McNaughton Languages

M. Beaudry[1], **M. Holzer**[2,*], **G. Niemann**[3] and **F. Otto**[3]

[1] Département de mathématique et informatique
Université de Sherbrooke
Sherbrooke, Québec, Canada J1K 2R1
`beaudry@dmi.usherb.ca`

[2] Institut für Informatik, Technische Universität München
Arcisstraße 21, 80290 München, Germany
`holzer@informatik.tu-muenchen.de`

[3] Fachbereich Mathematik/Informatik
Universität Kassel, 34109 Kassel, Germany
{`niemann,otto`}`@theory.informatik.uni-kassel.de`

November 1, 2000

Abstract

In 1988 the Church-Rosser languages were introduced by McNaughton et al. as the languages that are recognized by finite, length-reducing and confluent string-rewriting systems using extra non-terminal symbols. Here we generalize this concept by considering classes of languages that are obtained by other types of string-rewriting systems. To honour Robert McNaughton's original contribution to this development we call the resulting languages *McNaughton languages* and the resulting language classes *McNaughton classes*. Here it is shown that the concept of McNaughton classes is as powerful as the notion of the Turing machine or the notion of the phrase-structure grammar. We investigate the relationships between the various classes of McNaughton languages, and we consider the closure and non-closure properties of these language classes. Also we address the complexity of the fixed and the general membership problems for these classes.

1 Introduction

In [MNO88] the class CRL of *Church-Rosser languages* was introduced. A language $L \subseteq \Sigma^*$ is called a Church-Rosser language, if it consists of those strings $w \in \Sigma^*$ that, placed into the context $t_1 w t_2$ of certain auxiliary strings t_1 and t_2, reduce to a certain final "accepting" symbol with respect to a finite, length-reducing, and confluent string-rewriting system. Apart from the final symbol and the symbols occurring in the context strings t_1 and t_2, also other non-terminal symbols are allowed in this definition. An essential observation in relation to this definition is the fact that, although the rewriting process with respect to the string-rewriting system considered is inherently non-deterministic, the confluence of the system ensures that each reduction sequence will lead to the same result. And indeed, it has been observed that

*Most of the work presented here was performed, while M. Holzer was visiting the Département d'I.R.O., Université de Montréal, Montréal, Québec, Canada H3C 3J7.

the class CRL of Church-Rosser languages can be interpreted as the deterministic variant of the class GCSL of *growing context-sensitive languages* [BO98, NO98].

As the string-rewriting system is the central part in the definition of a Church-Rosser language, it is only natural to ask for the influence of the various restrictions that are placed on these string-rewriting systems on the class of languages that are defined in this way. Here we present a detailed study of the language classes that are obtained by admitting other classes of finite string-rewriting systems in the definition. To honour Robert McNaughton's original contribution to this development we call the resulting language classes *McNaughton classes*. Here it is shown that the concept of McNaughton classes is as powerful as the notion of the Turing machine or the notion of the phrase-structure grammar. In fact the class of recursively enumerable languages is obtained as the largest class of McNaughton languages, and all the classes in the Grzcegorczyk hierarchy above level two as well as the well-known complexity classes P and NP and the classes CSL and CFL of the Chomsky hierarchy can be characterized as classes of McNaughton languages.

In fact, we consider two families of McNaughton classes: those defined by finite (terminating) string-rewriting systems, and those defined by finite, terminating and confluent systems. As we will see the former correspond in some sense to nondeterministic complexity and language classes, while the later correspond to deterministic complexity and language classes. The class GCSL of growing context-sensitive languages, which will be characterized as a (non-confluent) McNaughton class, will play a special role in the hierarchy of McNaughton classes, as it is related by inclusion to every other class. Further, all the McNaughton classes above GCSL have well-known characterizations as complexity classes, while below GCSL the class CFL of context-free languages is the only McNaughton class (apart from the class CRL, of course) that has a well-known characterization in terms of grammars or automata. In particular, some McNaughton classes are obtained naturally that fall into the gap between the class REG of regular languages and the class CSL of context-sensitive languages.

The paper is structured as follows. After establishing the concept of McNaughton classes in Section 2 we will show that the most general of these classes coincide with the class of recursively enumerable languages. Also the classes in the Grzegorczyk hierarchy, the class of primitive-recursive languages and the classes P and NP are described as McNaughton classes.

In Section 3 we introduce various McNaughton classes that are contained in the class CSL of context-sensitive languages. We obtain characterizations for the class CSL and the class DCSL of deterministic context-sensitive languages, and of course the classes CRL and GCSL reappear in this context. In particular, we will consider those classes of McNaughton languages that are defined by finite string-rewriting systems that are monadic or special. The monadic McNaughton languages will be shown to coincide with CFL, but surprizingly we have no characterization of the class DCFL of deterministic context-free languages in terms of McNaughton classes, as the class con-mon-McNL of McNaughton languages that are defined by finite, monadic, and confluent string-rewriting systems is just a proper subclass of DCFL. The classes sp-McNL and con-sp-McNL of McNaughton languages defined by finite special string-rewriting systems that are non-confluent, respectively confluent, are even incomparable to the class REG of regular languages. However, by establishing a normal form for systems specifying languages of this class, we can at least show that each language from the class con-sp-McNL is *Church-Rosser congruential*, which is another notion introduced in [MNO88]. This section closes with a diagram summarizing the known inclusion relations between the classes discussed in the paper.

In Section 4 we investigate the closure and non-closure properties of the lower classes of the hierarchy of McNaughton classes. We will see that each McNaughton class is closed under left- and right-derivatives, and obviously they are all closed under reversal, too. On the other hand, the classes $\mathsf{con\text{-}sp\text{-}McNL}$, $\mathsf{sp\text{-}McNL}$, and $\mathsf{con\text{-}mon\text{-}McNL}$ do not seem to have many additional closure properties, as they are neither closed under union, nor under intersection, nor under product, nor under λ-free morphisms, to mention just a few. Also we show that for $k \geq 1$ the classes $\bigcup^k \mathsf{con\text{-}sp\text{-}McNL}$ and $\bigcup^k \mathsf{con\text{-}mon\text{-}McNL}$ form strict hierarchies with respect to inclusion.

Finally, in Section 5 we address the complexity of the fixed and general membership problems for the classes $\mathsf{con\text{-}sp\text{-}McNL}$, $\mathsf{sp\text{-}McNL}$, $\mathsf{con\text{-}mon\text{-}McNL}$, and $\mathsf{mon\text{-}McNL}$. For the non-confluent classes these problems are $\mathsf{LOG(CFL)}$-complete, while for the confluent classes we obtain $\mathsf{LOG(DCFL)}$ as an upper bound only, and the lower bounds obtained are much weaker. In the concluding section we will present some open problems and point out some ideas for future research.

2 Definitions and first results

Let Σ denote a finite alphabet. Then Σ^* is the set of all strings over Σ including the empty string λ, and Σ^+ is the set of all non-empty strings over Σ. As usual $|w|$ denotes the length of the string w, and numerical exponents are used to abbreviate strings, where $w^0 = \lambda$ and $w^{n+1} = w^n w$ for all $w \in \Sigma^*$ and $n \in \mathbb{N}$.

We assume that the reader is familiar with the basic concepts of formal language and automata theory. As our standard reference concerning this field we use the monographs by Hopcroft and Ullman [HU79] and Harrison [Har79]. In addition to this classical field of computer science we need some background from the theory of string-rewriting systems.

A *string-rewriting system* on Σ is a set R of pairs of strings from Σ^*. Usually a pair $(\ell, r) \in R$ is written as $(\ell \to r)$, and it is called a *rewrite rule*. By $\mathrm{dom}(R)$ we denote the set $\mathrm{dom}(R) = \{\ \ell \mid \exists r \in \Sigma^* : (\ell \to r) \in R\ \}$ of all left-hand sides of rules of R, and $\mathrm{range}(R) = \{\ r \mid \exists \ell \in \Sigma^* : (\ell \to r) \in R\ \}$ is the set of all right-hand sides.

The *reduction relation* $\to_R^*$ on Σ^* that is induced by R is the reflexive and transitive closure of the *single-step reduction relation*

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

For a string $u \in \Sigma^*$, if there exists a string v such that $u \to_R v$ holds, then u is called *reducible* modulo R, v is a *direct descendant* of u, and u is an *direct ancester* of v. If such a string v does not exist, then u is called *irreducible* modulo R. By $\Delta_R^*(u)$ we denote the set of all descendants of u, that is, $\Delta_R^*(u) = \{\ v \mid u \to_R^* v\ \}$, $\nabla_R^*(v)$ is the set of all ancestors of v, that is, $\nabla_R^*(v) = \{\ u \mid u \to_R^* v\ \}$, and by $\mathrm{IRR}(R)$ we denote the set of all irreducible strings modulo R. It is easily seen that $\mathrm{IRR}(R)$ is a regular language for each finite string-rewriting system R. By $\leftrightarrow_R^*$ we denote the *Thue congruence* on Σ^* that is induced by R. It is the smallest equivalence relation on Σ^* containing the single-step reduction relation $\to_R$.

Here we will be interested in certain restricted types of string-rewriting systems. A string-rewriting system R is called

- *terminating* or *noetherian* if there is no infinite sequence of reduction steps $w \to_R w_1 \to_R w_2 \to_R \cdots$,
- *locally confluent* if, for all $u, v, w \in \Sigma^*$, $u \to_R v$ and $u \to_R w$ imply that there exists some $z \in \Sigma^*$ such that $v \to_R^* z$ and $w \to_R^* z$ hold,
- *confluent* if, for all $u, v, w \in \Sigma^*$, $u \to_R^* v$ and $u \to_R^* w$ imply that there exists some $z \in \Sigma^*$ such that $v \to_R^* z$ and $w \to_R^* z$ hold, and

- *convergent* if it is both terminating and confluent.

If reduction sequences are seen as computations, then the termination property expresses the fact that each computation terminates after finitely many steps, and the confluence property expresses the fact that the resulting irreducible descendant of a given string is unique, if it exists. Thus, for a convergent system R the set $\mathrm{IRR}(R)$ of irreducible strings is a complete set of unique representatives for the Thue congruence $\leftrightarrow_R^*$.

For future reference we restate a technical characterization of local confluence. This characterization is based on the notion of *critical pairs*. Let R be a string-rewriting system on Σ. There are two kinds of critical pairs for R. First, if there are two rules $\ell \to r$ and $\ell' \to r'$ in R such that $\ell = x\ell'y$ for some $x, y \in \Sigma^*$, then the string ℓ can be reduced by either of the two rules. The one rule yields the string r, the other yields the string $xr'y$. If the system R is to be locally confluent, then the strings r and $xr'y$ must have a common descendant. Accordingly, the pair $(r, xr'y)$ is called a *critical pair* of R. Secondly, if $\ell = xy$ and $\ell' = yz$ for some non-empty strings $x, y, z \in \Sigma^*$, then the string xyz can be reduced by either rule: replacing the redex xy $(= \ell)$ by r yields the string rz, and replacing the redex yz $(= \ell')$ by r' yields the string xr'. Then the pair (rz, xr') is a *critical pair* of R.

It can be shown that R is locally confluent if and only if, for each critical pair (z, z') of R, z and z' have a common descendant modulo R. In particular, if R has no critical pairs at all, that is, if the left-hand sides of the rules of R do not overlap, then R is locally confluent. For additional information concerning these notions we refer to the literature, where the monograph [BO93] serves as our main reference on string-rewriting systems.

Finally we come to the main definition of this paper. 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 R on Γ, strings $t_1, t_2 \in (\Gamma \smallsetminus \Sigma)^* \cap \mathrm{IRR}(R)$, and a letter $Y \in (\Gamma \smallsetminus \Sigma) \cap \mathrm{IRR}(R)$ such that, for all $w \in \Sigma^*$, the following two statements are equivalent:

(1.) $w \in L$;

(2.) $t_1wt_2 \to_R^* Y$, that is, $t_1wt_2 \in \nabla_R^*(Y)$.

Here the symbols of Σ are *terminals*, while those of $\Gamma \smallsetminus \Sigma$ can be seen as *non-terminals*. We say that the McNaughton language L is *specified* by the four-tuple (R, t_1, t_2, Y). This fact will be expressed as $L = L(R, t_1, t_2, Y)$.

By placing restrictions on the finite string-rewriting systems used we will obtain certain subclasses of the class McNL of all McNaughton languages. Below we will see that the above notion is as powerful as the notion of the Turing machine or the notion of the phrase-structure grammar. In fact for sufficiently large complexity classes $\mathcal{C}$ we will see that by restricting the derivation length of the string-rewriting systems used through a function from $\mathcal{C}$ we obtain the complexity class $\mathcal{C}$. However, for more restricted string-rewriting systems we will obtain language classes which do not seem to coincide with Turing machine time-complexity classes. A special case, for example, is the class CRL of Church-Rosser languages [MNO88].

Theorem 1. *The class* McNL *coincides with the class of recursively enumerable sets.*

Proof. "$\subseteq$": Let $L \subseteq \Sigma^*$ be a McNaughton language, let R be a finite string-rewriting system on $\Gamma \supsetneq \Sigma$, and let $t_1, t_2 \in (\Gamma \smallsetminus \Sigma)^* \cap \mathrm{IRR}(R)$ and $Y \in (\Gamma \smallsetminus \Sigma) \cap \mathrm{IRR}(R)$ such that, for each $w \in \Sigma^*$, $w \in L$ iff $t_1wt_2 \to_R^* Y$. Given a string $w \in \Sigma^*$, a Turing machine can systematically enumerate the set $\Delta_R^*(t_1wt_2)$ of descendants of t_1wt_2 modulo R. Now w is accepted, if and when Y is encountered. Thus, L is recursively enumerable.

"$\supseteq$": Let $L \subseteq \Sigma^*$ be recursively enumerable. Then there exists a nondeterministic single-tape Turing machine $M = (Q, \Gamma, q_0, q_a, \delta)$ such that L is the language accepted by M. Here Q is the finite set of states, $\Gamma \supset \Sigma \cup \{\sqcup\}$ is the tape alphabet, where $\sqcup$ denotes the blank symbol, $q_0 \in Q$ is the initial state, $q_a \in Q$ is the unique halting (accepting) state, and

$$\delta \subseteq (Q \smallsetminus \{q_a\}) \times \Gamma \times (Q \smallsetminus \{q_0\}) \times \Gamma \times \{\texttt{left}, \texttt{right}\}$$

is the transition relation, that is, if $(q, a, q', c, \texttt{move}) \in \delta$, then M can perform the following actions if it is in state q seeing a in the actual tape square: replace a by c, move the head one step according to $\texttt{move} \in \{\texttt{left}, \texttt{right}\}$, and enter state q'. As usual the configurations of M are encoded as strings from $\Gamma^* \cdot Q \cdot \Gamma^*$ (see, e.g., [HU79]), and by $\vdash_M$ we express the single-step computation relation of M on the set of configurations. Hence, for each $w \in \Sigma^*$, $w \in L$ if and only if M has a computation of the form $q_0 w \vdash_M u_1 q_1 w_1 \vdash_M \cdots \vdash_M u_n q_a v_n$ for some $u_n, v_n \in \Gamma^*$, $n \geq 1$.

Let $\Omega := Q \cup \Gamma \cup \overline{\Gamma} \cup \{¢, \$, \#\}$, where $\overline{\Gamma} := \{ \bar{a} \mid a \in \Gamma \}$, and let R be the string-rewriting system containing the following two groups of rules:

(a) Rules for simulating the Turing machine M:

$$\left.\begin{array}{rcllcl}
q_i a & \to & \bar{c} q_j & \text{if} & (q_i, a, q_j, c, \texttt{right}) & \in \delta, \\
q_i \$ & \to & \bar{c} q_j \$ & \text{if} & (q_i, \sqcup, q_j, c, \texttt{right}) & \in \delta, \\
\bar{d} q_i a & \to & q_j d c & \text{if} & (q_i, a, q_j, c, \texttt{left}) & \in \delta \\
\bar{d} q_i \$ & \to & q_j d c \$ & \text{if} & (q_i, \sqcup, q_j, c, \texttt{left}) & \in \delta \\
¢ q_i a & \to & ¢ q_j b c & \text{if} & (q_i, a, q_j, c, \texttt{left}) & \in \delta, \\
¢ q_i \$ & \to & ¢ q_j b c \$ & \text{if} & (q_i, \sqcup, q_j, c, \texttt{left}) & \in \delta.
\end{array}\right\} \text{ for all } \bar{d} \in \overline{\Gamma},$$

(b) Rules for erasing final configurations of M:

$$\begin{array}{rcll}
q_a c & \to & q_a & \text{for all } c \in \Gamma, \\
\bar{c} q_a \$ & \to & q_a \$ & \text{for all } \bar{c} \in \overline{\Gamma}, \\
¢ q_a \$ & \to & \# & .
\end{array}$$

Let $t_1 := ¢ q_0$, $t_2 := \$$, and $Y := \#$.

Claim: $t_1 w t_2 \to_R^* Y$ iff $w \in L$.

Proof. If $w \in L$, then there exists an accepting computation of the form $q_0 w \vdash_M u_1 q_1 v_1 \vdash_M \cdots \vdash_M u_n q_a v_n$. Hence, $¢ q_0 w \$ \to_R ¢ \bar{u}_1 q_1 v_1 \$ \to_R \cdots \to_R ¢ \bar{u}_n q_a v_n \$ \to_R^* ¢ q_a \$ \to \#$, where $^{-} : \Gamma^* \to \overline{\Gamma}^*$ denotes the obvious morphism.

On the other hand, if $¢ q_0 w \$ \to_R^* \#$, then $¢ q_0 w \$ \to_R^* ¢ \bar{u} q_a v \$$ for some $u, v \in \Gamma^*$, and this reduction corresponds to an accepting computation of M, that is, $w \in L$. □□

It is easily verified that the only overlaps between left-hand sides of rules of the system R in the proof above result from transitions that are applicable to the same configurations. Thus, if the Turing machine M is deterministic, then the system R constructed from M has no overlaps, that is, it is locally confluent. In general, R will not be terminating, but it can nevertheless be shown that R is confluent [Hue80].

A McNaughton language will be called *confluent* if it is specified by a finite confluent string-rewriting system. By con-McNL we denote the class of confluent McNaughton languages. As each recursively enumerable set is accepted by some deterministic Turing machine, the observation above yields the following consequence.

Corollary 2. *The class* con-McNL *of confluent McNaughton languages coincides with the class of recursively enumerable sets.*

Next we restrict our attention to those McNaughton languages for which the string-rewriting system used in the specification has a bounded reduction length.

Let R be a string-rewriting system on Γ, and let $w \in \Gamma^*$. Then $\mathrm{rl}_R : \Gamma^* \to \mathbb{N} \cup \{\infty\}$ is the following function:

$$\mathrm{rl}_R(w) := \begin{cases} \max\{\, m \mid \exists w_0, w_1, \ldots, w_m \in \Gamma^* : \\ \qquad\qquad w = w_0 \to_R w_1 \to_R \cdots \to_R w_m \,\} & \text{if such a maximum exists,} \\ \infty & \text{otherwise.} \end{cases}$$

Further, $\mathrm{Rl}_R : \mathbb{N} \to \mathbb{N} \cup \{\infty\}$ is the following function:

$$\mathrm{Rl}_R(n) := \max\{\, \mathrm{rl}_R(w) \mid w \in \Gamma^*, |w| = n \,\}.$$

Observe that a finite system R is terminating iff $\mathrm{Rl}_R(\mathbb{N}) \subseteq \mathbb{N}$ holds.

The string-rewriting system R is *f-reduction bounded*, where $f : \mathbb{N} \to \mathbb{N}$, if $\mathrm{Rl}_R(n) \leq f(n)$ holds for all $n \in \mathbb{N}$. For a class $\mathcal{C}$ of functions R is called *$\mathcal{C}$-reduction bounded*, if R is f-reduction bounded for some function $f \in \mathcal{C}$.

For a function $f : \mathbb{N} \to \mathbb{N}$ we obtain the classes of *f-bounded McNaughton languages* f-b-McNL and of *confluent f-bounded McNaughton languages* con-f-b-McNL by admitting only finite (confluent) string-rewriting systems that are f-reduction bounded. For a class $\mathcal{C}$ of functions, we obtain the language classes $\mathcal{C}$-b-McNL and con-$\mathcal{C}$-b-McNL in this way.

Let E_n denote the n-th level of the Grzegorczyk hierarchy [Grz53, Tou84]. A language is called *E_n-decidable* if its characteristic function belongs to the class E_n.

Theorem 3. *For all $n \geq 3$, E_n-b-McNL= con-E_n-b-McNL= E_n-decidable languages.*

Proof. Let $L \in \mathsf{E}_n$-b-McNL. Then there exist a function $f \in \mathsf{E}_n$, an f-reduction bounded string-rewriting system R, and strings t_1, t_2, Y such that, for all $w \in \Sigma^*$, $w \in L$ iff $t_1 w t_2 \to_R^* Y$. Since R is f-reduction bounded, the reduction sequence $t_1 w t_2 \to_R^* Y$ has length $m \leq f(|t_1 w t_2|)$. Now a Turing machine can enumerate all reduction sequences starting with $t_1 w t_2$, checking whether or not Y is obtained. Since $f \in \mathsf{E}_n$ and $n \geq 3$, this can be done by a machine that is E_n-time bounded. Hence, L is E_n-decidable.

If L is E_n-decidable, then there exist a function $g \in \mathsf{E}_n$ and a deterministic single-tape Turing machine M that decides membership of L with time bound g. As in the proof of Theorem 1 this Turing machine can be simulated by a string-rewriting system.

We can assume without loss of generality that starting with an arbitrary configuration uqv, M will halt within at most $g(|uv|)$ many steps [MO85, Bir98]. Hence, the resulting string-rewriting system is h-reduction bounded for some function $h \in \mathsf{E}_n$. In addition, it is confluent, which completes the proof of Theorem 3. $\Box$

The class of primitive-recursive functions coincides with the union $\bigcup_{n \geq 0} \mathsf{E}_n$. In particular, it is also a Turing machine time-complexity class. Thus, by using essentially the same proof we obtain the following result, where prim-rec-McNL (con-prim-rec-McNL) denotes the class of (confluent) McNaughton languages that are bounded by a primitive-recursive function.

Corollary 4. *The classes prim-rec-McNL and con-prim-rec-McNL coincide with the class of languages that are primitive-recursively decidable.*

On the other hand it is known that a function f belongs to the Grzegorczyk class E_2 if and only if is computed by a deterministic Turing machine in polynomial time. Hence, the proof of Theorem 3 also yields the following result.

Corollary 5.

(a) E_2-b-McNL= NP.

(b) con-E_2-b-McNL= P.

This result very nicely illustrates the fact that confluent McNaughton classes correspond to deterministic complexity or language classes, while non-confluent McNaughton classes correspond to non-deterministic complexity or language classes.

3 Subclasses of the context-sensitive languages

In this section we will see that the class CSL of context-sensitive languages coincides with a particular class of McNaughton languages. In addition, we will present several classes of McNaughton languages that are proper subclasses of the class CSL.

A McNaughton language is called *non-increasing* if it is defined using a finite terminating string-rewriting system R such that $|\ell| \geq |r|$ holds for each rule $(\ell \to r) \in R$. The resulting class of languages is denoted by non-in-McNL. A McNaughton language is called *length-reducing* if it is defined using a finite string-rewriting system R that is length-reducing, that is, for each rule $(\ell \to r) \in R$ we have $|\ell| > |r|$. The resulting language class is denoted by lr-McNL.

Instead of using the length of strings as the means to orient rules, we can also use a weight-function. A function $\varphi : \Sigma \to \mathbb{N}_+$ is called a *weight-function.* Its extension to Σ^*, which we will also denote by φ, is defined inductively through $\varphi(\lambda) = 0$ and $\varphi(wa) = \varphi(w) + \varphi(a)$ for all $w \in \Sigma^*$ and $a \in \Sigma$. A particular weight-function is the length-function $|\cdot| : \Sigma \to \mathbb{N}_+$, which assigns each letter the weight (length) 1.

A McNaughton language is called *non-weight-increasing* if it is defined using a finite terminating string-rewriting system R such that there exists a weight-function φ satisfying $\varphi(\ell) \geq \varphi(r)$ for each rule $(\ell \to r) \in R$. The resulting class of languages is denoted by non-win-McNL. Finally, a McNaughton language is called *weight-reducing* if it is defined using a weight-reducing string-rewriting system R, that is, there exists a *weight-function* $\varphi : \Sigma \to \mathbb{N}_+$ satisfying $\varphi(\ell) > \varphi(r)$ for each rule $(\ell \to r) \in R$. The resulting language class is denoted by wr-McNL.

By requiring in addition that R is confluent, we obtain the classes con-non-win-McNL of *confluent, non-weight-increasing McNaughton languages*, con-wr-McNL of *confluent, weight-reducing McNaughton languages*, con-non-in-McNL of *confluent, non-increasing McNaughton languages*, and con-lr-McNL of *confluent, length-reducing McNaughton languages.*

Concerning the non-increasing and non-weight-increasing classes, we have the following characterization.

Theorem 6.

(a) non-in-McNL = non-win-McNL = CSL.

(b) con-non-in-McNL = con-non-win-McNL = DCSL.

Proof. (a) If $L \subseteq \Sigma^*$ is a non-increasing McNaughton language, then there exists a finite, terminating string-rewriting system R such that

(1.) $\forall(\ell \to r) \in R : |\ell| \geq |r|$, and

(2.) $\forall w \in \Sigma^* : w \in L$ iff $t_1 w t_2 \to_R^* Y$.

Given a string $w \in \Sigma^*$, a non-deterministic single-tape Turing machine can simply guess the reduction sequence $t_1wt_2 \to_R^* Y$. For simulating this sequence linear space suffices, that is, $L \in \mathsf{NSPACE}(n) = \mathsf{CSL}$.

If R is in addition confluent, then a deterministic single-tape machine can compute the left-most reduction sequence starting with t_1wt_2. This sequence ends with Y iff $w \in L$. Hence, in this case $L \in \mathsf{DSPACE}(n) = \mathsf{DCSL}$.

If $L \in \mathsf{CSL}$, then there exists a non-deterministic single-tape machine M accepting L with linear space. Without loss of generality we can assume that starting from an arbitrary configuration M halts without using additional space. As in the proof of Theorem 1 M can be simulated by a string-rewriting system R. As M always halts, R is terminating, and since M uses no extra space, R needs no rules for extending the work space to the right or to the left beyond the border markers \$ or ¢, respectively, that is, R can be chosen in such a way that all rules $(\ell \to r) \in R$ satisfy $|\ell| \geq |r|$. Hence, $L \in$ non-in-McNL. If M is deterministic, then R is in addition confluent.

Finally, let us consider the non-weight-reducing classes. Because every non-increasing system is also a non-weight-increasing one, we only have to show that a non-weight-increasing system R can be simulated by a Turing machine that is linearly space bounded. Let $c_{max} = \max\{ \varphi(a) \mid a \in \Gamma \}$, where Γ is the underlying alphabet, and φ is the corresponding weight-function. For every non-empty input w we have $\varphi(t_1wt_2) \leq c_{max} \cdot |w| + \varphi(t_1t_2) \leq c \cdot |w|$ for some constant c. Because the length of the longest possible string in a reduction sequence $t_1wt_2 \to_R^* Y$ is less than or equal to $c \cdot |w|$ – the worst case may occur if $\min\{ \varphi(a) \mid a \in \Gamma \}$ equals 1 – the space bound remains linear. Thus, non-win-McNL $\subseteq$ CSL and con-non-win-McNL $\subseteq$ DCSL, which completes the proof. □

The length- and weight-reducing classes can be characterized as follows, where GCSL denotes the class of *growing context-sensitive languages* [DW86].

Theorem 7. wr-McNL = lr-McNL = GCSL.

Proof. If $L \in$ GCSL, then $L = L(M)$ for some shrinking two-pushdown automaton M [Bun96]. From M we obtain a weight-reducing string-rewriting system R simulating the computations of M. It follows that the $L \in$ wr-McNL (see [BO98] proof of Thm. 4.3).

The inclusion wr-McNL $\subseteq$ lr-McNL can be shown exactly as the inclusion GCRL $\subseteq$ con-lr-McNL is shown in [NO98], where GCRL denotes the class of *generalized Church-Rosser languages*, which is just the class con-wr-McNL in our notation.

Finally, if $L \in$ wr-McNL, then from (R, t_1, t_2, Y) we can construct a *growing context-sensitive grammar* generating the language $t_1 \cdot L \cdot t_2$, that is, $t_1 \cdot L \cdot t_2 \in$ GCSL. Here a grammar $G = (N, T, P, S)$ is called *growing context-sensitive* if $|\ell| < |r|$ holds for each production $(\ell \to r) \in P$ satisfying $\ell \neq S$. Since GCSL is closed under the operation of removing border markers [Bun96], this yields $L \in$ GCSL. Hence, wr-McNL = lr-McNL = GCSL. □

By definition the classes con-lr-McNL and CRL coincide [MNO88]. As con-wr-McNL = con-lr-McNL is shown in [NO98], we have the following analogue to the above result.

Corollary 8. con-wr-McNL = con-lr-McNL = CRL.

It is known that the class CRL is incomparable to the class CFL of context-free languages [BO98], while it contains the class DCFL of deterministic context-free languages [MNO88]. It is, however, still an open problem of whether or not CRL contains the class UCFL of all *unambiguous context-free languages*.

Next we introduce further classes of McNaughton languages by placing additional restrictions on the string-rewriting systems that are used in their specifications.

A string-rewriting system R on Γ is called *monadic* if it is length-reducing and if $|r| \leq 1$ holds for each rule $(\ell \to r)$ of R, and it is called *special* if it is length-reducing, and the right-hand side of each rule is the empty string.

A language $L \subseteq \Sigma^*$ is a *special*, respectively, a *monadic McNaughton language* if it is defined using a finite string-rewriting system that is special, respectively monadic. The corresponding language classes are denoted as $\mathsf{sp\text{-}McNL}$ and $\mathsf{mon\text{-}McNL}$, respectively. By requiring in addition that the string-rewriting systems are confluent, we obtain the class $\mathsf{con\text{-}sp\text{-}McNL}$ of *confluent special McNaughton languages* and the class $\mathsf{con\text{-}mon\text{-}McNL}$ of *confluent monadic McNaughton languages.*

Theorem 9. $\mathsf{mon\text{-}McNL} = \mathsf{CFL}$.

Proof. Let L be a monadic McNaughton language specified by (R, t_1, t_2, Y). In [BJW82] Theorem 2.2 it is shown that for every monadic string-rewriting system R the language $\nabla_R^*(Y) = \{\, w \mid w \to_R^* Y \,\}$ is context-free. Because context-free languages are closed under intersection with regular languages, *left-derivatives* $({}_{t_1}\delta(L) := \{\, w \mid t_1 w \in L \,\})$ and *right-derivatives* $(\delta_{t_2}(L) := \{\, w \mid w t_2 \in L \,\})$, the language L is context-free.

For the converse inclusion let $G := (N, T, P, S)$ be a context-free grammar in Chomsky Normal-form, that is, every rule is either of the form $A \to BC$ or $A \to a$, where $A, B, C \in N$ and $a \in T$. In addition, G contains the rule $S \to \lambda$ if $\lambda \in L(G)$ holds. From G we construct a string-rewriting system R which contains the following rules:

$$
\begin{aligned}
R \;:=\; & \{t_1 S t_2 \to Y\} \cup \{\, t_1 a t_2 \to Y \mid (S \to a) \in P,\, a \in T \cup \{\lambda\} \,\} \\
& \cup \{\, BC \to A \mid (A \to BC) \in P \,\} \\
& \cup \{\, bC \to A \mid (A \to BC) \text{ and } (B \to b) \text{ in } P \,\} \\
& \cup \{\, Bc \to A \mid (A \to BC) \text{ and } (C \to c) \text{ in } P \,\} \\
& \cup \{\, bc \to A \mid (A \to BC),\, (B \to b), \text{ and } (C \to c) \text{ in } P \,\}.
\end{aligned}
$$

Here $\Gamma := N \cup T \cup \{Y, t_1, t_2\}$, where Y, t_1, and t_2 are new symbols. Finally let $\Sigma := T$. Then the language $L(R, t_1, t_2, Y)$ coincides with $L(G)$. □

Obviously, $\mathsf{con\text{-}mon\text{-}McNL} \subseteq \mathsf{mon\text{-}McNL}$ holds. On the other hand, we have the following inclusion.

Theorem 10. $\mathsf{REG} \subseteq \mathsf{con\text{-}mon\text{-}McNL} \subseteq \mathsf{DCFL}$.

Proof. The first inclusion follows immediately from the proof of [MNO88] Theorem 2.1, where a deterministic finite-state automaton working from left to right is simulated by a finite, monadic, and confluent string-rewriting system. However, in order to demonstrate the power of confluent monadic systems we give an alternative proof based on the characterization of the regular languages by monoids.

Let $(M, \circ)$ be a monoid, and let h a morphism that maps Σ to M. Its extension to Σ^* is defined as usual. A language $L \subseteq \Sigma^*$ is accepted by the monoid $(M, \circ)$ if L coincides with the set $\{\, w \mid w \in \Sigma^* \text{ and } h(w) \in F \,\}$ for some $F \subseteq M$.

Let $L \subseteq \Sigma^*$ be a regular language. Then there exist a finite monoid $(M, \circ)$ and a set $F \subseteq M$ that accepts L [Ber79]. From M and F we construct a finite monadic string-rewriting system R as follows:

$$\begin{aligned} R \;:=\; & \{\, t_1 e t_2 \to Y \mid e \in F \,\} \cup \{\, t_1 t_2 \to Y \mid 1_M \in F \,\} \\ & \cup \{\, t_1 a t_2 \to Y \mid a \in \Sigma \text{ and } h(a) \in F \,\} \\ & \cup \{\, ab \to g \mid a, b \in \Sigma \text{ and } h(a) \circ h(b) = g \,\} \\ & \cup \{\, af \to g \mid a \in \Sigma,\ f \in M \text{ and } h(a) \circ f = g \,\} \\ & \cup \{\, eb \to g \mid b \in \Sigma,\ e \in M \text{ and } e \circ h(b) = g \,\} \\ & \cup \{\, ef \to g \mid e, f \in M \text{ and } e \circ f = g \,\}, \end{aligned}$$

where 1_M is the identity element of $(M, \circ)$, and $\Gamma := \Sigma \cup M \cup \{Y, t_1, t_2\}$. Here we assume that Σ and M are disjoint sets, and that Y, t_1, and t_2 are three new symbols. It is easily seen that R is confluent, and that (R, t_1, t_2, Y) specifies the language L. Hence, L belongs to the class $\mathsf{con\text{-}mon\text{-}McNL}$, that is, $\mathsf{REG} \subseteq \mathsf{con\text{-}mon\text{-}McNL}$.

To prove the second inclusion let $L \in \mathsf{con\text{-}mon\text{-}McNL}$, that is, there exists a finite, confluent, and monadic string-rewriting system R such that, for all $w \in \Sigma^*$, $w \in L$ iff $t_1 w t_2 \to_R^* Y$. Thus, $t_1 \cdot L \cdot t_2 = [Y]_R \cap t_1 \cdot \Sigma^* \cdot t_2$. By [BJW82] Theorem 3.9 $[Y]_R \in \mathsf{DCFL}$, and so $t_1 \cdot L \cdot t_2 \in \mathsf{DCFL}$. It follows that $L \in \mathsf{DCFL}$. $\Box$

Actually these inclusions are proper as we will see in the following

Theorem 11. $\mathsf{con\text{-}mon\text{-}McNL} \subsetneq \mathsf{DCFL}$.

Proof. Consider the language $L := \{\ ca^n b^n \mid n \geq 1\ \} \cup \{\ da^n b^{2n} \mid n \geq 1\ \} \in \mathsf{DCFL}$. It is well-known that $L^R = \{\ b^n a^n c \mid n \geq 1\ \} \cup \{\ b^{2n} a^n d \mid n \geq 1\ \} \notin \mathsf{DCFL}$. Since the class $\mathsf{con\text{-}mon\text{-}McNL}$ is obviously closed under reversal, this means that $L \notin \mathsf{con\text{-}mon\text{-}McNL}$. $\Box$

The proof above is based on the fact that there are languages in DCFL the reversals of which are not deterministic context-free. This observation implies the question of whether a language L is in $\mathsf{con\text{-}mon\text{-}McNL}$, whenever L and L^R are both deterministic context-free. Next we will answer this question in the negative.

Let $A := \{\ a^n b^n \mid n \geq 1\ \}$ and $B := \{\ a^m b^{2m} \mid m \geq 1\ \}$, and let $L_p := A \cdot B = \{\ a^n b^n a^m b^{2m} \mid n, m \geq 1\ \}$. It is obvious that L_p as well as L_p^R is deterministic context-free.

Lemma 12. *L_p does not belong to the class* $\mathsf{con\text{-}mon\text{-}McNL}$.

Proof. Assume to the contrary that $L_p = L(R, t_1, t_2, Y)$, where R is a finite, monadic, and confluent string-rewriting system on $\Gamma := \Sigma \cup \Delta$, $\Sigma := \{a, b\}$, Δ contains the symbol $Y \in \mathrm{IRR}(R)$, and $t_1, t_2 \in \mathrm{IRR}(R)$. By general results on string-rewriting systems we can assume without loss of generality that the system R is *interreduced*, that is, for each rule $(\ell \to r) \in R$, $\ell \in \mathrm{IRR}(R \smallsetminus \{\ell \to r\})$ and $r \in \mathrm{IRR}(R)$ (see, e.g., [BO93] Section 2.2). For $w \in \Gamma^*$, $\mathrm{nf}(w)$ will be used to denote the unique irreducible descendant of w mod R.

Claim 1. For all $n \neq m$, $\mathrm{nf}(a^n b^n) \neq \mathrm{nf}(a^m b^m)$.

Proof. Assume to the contrary that $\mathrm{nf}(a^n b^n) = \mathrm{nf}(a^{n+k} b^{n+k})$ for some $n, k \geq 1$. Then $t_1 ab a^n b^{2n} t_2 \to_R^* t_1 ab\, \mathrm{nf}(a^n b^n) b^n t_2 \to_R^* Y$, as $aba^n b^{2n} \in L_p$. Analogously, we obtain

$$t_1 ab a^{n+k} b^{n+k} b^n t_2 \to_R^* t_1 ab\, \mathrm{nf}(a^{n+k} b^{n+k}) b^n t_2 = t_1 ab\, \mathrm{nf}(a^n b^n) b^n t_2 \to_R^* Y.$$

This, however, contradicts the fact that $aba^{n+k} b^{2n+k} \notin L_p$, as $2n + k < 2(n + k)$ holds. Thus, for all $n \neq m$, $\mathrm{nf}(a^n b^n) \neq \mathrm{nf}(a^m b^m)$. $\Box$

Claim 1 implies that the language $\Delta_R^*(\{\ a^n b^n \mid n \geq 1\ \}) \cap \mathrm{IRR}(R)$ is infinite. Let $\mathrm{nf}(a^n b^n) = c_1 \cdots c_m$ for some letters $c_1, \ldots, c_m \in \Gamma$. As $a^n b^n \to_R^* c_1 \cdots c_m$ and as R is monadic, a^nb^n has a factorization of the form $a^nb^n = u_1u_2\cdots u_m$ such that $u_j \to_R^* c_j$ for all $j = 1,\dots,m$. Hence, there is an index $k \in \{1,\dots,m\}$ such that $u_1 = a^{i_1}, u_2 = a^{i_2},\dots,u_{k-1} = a^{i_{k-1}}, u_k = a^{i_k}b^{j_k}, u_{k+1} = b^{j_{k+1}},\dots,u_m = b^{j_m}$, where $\sum_{\ell=1}^{k} i_\ell = n = \sum_{\ell=k}^{m} j_\ell$. Let $m(n)$ denote the length of $\mathrm{nf}(a^nb^n)$, and let $k(n)$ denote the minimal value of k among all factorizations of a^nb^n of the above form. By Claim 1 $m(n)$ is unbounded as n grows to infinity.

Claim 2. $k(n)$ is unbounded as n grows to infinity.

Proof. Assume to the contrary that $k(n) \leq \alpha$ holds for all $n \geq 1$, where $\alpha \in \mathbb{N}$ is a constant. Hence, for each $n \geq 1$ there exists an integer $j_{k(n)} \geq 0$ such that $a^nb^{j_{k(n)}} \to_R^* c_1c_2\cdots c_{k(n)} \in \bigcup_{i=0}^{\alpha} \Gamma^i =: \Gamma^{\leq\alpha}$. Thus, there are $n, m \geq 1$ such that

$$a^nb^{j_{k(n)}} \to_R^* c_1c_2\cdots c_{k(n)} = c_1c_2\cdots c_{k(n+m)} \leftarrow_R^* a^{n+m}b^{j_{k(n+m)}}$$

holds. In particular, this means that $k(n) = k(n+m)$. Now

$$t_1a^nb^{j_{k(n)}}b^{n-j_{k(n)}}ab^2t_2 \to_R^* t_1c_1\cdots c_{k(n)}b^{n-j_{k(n)}}ab^2t_2 \to_R^* Y,$$

as $a^nb^{j_{k(n)}}b^{n-j_{k(n)}}ab^2 = a^nb^nab^2 \in L_p$. Hence, we also have the following sequence of reductions:

$$t_1a^{n+m}b^{j_{k(n+m)}}b^{n-j_{k(n)}}ab^2t_2 \to_R^* t_1c_1\cdots c_{k(n)}b^{n-j_{k(n)}}ab^2t_2 \to_R^* Y,$$

which implies that $j_{k(n+m)} + n - j_{k(n)} = n + m$ holds. This in turn yields $j_{k(n)} = j_{k(n+m)} = j_{k(n)} + m$, that is, $m = 0$, thus contradicting the choice of m. Hence, we can conclude that the indices $k(n)$ are unbounded as n grows to infinity. □

By a symmetric argument we also obtain the following unboundedness result.

Claim 3. The difference $m(n) - k(n)$ is unbounded as n grows to infinity.

For each $n \in \mathbb{N}$, $a^nb^nab^2 \in L_p$, that is, we have a reduction sequence of the following form:

$$t_1a^nb^nab^2t_2 \to_R^* t_1\mathrm{nf}(a^nb^n)\mathrm{nf}(ab^2t_2) = t_1c_1\cdots c_{k(n)}\cdots c_{m(n)}t_3 \to_R^* Y,$$

where $t_3 := \mathrm{nf}(ab^2t_2)$. Let $\gamma := \max\{|t_1|, |t_2|+3\}$. Then $t_1, c_1\cdots c_{m(n)}, t_3 \in \mathrm{IRR}(R)$ satisfying $|t_1|, |t_3| \leq \gamma$.

Let $m := (|\Gamma|+1)^{|\gamma|}\cdot|R|\cdot\rho+\rho$, where $\rho := \max\{\ |\ell| \mid \ell \in \mathrm{dom}(R)\ \}$. By Claims 2 and 3 there exists an integer n such that $k(n) > m$ and $m(n)-k(n) > m$. As $t_1c_1\cdots c_{k(n)}c_{k(n)+1}\cdots c_{m(n)}t_3 \to_R^* Y$, where $|t_1|, |t_3| \leq \gamma$ and $k(n), m(n)-k(n) > m$, this implies that the prefix $t_1c_1\cdots c_{k(n)}$ or the suffix $c_{k(n)+1}\cdots c_{m(n)}t_3$ reduces to a string of length at most ρ, since R is a monadic system.

We consider the case that $t_1c_1\cdots c_{k(n)} \to_R^* u$ for some $u \in \Delta^*$, $|u| \leq \rho$, the other case being symmetric. As $t_1, c_1\cdots c_{k(n)} \in \mathrm{IRR}(R)$ and R is monadic, the redex for each step of this reduction sequence is contained in the prefix of length $|t_1| + \rho - 1$ of the actual string, that is, if $t_1c_1\cdots c_{k(n)} \to_R^* v = v_1\ell v_2 \to_R v_1rv_2$ for some $(\ell \to r) \in R$, then $|v_1| < |t_1|$. As $k(n) > (|\Gamma|+1)^{|\gamma|}\cdot|R|\cdot\rho+\rho$, this reduction sequence contains $\sigma > (|\Gamma|+1)^{|\gamma|}\cdot|R|$ steps. Hence, there are two steps in which the same prefix v_1 and the same rule $(\ell \to r)$ occur, that is,

$$t_1c_1\cdots c_{k(n)} \to_R^* v = v_1\ell v_2 \to_R v_1rv_2 \to_R^* v_1\ell v_4 \to_R v_1rv_4 \to_R^* u.$$

Thus, $v_2 = v_3 v_4$ for some $v_3 \in \Delta^+$ such that $v_1 r v_3 \to_R^* v_1 \ell$. Then there are indices $j_1 < j_2 \leq k(n)$ such that $t_1 c_1 \cdots c_{j_1 - 1} \to_R^* v_1 \ell, c_{j_1} \cdots c_{j_2} = v_3$ and $c_{j_2+1} \cdots c_{k(n)} = v_4$. Let $\beta := \sum_{\mu=j_1}^{j_2} i_\mu$, that is, $a^\beta = a^{i_{j_1}} a^{i_{j_1+1}} \cdots a^{i_{j_2}} \to_R^* c_{j_1} c_{j_{1+1}} \cdots c_{j_2} = v_3$. Then

$$\begin{array}{rcl} t_1 a^{n-\beta} b^n a b^2 t_2 & \to_R^* & t_1 c_1 \cdots c_{j_1-1} c_{j_2+1} \cdots c_{k(n)} \cdots c_{m(n)} t_3 \\ & \to_R^* & v_1 \ell c_{j_2+1} \cdots c_{k(n)} \cdots c_{m(n)} t_3 \\ & = & v_1 \ell v_4 c_{k(n)+1} \cdots c_{m(n)} t_3 \\ & \to_R^* & u c_{k(n)+1} \cdots c_{m(n)} t_3 \\ & \to_R^* & Y, \end{array}$$

which contradicts the fact that $a^{n-\beta} b^n a b^2 \notin L_p$. Thus, L_p does not belong to the class con-mon-McNL. □

This fact yields the following consequence.

Corollary 13. con-mon-McNL $\subsetneq \{ L \mid L \in$ DCFL *and* $L^R \in$ DCFL $\}$.

Based on the relationship between the various restrictions that are placed on the defining string-rewriting systems, the above inclusion results yield an inclusion hierarchy of McNaughton languages. However, within this hierarchy it remains to determine the exact position of the class sp-McNL. Further, we want to resolve the problem of which of these inclusions are proper.

Theorem 14. REG $\not\subseteq$ sp-McNL *and* con-sp-McNL $\not\subseteq$ REG.

Proof. Let $L_1 := ab^*ab^*a \in$ REG, and let R be a special string-rewriting system such that $t_1 ab^n ab^m a t_2 \to_R^* Y$ holds for all $n, m \geq 0$. Since R is a finite special system, it must contain a rule $(\ell \to \lambda)$ such that $\ell \in \{a,b\}^+$. If $\ell = b^k$ for some $k \geq 1$, then $t_1 b^k ab^n ab^m a t_2 \to_R^* Y$, and if $|\ell|_a \geq 1$, then $t_1 \ell ab^n ab^m a t_2 \to_R^* Y$, that is, $b^k ab^n ab^m a \in L$ or $\ell ab^n ab^m a \in L$, where L denotes the language specified by (R, t_1, t_2, Y). Hence, $L \neq L_1$, that is, REG $\not\subseteq$ sp-McNL.

Conversely, let $R := \{ab \to \lambda\}$, $t_1 := \lambda$, and $t_2 := Y$. Then, for all $w \in \{a,b\}^*$, $t_1 w t_2 \to_R^* Y$ iff $w \to_R^* \lambda$ iff $|w|_a = |w|_b$ and $|v|_a \geq |v|_b$ for each prefix v of w. Hence, the language L defined by (R, t_1, t_2, Y) is the Dyck language D_1^*, which is not regular. Since R is special and confluent, we see that con-sp-McNL $\not\subseteq$ REG holds. □

As con-sp-McNL $\subseteq$ con-mon-McNL, we obtain the following proper inclusions.

Corollary 15.

(a) REG $\subsetneq$ con-mon-McNL, *and* con-sp-McNL $\subsetneq$ con-mon-McNL.

(b) sp-McNL $\subsetneq$ mon-McNL.

Next we compare the class sp-McNL to the classes con-mon-McNL and DCFL. From Theorem 14 we already know that con-mon-McNL $\not\subseteq$ sp-McNL holds. However, also the converse inclusion does not hold. In fact we have the following result.

Theorem 16. sp-McNL $\not\subseteq$ DCFL.

Proof. Let $R := \{ab \to \lambda, abb \to \lambda\}$, $t_1 := \lambda$, and $t_2 := Y$, where Y is a new symbol. By L we denote the language that is specified by (R, t_1, t_2, Y). Then $L \in$ sp-McNL. Now consider the subset $L \cap a^*b^*$. It is easily seen that $L \cap a^*b^* = \{ a^n b^m \mid 0 \leq n \leq m \leq 2n \}$. From Ogden's lemma for deterministic context-free languages (see [Har79] Lemma 11.8.3) it follows easily that this language is not in DCFL, and since the class DCFL is closed under intersection with regular languages, it follows that L itself does not belong to DCFL. Thus, sp-McNL $\not\subseteq$ DCFL. $\square$

Hence, we have the following consequence.

Corollary 17. *The classes* DCFL *and* sp-McNL *are incomparable under inclusion.*

We close this section by comparing the various classes of McNaughton languages to still another class of languages. A language $L \subseteq \Sigma^*$ is called a *Church-Rosser congruential language* if there exist a finite, length-reducing, and confluent string-rewriting system R on Σ and finitely many irreducible strings $w_1, w_2, \dots, w_n \in \Sigma^*$ such that $L = \bigcup_{i=1}^{n} [w_i]_R$. In [MNO88] it is shown that the class CRCL of all Church-Rosser congruential languages is properly contained in the class CRL, and that it contains languages that are not context-free. Hence, CRCL is incomparable to the class CFL of context-free languages. Further, there are deterministic context-free languages that are not congruential, for example, the language $L := \{\ a^n b^n \mid n \geq 1\ \} \cup b^*$ has this property, and so CRCL is also incomparable to the class DCFL. In fact, the language L is even a confluent and monadic McNaughton language. To see this let $\Gamma := \{a, b, A, B, Y, ¢, \$\}$, let $t_1 := ¢$, $t_2 := \$$, and let R denote the following monadic string-rewriting system:

$$R := \{ab \to A, aAb \to A, ¢A\$ \to Y, ¢b \to B, Bb \to B, B\$ \to Y\}.$$

Then R is confluent, and it is easily verified that, for each $w \in \{a, b\}^*$, $¢w\$ \to_R^* Y$ holds if and only if $w \in L$. Hence, we have the following result.

Theorem 18. *The classes* con-mon-McNL *and* CRCL *are incomparable under inclusion.*

Here we want to compare the class CRCL to the classes con-sp-McNL and sp-McNL. For that we will make use of the following characterizations.

Lemma 19. *Let $L \subseteq \Sigma^*$ be a special McNaughton language. Then there exist an alphabet $\Gamma := \Sigma \cup \Theta \cup \{¢, \$, Y\}$, a finite special string-rewriting system $R := R_1 \cup R_2$ on Γ and irreducible strings $t_1, t_2 \in \Theta^*$ such that the following conditions are satisfied:*

(a) $(R, ¢t_1, t_2\$Y, Y)$ *specifies the language* L,

(b) $R_1 \subseteq \Sigma^+ \times \{\lambda\}$,

(c) $R_2 \subseteq (¢t_1 \cdot \Sigma^* \cdot t_2\$) \times \{\lambda\}$ *has no critical pairs, and*

(d) for all $w \in \Sigma^*$, $w \in L$ *if and only if there exists some* $\hat{w} \in \Sigma^*$ *such that* $¢t_1 w t_2\$ \to_{R_1}^* ¢t_1 \hat{w} t_2\$ \to_{R_2} \lambda$.

Proof. Let $L \subseteq \Sigma^*$ be a special McNaughton language, and let R' be a finite special string-rewriting system on $\Gamma' := \Sigma \cup \Theta$, let $t_1, t_2 \in \Theta^* \cap \mathrm{IRR}(R')$, and $Y' \in \Theta \cap \mathrm{IRR}(R')$ such that (R', t_1, t_2, Y') specifies the language L. Hence, for all $w \in \Sigma^*$, $w \in L$ iff $t_1 w t_2 \to_{R'}^* Y'$. As R' is a special system, this means that either $t_1 = Y' t_1'$ for some $t_1' \in \Theta^*$ satisfying $t_1' w t_2 \to_{R'}^* \lambda$ or $t_2 = t_2' Y'$ for some $t_2' \in \Theta^*$ satisfying $t_1 w t_2' \to_{R'}^* \lambda$. Without loss of generality we assume the latter.

Obviously only finitely many steps in this reduction sequence can involve symbols from t_1 or t_2'. Thus, by commuting independent steps we can transform this sequence into one of the following form, where $R_1 := R' \cap (\Sigma^+ \times \{\lambda\})$ and $R_3 := R' \smallsetminus R_1$:

$$t_1 w t_2' \to_{R_1}^* t_1 w_1 t_2' \to_{R_3}^* \lambda \text{ for some } w_1 \in \Sigma^*.$$

Each rule of R_3 contains a nonterminal on its left-hand side. Therefore the sequence $t_1w_1t_2' \to_{R_3}^* \lambda$ can contain at most $|t_1t_2'|$ many steps. Thus, we obtain that $|w_1| \leq |t_1t_2'|\cdot(\rho-1)$, where $\rho := \max\{ |\ell| \mid \ell \in \mathrm{dom}(R') \}$.

Now let $¢, \$$ and Y be three new symbols, and let $\Gamma := \Sigma \cup \Theta \cup \{¢, \$, Y\}$. We define a new system R_2 on Γ as follows:

$$R_2 := \{ ¢t_1w_1t_2'\$ \to \lambda \mid w_1 \in \Sigma^*, w_1 \in L, \text{ and } |w_1| \leq |t_1t_2'| \cdot (\rho - 1) \}.$$

Then R_1 and R_2 satisfy the conditions (b) and (c). Further, from the considerations above we easily see that condition (d) is satisfied as well, and that the language L is specified by the four-tuple $(R_1 \cup R_2, ¢t_1, t_2'\$Y, Y)$. □

For the class con-sp-McNL we obtain the following strengthened version of the above result.

Lemma 20. *Let $L \subseteq \Sigma^*$ be an element of the class* con-sp-McNL. *Then there exist an alphabet $\Gamma := \Sigma \cup \Theta \cup \{¢, \$, Y\}$, a finite, special, and confluent string-rewriting system $R := R_1 \cup R_2$ over Γ and irreducible strings $t_1, t_2 \in \Theta^*$ such that the following conditions are satisfied:*

(a) $(R, ¢t_1, t_2\$Y, Y)$ specifies the language L,

(b) R_1 is a confluent string-rewriting system on Σ,

(c) $R_2 \subseteq (¢t_1 \cdot \mathrm{IRR}(R_1) \cdot t_2\$) \times \{\lambda\}$ has no critical pairs, and

(d) for all $w \in \Sigma^*$, $w \in L$ if and only if there exists some $\hat{w} \in \mathrm{IRR}(R_1)$ such that $¢t_1wt_2\$ \to_{R_1}^* ¢t_1\hat{w}t_2\$ \to_{R_2} \lambda$.

Proof. This follows as the corresponding statements in the proof of the previous lemma by observing the following. If $t_1wt_2' \to_{R'}^* \lambda$, then $w \to_{R_1}^* \hat{w}$ for some $\hat{w} \in \mathrm{IRR}(R_1)$, and $t_1\hat{w}t_2' \to_{R_3}^* \lambda$. Thus, we can choose R_2 as follows:

$$R_2 := \{ ¢t_1w_1t_2'\$ \to \lambda \mid w_1 \in \mathrm{IRR}(R_1) \cap \Sigma^*, w_1 \in L_1 \text{ and } |w_1| \leq |t_1t_2'| \cdot (\rho - 1) \}.$$

Then R_1 and R_2 are both confluent, and since there is no overlap between a left-hand side of R_1 and a left-hand side of R_2, the combined system $R = R_1 \cup R_2$ is confluent, too. The remaining statements then follow as above. □

Based on this characterization we obtain the following inclusion.

Theorem 21. con-sp-McNL $\subsetneq$ CRCL.

Proof. Let $L \subseteq \Sigma^*$ be a language from con-sp-McNL, and let R be a special and confluent string-rewriting system on Γ such that, for all $w \in \Sigma^*$, $t_1wt_2 \to_R^* Y$ iff $w \in L$.

By Lemma 20 we can assume that $R = R_1 \cup R_2$, where $R_1 \subseteq \Sigma^+ \times \{\lambda\}$ is confluent, and $R_2 = \{ ¢t_1'wt_2'\$ \to \lambda \mid w \in L \cap \mathrm{IRR}(R_1), \text{ and } |w| \leq c \}$, where c is a constant depending on R, and $t_1', t_2' \in (\Gamma \smallsetminus \Sigma)^* \cap \mathrm{IRR}(R)$.

Consider the system R_1. It is finite, special, and confluent. Further, let $W := \{ w \mid ¢t_1'wt_2'\$ \in \mathrm{dom}(R_2) \} \subseteq \mathrm{IRR}(R)$. Then W is a finite set.

Claim: $L = \bigcup\limits_{w \in W} [w]_{R_1}$.

Proof. Let $w \in W$, and let $u \leftrightarrow_{R_1}^* w$. Then $t_1ut_2 = ¢t_1'ut_2'\$ \to_{R_1}^* ¢t_1'wt_2'\$ \to_{R_2} \lambda$, and hence, $t_1ut_2Y \to_R^* Y$, that is, $u \in L$. Conversely, if $u \in L$, then $¢t_1'ut_2'\$Y \to_R^* Y$, and so $u \to_{R_1}^* w$ for some $w \in W$. Thus, $L = \bigcup\limits_{w \in W} [w]_{R_1}$. □

Since $\mathsf{CRCL} \not\subseteq \mathsf{DCFL}$, the inclusion $\mathsf{con\text{-}sp\text{-}McNL} \subseteq \mathsf{CRCL}$ is proper. $\square$

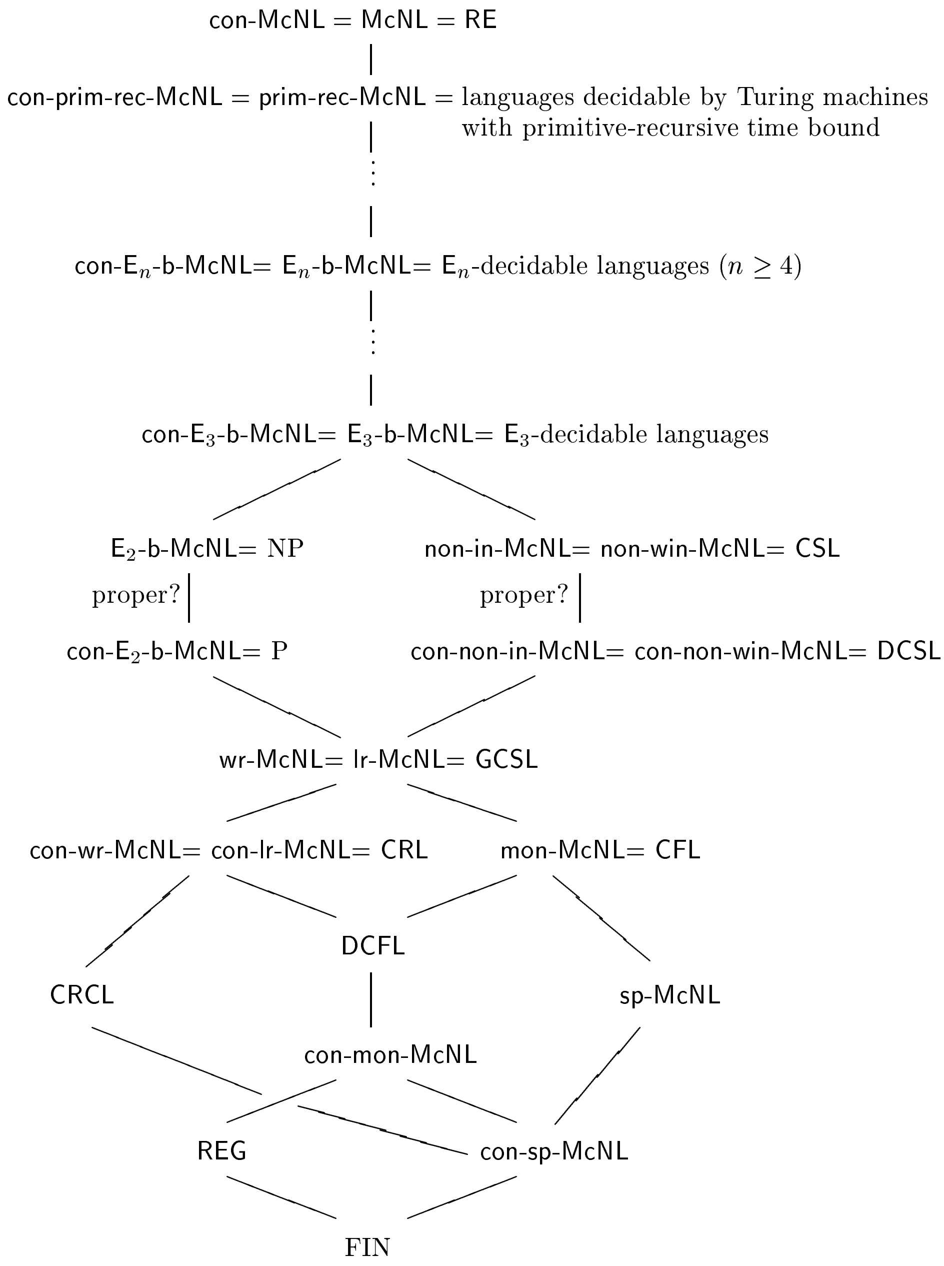

Figure 1: The hierarchy of McNaughton classes

On the other hand we have the following result.

Theorem 22. $\mathsf{sp\text{-}McNL} \not\subseteq \mathsf{CRCL}$.

Proof. Let $R := \{ab \to \lambda, abb \to \lambda\}$, $t_1 := \lambda$, $t_2 := Y$. Then the language L specified by (R, t_1, t_2, λ) belongs to $\mathsf{sp\text{-}McNL}$. Since $L = \nabla_R^*(\lambda)$, we see that each $w \in L$ satisfies the following inequalities: $|w|_a \leq |w|_b \leq 2|w|_a$, and $w = \lambda$ or $w \in a \cdot \Sigma^* \cdot b$. Assume that $L \in \mathsf{CRCL}$, that is, there exist a finite, length-reducing and confluent string-rewriting system S on $\Sigma := \{a, b\}$ and strings $w_1, \ldots, w_n \in \mathrm{IRR}(S)$ such that $L = \bigcup_{i=1}^{n} [w_i]_S = \bigcup_{i=1}^{n} \nabla_S^*(w_i)$.

Since $\{ a^n b^m \mid 0 \le n \le m \le 2n \} \subseteq L$, S must contain a rule of the form $a^k b^\ell \to w$, where $|w| < k + \ell$. Let $|w|_a = k - i$ and $|w|_b = \ell - j$. Then $|w| = k - i + \ell - j = k + \ell - i - j$, that is, $i + j > 0$.

If $j > i$, then $a^{k+\ell} b^{\ell+k} \to a^\ell w b^k =: u$, where $|u|_a = \ell + k - i$ and $|u|_b = \ell - j + k < \ell + k - i = |u|_a$. Since $a^{k+\ell} b^{\ell+k} \in L$, also $u \in L$, a contradiction. Hence, we have $j \le i$.

Assume that $j = i$, that is, $a^{k+\ell} b^{2k+2\ell} \to a^\ell w b^{\ell+2k} =: u$, where $|u|_a = \ell + k - i$ and $|u|_b = \ell - j + \ell + 2k = 2\ell + 2k - j = 2\ell + 2k - i$. Since $2|u|_a = 2\ell + 2k - 2i < 2\ell + 2k - i = |u|_b$, this implies that $u \notin L$, contradicting the choice of u.

Finally, if $j < i$, then $a^{k+\ell} b^{k+\ell} \to a^\ell w b^k =: u$, where $|u|_a = \ell + k - i > \ell + k - j = |u|_b$, again reaching the same contradiction. Thus, $L \notin \mathsf{CRCL}$, that is, $\mathsf{sp\text{-}McNL} \not\subset \mathsf{CRCL}$. □

Thus, in summary we have the hierarchy of classes of McNaughton languages depicted in Figure 1, where RE is the class of all recursively enumerable languages, and FIN denotes the class of all finite languages. All inclusions are known to be proper apart from the well-known LBA- and P-NP-problems. At this time it remains open whether or not sp-McNL is contained in CRL. Also it is a long-standing open problem whether or not each regular language is in CRCL. For a partial result showing that at least all regular languages of polynomial density are in CRCL see [Nie00].

The example languages used in establishing some of the separation results are given in the following table, where symDCFL denotes the class of deterministic context-free languages L for which L^R is also deterministic context-free. These languages nicely illustrate the power of the various language classes.

		Language	contained in
L	$=$	$\{ ca^n b^n \mid n \ge 1 \} \cup \{ da^n b^{2n} \mid n \ge 1 \}$	DCFL $\smallsetminus$ con-mon-McNL
L_p	$=$	$\{ a^n b^n a^m b^{2m} \mid n, m \ge 1 \}$	symDCFL $\smallsetminus$ con-mon-McNL
L	$=$	ab^*ab^*a	REG $\smallsetminus$ sp-McNL
D_1^*	$=$	$\{ w \in \{a,b\}^* \mid \|w\|_a = \|w\|_b \}$	con-sp-McNL $\smallsetminus$ REG
L	$=$	$\{ a^n b^n \mid n \ge 0 \} \cup \{b\}^*$	con-mon-McNL $\smallsetminus$ CRCL

Table 1: Some example languages

4 Some closure and non-closure properties

All the McNaughton classes above DCFL are well-known language or complexity classes, and as such their closure properties have been investigated before. Therefore we concentrate here on the classes con-sp-McNL, sp-McNL, and con-mon-McNL.

From the definition it is immediate that each class of McNaughton languages is closed under reversal. In addition, we will see that each class of McNaughton languages is also closed under left- and right-derivatives, where for $L \subseteq \Sigma^*$ and $u \in \Sigma^*$, ${}_u\delta(L) := \{ w \mid uw \in L \}$ denotes the *left-derivative* and $\delta_u(L) := \{ w \mid wu \in L \}$ denotes the *right-derivative* of L with respect to u.

Theorem 23. *Each McNaughton class is closed under left-derivatives and right-derivatives.*

Proof. In [MNO88] this result is proved for the class con-lr-McNL. Actually it suffices to prove this result for the special case of taking the derivative with respect to a string of length one, that is, with respect to a single letter. For this case the proof given in [MNO88] also applies to the class con-mon-McNL and all the classes containing it. Thus, it remains to consider the classes sp-McNL and con-sp-McNL.

For the class sp-McNL the proof is rather straightforward. Let R be a special string-rewriting system on $\Gamma := \Sigma \cup \Theta$, and let $t_1, t_2 \in \Theta^*$ and $Y \in \Theta$ such that (R, t_1, t_2, Y) specifies the special McNaughton language $L \subseteq \Sigma^*$. Further, let $a \in \Sigma$. We will construct a specification of the right-derivative $\delta_a(L)$ as a special McNaughton language.

Let A be a new letter, and let $\Theta' := \Theta \cup \{A\}$ and $\Gamma' := \Sigma \cup \Theta'$. Further, let $\pi : \Gamma'^* \to \Gamma^*$ be the morphism that is the identity on Γ^* and that maps A onto a, and let $t_1' := t_1$ and $t_2' := At_2$. Finally, let R' be the special string-rewriting system that is defined as follows:

$$R' := R \cup \{\, xAy \to \lambda \mid (xay \to \lambda) \in R \,\}.$$

We claim that (R', t_1', t_2', Y) specifies the language $\delta_a(L)$. Indeed, for $w \in \Sigma^*$, we have $t_1' w t_2' = t_1 w A t_2 \to_{R'}^* Y$ if and only if $t_1 w a t_2 \to_R^* Y$ if and only if $wa \in L$ if and only if $w \in \delta_a(L)$. Hence, $\delta_a(L) \in$ sp-McNL. Thus, the class sp-McNL is closed under right-derivatives. By symmetry it is also closed under left-derivatives.

Unfortunately, in the construction above the system R' need not be confluent, even if R is. For example, if $(ab \to \lambda) \in R$ and $(ba \to \lambda) \in R$, then R' contains the rules $(ab \to \lambda)$ and $(bA \to \lambda)$. Thus, $a \leftarrow_{R'} abA \to_{R'} A$, but a and A do not have a common descendant modulo R'. Hence, for the case of special systems that are confluent, we need a different construction.

Let $L \subseteq \Sigma^*$ be a language from con-sp-McNL, and let $a \in \Sigma$. We want to show that the right-derivative $\delta_a(L)$ also belongs to con-sp-McNL. We can assume that L is specified by a finite, special, and confluent string-rewriting system $R = R_1 \cup R_2$ of the form described in Lemma 20.

Let A be a new letter, and let $\Gamma' := \Gamma \cup \{A\}$. We consider the string-rewriting system

$$R_a := R_1 \cup \{\, ¢ t_1 w A t_2 \$ \to \lambda \mid w \in \Sigma^* \cap \mathrm{IRR}(R), |w| \leq (|t_1 t_2| + 1) \cdot \rho, wa \in L \,\},$$

where $\rho := \max\{\, |\ell| \mid \ell \in \mathrm{dom}(R) \,\}$. Then R_a is a finite special string-rewriting system that is confluent.

Claim. $(R_a, ¢t_1, At_2\$Y, Y)$ specifies the language $\delta_a(L)$.

Proof. If $¢t_1 w A t_2 \$ Y \to_{R_a}^* Y$, then $¢t_1 w A t_2 \$ \to_{R_a}^* \lambda$. Hence, $w \to_{R_1}^* w'$ for some $w' \in \Sigma^*$ such that $|w'| \leq (|t_1 t_2| + 1) \cdot \rho$, and $w'a \in L$. Then $t_1 w a t_2 Y \to_{R_1}^* t_1 w' a t_2 Y \to_{R_1}^* t_1 w'' t_2 Y \to_{R_2} Y$ for some $w'' \in \mathrm{IRR}(R)$, that is, $wa \in L$. Thus, $w \in \delta_a(L)$.

Conversely, if $w \in \delta_a(L)$, then $t_1 w a t_2 Y \to_R^* Y$. Hence, $w \to_{R_1}^* w'$ for some $w' \in \Sigma^* \cap \mathrm{IRR}(R)$, and $t_1 w' a t_2 Y \to_R^* Y$. Thus, $|w'| \leq (|t_1 t_2| + 1) \cdot \rho$, and $w'a \in L$. Hence, $(¢t_1 w' A t_2 \$ \to \lambda) \in R_a$, that is, $¢t_1 w A t_2 \$ Y \to_{R_a}^* ¢t_1 w' A t_2 \$ Y \to_{R_a} Y$. Thus, $(R_a, ¢t_1, t_2\$Y, Y)$ does indeed specify the language $\delta_a(L)$. □

It follows that also the class con-sp-McNL is closed under right-derivatives. Closure under left-derivatives follows by symmetry. □

It is known that the class CRL is neither closed under union nor under intersection [NO98]. Here we will see that the same is true for the classes con-mon-McNL, con-sp-McNL and sp-McNL.

Theorem 24. *The language classes* con-mon-McNL*,* con-sp-McNL*, and* sp-McNL *are not closed under union.*

Proof. Let $R_1 := \{ab \to \lambda\}$ and $R_2 := \{abb \to \lambda\}$, $t_1 := \lambda$, and $t_2 := Y$. We consider the languages L_1 and L_2 specified by (R_1, t_1, t_2, Y) and (R_2, t_1, t_2, Y), respectively. Both languages L_1 and L_2 belong to con-sp-McNL, but $L_1 \cup L_2 \notin$ DCFL, since DCFL is closed under intersection with regular languages and $(L_1 \cup L_2) \cap a^*b^* = \{ a^nb^n \mid n \geq 0 \} \cup \{ a^nb^{2n} \mid n \geq 0 \}$. Thus, the classes con-sp-McNL and con-mon-McNL are not closed under union.

Next let $R_3 := \{b \to \lambda\}$, and let $R_4 := \{¢a\$ \to \lambda\}$. Then R_3 and R_4 are finite special string-rewriting systems that are confluent. The four-tuple (R_3, λ, Y, Y) specifies the language $L_3 := b^*$, and $(R_4, ¢, \$Y, Y)$ specifies the language $L_4 := \{a\}$. Thus, $L_3, L_4 \in$ con-sp-McNL.

Let L denote the union $L := L_3 \cup L_4$. We claim that L does not belong to the class sp-McNL. Assume to the contrary that there exist a finite and special string-rewriting system R on $\Gamma := \{a, b, Y\} \cup \Theta$ and strings $t_1, t_2 \in (\Theta \cup \{Y\})^* \cap \mathrm{IRR}(R)$ such that, for all $w \in \{a, b\}^*$, $w \in L$ iff $t_1wt_2 \to_R^* Y$ holds. Since $L = \{a\} \cup \{ b^n \mid n \geq 0 \}$, we see that R must contain a rule of the form $(b^m \to \lambda)$ for some $m > 0$. Since $a \in L$, we have $t_1at_2 \to_R^* Y$. Thus, we obtain the reduction sequence $t_1ab^mt_2 \to_R t_1at_2 \to_R^* Y$, which shows that also ab^m belongs to the language that is specified by (R, t_1, t_2, Y). Since $ab^m \notin L$, this is a contradiction. Thus, L does not belong to the class sp-McNL, that is, this class is not closed under union either. □

Since mon-McNL = CFL, this class is closed under union, but it is neither closed under intersection nor under complementation. Next we will consider the operations of intersection and complementation for the three classes sp-McNL, con-sp-McNL and con-mon-McNL.

Theorem 25. *The language classes* con-sp-McNL*,* sp-McNL*, and* con-mon-McNL *are not closed under intersection.*

Proof. Let $A := \{a_1, \ldots, a_n\}$ and $B := \{b_1, \ldots, b_n\}$ such that $A \cap B = \emptyset$. Further, for each subset $I \subseteq \{1, \ldots, n\}$, we define the language D_I as the solution of the equation

$$X = \lambda + X \cdot X + \bigcup_{i \in I} a_i \cdot X \cdot b_i + \bigcup_{j \notin I} (a_j \cdot X + b_j \cdot X),$$

and by R_I we denote the special string-rewriting system

$$R_I := \{ a_ib_i \to \lambda \mid i \in I \} \cup \{ a_j \to \lambda, b_j \to \lambda \mid j \notin I \}.$$

It is easily seen that R_I is confluent and that the four-tuple (R_I, λ, Y, Y) specifies the language D_I. Thus, $D_I \in$ con-sp-McNL for all $I \subseteq \{1, \ldots, n\}$.

Now let $I, J \subseteq \{1, \ldots, n\}$ such that $I \not\subseteq J$ and $J \not\subseteq I$. We claim that $D_I \cap D_J \notin$ CFL. Let $i \in I \smallsetminus J$ and $j \in J \smallsetminus I$. Then

$$D_I \cap D_J \cap a_i^*a_j^*b_i^*b_j^* = \{ a_i^na_j^mb_i^nb_j^m \mid n, m \geq 0 \},$$

which is not context-free by Ogden's lemma [HU79]. Hence, none of three language classes mentioned in the statement of the theorem is closed under intersection. □

Concerning the complementation we have the following negative results.

Theorem 26. *The class* con-sp-McNL *is not closed under complementation.*

Proof. Let $\Sigma := \{a\}$. The language $L := \{\lambda\}$ clearly belongs to con-sp-McNL. For example, it is specified by the string-rewriting system $R := \{¢\$ \to \lambda\}$ and the strings $t_1 := ¢$ and $t_2 := \$Y$. We claim that the complement $L^c = a^+$ does not belong to con-sp-McNL.

Assume to the contrary that L^c belongs to the class con-sp-McNL. Then by Lemma 20 we can assume that there is a finite, confluent, and special string-rewriting system $R_c = R_1 \cup R_2$ on some finite alphabet $\Gamma = \{a\} \cup \Theta$, and strings $s_1, s_2 \in \Theta^* \cap \mathrm{IRR}(R_c)$ and $Y \in \Theta \cap \mathrm{IRR}(R_c)$ such that (R_c, s_1, s_2Y, Y) specifies the language L^c. Here R_1 is a confluent system on $\{a\}$, $R_2 \subseteq (s_1 \cdot \mathrm{IRR}(R_1) \cdot s_2) \times \{\lambda\}$, and for all $n \in \mathbb{N}$, $n \geq 1$ if and only if there exists some $m \geq 0$

such that $s_1 a^n s_2 \to_{R_1}^* s_1 a^m s_2 \to_{R_2} \lambda$. Since R_2 is a finite system, R_1 must contain a rule of the form $(a^i \to \lambda)$ for some $i > 0$. Since $a^i \in L^c$, this means that $s_1 a^i s_2 \to_{R_1} s_1 s_2 \to_{R_2} \lambda$ holds, that is, $(s_1 s_2 \to \lambda)$ is a rule of R_2. However, this implies that λ belongs to the language specified by $(R_c, s_1, s_2 Y, Y)$, contradicting our assumption that this language coincides with the language L^c. Thus, L^c is not a confluent special McNaughton language. $\square$

The proof above depends on Lemma 20 in a crucial way. Indeed the language a^+ does belong to the class sp-McNL, as it is specified by the system $R_c = \{a^2 \to \lambda, ¢a\$ \to \lambda, ¢a^2\$ \to \lambda\}$. Nevertheless we have the following non-closure property for sp-McNL.

Theorem 27. *The class* sp-McNL *is not closed under complementation.*

Proof. Let $\Sigma := \{a, b\}$, and let $L := a^+$. As seen above $L \in$ sp-McNL. We claim that its complement $L^c = \Sigma^* \smallsetminus L$ does not belong to sp-McNL. Observe that $L^c = \{ w \in \Sigma^* \mid |w|_b > 0 \} \cup \{\lambda\}$.

Assume to the contrary that R_c is a finite special string-rewriting system on some finite alphabet $\Gamma := \{a, b\} \cup \Theta$, and that $s_1, s_2 \in \Theta^* \cap \mathrm{IRR}(R_c)$ and $Y \in \Theta \cap \mathrm{IRR}(R_c)$ are chosen such that (R_c, s_1, s_2, Y) specifies the language L^c. Since $ba^n \in L^c$ for all $n \geq 0$, we see that R_c contains a rule of the form $(a^m \to \lambda)$ for some $m > 0$. On the other hand, since $\lambda \in L^c$, we have $s_1 s_2 \to_{R_c}^* Y$. Thus, $s_1 a^m s_2 \to_{R_c} s_1 s_2 \to_{R_c}^* Y$, which means that also the string a^m belongs to the language specified by (R_c, s_1, s_2, Y). Hence, this language does not coincide with the language L^c, that is, L^c is not in sp-McNL. $\square$

The closure of the class con-mon-McNL with respect to complementation remains open at this time.

Using essentially the same argument as in the proof above it can easily be shown that the language $L := b \cdot a^*$ is not a member of the class sp-McNL, while $L_1 := \{b\}$ and $L_2 := a^*$ are both in con-sp-McNL. Further, in Lemma 12 we have seen that $L_p = A \cdot B$ does not belong to the class con-mon-McNL, while $A = \{ a^n b^n \mid n \geq 1 \}$ and $B = \{ a^m b^{2m} \mid m \geq 1 \}$ are obviously in con-mon-McNL. Thus, we have the following non-closure property.

Corollary 28. *The classes* con-sp-McNL, sp-McNL *and* con-mon-McNL *are not closed under product.*

By Theorem 14 sp-McNL does not contain the regular languages, while $\Sigma^* \in$ con-sp-McNL holds for each finite alphabet Σ. Hence, we see that neither the class con-sp-McNL nor the class sp-McNL is closed under intersection with regular sets. On the other hand, all the classes in Figure 1 above the class DCFL are closed under intersection with regular sets. Thus, it remains to consider this operation for the class con-mon-McNL. We suspect that this class is indeed closed under this operation, but currently we have no proof yet.

Next we turn to the closure under morphisms and inverse morphisms. Here we have the following negative results.

Theorem 29. *The classes* con-sp-McNL, sp-McNL *and* con-mon-McNL *are not closed under (λ-free) morphisms.*

Proof. Let L denote the language $L := a^*$, let $\Theta := \{c, d\}$, and let $h : a^* \to \Theta^*$ be the morphism induced by $a \mapsto cd$. Then $h(L) = (cd)^*$. While L obviously belongs to the class con-sp-McNL, we claim that $h(L)$ is not even a special McNaughton language.

Assume to the contrary that $h(L)$ is specified by (R, t_1, t_2, Y), where R is a finite special string-rewriting system on some alphabet Γ containing $\Theta \cup \{Y\}$. As $t_1 (cd)^n t_2 \to_R^* Y$ holds for each $n \geq 0$, R must contain a rule of the form $(cd)^i \to \lambda$, $(cd)^i c \to \lambda$, $(dc)^i \to \lambda$ or $(dc)^i d \to \lambda$. In each case it follows that the language specified by (R, t_1, t_2, Y) also contains a string that has a substring of the form c^2. For example, if $(cd)^i \to \lambda$ is a rule of R, then $ccd(cd)^{i-1}d(cd)^{i-1} \to_R^2 \lambda$. This, however, contradicts our assumption that (R, t_1, t_2, Y) specifies the language $h(L)$. Thus, $h(L)$ is not contained in sp-McNL, that is, neither con-sp-McNL nor sp-McNL is closed under (λ-free) morphisms.

For the class con-mon-McNL, we consider the language

$$L := \{\, a^n cb^n \mid n \geq 0 \,\} \cup \{\, a^n db^{2n} \mid n \geq 0 \,\},$$

which is obviously contained in con-mon-McNL. Let $h : \{a, b, c, d\}^* \to \{a, b\}^*$ be the morphism that is defined through $a \mapsto a$, $b \mapsto b$, $c \mapsto ab$, and $d \mapsto ab^2$. Then

$$h(L) = \{\, a^n b^n \mid n \geq 1 \,\} \cup \{\, a^n b^{2n} \mid n \geq 1 \,\},$$

which is not deterministic context-free. Thus, $h(L)$ does not belong to con-mon-McNL, that is, the class con-mon-McNL is not closed under (λ-free) morphisms either. □

Concerning inverse morphisms the situation is not much better, as far as our current knowledge goes.

Theorem 30. *The classes* con-sp-McNL *and* sp-McNL *are not closed under inverse morphisms.*

Proof. Let $L \subseteq \{a, b\}^*$ be the language that is specified by (R, λ, Y, Y), where $R := \{ab \to \lambda\}$. Then $L \in$ con-sp-McNL, and it is easily seen that

$$L = \{\, w \in \{a, b\}^* \mid |w|_a = |w|_b, \text{ and for each prefix } u \text{ of } w, |u|_a \geq |u|_b \,\}.$$

Let $\Gamma := \{a, b, c, d\}$, and let $h : \Gamma^* \to \{a, b\}^*$ be the morphism that is induced by $a \mapsto a$, $b \mapsto b$, $c \mapsto ba^3$, and $d \mapsto b^2 a$. Let $L_1 := h^{-1}(L) = \{\, z \in \Gamma^* \mid h(z) \in L \,\}$. As $L \subseteq a \cdot \{a, b\}^* \cdot b \cup \lambda$, it follows that $L_1 \subseteq a \cdot \Gamma^* \cdot b \cup \lambda$.

Claim. $L_1 \notin$ sp-McNL.

Proof. Assume to the contrary that L_1 is specified by (R_1, t_1, t_2, Y), where R_1 is a finite special string-rewriting system, that is, for all $z \in \Gamma^*$, $z \in L_1$ if and only if $t_1 z t_2 \to_{R_1}^* Y$ holds. For all $n \geq 1$, we have $a(cd)^n d^n b \in L_1$, as

$$\begin{array}{rcl} h(a(cd)^n d^n b) & = & a(ba^3b^2a)^n(b^2a)^n b \quad \to_R^* \quad a(ba^2)^n(b^2a)^n b \\ & \to_R^* & a^{n+1}b^{n+1} \qquad\qquad\quad\; \to_R^* \quad \lambda. \end{array}$$

Thus, $t_1 a(cd)^n d^n b t_2 \to_{R_1}^* Y$ holds for all $n \geq 1$, which implies that R_1 contains a rule $\ell \to \lambda$ such that ℓ is a factor of $(cd)^n d^n$ for some $n \geq 1$. Hence, $t_1 \ell ab t_2 \to_{R_1} t_1 ab t_2 \to_{R_1}^* Y$, which implies that $\ell ab \in L_1$. This, however, contradicts our observation above that each non-empty element of L_1 begins with the letter a. Thus, L_1 is not a special McNaughton language. □

This implies that neither con-sp-McNL nor sp-McNL is closed under inverse morphisms. □

Currently it remains open whether or not the class con-mon-McNL is closed under inverse morphisms. The following table summarizes the closure and non-closure properties we have established for the classes con-sp-McNL, sp-McNL, and con-mon-McNL. By + we indicate that a class is closed under the corresponding operation, − indicates that it is not closed, and ? indicates that the answer is currently still open. For comparison we also include the classes CFL, CRL, and GCSL in this table.

	$\cup$	$\cap$	co	$\cdot$	$\cap$REG	inv. hom.	λ-free hom.	hom.
con-sp-McNL	−	−	−	−	−	−	−	−
sp-McNL	−	−	−	−	−	−	−	−
con-mon-McNL	−	−	?	−	?	?	−	−
DCFL	−	−	+	−	+	+	−	−
CFL=mon-McNL	+	−	−	+	+	+	+	+
CRL=con-lr-McNL	−	−	+	−	+	+	−	−
GCSL=lr-McNL	+	−	−	+	+	+	+	−

Table 2: Closure properties

We close this section by establishing some hierarchies of languages that are obtained as unions of bounded numbers of confluent and monadic, respectively special, McNaughton languages. These results are an adaptation of results obtained by Herzog for the context-free languages [Her97].

Herzog explores context-free languages with bounded nondeterminism and bounded ambiguity. The amount of ambiguity of a context-free language is measured through the number of left-most derivations of a string in the language considered, or equivalently through the number of accepting computations of a pushdown automaton for this language, while the amount of nondeterminism is defined as the product of the number of nondeterministic choices along an accepting computation of a pushdown automaton for this language (see [Her97] for details). Herzog proves that the union $\bigcup^k \mathsf{DCFL}$ consists exactly of those context-free languages that are inherently nondeterministic of degree at most k. Here $\bigcup^k \mathsf{DCFL}$ denotes the class of languages that can be expressed as a union of up to k deterministic context-free languages. In his paper he considers two families of context-free languages, which are also of interest to us.

For $k \geq 1$ and $i = 1, \dots, k$, let

$$B_{k,i} := \{\, b_1^n b_2^{i \cdot n} \mid n \geq 1 \,\},$$

$$A_{k,i} := \{ a_1^{n_1} a_2^{n_2} a_3^{n_2} \cdots a_{2i-1}^{n_i} a_{2i}^{n_1} a_{2i+1}^{n_{i+1}} a_{2i+2}^{n_{i+1}} \cdots a_{2k-1}^{n_k} a_{2k}^{n_k} \mid n_1, n_2, \dots, n_k \geq 1 \,\},$$

and

$$B_k := \bigcup_{i=1}^{k} B_{k,i} \quad \text{and} \quad A_k := \bigcup_{i=1}^{k} A_{k,i}.$$

Herzog shows that the language A_k is inherently ambiguous of degree k, that the language B_k is unambiguous and nondeterministic of degree k, and that the union $A_k \cup B_\ell$ is inherently ambiguous of degree k and nondeterministic of degree ℓ for $\ell \geq k$.

Analogously to DCFL the classes $\mathsf{con\text{-}mon\text{-}McNL}$ and $\mathsf{con\text{-}sp\text{-}McNL}$ are not closed under union (Theorem 24). Therefore we are interested in the classes

$$\bigcup^{k} \mathsf{con\text{-}mon\text{-}McNL} \quad \text{and} \quad \bigcup^{k} \mathsf{con\text{-}sp\text{-}McNL}$$

of languages that can be expressed as a union of at most k languages from $\mathsf{con\text{-}mon\text{-}McNL}$, respectively from $\mathsf{con\text{-}sp\text{-}McNL}$. Obviously, we have the following chain of inclusions:

$$\bigcup^{k} \mathsf{con\text{-}sp\text{-}McNL} \subseteq \bigcup^{k} \mathsf{con\text{-}mon\text{-}McNL} \subseteq \bigcup^{k} \mathsf{DCFL}.$$

Theorem 31. *For $k \geq 1$, the languages A_k and B_k belong to the class $\bigcup^k$ con-mon-McNL.*

Proof. It suffices to construct a specification of the form (R, t_1, t_2, Y) with a finite, monadic, and confluent string-rewriting system R for each of the languages $A_{k,i}$ and $B_{k,i}$, $k \geq 1$ and $i = 1, \ldots, k$.

Let $\Sigma := \{b_1, b_2\}$, let $\Gamma := \Sigma \cup \{¢, \$, C, Y\}$, $t_1 := ¢$, $t_2 := \$$, and

$$R := \{b_1 b_2^i \to C, b_1 C b_2^i \to C, ¢C\$ \to Y\}.$$

Then R is a finite monadic string-rewriting system on Γ, and since R does not have any non-trivial overlaps, it is obviously confluent. Further, $t_1, t_2, Y \in \mathrm{IRR}(R)$. Finally, for all $w \in \Sigma^*$, it is easily seen that $t_1 w t_2 = ¢w\$ \to_R^* Y$ iff $w = b_1^n b_2^{i \cdot n}$ for some $n \geq 1$, that is, iff $w \in B_{k,i}$. Thus, we see that $B_{k,i} \in$ con-mon-McNL.

Next we turn to the language $A_{k,i}$. Let $\Sigma := \{a_1, a_2, \ldots, a_{2k}\}$, $\Gamma := \Sigma \cup \{¢, \$, C_2, C_4, \ldots, C_{2k}, Y\}$, $t_1 := ¢$, $t_2 := \$$, and let R be the system that contains the following rules:

$$\begin{array}{lcl}
a_2 a_3 & \to & C_2, \\
a_2 C_2 a_3 & \to & C_2, \\
a_4 a_5 & \to & C_4, \\
a_4 C_4 a_5 & \to & C_4, \\
 & \vdots & \\
a_{2i-2} a_{2i-1} & \to & C_{2i-2}, \\
a_{2i-2} C_{2i-2} a_{2i-1} & \to & C_{2i-2}, \\
a_1 C_2 C_4 \cdots C_{2i-2} a_{2i} & \to & C_{2i}, \\
a_1 C_{2i} a_{2i} & \to & C_{2i}, \\
a_{2i+1} a_{2i+2} & \to & C_{2i+2}, \\
a_{2i+1} C_{2i+2} a_{2i+2} & \to & C_{2i+2}, \\
 & \vdots & \\
a_{2k-1} a_{2k} & \to & C_{2k}, \\
a_{2k-1} C_{2k} a_{2k} & \to & C_{2k}, \\
¢ C_{2i} C_{2i+2} \cdots C_{2k} \$ & \to & Y.
\end{array}$$

Then R is a finite monadic system on Γ, and as R does not have any non-trivial overlaps, it is confluent. Further, $t_1, t_2, Y \in \mathrm{IRR}(R)$. Finally, for all $w \in \Sigma^*$, it is easily seen that $t_1 w t_2 = ¢w\$ \to_R^* Y$ iff $w = a_1^{n_1} a_2^{n_2} a_3^{n_2} \cdots a_{2i-1}^{n_i} a_{2i}^{n_1} a_{2i+1}^{n_{i+1}} a_{2i+2}^{n_{i+1}} \cdots a_{2k-1}^{n_k} a_{2k}^{n_k}$ for some $n_1, n_2, \ldots, n_k \geq 1$, that is, iff $w \in A_{k,i}$. Thus, $A_{k,i} \in$ con-mon-McNL. □

Since $B_k \in \bigcup^k \mathsf{DCFL} \smallsetminus \bigcup^{k-1} \mathsf{DCFL}$ for all $k \geq 2$, the result above yields the following consequence by Theorem 11.

Corollary 32. *The classes $\bigcup^k$ con-mon-McNL, $k \geq 1$, form a strict hierarchy with respect to inclusion.*

In the proof of Theorem 11 we have seen that the deterministic context-free language $L := \{ ca^n b^n \mid n \geq 1 \} \cup \{ da^n b^{2n} \mid n \geq 1 \}$ does not belong to the class con-mon-McNL. Since the alphabets $\{a, b, c, d\}$ and $\{b_1, b_2\}$ are disjoint, the product $C_k := B_k \cdot L$ belongs to the class $\bigcup^k$ DCFL. On the other hand, since obviously $L \in \bigcup^2$ con-mon-McNL, it follows from Theorem 31 that $C_k \in \bigcup^{2k}$ con-mon-McNL. However, the number $2k$ is a sharp bound as shown by the following result.

Lemma 33. *The language $C_k := B_k \cdot L$ does not belong to the class $\bigcup^\ell$ con-mon-McNL for any $\ell < 2k$.*

Proof. If $C_k \in \bigcup^\ell$ con-mon-McNL for some ℓ, then its reversal

$$C_k^R = \{\ b^n a^n c b_2^{i\cdot m} b_1^m, b^{2n} a^n d b_2^{i\cdot m} b_1^m \mid n, m \geq 1,\ i = 1, \ldots, k\ \}$$

also belongs to $\bigcup^\ell$ con-mon-McNL, and hence, $C_k^R \in \bigcup^\ell$ DCFL. We will show that this leads to a contradiction if $\ell < 2k$.

Assume that, for some $\ell < 2k$, $L_1, \ldots, L_\ell$ are deterministic context-free languages such that $C_k^R = \bigcup_{j=1}^\ell L_j$, let μ denote the maximum of the constants from the pumping lemma for deterministic context-free languages (see [Her97] Lemma 5) for the languages L_1 to L_ℓ, and let τ denote the factorial of μ. We consider the following strings:

$$\begin{array}{rcl} x_i & := & b^\tau a^\tau c b_2^{i\cdot\tau} b_1^\tau \quad , \quad i = 1, \ldots, k, \\ y_i & := & b^{2\tau} a^\tau d b_2^{i\cdot\tau} b_1^\tau \quad , \quad i = 1, \ldots, k. \end{array}$$

Since $\ell < 2k$, at least two of these strings belong to the same language L_j. We have to distinguish between three cases.

Case 1: There exist $r, s \in \{1, \ldots, k\}$, $r \neq s$, such that $x_r \in L_j$ and $x_s \in L_j$. This yields a contradiction just as in the proof of [Her97] Lemma 11.

Case 2: There exist $r, s \in \{1, \ldots, k\}$, $r \neq s$, such that $y_r \in L_j$ and $y_s \in L_j$. This is symmetric to Case 1.

Case 3: There exist $r, s \in \{1, \ldots, k\}$ such that $x_r \in L_j$ and $y_s \in L_j$. We apply the pumping lemma for deterministic context-free languages to L_j. We mark all the letters a in $x_r = b^\tau a^\tau c b_2^{r\cdot\tau} b_1^\tau$. Then there exists a factorization $x_r = v_1 v_2 v_3 v_4 v_5$ satisfying all of the following conditions:

(1.) $v_2 \neq \lambda$,

(2.) $v_1 v_2^n v_3 v_4^n v_5 \in L_j$ for all $n \geq 0$,

(3.) v_1, v_2 and v_3 each contain marked positions, or v_3, v_4 and v_5 each contain marked positions,

(4.) $v_2 v_3 v_4$ contains at most μ marked positions, and

(5.) if $v_5 \neq \lambda$, then for all $m, n \geq 0$ and all strings u,

$$v_1 v_2^{m+n} v_3 v_4^n u \in L_j \text{ iff } v_1 v_2^m v_3 u \in L_j.$$

From conditions (2.) and (3.) it follows that $v_2 = b^\nu$ and $v_4 = a^\nu$ for some ν satisfying $0 < \nu \leq \mu$. This yields that $v_5 = a^t c b_2^{r\cdot\tau} b_1^\tau$ for some $t \leq \tau - \nu$. Let u denote the string $u := v_4 a^t d b_2^{s\cdot\tau} b_1^\tau$, and let $m := \tau/\nu$. Since $\tau = \mu!$, the number m is an integer. We obtain that

$$v_1 v_2^{m+1} v_3 u = b^{\tau + m\cdot\nu} a^\tau d b_2^{s\cdot\tau} b_1^\tau = b^{2\tau} a^\tau d b_2^{s\cdot\tau} b_1^\tau = y_s,$$

that is, $v_1 v_2^{m+1} v_3 u \in L_j$. Hence, condition (5.) above implies that also

$$v_1 v_2^{m+2} v_3 v_4 u = b^{2\tau+\nu} a^{\tau+\nu} d b_2^{s\cdot\tau} b_1^\tau \in L_j.$$

However, since $0 < \nu < \tau$, this string does not belong to the language C_k^R. As this contradicts the pumping lemma, we see that C_k^R does indeed not belong to $\bigcup^{<2k}$ DCFL. □

Hence, we have the following proper inclusion.

Corollary 34. *For each $k \geq 1$, $\bigcup^k$ con-mon-McNL is properly contained in $\bigcup^k$ DCFL.*

Analogously it can be shown that the language $(A_k \cup B_\ell) \cdot L$ will be ambiguous of degree k and nondeterministic of degree ℓ for $\ell > k$. Hence, with respect to the degree of nondeterminism and the degree of ambiguity, the classes $\bigcup^k$ con-mon-McNL yield the same strict hierarchies as the corresponding classes $\bigcup^k$ DCFL.

In fact this result can even be extended to the bounded unions of confluent special McNaughton languages as follows. For $k \geq 1$ and $i \in \{1, \ldots, k\}$, let $D_{k,i}$ denote the set of strings that is the solution of the equation

$$X = \lambda + X \cdot X + a \cdot X \cdot b^i,$$

and take $D_k := \bigcup_{i=1}^k D_{k,i}$. Further, define $E_{k,i}$ as the solution of the equation

$$X = \lambda + X \cdot X + a_1 \cdot X \cdot a_{2i} + \bigcup_{1 \leq j < i} a_{2j} \cdot X \cdot a_{2j+1} + \bigcup_{i < j \leq k} a_{2j-1} \cdot X \cdot a_{2j},$$

and let $E_k := \bigcup_{i=1}^k E_{k,i}$.

Lemma 35. *For each $k \geq 1$, D_k and E_k belong the the class $\bigcup^k$ con-sp-McNL.*

Proof. Obviously, the language $D_{k,i}$ is specified by the four-tuple $(R_i, ¢, \$Y, Y)$, where R_i denotes the following finite, special, and confluent string-rewriting system on $\Gamma := \{a, b, ¢, \$, Y\}$:

$$R := \{¢\$ \rightarrow \lambda, ab^i \rightarrow \lambda\}.$$

Further, $E_{k,i}$ is specified by $(S_i, ¢, \$Y, Y)$, where S_i is the following system on $\Theta := \{a_1, \ldots, a_{2k}, ¢, \$, Y\}$:

$$S_i := \{¢\$ \rightarrow \lambda, a_1 a_{2i} \rightarrow \lambda, a_{2j} a_{2j+1} \rightarrow \lambda\, (1 \leq j < i), a_{2j-1} a_{2j} \rightarrow \lambda\, (i < j \leq k)\}.$$

Thus, we see that $D_k, E_k \in \bigcup^k$ con-sp-McNL. □

As in the proof of [Her97] Lemma 11 it can be shown that $D_k \notin \bigcup^{k-1}$ DCFL, and hence, $D_k \notin \bigcup^{k-1}$ con-sp-McNL. Thus, we see the following.

Corollary 36. *The classes $\bigcup^k$ con-sp-McNL, $k \geq 1$, form a strict hierarchy with respect to inclusion.*

It is easily seen that the language B_k is not contained in $\bigcup^k$ con-sp-McNL. Hence, we have the following consequence.

Corollary 37. *For each $k \geq 1$, $\bigcup^k$ con-sp-McNL is properly contained in $\bigcup^k$ con-mon-McNL.*

Finally, following the proof of [Her97] Lemma 10 it can be shown that the language E_k is inherently ambiguous of degree k. Thus, already $\bigcup^k$ con-sp-McNL contains languages that are inherently ambiguous of degree k.

5 On the complexity of McNaughton languages

Finally we turn to the task of determining the degree of complexity of those McNaughton languages that are specified by monadic or special string-rewriting systems. For our considerations we will need the complexity classes AC^0, NC^1, L, LOG(DCFL), LOG(CFL), and P.

Here AC^0 is the class of languages that are accepted by uniform Boolean circuits of constant depth (and polynomial size) with AND- and OR-gates of unbounded fan-in, NC^1 is the class of languages that are accepted by uniform Boolean circuits of logarithmic depth and polynomial size with AND- and OR-gates of bounded fan-in, L is the class of languages accepted by deterministic logarithmically space-bounded Turing machines, and $\mathsf{LOG(DCFL)}$ (respectively $\mathsf{LOG(CFL)}$) is the class of languages accepted by deterministic (respectively nondeterministic) auxiliary pushdown automata with a logarithmically space-bounded work tape in polynomial time. Finally, P is the class of languages accepted by deterministic Turing machines in polynomial time.

It is known that the following chains of inclusions hold between these complexity classes (see, e.g., [Joh90]):

$$\mathsf{AC}^0 \subseteq \mathsf{NC}^1 \subseteq \mathsf{L} \subseteq \mathsf{LOG(DCFL)} \subseteq \mathsf{LOG(CFL)} \subseteq \mathsf{P}.$$

For establishing completeness or hardness of a problem (language) within a complexity class we will in general use *deterministic many-one log-space reductions.* Here a language $A \subseteq \Sigma^*$ is said to *reduce* to a language $B \subseteq \Gamma^*$ with respect to a deterministic many-one log-space reduction, if there exists a function $\varphi : \Sigma^* \to \Gamma^*$ that is computable by a logarithmically space-bounded deterministic Turing machine and that satisfies the following equivalence for each string $w \in \Sigma^*$: $w \in A \quad$ if and only if $\quad \varphi(w) \in B$.

The language A is called *log-space many-one equivalent* to B, if A reduces to B and B reduces to A. This will be expressed as $A \equiv_m^{\log} B$. Further A is *hard* for a complexity class $\mathcal{C}$ if each language $C \in \mathcal{C}$ reduces to A, and it is *complete* for $\mathcal{C}$, if it is hard for $\mathcal{C}$ and belongs to $\mathcal{C}$. Finally, two complexity classes $\mathcal{C}_1$ and $\mathcal{C}_2$ are log-space many-one equivalent, if each $L \in \mathcal{C}_1$ reduces to some $L' \in \mathcal{C}_2$, and also each $L' \in \mathcal{C}_2$ reduces to some $L \in \mathcal{C}_1$.

From the results of Section 3 we obtain the following inclusions and equivalences:

$$\mathsf{REG} \subseteq \mathsf{con\text{-}mon\text{-}McNL} \subseteq \mathsf{DCFL} \equiv_m^{\log} \mathsf{LOG(DCFL)} \subseteq \mathsf{LOG(CFL)} \equiv_m^{\log} \mathsf{CFL} = \mathsf{mon\text{-}McNL},$$

$$\mathsf{FIN} \subseteq \mathsf{con\text{-}sp\text{-}McNL} \subseteq \mathsf{sp\text{-}McNL} \subseteq \mathsf{CFL} \equiv_m^{\log} \mathsf{LOG(CFL)},$$

and

$$\mathsf{FIN} \subseteq \mathsf{con\text{-}sp\text{-}McNL} \subseteq \mathsf{con\text{-}mon\text{-}McNL} \subseteq \mathsf{DCFL} \equiv_m^{\log} \mathsf{LOG(DCFL)}.$$

Here we are interested in the complexity of the fixed and the general membership problem for certain classes of McNaughton languages.

The *fixed membership problem* for a McNaughton language $L \subseteq \Sigma^*$ is defined as follows:

INSTANCE	:	A string $w \in \Sigma^*$.
QUESTION	:	Does w belong to L?

For a class $\mathcal{C}$ of McNaughton languages we obtain the *general membership problem* for $\mathcal{C}$ as the following natural generalization of the above problem:

INSTANCE	:	A specification (R, t_1, t_2, Y) of a language $L \in \mathcal{C}$ and a string w.
QUESTION	:	Does w belong to the language L, that is, does $t_1 w t_2 \to_R^* Y$ hold?

Since $\mathsf{CFL} = \mathsf{mon\text{-}McNL}$, we see that the fixed membership problem for $\mathsf{mon\text{-}McNL}$ is $\mathsf{LOG(CFL)}$-complete, that is, for each language $L' \in \mathsf{LOG(CFL)}$ there exists a language $L \in \mathsf{mon\text{-}McNL}$ such that L' reduces to the fixed membership problem for L. Actually, from the existence of a hardest context-free language [Gre73] we see that there exists a fixed language in $\mathsf{mon\text{-}McNL}$ such that the fixed membership problem for this language is already LOG(CFL)-complete. In particular, this implies that the general membership problem for mon-McNL is LOG(CFL)-hard. The reduction sequence $t_1wt_2 \to_R^* Y$ with respect to a finite monadic string-rewriting system R can be simulated in a straightforward way by a nondeterministic log-space and polynomially time-bounded auxiliary pushdown automaton. The polynomial time-bound is obtained from the fact that all the rules of a monadic string-rewriting system are length-reducing. This means that the general membership problem for mon-McNL is also LOG(CFL)-complete. This sharply contrasts the corresponding result for context-free grammars, as the general membership problem for them is P-complete in general [JL77]. Thus, in summary we have the following results for the class mon-McNL.

Theorem 38. *The fixed and the general membership problem for the class* mon-McNL *are* LOG(CFL)*-complete.*

Next we turn our attention to the confluent monadic McNaughton languages. Although we have con-mon-McNL $\subseteq$ DCFL, the hardest deterministic context-free language given in [Sud78] is unfortunately not a confluent monadic McNaughton language, as its reversal is not deterministic context-free. Therefore, we obtain LOG(DCFL) only as an upper bound for the membership problems for the class con-mon-McNL.

Theorem 39. *The fixed and the general membership problems for the class* con-mon-McNL *belong to the complexity class* LOG(DCFL)*.*

As a lower bound we will establish L-hardness for these problems. In order to prove this result we need less powerful reductions. An NC^1 *many-one reduction* from a language $A \subseteq \Sigma^*$ to a language $B \subseteq \Gamma^*$ is given through a function $\varphi : \Sigma^* \to \Gamma^*$ of polynomial growth [Bus87] such that, for each $w \in \Sigma^*$, $w \in A$ iff $\varphi(w) \in B$ holds, and the predicate $P_\varphi(c, i, w) \iff$ 'c is the i-th symbol of $\varphi(w)$' is in NC^1. Analogously, AC^0 *many-one reductions* are defined.

For establishing the lower bound we proceed as follows. In [MNO88] DCFL $\subseteq$ CRL is proved by showing how to construct a confluent and length-reducing string-rewriting system that simulates the shift-reduce deterministic parsing of an LR(1) grammar. From [HL93] it is known that there exists a fixed linear LR(1) language L_0 which is L-complete with respect to NC^1 many-one reductions. Hence, in order to establish L-hardness for the fixed and the general membership problems for the class con-mon-McNL it suffices to construct a specification of a language $L_1 \in$ con-mon-McNL such that L_0 reduces to L_1 with respect to an NC^1 many-one reduction.

Lemma 40. *Each linear* LR(1) *language reduces to a confluent and monadic McNaughton language with respect to an* NC^1 *many-one reduction.*

Proof. Let L be a linear LR(1) language over Σ. From a linear LR(1) grammar for L an LR-style parser can be obtained as in [Har79] Section 13.4. By T we denote the finite set of tables from this construction. Since we are not interested in actually constructing a parse for the strings in L, we can simply forget about the output tape of the parser. Thus, for our purposes the parser is a deterministic pushdown automaton. From its construction we see that it has the following properties:

- At each step of a computation the contents of the pushdown is a string from the set $T_0 \cdot (VT)^*$, where $T_0 \in T$ is a special initial table, and $V := N \cup \Sigma$, where N is the set of nonterminals of the linear LR(1) grammar generating L.
- The action of the parser is specified through two functions

$$f : T \times (\Sigma \cup \{\$\}) \to \{\mathsf{shift}, \mathsf{error}\} \cup \{ \ \mathsf{reduce}\ p \mid p \in P \ \}$$

and

$$g : T \times V \to T \cup \{\mathsf{error}\},$$

where P is the set of productions of the linear $\mathsf{LR}(1)$ grammar generating L, and \$ is a special symbol.

A *configuration* of the parser is given through a string $\gamma T_1 az\$$, where $\gamma T_1 \in T_0 \cdot (VT)^*$ is the contents of the pushdown, $T_1 \in T$ is the topmost symbol on the pushdown, and $az \in \Sigma^*$ is the remaining part of the input, where $a \in \Sigma$ is the symbol currently being scanned, and $\$ \notin \Sigma$ is used as a delimiter signalling the end of the input.

By $\vdash$ we denote the single-step computation relation of the parser. The configuration *con* that is obtained from $\gamma T_1 az\$$ in a single step is defined as follows:

(1.) if $f(T_1, a) = \mathsf{shift}$ and $g(T_1, a) = T_2$, then $con := \gamma T_1 a T_2 z\$$;

(2.) if $az = \lambda$, $f(T_1, \$) = \mathsf{reduce}\ p$, where $p = (S \to r) \in P$ and S is the start symbol of the linear $\mathsf{LR}(1)$ grammar, and $\gamma = T_0\gamma_1$ for some γ_1 satisfying $|\gamma_1| = 2 \cdot |r|$, then $con := \mathsf{accept}$;

(3.) if case (2.) does not apply, but $f(T_1, a) = \mathsf{reduce}\ p$, where $p = (A \to r) \in P$, $|r| = m$, and $\gamma = \gamma_1 T' \gamma_2$, $|\gamma_2| = 2m$, and $g(T', A) = T_2$, then $con := \gamma_1 T' A T_2 az\$$;

(4.) $con := \mathsf{error}$ in all other cases.

Given a string $w \in \Sigma^*$ as input, the parser starts in the initial configuration $T_0 w\$$. If $w \notin L$, then after a finite number of steps it will reach the error configuration. On the other hand, if $w \in L$, then it will eventually reach the accept configuration. As the underlying grammar is linear, we see that a computation of the parser consists of two phases. In phase 1 it only performs shift operations, thus shifting a prefix of the given input together with the corresponding tables onto the pushdown. Phase 2 starts with the first reduce operation that is performed. After a reduce operation some shift operations may follow, but their number is bounded by the length of the longest right-hand side of a production of the grammar. Thus, phase 2 consists of a sequence of reduce operations such that between two successive reduce operations a sequence of at most c shift operations takes place, where c is a constant depending on the grammar.

Now the announced reduction will proceed as follows. From the parser we will construct a finite, monadic and confluent string-rewriting system that simulates the second phase of the computation of the parser. Given a string $w \in \Sigma^*$, an NC^1 reduction will transform w into an encoding of the configuration of the parser that corresponds to the end of the first phase of its computation on input w. This NC^1 reduction will generate this encoding and verify that indeed it corresponds to the intended configuration. Then using the monadic string-rewriting system this string reduces to some special symbol Y if and only if the parser accepts on input w. Thus, in this way the language L will be reduced to a language $L' \in \mathsf{con\text{-}mon\text{-}McNL}$.

During a reduce operation the length of the actual configuration of the parser may remain unchanged, if the underlying production is length-preserving. Further, the shift operations between reduce operations will even increase the length of the actual configuration. Thus, in order to simulate this phase of the parser's computation by a monadic string-rewriting system we must compress several symbols into a single one.

A derivation in the linear $\mathsf{LR}(1)$ grammar can contain at most c_1 consecutive applications of length-preserving productions without containing a loop, where c_1 is the number of non-terminals in the grammar. However, since the parser is a deterministic device, we can revise it such that, instead of entering a loop, it simply enters the error state. Thus, in phase 2 of the parser's computation we have at most c_1 reduce operations preserving the length of the actual configuration and c shift operations before a reduce operation is used that actually reduces the length of the actual configuration. Observe that length-preserving productions do not necessitate any intermediate shift operations. Accordingly we need new 'super' symbols that encode a window of sufficient size for storing the topmost part of the pushdown and the actual input symbol.

The result of a reduce operation depends on the topmost table in the pushdown that is not deleted during this reduce operation. As the string-rewriting system is to be monadic, this means that this symbol together with the symbols generated by the reduce operation has to be stored in the new super symbol. Further, the effect of up to c shift operations must be stored in this way. Hence, we need super symbols that encode windows of size up to $(c+2)\cdot 2+1$, where we need up to $(c+2)\cdot 2$ places for storing the topmost part of the pushdown and one place for storing the actual input symbol.

The NC^1 reduction will transform an input string of the form $w = a_1a_2\cdots a_n$, $a_1,\ldots,a_n \in \Sigma$, into the following string:

$$\cent T_0 a_1' T_1 \cdots a_{j-1}' T_{j-1} [a_j T_j, a_{j+1}] a_{j+2} \cdots a_n \$\$,$$

where $T_0, T_1, \ldots, T_j \in T$, $a_1', \ldots, a_{j-1}' \in \Sigma'$, which is an alphabet in one-to-one correspondence to Σ, $[a_jT_j, a_{j+1}], \cent, \$$ are new symbols, and

- $f(T_i, a_{i+1}) = \mathsf{shift}$ and $g(T_i, a_{i+1}) = T_{i+1}$ for all $i = 0, 1, \ldots, j-1$, and
- $f(T_j, a_{j+1}) = \mathsf{reduce}\ p$ for some production p.

As the parser is fixed, this transformation can indeed be performed by an NC^1 reduction. Let Γ denote the alphabet

$$\begin{aligned} \Gamma \ := \ & \Sigma \cup \Sigma' \cup N \cup T \cup \{\cent, \$, E, Y\} \cup \\ & \{\ [a_1T_1 \cdots a_mT_m, b] \mid 1 \le m \le c+2, a_i \in \Sigma \cup N, T_i \in T, 1 \le i \le m, b \in \Sigma \cup \{\$\}\ \}, \end{aligned}$$

and let R be the finite string-rewriting system that consists of the following rules:

(i) Rules to simulate shift operations:

$$\begin{array}{lcll}
[a_1T_1 \cdots a_mT_m, b]c & \to & [a_1T_1 \cdots a_mT_m bT_{m+1}, c] & \text{if } m < c+2, c \in \Sigma, f(T_m, b) = \mathsf{shift} \\
 & & & \ \text{and } g(T_m, b) = T_{m+1}, \\
[a_1T_1 \cdots a_mT_m, b]\$ & \to & [a_1T_1 \cdots a_mT_m bT_{m+1}, \$] & \text{if } m < c+2, f(T_m, b) = \mathsf{shift} \\
 & & & \ \text{and } g(T_m, b) = T_{m+1}, \\
[a_1T_1 \cdots a_mT_m, b]c & \to & E & \text{if } m = c+2 \text{ and } f(T_m, b) = \mathsf{shift}, \\
 & & & \ \text{or } m < c+2, f(T_m, b) = \mathsf{shift} \\
 & & & \ \text{and } g(T_m, b) = \mathsf{error}, \\
[a_1T_1 \cdots a_mT_m, b]\$ & \to & E & \text{if } m = c+2 \text{ and } f(T_m, b) = \mathsf{shift}, \\
 & & & \ \text{or } m < c+2, f(T_m, b) = \mathsf{shift} \\
 & & & \ \text{and } g(T_m, b) = \mathsf{error}.
\end{array}$$

Here $a_i \in \Sigma \cup N$, $T_i \in T$, $b \in \Sigma$, and $c \in \Sigma$.

(ii) Rules to simulate reduce operations:

$T_0 a'_1 T_1 \cdots T_{j-1}[a_j T_j \cdots a_m T_m, \$] \to Y$ for $j = 1, \ldots, m$,
 if $f(T_m, \$) = \mathsf{reduce}\ p$ and $p = (S \to a_1 \cdots a_m)$,
$T_0 a'_1 T_1 \cdots T_{j-1}[a_j T_j \cdots a_m T_m, \$] \to E$ for $j = 1, \ldots, m$,
 if $f(T_m, \$) = \mathsf{reduce}\ p$ and $p = (S \to b_1 \cdots b_m)$ for some $b_1 \cdots b_m \neq a_1 \cdots a_m$,
$a'_1 T_1 \cdots T_{j-1}[a_j T_j \cdots a_m T_m, b] \to [a_1 T_1 A \hat{T}_2, b]$ for $j = 2, \ldots, m$,
 if $f(T_m, b) = \mathsf{reduce}\ p$, $p = (A \to a_2 \cdots a_m)$ and $g(T_1, A) = \hat{T}_2$,
$a'_1 T_1 a'_2 T_2 \cdots T_{j-1}[a_j T_j \cdots a_m T_m, b] \to E$ for $j = 1, \ldots, m$,
 if $f(T_m, b) = \mathsf{reduce}\ p$, $p = (A \to a_2 \cdots a_m)$ and $g(T_1, A) = \mathsf{error}$,
 or $f(T_m, b) = \mathsf{reduce}\ p$ and $p = (A \to b_2 \cdots b_m)$ for some $b_2 \cdots b_m \neq a_2 \cdots a_m$,
$[a_1 T_1 \cdots a_j T_j a_{j+1} T_{j+1} \cdots a_m T_m, b] \to [a_1 T_1 \cdots a_j T_j A \hat{T}_{j+1}, b]$
 if $f(T_m, b) = \mathsf{reduce}\ p$, $p = (A \to a_{j+1} \cdots a_m)$ and $g(T_j, A) = \hat{T}_{j+1}$,
$[a_1 T_1 \cdots a_j T_j a_{j+1} T_{j+1} \cdots a_m T_m, b] \to E$
 if $f(T_m, b) = \mathsf{reduce}\ p$, $p = (A \to a_{j+1} \cdots a_m)$ and $g(T_j, A) = \mathsf{error}$,
 or $f(T_m, b) = \mathsf{reduce}\ p$ and $p = (A \to b_{j+1} \cdots b_m)$ for some $b_{j+1} \cdots b_m \neq a_{j+1} \cdots a_m$.

Here $a'_i \in \Sigma'$, $T_i \in T$, $a_j \in \Sigma \cup N$, and $b \in \Sigma \cup \{\$\}$.

(iii) Rules to clean up encodings of final configurations:

$$\left.\begin{array}{lcl} xE & \to & E \\ Ex & \to & E \end{array}\right\} \text{ for all } x \in \Gamma,$$

$$\left.\begin{array}{lcl} Yy & \to & Y \\ yY & \to & Y \end{array}\right\} \text{ for all } y \in \Gamma \smallsetminus \{E\}.$$

As the parser cannot get into a loop performing reduce operations with productions of the form $A \to B$, where $A, B \in N$, the system R is terminating. As f and g are functions, there are no overlaps between the various rules simulating the parser. Hence, R is a finite system that is terminating and confluent. Further, it is easily seen that $w = a_1 \cdots a_n \in L$ if and only if

$$¢T_0 a'_1 T_1 \cdots a'_{j-1} T_{j-1}[a_j T_j, a_{j+1}] a_{j+2} \cdots a_n \$\$ \to_R^* Y.$$

Thus, the reduction above reduces the language L to the McNaughton language specified by $(R, ¢, \$\$, Y)$. Unfortunately, the system R is not monadic in the sense of our definition, since it may contain length-preserving rules. Hence, we must transform the system R into a finite, monadic and confluent system. Fortunately, this is easily done.

First we replace the right-hand side of each rule of R by its unique irreducible descendant modulo R. Let R' denote the system thus obtained. Then R' is still finite, terminating, and confluent, R' and R generate the same Thue congruence $\leftrightarrow^*$ on Γ^*, and $\mathrm{IRR}(R') = \mathrm{IRR}(R)$ (see, e.g., [BO93] Lemma 2.2.11). Further, R' can be decomposed as $R' = R_1 \cup R_2$, where R_1 consists of length-reducing rules only, and $R_2 \subseteq \Gamma \times \Gamma$.

From the definition of R we see that the left-hand side of each rule of R_1 is irreducible modulo R_2. Thus, R_1 is a finite, monadic, and confluent system. As the string $¢T_0 a'_1 T_1 \cdots T_{j-1}[a_j T_j, a_{j+1}] a_{j+2} \cdots a_n \$\$$ is also irreducible modulo R_2, we see that

$$¢T_0 a'_1 T_1 \cdots a'_{j-1} T_{j-1}[a_j T_j, a_{j+1}] a_{j+2} \cdots a_n \$\$ \to_R^* Y$$

if and only if

$$¢T_0a'_1T_1 \cdots a'_{j-1}T_{j-1}[a_jT_j, a_{j+1}]a_{j+2} \cdots a_n\$\$ \to^*_{R_1} Y.$$

Hence, L has been reduced to the confluent monadic McNaughton language that is specified by $(R_1, ¢, \$\$, Y)$. □

This gives the following result.

Theorem 41. *The fixed and the general membership problems for the class* con-mon-McNL *are* L*-hard with respect to* NC^1 *many-one reductions.*

Finally we turn to the two classes of special McNaughton languages. There we have the following situation.

Theorem 42. *The fixed and the general membership problems for the class* con-sp-McNL *are* NC^1*-hard with respect to* AC^0 *many-one reductions, and they belong to* LOG(DCFL).

Proof. The containment in LOG(DCFL) is obvious due to the inclusion con-sp-McNL $\subseteq$ con-mon-McNL. The lower bound is obtained as follows.

According to [Bus87] the *Boolean formula evaluation problem* is NC^1-complete under AC^0 many-one reductions. This problem is defined as follows:

INSTANCE	:	A completely bracketed Boolean formula $\alpha \in \{0, 1, (,), \wedge, \vee, \sim\}^*$.
TASK	:	Determine the truth value of α.

Let $\Sigma := \{0, 1, (,), \wedge, \vee, \sim\}$, $\Gamma := \Sigma \cup \{\#, Y\}$, $t_1 := \#$, $t_2 := Y$, and let R be the special string-rewriting system that consists of the following rules:

$$\begin{array}{lcl@{\qquad\qquad}lcl}
(0 \wedge 0)1 & \to & \lambda & (0 \vee 0)1 & \to & \lambda \\
(0 \wedge 1)1 & \to & \lambda & 0(0 \vee 1) & \to & \lambda \\
(1 \wedge 0)1 & \to & \lambda & 0(1 \vee 0) & \to & \lambda \\
0(1 \wedge 1) & \to & \lambda & 0(1 \vee 1) & \to & \lambda \\
0(\sim 0) & \to & \lambda & (\sim 1)1 & \to & \lambda \\
\#1 & \to & \lambda. & & &
\end{array}$$

As R has no non-trivial overlaps, it is confluent. Hence, the language L specified by (R, t_1, t_2, Y) belongs to con-sp-McNL. We will show that the Boolean formula evaluation problem reduces to the membership problem for L.

Let $\alpha \in \Sigma^*$ be a Boolean formula. We encode α by inserting 0 in front of every opening bracket and 1 after each closing bracket. For instance, if $\alpha = ((0 \vee 1) \wedge (\sim 0))$, then we obtain the string $0(0(0 \vee 1)1 \wedge 0(\sim 0)1)1$. Obviously, this transformation can be realized through an AC^0 reduction.

We claim that the formula α is true if and only if its encoding $e(\alpha)$ belongs to L. In fact, for each operator $op \in \{\wedge, \vee, \sim\}$ the corresponding rules reduce a substring of the form $0(a_1 \, op \, a_2)1$ to the truth value of the subformula $(a_1 \, op \, a_2)$. Thus, if the formula α is true, then $\#e(\alpha)Y \to^*_R \#1Y \to_R Y$. On the other hand, if α is false, then $\#e(\alpha)Y \to^*_R \#0Y \in \mathrm{IRR}(R)$.

For example, for the formula $\alpha = ((0 \vee 1) \wedge (\sim 0))$ we obtain the following reduction sequence:

$$\#0(0(0 \vee 1)1 \wedge 0(\sim 0)1)1Y \to^2 \#0(1 \wedge 1)1Y \to \#1Y \to Y.$$

This completes the proof of Theorem 42. □

For the special non-confluent McNaughton languages we obtain the following bounds, generalizing the construction in the proof above.

Theorem 43. *The fixed and the general membership problems for the class* sp-McNL *are* LOG(CFL)*-complete with respect to* NC^1 *many-one reductions.*

Proof. For deriving these completeness results we make use of the following problem. Let $(G, \circ)$ be a *groupoid*, where $G = \{1, 2, \ldots, m\}$, and let $F \subset G$ be a set of *target elements*. Thus, $\circ$ is a binary operation on G, and 1 is an identity element for $\circ$. However, in contrast to the situation for monoids the operation $\circ$ is in general not associative. Thus, if $(i_1, i_2, \ldots, i_n)$ is a sequence of elements of G, then by $G(i_1, i_2, \ldots, i_n)$ we denote the set of all elements of G that can be obtained by parenthesizing this sequence and then evaluating the resulting expression in G. Now the *evaluation problem* for (G, F) is the following decision problem:

INSTANCE	:	A sequence $(i_1, i_2, \ldots, i_n)$ of elements of G.
QUESTION	:	Can this sequence be evaluted to give an element of F, that is, is $G(i_1, i_2, \ldots, i_n) \cap F$ non-empty?

It is known that there exists a fixed finite groupoid $(G, \circ)$ and a fixed subset F of G such that the evaluation problem for (G, F) is LOG(CFL)-complete under NC^1 reductions ([BLM93] Corollary 3.4). For the following considerations let $(G, \circ)$ be such a groupoid, $G = \{1, \ldots, m\}$, and let F be a corresponding set of target elements.

Let $\Sigma := \{a_1, a_2, \ldots, a_m, \#, \$\}$, $u_G := \$a_1 \cdots a_m\$$, $u_i := \$a_1 \cdots a_i$, and $\bar{u}_i := a_{i+1} \cdots a_m\$$ for $i = 1, \ldots, m$. Then $u_i\bar{u}_i = u_G$ holds for all $1 \leq i \leq m$. Further, let $\Gamma := \Sigma \cup \{C, D, Y\}$, $t_1 := C$, $t_2 := DY$, and let R be the finite special string-rewriting system containing the following rules:

$$\begin{array}{lll} \# & \to \lambda & \\ u_G & \to \lambda & \\ \bar{u}_k u_i \# u_j \# & \to \lambda & \text{if } i \circ j = k \text{ in } G, \text{ and} \\ C u_i D & \to \lambda & \text{if } i \in F. \end{array}$$

Finally, a sequence $(i_1, i_2, \ldots, i_n)$ of elements of G is encoded through the string

$$u_G^n u_{i_1} \#^n u_G^n u_{i_2} \#^n \ldots u_G^n u_{i_n} \#^n.$$

Observe that the encoding of the sequence $(i_1, i_2, \ldots, i_n)$ can be performed by an NC^1 reduction.

Now using the reduction relation induced by R the multiplication $i \circ j$ of two elements i and j of G can be simulated. First one has to remove the separating u_G^n block as well as the separating #-symbols, except one # that must remain in order to apply the necessary rule for the multiplication. For instance, if $i \circ j = \ell_1$ in G, then

$$\begin{array}{rl} u_G^n u_i \#^n u_G^n u_j \#^n u_G^n u_k \#^n & \to_R^* \; u_G^n u_i \#^n u_j \#^n u_G^n u_k \#^n \\ & \to_R^* \; u_G^n u_i \# u_j \#^n u_G^n u_k \#^n \\ & \to_R^* \; u_G^{n-1} u_{\ell_1} \#^{n-1} u_G^n u_k \#^n \end{array}$$

and if $j \circ k = \ell_2$, then

$$\begin{array}{rl} u_G^n u_i \#^n u_G^n u_j \#^n u_G^n u_k \#^n & \to_R^* \; u_G^n u_i \#^n u_G^n u_j \#^n u_k \#^n \\ & \to_R^* \; u_G^n u_i \#^n u_G^n u_j \# u_k \#^n \\ & \to_R^* \; u_G^n u_i \#^n u_G^{n-1} u_{\ell_2} \#^{n-1}. \end{array}$$

Thus, for a sequence $(i_1, i_2, \ldots, i_n)$ of elements of G there exists a bracketing such that the corresponding evaluation yields an element of F if and only if there exists a reduction from the string $u_G^n u_{i_1} \#^n u_G^n u_{i_2} \#^n \ldots u_G^n u_{i_n} \#^n$ to a string of the form u_i with $i \in F$, and this holds if and only if $t_1 u_G^n u_{i_1} \#^n u_G^n u_{i_2} \#^n \dots u_G^n u_{i_n} \#^n t_2$ reduces to Y. Thus, the evaluation problem for (G, F) reduces to the membership problem for the special McNaughton language that is specified by (R, t_1, t_2, Y). Hence, the fixed membership problem for sp-McNL is complete for LOG(CFL). Obviously, the above construction implies that also the general membership problem for sp-McNL is complete for LOG(CFL). □

The following table summarizes the complexity results derived in this section.

	fixed membership	general membership
mon-McNL	LOG(CFL)-complete under $\leq_m^{\log}$	LOG(CFL)-complete under $\leq_m^{\log}$
con-mon-McNL	$\subseteq$ LOG(DCFL) L-hard under $\leq_m^{\mathsf{NC}^1}$	$\subseteq$ LOG(DCFL) L-hard under $\leq_m^{\mathsf{NC}^1}$
sp-McNL	LOG(CFL)-complete under $\leq_m^{\mathsf{NC}^1}$	LOG(CFL)-complete under $\leq_m^{\mathsf{NC}^1}$
con-sp-McNL	$\subseteq$ LOG(DCFL) NC^1-hard under $\leq_m^{\mathsf{AC}^0}$	$\subseteq$ LOG(DCFL) NC^1-hard under $\leq_m^{\mathsf{AC}^0}$

Table 3: Complexity results

For the classes of confluent special or monadic McNaughton languages the given upper and lower bounds for the membership problems do not match. Thus, some work remains to be done in order to obtain a complete classification.

6 Conclusion

By considering various classes of confluent and non-confluent string-rewriting systems we have obtained two families of McNaughton classes. As it turned out this approach is as powerful as the notion of the phrase-structure grammar or that of the Turing machine. In fact, many Turing machine time-complexity classes as well as the upper classes of the Chomsky hierarchy can be characterized in this way.

The inclusion relations between the various classes of McNaughton languages considered could be settled with two famous exceptions: the P-NP-problem as well as the LBA-problem occur as special cases.

Then we turned to those McNaughton languages that are specified by monadic or special string-rewriting systems. Here several closure and non-closure results have been obtained, but in particular for the class con-mon-McNL various questions remain.

Here is a short list containing the open problems that we encountered:

1. Is the class sp-McNL contained in CRL?
2. Is the class UCFL contained in CRL? In particular, is the language $\{ ww^R \mid w \in \{a, b\}^* \}$ of all palindromes of even length a Church-Rosser language [MNO88]? Further, it is not known whether or not sp-McNL is contained in UCFL, however, we expect that sp-McNL contains some context-free languages that are inherently ambiguous.
3. Is REG contained in CRCL?
4. Is the class con-mon-McNL closed under complementation, intersection with regular sets, or inverse morphisms?

5. Are the fixed and the general membership problems for con-sp-McNL LOG(DCFL)-hard? Find a proof or more efficient algorithms.

6. Are these problems for the class con-mon-McNL LOG(DCFL)-hard?

We have seen that neither the class con-sp-McNL nor the class sp-McNL contain all the regular sets. Thus, it is certainly of interest to investigate the properties of the intersections REG $\cap$ con-sp-McNL and REG $\cap$ sp-McNL. What closure properties do they have? How can the languages in these intersections be characterized?

Finally, as we have obtained a hierarchy of classes of McNaughton languages the questions of the following form are very natural:

INSTANCE	:	A McNaughton language L of some type T_1.
QUESTION	:	Is L a McNaughton language of type T_2?

For which types T_1 and T_2 is this question decidable, and in the affirmative what is the resulting degree of complexity?

The various classes of McNaughton languages have been obtained as a generalization of the Church-Rosser languages. The Church-Rosser congruential languages can be seen as the *pure* variants of the Church-Rosser languages. Accordingly, this notion can also be generalized similarly, thus defining a hierarchy of *McNaughton congruential languages*.

Also instead of only distinguishing between confluent and non-confluent string-rewriting systems we could consider various classes of *weakly confluent* systems [MNOZ93], which would lead to a family consisting of classes of *weakly confluent McNaughton languages*.

References

[Ber79] J. Berstel. *Transductions and Context-free Languages*. Teubner Studienbücher, Teubner-Verlag, Stuttgart, 1979.

[Bir98] J.C. Birget. Infinite string rewrite systems and complexity. *Journal of Symbolic Computation*, 25:759–793, 1998.

[BJW82] R.V. Book, M. Jantzen, and C. Wrathall. Monadic Thue systems. *Theoretical Computer Science*, 19:231–251, 1982.

[BLM93] F. Bédard, F. Lemieux, and P. McKenzie. Extensions to Barrington's M-program model. *Theoretical Computer Science*, 107:31–61, 1993.

[BO93] R.V. Book and F. Otto. *String-Rewriting Systems*. Springer-Verlag, New York, 1993.

[BO98] G. Buntrock and F. Otto. Growing context-sensitive languages and Church-Rosser languages. *Information and Computation*, 141:1–36, 1998.

[Bun96] G. Buntrock. *Wachsende kontext-sensitive Sprachen*. Habilitationsschrift, Fakultät für Mathematik und Informatik, Universität Würzburg, July 1996.

[Bus87] S. Buss. The Boolean formula value problem is in ALOGTIME. In *Proc. of 19th STOC*, pages 123–131. ACM Press, New York, 1987.

[DW86] E. Dahlhaus and M. Warmuth. Membership for growing context-sensitive grammars is polynomial. *Journal of Computer and System Sciences*, 33:456–472, 1986.

[Gre73] S. Greibach. The hardest context-free language. *SIAM Journal on Computing*, 2:304–310, 1973.

[Grz53] A. Grzegorczyk. Some classes of recursive functions. *Rozprawy Matematycne*, 4:1–45, 1953.

[Har79] M. Harrison. *Introduction to Formal Language Theory.* Addison-Wesley, Reading, Massachusetts, 1979.

[Her97] Ch. Herzog. Pushdown automata with bounded nondeterminism and bounded ambiguity. *Theoretical Computer Science*, 181:141–157, 1997.

[HL93] M. Holzer and K.-J. Lange. On the complexities of linear LL(1) and LR(1) grammars. In Z. Ésik, editor, *Proceedings of the 9th International Conference on Fundamentals of Computation Theory*, Lecture Notes in Computer Science 710, pages 299–308. Springer-Verlag, Berlin, 1993.

[HU79] J.E. Hopcroft and J.D. Ullman. *Introduction to Automata Theory, Languages, and Computation.* Addison-Wesley, Reading, Massachusetts, 1979.

[Hue80] G. Huet. Confluent reductions: abstract properties and applications to term rewriting systems. *Journal of the Association for Computing Machinery*, 27:729–821, 1980.

[JL77] N.D. Jones and W.T. Laaser. Complete problems for deterministic polynomial time. *Theoretical Computer Science*, 3:105–117, 1977.

[Joh90] D.S. Johnson. A catalog of complexity classes. In J. van Leeuwen, editor, *Handbook of Theoretical Computer Science*, pages 67–161. The MIT Press/Elsevier, Cambridge/Amsterdam, 1990.

[MNO88] R. McNaughton, P. Narendran, and F. Otto. Church-Rosser Thue systems and formal languages. *Journal of the Association for Computing Machinery*, 35:324–344, 1988.

[MNOZ93] K. Madlener, P. Narendran, F. Otto, and L. Zhang. On weakly confluent monadic string-rewriting systems. *Theoretical Computer Science*, 113:119–165, 1993.

[MO85] K. Madlener and F. Otto. Pseudo-natural algorithms for the word problem for finitely presented monoids and groups. *Journal of Symbolic Computation*, 1:383–418, 1985.

[Nie00] G. Niemann. Regular languages and Church-Rosser congruential languages. In R. Freund and A. Kelemenová, editors, *Grammar Systems 2000, Proceedings*, pages 359–370. Silesian University, Opava, 2000.

[NO98] G. Niemann and F. Otto. The Church-Rosser languages are the deterministic variants of the growing context-sensitive languages. In M. Nivat, editor, *Foundations of Software Science and Computation Structures, Proceedings FoSSaCS'98*, Lecture Notes in Computer Science 1378, pages 243–257. Springer-Verlag, Berlin, 1998.

[Sud78] H. Sudborough. On the tape complexity of deterministic context-free languages. *Journal of the Association for Computing Machinery*, 25:405–414, 1978.

[Tou84] G.J. Tourlakis. *Computability.* Reston Publishing Company, Reston, Virginia, 1984.