AAECC 15, 149–171 (2004)
Digital Object Identifier (DOI) 10.1007/s00200-004-0162-8

AAECC
Applicable Algebra in Engineering, Communication and Computing

Match-Bounded String Rewriting Systems

Alfons Geser[1,⋆], Dieter Hofbauer[2], Johannes Waldmann[3]

[1]National Institute of Aerospace, 144 Research Drive, Hampton, VA 23666, USA (e-mail: geser@nianet.org)
[2]Mühlengasse 16, 34125 Kassel, Germany (e-mail: dieter@theory.informatik.uni-kassel.de)
[3]Hochschule für Technik, Wirtschaft und Kultur (FH) Leipzig, Fb IMN, PF 30 11 66, 04251 Leipzig, Germany (e-mail: waldmann@imn.htwk-leipzig.de)

Received: October 13, 2003; revised version: June 7, 2004
Published online: October 14, 2004 – © Springer-Verlag 2004

Abstract. We introduce a new class of automated proof methods for the termination of rewriting systems on strings. The basis of all these methods is to show that rewriting preserves regular languages. To this end, letters are annotated with natural numbers, called *match heights*. If the minimal height of all positions in a redex is h then every position in the reduct will get height $h + 1$. In a *match-bounded* system, match heights are globally bounded. Using recent results on *deleting* systems, we prove that rewriting by a match-bounded system preserves regular languages. Hence it is decidable whether a given rewriting system has a given match bound. We also provide a criterion for the absence of a match-bound. It is still open whether match-boundedness is decidable. Match-boundedness for all strings can be used as an automated criterion for termination, for match-bounded systems are terminating. This criterion can be strengthened by requiring match-boundedness only for a restricted set of strings, namely the set of right hand sides of forward closures.

Keywords: String Rewriting System, Semi-Thue System, Termination, Match-bounded

1 Introduction

Rewriting is a model of computation. It allows one to handle questions like *termination* (there is no infinite computation), *correctness* (no erroneous configuration is reachable), and *normalization* (a final configuration is reachable).

⋆ Partly supported by the National Aeronautics and Space Administration under NASA Contract No. NAS1-97046 while this author was in residence at the NIA.

These questions can be stated in terms of sets of descendants: For a rewriting system R and a language L let $R^*(L) = \{y \mid x \in L, x \to_R^* y\}$. Now R is *correct* for L iff $R^*(L) \cap \text{Err} = \emptyset$, and R is *normalizing* for L iff $L \subseteq R^{-*}(\text{Final})$, with Err and Final denoting the set of erroneous and final configurations, respectively, and R^- denoting R where left and right hand sides of rules are exchanged. Starting from classical program analysis, recent applications include verification of XML transformations [3] and cryptographic protocols [10].

In view of these applications, the reachability relation R^* should effectively preserve language classes with good decidability and closure properties—like the class of regular languages.

Some of us recently showed [18] that *deleting* string rewriting systems preserve regular languages. A string rewriting system R is called *deleting* if there exists a partial ordering on its alphabet such that each letter in the right hand side of a rule is less than some letter in the corresponding left hand side. Deleting systems can be understood as the inverses of *context limited* grammars as defined and investigated by Hibbard [17]. Deleting rewriting systems terminate and have linearly bounded derivational complexity.

In the present paper, we transfer these results to *match-bounded* string rewriting. Letters are annotated with numbers, which we call *match heights*. A position in a reduct will get height $h+1$ if the minimal height of all positions in the redex is h. A rewriting system is *match-bounded* if match heights of derivations are globally bounded. In this case its annotated system is finite and deleting. Termination and regularity preservation carry over from the annotated to the original system. The recognizing automaton for the set of descendants modulo the annotated system is a certificate for match-boundedness.

The concept of match heights is a generalization of *change heights* introduced by Ravikumar [26] to investigate preservation of regularity under rewriting modulo length-preserving systems.

Every match-bounded system terminates, and effectively preserves regularity of languages. Therefore it is decidable whether a given system has a given match-bound. This makes match-boundedness a new automatic criterion for termination. The criterion applies for instance to Zantema's System $\{a^2b^2 \to b^3a^3\}$ (match-bound 4) for which hitherto all automated termination proof methods failed.

We also study *RFC-match-boundedness*, a variant of the criterion, where a system has to be match-bounded only for the set of right hand sides of its forward closures. By a result of Dershowitz, termination there is sufficient for uniform termination.

Basic definitions, results and examples are given in Sections 3 and 4, while in Sections 5 and 6 we discuss how to verify or refute match-boundedness. In Section 7 we introduce RFC-match-boundedness, and consider some variants of this notion in Section 8. All main criteria are implemented (Section 9). Section 10 contains a short comparison of our new termination criteria with

Zantema's Termination Hierarchy. We conclude by discussing ramifications for further research in Section 11.

Some of the results reported here have been presented at the 28th International Symposium on Mathematical Foundations of Computer Science MFCS 2003 at Bratislava, Slovak Republic [12] and at the 6th International Workshop on Termination WST 2003 at Valencia, Spain [13].

2 Preliminaries

We mostly stick to standard notations for strings and string rewriting, as e.g. in [2]. We use ϵ for the *empty string*, and $|x|$ is the *length* of a string x. By $\mathcal{L}(A)$ we denote the language accepted by a finite automaton A, and REG denotes the class of regular languages. Further, $\text{factor}(L) = \{y \in \Sigma^* \mid \exists x, z \in \Sigma^* : xyz \in L\}$ for a language $L \subseteq \Sigma^*$.

A *string rewriting system* over a (finite or infinite) 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* $\ell \to r$. By $\text{lhs}(R)$ and $\text{rhs}(R)$ we denote the sets of left (resp. right) hand sides of R. The reflexive and transitive closure of $\to_R$ is $\to_R^*$, often abbreviated as R^*, and $\to_R^+$ or R^+ denote the transitive closure. An *R-derivation* is a (finite or infinite) sequence $(x_0, x_1, \dots)$ with $x_i \to_R x_{i+1}$ for all i. We call *R terminating on $L \subseteq \Sigma^*$* if there is no infinite derivation starting with some $x_0 \in L$. If $L = \Sigma^*$, we call *R terminating*. In order to classify lengths of derivations, define the *derivation height* function modulo R on Σ^* by $\text{dh}_R(x) = \max\{n \in \mathbb{N} \mid \exists y \in \Sigma^* : x \to_R^n y\}$; here, $\to_R^n$ denotes the n-fold composition of $\to_R$. The *derivational complexity* of R is defined as the function $n \mapsto \max\{\text{dh}_R(x) \mid |x| \le n\}$ on $\mathbb{N}$.

A rewriting rule $\ell \to r$ is called *context-free* if $|\ell| \le 1$, and a rewriting system is context-free if all its rules are.

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 $\rho^- = \{(b, a) \mid (a, b) \in \rho\} \subseteq B \times A$, and we say that ρ satisfies the property *inverse P* if ρ^- satisfies P. Thus, the set of *descendants* of a language $L \subseteq \Sigma^*$ modulo some 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 a substitution s is uniquely determined by the languages $s(a)$ for $a \in \Sigma$. If each language $s(a)$ for $a \in \Sigma$ is finite, then s is a *finite* substitution.

Now we recall definitions and results regarding *deleting* string rewriting systems [18], a topic that goes back to Hibbard [17]. 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 ([18]) *Every deleting string rewriting system is terminating, and has linear derivational complexity.*

This is due to the fact that a $>$-deleting system can be ordered by the multiset extension of the precedence $>$. The linear factor may be exponential in the size of the alphabet.

Furthermore, we have the following decomposition result.

Theorem 1 ([18]) *Let R be a deleting string rewriting system over Σ. Then there effectively are 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*

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

As a consequence, inverse deleting systems effectively preserve context-freeness, a result by Hibbard [17]. As another consequence we get:

Corollary 1 ([18]) *Deleting string rewriting systems effectively preserve regularity.*

Throughout the paper, all existential statements are effective in that there are algorithms that construct witnesses. This presumes that all input parameters are effectively given. In particular, a class of rewrite systems is said to *effectively preserve regularity* if there is an algorithm that, given a finite system R from this class and a finite automaton A, yields a finite automaton accepting the language $R^*(\mathcal{L}(A))$.

3 Match-Bounded String Rewriting Systems

We will now apply the theory of deleting systems to obtain results for *match-bounded* rewriting. A derivation is match-bounded if dependencies between rule applications are limited. To make this precise, we will 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.

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 such that $\epsilon \notin \mathrm{lhs}(R)$, we 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 non-empty R, the system $\mathrm{match}(R)$ is always infinite. Note that systems with $\epsilon \in \mathrm{lhs}(R)$ are trivially non-terminating.

Every derivation modulo $\mathrm{match}(R)$ corresponds to a derivation modulo R of the same length (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 (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,

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

Definition 1 *A string rewriting system R over Σ is called* match-bounded for $L \subseteq \Sigma^*$ by $c \in \mathbb{N}$ *if* $\epsilon \notin \mathrm{lhs}(R)$ *and* $\mathrm{match}(R)^*(\mathrm{lift}_0(L)) \subseteq (\Sigma \times \{0, \ldots, c\})^*$. *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, 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 $\mathrm{match}(R)$ may be replaced by a finite restriction. Denote by $\mathrm{match}_c(R)$ the restriction of $\mathrm{match}(R)$ to the alphabet $\Sigma \times \{0, \ldots, c\}$.

By definition of match-boundedness we immediately get the following result.

Lemma 1 *If R is match-bounded for L by c, then*

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

Lemma 2 *For all R with $\epsilon \notin \mathrm{lhs}(R)$ and all $c \in \mathbb{N}$, the system* $\mathrm{match}_c(R)$ *is deleting.*

Proof Use the precedence $>$ on $\Sigma \times \{0, \ldots, c\}$ where $(a, m) > (b, n)$ iff $m < n$. (Letters of minimal match height are maximal in the precedence.) $\square$

Theorem 2 *If R is match-bounded for L, then R is terminating on L.*

Proof An infinite R-derivation starting from an element of L can be transformed into an infinite match(R)-derivation from an element of $\text{lift}_0(L)$. The latter, given that R is match-bounded for L by c, is a $\text{match}_c(R)$-derivation. However, $\text{match}_c(R)$ is deleting by Lemma 2 and hence terminating by Proposition 1. $\square$

Likewise, Proposition 1 implies linearly bounded derivation lengths for match-bounded systems.

Proposition 2 *Every match-bounded string rewriting system has linear derivational complexity.*

We conclude this section with a few examples.

Example 1 The system $R_1 = \{ab \to bc\}$ is match-bounded by 1, system $R_2 = \{aa \to aba\}$ is match-bounded by 2, system $R_3 = \{ab \to ac, ca \to bc\}$ is match-bounded by 2, and system $R_4 = \{ab \to ac, ca \to b\}$ is match-bounded by 3. It is easily checked that the claimed heights are reached: For R_1 consider the derivation $\underline{a_0b_0} \to_{\text{match}(R_1)} b_1c_1$, for R_2 we have $\underline{a_0a_0}a_0a_0 \to_{\text{match}(R_2)} a_1b_1a_1\underline{a_0a_0} \to_{\text{match}(R_2)} a_1b_1\underline{a_1a_1}b_1a_1 \to_{\text{match}(R_2)} a_1b_1a_2b_2a_2b_1a_1$, for R_3 the derivation $\underline{a_0b_0}a_0 \to_{\text{match}(R_3)} a_1\underline{c_1a_0} \to_{\text{match}(R_3)} \underline{a_1b_1}c_1 \to_{\text{match}(R_3)} a_2c_2c_1$ shows that height 2 is reached, and for R_4 we have $\underline{a_0b_0}a_0 \to_{\text{match}(R_4)} a_1\underline{c_1a_0} \to_{\text{match}(R_4)} \underline{a_1b_1} \to_{\text{match}(R_4)} a_2c_2$, thus $(a_0b_0a_0)^2 \to^*_{\text{match}(R_4)} a_2\underline{c_2a_2}c_2 \to_{\text{match}(R_4)} a_2b_3c_2$. Here, rexedes are underlined. In Section 5, we explain how to check automatically that these are indeed upper bounds for match-heights.

The next example illustrates that any number can be a least match bound.

Example 2 The bubble sort system $B_2 = \{ab \to ba\}$ over the two-letter alphabet $\{a, b\}$ is match-bounded for a^*b^n by n, but not by $n - 1$. The system $\{a_i \to a_{i+1} \mid 0 \le i < n\}$ over alphabet $\Sigma = \{a_i \mid 0 \le i \le n\}$ is match-bounded (for Σ^*) by n, but not by $n - 1$. As a variant of the previous example, now over a fixed alphabet, consider the system $\{ab^ic \to ab^{i+1}c \mid 0 \le i < n\}$ over $\{a, b, c\}$; it is match-bounded by n, but not by $n - 1$. The same holds true for the length-preserving variant $\{ab^ic^{n-i+1} \to ab^{i+1}c^{n-i} \mid 0 \le i < n\}$.

Example 3 System B_2 is not match-bounded (for $\{a, b\}^*$) since it has quadratic derivational complexity, contradicting the conclusion of Proposition 2.

Dually to Lemma 2, we have:

Proposition 3 *If R is deleting, then R is match-bounded.*

Proof Assume R over Σ is deleting for the precedence $>$ on Σ. Then R is match-bounded by the maximal height (i.e., length of a descending chain) in $(\Sigma, >)$. □

Example 4 The system $\{ba \to cb, bd \to d, cd \to de\}$ is match-bounded by 2, since it is deleting for the precedence $a > b > d, a > c > e, c > d$.

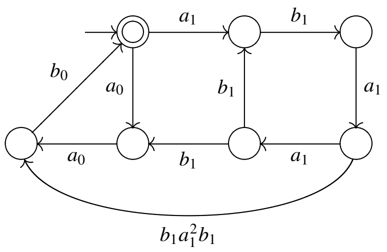
4 Match-Bounded Systems Preserve Regularity

Here, we elaborate on the fact that match-bounded string rewriting systems always preserve regularity. The section concludes on a short comparison of match-boundedness to the related concept of change-boundedness [26].

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

Proof By Lemma 1, $R^*(L) = \text{base}(\text{match}_c(R)^*(\text{lift}_0(L)))$. As $\text{match}_c(R)$ is deleting by Lemma 2, thus effectively regularity preserving by Corollary 1, and since the class of regular languages is effectively closed under morphisms, we are done. □

Example 5 For $R = \{aaba \to abaab\}$ (cf. [20], p. 118) and $L = (aab)^*$, the language $\text{match}(R)^*(\text{lift}_0(L))$ is accepted by the following automaton. We use generalized automata where transitions are labelled by words instead of single letters.



By stripping heights from all letters, one obtains an automaton accepting $R^*(L)$.

Example 6 The bubble sort system $B_2 = \{ab \to ba\}$ is not regularity preserving, since $B_2^*((ab)^*) \cap b^*a^* = \{b^n a^n \mid n \geq 0\}$ is not regular. So Theorem 3 implies that B_2 is not match-bounded. (Cf. Example 3 for another indirect proof, and Example 10 for a direct proof of the same fact.)

However, not every regularity preserving string rewriting system is match-bounded. For instance, the system $\{aa \to a\}$ constitutes a counterexample. As a *monadic* system (i.e., $|\ell| > |r| \leq 1$ for $(\ell \to r) \in R$) it preserves regularity [1, 2], but it is not match-bounded as proven in Example 12 below.

Remark 1 There are terminating and regularity preserving systems with high derivational complexity as we are going to demonstrate. For an alphabet Σ, define the string rewriting system $\text{Embed}(\Sigma) = \{a \to \epsilon \mid a \in \Sigma\}$. By an application of Higman's Lemma, the *subword language* $\text{Embed}(\Sigma)^*(L)$ is regular for *each* language L over Σ, cf. Theorem 7.3 in [4]. This implies that for any string rewriting system R over Σ, the system $R \cup \text{Embed}(\Sigma)$ preserves (in fact, *generates*) regularity.

Termination of $R \cup \text{Embed}(\Sigma)$ is called *simple termination* of R. By the above, every simply terminating rewriting system R can be extended to a (simply) terminating and regularity preserving system while keeping or increasing its derivational complexity. E.g., $\{ab \to ba, a \to \epsilon, b \to \epsilon\}$ preserves regularity, and has quadratic complexity.

Example 7 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 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", see [25] for its history. The system P^- is match-bounded by 2, so we obtain yet another proof of that result.

Remark 2 Ravikumar [26] proves that P^- preserves regularity by considering the system's *change-bound* (which is 4). Change-boundedness is similar to match-boundedness. Given a length-preserving string rewriting system R (viz. $|\ell| = |r|$ for every rule $\ell \to r$), define the system

$$\text{change}(R) = \{\ell \to r \mid (\text{base}(\ell) \to \text{base}(r)) \in R,\ \text{height}(\text{succ}(\ell)) = \text{height}(r)\}$$

over alphabet $\Sigma \times \mathbb{N}$, where succ is the morphism succ : $(\Sigma \times \mathbb{N})^* \to (\Sigma \times \mathbb{N})^*$ induced by succ : $(a, h) \mapsto (a, h+1)$. For instance, the system change($\{ab \to bc\}$) contains the rules

$$a_0b_0 \to b_1c_1,\ a_0b_1 \to b_1c_2,\ a_1b_0 \to b_2c_1,\ a_1b_1 \to b_2c_2,\ a_0b_2 \to b_1c_3,\ \ldots$$

Ravikumar proves that if change$(R)^*(\text{lift}_0(L))$ has bounded height, then R preserves regularity of L. In contrast to change-bounds, match-bounds are also applicable to non-length-preserving systems. For a length-preserving system R, match(R) will always give smaller or equal heights, so our result directly implies Ravikumar's. In fact, it can also be shown conversely that match-boundedness implies change-boundedness for length-preserving systems.

5 Verification of Match-Bounds

In this section, we show that match-boundedness by a given bound is decidable. Hence the existence of a match-bound is recursively enumerable. In the next section we address the complementary problem.

Any given automaton over alphabet $\Sigma \times \mathbb{N}$ can be seen as a potential certificate of the fact that a system R is match-bounded for a language L by some bound c, and hence of termination of R on L. An automaton A is said to *certify that R is match-bounded for L by c* if its accepted language $\mathcal{L}(A)$

1. includes $\text{lift}_0(L)$,
2. is closed under rewriting modulo $\text{match}_{c+1}(R)$, and
3. contains no letter of height $c + 1$.

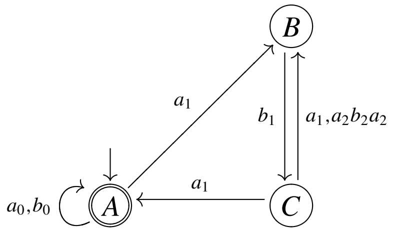
The first two items imply that $\text{match}_{c+1}(R)^*(\text{lift}_0(L))$ is included in the accepted language. Validity of such a certificate can be decided by standard algorithms for finite automata.

Theorem 4 *If a finite automaton A certifies that R is match-bounded for L by c, then R is match-bounded for L by c.*

Proof We have $\text{match}_{c+1}(R)^*(\text{lift}_0(L)) \subseteq \mathcal{L}(A) \subseteq (\Sigma \times \{0, \ldots, c\})^*$ and hence $\text{match}(R)^*(\text{lift}_0(L)) \subseteq (\Sigma \times \{0, \ldots, c\})^*$. □

Corollary 2 *If a finite automaton A certifies that R is match-bounded for L, then R terminates with linear derivational complexity on L.*

Example 8 For $R = \{aa \to aba\}$ and $L = \{a, b\}^*$, the set $\text{match}_3(R)^*(\text{lift}_0(L))$ is accepted by the following (non-deterministic) automaton. (Again, we use an obvious generalized notation where transitions are labelled by sets of words.)



redex path	reduct path
$A \xrightarrow{a_0} A \xrightarrow{a_0} A$	$A \xrightarrow{a_1} B \xrightarrow{b_1} C \xrightarrow{a_1} A$
$A \xrightarrow{a_0} A \xrightarrow{a_1} B$	$A \xrightarrow{a_1} B \xrightarrow{b_1} C \xrightarrow{a_1} B$
$C \xrightarrow{a_1} A \xrightarrow{a_0} A$	$C \xrightarrow{a_1} B \xrightarrow{b_1} C \xrightarrow{a_1} A$
$C \xrightarrow{a_1} A \xrightarrow{a_1} B$	$C \xrightarrow{a_2b_2a_2} B$

Closure under $\text{match}_3(R)$ can be verified by checking off the table on the right. Since the highest label is 2, the automaton certifies that R is match-bounded by 2, as claimed in the introductory Example 1.

An obvious procedure for the construction of a certifying finite automaton starts from a given automaton that accepts $\text{lift}_0(L)$, and adds a fresh path $p \xrightarrow{r} q$ whenever a path $p \xrightarrow{\ell} q$ and a rule $\ell \to r$ in $\text{match}_{c+1}(R)$ exist. This procedure may add states to the automaton forever. One may envisage various heuristics for the re-use of states to make the procedure terminate, see [14]. However, every heuristics runs the risk of over-estimating the language and may so fail in certifying.

We offer a completely different method that always yields a certifying finite automaton; moreover the automaton is *exact*, i.e., $\mathcal{L}(A)$ coincides with $\text{match}_{c+1}(R)^*(\text{lift}_0(L))$. Thus we can also determine effectively the *least* match-bound of a match-bounded system.

Theorem 5 *The following problem is decidable:*

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

Proof In a given automaton that accepts L, annotate every letter by 0, yielding an automaton that accepts $\text{lift}_0(L)$. By Lemma 2 and Corollary 1, an automaton accepting $L_{c+1} = \text{match}_{c+1}(R)^*(\text{lift}_0(L))$ can be constructed. Then R is match-bounded for L by c if and only if $L_{c+1} \subseteq (\Sigma \times \{0, \ldots, c\})^*$. □

For an implementation, the growth of $|\text{match}_c(R)|$ as a function in c is problematic. Hence we want to replace $\text{match}_c(R)$ by some small system S such that $\text{match}_c(R)^*|_{\text{lift}_0(L)} = S^*|_{\text{lift}_0(L)}$. The basic idea is to start from the empty set, and to add only rules that are necessary.

When computing $\text{match}_c(R)^*(\text{lift}_0(L))$, we may restrict attention to those rules of $\text{match}_c(R)$ that are *accessible* in derivations starting from $\text{lift}_0(L)$. For a language $L \subseteq \Sigma^*$, a system R over Σ, and a system $S \subseteq \text{match}(R)$ define

$$\text{accessible}(L, R, S) = \text{match}(R) \cap (\text{factor}(S^*(\text{lift}_0(L))) \times (\Sigma \times \mathbb{N})^*).$$

Note that this construction is effective if a finite system S and a regular language L are effectively given. We construct a sequence of rewriting systems

R_i by $R_0 = \emptyset$ and $R_{i+1} = \text{accessible}(L, R, R_i)$. Induction on i shows $R_i \subseteq \text{match}_i(R)$ for $i \geq 0$. In particular, every system R_i is finite. By induction on i, using that $S \subseteq S'$ implies $\text{accessible}(L, R, S) \subseteq \text{accessible}(L, R, S')$, one also proves that $R_i \subseteq R_{i+1}$. Define $R_\infty = \bigcup_{i \in \mathbb{N}} R_i$. Clearly, $R_\infty^*(\text{lift}_0(L)) = \text{match}(R)^*(\text{lift}_0(L))$. If R is match-bounded for L by c, then R_∞ is a subset of $\text{match}_c(R)$; so R_∞ is finite, and there is an index N such that $R_N = R_{N+1} = \cdots$. If R is not match-bounded for L, then R_∞ contains for each c a rule with height c, and therefore is infinite. We remark that the enumeration of R_i up to $i = |\,\text{match}_c(R)| + 1$ can be used as an alternative decision procedure for Theorem 5.

Example 9 Proving termination of the one-rule system $Z = \{a^2b^2 \to b^3a^3\}$ is known as *Zantema's Problem*. This is a "modern classic" in rewriting [5, 8, 20, 28, 29, 33], as it provides a test case where all previous automated methods for termination proofs fail. Our algorithm constructs in 6 iterations a deterministic automaton with 85 states. This automaton recognizes $\text{match}(Z)^*(\text{lift}_0(\Sigma^*))$ and certifies that Z is match-bounded for Σ^* by 4. This also proves that Z has only linear derivational complexity, a result by Tahhan-Bittar [29].

6 A Proof Method for Unbounded Match Heights

In this section we provide a sufficient condition for the absence of a match bound. We leave decidability of match-boundedness as an open problem.

Sometimes we can also verify automatically that a given rewriting system R ($\epsilon \notin \text{lhs}(R)$) is *not* match-bounded for a language L. For this purpose, we want a non-empty witnessing language $W \subseteq L$ such that every element in W can be reached from some element in W by an all-height increasing derivation. By chaining such derivations, strings of arbitrary height can be derived, disproving match-boundedness. In the remainder of this section we formalize this argument.

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 $\mathbb{N}^n$. We assume $W \subseteq \Sigma^+$. A string $y \in W$ is *reached* from $x \in W$ if there is a derivation $\text{lift}_0(x) \to^*_{\text{match}(R)} py'q$ for some string $y' \geq \text{lift}_1(y)$ and strings p, q. Now every element in W is reached from an element in W if $W \subseteq \text{raised}(R, W)$, where the latter set of strings is defined by

$$\text{raised}(R, W) = \text{base}(\text{factor}(\text{match}(R)^*(\text{lift}_0(W))) \cap (\Sigma \times (\mathbb{N} \setminus \{0\}))^+).$$

First we observe that a $\text{match}(R)$-derivation can always be raised to greater heights since the two relations $\geq$ and $\to_{\text{match}(R)}$ commute:

Lemma 3 $\geq \circ \to_{\text{match}(R)} \subseteq \to_{\text{match}(R)} \circ \geq$.

Proposition 4 *Let R be a string rewriting system such that $\epsilon \notin \operatorname{lhs}(R)$, and let W be a non-empty language, both over Σ. If $W \subseteq \operatorname{raised}(R, W)$, then R is not match-bounded for W.*

Proof We prove a stronger claim: If $W \subseteq \operatorname{raised}(R, W)$ then, for every $c \geq 0$,

$$W \subseteq \operatorname{base}(\operatorname{factor}(\operatorname{match}(R)^*(\operatorname{lift}_0(W))) \cap (\Sigma \times \{c, c+1, \dots\})^+).$$

In other words, every element of W can receive unbounded match heights. We prove this claim by induction on c. Consider $y \in W$. For $c = 0$ we obtain $y \in \operatorname{base}(\operatorname{factor}(\operatorname{lift}_0(y)) \cap (\Sigma \times \mathbb{N})^+)$. So assume $c > 0$. By inductive hypothesis there is a string $u \in W$ and a derivation

$$\operatorname{lift}_0(u) \rightarrow^*_{\operatorname{match}(R)} p y' q$$

with $y' \geq \operatorname{lift}_{c-1}(y)$. Since $u \neq \epsilon$, this derivation can be relabelled to a derivation

$$\operatorname{lift}_1(u) \rightarrow^*_{\operatorname{match}(R)} \operatorname{succ}(p) \operatorname{succ}(y') \operatorname{succ}(q)$$

with $\operatorname{succ}(y') \geq \operatorname{succ}(\operatorname{lift}_{c-1}(y)) = \operatorname{lift}_c(y)$, where succ is the morphism defined in Section 4, increasing the height of each position by 1. Since $u \in W \subseteq \operatorname{raised}(R, W)$, there is $v \in W$ and a derivation

$$\operatorname{lift}_0(v) \rightarrow^*_{\operatorname{match}(R)} p' u' q' \tag{1}$$

with $u' \geq \operatorname{lift}_1(u)$. By Lemma 3 we get a derivation

$$u' \rightarrow^*_{\operatorname{match}(R)} p'' y'' q'' \tag{2}$$

for some $y'' \geq \operatorname{succ}(y')$. We conclude by composing (1) and (2) into a derivation

$$\operatorname{lift}_0(v) \rightarrow^*_{\operatorname{match}(R)} p' u' q' \rightarrow^*_{\operatorname{match}(R)} p' p'' y'' q'' q'$$

with $y'' \geq \operatorname{lift}_c(y)$. □

A slightly weaker version of Proposition 4 is obtained as follows: Define

$$\operatorname{raised}_c(R, W) = \operatorname{base}(\operatorname{factor}(\operatorname{match}_c(R)^*(\operatorname{lift}_0(W))) \cap (\Sigma \times (\mathbb{N} \setminus \{0\}))^+),$$

and replace $\operatorname{raised}(R, W)$ in Proposition 4 by $\operatorname{raised}_c(R, W)$:

Corollary 3 *Let R be a string rewriting system such that $\epsilon \notin \operatorname{lhs}(R)$, and let W be a non-empty language, both over Σ. If $W \subseteq \operatorname{raised}_c(R, W)$ for some $c \in \mathbb{N}$, then R is not match-bounded for W.*

This version can be effectively checked if a finite system R, a number $c \in \mathbb{N}$, and a regular language W are effectively given.

Example 10 The system $B_2 = \{ab \to ba\}$ (cf. Example 6) is not match-bounded for Σ^*. Take $W = (ab)^+$. Then $\text{raised}_1(B_2, W) = \text{factor}((ba)^+) \supseteq W$.

Example 11 Neither is $R = \{aabb \to ba\}$ match-bounded, as witnessed by $W = \{a, b\}^+ = \text{raised}_1(R, W)$. This can be seen as follows. Define the two morphisms $\phi : a \mapsto a, b \mapsto abb$ and $\psi : a \mapsto aab, b \mapsto b$. Then, for each $y \in \Sigma^*$, there are derivations

$$a\ \phi(y) \to_R^* y\ a \quad \text{and} \quad \psi(y)\ b \to_R^* b\ y,$$

and these can be combined to a derivation

$$a\ \phi(\psi(y))\ abb = a\ \phi(\psi(y)\ b) \to_R^* \psi(y)\ b\ a \to_R^* b\ y\ a.$$

When lifting this to a $\text{match}(R)$-derivation, starting from heights 0, all final heights are 1. This proves that for each $y \in W = \Sigma^+$, there is $x = a\phi(\psi(y))abb \in \Sigma^+ = W$ with the required property. In contrast, the system $\{a^3b^3 \to b^2a^2\}$ is match-bounded by 2.

Example 12 The regularity preserving system $R = \{aa \to a\}$ is not match-bounded: check that $W = \{a^{2^n} \mid n \in \mathbb{N}\} \subseteq \text{raised}_1(R, W)$. Alternatively, $W' = a^+ \subseteq \text{raised}_1(R, W')$.

Example 13 The system $R = \{ab \to bba\}$ is not match-bounded for Σ^* because it admits derivations $a^n b \to_R^{2^n - 1} b^{2^n} a^n$ of exponential lengths. Another proof can be given by Proposition 4. One shows by induction that, for $k \geq 0$,

1. $a_k b_k^i \to_{\text{match}(R)}^* b_{k+1}^{2i} a_{k+1}$ for $i > 0$, and
2. $a_0 b_1 b_1 b_2^2 \ldots b_k^{2^{k-1}} \to_{\text{match}(R)}^* b_1 b_1 b_2^2 \ldots b_{k+1}^{2^k} a_{k+1}$.

So $a_0^m b_0$, $m > 1$, rewrites to a string that contains the factor $a_{m-1} \ldots a_1 b_1$:

$$a_0^m b_0 \to_{\text{match}(R)} a_0^{m-1} b_1^2 a_1 \to_{\text{match}(R)}^* b_1 b_1 b_2^2 \ldots b_{m-1}^{2^{m-2}} a_{m-1} \ldots a_1 b_1 a_1.$$

Hence $W \subseteq \text{raised}(R, W)$ for $W = a^+ b$. Note that this set of witnesses is regular, but the derivations verifying the witnesses are not globally match-bounded. On the other hand, we can have match-bounded verification for the non-regular set of witnesses $W' = \{ab^{2^n} ab^{2^{n-1}} \ldots ab \mid n \in \mathbb{N}\}$, since $W' \subseteq \text{raised}_1(R, W')$.

Looping string rewriting systems form a particular subclass of the class of all non-terminating systems. A *loop* is a derivation of the form $s \to_R^+ psq$ for strings $s, p, q \in \Sigma^*$. As it turns out, the existence of a loop can be characterized in terms of *finite* sets of witnesses, as follows.

Proposition 5 *A string rewriting system R admits a loop if and only if there is a finite, non-empty set W such that $W \subseteq \text{raised}(R, W)$.*

Proof If R admits a loop then it also admits a loop $s \to_R^+ psq$ during which every position between letters is touched [15]. So $\text{lift}_0(s) \to_{\text{match}(R)}^+ p's'q'$ for some $s' \geq \text{lift}_1(s)$. The claim holds with $W = \{s\}$. Conversely, let $W \subseteq \text{raised}(R, W)$. Then for every $k > 0$ there is a sequence $w_0, w_1, \ldots, w_k$ such that $w_{i+1} \in \text{raised}(R, \{w_i\})$ for $0 \leq i < k$, thus $w_j \in \text{raised}(R, \{w_i\})$ for $0 \leq i < j \leq k$. For $k = |W|$, by the pigeonhole principle, there are $i < j$ such that $w_i = w_j$. Hence $w_i = w_j \in \text{factor}(R^+(\{w_i\}))$ forms the desired loop. □

The converse of Proposition 4 is open:

Problem 1 *Does every string rewriting system R such that $\epsilon \notin \text{lhs}(R)$ that is not match-bounded have a non-empty set $W \subseteq \text{raised}(R, W)$?*

If the stronger statement " ... have some $c \in \mathbb{N}$ and a non-empty *regular* set $W \subseteq \text{raised}_c(R, W)$" holds then match-boundedness is decidable: One can simultaneously enumerate these certificates (c, W) along with certificates for match-boundedness (according to Theorem 5). Example 13 seems to indicate that the stronger statement is false. So the following remains open:

Problem 2 *Is match-boundedness decidable?*

7 Match-Bounds for Forward Closures

We have shown that match-boundedness for L is a criterion for termination on L. To prove termination on Σ^* however, the obvious choice $L = \Sigma^*$ may be too restrictive as it even entails linear derivational complexity. We are going to show that the set of right hand sides of forward closures [21, 7] is a better choice for L. For a string rewriting system R over Σ, the set of *forward closures* $\text{FC}(R) \subseteq \Sigma^* \times \Sigma^*$ is defined as the least set containing R such that

- if $(u, v) \in \text{FC}(R)$ and $v \to_R w$, then $(u, w) \in \text{FC}(R)$ (*inside reduction*), and
- if $(u, v\ell_1) \in \text{FC}(R)$ and $(\ell_1\ell_2 \to r) \in R$ for strings $\ell_1 \neq \epsilon$, $\ell_2 \neq \epsilon$, then $(u\ell_2, vr) \in \text{FC}(R)$ (*right extension*).

Let $\text{RFC}(R)$ denote the set of right hand sides of forward closures. Equivalently, $\text{RFC}(R)$ is the least subset of Σ^* containing $\text{rhs}(R)$ such that

- if $v \in \text{RFC}(R)$ and $v \to_R w$, then $w \in \text{RFC}(R)$, and
- if $v\ell_1 \in \text{RFC}(R)$ and $(\ell_1\ell_2 \to r) \in R$ for $\ell_1 \neq \epsilon$, $\ell_2 \neq \epsilon$, then $vr \in \text{RFC}(R)$.

Theorem 6 ([6]) *A string rewriting system R is terminating on Σ^* if and only if R is terminating on* $\mathrm{RFC}(R)$.

Theorem 2 for $L = \mathrm{RFC}(R)$ yields:

Corollary 4 *Every string rewriting system R that is match-bounded for* $\mathrm{RFC}(R)$ *is terminating.*

Example 14 The system $R = \{aa \to aba\}$ (cf. Example 8) is match-bounded for $\mathrm{RFC}(R)$ by 0 since the set $\mathrm{RFC}(R) = (ab)^+a$ consists of strings in normal form. Therefore, R is terminating.

We can obtain $\mathrm{RFC}(R)$ as a set of descendants modulo the rewriting system

$$R_\# = R \cup \{\ell_1\# \to r \mid (\ell_1\ell_2 \to r) \in R, \ell_1 \neq \epsilon, \ell_2 \neq \epsilon\}$$

over alphabet $\Sigma \cup \{\#\}$, where right extension is simulated via the new end-marker $\# \notin \Sigma$. Indeed:

Lemma 4 *Let R be a string rewriting system over Σ, where $\# \notin \Sigma$. Then*

$$\mathrm{RFC}(R) = R_\#^*(\mathrm{rhs}(R) \cdot \#^*) \cap \Sigma^*.$$

Proof This follows from $\mathrm{RFC}(R) \cdot \#^* = R_\#^*(\mathrm{rhs}(R) \cdot \#^*)$. The inclusion from left to right is shown by induction over the definition of $\mathrm{RFC}(R)$. Conversely, $\to_{R_\#}^n(\mathrm{rhs}(R) \cdot \#^*) \subseteq \mathrm{RFC}(R) \cdot \#^*$ is shown by induction over n. □

Theorem 7 *A string rewriting system R is terminating if and only if $R_\#$ is terminating on* $\mathrm{rhs}(R) \cdot \#^*$.

Proof By virtue of Theorem 6 it is sufficient to show that R is terminating on $\mathrm{RFC}(R)$ iff $R_\#$ is terminating on $\mathrm{rhs}(R) \cdot \#^*$. If $R_\#$ is terminating on $\mathrm{rhs}(R) \cdot \#^*$, then it is terminating also on $R_\#^*(\mathrm{rhs}(R) \cdot \#^*) \supseteq \mathrm{RFC}(R)$ by Lemma 4. So particularly $R \subseteq R_\#$ is terminating on $\mathrm{RFC}(R)$. Conversely, every infinite $R_\#$-derivation contains only finitely many steps from $R_\# \setminus R$, as each such step consumes a # symbol. Hence there exists a string in $R_\#^*(\mathrm{rhs}(R) \cdot \#^*) = \mathrm{RFC}(R) \cdot \#^*$ which initiates an infinite R-derivation. The # symbols can be stripped off, yielding an infinite R-derivation from $\mathrm{RFC}(R)$. □

Definition 2 *A string rewriting system R is* RFC-match-bounded by c *if $R_\#$ is match-bounded for* $\mathrm{rhs}(R) \cdot \#^*$ *by c.*

As an immediate consequence, Theorem 3 together with Lemma 4 yield:

Corollary 5 *If a string rewriting system R is RFC-match-bounded by c, then the language* $\mathrm{RFC}(R)$ *is effectively regular.*

Recall that $R_\#$ is match-bounded for $\mathrm{rhs}(R) \cdot \#^*$ if and only if $R_\#$ is match-bounded for $R_\#^*(\mathrm{rhs}(R) \cdot \#^*)$.

Lemma 5 *If a string rewriting system R is RFC-match-bounded, then R is match-bounded for* $\mathrm{RFC}(R)$.

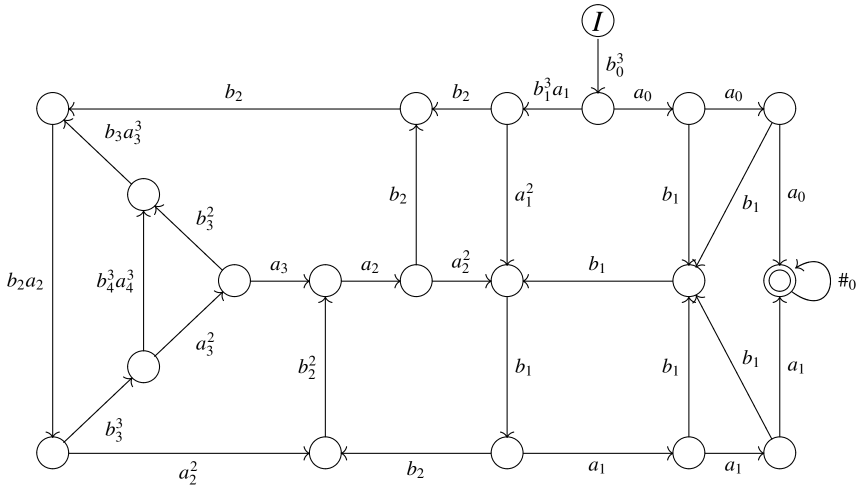
Proof If $R_\#$ is match-bounded for $R_\#^*(\mathrm{rhs}(R) \cdot \#^*)$, then R is match-bounded for $\mathrm{RFC}(R)$ as $R \subseteq R_\#$ and, by Lemma 4, $\mathrm{RFC}(R) \subseteq R_\#^*(\mathrm{rhs}(R) \cdot \#^*)$. □

However, RFC-match-boundedness and match-boundedness for $\mathrm{RFC}(R)$ are not equivalent, see Example 20 for a counterexample.

Combining Lemma 5 with Corollary 4 we obtain the following termination criterion.

Theorem 8 *Every RFC-match-bounded string rewriting system is terminating.*

Example 15 Zantema's system $Z = \{a^2b^2 \to b^3a^3\}$ from Example 9 is RFC-match-bounded by 4, as the following finite automaton with initial state I accepts the language $\mathrm{match}_5(Z_\#)^*(\mathrm{lift}_0(\mathrm{rhs}(Z) \cdot \#^*))$.



This automaton certifies termination of Z, cf. the discussion at the beginning of Section 5.

Example 16 The system $B_2 = \{ab \to ba\}$ from Example 6 is RFC-match-bounded by 1 since $\text{match}(B_{2\#})^*(\text{lift}_0(\text{rhs}(B_2) \cdot \#^*)) = b_0(a_0 \cup b_1^+ a_1)\#_0^*$. It is not match-bounded, see Example 10.

Example 17 The system $R = \{ab \to bba\}$ is RFC-match-bounded by 1. Here, as can easily be seen, $\text{match}(R_\#)^*(\text{lift}_0(\text{rhs}(R) \cdot \#^*)) = b_0^2(a_0 \cup (b_1^2)^+ a_1)\#_0^*$. Again, this system is not match-bounded, see Example 13.

Examples 16 and 17 show that RFC-match-bounded systems, unlike match-bounded systems, may have non-linear derivational complexities. We do not know of an RFC-match-bounded system with longer than exponential derivations.

Example 18 The bubble-sort system over a three-letter alphabet, $B_3 = \{ab \to ba, ac \to ca, bc \to cb\}$, is not match-bounded for $\text{RFC}(B_3)$, and hence not RFC-match-bounded. To prove it, check $b^+c^+a \subseteq \text{RFC}(B_3)$, and observe that $\{bc \to cb\} \subseteq B_3$ is not match-bounded for b^+c^+, cf. Example 6. In contrast, all proper subsystems of B_3 are RFC-match-bounded by 1.

Example 19 For $R = \{ab \to baa\}$, we have $\text{RFC}(R) \cap b^*a^* = \{b^n a^{2^n} | n \geq 1\} \notin \text{REG}$. By Corollary 5, R is not RFC-match-bounded, in contrast to Example 17. This shows that the class of RFC-match-bounded systems is not closed under reversal, i.e., under the operation $R \mapsto \{\text{rev}(\ell) \to \text{rev}(r) \mid (\ell \to r) \in R\}$, where $\text{rev}(a_1 a_2 \ldots a_n) = a_n \ldots a_2 a_1$ for $a_i \in \Sigma$. (Note that the class of terminating systems is trivially closed under reversal.)

8 RFC-Match-boundedness and Related Conditions

As a sufficient condition for termination of a string rewriting system R, we introduced match-boundedness of R for $\text{RFC}(R)$. In order to construct $\text{RFC}(R)$, we used the enriched system $R_\#$. This system contains additional rules that subtly influence match heights, as indicated in this section.

Example 20 Here, we will present an example demonstrating that the inverse of Lemma 5 does not hold true. We claim that the string rewriting system R over alphabet $\{a, b, c, d, e\}$ with rules

$$\{a \to b,\ b \to cd,\ de \to a,\ cb \to a\}$$

is match-bounded for $\text{RFC}(R)$, but not RFC-match-bounded.

Claim 1 R is match-bounded for $\text{RFC}(R)$. Indeed, it is straightforward to verify that R is match-bounded by 3 for $\text{RFC}(R) = c^*a \cup c^*b \cup c^+d$.

Claim 2 R is not RFC-match-bounded. This is a direct consequence of the fact that, for $z \in \{0, 1\}$ and for any $n \geq 1$,

$$a_z \#_0^{2^n-1} \to^*_{\mathrm{match}(R_\#)} a_{2n+1}.$$

The proof is by induction on n. We have $a_z\#_0 \to b_{z+1}\#_0 \to c_{z+2}d_{z+2}\#_0 \to c_{z+2}a_1 \to c_{z+2}b_2 \to a_3$ for $n = 1$, and for $n > 1$ we obtain

$$a_z\#_0^{2^n-1} \to^* a_{2n-1}\#_0^{2^{n-1}} \to b_{2n}\#_0^{2^{n-1}} \to c_{2n+1}d_{2n+1}\#_0^{2^{n-1}} \to$$
$$c_{2n+1}a_1\#_0^{2^{n-1}-1} \to^* c_{2n+1}a_{2n-1} \to c_{2n+1}b_{2n} \to a_{2n+1},$$

the induction hypothesis being applied twice. Throughout, rewriting is modulo $\mathrm{match}(R_\#)$.

Example 21 Even if a string rewriting system R is both match-bounded for $\mathrm{RFC}(R)$ and RFC-match-bounded, the corresponding least match-bounds may differ by any given number $k > 0$. This is shown for the system

$$R = \{a_{i-1} \to a_i, b_{i-1} \to b_i \mid 1 \leq i < k\} \cup \{a_{k-1} \to cd, de \to b_0, cb_{k-1} \to a_0\}$$

over alphabet $\{a_0, \ldots, a_{k-1}, b_0, \ldots, b_{k-1}, c, d, e\}$. As is easily seen, R is match-bounded for $\mathrm{RFC}(R)$ by $k + 1$, whereas $R_\#$ is match-bounded for $\mathrm{rhs}(R) \cdot \#^*$ by $2k + 1$. So the difference between these bounds is indeed k.

For completeness' sake we also mention a sufficient criterion for RFC-match-boundedness. We will use the set of left hand sides of forward closures of a rewriting system R, denoted by $\mathrm{LFC}(R)$.

We remark that computation of $\mathrm{LFC}(R)$ seems to require the construction of the full set $\mathrm{FC}(R)$, a step that could be avoided for $\mathrm{RFC}(R)$.

Proposition 6 *If a string rewriting system R is match-bounded for* $\mathrm{LFC}(R)$ *by c, then R is RFC-match-bounded by c.*

Proof For any step modulo $R_\#$ that uses a rule $\ell_1\# \to r$, it is possible to reconstruct some string ℓ_2 with $\ell_1\ell_2 \to r$ in R that # represents. This transformation preserves match heights. □

Example 22 The least RFC-match-bound of $R = \{aa \to aba\}$ from Example 8 is 1. The least match-bound of R for $\mathrm{LFC}(R) = aa^+$, however, is 2.

Example 23 The system $R = \{aba \to a, ab \to ba\}$ is RFC-match-bounded by 1, but R is not match-bounded for $a(ba)^+ \subseteq \mathrm{LFC}(R)$.

Corollary 6 *Every match-bounded string rewriting system is RFC-match-bounded by at most the same bound.*

Recall that the converse of Corollary 6 does not hold, as Example 16 demonstrates.

9 Implementing Match-Bounds: Matchbox

We have implemented the algorithms presented in this paper (Theorems 5 and 8) in a program called `Matchbox`. It can be accessed via a CGI-interface at

http://theo1.informatik.uni-leipzig.de/matchbox/

and its Haskell source is available. The program fared quite well in the recent "termination competition" held at the 6th International Workshop on Termination (WST 2003) at Valencia, Spain. Unlike its competitors, however, `Matchbox` only addresses string rewriting.

In particular, `Matchbox` is able to prove termination for a large number of one-rule string rewrite systems for which all standard automated methods (like path orderings and polynomial interpretations) fail, and for which only complicated ad-hoc proofs were known, if any. The list below contains those one-rule systems that are left from an attempt to classify termination of all (approx. $6.7 \cdot 10^9$) one-rule systems $\{\ell \to r\}$ where $|\ell| < |r| \leq 9$. They could not be solved by any known method [11].

$$\begin{array}{rr}
\{abaab \to baabbaa\}, & \{babbaa \to abbaabba\}, \\
\{aabaaab \to baaabbaaa\}, & \{ababaab \to baabbabaa\}, \\
\{baabba \to aabbaaabb\}, & \{caabca \to aabccaabc\}, \\
\{aabaaba \to abaabaaab\}, & \{abaab \to baabbaaba\}.
\end{array}$$

`Matchbox` yields proofs that all these systems are RFC-match-bounded by 2.

10 A Comparison to the Termination Hierarchy

We have shown that for string rewriting systems R the following implications are valid:

$$\begin{array}{r}
\text{match-bounded} \Rightarrow \\
\text{match-bounded for } \mathrm{LFC}(R) \Rightarrow \\
\text{RFC-match-bounded} \Rightarrow \\
\text{match-bounded for } \mathrm{RFC}(R) \Rightarrow \\
\text{terminating}
\end{array}$$

None of these implications can be reversed: The system $\{ab \to ba\}$ from Example 10 is match-bounded for $\mathrm{LFC}(R) = ab^+$, but not match-bounded. Example 23 is RFC-match-bounded but not match-bounded for $\mathrm{LFC}(R)$. Example 20 contains a counterexample to the converse of the third implication. Finally, the system B_3 from Example 18 is terminating though not match-bounded for $\mathrm{RFC}(B_3)$. One may wonder how these results relate to Zantema's *termination hierarchy* [31, 32]:

$$\begin{array}{l} \text{polynomially terminating} \Rightarrow \omega\text{-terminating} \Rightarrow \text{totally terminating} \Rightarrow \\ \quad \text{simply terminating} \Rightarrow \text{non-self-embedding} \Rightarrow \text{terminating} \end{array}$$

As it turns out, this hierarchy is orthogonal to all four properties mentioned above. Indeed, $B_3 = \{ab \to ba, ac \to ca, bc \to cb\}$ is polynomially terminating (choose $n \mapsto 3n + 1$, $n \mapsto 2n + 1$ and $n \mapsto n + 1$ as interpretation for a, b and c respectively), but not match-bounded for $\mathrm{RFC}(R)$. And $\{aa \to aba\}$ is a self-embedding system that is nevertheless match-bounded. So for any level X from the first four of the former hierarchy and any level Y from the first five of the latter hierarchy, we have $B_3 \in Y \setminus X$ and $\{aa \to aba\} \in X \setminus Y$.

11 Conclusion

If the flow of information during rewriting is suitably restricted, some desirable properties hold: termination, bounded derivational complexity, or preservation of regular languages. For instance, McNaughton [22] and independently Ferreira and Zantema [9] use extra letters to indicate the absence of information flow through certain positions. Kobayashi et al. [19] restrict derivations by using markers for the start and the end of a redex. Sénizergues [28] constructs finite automata to solve the termination problem for certain one-rule string rewriting systems. Moczydłowski and Geser [23, 24] restrict the way the right hand side of a rule may be consumed in order to simulate the rewrite relation by the computation of a pushdown automaton.

With our concepts of deleting and match-bounded string rewriting, we aim at extending these approaches to a systematic theory of *termination by language properties*. Regularity preservation forms a basis for automated termination proofs. We present two variants to demonstrate some of the potential of this new approach. Match-boundedness on the set of all strings over the given alphabet is easiest to conceive. On the other hand, match-boundedness on more restricted sets, for instance the right hand sides of forward closures, may significantly enlarge the application domain. Both variants can solve hard examples, like Zantema's system. Since match-boundedness entails RFC-match-boundedness, the latter variant is to be preferred for use in implementations.

We expect these powerful criteria to enable some major progress in the decision problem of uniform termination of one-rule string rewriting systems, a problem open for 13 years [20] (see also [27, Problem 21]). Our hope is supported by the fact that some hard one-rule systems can now be proven terminating automatically.

Single-player games like Peg Solitaire can be analyzed through the construction of reachability sets. It is challenging to extend this approach to two-player rewriting games [30]. Interesting properties are termination, which is necessary for a well-defined game, or regularity of winning sets. Even the impartial case is hard; here the central question is whether Grundy values are bounded.

It seems natural to carry over the notion of match-boundedness to term rewriting, in order to obtain both closure properties and new automated termination proof methods. For this purpose the theory of finite tree automata and regular tree languages comes in handy. However, a straightforward generalization from string to term rewriting is impossible in case non-linear rewrite rules are present.

Acknowledgements. This research was supported in part by the National Aeronautics and Space Administration (NASA) 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 helpful comments of three anonymous referees are gratefully acknowledged, and we thank Hans Zantema for valuable hints and discussions

References

1. Book, R.V., Jantzen, M., Wrathall, C.: Monadic Thue systems. Theoret. Comput. Sci. **19**, 231–251 (1982)
2. Book, R.V., Otto, F.: String-Rewriting Systems. Texts Monogr. Comput. Sci. Springer-Verlag, New York, 1993
3. Chidlovskii, B.: Using regular tree automata as XML schemas. In: J. Hoppenbrouwers, T. de Souza Lima, M. Papazoglou, A. Sheth (eds.), Proc. IEEE Advances in Digital Libraries ADL-00, IEEE Comput. Society, 2000, pp. 89–98
4. Choffrut, C., Karhumäki, J.: Combinatorics of words. In: G. Rozenberg, A. Salomaa (eds.), Handbook of Formal Languages. Vol. **1**, Springer-Verlag, 1998, pp. 329–438
5. Coquand, T., Persson, H.: A proof-theoretical investigation of Zantema's problem. In: M. Nielsen, 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
6. Dershowitz, N.: Termination of linear rewriting systems. In: S. Even, O. Kariv (eds.), Proc. 8th Int. Coll. Automata, Languages and Programming ICALP-81. Lecture Notes in Comput. Sci. Vol. **115**, Springer-Verlag, 1981, pp. 448–458
7. Dershowitz, N.: Termination of rewriting. J. Symbolic Comput. **3**(1–2), 69–115 (1987)
8. Dershowitz, N., 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
9. Ferreira, M.C.F., Zantema, H.: Dummy elimination: Making termination easier. In: H. Reichel (ed.), 10th Int. Symp. Fundamentals of Computation Theory FCT-95. Lecture Notes in Comput. Sci. Vol. **965**, Springer-Verlag, 1995, pp. 243–252
10. Genet, T., Klay, F.: Rewriting for cryptographic protocol verification. In: D. A. McAllester (ed.), 17th Int. Conf. Automated Deduction CADE-17. Lecture Notes in Artificial Intelligence Vol. **1831**, Springer-Verlag, 2000, pp. 271–290

11. Geser, A.: Is Termination Decidable for String Rewriting with only One Rule? Habilitationsschrift. Eberhard-Karls-Universität Tübingen, Germany, 2001
12. Geser, A., Hofbauer, D., Waldmann, J.: Match-bounded string rewriting systems. In: B. Rovan, 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
13. Geser, A., Hofbauer, D., Waldmann, J.: Match-bounded string rewriting systems and automated termination proofs. In: A. Rubio (ed.), Proc. 6th Int. Workshop on Termination WST-03. Technical Report DSIC-II/15/03, Universidad Politécnica de Valencia, Spain, 2003, pp. 19–22
14. Geser, A., Hofbauer, D., Waldmann, J., Zantema, H.: Finding finite automata that certify termination of string rewriting. In: K. Salomaa, S. Yu (eds.), Proc. 9th Int. Conf. Implementation and Application of Automata CIAA-04. Lecture Notes in Comput. Sci. to appear. Springer-Verlag, 2004
15. Geser, A., Zantema, H.: Non-looping string rewriting. RAIRO – Theoret. Inform. Appl. **33**, 279–302 (1999)
16. Ginsburg, S., Greibach, S.A.: Mappings which preserve context sensitive languages. Inform. Control. **9**(6), 563–582 (1966)
17. Hibbard, T.N.: Context-limited grammars. J. ACM **21**(3), 446–453 (1974)
18. Hofbauer, D., Waldmann, J.: Deleting string rewriting systems preserve regularity. In: Z. Ésik, Z. Fülöp (eds.), Proc. 7th Int. Conf. Developments in Language Theory DLT-03. Lecture Notes in Comput. Sci. Vol. **2710**, Springer-Verlag, 2003, pp. 337–348. Full version accepted for publication in Theoret. Comput. Sci.
19. Kobayashi, Y., Katsura, M., Shikishima-Tsuji, K.: Termination and derivational complexity of confluent one-rule string-rewriting systems. Theoret. Comput. Sci. **262**(1-2), 583–632 (2001)
20. Kurth, W.: Termination und Konfluenz von Semi-Thue-Systemen mit nur einer Regel. Dissertation, Technische Universität Clausthal, Germany, 1990
21. Lankford, D.S., Musser, D.R.: A finite termination criterion. Technical Report, Information Sciences Institute, Univ. of Southern California, Marina-del-Rey, CA, 1978
22. McNaughton, R.: Semi-Thue systems with an inhibitor. J. Automat. Reason. **26**, 409–431 (2001)
23. Moczydłowski, W.: Jednoregułowe systemy przepisywania słów. Masters thesis, Warsaw University, Poland, 2002
24. Moczydłowski, W., Geser, A.: Termination of single-threaded one-rule Semi-Thue systems – revised version. Unpublished manuscript, National Institute of Aerospace, Hampton, VA, 2004. Available at http://research.nianet.org/~geser/papers/single-revised.html
25. Moore, C., Eppstein, D.: One-dimensional peg solitaire, and duotaire. In: R.J. Nowakowski (ed.), More Games of No Chance. Cambridge Univ. Press, 2003
26. Ravikumar, B.: Peg-solitaire, string rewriting systems and finite automata. In: H.-W. Leong, H. Imai, 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
27. The RTA list of open problems. http://www.lsv.ens-cachan.fr/rtaloop/
28. 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
29. Tahhan Bittar, E.: Complexité linéaire du problème de Zantema. C. R. Acad. Sci. Paris Sér. I Inform. Théor. t. **323**, 1201–1206 (1996)
30. Waldmann, J.: Rewrite games. In: S. Tison (ed.), Proc. 13th Int. Conf. Rewriting Techniques and Applications RTA-02. Lecture Notes in Comput. Sci. Vol. **2378**, Springer-Verlag, 2002, pp. 144–158
31. Zantema, H.: Termination of term rewriting: interpretation and type elimination. J. Symbolic Comput. **17**(1), 23–50 (1994)

32. Zantema, H.: The termination hierarchy for term rewriting. Appl. Algebra Engrg. Comm. Comput. **12**(1-2), 3–19 (2001)
33. Zantema, H., Geser, A.: A complete characterization of termination of $0^p 1^q \to 1^r 0^s$. Appl. Algebra Engrg. Comm. Comput. **11**(1), 1–25 (2000)