Journal of Automated Reasoning (2005) 34: 365–385
DOI: 10.1007/s10817-005-9024-8

Termination Proofs for String Rewriting Systems via Inverse Match-Bounds

ALFONS GESER[1], DIETER HOFBAUER[2] and JOHANNES WALDMANN[3]
[1]*Hochschule für Technik, Wirtschaft und Kultur (FH) Leipzig, Fachbereich EIT, Postfach 30 11 66, D-04251, Leipzig, Germany. e-mail: geser@fbeit.htwk-leipzig.de*
[2]*Mühlengasse 16, D-34125, Kassel, Germany. e-mail: dieter@theory.informatik.uni-kassel.de*
[3]*Hochschule für Technik, Wirtschaft und Kultur (FH) Leipzig, Fachbereich IMN, Postfach 30 11 66, D-04251, Leipzig, Germany. e-mail: waldmann@imn.htwk-leipzig.de*

Abstract. Annotating a letter by a number, one can record information about its history during a rewrite derivation. In each rewrite step, numbers in the reduct are updated depending on the redex numbering. A string rewriting system is called *match-bounded* if there is a global upper bound to these numbers. Match-boundedness is known to be a strong sufficient criterion for both termination and preservation of regular languages. We show that the string rewriting systems whose inverse (left- and right-hand sides exchanged) is match-bounded also have exceptional properties, but slightly different ones. Inverse match-bounded systems need not terminate; they effectively preserve context-free languages; their sets of normalizable strings and their sets of immortal strings are effectively regular. These languages can be used to decide the normalization, the uniform normalization, the termination, and the uniform termination problem for inverse match-bounded systems. We also prove that the termination problem is decidable in linear time and that a certain strong reachability problem is decidable, thereby solving two open problems of McNaughton's. Like match-bounds, inverse match-bounds entail linear derivational complexity on the set of terminating strings.

Key words: string rewriting systems, semi-Thue systems, termination, normalization, reachability, context-free languages, regular languages, match-bounded, inhibitor.

1. Introduction

The termination and uniform termination problems are undecidable for string rewriting systems (also called semi-Thue systems). The two problems amount to the membership and emptiness problems for the set of immortal strings, that is strings that initiate an infinite derivation. For any class of string rewriting systems where this set is effectively a regular language, the two problems are decidable. By "effectively regular" we mean that there is an algorithm that constructs a finite automaton accepting the language. This is the basis of a new automated termination criterion.

By annotating a letter with a number, its *match height*, one can record information about its history during a reduction. In each rewrite step, the reduct numbering is updated in such a way that its minimal match height exceeds the

minimal match height in the redex. A string rewriting system is called *match-bounded* if there is an upper bound to these match heights. Every match-bounded system can be encoded into a *deleting* system [13]. The descendant relation of a deleting system can be decomposed into components that are known to preserve regularity. So match-bounded systems effectively preserve regularity of languages [9]. This can be used to decide match-boundedness by a fixed bound via an automaton construction. Moreover, match-bounded string rewriting systems terminate and have linear derivational complexity.

In the present article, we focus on string rewriting systems whose *inverse* is match-bounded; that is, by exchanging left- and right-hand sides of the rules, one gets a match-bounded system. This class covers McNaughton's inhibitor systems [15] and the "special" grammars from Ginsburg and Greibach [11, Section IV], for instance. In contrast to match-bounded systems, inverse match-bounded systems need not terminate.

We observe that the set of normalizing strings is effectively regular for inverse match-bounded systems. Hence, it is decidable whether such systems are normalizing. As another consequence of the above mentioned decomposition result, inverse match-bounded systems effectively preserve context-freeness of languages. This shows that certain strong reachability problems are decidable, thereby solving an open problem of McNaughton's [15].

Our main achievement is to prove that for an inverse match-bounded system the set of immortal strings is effectively regular. As explained before, this proves that the termination and the uniform termination problem are decidable for this class of string rewriting systems. In particular, the termination problem is decidable in linear time, solving another open problem of McNaughton's. The derivational complexity on the set of terminating strings is shown to be linearly bounded.

The strength of this approach lies in the possibility to decide termination on given languages from any family that is closed under intersection with regular sets and for which emptiness is decidable, instead of merely deciding uniform termination. Algorithms for constructing finite automata accepting the language of immortal strings have already been presented by McNaughton [15] and Sénizergues [18] for the classes of inhibitor systems and for certain one-rule systems, respectively.

The article is organized as follows. After recalling definitions and essential results on deleting and match-bounded rewriting systems in Sections 3 and 4, we study inverse match-boundedness in Section 5 (normalization properties, preservation of context-freeness, reachability properties) and Section 6 (termination properties). Section 7 gives an in-depth view into the constructions. Derivational complexity is investigated in Section 8. Finally, we briefly describe an implementation of our algorithms in Section 9.

Some of the results from Section 6 have been presented at the 28th International Symposium on Mathematical Foundations of Computer Science MFCS 2003 at Bratislava, Slovak Republic [8].

2. Preliminaries

Standard notations for strings and string rewriting can be found, for instance, in Book and Otto [2]. A *string rewriting system* over an alphabet Σ is a relation $R \subseteq \Sigma^* \times \Sigma^*$, inducing the *rewrite relation* $\to_R = \{(x\ell y, xry) \mid x, y \in \Sigma^*, (\ell, r) \in R\}$ on Σ^*. Unless indicated otherwise, all rewriting systems are finite. Pairs (ℓ, r) from R are frequently referred to as *rules* $\ell \to r$, and by $\mathrm{lhs}(R)$ and $\mathrm{rhs}(R)$ we denote the sets of left- and right-hand sides of R, respectively. The reflexive and transitive closure of $\to_R$ is $\to_R^*$, often abbreviated as R^*, and $\to_R^+$ or R^+ denotes the transitive closure. An *R-derivation* is a (finite or infinite) sequence $(x_0, x_1, \ldots)$ with $x_i \to_R x_{i+1}$ for all $i > 0$. Define the set of *immortal* strings $\mathrm{Im}(R)$ as the set of all $x_0 \in \Sigma^*$ that initiate an infinite R-derivation. We call R *terminating on* $L \subseteq \Sigma^*$ if $\mathrm{Im}(R) \cap L = \emptyset$. If R is terminating on Σ^*, we say that R is *terminating*.

In order to classify lengths of derivations for terminating systems, define the *derivation height* function modulo R on Σ^* by $\mathrm{dh}_R(x) = \max\{n \in \mathbb{N} \mid \exists y \in \Sigma^* : x \to_R^n y\}$. The *derivational complexity* of R is defined as the function $n \mapsto \max\{\mathrm{dh}_R(x) \mid |x| \leq n\}$ on $\mathbb{N}$.

A rewriting rule $\ell \to r$ is called *context-free* if $|\ell| \leq 1$, and a rewriting system is context-free if all its rules are. Throughout we use ϵ for the *empty string* and $|x|$ for the *length* of a string x.

The composition of two relations $\rho \subseteq A \times B$ and $\sigma \subseteq B \times C$ is $\rho \circ \sigma = \{(a, c) \mid \exists b \in B : (a, b) \in \rho, (b, c) \in \sigma\}$. The restriction of $\rho \subseteq A \times B$ to $A' \subseteq A$ is $\rho|_{A'} = \{(a, b) \in \rho \mid a \in A'\}$. For $\rho \subseteq A \times B$ let $\rho(a) = \{b \in B \mid (a, b) \in \rho\}$ for $a \in A$ and $\rho(A') = \bigcup_{a \in A'} \rho(a)$ for $A' \subseteq A$. The inverse of ρ is $\rho^- = \{(b, a) \mid (a, b) \in \rho\} \subseteq B \times A$. Define $\mathrm{Inf}(\rho) = \{a \in A \mid \rho(a) \text{ is infinite}\}$; the relation ρ is *finitely branching* if $\mathrm{Inf}(\rho) = \emptyset$.

The set of *descendants* of a language $L \subseteq \Sigma^*$ modulo some string rewriting system R is $R^*(L)$. The system R is said to *preserve regularity (context-freeness)* if $R^*(L)$ is a regular (context-free) language whenever L is.

A relation $s \subseteq \Sigma^* \times \Gamma^*$ is a *substitution* if $s(\epsilon) = \{\epsilon\}$ and $s(xy) = s(x)s(y)$ for $x, y \in \Sigma^*$, so s is uniquely determined by the languages $s(a)$ for $a \in \Sigma$. For a family of languages $\mathcal{L}$ over Γ, the substitution s is an $\mathcal{L}$-substitution if $s(a) \in \mathcal{L}$ for $a \in \Sigma$. For instance, if $\mathcal{L}$ is the family of finite (context-free) languages, then s is a *finite* (*context-free*) *substitution*. If $\epsilon \notin s(a)$ for every $a \in \Sigma$, then s is *epsilon-free*. Note that a finite substitution is finitely branching, and the same holds for the inverse of a finite and epsilon-free substitution.

3. Deleting String Rewriting Systems

In this section, we briefly recall definitions and results regarding *deleting* string rewriting systems [13], a topic that can be traced back to Hibbard [12]. This class of string rewriting systems enjoys a strong decomposition property. As an

immediate consequence, deleting systems preserve regularity of languages, and inverse deleting systems preserve context-freeness. All these results will be frequently used in the sequel. For proofs and for a description of the decomposition algorithm, we refer to [13].

DEFINITION 1. *A string rewriting system R over an alphabet Σ is* $>$-deleting *for an irreflexive partial ordering $>$ on Σ (a* precedence*) if $\epsilon \notin \mathrm{lhs}(R)$, and if for each rule $\ell \to r$ in R and for each letter a in r, there is some letter b in ℓ with $b > a$. The system R is deleting if it is $>$-deleting for some precedence $>$.*

PROPOSITION 1 ([13]). *Every deleting string rewriting system is terminating and has linear derivational complexity.*

Furthermore, we have an effective decomposition result for deleting systems.

Throughout the paper, an existence is called *effective* if there is an algorithm that constructs a witness. This presumes that all inputs are effectively given. In particular, a language is called *effectively regular* if there effectively exists a finite automaton accepting it. And a rewriting system R *effectively preserves regularity* if there is an algorithm that, given a finite automaton accepting a language L, constructs an automaton accepting $R^*(L)$. The notions *effectively context-free* and *effectively preserving context-freeness* are used analogously.

THEOREM 1 ([13]). *Let R be a deleting string rewriting system over Σ. Then there effectively exist an extended alphabet $\Gamma \supseteq \Sigma$, a finite substitution $s \subseteq \Sigma^* \times \Gamma^*$, and a context-free string rewriting system C over Γ such that*

$$R^* = (s \circ C^{-}{}^*) \cap (\Sigma^* \times \Sigma^*).$$

For a binary relation ρ and a property P, we say that ρ satisfies *inverse P* if ρ^- satisfies P. So R is called inverse deleting if R^- is deleting.

COROLLARY 1 ([12, 13]). *Every inverse deleting string rewriting system effectively preserves context-freeness.*

COROLLARY 2 ([13]). *Every deleting string rewriting system effectively preserves regularity.*

In the present paper, we will frequently refer to a slightly specialized version of the above theorem.

COROLLARY 3. *Let R be a deleting string rewriting system over Σ such that $\epsilon \notin \mathrm{rhs}(R)$. Then there effectively exists an extended alphabet $\Gamma \supseteq \Sigma$, an epsilon-free finite substitution $s \subseteq \Sigma^* \times \Gamma^*$, and an epsilon-free context-free substitution $c \subseteq \Sigma^* \times \Gamma^*$ such that*

$$R^* = s \circ c^{-}.$$

As a direct consequence, we get $R^{-}{}^{*} = R^{*-} = (s \circ c^{-})^{-} = c \circ s^{-}$.

Proof. By Theorem 1, we have $R^{*} = (s \circ C^{-}{}^{*}) \cap (\Sigma^{*} \times \Sigma^{*})$, where $s \subseteq \Sigma^{*} \times \Gamma^{*}$ is a finite substitution and C is a context-free string rewriting system over Γ. Reviewing the construction in [13], we find that s is epsilon-free and that no ϵ can occur on either side of C, thus $C^{*} \cap (\Sigma^{*} \times \Gamma^{*})$ coincides with an epsilon-free context-free substitution $c \subseteq \Sigma^{*} \times \Gamma^{*}$. Note that $C^{-}{}^{*} = C^{*-}$. □

REMARK 1. The assumption $\epsilon \notin \mathrm{rhs}(R)$ in Corollary 3 cannot be dropped. Consider the example $R = \{aa \to \epsilon\}$, and assume $R^{*} = s \circ c^{-}$ for substitutions s and c. We have $c(\epsilon) = \{\epsilon\}$, since c is a substitution. Now, $(aa, \epsilon) \in R^{*}$ implies $(aa, \epsilon) \in s$, thus $\epsilon \in s(a)$. But then we have $(a, \epsilon) \in s \circ c^{-}$, contradicting $(a, \epsilon) \notin R^{*}$. Note that $R = \{a \to \epsilon\}$ is not a counterexample, as in this case $R^{*} = s$ for the substitution $s : a \mapsto \{a, \epsilon\}$.

4. Match-Bounded String Rewriting Systems

The theory of deleting systems can be applied to obtain results for *match-bounded* rewriting. A derivation is match-bounded if dependencies between rule applications are limited. To make this precise, we annotate positions in strings by natural numbers that indicate their *match height*. Positions in a reduct will get height $h + 1$ if the minimal height of all positions in the corresponding redex was h. In this section, we summarize essential results from [8, 9].

Given an alphabet Σ, define the morphisms $\mathrm{lift}_c : \Sigma^{*} \to (\Sigma \times \mathbb{N})^{*}$ for $c \in \mathbb{N}$ by $\mathrm{lift}_c : a \mapsto (a, c)$, $\mathrm{base} : (\Sigma \times \mathbb{N})^{*} \to \Sigma^{*}$ by $\mathrm{base} : (a, c) \mapsto a$, and $\mathrm{height} : (\Sigma \times \mathbb{N})^{*} \to \mathbb{N}^{*}$ by $\mathrm{height} : (a, c) \mapsto c$. For $x \in \mathbb{N}^{*}$ let $\min(x)$ denote the minimum over x; we leave $\min(\epsilon)$ undefined because we do not need it.

For a string rewriting system R over Σ with $\epsilon \notin \mathrm{lhs}(R)$ define the rewriting system

$$\mathrm{match}(R) = \{\ell' \to \mathrm{lift}_c(r) \mid (\ell \to r) \in R, \mathrm{base}(\ell') = \ell,\ c = 1 + \min(\mathrm{height}(\ell'))\}$$

over alphabet $\Sigma \times \mathbb{N}$. For instance, the system $\mathrm{match}(\{ab \to bc\})$ contains the rules $a_0b_0 \to b_1c_1$, $a_0b_1 \to b_1c_1$, $a_1b_0 \to b_1c_1$, $a_1b_1 \to b_2c_2$, $a_0b_2 \to b_1c_1, \ldots$, writing x_c as abbreviation for (x, c). For nonempty R, the system $\mathrm{match}(R)$ is always infinite.

Every $\mathrm{match}(R)$-derivation corresponds to an R-derivation of the same length (i.e., for $x, y \in (\Sigma \times \mathbb{N})^{*}$, if $x \to_{\mathrm{match}(R)} y$, then $\mathrm{base}(x) \to_R \mathrm{base}(y)$) and vice versa (i.e., for $v, w \in \Sigma^{*}$ and $x \in (\Sigma \times \mathbb{N})^{*}$, if $v \to_R w$ and $\mathrm{base}(x) = v$, then there is $y \in (\Sigma \times \mathbb{N})^{*}$ such that $\mathrm{base}(y) = w$ and $x \to_{\mathrm{match}(R)} y$). In particular, for $n \in \mathbb{N}$ we have $R^{n} = \mathrm{lift}_0 \circ \mathrm{match}(R)^{n} \circ \mathrm{base}$, thus

$$R^{*} = \mathrm{lift}_0 \circ \mathrm{match}(R)^{*} \circ \mathrm{base}.$$

It is convenient to compare the height vectors of strings that have the same base. For $u, v \in (\Sigma \times \mathbb{N})^*$ we write $u \geq v$ if $\text{base}(u) = \text{base}(v)$ and $\text{height}(u) \geq_n \text{height}(v)$, where $\geq_n$ denotes the pointwise greater-or-equal ordering on $\mathbb{N}^n$. The relations $\geq$ and $\rightarrow_{\text{match}(R)}$ commute.

LEMMA 1 ([9]). $\geq \circ \rightarrow_{\text{match}(R)} \subseteq \rightarrow_{\text{match}(R)} \circ \geq$.

DEFINITION 2. *A string rewriting system R over Σ is called* match-bounded *for $L \subseteq \Sigma^*$ by $c \in \mathbb{N}$ if $\epsilon \notin \text{lhs}(R)$ and*

$$\text{match}(R)^*(\text{lift}_0(L)) \subseteq (\Sigma \times \{0, \ldots, c\})^*.$$

System R is match-bounded *for L if there is a constant c such that R is match-bounded for L by c. If we omit L, then it is understood that $L = \Sigma^*$.*

Obviously, a system that is match-bounded for L is also match-bounded for any subset of L by the same bound. Further, by Lemma 1, if R is match-bounded for L, then R is match-bounded for $R^*(L)$, again by the same bound.

For a match-bounded system R, the infinite system $\text{match}(R)$ may be replaced by a finite restriction. Denote by $\text{match}_c(R)$ the restriction of $\text{match}(R)$ to the alphabet $\Sigma \times \{0, 1, \ldots, c\}$, that is

$$\text{match}_c(R) = \text{match}(R) \cap ((\Sigma \times \{0, \ldots, c\})^* \times (\Sigma \times \{0, \ldots, c\})^*).$$

LEMMA 2 ([9]). *If R is match-bounded for L by c, then*

$$R^*|_L = (\text{lift}_0 \circ \text{match}_c(R)^* \circ \text{base})|_L.$$

LEMMA 3 ([9]). *For all $c \in \mathbb{N}$, the system* $\text{match}_c(R)$ *is deleting.*

Dually we know that deleting systems are match-bounded. Lemma 3 makes results from Section 3 applicable. By Proposition 1, match-bounded systems terminate and have linearly bounded derivational complexity. Further, Corollary 2 implies that match-bounded systems preserve regularity; therefore, match-boundedness by a given bound is decidable. These results are stated here for later reference.

THEOREM 2 ([9]). *If R is match-bounded for L, then R is terminating on L.*

PROPOSITION 2 ([9]). *Every match-bounded string rewriting system has linear derivational complexity.*

THEOREM 3 ([9]). *If R is match-bounded for a regular language L by c,* then $R^*(L)$ *is effectively regular.*

THEOREM 4 ([9]). *The following problem is decidable:*

GIVEN: *A string rewriting system R;* a regular language L; $c \in \mathbb{N}$.
QUESTION: *Is R match-bounded for L by c?*

5. Inverse Match-Bounded String Rewriting Systems

Our focus in this article is on string rewriting systems that are inverse match-bounded, that is systems obtained from match-bounded ones by exchanging left and right hand sides of rules. Inverse match-bounded systems have both interesting applications and nice decidability properties. In this section, we show that the set of normalizing strings can be effectively determined for this class of rewriting systems; therefore normalization is decidable. Further, since context-free languages are preserved, we get decidability of a strong version of the reachability problem.

EXAMPLE 1. Peg solitaire is a one-person game. The objective is to remove pegs from a board. A move consists of one peg X hopping over an adjacent peg Y, landing on the empty space on the opposite side of Y. After the hop, Y is removed. Peg solitaire on a one-dimensional board corresponds to the string rewriting system

$$P = \{\blacksquare\blacksquare\square \to \square\square\blacksquare,\ \square\blacksquare\blacksquare \to \blacksquare\square\square\},$$

where $\blacksquare$ stands for "peg" and $\square$ for "empty." One is interested in winning positions, that is the language of all positions that can be reduced to one single peg, which is $P^{-}{}^*(\square^*\blacksquare\square^*)$. Regularity of $P^{-}{}^*(\square^*\blacksquare\square^*)$ is a "folklore theorem" [16]. Ravikumar [17] proved regularity using *change bounds*. Change bounds are closely related to match bounds but defined only for length-preserving string rewriting systems.

The system P is inverse match-bounded by 2, so we obtain yet another proof of that result. Note that in this example, P and its inverse P^{-} are isomorphic.

EXAMPLE 2. McNaughton [15] introduced a class of string rewriting systems that has good decidability properties. A system R is called an *inhibitor system* if there is a letter $\iota \notin \Sigma$, the *inhibitor*, such that $\ell \in \Sigma^+$ and $r \in (\Sigma \cup \{\iota\})^* \setminus \Sigma^*$ for every rule $\ell \to r$ in R. Each inhibitor system is inverse deleting for the ordering that makes the inhibitor ι greater than every other letter; hence it is inverse match-bounded by 1.

For example, the inhibitor system $R = \{baa \to aa\iota babba\}$ is inverse match-bounded by 1 as height 2 does not occur in

$$\text{match}_2(R^-)^*(\{a_0, b_0, \iota_0\}^*) = (\{a_0, \iota_0, b_0\} \cup b_1\{a_1, b_1\}^* a_1^2)^*.$$

The automaton for $\text{match}_2(R^-)^*(\{a_0, b_0, \iota_0\}^*)$ can be constructed because $\text{match}_2(R^-)$ is deleting by Lemma 3, hence effectively regularity preserving by Corollary 2.

EXAMPLE 3. The systems $\{ab \to ba\}$ and $\{abb \to ba\}$ are not inverse match-bounded [9].

5.1. NORMALIZATION PROPERTIES

A string is called *normalizing* if it has a descendant that is in normal form. A string rewriting system is called normalizing if every string is.

THEOREM 5. *For an inverse match-bounded string rewriting system, the set of normalizing strings is effectively regular.*

Proof. The set of normal forms is $\text{NF}(R) = \Sigma^* \setminus (\Sigma^* \cdot \text{lhs}(R) \cdot \Sigma^*)$, which is therefore regular. Hence by Theorem 3, the set of normalizing strings, $R^-{}^*(\text{NF}(R))$, is effectively regular. □

Note that a system R is normalizing if and only if $R^-{}^*(\text{NF}(R)) = \Sigma^*$, so normalization of inverse match-bounded systems is decidable.

COROLLARY 4. *The following problem is decidable:*

GIVEN: *An inverse match-bounded string rewriting system R.*
QUESTION: *Is R normalizing?*

EXAMPLE 4. The system $R = \{b^2ab^3 \to ab^6a\}$ is normalizing although it admits the loop $b^2ab^6 \to_R ab^6ab^3 \to ab^4ab^6a$ [6]. We get $\text{match}_2(R^-)^*(\{a_0, b_0\}^*) = (\{a_0, b_0\} \cup (b_1^2)^+ a_1 (b_1^3)^+)^*$; hence R is inverse match-bounded by 1. The construction of $R^-{}^*(\text{NF}(R)) = \Sigma^*$ by Theorem 5 yields another proof that R is normalizing.

5.2. PRESERVING CONTEXT-FREENESS

Inverse match-bounded systems preserve context-free languages. This result will be used in Sections 5.3 and 6.

THEOREM 6. *For a context-free language L, if a string rewriting system R is inverse match-bounded for $R^*(L)$, then $R^*(L)$ is effectively context-free.*

Proof. Let $\bar{L} = R^*(L)$, and let R^- be match-bounded for $\bar{L}$ by c. For $\rho = \text{lift}_0 \circ \text{match}_c(R^-)^* \circ \text{base}$ we get $R^{-*}|_{\bar{L}} = \rho|_{\bar{L}}$ by Lemma 2. Intersection with $\bar{L} \times \bar{L}$ on both sides of this equation yields $R^{-*} \cap (\bar{L} \times \bar{L}) = \rho \cap (\bar{L} \times \bar{L})$, which by $R^{-*} = R^{*-}$ is equivalent to $R^* \cap (\bar{L} \times \bar{L}) = \rho^- \cap (\bar{L} \times \bar{L})$. We have $R^*(\bar{L}) \subseteq \bar{L}$, and by definition of ρ one proves $\rho^-(\bar{L}) \subseteq \bar{L}$, hence $R^*|_{\bar{L}} = \rho^-|_{\bar{L}}$, so $R^*(L) = \rho^-(L)$ from $L \subseteq \bar{L}$.

The system $\text{match}_c(R^-)$ is deleting by Lemma 3, so $\text{match}_c(R^-)^-$ effectively preserves context-freeness by Corollary 1. Also, the inverse morphisms base^- and lift_0^- effectively preserve context-freeness, so the same is true for $\rho^- = \text{base}^- \circ \text{match}_c(R^-)^{-*} \circ \text{lift}_0^-$. □

In order to apply Theorem 6, we need to establish that R is inverse match-bounded for $R^*(L)$. This can be done by using Theorem 4 by verifying that R is inverse match-bounded for some regular language $M \supseteq R^*(L)$. The obvious choice $M = \Sigma^*$ leads to the following corollary.

COROLLARY 5. *Every inverse match-bounded string rewriting system effectively preserves context-freeness.*

REMARK 2. Under the weaker assumption that the rewriting system R is inverse match-bounded just for the context-free language L, we cannot guarantee that $R^*(L)$ is again context-free. This is shown by the following example. Consider $R = \{\epsilon \to abc, ac \to ca, ab \to ba, bc \to cb\}$ over alphabet $\{a, b, c\}$ and the language $L = \{\epsilon\}$. It is not difficult to see that $R^*(L) \cap c^*b^*a^* = \{c^n b^n a^n \mid n \geq 0\}$, a language that is well known not to be context-free. The system R, however, is inverse match-bounded for L by 0, since L contains normal forms modulo R^- only.

EXAMPLE 5. Ginsburg and Greibach [11, Section IV] consider grammars where each rule has a nonempty string of nonterminals as left-hand side, and at least one occurrence of a terminal letter in the right-hand side, and they show that these grammars generate context-free languages. Corollary 5 provides another proof of this result. Indeed, by the same argument as for inhibitor systems, such a grammar G is always inverse match-bounded by 1 since in G^--derivations, all terminal letters have height 0.

EXAMPLE 6. The system $R = \{ab \to da, ac \to acc\}$ is inverse match-bounded by 1. For the context-free language $L = \{ab^n c^n \mid n \geq 1\}$ we get the context-free language $R^*(L) = \{d^{n_1} ab^{n_2} c^n \mid n_1, n_2 \geq 0, n_1 + n_2 = n \geq 1\} \cup \{d^n ac^m \mid m > n \geq 1\}$.

5.3. REACHABILITY PROPERTIES

We have seen that inverse match-bounded string rewriting systems effectively preserve context-freeness. As an application, we obtain decidability of a strong version of the reachability problem. The reachability problem for a class $\mathcal{C}$ of string rewriting systems is defined as

GIVEN: A string rewriting system $R \in \mathcal{C}$; two strings x and y.
QUESTION: Does $x \to_R^* y$ hold?

For the class $\mathcal{C}$ of terminating systems, reachability is easily decided by generating the finite tree of derivations starting from x and checking whether y appears in this tree. By symmetry, one solves the problem for the class of inverse terminating systems, which includes the inverse match-bounded systems. This is generalized considerably by the following result.

THEOREM 7. *The following problem is decidable:*

GIVEN: *An inverse match-bounded string rewriting system R; a context-free language L; a regular language M.*
QUESTION: *Does $\exists x \in L\ \exists y \in M : x \to_R^* y$ hold?*

Proof. The question is equivalent to $R^*(L) \cap M \neq \emptyset$, so decidability is a consequence of the fact that $R^*(L)$ is effectively context-free by Corollary 5. □

Note that this cannot be extended to context-free languages M, even in the special case where $R = \emptyset$. The reason is that the question whether $L \cap M = \emptyset$ holds for given context-free languages L and M is undecidable.

COROLLARY 6. *The following problem is decidable:*

GIVEN: *An inverse match-bounded string rewriting system R over Σ; two strings $x, y \in \Sigma^*$.*
QUESTION: *Does $\exists u, v \in \Sigma^* : x \to_R^* uyv$ hold?*

Proof. Employ Theorem 7 for $L = \{x\}$ and $M = \Sigma^*\{y\}\Sigma^*$.

EXAMPLE 7. Recall the inhibitor systems from Example 2. The reachability problem is decidable for the class of inhibitor systems [15]. An alternative proof is by Theorem 7. Corollary 6 affirmatively answers McNaughton's [15] Open Question 4: Is the following problem decidable?

GIVEN: *An inhibitor string rewriting system R over Σ; strings $x, y \in \Sigma^*$.*
QUESTION: *Does $\exists u, v \in \Sigma^* : x \to_R^* uyv$ hold?*

6. Decidable Termination Properties

In this section, we prove that termination is decidable for inverse match-bounded systems. This is done by showing that the set of immortal strings is effectively regular for inverse deleting, and so for inverse match-bounded systems. Examples illustrate our approach.

LEMMA 4. *Let* $c \subseteq \Sigma^* \times \Gamma^*$ *be a substitution, and let* K *be a regular language over* Γ. *Then* $\mathrm{Inf}(c \cap (\Sigma^* \times K))$ *is regular.*

Proof. Consider a finite automaton A with state set Q that accepts K. For $p, q \in Q$, denote by $L(A, p, q)$ the set of strings x for which there is a path $p \xrightarrow{x} q$ in A. We define an automaton B over alphabet $\Sigma \times \{F, I\}$ as follows. The sets of states, initial states, and final states of B and A coincide. For $p, q \in Q$ and $a \in \Sigma$, B contains the transition

- $p \xrightarrow{(a,I)} q$ iff the language $c(a) \cap L(A, p, q)$ is infinite,
- $p \xrightarrow{(a,F)} q$ iff the language $c(a) \cap L(A, p, q)$ is finite and nonempty.

We claim that $x = a_1 \dots a_n \in \mathrm{Inf}(c \cap (\Sigma^* \times K))$ for $a_i \in \Sigma$ and $n > 0$ if and only if there is an accepting path in B that is labelled by $(a_1, b_1) \dots (a_n, b_n)$ where at least one b_i equals I. [Note that $c(\epsilon) = \{\epsilon\}$, thus $\epsilon \notin \mathrm{Inf}(c)$.] This can be seen as follows.

By definition, $x \in \mathrm{Inf}(c \cap (\Sigma^* \times K))$ if and only if $c(x) \cap K$ is infinite. Each string $y \in c(x)$ is of the form $y = y_1 \dots y_n$ with $y_i \in c(a_i)$. Thus, $y \in c(x) \cap K$ if and only if there is a path $q_0 \xrightarrow{y_1} q_1 \xrightarrow{y_2} \dots \xrightarrow{y_n} q_n$ in A accepting y, i.e., with q_0 initial and q_n final. Denote by P the set of all such sequences $q_0 \dots q_n$. We have $y_i \in c(a_i) \cap L(A, q_{i-1}, q_i)$ by construction, and

$$c(x) \cap K = \bigcup_{q_0 \dots q_n \in P} \big(c(a_1) \cap L(A, q_0, q_1)\big) \cdot \dots \cdot \big(c(a_n) \cap L(A, q_{n-1}, q_n)\big).$$

A finite union of languages is infinite if at least one summand is infinite, and a finite product of languages is infinite if no factor is empty and at least one factor is infinite. Thus, $c(x) \cap K$ is infinite if and only if there is a sequence $q_0 \dots q_n \in P$ such that each $c(a_i) \cap L(A, q_{i-1}, q_i)$ is nonempty and at least one $c(a_i) \cap L(A, q_{i-1}, q_i)$ is infinite. This is equivalent to the existence of a sequence $b_1 \dots b_n \in \{F, I\}^n \setminus F^n$ such that $(a_1, b_1) \dots (a_n, b_n) \in L(B)$. Therefore,

$$\mathrm{Inf}(c \cap (\Sigma^* \times K)) = \pi(L(B) \setminus (\Sigma \times \{F\})^*)$$

where $\pi : (\Sigma \times \{I, F\})^* \to \Sigma^*$ is the morphism induced by $\pi : (a, b) \mapsto a$. □

Recall that a finite transduction [1, 21] is a relation on strings induced by a finite transducer, which can be seen as a finite automaton with output. Finite

transductions effectively preserve regularity of languages, and the inverse of a finite transduction is again a finite transduction.

LEMMA 5. *Let Σ, Γ, Δ be alphabets, let $c \subseteq \Sigma^* \times \Gamma^*$ be a substitution, and let $T \subseteq \Gamma^* \times \Delta^*$ be a finitely branching finite transduction such that also T^- is finitely branching. Then* $\mathrm{Inf}(c \circ T)$ *is regular.*

Proof. We have $\mathrm{Inf}(c \circ T) \subseteq \mathrm{Inf}(c \cap (\Sigma^* \times T^-(\Delta^*)))$ because T is finitely branching, and $\mathrm{Inf}(c \circ T) \supseteq \mathrm{Inf}(c \cap (\Sigma^* \times T^-(\Delta^*)))$ because T^- is finitely branching. Thus, $\mathrm{Inf}(c \circ T) = \mathrm{Inf}(c \cap (\Sigma^* \times T^-(\Delta^*)))$, and we conclude by Lemma 4. □

REMARK 3. The regularity results in Lemma 4 and Lemma 5 are effective if c is an $\mathcal{L}$-substitution for a family $\mathcal{L}$ of languages that is closed under intersection with regular sets and for which emptiness and finiteness are decidable. This is the case, for example, for the family of context-free languages.

LEMMA 6. *For each inverse deleting string rewriting system R, the set $\mathrm{Inf}(R^*)$ is effectively regular.*

Proof. Let R be a system over alphabet Σ such that R^- is deleting. If $\epsilon \in \mathrm{lhs}(R)$ then $\mathrm{Inf}(R^*) = \Sigma^*$, so we may assume $\epsilon \notin \mathrm{lhs}(R)$. Then $R^* = c \circ s^-$ by Corollary 3, where c is a context-free substitution and s is a finite epsilon-free substitution. The claim follows by Lemma 5 and Remark 3. □

The sets $\mathrm{Im}(R)$ and $\mathrm{Inf}(R^*)$ are connected by the following equation.

LEMMA 7. *For each string rewriting system R,*

$$\mathrm{Im}(R) = \mathrm{Inf}(R^*) \cup \{y \in \Sigma^* \mid \exists x \in \Sigma^* : y \to_R^* x \to_R^+ x\}.$$

Proof. Each $x \in \mathrm{Im}(R)$ either initiates a derivation with infinitely many pairwise distinct strings or initiates a derivation that ends in a cycle. Conversely, a derivation that ends in a cycle is obviously infinite, and a string that has infinitely many descendants initiates an infinite derivation because $\to_R$ is finitely branching. □

If R is inverse deleting, then R^- is deleting, hence terminating; hence there is no $x \in \Sigma^*$ such that $x \to_R^+ x$.

LEMMA 8. *For each inverse deleting string rewriting system R,*

$$\mathrm{Im}(R) = \mathrm{Inf}(R^*).$$

LEMMA 9. *Let the string rewriting system R be inverse match-bounded by c, and let $S = match_c(R^-)^-$. Then for every infinite derivation $x_0 \to_R x_1 \to_R \cdots$ there is an infinite derivation $x'_0 \to_S x'_1 \to_S \cdots$ such that* $\text{base}(x'_i) = x_i$ *for all* $i \geq 0$. *Therefore,*

$$\text{Im}(R) = \text{base}(\text{Im}(S)).$$

Proof. For every finite initial segment $x_0 \to_R x_1 \to_R \ldots \to_R x_j$ of the given derivation, we construct (by the remarks before Definition 2) a derivation $x_{0j} \to_S x_{1j} \to_S \ldots \to_S x_{jj} = \text{lift}_0(x_j)$ such that $\text{base}(x_{ij}) = x_i$. By induction on $j - i$, using the implication

$$x_{ij'} \geq x_{ij} \leftarrow_S x_{i-1,j} \quad \Rightarrow \quad x_{ij'} \leftarrow_S x_{i-1,j'} \geq x_{i-1,j}$$

for the inductive step, we have $x_{ij} \leq x_{ij'}$ for all $0 \leq i \leq j \leq j'$. Define x'_i as the maximum of the finite set $\{x_{ij} \mid j \geq i\} \subseteq \text{base}^{-1}(x_i) \cap \text{height}^{-1}(\{0, \ldots, c\}^*)$. Then for all $i \geq 0$ we have $x'_i \to_S x'_{i+1}$ since there is an index $j > i$ such that $x'_i = x_{ij} \to_S x_{i+1,j} = x'_{i+1}$. □

THEOREM 8. *If a string rewriting system R is inverse match-bounded by c,* then $\text{Im}(R)$ *is effectively regular.*

Proof. By Lemma 3, the system $S = \text{match}_c(R^-)^-$ is inverse deleting. Therefore, $\text{Im}(S)$ is regular by Lemmas 6 and 8; thus also $\text{Im}(R)$ is regular as $\text{Im}(R) = \text{base}(\text{Im}(S))$ by Lemma 9. □

COROLLARY 7. *Let $\mathcal{L}$ be a family of languages that is closed under intersection with regular sets, and for which emptiness is decidable. Then the following problem is decidable:*

GIVEN: *A language $L \in \mathcal{L}$; an inverse match-bounded string rewriting system R.*
QUESTION: *Is R terminating on L?*

Proof. By Theorem 8, $\text{Im}(R)$ is regular, so emptiness of $\text{Im}(R) \cap L$ is decidable. □

COROLLARY 8. *Termination and uniform termination are decidable for the class of inverse match-bounded string rewriting systems.*

Proof. Choose $L = \{x\}$ to decide whether there is an infinite derivation starting from string x, and choose $L = \Sigma^*$ to decide uniform termination. □

EXAMPLE 8. The one-rule systems $\{babaa \to aaababab\}$, $\{aabaaba \to abaabaaab\}$, $\{aaabbab \to abbaaabba\}$, and $\{caabca \to aabccaabc\}$ are inverse match-bounded by 2, 3, 3, and 2, respectively. Each of these systems R satisfies $\text{Im}(R) = \emptyset$; hence R terminates.

EXAMPLE 9. McNaughton [15] shows that termination and uniform termination are decidable for the class of inhibitor systems; see Example 2. We can give an alternative proof of this result by Corollary 8.

As the membership problem for a fixed regular language is decidable in linear time, we obtain the following.

COROLLARY 9. *For each inverse match-bounded string rewriting system R, its termination problem is decidable in linear time:*

GIVEN: *A string x.*
QUESTION: *Is R terminating on x?*

EXAMPLE 10. Recall the inhibitor systems from Example 2. Corollary 9 affirmatively answers McNaughton's [15] Open Problem 2: For every inhibitor system, is its termination problem decidable in polynomial time?

We conclude this section with an example known as *Zantema's Problem*. Proving termination of this one-rule string rewriting system is a challenging problem [3, 5, 14, 18, 19, 23] where all previous automated methods for termination proofs fail.

EXAMPLE 11. Termination of the system

$$Z = \{a^2b^2 \to b^3a^3\}$$

can be proven by verifying [9] that Z is match-bounded by 4. Alternatively, we can check that Z is inverse match-bounded by 2 and satisfies $\mathrm{Im}(Z) = \emptyset$, so it terminates. The automaton for $\mathrm{match}(Z^-)^*(\mathrm{lift}_0(\Sigma^*))$ has 199 states. The intermediate constructions according to Theorem 1 involve much larger automata (up to 1576 states with 15,999 transitions) over much larger alphabets (up to 283 letters). The approximate construction [10] for match-boundedness by 4, in contrast, needs only 34 states.

7. More Examples

In contrast to match-boundedness, inverse match-boundedness does not entail termination. Here we collect a few examples of this kind. In each case, the language $\mathrm{Im}(R)$ has been computed according to the algorithm described in the last section. This is carried out in detail for the first example.

EXAMPLE 12. The inhibitor system

$$R = \{ab \to bb\iota aa\}$$

over alphabet $\Sigma = \{a, b, \iota\}$ is inverse match-bounded by 1 (cf. Example 2). The system is nonterminating; for instance, it admits the loop $abb \to bb\iota aab \to bb\iota abb\iota aa$. Below we will show that the language $\mathrm{Im}(R)$ consists of all strings that contain one of the factors *aab*, *abab*, or *abb*:

$$\mathrm{Im}(R) = \Sigma^*\{aab, abab, abb\}\Sigma^*.$$

Step 1. In order to automatically verify that R is inverse match-bounded by 1, we compute $\mathrm{match}_2(R^-)^*(\mathrm{lift}_0(\Sigma^*))$. For this purpose, it suffices to consider the *accessible* rules from $\mathrm{match}_2(R^-)$, that is, those rules that can occur in derivations starting from strings in $\mathrm{lift}_0(\Sigma^*)$; see [9, Section 5]. For this example, we obtain the nine-rule system

$$S^- = \{b_p b_q \iota_0 a_r a_s \to a_1 b_1 \mid pq \in L, rs \in R\}$$

over the alphabet $\overline{\Sigma} = \{a_0, a_1, b_0, b_1, \iota_0\}$, where

$$L = \{00, 10, 11\} \text{ and } R = \{00, 01, 11\}.$$

Height 2 never occurs in S^-; hence R is inverse match-bounded by 1. So S^- is deleting by Lemma 3, and $\mathrm{Im}(S) = \mathrm{Inf}(S^*)$ by Lemma 8.

Step 2. Next we determine a decomposition of S^* according to Corollary 3. Specifically, we seek an extended alphabet $\Gamma \supseteq \overline{\Sigma}$, an epsilon-free finite substitution $s \subseteq \overline{\Sigma}^* \times \Gamma^*$, and a context-free substitution $c \subseteq \overline{\Sigma}^* \times \Gamma^*$ such that

$$S^* = c \circ s^-.$$

This decomposition is the result of the nondeterministic transformation procedure described in [13]. Depending on the chosen strategy we get the following result:

$$\Gamma = \overline{\Sigma} \cup \{c_{l,r}, d_{l,r} \mid l \in L, r \in R\},$$

that is, we add 18 fresh auxiliary letters. The finite substitution s is

$$\begin{aligned} s : x &\mapsto \{x\} \text{ for } x \in \{a_0, a_1, b_0, b_1\}, \\ s : \iota_0 &\mapsto \{\iota_0\} \cup \{c_{l,r} d_{l,r} \mid l \in L, r \in R\}. \end{aligned}$$

The context-free substitution c is given as

$$c(u) = C^*(u)$$

for $u \in \overline{\Sigma}^*$, where C is the string rewriting system (with 18 rules)

$$\begin{aligned} C = \{&a_1 \to b_p b_q c_{pq,rs} \mid pq \in L, rs \in R\} \cup \\ &\{b_1 \to d_{pq,rs} a_r a_s \mid pq \in L, rs \in R\}. \end{aligned}$$

In particular, $c(x) = \{x\}$ for $x \in \{a_0, b_0, \iota_0\}$, whereas $c(a_1)$ and $c(b_1)$ are infinite context-free languages.

Step 3. Instantiating s^- for T, and $\overline{\Sigma}$ for both Σ and Δ in the proof of Lemma 5 we get

$$\mathrm{Inf}(S^*) = \mathrm{Inf}(c \circ s^-) = \mathrm{Inf}(c \cap (\overline{\Sigma}^* \times s(\overline{\Sigma}^*))).$$

Next we apply the construction from the proof of Lemma 4 for $K = s(\overline{\Sigma}^*)$.

First, we need an automaton A accepting K. As the set of states we choose $\{Z\} \cup (L \times R)$ (that is, 10 states) with Z being initial and final. Further, A contains the following 23 transitions, where $x \in \overline{\Sigma}$ and $l \in L$, $r \in R$:

$$Z \xrightarrow{x} Z, \qquad Z \xrightarrow{c_{l,r}} (l,r), \qquad (l,r) \xrightarrow{d_{l,r}} Z.$$

From A we construct an automaton B as follows. Take the same set of states, and keep initial and final ones. In order to determine the transitions of B, we have to consider all sets of the form $c(x) \cap L(A,p,q)$ with p and q being states in A and $x \in \overline{\Sigma}$. From these 500 sets, the following 131 are nonempty ($x \in \overline{\Sigma}$ and $l, \bar{l} \in L$, $r, \bar{r} \in R$):

$$\begin{array}{ll}
c(x) \cap L(A,Z,Z), & \\
c(a_1) \cap L(A,Z,(l,r)), & c(b_1) \cap L(A,(l,r),Z), \\
c(a_1) \cap L(A,(l,r),(10,\bar{r})), & c(b_1) \cap L(A,(\bar{l},01),(l,r)), \\
c(a_1) \cap L(A,(l,r),(11,\bar{r})), & c(b_1) \cap L(A,(\bar{l},11),(l,r)),
\end{array}$$

Among these, there are 18 infinite sets ($l \in L$, $r \in R$):

$$c(a_1) \cap L((l,11),(11,r)), \qquad c(b_1) \cap L((l,11),(11,r)).$$

Therefore, automaton B has 18 transitions with labels of the form (x,I) and 113 transitions with labels (x,F). Finally, we compute the language $\mathrm{Inf}(S^*) = \pi(L(B) \setminus (\overline{\Sigma} \times \{F\})^*)$ by collecting all accepting paths in B that contain a transition with label (x,I). In our case, we obtain

$$\mathrm{Inf}(S^*) = \pi\Big(\bigcup_{l \in L, r \in R} L(B,Z,(l,11)) \cdot \{(a_1,I),(b_1,I)\} \cdot L(B,(11,r),Z) \Big).$$

Since $\pi(\bigcup_{l \in L} L(B,Z,(l,11))) = \overline{\Sigma}^* a_1 b_1^*$ and $\pi(\bigcup_{r \in R} L(B,(11,r),Z)) = a_1^* b_1 \overline{\Sigma}^*$, we get

$$\mathrm{Inf}(S^*) = \overline{\Sigma}^* a_1 b_1^* \{a_1, b_1\} a_1^* b_1 \overline{\Sigma}^* = \overline{\Sigma}^* a_1 \{a_1, b_1 a_1, b_1\} b_1 \overline{\Sigma}^*.$$

Eventually, Lemma 9 yields the announced result:

$$\mathrm{Im}(R) = \mathrm{base}(\mathrm{Im}(S)) = \mathrm{base}(\mathrm{Inf}(S^*)) = \Sigma^* \{aab, abab, abb\} \Sigma^*.$$

EXAMPLE 13. The inhibitor system $R = \{baa \to aa\iota babba\}$ from Example 2 has a loop of length 3 initiated by $baaa$. Indeed,

$$baaa \in \mathrm{Im}(R) = \Sigma^*\{ba, baab\}b^*aa\Sigma^*.$$

EXAMPLE 14. The system $R = \{baaba \to aababbaab\} = \{\ell \to r\}$ is inverse match-bounded by 2 and satisfies $\mathrm{Im}(R) = \Sigma^*M\Sigma^*$ for

$$M = \{\ell aa, \ell\ell, b\ell a\} \cup \{\ell a, \ell b, \ell baab, bbaab\}b^*\ell.$$

For instance, ℓaa initiates a loop of length 4.

EXAMPLE 15. The system $R = \{aabaaba \to abaaabaab\} = \{\ell \to r\}$ is inverse match-bounded by 3; we have $\mathrm{Im}(R) = \Sigma^*\{\ell aa, a\ell a, ara\}\Sigma^*$. The string ℓaa initiates a loop of length 3.

EXAMPLE 16. System $R = \{bca(bc)^3 \to a(bc)^3c(bc)^2b\} = \{\ell \to r\}$ from [7] is inverse match-bounded by 2 and admits a loop of length 3 starting from $bc\ell c$. We have $\mathrm{Im}(R) = \Sigma^*M\Sigma^*$, where

$$M = \{bc\ell c, \ell ca(bc)^3, (bc)^2\ell, \ell c\ell, \ell cbc\ell\}.$$

EXAMPLE 17. Consider the system $R = \{ab \to da, ac \to acc\}$ over $\Sigma = \{a, b, c, d\}$ from Example 6, which is inverse match-bounded by 1. Here we obtain $\mathrm{Im}(R) = \Sigma^*ab^*c\Sigma^*$.

REMARK 4. Looping one-rule string rewriting systems exist that are not inverse match-bounded. As an example, consider the system $R = \{ba \to aabb\}$, which admits the loop $bba \to_R baabb \to_R aabbabb$. It is not inverse match-bounded [9, Example 11]. Actually, R does not preserve context-free languages, as we can show that $R^*(baa)$ is not context-free. This, too, proves that R is not inverse match-bounded by Corollary 5.

8. Derivational Complexity

Inverse match-bounded systems R have linear size growth for strings that are not in $\mathrm{Im}(R) = \mathrm{Inf}(R^*)$. Hence, inverse match-bounded systems have linear derivational complexity on the sets of terminating strings.

THEOREM 9. *For each inverse match-bounded string rewriting system R there is a constant $k \in \mathbb{N}$ such that $|y| \leq k \cdot |x|$ for all $x \in \Sigma^* \setminus \mathrm{Inf}(R^*)$ and $y \in R^*(x)$.*

Proof. Since the morphisms base and lift preserve lengths of strings, we may assume that R is inverse deleting by Lemma 2 and Lemma 3. As in the proof of Lemma 6, we can exclude the case $\epsilon \in \mathrm{lhs}(R)$. Then $R^* = c \circ s^-$ by Corollary 3,

where $c \subseteq \Sigma^* \times \Gamma^*$ is a context-free substitution and $s \subseteq \Sigma^* \times \Gamma^*$ is a finite epsilon-free substitution. Further, we have $\mathrm{Inf}(R^*) = \mathrm{Inf}(c \circ s^-) = \mathrm{Inf}(c \cap (\Sigma^* \times K))$ as in the proof of Lemma 5 for $K = s(\Sigma^*)$.

We change the automaton B constructed in the proof of Lemma 4 to an automaton B' over the alphabet $\Sigma \times (\mathbb{N} \cup \{I\})$ as follows. In the case where $c(a) \cap L(A, p, q)$ is finite and nonempty, we draw an arrow labelled by (a, m) where m is the length of the longest string in $c(a) \cap L(A, p, q)$. Now define $\mu : \Sigma^* \setminus \mathrm{Inf}(R^*) \to \mathbb{N}$ by

$$\mu(a_1 \ldots a_n) = \max\{\sum_{i=1}^{n} m_i \mid (a_1, m_1) \ldots (a_n, m_n) \in L(B') \cap (\Sigma \times \mathbb{N})^*\}.$$

One verifies for all $x \in \Sigma^* \setminus \mathrm{Inf}(R^*)$ that $\mu(x)$ is the maximal length of strings in the set $R^*(x)$. One possible choice of k is therefore obtained as the maximum of all second components of labels in B'. □

COROLLARY 10. *Every inverse match-bounded string rewriting system has linear derivational complexity on its set of terminating strings.*

Proof. If R^- is a match-bounded system, then by Proposition 2, there is a constant k' such that $x \to_R^n y$ (i.e., $y \to_{R^-}^n x$) implies $n \leq k' \cdot |y|$ for strings x and y. On the other hand, $|y| \leq k \cdot |x|$ for some constant k by Theorem 9. Hence, $n \leq k' \cdot k \cdot |x|$. □

REMARK 5. By Corollary 10, systems like $\{ab \to ba\}$ and $\{ab \to bba\}$, which have quadratic and exponential complexity respectively, cannot be proven terminating by inverse match-bounds. This limits the applicability of inverse match-bounds as a general termination proof method.

REMARK 6. Bounds for k and k' can be determined effectively. In the worst case, the constant k' can be exponential in the size of the underlying alphabet $\Sigma \times \{0, \ldots, c\}$ if R^- is match-bounded by c [13]. For an algorithm to compute constant k see the proof of Theorem 9 above.

All terminating, inverse match-bounded examples in Section 6 have linear derivational complexity. In particular, this is true for Example 11 (*Zantema's Problem*), a result due to Tahhan Bittar [19].

9. Implementing Match-Bounds: Matchbox

Our program *Matchbox* [20] can verify that a given string rewriting system R is match-bounded by a given number for a given regular language. Hence it can verify inverse match-boundedness (for Σ^*) as well, and it provides an

implementation of Theorem 5 for computing the set of normalizing strings. Further, *Matchbox* constructs $\mathrm{Im}(R)$ according to Theorem 8.

The program can be accessed via a CGI-interface at

`http://dfa.imn.htwk-leipzig.de/matchbox/`,

its Haskell source is available. In particular, *Matchbox* is able to prove termination of a large number of string rewriting systems for which all standard automated methods, for example, path orderings, polynomial interpretations, and dependency pairs [4, 22], fail.

10. Conclusion

We studied properties related to termination that can be solved automatically for inverse match-bounded string rewriting systems, that is, for systems the inverse of which is match-bounded. Inverse match-boundedness is a natural extension of criteria from the literature that are known to preserve context-free languages or to have a decidable uniform termination problem.

By a decomposition into elementary language theoretic relations, the inverse rewrite relation preserves regularity, and the rewrite relation preserves context-freeness. Hence, the set of normalizing strings, and by a more involved argument, the set of immortal strings are effectively regular. It follows that normalization, uniform normalization, termination and uniform termination, are decidable properties for inverse match-bounded systems. Using these results, we solve two open problems of McNaughton's about inhibitor systems.

The derivational complexity of an inverse match-bounded system is linearly bounded on the set of terminating strings. On the positive side, this means that showing an inverse match-bound is a method for proving linear derivational complexity, which may be more efficient than showing a match-bound for the same purpose. Moreover, it is the first criterion for linear derivational complexity on the set of terminating strings that applies to non-terminating systems. On the negative side, it means that the applicability of the above mentioned strong decision procedures is limited. However, many hard termination problems, for example, Zantema's Problem, turn out to be linearly terminating. Inverse match bounds are one method to find such proofs automatically – and to our knowledge, it is the first such method that can decide termination and normalization for arbitrary regular sets of words.

Match-boundedness and inverse match-boundedness, as proof methods for uniform termination, seem to be correlated: all our terminating, inverse match-bounded, examples are also match-bounded. We would like to learn more about the precise nature of this surprising observation.

Can this approach be extended to term rewriting? For such an extension, one needs the notion of a deleting term rewriting system such that a decomposition theorem holds. We have not found a satisfying definition yet.

Acknowledgements

This research was initiated while the last two authors were visiting scientists at the Institute for Computer Applications in Science and Engineering (ICASE), NASA Langley Research Center (LaRC), Hampton, VA, in September 2002. The first author was supported by the National Aeronautics and Space Administration under NASA Contract No. NAS1-97046.

We thank Hans Zantema for valuable hints and fruitful discussions, and the anonymous referees for helpful suggestions.

References

1. Berstel, J.: *Transductions and Context-Free Languages*, Teubner, Stuttgart, 1979.
2. Book, R. V. and Otto, F.: *String-Rewriting Systems*, Texts Monogr. Comput. Sci. Springer-Verlag, New York, 1993.
3. Coquand, T. and Persson, H.: A proof-theoretical investigation of Zantema's problem, in M. Nielsen and W. Thomas (eds.), *Proc. 11th Annual Conf. of the EACSL CSL-97*, *Lecture Notes in Comput. Sci*., Vol. 1414, Springer-Verlag, 1998, pp. 177–188.
4. Dershowitz, N.: Termination of rewriting, *J. Symb. Comput.* **3**(1–2) (1987), 69–115.
5. Dershowitz, N. and Hoot, C.: Topics in termination, in C. Kirchner (ed.), *Proc. 5th Int. Conf. Rewriting Techniques and Applications RTA-93*, *Lecture Notes in Comput. Sci*., Vol. 690, Springer-Verlag, 1993, pp. 198–212.
6. Geser, A.: Note on normalizing, non-terminating one-rule string rewriting systems, *Theoret. Comput. Sci.* **243** (2000), 489–498.
7. Geser, A.: Termination of string rewriting rules that have one pair of overlaps, in R. Nieuwenhuis (ed.), *Proc. 14th Int. Conf. Rewriting Techniques and Applications RTA-03*, *Lecture Notes in Comput. Sci*, Vol. 2706, Springer-Verlag, pp. 410–423.
8. Geser, A., Hofbauer, D. and Waldmann, J.: Match-bounded string rewriting systems, in B. Rovan and P. Vojtas (eds.), *Proc. 28th Int. Symp. Mathematical Foundations of Computer Science MFCS-03*, *Lecture Notes in Comput. Sci*., Vol. 2747, Springer-Verlag, 2003, pp. 449–459.
9. Geser, A., Hofbauer, D. and Waldmann, J.: Match-bounded string rewriting systems, *Appl. Algebra Eng. Commun. Comput.* **15**(3–4) (2004), 149–171.
10. Geser, A., Hofbauer, D., Waldmann, J. and Zantema, H.: Finding finite automata that certify termination of string rewriting systems, *Int. J. Found. Comput. Sci.* **16**(3) (2005), 471–486.
11. Ginsburg, S. and Greibach, S. A.: Mappings which preserve context sensitive languages, *Inf. Control* **9**(6) (1966), 563–582.
12. Hibbard, T. N.: Context-limited grammars, *J. ACM* **21**(3) (1974), 446–453.
13. Hofbauer, D. and Waldmann, J.: Deleting string rewriting systems preserve regularity, *Theoret. Comput. Sci.* **327** (2004), 301–317.
14. Kurth, W.: Termination und Konfluenz von Semi-Thue-Systemen mit nur einer Regel. Dissertation, Technische Universität Clausthal, Germany, 1990.
15. McNaughton, R.: Semi-Thue systems with an inhibitor, *J. Autom. Reason.* **26** (2001), 409–431.
16. Moore, C. and Eppstein, D.: One-dimensional peg solitaire, and duotaire, in R. J. Nowakowski (ed.), *More Games of No Chance*, *MSRI Publications* 42, Cambridge Univ. Press, 2002, pp. 341–350.
17. Ravikumar, B.: Peg-solitaire, string rewriting systems and finite automata, in H.-W. Leong, H. Imai and S. Jain (eds.), *Proc. 8th Int. Symp. Algorithms and Computation ISAAC-97*, *Lecture Notes in Comput. Sci*., Vol. 1350, Springer-Verlag, 1997, pp. 233–242.

18. Sénizergues, G.: On the termination problem for one-rule semi-Thue systems, in H. Ganzinger (ed.), *Proc. 7th Int. Conf. Rewriting Techniques and Applications RTA-96*, *Lecture Notes in Comput. Sci*., Vol. 1103, Springer-Verlag, 1996, pp. 302–316.
19. Tahhan Bittar, E.: Complexité linéaire du problème de Zantema, *C. R. Acad. Sci. Paris Sér. I Inform. Théor.* **323** (1996), 1201–1206.
20. Waldmann, J.: Matchbox: A tool for match-bounded string rewriting, in V. van Oostrom (ed.), *Proc. 15th Int. Conf. Rewriting Techniques and Applications RTA-04*, *Lecture Notes in Comp. Sci*. Vol. 3091, Springer-Verlag, 2004, pp. 85–94.
21. Yu, S.: Regular languages, in G. Rozenberg and A. Salomaa (eds.), *Handbook of Formal Languages*, Vol. 1, Springer-Verlag, 1998, pp. 41–110.
22. Zantema, H.: Termination, in *Terese*, *Term Rewriting Systems*, Cambridge Univ. Press, 2003, pp. 181–259.
23. Zantema, H. and Geser, A.: A complete characterization of termination of $0^p1^q \to 1^r0^s$, *Appl. Algebra Eng. Commun. Comput*. **11**(1) (2000), 1–25.