

# On h-Lexicalized Automata and h-Syntactic Analysis

Martin Plátek<sup>1</sup>, F. Otto<sup>2</sup>, and F. Mráz<sup>1</sup> \*

<sup>1</sup> 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

<sup>2</sup> Fachbereich Elektrotechnik/Informatik, Universität Kassel  
D-34109 Kassel, Germany  
otto@theory.informatik.uni-kassel.de

*Abstract:* Following some previous studies on list automata and restarting automata, we introduce a generalized and refined model – the *h-lexicalized restarting list automaton* (LxRLAW). We argue that this model is useful for expressing transparent variants of lexicalized syntactic analysis, and analysis by reduction in computational linguistics. We present several subclasses of LxRLAW and provide some variants and some extensions of the Chomsky hierarchy, including the variant for the lexicalized syntactic analysis. We compare the input languages, which are the languages traditionally considered in automata theory, to the so-called *basic* and *h-proper languages*. The basic and h-proper languages allow stressing the transparency of h-lexicalized restarting automata for a superclass of the context-free languages by the so-called complete correctness preserving property. Such a type of transparency cannot be achieved for the whole class of context-free languages by traditional input languages. The transparency of h-lexicalized restarting automata is illustrated by two types of hierarchies which separate the classes of infinite and the classes of finite languages by the same tools.

## 1 Introduction

Chomsky introduced his well-known hierarchy of grammars to formulate the phrase-structure (immediate constituents) syntax of natural languages. However, the syntax of most European languages (including English) is often considered as a lexicalized syntax. In other words, the categories of Chomsky are bound to immediate constituents (phrases), while by lexicalized syntax they are bound to individual word-forms. In order to give a general theoretical basis for lexicalized syntax that is comparable to the Chomsky hierarchy, we follow some previous studies of list and restarting automata (see [1, 4, 5, 13]) and introduce a generalized and refined model that formalizes lexicalization in a natural way – the *h-lexicalized restarting list automaton with a look-ahead window* (LxRLAW). We argue that, through the use of restarting operations

and basic and h-proper languages, this new model is better suited for modeling (i) lexicalized syntactic analysis (h-syntactic analysis) and specially (ii) (lexicalized) analysis by reduction of natural languages (compare [6, 7]).

Analysis by reduction is a technique for deciding the correctness of a sentence. It is based on a stepwise simplification by reductions preserving the (in)correctness of the sentence until a short sentence is obtained for which it is easy to decide its correctness. Restarting automata were introduced as an automata model for analysis by reduction. While modeling analysis by reduction, restarting automata are forced to use very transparent types of computations. Nevertheless, they still can recognize a proper superset of the class of context-free languages (CFL). The first observation, which supports our argumentation, is that LxRLAW allow characterizations of the Chomsky hierarchy for classes of languages and for h-syntactic analysis as well.

An LxRLAW  $M$  is a device with a finite state control and a read/write window of a fixed size. This window can move in both directions along a tape (that is, a list of items) containing a word delimited by sentinels. The LxRLAW-automaton  $M$  uses an input alphabet and a working alphabet that contains the input alphabet. Lexicalization of  $M$  is given through a morphism which binds the individual symbols from the working alphabet to symbols from the input alphabet.  $M$  can decide (in general non-deterministically) to rewrite the contents of its window: it may delete some items from the list (moving its window in one or the other direction), insert some items into the list, and/or replace some items. In addition,  $M$  can perform a *restart operation* which causes  $M$  to place its window at the left end of the tape, so that the first symbol it contains is the left border marker, and to reenter its initial state.

In the technical part we adjust some known results to results on several subclasses of LxRLAW that are obtained by restricting its set of operations to certain subsets, and we also provide some variants and extensions of the Chomsky hierarchy of languages. E.g., an LxRLAI uses an input alphabet only.

We recall and newly introduce some constraints that are suitable for restarting automata, and we outline ways for new combinations of constraints, and more transparent computations.

\*The first and the third author were partially supported by the Czech Science Foundation under the project 15-04960S.

Section 2 contains the basic definitions. The characterizations of the Chomsky hierarchy by certain types of LxRLAW is provided in Section 3.

In Section 4 we introduce RLWW-automata as a restricted type of LxRLAW and give several new constraints for them. By the basic and h-proper languages we show the transparency of h-lexicalized restarting automata through the so-called complete correctness preserving property. Using the new constraints and the basic and h-proper languages, we are able to separate classes of finite languages in a similar way as the classes of infinite languages and establish in this way new hierarchies, which can create a suitable tool for the classification of syntactic phenomena for computational linguistics. The paper concludes with Section 5.

## 2 Definitions

In what follows, we use  $\subset$  to denote the proper subset relation. Further, we will sometimes use regular expressions instead of the corresponding regular languages. Finally, throughout the paper  $\lambda$  will denote the empty word and  $\mathbb{N}_+$  will denote the set of all positive integers.

An *h-lexicalized two-way restarting list automaton*, LxRLAW for short, is a one-tape machine  $M = (Q, \Sigma, \Gamma, \mathfrak{c}, \$, q_0, k, \delta, h)$ , where  $Q$  is the finite set of states,  $\Sigma$  is the finite input alphabet,  $\Gamma$  is the finite working alphabet containing  $\Sigma$ , the symbols  $\mathfrak{c}, \$ \notin \Gamma$  are the markers for the left and right border of the work space, respectively,  $h : \Gamma \cup \{\mathfrak{c}, \$\} \rightarrow \Sigma \cup \{\mathfrak{c}, \$\}$  is a mapping creating a (letter) morphism from  $\mathfrak{c}\Gamma^*\$$  to  $\mathfrak{c}\Sigma^*\$$  such that, for each  $a \in \Sigma \cup \{\mathfrak{c}, \$\}$ ,  $h(a) = a$ ;  $q_0 \in Q$  is the initial state,  $k \geq 1$  is the size of the *read/write window*, and

$$\delta : Q \times \mathcal{P}\mathcal{C}^{\leq k} \rightarrow \mathcal{P}((Q \times (\{\text{MVR}, \text{MVL}\} \cup \{\text{W}(v), \text{SR}(v), \text{SL}(v), \text{l}(v)\})) \cup \{\text{Restart}, \text{Accept}, \text{Reject}\})$$

is the *transition relation*. Here  $\mathcal{P}(S)$  denotes the powerset of a set  $S$ ,

$$\mathcal{P}\mathcal{C}^{\leq k} := (\mathfrak{c} \cdot \Gamma^{k-1}) \cup \Gamma^k \cup (\Gamma^{\leq k-1} \cdot \$) \cup (\mathfrak{c} \cdot \Gamma^{\leq k-2} \cdot \$),$$

is the set of *possible contents* of the read/write window of  $M$ , and  $v \in \mathcal{P}\mathcal{C}^{\leq n}$ , where  $n \in \mathbb{N}$ .

According to the transition relation, if  $M$  is in state  $q$  and sees the string  $u$  in its read/write window, it can perform nine different steps, where  $q' \in Q$ :

1. A *move-right step*  $(q, u) \rightarrow (q', \text{MVR})$  assumes that  $(q', \text{MVR}) \in \delta(q, u)$  and  $u \neq \$$ . It causes  $M$  to shift the read/write window one position to the right and to enter state  $q'$ .
2. A *move-left step*  $(q, u) \rightarrow (q', \text{MVL})$  assumes that  $(q', \text{MVL}) \in \delta(q, u)$  and  $u \notin \mathfrak{c} \cdot \Gamma^* \cdot \{\lambda, \$\}$ . It causes  $M$  to shift the read/write window one position to the left and to enter state  $q'$ .

3. A *rewrite step*  $(q, u) \rightarrow (q', \text{W}(v))$  assumes that  $(q', \text{W}(v)) \in \delta(q, u)$ ,  $|v| = |u|$ , and the sentinels are at the same positions in  $u$  and  $v$  (if at all). It causes  $M$  to replace the contents  $u$  of the read/write window by the string  $v$ , and to enter state  $q'$ . The head does not change its position.
4. An *S-right step*  $(q, u) \rightarrow (q', \text{SR}(v))$  assumes that  $(q', \text{SR}(v)) \in \delta(q, u)$ ,  $v$  is shorter than  $u$ , containing all sentinels in  $u$ . It causes  $M$  to replace  $u$  by  $v$  and to enter state  $q'$ ; the new position of the head is on the first item of  $v$  (the contents of the window is thus ‘completed’ from the right; the positional distance to the right sentinel decreases).
5. An *S-left step*  $(q, u) \rightarrow (q', \text{SL}(v))$  assumes that  $(q', \text{SL}(v)) \in \delta(q, u)$ ,  $v$  is shorter than  $u$ , containing all sentinels in  $u$ . It causes  $M$  to replace  $u$  by  $v$ , to enter state  $q'$ , and to shift the head position by  $|u| - |v|$  items to the left – but to the left sentinel  $\mathfrak{c}$  at most (the contents of the window is ‘completed’ from the left; the distance to the left sentinel decreases if the head position was not already at  $\mathfrak{c}$ ).
6. An *insert step*  $(q, u) \rightarrow (q', \text{l}(v))$  assumes that  $(q', \text{l}(v)) \in \delta(q, u)$ ,  $u$  is a proper subsequence of  $v$  (keeping the obvious sentinel constraints). It causes  $M$  to replace  $u$  by  $v$  (by inserting the relevant items), and to enter state  $q'$ . The head position is shifted by  $|v| - |u|$  to the right (the distance to the left sentinel increases).
7. A *restart step*  $(q, u) \rightarrow \text{Restart}$  assumes that  $\text{Restart} \in \delta(q, u)$ . It causes  $M$  to place its read/write window onto the left end of the tape, so that the first symbol it sees is the left border marker  $\mathfrak{c}$ , and to reenter the initial state  $q_0$ .
8. An *accept step*  $(q, u) \rightarrow \text{Accept}$  assumes that  $\text{Accept} \in \delta(q, u)$ . It causes  $M$  to halt and accept.
9. A *reject step*  $(q, u) \rightarrow \text{Reject}$  assumes that  $\text{Reject} \in \delta(q, u)$ . It causes  $M$  to halt and reject.

A *configuration* of  $M$  is a string  $\alpha q \beta$  where  $q \in Q$ , and either  $\alpha = \lambda$  and  $\beta \in \{\mathfrak{c}\} \cdot \Gamma^* \cdot \{\$\}$  or  $\alpha \in \{\mathfrak{c}\} \cdot \Gamma^*$  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 head scans the first  $k$  symbols of  $\beta$  or all of  $\beta$  when  $|\beta| < k$ . A *restarting configuration* is of the form  $q_0 \mathfrak{c} w \$$ , where  $w \in \Gamma^*$ ; if  $w \in \Sigma^*$ , then  $q_0 \mathfrak{c} w \$$  is an *initial configuration*. We see that any initial configuration is also a restarting configuration. Any restart transfers  $M$  into a restarting configuration.

In general, the automaton  $M$  is *non-deterministic*, that is, there can be two or more steps (instructions) with the same left-hand side  $(q, u)$ , and thus, there can be more than one computation for an input word. If this is not the case, the automaton is *deterministic*.

A *computation* of  $M$  is a sequence  $C = C_0, C_1, \dots, C_j$  of configurations, where  $C_0$  is an initial configuration and  $C_{i+1}$  is obtained by a step of  $M$  from  $C_i$ , for all  $0 \leq i < j$ .

An *input word*  $w \in \Sigma^*$  is *accepted* by  $M$ , if there is a computation which starts with the initial configuration  $q_0 \epsilon w \$$  and ends by executing an Accept instruction. By  $L(M)$  we denote the language consisting of all input words accepted by  $M$ ; we say that  $M$  *accepts the input language*  $L(M)$ .

A *basic (or characteristic, or working) word*  $w \in \Gamma^*$  is *accepted* by  $M$ , if there is a computation which starts with the restarting configuration  $q_0 \epsilon w \$$  and ends by executing an Accept instruction. By  $L_C(M)$  we denote the language consisting of all basic words accepted by  $M$ ; we say that  $M$  *accepts the basic (characteristic) language*  $L_C(M)$ .

Further, we take  $L_{\text{hP}}(M) = \{h(w) \in \Sigma^* \mid w \in L_C(M)\}$ , and we say that  $M$  *recognizes (accepts) the h-proper language*  $L_{\text{hP}}(M)$ .

Finally, we take  $L_A(M) = \{(h(w), w) \mid w \in L_C(M)\}$  and we say that  $M$  *determines the h-syntactic analysis*  $L_A(M)$ .

We say that, for  $x \in \Sigma^*$ ,  $L_A(M, x) = \{(x, y) \mid y \in L_C(M), h(y) = x\}$  is the *h-syntactic analysis* for  $x$  by  $M$ . We see that  $L_A(M, x)$  is non-empty only for  $x \in L_{\text{hP}}(M)$ .

In the following we only consider finite computations of LxRLAW-automata, which finish either by an accept or by a reject operation.

An LxRLAI  $M$  is an LxRLAW for which the input alphabet and the working alphabet are equal.

**Fact 1. (Equality of Languages for LxRLAI-automata).** For each LxRLAI-automaton  $M$ ,  $L(M) = L_C(M) = L_{\text{hP}}(M)$ .

– **Cycles, tails:** Any finite computation of an LxRLAW-automaton  $M$  consists of certain phases. Each phase starts in a restarting configuration. In a phase called a *cycle*, the window moves along the tape performing non-restarting operations until a Restart operation is performed and thus a new restarting configuration is reached. If after a restart configuration no further restart operation is performed, any finite computation necessarily finishes in a halting configuration – such a phase is called a *tail*.

– **Cycle-rewritings:** We use the notation  $q_0 \epsilon u \$ \vdash_M^c q_0 \epsilon v \$$  to denote a cycle of  $M$  that begins with the restarting configuration  $q_0 \epsilon u \$$  and ends with the restarting configuration  $q_0 \epsilon v \$$ . Through this relation we define the relation of *cycle-rewriting* by  $M$ . We write  $u \Rightarrow_M^c v$  iff  $q_0 \epsilon u \$ \vdash_M^c q_0 \epsilon v \$$  holds. The relation  $u \Rightarrow_M^{c*} v$  is the reflexive and transitive closure of  $u \Rightarrow_M^c v$ .

We point out that the cycle-rewriting is a very important feature of LxRLAW.

– **Reductions:** If  $u \Rightarrow_M^c v$  is a cycle-rewriting by  $M$  such that  $|u| > |v|$ , then  $u \Rightarrow_M^c v$  is called a *reduction* by  $M$ .

## 2.1 Further Refinements on LxRLAW

Here we introduce some constrained types of rewriting steps which assume  $q, q' \in Q$  and  $u \in \mathcal{P}^{\leq k}$ .

A *delete-right step*  $(q, u) \rightarrow (q', \text{DR}(v))$  is an S-right step  $(q, u) \rightarrow (q', \text{SR}(v))$  such that  $v$  is a proper sub-sequence of  $u$ , containing all the sentinels from  $u$  (if any).

A *delete-left step*  $(q, u) \rightarrow (q', \text{DL}(v))$  is an S-left step  $(q, u) \rightarrow (q', \text{SL}(v))$  such that  $v$  is a proper sub-sequence of  $u$ , containing all the sentinels from  $u$  (if any).

A *contextual-left step*  $(q, u) \rightarrow (q', \text{CL}(v))$  is an S-left step  $(q, u) \rightarrow (q', \text{SL}(v))$ , where  $u = v_1 u_1 v_2 u_2 v_3$  and  $v = v_1 v_2 v_3$  for some  $v_1, u_1, v_2, u_2, v_3$ , such that  $v$  contains all the sentinels from  $u$  (if any).

A *contextual-right step*  $(q, u) \rightarrow (q', \text{CR}(v))$  is an S-right-step  $(q, u) \rightarrow (q', \text{SR}(v))$ , where  $u = v_1 v_2 v_3$  and  $v = v_1 v_3$  for some  $v_1, v_2, v_3$ , such that  $v$  contains all the sentinels from  $u$  (if any).

Note that the contextual-right step is not symmetrical to the contextual-left step. We will use this fact in Section 4.

The set  $OG = \{\text{MVL}, \text{MVR}, \text{W}, \text{SL}, \text{SR}, \text{DL}, \text{DR}, \text{CL}, \text{CR}, \text{l}, \text{Restart}\}$  represents the set of types of steps, which can be used for characterizations of subclasses of LxRLAW. This set does not contain the symbols Accept and Reject, corresponding to halting steps, as they are used for all LxRLAW-automata. Let  $T \subseteq OG$ . We denote by T-automata the subset of LxRLAW-automata which only use transition steps from the set  $T \cup \{\text{Accept}, \text{Reject}\}$ . For example,  $\{\text{MVR}, \text{W}\}$ -automata only use move-right steps, W-steps, Accept steps, and Reject steps.

*Notations.* For brevity, the prefix *det-* will be used to denote the property of being deterministic. For any class  $\mathcal{A}$  of automata,  $\mathcal{L}(\mathcal{A})$  will denote the class of input languages that are recognized by automata from  $\mathcal{A}$ ,  $\mathcal{L}_C(\mathcal{A})$  will denote the class of basic languages that are recognized by automata from  $\mathcal{A}$ , and  $\mathcal{L}_{\text{hP}}(\mathcal{A})$  will denote the class of h-proper languages that are recognized by automata from  $\mathcal{A}$ . Moreover,  $\mathcal{L}_A(\mathcal{A})$  will denote the class of h-syntactic analyses that are determined by automata from  $\mathcal{A}$ . Let us note that we use the more simple notation  $\mathcal{L}_P(\mathcal{A})$  in [15] and other papers in a different sense than the denotation  $\mathcal{L}_{\text{hP}}(\mathcal{A})$  here.

For a natural number  $k \geq 1$ ,  $\mathcal{L}(k\text{-}\mathcal{A})$  ( $\mathcal{L}_C(k\text{-}\mathcal{A})$  or  $\mathcal{L}_{\text{hP}}(k\text{-}\mathcal{A})$ ) will denote the class of input (basic or h-proper) languages that are recognized by those automata from  $\mathcal{A}$  that use a read/write window of size  $k$ .

– **Monotonicity of Rewritings.** We introduce various notions of *monotonicity* as important types of constraints for computations of LxRLAW-automata.

Let  $M$  be an LxRLAW-automaton, and let  $C = C_k, C_{k+1}, \dots, C_j$  be a sequence of configurations of  $M$ , where  $C_{i+1}$  is obtained by a single transition step of  $M$  from  $C_i$ ,  $k \leq i < j$ . We say that  $C$  is a *sub-computation* of  $M$ .

Let  $RW \subseteq \{\text{W}, \text{SR}, \text{SL}, \text{DR}, \text{DL}, \text{CR}, \text{CL}, \text{l}\}$ . Then we denote by  $W(C, RW)$  the maximal (scattered) sub-sequence

of  $C$ , which contains those configurations from  $C$  that correspond to  $RW$ -steps (that is, those configurations in which a transition step of one of the types from the set  $RW$  is applied). We say that  $W(C, RW)$  is the *working sequence* of  $C$  determined by  $RW$ .

Let  $C$  be a sub-computation of an LxRLAW-automaton  $M$ , and let  $C_w = \mathfrak{c}\alpha q\beta\mathfrak{s}$  be a configuration from  $C$ . Then  $|\beta\mathfrak{s}|$  is the *right distance* of  $C_w$ , which is denoted by  $D_r(C_w)$ , and  $|\mathfrak{c}\alpha|$  is the *left distance* of  $C_w$ , which is denoted by  $D_l(C_w)$ .

We say that a working sequence  $W(C, RW) = (C_1, C_2, \dots, C_n)$  is *RW-monotone* (or *RW-right-monotone*) if  $D_r(C_1) \geq D_r(C_2) \geq \dots \geq D_r(C_n)$ .

A sub-computation  $C$  of  $M$  is *RW-monotone* if  $W(C, RW)$  is *RW-monotone*. If we write (right-)monotone, we actually mean  $\{W, SR, SL, DR, DL, CR, CL, I\}$ -right-monotone, that is, monotonicity with respect to any type of (allowed) rewriting and inserting for the corresponding type of automaton. By completely-(right-)monotone, we actually mean monotone with respect to each configuration of the computation.

For each of the prefixes  $X \in \{RW, \lambda, \text{completely}\}$  we say that  $M$  is *X-monotone* if each of its (sub)computations is *X-monotone*.

**Fact 2.** *Let  $M$  be an  $\{MVR, SR, SL, W, I\}$ -automaton. Then  $M$  is completely-monotone.*

**Remark on PDA.** It is not hard to see that a 1- $\{MVR, SR, SL, W, I\}$ -automaton is a type of normalized pushdown automaton. The top of the pushdown is represented by the position of the head, and the content of the pushdown is represented by the part of the tape between the left sentinel and the position of the head. In fact, in a very similar way the pushdown automaton was introduced by Chomsky. A  $k$ - $\{MVR, SR, SL, W, I\}$ -automaton can be interpreted as a pushdown automaton with a  $k$ -lookahead, and with a limited look under the top of the pushdown at the same time. (det-)PDAs can be simulated even by (det-)1- $\{MVR, SL, W\}$ -automata. In the following, we will consider the automata which fulfill the condition of completely-right-monotonicity for pushdown automata (PDA).

### 3 Characterizations of the Chomsky Hierarchy

We transform and enhance some results from [1, 5] concerning input languages to basic and h-proper languages.

det- $\{MVR\}$ - and  $\{MVR\}$ -automata work like deterministic and nondeterministic finite-state automata, respectively. The only difference is that such automata can accept or reject without visiting all symbols of an input word. Nevertheless, these automata can be simulated by deterministic and nondeterministic finite-state automata, respectively, which instead of an Accept-step enter a special accepting state in which they scan the rest of the input

word. Since the regular languages are closed under homomorphisms, we have the following proposition.

**Proposition 3.** *For  $\mathcal{X} \in \{\mathcal{L}, \mathcal{L}_C, \mathcal{L}_{hP}\}$ , the classes  $\mathcal{X}(1\text{-det-}\{MVR\})$  and  $\mathcal{X}(1\text{-}\{MVR\})$  coincide with the class REG of regular languages.*

Observe that for LxRLAW-automata with window size 1, the operation SR coincides with the operations DR and CR, and that the operation SL coincides with the operations DL and CL, and that in this situation all these operations just delete the symbol currently inside the window.

**Proposition 4.** *For  $\mathcal{X} \in \{\mathcal{L}, \mathcal{L}_C, \mathcal{L}_{hP}\}$ :*

$$\mathcal{X}(\{MVR, CL, W\}) \subseteq \text{CFL},$$

where CFL is the class of context-free languages.

*Proof.* Any  $\{MVR, CL, W\}$ -automaton  $M = (Q, \Sigma, \Gamma, \mathfrak{c}, \mathfrak{s}, q_0, k, \delta, h)$  can be simulated by a pushdown automaton (PDA)  $P$  which stores the current content of the window of  $M$  in its control unit (as a state). On an input word  $w$ ,  $P$  first tries to read the first  $k$  symbols of  $w$  and to store them within its control unit. If  $w$  is of length less than  $k$ , then  $P$  accepts or rejects  $w$  upon encountering the right sentinel  $\mathfrak{s}$ . Otherwise,  $P$  continues while preserving the following invariant: the contents of the pushdown store of  $P$  equals the part the tape of  $M$  to the left of its current position, the contents of the window of  $M$  and its state are both stored in the control unit of  $P$ , and the rest of the tape of  $M$  to the right of its window is the unread part of the input of  $P$ .

Hence,  $P$  accepts exactly  $L(M)$  by entering an accepting state and reading the rest of its input whenever  $M$  performs an Accept-step. If we include all working symbols of  $M$  into the input alphabet of  $P$ , we obtain a PDA  $P'$  such that  $L(P') = L_C(M)$ , thus the basic language of  $M$  is context-free. As CFL is closed under homomorphisms, also the h-proper language of  $M$  is context-free, too.  $\square$

**Proposition 5.** *For  $\mathcal{X} \in \{\mathcal{L}, \mathcal{L}_C, \mathcal{L}_{hP}\}$ , the class  $\mathcal{X}(1\text{-}\{MVR, CL, W\})$  coincides with the class CFL of context-free languages.*

*Proof.* According to Proposition 4, each language accepted by a 1- $\{MVR, CL, W\}$ -automaton as an input language or as a basic language is context-free. It remains to prove the opposite inclusion. Let  $L$  be a context-free language. Clearly, the empty language and the language  $\{\lambda\}$  can be accepted by a 1- $\{MVR, CL, W\}$ -automaton.

W.l.o.g. we can suppose that  $L \setminus \{\lambda\}$  is generated by a context-free grammar  $G = (\Pi, \Sigma, S, P_G)$  in Chomsky normal form. We can construct a PDA  $P$  accepting the language  $L \setminus \{\lambda\}$  by empty store. For each nonempty word  $w \in L$ , the PDA  $P$  can guess and simulate a rightmost derivation of  $w$  according to  $G$  in reverse order. That is,  $P$  will perform a bottom-up analysis of  $w$  which uses a shift-operation that moves the next input symbol onto the pushdown and reductions according to the rewrite rules from  $P_G$ . The reduction according to a rule of the form  $X \rightarrow x$ , for  $X \in \Pi$ ,  $x \in \Sigma$ , consists of popping  $x$  from the

top of the pushdown and pushing  $X$  onto the pushdown. The reduction according to a rule of the form  $X \rightarrow YZ$ , where  $X, Y, Z$  are nonterminals of  $G$ , consists of popping  $YZ$  (in reversed order) from the pushdown and pushing  $X$  onto the pushdown.

The empty word  $\lambda$  can be immediately accepted or rejected by a 1- $\{\text{MVR}, \text{CL}, \text{W}\}$ -automaton  $M$  when  $\lambda \in L$  or  $\lambda \notin L$ , respectively. For each nonempty word  $w \in \Sigma^*$ , each computation of  $P$  on  $w$  can be simulated by  $M$  in such a way that the top of the pushdown will be in the window of  $M$ . The rest of the contents of the pushdown of  $P$  will be stored on the part of the tape of  $M$  to the left of the position of its window.

A shift-operation of  $P$  can be simulated by a MVR-step. The reduction according to a rule of the form  $X \rightarrow x$ , for  $X \in \Pi$ ,  $x \in \Sigma$ , will be simulated by the W-step which rewrites  $x$  in the window of  $M$  by  $X$ . The reduction according to a rule of the form  $X \rightarrow YZ$ , where  $X, Y, Z \in \Pi$ , will be simulated by the CL-step which deletes the tape cell containing  $Z$  and a W-step which rewrites the symbol  $Y$  in the window of  $M$  by  $X$ . As  $P$  accepts when the pushdown contains the single symbol  $S$ ,  $M$  will perform an Accept-step, when the only symbol on its tape is the initial nonterminal  $S$ . Clearly,  $L(M) = L(P)$ . Additionally,  $M$  can check that after each MVR-step the symbol which appears within its window is either the sentinel  $\$$  or a terminal from  $\Sigma$ . In this way it is ensured that  $M$  does not accept any word containing a working symbol, hence  $L_C(M) = L_{\text{hP}}(M) = L$ .  $\square$

It is easy to see that a linear-bounded automaton working on a tape containing an input word delimited by sentinels directly corresponds to a 1- $\{\text{MVR}, \text{MVL}, \text{W}\}$ -automaton. We can also see that a  $\{\text{MVR}, \text{MVL}, \text{W}\}$ -automaton can be simulated by a 1- $\{\text{MVR}, \text{MVL}, \text{W}\}$ -automaton. Recall that the class CSL is closed under non-erasing homomorphisms. Therefore, we have the following statement.

**Proposition 6.** *For  $\mathcal{X} \in \{\mathcal{L}, \mathcal{L}_C, \mathcal{L}_{\text{hP}}\}$ , the class  $\mathcal{X}(\{\text{MVR}, \text{MVL}, \text{W}\})$  coincides with the class CSL of context-sensitive languages.*

By adding the ability to insert cells within the working tape to the operations MVR, MVL and W, we can easily simulate an arbitrary Turing machine. Hence we have the following proposition.

**Proposition 7.** *For  $\mathcal{X} \in \{\mathcal{L}, \mathcal{L}_C, \mathcal{L}_{\text{hP}}\}$ , the class  $\mathcal{X}((\text{det-})\{\text{MVR}, \text{MVL}, \text{W}, \text{I}\})$  coincides with the class RE of recursively enumerable languages.*

From the previous results we obtain the following variant of the Chomsky hierarchy for the classes of the h-syntactic analysis which follows from the corresponding hierarchies for the classes of h-proper and basic languages.

**Corollary 1.** *We have the following hierarchy:*

$$\mathcal{L}_A(\{\text{MVR}\}) \subset \mathcal{L}_A(\{\text{MVR}, \text{CL}, \text{W}\}),$$

$$\begin{aligned} \mathcal{L}_A(\{\text{MVR}, \text{CL}, \text{W}\}) &\subset \mathcal{L}_A(\{\text{MVR}, \text{MVL}, \text{W}\}), \\ \mathcal{L}_A(\{\text{MVR}, \text{MVL}, \text{W}\}) &\subset \mathcal{L}_A(\{\text{MVR}, \text{MVL}, \text{W}, \text{I}\}). \end{aligned}$$

In a similar way we can obtain the deterministic variants of the Chomsky hierarchy.

## 4 RLWW-Automata with New Constraints

Here we formulate the h-lexicalized two-way restarting automaton in weak accepting (cyclic) form, and some of its subclasses. By considering basic and h-proper languages, this new type of automaton is close to the method of analysis by reduction (see [6]), its computations are transparent, and it reflects well the structure of its basic and h-proper languages.

An *h-lexicalized two-way restarting automaton in weak accepting form* (originally called *cyclic form*)  $M$ , an *hRLWW-automaton* for short, is a  $\{\text{MVR}, \text{MVL}, \text{SL}, \text{SR}, \text{Restart}\}$ -automaton, which uses an SL-step or SR-step exactly once in each cycle (only one of them), and directly accepts only words that fit into its window.

An hRLWW-automaton is called an *hRLW-automaton* if its working alphabet coincides with its input alphabet. Note that in this situation, each restarting configuration is necessarily an initial configuration.

An hRLW-automaton is called an *hRL-automaton* if each of its rewrite steps is a DL- or DR-step.

An hRL-automaton is called an *hRLC-automaton* if each of its rewrite steps is a CL- or CR-step.

An hRLWW-automaton is called an *hRLWWC-automaton* if each of its rewrite steps is a CL-step or CR-step.

An hRLWW-automaton is called an *hRRWW-automaton* if it does not use any MVL-steps. Analogously, we obtain *hRRW-automata*, *hRR-automata*, and *hRRC-automata*.

We see that for hRLWW-automata, all cycle-rewritings are reductions. We also have the following simple facts, which illustrate the transparency of computations of hRLWW-automata due their basic and h-proper languages.

**Fact 8. (Complete Correctness Preserving Property).**

*Let  $M$  be a deterministic hRLWW-automaton, let  $C = C_0, C_1, \dots, C_n$  be a computation of  $M$ , and let  $\text{cu}_i\$$  be the tape contents of the configuration  $C_i$ ,  $0 \leq i \leq n$ . If  $u_i \in L_C(M)$  for some  $i$ , then  $u_j \in L_C(M)$  for all  $j = 0, 1, \dots, n$ .*

**Fact 9. (Complete Error Preserving Property).**

*Let  $M$  be a deterministic hRLWW-automaton, let  $C = C_0, C_1, \dots, C_n$  be a computation of  $M$ , and let  $\text{cu}_i\$$  be the tape contents of the configuration  $C_i$ ,  $0 \leq i \leq n$ . If  $u_i \notin L_C(M)$  for some  $i$ , then  $u_j \notin L_C(M)$  for all  $j = 0, 1, \dots, n$ .*

**Fact 10. (Prefix Correctness Preserving Property).**

*Let  $M$  be an hRLWW-automaton, let  $C = C_0, C_1, \dots, C_n$  be a computation of  $M$ , and let  $\text{cu}_i\$$  be the tape contents of the configuration  $C_i$ ,  $0 \leq i \leq n$ . If  $u_i \in L_C(M)$  for some  $i$ , then  $u_j \in L_C(M)$  for all  $j \leq i$ .*

**Fact 11. (Suffix Error Preserving Property).**

Let  $M$  be an hRLWW-automaton, let  $C = C_0, C_1, \dots, C_n$  be a computation of  $M$ , and let  $cu_i\$$  be the tape contents of the configuration  $C_i$ ,  $0 \leq i \leq n$ . If  $u_i \notin L_C(M)$  for some  $i$ , then  $u_j \notin L_C(M)$  for all  $j \geq i$ .

**Corollary 2. (Equality of Languages for hRLW-automata).**

For each hRLW-automaton  $M$ ,  $L(M) = L_C(M) = L_{hP}(M)$ .

From the last corollary above, the Complete Error and Complete Correctness Preserving Properties for deterministic hRLW-automata, and the Suffix Error Preserving Property and the Prefix Correctness Preserving Property for hRLW-automata follow for their input, basic, and h-proper languages.

**4.1 On Regular Languages and Correctness Preserving Computations**

**Proposition 12.** *The class REG is characterized by basic and h-proper languages of deterministic hRLWW-automata which use only the operations MVR, CR, Restart, and which use the operation CR only when the window is at the position just to the right of the left sentinel. We denote such automata by det-Rg2.*

*Proof.* We outline only the main ideas of the proof. First we show that any basic language and h-proper language of a det-Rg2 is regular. Let  $M_k$  be a  $k$ -det-Rg2-automaton. It is not hard to see that  $M_k$  can be simulated by a finite automaton  $A_{k+1}$  with a stack of size  $k+1$  stored within its finite control, and therefore  $L_C(M_k) = L(A_{k+1})$ , and  $L_{hP}(M_k) = L_{hP}(A_{k+1})$ . Since the regular languages are closed under homomorphisms,  $L_{hP}(A_{k+1})$  is a regular language, too.

On the other hand, we can see from the pumping lemma for regular languages that any regular language  $L \subseteq \Sigma^*$  can be accepted by a det-Rg2-automaton with working (and input) alphabet  $\Sigma$ .  $\square$

**Remark.** Since det-Rg2-automata are deterministic hRLWW-automata, we see that they are completely correctness preserving for their basic languages.

**4.2 Monotonicity and h-Proper Languages**

As an hRRWWC-automaton is an hRRWW-automaton which uses only CL-operations instead of SL-operations and CR-operations instead of SR-operations, we can prove the following theorem in a similar way as it was done for a stronger version of hRRWW-automata in the technical report [15].

**Theorem 13.**  $CFL = \mathcal{L}_{hP}(\text{det-mon-hRLWWC})$ .

*Proof.* The proof is based mainly on ideas from [15]. Here it is transformed for hRLWW-automata with the constraint of the weak accepting form.

If  $M = (Q, \Sigma, \Gamma, \mathfrak{c}, \$, q_0, k, \delta, h)$  is a right-monotone hRLWW-automaton, then its characteristic language  $L_C(M)$  is context-free [14]. As  $L_{hP}(M) = h(L_C(M))$ , and as CFL is closed under morphisms, it follows that  $L_{hP}(M)$  is also context-free.

Conversely, assume that  $L \subseteq \Sigma^*$  is a context-free language. Without loss of generality we may assume that  $L$  does not contain the empty word. Thus, there exists a context-free grammar  $G = (N, \Sigma, S, P)$  for  $L$  which is in Greibach normal form, that is, each rule of  $P$  has the form  $A \rightarrow a\alpha$  for some string  $\alpha \in N^*$  and some letter  $a \in \Sigma$ . For the following construction we assume that the rules of  $G$  are numbered from 1 to  $m$ .

From  $G$  we construct a new grammar  $G' := (N, \Delta, S, P')$ , where  $\Delta := \{(\nabla_i, a) \mid 1 \leq i \leq m \text{ and the } i\text{-th rule of } G \text{ has the form } A \rightarrow a\alpha\}$  is a set of new terminal symbols that are in one-to-one correspondence to the rules of  $G$ , and

$$P' := \{A \rightarrow (\nabla_i, a)\alpha \mid A \rightarrow a\alpha \text{ is the } i\text{-th rule of } G, 1 \leq i \leq m\}.$$

Obviously, a word  $\omega \in \Delta^*$  belongs to  $L(G')$  if and only if  $\omega$  has the form  $\omega = (\nabla_{i_1}, a_1)(\nabla_{i_2}, a_2) \cdots (\nabla_{i_n}, a_n)$  for some integer  $n > 0$ , where  $a_1, a_2, \dots, a_n \in \Sigma$ ,  $i_1, i_2, \dots, i_n \in \{1, 2, \dots, m\}$ , and the sequence of these indices describes a (left-most) derivation of  $w := a_1 a_2 \cdots a_n$  from  $S$  in  $G$ . Let us take  $h((\nabla_i, a)) = a$  for all  $(\nabla_i, a) \in \Delta$ . Then it follows that  $h(L(G')) = L(G) = L$ . From  $\omega$  this derivation can be reconstructed deterministically. In fact, the language  $L(G')$  is deterministic context-free. Hence, there exists a deterministic right-monotone hRRC-automaton  $M$  for this language (see [4]). By interpreting the symbols of  $\Delta$  as auxiliary symbols, we obtain a deterministic right-monotone hRRWWC-automaton  $M'$  such that  $h(L_C(M')) = L_{hP}(M') = h(L(M)) = h(L(G')) = L$ . Observe that the input language  $L(M')$  of  $M'$  is actually empty.  $\square$

**Remark.** By means of h-lexicalized restarting automata, the above construction corresponds to the linguistic effort to obtain a set of categories (auxiliary symbols) that ensures the correctness preserving property for the respective analysis by reduction. Note that in its reductions, the automaton  $M'$  above only uses delete operations (in a special way). This is highly reminiscent of the basic (elementary) analysis by reduction learned in (Czech) basic schools.

**4.3 Hierarchies**

In this subsection we will show similar results for finite and infinite classes of basic and h-proper languages and for classes of h-syntactic analysis given by certain subclasses of hRLWW-automata. At first we introduce some useful notions.

For a (sub)class  $X$  of hRLWW-automata, by  $\text{fin}(i)\text{-}X$  we denote the subclass of  $X$ -automata which can perform at

most  $i$  reductions, and by  $\text{fin-}X$  we denote the union of  $\text{fin}(i)\text{-}X$  over all natural numbers  $i$ .

By  $L_C(M, i)$  we denote the subset of  $L_C(M)$  containing all words accepted by at most  $i$  reductions. We take  $L_{\text{hP}}(M, i) = h(L_C(M, i))$ .

**Proposition 14.** *Let  $i \geq 0$ , and let  $M$  be an hRLWW-automaton. Then there is a  $\text{fin}(i)$ -hRLWW-automaton  $M_1$  such that  $L_C(M, i) = L_C(M_1)$ , and if  $u \Rightarrow_{M_1} v$ , then  $u \Rightarrow_M v$ . If  $M$  is deterministic, then  $M_1$  is deterministic as well.*

**Proof.** The basic idea of the construction of  $M_1$  is to add to the finite control of  $M$  the counter of possible cycles on the starting word by  $M$ . While simulating the first cycle of  $M$ ,  $M_1$  already simulates the next up to  $i - 1$  cycles of  $M$  rejecting all words which are not acceptable with at most  $i$  cycles by  $M$ .  $\square$

We can see that a similar proposition can also be shown for hRRWW-automata.

For a positive integer  $k$ , let the prefix  $\text{de}(k)$ - denote hRLWW-automata which delete at most  $k$  symbols in each reduction.

The next proposition lays the foundation to the desired hierarchies. In order to show the next proposition we consider the following sequence of languages for  $k \geq 1$ :  $L_{\text{rg}} = \{(a^k)^i \mid i \geq 1\}$ .

**Proposition 15.**  $\mathcal{L}_{\text{hP}}(k\text{-de}(k)\text{-det-fin}(1)\text{-Rg2}) \setminus \mathcal{L}_{\text{hP}}((k-1)\text{-hRLWW}) \neq \emptyset$ , for all  $k \geq 2$ .

**Proof.** We outline the basic ideas only. We can construct a  $k\text{-det-Rg2}$ -automaton  $MR_k$  such that  $L_C(MR_k) = L_{\text{hP}}(MR_k) = L_{\text{rg}}$ . For  $MR_k$  a window of the size  $k$  suffices, because  $MR_k$  can first move its window to the right of the left sentinel. If it sees  $a^k$ , then it performs a CR-step which deletes  $a^k$ . Next, if the window contains only the right sentinel, the automaton accepts, otherwise it restarts.

From Proposition 14 we can see that there is a  $k\text{-det-Rg2}$ -automaton  $MR'_k$  such that  $L_C(MR'_k) = L_{\text{hP}}(MR'_k) = L_{\text{hP}}(MR_k, 1)$ .

On the other hand, there is no  $(k-1)$ -hRLWW-automaton accepting  $L_{\text{hP}}(MR'_k)$  as its h-proper language. For a contradiction, let us suppose that  $L_{\text{hP}}(M) = L_{\text{hP}}(MR'_k)$  for a  $(k-1)$ -hRLWW-automaton  $M$ . Then  $M$  accepts a word  $w \in L_C(M)$  such that  $h(w) = a^k \in L_{\text{hP}}(MR_k, 1)$ . In an accepting computation on  $w$ , automaton  $M$  must perform at least one reduction, but it can delete at most  $(k-1)$  symbols by which it obtains a word  $w'$  of length between 1 and  $k-1$ , hence  $h(w') \notin L_{\text{hP}}(MR'_k)$  – a contradiction to  $L_{\text{hP}}(M) = L_{\text{hP}}(MR'_k)$ .  $\square$

**Corollary 3.** *For all types  $X \in \{\text{det-Rg2}, \text{hRR}, \text{hRRC}, \text{hRRW}, \text{hRRWWC}, \text{hRRWW}, \text{hRL}, \text{hRLC}, \text{hRLW}, \text{hRLWWC}, \text{hRLWW}\}$ , all prefixes  $pr_1 \in \{\lambda, \text{fin}\}$ , all prefixes  $pref_X \in \{\lambda, \text{mon}, \text{det}, \text{det-mon}\}$ , and all  $k \geq 2$ , the following holds:*

$$\begin{aligned} \mathcal{L}_{\text{hP}}(k\text{-}pr_1\text{-}pref_X\text{-}X) &\subset \mathcal{L}_{\text{hP}}((k+1)\text{-}pr_1\text{-}pref_X\text{-}X) \text{ and} \\ \mathcal{L}_{\text{hP}}(\text{de}(k)\text{-}pr_1\text{-}pref_X\text{-}X) &\subset \mathcal{L}_{\text{hP}}(\text{de}(k+1)\text{-}pr_1\text{-}pref_X\text{-}X). \end{aligned}$$

**Remark.** The next theorem enables us to classify finite linguistic observations in a similar way as infinite formal languages. It refines a part of the Chomsky hierarchy and gives a useful tool for classifications of several syntactic phenomena of natural languages. In order to show the next theorem we consider the following sequences of languages for  $k \geq 1$ :  $L_{\text{cf}_{k+1}} = \{a^i b^{i-k} \mid i \geq 1\}$  and  $L_{\text{cs}_{k+1}} = \{a^i b^i c^{i-k}, a^{i+1} b^i c^{i-k} \mid i \geq 1\}$ .

**Theorem 16.** *For  $X = \text{hRLWWC}$ , and  $k \geq 2$  the following holds:*

- (1)  $\mathcal{L}_{\text{hP}}(k\text{-det-Rg2}) \subset \mathcal{L}_{\text{hP}}(k\text{-det-mon-}X) \subset \mathcal{L}_{\text{hP}}(k\text{-det-}X)$ ,
- (2)  $\mathcal{L}_A(k\text{-det-Rg2}) \subset \mathcal{L}_A(k\text{-det-mon-}X) \subset \mathcal{L}_A(k\text{-det-}X)$ ,
- (3)  $\mathcal{L}_{\text{hP}}(k\text{-det-fin-Rg2}) \subset \mathcal{L}_{\text{hP}}(k\text{-det-fin-mon-}X) \subset \mathcal{L}_{\text{hP}}(k\text{-det-fin-}X)$ ,
- (4)  $\mathcal{L}_A(k\text{-det-fin-Rg2}) \subset \mathcal{L}_A(k\text{-det-fin-mon-}X) \subset \mathcal{L}_A(k\text{-det-fin-}X)$ .

**Proof.** We outline the main ideas of the proof. For  $k \geq 2$  and  $X = \text{hRLWWC}$ , the following inclusions follow from definitions:

- (1)  $\mathcal{L}_{\text{hP}}(k\text{-det-Rg2}) \subseteq \mathcal{L}_{\text{hP}}(k\text{-det-mon-}X) \subseteq \mathcal{L}_{\text{hP}}(k\text{-det-}X)$ ,
- (2)  $\mathcal{L}_A(k\text{-det-Rg2}) \subseteq \mathcal{L}_A(k\text{-det-mon-}X) \subseteq \mathcal{L}_A(k\text{-det-}X)$ ,
- (3)  $\mathcal{L}_{\text{hP}}(k\text{-det-fin-Rg2}) \subseteq \mathcal{L}_{\text{hP}}(k\text{-det-fin-mon-}X) \subseteq \mathcal{L}_{\text{hP}}(k\text{-det-fin-}X)$ ,
- (4)  $\mathcal{L}_A(k\text{-det-fin-Rg2}) \subseteq \mathcal{L}_A(k\text{-det-fin-mon-}X) \subseteq \mathcal{L}_A(k\text{-det-fin-}X)$ .

It remains to show that all these inclusions are proper. We use the sequence of context-free languages  $L_{\text{cf}_k}$  to show the first proper inclusion in all four propositions.

For any natural number  $k \geq 2$ , it is not hard to construct a  $k\text{-det-mon-hRLC}$ -automaton  $M_{\text{cf}_k}$  such that  $L_C(M_{\text{cf}_k}) = L_{\text{hP}}(M_{\text{cf}_k}) = L_{\text{cf}_k}$ . By applying Proposition 14 for  $i = k$ , we can construct a  $k\text{-det-mon-fin}(k)\text{-hRLC}$ -automaton  $M_{\text{cf}'_k}$  such that  $L_C(M_{\text{cf}'_k}) = L_{\text{hP}}(M_{\text{cf}'_k}) = L_{\text{hP}}(M_{\text{cf}_k}, k)$ .

For a contradiction, let us suppose that there is a  $k\text{-det-Rg2}$ -automaton  $M_k$  such that  $L_{\text{hP}}(M_k) = L_{\text{hP}}(M_{\text{cf}_k}, k)$ . Let us consider the word  $a^{k+1} b^{(k+1)(k-1)} \in L_{\text{hP}}(M_{\text{cf}'_k})$ . As this word is longer than  $k$ ,  $M_k$  must use at least one cycle to accept a word  $w$  such that  $h(w) = a^{k+1} b^{(k+1)(k-1)}$ . Since any  $k\text{-det-Rg2}$ -automaton can delete only some of the first  $k$  symbols we have a contradiction to the complete correctness preserving property of  $M_k$ . Thus, the first inclusion is proper for all four propositions of the theorem.

Using languages  $L_{\text{cs}_k}$  we can show the second inclusion to be proper in all four propositions. Let  $k \geq 2$  be an integer. It is easy to construct a  $k\text{-det-hRLC}$ -automaton  $M_{\text{cs}_k}$  such that  $L_C(M_{\text{cs}_k}) = L_{\text{hP}}(M_{\text{cs}_k}) = L_{\text{cs}_k}$ . By applying Proposition 14 for  $i = k$ , we can construct a  $k\text{-det-fin}(k)\text{-hRLC}$ -automaton  $M_{\text{cs}'_k}$  such that  $L_C(M_{\text{cs}'_k}) = L_{\text{hP}}(M_{\text{cs}'_k}) = L_{\text{hP}}(M_{\text{cs}_k}, k)$ .

In order to derive a contradiction, let us assume that there is a  $k\text{-mon-det-hRLWWC}$ -automaton  $M_k$  such that

$L_{\text{hP}}(M_k) = L_{\text{hP}}(MCS_k, k)$ . Let us consider the word  $a^{k+1}b^{k+1}c^{(k+1)(k-1)} \in L_{\text{hP}}(MCS_k)$ . As this word is longer than  $2k$ ,  $M_k$  must use at least two cycles to accept a word  $w$  such that  $h(w) = a^{k+1}b^{k+1}c^{(k+1)(k-1)}$ . Because of the correctness preserving property  $M_k$  must reduce  $w$  into a word  $w_1$  such that  $h(w_1) = a^{k+1}b^k c^{k \cdot (k-1)}$ , and then it must reduce  $w_1$  to a word  $w_2$  such that  $h(w_2) = a^k b^k c^{k \cdot (k-1)}$ . But these two reductions violate the condition of monotonicity – a contradiction to the monotonicity constraint of  $M_k$ . Hence, the second inclusion is proper in all four propositions of the theorem.  $\square$

## 5 Conclusion

We have introduced the h-lexicalized restarting list automaton (LxRLAW), which yields a formal environment that is useful for expressing the lexicalized syntax in computational linguistics. We presented a basic variant of the Chomsky hierarchy of LxRLAW-automata, which can be interpreted as a hierarchy of classes of input, basic, and h-proper languages, and a hierarchy of h-syntactic analyses as well.

In the main part of the paper we have concentrated on h-lexicalized (deterministic) hRLWW-automata fulfilling the constraint of weak accepting form. We have stressed the transparency of computations of these automata for their basic and h-proper languages due to the complete correctness preserving property. We believe that the automata having the complete correctness preserving property constitute a meaningful class of automata and that they cover a significant class of languages, including the class CFL.

The newly added property of weak accepting form is particularly important for computational linguistics, since it allows to use finite observations (languages) for classifications of classes of infinite and finite languages as well, as we have shown in Section 4.3.

hRLWW-automata can be considered as a refined analytical counterpart to generative Marcus contextual grammars (see e.g. [8]) and Novotný's pure grammars (see e.g. [12]). Contextual and pure grammars work with the (complete) generative correctness preserving property. We have applied many useful techniques for the formalization of syntactic analysis of Czech sentences from Ladislav Nebeský (see e.g. [11]).

We plan in the close future to show a transfer from the complete correctness preserving monotone hRRWW-automaton to the complete correctness preserving monotone LxRLAW-automaton without any restart, which in fact works like a push-down automaton. Such an automaton computes in linear time. It is also possible to obtain similar hierarchies for this type of automata as for the hRRWW-automata from Section 4.3.

We also plan to study some relaxations of the complete correctness preserving property.

## References

- [1] Michal P. Chytil, Martin Plátek, Jörg Vogel: A note on the Chomsky hierarchy. *Bulletin of the EATCS* 27: 23–30 (1985)
- [2] Petr Jančar, František Mráz, Martin Plátek, Jörg Vogel: Restarting automata. In: Horst Reichel (ed.): FCT'95, Proc., pages 283–292, LNCS 965, Springer, Berlin (1995)
- [3] Petr Jančar, František Mráz, Martin Plátek: Forgetting Automata and Context-Free Languages. *Acta Informatica* 33: 409–420 (1996)
- [4] Petr Jančar, František Mráz, Martin Plátek, Jörg Vogel: On monotonic automata with a restart operation. *J. Autom. Lang. Comb.* 4: 287–311 (1999)
- [5] Petr Jančar, Martin Plátek, Jörg Vogel: Generalized linear list automata. ITAT 2004, Univerzita P. J. Šafárika v Košiciach, 2005, p. 97–105, ISBN 80-7097-589-X (2005)
- [6] Markéta Lopatková, Martin Plátek, Vladislav Kuboň: Modeling syntax of free word-order languages: Dependency analysis by reduction. In: Václav Matoušek, Pavel Mautner, Tomáš Pavelka (eds.), TSD 2005, Proceedings, pages 140–147, LNCS 3658, Springer, Berlin (2005)
- [7] Markéta Lopatková, Martin Plátek, Petr Sgall: Towards a formal model for functional generative description: Analysis by reduction and restarting automata. *Prague Bull. Math. Linguistics* 87: 7–26 (2007)
- [8] Solomon Marcus: Contextual grammars and natural languages. *Handbook of Formal Languages*, pages 215–235, Springer, Berlin (1997)
- [9] František Mráz: Lookahead hierarchies of restarting automata. *J. Autom. Lang. Comb.* 6: 493–506 (2001)
- [10] František Mráz, Friedrich Otto, Martin Plátek: The degree of word-expansion of lexicalized RRWW-automata – A new measure for the degree of nondeterminism of (context-free) languages. *Theoretical Computer Science* 410: 3530–3538 (2009)
- [11] Ladislav Nebeský: On One Formalization of Sentence Analysis. *Slovo a slovesnost*, 104–107, (1962)
- [12] Miroslav Novotný: With Algebra from Language to Grammar and back (in Czech: S algebrou od jazyka ke gramatice a zpět). Academia, Praha (1988)
- [13] Friedrich Otto: Restarting automata and their relations to the Chomsky hierarchy. In: Zoltan Esik, Zoltan Fülöp (eds.): Developments in Language Theory, Proceedings of DLT'2003, pages 55–74, LNCS 2710, Springer, Berlin (2003)
- [14] Martin Plátek: Two-way restarting automata and j-monotonicity. In: Leszek Pacholski, Peter Ružička (eds.): SOFSEM'01, Proc., pages 316–325, LNCS 2234, Springer, Berlin (2001)
- [15] Martin Plátek, Friedrich Otto, František Mráz: On h-Lexicalized Restarting List Automata, Technical report, [www.theory.informatik.uni-kassel.de/projekte/RL2016v6.4.pdf](http://www.theory.informatik.uni-kassel.de/projekte/RL2016v6.4.pdf), Kassel (2017)