Decomposing Terminating Rewrite Relations

Jörg Endrullis[1], Dieter Hofbauer[2], and Johannes Waldmann[3]

[1] Department of Computer Science, Vrije Universiteit Amsterdam
De Boelelaan 1081, 1081 HV Amsterdam, The Netherlands.
Email: joerg@few.vu.nl
[2] Mühlengasse 16, D-34125 Kassel, Germany.
Email: dieter@theory.informatik.uni-kassel.de
[3] Hochschule für Technik, Wirtschaft und Kultur (FH) Leipzig
Fb IMN, PF 30 11 66, D-04251 Leipzig, Germany.
Email: waldmann@imn.htwk-leipzig.de

1 Introduction

We decompose an arbitrary rewrite relation into the product of a context-free system and an inverse context-free system with empty right-hand sides. By requiring both of these relations to be terminating, we lose computational completeness and arrive at the class of *deleting* rewriting systems [5].

Our new treatment allows to efficiently construct the rewrite closure of a regular language with respect to deleting or *match-bounded* [3] rewriting. Previous implementations of this method either used a complete but inefficient decomposition algorithm [5] leading to impracticable resource consumption, or incomplete approximation algorithms [4]. Our new algorithm is both efficient and exact, thereby improving the power of automated termination provers that use the method of match-bounds.

2 Decomposing String Rewriting Systems

We denote *context-free* rewriting systems $\mathrm{CF} = \{R \mid \forall(\ell \to r) \in R : |\ell| \leq 1\}$, its subclass $\mathrm{CF}_0 = \{R \mid \forall(\ell \to r) \in R : |\ell| = 0\}$ and $\mathrm{SN} = \{R \mid \mathrm{SN}(\to_R)\}$. For a class $\mathcal{C}$ of string rewriting systems let $\mathcal{C}^- = \{R^- \mid R \in \mathcal{C}\}$.

Definition 1. *Let R be a string rewriting system over Σ, let S and T be string rewriting systems over $\Gamma \supseteq \Sigma$. Then the pair (S, T) is a* decomposition of R *if*

$$\to_R^* = (\to_S^* \circ \to_T^*) \cap (\Sigma^* \times \Sigma^*).$$

If additionally $S \in \mathcal{S}$ and $T \in \mathcal{T}$ for classes of string rewriting systems $\mathcal{S}$ and $\mathcal{T}$, then (S, T) is called an $(\mathcal{S}, \mathcal{T})$-decomposition *of R.*

The set of strings over a given alphabet is a monoid w.r.t. to concatenation, but this operation is not invertible. We introduce *formal left and right inverses* of letters. For a given alphabet Σ, define alphabets $\overrightarrow{\Sigma} = \{\overrightarrow{a} \mid a \in \Sigma\}$ and

$\overleftarrow{\Sigma} = \{\overleftarrow{a} \mid a \in \Sigma\}$, and let $\overline{\Sigma} = \Sigma \cup \overrightarrow{\Sigma} \cup \overleftarrow{\Sigma}$. We extend $\overrightarrow{}$ and $\overleftarrow{}$ from letters to strings by $\overrightarrow{a_1 \cdots a_n} = \overrightarrow{a_n} \cdots \overrightarrow{a_1}$ and $\overleftarrow{a_1 \cdots a_n} = \overleftarrow{a_n} \cdots \overleftarrow{a_1}$. The behaviour of inverse letters is expressed by the rewriting systems $\overrightarrow{E_\Sigma} = \{\overrightarrow{a}a \to \epsilon \mid a \in \Sigma\}$ and $\overleftarrow{E_\Sigma} = \{a\overleftarrow{a} \to \epsilon \mid a \in \Sigma\}$. We write $\overrightarrow{E}$ for $\overrightarrow{E_\Sigma}$ and $\overleftarrow{E}$ for $\overleftarrow{E_\Sigma}$, if Σ is clear from the context. Let $E = \overrightarrow{E} \cup \overleftarrow{E}$. Observe that $\overrightarrow{x}x \to_E^* \epsilon \leftarrow_E^* x\overleftarrow{x}$ for $x \in \Sigma^*$. The above construction is standard. The congruence relation generated by $\to_E$ is called the *Shamir congruence* in [6] II.6.2.

Definition 2. *For string rewriting systems R and S over $\overline{\Sigma}$ write $R \curvearrowright S$ if S results from R by replacing a rule $xa \to r$ by $x \to r\overrightarrow{a}$, or replacing a rule $ax \to r$ by $x \to \overleftarrow{a}r$, where $a \in \Sigma$. Let $\sim$ denote the equivalence generated by $\curvearrowright$. We say that R and S are* conjugates *if $R \sim S$.*

A finite system R has only finitely many conjugates, among them R, so the union of all its conjugates is finite. In the sequel, we denote this union by $C(R)$.

Lemma 1. *For every string rewriting system R over Σ,*

(1) $\to_{C(R)\cup E}^* \cap (\Sigma^* \times \Sigma^*) \subseteq \to_R^*$ (correctness), and
(2) $\to_R^* \subseteq \to_C^* \circ \to_E^*$ (completeness), for every context-free conjugate C of R.

Theorem 1. *Let R be a string rewriting system over Σ. Then $(C(R), E)$ is a decomposition of R, and if C is a context-free conjugate of R, then (C, E) is a $(\mathrm{CF}, \mathrm{CF}_0^-)$-decomposition of R.*

Every string rewriting system has a $(\mathrm{CF}, \mathrm{CF}_0^-)$-decomposition (C, E). We are especially interested in terminating decompositions.

Definition 3. *A string rewriting system R over Σ is called* deleting *if there is an irreflexive partial ordering $>$ on Σ such that for each $(\ell \to r) \in R$ there is some letter a in ℓ so that for each letter b in r, $a > b$.*

Lemma 2. *For a string rewriting system R, the following conditions are equivalent: (1) There is a terminating context-free conjugate of R. (2) R is deleting.*

Corollary 1. *Let R be a deleting string rewriting system, then*

(1) R has a $(\mathrm{SN} \cap \mathrm{CF}, \mathrm{SN} \cap \mathrm{CF}_0^-)$-decomposition, and
(2) [5] R preserves regularity and R^- preserves context-freeness.

Example 1. The rewriting system $R = \{ba \to cb, bd \to d, cd \to de, d \to \epsilon\}$ is deleting w.r.t. the ordering $a > b > c > d > e$. A terminating context-free conjugate of R is $C = \{a \to \overleftarrow{b}cb, b \to d\overrightarrow{d}, c \to de\overrightarrow{d}, d \to \epsilon\}$.

Following [3], we annotate letters by numbers, called *match heights*, to get more detailed information on rewrite sequences. We switch to the extended alphabet $\Gamma = \Sigma \times \mathbb{N}$ and abbreviate a_n for (a, n) in Γ. Define morphisms $\mathrm{base} : \Gamma \to \Sigma$, $\mathrm{height} : \Gamma \to \mathbb{N}$, and, for $n \in \mathbb{N}$, $\mathrm{lift}_n : \Sigma \to \Gamma$ by $\mathrm{base}(a_n) = a$, height$(a_n) = n$ and $\text{lift}_n(a) = a_n$. For a rewriting system R over Σ where $\epsilon \notin \text{lhs}(R)$ define the rewriting system

$$\text{match}(R) = \{\ell' \to \text{lift}_{m+1}(r) \mid (\ell \to r) \in R,\ \text{base}(\ell') = \ell,\ m = \min \text{height}(\ell')\}$$

over Γ. It simulates R-rewriting as $\to_R^* = \text{lift}_0 \circ \to_{\text{match}(R)}^* \circ \text{base}$. For a system S over $\Sigma \times \mathbb{N}$ let S_c denote the restriction of S to $\Sigma \times \{0, \dots, c\}$. The system R is called *match-bounded by* $c \in \mathbb{N}$ if $\to_{\text{match}(R)}^*(\text{lift}_0(\Sigma^*)) \subseteq (\Sigma \times \{0, \dots, c\})^*$.

Each system $\text{match}_c(R)$ is deleting w.r.t. the ordering defined by $a_m > b_n$ if $m < n$ and hence has a $(\text{SN} \cap \text{CF}, \text{SN} \cap \text{CF}_0^-)$-decomposition (C, E). Due to the special and uniform structure of $\text{match}(R)$, this decomposition can be improved. Giving up uniqueness of the inverses, we increase the "computational power" of inverses in using the rewriting system

$$E' = \{\overrightarrow{a_i} a_j \to \epsilon,\ a_j \overleftarrow{a_i} \to \epsilon \mid a \in \Sigma,\ j \geq i \geq 0\},$$

again over $\overline{\Gamma}$. In this extended sense, $\overrightarrow{a_2}$ becomes the left inverse of all letters $a_2, a_3, \dots$, for instance. Note that $E \subseteq E'$ and $C' \subseteq C$. With these more general inverses we obtain a succinct and efficient decomposition of $\text{match}(R)$.

$$\begin{aligned} C' = \{&\text{lift}_i(a) \to \text{lift}_i(\overleftarrow{x})\, \text{lift}_{i+1}(r)\, \text{lift}_i(\overrightarrow{y}) \mid \\ &(xay \to r) \in R,\ a \in \Sigma,\ x, y \in \Sigma^*,\ i \geq 0\} \end{aligned}$$

Theorem 2. (C', E') *is a* $(\text{SN} \cap \text{CF}, \text{SN} \cap \text{CF}_0^-)$*-decomposition of* $\text{match}(R)$.

Corollary 2. (C'_c, E'_c) *is a* $(\text{SN} \cap \text{CF}, \text{SN} \cap \text{CF}_0^-)$*-decomposition of* $\text{match}_c(R)$.

Corollary 3. *Every match-bounded string rewriting system has a* $(\text{SN} \cap \text{CF}, \text{SN} \cap \text{CF}^-)$*-decomposition.*

Example 2. Take $R = \{aa \to aba\}$, and consider decompositions of $\text{match}_2(R)$. This is Example 1 from [4], which contains a $(\text{SN} \cap \text{CF}, \text{SN} \cap \text{CF}^-)$ decomposition where both parts have 7 rules. By Corollary 2 we get $C'_2 = \{a_0 \to \overleftarrow{a_0} a_1 b_1 a_1,\ a_0 \to a_1 b_1 a_1 \overrightarrow{a_0},\ a_1 \to \overleftarrow{a_1} a_2 b_2 a_2,\ a_1 \to a_2 b_2 a_2 \overrightarrow{a_1}\}$ with 4 rules, and $E'_2 = \{\overrightarrow{a_0} a_0 \to \epsilon,\ \overrightarrow{a_0} a_1 \to \epsilon, \dots\}$ with 24 rules. In contrast, C_2 contains C'_2 and 6 additional rules $a_0 \to \overleftarrow{a_1} a_1 b_1 a_1,\ a_0 \to a_1 b_1 a_1 \overrightarrow{a_1}, \dots$, while $E_2 \subset E'_2$ and $|E_2| = 12$.

The result states that the drastic reduction from C_c to C'_c can be compensated by moderately enlarging E_c to E'_c. Note that $|C'_c| \leq |R| \cdot m \cdot c$ and $|C_c| \leq |R| \cdot m \cdot (c+1)^m$ for $m = \max\{|\ell| \mid \ell \in \text{lhs}(R)\}$, whereas $|E'_c| = |\Sigma| \cdot O(c^2)$ and $|E_c| = |\Sigma| \cdot O(c)$.

3 Automata

For the application of automated proofs of termination we are interested in finite automata A that represent sets of descendants with respect to $\text{match}_c(R)$.

An *automaton* (with epsilon transitions) $A = (\Sigma, Q, I, F, \delta)$ consists of an alphabet Σ, a set of states Q, sets $I, F \subseteq Q$ of initial and final states resp., and a transition relation $\delta \subseteq Q \times (\Sigma \cup \{\epsilon\}) \times Q$. A path $p \to_A q$ is called *ϵ-minimal* if it neither starts nor ends with an ϵ-transition.

Definition 4. *An automaton A over Σ is* compatible *(resp.* exactly compatible*) with a rewriting system R over Σ and a language L over Σ if $L \subseteq \mathcal{L}(A)$ (resp. $\to_R^* (L) = \mathcal{L}(A)$) and for each pair of states $p, q \in A$ and rule $(\ell \to r) \in R$ with $p \xrightarrow{\ell}_A q$, it holds that $p \xrightarrow{r}_A q$. If we omit L, then $L = \mathcal{L}(A)$.*

We will construct compatible representations of descendants of $\mathcal{L}(A)$ under rewriting. Therefore we give non-deterministic algorithms on automata.

Definition 5. *For automata A, B over Σ and states $p, q \in Q(A)$ and $w \in \Sigma^*$, we write $A \xrightarrow{(p,w,q)} B$ if B is obtained from A by adding transitions and states:*

- *if $|w| \leq 1$, then $Q(B) = Q(A)$ and $\delta(B) = \delta(A) \cup (p, w, q)$, and*
- *if $|w| > 1$, then $Q(B) = Q(A) \uplus \{s_1, \dots, s_{|w|-1}\}$ and B contains a path labelled w form p to q along the fresh states $s_1, \dots, s_{|w|-1}$.*

Definition 6. *For automata A, B over Σ and a rewriting system R over Σ, we write $A \xrightarrow{R} B$ if there exist states $p, q \in Q(A)$ and a rule $(\ell \to r) \in R$ such that there exists an ϵ-minimal path $p \xrightarrow{\ell}_A q$, $\neg(p \xrightarrow{r}_A q)$ and $A \xrightarrow{(p,r,q)} B$.*

Lemma 3. *Let R be rewriting system over Σ such that (1) R is terminating and context-free, or (2) R is inverse context-free. Then $\xrightarrow{R}$ is terminating, and for all automata A,B over Σ with $A \xrightarrow{R}{}^! B$, the automaton B is exactly compatible with R and $\mathcal{L}(A)$.*

Lemma 4 (off-line construction). *Let R be a string rewriting system with $(\mathrm{SN} \cap \mathrm{CF}, \mathrm{CF}_0^-)$-decomposition (C, E) such that C is a conjugate of R. If we have $A_0 \xrightarrow{C}{}^! A_1 \xrightarrow{E}{}^! A_2$ for automata A_0, A_1, A_2, then $A_2|_\Sigma$ exactly compatible with R and $\mathcal{L}(A_0)$.*

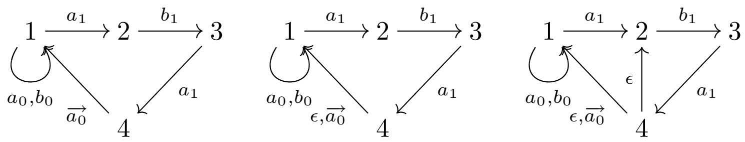
By $A_2|_\Sigma$ we denote the restriction of A_2 to the original alphabet. The off-line construction is inefficient since it introduces states and transitions that turn out to be unreachable later (i.e., they are in A_2, but not in $A_2|_\Sigma$). We give a method that constructs only accessible states and transitions.

Definition 7. *For a rewrite system R with $(\mathrm{SN} \cap \mathrm{CF}, \mathrm{CF}_0^-)$-decomposition (C, E) such that C is conjugate to R, and automata A, B we write $A \xrightarrow{R,C} B$ if there is a rule $(\ell \to r) \in R$ with $\ell = xay$ such that $p \xrightarrow{x} p' \xrightarrow{a} q' \xrightarrow{y} q$ is an ϵ-minimal path in A with $(p', a, q') \in \delta(A)$ and $\neg(p \xrightarrow{r}_A q)$ and $(a \to \overleftarrow{x}\, r\, \overrightarrow{y}) \in C$ such that $A \xrightarrow{(p', \overleftarrow{x}\, r\, \overrightarrow{y}, q')} B$.*

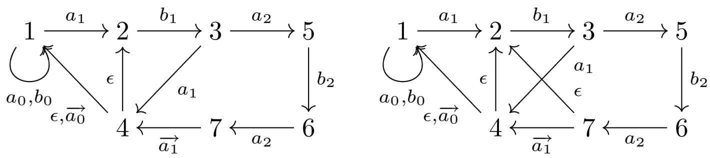
Lemma 5 (on-line construction). *For R, C, E as in Definition 7, the relation $\xrightarrow{R,C} \circ \xrightarrow{E}{}^!$ is terminating, and for automata A over Σ, B over Γ such that $A (\xrightarrow{R,C} \circ \xrightarrow{E}{}^!)^! B$, it holds that $B|_\Sigma$ is exactly compatible with R and $\mathcal{L}(A)$.*

This construction can be used to search for a certificate of match-boundedness. Starting with an automaton A for $\mathrm{lift}_0(L)$, we use the decomposition (C', E') of $\mathrm{match}(R)$. The construction stops if and only if R is match-bounded, yielding an exactly compatible automaton in the latter case.

Example 3. For $R = \{aa \to aba\}$ over $\Sigma = \{a, b\}$, we show how to verify that R is match-bounded by 2 (and thus terminating) for $L = \Sigma^*$. We start with an automaton $A_0 =$ a_0,b_0 1 , representing $\mathrm{lift}_0(L)$. (For all automata in this example, state 1 is both initial and final.) A_0 contains a $\mathrm{match}(R)$-redex path $1 \xrightarrow{a_0} 1 \xrightarrow{a_0} 1$. We choose the conjugate $a_0 \to a_1 b_1 a_1 \overrightarrow{a_0}$ and add its right-hand side, getting A_1 (left). It contains two E'-redex paths $4 \xrightarrow{\overrightarrow{a_0}} 1 \xrightarrow{a_0} 1$ and $4 \xrightarrow{\overrightarrow{a_0}} 1 \xrightarrow{a_1} 2$, so we add the transitions $4 \xrightarrow{\epsilon} 1$ (middle), and $4 \xrightarrow{\epsilon} 2$ resulting in A_3 (right).



Now there is a $\mathrm{match}(R)$-redex path $3 \xrightarrow{a_1} 4 \xrightarrow{\epsilon} 1 \xrightarrow{a_1} 2$. We choose a conjugate $a_1 \to a_2 b_2 a_2 \overrightarrow{a_1}$ and add its right-hand side as a path from 3 to 4 (left). Now there is an E'-redex $7 \xrightarrow{\overrightarrow{a_1}} 4 \xrightarrow{\epsilon} 1 \xrightarrow{a_1} 2$, so we add a transition $7 \xrightarrow{\epsilon} 2$, resulting in A_5 (right). A_5 is compatible with $\mathrm{match}(R)$.



To conclude, we consider the system $R = \{caac \to aaa,\ b \to aca,\ aba \to bb\}$. Our on-line algorithm constructs (within a few seconds) an exactly compatible automaton with about 30.000 states that certifies the RFC-match-bound 12.

References

1. R. V. Book, M. Jantzen, and C. Wrathall. Monadic Thue systems. *Theoret. Comput. Sci.*, 19:231–251, 1982.
2. J. Endrullis. *Effiziente Algorithmen für deleting und match-bounded Wortersetzungssysteme.* Diplomarbeit, Universität Leipzig, Germany, 2005.
3. A. Geser, D. Hofbauer and J. Waldmann. Match-bounded string rewriting systems. *Appl. Algebra Engrg. Comm. Comput.*, 15(3-4):149-171, 2004.
4. A. Geser, D. Hofbauer, J. Waldmann, and H. Zantema. Finding finite automata that certify termination of string rewriting. *Internat. J. Found. Comput. Sci.* 16(3):471–486, 2005.
5. D. Hofbauer and J. Waldmann. Deleting string rewriting systems preserve regularity. *Theoret. Comput. Sci.*, 327(3):301–317, 2004.
6. J. Sakarovitch. *Eléments de Théorie des Automates.* Vuibert, Paris, 2003.