RAIRO-Theor. Inf. Appl. 48 (2014) 61–84
DOI: 10.1051/ita/2014001

Available online at:
www.rairo-ita.org

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

Friedrich Otto[1], Peter Černo[2] and František Mráz[2]

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

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 $L_{\text{Glad}} = \{ wcw^R cw \mid w \in \{a,b\}^* \}$, 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 Δ-*clearing restarting automata* and the Δ^*-*clearing restarting automata* that can use a marker symbol Δ 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 Δ- or Δ^*-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 λ, that is, if and only if w can be rewritten to λ 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 GCSL of *growing context-sensitive languages* [7,11], the class CRL of *Church–Rosser languages* [18], and the class con-gen-mon-McNL of *confluent generalized monadic McNaughton languages* [17] will appear prominently.

Notation. In the following all alphabets considered will be finite. For an alphabet Σ, Σ^* is used to denote the set of all words over Σ including the empty word λ. 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 $L^R = \{ w^R \mid w \in L \}$. 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 Σ. 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, $\mathcal{L}(\mathsf{A})$ 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 Σ consists of (finitely many) pairs of strings from Σ^*, called *rewrite rules*, which are written as $(\ell \to r)$. By $\mathrm{dom}(S)$ we denote the set $\mathrm{dom}(S) = \{ \ell \mid \exists r \in \Sigma^* : (\ell \to r) \in S \}$ of left-hand sides of rules of S. The *reduction relation* $\Rightarrow_S^*$ on Σ^* that is induced by S is the reflexive and transitive closure of the *single-step reduction relation*

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

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, $\Delta_S^*(u) = \{\, v \mid u \Rightarrow_S^* v \,\}$, $\nabla_S^*(v) = \{\, u \mid u \Rightarrow_S^* v \,\}$ 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 Σ^* that is induced by S. It is the smallest equivalence relation on Σ^* 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 $w \Rightarrow_S w_1 \Rightarrow_S w_2 \Rightarrow_S \ldots$,
- *weight-reducing*, if there exists a weight function g satisfying $g(\ell) > g(r)$ for each rule $(\ell \to r) \in S$,
- *length-reducing*, if $|\ell| > |r|$ for each rule $(\ell \to r) \in S$,
- *generalized monadic*, if $|\ell| \geq |r|$ and $|r| \leq 1$ for each rule $(\ell \to r) \in S$,
- *monadic*, if $|\ell| > |r|$ and $|r| \leq 1$ for each rule $(\ell \to r) \in S$,
- *confluent*, if, for all $u, v \in \Sigma^*$, $u \Leftrightarrow_S^* v$ 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 confluent (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 Σ. If there are two rules $(\ell \to r)$ and $(\ell' \to r')$ in S such that $\ell = u\ell' v$ for some $u, v \in \Sigma^*$, then the word ℓ 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: $uvw = \ell w \Rightarrow_S rw$ and $uvw = u\ell' \Rightarrow_S ur'$. 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 Γ strictly containing Σ, a finite string-rewriting system S on Γ, 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 Σ 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 Γ:

$$S = \{¢aaaa \to ¢aaF, Faa \to aF, F\$ \to \$, ¢aa\$ \to Y, ¢a\$ \to Y\}.$$

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 $L_{\text{expo}} \in$ con-lr-McNL. Concerning these families the following results are known.

Theorem 2.3 [3,17]**.**
(a) GCSL = wr-McNL = lr-McNL.
(b) CRL = con-wr-McNL = con-lr-McNL.
(c) CFL = gen-mon-McNL = mon-McNL.
(d) REG $\subsetneq$ con-mon-McNL $\subseteq$ con-gen-mon-McNL $\subsetneq$ symDCFL.

Here REG and CFL denote the classes of regular and context-free languages, and symDCFL = DCFL $\cap$ DCFL^R, that is, a language L belongs to 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 Σ is an input alphabet, Γ is a working alphabet

containing Σ, and I is a finite set of *instructions* of the form $(u\,|\,x \to y\,|\,v)$, where $x, y \in \Gamma^*$ such that $g(x) > g(y)$ for some weight function $g : \Gamma^* \to \mathbb{N}$, $u \in \{\lambda, ¢\} \cdot \Gamma^*$, and $v \in \Gamma^* \cdot \{\lambda, \$\}$. Here ¢ and \$ are the left and right sentinels, which are not elements of Γ.

The lc-R-automaton $M = (\Sigma, \Gamma, I)$ induces a reduction relation $\vdash_M^c$ on Γ^* that is defined as follows: for each $w, z \in \Gamma^*$, $w \vdash_M^c z$, if there exist words $w_1, w_2 \in \Gamma^*$ and an instruction $(u\,|\,x \to y\,|\,v)$ in I such that $w = w_1 x w_2$, $z = w_1 y w_2$, 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 $\vdash_M^{c^*}$ we denote the reflexive and transitive closure of $\vdash_M^c$. The language accepted by the lc-R-automaton M is $L(M) = \{\, w \in \Sigma^* \mid w \vdash_M^{c^*} \lambda \,\}$.

An lc-R-automaton M accepts exactly the set of input words which can be reduced to λ. Obviously, λ is in $L(M)$ for each lc-R-automaton M. Accordingly, if L is a language that does not contain λ 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 Σ are equal, denoted as $L \doteq L'$, 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 $I = \{(\lambda\,|\,acbb \to c\,|\,\lambda), (¢\,|\,c \to \lambda\,|\,\$)\}$. Then $aaacbbbbbb \vdash_M^c aacbbbb \vdash_M^c acbb \vdash_M^c c \vdash_M^c \lambda$, and so the word a^3cb^6 belongs to $L(M)$. It is easily seen that $L(M) = \{\, a^n cb^{2n} \mid n \geq 0 \,\}$.

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 $\mathcal{R}_0'$. 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 $M = (\Sigma, \Gamma, I)$ is of type

- $\mathcal{R}_0$, if I only contains instructions of the form $(u\,|\,x \to y\,|\,v)$, where $|x| > |y|$;
- $\mathcal{R}_1'$, if I only contains instructions of the form $(u\,|\,x \to y\,|\,v)$, where $|y| \leq 1$, and $x \in \Gamma^+$;
- $\mathcal{R}_1$, if I only contains instructions of the form $(u\,|\,x \to y\,|\,v)$, where $|y| \leq 1$, and $x \in \Gamma^+$ such that $|x| > |y|$;
- $\mathcal{R}_2'$, if I only contains instructions of the form $(u\,|\,x \to y\,|\,v)$, where $|y| \leq 1$, $u \in \{\lambda, ¢\}$, $v \in \{\lambda, \$\}$, and $x \in \Gamma^+$;
- $\mathcal{R}_2$, if I only contains instructions of the form $(u\,|\,x \to y\,|\,v)$, where $|y| \leq 1$, $u \in \{\lambda, ¢\}$, $v \in \{\lambda, \$\}$, and $x \in \Gamma^+$ such that $|x| > |y|$;
- $\mathcal{R}_3'$, if I only contains instructions of the form $(u\,|\,x \to y\,|\,\$)$, where $|y| \leq 1$, $u \in \{\lambda, ¢\}$, and $x \in \Gamma^+$;
- $\mathcal{R}_3$, if I only contains instructions of the form $(u\,|\,x \to y\,|\,\$)$, where $|y| \leq 1$, $u \in \{\lambda, ¢\}$, and $x \in \Gamma^+$ such that $|x| > |y|$.

For any $\mathcal{R} \in \{\mathcal{R}_0, \mathcal{R}_0', \mathcal{R}_1, \mathcal{R}_1', \mathcal{R}_2, \mathcal{R}_2', \mathcal{R}_3, \mathcal{R}_3'\}$, we will refer to lc-R-automata of type $\mathcal{R}$ as lc-R[$\mathcal{R}$]-automata. In [1], Basovník 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 $M = (\Sigma, \Gamma, I)$ be an lc-R-automaton. With M we can associate the finite string-rewriting system $R(M) = \{\, uxv \to uyv \mid (u \,|\, x \to y \,|\, v) \in I \,\}$. Then $R(M)$ induces a *reduction relation* $\Rightarrow^*_{R(M)}$ on the set of bordered words $\mathfrak{c} \cdot \Gamma^* \cdot \$$, 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 $\mathfrak{c}w\$ \Rightarrow_{R(M)} \mathfrak{c}z\$$ iff $w \vdash^c_M z$ holds. Accordingly, we see that

$$L(M) = \{\, w \in \Sigma^* \mid w \vdash^{c^*}_M \lambda \,\} = \{\, w \in \Sigma^* \mid \mathfrak{c}w\$ \Rightarrow^*_{R(M)} \mathfrak{c}\$ \,\}.$$

By taking $S(M) = R(M) \cup \{\mathfrak{c}\$ \to Y\}$, where Y is a new letter, we obtain a finite string-rewriting system on $\Gamma' = \Gamma \cup \{\mathfrak{c}, \$, Y\}$ such that $L(M) = \{\, w \in \Sigma^* \mid \mathfrak{c}w\$ \Rightarrow^*_{S(M)} Y \,\}$ holds. It follows that $L(M)$ is the *McNaughton language* that is specified by the four-tuple $(S(M), \mathfrak{c}, \$, Y)$.

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[$\mathcal{R}_0$]-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]) $T = (Q, \Sigma, \Gamma, \delta, q_0, \bot, \lambda, \lambda, \{q_f\})$ that accepts L. Here Γ is the tape alphabet of T that includes the input alphabet Σ 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 $\overline{\Gamma} = \{\, \overline{a} \mid a \in \Gamma \,\}$ be a new alphabet in one-to-one correspondence to Γ such that Q, Γ, and $\overline{\Gamma}$ are pairwise disjoint, let $\overline{\ }: \Gamma^* \to \overline{\Gamma}^*$ denote the corresponding

isomorphism that is induced by $a \mapsto \overline{a}$ ($a \in \Gamma$), and let $\Delta = Q \cup \Gamma \cup \overline{\Gamma}$. In analogy to the proof of [20] Lemma 4.1, we can construct a finite length-reducing string-rewriting system R on Δ such that, for all $w \in \Sigma^*$, $w \in L$ iff $\overline{\vdash} q_0 w \dashv \Rightarrow_R^* q_f$. Here the final state q_f is introduced by specific rules of the form $\overline{\vdash}\overline{u}qv\dashv \to q_f$. Now from R we obtain an lc-R-automaton $M = (\Sigma, \Delta, I)$ by taking

$$\begin{aligned} I = {} & \{\, (\lambda \mid \ell \to r \mid \lambda) \mid (\ell \to r) \in R, |\ell|_{\overline{\vdash}} = |\ell|_{\dashv} = 0 \,\} \cup \\ & \{\, (\mathrm{¢} \mid u \to v \mid \lambda) \mid (\overline{\vdash} u \to \overline{\vdash} v) \in R, |u|_{\dashv} = 0 = |u|_{q_0} \,\} \cup \\ & \{\, (\mathrm{¢} \mid u \to v \mid \lambda) \mid (\overline{\vdash} q_0 u \to \overline{\vdash} v) \in R, |u|_{\dashv} = 0 \,\} \cup \\ & \{\, (\lambda \mid u \to v \mid \$) \mid (u\dashv \to v\dashv) \in R, |u|_{\overline{\vdash}} = 0 \,\} \cup \\ & \{\, (\mathrm{¢} \mid u \to v \mid \$) \mid (\overline{\vdash} u\dashv \to \overline{\vdash} v\dashv) \in R, |u|_{q_0} = 0 \,\} \cup \\ & \{\, (\mathrm{¢} \mid u \to v \mid \$) \mid (\overline{\vdash} q_0 u\dashv \to \overline{\vdash} v\dashv) \in R \,\} \cup \\ & \{\, (\mathrm{¢} \mid u \to \lambda \mid \$) \mid (\overline{\vdash} q_0 u\dashv \to q_f) \in R, |u| > 0\} \cup \\ & \{\, (\mathrm{¢} \mid u \to \lambda \mid \$) \mid (\overline{\vdash} u\dashv \to q_f) \in R, |u|_{q_0} = 0 \,\}. \end{aligned}$$

Then M is of type $\mathcal{R}_0$, and for all $w \in \Sigma^+$,

$$w \in L(M) \text{ iff } w \vdash_M^{c^*} \lambda \text{ iff } \mathrm{¢}w\$ \Rightarrow_{R(M)}^* \mathrm{¢}\$ \text{ iff } \overline{\vdash} q_0 w \dashv \Rightarrow_R^* q_f \text{ iff } w \in L.$$

Thus, $L(M) \doteq L$. This completes the proof of Theorem 3.4. □

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 $r = uxv$ for some $A \in N$, $u, v \in (N \cup T)^*$, and $x \in (N \cup T)^+$. By taking

$$I(G) = \{\, (u \,|\, x \to A \,|\, v) \mid (uAv \to uxv) \in P \,\} \cup \{\, (\mathrm{¢} \,|\, r \to \lambda \,|\, \$) \mid (S \to r) \in P \,\},$$

we obtain an lc-R-automaton $M(G) = (T, N \cup T, I(G))$. It is easily seen that $M(G)$ is of type $\mathcal{R}_1'$, 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$[\mathcal{R}_1']$-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. $\mathcal{L}(\text{lc-R}[\mathcal{R}_1']) = \text{GCSL}$.

If $M = (\Sigma, \Gamma, I)$ is an lc-R$[\mathcal{R}_1]$-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 $(u \,|\, x \to \lambda \,|\, v) \in I$ 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$[\mathcal{R}_1]$*-automaton, then there exists an equivalent* lc-R*-automaton $M' = (\Sigma, \Gamma, I')$ of type $\mathcal{R}_1$ such that $|y| = 1$ for all instructions $(u \,|\, x \to y \,|\, v) \in I'$ satisfying $u \neq \mathrm{¢}$ or $v \neq \$$. In fact, for all $w, z \in \Gamma^*$, $w \vdash_M^c z$ if and only if $w \vdash_{M'}^c z$.*

Proof. To obtain M' from M, we simply replace each instruction of the form $i = (u \,|\, x \to \lambda \,|\, v) \in I$. If $u = u_1 A$ for some $A \in \Gamma$, then we replace i by the instruction $i' = (u_1 \,|\, Ax \to A \,|\, v)$; if $u = \lambda$ or $u = \mathbb{¢}$ and $v = Bv_1$ for some $B \in \Gamma$, then we replace i by the instruction $i' = (u \,|\, xB \to B \,|\, v_1)$; if $u = \mathbb{¢}$ and $v = \lambda$, then we replace i by the set of instructions $I'(i) = \{\, (\mathbb{¢} \,|\, xA \to A \,|\, \lambda) \mid A \in \Gamma \,\} \cup \{(\mathbb{¢} \,|\, x \to \lambda \,|\, \$)\}$; if $u = \lambda$ and $v = \$$, then we replace i by the set of instructions $I'(i) = \{\, (\lambda \,|\, Ax \to A \,|\, \$) \mid A \in \Gamma \,\} \cup \{(\mathbb{¢} \,|\, x \to \lambda \,|\, \$)\}$; and if $u = \lambda = v$, then we replace i by the set of instructions $I'(i) = \{\, (\lambda \,|\, xA \to A \,|\, \lambda), (\lambda \,|\, Ax \to A \,|\, \lambda) \mid A \in \Gamma \,\} \cup \{(\mathbb{¢} \,|\, x \to \lambda \,|\, \$)\}$. Then, for all $w, z \in \Gamma^*$, if $w \vdash_M^c z$ using instruction i, then $w \vdash_{M'}^c z$ by instruction i' or by one of the instructions of $I'(i)$, and conversely, if $w \vdash_{M'}^c z$ by instruction i' or by one of the instructions of $I'(i)$, then $w \vdash_M^c z$ by instruction i. □

Now let $M = (\Sigma, \Gamma, I)$ be an $\mathsf{lc\text{-}R}[\mathcal{R}_1]$-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 $R(M)^{-1} = \{\, (v \to u) \mid (u \to v) \in R(M) \,\}$. From $R(M)^{-1}$ we can construct a length-increasing context-sensitive grammar $G(M) = (\Gamma', \Sigma', S, R')$ that generates the language $L(G(M)) = \mathbb{¢} \cdot L(M) \cdot \$$. Thus, the language $\mathbb{¢} \cdot L(M) \cdot \$$ 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

$$I(G) = \{\, (u \,|\, x \to A \,|\, v) \mid (uAv \to uxv) \in P \,\} \cup \{\, (\mathbb{¢} \,|\, r \to \lambda \,|\, \$) \mid (S \to r) \in P \,\},$$

we obtain an $\mathsf{lc\text{-}R}$-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. $\mathcal{L}(\mathsf{lc\text{-}R}[\mathcal{R}_1]) = \mathsf{GACSL}$.

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

Let $M = (\Sigma, \Gamma, I)$ be an $\mathsf{lc\text{-}R}[\mathcal{R}_2']$-automaton. Then, for each instruction $(u \,|\, x \to y \,|\, v) \in I$, we have $u \in \{\lambda, \mathbb{¢}\}$, $v \in \{\lambda, \$\}$, $|x| \geq 1$, $|y| \leq 1$, 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) $R_{\mathrm{bif}} = \{\, \mathbb{¢}x\$ \to \mathbb{¢}y\$ \mid (\mathbb{¢} \,|\, x \to y \,|\, \$) \in I \,\}$, the *bifix rules* of $R(M)$,
(b) $R_{\mathrm{pre}} = \{\, \mathbb{¢}x \to \mathbb{¢}y \mid (\mathbb{¢} \,|\, x \to y \,|\, \lambda) \in I \,\}$, the *prefix rules* of $R(M)$,
(c) $R_{\mathrm{suf}} = \{\, x\$ \to y\$ \mid (\lambda \,|\, x \to y \,|\, \$) \in I \,\}$, the *suffix rules* of $R(M)$,
(d) $R_{\mathrm{inf}} = \{\, x \to y \mid (\lambda \,|\, x \to y \,|\, \lambda) \in I \,\}$, the *infix rules* of $R(M)$.

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

$$L(M) = \{\, w \in \Sigma^* \mid \mathbb{¢}w\$ \Rightarrow_{R(M)}^* \mathbb{¢}\$ \,\} = \{\, w \in \Sigma^* \mid \exists \alpha \in B(M) : \mathbb{¢}w\$ \Rightarrow_{R'}^* \mathbb{¢}\alpha\$ \,\},$$

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

Conversely, if $L \subseteq \Sigma^*$ 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 $\mathsf{lc\text{-}R}$-automaton $M(G) = (\Sigma, N \cup \Sigma, I)$ by taking

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

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

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

Finally, let $M = (\Sigma, \Gamma, I)$ be an $\mathsf{lc\text{-}R}[\mathcal{R}'_3]$-automaton, that is, for all instructions $(u \,|\, x \to y \,|\, v) \in I$, we have $u \in \{\lambda, ¢\}$, $v = \$$, $|x| \geq 1$, $|y| \leq 1$, 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) $R_{\rm bif} = \{ ¢x\$ \to ¢y\$ \mid (¢ \,|\, x \to y \,|\, \$) \in I \}$, the *bifix rules* of $R(M)$,
(b) $R_{\rm suf} = \{ x\$ \to y\$ \mid (\lambda \,|\, x \to y \,|\, \$) \in I \}$, the *suffix rules* of $R(M)$.

As above, the set $B(M) = \{ \alpha \in \Gamma^* \mid ¢\alpha\$ \in \mathrm{dom}(R_{\rm bif}) \text{ and } ¢\alpha\$ \Rightarrow^*_{R(M)} ¢\$ \} \cup \{\lambda\}$ is finite, and

$$L(M) = \{ w \in \Sigma^* \mid ¢w\$ \Rightarrow^*_{R(M)} ¢\$ \} = \{ w \in \Sigma^* \mid \exists \alpha \in B(M) : ¢w\$ \Rightarrow^*_{R_{\rm suf}} ¢\alpha\$ \},$$

that is, $¢ \cdot L(M) \cdot \$ = \nabla^*_{R_{\rm suf}}(¢ \cdot B(M) \cdot \$) \cap (¢ \cdot \Sigma^* \cdot \$)$. For the suffix-rewriting system $P(M) = \{ y \to x \mid (x\$ \to y\$) \in R_{\rm suf} \}$, we obtain $L(M) = \Delta^*_{P(M)}(B(M)) \cap \Sigma^*$. As $B(M)$ is finite, and, hence, regular, it follows that the set of descendants $\Delta^*_{P(M)}(B(M))$ 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 $L \subseteq \Sigma^*$ 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 $\mathsf{lc\text{-}R}$-automaton $M(G) = (\Sigma, N \cup \Sigma, I)$ by taking

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

Then $M(G)$ is of type $\mathcal{R}_3$, and $L(M(G)) \doteq L$. 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.

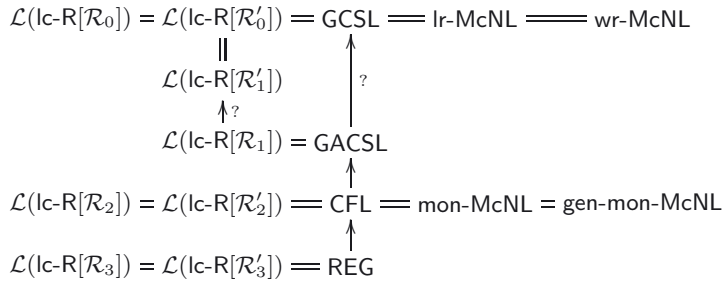

FIGURE 1. Hierarchy of language classes that are accepted by the various types of limited context restarting automata. Question marks indicate inclusions not known to be proper.

Theorem 3.9. $\mathcal{L}(\text{lc-R}[\mathcal{R}'_3]) = \mathcal{L}(\text{lc-R}[\mathcal{R}_3]) = \text{REG}$.

The results on the various lc-R-automata are summarized by the diagram in Figure 1.

4. CONFLUENT LIMITED CONTEXT RESTARTING AUTOMATA

As defined in Definition 3.1, an lc-R-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 $\mathbb{¢}w\$$ into the configuration with tape content $\mathbb{¢}\$$. In general, there will be many different computations of M that start from the configuration with tape content $\mathbb{¢}w\$$, and only some of them will derive the tape content $\mathbb{¢}\$$. As this phenomenon complicates the problem of deciding membership in $L(M)$, we are interested in lc-R-automata for which all computations from $\mathbb{¢}w\$$ lead to $\mathbb{¢}\$$, if $w \in L(M)$. Actually, as the reduction relation $\vdash_M^c$ corresponds to the single-step reduction relation $\Rightarrow_{R(M)}$ that is induced by the string-rewriting system $R(M)$ on the set of bordered words $\mathbb{¢} \cdot \Gamma^* \cdot \$$, this would lead to considering lc-R-automata M for which the string-rewriting system $R(M)$ is *confluent on the congruence class* $[\mathbb{¢}\$]_{R(M)}$. 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 [22]. Therefore, we turn to lc-R-automata that satisfy an even stronger restriction than confluence on a particular congruence class.

Definition 4.1. An lc-R-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 lc-R-automaton, confluence is a decidable property of lc-R-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 $I = \{(\mathfrak{c} \,|\, aaaa \to aaF \,|\, \lambda), (\lambda \,|\, Faa \to aF \,|\, \lambda), (\lambda \,|\, F \to \lambda \,|\, \$), (\mathfrak{c} \,|\, aa \to \lambda \,|\, \$), (\mathfrak{c} \,|\, a \to \lambda \,|\, \$)\}$. Then $M = (\Sigma, \Gamma, I)$ is an lc-R-automaton of type $\mathcal{R}_0$ that accepts the language L_{expo}, and it is easily checked that M is confluent.

We will use the prefix con- to denote types of confluent lc-R-automata. Further, for each type $\mathcal{R} \in \{\, \mathcal{R}'_i, \mathcal{R}_i \mid i \in \{0,1,2,3\} \,\}$, lc-R[con-$\mathcal{R}$] will denote the class of lc-R-automata of type $\mathcal{R}$ 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-$\mathcal{R}'_0$]-automaton, then $S(M) = R(M) \cup \{\mathfrak{c}\$ \to Y\}$ is a finite weight-reducing string-rewriting system that is confluent, and $L(M)$ is simply the McNaughton language that is specified by $(S(M), \mathfrak{c}, \$, Y)$. 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 $\mathcal{R}_0$ can be constructed for L. Hence, we obtain the following characterization.

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

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 $(S, \mathfrak{c}, \$, Y)$, where S is a finite, weight-reducing, and confluent string-rewriting system that consists of rules of the form $(uxv \to uyv)$, where $u, v \in \Gamma^*$, $x \in \Gamma^+$, and $|y| \leq 1$, and rules of the form $(\mathfrak{c}x\$ \to Y)$, 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 con-$\mathcal{R}'_1$, this gives the following characterization.

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

For the class of languages that are accepted by confluent lc-R-automata of type $\mathcal{R}_1$, 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 $L_{\mathrm{expo5}} = \{\, a^{5^n} \mid n \geq 0 \,\}$ is accepted by an lc-R[con-$\mathcal{R}_1$]-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* $\mathcal{L}(\text{lc-R}[\text{con-}\mathcal{R}_1])$ *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. $\mathcal{L}(\text{lc-R}[\text{con-}\mathcal{R}'_2]) \subseteq \text{con-gen-mon-McNL}$.

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:

$$\begin{aligned}
&\text{(a) } R_{\text{bif}} = \{\, ¢x\$ \to ¢y\$ \mid (¢ \,|\, x \to y \,|\, \$) \in I \,\}, \text{ the } \textit{bifix rules} \text{ of } R(M),\\
&\text{(b) } R_{\text{pre}} = \{\, ¢x \to ¢y \mid (¢ \,|\, x \to y \,|\, \lambda) \in I \,\}, \quad \text{the } \textit{prefix rules} \text{ of } R(M),\\
&\text{(c) } R_{\text{suf}} = \{\, x\$ \to y\$ \mid (\lambda \,|\, x \to y \,|\, \$) \in I \,\}, \quad \text{the } \textit{suffix rules} \text{ of } R(M),\\
&\text{(d) } R_{\text{inf}} = \{\, x \to y \mid (\lambda \,|\, x \to y \,|\, \lambda) \in I \,\}, \quad \text{the } \textit{infix rules} \text{ of } R(M).
\end{aligned}$$

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

$$S(M) = R(M) \cup \{¢\$ \to Y\}$$

is confluent, as the additional rule $(¢\$ \to Y)$ 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 $L(M) = L(G, ¢, \$, Y)$ holds. This transformation will be realized in three steps. We use the notation $\text{nf}(w)$ to denote the unique irreducible descendant of the word $w \in (\Gamma \cup \{¢, \$, Y\})^*$ 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 ρ to denote the number $\rho = \max\{\, |x| \mid \exists u, y, v : (u \,|\, x \to y \,|\, v) \in I \,\}$.

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

$$\begin{aligned}
R'_{\text{bif}} = {} & \{\, ¢w\$ \to Y \mid w \in \Gamma^*, |w| \le 2 \cdot \rho, \text{nf}(¢w\$) = Y \,\} \cup\\
& \{\, ¢w\$ \to N \mid w \in \Gamma^*, |w| \le 2 \cdot \rho, \text{nf}(¢w\$) \neq Y \,\}.
\end{aligned}$$

Then $(¢\$ \to Y) \in R'_{\text{bif}}$, and for all $(¢w\$ \to N) \in R'_{\text{bif}}$, we have $|w| \ge 1$. Let

$$S_1(M) = R'_{\text{bif}} \cup R_{\text{pre}} \cup R_{\text{suf}} \cup R_{\text{inf}}. \tag{4.1}$$

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

Claim 4.7. For all $w \in \Gamma^*$, $¢w\$ \Rightarrow^*_{S(M)} Y$ iff $¢w\$ \Rightarrow^*_{S_1(M)} Y$.

Proof. For $w \in \Gamma^*$, if $\mathfrak{c}w\$ \Rightarrow^*_{S(M)} Y$, then either $\mathfrak{c}w\$ \Rightarrow^*_{R_{\mathrm{pre}} \cup R_{\mathrm{suf}} \cup R_{\mathrm{inf}}} \mathfrak{c}\$$, or there exists a rule $(\mathfrak{c}x\$ \to \mathfrak{c}y\$) \in R_{\mathrm{bif}}$ such that $\mathfrak{c}w\$ \Rightarrow^*_{R_{\mathrm{pre}} \cup R_{\mathrm{suf}} \cup R_{\mathrm{inf}}} \mathfrak{c}x\$ \Rightarrow \mathfrak{c}y\$ \Rightarrow^*_{S(M)} Y$ holds. As $|x| \leq \rho$, we see that $(\mathfrak{c}x\$ \to Y)$ is contained in R'_{bif}, and hence, $\mathfrak{c}w\$ \Rightarrow^*_{S_1(M)} Y$ follows. Conversely, if $(\mathfrak{c}x\$ \to Y)$ is a rule of R'_{bif}, then $\mathfrak{c}w\$ \Rightarrow^*_{S(M)} Y$ holds. Thus, for any $w \in \Gamma^*$, if $\mathfrak{c}w\$ \Rightarrow^*_{S_1(M)} Y$, then also $\mathfrak{c}w\$ \Rightarrow^*_{S(M)} Y$. □

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

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 $R_{\mathrm{pre}} \cup R_{\mathrm{inf}}$ and $R_{\mathrm{suf}} \cup R_{\mathrm{inf}}$ of $S_1(M)$ are also confluent. Thus, it remains to consider the critical pairs that result from overlapping a rule of R_{pre} with a rule of R_{suf}, and those that result from overlaps with rules from R'_{bif}.

First, let $(\mathfrak{c}x \to \mathfrak{c}y) \in R_{\mathrm{pre}}$ and $(x'\$ \to y'\$) \in R_{\mathrm{suf}}$ such that $x = x_1x_2$ and $x' = x_2x_3$ for some non-empty factor x_2. Then

$$\mathfrak{c}yx_3\$ \Leftarrow_{R_{\mathrm{pre}}} \mathfrak{c}xx_3\$ = \mathfrak{c}x_1x_2x_3\$ = \mathfrak{c}x_1x'\$ \Rightarrow_{R_{\mathrm{suf}}} \mathfrak{c}x_1y'\$.$$

As $S(M)$ is confluent, $\mathrm{nf}(\mathfrak{c}yx_3\$) = \mathrm{nf}(\mathfrak{c}x_1y'\$)$. Now $|yx_3|, |x_1y'| \leq |x_1x_2x_3| < 2 \cdot \rho$, and hence, R'_{bif} contains the rules $(\mathfrak{c}yx_3\$ \to Z)$ and $(\mathfrak{c}x_1y'\$ \to Z)$ for some symbol $Z \in \{Y, N\}$. Thus, the critical pair $(\mathfrak{c}yx_3\$, \mathfrak{c}x_1y'\$)$ resolves mod $S_1(M)$.

Obviously, there are no overlaps between different rules of R'_{bif}. Hence, it remains to consider the cases that $(\mathfrak{c}w\$ \to Z) \in R'_{\mathrm{bif}}$ and $(x \to y) \in R_{\mathrm{inf}}$ such that $w = w_1xw_2$, or $(\mathfrak{c}x \to \mathfrak{c}y) \in R_{\mathrm{pre}}$ such that $w = xw_2$, or $(x\$ \to y\$) \in R_{\mathrm{suf}}$ such that $w = w_1x$. Here we only consider the first of these cases, as the other two are dealt with in the same way. We have

$$\mathfrak{c}w_1yw_2\$ \Leftarrow_{R_{\mathrm{inf}}} \mathfrak{c}w_1xw_2\$ = \mathfrak{c}w\$ \Rightarrow_{R'_{\mathrm{bif}}} Z.$$

As $S(M)$ is confluent, it follows that $\mathrm{nf}(\mathfrak{c}w_1yw_2\$) = \mathrm{nf}(\mathfrak{c}w\$)$. Further, $|w_1yw_2| \leq |w| \leq 2 \cdot \rho$, and hence, R'_{bif} also contains the rule $(\mathfrak{c}w_1yw_2\$ \to Z)$. Thus, also the critical pair $(\mathfrak{c}w_1yw_2\$, Z)$ resolves. It follows that the system $S_1(M)$ is indeed confluent. □

Step 2. Next we separate those letters that are (prefix or suffix) reducible to λ from the other ones. 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 R'_{bif}, nothing changes by this operation, but for a rule $(x \to y) \in R_{\mathrm{inf}}$, if $|y| = 1$, then y may reduce to another letter $y' \in \Gamma$ or to λ, and analogously for the rules of R_{pre} and R_{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 $\Gamma_0 = \{ A \in \Gamma \mid (A \to \lambda) \in R_{\mathrm{inf}} \}$, that is, Γ_0 is the set of letters that reduce to λ. We now delete all rules from R_{pre}, R_{suf} and R_{inf} that properly contain an occurrence of a letter from Γ_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 $\Gamma_p = \{ A \in \Gamma \setminus \Gamma_0 \mid (\mathbb{¢}A \to \mathbb{¢}) \in R_{\mathrm{pre}} \}$, that is, Γ_p is the set of letters that are prefix reducible to λ. From R_{pre} we now delete all those rules for which the left-hand sides have a proper prefix of the form $\mathbb{¢}A$ for some letter $A \in \Gamma_p$. Analogously, let $\Gamma_s = \{ A \in \Gamma \setminus \Gamma_0 \mid (A\$ \to \$) \in R_{\mathrm{suf}} \}$. From R_{suf} we delete all those rules for which the left-hand sides have a proper suffix of the form $A\$$ for some letter $A \in \Gamma_s$. By $S_2(M)$ we denote the system that is obtained by these operations. It has the form

$$S_2(M) = R'_{\mathrm{bif}} \cup R'_{\mathrm{pre}} \cup R'_{\mathrm{suf}} \cup R'_{\mathrm{inf}}, \tag{4.2}$$

where R'_{inf} consists of two subsystems, $R_{\mathrm{inf},1} = \{ A \to \lambda \mid A \in \Gamma_0 \}$ and $R_{\mathrm{inf},2} = \{ (x \to y) \in R_{\mathrm{inf}} \mid x \in (\Gamma \setminus \Gamma_0)^+ \}$, R'_{pre} consists of two subsystems, $R_{\mathrm{pre},1} = \{ \mathbb{¢}A \to \mathbb{¢} \mid A \in \Gamma_p \}$ and

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

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

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

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 $(S_2(M), \mathbb{¢}, \$, Y)$.

Step 3. The subsystem $R'_{\mathrm{bif}} \cup R'_{\mathrm{inf}}$ of $S_2(M)$ contains generalized monadic rules only, but $R_{\mathrm{pre},2}$ and $R_{\mathrm{suf},2}$ 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

$$\hat{\Gamma} = \Gamma' \cup \{ \mathbb{¢}A \mid A \in \Gamma \setminus (\Gamma_0 \cup \Gamma_p) \} \cup \{ A_\$ \mid A \in \Gamma \setminus (\Gamma_0 \cup \Gamma_s) \}, \tag{4.3}$$

that is, for each letter A that is not prefix reducible to λ, we add a letter $\mathbb{¢}A$, and for each letter A that is not suffix reducible to λ, we add a letter $A_\$$. To simplify the notation, we write $\mathbb{¢}w$ to denote the word that is obtained from w by replacing the first letter A of w by the symbol $\mathbb{¢}A$, and analogously for $w_\$$, and we define a morphism $\psi : \hat{\Gamma}^* \to \Gamma'^*$ through $A \mapsto A$ $(A \in \Gamma)$, $\mathbb{¢}A \mapsto \mathbb{¢}A$ $(A \in \Gamma \setminus (\Gamma_0 \cup \Gamma_p))$, $A_\$ \mapsto A\$$ $(A \in \Gamma \setminus (\Gamma_0 \cup \Gamma_s))$, and $Z \mapsto Z$ $(Z \in \{\mathbb{¢}, \$, Y, N\})$.

We define the system G as

$$G = \hat{R}_{\mathrm{pre}} \cup \hat{R}_{\mathrm{suf}} \cup \hat{R}_{\mathrm{bif}} \cup R'_{\mathrm{inf}}, \tag{4.4}$$

where the subsystems $\hat{R}_{\mathrm{pre}}$, $\hat{R}_{\mathrm{suf}}$, and $\hat{R}_{\mathrm{bif}}$ are defined as follows:

$$\begin{array}{lcll}
(\mathrm{a})\ \hat{R}_{\mathrm{pre}} & = & \{\, \mathfrak{c}A \to {}_{\mathfrak{c}}A & \mid\ A \in \Gamma \setminus (\Gamma_0 \cup \Gamma_p)\,\} \cup R_{\mathrm{pre},1} \cup \\
& & \{\, {}_{\mathfrak{c}}x \to {}_{\mathfrak{c}}y & \mid\ (\mathfrak{c}x \to \mathfrak{c}y) \in R_{\mathrm{pre},2} \text{ and } y \in \Gamma\,\} \cup \\
& & \{\, {}_{\mathfrak{c}}x \to \mathfrak{c} & \mid\ (\mathfrak{c}x \to \mathfrak{c}) \in R_{\mathrm{pre},2}\,\} \cup \\
& & \{\, {}_{\mathfrak{c}}x_1x \to {}_{\mathfrak{c}}y & \mid\ (x_1x \to y) \in R_{\mathrm{inf},2}, x_1, y \in \Gamma \setminus (\Gamma_0 \cup \Gamma_p)\,\} \cup \\
& & \{\, {}_{\mathfrak{c}}x_1x \to \mathfrak{c} & \mid\ (x_1x \to y) \in R_{\mathrm{inf},2}, x_1 \in \Gamma \setminus (\Gamma_0 \cup \Gamma_p), y \in \Gamma_p\,\} \cup \\
& & \{\, {}_{\mathfrak{c}}x_1x \to \mathfrak{c} & \mid\ (x_1x \to \lambda) \in R_{\mathrm{inf},2}, x_1 \in \Gamma \setminus (\Gamma_0 \cup \Gamma_p)\,\}.
\end{array}$$

Observe that $(\mathfrak{c}x \to \mathfrak{c}y) \in R_{\mathrm{pre},2}$ implies that $y \notin \Gamma_p$, as by construction the right-hand side of each rule of $S_2(M)$ is irreducible. Hence, all the marked symbols ${}_{\mathfrak{c}}x$, ${}_{\mathfrak{c}}x_1$, and ${}_{\mathfrak{c}}y$ occurring in the definition of $\hat{R}_{\mathrm{pre}}$ are defined.

$$\begin{array}{lcll}
(\mathrm{b})\ \hat{R}_{\mathrm{suf}} & = & \{\, A\$ \to A_{\$} & \mid\ A \in \Gamma \setminus (\Gamma_0 \cup \Gamma_s)\,\} \cup R_{\mathrm{suf},1} \cup \\
& & \{\, x_{\$} \to y_{\$} & \mid\ (x\$ \to y\$) \in R_{\mathrm{suf},2} \text{ and } y \in \Gamma\,\} \cup \\
& & \{\, x_{\$} \to \$ & \mid\ (x\$ \to \$) \in R_{\mathrm{suf},2}\,\} \cup \\
& & \{\, xb_{\$} \to y_{\$} & \mid\ (xb \to y) \in R_{\mathrm{inf},2}, b, y \in \Gamma \setminus (\Gamma_0 \cup \Gamma_s)\,\} \cup \\
& & \{\, xb_{\$} \to \$ & \mid\ (xb \to y) \in R_{\mathrm{inf},2}, b \in \Gamma \setminus (\Gamma_0 \cup \Gamma_s), y \in \Gamma_s\,\} \cup \\
& & \{\, xb_{\$} \to \$ & \mid\ (xb \to \lambda) \in R_{\mathrm{inf},2}, b \in \Gamma \setminus (\Gamma_0 \cup \Gamma_s)\,\}.
\end{array}$$

As above, $(x\$ \to y\$) \in R_{\mathrm{suf},2}$ implies that $y \notin \Gamma_s$. Hence, all the marked symbols $x_{\$}, b_{\$}$, and $y_{\$}$ occurring in the definition of $\hat{R}_{\mathrm{suf}}$ are defined.

$$\begin{array}{lcl}
(\mathrm{c})\ \hat{R}_{\mathrm{bif}} & = & R'_{\mathrm{bif}} \cup \\
& & \{\, \mathfrak{c}awb\$ \to Y, \mathfrak{c}awb_{\$} \to Y,\ {}_{\mathfrak{c}}awb_{\$} \to Y \mid a \in \Gamma \setminus (\Gamma_0 \cup \Gamma_p), w \in \Gamma^*, \\
& & b \in \Gamma \setminus (\Gamma_0 \cup \Gamma_s), 1 \le |awb| \le 2 \cdot \rho, \mathrm{nf}(\mathfrak{c}awb\$) = Y\,\} \cup \\
& & \{\, \mathfrak{c}awb\$ \to N, \mathfrak{c}awb_{\$} \to N,\ {}_{\mathfrak{c}}awb_{\$} \to N \mid a \in \Gamma \setminus (\Gamma_0 \cup \Gamma_p), w \in \Gamma^*, \\
& & b \in \Gamma \setminus (\Gamma_0 \cup \Gamma_s), 1 \le |awb| \le 2 \cdot \rho, \mathrm{nf}(\mathfrak{c}awb\$) \neq Y\,\}.
\end{array}$$

Here, for each letter $A \in \Gamma$, the rules $(\mathfrak{c}A\$ \to Z)$, $({}_{\mathfrak{c}}A\$ \to Z)$, and $(\mathfrak{c}A_{\$} \to Z)$ are in $\hat{R}_{\mathrm{bif}}$ for some symbol $Z \in \{Y, N\}$, provided ${}_{\mathfrak{c}}A$ and/or $A_{\$}$ 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 $(G, \mathfrak{c}, \$, Y)$ 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 $\mathfrak{c}Au \Rightarrow^*_{S_2(M)} \mathfrak{c}Bv$ and $\mathfrak{c}Bv \in \mathrm{IRR}(S_2(M))$. Then $\mathfrak{c}Au \Rightarrow^*_G {}_{\mathfrak{c}}Bv$, and in addition, if $A \notin \Gamma_0 \cup \Gamma_p$, then also ${}_{\mathfrak{c}}Au \Rightarrow^*_G {}_{\mathfrak{c}}Bv$.

Proof. We proceed by Noetherian induction based on the well-founded partial ordering $\Rightarrow^+_{S_2(M)}$ on $\mathfrak{c} \cdot \Gamma^*$. If $\mathfrak{c}Au$ is irreducible mod $S_2(M)$, then $Au = Bv$. Hence, $A = B \notin \Gamma_0 \cup \Gamma_p$, and $\mathfrak{c}Au \Rightarrow_G {}_{\mathfrak{c}}Au = {}_{\mathfrak{c}}Bv$ follows.

Now assume that $\mathfrak{c}Au \Rightarrow_{S_2(M)} \mathfrak{c}Dw$, where $D \in \Gamma$ and $w \in \Gamma^*$. Then $\mathfrak{c}Dw \Rightarrow^*_{S_2(M)} \mathfrak{c}Bv$, as $S_2(M)$ is confluent. Hence, from the induction hypothesis we know that $\mathfrak{c}Dw \Rightarrow^*_G {}_{\mathfrak{c}}Bv$ holds, and that also ${}_{\mathfrak{c}}Dw \Rightarrow^*_G {}_{\mathfrak{c}}Bv$ holds, if $D \notin \Gamma_0 \cup \Gamma_p$.

If $\mathfrak{c}Au$ is rewritten into $\mathfrak{c}Dw$ by applying a rule $(x \to y) \in R_{\mathrm{inf},2}$ to a factor of u, then $A = D$ is not touched in this step, and $\mathfrak{c}Au \Rightarrow_G \mathfrak{c}Aw = \mathfrak{c}Dw$.

If $\mathfrak{c}Au$ is rewritten into $\mathfrak{c}Dw$ by applying a rule $(\mathfrak{c}Ax \to \mathfrak{c}y) \in R_{\mathrm{pre},2}$, then $u = xz$ for some $z \in \Gamma^*$, $yz = Dw$, and $A \notin \Gamma_0 \cup \Gamma_p$. Hence, $(\mathfrak{c}A \to {}_{\mathfrak{c}}A) \in \hat{R}_{\mathrm{pre}}$,

and either $(\,\text{¢}Ax \to \,\text{¢}y) \in \hat{R}_{\text{pre}}$ (if $y \in \Gamma$) or $(\,\text{¢}Ax \to \text{¢}) \in \hat{R}_{\text{pre}}$ (if $y = \lambda$). Thus, $\text{¢}Au = \text{¢}Axz \Rightarrow_G \,\text{¢}Axz \Rightarrow_G \,\text{¢}yz = \,\text{¢}Dw$ in the former case, and $\text{¢}Au = \text{¢}Axz \Rightarrow_G \,\text{¢}Axz \Rightarrow_G \text{¢}z = \text{¢}Dw$, in the latter case.

If $\text{¢}Au$ is rewritten into $\text{¢}Dw$ by applying a rule $(Ax \to y) \in R_{\text{inf},2}$ to the prefix Ax of Au, then $u = xz$ for some $z \in \Gamma^*$, $yz = Dw$, and $A \notin \Gamma_0$. If $A \notin \Gamma_p$, either, then $\hat{R}_{\text{pre}}$ contains the rule $(\,\text{¢}Ax \to \,\text{¢}y)$, if $y \in \Gamma \setminus (\Gamma_0 \cup \Gamma_p)$, and it contains the rule $(\,\text{¢}Ax \to \text{¢})$, 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 $(Ax \to y)$. However, $(\text{¢}A \to \text{¢})$ is a rule of $R_{\text{pre},1}$, and hence, we obtain $\text{¢}Au = \text{¢}Axz \Rightarrow_{S_2(M)} \text{¢}xz$, and as $S_2(M)$ is confluent, we have $\text{¢}xz \Rightarrow^*_{S_2(M)} \text{¢}Bv$. Hence, by induction hypothesis $\text{¢}xz \Rightarrow^*_G \,\text{¢}Bv$. It follows that $\text{¢}Au = \text{¢}Axz \Rightarrow_G \text{¢}xz \Rightarrow^*_G \,\text{¢}Bv$. □

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 $\text{¢}Au \Rightarrow^*_{S_2(M)} \text{¢}$. Then $\text{¢}Au \Rightarrow^*_G \text{¢}$, and in addition, if $A \notin \Gamma_0 \cup \Gamma_p$, then also $\,\text{¢}Au \Rightarrow^*_G \text{¢}$.

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 $\text{¢}w\$ \Rightarrow^*_{S_2(M)} Y$, then $\text{¢}w\$ \Rightarrow^*_G Y$.

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

$$\text{¢}w\$ \Rightarrow^*_{R'_{\text{pre}} \cup R'_{\text{inf}}} \text{¢}u\$ \Rightarrow^*_{R'_{\text{suf}} \cup R'_{\text{inf}}} \text{¢}v\$ \Rightarrow_{R'_{\text{bif}}} Y,$$

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 $\text{¢}w\$ \Rightarrow^*_G \text{¢}\$ \Rightarrow_G Y$. If $u = Ax$ for some $A \in \Gamma$ and $x \in \Gamma^*$, then by Claim 4.9, $\text{¢}w\$ \Rightarrow^*_G \,\text{¢}Ax\$$. The reduction $\text{¢}u\$ = \text{¢}Ax\$ \Rightarrow^*_{R'_{\text{suf}} \cup R'_{\text{inf}}} \text{¢}v\$$ can be written as

$$\text{¢}Ax\$ \Rightarrow^*_{R'_{\text{suf}} \cup R'_{\text{inf}}} \text{¢}Az\$ \Rightarrow^*_{R'_{\text{suf}} \cup R'_{\text{inf}}} \text{¢}v\$,$$

where $z\$$ is the irreducible descendant of $x\$$ mod $R'_{\text{suf}} \cup R'_{\text{inf}}$.

If $Az\$$ is reducible mod $R'_{\text{suf}} \cup R'_{\text{inf}}$, then it follows from the form of the rules of $S_2(M)$ that $|z| \leq \rho$. If $z = \lambda$, then we have $x\$ \Rightarrow^*_G \$$ by the suffix variant of Claim 4.10, which yields $\,\text{¢}Ax\$ \Rightarrow^*_G \,\text{¢}A\$ \Rightarrow_G Y$. Further, if $z = z'B$ for some $B \in \Gamma$ and $z' \in \Gamma^*$, then the suffix variant of Claim 4.9 shows that $x\$ \Rightarrow^*_G z'B_\$$, and hence, we obtain $\,\text{¢}Ax\$ \Rightarrow^*_G \,\text{¢}Az'B_\$ \Rightarrow_G Y$, as $(\,\text{¢}Az'B_\$ \to Y) \in \hat{R}_{\text{bif}}$. This completes the proof of Claim 4.11. □

From the construction of the rules of G it follows immediately that, for each rule $(x \to y) \in G$, $\psi(x) \Rightarrow^*_{S_2(M)} \psi(y)$ holds. Hence, together with Claim 4.11 this yields the following statement.

Claim 4.12. For all $w \in \Gamma^*$, $\mathbb{¢}w\$ \Rightarrow^*_G Y$ iff $\mathbb{¢}w\$ \Rightarrow^*_{S_2(M)} Y$.

As noted before, $L(M)$ is the McNaughton language that is specified by the tuple $(S_2(M), \mathbb{¢}, \$, Y)$. Thus, Claim 4.12 shows that $L(M) = L(G, \mathbb{¢}, \$, Y)$ 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 $\hat{R}_{\mathrm{bif}}$ and R'_{inf} are confluent. Furthermore, it follows as in the proof of Claim 4.8 above that all overlaps between a rule of $\hat{R}_{\mathrm{bif}}$ and a rule from $\hat{R}_{\mathrm{pre}} \cup \hat{R}_{\mathrm{suf}} \cup R'_{\mathrm{inf}}$ resolve. Thus, it remains to consider the case that a rule from $\hat{R}_{\mathrm{pre}} \cup \hat{R}_{\mathrm{suf}}$ overlaps with a rule from $\hat{R}_{\mathrm{pre}} \cup \hat{R}_{\mathrm{suf}} \cup R'_{\mathrm{inf}}$. As $\hat{R}_{\mathrm{pre}}$ and $\hat{R}_{\mathrm{suf}}$ are defined in a symmetric way, it suffices to consider the case of a rule of $\hat{R}_{\mathrm{pre}}$.

If $(\mathbb{¢}A \to \ \mathbb{¢}A) \in \hat{R}_{\mathrm{pre}}$, then $A \in \Gamma \setminus (\Gamma_0 \cup \Gamma_p)$. Hence, $\mathbb{¢}A$ does not overlap with the left-hand side of any other rule of $\hat{R}_{\mathrm{pre}}$. If $(Ax \to y) \in R'_{\mathrm{inf}}$, then $(\mathbb{¢}Ax \to \ \mathbb{¢}y) \in \hat{R}_{\mathrm{pre}}$, if $y \in \Gamma \setminus (\Gamma_0 \cup \Gamma_p)$, and $(\mathbb{¢}Ax \to \mathbb{¢}) \in \hat{R}_{\mathrm{pre}}$, if $y \in \Gamma_p \cup \{\lambda\}$. In the former case, $(\mathbb{¢}y \to \ \mathbb{¢}y) \in \hat{R}_{\mathrm{pre}}$, and in the latter case we have $\mathbb{¢}y \Rightarrow^*_{R_{\mathrm{pre},1}} \mathbb{¢}$, that is, the critical pair $(\mathbb{¢}Ax, \mathbb{¢}y)$ resolves. Finally, if $(A\$ \to A_\$) \in \hat{R}_{\mathrm{suf}}$, then $A \notin \Gamma_s$, either. Hence, $\mathbb{¢}A\$ \Leftarrow_G \mathbb{¢}A\$ \Rightarrow_G \mathbb{¢}A_\$$, that is, $(\mathbb{¢}A\$, \mathbb{¢}A_\$)$ is a critical pair of G. However, $\hat{R}_{\mathrm{bif}}$ then contains the two rules $(\mathbb{¢}A\$ \to Z)$ and $(\mathbb{¢}A_\$ \to Z)$ for some value of $Z \in \{Y, N\}$, which shows that the above critical pair resolves.

Next assume that $(\mathbb{¢}Ax \to \ \mathbb{¢}y) \in \hat{R}_{\mathrm{pre}}$ and that $(\mathbb{¢}Ax_1 \to \ \mathbb{¢}y') \in \hat{R}_{\mathrm{pre}}$, where $x = x_1x_2$ for some word $x_2 \in \Gamma^*$. Then $\mathbb{¢}y \Leftarrow_G \ \mathbb{¢}Ax = \mathbb{¢}Ax_1x_2 \Rightarrow_G \ \mathbb{¢}y'x_2$, that is, we obtain the critical pair $(\mathbb{¢}y, \mathbb{¢}y'x_2)$. As $S_2(M)$ is confluent, $\mathbb{¢}y$ and $\mathbb{¢}y'x_2$ have a common irreducible descendant mod $R'_{\mathrm{pre}} \cup R'_{\mathrm{inf}}$, 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 $\hat{R}_{\mathrm{pre}}$ with a rule of $\hat{R}_{\mathrm{pre}} \cup R'_{\mathrm{inf}}$ resolve, too. Finally, if $(\mathbb{¢}Ax \to \ \mathbb{¢}y) \in \hat{R}_{\mathrm{pre}}$ and $(wB_\$ \to z_\$) \in \hat{R}_{\mathrm{suf}}$ 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 $(\mathbb{¢}yx_3B_\$, \mathbb{¢}Ax_1z_\$)$. However, as $|yx_3B|, |Ax_1z| \leq |Ax_1x_2x_3B| \leq 2 \cdot \rho$, we see that $\hat{R}_{\mathrm{bif}}$ contains the two rules $(\mathbb{¢}yx_3B_\$ \to Z)$ and $(\mathbb{¢}Ax_1z_\$ \to Z)$ 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 $\hat{R}_{\mathrm{pre}}$ with a rule of $\hat{R}_{\mathrm{suf}}$ 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. $\mathcal{L}(\text{lc-R}[\text{con-}\mathcal{R}_2]) \subseteq \text{con-mon-McNL}$.

We continue with the converse of Theorem 4.6.

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

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 Γ that properly contains the input alphabet Σ, $t_1, t_2 \in (\Gamma \setminus \Sigma)^*$ and $Y \in \Gamma \setminus \Sigma$ are irreducible mod S, and $L \subseteq \Sigma^*$. According to [17] we can assume that S is terminating and *interreduced*, that is, for each rule $(x \to y)$ of S, $y \in \text{IRR}(S)$ and $x \in \text{IRR}(S \setminus \{x \to y\})$. Furthermore, we can assume that S does not contain any rules with right-hand side λ. 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 $L(M) \doteq L$ holds. This automaton will work on an extended alphabet $\hat{\Gamma}$ that is defined as follows:

$$\begin{aligned}\hat{\Gamma} = \Gamma \cup \{\, [\alpha\rangle, \langle\alpha] \mid \alpha \in \text{IRR}(S), |\alpha| \le \rho + 1 \,\} \\ \cup \{\, [\alpha] \mid \alpha \in \text{IRR}(S), |\alpha| \le 3 \cdot \rho + 2 \,\},\end{aligned} \tag{4.5}$$

where $\rho = \max\{\, |w| \mid w \in \{t_1, t_2\} \cup \{\, x \mid (x \to y) \in S \,\} \,\}$. A symbol of the form $[\alpha\rangle$ will be used to encode the normal form α of a word beginning with t_1, a symbol of the form $\langle\alpha]$ will be used to encode the normal form α of a word ending in t_2, and finally, a symbol of the form $[\alpha]$ will be used to encode the normal form α 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 $I = I_{\text{inf}} \cup I_{\text{pre}} \cup I_{\text{suf}} \cup I_{\text{bif}}$:

(a) $I_{\text{inf}} = \{\, (\lambda \,|\, x \to y \,|\, \lambda) \mid (x \to y) \in S \,\}$.

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

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

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

(d) $I_{\text{bif}} = \{\, (\text{¢} \,|\, [\alpha\rangle u \to [\text{nf}(\alpha u t_2)] \,|\, \$) \mid u \in \text{IRR}(S), 0 \le |u| \le \rho + 1 \,\} \cup$
$\{\, (\text{¢} \,|\, u\langle\alpha] \to [\text{nf}(t_1 u \alpha)] \,|\, \$) \mid u \in \text{IRR}(S), 0 \le |u| \le \rho + 1 \,\} \cup$
$\{\, (\text{¢} \,|\, [\alpha\rangle u\langle\gamma] \to [\text{nf}(\alpha u \gamma)] \,|\, \$) \mid u \in \text{IRR}(S), 0 \le |u| \le \rho \,\} \cup$
$\{(\text{¢} \,|\, [Y] \to \lambda \,|\, \$)\}$.

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. $L(M) \doteq L$.

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 $t_1wt_2 \Rightarrow_S^* Y$. As S does not contain any rule with empty right-hand side, this reduction sequence can be factored as $t_1wt_2 \Rightarrow_S^* t_1\mathrm{nf}(w)t_2 \Rightarrow_S^* x \Rightarrow_S Y$ for some rule $(x \to Y)$ of S. Obviously, the automaton M can execute the sequence of cycles $w \vdash_M^{c^*} \mathrm{nf}(w)$ using the instructions of I_{inf}.

If $\mathrm{nf}(w) = a \in \Gamma$, then $\mathrm{nf}(w) = a \vdash_M^c [\mathrm{nf}(t_1a)\rangle \vdash_M^c [Y] \vdash_M^c \lambda$, as $\mathrm{nf}(t_1at_2) = Y$. If $\mathrm{nf}(w) = avb$, where $a, b \in \Gamma$ and $v \in \Gamma^*$, then $\mathrm{nf}(w) = avb \vdash_M^c [\mathrm{nf}(t_1a)\rangle vb \vdash_M^c [\mathrm{nf}(t_1a)\rangle v\langle \mathrm{nf}(bt_2)]$. If $|v| \leq \rho$, then $(\mathfrak{c} \,|\, [\mathrm{nf}(t_1a)\rangle v\langle \mathrm{nf}(bt_2)] \to [Y] \,|\, \$) \in I_{\mathrm{bif}}$. Finally, if $|v| > \rho$, then $\mathrm{nf}(t_1a)\, v$ or $v\, \mathrm{nf}(bt_2)$ (or both) are reducible mod S, as $\mathrm{nf}(t_1a)\, v\, \mathrm{nf}(bt_2)$ reduces to the word x mod S, and $|x| \leq \rho$. Hence, we obtain $[\mathrm{nf}(t_1a)\rangle v\langle \mathrm{nf}(bt_2)] \vdash_M^{c^*} [\mathrm{nf}(t_1av_1)\rangle v_2\langle \mathrm{nf}(v_3bt_2)]$ using instructions from I_{pre} and/or I_{suf}, where $v = v_1v_2v_3$ and $|v_2| \leq \rho$. As $\mathrm{nf}(t_1av_1)\, v_2\, \mathrm{nf}(v_3bt_2) \Rightarrow_S^* Y$, we can now use an instruction from I_{bif} to obtain $[\mathrm{nf}(t_1av_1)\rangle v_2\langle \mathrm{nf}(v_3bt_2)] \vdash_M^c [Y] \vdash_M^c \lambda$. 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 $w \vdash_M^{c^*} \lambda$, and as M has only a single instruction with empty right-hand side, we see that $w \vdash_M^{c^*} [Y]$ holds. We define a morphism $\psi : (\hat{\Gamma} \cup \{\mathfrak{c}, \$\})^* \to (\Gamma \cup \{\mathfrak{c}, \$\})^*$ by taking $a \mapsto a$ $(a \in \Gamma \cup \{\mathfrak{c}, \$\})$, $[\alpha\rangle \mapsto \alpha$, $\langle\alpha] \mapsto \alpha$, and $[\alpha] \mapsto \alpha$ for all values of α. From the definition of the instructions in I, the following properties are easily derived, where $u, v, \alpha, \gamma \in \Gamma^*$:

(a) If $u \vdash_M^c [\alpha\rangle v$, then $t_1u \Rightarrow_S^* \alpha v$.
(b) If $u \vdash_M^c v\langle\alpha]$, then $ut_2 \Rightarrow_S^* v\alpha$.
(c) If $[\alpha\rangle u \vdash_M^c [\alpha\rangle v\langle\gamma]$, then $\alpha ut_2 \Rightarrow_S^* \alpha v\gamma$.
(d) If $u\langle\gamma] \vdash_M^c [\alpha\rangle v\langle\gamma]$, then $t_1u\gamma \Rightarrow_S^* \alpha v\gamma$.
(e) If $u \vdash_M^c v$, where either both, u and v, begin with a letter of the form $[\alpha\rangle$ or none of them does, and where either both end with a letter of the form $\langle\gamma]$ or none of them does, then $\psi(u) \Rightarrow_S^* \psi(v)$.
(f) If $[\alpha\rangle u \vdash_M^c [\gamma]$, then $\alpha ut_2 \Rightarrow_S^* \gamma$.
(g) If $u\langle\alpha] \vdash_M^c [\gamma]$, then $t_1u\alpha \Rightarrow_S^* \gamma$.
(h) If $[\alpha\rangle u\langle\gamma] \vdash_M^c [v]$, then $\alpha u\gamma \Rightarrow_S^* v$.

By induction on the number of cycles in the computation $w \vdash_M^{c^*} [Y]$ it can now be shown easily that $t_1wt_2 \Rightarrow_S^* Y$ 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. □

Proof of Claim 4.17. The string-rewriting system $R(M)$ consists of four subsystems R_{inf}, R_{pre}, R_{suf}, and R_{bif} that are obtained by turning the corresponding sets of instructions into rewrite rules.

By our hypothesis, the subsystem R_{inf}, which coincides with the string-rewriting system S, is confluent.

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

Thus, it remains to consider those critical pairs that result from overlapping rules of different subsystems. Let $(\mathfrak{c}a \to \mathfrak{c}[\mathrm{nf}(t_1 a)\rangle) \in R_{\mathrm{pre}}$ and $(ax \to y) \in R_{\mathrm{inf}}$. From these two rules we obtain the critical pair $(\mathfrak{c}[\mathrm{nf}(t_1 a)\rangle x, \mathfrak{c}y)$. As S is strictly monadic and interreduced, we have $y \in \Gamma \cap \mathrm{IRR}(S)$. Hence, R_{pre} contains the rule $(\mathfrak{c}y \to \mathfrak{c}[\mathrm{nf}(t_1 y)\rangle)$. Further, $x \in \mathrm{IRR}(S)$, $|x| \leq \rho - 1$, and $\mathrm{nf}(t_1 ax) = \mathrm{nf}(t_1 y)$, that is, $|\mathrm{nf}(t_1 ax)| \leq \rho + 1$. Thus, R_{pre} also contains the rule $(\mathfrak{c}[\mathrm{nf}(t_1 a)\rangle x \to \mathfrak{c}[\mathrm{nf}(t_1 ax)\rangle)$, which shows that the critical pair above resolves.

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

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

It remains to consider those critical pairs that result from overlapping a rule of R_{bif} with a rule of the other three subsystems. First, observe that there are no such overlaps with rules of R_{inf}. Now let $(\mathfrak{c}[\alpha\rangle uv\$ \to \mathfrak{c}[\mathrm{nf}(\alpha uvt_2)]\$) \in R_{\mathrm{bif}}$ and $(\mathfrak{c}[\alpha\rangle u \to \mathfrak{c}[\mathrm{nf}(\alpha u)\rangle) \in R_{\mathrm{pre}}$. These rules yield the critical pair $(\mathfrak{c}[\mathrm{nf}(\alpha uvt_2)]\$, \mathfrak{c}[\mathrm{nf}(\alpha u)\rangle v\$)$. But then R_{bif} also contains the rule $(\mathfrak{c}[\mathrm{nf}(\alpha u)\rangle v\$ \to \mathfrak{c}[\mathrm{nf}(\alpha uvt_2)]\$)$, which resolves this critical pair. All other critical pairs that result from an overlap with a rule of R_{pre} also resolve mod R_{bif}, and by symmetry the same holds for the critical pairs that result from overlaps with a rule of R_{suf}. Thus, we have shown that the system $R(M)$ is indeed confluent. □

This completes the proof of Theorem 4.15. □

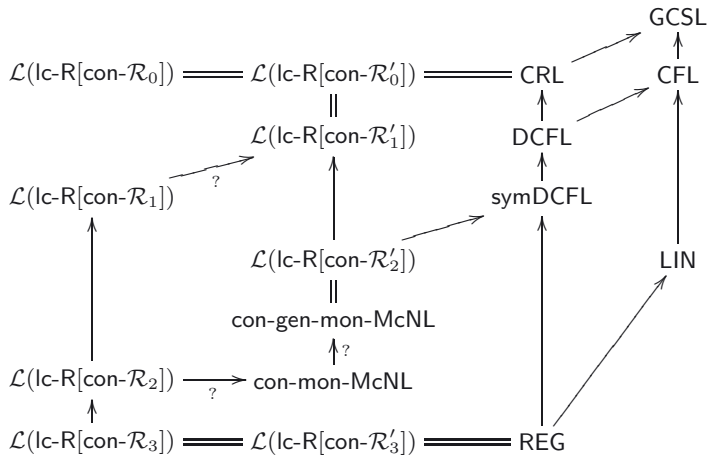

FIGURE 2. Hierarchy of language classes that are accepted by the various types of confluent limited context restarting automata. Question marks indicate inclusions not known to be proper.

Together Theorems 4.6 and 4.15 yield the following equivalence.

Corollary 4.18. $\mathcal{L}(\text{lc-R}[\text{con-}\mathcal{R}'_2]) = \text{con-gen-mon-McNL}$.

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 $q, q' \in Q$:

$$I = \{\, (\mathfrak{c} \,|\, ab \to q \,|\, \lambda) \mid \delta(q_0, ab) = q \,\} \cup \{\, (\mathfrak{c} \,|\, qa \to q' \,|\, \lambda) \mid \delta(q, a) = q' \,\} \cup \{\, (\mathfrak{c} \,|\, q \to \lambda \,|\, \$) \mid q \in F \,\} \cup \{\, (\mathfrak{c} \,|\, a \to \lambda \,|\, \$) \mid a \in \Sigma \cap L^R \,\}.$$

It is easily seen that $L(M) \doteq L^R$, and that the string-rewriting system $R(M)$ is confluent. By taking $M' = (\Sigma, Q \cup \Sigma, I')$, where $I' = \{\, (\lambda \,|\, u^R \to v^R \,|\, \$) \mid (\mathfrak{c} \,|\, u \to v \,|\, \lambda) \in I \,\} \cup \{\, (\mathfrak{c} \,|\, u^R \to v^R \,|\, \$) \mid (\mathfrak{c} \,|\, u \to v \,|\, \$) \in I \,\}$, 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. $\mathcal{L}(\text{lc-R}[\text{con-}\mathcal{R}'_3]) = \mathcal{L}(\text{lc-R}[\text{con-}\mathcal{R}_3]) = \text{REG}$.

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

[1] S. Basovník, Learning restricted restarting automata using genetic algorithm. Master's thesis, Charles University. Faculty of Mathematics and Physics, Prague, Czech (2010).

[2] S. Basovník and F. Mráz, Learning limited context restarting automata by genetic algorithms, in *Theorietag*, edited by J. Dassow and B. Truthe. Otto-von-Guericke-Universität, Magdeburg (2011) 1–4.

[3] M. Beaudry, M. Holzer, G. Niemann and F. Otto, McNaughton families of languages. *Theoret. Comput. Sci.* **290** (2003) 1581–1628.

[4] R.V. Book and F. Otto, String-Rewriting Systems. Springer, New York (1993).

[5] J.R. Büchi, Regular canonical systems. *Arch. f. Math. Logik Grundlagenf.* **6** (1964) 91–111.

[6] G. Buntrock, *Wachsende kontext-sensitive Sprachen*. Habilitationsschrift. Fakultät für Mathematik und Informatik, Universität Würzburg (1996).

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

[8] J. Čejka, *Learning correctness preserving reduction analysis*. BSc Project, Charles University. Faculty of Mathematics and Physics, Prague, Czech (2003).

[9] P. Černo and F. Mráz, Clearing restarting automata. *Fund. Inf.* **104** (2010) 17–54.

[10] P. Černo and F. Mráz, Δ-clearing restarting automata and CFL, in edited by G. Mauri and A. Leporati. *DLT 2011*, in vol. 6795 of *Lect. Notes Comput. Sci.* Springer, Berlin (2011) 153–164.

[11] E. Dahlhaus and M. Warmuth, Membership for growing context-sensitive grammars is polynomial. *J. Comput. System Sci.* **33** (1986) 456–472.

[12] D. Hofbauer and J. Waldmann, Deleting string rewriting systems preserve regularity. *Theoret. Comput. Sci.* **327** (2004) 301–317.

[13] P. Hoffmann, Learning restarting automata by genetic algorithms, in *SOFSEM 2002: Student Research Forum*, edited by M. Bieliková. Masaryk University, Brno (2002) 15–20.

[14] P. Jančar, F. Mráz, M. Plátek and J. Vogel, Restarting automata, *FCT'95*, in vol. 965 of *Lect. Notes Comput. Sci.*, edited by H. Reichel. Springer, Berlin (1995) 283–292.

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

[16] D. Knuth and P. Bendix, Simple word problems in universal algebras, in *Comput. Problems in Abstract Algebra*, edited by J. Leech. Pergamon Press, New York (1970) 263–297.

[17] P. Leupold and F. Otto, On McNaughton families of languages that are specified by some variants of monadic string-rewriting systems. *Fund. Inf.* **112** (2011) 219–238.
[18] R. McNaughton, P. Narendran and F. Otto, Church-Rosser Thue systems and formal languages. *J. Assoc. Comput. Mach.* **35** (1988) 324–344.
[19] F. Mráz, F. Otto and M. Plátek, Learning analysis by reduction from positive data, in *ICGI 2006*, in vol. 4201 of *Lect. Notes Comput. Sci.*, edited by Y. Sakakibara, S. Kobayashi, K. Sato, T. Nishino and E. Tomita. Springer, Berlin (2006) 125–136.
[20] 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.
[21] G. Niemann and J.R. Woinowski, The growing context-sensitive languages are the acyclic context-sensitive languages, in *DLT 2002*, in vol. 2295 of *Lect. Notes Comput. Sci.*, edited by W. Kuich, G. Rozenberg and A. Salomaa. Springer, Berlin (2002) 197–205.
[22] F. Otto, On deciding the confluence of a finite string-rewriting system on a given congruence class. *J. Comput. System Sci.* **35** (1987) 285–310.
[23] F. Otto, Restarting automata, *Recent Advances in Formal Languages and Applications*, in vol. 25 of *Studies in Comput. Intelligence*, edited by Z. Ésik, C. Martin-Vide and V. Mitrana. Springer, Berlin (2006) 269–303.
[24] F. Otto, P. Černo and F. Mráz, Limited context restarting automata and McNaughton families of languages, in *Fourth Workshop on Non-Classical Models for Automata and Applications (NCMA 2012), Proc.*, books@ocg.at, Band 290, edited by R. Freund, M. Holzer, B. Truthe and U. Ultes-Nitsche. Oesterreichische Computer Gesellschaft, Wien (2012) 165–180.
[25] J. Woinowski, The context-splittable normal form for Church-Rosser language systems. *Inform. Comput.* **183** (2003) 245–274.

Communicated by M. Holzer.
Received January 28, 2013. Accepted January 8, 2014.