RAIRO-Theor. Inf. Appl. 43 (2009) 653–665
DOI: 10.1051/ita/2009013

Available online at:
www.rairo-ita.org

LEFT-TO-RIGHT REGULAR LANGUAGES AND TWO-WAY RESTARTING AUTOMATA

Friedrich Otto[1]

Abstract. It is shown that the class of left-to-right regular languages coincides with the class of languages that are accepted by monotone deterministic RL-automata, in this way establishing a close correspondence between a classical parsing algorithm and a certain restricted type of analysis by reduction.

Mathematics Subject Classification. 68Q45.

1. Introduction

The left-to-right regular grammars were introduced by Čulik and Cohen [4] as a generalization of LR(k) grammars. As it turned out the class LRR of left-to-right regular languages, that is, the class of languages that are generated by left-to-right regular grammars, properly extends the class DCFL of deterministic context-free languages, while it is strictly included in the class UCFL of unambiguous context-free languages. Left-to-right regular languages are of interest as they can be parsed deterministically in linear time by a two-scan algorithm. However, this algorithm is considered to be impractical for real applications for two reasons (see [1]): first the finite-state preprocessor, which realizes the first phase of the two-scan algorithm, is assumed to be given together with the grammar. No method is given for constructing it – in fact, the problem of deciding whether such a preprocessor exists is undecidable [6]. Secondly, the right-to-left scan of the first phase seems to be difficult for large inputs. Therefore, various practical implementations of parsers have been proposed that extend LR(0) parsers by techniques to compute look-ahead information, even for arbitrary look-aheads [2,5,25]. However, in contrast to the linear time bound for the two-scan algorithm, this approach yields

Keywords and phrases. Left-to-right regular grammar, two-way restarting automaton, monotonicity.

[1] Fachbereich Elektrotechnik/Informatik, Universität Kassel, 34109 Kassel, Germany; otto@theory.informatik.uni-kassel.de

only a quadratic time bound in the worst case, even though this "rarely (if ever) occurs in practical situations" (see [2]). Further, these implementations can only handle certain subclasses of the left-to-right regular languages.

The restarting automaton, on the other hand, was defined by Jančar *et al.* [7] to model the *analysis by reduction*, a technique used in linguistics to analyze sentences of natural languages with free word-order. A restarting automaton has a finite-state control and a read/write window of a fixed size that works on a flexible tape delimited by sentinels. It works in cycles. In each cycle it starts in its initial state with its read/write window in the leftmost position, scanning the left sentinel and the prefix of the current tape content. It can move its window on the tape one cell at a time by performing move-right and move-left steps until, at some place, it decides to rewrite the part of the tape content in its window by a shorter string, in this way also shortening the tape. After that it may perform further move operations until it eventually executes a restart. Such a restart places the window back over the left hand of the tape and resets the finite-state control to the initial state. Then the next cycle starts on the now shortened tape. The automaton halts by either performing an explicit accept instruction, or by entering a configuration for which its control unit has no further instructions, in which case it rejects.

In fact, many different models of restarting automata have been developed (see, *e.g.*, [23] for an overview). For example, monotone deterministic R-automata accept exactly the deterministic context-free languages, while monotone RLWW-automata characterize the class CFL of context-free languages [8]. Further, deterministic RWW-automata accept the *Church-Rosser languages* (CRL) of [15,19], which are of interest as they have linear-time decidable membership problems. Observe, however, that in general a deterministic restarting automaton (of any form) may execute up to n cycles on an input of length n, which yields a quadratic time-bound. The class CRL properly includes the deterministic context-free languages, but it is incomparable under inclusion to the class of unambiguous context-free languages [9]. The intersection of the classes CRL and CFL has been investigated only recently [13], and it has been shown that the problem of deciding whether a given Church-Rosser or context-free language belongs to this intersection is complete for the second level of the arithmetic hierarchy.

In [11] it is shown that monotone deterministic two-way restarting automata (that is, det-mon-RL-automata) define a language class that strictly includes the class DCFL, and that is rather robust in that it is also characterized by various other types of monotone deterministic two-way restarting automata. Actually, this class of languages is a proper subclass of CRL [23].

Here we show that the class of languages accepted by det-mon-RL-automata coincides with the class LRR of left-to-right regular languages. By symmetry it follows that the class of languages accepted by left-monotone deterministic RL-automata coincides with the class RLR of right-to-left regular languages of Čulik and Cohen [4]. In this way we establish a close correspondence between a certain classical parsing algorithm and a restricted type of analysis by reduction. In particular, we see that monotone deterministic RL-automata yield recognizers for left-to-right regular languages that are asymptotically of the same worst case complexity as the parsers mentioned above. In addition, our results imply immediately that the class LRR as well as the class RLR are proper subsets of the intersection of the class CRL of Church-Rosser languages with the class UCFL of unambiguous context-free languages.

This paper is structured as follows. In Section 2 we describe the types of restarting automata we will be dealing with in short. In addition, we provide a new, simplified proof for a result from [11] stating that a language that is accepted by a left-monotone deterministic RL-automaton can be transformed by an injective deterministic transduction into the reversal of a deterministic context-free language. In Section 3 we restate the definition of the left-to-right regular grammars and languages and some of their main properties from [4]. Then we establish our main result in Section 4. In its proof the result on left-monotone deterministic RL-automata mentioned above plays a crucial role. The paper closes with a number of open problems related to our result.

2. Two-way restarting automata

Here we describe in short the type of restarting automaton we will be dealing with. More details on restarting automata in general can be found in [22,23].

A *two-way restarting automaton*, RLWW-automaton for short, is a nondeterministic machine $M = (Q, \Sigma, \Gamma, ¢, \$, q_0, k, \delta)$ with a finite set of internal states Q, a flexible tape, and a read/write window of a fixed size $k \geq 1$. The work space is limited by the left sentinel ¢ and the right sentinel \$, which cannot be removed from the tape. In addition to the input alphabet Σ, the tape alphabet Γ of M may contain a finite number of so-called auxiliary symbols. The behaviour of M is described by the transition relation δ that associates a finite set of transition steps to each pair (q, u) consisting of a state q and a possible content u of the read/write window. There are five types of transition steps:

1. A *move-right step* (MVR) causes M to shift the read/write window one position to the right and to change the state.
2. A *move-left step* (MVL) causes M to shift the read/write window one position to the left and to change the state.
3. A *rewrite step* causes M to replace the content u of the read/write window by a shorter string v, thereby shortening the tape, and to change the state.
4. A *restart step* causes M to place its read/write window over 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* causes M to halt and accept.

If $\delta(q, u) = \emptyset$ for some pair (q, u), then M necessarily halts, and we say that M *rejects* in this situation. Further, it is required that, when ignoring move operations, rewrite and restart steps alternate in each computation of M, with a rewrite step coming first. In general, the automaton M is *nondeterministic*, that is, there can be two or more instructions with the same left-hand side (q, u). If that is not the case, the automaton is *deterministic.*

A *configuration* of M is a string $\alpha q \beta$ where q is a state, and either $\alpha = \lambda$ (the empty string) and $\beta \in \{¢\}\cdot\Gamma^*\cdot\{\$\}$ or $\alpha \in \{¢\}\cdot\Gamma^*$ and $\beta \in \Gamma^*\cdot\{\$\}$; here q represents the current state, $\alpha\beta$ is the current content of the tape, and it is understood that the window contains the first k symbols of β or all of β when $|\beta| \leq k$. A *restarting configuration* is of the form $q_0¢w\$$, where q_0 is the initial state and $w \in \Gamma^*$; if $w \in \Sigma^*$ is an input word, then $q_0¢w\$$ is called an *initial configuration.*

We observe that any finite computation of a two-way restarting automaton M consists of certain phases. A phase, called a *cycle*, starts in a restarting configuration. The window is moved along the tape by executing MVR and MVL operations and a single rewrite operation until a restart operation is performed and thus a new restarting configuration is reached. The part after the last restart operation is called a *tail.* By $u \vdash_M^c v$ we denote a cycle of M that transforms the restarting configuration $q_0¢u\$$ into the restarting configuration $q_0¢v\$$. By $\vdash_M^{c^*}$ we denote the reflexive and transitive closure of $\vdash_M^c$.

A *word* $w \in \Sigma^*$ *is accepted by* M*,* if there is a computation which, starting with the initial configuration $q_0¢w\$$, finishes by executing an accept instruction. By $L(M)$ we denote the language consisting of all words accepted by M; this is the *language recognized* (*accepted*) by M.

We are also interested in various restricted types of restarting automata. They are obtained by combining two types of restrictions:

(a) Restrictions on the movement of the read/write window (expressed by the first part of the class name): RL- denotes no restriction, RR- means that no MVL operations are available, R- means that no MVL operations are available and that each rewrite step is immediately followed by a restart.
(b) Restrictions on the rewrite instructions (expressed by the second part of the class name): -WW denotes no restriction, -W means that no auxiliary symbols are available (that is, $\Gamma = \Sigma$), -λ means that no auxiliary symbols are available and that each rewrite step is simply a deletion (that is, if $(q', v) \in \delta(q, u)$ is a rewrite instruction of M, then v is obtained from u by deleting some symbols).

Also some generalizations of restarting automata have been introduced and studied. Here we are interested in the so-called *shrinking* restarting automaton. A *shrinking* RLWW-automaton $M = (Q, \Sigma, \Gamma, ¢, \$, q_0, k, \delta)$ has the same components as an RLWW-automaton with the exception that it is not required that $|v| < |u|$ holds for each rewrite instruction $(q', v) \in \delta(q, u)$ of M. Instead there must exist a *weight function* $\varphi : \Gamma \cup \{¢, \$\} \to \mathbb{N}_+$ such that, for each rewrite step $(q', v) \in \delta(q, u)$, $\varphi(u) > \varphi(v)$ holds. Here φ is extended to a morphism $\varphi : (\Gamma \cup \{¢, \$\})^* \to \mathbb{N}$ by taking $\varphi(\lambda) := 0$ and $\varphi(wa) := \varphi(w) + \varphi(a)$ for all $w \in (\Gamma \cup \{¢, \$\})^*$ and all $a \in \Gamma \cup \{¢, \$\}$. We use the prefix s- to denote types of shrinking restarting automata.

Concerning the expressive power of deterministic restarting automata the following results have been obtained. Here CRL denotes the class of *Church-Rosser* *languages* of [15,19], which can be seen as the deterministic variants of the *growing context-sensitive languages* [21].

Theorem 2.1 [10,20]**.**
$\mathsf{CRL} = \mathcal{L}(\mathsf{det\text{-}sRRWW}) = \mathcal{L}(\mathsf{det\text{-}RRWW}) \subsetneq \mathcal{L}(\mathsf{det\text{-}RLWW})$.

Here the strictness of the inclusion of CRL in $\mathcal{L}(\mathsf{det\text{-}RLWW})$ follows from the observation that there exists a deterministic RLWW-automaton for the copy language $L_{\text{copy}} := \{\, w\#w \mid w \in \{a,b\}^* \,\}$, which is not even growing context-sensitive [3,14].

Finally, we need two monotonicity conditions for restarting automata. Each cycle C of a restarting automaton M contains a unique configuration $\alpha q\beta$ in which a rewrite step is applied. Then $|\beta|$ is called the *right distance* of C, denoted as $D_r(C)$, and $|\alpha|$ is called the *left distance* of C, denoted as $D_l(C)$.

A sequence of cycles $S = (C_1, C_2, \ldots, C_n)$ of M is called *monotone* (or *right-monotone*) if $D_r(C_1) \geq D_r(C_2) \geq \cdots \geq D_r(C_n)$, and it is called *left-monotone* if $D_l(C_1) \geq D_l(C_2) \geq \cdots \geq D_l(C_n)$. A computation of M is called *monotone* (*left-monotone*) if the corresponding sequence of cycles is monotone (left-monotone). Finally, M itself is called *monotone* (*left-monotone*) if all its computations that start from an initial configuration are monotone (left-monotone). We use the prefix mon- (left-mon-) to denote monotone (left-monotone) types of restarting automata.

The following results have been obtained on monotone and left-monotone deterministic two-way restarting automata. Here w^R denotes the *reversal* (or *mirror image*) of a word w, L^R denotes the *reversal* $\{\, w^R \mid w \in L \,\}$ of a language L, and for a language class $\mathcal{L}$, $\mathcal{L}^R = \{\, L^R \mid L \in \mathcal{L} \,\}$.

Theorem 2.2 [11,12]**.**

(a) *For all* $\mathsf{X} \in \{\lambda, \mathsf{W}, \mathsf{WW}\}, \mathcal{L}(\mathsf{det\text{-}left\text{-}mon\text{-}(s)RLX}) = (\mathcal{L}(\mathsf{det\text{-}mon\text{-}(s)RLX}))^R$.

(b) $\mathcal{L}(\mathsf{det\text{-}mon\text{-}RL}) = \mathcal{L}(\mathsf{det\text{-}mon\text{-}RLWW}) = \mathcal{L}(\mathsf{det\text{-}mon\text{-}sRLWW})$.

(c) $\mathcal{L}(\mathsf{det\text{-}left\text{-}mon\text{-}RL}) = \mathcal{L}(\mathsf{det\text{-}left\text{-}mon\text{-}RLWW}) = \mathcal{L}(\mathsf{det\text{-}left\text{-}mon\text{-}sRLWW})$
$= \mathcal{L}(\mathsf{det\text{-}left\text{-}mon\text{-}RWW}) = \mathcal{L}(\mathsf{det\text{-}left\text{-}mon\text{-}RRWW})$.

From the above results we easily obtain the following proper inclusions.

Corollary 2.3 [23]**.** $\mathcal{L}(\mathsf{det\text{-}mon\text{-}RL}) \subsetneq \mathsf{CRL}$ *and* $\mathcal{L}(\mathsf{det\text{-}left\text{-}mon\text{-}RL}) \subsetneq \mathsf{CRL}$.

Proof. As $\mathcal{L}(\mathsf{det\text{-}left\text{-}mon\text{-}RL}) = \mathcal{L}(\mathsf{det\text{-}left\text{-}mon\text{-}RRWW}) \subseteq \mathcal{L}(\mathsf{det\text{-}RRWW}) = \mathsf{CRL}$, we have the second inclusion. Further, as the class CRL is closed under reversal, and as $\mathcal{L}(\mathsf{det\text{-}mon\text{-}RL}) = (\mathcal{L}(\mathsf{det\text{-}left\text{-}mon\text{-}RL}))^R$, we see that also the first inclusion holds.

To prove the strictness of these inclusions we consider the language

$$L := \{\, a^n b^n \mid n \geq 0 \,\} \cup \{\, a^n b^m \mid m > 2n \geq 0 \,\}.$$

In [23] it is shown that L is a Church-Rosser language, while in [8] it is shown that this language is not accepted by any RR-automaton. As nondeterministic RR-automata and RL-automata accept exactly the same languages (see, *e.g.*, [24]), and as $\mathcal{L}(\mathsf{det\text{-}mon\text{-}RL}) \cup \mathcal{L}(\mathsf{det\text{-}left\text{-}mon\text{-}RL}) \subseteq \mathcal{L}(\mathsf{RL})$, it follows that L is not accepted by any monotone deterministic RL-automaton nor by any left-monotone deterministic RL-automaton. □

Below we will need the details of a deterministic transducer that maps the language accepted by a left-monotone deterministic RL-automaton onto the reversal of a deterministic context-free language. Therefore, we provide a detailed description of this construction, which is actually simpler than a similar construction given in [11].

Let $L \in \mathcal{L}(\mathsf{det\text{-}left\text{-}mon\text{-}RL})$. Then by Theorem 2.2 (c), there exists a left-monotone deterministic RWW-automaton $M = (Q, \Sigma, \Gamma, ¢, \$, q_0, k, \delta)$ such that $L(M) = L$. This RWW-automaton can be described more succinctly through a finite sequence of *rewriting meta-instructions* $I_1, \dots, I_n$ and a single *accepting meta-instruction* I_0 (see [23]). Here each I_i is of the form $I_i = (E_i, u_i \to v_i)$, where, for each $i = 1, \dots, n$, E_i is a regular language, and u_i and v_i are words satisfying $k \geq |u_i| > |v_i|$. By adjusting the size of the read/write window of M accordingly, we may even assume that $|v_i| \geq 2$, implying that v_i is neither empty nor that it is just a sentinel. Further, $I_0 = (E_0, \mathsf{Accept})$, where E_0 is a regular language. In a restarting configuration $q_0¢w\$$, the meta-instruction I_i is applicable, if w admits a factorization of the form $w = w_1 u_i w_2$ such that $¢w_1 \in E_i$ holds. In this situation $q_0¢w\$$ yields the restarting configuration $q_0¢w_1 v_i w_2\$$ in one cycle, that is, $w \vdash_M^c w_1 v_i w_2$ holds. If $¢w\$ \in E_0$, then I_0 is applicable to the restarting configuration $q_0¢w\$$, and M accepts in a tail computation.

As M is deterministic, we can assume that to any given restarting configuration at most a single meta-instruction is applicable. In fact, we may assume that the regular languages E_i, $1 \leq i \leq n$, are chosen in such a way that each word $w \in \Gamma^*$ admits at most one factorization of the form $w = w_1 u w_2$ such that there exists an index i with $¢w_1 \in E_i$ and $u = u_i$.

Now we associate a deterministic transducer to the RWW-automaton M as follows. For $i = 1, \dots, n$, let $A_i = (Q_i, \Gamma \cup \{¢, \$\}, \delta_i, q_0^{(i)}, F_i)$ be a complete deterministic finite-state acceptor (DFA) for the language $¢ \cdot E_i$. Then we can construct the product automaton $P := (Q_P, \Gamma \cup \{¢, \$\}, \delta_P, q_0^{(P)}, Q_P)$ of the DFAs $A_1, \dots, A_n$, that is, $Q_P := Q_1 \times \dots \times Q_n$, $q_0^{(P)} := (q_0^{(1)}, \dots, q_0^{(n)})$, and $\delta_P((q_{i_1}^{(1)}, \dots, q_{i_n}^{(n)}), a) := (\delta_1(q_{i_1}^{(1)}, a) \dots, \delta_n(q_{i_n}^{(n)}, a))$ for all states $q_{i_j}^{(j)} \in Q_j$, $1 \leq j \leq n$, and all $a \in \Gamma \cup \{¢, \$\}$. The deterministic transducer G is defined as $G := (Q_G, \Gamma, \Delta, \delta_G, \lambda_G, q_0^{(G)}, Q_G)$, where $Q_G := Q_P \cup \{q_0^{(G)}\}$, the output alphabet is $\Delta := \Gamma \times Q_P$, the transition function δ_G is defined through

$$\begin{array}{rcll} \delta_G(q_0^{(G)}, a) & := & \delta_P(q_0^{(P)}, ¢a) & \text{for all } a \in \Gamma, \\ \delta_G(q, a) & := & \delta_P(q, a) & \text{for all } q \in Q_P \text{ and all } a \in \Gamma, \end{array}$$

and the output function λ_G is given through

$$\begin{array}{rcll} \lambda_G(q_0^{(G)}, a) & := & (a, \delta_P(q_0^{(P)}, ¢)) & \text{for all } a \in \Gamma, \\ \lambda_G(q, a) & := & (a, q) & \text{for all } q \in Q_P \text{ and all } a \in \Gamma. \end{array}$$

Let $a_1, a_2, \ldots, a_m \in \Gamma$, and let $p_i := \delta_P(q_0^{(P)}, ¢a_1 \ldots a_i)$ for all $i = 0, 1 \ldots, m$. Then $\delta_G(q_0^{(G)}, a_1 a_2 \ldots a_i) = p_i$ for all $i = 1, \ldots, m$, and

$$G(a_1 a_2 \ldots a_m) = (a_1, p_0)(a_2, p_1) \ldots (a_m, p_{m-1}).$$

Now the main property of this construction is given by the following lemma.

Lemma 2.4. *The language $G(L)$ is the reversal of a deterministic context-free language, that is, $(G(L))^R \in \mathsf{DCFL}$.*

Proof. As the class DCFL of deterministic context-free languages coincides with the class of languages that are accepted by monotone deterministic RWW-automata [8], we present a monotone deterministic RWW-automaton M' with input alphabet $\Gamma \times Q_P$ and tape alphabet $(\Gamma \times Q_P) \cup \Gamma$ for the language $L' := (G(L))^R$. For describing M', we need the morphism $\rho : ((\Gamma \times Q_P) \cup \Gamma)^* \to \Gamma^*$, which is defined by $(a, q) \mapsto a$ and $a \mapsto a$ for all $a \in \Gamma$ and $q \in Q_P$.

Let $w \in \Sigma^*$ be an input word. Given the word $w' := (G(w))^R$ as input, M' will simulate the computation of M on input w. Scanning w' from left to right, M' tries to find the rewrite step that M would apply to w. At the same time it simulates a DFA A_0 for the language E_0^R in its finite-state control. If $¢w\$ \in E_0$, that is, if M accepts w in a tail computation, then M' will recognize this through the simulation of A_0, and accordingly it will accept w' in a tail computation, too. Assume now that $w \vdash_M^c z$ holds, where meta-instruction I_i is applied for some $1 \leq i \leq n$. Then $w = w_1 u_i w_2$ such that $¢w_1 \in E_i$, and $z = w_1 v_i w_2$. Then $G(w) = G(w_1 u_i w_2) = G(w_1) U_i W_2$, where U_i and W_2 denote the output that G produces for the infix u_i and the suffix w_2, respectively, while processing $w_1 u_i w_2$. Given $(G(w))^R = W_2^R U_i^R (G(w_1))^R$ as input, M' will detect the image U_i^R of the factor u_i to be rewritten from the information provided by the state of P encoded by the transducer G in the first letter of the suffix $(G(w_1))^R$. Then M' will replace the factor U_i^R by the string v_i^R. Actually, in order to ensure that this approach works, M' needs a read/write window of size $k + 1$.

Through the above rewrite process the tape contents $¢W_2^R v_i^R (G(w_1))^R\$$ is obtained, which is not anymore of the form $¢(G(x))^R\$$ for any $x \in \Gamma^*$, as v_i is non-empty by our assumption above. However, the RWW-automaton M is left-monotone, that is, the next rewrite step it executes on $¢w_1 v_i w_2\$$ has left distance at most $|w_1| + 1$, that is, it replaces a factor that starts at the same place as the factor v_i or to the left of that place. Thus, the necessary information about this rewrite step is still encoded in the word $G(w_1)$. Accordingly, when M' starts the next cycle with tape contents $¢W_2^R v_i^R (G(w_1))^R\$$, then it simply has to perform move-right steps until it discovers the suffix $(G(w_1))^R$, from which it can then extract the necessary information on the next rewrite step of M that it has to simulate. If another rewrite step is detected, then M' simulates it as above; otherwise, it scans the tape completely from left to right, thereby simulating the DFA A_0 on $¢ \cdot \rho(W_2^R v_i^R (G(w_1))^R) \cdot \$$. In case the A_0-computation being simulated is accepting, M' accepts as well.

It follows that M' is a deterministic RWW-automaton, and that it is monotone. For all words $w \in \Sigma^*$, we see that M' accepts on input $(G(w))^R$ if and only if $w \in L$, that is, $L(M') \cap (G(\Sigma^*))^R = L'$. Thus, $L(M')$ is a deterministic context-free language that has the property that $L(M') \cap (G(\Sigma^*))^R = L'$. As the class DCFL is closed under intersection with regular languages, it follows that the language L' is in DCFL, too. □

3. Left-to-right regular languages

Here we restate the relevant definitions and results on left-to-right regular grammars and languages from [4].

Let $G = (N, T, P, S)$ be a context-free grammar, where N is the set of nonterminals, T is the set of terminals, $S \in N$ is the start symbol, and $P \subsetneq N \times (N \cup T)^*$ is a finite set of productions. By $\Rightarrow^*$ we denote the derivation relation induced by G, and $L(G) := \{ w \in T^* \mid S \Rightarrow^* w \}$ is the language generated by G. Without loss of generality we may assume that S does not occur on the right-hand side of any production. By $\overset{R}{\Rightarrow}{}^*$ we denote the rightmost derivation relation induced by G. A *right sentential form* is any string $\alpha \in (N \cup T)^*$ such that $S \overset{R}{\Rightarrow}{}^* \alpha$ holds.

Definition 3.1. Let $\pi = \{E_1, E_2, \ldots, E_n\}$ be a partition of T^* into n disjoint regular sets. Then $x \equiv y \bmod(\pi)$ denotes the fact that x and y belong to the same subset E_i.

A context-free grammar $G = (N, T, P, S)$ is called LR(π), if, for any right-most derivations of the form

$$S \overset{R}{\Rightarrow}{}^* \alpha_1 A_1 y_1 \overset{R}{\Rightarrow} \alpha_1 \gamma y_1 \quad \text{and} \quad S \overset{R}{\Rightarrow}{}^* \alpha_2 A_2 y_3 \overset{R}{\Rightarrow} \alpha_1 \gamma y_2,$$

where $A_1, A_2 \in N$, $\alpha_1, \alpha_2, \gamma \in (N \cup T)^*$, and $y_1, y_2, y_3 \in T^*$, $y_1 \equiv y_2 \bmod(\pi)$ implies that $A_1 = A_2$, $\alpha_1 = \alpha_2$ and $y_2 = y_3$.

A context-free grammar is called *left-to-right regular* (LR-*regular*) if it is LR(π) for some regular partition π of T^*. A language L is *left-to-right regular* if $L = L(G)$ for some LR-regular grammar G. By LRR we denote the class of left-to-right regular languages.

Concerning the language class LRR the following results are known.

Theorem 3.2 [4].

(1) DCFL $\subsetneq$ LRR $\subsetneq$ UCFL.
(2) *The class* LRR *is incomparable under inclusion to the class* DCFL^R.
(3) *The class* LRR *is an* abstract family of deterministic languages*, that is, it is closed under marked union, marked Kleene star, and inverse marked* GSM *mappings.*
(4) *It is decidable whether a given left-to-right regular language is regular.*

By RLR we denote the class of *right-to-left regular languages*. These are just the mirror images of the left-to-right regular languages, that is, RLR $=$ LRR^R.

4. Main result

Here we will establish our main result.

Theorem 4.1. $\mathsf{LRR} = \mathcal{L}(\mathsf{det\text{-}mon\text{-}RL})$.

By symmetry this yields the following result.

Corollary 4.2. $\mathsf{RLR} = \mathcal{L}(\mathsf{det\text{-}left\text{-}mon\text{-}RL})$.

Proof. Theorem 4.1 yields $\mathsf{RLR} = \mathsf{LRR}^R = (\mathcal{L}(\mathsf{det\text{-}mon\text{-}RL}))^R$, which implies $\mathsf{RLR} = \mathcal{L}(\mathsf{det\text{-}left\text{-}mon\text{-}RL})$ by Theorem 2.2 (a). □

It remains to prove Theorem 4.1. From the results on monotone and left-monotone deterministic RL-automata mentioned above it follows quite easily that each left-to-right regular language is accepted by a monotone deterministic RL-automaton.

Lemma 4.3. $\mathsf{LRR} \subseteq \mathcal{L}(\mathsf{det\text{-}mon\text{-}RL})$.

Proof. Let L be generated by an $\mathsf{LR}(\pi)$ grammar. Then according to the proof of Theorem 2.1 of [4], there exists a particular deterministic transducer G such that the language $L_0 := (G(L^R))^R$ is deterministic context-free. Hence, there exists a monotone deterministic R-automaton M_0 such that $L(M_0) = L_0$ [8]. Now a deterministic shrinking RLWW-automaton M can recognize the language L by proceeding as follows:

- M first moves its read/write window all the way across its tape, checking that no auxiliary symbols are on the tape. In the affirmative it knows that it has started from an initial configuration, while otherwise it realizes that it has already executed one or more cycles.
- If M started from an initial configuration, then, beginning at the right end of the tape, it simulates the deterministic transducer G (but without writing down the output generated) moving left until it reaches the left-most letters of the tape content. These are then replaced by their images under G, and M restarts. Obviously a weight function can be chosen in such a way that this process is indeed weight reducing.
- If M started from a non-initial restarting configuration, then, beginning at the right end of the tape, it simulates the deterministic transducer G (but without writing down the output generated) moving left until it reaches the position where the last rewrite step was performed. This place is easily detected through the presence of auxiliary symbols. If at that place a rewrite step of M_0 is applicable, then M executes this step and restarts; otherwise it replaces the next few symbols to the right of that position by their images under G and restarts.

As M_0 is monotone, these two processes (the simulation of G and the simulation of the subsequent computation of M_0) can indeed be interleaved in the way described above. Thus, $L(M) = L$ holds. In addition, it follows immediately that the resulting deterministic shrinking RLWW-automaton is monotone. Hence,

$L \in \mathcal{L}(\mathsf{det\text{-}mon\text{-}RL})$ by Theorem 2.2 (b), which implies that $\mathsf{LRR} \subseteq \mathcal{L}(\mathsf{det\text{-}mon\text{-}RL})$ holds. $\square$

The proof of the converse inclusion is somewhat more involved.

Lemma 4.4. $\mathcal{L}(\mathsf{det\text{-}mon\text{-}RL}) \subseteq \mathsf{LRR}$.

Proof. Assume that the language $L \subseteq \Sigma^*$ is accepted by a monotone deterministic RL-automaton. Then its mirror image L^R is accepted by a left-monotone deterministic RL-automaton (Thm. 2.2 (a)). Thus, by Theorem 2.2 (c) it is also accepted by a left-monotone deterministic RWW-automaton $M = (Q, \Sigma, \Gamma, ¢, \$, q_0, k, \delta)$. As described in detail in the paragraphs preceding Lemma 2.4 we can construct a particular deterministic transducer G from M such that the language $L' := (G(L^R))^R$ is deterministic context-free.

For each state q of the product automaton P considered in the construction of G, let $\pi_q := \{\, w \in \Sigma^* \mid \delta_P(q_0^{(P)}, ¢w^R) = q \,\}$. Then $\pi := (\pi_q)_{q \in Q_P}$ is a finite regular partition of Σ^*, as all component automata $A_1, \ldots, A_n$ of P are complete DFAs. We say that two words $x, y \in \Sigma^*$ are congruent mod π, denoted as $x \equiv y \bmod(\pi)$, if x and y belong to the same set π_q. In fact, this relation is a left congruence on Σ^*. If $x, y, z \in \Sigma^*$ are any words satisfying $x \equiv y \bmod(\pi)$, then $\delta_P(q_0^{(P)}, ¢x^R) = \delta_P(q_0^{(P)}, ¢y^R)$. Hence,

$$\delta_P(q_0^{(P)}, ¢(zx)^R) = \delta_P(q_0^{(P)}, ¢x^R z^R) = \delta_P(q_0^{(P)}, ¢y^R z^R) = \delta_P(q_0^{(P)}, ¢(zy)^R),$$

which shows that $zx \equiv zy \bmod(\pi)$ holds.

Let $w = a_m \ldots a_2 a_1$, where $a_1, \ldots, a_m \in \Sigma$. Further, let $p_0 := \delta_P(q_0^{(P)}, ¢)$, and let $p_i := \delta_P(q_0^{(P)}, ¢a_1 \ldots a_i)$ $(1 \leq i \leq m)$, that is, p_i is the index of the subset of π that contains the suffix $a_i \ldots a_1$ of w. Then it follows immediately from the construction of G that $G(w^R) = (a_1, p_0)(a_2, p_1) \ldots (a_m, p_{m-1})$ holds. We now define a mapping $f_\pi : \Sigma^* \to ¢ \cdot (\Sigma \times Q_P)^* \cdot \$$ by taking

$$f_\pi(a_m \ldots a_1) := ¢(a_m, p_{m-1}) \ldots (a_1, p_0)\$,$$

that is, $f_\pi(w) = ¢ \cdot (G(w^R))^R \cdot \$$. Obviously, f_π is an injective mapping. Further, let $h_\pi : ((\Sigma \times Q_P) \cup \{¢, \$\})^* \to \Sigma^*$ be the morphism that is defined through $(a, q) \mapsto a$ $(a \in \Sigma,\ q \in Q_P)$, $¢ \mapsto \lambda$, and $\$ \mapsto \lambda$. If R_π denotes the range $R_\pi := f_\pi(\Sigma^*)$ of f_π, then f_π^{-1} is simply the restriction of h_π to R_π. Thus, h_π is one-to-one on R_π.

As the language $(G(L^R))^R$ is deterministic context-free, the language $f_\pi(L) = ¢ \cdot (G(L^R))^R \cdot \$$ is generated by an $\mathsf{LR}(0)$ grammar $\mathcal{G}'$. By replacing each occurrence of each terminal symbol in $\mathcal{G}'$ by its image under h_π, we obtain a grammar $\mathcal{G}$. This grammar generates the language $h_\pi(f_\pi(L)) = L$, and according to Lemma 3.2 of [4] it is an $\mathsf{LR}(\pi)$ grammar. This implies that L is a left-to-right regular language. This completes the proof of Lemma 4.4. $\square$

Based on Theorem 4.1 and Corollary 2.3 we now obtain the following proper inclusions.

Corollary 4.5. $\mathsf{LRR} \subsetneq \mathsf{CRL} \cap \mathsf{UCFL}$ *and* $\mathsf{RLR} \subsetneq \mathsf{CRL} \cap \mathsf{UCFL}$.

Proof. It is an immediate consequence of our results that both LRR and RLR are contained in CRL. Further, LRR is contained in UCFL by Theorem 3.2, and as UCFL is closed under reversal, it follows that RLR is contained in UCFL, too.

To prove that these inclusions are strict we again consider the language $L = \{ a^n b^n \mid n \geq 0 \} \cup \{ a^n b^m \mid m > 2n \geq 0 \}$ from the proof of Corollary 2.3. As shown there this language separates both $\mathsf{LRR} = \mathcal{L}(\mathsf{det\text{-}mon\text{-}RL})$ and $\mathsf{RLR} = \mathcal{L}(\mathsf{det\text{-}left\text{-}mon\text{-}RL})$ from the class CRL. It remains to show that L is an unambiguous context-free language. However, it is easily seen that L is generated by the context-free grammar $G := (N, \{a, b\}, S, P)$, where $N := \{S, A, B, C\}$ and $P := \{S \to A + B, A \to aAb + \lambda, B \to aBbb + C, C \to bC + b\}$. As this grammar is clearly unambiguous, it follows that $L \in \mathsf{CRL} \cap \mathsf{UCFL}$. □

Observe that the language classes CRL and UCFL are incomparable under inclusion. This follows from the fact that CRL contains some non-context-free languages [15] and the fact that the unambiguous context-free language $L_{\mathrm{pal}} := \{ ww^R \mid w \in \{a, b\}^* \}$ is not Church-Rosser [9].

5. Concluding remarks

In [18] the so-called *nonforgetting* restarting automaton was introduced, which, when executing a restart operation, simply changes its internal state as with any other operation, instead of resetting it to the initial state. In [16,17] various types of monotone nonforgetting restarting automata are investigated. Among others the following results are obtained, where the prefix nf- is used to denote classes of nonforgetting restarting automata.

Theorem 5.1 [16,17]**.**

$$\begin{array}{ccccc}
\mathsf{DCFL} & = & \mathcal{L}(\mathsf{det\text{-}mon\text{-}nf\text{-}RWW}) & \subsetneq & \mathcal{L}(\mathsf{det\text{-}mon\text{-}nf\text{-}RR}) \\
 & \subsetneq & \mathcal{L}(\mathsf{det\text{-}mon\text{-}nf\text{-}RRW}) & \subsetneq & \mathcal{L}(\mathsf{det\text{-}mon\text{-}nf\text{-}RRWW}) \\
 & = & \mathcal{L}(\mathsf{det\text{-}mon\text{-}nf\text{-}RLWW}) & = & \mathcal{L}(\mathsf{det\text{-}mon\text{-}RL}).
\end{array}$$

Thus, with $\mathcal{L}(\mathsf{det\text{-}mon\text{-}nf\text{-}RR})$ and $\mathcal{L}(\mathsf{det\text{-}mon\text{-}nf\text{-}RRW})$ we have two language classes that are strictly above DCFL and below the class LRR. Can these two classes also be characterized in terms of more classical types of grammars or automata?

In [26] it is shown that there is a strict hierarchy of language classes that strictly include the class LRR and that are included in UCFL. Among these are the so-called $\mathsf{BCP}(m, n)$ languages and the $\mathsf{LR}(k, \infty)$ languages (see [26] for the definitions). For example, it is shown there that the language

$$L := \{ a^n b^n c^m d^{m+k} \mid n, m, k \geq 1 \} \cup \{ a^n b^{2n} c^m d^m \mid n, m \geq 1 \}$$

is a $\mathsf{BCP}(1, 1)$ language that does not belong to the class LRR. Which of these language classes is still contained in CRL (and therewith in the intersection of CRL and UCFL)? For example, the above language L is easily seen to be accepted by a left-monotone deterministic RL-automaton, and so it is contained in RLR. Further, the language $L' := \{ a^ncb^n, a^ncb^{2n} \mid n \geq 1 \}$ is an $\mathsf{LR}(1,\infty)$ language that is not generated by any $\mathsf{FSPA}(k)$ grammar (see [26]). However, analogously to the proof that the language considered in the proof of Corollary 2.3 is Church-Rosser, it can be shown that L' is Church-Rosser, too. Thus, the intersection of CRL with UCFL contains languages that are not even generated by any $\mathsf{FSPA}(k)$ grammars.

Acknowledgements. The author wants to thank Sylvain Schmitz from the Université de Nice at Sophia Antipolis for pointing him to the literature on LR-parsers and for very helpful comments on a preliminary version of this paper. Also he thanks Hartmut Messerschmidt from Universität Kassel for fruitful discussions on the results presented here and their proofs.

References

[1] T.P. Baker, Extending lookahead for LR parsers. *J. Comput. System. Sci.* **22** (1981) 243–259.

[2] M. Bermudez and K. Schimpf, Practical arbitrary lookahead LR parsing. *J. Comput. System. Sci.* **41** (1990) 230–250.

[3] G. Buntrock and F. Otto, Growing context-sensitive languages and Church-Rosser languages. *Inform. Comput.* **141** (1998) 1–36.

[4] K. Čulik II and R. Cohen, LR-regular grammars - an extension of LR(k) grammars. *J. Comput. System. Sci.* **7** (1973) 66–96.

[5] J. Farré and J. Fortes Gálvez, A bounded graph-connect construction for LR-regular parsers, in *Proc. Compiler Construction, CC 2001. Lect. Notes Comput. Sci.* **2027** (2001) 244–258.

[6] S. Heilbrunner, A metatheorem for undecidable properties of formal languages and its application to LRR and LLR grammars and languages. *Theoret. Comput. Sci.* **23** (1983) 49–68.

[7] P. Jančar, F. Mráz, M. Plátek and J. Vogel, Restarting automata, in *Proc. FCT 1995. Lect. Notes Comput. Sci.* **965** (1995) 283–292.

[8] P. Jančar, F. Mráz, M. Plátek and J. Vogel, On monotonic automata with a restart operation. *J. Autom. Lang. Comb.* **4** (1999) 287–311.

[9] T. Jurdziński and K. Loryś, Church-Rosser languages *vs.* UCFL, in *Proc. ICALP 2002. Lect. Notes Comput. Sci.* **2380** (2002) 147–158.

[10] T. Jurdziński and F. Otto, Shrinking restarting automata. *Int. J. Found. Comput. Sci.* **18** (2007) 361–385.

[11] T. Jurdziński, F. Mráz, F. Otto and M. Plátek, Monotone deterministic RL-automata don't need auxiliary symbols, in *Proc. DLT 2005. Lect. Notes Comput. Sci.* **3572** (2005) 284–295.

[12] T. Jurdziński, F. Mráz, F. Otto and M. Plátek, Degrees of non-monotonicity for restarting automata. *Theoret. Comput. Sci.* **369** (2006) 1–34.

[13] M. Kutrib and A. Malcher, When Church-Rosser becomes context-free. *Int. J. Found. Comput. Sci.* **18** (2007) 1293–1302.

[14] C. Lautemann, One pushdown and a small tape, in *Dirk Siefkes zum 50. Geburtstag*, edited by K.W. Wagner, Technische Universität Berlin and Universität Augsburg (1988) 42–47.

[15] R. McNaughton, P. Narendran and F. Otto, Church-Rosser Thue systems and formal languages. *J. ACM* **35** (1988) 324–344.

[16] H. Messerschmidt, *CD-Systems of Restarting Automata.* Doctoral Dissertation, Fachbereich Elektrotechnik/Informatik, Universität Kassel (2008).

[17] H. Messerschmidt and F. Otto, On nonforgetting restarting automata that are deterministic and/or monotone, in *Proc. CSR 2006. Lect. Notes Comput. Sci.* **3967** (2006) 247–258.

[18] H. Messerschmidt and H. Stamer, Restart-Automaten mit mehreren Restart-Zuständen, in *Proc. Workshop "Formale Methoden in der Linguistik" und 14. Theorietag "Automaten und Formale Sprachen"*, edited by H. Bordihn, Institut für Informatik, Universität Potsdam (2004) 111–116.

[19] P. Narendran, *Church-Rosser and Related Thue Systems.* Ph.D. thesis, Rensselaer Polytechnic Institute, Troy, New York (1984).

[20] G. Niemann and F. Otto, Further results on restarting automata, in *Proc. Words, Languages and Combinatorics III*, edited by M. Ito and T. Imaoka, World Scientific, Singapore (2003) 353–369.

[21] G. Niemann and F. Otto, The Church-Rosser languages are the deterministic variants of the growing context-sensitive languages. *Inform. Comput.* **197** (2005) 1–21.

[22] F. Otto, Restarting automata and their relations to the Chomsky hierarchy, in *Proc. DLT 2003. Lect. Notes Comput. Sci.* **2710** (2003) 55–74.

[23] F. Otto, Restarting automata, in *Recent Advances in Formal Languages and Applications*, edited by Z. Ésik, C. Martin-Vide and V. Mitrana. *Studies in Computational Intelligence* **25** (2006) 269–303.

[24] M. Plátek, Two-way restarting automata and j-monotonicity, in *Proc. SOFSEM 2001. Lect. Notes Comput. Sci.* **2234** (2001) 316–325.

[25] B. Seité, A YACC extension for LRR grammar parsing. *Theoret. Comput. Sci.* **52** (1987) 91–143.

[26] T. Szymanski and J. Williams, Noncanonical extensions of bottom-up parsing techniques. *SIAM J. Comput.* **5** (1976) 231–250.

Communicated by C. Choffrut.
Received December 19, 2007. Accepted March 5, 2009.