ON THE CLASSES OF LANGUAGES ACCEPTED BY LIMITED CONTEXT RESTARTING AUTOMATA *, **, ***

FRIEDRICH OTTO¹, PETER ČERNO² AND FRANTIŠEK MRÁZ²

Abstract. In the literature various types of restarting automata have been studied that are based on contextual rewriting. A word $w$ is accepted by such an automaton if, starting from the initial configuration that corresponds to input $w$, the word $w$ is reduced to the empty word by a finite number of applications of these contextual rewritings. This approach is reminiscent of the notion of McNaughton families of languages. Here we put the aforementioned types of restarting automata into the context of McNaughton families of languages, relating the classes of languages accepted by these automata in particular to the class GCSL of growing context-sensitive languages and to the class CRL of Church–Rosser languages.

Mathematics Subject Classification. 68Q45.

Keywords and phrases. Restarting automaton, contextual rewriting, McNaughton family of languages.

* Some of the results of this paper have been announced at NCMA 2012 in Fribourg, Switzerland, August 2012. An extended abstract appeared in the proceedings of that conference [24].

** This paper was prepared while the first author was visiting at Charles University in Prague. He gratefully acknowledges the hospitality of the Faculty of Mathematics and Physics.

*** The second and the third authors were supported by the Grant Agency of the Czech Republic under the projects P103/10/0783 and P202/10/1333 and by the Grant Agency of Charles University under project 272111/A-INF/MFF.

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

² Charles University, Faculty of Mathematics and Physics, Department of Computer Science, Malostranské nám. 25, 11800 Praha 1, Czech Republic. petercerno@gmail.com; mraz@ksvi.ms.mff.cuni.cz

Article published by EDP Sciences © EDP Sciences 2014
1. Introduction

Restarting automata have been introduced to model the linguistic technique of analysis by reduction [14]. By now many different types of restarting automata have been defined and studied intensively, see for example [23]. The deterministic context-free languages, the context-free languages, the Church–Rosser languages and the growing context-sensitive languages have all been characterized by certain types of restarting automata. Further, it has been shown that also some interesting languages, like the copy language \( L_{\text{copy}} = \{ wcw \mid w \in \{a, b\}^* \} \) and the Gladkij language \( L_{\text{Glad}} = \{ wcw^Rcw \mid w \in \{a, b\}^* \} \), which are not growing context-sensitive, are accepted by certain types of restarting automata. Interestingly, a restarting automaton is not only useful for accepting a language, but in addition, a restarting automaton (“analysis by reduction”) enables error localization in rejected words/sentences (see, e.g. [15]).

Therefore, several attempts for learning restarting automata by genetic algorithms have been made [8, 13]. Unfortunately, the results are far from being applicable. Another method based on the concept of identification in the limit from positive data was proposed in [19]. This method uses positive samples of simplifications (reductions) and positive samples of so-called simple words (sentences) of the language to be learnt. It applies to learning a subclass of restarting automata called strictly locally testable restarting automata. Their definition as well as the protocol for learning them is based on the notion of strictly locally testable languages. As it turned out, the strictly locally testable restarting automata are quite expressive, as they accept a proper superclass of the growing context-sensitive languages.

In [9] an even simpler version of the restarting automaton was introduced: the so-called clearing restarting automaton. While in general a restarting automaton scans its tape content from left to right until it detects a position to which a rewrite operation applies, the rewriting done by a clearing restarting automaton only depends on the context of a fixed size around the subword to be rewritten. In fact, a clearing restarting automaton can only delete symbols. For these automata a simple learning algorithm exists, but not surprisingly, clearing restarting automata are quite limited in their expressive power. They accept all regular languages and even some languages that are not context-free, but they do not even accept all context-free languages. Accordingly, they were extended to the so-called \( \Delta \)-clearing restarting automata and the \( \Delta^* \)-clearing restarting automata that can use a marker symbol \( \Delta \) in their rewrite operations. It turned out that these types of restarting automata only accept languages that are growing context-sensitive [24], but that they can accept all context-free languages [10]. However, it is still open whether or not there is a growing context-sensitive language that is not accepted by any \( \Delta \)- or \( \Delta^* \)-clearing restarting automaton.

Finally, in [2] limited context restarting automata were defined as an extension of the clearing restarting automaton. Also these automata apply rewrite steps only based on context information, but their rewrite rules are more general. In fact, the
most general form of these automata accepts exactly the growing context-sensitive languages. In addition, in [2] a special version of a genetic algorithm is proposed to learn these automata from positive and negative samples.

As a limited context restarting automaton applies its rewrite operations based on context information, it can be interpreted as executing reductions with respect to a finite string-rewriting system. Furthermore, a word \( w \) belongs to the language \( L(M) \) that is accepted by a given limited context restarting automaton \( M \), if and only if \( M \) can reduce \( w \) to the empty word \( \lambda \), that is, if and only if \( w \) can be rewritten to \( \lambda \) by the induced string-rewriting system. This is essentially the same concept as the one that underlies the notion of McNaughton families of languages studied in [3]. Accordingly, a detailed study of the correspondence between the various types of limited context restarting automata on the one hand and the various McNaughton families of languages of [3] on the other hand is called for. Here we present such a study. For this we first repeat the definitions of the concept of a McNaughton family of languages and of the various types of limited context restarting automata. Then we relate the language classes accepted by these automata to the classes of the classical Chomsky hierarchy and to certain McNaughton families of languages, among which the class GC\(S_L \) of growing context-sensitive languages [7,11], the class CR\(L \) of Church–Rosser languages [18], and the class con-gen-mon-Mc\(N_L \) of confluent generalized monadic McNaughton languages [17] will appear prominently.

**Notation.** In the following all alphabets considered will be finite. For an alphabet \( \Sigma \), \( \Sigma^* \) is used to denote the set of all words over \( \Sigma \) including the empty word \( \lambda \). For \( w \in \Sigma^* \), \(|w|\) denotes the length of \( w \), and \( w^R \) is used to denote the reversal (or mirror image) of \( w \). Accordingly, for a language \( L \subseteq \Sigma^* \), \( L^R \) denotes the language \( \{ w^R \mid w \in L \} \). By \( \mathbb{N} \) we denote the set of non-negative integers. A weight function is a mapping \( g : \Sigma \to \mathbb{N} \) that assigns a positive weight \( g(a) \) to each letter \( a \) of \( \Sigma \). It is extended to arbitrary words by taking \( g(\lambda) = 0 \) and \( g(wa) = g(w) + g(a) \) for all words \( w \in \Sigma^* \) and all \( a \in \Sigma \). Finally, for any type \( A \) of automaton, \( L(A) \) is used to denote the class of languages accepted by automata of this type.

2. **McNaughton families of languages**

A string-rewriting system \( S \) on an alphabet \( \Sigma \) consists of (finitely many) pairs of strings from \( \Sigma^* \), called rewrite rules, which are written as \( (\ell \to r) \). By \( \text{dom}(S) \) we denote the set \( \text{dom}(S) = \{ \ell \mid \exists r \in \Sigma^* : (\ell \to r) \in S \} \) of left-hand sides of rules of \( S \). The reduction relation \( \Rightarrow_S^\ast \) on \( \Sigma^* \) that is induced by \( S \) is the reflexive and transitive closure of the single-step reduction relation

\[
\Rightarrow_S = \{ (ulv, urv) \mid (\ell \to r) \in S, u, v \in \Sigma^* \}.
\]

For a string \( u \in \Sigma^* \), if there exists a string \( v \) such that \( u \Rightarrow_S v \) holds, then \( u \) is called reducible mod \( S \). If such a string \( v \) does not exist, then \( u \) is called
irreducible mod $S$. By $\Delta^*_S(u)$ we denote the set of all descendants of $u$, that is, 
$\Delta^*_S(u) = \{ v \mid u \Rightarrow^*_S v \}$, and $\overline{\Delta^*_S(v)} = \{ u \mid u \Rightarrow^*_S v \}$ is the set of all ancestors of $v$, 
and $\mathrm{IRR}(S)$ denotes the set of all irreducible strings mod $S$. It is easily seen that 
$\mathrm{IRR}(S)$ is a regular language for each finite string-rewriting system $S$. By $\Leftrightarrow^*_S$, 
we denote the Thue congruence on $\Sigma^*$ that is induced by $S$. It is the smallest 
equivalence relation on $\Sigma^*$ containing the single-step reduction relation $\Rightarrow_S$.

Here we are interested in certain restricted types of string-rewriting systems. A 
string-rewriting system $S$ is called

- terminating, if there is no infinite sequence of reduction steps $w \Rightarrow_S w_1 \Rightarrow_S \ldots \Rightarrow_S w_k \Rightarrow_S \ldots$, 
- weight-reducing, if there exists a weight function $g$ satisfying $g(\ell) > g(r)$ for 
each rule $(\ell \rightarrow r) \in S$, 
- length-reducing, if $|\ell| > |r|$ for each rule $(\ell \rightarrow r) \in S$, 
- generalized monadic, if $|\ell| \geq |r|$ and $|r| \leq 1$ for each rule $(\ell \rightarrow r) \in S$, 
- monadic, if $|\ell| > |r|$ and $|r| \leq 1$ for each rule $(\ell \rightarrow r) \in S$, 
- confluent, if, for all $u, v, w \in \Sigma^*$, $u \Leftrightarrow^*_S v$ implies that there exists some $z \in \Sigma^*$ 
such that $u \Rightarrow^*_S z$ and $v \Rightarrow^*_S z$ hold, and 
- convergent, if it is both terminating and confluent.

Obviously, each weight-reducing system is terminating, and each length-reducing 
system is weight-reducing. For a convergent system $S$, the set $\mathrm{IRR}(S)$ of irreducible 
strings is a complete set of unique representatives for the Thue congruence $\Leftrightarrow^*_S$ 
(see, e.g., [4]).

While it is undecidable in general whether a finite string-rewriting system is con-
fluent (see, e.g., [4]), confluence is a decidable property for finite string-rewriting 
systems that are terminating. Let $S$ be a string-rewriting system on $\Sigma$. If there are 
two rules $(\ell \rightarrow r)$ and $(\ell' \rightarrow r')$ in $S$ such that $\ell = u\ell'v$ for some $u, v \in \Sigma^*$, then the 
word $\ell$ can be rewritten by either of the two rules: $\ell \Rightarrow_S r$ and $\ell = u\ell'v \Rightarrow_S ur'v$. 
If the system $S$ is to be confluent, then the words $r$ and $ur'v$ must have a common 
descendant. Accordingly, the pair $(r, ur'v)$ is called a critical pair of $S$. Furthermore, if $\ell = uv$ and $\ell' = vw$ for some words $u, v, w \in \Sigma^+$, then the word $uvw$ can 
be rewritten by either rule: $uvw = lw \Rightarrow_S rw$ and $uvw = ul'v \Rightarrow_S ur'v$. Hence, also 
$(rw, ur')$ is a critical pair of $S$.

**Proposition 2.1** [16]. A terminating string-rewriting system is confluent if and 
only if, for each critical pair $(p, q)$ of $S$, $p$ and $q$ have a common descendant mod $S$.

If $S$ is finite, then it has only finitely many critical pairs, which can be computed. 
Hence, it follows immediately that confluence is decidable for finite terminating 
string-rewriting systems.

Next we come to the notion of McNaughton family of languages. A language 
$L \subseteq \Sigma^*$ is called a McNaughton language, if there exist a finite alphabet $\Gamma$ strictly 
containing $\Sigma$, a finite string-rewriting system $S$ on $\Gamma$, strings $t_1, t_2 \in (\Gamma \setminus \Sigma)^* \cap$ 
$\mathrm{IRR}(S)$, and a letter $Y \in (\Gamma \setminus \Sigma) \cap \mathrm{IRR}(S)$ such that, for all $w \in \Sigma^*$, $w \in L$ if and 
only if $t_1 w t_2 \Rightarrow^*_S Y$. Here the symbols of $\Sigma$ are terminals, while those of $\Gamma \setminus \Sigma$
can be seen as nonterminals. We say that the McNaughton language $L$ is specified by the four-tuple $(S, t_1, t_2, Y)$. This fact will be expressed as $L = L(S, t_1, t_2, Y)$.

We illustrate this definition by a simple example.

**Example 2.2.** Let $\Sigma = \{a\}$, let $\Gamma = \{a, \#, F, Y\}$, and let $S$ be the following finite and length-reducing string-rewriting system on $\Gamma$:

$$S = \{\#aaaa \rightarrow \#aaF, Faa \rightarrow aF, FS \rightarrow \#, \#aa$ \rightarrow Y, \#a$ \rightarrow Y\}.$$ 

This system does not have any critical pairs, and hence, it is confluent. Now, for all $m \in \mathbb{N}$, $\#a^m \Rightarrow^*_S Y$ if and only if $m = 2^n$ for some $n \geq 0$, which implies that the McNaughton language $L(S, \#, \#, Y)$ is the language $L_{\text{expo}} = \{a^{2^n} \mid n \in \mathbb{N}\}$.

By placing restrictions on the finite string-rewriting systems used we obtain certain families of McNaughton languages. A McNaughton language is called weight-reducing (length-reducing), if it is defined using a finite string-rewriting system that is weight-reducing (length-reducing). The resulting class of languages is denoted by wr-McNL (lr-McNL). A McNaughton language is called (generalized) monadic, if it is defined using a finite string-rewriting system that is (generalized) monadic. The resulting language classes are denoted by gen-mon-McNL and mon-McNL. By requiring, in addition, that the string-rewriting system is confluent, we obtain the McNaughton families con-wr-McNL, con-lr-McNL, con-gen-mon-McNL, and con-mon-McNL. Thus, Example 2.2 shows that $L_{\text{expo}} \in \text{con-lr-McNL}$. Concerning these families the following results are known.

**Theorem 2.3** [3,17].

(a) $\text{GCSL} = \text{wr-McNL} = \text{lr-McNL}$.

(b) $\text{CRL} = \text{con-wr-McNL} = \text{con-lr-McNL}$.

(c) $\text{CFL} = \text{gen-mon-McNL} = \text{mon-McNL}$.

(d) $\text{REG} \subset \text{con-mon-McNL} \subset \text{con-gen-mon-McNL} \subset \text{symDCFL}$.

Here REG and CFL denote the classes of regular and context-free languages, and $\text{symDCFL} = \text{DCFL} \cap \text{DCFL}^R$, that is, a language $L$ belongs to $\text{symDCFL}$, if both, $L$ and $L^R$, are deterministic context-free. It is still open whether the second inclusion in (d) is proper. In fact, it is shown in [17] that the families con-mon-McNL and con-gen-mon-McNL coincide, if and only if the former is closed under inverse strictly alphabetic morphisms.

### 3. Limited context restarting automata

The **limited context restarting automaton**, abbreviated as lc-R-automaton, was introduced in [2] as a generalization of the clearing restarting automaton. Here we introduce a slightly generalized version which uses weight-reducing rules instead of length-reducing ones.

**Definition 3.1.** A **limited context restarting automaton** $M$ is defined through a triple $M = (\Sigma, \Gamma, I)$, where $\Sigma$ is an input alphabet, $\Gamma$ is a working alphabet
containing $\Sigma$, and $I$ is a finite set of instructions of the form $(u \mid x \rightarrow y \mid v)$, where $x, y \in \Gamma^*$ such that $g(x) > g(y)$ for some weight function $g : \Gamma^* \rightarrow \mathbb{N}$, $u \in \{\lambda, \$\} \cdot \Gamma^*$, and $v \in \Gamma^* \cdot \{\lambda, \$\}$. Here $\$ and $\$ are the left and right sentinels, which are not elements of $\Gamma$.

The lc-$R$-automaton $M = (\Sigma, I, I)$ induces a reduction relation $\vdash^c_M$ on $\Gamma^*$ that is defined as follows: for each $w, z \in \Gamma^*$, $w \vdash^c_M z$, if there exist words $w_1, w_2 \in \Gamma^*$ and an instruction $(u \mid x \rightarrow y \mid v)$ in $I$ such that $w = w_1xw_2$, $z = w_1yw_2$, $u$ is a suffix of $w_1$ and $v$ is a prefix of $w_2\$$. Thus, the factor $x$ is rewritten into $y$, if it appears within the context $uxv$. By $\vdash^c_M$ we denote the reflexive and transitive closure of $\vdash^c_M$. The language accepted by the lc-$R$-automaton $M$ is $L(M) = \{ w \in \Sigma^* \mid w \vdash^c_M \lambda \}$.

An lc-$R$-automaton $M$ accepts exactly the set of input words which can be reduced to $\lambda$. Obviously, $\lambda$ is in $L(M)$ for each lc-$R$-automaton $M$. Accordingly, if $L$ is a language that does not contain $\lambda$ as an element, then $L$ is not accepted by any lc-$R$-automaton. In order to overcome this problem, we will consider equality of languages only up to the empty word, that is, we will say that two languages $L$ and $L'$ on $\Sigma$ are equal, denoted as $L = L'$, if $L \cap \Sigma^+ = L' \cap \Sigma^+$ holds.

**Example 3.2.** Let $M = (\{a, b, c\}, \{a, b, c\}, I)$ be the lc-$R$-automaton that is defined by the set of instruction $I = \{(\lambda \mid acbb \rightarrow c \mid \lambda), (\$ \mid c \rightarrow \lambda \mid \$)\}$. Then $aaacbbbbbb \vdash^c_M aacbbbb \vdash^c_M acbb \vdash^c_M c \vdash^c_M \lambda$, and so the word $a^3cb^6$ belongs to $L(M)$. It is easily seen that $L(M) = \{ a^nbc^{2n} \mid n \geq 0 \}$.

Recall from Definition 3.1 that all instructions of an lc-$R$-automaton are necessarily weight-reducing. These most general lc-$R$-automata are said to be of type $R_0$. We now define some restricted types of lc-$R$-automata by putting additional restrictions on the form of their instructions. We say that an lc-$R$-automaton $M = (\Sigma, I, I)$ is of type

- $R_0$, if $I$ only contains instructions of the form $(u \mid x \rightarrow y \mid v)$, where $|x| > |y|$;
- $R'_0$, if $I$ only contains instructions of the form $(u \mid x \rightarrow y \mid v)$, where $|y| \leq 1$, and $x \in \Gamma^+$;
- $R_1$, if $I$ only contains instructions of the form $(u \mid x \rightarrow y \mid v)$, where $|y| \leq 1$, and $x \in \Gamma^+$ such that $|x| > |y|$;
- $R'_1$, if $I$ only contains instructions of the form $(u \mid x \rightarrow y \mid v)$, where $|y| \leq 1$, and $x \in \Gamma^+$ such that $|x| > |y|$;
- $R_2$, if $I$ only contains instructions of the form $(u \mid x \rightarrow y \mid v)$, where $|y| \leq 1$, $u \in \{\lambda, \$\}$, $v \in \{\lambda, \$\}$, and $x \in \Gamma^+$ such that $|x| > |y|$;
- $R'_2$, if $I$ only contains instructions of the form $(u \mid x \rightarrow y \mid v)$, where $|y| \leq 1$, $u \in \{\lambda, \$\}$, $v \in \{\lambda, \$\}$, and $x \in \Gamma^+$ such that $|x| > |y|$;
- $R_3$, if $I$ only contains instructions of the form $(u \mid x \rightarrow y \mid \$)$, where $|y| \leq 1$, $u \in \{\lambda, \$\}$, and $x \in \Gamma^+$ such that $|x| > |y|$.

For any $R \in \{R_0, R'_0, R_1, R'_1, R_2, R'_2, R_3, R'_3\}$, we will refer to lc-$R$-automata of type $R$ as lc-$R[R]$-automata. In [1], Basovnk studied length-reducing lc-$R$-automata, obtaining the following three characterizations.
Theorem 3.3 [1].
(a) $\mathcal{L}(\text{lc-R}[\mathcal{R}_0]) = \text{GCSL}$. (b) $\mathcal{L}(\text{lc-R}[\mathcal{R}_2]) = \text{CFL}$. (c) $\mathcal{L}(\text{lc-R}[\mathcal{R}_3]) = \text{REG}$.

In the current section we complete these results by also studying the other five types of lc-R-automata.

Let $M = (\Sigma, I, I)$ be an lc-R-automaton. With $M$ we can associate the finite string-rewriting system $R(M) = \{ u x v \to u x y v \mid (u \mid x \to y \mid v) \in I \}$. Then $R(M)$ induces a reduction relation $\Rightarrow^*_R(M)$ on the set of bordered words $\frak{T} \cdot \Gamma^* \cdot \frak{S}$, which is the reflexive and transitive closure of the single-step reduction relation $\Rightarrow^*_R(M)$ (see Sect. 2). Thus, for all $w, z \in \Gamma^*$, we have $qw\frak{S} \Rightarrow_R(M) qz\frak{S}$ iff $w \vdash^*_M z$ holds. Accordingly, we see that

$$L(M) = \{ w \in \Sigma^* \mid w \vdash^*_M \lambda \} = \{ w \in \Sigma^* \mid qw\frak{S} \Rightarrow^*_R(M) q\frak{S} \}.$$

By taking $S(M) = R(M) \cup \{ \frak{T}\frak{S} \to Y \}$, where $Y$ is a new letter, we obtain a finite string-rewriting system on $\Gamma' = \Gamma \cup \{ \frak{T}, \frak{S}, Y \}$ such that $L(M) = \{ w \in \Sigma^* \mid w\frak{S} \Rightarrow^*_S(M) \}$. It follows that $L(M)$ is the McNaughton language that is specified by the four-tuple $(S(M), \frak{T}, \frak{S}, Y)$.

If $M$ is of type $\mathcal{R}_0$, then the string-rewriting system $S(M)$ is weight-reducing. Hence, it follows from Theorem 2.3(a) that $L(M) \in \text{GCSL}$. Together with Theorem 3.3(a), this yields the following characterization, as each lc-R-automaton of type $\mathcal{R}_0$ is obviously also of type $\mathcal{R}_0$.

**Theorem 3.4.** $\mathcal{L}(\text{lc-R}[\mathcal{R}_0]) = \mathcal{L}(\text{lc-R}[\mathcal{R}_0]) = \text{GCSL}$.

In the proof of Theorem 3.3(a) in [1], the author constructs an lc-R-[R_0]-automaton $M$ for the language $L(G)$ from a given growing context-sensitive grammar $G$. Basovník’s thesis [1] is difficult to access. Therefore, we present a full proof for Theorem 3.4 using a construction that is based on the two-pushdown automata of [7], as we will refer back to this proof later.

**Proof of Theorem 3.4.** Because of the above observation, it suffices to show that each growing context-sensitive language is accepted by some lc-R-automaton of type $\mathcal{R}_0$. Hence, assume that $L \subseteq \Sigma^*$ is growing context-sensitive. We construct an lc-R[\mathcal{R}_0]-automaton $M$ for $L$. As $L$ is growing context-sensitive, there exists a length-reducing two-pushdown automaton (TPDA, for short, see, e.g., [7]) $T = (Q, \Sigma, \Gamma, \delta, q_0, \bot, \lambda, \{ q_f \})$ that accepts $L$. Here $\Gamma$ is the tape alphabet of $T$ that includes the input alphabet $\Sigma$ as well as the special bottom marker $\bot$ for the pushdowns. Thus, for all $w \in \Sigma^*, w \in L$ if and only if $T$ accepts starting from the initial configuration $\bot q_0 w \bot$, that is, if and only if $\bot q_0 w \bot \vdash^*_T q_f$ holds (see [20] Lems. 3.4 and 4.1). Actually, we may even assume without loss of generality that the initial step of a computation of $T$ that starts from an initial configuration of the form $\bot q_0 w \bot$ reduces the overall length of the configuration by at least two, and that $T$ never enters its initial state $q_0$ during a computation.

Let $\overline{T} = \{ \pi \mid a \in \Gamma \}$ be a new alphabet in one-to-one correspondence to $\Gamma$ such that $Q, \Gamma$, and $\overline{T}$ are pairwise disjoint, let $\overline{\cdot} : \Gamma^* \to \overline{\Gamma}^*$ denote the corresponding
isomorphism that is induced by \( a \mapsto \overline{a} \) \((a \in \Gamma)\), and let \( \Delta = Q \cup \Gamma \cup \overline{T} \). In analogy to the proof of [20] Lemma 4.1, we can construct a finite length-reducing string-rewriting system \( R \) on \( \Delta \) such that, for all \( w \in \Sigma^* \), \( w \in L \) iff \( \overline{\rew_{q_0} w \perp} \Rightarrow^*_R q_f \). Here the final state \( q_f \) is introduced by specific rules of the form \(| \overline{\rew q_0 w \perp} \Rightarrow q_f \). Now from \( R \) we obtain an lc-R-automaton \( M = (\Sigma, \Delta, I) \) by taking

\[
I = \{ (\lambda \mid \ell \to r \mid \lambda) \mid (\ell \to r) \in R, |\ell|^I = |r|^I = 0 \} \cup \\
\{ (\ell \mid u \to v \mid \lambda) \mid (\rew u \Rightarrow \rew v) \in R, |u|^I = 0 \Rightarrow |v|^I = 0 \} \cup \\
\{ (\ell \mid u \to v \mid \$) \mid (u \perp \Rightarrow v \perp) \in R, |u|^I = 0 \Rightarrow |v|^I = 0 \} \cup \\
\{ (\ell \mid u \to v \mid \$) \mid (\rew u \Rightarrow \rew v \perp) \in R, |u|^I = 0 \Rightarrow |v|^I = 0 \} \cup \\
\{ (\ell \mid u \to \lambda \mid \$) \mid (\rew q_0 u \perp \Rightarrow q_f) \in R, |u|^I = 0 \Rightarrow |q_f|^I = 0 \}\].

Then \( M \) is of type \( R_0 \), and for all \( w \in \Sigma^+ \),

\[
w \in L(M) \Leftrightarrow w \not\vdash^*_M \lambda \Leftrightarrow \rew w \Rightarrow^*_R \rew (M) \rew \rew \Rightarrow \rew q_f \iff w \in L.
\]

Thus, \( L(M) = L \). This completes the proof of Theorem 3.4. \( \square \)

Let \( G = (N, T, S, P) \) be a weight-increasing context-sensitive grammar, that is, there exists a weight function \( g \) such that \( g(\ell) < g(r) \) for each rule \( (\ell \to r) \) of \( P \), and in addition, each rule \( (\ell \to r) \) has the form \( \ell = uAv \) and \( r = uwx \) for some \( A \in N, u, v \in (N \cup T)^+ \), and \( x \in (N \cup T)^+ \). By taking

\[
I(G) = \{ (u \mid x \to A \mid v) \mid (uAv \Rightarrow uwx) \in P \} \cup \{ (\ell \mid r \to \lambda \mid \$) \mid (S \to r) \in P \},
\]

we obtain an lc-R-automaton \( M(G) = (T, N \cup T, I(G)) \). It is easily seen that \( M(G) \) is of type \( R_1 \), and that \( L(M(G)) = L(G) \cup \{ \lambda \} \) holds. Thus, ignoring the special case of the empty word we see that all languages that are generated by weight-increasing context-sensitive languages are accepted by lc-R[\( R_1^c \)]-automata. However, the class of languages that are generated by weight-increasing context-sensitive grammars, which is known as the class ACSL of acyclic context-sensitive languages, coincides with the class GCSL of growing context-sensitive languages [21]. Hence, we obtain the following characterization.

**Theorem 3.5.** \( \mathcal{L}(\text{lc-R}[R_1^c]) = \text{GCSL} \).

If \( M = (\Sigma, \Gamma, I) \) is an lc-R[\( R_1^c \)]-automaton, then the rules of \( R(M) \) have the form \( (uxv \to uyv) \), where \( |x| > |y| \) and \( |y| \leq 1 \). By replacing each instruction of the form \( (u \mid x \to \lambda \mid v) \in I \) by finitely many rules with a revised context, the following technical result can be derived.

**Lemma 3.6.** If \( M = (\Sigma, \Gamma, I) \) is an lc-R[\( R_1^c \)]-automaton, then there exists an equivalent lc-R-automaton \( M' = (\Sigma, \Gamma, I') \) of type \( R_1 \) such that \( |y| = 1 \) for all instructions \( (u \mid x \to y \mid v) \in I' \) satisfying \( u \neq \$ \) or \( v \neq \$ \). In fact, for all \( w, z \in \Gamma^* \), \( w \vdash^c_M z \) if and only if \( w \vdash^c_{M'} z \).
Proof. To obtain $M'$ from $M$, we simply replace each instruction of the form $i = (u \mid x \rightarrow \lambda \mid v) \in I$. If $u = u_1A$ for some $A \in I$, then we replace $i$ by the instruction $i' = (u_1 \mid Ax \rightarrow A \mid v)$; if $u = \lambda$ or $u = \epsilon$ and $v = Bv_1$ for some $B \in I$, then we replace $i$ by the instruction $i' = (u \mid xB \rightarrow B \mid v_1)$; if $u = \epsilon$ and $v = \lambda$, then we replace $i$ by the set of instructions $I'(i) = \{(\lambda \mid xA \rightarrow A \mid \lambda) \mid A \in I\} \cup \{(\lambda \mid x \rightarrow \lambda \mid \lambda) \mid \lambda \in \Sigma\}$. Then, for all $w, z \in I^*$, if $w \vdash_M z$ using instruction $i$, then $w \vdash_M z$ by instruction $i'$ or by one of the instructions of $I'(i)$, and conversely, if $w \vdash_M z$ by instruction $i'$ or by one of the instructions of $I'(i)$, then $w \vdash_M z$ by instruction $i$. 

Now let $M = (\Sigma, \Gamma, I)$ be an lc-R[\mathcal{R}_1]-automaton that satisfies the properties of Lemma 3.6, let $R(M)$ be the corresponding string-rewriting system, and let $R(M)^{-1}$ denote the system $R(M)^{-1} = \{(v \rightarrow u) \mid (u \rightarrow v) \in R(M)\}$. From $R(M)^{-1}$ we can construct a length-increasing context-sensitive grammar $G(M) = (I', \Sigma', S, \mathcal{R}')$ that generates the language $L(G(M)) = \mathcal{R} \cdot L(M) \cdot \Sigma$. Thus, the language $\mathcal{R} \cdot L(M) \cdot \Sigma$ is a growing acyclic context-sensitive language (see, e.g., [21]). It is known from [6] that the class GACSL of growing acyclic context-sensitive languages is closed under the operations of removing left and right end markers. Hence, the language $L(M)$ is growing acyclic context-sensitive, too. Conversely, if $G = (N, T, S, P)$ is a length-increasing context-sensitive grammar, then by taking

$I(G) = \{(u \mid x \rightarrow A \mid v) \mid (uAv \rightarrow uxv) \in P\} \cup \{(\epsilon \mid r \rightarrow \lambda \mid \Sigma) \mid (S \rightarrow r) \in P\}$,

we obtain an lc-R-automaton $M(G) = (T, N \cup T, I(G))$ of type $\mathcal{R}_1$ such that $L(M(G)) = L(G) \cup \{\lambda\}$ holds. Thus, we have the following characterization.

**Theorem 3.7.** $L(\text{lc-R}[\mathcal{R}_1]) = \text{GACSL}$.

It is known that the class GACSL properly contains the class CFL of context-free languages, and it is obviously contained in ACFL = GCSL. However, it is an open problem whether this containment is strict or not.

Let $M = (\Sigma, \Gamma, I)$ be an lc-R[\mathcal{R}_2]-automaton. Then, for each instruction $(u \mid x \rightarrow y \mid v) \in I$, we have $u \in \{\lambda, \epsilon\}$, $v \in \{\lambda, \Sigma\}$, $|x| \geq 1$, $|y| \leq 1$, and $g(x) > g(y)$ for a fixed weight function $g$. Hence, the corresponding string-rewriting system $R(M)$ can be split into four disjoint subsystems:

(a) $R_{\text{bif}} = \{\epsilon x\Sigma \rightarrow \epsilon y\Sigma \mid (\epsilon \mid x \rightarrow y \mid \Sigma) \in I\}$, the bifix rules of $R(M)$,
(b) $R_{\text{pre}} = \{\epsilon x \rightarrow y \mid (\epsilon \mid x \rightarrow y \mid \lambda) \in I\}$, the prefix rules of $R(M)$,
(c) $R_{\text{suf}} = \{x\Sigma \rightarrow y\Sigma \mid (\lambda \mid x \rightarrow y \mid \Sigma) \in I\}$, the suffix rules of $R(M)$,
(d) $R_{\text{inf}} = \{x \rightarrow y \mid (\lambda \mid x \rightarrow y \mid \lambda) \in I\}$, the infix rules of $R(M)$.

Let $B(M) = \{\alpha \in \Gamma^* \mid \alpha \Sigma \in \text{dom}(R_{\text{bif}})$ and $\alpha \Sigma \Rightarrow_{R(M)}^{*} \epsilon \Sigma\}$ $\cup \{\lambda\}$. As $R(M)$ is finite, so is the set $B(M)$. Let $R' = R_{\text{pre}} \cup R_{\text{suf}} \cup R_{\text{inf}}$. Then

$L(M) = \{w \in \Sigma^* \mid \epsilon w \Sigma \Rightarrow_{R(M)}^{*} \epsilon \Sigma\} \cup \{w \in \Sigma^* \mid \exists \alpha \in B(M) : \epsilon w \Sigma \Rightarrow_{R'}^{*} \epsilon \alpha \Sigma\}$,
that is, \( \psi \cdot L(M) \cdot \$ = \nabla_{R'}^\ast (\psi \cdot B(M) \cdot \$) \cap (\psi \cdot \Sigma^\ast \cdot \$). \) Recall that \( \nabla_{R'}^\ast (\alpha) \) denotes the set of ancestors of \( \alpha \) mod \( R' \), and \( \nabla_{R'}^\ast (B) = \bigcup_{\alpha \in B} \nabla_{R'}^\ast (\alpha) \) for any set \( B \). Now we define a mixed rewriting system (see, e.g., [12]) \( P(M) = P_{\text{pre}} \cup P_{\text{suf}} \cup P_{\text{inf}} \) by taking the prefix-rewriting system\(^3\) \( P_{\text{pre}} = \{ x \to y \mid (x \to y) \in R_{\text{pre}} \} \), the suffix-rewriting system\(^4\) \( P_{\text{suf}} = \{ x \to y \mid (x \$ \to y\$) \in R_{\text{suf}} \} \), and the string-rewriting system \( P_{\text{inf}} = R_{\text{inf}} \). Then \( L(M) = \nabla_{P(M)}^\ast (B(M)) \cap \Sigma^\ast \). As \( P(M) \) only contains generalized monadic rules (see, e.g., [17]), it follows that the set \( \nabla_{P(M)}^\ast (B(M)) \) is context-free, which in turn implies that \( L(M) \) is a context-free language.

Conversely, if \( L \subseteq \Sigma^\ast \) is a context-free language, then there exists a context-free grammar \( G = (N, \Sigma, S, P) \) for \( L \cap \Sigma^{\geq 2} \) such that, for each rule \((\ell \to r) \) of \( P \), we have \( |\ell| = 1 \) and \( |r| = 2 \). We now obtain an lc-R-automaton \( M(G) = (\Sigma, N \cup \Sigma, I) \) by taking

\[
I = \{ (\lambda | r \to \ell | \lambda) \mid (\ell \to r) \in P \} \cup \{ (\psi | a \to \lambda | \$) \mid a \in (\Sigma \cap L) \cup \{S\} \}.
\]

Then \( M(G) \) is of type \( \mathcal{R}_2 \), and \( L(M(G)) \equiv L \). Thus, we have the following characterization.

**Theorem 3.8.** \( L(\text{lc-R}[\mathcal{R}_2]) = L(\text{lc-R}[\mathcal{R}_2]) = \text{CFL} \).

Finally, let \( M = (\Sigma, I, \mathcal{I}) \) be an lc-R[\mathcal{R}_2]-automaton, that is, for all instructions \((u | x \to y | v) \in I \), we have \( u \in \{\lambda, \psi\}, v = \$, \( |x| \geq 1 \), \(|y| \leq 1 \), and \( g(x) > g(y) \) for a fixed weight function \( g \). Hence, the corresponding string-rewriting system \( R(M) \) can be split into two disjoint subsystems:

- (a) \( R_{\text{bif}} = \{ \psi x \$ \to \psi y \$ \mid (\psi | x \to y | \$) \in I \} \), the bifix rules of \( R(M) \),
- (b) \( R_{\text{suf}} = \{ x \$ \to y \$ \mid (\lambda | x \to y | \$) \in I \} \), the suffix rules of \( R(M) \).

As above, the set \( B(M) = \{ \alpha \in \Gamma^\ast | \psi \alpha \$ \in \text{dom}(R_{\text{bif}}) \) and \( \psi \alpha \$ \Rightarrow_{R(M)} \$ \} \cup \{ \lambda \} \) is finite, and

\[
L(M) = \{ w \in \Sigma^\ast | \psi w \$ \Rightarrow_{R(M)}^\ast \$ \} = \{ w \in \Sigma^\ast | \exists \alpha \in B(M) : \psi w \$ \Rightarrow_{R_{\text{suf}}}^\ast \psi \alpha \$ \},
\]

that is, \( \psi \cdot L(M) \cdot \$ = \nabla_{R_{\text{suf}}}^\ast (\psi \cdot B(M) \cdot \$) \cap (\psi \cdot \Sigma^\ast \cdot \$). \) For the suffix-rewriting system \( P(M) = \{ y \to x \mid (x \$ \to y \$) \in R_{\text{suf}} \} \), we obtain \( L(M) = \Delta_{P(M)}^\ast (B(M)) \cap \Sigma^\ast \). As \( B(M) \) is finite, and, hence, regular, it follows that the set of descendants \( \Delta_{P(M)}^\ast (B(M)) \) of this set with respect to the suffix rewriting system \( P(M) \) is regular [5], which in turn implies that \( L(M) \) is regular.

Conversely, if \( L \subseteq \Sigma^\ast \) is a regular language, then there exists a regular grammar \( G = (N, \Sigma, S, P) \) for \( L \cap \Sigma^{\geq 2} \) such that, for each rule \((\ell \to r) \) of \( P \), we have \( \ell \in N \) and \( r \in \Sigma \cdot N \cup \Sigma^2 \). We now obtain an lc-R-automaton \( M(G) = (\Sigma, N \cup \Sigma, I) \) by taking

\[
I = \{ (\lambda | r \to \ell | \$) \mid (\ell \to r) \in P \} \cup \{ (\psi | a \to \lambda | \$) \mid a \in (L \cap \Sigma) \cup \{S\} \}.
\]

Then \( M(G) \) is of type \( \mathcal{R}_3 \), and \( L(M(G)) \equiv L \). Thus, we have the following characterization.

---

\(^3\)The rules of a prefix-rewriting system can only be applied to the prefix of a word.

\(^4\)The rules of a suffix-rewriting system can only be applied to the suffix of a word.
Theorem 3.9. \( L(\text{lcr}[R'_3]) = L(\text{lcr}[R_3]) = \text{REG} \).

The results on the various \text{lcr}-automata are summarized by the diagram in Figure 1.

4. Confluent limited context restarting automata

As defined in Definition 3.1, an \text{lcr}-automaton \( M = (\Sigma, \Gamma, I) \) is a nondeterministic device. A word \( w \in \Sigma^* \) belongs to the language \( L(M) \) accepted by \( M \), if and only if there exists a computation of \( M \) that transforms the initial configuration with tape content \( \downarrow \cdot w \cdot \$ \) into the configuration with tape content \( \uparrow \cdot \$ \). In general, there will be many different computations of \( M \) that start from the configuration with tape content \( \downarrow \cdot w \cdot \$ \), and only some of them will derive the tape content \( \uparrow \cdot \$ \). As this phenomenon complicates the problem of deciding membership in \( L(M) \), we are interested in \text{lcr}-automata for which all computations from \( \downarrow \cdot w \cdot \$ \) lead to \( \uparrow \cdot \$ \), if \( w \in L(M) \). Actually, as the reduction relation \( \vdash^c_M \) corresponds to the single-step reduction relation \( \Rightarrow_{R(M)}^c \) that is induced by the string-rewriting system \( R(M) \) on the set of bordered words \( \Phi \cdot \Gamma^* \cdot \$ \), this would lead to considering \text{lcr-automata} \( M \) for which the string-rewriting system \( R(M) \) is confluent on the congruence class \( [\downarrow \cdot \Phi]_{R(M)} \). Unfortunately, it is undecidable in general whether a finite string-rewriting system is confluent on a given congruence class, even if the given finite system only contains length-reducing rules \cite{22}. Therefore, we turn to \text{lcr-automata} that satisfy an even stronger restriction than confluence on a particular congruence class.

Definition 4.1. An \text{lcr-automaton} \( M = (\Sigma, \Gamma, I) \) is called confluent if the corresponding string-rewriting system \( R(M) \) is confluent.

As \( R(M) \) is a finite and terminating string-rewriting system for each \text{lcr-automaton}, confluence is a decidable property of \text{lcr-automata} (see Prop. 2.1.
and the subsequent paragraph). We illustrate the above definition by a simple example.

Example 4.2. Let $\Sigma = \{a\}$, let $\Gamma = \{a, F\}$, and let $I = \{(\# | aaaa \rightarrow aaF | \lambda), (\lambda | Faa \rightarrow aF | \lambda), (\lambda | F \rightarrow \lambda | \$), (\# | aa \rightarrow \lambda | \$), (\# | a \rightarrow \lambda | \$)\}$. Then $M = (\Sigma, \Gamma, I)$ is an lc-R-automaton of type $R_0$ that accepts the language $L_{\text{expo}}$, and it is easily checked that $M$ is confluent.

We will use the prefix $\text{con-}$ to denote types of confluent lc-R-automata. Further, for each type $R \in \{R'_i, R_i \mid i \in \{0, 1, 2, 3\}\}$, lc-R[con-$R$] will denote the class of lc-R-automata of type $R$ that are confluent.

In the following we study the expressive power of the various types of confluent lc-R-automata. As in the previous section we consider the various types in turn, from the most general one to the most restricted one.

If $M = (\Sigma, \Gamma, I)$ is an lc-R[con-$R'_0$]-automaton, then $S(M) = R(M) \cup \{\# \$ \rightarrow Y\}$ is a finite weight-reducing string-rewriting system that is confluent, and $L(M)$ is simply the McNaughton language that is specified by $(S(M), \#, \$, Y). Thus, $L(M)$ is a Church–Rosser language [18]. On the other hand, if $L \subseteq \Sigma^*$ is a Church–Rosser language, then it is accepted by a length-reducing deterministic two-pushdown automaton [20]. Following the proof of Theorem 3.4, a confluent lc-R-automaton of type $R_0$ can be constructed for $L$. Hence, we obtain the following characterization.

**Theorem 4.3.** $L(\text{lc-R[con-$R'_0$]}) = L(\text{lc-R[con-$R_0$]}) = \text{CRL}.$

In [25] Woinowski introduced a normal form for presentations of Church–Rosser languages, called context-splittable Church–Rosser language systems. Such a presentation is of the form $(S, \#, \$, Y), where $S$ is a finite, weight-reducing, and confluent string-rewriting system that consists of rules of the form $(uxv \rightarrow uyyv)$, where $u, v \in \Gamma^*, x \in \Gamma^+$, and $|y| \leq 1$, and rules of the form $(\# \$ \rightarrow Y)$, where $x \in \Gamma^+$. As each Church–Rosser language admits a presentation of this form [25], and as a presentation of this form can immediately be translated into an lc-R-automaton of type $R'_1$, this gives the following characterization.

**Theorem 4.4.** $L(\text{lc-R[con-$R'_1$]}) = \text{CRL}.$

For the class of languages that are accepted by confluent lc-R-automata of type $R_1$, we have no characterization result yet. On the one hand, these automata can only accept (certain) Church–Rosser languages, and hence, they do not accept all context-free languages. On the other hand, the language $L_{\text{expo5}} = \{a^5^n \mid n \geq 0\}$ is accepted by an lc-R[con-$R_1$]-automaton, as shown in [24]. As this language is not context-free, we can at least conclude the following.

**Corollary 4.5.** The language class $L(\text{lc-R[con-$R_1$]})$ is incomparable to the class CFL with respect to inclusion.
It remains open whether lc-R-automata of type con-\(\mathcal{R}_1\) accept all Church–Rosser languages, in fact, it is even open whether they accept at least all deterministic context-free languages.

Now we turn to the confluent lc-R-automata of type \(\mathcal{R}'_2\).

**Theorem 4.6.** \(L(lc-R[con-\mathcal{R}'_2]) \subseteq con-gen-mon-McNL\).

Recall from Section 2 that \(con-gen-mon-McNL\) denotes the family of confluent generalized monadic McNaughton languages.

**Proof.** Let \(M = (\Sigma, \Gamma, I)\) be an \(lc-R[con-\mathcal{R}'_2]\)-automaton. As in the discussion that led to Theorem 3.8, the corresponding string-rewriting system \(R(M)\) can be split into four disjoint subsystems:

(a) \(R_{bif} = \{ qx$ \rightarrow qy$ \mid (q \mid x \rightarrow y \mid \$) \in I \}\), the bifix rules of \(R(M)\),
(b) \(R_{pre} = \{ qx \rightarrow qy \mid (q \mid x \rightarrow y \mid \lambda) \in I \}\), the prefix rules of \(R(M)\),
(c) \(R_{suf} = \{ x$ \rightarrow y$ \mid (\lambda \mid x \rightarrow y \mid \$) \in I \}\), the suffix rules of \(R(M)\),
(d) \(R_{inf} = \{ x \rightarrow y \mid (\lambda \mid x \rightarrow y \mid \lambda) \in I \}\), the infix rules of \(R(M)\).

Observe that together with \(R(M)\), also the string-rewriting system

\[
S(M) = R(M) \cup \{ q\$ \rightarrow Y \}
\]

is confluent, as the additional rule \(q\$ \rightarrow Y\) does not yield any additional critical pairs. In what follows, we will transform the system \(S(M)\) into a confluent generalized monadic string-rewriting system \(G\) on an extended alphabet \(\hat{\Gamma}\) such that \(L(M) = L(G, q, \$, Y)\) holds. This transformation will be realized in three steps. We use the notation \(nf(w)\) to denote the unique irreducible descendant of the word \(w \in (\Gamma' \cup \{q, \$, Y\})^*\) with respect to the string-rewriting system \(S(M)\). Recall that \(S(M)\) is convergent, as it is weight-reducing and confluent, and hence, these normal forms exist. Furthermore, we take \(\rho\) to denote the number \(\rho = \max\{ |x| \mid \exists u, y, v : (u \mid x \rightarrow y \mid v) \in I \}\).

**Step 1.** First we replace the subsystem \(R_{bif} \cup \{ q\$ \rightarrow Y \}\) by a new set of bifix rules \(R'_{bif}\) on \(\Gamma' = \Gamma' \cup \{q, \$, Y, N\}\), where \(N\) is another new letter. The system \(R'_{bif}\) is defined as follows:

\[
R'_{bif} = \{ qw$ \rightarrow Y \mid w \in \Gamma'^*, |w| \leq 2 \cdot \rho, nf(qw$) = Y \}\cup
\{ qw$ \rightarrow N \mid w \in \Gamma'^*, |w| \leq 2 \cdot \rho, nf(qw$) \neq Y \}.
\]

Then \(q\$ \rightarrow Y\) \(\in R'_{bif}\), and for all \(qw$ \rightarrow N\) \(\in R'_{bif}\), we have \(|w| \geq 1\). Let

\[
S_1(M) = R'_{bif} \cup R_{pre} \cup R_{suf} \cup R_{inf}.
\]

Obviously, \(S_1(M)\) is a finite and weight-reducing string-rewriting system.

**Claim 4.7.** For all \(w \in \Gamma'^*\), \(qw$ \Rightarrow^*_S(M) Y\) iff \(qw$ \Rightarrow^*_S(M) Y\).
Proof. For \( w \in \Gamma^* \), if \( \psi w \$ \Rightarrow^*_S(M) Y \), then either \( \psi w \$ \Rightarrow^*_{\text{pre}} \cup \text{R}_{\text{suf}} \cup \text{R}_{\text{inf}} \psi \$ \), or there exists a rule \( (\psi x \$ \Rightarrow \psi y \$) \in \text{R}_{\text{bif}} \) such that \( \psi w \$ \Rightarrow^*_{\text{pre}} \cup \text{R}_{\text{suf}} \cup \text{R}_{\text{inf}} \psi x \$ \Rightarrow \psi y \$ \Rightarrow^*_S(M) Y \) holds. As \( |x| \leq \rho \), we see that \( (\psi x \$ \Rightarrow Y) \) is contained in \( \text{R}_{\text{bif}}' \), and hence, \( \psi w \$ \Rightarrow^*_{S_1(M)} Y \) follows. Conversely, if \( (\psi x \$ \Rightarrow Y) \) is a rule of \( \text{R}_{\text{bif}}' \), then \( \psi w \$ \Rightarrow^*_{S_1(M)} Y \) holds. Thus, for any \( w \in \Gamma^* \), if \( \psi w \$ \Rightarrow^*_{S_1(M)} Y \), then also \( \psi w \$ \Rightarrow^*_{S(M)} Y \). \( \Box \)

It follows that \( L(M) \) is the McNaughton language that is specified by the 4-tuple \((S_1(M), \psi, \$, Y)\).

Claim 4.8. The string-rewriting system \( S_1(M) \) is confluent.

Proof. As the system \( S(M) \) is confluent, it follows immediately that the subsystems \( \text{R}_{\text{pre}} \cup \text{R}_{\text{inf}} \) and \( \text{R}_{\text{suf}} \cup \text{R}_{\text{inf}} \) of \( S_1(M) \) are also confluent. Thus, it remains to consider the critical pairs that result from overlapping a rule of \( \text{R}_{\text{pre}} \) with a rule of \( \text{R}_{\text{suf}} \), and those that result from overlaps with rules from \( \text{R}_{\text{bif}}' \).

First, let \((\psi x \Rightarrow \psi y) \in \text{R}_{\text{pre}} \) and \((x' \$ \Rightarrow y' \$) \in \text{R}_{\text{suf}} \) such that \( x = x_1 x_2 \) and \( x' = x_2 x_3 \) for some non-empty factor \( x_2 \). Then

\[
\psi y x_3 \$ \Leftarrow \psi x x_3 \$ = \psi x_1 x_2 x_3 \$ = \psi x_1 x' \$ \Rightarrow_{\text{R}_{\text{suf}}} \psi x_1 y' \$.
\]

As \( S(M) \) is confluent, \( \text{nf}(\psi y x_3 \$) = \text{nf}(\psi x_1 y' \$) \). Now \( |\psi y x_3|, |\psi x_1 y'| \leq |\psi x_1 x_2 x_3| < 2 \cdot \rho \), and hence, \( \text{R}_{\text{bif}}' \) contains the rules \((\psi y x_3 \$ \Rightarrow Z)\) and \((\psi x_1 y' \$ \Rightarrow Z)\) for some symbol \( Z \in \{Y, N\} \). Thus, the critical pair \((\psi y x_3 \$, \psi x_1 y' \$)\) resolves mod \( S_1(M) \).

Obviously, there are no overlaps between different rules of \( \text{R}_{\text{bif}}' \). Hence, it remains to consider the cases that \((\psi w \$ \Rightarrow Z) \in \text{R}_{\text{bif}}' \) and \((x \Rightarrow y) \in \text{R}_{\text{inf}} \) such that \( w = w_1 x w_2 \), or \((\psi x \Rightarrow \psi y) \in \text{R}_{\text{pre}} \) such that \( w = w_1 x \). Here we only consider the first of these cases, as the other two are dealt with in the same way. We have

\[
\psi w_1 y w_2 \$ \Leftarrow_{\text{R}_{\text{inf}}} \psi w_1 x w_2 \$ = \psi w \$ \Rightarrow_{\text{R}_{\text{bif}}'} Z.
\]

As \( S(M) \) is confluent, it follows that \( \text{nf}(\psi w_1 y w_2 \$) = \text{nf}(\psi w \$) \). Further, \( |\psi w_1 y w_2| \leq |w| \leq 2 \cdot \rho \), and hence, \( \text{R}_{\text{bif}}' \) also contains the rule \((\psi w_1 y w_2 \$ \Rightarrow Z)\). Thus, also the critical pair \((\psi w_1 y w_2 \$, Z)\) resolves. It follows that the system \( S_1(M) \) is indeed confluent. \( \Box \)

Step 2. Next we separate those letters that are (prefix or suffix) reducible to \( \lambda \) from the others. In this way we will transform the system \( S_1(M) \) into a system \( S_2(M) \). This is done as follows. First the right-hand side of each rule of \( S_1(M) \) is replaced by its unique irreducible descendant mod \( S_1(M) \). For the subsystem \( \text{R}_{\text{bif}}' \), nothing changes by this operation, but for a rule \((x \Rightarrow y) \in \text{R}_{\text{inf}}, \) if \(|y| = 1, \) then \( y \) may reduce to another letter \( y' \in \Gamma \) or to \( \lambda \), and analogously for the rules of \( \text{R}_{\text{pre}} \) and \( \text{R}_{\text{suf}} \). The system obtained in this way is still confluent and equivalent to \( S_1(M) \) (see, e.g., [4] Lem. 2.2.11).
Next, let $I_0 = \{ A \in \Gamma \mid (A \to \lambda) \in R_{\inf} \}$, that is, $I_0$ is the set of letters that reduce to $\lambda$. We now delete all rules from $R_{\pre}$, $R_{\suff}$ and $R_{\inf}$ that properly contain an occurrence of a letter from $I_0$ on their left-hand sides. It follows from the confluence property that the resulting system is still confluent and equivalent to $S_1(M)$ (see [4] Thm. 2.2.13 and the remark preceding Thm. 2.2.14).

Finally, let $I_p = \{ A \in \Gamma \setminus I_0 \mid (\psi A \to \bullet) \in R_{\pre} \}$, that is, $I_p$ is the set of letters that are prefix reducible to $\lambda$. From $R_{\pre}$ we now delete all those rules for which the left-hand sides have a proper prefix of the form $\psi A$ for some letter $A \in I_p$.

Analogously, let $I_s = \{ A \in \Gamma \setminus I_0 \mid (A\$ \to \$) \in R_{\suff} \}$. From $R_{\suff}$ we delete all those rules for which the left-hand sides have a proper suffix of the form $A\$ for some letter $A \in I_s$. By $S_2(M)$ we denote the system that is obtained by these operations. It has the form

$$S_2(M) = R'_{\bif} \cup R'_{\pre} \cup R'_{\suff} \cup R'_{\inf}, \quad (4.2)$$

where $R'_{\inf}$ consists of two subsystems, $R_{\inf,1} = \{ A \to \lambda \mid A \in I_0 \}$ and $R_{\inf,2} = \{ (x \to y) \in R_{\inf} \mid x \in (\Gamma \setminus I_0)^+ \}$, $R'_{\pre}$ consists of two subsystems, $R_{\pre,1} = \{ \psi A \to \bullet \mid A \in I_p \}$ and

$$R_{\pre,2} = \{ (\psi x \to \psi y) \in R_{\pre} \mid x \text{ does not begin with a letter from } I_p \},$$

and $R'_{\suff}$ consists of $R_{\suff,1} = \{ A\$ \to \$ \mid A \in I_s \}$ and

$$R_{\suff,2} = \{ (x\$ \to \$y) \in R_{\suff} \mid x \text{ does not end with a letter from } I_s \}.$$ 

Again it follows from the confluence property of $S_1(M)$ that the system $S_2(M)$ is still confluent and equivalent to $S_1(M)$. In particular, $L(M)$ is the McNaughton language that is specified by the tuple $(S_2(M), \psi, \$, $Y)$.

**Step 3.** The subsystem $R'_{\bif} \cup R'_{\inf}$ of $S_2(M)$ contains generalized monadic rules only, but $R'_{\pre,2}$ and $R_{\suff,2}$ may contain some rules that are not generalized monadic. Thus, we must replace these rules by some generalized monadic rules. For doing so we introduce the alphabet

$$\hat{\Gamma} = \Gamma' \cup \{ \psi A \mid A \in \Gamma \setminus (I_0 \cup I_p) \} \cup \{ A\$ \mid A \in \Gamma \setminus (I_0 \cup I_s) \}, \quad (4.3)$$

that is, for each letter $A$ that is not prefix reducible to $\lambda$, we add a letter $\psi A$, and for each letter $A$ that is not suffix reducible to $\lambda$, we add a letter $A\$. To simplify the notation, we write $\psi w$ to denote the word that is obtained from $w$ by replacing the first letter $A$ of $w$ by the symbol $\psi$, and analogously for $w\$, and we define a morphism $\psi : \hat{\Gamma}^* \to \Gamma^*$ through $A \mapsto A (A \in \Gamma)$, $\psi A \mapsto \psi A (A \in \Gamma \setminus (I_0 \cup I_p))$, $A\$ $\mapsto A\$ (A $\in \Gamma \setminus (I_0 \cup I_s))$, and $Z \mapsto Z (Z \in \{ \psi, \$, Y, $N \})$.

We define the system $G$ as

$$G = \hat{R}_{\pre} \cup \hat{R}_{\suff} \cup \hat{R}_{\bif} \cup \hat{R}_{\inf}, \quad (4.4)$$
where the subsystems $\hat{R}_{\text{pre}}$, $\hat{R}_{\text{suf}}$, and $\hat{R}_{\text{bif}}$ are defined as follows:

(a) $\hat{R}_{\text{pre}} = \{ qA \rightarrow qA \mid A \in \Gamma \setminus (I_0 \cup I_p) \} \cup R_{\text{pre},1} \cup \{ qx \rightarrow qy \mid (qx \rightarrow qy) \in R_{\text{pre},2} \text{ and } y \in \Gamma \} \cup \{ qx \rightarrow q \mid (qx \rightarrow q) \in R_{\text{pre},2} \} \cup \{ qx_1 \rightarrow qy \mid (x_1x \rightarrow y) \in R_{\text{inf},2}, x_1, y \in \Gamma \setminus (I_0 \cup I_p) \} \cup \{ qx_1 \rightarrow q \mid (x_1x \rightarrow y) \in R_{\text{inf},2}, x_1 \in \Gamma \setminus (I_0 \cup I_p), y \in I_p \} \cup \{ qx_1 \rightarrow q \mid (x_1x \rightarrow \lambda) \in R_{\text{inf},2}, x_1 \in \Gamma \setminus (I_0 \cup I_p) \}.$

Observe that $(qx \rightarrow qy) \in R_{\text{pre},2}$ implies that $y \notin I_p$, as by construction the right-hand side of each rule of $S_2(M)$ is irreducible. Hence, all the marked symbols $qx$, $qx_1$, and $qy$ occurring in the definition of $\hat{R}_{\text{pre}}$ are defined.

(b) $\hat{R}_{\text{suf}} = \{ A\$ \rightarrow A\$ \mid A \in \Gamma \setminus (I_0 \cup I_s) \} \cup R_{\text{suf},1} \cup \{ xs \rightarrow ys \mid (xs \rightarrow ys) \in R_{\text{suf},2} \text{ and } y \in \Gamma \} \cup \{ xs \rightarrow \$ \mid (xs \rightarrow \$) \in R_{\text{suf},2} \} \cup \{ xsb \rightarrow ys \mid (xb \rightarrow y) \in R_{\text{inf},2}, b, y \in \Gamma \setminus (I_0 \cup I_s) \} \cup \{ xsb \rightarrow \$ \mid (xb \rightarrow y) \in R_{\text{inf},2}, b \in \Gamma \setminus (I_0 \cup I_s), y \in I_s \} \cup \{ xsb \rightarrow \$ \mid (xb \rightarrow \lambda) \in R_{\text{inf},2}, b \in \Gamma \setminus (I_0 \cup I_s) \}.

As above, $(xs \rightarrow ys) \in R_{\text{suf},2}$ implies that $y \notin I_s$. Hence, all the marked symbols $xs$, $bs$, and $ys$ occurring in the definition of $\hat{R}_{\text{suf}}$ are defined.

(c) $\hat{R}_{\text{bif}} = R_{\text{bif}}' \cup \{ qawb\$ \rightarrow Y, qawb\$ \rightarrow Y, qawb\$ \rightarrow Y \mid a \in \Gamma \setminus (I_0 \cup I_p), w \in \Gamma^*, b \in \Gamma \setminus (I_0 \cup I_s), 1 \leq |awb| \leq 2 \cdot \rho, \operatorname{nf}(qawb\$) = Y \} \cup \{ qawb \rightarrow N, qawb \rightarrow N, qawb \rightarrow N \mid a \in \Gamma \setminus (I_0 \cup I_p), w \in \Gamma^*, b \in \Gamma \setminus (I_0 \cup I_s), 1 \leq |awb| \leq 2 \cdot \rho, \operatorname{nf}(qawb) \neq Y \}.

Here, for each letter $A \in \Gamma$, the rules $(qA\$ \rightarrow Z)$, $(qA\$ \rightarrow Z)$, and $(qA_s \rightarrow Z)$ are in $\hat{R}_{\text{bif}}$ for some symbol $Z \in \{Y, N\}$, provided $qA$ and/or $A_s$ are defined.

Then $G$ is a finite generalized monadic string-rewriting system, and it is shown easily that $G$ is weight-reducing. Below we will prove that $G$ is also confluent, and that the tuple $(G, \{A, S, Y\})$ specifies the language $L(M)$. In preparation for this proof we first relate reduction sequences of $S_2(M)$ to reduction sequences of $G$ and vice versa.

**Claim 4.9.** Let $A, B \in \Gamma$ and $u, v \in \Gamma^*$ such that $qAu \Rightarrow_S^{*} S_2(M) qBv$ and $qBv \in \text{IRR}(S_2(M))$. Then $qAu \Rightarrow^{*} G qBv$, and in addition, if $A \notin I_0 \cup I_p$, then also $qAu \Rightarrow^{*} G qBv$.

**Proof.** We proceed by Noetherian induction based on the well-founded partial ordering $\Rightarrow_S^{+} S_2(M)$ on $q \cdot \Gamma^*$. If $qAu$ is irreducible mod $S_2(M)$, then $Au = Bv$. Hence, $A = B \notin I_0 \cup I_p$, and $qAu \Rightarrow G qAu = qBv$ follows.

Now assume that $qAu \Rightarrow S_2(M) qDw$, where $D \in \Gamma$ and $w \in \Gamma^*$. Then $qDw \Rightarrow^{*} S_2(M) qBv$, as $S_2(M)$ is confluent. Hence, from the induction hypothesis we know that $qDw \Rightarrow G qBv$ holds, that also $qDw \Rightarrow^{*} G qBv$ holds, if $D \notin I_0 \cup I_p$.

If $qAu$ is rewritten into $qDw$ by applying a rule $(x \rightarrow y) \in R_{\text{inf},2}$ to a factor of $u$, then $A = D$ is not touched in this step, and $qAu \Rightarrow G qAw = qDw$.

If $qAu$ is rewritten into $qDw$ by applying a rule $(qAx \rightarrow qy) \in R_{\text{pre},2}$, then $u = xz$ for some $z \in \Gamma^*$, $yz = Dw$, and $A \notin I_0 \cup I_p$. Hence, $(qA \rightarrow qA) \in \hat{R}_{\text{pre}}$. 

76
F. OTTO ET AL.
and either $(Ax \rightarrow Ay) \in \hat{R}_{\text{pre}}$ (if $y \in \Gamma$) or $(Ax \rightarrow a) \in \hat{R}_{\text{pre}}$ (if $y = \lambda$). Thus, $\mathcal{A}u = \mathcal{A}xz \Rightarrow_G \mathcal{A}xz \Rightarrow_G \mathcal{A}yw = \mathcal{A}Dw$ in the former case, and $\mathcal{A}u = \mathcal{A}xz \Rightarrow_G \mathcal{A}xz \Rightarrow_G \mathcal{A}z = \mathcal{A}Dw$, in the latter case.

If $\mathcal{A}u$ is rewritten into $\mathcal{A}Dw$ by applying a rule $(Ax \rightarrow y) \in R_{\text{inf,2}}$ to the prefix $Ax$ of $Au$, then $u = xz$ for some $z \in \Gamma^*$, $yz = Dw$, and $A \not\in \Gamma_0$. If $A \not\in \Gamma_p$, either, then $\hat{R}_{\text{pre}}$ contains the rule $(Ax \rightarrow Ay)$, if $y \in \Gamma \setminus (\Gamma_0 \cup \Gamma_p)$, and it contains the rule $(Ax \rightarrow a)$, if $y \in \Gamma_p \cup \{\lambda\}$. Hence, this case is analogous to the previous one. If, however, $A \in \Gamma_p$, then the situation is different, as in this case $\hat{R}_{\text{pre}}$ does not contain any prefix rule that is derived from the infix rule $(Ax \rightarrow y)$. However, $(\mathcal{A}A \rightarrow \mathcal{A})$ is a rule of $R_{\text{pre,1}}$, and hence, we obtain $\mathcal{A}u = \mathcal{A}xz \Rightarrow_{S_2(M)} \mathcal{A}xz$, and $z \in \Gamma^*$, $\mathcal{A}z = \mathcal{A}Dw$, in the latter case.

If $\mathcal{A}u$ is rewritten into $\mathcal{A}Dw$ by applying a rule $(Ax \rightarrow y) \in R_{\text{inf,2}}$ to the prefix $Ax$ of $Au$, then $u = xz$ for some $z \in \Gamma^*$, $yz = Dw$, and $A \not\in \Gamma_0$. If $A \not\in \Gamma_p$, either, then $\hat{R}_{\text{pre}}$ contains the rule $(Ax \rightarrow Ay)$, if $y \in \Gamma \setminus (\Gamma_0 \cup \Gamma_p)$, and it contains the rule $(Ax \rightarrow a)$, if $y \in \Gamma_p \cup \{\lambda\}$. Hence, this case is analogous to the previous one. If, however, $A \in \Gamma_p$, then the situation is different, as in this case $\hat{R}_{\text{pre}}$ does not contain any prefix rule that is derived from the infix rule $(Ax \rightarrow y)$. However, $(\mathcal{A}A \rightarrow \mathcal{A})$ is a rule of $R_{\text{pre,1}}$, and hence, we obtain $\mathcal{A}u = \mathcal{A}xz \Rightarrow_{S_2(M)} \mathcal{A}xz$, and as $S_2(M)$ is confluent, we have $\mathcal{A}xz \Rightarrow_{S_2(M)} \mathcal{A}Bv$. Hence, by induction hypothesis $\mathcal{A}xz \Rightarrow_{S_2(M)} \mathcal{A}Bv$. It follows that $\mathcal{A}u = \mathcal{A}xz \Rightarrow_G \mathcal{A}xz \Rightarrow_G \mathcal{A}Bv$. \hfill \Box

In the same manner also the following technical result can be derived.

**Claim 4.10.** Let $A \in \Gamma$ and $u \in \Gamma^*$ such that $\mathcal{A}u \Rightarrow_{S_2(M)} \mathcal{A}$. Then $\mathcal{A}u \Rightarrow_{S_2(M)} \mathcal{A}$, and in addition, if $A \not\in \Gamma_0 \cup \Gamma_p$, then also $\mathcal{A}u \Rightarrow_{S_2(M)} \mathcal{A}$.

By symmetry, statements corresponding to Claims 4.9 and 4.10 also hold for $S_2(M)$-reductions that begin with a word of the form $uA\$. These technical results will now be used to prove the following fundamental property of $G$.

**Claim 4.11.** For all $w \in \Gamma^*$, if $\mathcal{A}w \Rightarrow_{S_2(M)} Y$, then $\mathcal{A}w \Rightarrow_{S_2(M)} Y$.

**Proof.** Let $w \in \Gamma^*$ such that $\mathcal{A}w \Rightarrow_{S_2(M)} Y$ holds. If $|w| \leq 2 \cdot \rho$, then we have $(\mathcal{A}w \rightarrow Y) \in R_{\text{bif}} \subseteq \hat{R}_{\text{bif}}$, and hence, $\mathcal{A}w \Rightarrow_Y Y$ holds. If $|w| > 2 \cdot \rho$, then the above $S_2(\mathcal{M})$-reduction can be replaced by a reduction of the following form:

$$
\mathcal{A}w \Rightarrow_{R_{\text{pre}} \cup R_{\text{inf}}} \mathcal{A}w \Rightarrow_{R_{\text{inf}} \cup R_{\text{inf}}} \mathcal{A}w \Rightarrow_{R_{\text{inf}}} Y,
$$

where $u$ is irreducible mod $R_{\text{pre}} \cup R_{\text{inf}}$, and $|v| < 2 \cdot \rho$.

If $u = \lambda$, then by Claim 4.10 we have $\mathcal{A}w \Rightarrow_{S_2(M)} \mathcal{A}w \Rightarrow_{S_2(M)} Y$. If $u = Ax$ for some $A \in \Gamma$ and $x \in \Gamma^*$, then by Claim 4.9, $\mathcal{A}w \Rightarrow_{S_2(M)} \mathcal{A}Az$. The reduction $\mathcal{A}w = \mathcal{A}Ax \Rightarrow_{R_{\text{inf}} \cup R_{\text{inf}}} \mathcal{A}Az \Rightarrow_{R_{\text{inf}} \cup R_{\text{inf}}} \mathcal{A}w$, can be written as

$$
\mathcal{A}Ax \Rightarrow_{R_{\text{inf}} \cup R_{\text{inf}}} \mathcal{A}Az \Rightarrow_{R_{\text{inf}} \cup R_{\text{inf}}} \mathcal{A}w,
$$

where $z\$ is the irreducible descendant of $x\$ mod $R_{\text{inf}} \cup R_{\text{inf}}$. If $Az\$ is reducible mod $R_{\text{inf}} \cup R_{\text{inf}}$, then it follows from the form of the rules of $S_2(M)$ that $|z| \leq \rho$. If $z = \lambda$, then we have $\mathcal{A}z \Rightarrow_{S_2(M)} \mathcal{A}z \Rightarrow_{S_2(M)} Y$ by the suffix variant of Claim 4.10, which yields $\mathcal{A}Az \Rightarrow_{S_2(M)} \mathcal{A}Az \Rightarrow_{S_2(M)} Y$. Further, if $z = z'B$ for some $B \in \Gamma$ and $z' \in \Gamma^*$, then the suffix variant of Claim 4.9 shows that $\mathcal{A}z' \Rightarrow_{S_2(M)} z'B_\$, and hence, we obtain $\mathcal{A}Az \Rightarrow_{S_2(M)} \mathcal{A}Az' \Rightarrow_{S_2(M)} \mathcal{A}z'B_\$ and $\mathcal{A}z'B_\Rightarrow_{S_2(M)} Y$, as $(\mathcal{A}z'B_\rightarrow Y) \in \hat{R}_{\text{bif}}$. This completes the proof of Claim 4.11. \hfill \Box
From the construction of the rules of $G$ it follows immediately that, for each rule $(x \rightarrow y) \in G$, $\psi(x) \Rightarrow^*_{S_2(M)} \psi(y)$ holds. Hence, together with Claim 4.11 this yields the following statement.

Claim 4.12. For all $w \in \Gamma^*$, $\varphi w\$ \Rightarrow^*_G Y$ if $\varphi w\$ \Rightarrow^*_S(M) Y$.

As noted before, $L(M)$ is the McNaughton language that is specified by the tuple $(S_2(M), \varphi, \$, Y). Thus, Claim 4.12 shows that $L(M) = L(G, \varphi, \$, Y)$ holds. To complete the proof of Theorem 4.6 it remains to establish the following claim.

Claim 4.13. The string-rewriting system $G$ is confluent.

Proof. Obviously, the subsystems $\hat{R}_{\text{bif}}$ and $R'_{\text{inf}}$ are confluent. Furthermore, it follows as in the proof of Claim 4.8 above that all overlaps between a rule of $\hat{R}_{\text{bif}}$ and a rule from $\hat{R}_{\text{pre}} \cup \hat{R}_{\text{suf}} \cup R'_{\text{inf}}$ resolve. Thus, it remains to consider the case that a rule from $\hat{R}_{\text{pre}} \cup \hat{R}_{\text{suf}}$ overlaps with a rule from $\hat{R}_{\text{pre}} \cup \hat{R}_{\text{suf}} \cup R'_{\text{inf}}$. As $\hat{R}_{\text{pre}}$ and $\hat{R}_{\text{suf}}$ are defined in a symmetric way, it suffices to consider the case of a rule of $\hat{R}_{\text{pre}}$.

If $(\varphi A \rightarrow \varphi A) \in \hat{R}_{\text{pre}}$, then $A \in \Gamma \setminus (\Gamma_0 \cup \Gamma_\rho)$. Hence, $\varphi A$ does not overlap with the left-hand side of any other rule of $\hat{R}_{\text{pre}}$. If $(Ax \rightarrow y) \in R'_{\text{inf}}$, then $(\varphi Ax \rightarrow \varphi y) \in \hat{R}_{\text{pre}}$, if $y \in \Gamma \setminus (\Gamma_0 \cup \Gamma_\rho)$, and $(\varphi Ax \rightarrow \varphi) \in \hat{R}_{\text{pre}}$, if $y \in \Gamma_\rho \cup \{\lambda\}$. In the former case, $(\varphi y \rightarrow \varphi y) \in \hat{R}_{\text{pre}}$, and in the latter case we have $\varphi y \Rightarrow^*_{R_{\text{pre},1}} \varphi$, that is, the critical pair $(\varphi Ax, \varphi y)$ resolves. Finally, if $(\varphi A\$ \rightarrow A\$) \in \hat{R}_{\text{suf}}$, then $A \not\in \Gamma_s$, either. Hence, $\varphi A\$ $\Rightarrow_G \varphi A\$ $\Rightarrow_G \varphi A\$, that is, $(\varphi A\$, $\varphi A\$) is a critical pair of $G$.

However, $\hat{R}_{\text{bif}}$ then contains the two rules $(\varphi A\$ $\rightarrow Z)$ and $(\varphi A\$ $\rightarrow Z)$ for some value of $Z \in \{Y, N\}$, which shows that the above critical pair resolves.

Next assume that $(\varphi Ax \rightarrow \varphi y) \in \hat{R}_{\text{pre}}$ and that $(\varphi Ax_1 \rightarrow \varphi y') \in \hat{R}_{\text{pre}}$, where $x = x_1x_2$ for some word $x_2 \in \Gamma^*$. Then $\varphi y \Rightarrow_G \varphi Ax = \varphi Ax_1x_2 \Rightarrow_G \varphi y'x_2$, that is, we obtain the critical pair $(\varphi y, \varphi y'x_2)$. As $S_2(M)$ is confluent, $\varphi y$ and $\varphi y'x_2$ have a common irreducible descendant mod $R'_{\text{pre}} \cup R'_{\text{inf}}$, and hence, we see from Claims 4.9 and 4.10 that this critical pair resolves.

In the same manner it can be shown that all other critical pairs that result from overlapping a rule of $\hat{R}_{\text{pre}}$ with a rule of $\hat{R}_{\text{pre}} \cup R'_{\text{inf}}$ resolve, too. Finally, if $(\varphi Ax \rightarrow \varphi y) \in \hat{R}_{\text{pre}}$ and $(wB_\$ \rightarrow z\$) \in \hat{R}_{\text{suf}}$ such that $x = x_1x_2$ and $w = x_2x_3$ for a non-empty word $x_2 \in \Gamma^*$, then we obtain the critical pair $(\varphi yx_3B_\$, $\varphi Ax_1z\$). However, as $|yx_3B|, |Ax_1z| \leq |Ax_1x_2| \leq 2 \cdot \rho$, we see that $\hat{R}_{\text{bif}}$ contains the two rules $(\varphi yx_3B_ \rightarrow Z)$ and $(\varphi Ax_1z\$ $\rightarrow Z)$ for the appropriate symbol $Z \in \{Y, N\}$. Thus, also this critical pair resolves, and all other critical pairs that result from overlapping a rule of $\hat{R}_{\text{pre}}$ with a rule of $\hat{R}_{\text{suf}}$ can be shown to resolve in the same way. It follows that the system $G$ is confluent.

Thus, $G$ is actually a finite, weight reducing, generalized monadic string-rewriting system that is confluent, and hence, Claim 4.12 shows that $L(M)$ is indeed a confluent generalized monadic McNaughton language.
If $M = (\Sigma, \Gamma, I)$ is a confluent lc-R-automaton of type $\mathcal{R}_2$, then it turns out that the string-rewriting system $G$ constructed in the proof above is finite, monadic, and confluent. Hence, we obtain the following inclusion.

**Theorem 4.14.** $\mathcal{L}(\text{lc-R[con-$\mathcal{R}_2$]}) \subseteq \text{con-mon-McNL}$.

We continue with the converse of Theorem 4.6.

**Theorem 4.15.** $\text{con-gen-mon-McNL} \subseteq \mathcal{L}(\text{lc-R[con-$\mathcal{R}_2'$]})$.

**Proof.** Let $L = L(S, t_1, t_2, Y)$, where $S$ is a finite, confluent, generalized monadic string-rewriting system on a finite alphabet $\Gamma$ that properly contains the input alphabet $\Sigma$, $t_1, t_2 \in (\Gamma \setminus \Sigma)^*$ and $Y \in \Gamma \setminus \Sigma$ are irreducible mod $S$, and $L \subseteq \Gamma^*$. According to [17] we can assume that $S$ is terminating and interreduced, that is, for each rule $(x \rightarrow y)$ of $S$, $y \in \text{IRR}(S)$ and $x \in \text{IRR}(S \setminus \{x \rightarrow y\})$. Furthermore, we can assume that $S$ does not contain any rules with right-hand side $\lambda$. Actually, it is quite easy to provide a weight function $g$ such that $S$ is weight-reducing with respect to $g$. As $S$ is terminating and confluent, each word $w \in \Gamma^*$ has a unique normal form in $\text{IRR}(S)$, which we denote by $\text{nf}(w)$ as before.

From $S$ we now construct a confluent lc-R-automaton $M$ of type $\mathcal{R}_2'$ such that $L(M) = L$ holds. This automaton will work on an extended alphabet $\hat{\Gamma}$ that is defined as follows:

$$
\hat{\Gamma} = \Gamma \cup \{ [\alpha], \langle \alpha \rangle \mid \alpha \in \text{IRR}(S), |\alpha| \leq \rho + 1 \} 
\cup \{ [\alpha] \mid \alpha \in \text{IRR}(S), |\alpha| \leq 3 \cdot \rho + 2 \},
$$

where $\rho = \max \{|w| \mid w \in \{t_1, t_2\} \cup \{ x \mid (x \rightarrow y) \in S \} \}$. A symbol of the form $[\alpha]$ will be used to encode the normal form $\alpha$ of a word beginning with $t_1$, a symbol of the form $\langle \alpha \rangle$ will be used to encode the normal form $\alpha$ of a word ending in $t_2$, and finally, a symbol of the form $[\alpha]$ will be used to encode the normal form $\alpha$ of a word that begins with $t_1$ and ends in $t_2$ (see the construction below).

The lc-R-automaton $M = (\Sigma, \hat{\Gamma}, I)$ is defined by the following sets of instructions, where $I = I_{\text{int}} \cup I_{\text{pre}} \cup I_{\text{suf}} \cup I_{\text{bif}}$:

(a) $I_{\text{int}} = \{ (\lambda \mid x \rightarrow y \mid \lambda) \mid (x \rightarrow y) \in S \}$.

Observe that $|x| \geq |y| = 1$ holds for each of these instructions.

(b) $I_{\text{pre}} = \{ (\# \mid a \rightarrow [\text{nf}(t_1 a)] \mid \lambda) \mid a \in \Gamma \cap \text{IRR}(S) \} \cup
\{ (\# \mid [\alpha] u \rightarrow [\text{nf}(\alpha u)] \mid \lambda) \mid u \in \text{IRR}(S), 1 \leq |u| \leq \rho - 1, \text{ and } |\text{nf}(\alpha u)| \leq \rho + 1 \},$

(c) $I_{\text{suf}} = \{ (\lambda \mid b \rightarrow [\text{nf}(b t_2)] \mid \$) \mid b \in \Gamma \cap \text{IRR}(S) \} \cup
\{ (\lambda \mid u [\alpha] \rightarrow [\text{nf}(u \alpha)] \mid \$) \mid u \in \text{IRR}(S), 1 \leq |u| \leq \rho - 1, \text{ and } |\text{nf}(u \alpha)| \leq \rho + 1 \},$

(d) $I_{\text{bif}} = \{ (\# \mid [\alpha] u \rightarrow [\text{nf}(\alpha u t_2)] \mid \$) \mid u \in \text{IRR}(S), 0 \leq |u| \leq \rho + 1 \} \cup
\{ (\# \mid u [\alpha] \rightarrow [\text{nf}(t_1 u \alpha)] \mid \$) \mid u \in \text{IRR}(S), 0 \leq |u| \leq \rho + 1 \} \cup
\{ (\# \mid [\alpha] u \gamma \rightarrow [\text{nf}(\alpha u \gamma)] \mid \$) \mid u \in \text{IRR}(S), 0 \leq |u| \leq \rho \} \cup
\{ (\# \mid Y \rightarrow \lambda \mid \$) \}.$
It is easily checked that $M$ is of type $\mathcal{R}_2$. Observe that all subsystems of $I$ will in
general contain some length-preserving instructions. The proof of Theorem 4.15
will now be completed by establishing the following two claims.

**Claim 4.16.** $L(M) \cong L$.

**Claim 4.17.** The string-rewriting system $R(M)$ obtained from $I$ is confluent.

*Proof of Claim 4.16.* Let $w \in L$ such that $|w| \geq 1$. Then $w \in \Sigma^+$, and $t_1 wt_2 \Rightarrow^*_S Y$.
As $S$ does not contain any rule with empty right-hand side, this reduction sequence
can be factored as $t_1 wt_2 \Rightarrow^*_S t_1 \mathrm{nf}(w)t_2 \Rightarrow^*_S x \Rightarrow^*_S Y$ for some rule $(x \rightarrow Y)$ of $S$.
Obviously, the automaton $M$ can execute the sequence of cycles $w \vdash^c_M \mathrm{nf}(w)$ using
the instructions of $I_{\mathrm{inf}}$.

If $\mathrm{nf}(w) = a \in \Gamma$, then $\mathrm{nf}(w) = a \vdash^c_M [\mathrm{nf}(t_1 a)] \vdash^c_M [Y] \vdash^c_M \lambda$, as $\mathrm{nf}(t_1 at_2) = Y$.
If $\mathrm{nf}(w) = avb$, where $a, b \in \Gamma$ and $v \in \Gamma^*$, then $\mathrm{nf}(w) = avb \vdash^c_M [\mathrm{nf}(t_1 a)]vb \vdash^c_M
[\mathrm{nf}(t_1 a)]v[\mathrm{nf}(bt_2)]$. If $|v| \leq \rho$, then $(\{[\mathrm{nf}(t_1 a)]v[\mathrm{nf}(bt_2)] \rightarrow [Y] \}$ $\in I_{\mathrm{bif}}$. Fi-
ally, if $|v| > \rho$, then $\mathrm{nf}(t_1 a)v$ or $v[\mathrm{nf}(bt_2)$ (or both) are reducible mod $S$, as
$\mathrm{nf}(t_1 a)v[\mathrm{nf}(bt_2)$ reduces to the word $x$ mod $S$, and $|x| \leq \rho$. Hence, we obtain
$[\mathrm{nf}(t_1 a)]v[\mathrm{nf}(bt_2)] \vdash^c_M [\mathrm{nf}(t_1 avt_1)]v[\mathrm{nf}(v_2 bt_2)]$ using instructions from $I_{\mathrm{pre}}$ and/or
$I_{\mathrm{suf}}$, where $v = v_1 v_2 v_3$ and $|v_2| \leq \rho$. As $\mathrm{nf}(t_1 avt_1)v[\mathrm{nf}(v_2 bt_2) \Rightarrow^*_S Y$, we can now
use an instruction from $I_{\mathrm{bif}}$ to obtain $[\mathrm{nf}(t_1 avt_1)]v[\mathrm{nf}(v_2 bt_2)] \vdash^c_M [Y] \vdash^c_M \lambda$. Thus,
in each case $w$ is accepted by $M$, which proves that $L \subseteq L(M)$ holds.

Conversely, let $w \in L(M)$ such that $|w| \geq 1$. Then $w \vdash^c_M \lambda$, and as $M$ has only
a single instruction with empty right-hand side, we see that $w \vdash^c_M [Y]$ holds. We
define a morphism $\psi : (\hat{\Gamma} \cup \{\delta, \emptyset\})^* \rightarrow (\hat{\Gamma} \cup \{\delta, \emptyset\})^*$ by taking $a \rightarrow a$ $(a \in \Gamma \cup \{\delta, \emptyset\}),
[\alpha] \rightarrow \alpha, [\alpha] \rightarrow \alpha, \text{ and } [\alpha] \rightarrow \alpha$ for all values of $\alpha$. From the definition of the
instructions in $I$, the following properties are easily derived, where $u, v, \alpha, \gamma \in \Gamma^*$:

(a) If $u \vdash^c_M [\alpha]v$, then $t_1 u \Rightarrow^*_S \alpha v$.
(b) If $u \vdash^c_M v[\alpha]$, then $ut_2 \Rightarrow^*_S v\alpha$.
(c) If $[\alpha]u \vdash^c_M [\alpha]v[\gamma]$, then $\alpha ut_2 \Rightarrow^*_S \alpha v\gamma$.
(d) If $u[\gamma] \vdash^c_M [\alpha]v[\gamma]$, then $t_1 u\gamma \Rightarrow^*_S \alpha v\gamma$.
(e) If $u \vdash^c_M v$, where either both, $u$ and $v$, begin with a letter of the form $[\alpha]$
or none of them does, and where either both end with a letter of the form
$[\gamma]$ or none of them does, then $\psi(u) \Rightarrow^*_S \psi(v)$.
(f) If $[\alpha]u \vdash^c_M [\gamma]$, then $\alpha ut_2 \Rightarrow^*_S \gamma$.
(g) If $u[\alpha] \vdash^c_M [\gamma]$, then $t_1 u\alpha \Rightarrow^*_S \gamma$.
(h) If $u[\alpha] \vdash^c_M [v]$, then $\alpha uv\gamma \Rightarrow^*_S v$.

By induction on the number of cycles in the computation $w \vdash^c_M [Y]$ it can now be
shown easily that $t_1 wt_2 \Rightarrow^*_S Y$ holds, which implies that $w \in L$.

Thus, in summary we have shown that $L(M) = L \cup \{\lambda\}$. This completes the
proof of Claim 4.16. \hfill $\square$

*Proof of Claim 4.17.* The string-rewriting system $R(M)$ consists of four subsystems
$R_{\mathrm{inf}}, R_{\mathrm{pre}}, R_{\mathrm{suf}}$, and $R_{\mathrm{bif}}$ that are obtained by turning the corresponding sets of
instructions into rewrite rules.
By our hypothesis, the subsystem $R_{\text{inf}}$, which coincides with the string-rewriting system $S$, is confluent.

Next we consider the subsystem $R_{\text{pre}}$. Assume that $(\varphi[\alpha]u \rightarrow \varphi[\text{nf}(\alpha u)])$ and $(\varphi[\alpha]uv \rightarrow \varphi[\text{nf}(\alpha uv)])$ are both rules of $R_{\text{pre}}$, where $1 \leq |u| < |uv| \leq \rho - 1$. Then $(\varphi[\text{nf}(\alpha u)]v \rightarrow \varphi[\text{nf}(\alpha uv)])$ is also a member of $R_{\text{pre}}$, which shows that the critical pair $(\varphi[\text{nf}(\alpha u)]v, \varphi[\text{nf}(\alpha uv)])$ resolves. Hence, the subsystem $R_{\text{pre}}$ is confluent, and by symmetry also $R_{\text{suf}}$ is confluent. Finally, as there are no non-trivial overlaps between rules of $R_{\text{bif}}$, also $R_{\text{bif}}$ is confluent.

Thus, it remains to consider those critical pairs that result from overlapping rules of different subsystems. Let $(\varphi a \rightarrow \varphi[\text{nf}(t_1a)]) \in R_{\text{pre}}$ and $(ax \rightarrow y) \in R_{\text{inf}}$. From these two rules we obtain the critical pair $(\varphi[\text{nf}(t_1a)]x, \varphi y)$. As $S$ is strictly monadic and interreduced, we have $y \in \Gamma \cap \text{IRR}(S)$. Hence, $R_{\text{pre}}$ contains the rule $(\varphi y \rightarrow \varphi[\text{nf}(t_1y)])$. Further, $x \in \text{IRR}(S)$, $|x| \leq \rho - 1$, and $\text{nf}(t_1ax) = \text{nf}(t_1y)$, that is, $|\text{nf}(t_1ax)| \leq \rho + 1$. Thus, $R_{\text{pre}}$ also contains the rule $(\varphi[\text{nf}(t_1a)]x \rightarrow \varphi[\text{nf}(t_1ax)])$, which shows that the critical pair above resolves.

If $(\varphi[\alpha]uv \rightarrow \varphi[\text{nf}(\alpha uv)]) \in R_{\text{pre}}$ and $(uv \rightarrow y) \in R_{\text{inf}}$ for some non-empty word $v$, then we obtain the critical pair $(\varphi[\text{nf}(\alpha uv)]w, \varphi[\alpha uy])$. As $S$ is confluent, we know that $\text{nf}(\alpha uvw) = \text{nf}(\alpha uy)$. So let this normal form be $z = z_1 \ldots z_k$, where $k \geq 1$ and $z_1, \ldots, z_k \in \Gamma$. If $k \leq \rho + 1$, then $(\varphi[\text{nf}(\alpha uv)]w \rightarrow \varphi[z]) \in R_{\text{pre}}$ and $(\varphi[\alpha]nf(uy) \rightarrow \varphi[z]) \in R_{\text{pre}}$, and so the critical pair resolves. If, however, $k > \rho + 1$, then $\varphi[\text{nf}(\alpha uv)]w \Rightarrow_{R_{\text{pre}}} \varphi[z_1 \ldots z_{\rho+1}]z_{\rho+2} \ldots z_k$ and $\varphi[\alpha]uy \Rightarrow_{R_{\text{inf}}} \varphi[\alpha]nf(uy) \Rightarrow_{R_{\text{pre}}} \varphi[z_1 \ldots z_{\rho+1}]z_{\rho+2} \ldots z_k$, as for all rules $(c_1 | \ell \rightarrow r | c_2) \in I$, we have $|r| \leq 1$. Thus, the critical pair above also resolves in this case. It follows that the subsystem $R_{\text{pre}} \cup R_{\text{inf}}$ is confluent, and by symmetry also the subsystem $R_{\text{suf}} \cup R_{\text{inf}}$ is confluent.

Finally, let $(\varphi a \rightarrow \varphi[\text{nf}(t_1a)]) \in R_{\text{pre}}$. If $(a\varnothing \rightarrow \langle \text{nf}(at_2)\rangle \varnothing) \in R_{\text{suf}}$, this yields the critical pair $(\varphi[\text{nf}(t_1a)]\varnothing, \varphi[\text{nf}(at_2)]\varnothing)$. As $\text{nf}(t_1a)t_2 = \text{nf}(t_1 \text{nf}(at_2))$, we see that $R_{\text{bif}}$ contains the rules $(\varphi[\text{nf}(t_1a)]\varnothing \rightarrow \varphi[\text{nf}(t_1at_2)]\varnothing)$ and $(\varphi[\text{nf}(at_2)]\varnothing \rightarrow \varphi[\text{nf}(t_1at_2)]\varnothing)$, that is, the critical pair resolves. Further, if $R_{\text{suf}}$ contains the rule $(\alpha u\langle \alpha \rangle \varnothing \rightarrow \langle \text{nf}(\alpha u)\rangle \varnothing)$, then the critical pair $(\varphi[\text{nf}(t_1a)]u\varnothing \alpha, \varphi[\text{nf}(\alpha u)]\varnothing)$ is obtained. As $|u| < \rho$, and as $\text{nf}(t_1a\alpha) = \text{nf}(t_1 \text{nf}(\alpha u))$, we see that this critical pair resolves mod $R_{\text{bif}}$. It follows analogously that also the other critical pairs that result from overlapping a rule of $R_{\text{pre}}$ with a rule of $R_{\text{suf}}$ resolve mod $R_{\text{bif}}$.

It remains to consider those critical pairs that result from overlapping a rule of $R_{\text{bif}}$ with a rule of the other three subsystems. First, observe that there are no such overlaps with rules of $R_{\text{inf}}$. Now let $(\varphi[\alpha]uv\varnothing \rightarrow \varphi[\text{nf}(\alpha ut_2)]\varnothing) \in R_{\text{bif}}$ and $(\varphi[\alpha]u \rightarrow \varphi[\text{nf}(\alpha u)]) \in R_{\text{pre}}$. These rules yield the critical pair $(\varphi[\text{nf}(\alpha ut_2)]\varnothing, \varphi[\text{nf}(\alpha u)]\varnothing)$. But then $R_{\text{bif}}$ also contains the rule $(\varphi[\text{nf}(\alpha u)]\varnothing \rightarrow \varphi[\text{nf}(\alpha ut_2)]\varnothing)$, which resolves this critical pair. All other critical pairs that result from an overlap with a rule of $R_{\text{pre}}$ also resolve mod $R_{\text{bif}}$, and by symmetry the same holds for the critical pairs that result from overlaps with a rule of $R_{\text{suf}}$. Thus, we have shown that the system $R(M)$ is indeed confluent.

□

This completes the proof of Theorem 4.15.
Together Theorems 4.6 and 4.15 yield the following equivalence.

**Corollary 4.18.** \( \mathcal{L}(\text{lc-R[con-} \mathcal{R}_3') = \text{con-gen-mon-McNL} \).  

It currently remains open whether the converse of Theorem 4.14 holds as well. Observe that the lc-R-automaton \( M \) constructed in the proof of Theorem 4.15 contains length-preserving instructions even when the string-rewriting system \( S \) is length-reducing.

Finally, we turn to the confluent lc-R-automata of types \( \mathcal{R}_3' \) and \( \mathcal{R}_3 \). If \( M \) is an lc-R[con-} \mathcal{R}_3'-automaton, then \( L(M) \) is a regular language by Theorem 3.9.  

Conversely, if \( L \subseteq \Sigma^* \) is a regular language, then there exists a deterministic finite-state acceptor \( A = (Q, \Sigma, q_0, F, \delta) \) that accepts \( L^R \). We define an lc-R-automaton \( M = (\Sigma, Q \cup \Sigma, I) \) as follows, where \( a, b \in \Sigma \) and \( q, q' \in Q \):

\[
I = \{ (q \mid ab \rightarrow q \mid \lambda) \mid \delta(q_0, ab) = q \} \cup \{ (q \mid qa \rightarrow q' \mid \lambda) \mid \delta(q, a) = q' \} \cup \{ (q \mid q \rightarrow \lambda \mid \$) \mid q \in F \} \cup \{ (q \mid a \rightarrow \lambda \mid \$) \mid a \in \Sigma \cap L^R \}.
\]

It is easily seen that \( L(M) = L^R \), and that the string-rewriting system \( R(M) \) is confluent. By taking \( M' = (\Sigma, Q \cup \Sigma, I') \), where \( I' = \{ (\lambda \mid u^R \rightarrow v^R \mid \$) \mid (q \mid u \rightarrow v \mid \lambda) \in I \} \cup \{ (q \mid u^R \rightarrow v^R \mid \$) \mid (q \mid u \rightarrow v \mid \$) \in I \} \), we obtain a confluent lc-R-automaton of type \( \mathcal{R}_3 \) that accepts the language \( L \). Thus, we have the following characterization.

**Theorem 4.19.** \( \mathcal{L}(\text{lc-R[con-} \mathcal{R}_3') = \mathcal{L}(\text{lc-R[con-} \mathcal{R}_3]) = \text{REG} \).

The diagram in Figure 2 summarizes our results on confluent lc-R-automata.
5. Concluding remarks

We have studied the relationship between various classes of limited context restarting automata on the one hand and certain McNaughton families of languages on the other hand. We have seen that the class $GCSL$ of growing context-sensitive languages is an upper bound for all the types of limited context restarting automata considered, and that this upper bound is attained by three classes of these automata. Under the additional requirement of confluence, the Church–Rosser languages form an upper bound, which is reached by the three most general types of these automata. On the other hand, for the most restricted types of lc-R-automata, we just obtain the regular languages, both in the confluent and the non-confluent case. However, for most of the intermediate systems, the question for an exact characterization of the classes of languages accepted remains open. In fact, for these types of systems it even remains unsolved whether weight-reducing lc-R-automata are more expressive than length-reducing lc-R-automata of the same type.

References


Communicated by M. Holzer.

Received January 28, 2013. Accepted January 8, 2014.