ITAT 2013 Proceedings, CEUR Workshop Proceedings Vol. 1003, pp. 2–9
http://ceur-ws.org/Vol-1003, Series ISSN 1613-0073, © 2013 M. Lopatková, M. Plátek

Formalization of Word-Order Shifts by Restarting Automata*

Markéta Lopatková and Martin Plátek

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

Abstract: The phenomenon of word order freedom plays an important role in syntactic analysis of many natural languages. This paper introduces a notion of a *word order shift*, an operation reflecting the degree of word order freedom of natural languages. The word order shift is an operation based upon word order changes preserving syntactic correctness, individual word forms, their morphological characteristics, and their surface dependency relations in a course of a stepwise simplification of a sentence, a so called *analysis by reduction*.

The paper provides linguistic motivation for this operation and concentrates on a formal description of the whole mechanism through a special class of automata, so called restarting automata with the shift operation enhanced with a structured output, and their computations.

The goal of this study is to clarify the properties of computations needed to perform (enhanced) analysis by reduction for free word order languages.

1 Introduction

The phenomenon of word order freedom plays an important role in syntactic analysis of many natural languages. A relatively free word order combined with additional linguistic constraints constitutes a big challenge for a formal description of syntactic structures.Here we introduce a notion of a *word order shift*, an operation reflecting the degree of word order freedom of natural languages. The word order shift is a word order change preserving syntactic correctness, individual word forms, their morphological characteristics, and their surface dependency relations in the course of a stepwise simplification of a sentence under investigation, referred to as an *analysis by reduction*. Section 2 provides a linguistic motivation and informal description of the process.

Section 3 concentrates on a formal description of the whole mechanism through a special class of automata, so called restarting automata with the shift operation enhanced with structured output, pRL_e-automata, and their computations.

Previous Work. The paper [6] introduced a specific approach to the problem of measuring the word-order freedom. It focused upon an interplay of two phenomena related to word order: a *(non-)projectivity* of a sentence and a *number of word order shifts* within the analysis by reduction. This interplay was exemplified on a sample of Czech sentences with clitics and non-projectivities (long distance dependencies, see esp. [10, 4]). This approach is based upon the method of analysis reduction – a stepwise simplification of a sentence preserving its correctness. The analysis by reduction had been applied in a relatively free manner with no constraints on the projectivity of its individual steps. Our investigation focused on a set of syntactically analyzed Czech sentences from the Prague Dependency Treebank (PDT) [2]. The experiment on treebank data showed that with only basic constraints on the analysis by reduction, see [6], the minimal number of necessary shifts for any sentence from the test set did not exceed one. This number actually confirmed the initial assumption for Czech as a language with relatively high degree of word-order freedom; nevertheless, we have continued the search for more complicated sentences. This lead to the discovery of a special construction requiring (at least) two shifts, as it is discussed in Section 2.1.

The analysis by reduction has served as a motivation for new a family of automata, so called *restarting automata*, see esp. [14, 13]. As a first step, analysis by reduction was modelled as a formal string to string translation using a suitable model of restarting automata [9]. Then a class of enhanced restarting automata with an output consisting of a single DR-tree was introduced, [16]. Finally, the model was further enhanced – a model of restarting automata that assign a set of dependency structures (DR-structures) to every reduction of an input sentence was discussed, [15]. DR-structures can capture a set of dependency trees representing sentence on different layers of linguistic description in a parallel way. Here we further enrich the performance of these automata with the shift operation.

2 Analysis by Reduction

Let us first describe the main ideas behind the method used for sentence analysis. *Analysis by reduction (*AR*)* is based on a stepwise simplification of an analyzed sentence, see [8]. It defines possible sequences of reductions (deletions) in the sentence – each step of AR is represented by *deleting* a sentence member, possibly accompanied by *rewriting* a word of the sentence (and thus its shortening); in specific cases, deleting is accompanied by a *shift* of a word to another word order position.

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

Let us stress the basic constraints imposed on the analysis by reduction:

(i) the obvious constraint on *preserving individual words (word forms)*, their morphological characteristics and/or their surface dependency relations;

(ii) the constraint on *preserving the correctness* (i.e., a grammatically correct sentence must remain correct after its simplification);

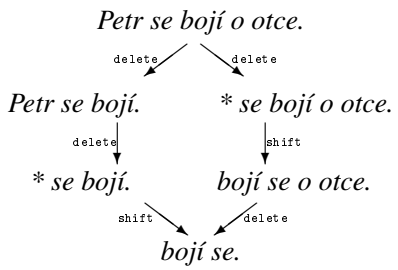
(iii) moreover, the application of the *shift operation is limited* only to cases when a shift is enforced by the correctness preserving principle of AR, i.e., a simple deletion would result in an incorrect word order and thus violate the principle (ii) above.

As a consequence of AR, it is possible to derive formal dependency relations between individual sentence members – pairs of governing and dependent nodes – based on the possible order(s) of reductions, as it is described in [8, 15], see ex. (1) below. Compare also with other approaches aiming to determine dependency relations summed up esp. in [1] and [12].

Example:

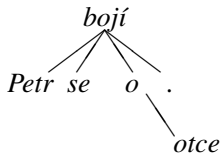
(1) *Petr se bojí o otce.*
'Peter – REFL – fears – for – father'
'Peter fears for his father.'

The analysis by reduction can be summarized in the following scheme:



Example (1) 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 simple deletion of the prepositional group *o otce* 'for father' (following the constraint on correctness of 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 an incorrect word order variant starting with the clitic *se* (such position of a clitic is forbidden in Czech). Thus the shift operation correcting the word order is enforced $\rightarrow_{shift}$ *bojí se o otce*.
As these possible reductions are independent of each other, we can conclude that the words *Petr* and *o otce* 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*.
Now, we can proceed in a similar way until we get the minimal correct simplified sentence $\rightarrow$ *bojí se.*

We can notice that the order of reductions reflects the dependency relations in the corresponding dependency tree. Informally, the words are 'cut from the bottom of the tree' (in other words, the 'pruning' operation is applied on the tree); i.e., a governing node must be preserved in a simplified sentence until all its dependent words are deleted.[1] In other words, AR makes it possible to identify the dependency tree for sentence (1):



The analysis by reduction has been designed to model a human who is able to check the correctness of the analysis. However, it is possible to model correct reductions on the basis of large corpora of natural language sentences, see [11].

2.1 The role of shifts. The above mentioned experiment on a set of non-projective sentences from PDT [2] (namely on the sentences with a non-projectivity given by a modal verb, see [3]) constituted only the first step.

As a follow-up experiment, we have decided to supplement the data with 'suspicious' types of sentences identified in previous work on Czech word order freedom, esp. in [4]. It allowed to include also less frequent phenomena into our investigations. As a result, we found a well-formed Czech sentence that requites at least two shifts in the course of AR, see the following example (2).

Let us stress that the role of shifts in the course of the analysis by reduction was limited to cases where a shift 'corrects' word order of a simplified sentence – this concerns primarily the cases when an input sentence contains a clitic (as in sentences (1)-(2)).

Example:

(2) *S těžkým se bála pomoci úkolem.*
'with – difficult – REFL – (she) was afraid – to help – task'
'With a *difficult* task, she was afraid to help.'

Although the sample sentence is rather strange, the splitting of the prepositional group is a grammatical construction in Czech allowing to put a strong stress on an adjective modifying the noun and not on the whole prepositional group.

Due to the dependency relations present in the sentence there is only one possibility how to reduce it, the reduction of the adjective *těžkým* 'difficult'. Unfortunately, it results in a syntactically incorrect word order:
$\rightarrow_{delete}$ * *s se bála pomoci úkolem.*
This situation can be corrected in two possible ways, let us focus on one of them:

[1] Let us note that some relations – e.g., the relation between a preposition and its 'head' noun as well as a verb and some clitic – are rather technical (not only) from the point of view of AR: such pairs must be reduced in a single step (despite being represented as two nodes in a dependency tree); here we adhere to the practice used in the PDT annotation.

$\rightarrow_{shift}$ *s úkolem se bála pomoci.* (A shift of the noun *úkolem* 'task' next to the preposition.)
$\rightarrow_{delete}$ * *se bála pomoci.* (Unfortunately, the next reduction must remove the prepositional group *s úkolem* 'with task' making the sentence again ungrammatical due to the clitic on the sentence first position.)
$\rightarrow_{shift}$ *bála se pomoci.* (Now we can repair the sentence by shifting the verb *bála* to the left.)
$\rightarrow_{delete}$ *bála se.* (The reduction of the word *pomoci* 'help' ends the process.)
Similarly for the second sequence of reductions and shifts. Regardless of the selected sequence, we must apply at least two shifts in the course of AR.

Remark: Note that we limit our observations to simple sentences, i.e., to sentences with a single clause. The reason is obvious, complex sentences might blur the whole picture by bringing additional phenomena into the play. For example, a coordination of clauses might completely change the total number of shifts in a complex sentence regardless of the phenomena contained in individual clauses. In general, each (coordinated) clause can induce a shift in the course of AR, thus the overall number of shifts cannot be bounded.

3 Restarting Automaton with a Structured Output

The crucial issue of the application of a word-order shift in the process of AR is actually identifying WHAT should be shifted WHERE. In order to model the whole process by means of a special kind of an automaton, we have to use special markers – pebbles, which mark the possible targets of shift locations. For this purpose we are going to define a restarting automaton with a finite set of special meta-instructions using pebbles – an pRL-automaton. The automaton with pebbles was originally introduced in [15] in order to define dependency output structures, and we are going to enhance its abilities with the shift operation.

We proceed in several steps. First, a restarting automaton with pebbles is enhanced with a shift operation (Section 3.1). Second, a structured output is added (Section 3.2). Finally, a computation of such enhanced automaton is described (Section 3.3).

3.1 Restarting Automaton with the Shift Operation Operating on Strings

Here we present a class of nondeterministic pRL-automata that are suitable for modeling AR – these automata allow for checking the whole input sentence (and marking selected words with pebbles) prior to any changes. It resembles a linguist who can read the whole sentence first, and then can reduce the sentence in a correct way. Thus he/she makes correct steps of the analysis by reduction. We have chosen a nondeterministic model to enable various sequences of reductions. This can serve for the verification of the (in)dependency between the individual parts of a sentence, as was sketched in Section 2.

A pRL-*automaton* M is (in general) a nondeterministic machine with a finite alphabet Γ, a head q (window of size 1) working on a flexible tape delimited by the left sentinel ¢ and the right sentinel \$ (¢, \$ $\notin \Gamma$), and two finite sets of restarting meta-instructions $R(M)$ and accepting meta-instructions $A(M)$, respectively. Such an automaton makes use of k pebbles $p_1 \cdots, p_k$.

Let us first describe informally the way how an pRL-automaton works. Each of its finite computations consists of certain phases called cycles, and a last – halting – phase called a tail. In each cycle, the automaton performs two passes through the tape: during the first pass, it marks selected symbols of a processed sentence with pebbles; then during the second pass, it performs its editing operations described by meta-instructions from $R(M)$; the operations can be applied (only) on the symbols marked by pebbles. In each (accepting) tail, the automaton marks all symbols by pebbles – according to a meta-instruction from $A(M)$ –, halts, and accepts the analyzed sentence.

3.1.1 Restarting meta-instructions. Each cycle of the pRL-automaton M is controled by a single restarting meta-instruction $I_{\mathrm{R}} \in R(M)$ of the form

$$I_{\mathrm{R}} = (E_0, I_1, \ldots I_s; O_p;\ \mathsf{Restart}) \tag{1}$$

where:
– each I_i is an instruction of the form $(a_i, E_i), 1 \leq i \leq s$,
– each E_i is a regular language over Γ; E_i is interpreted as the i-th context of I_{R} (the contexts are usually represented by regular expressions);
– each instruction I_i marks the symbol $a_i \in \Gamma$ from the tape with the pebble p_i within the first pass;
– $O_p = \{o_1, \cdots, o_p\}$ is an ordered set of editing operations, $o_j \in \{\mathrm{dl}[i], \mathrm{wr}[i,b], \mathrm{sh}[i,l] | 1 \leq i,l \leq s, i \neq l, b \in \Gamma\}$,
– o_j is the j-th editing operation performed in the second pass:
— if $o_j = \mathrm{dl}[i]$ then it deletes the symbol a_i,
— if $o_j = \mathrm{wr}[i,b]$ then it rewrites a_i by b,
— if $o_j = \mathrm{sh}[i,l]$ then it shifts a_i to the position behind a_l.
We require a 'uniqueness' of operations on a single symbol, i.e., if $\mathrm{dl}[i] \in O_p$ then O_p does not contain the operations $\mathrm{wr}[i,b], \mathrm{sh}[i,l]$ for any b,l.

Performing a meta-instruction: For an (input) sentence $w \in \Gamma^*$, we define the *tape inscription* ¢w\$, and a *reduction* $w \vdash^c_{I_{\mathrm{R}}} w'$. When trying to execute a cycle C by performing the meta-instruction I_{R}, it starts with the tape inscription ¢w\$, M will get stuck (and so reject) if w does not permit a factorization of the form $w = v_0 a_1 v_1 a_2 \ldots v_{s-1} a_s v_s$ ($v_i \in E_i$ for all $i = 0, \ldots, s$). On the other hand, if w permits factorizations of this form, then one of them is (nondeterministically) chosen.

The automaton processes the input word in two passes:
1. During the first pass of C, M puts the pebble p_i on each a_i (this corresponds to the instruction I_i, for each i, $1 \leq i \leq s$).
2. During the second pass of C controlled by the ordered set of operations O_p, the tape inscription ¢w\$ is transformed (reduced) into a new tape inscription $w' = v_0 r_1 v_1 \cdots v_{s-1} r_s v_s$, where $1 \leq i \leq s$:

- In the second pass of C, M performs its operations in the prescribed order $o_1, o_2, \cdots, o_p$.
- $r_i = \lambda$ if there is a deleting operation $o_j \in O_p$ such that $o_j = \mathrm{dl}[i]$;
  i.e., a_i is deleted during the cycle C;
- $r_i = b$ if there is a rewriting operation $o_j \in O_p$ such that $o_j = \mathrm{wr}[i,b]$;
  i.e., a_i is rewritten with b during the cycle C;
- $r_i = \lambda$ and $r_j = a_l a_i$ if there is a shifting operation $o_j \in O_p$ such that $o_j = \mathrm{sh}[i,l]$;
  i.e., a_i is shifted behind the symbol marked by the pebble p_l during the cycle C;
- $r_i = a_i$ if there is no operation $o_j \in O_p$ applicable on a_i.
  (Note that such an a_i may – together with symbols where some of above mentioned operation is applied – create the output structure, see the following section, and thus they must be marked with pebbles.)
- At the end of each cycle, the Restart operation is performed: all the pebbles are removed from the new tape inscription w', and the head is placed on the left sentinel.

Of course, we can delete, rewrite or shift neither the left sentinel ¢ nor the right sentinel \$. It is required that during a cycle, M executes at least one delete operation.

If no further cycle is performed, each finite (accepting) computation necessarily finishes in a tail performed according to an accepting meta-instruction.

3.1.2 Accepting meta-instructions. Tails of accepting computations are described by a set accepting meta-instructions $A(M)$, each of the form:

$$I_{\mathrm{A}} = (a_1, \ldots, a_s, \mathsf{Accept}), \tag{2}$$

where a_i are symbols from Γ.
Performing a meta-instruction: The tail performed by the meta-instruction I_{A} starts with the inscription on the tape ¢z\$; if $z = a_1 \cdots a_s$, then M marks each symbol a_i with the pebble p_i, and accepts z (and the whole computation as well). Otherwise, the computation halts with rejection. Note that only operations distributing pebbles are allowed within an accepting meta-instruction – they will serve for building the structured output, as it is described in the following sections.

The notation $u \vdash_M^c v$ (a reduction of u to v by M) denotes a reduction performed during a cycle of M that begins with the tape inscription ¢u\$ and ends with the tape inscription ¢v\$; the relation $\vdash_M^{c^*}$ is the reflexive and transitive closure of $\vdash_M^c$. By $\mathsf{REM}(M)$ we denote the set of reductions performed by M.

A string $w \in \Gamma^*$ is *accepted* by M, if there is an accepting computation which starts with the tape inscription ¢w\$ and ends by executing an *accept* operation. By $L(M)$ we denote the language consisting of all words accepted by M; we say that M *recognizes (accepts) language* $L(M)$.[2]

Note that we limit ourselves only to such combinations of operations that are adequate (and necessary) for known linguistic examples.

Example: Let us illustrate the notions of restarting and accepting meta-instructions on the analysis of sentence (1) from Section 2. This example formalizes the AR of the sentence *Petr se bojí o otce.* 'Peter fears for his father., namely the left branch of its AR (see the scheme in Section 2). The respective pRL$_e$-automaton M is described by two restarting meta-instructions I_{R_1} and I_{R_2}, and one accepting meta-instruction I_A, namely:

$I_{R_1} = (E_0, I_1^1, I_2^1, I_3^1; O_2^1; \mathsf{Restart})$
$E_0 = \{Petr\ se\}$, $I_1^1 = (\ bojí\ , \{\lambda\})$
$O_2^1 = \{\mathrm{dl}[2], \mathrm{dl}[3]\}$, $I_2^1 = (\ o\ , \{\lambda\})$
$I_3^1 = (\ otce\ , \{\cdot\})$
$I_{R_2} = (E_0, I_1^2, I_2^2, I_3^2; O_2^2; \mathsf{Restart})$
$E_0 = \{\lambda\}$, $I_1^2 = (\ Petr, \{\lambda\})$
$O_2^2 = \{\mathrm{dl}[1], \mathrm{sh}[2,3]\}$, $I_2^2 = (\ se, \{bojí\})$
$I_3^2 = (\ bojí\ , \{\lambda\})$
$I_A = (\ bojí\ ,\ se, .\ , \mathsf{Accept})$

The computation consists of two cycles. Within the first cycle described by the meta-instruction I_{R_1}, the instructions I_1^1, I_2^1 and I_3^1 put pebbles p_1, p_2 and p_3 on the symbols containing the words *bojí* (with the left context *Petr se* and empty right context), *o*, and *otce*, respectively; the operations dl[2] and dl[3] delete the word *o* (marked with the pebble p_2) and *otce* (marked with the pebble p_3), respectively. Then the automaton restarts (in particular, all pebbles are removed). Similarly in the second cycle (meta-instruction I_{R_2}), the instructions I_1^2, I_2^2 and I_3^2 mark the respective words and then operation dl[1] deletes the word *Petr* (marked with p_1) and the operation sh[2,3] shifts the word *se* with the pebble p_2 behind the word *bojí* (with the pebble p_3). The accepting instruction I_A just marks the remaining words *bojí* and *se* with pebbles.

The following property – obviously ensured by pRL-automata – has a crucial role in our applications of restarting automata.

Weak Correctness Preserving Property. Any pRL-automaton M is *weakly correctness preserving*, i.e., (for all

[2] Note that language $L(M)$ accepted by the automaton M is the same language as $L_C(M)$ in [15].

$u, v \in \Gamma^*$)
$v \in L(M)$ and $u \vdash_M^{c^*} v$ imply that $u \in L(M)$.

Our goal is to model the analysis by reduction by automata with the following stronger property:

Correctness Preserving Property. A pRL-automaton M is *correctness preserving*, if (for all $u, v \in \Gamma^*$) $u \in L(M)$ and $u \vdash_M^{c^*} v$ imply that $v \in L(M)$.

3.2 Restarting Automaton with the Shift Operation Enhanced with Structured Output

In this section, we introduce enhanced restarting automata with structured output – the so-called pRL$_e$-automata. During their enhanced computations, these automata build – with the help of pebbles – so called DR-structures, i.e., graphs consisting of items representing individual occurrences of symbols (words) in the input sentence and directed edges between them.

In order to represent a structured output by an pRL$_e$-automaton, the items of flexible tape (representing just sentinels and symbols (= words) of a processed sentence) are enriched with two integers denoting horizontal and vertical position of a respective word in the resulting DR-structure.

An pRL$_e$-automaton is specified in two steps: First, instructions of an pRL-automaton are enhanced with graphs (Sections 3.2.1 and 3.2.2). Second, an application of these enhanced instructions (i.e., its computations) are specified with the help of DR-structures (Sections 3.2.3 and 3.3).

A pRL$_e$-automaton $M_\mathrm{e} = (\Gamma, ¢, \$, ER(M_\mathrm{e}), EA(M_\mathrm{e}))$ consists of an alphabet Γ, sentinels $¢, \$ \notin \Gamma$, a set of enhanced restarting meta-instructions $ER(M_\mathrm{e})$, and a set of enhanced accepting meta-instructions $EA(M_\mathrm{e})$.

An enhanced meta-instruction is a pair $I_{\mathrm{EX}} = (I_X, G)$, where I_X is a (restarting or accepting) meta-instruction of a pRL-automaton, and G is a directed acyclic graph $G = (U, H)$. The graph G represents the required structure of symbols processed during the application of the meta-instruction I_X. In this paper we request G to be a rooted tree. (This restriction ensures linguistically adequate output structures.)

3.2.1 Enhanced restarting meta-instruction I_{ER}.

Let I_R be a restarting meta-instruction of the form (1) introduced in section 3.1.1. We define the corresponding graph $G = (U, H)$ in the following way:

1. The set of nodes U consists of three disjunct subsets U_n, U_w, and U_p:

- $U_n = \{[i, a_i] \mid \mathrm{dl}[i] \in O_p \text{ or } \mathrm{wr}[i, b] \in O_p,\ 1 \le i \le s\}$. We can see that U_n covers all (original) words which are deleted or rewritten words within I_R ($a_i \in \Gamma$ is the word marked with a pebble p_i).
- $U_w = \{[i, b] \mid \mathrm{wr}[i, b] \in O_p,\ 1 \le i \le s\}\}$. The set U_w represents words resulting from the rewritten operations within I_R.
- U_p contains the root of G of the form $[i, a_i]$ (unless the root is already contained in U_w).

2. An edge $(u, v) \in H$ represents:

- either *deleting*: if $\mathrm{dl}[i] \in O_p$ then the edge has the form $u = [i, a_i]$, and $v = [j, a_j]$ for some $j \neq i$ (j is interpreted as a position of respective governing node);
- or *rewriting*: if $\mathrm{wr}[i, b] \in O_p$ then the edge has the form $u = [i, a_i]$, and $v = [i, b]$ (i.e., u represents the original symbol a_i and v the rewritten)

3.2.2 Enhanced accepting meta-instruction I_A.

Let I_A be an accepting instruction of the form $I_A = (a_1, \ldots, a_s, \mathsf{Accept})$ (i.e., as it is introduced in Section 3.1, $X = A$). Then a corresponding enhanced accepting meta-instruction has a form $I_{EA} = (I_A, G)$, where $G = (U, H)$ is an enhancing graph; the symbols $a_1, \ldots, a_s$ are pebbled and they correspond to the nodes $\{[1, a_1], [2, a_2] \ldots, [s, a_s]\}$ of U. For each edge $h = (u, v) \in H$, we suppose that $u = [a_i, i]$, $v = [a_j, j]$ ($1 \le i, j \le s$ and $i \neq j$).

Example: Let us return to the analysis of sentence (1) *Petr se bojí o otce.* 'Peter fears for his father.', see Section 2. In Section 3.1, we have already presented two restarting instructions I_{R_1} and I_{R_2}, and one accepting instruction I_A. Here we enrich these instructions with a graph describing the (part of) resulting DR-structures – the respective pRL$_e$-automaton M_e is described by enhanced instructions $I_{ER_1} = (I_{R_1}, G_1)$, $I_{ER_2} = (I_{R_2}, G_2)$, and $I_{EA} = (I_A, G_A)$, namely:

$G_1 = (U_1, H_1)$ $\quad U_1 = \{[1, \textit{bojí}], [2, o], [3, \textit{otce}]\}$
$\quad H_1 = \{([2, o], [1, \textit{bojí}]), ([3, \textit{otce}], [2, o])\}$
$G_2 = (U_2, H_2)$ $\quad U_2 = \{[1, \textit{Petr}], [3, \textit{bojí}]\}$
$\quad H_2 = \{([1, \textit{Petr}], [3, \textit{bojí}])\}$
$G_A = (U_A, H_A)$ $\quad U_A = \{[1, \textit{bojí}], [2, \textit{se}], [3, \cdot]\}$
$\quad H_A = \{([2, \textit{se}], [1, \textit{bojí}]), ([3, \cdot], [1, \textit{bojí}])\}$

3.2.3 DR-structures.

In this paragraph, we introduce so-called DR-structures (Delete Rewrite structures), which serve for the representation of output structures of restarting automata. A DR-structure captures syntactic units (words and their categories used in an input sentence and also in its reduced forms) 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 DR-structure is a directed acyclic graph $D = (V, E)$, where V is a finite set of nodes, and $E \subset V \times V$ is a finite set of edges. A *node* $u \in V$ is a tuple $u = [i, j, a]$, where a is a symbol (word) assigned to the node, and i, j are natural numbers, i represents the horizontal position of the node u (i.e., the word order in an input sentence), j represents the vertical position of u (it is equal to a number of rewritings of a word on a particular horizontal position). Each *edge* $h = (u, v) \in E$, where $u = [i_u, j_u, a]$ and $v = [i_v, j_v, b]$

for some non-negative integers i_u, i_v, j_u, j_v and $a, b \in \Gamma$, is either

- *oblique edge* iff $i_u \neq i_v$;
  such edges are interpreted as representing syntactic relations between respective symbols (words), or
- *vertical edge* iff $i_u = i_v$ and $j_v = j_u + 1$;
  such edges are interpreted as representing a rewriting of a symbol (word) on the respective position i_u .

We say that $D = (V,E)$ is a DR-tree if the graph D is a rooted tree (i.e., all maximal paths in D end in its single root).

Remark: It is important to say that a DR-structure preserves word order of an original sentence, i.e., the order of nodes in DR-structure (corresponding to words and their categories) is not affected by (possible) shifts. However, shifts affect the way how a DR-structure corresponds to individual reductions of the automaton – an information of both original and changed positions of individual words (and their categories) must be recorded during the computation.

3.3 Computations and (Structured) Languages by pRL$_e$-automata

Based on the definition, it is clear that for a pRL$_e$-automaton M_e, a reduction relation $\vdash_{M_\mathrm{e}}$ can be defined in the same way as the reduction relation $\vdash_M$ of the corresponding pRL-automaton M; further, M_e accepts the same language as the M (leaving aside output DR-structures), and we can see that a set of reductions $\mathsf{REM}(M_e) = \mathsf{REM}(M)$, as well.

Similarly as in [15], computation of an pRL$_e$-automaton M_e is characterized by sequences of its configurations. A *configuration* is a pair of the form $C_w = (T_w, D_w)$, where T_w (so called *tape content* of C_w) is a string of items representing current sentence w on the tape and $D_w = (V,E)$ is a DR-structure (see above). The items of T_w create a subset of V such that T_w contains just all roots of individual components (trees) of the DR-structure D_w (thus each item $[i, j, x_i]$ of T_w consists of a word position i (i.e., the horizontal index), a symbol x_i on this position (word of the original sentence), and the number of its rewritings j). Moreover, T_w contains at most one item for each horizontal position i.

Note that – contrary to [15] – the items of T_w are ordered; however, their order need not reflect the original word order of an input sentence as the items (triples) may be re-ordered by a shift operation.

An (accepting) computation C of M_e on an input sentence w starts in the initial configuration $C_w^{in} = (T_w^{in}, D_w^{in})$: for the given sentence $w = x_1 \dots x_n$ ($x_j \in \Gamma$, $1 \leq j \leq n$), the initial tape content consists in the string of items $T_w^{in} = [1,0,x_1] \cdots [n,0,x_n]$, and the DR-structure $D_w^{in} = (V_w^{in}, \emptyset)$, where $V_w^{in} = \{[1,0,x_1], \cdots, [n,0,x_n]\}$. The computation continues by performing several cycles, and it ends by a tail.

A cycle means an application of an enhanced restarting meta-instruction on a given configuration, resulting in a new configuration. During each cycle of C, some items of the tape content are deleted, rewritten, or shifted (as in a cycle of the original automaton without structures); further, some edges and (rewritten) nodes are added to the DR-structure of the corresponding configuration, in accordance with the graph attached to the respective enhanced meta-instruction (see below for more formal description).

A tail consists in an application of an enhanced accepting meta-instruction – it results in an acceptance of a configuration; further, some edges are added to the current DR-structure in accordance with a graph attached to this enhanced meta-instruction.

Let us stress that during a computation, some nodes and edges may be added to the DR-structures of the computation, and never removed. At the end of each computation, we obtain a DR-tree covering the whole input sentence.

3.3.1 Application of an enhanced restarting meta-instruction I_{ER}.

Let $I_{ER} = (I_R, G)$ be an enhanced restarting instruction in the previously defined form, with $G = (U,H)$ (see Section 3.2). An application of I_{ER} on an enhanced configuration $C_w = (T_w, D_w)$ of a word w results in a new configuration $C_{w'} = (T_{w'}, D_{w'})$. The application consists in the following steps:

1. Choosing a factorization of w of the form $w = v_0 a_1 v_1 a_2 \dots v_{s-1} a_s v_s$ such that $v_i \in E_i$ for all $i = 0, \dots, s$ (see the form of a restarting meta-instruction (1)). Let us consider that the items from T_w containing the symbols $a_1, \cdots, a_s$ are pebbled.
   If w does not permit any factorization of this form, then I_{ER} cannot be applied on the configuration C_w.
2. Transforming the tape content T_w containing the sentence w into the tape $T_{w'}$ containing the sentence $w' = v_0 r_1 v_1 \cdots v_{s-1} r_s v_s$; r_i ($1 \leq i \leq s$), are described in section 3.1). The transformation of T_w to $T_{w'}$ keeps the following rules:
   - each item $v_i \in T_w$ ($1 \leq i \leq s$) which does not undergo an editing operation is copied to $T_{w'}$,
   - no deleted item from T_w is transferred to $T_{w'}$, and
   - each item $[k_i, j_i, a_i]$ from T_w that corresponds to a rewritten symbol a_i (hence $o_i = \mathrm{wr}[i,b]$, $b \in \Gamma$, $1 \leq i \leq s$) is replaced by $[k_i, j_i + 1, b]$ in $T_{w'}$;
   - the shift of an item $[i, j, a_i]$ is performed in such a way that the indices i, j remain unchanged in $T_{w'}$.
3. All edges and nodes from the DR-structure D_w are transferred into the new DR-structure $D_{w'}$.

4. For each edge $e \in H$, a new edge is inserted into the new DR-structure $D_{w'} = (V', E')$:

   - if $e = ([i,a_i],[r,a_r])$ is a deleting edge from H $(1 \leq r \leq s)$, and $[k_i,j_i,a_i],[k_r,j_r,a_r] \in T_w$ are the items covering the pebbled symbols a_i and a_r, respectively, then an oblique edge $e_o = ([k_i,j_i,a_i],[k_r,j_r,a_r])$ is inserted into $D_{w'}$ (i.e., into E').
   - if $e = ([i,a_i],[i,b_i])$ is a rewriting edge and $[k_i,j_i,a_i] \in T_w$ is the item covering the pebbled symbol a_i, then a vertical edge e_v is inserted into E': e_v leads from the item $[k_i,j_i,a_i]$ to the item $[k_i,j_i+1,b_i]$.
     At the same time the node $[k_i,j_i+1,b_i]$ is added to the set V'.

We say that the configuration $C_{w'}$ can be reached in one *(restarting) step* from the configuration C_w by M_e (using the instruction I_ER) and denote it by $C_w \models^{I_\mathrm{ER}}_{M_\mathrm{e}} C_{w'}$, or simply by $C_w \models_{M_\mathrm{e}} C_{w'}$.

3.3.2 Application of an enhanced accepting meta-instruction I_{EA}.

Let $I_{EA} = (I,G)$ be an enhanced accepting meta-instruction, where $I = (a_1,\ldots,a_s,\mathsf{Accept})$ and $G = (U,H)$. Let $C_w = (T,D)$ be an enhanced configuration of M_e with the tape content $T = [i_1,j_1,a_1],\cdots,[i_s,j_s,a_s]$ covering the word $w = a_1\cdots a_s$. Then the application of I_{EA} on the configuration C_w results in a new configuration $C_{w'} = (T',D')$, where $D' = (V',E')$:

1. $T' = [i_1,j_1,a_1],\cdots,[i_s,j_s,a_s] = T$;
2. all edges and nodes from the DR-structure D are transferred into the new DR-structure D';
3. moreover, for each edge $e = ([r,a_r],[s,a_t]) \in H$ the new edge $([i_r,j_r,a_{i_r}],[i_t,j_t,a_{i_t}])$ is inserted into D'.

Similarly as in the previous case, we say that the configuration $C_{w'}$ can be reached in one *(accepting) step* from the configuration C_w by M_e using the instruction I_EA, and denote it by $C \models^{I_\mathrm{EA}}_{M_\mathrm{e}} C'$.

We can finally define an accepting computation of an enhanced pRL$_e$-automaton and subsequently also a DR-language consisting from the output DR-structures.

Let $\mathscr{C} = C_0,C_1,\ldots,C_k$ be a sequence of configurations such that
– C_0 is the initial configuration for a word w, – $C_i \models^{I_i}_{M_\mathrm{e}} C_{i+1}$ holds for all $i = 0,\ldots,k-1$,
– I_i are restarting instructions for all $i = 0,\ldots,k-2$,
– I_{k-1} is an accepting instruction.
Then we say that $\mathscr{C}$ is an *accepting computation* of M_e, the word w is *accepted* by M_e, and the DR-structure D_k (from the last configuration $C_k = (T_k,D_k)$) is the *output* DR-*structure* of the computation $\mathscr{C}$. We write $\mathsf{DR}(\mathscr{C}) = D_k$.

Let $AC(M_\mathrm{e})$ denote the set of all accepting computations of the pRL$_e$-automaton M_e. Then the DR-language of M_e is the set $\mathsf{DR}(M_\mathrm{e}) = \{\mathsf{DR}(\mathscr{C}) \mid \mathscr{C} \in AC(M_\mathrm{e})\}$.

Remark. From the linguistic point of view, $\mathsf{DR}(M_\mathrm{e})$ should be considered as a set of (shallow) dependency-based syntactic trees / analyses computed by the automaton M_e.

Example: Let us return to the example (1), section 2, *Petr se bojí o otce.* 'Peter fears for his father.' In Sections 3.1.2 and 3.2.2, the pRL$_e$-automaton M_e – described by two restarting meta-instructions and one accepting meta-instruction together with the graphs enhancing these meta-instructions – was presented. The sequence $S = I_{ER_1},I_{ER_2},I_{EA}$ performs the left branch of the analysis by reduction. In the course of the corresponding computation, the DR-tree D_3 is gradually constructed.

Now we can present the accepting computation $\mathscr{C} = C_0,C_1,C_2,C_3$ on the sentence (1) determined by the sequence of meta-instructions S. Let $C^{in} = (T^{in},D^{in})$, $C_1 = (T_1,D_1)$, $C_2 = (T_2,D_2)$, $C_3 = (T_3,D_3)$, C_0 is the initial configuration of $\mathscr{C}$, C_3 is the accepting configuration. Further we gradually describe T^{in},T_1,T_2,T_3, and $D^{in},\mathsf{D}_1,D_2,D_3$, where $D_3 = \mathsf{DR}(\mathscr{C})$.

$T^{in} = [1,0,Petr][2,0,se][3,0,bojí][4,0,o][5,0,otce][6,0,\cdot]$
$D^{in} = (V^{in},\emptyset)$
$V^{in} = \{[1,0,Petr],[2,0,se],[3,0,bojí],[4,0,o],[5,0,otce][6,0,\cdot]\}$

$T_1 = [1,0,Petr][2,0,se][3,0,bojí][6,0,\cdot]$
$D_1 = (V^{in},E_1)$
$E_1 = \{([4,0,o],[3,0,bojí]),([5,0,otce],[4,0,o])\}$

$T_2 = [3,0,bojí][2,0,se][6,0,\cdot]$
$D_2 = (V^{in},E_1 \cup E_2)$
$E_2 = \{([1,0,Petr],[3,0,bojí])\}$

$T_3 = T_2$
$D_3 = (V^{in},E_1 \cup E_2 \cup E_3)$
$E_3 = \{([2,0,se],[3,0,bojí]),([6,0,\cdot],[3,0,bojí])\}$

We can see that the presented example is very simple – the context languages from the instructions contain always one string (sometimes the empty one λ). Moreover, we have not used the operation wr$[i]$ at all. That caused that all items used in $\mathscr{C}$ have their vertical index equal to 0.

We can notice that D_3 is a formalization of the dependency tree from the example (1). (Let us note that a computation performing the another branch of AR should create the same DR-tree.)

Observations. Let us finally (and just informally) formulate some observations concerned the presented notions.

1. The classes of introduced languages, sets of reductions, and DR-languages create infinite hierarchies with respect to the number of used pebbles. We are able to prove exactly such propositions.

2. On the other hand, we can observe that very few pebbles (six, according to our preliminary hypothesis) suffice

to analyze adequately any sentence from the Prague Dependency Treebank (see [2]). It means that the complexity of analysis by reduction of natural languages is not too hight; however, it is not too trivial.

3. In a similar way, we can see that the number of used shifts in one cycle does not need to exceed one.

4. As far as we can foresee, the use of rewritings within the analysis by reduction is forced only by two linguistic phenomena: coordinations and noun groups with numerals. Moreover, the number of rewritings in one cycle does not need to exceed one.

5. Further, it not hard to see that the shift operation cannot be simulated by rewritings within the class of pRL_e-automata, due to the limited number of rewritings in encoded in meta-instructions.

6. Finally, the measure of word-order complexity from [4] is not bounded by any natural number for DR-languages of pRL_e-automata.

Conclusion and Perspectives

The main novelty of the work presented in the paper is an enhancement of a formal system in a way that makes it possible to capture adequately relations between linguistic notions in the course of (enhanced) analysis by reductions, namely it describes dependency structures on the one side, and (complexity of) word order on the other side (for the clarity reason, we do not distinguish between input symbols and their categories in this paper). However, the investigation of these notions is complete neither from the linguistic point of view, nor from the point of view of the formal automata theory.

As for the future work, we plan to continue the research in the direction of combining more types of linguistic phenomena which influence the degree of word order freedom for natural languages as well as enhancing the proposed formal system so that it would capture the added phenomena. Further research of the presented type of automata should present its properties in an exact way, as well.

Further, we plan to propose a refinement of AR allowing to set constraints on possible reduction steps, namely a constraint on *projective reductions* only, as the additional experiments with a different or modified set of constraints applied on the shift operation will probably provide new measures of word order freedom. An interesting research direction in this respect would be the investigation of projectivization of sentences prior to the individual reductions of AR, which might completely change the picture and allow for a more detailed investigation of an interplay between clitics and non-projective sentences, see also [5].

Further investigation of the shift operation might also help to describe the relationship between different levels of formal description of natural languages in the Functional Generative Description, namely between the tectogrammatical and analytical layer. This would strengthen the formal background of the theory.

References

[1] Gerdes, K., Kahane, S.: Defining dependencies (and constituents). In: Gerdes, Hajičová, Wanner (eds.) Proceedings of Depling 2011. pp. 17–27. Barcelona (2011)

[2] Hajič, J., Panevová, J., Hajičová, E., Sgall, P., Pajas, P., Štěpánek, J., Havelka, J., Mikulová, M., Žabokrtský, Z., Ševčíková-Razímová, M.: Prague Dependency Treebank 2.0. Linguistic Data Consortium, Philadelphia (2006)

[3] Hajičová, E., Havelka, J., Sgall, P., Veselá, K., Zeman, D.: Issues of Projectivity in the Prague Dependency Treebank. The Prague Bulletin of Mathematical Linguistics 81, 5–22 (2004)

[4] Holan, T., Kuboň, V., Oliva, K., Plátek, M.: On Complexity of Word Order. Les grammaires de dépendance – Traitement automatique des langues 41(1), 273–300 (2000)

[5] Kuboň, V., Lopatková, M.: A case study of a free word order (manuscript)

[6] Kuboň, V., Lopatková, M., Plátek, M.: Studying formal properties of a free word order language. In: Youngblood, G., McCarthy, P. (eds.) Proceedings of the FLAIRS 25 Conference. pp. 300–305. AAAI Press, Palo Alto (2012)

[7] Kunze, J.: Abhängigkeitsgrammatik, Studia Grammatica, vol. XII. Akademie Verlag, Berlin (1975)

[8] Lopatková, M., Plátek, M., Kuboň, V.: Modeling Syntax of Free Word-Order Languages: Dependency Analysis by Reduction. In: Matoušek, V. et al. (ed.) Proceedings of TSD 2005. LNCS, vol. 3658, pp. 140–147. Springer (2005)

[9] Lopatková, M., Plátek, M., Sgall, P.: Towards a Formal Model for Functional Generative Description: Analysis by Reduction and Restarting Automata. The Prague Bulletin of Mathematical Linguistics 87, 7–26 (2007)

[10] Marcus, S.: Sur la notion de projectivité. Zeitschrift fur Mathematische Logik und Grundlagen der Mathematik 11(1), 181–192 (1965)

[11] Mareček, D., Žabokrtský, Z.: Exploiting reducibility in unsupervised dependency parsing. In: Proceedings of EMNLP-CoNLL 2012. pp. 297–307. ACL (2012)

[12] Mel'čuk, I.A.: Dependency in language. In: Gerdes, Hajičová, Wanner (eds.) Proceedings of Depling 2011. pp. 1–16. Barcelona (2011)

[13] Otto, F.: Restarting Automata and Their Relation to the Chomsky Hierarchy. In: Ésik, Z., Fülöp, Z. (eds.) Proceedings of DLT 2003. LNCS, vol. 2710, pp. 55–74. Springer, Berlin (2003)

[14] Otto, F.: Restarting Automata. In: Ésik, Z., Martin-Vide, C., Mitrana, V. (eds.) Recent Advances in Formal Languages and Applications, Studies in Computational Intelligence. vol. 25, pp. 269–303. Springer-Verlag, Berlin (2006)

[15] Plátek, M., Mráz, F., Lopatková, M.: (In)Dependencies in Functional Generative Description by Restarting Automata. In: Bordihn, H. et al. (ed.) Proceedings of NCMA 2010. books@ocg.at, vol. 263, pp. 155–170. Österreichische Computer Gesellschaft, Wien, Austria (2010)

[16] Plátek, M., Mráz, F., Lopatková, M.: Restarting Automata with Structured Output and Functional Generative Description. In: Dediu A. et al. (ed.) Proceedings of LATA 2010. LNCS, vol. 6031, pp. 500–511. Springer, Berlin Heidelberg (2010)