On h-Lexicalized Restarting Automata

Martin Plátek
Charles University
Department of Computer Science
Malostranské nám. 25
118 00 PRAHA, Czech Republic
martin.platek@mff.cuni.cz

Friedrich Otto
Universität Kassel
Fachbereich Elektrotechnik/Informatik
34109 KASSEL, Germany
otto@theory.informatik.uni-kassel.de

Following some previous studies on restarting automata, we introduce a refined model – the *h-lexicalized restarting automaton* (h-RLWW). We argue that this model is useful for expressing lexicalized syntax in computational linguistics. We compare the input languages, which are the languages traditionally considered in automata theory, to the so-called *basic* and *h-proper languages*, which are (implicitly) used by categorial grammars, the original tool for the description of lexicalized syntax. The basic and h-proper languages allow us to stress several nice properties of h-lexicalized restarting automata, and they are suitable for modeling the analysis by reduction and, subsequently, for the development of categories of a lexicalized syntax. Based on the fact that a two-way deterministic monotone restarting automaton can be transformed into an equivalent deterministic monotone RL-automaton in (Marcus) contextual form, we obtain a transformation from monotone RLWW-automata that recognize the class CFL of context-free languages as their input languages to deterministic monotone h-RLWW-automata that recognize CFL through their h-proper languages. Through this transformation we obtain automata with the *complete correctness preserving property* and an infinite hierarchy within CFL, based on the size of the read/write window. Additionally, we consider h-RLWW-automata that are allowed to perform multiple rewrite steps per cycle, and we establish another infinite hierarchy above CFL that is based on the number of rewrite steps that may be executed within a cycle. The corresponding separation results and their proofs illustrate the transparency of h-RLWW-automata that work with the (complete or cyclic) correctness preserving property.

1 Introduction

The linguistic technique of 'analysis by reduction' is used to analyze sentences of natural languages with a high degree of word-order freedom like, e.g., Czech, Latin, or German (see, e.g., [13]). A human reader is supposed to understand the meaning of a given sentence before he starts to analyze it. Analysis by reduction (partially) simulates such a behavior by analyzing sentences, where morphological and syntactical tags have been added to the word-forms and punctuation marks (see, e.g., [14]).

In [5] the restarting automaton was presented as a formal device to model the naive (i.e. non-tagged) analysis by reduction. Such a restarting automaton has a finite-state control and a flexible tape with endmarkers on which a window of fixed finite size operates. The automaton works in cycles, where each cycle begins with the automaton being in its initial state with the window over the left end of the tape. Now it scans the tape from left to right until, at some place, it performs a combined rewrite/restart operation that deletes one or more symbols from the window, moves the window back to the left end of the tape, and returns the automaton to the initial state. As the tape is flexible, it adjusts automatically to the now shortened inscription. This model, nowadays called R-automaton, is quite restricted. Accordingly, in subsequent years this model was extended by allowing more general rewrite operations, by admitting additional symbols (so-called auxiliary symbols) in the working alphabet, and by separating the rewrite operation from the restart operation. In addition, models were proposed that can move their window in

E. Csuhaj-Varjú, P. Dömösi, Gy. Vaszil (Eds.): 15th International
Conference on Automata and Formal Languages (AFL 2017)
EPTCS 252, 2017, pp. 219–233, doi:10.4204/EPTCS.252.21

both directions, enabling them to first scan the given input completely before executing any rewrite step (for a survey see [18] or [19]).

To model the analysis by reduction on sequences of tagged items, *basic languages* and *proper languages* of restarting automata were considered in [17]. While the *input language* of an automaton M just consists of all input words that M can accept, the *basic language* of M consists of all words over the working alphabet that M can accept, and the *proper language* consists of all words that are obtained from the basic language by erasing all non-input (that is, auxiliary) symbols. Now one can argue that the auxiliary symbols represent the tags, and so, the basic language models the sequences of tagged items that are accepted. However, as it turned out, there are already deterministic restarting automata for which these proper languages are not even recursive, although the input language (and the basic language) of each deterministic restarting automaton is decidable in polynomial time.

Hence, *lexicalized* types of restarting automata were introduced in [17], in which the use of auxiliary symbols is somewhat restricted. A lexicalized restarting automaton is deterministic and there is a positive constant c such that essentially each factor of a word from the basic language that only consists of auxiliary symbols has length at most c. One of the main results of [17] states that the class of proper languages of lexicalized monotone RRWW-automata (see Sections 2 and 3 for the definitions) coincides with the class of context-free languages.

Here, in order to give a theoretical basis for lexicalized syntax, we introduce a model of the restarting automaton that formalizes lexicalization in a similar way as categorial grammars (see e.g. [2]) – the *h-lexicalized restarting automaton* (h-RLWW). This model is obtained from the two-way restarting automaton of [21] by adding a letter-to-letter morphism h that assigns an input symbol to each working symbol. Then the *h-proper language* of an h-RLWW-automaton M consists of all words $h(w)$, where w is taken from the basic language of 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 information (tags). We argue that this new model is better suited to the modeling of the lexicalized syntactic analysis and (lexicalized) analysis by reduction of natural languages (compare [13, 14]) through the use of basic and h-proper languages. We stress the fact that this model works directly with the text-editing operations (rewrite, delete).

We recall some constraints that are typical for restarting automata, and we outline ways for new combinations of constraints. As our basic technical result, we show that the expressive power of two-way deterministic monotone restarting automata (det-mon-RLWW-automata) does not decrease, if we use the corresponding type of automaton which instead of rewritings can only delete symbols (det-mon-RL-automata) and which are in the so-called (Marcus) *contextual form* (for *contextual grammars* see [15, 23]). In fact, these types of automata all characterize the class LRR of left-to-right regular languages [3]. The technique for this transformation is derived from the linguistic techniques for dependency syntax.

Then we show that the h-proper languages of monotone h-RRWW- and h-RLWW-automata just yield the context-free languages, both in the deterministic and in the nondeterministic case. In particular, this means that h-proper languages of deterministic monotone h-RLWW-automata properly extend the input languages of this type of automaton. Then we prove that the h-proper language of any h-RLWW-automaton occurs as the input language of some *shrinking* h-RLWW-automaton, but that there are input languages of RLWW-automata that do not occur as h-proper languages of h-RLWW-automata. Finally, based on the size of the read/write window, we establish infinite ascending hierarchies of language classes of h-proper languages for several subtypes of h-RLWW-automata. In particular, we obtain hierarchies within LRR and within CFL that have LRR and CFL as their limits. In addition, we also study h-RLWW-automata that are allowed to execute several rewrite steps within a cycle, and we establish infinite ascending hierarchies above CFL that are based on the number of rewrite steps that are executed within a cycle.

The paper is structured as follows. In Section 2, we introduce our model and its submodels, we define the h-proper languages, and we state the complete correctness preserving property and the complete error preserving property for the basic languages of deterministic h-RLWW-automata. In Section 3, we present the aforementioned characterization of the context-free languages through h-proper languages of deterministic monotone h-RLWW-automata, and we discuss the relationship between input and h-proper languages for some types of h-RLWW-automata. Then, in Section 4, we derive the aforementioned hierarchies inside LRR and CFL, and in Section 5 we consider h-RLWW-automata with several rewrites per cycle. The paper concludes with Section 6 in which we summarize our results and state some problems for future work.

2 Definitions

We start with the definition of the two-way restarting automaton. For technical reasons we propose a slight modification from the original definition given in [21].

Definition 1 *A* two-way restarting automaton*, an* RLWW-automaton *for short, is a machine with a single flexible tape and a finite-state control. It is defined through an 8-tuple $M = (Q,\Sigma,\Gamma,¢,\$,q_0,k,\delta)$, where Q is a finite set of states, Σ is a finite input alphabet, and Γ is a finite working alphabet containing Σ. The symbols from $\Gamma \smallsetminus \Sigma$ are called* auxiliary symbols*. Further, the symbols $¢,\$ \notin \Gamma$, 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*, 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,*

$$\mathscr{PC}^{\leq k} = (¢ \cdot \Gamma^{k-1}) \cup \Gamma^k \cup (\Gamma^{\leq k-1} \cdot \$) \cup (¢ \cdot \Gamma^{\leq k-2} \cdot \$)$$

is the set of possible contents *of the read/write window of M, and $v \in \mathscr{PC}^{\leq k-1}$.*

The transition relation describes six different types of transition steps (or instructions):

1. *A* move-right step $(q,u) \longrightarrow (q',\mathsf{MVR})$ *assumes that $(q',\mathsf{MVR}) \in \delta(q,u)$, where $q,q' \in Q$ and $u \in \mathscr{PC}^{\leq k}$, $u \neq \$$. If M is in state q and sees the word u in its read/write window, then 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 $(q,u) \longrightarrow (q',\mathsf{MVL})$ *assumes that $(q',\mathsf{MVL}) \in \delta(q,u)$, where $q,q' \in Q$ and $u \in \mathscr{PC}^{\leq k}$, $u \notin ¢ \cdot \Gamma^* \cdot \{\lambda,\$\}$ (Here λ is used to denote the* empty word*). It causes M to shift the window one position to the left and to enter state q'.*

3. *An* SL-step $(q,u) \longrightarrow (q',\mathsf{SL}(v))$ *assumes that $(q',\mathsf{SL}(v)) \in \delta(q,u)$, where $q,q' \in Q$, $u \in \mathscr{PC}^{\leq k}$, and $v \in \mathscr{PC}^{\leq k-1}$, that v is shorter than u, and that 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 $(q,u) \longrightarrow \mathsf{Restart}$ *assumes that $\mathsf{Restart} \in \delta(q,u)$, where $q \in Q$ and $u \in \mathscr{PC}^{\leq k}$. It causes M to move its window to the left end of the tape, so that the first symbol it sees is the left sentinel ¢, and to reenter the initial state q_0.*

5. *An* accept step $(q,u) \longrightarrow \mathsf{Accept}$ *assumes that* $\mathsf{Accept} \in \delta(q,u)$*, where* $q \in Q$ *and* $u \in \mathscr{PC}^{\leq k}$*. It causes M to halt and accept.*

6. *A* reject step $(q,u) \longrightarrow \mathsf{Reject}$ *assumes that* $\mathsf{Reject} \in \delta(q,u)$*, where* $q \in Q$ *and* $u \in \mathscr{PC}^{\leq k}$*. It causes M to halt and reject.*

There are two differences to the original definition given in [21] in that we have explicit reject steps and in that after a rewrite, that is, an SL-step, the window is not moved but just refilled from the left. It is easily seen that these modifications do not influence the expressive power of the model.

A *configuration* of an RLWW-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 $w \in \Gamma^*$; if $w \in \Sigma^*$, then $q_0¢w\$$ is an *initial configuration*. We see that any initial configuration is also a restarting configuration, and that any restart transfers M into a restarting configuration.

In general, an RLWW-automaton M is *nondeterministic*, 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 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 an initial or a restarting configuration and C_{i+1} is obtained from C_i by a step of M, for all $0 \leq i < j$. In the following we only consider computations of RLWW-automata which end either by an accept or by a reject step.

– Cycles and tails: Any finite computation of an RLWW-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*. Here we require that M executes *exactly one* SL-step within any cycle, and that it does not execute any SL-step in a tail. The latter is again a deviation from the original model, but in the nondeterministic setting, this does not influence the expressive power of the model, either.

We use the notation $q_0¢u\$ \vdash_M^c q_0¢v\$$ to denote a cycle of M that begins with the restarting configuration $q_0¢u\$$ and ends with the restarting configuration $q_0¢v\$$. Through this relation we define the relation of *cycle-rewriting* by M. We write $u \Rightarrow_M^c v$ iff $q_0¢u\$ \vdash_M^c q_0¢v\$$ holds. 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-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.

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

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 $L_C(M)$ we denote the language consisting of all words from Γ^* that are accepted by M; we say that M *recognizes (or accepts) the basic (or characteristic) language* $L_C(M)$.

By $\mathrm{P}r^{\Sigma}$ we denote the projection from Γ^* onto Σ^*, that is, $\mathrm{P}r^{\Sigma}$ is the morphism defined by $a \mapsto a$ for all $a \in \Sigma$ and $A \mapsto \lambda$ for all $A \in \Gamma \smallsetminus \Sigma$. If $v = \mathrm{P}r^{\Sigma}(w)$, then v is the Σ*-projection* of w, and w is an *extended version* of v. The *proper language* of M is now the language $L_P(M) = \mathrm{P}r^{\Sigma}(L_C(M))$, that is, a word $v \in \Sigma^*$ belongs to $L_P(M)$ iff there exists an expanded version w of v such that $w \in L_C(M)$.

Finally, we come to the definition of the central notion of this paper, the h-lexicalized RLWW-automaton.

Definition 2 *An* h-lexicalized RLWW-automaton*, or* h-RLWW*-automaton, is a pair* $\hat{M} = (M,h)$*, where* $M = (Q,\Sigma,\Gamma,¢,\$,q_0,k,\delta)$ *is an RLWW-automaton and* $h : \Gamma \to \Sigma$ *is a letter-to-letter morphism satisfying* $h(a) = a$ *for all input letters* $a \in \Sigma$*. The input language* $L(\hat{M})$ *of* $\hat{M}$ *is simply the language* $L(M)$ *and the basic language* $L_C(\hat{M})$ *of* $\hat{M}$ *is the language* $L_C(M)$*. Finally, we take* $L_{hP}(\hat{M}) = h(L_C(M))$*, and we say that* $\hat{M}$ recognizes (or accepts) the h-proper language $L_{hP}(\hat{M})$.

Notation. For brevity, the prefix det- will be used to denote the property of being deterministic. For any type A of restarting automaton, $\mathscr{L}(\mathsf{A})$ will denote the class of input languages that are recognized by automata from A, $\mathscr{L}_C(\mathsf{A})$ will denote the class of basic languages that are recognized by automata from A, and $\mathscr{L}_{hP}(\mathsf{A})$ will denote the class of h-proper languages that are recognized by automata from A. For a natural number $k \geq 1$, $\mathscr{L}(k\text{-}\mathsf{A})$ ($\mathscr{L}_C(k\text{-}\mathsf{A})$ or $\mathscr{L}_{hP}(k\text{-}\mathsf{A})$) will denote the class of input (basic or h-proper) languages that are recognized by those automata from A that use a read/write window of size k.

By $\subset$ we denote the proper subset relation. Finally, $\mathbb{N}_+$ will denote the set of all positive integers.

2.1 Further refinements and constraints on (h-lexicalized) RLWW-automata

Here we introduce some constrained types of rewrite steps.

A *delete-left step* $(q,u) \to (q',\mathsf{DL}(v))$ assumes that $(q',\mathsf{SL}(v)) \in \delta(q,u)$ and that v is a proper 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 ¢. Hence, the contents of the window is 'completed' from the left, and so the distance of the window to the left sentinel decreases if the window was not already at the left sentinel.

A *contextual-left step* $(q,u) \to (q',\mathsf{CL}(v))$ assumes that $(q',\mathsf{SL}(v)) \in \delta(q,u)$, where $u = v_1u_1v_2u_2v_3$ 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 ¢.

An RLWW-automaton is an *RLW-automaton* if its working alphabet coincides with its input alphabet, that is, no auxiliary symbols are available to this automaton. Note that in this situation, each restarting configuration is necessarily an initial configuration.

An RLW-automaton is an *RL-automaton* if all its rewrite steps are DL-steps, and it is an *RLC*-automaton if all its rewrite steps are CL-steps. Further, an RLWW-automaton is an *RLWWC-automaton* (that is, an *RLWW-automaton in Marcus contextual form*) if all its rewrite steps are CL-steps. Similarly, an RLWW-automaton is an *RLWWD-automaton* if all its rewrite steps are DL-steps. Observe that when concentrating on input languages, then DL- and CL-steps ensure that no auxiliary symbols can ever occur on the tape; if, however, we are interested in basic or h-proper languages, then auxiliary symbols can play an important role even though a given RLWW-automaton uses only DL- or CL-steps. Therefore, we distinguish between RLWWC- and RLC-automata, and between RLWWD- and RL-automata.

An RLWW-automaton is an *RRWW-automaton* if it does not use any MVL-steps. From these automata, we obtain *RRW-*, *RR-*, *RRC-*, *RRWWD-*, and *RRWWC-automata* in an analogous way. Further, an RRWW-automaton is an *RWW-automaton* if it restarts immediately after executing an SL-step. From these automata, we obtain *RW-*, *R-*, *RC-*, *RWWD-*, and *RWWC-automata.* Obviously, from these types of automata, we obtain corresponding types of h-lexicalized automata, denoted by the prefix h-. Observe that for an h-RLW-automaton (M,h), the letter-to-letter morphism h is necessarily the identity mapping on Σ. Hence, for such an automaton, the input language, the basic language, and the h-proper language all coincide.

We have the following simple facts, which illustrate the transparency of computations of deterministic RLWW-automata with respect to their basic languages.

Fact 3 (Complete Correctness Preserving Property for $\mathscr{L}_C(\mathsf{det\text{-}RLWW})$)
Let M be a deterministic RLWW-automaton, let $C = C_0, C_1, \ldots, C_n$ be a computation of M, and let $\text{¢}u_i\$$ 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 $j = 0, 1, \ldots, n$.

Fact 4 (Complete Error Preserving Property for $\mathscr{L}_C(\mathsf{det\text{-}RLWW})$)
Let M be a deterministic RLWW-automaton, let $C = C_0, C_1, \ldots, C_n$ be a computation of M, and let $\text{¢}u_i\$$ 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 $j = 0, 1, \ldots, n$.

Fact 5 (Error Preserving Property for $\mathscr{L}_C(\mathsf{RLWW})$)
Let M be an RLWW-automaton, let $C = C_0, C_1, \ldots, C_n$ be a computation of M, and let $\text{¢}u_i\$$ 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 $j \geq i$.

2.2 Proper languages versus h-proper languages

As each SL-step is length-reducing, and as each cycle of each computation of an RLWW-automaton contains an application of an SL-step, it is easily seen that the membership problem for the basic language $L_C(M)$ is decidable in polynomial time for each deterministic RLWW-automaton M. On the other hand, in [17] it is shown that there exist deterministic RRWW-automata for which the corresponding proper languages are non-recursive. This fact is derived from the result that, for each recursively enumerable language $L \subseteq \Sigma_0^+$, where $\Sigma_0 = \{a, b\}$, there exists a deterministic RRWW-automaton M such that $L_P(M) \cap \Sigma_0^+ \cdot c = \varphi(L)$, where c is an additional symbol, and φ is a finite-state transduction. Thus, these proper languages are much more sophisticated than the basic languages. In particular, there is no general relation between the terminal symbols and the auxiliary symbols in a word from the basic language $L_C(M)$.

To remedy this, a notion of *lexicalization* was introduced in [17]. Let $\mathrm{NIR}(M)$ denote the set of all words from Γ^* that are not immediately rejected by M, that is, the words that are either accepted in a tail or that cause M to perform at least one cycle. An RRWW-automaton M is called *lexicalized* if it is deterministic and if there exists a positive integer constant j such that, whenever a word $v \in (\Gamma \smallsetminus \Sigma)^*$ is a factor of a word $w \in \mathrm{NIR}(M)$, then $|v| \leq j$, that is, factors of words from $\mathrm{NIR}(M)$ that only consist of auxiliary symbols are at most of length j. It was then shown that the proper language $L_P(M)$ of any lexicalized RRWW-automaton is growing context-sensitive.

In the current paper we use the notion of *h-lexicalization* to obtain a stronger correspondence between the auxiliary symbols and the terminal symbols in a word, as each auxiliary symbol is mapped to a terminal symbol through the given morphism h. This corresponds to the process of annotation of a given terminal word, replacing each given lexical item through a tuple containing this item together with morphological and lexical syntactic information, which then form the basis for the subsequent analysis by reduction.

3 Robustness of monotone RLWW-automata

We recall the notion of *monotonicity* as an important constraint for computations of restarting automata. Let M be an RLWW-automaton, and let $C = C_k, C_{k+1}, \ldots, C_j$ be a sequence of configurations of M, where

C_{i+1} is obtained by a single transition step from C_i, $k \leq i < j$. We say that C is a *subcomputation* of M. If $C_i = ¢\alpha q\beta\$$, then $|\beta\$|$ is the *right distance* of C_i, which is denoted by $D_r(C_i)$. We say that a subsequence $(C_{i_1}, C_{i_2}, \ldots, C_{i_n})$ of C is *monotone* if $D_r(C_{i_1}) \geq D_r(C_{i_2}) \geq \cdots \geq D_r(C_{i_n})$. A computation of M is called *monotone* if the corresponding subsequence 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 *monotone* if each of its computations is monotone. We use the prefix *mon-* to denote monotone types of RLWW-automata. This notion of monotonicity has been considered before in various papers. The following results have been established.

Theorem 6 [7, 10, 12, 20, 21]

(a) $\mathsf{CFL} = \mathscr{L}(\text{mon-RWW}) = \mathscr{L}(\text{mon-RRWW}) = \mathscr{L}(\text{mon-RLWW})$.

(b) $\mathsf{LRR} = \mathscr{L}(\text{det-mon-RL}) = \mathscr{L}(\text{det-mon-RLW}) = \mathscr{L}(\text{det-mon-RLWW})$.

(c) $\mathsf{DCFL} = \mathscr{L}(\text{det-mon-RC}) = \mathscr{L}(\text{det-mon-RR}) = \mathscr{L}(\text{det-mon-RRWW})$.

Here DCFL denotes the class of *deterministic context-free languages*, LRR is the class of *left-to-right regular languages* from [3], and CFL is the class of *context-free languages*. Actually the result in Theorem 6 (b) can be strengthened as follows.

Theorem 7 $\mathsf{LRR} = \mathscr{L}(\text{det-mon-RLC}) = \mathscr{L}(\text{det-mon-RLWW})$.

Thus, for each det-mon-RLWW-automaton M_a, there is a det-mon-RLC-automaton M_b that accepts the same input language.

Proof outline. To prove this result one must overcome the problem that there is no straightforward simulation of a det-mon-RLWW-automaton by a det-mon-RLC-automaton. This follows from the observation that any det-mon-RLC-automaton fulfills the Complete Correctness Preserving Property for its input language and that, moreover, all its rewrite steps are contextual. On the other hand, the det-mon-RLWW-automaton M_a will in general not satisfy any type of correctness preserving property for its input language. This means in particular that the reductions of M_a and M_b will in general differ substantially. The second difficulty results from the problem of how to ensure that M_b chooses the correct places for executing CL-steps in a deterministic fashion without the ability to use any non-input symbols.

In [22] these problems are solved through a sequence of transformations. First it is shown that each det-mon-RLWW-automaton M_a can be transformed into an equivalent automaton M_1 that uses additional length-preserving rewrite steps, that has a window of size two only, and that works in three well-defined passes. During the first pass, M_1 just applies a sequence of MVR-steps until it reaches the right sentinel. During the second pass M_1 applies length-preserving rewrite steps and MVL-steps until the window reaches the left sentinel again. In this way information is encoded at each position of the input on the suffix to the right of that position. Finally, in the third pass M_1 only applies length-preserving rewrite steps, MVR-, and SL-steps, simulating the actual reductions of M_a. In addition, M_1 is monotone with respect to its SL-steps. As M_1 has a window of size two, each of its SL-steps can be interpreted as replacing the left symbol inside the window and deleting the right symbol from the window. It follows that the computation of M_1 can be described by a tree-like graph, called an SL-graph, that has nodes in one-to-one correspondence to the initial tape contents. Essentially, an SL-graph describes through its (oriented) edges which symbols are deleted with which symbols as their immediate left neighbours. Based on some combinatorial arguments cutting lemmas can be established for these graphs. These in turn lead to a simulation of M_1 by an automaton M_2 that works in so-called *strong cycles*, and that uses CL-steps instead of SL-steps. Each strong cycle of M_2 consists of three passes, similar to the passes of M_1, where the third pass ends with a *strong restart* that resets the automaton into its initial state, replaces each symbol currently on the tape by its associated input symbol with respect to the letter-to-letter morphism of M_2, and moves the window to the left end of the tape. Further, the window of M_2 is quite large, as the CL-steps of M_2 do not correspond to the SL-steps of M_1, but rather a single CL-step simulates the effect of a whole sequence of SL-steps. Finally, it can be shown that the effect of the three passes of the strong cycles of M_2 and the subsequent strong restart step can be described through a sequence of four deterministic two-way finite-state transducers. As the class of functions that are computable by these transducers is closed under composition [1], it follows that the transformation on initial configurations that is induced by the strong cycles of M_2 can be realized by a deterministic two-way finite-state transducer. Now this transducer can be converted into a det-mon-RLC-automaton M_b that still accepts the same language as M_a. □

3.1 Robustness of monotonicity and h-proper languages

The following theorem extends the basic theorem from [17] and completes the aforementioned characterization of the context-free languages.

Theorem 8 $$\begin{aligned}\mathsf{CFL} &= \mathscr{L}_{hP}(\mathsf{det\text{-}mon\text{-}h\text{-}RRWWC}) = \mathscr{L}_{hP}(\mathsf{mon\text{-}h\text{-}RRWW})\\ &= \mathscr{L}_{hP}(\mathsf{det\text{-}mon\text{-}h\text{-}RLWW}) = \mathscr{L}_{hP}(\mathsf{mon\text{-}h\text{-}RLWW}).\end{aligned}$$

Proof. The proof is based mainly on ideas from [17]. Here it is carried over from monotone lexicalized RRWW-automata to monotone h-RLWW-automata.

Let $M = ((Q,\Sigma,\Gamma,¢,\$,q_0,k,\delta),h)$ be a monotone h-RLWW-automaton. Then the characteristic language $L_C(M)$ is context-free [7, 21]. As $L_{hP}(M) = h(L_C(M))$, and as CFL is closed under morphisms, it follows that $L_{hP}(M)$ is 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 that is in *Greibach normal form*, that is, each rule of P has the form $A \to a\alpha$ for some symbol $a \in \Sigma$ and for some word $\alpha \in N^*$ (see, e.g., [4]). For the following construction we assume that the rules of G are numbered from 1 to m.

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

Obviously, a word $\omega \in B^*$ belongs to $L(G')$ if and only if ω 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 $a_1,a_2,\ldots,a_n \in \Sigma$, $i_1,i_2,\ldots,i_n \in \{1,2,\ldots,m\}$, and the sequence of indices $(i_1,i_2,\ldots,i_n)$ describes a (left-most) derivation of $w = a_1a_2\cdots a_n$ from S in G. If we define a letter-to-letter morphism $h : B^* \to \Sigma^*$ by taking $h((\nabla_i,a)) = a$ for all $(\nabla_i,a) \in B$, then it follows that $h(L(G')) = L(G) = L$. From ω the derivation of w from S in G can be reconstructed deterministically. In fact, the language $L(G')$ is deterministic context-free. Hence, there exists a deterministic monotone RRC-automaton M for this language (see [7]). By interpreting the symbols of B as auxiliary symbols and by adding the terminal alphabet Σ, we obtain a deterministic monotone RRWWC-automaton M' from M such that $h(L_C(M')) = h(L(M)) = h(L(G')) = L$. Thus, (M',h) is a deterministic monotone h-RRWWC-automaton with h-proper language L. Observe that the input language $L(M')$ of M' is actually empty. □

Using automata, the construction above illustrates the linguistic effort to obtain a set of categories (auxiliary symbols) that ensures the correctness preserving property for the corresponding analysis by reduction. Note that in its reductions, the automaton M' above only uses delete steps (in a special way). This is highly reminiscent of the basic (elementary) analysis by reduction learned in (Czech) elementary schools.

From our results derived so far we obtain the following separation result, showing that for deterministic monotone h-RLWW-automata, the h-proper languages strictly contain the input languages.

Corollary 9 $\mathsf{LRR} = \mathscr{L}(\mathsf{det\text{-}mon\text{-}h\text{-}RLWW}) \subset \mathscr{L}_{hP}(\mathsf{det\text{-}mon\text{-}h\text{-}RLWW}\ ) = \mathsf{CFL}$.

In the following we will shortly consider another generalization of the RLWW-automaton, the so-called *shrinking* RLWW-automaton, see [11]. A shrinking RLWW-automaton M is defined just like an RLWW-automaton with the exception that it is no longer required that each rewrite step of M must be length-reducing. Instead it is required that there exists a weight function ω that assigns a positive integer $\omega(a)$ to each letter a of M's working alphabet Γ such that, for each cycle-rewriting $u \Rightarrow^c_M v$ of M, $\omega(u) > \omega(v)$ holds. Here the function ω is extended to a morphism $\omega : \Gamma^* \to \mathbb{N}$ as usual. We denote shrinking RLWW-automata by sRLWW. In the following (see the proof of the next proposition) we will work with a special, linguistically motivated, weight function, which we will call *lexical disambiguation*. Let us recall that reductions are cycle-rewritings that decrease the length of the tape contents.

Proposition 10 *For each h-RLWW-automaton M, there exists an h-sRLWW-automaton M_s such that $L(M_s) = L_{hP}(M_s) = L_{hP}(M)$, and there is a one-to-one correspondence between the sets of (length-reducing) reductions of M_s and M.*

Proof. Let $M = (\hat{M}, h) = ((Q, \Sigma, \Gamma, ¢, \$, \delta, q_0, k), h)$ be an h-RLWW-automaton with basic language $L_C(M)$ and h-proper language $L_{hP}(M) = h(L_C(M))$. We construct an h-sRLWW automaton $M_s = (\hat{M}_s, h_s) = ((Q_s, \Sigma, \Gamma_s, ¢, \$, \delta_s,\ q_0, k), h_s)$ with weight function ω_s as follows. Let $\Gamma_s = \Gamma \cup \{\, \hat{a} \mid a \in \Sigma \,\}$, where $\hat{a}$ ($a \in \Sigma$) are new working symbols such that $h_s(\hat{a}) = a$, and let $h_s(b) = h(b)$ for all $b \in \Gamma$.

We say that the *degree of lexical ambiguity* of an input symbol $a \in \Sigma$ is j, if the set $\{\, d \in \Gamma \mid h(d) = a \,\}$ has cardinality j. This is expressed as $\mathrm{dga}(a) = j$. Now we define the weight function ω_s by taking, for each $a \in \Sigma$, $\omega_s(a) = \mathrm{dga}(a) + 1$, and for each $b \in \Gamma_s \smallsetminus \Sigma$, $\omega_s(b) = 1$.

The automaton $\hat{M}_s$ will work in two phases. In the first phase, $\hat{M}_s$ nondeterministically performs the so-called *lexical analysis*, that is, the input symbols on the tape are replaced by symbols from $\Gamma_s \smallsetminus \Sigma$ in such a way that each $a \in \Sigma$ on the tape is rewritten by a symbol b such that $h_s(b) = a$. For example, this is achieved by processing the tape from right to left, replacing a single symbol in each cycle. Obviously, each of these rewritings is strictly weight-decreasing with respect to the weight function ω_s. Phase one ends as soon as the first symbol to the right of the left sentinel ¢ has been rewritten.

In the second phase $\hat{M}_s$ simply simulates the given RLWW-automaton $\hat{M}$ on the rewritten tape contents. During this phase the new auxiliary symbols $\hat{a}$ are used as substitutes for the input symbols a, that is, no input symbols occur on the tape during this phase. Now it follows that $L(M_s) = L_{hP}(M_s) = h(L_C(M)) = L_{hP}(M)$. As all the rewrite steps of $\hat{M}$ are strictly length-reducing, it follows that all the simulating steps of $\hat{M}_s$ are shrinking with respect to the weight function ω_s. Moreover, it is easily seen that the sets of reductions of M_s and M coincide (up to the replacement of a by $\hat{a}$ for all $a \in \Sigma$). □

The construction above illustrates the linguistic technique of composing lexical analysis with the analysis by reduction within a (basic) syntactic analyzer. Observe, however, that not every language from $\mathscr{L}(\mathsf{RLWW})$ is the h-proper language of some h-RLWW-automaton.

Proposition 11 *The language $L_e = \{\, a^{2^n} \mid n \in \mathbb{N} \,\} \in \mathscr{L}(\mathsf{det\text{-}RLWW})$ is not the h-proper language of any h-RLWW-automaton.*

Proof. It is not hard to construct a det-RLWW-automaton M_e such that $L(M_e) = L_e$ using the ideas from [6]. In fact, let $M_e = (\{q_0, q_1\}, \{a\}, \{a, b\}, ¢, \$, \delta_e, q_0, 3)$ be the deterministic RWW-automaton that is specified by the following transitions:

$$\begin{array}{lcllcl}
\delta_e(q_0, ¢a\$) & = & \mathsf{Accept}, & \delta_e(q_0, ¢b\$) & = & \mathsf{Accept}, \\
\delta_e(q_0, ¢aa) & = & (q_0, \mathsf{MVR}), & \delta_e(q_0, ¢bb) & = & (q_0, \mathsf{MVR}), \\
\delta_e(q_0, aaa) & = & (q_0, \mathsf{MVR}), & \delta_e(q_0, bbb) & = & (q_0, \mathsf{MVR}), \\
\delta_e(q_0, aa\$) & = & (q_1, \mathsf{SL}(b\$)), & \delta_e(q_0, bb\$) & = & (q_1, \mathsf{SL}(a\$)), \\
\delta_e(q_0, aab) & = & (q_1, \mathsf{SL}(bb)), & \delta_e(q_0, bba) & = & (q_1, \mathsf{SL}(aa)), \\
\delta_e(q_0, ¢ab) & = & \mathsf{Reject}, & \delta_e(q_0, ¢ba) & = & \mathsf{Reject}, \\
\delta_e(q_1, x) & = & \mathsf{Restart} & \text{for all } x \in \{a, b, ¢, \$\}^{\leq 3}. & &
\end{array}$$

Given a word $w = a^m$ as input, M_e will rewrite w from right to left, replacing each factor aa by the symbol b. This continues until no more rewrites of this form are possible. If m is uneven, then M_e halts and rejects, otherwise, the word b^{m_1} is obtained with $m_1 = \frac{m}{2}$. In the latter case, M_e will rewrite b^{m_1} from right to left, replacing each factor bb by the symbol a. Again this continues until no more rewrites of this form are possible. If m_1 is uneven, then M_2 halts and rejects, otherwise, the word a^{m_2} is obtained with $m_2 = \frac{m_1}{2} = \frac{m}{4}$. It can now be easily seen that $L(M_e) = L_e$.

To show that $L_e \notin \mathscr{L}_{hP}(\mathsf{h\text{-}RLWW})$, assume that $L_e = L_{hP}(M)$ for some h-RLWW-automaton $(M, h) = ((Q, \{a\}, \Gamma, ¢, \$, \delta, q_0, k), h)$. Let $z = a^{2^n} \in L_e$, where n is a sufficiently large integer satisfying $2^n - k > 2^{n-1}$, and such that M makes at least one cycle within an accepting computation on some word $w \in \Gamma^*$ satisfying $w \in L_C(M)$ and $h(w) = z$. It is easily seen that such a word w exists, since otherwise the set of words accepted by M would be a regular language. The accepting computation on w begins with a reduction of the form $w \Rightarrow_M^c w'$. Since $w \Rightarrow_M^c w'$ is a part of an accepting computation, it follows that $w' \in L_C(M)$, which in turn implies that $h(w') \in L_e$. Thus, $h(w') = a^m$ for some integer m satisfying $2^n - k \leq m < 2^n$. As $2^{n-1} < 2^n - k \leq m < 2^n$, this is a contradiction. Hence, L_e is not the h-proper language of any h-RLWW-automaton. □

Together with Proposition 10 this yields the following proper inclusion.

Corollary 12 $\mathscr{L}_{hP}(\mathsf{h\text{-}RLWW}) \subset \mathscr{L}(\mathsf{sRLWW})$.

It can be shown, however, that the language L_e, which is not the h-proper language of any h-RLWW-automaton, is in fact the h-proper language of a deterministic h-sRLWW-automaton that only has length-preserving rewrite steps.

4 Hierarchies based on window size

In this section we transfer, extend, and generalize the results of [16] from input languages to h-proper languages and from RRW-automata to h-RLWW-automata. We will show that the classes of h-proper languages that are accepted by h-RLWW-automata (and by several subclasses of h-RLWW-automata) form infinite ascending hierarchies with respect to the size of the read/write window. As separating witness languages, we can actually use the same languages as in [16]. In addition, we obtain a proper infinite ascending hierarchy within the left-to-right regular languages that converges to the class LRR, and we obtain a proper infinite ascending hierarchy within the context-free languages that converges to the complete class CFL. For input languages of RRWW- and RLWW-automata, an analogous result is impossible, as it follows from [16, 24] that the corresponding hierarchies for the input languages of RRWW- and RLWW-automata (and of several subclasses of RLWW-automata) collapse into only two classes: those that are accepted by RLWW-automata with window size one, and those that are accepted by RLWW-automata with window size two or larger. The new result on the hierarchy for the complete class CFL stresses the meaning of the main results from the previous section, since it is based on them.

Recall that an RWW-automaton must restart immediately after executing a rewrite step (see Subsection 2.1). From [16] we know that RWW-automata with a window of size one are fairly weak. Observe that the rewrite steps of any RLWW-automaton with window size one do just delete single symbols. Accordingly, we have the following characterization, where REG denotes the class of regular languages.

Lemma 13

$$\mathsf{REG} \;=\; \mathscr{L}(\mathsf{1\text{-}RC}) \;=\; \mathscr{L}(\mathsf{1\text{-}R}) \;=\; \mathscr{L}(\mathsf{1\text{-}RW}) \;=\; \mathscr{L}(\mathsf{1\text{-}RWW}) \;=\; \mathscr{L}_{hP}(\mathsf{1\text{-}h\text{-}RWW}).$$

On the other hand, RRWW-automata with a window of size one are more expressive – already a 1-RR-automaton can accept a non-context-free language [16].

Let D_1 denote the Dyck language over the alphabet $\{a_1, \bar{a}_1\}$, that is, D_1 is the language that is generated by the context-free grammar $G = (\{S\}, \{a_1, \bar{a}_1\}, S, P)$ with the set of rules $P = \{S \to a_1 S \bar{a}_1, S \to SS, S \to \lambda\}$. An alternative way to describe D_1 is to interpret a_1 as a left bracket and $\bar{a}_1$ as a right bracket, and then D_1 is the set of all words consisting of well-balanced brackets. The language D_1 is deterministic context-free, but it is not regular. However, it is accepted by a deterministic 2-det-mon-RC-automaton that just scans its tape from left to right and deletes the first factor $a_1\bar{a}_1$ that it encounters.

Lemma 14 [16] $D_1 \in \mathscr{L}(\mathsf{2\text{-}det\text{-}mon\text{-}RC})$.

The next technical lemma lays the foundation for the hierarchies mentioned above.

Lemma 15 *For all* $k \geq 1$, $\mathscr{L}((k+1)\text{-}\mathsf{det\text{-}mon\text{-}RC}) \smallsetminus \mathscr{L}_{hP}(k\text{-}\mathsf{h\text{-}RLWW}) \neq \emptyset$.

Proof. We provide a sequence of languages $\{L_k\}_{k=1}^{\infty}$ which satisfy

$$L_k \in \mathscr{L}((k+1)\text{-}\mathsf{det\text{-}mon\text{-}RC}) \smallsetminus \mathscr{L}_{hP}(k\text{-}\mathsf{h\text{-}RLWW}).$$

1. For $k = 1$, we take the language $L_1 = D_1$. By Lemma 14, $D_1 \in \mathscr{L}(\mathsf{2\text{-}det\text{-}mon\text{-}RC})$. Now assume that D_1 is the h-proper language of an h-RLWW-automaton $M_1 = ((Q, \Sigma, \Gamma, ¢, \$, q_0, 1, \delta), h)$ with window size one. As D_1 is not regular, there exists an integer n such that there is a word $w \in L_C(M_1)$ such that $h(w) = a_1^n \bar{a}_1^n \in D_1$ and w is not accepted by a tail computation of M_1. Now we consider an accepting computation of M_1 that begins with the restarting configuration $q_0¢w\$$. As this is not a tail computation, it contains a first cycle in which the word w is shortened to a word w'. As M_1 has window size one, this means that $|w'| = 2n-1 = |h(w')|$. Then $h(w') \in L_{hP}(M_1)$, but as D_1 does not contain any words of uneven length, this is a contradiction. Thus, D_1 is not the h-proper language of any 1-h-RLWW-automaton.

2. For $k \geq 2$, let $L_k = \{\, a^n c^{k-1} b^n \mid n \geq 0 \,\}$. One can easily design a det-mon-RC-automaton M'_k with a window of size $k+1$ such that $L_k = L(M'_k)$, which shows that $L_k \in \mathscr{L}((k+1)\text{-}\mathsf{det\text{-}mon\text{-}RC})$. Now assume that L_k is the h-proper language of an h-RLWW-automaton $M_k = ((Q, \Sigma, \Gamma, ¢, \$, q_0, k, \delta), h)$ with window size k. Again, as L_k is not regular, there exists an integer n such that there is a word $w \in L_C(M_k)$ such that $h(w) = a^n c^{k-1} b^n \in L_k$ and w is not accepted by a tail computation of M_k. Now we consider an accepting computation of M_k that begins with the restarting configuration $q_0¢w\$$. As this is not a tail computation, it contains a first cycle in which the word w is reduced to a shorter word w' such that $2n-1 \leq |w'| = |h(w')| \leq 2n+k-2$. However, as the window size of M_k is only k, we see that either the prefix corresponding to a^n or the suffix corresponding to b^n of the

word w is not altered through this rewrite step, which implies that $h(w')$ either has the prefix a^n or the suffix b^n. This is a contradiction, as no word from L_k of length at most $2n+k-2$ has prefix a^n or suffix b^n. Thus, it follows that L_k is not the h-proper language of any h-RLWW-automaton with window size k. □

Let us note that in the above proof the language $L_1' = \{a^n b^n \mid n \geq 0\}$ cannot be used to separate $\mathscr{L}(\text{1-R})$ from $\mathscr{L}(\text{2-R})$ or $\mathscr{L}(\text{1-RW})$ from $\mathscr{L}(\text{2-RW})$, etc., since $L_1' \notin \mathscr{L}(\text{2-RW})$ [16]. The above lemma yields the following hierarchy results.

Corollary 16 *For all* $X,Y \in \{\mathsf{RC},\mathsf{R},\mathsf{RW},\mathsf{h\text{-}RWW},\mathsf{h\text{-}RWWC},\mathsf{h\text{-}RWWD},\mathsf{RRC},\mathsf{RR},\mathsf{RRW},\mathsf{h\text{-}RRWW},\mathsf{h\text{-}RRWWC},\mathsf{h\text{-}RRWWD},\mathsf{RLC},\mathsf{RL},\mathsf{RLW},\mathsf{h\text{-}RLWW},\mathsf{h\text{-}RLWWC},\mathsf{h\text{-}RLWWD}\}$*, all prefixes* $\mathit{pref}_X,\mathit{pref}_Y \in \{\lambda,\mathsf{det},\mathsf{mon},\,\mathsf{det\text{-}mon}\}$*, and all* $k \geq 1$*, the following hold:*

(a) $\mathscr{L}_{hP}(k\text{-}\mathsf{pref}_X\text{-}X) \subset \mathscr{L}_{hP}((k+1)\text{-}\mathsf{pref}_X\text{-}X)$.

(b) $\mathscr{L}_{hP}((k+1)\text{-}\mathsf{pref}_X\text{-}X) \smallsetminus \mathscr{L}_{hP}(k\text{-}\mathsf{pref}_Y\text{-}Y) \neq \emptyset$.

For example, if $\mathit{pref}_X = \mathit{mon}$ and $X = \mathsf{R}$, the expression $\mathit{pref}_X\text{-}X$ denotes mon-R, and for $\mathit{pref}_X = \lambda$ and $X = \mathsf{RRW}$, the expression $\mathit{pref}_X\text{-}X$ denotes RRW.

From Theorem 7 and Corollary 16, we see that the classes of h-proper (and input) languages that are accepted by the deterministic monotone versions of RLC-, RL-, and RLW-automata form infinite strictly ascending hierarchies within the language class LRR.

Corollary 17 *For all* $\mathsf{X} \in \{\mathsf{RLC},\mathsf{RL},\mathsf{RLW}\}$*, the following hold:*

(a) *For all* $k \geq 1$, $\mathscr{L}_{hP}(k\text{-}\mathsf{det\text{-}mon}\text{-}X) = \mathscr{L}(k\text{-}\mathsf{det\text{-}mon\text{-}X}) \subset \mathscr{L}_{hP}((k+1)\text{-}\mathsf{det\text{-}mon\text{-}X}) \subset \mathsf{LRR}$.

(b) $\bigcup_{k=1}^{\infty} \mathscr{L}_{hP}(k\text{-}\mathsf{det\text{-}mon\text{-}X}) = \bigcup_{k=1}^{\infty} \mathscr{L}(k\text{-}\mathsf{det\text{-}mon\text{-}X}) = \mathsf{LRR}$.

From Theorem 8 and Corollary 16, we obtain that the classes of h-proper languages that are accepted by the monotone versions of deterministic and nondeterministic h-RRWW- and h-RLWW-automata form proper ascending hierarchies within the class CFL.

Corollary 18 *For all* $\mathsf{X} \in \{\mathsf{h\text{-}RRWW},\mathsf{h\text{-}RLWW}\}$ *and all pref* $\in \{\lambda,\, \mathit{det}\}$ *the following hold:*

(a) *For all* $k \geq 1$, $\mathscr{L}_{hP}(k\text{-}\mathsf{pref\text{-}mon\text{-}X}) \subset \mathscr{L}_{hP}((k+1)\text{-}\mathsf{pref\text{-}mon\text{-}X}) \subset \mathsf{CFL}$.

(b) $\bigcup_{k=1}^{\infty} \mathscr{L}_{hP}(k\text{-}\mathsf{pref\text{-}mon\text{-}X}) = \mathsf{CFL}$.

5 Restarting automata with multiple rewrites

In this section we consider still another generalization of the RLWW-automaton, the h-RLWW-automaton *with multiple rewrites* (mrRLWW-automaton for short). The aim of this generalization is to obtain a transparent tool which is strong enough to cover the (surface) syntax of natural languages. An mrRLWW-automaton M is defined just like an h-RLWW-automaton with the exception that it is required that, in each cycle of a computation, the automaton M must execute a positive number of SL-steps. By $\mathsf{mrRLWW}(j)$ we denote the class of mrRLWW-automata for which the number of SL-steps in a cycle is limited by the number j. For mrRLWW-automata we will consider similar subclasses as for h-RLWW-automata and we will use similar denotations for them. Further, we use the same important notions for them as for h-RLWW-automata.

It is easily seen that for deterministic mrRLWW-automata and their basic languages, the complete correctness (error) preserving property does not hold. However, we have the following useful facts on their computations.

Fact 19 (Cycle Error Preserving Property)
Let M be an mrRLWW-automaton. If $u \Rightarrow_M^{c^*} v$ *and* $u \notin L_C(M)$*, then* $v \notin L_C(M)$*.*

Fact 20 (Cycle Correctness Preserving Property)
Let M be a deterministic mrRLWW-automaton. If $u \Rightarrow_M^{c^*} v$ *and* $u \in L_C(M)$*, then* $v \in L_C(M)$*.*

These two facts ensure the transparency for computations of mrRLWW-automata and their basic and h-proper languages. From the results obtained above, we get the following corollary.

Corollary 21 *For all* $\mathsf{X} \in \{\mathsf{mrRRWW}, \mathsf{mrRRWWD}, \mathsf{mrRRWWC}, \mathsf{mrRLWW}, \mathsf{mrRLWWD}, \mathsf{mrRLWWC}\}$ *and all pref* $\in \{\lambda, \, det\}$ *the following hold:*

(a) $\mathsf{CFL} \ \subseteq \ \mathscr{L}_{hP}(\mathsf{pref\text{-}X})$.

(b) $\mathsf{CFL} \ \subset \ \mathscr{L}(\mathsf{mrRRWW}) \subseteq \mathscr{L}(\mathsf{mrRLWW})$.

This corollary ensures the basic request on the power of mrRLWW-automata and its variants. The following results support the idea that mrRLWW-automata are strong enough to cover the (surface) syntax of natural languages.

Lemma 22 *For all* $j \geq 1$, $\mathscr{L}(\mathsf{det\text{-}mrRRC}(j+1)) \smallsetminus \mathscr{L}_{hP}(\mathsf{mrRLWW}(j)) \neq \emptyset$*.*

Proof. We provide a sequence of languages $\{Lm_j\}_{j=1}^{\infty}$ which satisfy

$$Lm_j \in \mathscr{L}(\mathsf{det\text{-}mrRRC}(j+1)) \smallsetminus \mathscr{L}_{hP}(\mathsf{mrRLWW}(j)).$$

1. For $j = 1$, we take the language $Lm_1 = \{ucu \mid u \in \{a,b\}^*\}$. It is not hard to show that $Lm_1 \in \mathscr{L}(\mathsf{det\text{-}mrRRC}(2))$. In fact, this language is accepted by a det-mrRRC(2)-automaton M_1' which, in each cycle, simply deletes the first symbol and the first symbol following the symbol c if these two symbols coincide. In a tail M_1' just accepts c.

   Now assume that Lm_1 is the h-proper language of an h-RLWW-automaton $M_1 = ((Q,\Sigma,\Gamma,¢,\$,q_0,k,\delta),h)$. As Lm_1 is not regular, there exists an integer $n > k$ such that there is a word $w \in L_C(M_1)$ such that $h(w) = a^n b^n c a^n b^n \in Lm_1$ and w is not accepted by a tail computation of M_1. Now we consider an accepting computation of M_1 that begins with the restarting configuration $q_0¢w\$$. As this is not a tail computation, it contains a first cycle in which the word w is shortened to a word w'. As M_1 has window size k, and as it can use exactly one SL-step in a cycle, it follows than $h(w')$ cannot be from Lm_1. This is a contradiction, since any cycle in an accepting computation is correctness preserving. Thus, Lm_1 is not the h-proper language of any h-RLWW-automaton.

2. For $j \geq 2$, let $Lm_j = \{\, (uc)^j u \mid u \in \{a,b\}^* \,\}$. One can easily design a det-mrRRC($j+1$)-automaton M_j' for Lm_j, which shows that $Lm_j \in \mathscr{L}(\mathsf{det\text{-}mrRRC}(j+1))$.

   Now assume that Lm_j is the h-proper language of an mrRLWW(j)-automaton $M_j = ((Q,\Sigma,\Gamma,¢,\$,q_0,k,\delta),h)$. Again, as Lm_j is not regular, there exists an integer $n > k$ such that there is a word $w \in L_C(M_j)$ such that $h(w) = (a^n b^n c)^j a^n b^n \in Lm_j$ and w is not accepted by a tail computation of M_j. Now we consider an accepting computation of M_j that begins with the restarting configuration $q_0¢w\$$. As this is not a tail computation, it contains a first cycle in which the word w is reduced to a shorter word w', where at least one factor v of w satisfying $h(v) = a^n$ (or $h(v) = b^n$) is shortened or at least one c is deleted, while another factor y of w satisfying $h(y) = a^n$ (or $h(y) = b^n$) remains unchanged. This, however, means that $h(w') \notin Lm_j$, a contradiction. Thus, it follows that Lm_j is not the h-proper language of any mrRLWW(j)-automaton. □

Thus, we obtain the following consequences.

Corollary 23 *For all* $X,Y \in \{\mathsf{mrRRC}, \mathsf{mrRR}, \mathsf{mrRRW}, \mathsf{mrRRWW}, \mathsf{mrRRWWD}, \mathsf{mrRRWWC}, \mathsf{mrRLC}, \mathsf{mrRL}, \mathsf{mrRLW}, \mathsf{mrRLWW}, \mathsf{mrRLWWD}, \mathsf{mrRLWWC}\}$*, all prefixes* $\mathit{pref}_X, \mathit{pref}_Y \in \{\lambda, \mathsf{det}\}$*, and all* $k \geq 1$*, the following hold:*

(a) $\mathscr{L}_{hP}(\mathsf{pref}_X\text{-}X(k)) \quad \subset \quad \mathscr{L}_{hP}(\mathsf{pref}_X\text{-}X(k+1)).$

(b) $\mathscr{L}_{hP}(\mathsf{pref}_X\text{-}X(k+1)) \quad \setminus \quad \mathscr{L}_{hP}(\mathsf{pref}_Y\text{-}Y(k)) \neq \emptyset.$

We believe that already the class of mrRLWWD(2)-automata is strong enough to model lexicalized (surface) syntax of natural languages, that is, to model their analysis by reduction. In the future we will investigate the relationship of mrRLWWD(2)-automata to the class of mildly context-sensitive languages [8, 9].

6 Conclusion

We have introduced the h-lexicalized extension of the RLWW-automaton, which yields a formal environment that is useful for expressing the lexicalized syntax in computational linguistics. Then we presented the input, basic, and h-proper languages of these automata, and we compared the input languages, which are the languages that are traditionally considered in automata theory, to the basic and h-proper languages, which leads to *error preserving computations* for h-RLWW-automata, and in the deterministic case it yields *complete correctness preserving computations*. Based on the result that each det-mon-RLWW-automaton can be transformed into a det-mon-RLC-automaton that accepts the same input language, we obtained a transformation from mon-RLWW-automata that characterize the class CFL of context-free languages through their input languages to det-mon-h-RLWW-automata that characterize the class CFL through their h-proper languages. Through this transformation we have obtained automata with the complete correctness preserving property and infinite ascending hierarchies within the classes LRR and CFL, based on the size of the read/write window. Finally, we have introduced classes of restarting automata with transparent computing which are strong enough to model the syntax of natural languages (like, e.g., Czech, German or English). For the future we are interested in deriving a characterization of mildly context-sensitive languages [8, 9] by suitable classes of restarting (and list) automata that have the correctness preserving property.

Acknowledgement. The authors thank their colleague František Mráz for many helpful discussions on the results presented.

References

[1] Alfred V. Aho & Jeffrey D. Ullman (1970): *A characterization of two-way deterministic classes of languages. J. Comput. Syst. Sci.* 4, pp. 523-538, doi:10.1016/S0022-0000(70)80027-7.

[2] Yehoshua Bar-Hillel (1953): *A quasi-arithmetical notation for syntactic description. Language* 29, pp. 47-58, doi:10.2307/410452.

[3] Karel Čulik II & Rina Cohen (1973): *LR-regular grammars - an extension of LR(k) grammars. J. Comput. Syst. Sci.* 7, pp. 66-96, doi:10.1016/S0022-0000(73)80050-9.

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

[5] Petr Jančar, František Mráz, Martin Plátek & Jörg Vogel (1995): *Restarting automata.* In Horst Reichel editor: *FCT'95, Proc., LNCS* 965, Springer, Berlin, pp. 283-292, doi:10.1007/3-540-60249-6_60.

[6] Petr Jančar, František Mráz, Martin Plátek, Jörg Vogel (1997): *On restarting automata with rewriting.* In Gheorghe Păun, Arto Salomaa editors: *New Trends in Formal Languages, LNCS* 1218, Springer, Berlin, pp. 119-136, doi:10.1007/3-540-62844-4_8.

[7] Petr Jančar, František Mráz, Martin Plátek & Jörg Vogel (1999): *On monotonic automata with a restart operation. J. Autom. Lang. Comb.* 4, pp. 287-311.

[8] Aravind K. Joshi, K. Vijay-Shanker & David Weir (1991): *The convergence of mildly context-sensitive grammatical formalisms.* In Peter Sells, Stuart Shieber, Tom Wasow editors: *Foundational Issues in Natural Language Processing*, MIT Press, Cambridge MA, pp. 31-82.

[9] Aravind K. Joshi & Yves Schabes (1997): *Tree-adjoining grammars.* In Grzegorz Rozenberg, Arto Salomaa editors: *Handbook of Formal Languages*, vol. 3, Springer, Berlin, New York, pp. 69-123.

[10] Tomasz Jurdziński, František Mráz, Friedrich Otto & Martin Plátek (2005): *Monotone deterministic RL-automata don't need auxiliary symbols.* In Clelia De Felice, Antonio Restivo editors: *DLT 2005, Proc., LNCS* 3572, Springer, Berlin, pp. 284-295, doi:10.1007/11505877_25.

[11] Tomasz Jurdziński & Friedrich Otto (2007): *Shrinking restarting automata. Intern. J. Foundations Computer Science* 18, pp. 361-385, doi:10.1142/S0129054107004723.

[12] Tomasz Jurdziński, Friedrich Otto, František Mráz & Martin Plátek (2005): *Deterministic two-way restarting automata and Marcus contextual grammars. Fundam. Inform.* 64, pp. 217-228.

[13] Markéta Lopatková, Martin Plátek & Vladislav Kuboň (2005): *Modeling syntax of free word-order languages: Dependency analysis by reduction.* In Václav Matoušek, Pavel Mautner, Tomáš Pavelka editors: *TSD 2005, Proc., LNCS* 3658, Springer, Berlin, pp. 140-147, doi:10.1007/11551874_18.

[14] Markéta Lopatková, Martin Plátek & Petr Sgall (2007): *Towards a formal model for functional generative description: Analysis by reduction and restarting automata. Prague Bull. Math. Linguistics* 87, pp. 7-26.

[15] Solomon Marcus (1969): *Contextual grammars*, *Revue Roum. Math. Pures Appl.* 14, pp. 1473-1482.

[16] František Mráz (2001): *Lookahead hierarchies of restarting automata. J. Autom. Lang. Comb.* 6, pp. 493-506.

[17] František Mráz, Friedrich Otto & Martin Plátek (2009): *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, pp. 3530-3538, doi:10.1016/j.tcs.2009.03.017.

[18] Friedrich Otto (2003): *Restarting automata and their relations to the Chomsky hierarchy.* In Zoltán Ésik, Zoltan Fülöp editors: *DLT 2003, Proc., LNCS* 2710, Springer, Berlin, pp. 55-74, doi:10.1007/3-540-45007-6_5.

[19] Friedrich Otto (2006): *Restarting automata.* In Zoltán Ésik, Carlos Martín-Vide, Victor Mitrana editors: *Recent Advances in Formal Languages and Applications*, Studies in Computational Intelligence 25, Springer, Berlin, pp. 269-303, doi:10.1007/978-3-540-33461-3_11.

[20] Friedrich Otto (2009): *Left-to-right regular languages and two-way restarting automata. RAIRO-Theor. Inf. Appl.* 43, pp. 653-665. doi:10.1051/ita/2009013.

[21] Martin Plátek (2001): *Two-way restarting automata and j-monotonicity.* In Leszek Pacholski, Peter Ružička editors: *SOFSEM'01, Proc., LNCS* 2234, Springer, Berlin, pp. 316-325, doi:10.1007/3-540-45627-9_28.

[22] Martin Plátek, Friedrich Otto & František Mráz (2016): On h-lexicalized restarting list automata. Technical report. Available at `http://www.theory.informatik.uni-kassel.de/projekte/RL2016v6.4.pdf`.

[23] Grzegorz Rozenberg & Arto Salomaa, editors (1997): *Handbook of Formal Languages*, *Vol. 2: Linear Modeling: Background and Application*, Springer, Berlin, doi:10.1007/978-3-662-07675-0.

[24] Natalie Schluter (2015): *Restarting automata with auxiliary symbols restricted by lookahead size. Intern. J. Comput. Math.* 92, pp. 908-938, doi:10.1080/00207160.2014.926005.