Lexically Syntactic Characterization by Restarting Automata

Martin Plátek[1]*, František Mráz[1], and Dana Pardubská[2†]

[1] Charles University, Department of Computer Science
Malostranské nám. 25, 118 00 PRAHA 1, Czech Republic
martin.platek@mff.cuni.cz, frantisek.mraz@mff.cuni.cz

[2] Comenius University in Bratislava, Department of Computer Science
Mlynská Dolina, 84248 Bratislava, Slovakia
pardubska@dcs.fmph.uniba.sk

Abstract: Our long-term goal is to propose and support an advanced formal (and hopefully also software) environment (framework) for Functional (Generative) Description (FGD) of Czech ([8, 13]), and for similar formal descriptions(see e.g. [6]). This framework should enable to describe the grammaticality and ungrammaticality of a language in a building-kit way. This paper creates one further step to achieve this goal.

1 Introduction

We introduce and study the notion of *lexically syntactic characterization* (LSC) and its complexity features by *h-lexicalized two-way restarting automata* (hRLWW(i)-automata), that can rewrite at most i times per cycle, for $i \geq 1$, move in both directions (RL), and can re(w)rite using two alphabets (WW). Lexically syntactic characterization creates a characterization of a lexicalized syntax of a language. It is composed of four components: *basic language, h-proper language, h-lexicalized syntactic analysis, and analysis by reduction*. The h-lexicalized syntactic analysis formalizes the informal concept of lexical disambiguation of sentences. It is supposed that analysis by reduction satisfies the important basic correctness preserving property in order to express the full syntactic disambiguation of basic vocabulary (alphabet).

We stress the sensitivity of syntactic characterizations on the size of windows of the automata, on the number of allowed rewrite operations in one reduction of the automata, and on types of rewrite operations. The LSC's are sensitive on the mentioned features in a similar way for infinite as well as for finite (individual) syntactic characterizations. We present in that way useful tools for new types of complexity classifications for syntactic phenomena, and we observe that these phenomena in Czech are with respect to this classifications often simple.

One of our long-term goals is to cover an essential gap in theoretical tools supporting computational and corpus linguistics. Chomsky's and other types of phrase-structure grammars do not support syntactic lexical disambiguation nor analysis by reduction as these grammars work with categories bound to individual constituents related to constituent syntactic analysis. They do not support modeling of analysis by reduction with any kind of correctness preserving property, they do not support any type of sensitivity to the size of individual grammar (automata) rules (see several normal forms for context-free grammars, like Chomsky normal form [2]), and, finally, they do not support any kind of natural classification of finite syntactic constructions related to (natural) languages.

On the other hand, in traditional and corpus linguistics, only finite language phenomena can be observed. Now the lexically syntactic characterizations of hRLWWC(i)-automata with fixed window size, and in strong or weak cyclic form allow common classifications of finite syntactic phenomena as well as classifications of their infinite relaxations. All these classifications are based on the basic correctness preserving property and the strong (weak) cyclic form. The concept of hRLWWC(i)-automaton offers a rich set of constraints for expressing the grammaticality and ungrammaticality of individual natural language phenomena which are here formally expressed by LSC.

Let us recall that for restarting automata the (simple or general) monotonicity property characterizes context-free languages. We distinguish in a new way degrees of complexity of finite and infinite syntactic characterizations in order to naturally classify natural-language syntactic phenomena, which should not be considered (by linguistic intuition) as context-free. We are able to capture, e.g., the well-known Dutch sentence example below, see, e.g., the discussion in [5], which is, on one hand, finite, i.e. formally it is regular, but on the other hand, it is often considered (informally) as non-context free.

Let us recall that example. It is in the form of one branch of (naive, i.e. without the lexical disambiguation) analysis by reduction.

... *dat Jan Piet Marie de kinderen zag helpen leren zwemmen*

... *that Jan Piet Marie the children saw help teach swim*

The first reduction deletes the (isolated) words "Marie" and "leren":

... *dat Jan Piet de kinderen zag helpen zwemmen*

... *that Jan Piet the children saw help swim*

*The research was partially supported by the grant of the Czech Science Foundation No. 19-05704S by the authors stay at Institute of Computer Science, Czech Academy of Sciences.

†The research is partially supported by VEGA 2/0165/16

The second reduction deletes the (isolated) words "Piet" and "help":

... dat Jan de kinderen zag zwemmen

... that Jan the children saw swim

Example 1 at the end of this paper presents another natural language example with the full lexical disambiguation and full analysis by reduction.

A model of restarting automaton that formalizes lexicalized syntactic disambiguation in a similar way as categorial grammars (see, e.g., [1]) – the *h-lexicalized restarting automaton* (hRLWW) – was introduced in [10]. This model is obtained from the two-way restarting automaton by adding a symbol-to-symbol morphism h that assigns an input symbol to each working symbol. This morphism models the grammatical disambiguation of individual word-forms and punctuation marks. Then the *basic language* $L_{\mathrm{C}}(M)$ of an hRLWW-automaton M consists of all words over the working alphabet of M that are accepted by M, and the *h-proper language* $L_{\mathrm{hP}}(M)$ of M is obtained from $L_{\mathrm{C}}(M)$ through the morphism h.

Further, the set of pairs $\{(h(w),w) \mid w \in L_{\mathrm{C}}(M)\}$, denoted as $L_{\mathrm{A}}(M)$, is called the h-lexicalized syntactic analysis (LSA) by M. Thus, in this setting, the auxiliary symbols themselves play the role of the tagged items. That is, each auxiliary symbol b can be seen as a pair consisting of an input symbol $h(b)$ and some additional syntactico-semantic information (tags, categories).

Analysis by reduction is traditionally learned in Czech schools. It is used to analyze sentences of natural languages with a higher degree of word-order freedom like, e.g., Czech, Latin, or German. Usually, a human reader is supposed to understand the meaning of a given sentence before he starts to analyze it ([12]); h-lexicalized syntactic analysis based on the analysis by reduction (AR) simulates such a behavior by analysis of sentences, where morphological and syntactical tags have been added to the word-forms and punctuation marks (see, e.g., [8]). An important property of analysis by reduction is the so-called correctness preserving property. Using hRLWW(i)-automata the linguistic correctness preserving property is simulated by the formal notion of *basic correctness preserving property*.

Actually, the constrained hRLWW(i)-automata that are in a *strong cyclic form* preserve the essential part of the power of hRLWW(i)-automata but simultaneously they allow to extend the complexity results obtained for the classes of infinite syntactic characterizations also into the classes of finite syntactic characterizations. This is useful for classifications and the learning of individual phenomena in computational and corpus linguistics, where all the (syntactic) observation are of a finite nature. It also allows to design techniques for the localization of syntactic errors (grammar-checking).

Finally, we introduce the formal concept of *lexically syntactic characterization* (LSC), which creates a formal basis for an environment for syntactic characterizations of natural languages. An important component of LSC is analysis by reduction. Analysis by reduction is in its principle non-deterministic and correctness preserving. That is the reason for the use of non-deterministic, correctness preserving restarting automata. Also, the concept g-monotonicity is forced by the modelling of non-deterministic analysis by reduction. The correctness preserving property represents the full disambiguation for the basic alphabet, in other words for the (manually developed) lexical tags and/or categories.

We show some essential refinements of hierarchies related to the Chomsky hierarchy for formal languages in the area of LSC. We obtain in this way new tools for a natural new type of classification of syntactic phenomena connected with lexicalized syntax formulated in the terms of the theory of automata and formal languages. Let us yet stress that the achieved results are obtained through basic languages and the relation between basic and input alphabets (h-morphism) as the basis for this type of building-kit considerations. It is not possible to obtain similar tools by considering the well-known input languages which are commonly used in the automata theory, and in the theory of restarting automata, as well.

2 Definitions

By $\subseteq$ and $\subset$ we denote the subset and the proper subset relation, respectively. Further, we will sometimes use regular expressions instead of the corresponding regular languages. Finally, throughout the paper λ will denote the empty word.

We start with the definition of the two-way restarting automaton.

Definition 1. *Let i be a positive integer. A* two-way restarting automaton*, an* RLWW(i)*-automaton for short, is a machine with a flexible tape and a finite-state control. It is defined through a 9-tuple $M = (Q, \Sigma, \Gamma, \mathfrak{c}, \$, q_0, k, i, \delta)$, where Q is a finite set of states, Σ is a finite input alphabet, and $\Gamma (\supseteq \Sigma)$ is a finite working alphabet. The symbols from $\Gamma \setminus \Sigma$ are called* auxiliary symbols*. Q and Γ are disjoint. Further, the symbols $\mathfrak{c}, \$ \notin \Gamma$, called* sentinels*, are the markers for the left and right border of the workspace, respectively, $q_0 \in Q$ is the initial state, $k \geq 1$ is the size of the* read/write window*, $i \geq 1$ is the number of allowed rewrites in a cycle (see later), and*

$$\delta : Q \times \mathscr{PC}^{(\leq k)} \to \mathscr{P}((Q \times \{\mathsf{MVR}, \mathsf{MVL}, \mathsf{SL}(v)\}) \cup \{\mathsf{Restart}, \mathsf{Accept}, \mathsf{Reject}\})$$

is the transition relation*. Here $\mathscr{P}(S)$ denotes the powerset of a set S, $\mathscr{PC}^{(\leq k)}$ denotes the set of* possible contents *of the read/write window of size k:*

$$(\{\mathfrak{c}\} \cdot \Gamma^{k-1}) \cup \Gamma^k \cup (\Gamma^{\leq k-1} \cdot \{\$\}) \cup (\{\mathfrak{c}\} \cdot \Gamma^{\leq k-2} \cdot \{\$\})$$

and $v \in \mathscr{PC}^{(\leq k-1)}$.

Being in a state $q \in Q$ and seeing $u \in \mathscr{PC}^{(\leq k)}$ in its window, the automaton can perform six different types of transition steps (or instructions):

1. *A* move-right step *assumes that* $(q', \mathsf{MVR}) \in \delta(q,u)$, *where* $q' \in Q$ *and* u *does not end by the right sentinel* \$. *This move-right step causes* M *to shift the window one position to the right and to enter state* q'.

2. *A* move-left step *assumes that* $(q', \mathsf{MVL}) \in \delta(q,u)$, *where* $q' \in Q$ *and* u *does not start with the left sentinel* ¢. *It causes* M *to shift the window one position to the left and to enter state* q'.

3. *A* rewrite step with left shortening *(an* SL-step*) assumes that* $(q', \mathsf{SL}(v)) \in \delta(q,u)$, *where* $q' \in Q$, $v \in \mathscr{P}\mathscr{C}^{(\leq k-1)}$, v *is shorter than* u, *and* v *contains all the sentinels that occur in* u *(if any). It causes* M *to replace* u *by* v, *to enter state* q', *and to shift the window by* $|u| - |v|$ *items to the left – but at most to the left sentinel* ¢ *(that is, the contents of the window is 'completed' from the left, and so the distance to the left sentinel decreases, if the window was not already at* ¢*).*

4. *A* restart step *assumes that* $\mathsf{Restart} \in \delta(q,u)$. *It causes* M *to place its window at the left end of its tape, so that the first symbol it sees is the left sentinel* ¢, *and to reenter the initial state* q_0.

5. *An* accept step *assumes that* $\mathsf{Accept} \in \delta(q,u)$. *It causes* M *to halt and accept.*

6. *A* reject step *assumes that* $\mathsf{Reject} \in \delta(q,u)$. *It causes* M *to halt and reject.*

A *configuration* of an RLWW(i)-automaton M is a word $\alpha q \beta$, 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 contents of the tape, and it is understood that the read/write window contains the first k symbols of β or all of β if $|\beta| < k$. A *restarting configuration* is of the form $q_0 ¢ w \$$, where $w \in \Gamma^*$.

In general, an RLWW(i)-automaton M is *nondeterministic*, that is, $\delta(q,u)$ can contain two or more elements, for some state q and contents of the read/write window u, and thus, there can be more than one computation that start from a given restarting configuration. If this is not the case, the automaton is *deterministic*.

A *computation* of M is a sequence $C = C_0, C_1, \ldots, C_j$ of configurations of M, where C_0 is a restarting configuration and $C_{\ell+1}$ is obtained from C_ℓ by a step of M, for all $0 \leq \ell < j$. In the following we only consider computations of RLWW(i)-automata which are finite and end either by an accept or by a reject step.

Cycles and tails: Any finite computation of an RLWW(i)-automaton M consists of certain phases. A phase, called a *cycle*, starts in a restarting configuration, the window moves along the tape performing non-restarting steps until a restart step is performed and thus a new restarting configuration is reached. If no further restart step is performed, any finite computation necessarily finishes in a halting configuration – such a phase is called a *tail*. It is required that in each cycle RLWW(i)-automaton executes at most i rewrite steps (of type SL) but at least one SL-step. Moreover, it must not execute any rewrite step in a tail.

This induces the following relation of *cycle-rewriting* by M: $u \Rightarrow_M^c v$ iff there is a cycle that begins with the restarting configuration $q_0 ¢ u \$$ and ends with the restarting configuration $q_0 ¢ v \$$. The relation $\Rightarrow_M^{c^*}$ is the reflexive and transitive closure of $\Rightarrow_M^c$. We stress that the cycle-rewriting is a very important feature of an RLWW(i)-automaton. As each SL-step is strictly length-reducing, we see that $u \Rightarrow_M^c v$ implies that $|u| > |v|$. Accordingly, $u \Rightarrow_M^c v$ is also called a *reduction* by M.

A *basic (or characteristic) word* $w \in \Gamma^*$ is accepted by M if there is a computation which starts with the restarting configuration $q_0 ¢ w \$$ and ends by executing an accept step. By $L_{\mathrm{C}}(M)$ we denote the set of all words from Γ^* that are accepted by M; we say that M *recognizes (or accepts) the basic (or characteristic*[1]*) language* L_{C}.

We define the *set of* **c***orrect* **r***eduction***s** *by* M as
$CRS(M) = \{ u \Rightarrow_M^c v \mid u, v \in L_{\mathrm{C}}(M) \}$
the *analysis by reduction of a word* $u \in L_{\mathrm{C}}(M)$ *by* M as
$AR(M,u) = \{ x \Rightarrow_M^c z \in CRS(M) \mid u \Rightarrow_M^{c^*} x \}$, and
the *analysis by reduction recognized by* M as
$AR(M) = \{ AR(M,u) \mid u \in L_{\mathrm{C}}(M) \}$.

Finally, we come to the definition of the h-lexicalized RLWW(i)-automaton.

Definition 2. *Let* i *be a positive integer. An* h-lexicalized RLWW(i)-automaton*, or an* hRLWW(i)-automaton*, is a pair* $\widehat{M} = (M, h)$, *where* $M = (Q, \Sigma, \Gamma, ¢, \$, q_0, k, i, \delta)$ *is an* RLWW(i)*-automaton and* $h : \Gamma \to \Sigma$ *is a letter-to-letter morphism satisfying* $h(a) = a$ *for all input letters* $a \in \Sigma$. *The* basic language $L_{\mathrm{C}}(\widehat{M})$ *of* $\widehat{M}$ *is the language* $L_{\mathrm{C}}(M)$, *and the* analysis by reduction $AR(\widehat{M})$ *is the analysis by reduction* $AR(M)$.

Further we say that $\widehat{M}$ recognizes (or accepts) the h-proper language $L_{\mathrm{hP}}(\widehat{M}) = h(L_{\mathrm{C}}(M))$.

Finally, the set $L_{\mathrm{A}}(\widehat{M}) = \{ (h(w), w) \mid w \in L_{\mathrm{C}}(M) \}$ *is called the (h-)lexicalized syntactic analysis (shortly LSA) by* $\widehat{M}$.

For $x \in \Sigma^*$, $L_{\mathrm{A}}(\widehat{M}, x) = \{ (x, y) \mid y \in L_{\mathrm{C}}(M), h(y) = x \}$ *is the* lexicalized syntactic analysis *for* x *by* $\widehat{M}$. *We see that* $L_{\mathrm{A}}(\widehat{M}, x)$ *is non-empty only for* x *from* $L_{\mathrm{hP}}(\widehat{M})$.

Let us note that LSA formalizes the linguistic notion of *lexical disambiguation of sentences*. Each auxiliary symbol $x \in \Gamma \setminus \Sigma$ of a word from $L_{\mathrm{C}}(\widehat{M})$ can be considered as a disambiguated input symbol $h(x)$.

Definition 3. (Basic Correctness Preserving Property) *Let* M *be an* hRLWW(i)*-automaton. If* $u \Rightarrow_M^{c^*} v$ *and* $u \in L_{\mathrm{C}}(M)$ *induce that* $v \in L_{\mathrm{C}}(M)$, *and therewith* $h(v) \in L_{\mathrm{hP}}(M)$, *and* $(h(v), v) \in L_{\mathrm{A}}(M)$, *then we say that* M *is* basically correctness preserving.

The following fact ensures the transparency for computations of deterministic hRLWW(i)-automata.

[1] The subscript C is preserved from previous papers where basic languages were called characteristic languages.

Fact 1. *Let M be a deterministic* hRLWW(i)*-automaton. Then M is basically correctness preserving.*

Finally, we introduce the concept of *lexically syntactic characterization (LSC)*. Let M be an hRLWW(i)-automaton, and $u \in L_{\mathrm{C}}(M)$. We take as $L_{\mathrm{SC}}(M,u) = \{(u,h(u),AR(M,u))\}$. We say that $L_{\mathrm{SC}}(M,u)$ is the *lexically syntactic characterization of u by M*. Further we take as $L_{\mathrm{SC}}(M) = \{L_{\mathrm{SC}}(M,u) \mid u \in L_{\mathrm{C}}(M)\}$. We say that $L_{\mathrm{SC}}(M)$ is the *lexically syntactic characterization (LSC) recognized by M*.

Notations. For brevity, the prefixes det- and bcpp- will be used to denote the property of being deterministic and basically correctness preserving, respectively. For any class A of automata, $\mathscr{L}_{\mathrm{C}}(\mathsf{A})$ will denote the class of basic languages that are recognized by automata from A, and $\mathscr{L}_{\mathrm{hP}}(\mathsf{A})$ will denote the class of h-proper languages that are recognized by automata from A. $\mathscr{L}_{\mathrm{A}}(\mathsf{A})$ will denote the class of LSA (h-lexicalized syntactic analyses) that are defined by automata from A. $\mathscr{A}_{\mathrm{R}}(\mathsf{A})$ will denote the class of AR's (analyses by reduction) that are defined by automata from A. $\mathscr{L}_{\mathrm{SC}}(\mathsf{A})$ will denote the class of LSC (lexically syntactic characterizations) that are defined by automata from A.

For a natural number $k \geq 1$, $\mathscr{L}_{\mathrm{C}}(k\text{-}\mathsf{A})$, $\mathscr{L}_{\mathrm{hP}}(k\text{-}\mathsf{A})$, $\mathscr{L}_{\mathrm{A}}(k\text{-}\mathsf{A})$, $\mathscr{A}_{\mathrm{R}}(k\text{-}\mathsf{A})$, $\mathscr{L}_{\mathrm{SC}}(k\text{-}\mathsf{A})$ will denote the class of basic languages, h-proper languages, LSA's, AR's, and LSC's respectively, that are recognized by those automata from A that use a read/write window of size at most k. In other words the prefix k- means the restriction on the size of the read/write window.

2.1 Further Refinements, and Constraints on hRLWW(i)-Automata

Here we introduce some constrained types of rewrite steps (instructions) whose introduction is motivated by different types of linguistic reductions.

A *delete-left step*, written as $(q',\mathsf{DL}(v)) \in \delta(q,u)$, is a special type of an SL-step $(q',\mathsf{SL}(v)) \in \delta(q,u)$, where v is a proper (scattered) subsequence of u, containing all the sentinels from u (if any). It causes M to replace u by v (by deleting excessive symbols), to enter state q', and to shift the window by $|u|-|v|$ symbols to the left, but at most to the left sentinel $\mathfrak{c}$.

A *contextual-left step*, written as $(q',\mathsf{CL}(v)) \in \delta(q,u)$, is a special type of DL-step $(q',\mathsf{DL}(v)) \in \delta(q,u)$, where $u = v_1u_1v_2u_2v_3$, $u_1,u_2 \in \Gamma^*$, $|u_1u_2| \geq 1$ and $v = v_1v_2v_3$ such that v contains all the sentinels from u (if any). It causes M to replace u by v (by deleting the factors u_1 and u_2 of u), to enter state q', and to shift the window by $|u|-|v|$ symbols to the left, but at most to the left sentinel $\mathfrak{c}$.

An RLWW(i)-automaton is called an RLWWD(i)-*automaton* if all its rewrite steps are DL-steps. An RLWW(i)-automaton is called an RLWWC(i)-*automaton* if all its rewrite steps are CL-steps.

In the following we will use the corresponding notation also for subclasses of RLWW(i)- and hRLWW(i)-automata. Additionally, for a type X of RLWW(i)-automata and an integer $k \geq 1$, prefix k- will denote the subclass of X of automata of windows size at most k. For example, 3-det-hRLWWC(i) denotes the class of deterministic h-lexicalized RLWWC(i)-automata with window size at most 3.

We recall the notions of *monotonicity* and *g-monotonicity* (see [4]) as an important constraint for computations of RLWW(i)-automata. Let M be an RLWW(i)-automaton, and let $C = C_k, C_{k+1}, \ldots, C_j$ be a sequence of configurations of M, where $C_{\ell+1}$ is obtained by a single transition step from C_ℓ, $k \leq \ell < j$. We say that C is a *subcomputation* of M. If $C_\ell = \alpha q \beta$, then $|\beta|$ is the *right distance* of C_ℓ, which is denoted by $D_{\mathrm{r}}(C_\ell)$. We say that a sub-sequence $(C_{\ell_1}, C_{\ell_2}, \ldots, C_{\ell_n})$ of C, where $k \leq \ell_1 < \ell_2 \cdots < \ell_n \leq j$, is *monotone* if $D_{\mathrm{r}}(C_{\ell_1}) \geq D_{\mathrm{r}}(C_{\ell_2}) \geq \cdots \geq D_{\mathrm{r}}(C_{\ell_n})$. A computation of M is called *monotone* if the corresponding sub-sequence of rewrite configurations is monotone. Here a configuration is called a *rewrite configuration* if in this configuration an SL-step is being applied. Finally, M itself is called *g-monotone* if for each accepting computation of M there is an accepting computation of M with the same first (starting) configuration that is monotone. M is called *monotone* if all its computations are monotone. Note, that by notions of monotonicity the sequence of all rewritings in a sub-computation is considered, and the cycles are not considered. We use the prefix gmon- (mon-) to denote g-monotone (monotone) types of hRLWW(i)-automata. We can see that any mon-hRLWW(i)-automaton is also a gmon-hRLWW(i)-automaton. We work here with the g-monotonicity (cf. [4]) in order to adequately model the non-deterministic character of analysis by reduction. Non-deterministic analysis by reduction is traditionally used to determine the syntactic (in)dependencies in natural language (e.g., Czech) sentences (see, e.g., [7, 6]).

A restriction of the form of restarting automata called *strong cyclic form* can be transferred to hRLWW(i)-automata. An hRLWW M is said to be in *strong cyclic form* if $|uv| \leq k$ for each halting configuration $\mathfrak{c}uqv\$$ of M, where k is the size of the read/write window of M. Thus, before M can halt, it must erase sufficiently many letters from its tape. The prefix scf- will be used to denote restarting automata that are in strong cyclic form. The concept of strong cyclic form is useful for the presented sensitivity properties, and for techniques of grammar-checking (localization of syntactic errors) by hRLWW(i)-automata.

3 On Power and Sensitivity of Lexicalized Syntactic Characterizations

In this section we will study lexicalized syntactic characterizations (LSC) of scf-hRLWW(i)-automata by the study of their components, i.e. basic and h-proper lan-

guages, LSA's, and AR's. We will see that, with respect to LSC, scf-hRLWW(i)-automata (and their variants) are sensitive to several types of constraints, as, e.g., the window size, number of SL-operations (rewritings) in a cycle, (non)determinism, and types of allowed rewrite-operations. Through these constraints we essentially establish and refine the classifications which are derived from the linguistically relevant part of the Chomsky hierarchy and we will do so in several phases. In the first phase we refine the part corresponding to context-sensitive languages by the number of rewritings in a cycle. Next, by using the window size, we refine the individual areas of LSC that are given by the number of rewritings in a cycle. Finally, we use the AR's for the refinement by the nondeterminism and by the three different types of rewritings (SL, DL, CL). We consider all those types of constraints which are highly relevant for linguistic classifications.

The complexity of sentences can be measured also by the number of used reductions. That is the reason to consider the following concepts. For any RLWW(i)-automaton M, we use fin(j)-$L_{\mathrm{C}}(M)$ and fin(j)-$L_{\mathrm{hP}}(M)$ to denote the subsets of $L_{\mathrm{C}}(M)$ and $L_{\mathrm{hP}}(M)$, respectively, consisting of words accepted by computations with at most j reductions (cycles). Analogous notation is used also with any type X of RLWW(i)-automaton: fin(j)-$\mathscr{L}_{\mathrm{C}}(\mathsf{X})$ and fin(j)-$\mathscr{L}_{\mathrm{hP}}(\mathsf{X})$ denote the subclasses of $\mathscr{L}_{\mathrm{C}}(\mathsf{X})$ and $\mathscr{L}_{\mathrm{hP}}(\mathsf{X})$, respectively, consisting of languages accepted by computations with at most j reductions (cycles). Similarly we use the prefix fin(j)- for the classes of lexically syntactic characterizations, syntactic analyses, and analyses by reduction. E.g., fin(j)-$\mathscr{L}_{\mathrm{SC}}$(scf-RLWWC($i$)) denotes the class of syntactic characterizations obtained from $\mathscr{L}_{\mathrm{SC}}$(scf-RLWWC($i$)) by allowing only accepting computations with at most j reductions.

3.1 Sensitivity of scf-hRLWW(i)-Automata

Let us yet note that the prefix bcpp- means the basic correctness preserving property.

Subsequent sections are focused on results related to the sensitivity of scf-hRLWW(i)-automata. In particular, we show the sensitivity of the above mentioned automata on the size of the windows, and on the number of rewritings in a cycle.

Theorem 2. *For all $i,j,k \geq 1$ it holds the following:*

(1) $\mathscr{L}_{\mathrm{C}}(k\text{-det-mon-scf-RLWWC}(i)) \setminus \mathscr{L}_{\mathrm{hP}}(k\text{-scf-hRLWW}(i-1)) \neq \emptyset$,

(2) fin(j)-$\mathscr{L}_{\mathrm{C}}(k\text{-det-mon-scf-RLWWC}(i)) \setminus \mathscr{L}_{\mathrm{hP}}(k\text{-scf-hRLWW}(i-1)) \neq \emptyset$,

(3) fin(j)-$\mathscr{L}_{\mathrm{C}}(k\text{-det-mon-scf-RLWWC}(1)) \setminus \mathscr{L}_{\mathrm{hP}}((k-1)\text{-scf-hRLWW}(i)) \neq \emptyset$.

Proof. (1) For each $i,k \geq 1$, the language $L_1(i,k) = \{a^{k\cdot i\cdot \ell+k} \mid \ell \geq 0\}$ is both the basic and the h-proper language accepted by the following k-det-mon-scf-RLWWC(i)-automaton $M_1^{(i,k)}$. On input a^n, for some $n \geq 0$, the automaton:

1. rejects in a tail, if $n < k$;
2. accepts in a tail, if $n = k$;
3. deletes a^n by performing $\lceil \frac{n}{k} \rceil$ rewrites (all of then at the right end of the tape) and restarts, if $i \cdot k \geq n > k$;
4. rewrites the word a^n into the word $a^{n-k\cdot i}$ by executing i SL-steps each of which deletes the suffix a^k of the current tape contents and restarts, if $n > k \cdot i$.

Evidently, $M_1^{(i,k)}$ can be deterministic, monotone and in strong cyclic form.

Next we show that $L_1^{(i,k)}$ cannot be an h-proper language of any RLWW(i')-automaton in strong cyclic form with $i' < i$. For a contradiction, let M be a k-scf-hRLWW(i')-automaton such that $L_{\mathrm{hp}}(M) = L_1^{(i,k)}$, where $i' < i$. As $w = a^{k\cdot i+k} \in L_1^{(i,k)}$, $|w| > k$, and as M is in strong cyclic form, each accepting computation of M on input w must start by a cycle. As a^k is the only shorter word in $L_1^{(i,k)}$, the automaton must rewrite $a^{k\cdot i+k}$ into a word w' such that $h(w') = a^k$. Hence, it must delete $k \cdot i$ symbols. However, this is not possible, as M can rewrite at most $k \cdot i' < k \cdot i$ symbols in a cycle – a contradiction.

(2) For each $i,j,k \geq 1$, the language fin(j)-$L_{\mathrm{C}}(M_1^{(i,k)}) = \{a^{k\cdot i\cdot \ell+k} \mid j \geq \ell \geq 0\}$ is accepted by computations of the k-det-mon-scf-RLWWC(i)-automaton $M_1^{(i,k)}$ with at most j reductions.

In the same way as in the proof of the above claim, we can show that the language fin(j)-$L_{\mathrm{C}}(M_1^{(i,k)})$ cannot be h-proper language of any k-scf-hRLWWC(i')-automaton with $i' < i$.

(3) It is easy to construct a deterministic k-RLWWC-automaton $M_2^{(k)}$ accepting the language $L_2^{(k)} = \{a^k\}$. On input a^n, for some $n \geq 0$, the automaton:

1. rejects in a tail, if $n < k$;
2. accepts in a tail, if $n = k$;
3. deletes the first occurrence of a and restarts, if $n = \ell \cdot k$ for some $\ell > 1$;
4. deletes the suffix a^k and restarts, if $n = \ell \cdot k + m$ for some $\ell \geq 1$ and $1 \leq m \leq k-1$.

Evidently, $M_2^{(k)}$ is deterministic, monotone and in strong cyclic form. Moreover, $M_2^{(k)}$ accepts $L_2^{(k)}$ in computations which have no cycle.

Each non-empty h-proper language accepted by a $(k-1)$-scf-hRLWW(i)-automaton M contains at least one word of length at most $(k-1)$. The language $L_2^{(k)}$ is non-empty but it does not contain any word of length at most $k-1$. Hence it cannot be the h-proper language of any $(k-1)$-scf-hRLWW(i)-automaton. □

Obviously, $\mathscr{L}_{\mathrm{C}}(\mathsf{X}) \subseteq \mathscr{L}_{\mathrm{hP}}(\mathsf{X})$, for any type X of hRLWW-automata. Hence with Theorem 2 we have the following consequence.

Corollary 1. *For all $i,k \geq 1$ it holds the following:*

(1) $\mathscr{L}_{\mathrm{C}}(k\text{-bcpp-scf-hRLWW}(i)) \subset \mathscr{L}_{\mathrm{C}}(k\text{-bcpp-scf-hRLWW}(i+1))$,

(2) $\mathscr{L}_{\mathrm{hP}}(k\text{-bcpp-scf-hRLWW}(i)) \subset \mathscr{L}_{\mathrm{hP}}(k\text{-bcpp-scf-hRLWW}(i+1))$.

For bcpp-scp-automata without restriction on the size of read/window we have the following.

Theorem 3. *For all $i \geq 1$ it holds the following:*

(1) $\mathscr{L}_{\mathrm{C}}(\text{bcpp-scf-hRLWW}(i-1)) \subset \mathscr{L}_{\mathrm{C}}(\text{bcpp-scf-hRLWW}(i))$,

(2) $\mathscr{L}_{\mathrm{hP}}(\text{bcpp-scf-hRLWW}(i-1)) \subset \mathscr{L}_{\mathrm{hP}}(\text{bcpp-scf-hRLWW}(i))$.

Proof. The inclusion relations in both claims follow directly from the definitions. For any $i \geq 1$, we prove that both inclusions are proper using the following sample language $L_3^{(i)} = \{(a^n b^n)^i \mid n \geq 0\}$. This language is the basic and also the h-proper language of the following 2-det-scf-hRLWWD(i)-automaton $M_3^{(i)}$. On input $w \in \{a,b\}^*$, the automaton:

1. accepts, if $w = \lambda$,
2. deletes i occurrences of ab (one in each segment a^+b^+) and restarts, if w is from $(a^+b^+)^i$.
3. deletes the first occurrence of a and restarts, if w is from $(a^+b^+)^\ell$ for some positive integer $\ell \neq i$,
4. deletes the first occurrence of a and restarts, if w starts by a and w is not from $(a^+b^+)^i$,
5. deletes the first occurrence of a and restarts, if w starts by b and contains at least one a,
6. deletes the last occurrence of b and restarts, if $w = b^\ell$ for some $\ell > 1$,
7. rejects, if $w = b$.

The automaton does not use any auxiliary symbol. It is easy to see that $L(M_3^{(i)}) = L_{\mathrm{C}}(M_3^{(i)}) = L_{\mathrm{hP}}(M_3^{(i)}) = L_3^{(i)}$. As the automaton $M_3^{(i)}$ is deterministic, it has the basic correctness preserving property. Moreover, the automaton is in strong cyclic form.

On the other hand, let us suppose that $L_3^{(i)}$ is the basic language of a k-scf-hRLWW(i')-automaton M, for some $i' < i$. On input $w = (a^n b^n)^i$ for a sufficiently large $n \gg k$, the automaton M must perform at least one cycle $w \Rightarrow_M^c w'$, where $w' = (a^{n-j}b^{n-j})^i$, for some j, $k > j > 0$. For that, at least i rewriting steps are necessary – a contradiction.

In a similar way we can prove that $L_3^{(i)}$ cannot be the h-proper language of any scf-hRLWW(i')-automaton, for any $i' < i$. □

3.2 Relation of LSA and LSC to context-free and context-sensitive languages.

For $k \geq 1$, let k-CFL denote the class $\mathscr{L}_{\mathrm{hP}}(k\text{-gmon-bcpp-scf-hRLWW}(1))$.

Theorem 4. *Let $X \in \{$hRLWW(1), hRLWWD(1), hRLWWC(1)$\}$, and $k \geq 1$. Then*

(1) $\mathscr{L}_{\mathrm{hP}}(\text{bcpp-gmon-scf-}X) = CFL$,

(2) $\mathscr{L}_{\mathrm{C}}(k\text{-bcpp-gmon-scf-}X) \subset k\text{-}CFL$,

(3) $\mathscr{L}_{\mathrm{hP}}(k\text{-bcpp-gmon-scf-}X) \subset CFL$.

Proof. The h-proper languages of (mon-hRLWW(1))-automata are context-free because the basic languages of monotone hRLWW(1)-automata are context-free [10] and the class of context-free languages is closed under the application of morphisms. This can be easily extended to gmon-hRLWW(1)-automata similarly as in [4].

In [10], it was shown that each context-free language is an h-proper language of a det-mon-RLWWC(1)-automaton in a weak cyclic form. The weak cyclic form differs from the strong cyclic form in that it does not require to reject only words of length not longer than the size of the window. However, the proof from [10] can be easily adapted to the strong cyclic form either.

To prove both proper inclusions we can use the context-free language $L_2^{(k+1)}$ which cannot be h-proper (and therefore also not be the basic) language of any k-scf-RLWW(1)-automaton (see the proof of Theorem 3 (3)). □

The first claim of Theorem 4 presents the robustness of the class of h-proper languages with respect to several subclasses of gmon-scf-hRLWW(1)-automata. The claims (2) and (3) support the adequacy of the following notions which establish relations of LSA and LSC to context-free languages and to refined context-free languages.

Notations. We denote for $k \geq 1$ by k-CFLA the class $\mathscr{L}_{\mathrm{A}}(k\text{-gmon-bcpp-scf-hRLWW}(1))$, and by k-CFLSC the class $\mathscr{L}_{SC}(k\text{-gmon-bcpp-scf-hRLWW}(1))$. With this denotations we refine and enhance by hRLWW(1)-automata the concept of (restricted) context-free languages to the concept of context-free lexicalized syntactic analysis and to the concept of context-free lexicalized syntactic characterization. We can see that the union of k-CFL creates the class of CFL. We denote by CFLA the union of k-CFLA, and by CFLSC the union of k-CFLSC, for all $k \geq 1$.

3.3 Hierarchies of LSC's and LSA's.

Notations. For $i \geq 1$ we denote by LSC(i) the class $\mathscr{L}_{\mathrm{SC}}(\text{bcpp-scf-hRLWWC}(i))$ and by LSA(i) the class $\mathscr{L}_{\mathrm{A}}(\text{bcpp-scf-hRLWWC}(i))$. Taking the union over all natural numbers we obtain classes $\mathsf{LSC} = \bigcup_{i \in N} \mathsf{LSC}(i)$ and $\mathsf{LSA} = \bigcup_{i \in N} \mathsf{LSA}(i)$.

Corollary 2. *For all $i \geq 1$ it holds the following:*

(1) $\mathsf{CFLSC} \subset \mathsf{LSC}(1)$, $\mathsf{CFLA} \subset \mathsf{LSA}(1)$,

(2) $\mathscr{L}_{\mathrm{hP}}(\mathsf{bcpp\text{-}scf\text{-}hRLWWC}(i)) \subset \mathsf{CSL}$.

Proof. Claim (1) follows from Theorem 4. To prove assertion (2), note that $\mathsf{RLWW}(i)$-automaton can be simulated by a linear bounded automaton. On the other hand, the context-sensitive language $L_{\mathrm{e}} = \{a^{2^n} \mid n \geq 1\}$ cannot be the h-proper language of any k-scf-$\mathsf{hRLWW}(i)$-automaton M, for any $i,k \geq 1$, as, e.g., it should accept the input word $w = a^{2^{ik}}$. Because $|w| > k$, the automaton must perform at least one cycle $w \Rightarrow^{c}_{M} w'$ and $h(w')$ must belong to the h-proper language of M. However, this is not possible, as the automaton can shorten its tape contents by at most ik symbols in one cycle and therefore $2^{ik} > |w'| > 2^{ik-1}$ and hence $h(w') \notin L_{\mathrm{e}}$. □

The sensitivity of $\mathsf{hRLWW}(i)$-automata on the size of their windows can be utilized to essentially refine the hierarchies of LSC's. These refined hierarchies yield a fine classification of syntactic phenomena in lexicalized syntaxes of natural languages.

Note that prefix k- indicates the window size of the model in mind. So, k-$\mathsf{LSC}(i)$ is the class $\mathscr{L}_{\mathrm{SC}}(k\text{-}\mathsf{bcpp\text{-}scf\text{-}hRLWWC}(i))$; analogously k-$\mathsf{LSA}(i)$, and k-LSA. We say that k-$\mathsf{LSC}(i)$ is the set of *k-restricted lexically syntactic characterizations* of degree i, k-$\mathsf{LSA}(i)$ is the set of *k-restricted lexically syntactic analyses* of degree i. k-LSC is the set of *k-restricted lexically syntactic characterizations* and k-LSA is the set of *k-restricted lexically syntactic analyses.*

Then, the next corollary easily follows from Theorem 3.

Corollary 3. *For all $i \geq 1$, $k \geq 2$ it holds the following:*

(1) k-CFL $\subset (k+1)$-CFL

(2) k-$\mathsf{LSC}(i) \subset (k+1)$-$\mathsf{LSC}(i)$
k-$\mathsf{LSA}(i) \subset (k+1)$-$\mathsf{LSA}(i)$

(3) k-CFLSC $\subset k$-$\mathsf{LSC}(i) \subset k$-$\mathsf{LSC}(i+1)$ $\subset k$-LSC
k-CFLA $\subset k$-$\mathsf{LSA}(i) \subset k$-$\mathsf{LSA}(i+1)$ $\subset k$-LSA.

Let us recall that in order to bound the number of reductions in the computation we add prefix $\mathsf{fin}(j)$ to language constructions class.

Corollary 4. *For all $i,j \geq 1$, $k \geq 2$ it holds the following:*

(1) k-$\mathsf{fin}(j)$-$\mathsf{LSC}(i) \subset k$-$\mathsf{fin}(j)$-$\mathsf{LSC}(i+1)$,
k-$\mathsf{fin}(j)$-$\mathsf{LSA}(i) \subset k$-$\mathsf{fin}(j)$-$\mathsf{LSA}(i+1)$,

(2) k-$\mathsf{fin}(j)$-$\mathsf{LSC}(i) \subset (k+1)$-$\mathsf{fin}(j)$-$\mathsf{LSC}(i)$,
k-$\mathsf{fin}(j)$-$\mathsf{LSA}(i) \subset (k+1)$-$\mathsf{fin}(j)$-$\mathsf{LSA}(i)$.

Proof. The Corollary also follows from Corollary 1. □

The previous two corollaries witnesses the similarity of classifications between infinite and parametrized finite LCS's.

3.4 On Sensitivity of LSC's by analysis by reduction

We start this subsection by a small linguistic example which will help us to demonstrate some of the considerations about sensitivity of LSC's which is derived from the analysis by reduction.

Example 1. *Fig.1 below illustrates analysis by reduction corresponding to lexically disambiguated (tagged) sentence:*

(1) [*Rozhodl*.Pred] [*se*.AuxT] [*dnes*.Adv] [*odstoupit*.Obj] [..AuxK]
'(He) decided – REFL – today – (to) resign – .'
'He decided to resign today.'

that corresponds to the original untagged sentence:

(2) Rozhodl se dnes odstoupit.

We do not let the reflexive particle 'se' to be deleted, because we consider a deletion of a sole reflexive particle a forbidden reduction.

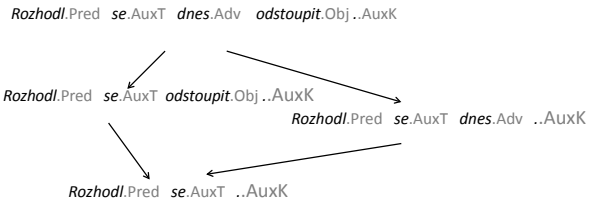

Figure 1: Lexical disambiguation and analysis by reduction of tagged sentence (1).

In order to obtain more fine and practical type of constraint we refine the notion of strong cyclic form. An k-hRLWW M is said to be in *strong cyclic form of degree i* if $|uv| \leq k \cdot i$ for each halting configuration $\mathrm{¢}uqv\$$ of M. The prefix $\mathsf{scf}(i)$- will be used to denote restarting automata that are in strong cyclic form of degree i. We will illustrate the notion by the previous example.

The analysis by reduction from the previous example has two branches. It is not hard to see that it can be simulated by a nondeterministic 1-bcpp-gmon-$\mathsf{scf}(3)$-hRLWWC(1)-automaton M_{ex} with the basic alphabet (vocabulary)

$$\{[\mathit{Rozhodl}.\mathsf{Pred}],\ [\mathit{se}.\mathsf{AuxT}],\ [\mathit{dnes}.\mathsf{Adv}],\ [\mathit{odstoupit}.\mathsf{Obj}],\ [..\mathsf{AuxK}]\},$$

with the input alphabet (vocabulary)

$$\{\mathit{Rozhodl},\ \mathit{se},\ \mathit{dnes},\ \mathit{odstoupit},\ .\}$$

and the morphism h given by the following set of equalities:

$h([\mathit{Rozhodl}.\mathsf{Pred}]) = \mathit{Rozhodl}$, $h([\mathit{se}.\mathsf{AuxT}) = \mathit{se}$,
$h([\mathit{odstoupit}.\mathsf{Obj}]) = \mathit{odstoupit}$, $h([\mathit{dnes}.\mathsf{Adv}]) = \mathit{dnes}$,
$h([..\mathsf{AuxK}]) = \text{'.'}$.

As no deterministic restarting automaton can provide an analysis by reduction with more than one branch, we obtain the following corollary.

Corollary 5. *For all $i,k \geq 1$, $j \geq 3$ it holds the following:*

(1) $\mathscr{A}_{\mathrm{R}}(k\text{-det-mon-scf}(j)\text{-hRLWW}(1)) \subset \mathscr{A}_{\mathrm{R}}(k\text{-bcpp-gmon-scf}(j)\text{-hRLWW}(1))$,

(2) $\mathscr{L}_{\mathrm{SC}}(k\text{-det-mon-scf}(j)\text{-hRLWW}(1)) \subset \mathscr{L}_{\mathrm{SC}}(k\text{-bcpp-gmon-scf}(j)\text{-hRLWW}(1))$,

(3) $\mathscr{A}_{\mathrm{R}}(k\text{-det-scf}(j)\text{-hRLWW}(i)) \subset \mathscr{A}_{\mathrm{R}}(k\text{-bcpp-scf}(j)\text{-hRLWW}(i))$,

(4) $\mathscr{L}_{\mathrm{SC}}(k\text{-det-scf}(j)\text{-hRLWW}(i)) \subset \mathscr{L}_{\mathrm{SC}}(k\text{-scf}(j)\text{-hRLWW}(i))$.

It is not hard to see that similar corollaries hold also if we add different combinations of constraints for the size of the window, for the number of allowed reductions, and for the different types of rewriting.

The following proposition can be easily shown using a set of small (finite) artificial examples of analysis by reduction.

Corollary 6. *For all $i,j,k \geq 1$ it holds the following:*

(1) $\mathscr{A}_{\mathrm{R}}(k\text{-bcpp-scf}(j)\text{-hRLWWC}(i)) \subset \mathscr{A}_{\mathrm{R}}(k\text{-bcpp-scf}(j)\text{-hRLWWD}(i)) \subset \mathscr{A}_{\mathrm{R}}(k\text{-bcpp-scf}(j)\text{-hRLWW}(i))$,

(2) $\mathscr{L}_{\mathrm{SC}}(k\text{-bcpp-scf}(j)\text{-hRLWWC}(i)) \subset \mathscr{L}_{\mathrm{SC}}(k\text{-bcpp-scf}(j)\text{-hRLWWD}(i)) \subset \mathscr{L}_{\mathrm{SC}}(k\text{-bcpp-scf}(j)\text{-hRLWW}(i))$.

4 Conclusion

We have introduced the concept of lexically syntactic characterization (LSC) by hRLWW(i)-automata. Our aim was to characterize exactly the grammatical and ungrammatical syntactic phenomena in terms close to lexicalized syntax of natural languages. LSC characterizes the explicative power of scf-hRLWW(i)-automata by basic languages.

We consider scf-hRLWW(i)-automata which satisfy the basic correctness preserving property. Together, the strong cyclic form and the basic correctness preserving property enforce the sensitivity to the number of rewritings in a cycle, to the size of the window, and the sensitivity with respect to finite syntactic phenomena. We have transferred syntactic features characterizing context-free syntactic phenomena from infinite to parametrized finite LCS's. Finally, we have introduced the concept of degree of strong cyclic form and outlined its meaning for characterization of the complexity of analysis by reduction of individual sentences.

Thanks to the long-time study of Prague Dependency Treebank (PDT), and manually made analysis by reduction on this material, we believe that the above defined class 12-LSC(2) is strong enough to model lexicalized surface syntax of Czech, that is, to model the LSC based on PDT.

Our long-term goal is to propose and support a formal (and possibly also software) environment for a further study and development of Functional Generative Description (FGD) of Czech (see [8, 13]). We believe that the LSC of full (four level) FGD can be described by tools very close to 24-LSC(4) (for the tools see [8]).

Finally, note that many practical problems in computational and corpus linguistic became decidable if we consider only languages parametrized by the size of the windows, or even easier by the finite number of reductions.

Aknowledgement. We thank to anonymous referees for their valuable comments.

References

[1] Bar-Hillel, Y.: A quasi-arithmetical notation for syntactic description. *Language* **29** (1953) 47–58

[2] Hopcroft, J. E., Ullman, J. D.: Introduction to Automata Theory, Languages, and Computation. Addison-Wesley, Reading, M.A. (1979)

[3] 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)

[4] Jančar, P., Mráz, F., Plátek, M., Vogel, J.: Different Types of Monotonicity for Restarting Automata. In: FST&TCS 1998. LNCS 1530, Springer, Berlin (1998) 343–354

[5] Kallmeyer, L.: Parsing Beyond Context-Free Grammars. Cognitive Technologies, Springer (2010)

[6] Kunze, J.: Abhängigkeitsgrammatik. Studia grammatica XII. Akademie-Verlag, Berlin (1969)

[7] 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, Proceedings. LNCS 3658, Springer, Berlin (2005) 140–147

[8] Lopatková, M., Plátek, M., Sgall, P.: Towards a formal model for functional generative description: Analysis by reduction and restarting automata. Prague Bull. Math. Linguistics **87** (2007) 7–26

[9] Niemann, G., Otto, F.: Restarting automata, Church-Rosser languages, and representations of r.e. languages. In: Developments In Language Theory: Foundations, Applications, and Perspectives, World Scientific (2000) 103–114

[10] Plátek, M., Otto, F., Mráz, F.: On h-lexicalized automata and h-syntactic analysis. In: ITAT 2017, Proc., CEUR Workshop Proceedings Vol. 1885 (2017) 40–47

[11] Plátek, M., Pardubská, D., Mráz, F.: Robustness versus Sensibility by Two-Way Restarting Automata. In: ITAT 2018, Proc., CEUR Workshop Proceedings Vol. 2203 (2018) 10–17

[12] Šmilauer, V.: Učebnice větného rozboru. Státní pedagogické nakladatelství (1958)

[13] Sgall, P.: Generativní popis jazyka a česká deklinace. Academia (1967)