Restarting Automata with Structured Output and Functional Generative Description

Martin Plátek, František Mráz, and Markéta Lopatková

Charles University in Prague, Czech Republic
martin.platek@mff.cuni.cz, frantisek.mraz@mff.cuni.cz,
lopatkova@ufal.mff.cuni.cz

Abstract. Restarting automata were introduced for modeling linguistically motivated analysis by reduction. In this paper we enhance these automata with a structured output in the form of a tree. Enhanced restarting automata can serve as a formal framework for the Functional Generative Description. In this framework, a natural language is described at four layers. Working simultaneously with all these layers, tectogrammatical dependency structures that represent the meaning of the sentence are derived.

1 Introduction

Functional Generative Description (FGD) is a stratificational dependency based descriptive system for the Czech language, which has been in development since 1960s, see esp. [9].

Stratification approaches split language description into layers, each layer providing complete description of sentence and having its own vocabulary and syntax. Here we focus on two layers of FGD, the layer of wordforms (w-layer) and the most abstract layer of meaning (called tectogrammatical layer in FGD, t-layer). At the w-layer, sentence is represented as a simple string of words and punctuation. The t-layer comprises language meaning; the core concepts of this layer are dependency, valency, and topic-focus articulation, see esp. [9].

FGD as a *dependency based approach* describes meaning of a sentence as (tectogrammatical) dependency tree – (meaningful) words are represented as nodes of the tree (each node being a complex unit capturing the lexical meaning, important morphological and syntactic characteristics); relations among words are represented by oriented edges between nodes of the tree [2].

We attempt to provide a formal model for natural language processing based on an elementary method of analysis by reduction. The *analysis by reduction* (AR henceforth, see [4,5], here Section 1.1), serves for describing syntactic structures of natural languages (and particularly languages with free word order). The framework of restarting automata meets the basic requirements set for natural language description stated in [5], i.e., (i) distinguishing the set of well-formed sentences (henceforth L_w) and the set of meaning structures (TR for tectogrammatical representation) and (ii) setting TSH relation comprising mutual relation between these two sets, relation of synonymy and ambiguity (homonymy).

In [5], FGD is modeled as a formal string to string translation using a suitable model of restarting automata. Here we enrich this class of restarting automata with structured output; that is, we define a new type of restarting transducers – *enhanced simple restarting automata* that output dependency trees derived from AR (Section 2). However, such a model is still too general. We formulate several additional restrictions that should be put on an enhanced simple restarting automaton $\mathsf{M}_{\mathsf{FGD}}$ (the last subsection in Section 3). As a result, we get a formal automaton, which helps linguists in further developing the theoretical model of FGD for the Czech language while it allows us to study its formal properties from the mathematical point of view. Section 3.1 summarizes the current model and proposes directions for further research.

This work does not focus on mathematical statements about restarting automata and dependency grammars. Restarting automata have been intensively studied, see e.g. [7,8]. Articles studying mathematical properties of enhanced restarting automata are to be expected in a near future. They will combine the methodologies of restarting automata and dependency grammars (see e.g. [1,2]).

1.1 Basic Principles of Analysis by Reduction

Analysis by reduction makes it possible to define formal dependency relations between particular sentence members. Roughly speaking, (i) a certain word depends on (modifies) another word from the sentence if it is possible to reduce this modifying word (and obtain a simplified correct sentence), and (ii) two words can be removed in an arbitrary order if and only if they are mutually independent. So whereas the basic operation in constituent-based approaches is the decomposition of a sentence into continuous parts representing simplified structures (phrases), in the (dependency) analysis by reduction it is possible to determine dependency relations between individual lexical words leaving aside their word order.

AR is based on a stepwise simplification of an analyzed sentence (enriched with metalanguage information from all layers of FGD); in each step, at least one word / symbol of the input string is deleted, which may lead to rewriting some other symbol. The sentence is simplified until so called *core predicative structure* is reached (typically formed by a sentence predicate and its valency complementations).

When simplifying input sentence, it is necessary to apply certain elementary principles assuring adequate analysis, primarily the principle of correctness preserving and the principle of completeness with respect to valency structure [4]. Moreover, each step of AR must be minimal from a certain point of view – any potential reduction step concerning less symbols in the sentence would violate the principle of completeness (it would lead to an incomplete sentence); it implies that only items fulfilling valency slots of a single governing word are processed in a single reduction step.

The basic principles of AR are exemplified on several reduction steps for particular Czech sentence (Example 1); they illustrate how the sentence is simplified and how fragments of its dependency tree are built (see [5] for more details).

Eng.	*w*-layer	*m*-layer	*a*-layer	*t*-layer
Our	*Našeho*	*můj*.PSMS4	Atr	
Karel	*Karla*	*Karel*.NNMS4	Obj	
				[*my*].t_ACT
(we) plan	*plánujeme*	*plánovat*.VB-P-	Pred	*plánovat*.f_PRED.VF1
				Karel.c_PAT
				my.f_APP
				[*my*].t_ACT
to send	*poslat*	*poslat*.Vf- - -	Obj	*poslat*.f_PAT.VF2
for	*na*	*na*.RR- - 4	AuxP	
next	*příští*	*příští*.AA4IS	Atr	
year	*rok*	*rok*.NNIS4	Adv	*rok*.f_THL
				příští.f_RSTR
to	*do*	*do*.RR- - 2	AuxP	
England	*Anglie*	*Anglie*.NNFS2	Adv	*Anglie*.f_DIR3
.	.	..Z: - - -	AuxK	

Fig. 1. Representation of the sample sentence from Example 1 at four layers of FGD (simplified)

Example 1
Našeho Karla plánujeme poslat na příští rok do Anglie. ([9], p. 241, modified)
our - Karel - (we) plan - to sent - for - next - year - to - England
Eng. 'It's our Karel whom we plan to send for the next year to England.'

Figure 1 presents a simplified representation of the sample sentence at four layers of FGD. Each column captures one layer of language description (*w*-, *m*-, *a*- and *t*-layer, respectively, see Section 1). Rows capture information related to particular words and punctuation marks (one or more rows for an individual word / punctuation, depending on its surface and deep word order, see [5]).

In the first reduction steps of AR, either word *našeho* 'our' or *příští* 'next' may be reduced (and the whole rows representing these words) in an arbitrary order – both simplified sentences are grammatically correct and they are complete. It implies that these two words are mutually independent and each of them modifies some word in the respective simplified sentence (see [4] for details).

Based on surface syntactic and morphological categories, pronoun *našeho* 'our' and adjective *příští* 'next' are analyzed as depending on proper name *Karla* 'Karel' and noun *rok* 'year', respectively. These reductions are reflected at *t*-layer as edges between respective nodes, namely edge '*my*.f_APP $\longrightarrow$ *Karel*.c_PAT' and edge '*příští*.f_RSTR $\longrightarrow$ *rok*.f_THL' (Fig. 2). When reducing simple dependents as *našeho* 'our' and *příští* 'next', only delete operation is required. The same is true if a word w can be reduced together with all its valency complementations in one step without a loss of completeness. On the other hand, if word w fills valency requirements of some other word (and thus cannot be simply reduced without a loss of completeness), rewriting operation is used for indicating the completeness of the original sentence (the reduced word w is rewritten by its syntactic category; such a structure is interpreted as a complete structure).

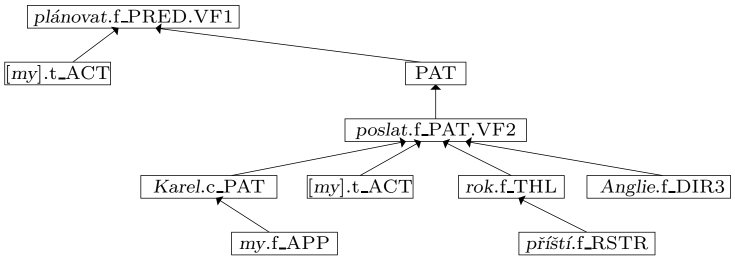

Fig. 2. Tectogrammatical tree for example sentence (1)

We can continue in a similar way and build tectogrammatical tree on the basis of individual reduction steps; the final tectogrammatical tree is in Figure 2.

2 Restarting Automata as a Formal Framework for FGD

Simple restarting automaton – *t*-sRL-automaton. An sRL-*automaton* M is a (in general) nondeterministic machine with a finite-state control Q, a finite working alphabet Γ, and a head (window of size 1) that works on a flexible tape delimited by the left sentinel ¢ and the right sentinel \$ ($¢, \$ \notin \Gamma$). For an input $w \in \Gamma^*$, the initial tape inscription is $¢w\$$. To process this input, M starts in its initial state q_0 with its window over the left end of the tape, scanning the left sentinel ¢. According to its transition relation, M performs *move-right* and *move-left steps*, which shift the window one position to the right or left, respectively, thereby possibly changing the state of M, *rewriting steps*, which rewrite the content of the window without a further move and change the state, and *delete steps*, which delete the content of the window (thus shorten the tape), change the state, and shift the window to the right neighbor of the deleted symbol. Of course, neither the left sentinel ¢ nor the right sentinel \$ must be deleted. At the right end of the tape M either halts and accepts, or it halts and rejects, or it *restarts*, that is, it places its window over the left end of the tape and reenters the initial state. It is required that before the first restart step and also between any two restart steps, M executes at least one delete operation.

A *configuration* of M is a string $\alpha q \beta$ where $q \in Q$, and either $\alpha = \lambda$ and $\beta \in \{¢\} \cdot \Gamma^* \cdot \{\$\}$ or $\alpha \in \{¢\} \cdot \Gamma^*$ and $\beta \in \Gamma^* \cdot \{\$\}$; here q represents the current state, $\alpha\beta$ is the current content of the tape, and it is understood that the window contains the first symbol of β. A *restarting configuration* is of the form $q_0¢w\$$, where $w \in \Gamma^*$.

A *cycle* starts in a restarting configuration, the window is moved along the tape by performing some operations until a restart operation is performed. If after a restart no further restart operation is performed, each finite computation necessarily finishes in a halting configuration – such a subcomputation is called a *tail*. We assume that no delete and rewrite operation is executed in a tail computation.

We use the notation $u \vdash_M^c v$ to denote a cycle of M that begins with the restarting configuration $q_0 ¢ u \$$ and ends with the restarting configuration $q_0 ¢ v \$$; the relation $\vdash_M^{c^*}$ is the reflexive and transitive closure of $\vdash_M^c$.

An input $w \in \Gamma^*$ is *accepted* by M, if there is an accepting computation which starts with the restarting configuration $q_0 ¢ w \$$. By $L_C(M)$ we denote the language consisting of all words accepted by M; we say that M *accepts the characteristic language* $L_C(M)$. By $S(M)$ we denote the *simple language of* M, which consists of all words that M accepts by tail computations. Observe that, for each $w \in \Gamma^*$, we have $w \in L_C(M)$ if and only if $w \vdash_M^{c^*} v$ holds for some word $v \in S(M)$.

A t-sRL-*automaton* ($t \geq 1$) is an sRL-automaton which uses at most t delete operations in a cycle, and for each $v \in S(M)$ it holds $|v| \leq t$.

Remark 1. Let us note, that t-sRL-automata are two-way nondeterministic automata which can check the whole sentence prior to any changes in a cycle. It resembles a linguist, who can read the whole sentence first and reduce the sentence in a correct way afterward. We choose nondeterministic model in order to obtain various sequences of possible reductions.

Similarly as in [7], we can describe a t-sRL-automaton by (meta-)instructions of the following two forms. A *restarting instruction* over Γ is defined as:

$$I_{\mathrm{R}} = (¢ \cdot E_0, a_1 \to b_1, E_1, a_2 \to b_2, E_2, \ldots, E_{s-1}, a_s \to b_s, E_s \cdot \$),$$

where $s \in \{1, \ldots t\}$, $b_1, \ldots, b_s \in \{\lambda\} \cup \Gamma$, λ denotes the empty string, $E_0, E_1, \ldots, E_s$ are regular languages over Γ (often represented by regular expressions), and $a_1, a_2, \ldots, a_s \in \Gamma$ correspond to symbols that are processed during the corresponding cycle of M; either all of them are deleted or one of them is rewritten and the rest is deleted. When trying to execute I_{R} starting from a configuration $q_0 ¢ w \$$, M gets stuck (and so reject), if w does not admit a factorization of the form $w = v_0 a_1 v_1 a_2 \ldots v_{s-1} a_s v_s$ such that $v_i \in E_i$ for all $i = 0, \ldots, s$. On the other hand, if w admits factorizations of this form, then one of them is chosen nondeterministically, and $q_0 ¢ w \$$ is transformed onto $q_0 ¢ v_0 b_1 v_1 b_2 \ldots v_{s-1} b_s v_s \$$.

Tails of accepting computations are described by *accepting instructions* over Γ of the form:

$$I_{\mathrm{A}} = (¢ \cdot E \cdot \$, \mathsf{Accept}),$$

where E is a finite language over Γ. In a configuration $¢ z \$$, a tail computation controlled by I_{A} finishes by accepting if $z \in E$, otherwise the computation halts with rejection.

Further we refer to a t-sRL-automaton as to a tuple $M = (\Gamma, ¢, \$, R(M), A(M))$, where Γ is a characteristic vocabulary (alphabet), $¢$ and $\$$ are sentinels not belonging to Γ, $R(M)$ is a finite set of restarting instructions over Γ and $A(M)$ is a finite set of accepting instructions over Γ.

The following property of restarting automata has a crucial role for modeling the analysis by reduction.

(Correctness Preserving Property). A t-sRL-automaton M is *correctness preserving* if $u \in L_C(M)$ and $u \vdash_M^{c^*} v$ imply that $v \in L_C(M)$.

It is known that all deterministic sRL-automata are correctness preserving. On the other hand, it is easy to design a nondeterministic sRL-automaton which is not correctness preserving (see [7]).

Enhanced *t*-sRL-automata. Enhanced t-sRL-automaton M_o is an extension of a t-sRL-automaton; it is described by instructions enhanced with trees. These trees are used to assign a tree structure to items (cells of the tape containing symbols) and consequently to assign an output to each accepting computation. Enhanced instruction can attach a tree to any item containing symbol which is not deleted during the corresponding cycle.

We will use tree structures denoted as DR-trees (Delete-Rewrite trees); DR-tree is a rooted ordered tree with edges oriented from its leaves to its root. Nodes of the trees are (some of) items deleted or rewritten during individual cycles. Each item has its horizontal position, which preserves left-to-right ordering of the input word. Vertical position of a node corresponds to the number of rewritings on the item(s) with the same horizontal position. The root of such a tree is one of the nodes of the tree which remain on the tape after an accepting tail. The edges of such trees are of the following two types:

- an *oblique* edge connects a marked item to some other marked item;
- a *vertical edge* connects a rewritten item to the original item. The horizontal position of the rewritten item is the same as the horizontal position of the original item. The vertical position of the rewritten item is by one higher than that of the original item. We can consider rewriting as creating a new item (cell) containing a new symbol and placing it in the same position of the tape as the original item.

Formally, a DR-tree $T = (V, H)$ is a rooted tree consisting of a finite set of nodes V and a finite set of oriented edges $H \subseteq V \times V$.

1. *A node* $u \in V$ is a triple $u = [i, j, a]$, where i, j are integers and a is a symbol from an alphabet Γ. Index i represents horizontal position of u. Index j represents vertical position of u and equals to the number of rewritings performed till a appeared in the corresponding cell of the tape (it is 0 when the cell was not rewritten).
2. Each *edge* $h = ([i_u, j_u, a], [i_v, j_v, b]) \in H$ is either oblique or vertical. We say that h is:
   - an *oblique edge*: if $i_u \neq i_v$;
   - a *vertical (rewriting) edge*: if $i_u = i_v$ and $j_v = j_u + 1$.

DR-trees are used in enhanced instructions for storing structural information contained in the deleted or rewritten parts of an input sentence. This information is combined with the DR-structures from accepting instructions into resulting trees.

Enhanced restarting instructions. Two kinds of restarting instructions are distinguished: instructions that rewrite one symbol and delete some other symbols, and instructions that delete symbols only.

A. An enhanced restarting instruction which rewrites one symbol is of the form:

$$I_{\mathrm{R}} = (¢ \cdot E_0, [a_1 \to \lambda]_1, E_1, [a_2 \to \lambda]_2, E_2, \ldots, E_{r-1}, [a_r \to b]_r, \\ E_r, [a_{r+1} \to \lambda]_{r+1}, \ldots E_{s-1}, [a_s \to \lambda]_s, E_s \cdot \$, T_{\mathrm{R}}),$$

where $I = (¢ \cdot E_0, a_1 \to \lambda, E_1, a_2 \to \lambda, \ldots, E_{r-1}, a_r \to b, E_r, a_{r+1} \to \lambda, \ldots, a_s \to \lambda, E_s \cdot \$)$ is a restarting instruction of an t-sRL-automaton. This instruction rewrites exactly one symbol a_r onto b and deletes symbols a_j, for $j = 1, \ldots, r-1, r+1, \ldots, s$. $T_{\mathrm{R}} = (V, H)$ is a DR-tree with $V \subseteq \{[i, 0, a_i] \mid 1 \le i \le s\} \cup \{[r, 1, b]\}$, root $[r, 1, b]$, oblique edges between nodes $[1, 0, a_1], \ldots, [s, 0, a_s]$ and one rewriting edge $([r, 0, a_r], [r, 1, b])$.

B. An enhanced restarting instruction which deletes only is of the form:

$$I_{\mathrm{D}} = (¢ \cdot E_0, [a_1 \to \lambda]_1, E_1, [a_2 \to \lambda]_2, E_2 \ldots, a_{r-1}, E_{r-1}, [a_r]_r, \\ E_r, [a_{r+1} \to \lambda]_{r+1}, \ldots, [a_{s+1} \to \lambda]_{s+1}, E_s \cdot \$, T_{\mathrm{D}}),$$

where $I = (¢ \cdot E_0, a_1 \to \lambda, E_1, a_2 \to \lambda, E_2, \ldots, E_{r-1} \cdot a_r \cdot E_r, a_{r+1} \to \lambda, \ldots, a_s \to \lambda, E_s \cdot \$)$ is a restarting instruction of an t-sRL-automaton. This instruction deletes symbols a_j, for $j = 1, \ldots, r-1, r+1, \ldots, s$. The symbol a_r is not deleted and actually denotes the item which will become the root of the DR-tree T_{D}. Thus $T_{\mathrm{D}} = (V, H)$, where $V \subseteq \{[i, 0, a_i] \mid 1 \le i \le s\}$, H contains oblique edges only and the root of T_{D} is $[r, 0, a_r]$.

Example 2. Let us consider the language $L_{abc} = \{a^n b^n c^n \mid n > 0\}$. During one reduction of a word (sentence) $a^n b^n c^n$, for some $n \ge 2$, we can delete the last symbol a, the symbol b preceding the last b will be considered for the root of DR-tree T_{D}, the last symbol b as well as the first symbol c will be deleted. The DR-tree T_{D} represents dependencies between deleted symbols of the word; it contributes to an incrementally built resulting tree. Hence the automaton has a single restarting instruction $I_{\mathrm{D}} = (¢ \cdot a^*, [a \to \lambda]_1, b^*, [b]_2, \lambda, [b \to \lambda]_3, \lambda, [c \to \lambda]_4, c^* \cdot \$, T_{\mathrm{D}})$, where $T_{\mathrm{D}} = (\{[2,0,b], [3,0,b], [4,0,c]\}, \{([4,0,c], [3,0,b]), ([3,0,b], [2,0,b])\})$ is depicted in Fig. 3 on the left. The DR-tree T_{D} indicates that the deleted symbol b depends on the non-deleted symbol b, that the deleted c depends on the deleted b, and further that the also deleted symbol a is not included into the dependency structure at all. Several trees different from T_{D} could be used, too. E.g., $T'_1 = (\{[2,0,b], [3,0,b], [4,0,c]\}, \{([3,0,b], [2,0,b]), ([4,0,c], [2,0,b])\})$ (Fig. 3 middle); this type of a DR-tree we will be later ruled-out.

$$\begin{tikzcd}
{[2,0,b]} & & & {[2,0,b]} & & & {[1,0,b]} & \\
& {[3,0,b]} \arrow[ul] & & {[3,0,b]} \arrow[u] & {[4,0,c]} \arrow[ul] & & & {[2,0,c]} \arrow[ul] \\
& & {[4,0,c]} \arrow[ul] & & & & &
\end{tikzcd}$$

Fig. 3. DR-trees T_{D} (left), T'_1 (middle) and T_{A} (right)

Enhanced accepting instructions. Enhanced accepting instructions cannot delete symbols, they can mark some symbols (items) of the tape and combine them into a resulting tree. They are of the form:

$$I_{\mathrm{A}} = (¢ \cdot E_0, [a_1]_1, E_1, [a_2]_2, E_2 \ldots, [a_s]_s, E_s \cdot \$, T_{\mathrm{A}}, \mathsf{Accept}),$$

where $I = (¢ \cdot E_0 \cdot a_1 \cdot E_1 \cdot a_2 \cdot E_2 \ldots a_s \cdot E_s \cdot \$, \mathsf{Accept})$ is an accepting instruction of an t-sRL-automaton. An t-sRL-automaton cannot rewrite during a tail computation, hence the tree $T_{\mathrm{A}} = (V, H)$ is a rooted DR-tree without rewriting edges. $V = \{[i, 0, a_i] \mid 1 \leq i \leq s\}$, H contains oblique edges only and the root of T_{A} is an arbitrary node from V.

Example 3. For the language $L_{abc} = \{a^n b^n c^n \mid n > 0\}$ from Example 2 we will need an accepting instruction, too. We can use the following

$$I_{\mathrm{A}} = (¢ \cdot a, [b]_1, \lambda, [c]_2, \$, T_{\mathrm{A}}, \mathsf{Accept}),$$

where $T_{\mathrm{A}} = (\{[1, 0, b], [2, 0, c]\}, \{([2, 0, c], [1, 0, b])\})$ (Fig. 3 right). Using this instruction, the item containing the last symbol b will become the root of the created tree and the last symbol c will become his descendant.

Computations of enhanced *t*-sRL-automata – combining DR-trees. Informally, each enhanced restarting instruction attaches a DR-tree to some item of the tape containing a non-deleted symbol. When an item with attached tree is used as a node of another tree in a subsequent reduction, it preserves the former descendant nodes. In this way bigger and bigger trees can be build.

An accepting computation of an enhanced t-sRL-automaton starts in an initial configuration with the input sentence (word) written on its tape. Automaton M performs several cycles according to its restarting instructions and the computation finishes by an accepting tail. Of course, in each cycle the items of the tape can contain several non-connected trees. The final accepting instruction produces the resulting DR-tree. The set of all DR-trees which can be obtained as a result of some accepting computations of an enhanced t-sRL-automaton M are called the TR-language of M and we denote it as $\mathsf{TR}(M)$.

Remark 2. Note that when we apply an enhanced instruction, the first elements of the nodes (which are small integers in DR-trees in the instructions of M) of resulting trees are always replaced by original positions of the corresponding items in the input word.

In step k of a computation of an enhanced t-sRL-automaton, we can record the number of rewritings applied on each non-deleted symbol from the beginning of the computation. To each symbol a_i forming the word $w = a_1 a_2 ... a_n$ on the tape after performing step k, we can assign a triple (h_i, v_i, a_i) containing its position h_i in the original (input) word and the number v_i of rewritings applied to the original symbol located at the position h_i. We denote the sequence $t_k = [h_1, v_1, a_1][h_2, v_2, a_2] \ldots [h_n, v_n, a_n]$ as an *extended tape content at step k*. Obviously, $1 \leq h_1 < h_2 < \ldots < h_n$ and $v_i \geq 0$, for all $i = 1, \ldots, n$. For each extended tape content $t' = [h'_1, v'_1, a'_1][h'_2, v'_2, a'_2] \ldots [h'_m, v'_m, a'_m]$, we have

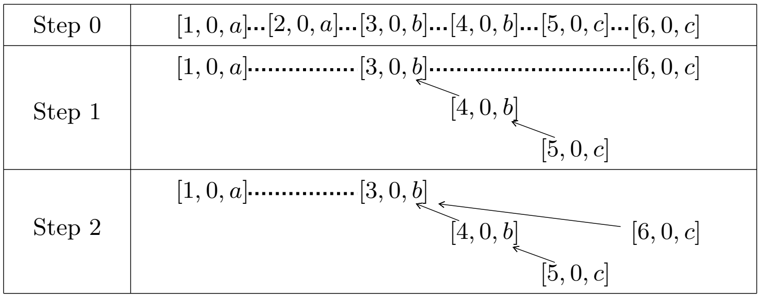

Fig. 4. A sample computation

a unique string of symbols $a'_1 \dots a'_m$. In particular, if w is an input word, i.e. prior to any reduction, we define the initial extended tape content as $\mathsf{Sp}(w) = [1, 0, a_1][2, 0, a_2] \dots [n, 0, a_n]$.

Example 4. Let $M = (\{a, b, c\}, ¢, \$, R(M), A(M))$ be an enhanced sRL-automaton with $R(M) = \{I_{\mathrm{D}}\}$, where I_{D} is the restarting instruction from Example 2, $A(M) = \{I_{\mathrm{A}}\}$, where I_{A} is the accepting instruction from Example 2. Then a computation of M on input word (sentence) *aabbcc* will produce the DR-tree depicted in the second step in Fig. 4. The root of the tree $[3, 0, b]$ corresponds to the third input symbol. The node $[5, 0, c]$ denotes the last but one symbol c which is dependent on the fourth symbol (b).

3 Representation of FGD by Enhanced *t*-sRL-Automaton

Our ultimate goal is to model FGD. In this section we introduce an enhanced t-sRL-automaton $\mathsf{M}_{\mathsf{FGD}}$ which makes it possible to describe relations between characteristic language, analysis by reduction, tectogrammatical dependency structures, characteristic relation and individual language layers that create the basis of (the formal framework for) FGD.

Let Σ, Γ be alphabets and $\Sigma \subseteq \Gamma$. In the following $\mathsf{Pr}^{\Sigma}(z)$, where $z \in \Gamma$, denotes the projection from Γ^* onto Σ^*, that is, Pr^{Σ} is the morphism defined by $a \mapsto a$ (for $a \in \Sigma$) and $A \mapsto \lambda$ (for $A \in \Gamma \setminus \Sigma$).

$\mathsf{M}_{\mathsf{FGD}} = (\Gamma, ¢, \$, R, A)$ is an enhanced nondeterministic t-sRL-automaton, which is *correctness preserving*. Its characteristic alphabet (vocabulary) consists of four parts $\Gamma = \Sigma_w \cup \Sigma_m \cup \Sigma_a \cup \Sigma_t$, which correspond to the respective layers of FGD (Sect. 1.1). $\mathsf{M}_{\mathsf{FGD}}$ can rewrite only symbols from Σ_t and such symbols can be rewritten onto symbols from Σ_t only. Recall that the symbols from individual layers can be quite complex. E.g., '*plánujeme*' is a symbol from the alphabet (vocabulary) Σ_w, '*plánovat*.VB-P-' is a symbol from Σ_m, 'Pred' is a symbol from Σ_a and '*plánovat*.f_PRED.VF1' is a symbol from Σ_t (see Fig. 1).

A language of layer $\ell \in \{w, m, s, t\}$ *accepted by* $\mathsf{M}_{\mathsf{FGD}}$ (denoted as $L_\ell(\mathsf{M}_{\mathsf{FGD}}) = \mathsf{Pr}^{\Sigma_\ell}(L_C(\mathsf{M}_{\mathsf{FGD}}))$) is the set of all sentences (strings) obtained from $L_C(\mathsf{M}_{\mathsf{FGD}})$

by removing all the symbols not belonging to the alphabet Σ_ℓ. In particular, $L_w(\mathsf{M_{FGD}})$ represents the set of all correct sentences defined by $\mathsf{M_{FGD}}$.

The characteristic language $L_C(\mathsf{M_{FGD}})$ contains input sequences (over Σ_w) interleaved with language information of the form of symbols from $\Sigma_m \cup \Sigma_a \cup \Sigma_t$. The language of correct Czech sentences is $L_w = \mathsf{Pr}^{\Sigma_w}(L_C(\mathsf{M_{FGD}}))$.

During an accepting computation, the automaton $\mathsf{M_{FGD}}$ collects exactly all the items with tectogrammatical symbols (symbols from Σ_t) into nodes of a resulting DR-tree. Remember that the TR-language of $\mathsf{M_{FGD}}$ represents the set of meanings described by FGD.

Let $z \in L_C(\mathsf{M_{FGD}})$ and let $\mathsf{TR}(z, \mathsf{M_{FGD}})$ denote the set of all DR-trees from $\mathsf{TR}(\mathsf{M_{FGD}})$ resulting from accepting computations of $\mathsf{M_{FGD}}$ on z. It is possible to formulate detailed requirements put on $\mathsf{M_{FGD}}$ in order to obtain *single* tree for each such input z (i.e. $|\mathsf{TR}(z, \mathsf{M_{FGD}})| = 1$, see Appendix).

Remark 3. Let us note that $L_t(\mathsf{M_{FGD}})$ is designed as a deterministic context-free language. Readers familiar with restarting automata can see that $L_C(\mathsf{M_{FGD}})$ is a deterministic context-sensitive language and $L_w(\mathsf{M_{FGD}})$ is a context-sensitive language.

Now we can define TR-*characteristic relation* $\mathsf{TSH}(\mathsf{M_{FGD}})$ of the automaton $\mathsf{M_{FGD}}$, which is the relation between sentence on the (w-layer) and its meaning (t-layer).

$$\mathsf{TSH}(\mathsf{M_{FGD}}) = \{(u, t) \mid \exists y \in L_C(\mathsf{M_{FGD}}), u = \mathsf{Pr}^{\Sigma_w}(y) \text{ and } t \in \mathsf{TR}(y, \mathsf{M_{FGD}})\}.$$

Remark 4. TR-characteristic relation represents important relations in the description of natural language – the relations of synonymy and ambiguity (homonymy). From the characteristic relation, the significant notions of analysis and synthesis can be derived.

For each $t \in \mathsf{TR}(\mathsf{M_{FGD}})$ we introduce TSH-*synthesis* using $\mathsf{M_{FGD}}$ as the set of wordforms u which are in TSH relation with the DR-tree t. Formally:

$$\textit{synt}\text{-}\mathsf{TSH}(\mathsf{M_{FGD}}, t) = \{u \mid (u, t) \in \mathsf{TSH}(\mathsf{M_{FGD}})\} .$$

Altogether, TSH-synthesis links the tectogrammatical representation, i.e. DR-tree t from $\mathsf{TR}(\mathsf{M_{FGD}})$ to all corresponding sentences u belonging to $L_w(\mathsf{M_{FGD}})$. This notion makes it possible to study synonymy and its degree.

Finally we introduce a notion dual to the TSH-synthesis – the notion of TSH-*analysis of a string u using* $\mathsf{M_{FGD}}$:

$$\textit{anal}\text{-}\mathsf{TSH}(\mathsf{M_{FGD}}, u) = \{t \mid (u, t) \in \mathsf{TSH}(\mathsf{M_{FGD}})\} .$$

For a given sentence u, TSH-analysis returns all its possible tectogrammatical representations t from $\mathsf{TR}(\mathsf{M_{FGD}})$. Hence it models ambiguity of particular surface sentences. This notion represents a formal definition of complete syntactic-semantic analysis using $\mathsf{M_{FGD}}$.

Reduction steps and DR-structures. $\mathsf{M_{FGD}}$ is a restricted instance of an enhanced t-sRL-automaton.

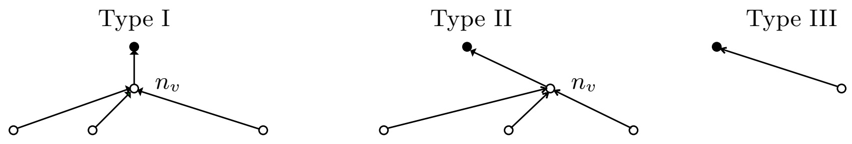

Fig. 5. Types of DR-trees used in $\mathsf{M}_{\mathsf{FGD}}$

A. We distinguish three types of restarting instructions used by $\mathsf{M}_{\mathsf{FGD}}$ (Fig. 5):

Type I instructions are rewriting instructions that process valency. Their corresponding trees are of the height 2 with a single inner node n_v representing a rewritten word v and one or more leaves representing valency complementations of v. The inner node n_v is connected to the root by a vertical edge, the leaves are attached to the inner node by oblique edges only. The rewriting (vertical) edge solves an incompleteness that would arise if v is simply deleted: v is rewritten by its syntactic category; the resulting structure is interpreted as complete with respect to the t-layer.

Type II instructions are deleting instructions that also process a word together with its valency complementations. Their corresponding trees are of the height 2 with a single inner node n_v representing a word v and leaves representing valency complementations of v. The inner node n_v is connected to the root by an oblique edge, the leaves are attached to n_v by oblique edges only. This type of instructions is used if v together with its complementations can be simply deleted without a loss of completeness at the t-layer.

Type III instructions are deleting instructions that process free modifications. The corresponding DR-trees have a root and only one leaf connected to the root by an oblique edge. These instructions are used to process simple reductions which delete a single dependent word.

B. Accepting instructions contain only DR-trees with all edges pointing to the root that corresponds to a governing node of a sentence. The root will become the root of the resulting tree of the whole computation.

C. Each resulting tree $T \in \mathsf{TR}(\mathsf{M}_{\mathsf{FGD}})$ is *projective* (with respect to its descendants); i.e., for each node n of the tree T all its descendants constitute a contiguous segment in the horizontal ordering of nodes of the tree T.

3.1 Concluding Remarks

In this paper, we pursue our studies of a formal model for natural language (Czech in particular) presented in [4,5]. We extend the presentation of the methodology of FGD so that it outputs neither lists of words nor lists of symbols but (tectogrammatical) DR-trees. Such DR-trees can express valencies and simple dependencies in Czech sentences. The model presented in this article captures synonymy and ambiguity as a relation between (surface) sentences and their DR-trees. In this way, we can describe the relation between analysis by reduction and dependency structures at the tectogrammatical layer in detail.

Moreover, we outline a formalization of the basic methodology of FGD in terms of automata theory (see Appendix). This is the main point of our paper as this methodology was presented till now in a way usual in traditional linguistics – that is, quite informally using numerous linguistic examples only.

We envisage that the proposed methodology is not FGD-specific and that similar approach can be used to obtain a formal frame for other language descriptions, as e.g. those presented in [6] and [3]. We plan to focus on them in the near future.

Appendix. Formal definition of enhanced computations of t-sRL-automata as well as more details on the principles of FGD can be found in the Appendix posted at the webpage `http://ufal.mff.cuni.cz/~lopatkova/lata10/app.pdf`.

Acknowledgment. The paper reports on the research supported by the grants of GAČR P202/10/1333 and P103/10/0783 and by the grant of MŠMT ČR MSM0021620838.

References

1. Gramatovici, R., Martín-Vide, C.: Sorted Dependency Insertion Grammars. Theor. Comput. Sci. 354(1), 142–152 (2006)
2. Holan, T., Kuboň, V., Oliva, K., Plátek, M.: Two Useful Measures of Word Order Complexity. In: Polguére, A., Kahane, S. (eds.) Proceedings of the Workshop Processing of Dependency-Based Grammars, Montréal, Quebeck, pp. 21–28 (1998)
3. Kunze, J.: Abhängigkeitsgrammatik. Studia Grammatica, vol. XII. Akademie Verlag, Berlin (1975)
4. Lopatková, M., Plátek, M., Kuboň, V.: Modeling Syntax of Free Word-Order Languages: Dependency Analysis by Reduction. In: Matoušek, V., Mautner, P., Pavelka, T. (eds.) TSD 2005. LNCS (LNAI), vol. 3658, pp. 140–147. Springer, Heidelberg (2005)
5. 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)
6. Mel'čuk, I.A.: Dependency Syntax: Theory and Practice. State University of New York Press, Albany (1988)
7. Messerschmidt, H., Mráz, F., Otto, F., Plátek, M.: Correctness Preservation and Complexity of Simple RL-automata. In: Ibarra, O.H., Yen, H.-C. (eds.) CIAA 2006. LNCS, vol. 4094, pp. 162–172. Springer, Heidelberg (2006)
8. Otto, F.: Restarting Automata and Their Relation to the Chomsky Hierarchy. In: Ésik, Z., Fülöp, Z. (eds.) DLT 2003. LNCS, vol. 2710, pp. 55–74. Springer, Heidelberg (2003)
9. Sgall, P., Hajičová, E., Panevová, J.: The Meaning of the Sentence in Its Semantic and Pragmatic Aspects. Reidel, Dordrecht (1986)