AAECC 2, 257–274 (1992)

AAECC
Applicable Algebra in Engineering, Communication and Computing

Completing a Finite Special String-Rewriting System on the Congruence Class of the Empty Word

Friedrich Otto

Fachbereich Mathematik/Informatik, Gesamthochschule Kassel, Postfach 101380, W-3500 Kassel, FRG

Received September 4, 1990; revised version September 9, 1991

Abstract. Based on a polynomial-time test for determining whether a finite special string-rewriting system R is e-confluent, a procedure for completing a finite special system R on $[e]_R$ is derived. The correctness and completeness of this procedure are proved. In addition, the special case of finite special string-rewriting systems presenting groups is considered.

Keywords: String-rewriting system, Monoid-presentation, e-confluence, Completion procedure

1. Introduction

In the present paper we are interested in special string-rewriting systems. A string-rewriting system R on an alphabet Σ is called **special** if each rule of R is of the form $l \to e$, where l is a non-empty word and e denotes the empty word. These systems are of particular interest for the following reasons. On the one hand, the process of rewriting modulo a finite special string-rewriting system is particularly simple, since it only amounts to the insertion and deletion of subwords. On the other hand, each finitely presented group G can be presented by a finite special string-rewriting system R on some alphabet Σ, i.e., G is isomorphic to the factor monoid $\mathfrak{M}_R := \Sigma^*/\leftrightarrow_R^*$ of the free monoid Σ^* generated by Σ modulo the Thue congruence $\leftrightarrow_R^*$ induced by R. However, although finite special string-rewriting systems are fairly simple with respect to the structure of their rules, it is in general not possible to obtain much information on the Thue congruence $\leftrightarrow_R^*$ or on the monoid $\mathfrak{M}_R$ from a given finite special string-rewriting system R on Σ. For example, it is undecidable in general whether the monoid $\mathfrak{M}_R$ presented by a finite special string-rewriting system R is a group [8]. In fact, the undecidability of Markov properties can be carried over to the class of monoids that are presented by finite special string-rewriting systems, thus establishing that many algebraic properties of $\mathfrak{M}_R$ are undecidable in this setting [14].

The situation improves dramatically when attention is restricted to finite special string-rewriting systems R that are confluent. Let $\rightarrow_R^*$ denote the reduction relation induced by R, which is obtained by allowing the rules of R to only be applied from left to right. Then R is called **confluent** if, for all $u, v \in \Sigma^*$, $u \leftrightarrow_R^* v$ implies that $u \rightarrow_R^* w$ and $v \rightarrow_R^* w$ for some $w \in \Sigma^*$. Thus, if R is a finite special string-rewriting system that is confluent, then each congruence class mod $\leftrightarrow_R^*$ contains a unique word of minimal length, and given any word $u \in \Sigma^*$, the minimal word v congruent to u can easily be obtained. Hence, the word problem for R is easily decidable, but also many other problems become easily decidable in this setting [1].

If R is a finite special string-rewriting system on Σ that is not confluent, then one way to try to solve the word problem for R consists in trying to construct a finite special string-rewriting system S on Σ such that S is equivalent to R, i.e., the congruences $\leftrightarrow_S^*$ and $\leftrightarrow_R^*$ coincide, and S is confluent. For example, let $\Sigma = \{a, b, c\}$ and $R = \{(abacab, e), (abac, e), (acab, e)\}$. Then $ab \,{}_R\!\leftarrow abacab \rightarrow_R e$, and so R is not confluent. However, $S = \{(ab, e), (ac, e)\}$ is a finite special system that is equivalent to R and that is confluent. On the other hand, let $\Sigma = \{b, c\}$ and $R = \{(b^2, e), (bcbc, e)\}$. Then $cbc \,{}_R\!\leftarrow b^2cbc \rightarrow_R b$, and so R is not confluent. If $w \in \Sigma^*$ satisfies $w \leftrightarrow_R^* e$, then $|w|_b \equiv 0 \bmod 2$ and $|w|_c \equiv 0 \bmod 2$ as can easily be seen, where $|w|_b (|w|_c)$ denotes the number of occurrences of the letter b (c) in w. Thus, neither b nor any factor of cbc is congruent to e mod R, i.e., whenever S is a special string-rewriting system that is equivalent to R, then $cbc \leftrightarrow_S^* b$, but cbc and b are both irreducible mod S. Hence, there is no special and confluent string-rewriting system on Σ that is equivalent to R.

It has been shown that a finitely presented group G can be presented by a finite special and confluent string-rewriting system if and only if G is isomorphic to the free product of finitely many (finite or infinite) cyclic groups [2]. Thus, the monoid $\mathfrak{M}_R$ presented by $\Sigma = \{a, b, c, d\}$ and $R = \{(ad, e), (da, e), (b^2, e), (c^2, e), (bcbc, e)\}$ cannot be presented by any finite special and confluent string-rewriting system on any set of generators Γ, since $\mathfrak{M}_R$ is isomorphic to the free product $\mathbb{Z} * (\mathbb{Z}_2 \times \mathbb{Z}_2)$ of the free group $\mathbb{Z}$ of rank 1 and the direct product $\mathbb{Z}_2 \times \mathbb{Z}_2$ of the cyclic group $\mathbb{Z}_2$ of order 2 with itself. However, let $R_0 := R \cup \{(cbcb, e)\}$. Then R_0 is a finite special system that is equivalent to R. Of course, R_0 is not confluent either, but it has the following interesting property: for all $w \in \Sigma^*$, if $w \leftrightarrow_{R_0}^* e$, when $w \rightarrow_{R_0}^* e$, i.e., R_0 is confluent on $[e]_{R_0}$ or **e-confluent**. In particular, this implies that the process of reduction mod R_0 yields a procedure to test membership in $[e]_R$. Furthermore, since the monoid $\mathfrak{M}_R$ is a group, the word problem for R is reducible to the membership problem for $[e]_R$. Hence, the process of reduction mod R_0 gives a method to solve the word problem for R. In fact, many problems become decidable when they are restricted to the class of finite special string-rewriting systems R that are e-confluent, e.g., the word problem, the conjugacy problem, the generalized word problem, etc. [12, 13]. Also the class of groups that can be presented by these systems in strictly larger than the class of groups presented by finite special and confluent string-rewriting systems. In fact, each group G that is isomorphic to the free product of a finitely generated free group and finitely many finite groups can be presented by a finite special string-rewriting system R that is e-confluent, and it has been conjectured that no other groups have such presentations [6].

Here we present a specialized completion procedure that, given a finite special string-rewriting system R on some alphabet Σ as input, tries to construct a finite special system S on Σ that is equivalent to R and that is e-confluent.

This procedure consists of two subroutines called NORMALIZATION and CONTEXT_RESOLVING, where the latter introduces new rules in order to make the string-rewriting system considered e-confluent, while the former deletes superfluous rules in order to keep the system as small as possible. It is shown that this procedure either terminates with a finite special system S, or it enumerates an infinite special system S. In either case is S equivalent to R and e-confluent. Further, it is shown that this procedure terminates whenever there exists a finite special system that is equivalent to R and e-confluent. Thus, our specialized completion procedure is correct and complete.

The above completion procedure, which is presented in Sect. 3, is based on a test for determining whether a finite special string-rewriting system is e-confluent [11]. This test, although being polynomial-time, is technically rather involved. For the special case of finite special string-rewriting systems presenting groups, a much simpler test is derived in Sect. 4. Based on this simplified test a specialized completion procedure for finite special string-rewriting systems presenting groups is then presented. This procedure consists of three subroutines called NORMALIZATION, SYMMETRIZATION, and CONTEXT_RESOLVING, where the latter two introduce new rules in order to make the string-rewriting system considered e-confluent, while the former one deletes superfluous rules. Again this completion procedure is shown to be correct and complete.

Finally, in Sect. 5 we point to the relation between the subroutine SYMMETRIZATION in our second completion procedure and the notion of symmetrized group-presentation as it is considered in small cancellation theory [5], and we state a few problems for future research.

2. Preliminary Results

Let Σ be a finite alphabet. Then Σ^* denotes the set of words over Σ including the empty word e. A **special string-rewriting system** R on Σ is a subset of $\Sigma^+ \times \{e\}$, where $\Sigma^+ = \Sigma^* - \{e\}$ denotes the set of non-empty words over Σ. The elements (l, e) of R are called **(rewrite) rules.** For all $u, v \in \Sigma^*$ and $(l, e) \in R$, $ulv \to_R uv$, i.e., $\to_R$ is the **single-step reduction relation** induced by R. Its reflexive and transitive closure $\to_R^*$ is the **reduction relation** induced by R. For $u, v \in \Sigma^*$, if $u \to_R^* v$, then u is an **ancestor** of v, and v is a **descendant** of u. By $\nabla_R^*(v)$ we denote the set of all ancestors of v, and $\Delta_R^*(u)$ denotes the set of all descendants of u. For a subset $L \subseteq \Sigma^*$, $\nabla_R^*(L) = \bigcup_{u \in L} \nabla_R^*(u)$, and $\Delta_R^*(L) = \bigcup_{u \in L} \Delta_R^*(u)$.

By $\leftrightarrow_R^*$ we denote the smallest equivalence relation on Σ^* that contains the single-step reduction relation $\to_R$. It is called the **Thue congruence** generated by R. For $w \in \Sigma^*$, $[w]_R = \{u \in \Sigma^* \mid u \leftrightarrow_R^* w\}$ is the congruence class of w mod R. Since $\leftrightarrow_R^*$ is in fact a congruence relation on Σ^*, the set $\mathfrak{M}_R := \{[w]_R \mid w \in \Sigma^*\}$ of congruence classes is a monoid under the operation $[u]_R \circ [v]_R = [uv]_R$ with identity $[e]_R$. This monoid is uniquely determined (up to isomorphism) by Σ and R, and hence, whenever $\mathfrak{M}$ is a monoid that is isomorphic to $\mathfrak{M}_R$, we call the ordered pair $(\Sigma; R)$ a **(monoid-) presentation** of $\mathfrak{M}$ with **generators** Σ and **defining relations** R.

We say that a subset $L \subseteq \Sigma^*$ is **closed under cyclic permutation** if $uv \in L$ implies $vu \in L$ for all $u, v \in \Sigma^*$. The following observation will be useful for our investigations.

Lemma 2.1. *Let R be a special string-rewriting system on Σ. Then the set $\nabla_R^*(e)$ of ancestors of the empty word* mod R *is closed under cyclic permutation if and only if, for all $u, v \in \Sigma^+$, if $(uv, e) \in R$, then $vu \to_R^* e$.*

Proof. Obviously, the above condition is necessary for $\nabla_R^*(e)$ to be closed under cyclic permutation. Thus, it remains to prove that it is also sufficient. Assume to the contrary that $\nabla_R^*(e)$ is not closed under cyclic permutation, and let $z \in \Sigma^+$ be a word of minimal length such that $z \to_R^* e$, but there is a cyclic permutation z_1 of z such that $z_1 \not\to_R^* e$. Then there is a cyclic permutation $y = ax$ of z, where $a \in \Sigma$ and $x \in \Sigma^*$, such that $y = ax \to_R^* e$ while $xa \not\to_R^* e$. Assume that there exists a word x_1 such that $x \to_R x_1$ and $ax_1 \to_R^* e$. Then $xa \to_R x_1 a$, and since $|x_1| < |x|$, the choice of z implies that with $ax_1 \to_R^* e$ also $x_1 a \to_R^* e$, thus contradicting the choice of $y = ax$. Hence, whenever $x \to_R x_1$, then $ax_1 \not\to_R^* e$. Thus, the reduction $ax \to_R^* e$ consists of a single step only, i.e. $(ax, e) \in R$. But then the above condition yields that $xa \to_R^* e$, again contradicting the choice of y. Thus, $\nabla_R^*(e)$ is indeed closed under cyclic permutation, if the above condition is satisfied. □

Given a finite special string-rewriting system R on Σ, and a regular set $L \subseteq \Sigma^*$ specified through a nondeterministic finite state acceptor (nfsa), an nsfa for the set $\Delta_R^*(L)$ can be constructed in polynomial time [1]. Since, for all $w \in \Sigma^*$, $w \in \nabla_R^*(e)$ if and only if $e \in \Delta_R^*(w)$, this means that the membership problem for the set $\nabla_R^*(e)$ is decidable in polynomial time for each finite special string-rewriting system R. Together with Lemma 2.1 this yields the following result.

Theorem 2.2. *The following problem is decidable in polynomial time:*
INSTANCE: *A finite special string-rewriting system R on Σ.*
QUESTION: *Is $\nabla_R^*(e)$ closed under cyclic permutation?*

Let R be a special string-rewriting system on Σ. We say that R is **confluent on** $[w]_R$ for some word $w \in \Sigma^*$, if there exists a word $w_0 \in IRR(R)$ such that $[w]_R \cap IRR(R) = \{w_0\}$. Here $IRR(R)$ denotes the set of words that are **irreducible** mod R, i.e. $IRR(R) = \{w \in \Sigma^* \mid \Delta_R^*(w) = \{w\}\}$. Thus, R is confluent on $[w]_R$ if all words in that class reduce to the same irreducible word, which then can serve as a normal form for this class. The system R is called **e-confluent** if it is confluent on $[e]_R$. In [11] a necessary and sufficient condition for R to be e-confluent is derived. This condition involves the following technical notions.

Let (l_1, e) and (l_2, e) be two rules of R. If $l_1 = xl_2y$ for some $x, y \in \Sigma^*$ satisfying $xy \neq e$, or if $l_1 x = yl_2$ for some $x, y \in \Sigma^*$ satisfying $0 < |y| < |l_1|$, then the pair (e, xy), respectively (x, y), is called a **critical pair** of R. By $UCP(R)$ we denote the set $\{(x, y) \mid (x, y)$ is a critical pair of R such that $\Delta_R^*(x) \cap \Delta_R^*(y) = \emptyset\}$ of **unresolvable critical pairs** of R. Observe that for R finite, this set can be computed in polynomial time.

The system R is called **normalized** if no left-hand side of a rule of R contains another left-hand side as a factor. If R is normalized, then R can only admit critical pairs of the second form. Further, if (p, q) is a critical pair of R, then p and q are irreducible.

For $u \in \Sigma^+$, let $RF_R(u)$ denote the set

$$RF_R(u) = \{v \in \Sigma^* \mid \exists k \geq 1\ \exists u_1, \ldots, u_k, v_1, \ldots, v_k \in \Sigma^+ : u = u_k \cdots u_1,\ v = v_1 \cdots v_k, \text{ and } (u_1 v_1, e), (u_2 v_2, e), \ldots, (u_k v_k, e) \in R\},$$

i.e., $RF_R(u)$ consists of all those words v that are right-inverses of $u \bmod R$, where there exists a reduction $uv \to_R^* e$ each step of which straddles the boundary between u and v. Thus, if R is finite, then $v \in RF_R(u)$ implies that $|v| \leqq (\lambda - 1) \cdot |u|$, where $\lambda := \max\{|l| \mid (l, e) \in R\}$, i.e. $RF_R(u)$ is a finite set. Analogously, the set

$$LF_R(u) = \{v \in \Sigma^* \mid \exists k \geqq 1\ \exists u_1, \ldots, u_k, v_1, \ldots, v_k \in \Sigma^+ : u = u_1 \cdots u_k, v = v_k \cdots v_1, \text{ and } (v_1 u_1, e), (v_2 u_2, e), \ldots, (v_k u_k, e) \in R\}$$

is finite for each finite special system R. Further, given a finite special system R and a word $u \in \Sigma^+$, nfsas for the sets $RF_R(u)$ and $LF_R(u)$ can be constructed in polynomial time [11]. Finally, in order to cover all possible cases we take $RF_R(e) := \{e\}$ and $LF_R(e) := \{e\}$.

In [11] the following technical result is obtained.

Theorem 2.3. *Let R be a normalized special string-rewriting system on Σ. Then R is e-confluent if and only if the following two conditions hold for each pair $(p, q) \in UCP(R)$:*

(i) $\forall p_1, p_2, p_3, x, y \in \Sigma^*$: if $p = p_1 p_2 p_3$ and $(x p_2 y, e) \in R$ such that $p_2 \neq e$ and $LF_R(p_1) \neq \varnothing \neq RF_R(p_3)$, then $\triangle_R^*(x \cdot LF_R(p_1) \cdot q \cdot RF_R(p_3) \cdot y) \cap IRR(R) = \{e\}$, and

(ii) $\forall q_1, q_2, q_3, x, y \in \Sigma^*$: if $q = q_1 q_2 q_3$ and $(x q_2 y, e) \in R$ such that $q_2 \neq e$ and $LF_R(q_1) \neq \varnothing \neq RF_R(q_3)$, then $\triangle_R^*(x \cdot LF_R(q_1) \cdot p \cdot RF_R(q_3) \cdot y) \cap IRR(R) = \{e\}$.

In particular, this result implies that it is decidable in polynomial time whether a finite special and normalized string-rewriting system R is e-confluent.

Observe that, if R is a finite system, then the sets of the form $\{x\} \cdot LF_R(p_1) \cdot \{q\} \cdot RF_R(p_3) \cdot \{y\}$, respectively $\{x\} \cdot LF_R(q_1) \cdot \{p\} \cdot RF_R(q_3) \cdot \{y\}$, are always finite. Therewith the sets of irreducible descendants of these sets are finite as well.

The completion procedure we are about to describe will be based on Theorem 2.3. If a normalized finite special string-rewriting system R does not satisfy the two conditions stated there, we add further rules to R, thus trying to obtain another special system R' such that R' is e-confluent, and R and R' are equivalent, i.e., the congruences $\leftrightarrow_R^*$ and $\leftrightarrow_{R'}^*$ coincide. However, in order to keep the system R' normalized, we will also delete rules whenever that is possible. The basis for this is the following observation.

Let R be a finite special string-rewriting system on Σ, and let (l_1, e) and (l_2, e) be rules of R. If l_1 is a proper factor of l_2, i.e., $l_2 = x l_1 y$ for some words $x, y \in \Sigma^*$, $xy \neq e$, then $x y_R \leftarrow x l_1 y = l_2 \to_R e$. For R to be e-confluent it must be possible to reduce xy to e. Obviously, during this reduction the rule (l_2, e) cannot be used. Thus, if $xy \to_R^* e$, then the system $R - \{(l_2, e)\}$ generates the same reduction relation as the system R. In particular, $R - \{(l_2, e)\}$ is equivalent to R, and the one system is e-confluent if and only if the other system is. Thus, instead of dealing with R we can deal with the smaller system. On the other hand, if $xy \not\to_R^* e$, then R is not e-confluent. Hence, to complete R on $[e]_R$, rules must be introduced that allow to reduce xy to e. So instead of R we may consider the system $(R - \{(l_2, e)\}) \cup \{(xy, e)\}$, which is equivalent to R, and which is smaller in the sense that a rule has been replaced by a smaller one. This process will be called **normalization**.

3. The Completion Procedure for Finite Special String-Rewriting Systems

Let R be a finite special string-rewriting system on Σ. We would like to obtain a special system S that is equivalent to R and e-confluent. To this end we first normalize R, and then we check whether or not R itself is e-confluent. If it is, we are done; otherwise, we must try to construct S from R. However, if R is not e-confluent, then one of the conditions of Theorem 2.3 is violated. Therefore, we have some information on a particular situation that violates the property of e-confluence for R. We now present a completion procedure that exploits this information. It consists of two subroutines: NORMALIZATION and CONTEXT_RESOLVING. The former realizes the process of normalization explained at the end of the previous section, while the latter adds new rules if the conditions of Theorem 2.3 are violated. Since the latter may destroy the effect of the former, and since new rules may lead to new unresolvable critical pairs, we have to keep applying these two subroutines repeatedly until a stable system is obtained.

Procedure 3.1. E-completion for finite special string-rewriting systems:
INPUT: A finite special string-rewriting system R on some alphabet Σ;
begin $i \leftarrow 0$; $R_i \leftarrow R$;
NORMALIZATION:
while $\exists l_1, l_2, x, y \in \Sigma^* : xy \neq e \wedge l_2 = xl_1y \wedge (l_1, e) \in R_i \wedge (l_2, e) \in R_i$ **do**
begin $R_i \leftarrow R_i - \{(l_2, e)\}$;
if $e \notin \triangle^*_{R_i}(xy)$ **then** $R_i \leftarrow R_i \cup \{(xy, e)\}$
end;
comment: At this point R_i is normalized;
CONTEXT_RESOLVING:
compute $UCP(R_i)$; $R'_i \leftarrow \emptyset$;
for all $(p, q) \in UCP(R_i)$ **do**
begin for all $p_1, p_2, p_3, x, y \in \Sigma^*$ **do**
if $p = p_1p_2p_3 \wedge p_2 \neq e \wedge (xp_2y, e) \in R_i$ **then**
begin $S_p \leftarrow (\triangle^*_{R_i}(x \cdot LF_{R_i}(p_1) \cdot q \cdot RF_{R_i}(p_3) \cdot y) \cap IRR(R_i)) - \{e\}$;
if $S_p \neq \emptyset$ **then** $R'_i \leftarrow R'_i \cup \{(l, e) \mid l \in S_p\}$
end;
for all $q_1, q_2, q_3, x, y \in \Sigma^*$ **do**
if $q = q_1q_2q_3 \wedge q_2 \neq e \wedge (xq_2y, e) \in R_i$ **then**
begin $S_q \leftarrow (\triangle^*_{R_i}(x \cdot LF_{R_i}(q_1) \cdot p \cdot RF_{R_i}(q_3) \cdot y) \cap IRR(R_i)) - \{e\}$;
if $S_q \neq \emptyset$ **then** $R'_i \leftarrow R'_i \cup \{(l, e) \mid l \in S_q\}$
end;
end;
if $R'_i \neq \emptyset$ **then**
begin $R_{i+1} \leftarrow R_i \cup R'_i$;
$i \leftarrow i + 1$;
(*) **goto** NORMALIZATION
end;
comment: At this point R_i is normalized and e-confluent;
OUTPUT: R_i
end.

We claim that the above procedure determines a finite special string-rewriting system R_i that is e-confluent and equivalent to R, whenever such a system exists. Otherwise it enumerates an infinite special system R_∞ having both these properties. As a first step towards proving this result we consider the subroutine NORMALIZATION. The following facts easily follow from the remarks at the end of Sect. 2.

Lemma 3.2. *Let R be a finite special string-rewriting system on Σ. Then on input R, the subroutine NORMALIZATION determines a finite special string-rewriting system R_0 on Σ such that R_0 is normalized, $\to_R \subseteq \to_{R_0}^*$, and R_0 is equivalent to R.*

Given a finite special string-rewriting system R as input, Procedure 3.1 computes a (finite or infinite) sequence of finite special string-rewriting systems $R_0, R_1, R_2, \ldots$, where R_{i-1} denotes the system that is determined by the subroutine NORMALIZATION during the i-th execution of the body of the goto-loop (*). Recall that if R_i is finite, then the sets S_p and $S_q((p,q) \in UCP(R_i))$ are finite. Thus, R_i' is finite, which in turn yields that R_{i+1} is finite.

Lemma 3.3. *For all $i \geq 0$, the following statements hold:*
(a) *R_i is normalized,*
(b) *R_i is equivalent to R, and*
(c) $\to_R^* \subseteq \to_{R_i}^* \subseteq \to_{R_{i+1}}^*$.

Proof. R_0 is determined by the subroutine NORMALIZATION from the input system R. Thus, by Lemma 3.2 R_0 is normalized and equivalent to R, and $\to_R^* \subseteq \to_{R_0}^*$. We proceed by induction on i. For $i \geq 1$, R_i is determined by the subroutine NORMALIZATION from the system $R_{i-1} \cup R_{i-1}'$. Hence, by Lemma 3.2 R_i is normalized and equivalent to $R_{i-1} \cup R_{i-1}'$, and $\to_{R_{i-1}} \cup \to_{R_{i-1}'} \subseteq \to_{R_i}^*$. Further, by the induction hypothesis R_{i-1} is equivalent to R, and $\to_R^* \subseteq \to_{R_{i-1}}^*$. Thus, $\to_R^* \subseteq \to_{R_i}^*$, and since $\leftrightarrow_{R_{i-1}} \subseteq \leftrightarrow_{R_{i-1}}^*$, $R_{i-1} \cup R_{i-1}'$ is equivalent to R, which implies that R_i is equivalent to R. This completes the proof of Lemma 3.3. □

From this lemma we can now easily derive the fact that Procedure 3.1 is correct, i.e., it satisfies the following statement.

Corollary 3.4. *Let R be a finite special string-rewriting system on Σ. If Procedure 3.1 terminates on input R, then it yields a finite special system R_i on Σ that is normalized, e-confluent, and equivalent to R.*

Proof. Procedure 3.1 terminates on input R, if, for some $i \geq 0$, then system R_i' is empty. In this case the finite special system R_i is taken as output. By Lemma 3.3 R_i is normalized and equivalent to R. Since $R_i' = \emptyset$, R_i satisfies conditions (i) and (ii) of Theorem 2.3. Hence, R_i is also e-confluent. □

Thus, whenever Procedure 3.1 terminates, then the system R_i constructed has indeed all the properties we want. It remains to show that this procedure does terminate whenever a special system S exists that is finite, equivalent to R, and e-confluent. As a first step towards proving this fact, we analyse the situation when Procedure 3.1 does not terminate.

Lemma 3.5. *Let R be a finite special string-rewriting system on Σ. If Procedure 3.1 does not terminate on input R, then it enumerates an infinite special system R_∞ that is normalized, equivalent to R, and e-confluent.*

Proof. Assume that Procedure 3.1 does not terminate on input R. Then it enumerates an infinite sequence $R_0, R_1, R_2, \ldots$ of finite special string-rewriting systems on Σ.

Because of Lemma 3.3(c) we have $IRR(R_{i+1}) \subseteq IRR(R_i)$. In fact, if $(l,e) \in R_i'$, then $l \in IRR(R_i)$ (see the construction of R_i' in the subroutine CONTEXT_RESOLVING), while $l \notin IRR(R_{i+1})$, since $\to_{R_{i+1}}^* \supseteq \to_{R_i} \cup \to_{R_i'}$ by Lemma 3.2. Thus, since Procedure 3.1 does not terminate after the $i+1$st execution of the body of the goto-loop (*), $R_i' \neq \emptyset$ implying that $IRR(R_{i+1}) \subset IRR(R_i)$. Further, if a rule (l,e) is deleted in the process of normalizing the system R_i, then $l \notin IRR(R_i)$, and therewith $l \notin IRR(R_j)$ for all $j \geq i$. This means that the rule (l,e) is not reintroduced at a later stage.

Now let $i \geq 0$ and let $(l,e) \in R_i$. If this rule is contained in R_j for all $j \geq i$, then this rule is called **persistent**. If (l,e) is not persistent, then it is deleted in the process of normalizing the system R_j for some $j > i$. Thus, R_j must contain a rule (l_1, e) such that l_1 is a proper factor of l. Now either (l_1, e) is a persistent rule, or (l_1, e) is again deleted in the process of normalizing the system R_k for some $k > j$, which means that R_k contains a rule (l_2, e) such that l_2 is a proper factor of l_1. However, this can only happen a finite number of times. Hence, there exist an index λ and a rule (x,e) such that x is a proper factor of l, and $(x,e) \in R_p$ for all $p \geq \lambda$. Thus, each rule $(l,e) \in \bigcup_{i \geq 0} R_i$ is either persistent, or there is a persistent rule (x,e) such that x is a proper factor of l.

Let $R_\infty := \{(l,e) \mid \exists j \geq 0 \forall i \geq j : (l,e) \in R_i\}$ be the set of persistent rules. The above discussion shows that R_∞ is an infinite special system. Procedure 3.1 can be interpreted as enumerating this system. Of course, this enumeration is not an effective one, since Procedure 3.1 does not identify the persistent rules. The discussion above also shows that $IRR(R_\infty) \subseteq \bigcap_{i \geq 0} IRR(R_i)$. In fact, the following holds.

Claim 1: $\to_{R_i} \subseteq \to_{R_\infty}^*$ for all $i \geq 0$.

Proof. It suffices to show that $l \to_{R_\infty}^* e$ for all $(l,e) \in \bigcup_{i \geq 0} R_i$. Assume to the contrary that there exists a rule $(l,e) \in \bigcup_{i \geq 0} R_i$ such that $l \not\to_{R_\infty}^* e$, and assume that (l,e) is chosen from all the rules having this property such that $|l|$ is minimal. Since $l \not\to_{R_\infty}^* e$, we have $(l,e) \notin R_\infty$. Hence, there is an index j such that the rule (l,e) is deleted in the process of normalizing R_j, i.e., $l = xl_1y$ for some $x, y \in \Sigma^*$, $xy \neq e$, and some rule $(l_1, e) \in R_j$. Since $|l_1| < |l|$, we have $l_1 \to_{R_\infty}^* e$ according to the choice of (l,e). Further, the way in which the subroutine NORMALIZATION works guarantees that $xy \to_{R_j}^* e$. Since $|xy| < |l|$, all the rules $(z,e) \in R_j$ used to reduce xy to e have the property that $z \to_{R_\infty}^* e$. Thus, $l = xl_1y \to_{R_\infty}^* xy \to_{R_\infty}^* e$, contradicting the choice of (l,e). This proves the claim. □

Hence, $\to_R \subseteq \to_{R_\infty}^*$ by Lemma 3.3(c). On the other hand, if $(l,e) \in R_\infty$, then $(l,e) \in R_j$ for some $j \geq 0$, and so $l \leftrightarrow_R^* e$ by Lemma 3.3(b). Thus, R_∞ is equivalent to R.

Claim 2: R_∞ is normalized.

Proof. Assume that (l_1, e) and (xl_1y, e) are both in R_∞, where $xy \neq e$. Then there is an index $j \geq 0$ such that $(l_1, e), (xl_1y, e) \in R_j$. However, this contradicts the fact that R_j is normalized. □

Finally, we can prove the following claim.

Claim 3: R_∞ is e-confluent.

Proof. Let $(p,q)\in UCP(R_\infty)$, let $p_1,p_2,p_3,x,y\in\Sigma^*$ such that $p=p_1p_2p_3$, $p_2\neq e$, and $(xp_2y,e)\in R_\infty$, and let $u\in LF_{R_\infty}(p_1)$ and $v\in RF_{R_\infty}(p_3)$. We must verify that $\triangle^*_{R_\infty}(xuqvy)\cap IRR(R_\infty)=\{e\}$ holds.

Since $(p,q)\in UCP(R_\infty)$, there are rules $(l_1,e),(l_2,e)\in R_\infty$ such that $l_1q=pl_2$, where $0<|p|<|l_1|$, and p and q do not have a common descendant mod R_∞. Since R_∞ only contains the persistent rules, there is an index $j\geqq 0$ such that $(l_1,e),(l_2,e),(xp_2y,e)\in R_i$ for all $i\geqq j$. Hence, (p,q) is a critical pair for all R_i, $i\geqq j$. Since $\to^*_{R_i}\subseteq\to^*_{R_\infty}$ for all $i\geqq j$ by Claim 1, we see that this pair cannot be resolved mod R_i for any $i\geqq j$, i.e., $(p,q)\in UCP(R_i)$ for all $i\geqq j$.

Since $u\in LF_{R_\infty}(p_1)$, we have $\mathrm{up}_1\to^*_{R_\infty}e$, and each step in this reduction straddles the boundary between u and p_1. Only a finite number of rules is used in this reduction, and hence, there is an index $k\geqq j$ such that $\mathrm{up}_1\to^*_{R_k}e$ coincides with the reduction $\mathrm{up}_1\to^*_{R_\infty}e$. Hence, $u\in LF_{R_k}(p_1)$. In fact, $u\in LF_{R_i}(p_1)$ for all $i\geqq k$. Analogously it is shown that $v\in RF_{R_i}(p_3)$ for all i that are sufficiently large. Thus, $xuqvy\in\{x\}\cdot LF_{R_i}(p_1)\cdot\{q\}\cdot RF_{R_i}(p_3)\cdot\{y\}$ for all sufficiently large indices i. Let i_0 be such an index. If $xuqvy\to^*_{R_{i_0}}e$, then $xuqvy\to^*_{R_\infty}e$ by Claim 1, i.e., $e\in\triangle^*_{R_\infty}(xuqvy)$. If $xuqvy\not\to^*_{R_{i_0}}e$, then $xuqvy\to^*_{R_{i_0}}w\in IRR(R_{i_0})-\{e\}$, and so $(w,e)\in R'_{i_0}$. Thus, $w\to^*_{R_{i_0+1}}e$ by Lemma 3.2, and hence, $xuqvy\to^*_{R_\infty}w\to^*_{R_\infty}e$ by Claim 1. This means that in any case $e\in\triangle^*_{R_\infty}(xuqvy)$.

Finally, assume that $xuqvy\to^*_{R_\infty}z$ for some $z\in IRR(R_\infty)-\{e\}$. Then $z\in\bigcap_{i\geqq 0}IRR(R_i)$, and $xuqvy\to^*_{R_i}z$ for all sufficiently large indices i. However, $z\in IRR(R_i)$ and $xuqvy\to^*_{R_i}z$ imply that $(z,e)\in R'_i$, which in turn yields that $z\notin IRR(R_{i+1})$, thus contradicting the above observation. Hence, $\triangle^*_{R_\infty}(xuqvy)\cap IRR(R_\infty)=\{e\}$. Since this holds for all $u\in LF_{R_\infty}(p_1)$ and $v\in RF_{R_\infty}(p_3)$, we can conclude that $\triangle^*_{R_\infty}(x\cdot LF_{R_\infty}(p_1)\cdot q\cdot RF_{R_\infty}(p_3)\cdot y)\cap IRR(R_\infty)=\{e\}$. Thus, it follows that R_∞ is indeed e-confluent.

This completes the proof of Lemma 3.5. □

Thus, on input a finite special string-rewriting system R, Procedure 3.1 always "computes" a special string-rewriting system R_∞ that is normalized, equivalent to R, and e-confluent. Procedure 3.1 terminates if and only if this system R_∞ is finite. Hence, it remains to characterize the condition under which this system R_∞ is indeed finite.

To this end let R be a finite special string-rewriting system on Σ, and let $S(R)$ denote the following special system:

$$S(R)=\{(l,e)\mid l\in[e]_R,\text{ but no proper factor of } l \text{ belongs to } [e]_R\}.$$

Lemma 3.6. *$S(R)$ is normalized, e-confluent, and equivalent to R.*

Proof. Obviously, $\leftrightarrow_{S(R)}\subseteq\leftrightarrow^*_R$, and $S(R)$ is normalized. On the other hand, if $w\in[e]_R$, then $w\to^*_{S(R)}e$. To prove this fact we proceed by induction on $|w|$. If no proper factor of w belongs to $[e]_R$, then $(w,e)\in S(R)$. Otherwise, $w=ulv$ for some $u,v\in\Sigma^*$, $uv\neq e$, and $(l,e)\in S(R)$. Then $w\to_{S(R)}uv$, and $uv\leftrightarrow^*_R e$. Since $|uv|<|w|$, we can conclude that $uv\to^*_{S(R)}e$ by the induction hypothesis. Thus, $w\to_{S(R)}uv\to^*_{S(R)}e$. Hence, $S(R)$ is equivalent to R, and $S(R)$ is e-confluent. □

As it will turn out, $S(R)$ is not just some normalized special string-rewriting system that is equivalent to R and e-confluent, but $S(R)$ is in fact the only system having all these properties.

Lemma 3.7. *Let T be a special string-rewriting system that is normalized, e-confluent, and equivalent to R. Then T coincides with the system $S(R)$.*

Proof. Let $(l, e) \in T$. Then $l \leftrightarrow_R^* e$, and so $l \rightarrow_{S(R)}^* e$. Thus, either $(l, e) \in S(R)$, or $l = uxv$ for some $u, v \in \Sigma^*$, $uv \neq e$, and some rule $(x, e) \in S(R)$. In the latter case, $x \leftrightarrow_R^* e$ implying that $x \leftrightarrow_T^* e$, i.e., $x \rightarrow_T^* e$, since T is e-confluent. However, this contradicts the fact that T is normalized. Thus, $T \subseteq S(R)$. Analogously, the converse inclusion can be verified, i.e., T actually coincides with $S(R)$. □

Thus, for each finite special string-rewriting system R there is a unique normalized special system that is e-confluent and equivalent to R. This coincides with the situation for length-reducing string-rewriting systems that are confluent everywhere [3]. For R the corresponding system $S(R)$ is either finite, in which case Procedure 3.1 must terminate on input R according to Lemma 3.5, or $S(R)$ is infinite, in which case Procedure 3.1 cannot terminate on input R according to Corollary 3.4. In either case Procedure 3.1 "computes" the system $S(R)$.

If T is a finite special system that is e-confluent and equivalent to R, then the process of normalization yields a finite subsystem T_1 of T that is still e-confluent and equivalent to R (see the discussion at the end of Sect. 2). Thus, $T_1 = S(R)$, and so $S(R)$ is finite in this case.

Combining all these results we obtain the following.

Corollary 3.8. *Let R be a finite special string-rewriting system on Σ. On input R, Procedure 3.1 computes a normalized special string-rewriting system $S(R)$ on Σ such that $S(R)$ is e-confluent and equivalent to R. Procedure 3.1 terminates if and only if the system $S(R)$ is finite, which happens if and only if there exists a finite special string-rewriting system S on Σ such that S is e-confluent and equivalent to R.*

Thus, Procedure 3.1 succeeds whenever there exists a finite special system that has all the required properties. Unfortunately, the following problem is undecidable [9.14]:

INSTANCE: A finite special string-rewriting system R on Σ.

QUESTION: Is the corresponding system $S(R)$ finite?

This means that it is undecidable in general whether or not Procedure 3.1 will terminate given a finite special string-rewriting system R as input. We close this section with a detailed example.

Example 3.9. Let $\Sigma = \{a, b, c, d\}$ and $R = \{ad \rightarrow e, da \rightarrow e, b^2 \rightarrow e, c^2 \rightarrow e, bcbc \rightarrow e\}$. Then the monoid $\mathfrak{M}_R$ is the free product $\mathbb{Z} * (\mathbb{Z}_2 \times \mathbb{Z}_2)$, which cannot be presented by any finite special and confluent string-rewriting system [2]. The system R is normalized, and $UCP(R) = \{(b, cbc), (c, bcb)\}$. For the critical pair $(p, q) := (b, cbc)$ the subroutine CONTEXT_RESOLVING performs the following computations:

(1) $p_1 = e$, $p_2 = b$, $p_3 = e$: Then $LF_R(p_1) = \{e\} = RF_R(p_3)$.

Now the following words $xp_2y \in \mathrm{dom}(R)$ are considered:

(i) $x = e$, $y = b$: $\triangle_R^*(x \cdot LF_R(p_1) \cdot q \cdot RF_R(p_3) \cdot y) \cap IRR(R) = \triangle_R^*(cbcb) \cap IRR(R) = \{cbcb\}$.

(ii) $x = b, y = e$: $\triangle_R^*(x \cdot LF_R(p_1) \cdot q \cdot RF_R(p_3) \cdot y) \cap IRR(R) = \triangle_R^*(bcbc) \cap IRR(R) = \{e\}$.
(iii) $x = e, y = cbc$: $\triangle_R^*(cbccbc) \cap IRR(R) = \{e\}$.
(iv) $x = bc, y = c$: $\triangle_R^*(bccbcc) \cap IRR(R) = \{e\}$.

(2) $q_1 = e, q_2 = cbc, q_3 = e$: Then $LF_R(q_1) = \{e\} = RF_R(q_3)$.
(i) $x = b, y = e$: $\triangle_R^*(x \cdot LF_R(q_1) \cdot p \cdot RF_R(q_3) \cdot y) \cap IRR(R) = \triangle_R^*(bb) \cap IRR(R) = \{e\}$.

(3) $q_1 = e, q_2 = cb, q_3 = c$: Then $LF_R(q_1) = \{e\}$ and $RF_R(q_3) = \{c\}$.
(i) $x = b, y = c$: $\triangle_R^*(bbcc) \cap IRR(R) = \{e\}$.

(4) $q_1 = c, q_2 = bc, q_3 = e$: Then $LF_R(q_1) = \{c, bcb\}$ and $RF_R(q_3) = \{e\}$.
(i) $x = e, y = bc$: $\triangle_R^*(\{c, bcb\} \cdot bbc) \cap IRR(R) = \{e\}$.
(ii) $x = bc, y = e$: $\triangle_R^*(bc \cdot \{c, bcb\} \cdot b) \cap IRR(R) = \{e\}$.

(5) $q_1 = e, q_2 = c, q_3 = bc$: Then $LF_R(q_1) = \{e\}$ and $RF_R(q_3) = \{bc, cb, c^2bc\}$.

For $x = c$ and $y = e$ we obtain $\triangle_R^*(cb \cdot \{bc, cb, c^2bc\}) \cap IRR(R) = \{e, cbcb\}$, while all other possible choices of x and y yield $\triangle_R^*(x \cdot LF_R(q_1) \cdot b \cdot RF_R(q_3) \cdot y) \cap IRR(R) = \{e\}$.

(6) $q_1 = c, q_2 = b, q_3 = c$: Then $LF_R(q_1) = \{c, bcb\}$ and $RF_R(q_3) = \{c\}$.

For $x = e$ and $y = b$ we obtain $\triangle_R^*(\{c, bcb\} \cdot bcb) \cap IRR(R) = \{e, cbcb\}$, while all other choices just yield the set $\{e\}$.

(7) $q_1 = cb, q_2 = c, q_3 = e$: Then $LF_R(q_1) = \{bc, b^2cb\}$ and $RF_R(q_3) = \{e\}$.
Again, for $x = c$ and $y = e$ we obtain the set $\{e, cbcb\}$, while all other possible choices yield the set $\{e\}$.

The critical pair (c, bcb) is symmetric to the first one. Hence, we obtain the system $R_1 := R_0 \cup \{cbcb \to e\}$. As it turns out this system is normalized and e-confluent. Procedure 3.1 terminates with output R_1. □

4. E-Completing Special String-Rewriting Systems that Present Groups

Let R be a special string-rewriting system on Σ. If the monoid $\mathfrak{M}_R$ is a group, then for all $u, v \in \Sigma^*$, $uv \leftrightarrow_R^* e$ implies that $vu \leftrightarrow_R^* e$, too, i.e., the congruence class $[e]_R$ is closed under cyclic permutation. If, in addition, R is e-confluent, then $[e]_R = \nabla_R^*(e)$ and hence, $\nabla_R^*(e)$ is closed under cyclic permutation. For finite R, this property is decidable in polynomial time (Theorem 2.2).

Now let us reconsider Example 3.9. The monoid $\mathfrak{M}_R$ is a group, $(bcbc, e) \in R$, but $c(bcb) \not\to_R^* e$. Hence, $\nabla_R^*(e)$ is not closed under cyclic permutation, and this immediately implies that R is not e-confluent. In fact, for special string-rewriting systems presenting groups we have the following simplified test for e-confluence.

Theorem 4.1. *Let R be a normalized special string-rewriting system on Σ such that the monoid $\mathfrak{M}_R$ is a group. Then R is e-confluent if and only if the following two conditions are satisfied:*

1. $\nabla_R^*(e)$ is closed under cyclic permutation, and
2. $\forall (p, q) \in UCP(R): (\triangle_R^*(q \cdot RF_R(p)) \cap IRR(R)) - \{e\} = \varnothing = (\triangle_R^*(p \cdot RF_R(q)) \cap IRR(R)) - \{e\}$.

Proof. First assume that R is e-confluent. Then $[e]_R = \nabla_R^*(e)$, and since $\mathfrak{M}_R$ is a group, $[e]_R$ is closed under cyclic permutation. Thus, condition (1.) is satisfied.

Further, let $(p,q)\in UCP(R)$, and let $v\in RF_R(p)$. Then $p\leftrightarrow_R^* q$, and $pv\rightarrow_R^* e$. Thus, $qv\leftrightarrow_R^* e$, and so, since R is e-confluent, e is the only irreducible descendant of qv. Hence, $(\triangle_R^*(q\cdot RF_R(p))\cap IRR(R))-\{e\}=\varnothing$, and by symmetry, $(\triangle_R^*(p\cdot RF_R(q))\cap IRR(R))-\{e\}=\varnothing$.

To prove the converse implication let $(p,q)\in UCP(R)$. By [10, Theorem 2.1] it suffices to show that $L_p(e)=L_q(e)$, where $L_p(e)=\{x\#y\mid x,y\in IRR(R), xpy\rightarrow_R^* e\}$ and $L_q(e)=\{x\#y\mid x,y\in IRR(R), xqy\rightarrow_R^* e\}$. Here $\#$ is an additional letter that is not in Σ. So let $x,y\in IRR(R)$ such that $xpy\rightarrow_R^* e$. By (1.) $\nabla_R^*(e)$ is closed under cyclic permutation, and so $pyx\rightarrow_R^* e$, too. Since R is a special system, this implies that there exists a word $z\in\Sigma^*$ such that $yx\rightarrow_R^* z$ and $z\in RF_R(p)$. By (2.) $qz\rightarrow_R^* e$ implying that $qyx\rightarrow_R^* qz\rightarrow_R^* e$. Again by (1.) this yields $xqy\rightarrow_R^* e$, i.e., $x\#y\in L_q(e)$. Thus, $L_p(e)\subseteq L_q(e)$. By symmetry we also obtain the converse inclusion, and so $L_p(e)=L_q(e)$. □

Observe that if R is a finite special system, then, for each $(p,q)\in UPC(R)$, the sets $\{q\}\cdot RF_R(p)$ and $\{p\}\cdot RF_R(q)$ are finite, and therewith the sets of descendants $\triangle_R^*(q\cdot RF_R(p))\cap IRR(R)$ and $\triangle_R^*(p\cdot RF_R(q))\cap IRR(R)$ are finite, too.

We now present a procedure that on input a finite special string-rewriting system R presenting a group tries to construct a special string-rewriting system S that is e-confluent and equivalent to R. This procedure contains three subroutines: NORMALIZATION, SYMMETRIZATION, and CONTEXT_RESOLVING. The first one realizes the process of normalization. and the second introduces new rules if necessary to obtain a system R_1 that is equivalent to R such that $\nabla_{R_1}^*(e)$ is closed under cyclic permutation. It is based on Lemma 2.1. The third one finally takes care of condition (2.) of Theorem 4.1. Since applications of the subroutines SYMMETRIZATION and CONTEXT_RESOLVING may destroy the effect obtained by previous applications of the subroutines NORMALIZATION and SYMMETRIZATION, respectively, we have to keep applying all three subroutines until a stable system is obtained.

Procedure 4.2. E-completion for finite special string-rewriting systems presenting groups:

INPUT: A finite special string-rewriting system R on some alphabet Σ such that the monoid $\mathfrak{M}_R$ presented by $(\Sigma;R)$ is a group;
begin $i\leftarrow 0$; $R_i\leftarrow R$;
NORMALIZATION:
while $\exists l_1,l_2,x,y\in\Sigma^*: xy\neq e\wedge l_2=xl_1y\wedge(l_1,e)\in R_i\wedge(l_2,e)\in R_i$ **do**
begin $R_i\leftarrow R_i-\{(l_2,e)\}$;
if $e\notin\triangle_{R_i}^*(xy)$ **then** $R_i\leftarrow R_i\cup\{(xy,e)\}$
end;
comment: At this point the system R_i is normalized;
SYMMETRIZATION:
while $\exists l_1,l_2\in\Sigma^+:(l_1l_2,e)\in R_i\wedge e\notin\triangle_{R_i}^*(l_2l_1)$ **do**
$R_i\leftarrow R_i\cup\{(l_2l_1,e)\}$;
comment: At this point $\nabla_{R_i}^*(e)$ is closed under cyclic permutation;
(*) **if** $\exists l_1,l_2,x,y\in\Sigma^*: xy\neq e\wedge l_2=xl_1y\wedge(l_1,e)\in R_i\wedge(l_2,e)\in R_i$ **then**
goto NORMALIZATION;
comment: At this point R_i is normalized, and $\nabla_{R_i}^*(e)$ is closed under cyclic permutation;

CONTEXT_RESOLVING:
compute $UCP(R_i)$; $R_i' \leftarrow \varnothing$;
for all $(p, q) \in UCP(R_i)$ **do**
begin $S_p \leftarrow (\triangle_{R_i}^*(q \cdot RF_{R_i}(p)) \cap IRR(R_i)) - \{e\}$;
$S_q \leftarrow (\triangle_{R_i}^*(p \cdot RF_{R_i}(q)) \cap IRR(R_i)) - \{e\}$;
if $S_p \neq \varnothing$ **then** $R_i' \leftarrow R_i' \cup \{(l, e) | l \in S_p\}$;
if $S_q \neq \varnothing$ **then** $R_i' \leftarrow R_i' \cup \{(l, e) | l \in S_q\}$
end;
if $R_i' \neq \varnothing$ **then**
begin $R_{i+1} \leftarrow R_i \cup R_i'$;
$i \leftarrow i + 1$;
(**) **goto** NORMALIZATION;
end;
comment: At this point R_i is normalized and e-confluent;
OUTPUT: R_i
end.

We claim that the above procedure determines a finite special string-rewriting system R_i that is e-confluent and that is equivalent to R, whenever such a system exists. Otherwise it enumerates an infinite special system R_∞ having both these properties. As a first step towards proving this claim we show that on input a finite special system R the innermost goto-loop (*) is executed only a finite number of times before the subroutine CONTEXT_RESOLVING is entered.

Lemma 4.3. *Let R be a finite special string-rewriting system on Σ such that the monoid $\mathfrak{M}_R$ is a group. Then, on input R, a finite number of iterations of the innermost goto-loop (*) yields a finite special string-rewriting system R_0 satisfying the following conditions:*

(i) R_0 is equivalent to R,
(ii) R_0 is normalized,
(iii) $\nabla_{R_0}^*$ is closed under cyclic permutation, and
(iv) $\rightarrow_R \subseteq \rightarrow_{R_0}^*$.

Proof. By Lemma 3.2 the subroutine NORMALIZATION determines a finite special system R' satisfying conditions (i), (ii), and (iv). Now the subroutine SYMMETRIZATION may add some rules of the form $(l_2 l_1, e)$, where $(l_1 l_2, e) \in R'$. Since R' is finite, only finitely many rules of this form can be added, and hence, this subroutine terminates with a finite special system R'' containing R'. Since the monoid $\mathfrak{M}_R$ is a group, R'' is equivalent to R, and by Lemma 2.1 $\nabla_{R''}^*(e)$ is closed under cyclic permutation. If the system R'' is also normalized, then $R_0 := R''$ satisfies all the conditions (i) to (iv), and the goto-loop (*) is left. Otherwise the subroutine NORMALIZATION is called again with R''.

Cycling through the subroutines NORMALIZATION and SYMMETRIZATION no rule (x, e) is ever generated such that $|x| > \lambda$, where $\lambda = \max\{|l| | (l, e) \in R\}$. Further, even if a rule (l_2, e) is deleted in the subroutine NORMALIZATION, we still have $l_2 \rightarrow^* e$. Thus, this rule will not be added again later on, neither in the subroutine NORMALIZATION nor in the subroutine SYMMETRIZATION. Hence, the goto-loop (*) terminates after a finite number of iterations, and it yields a finite special string-rewriting system R_0 with the stated properties. □

Given a finite special string-rewriting system R as input such that the monoid $\mathfrak{M}_R$ is a group, Procedure 4.2 computes a (finite or infinite) sequence of finite special string-rewriting systems $R_0, R_1, R_2, \ldots$. Here R_{i-1} denotes the system that is determined by the subroutines NORMALIZATION and SYMMETRIZATION (i.e., in the goto-loop (*)) during the i-th execution of the body of the outer goto-loop (**). Based on Lemma 4.3 the following properties of these systems can be derived.

Lemma 4.4. *For all $i \geq 0$, the following statements hold:*
(a) R_i is normalized,
(b) R_i is equivalent to R,
(c) $\nabla^*_{R_i}(e)$ is closed under cyclic permutation, and
(d) $\to^*_R \subseteq \to^*_{R_i} \subseteq \to^*_{R_{i+1}}$.

Proof. Analogously to the proof of Lemma 3.3. □

It now easily follows from Theorem 4.1 and Lemma 4.4 that Procedure 4.2 is correct.

Corollary 4.5. *Let R be a finite special string-rewriting system on Σ such that the monoid $\mathfrak{M}_R$ is a group. If Procedure 4.2 terminates on input R, then it yields a finite special system R_i on Σ that is normalized, e-confluent, and equivalent to R.*

As a first step towards proving that Procedure 4.2 is also complete, we now investigate the situation when this procedure does not terminate.

Lemma 4.6. *Let R be a finite special string-rewriting system on Σ such that the monoid $\mathfrak{M}_R$ is a group. If Procedure 4.2 does not terminate on input R, then it enumerates an infinite special system R_∞ that is normalized, e-confluent, and equivalent to R.*

Proof. Assume that Procedure 4.2 does not terminate on input R, i.e., it enumerates an infinite sequence $R_0, R_1, R_2, \ldots$ of finite special string-rewriting systems on Σ. By Lemma 4.4(d) $\to^*_{R_i} \subseteq \to^*_{R_{i+1}}$ for all $i \geq 0$. In fact, since Procedure 4.2 does not terminate with the system R_i, we have $R'_i \neq \emptyset$, and so $\to^*_{R_i} \subset \to^*_{R_i \cup R'_i} \subseteq \to^*_{R_{i+1}}$ and $IRR(R_{i+1}) \subset IRR(R_i)$. In particular, a rule that is deleted in the process of normalizing the system R_i is not reintroduced at a later stage.

As in the proof of Lemma 3.5 we observe that Procedure 4.2 enumerates an infinite special system $R_\infty := \{(l,e) \mid \exists j \geq 0\ \forall i \geq j : (l,e) \in R_i\}$ of persistent rules. Again this system, the rules of which Procedure 4.2 does not identify effectively, satisfies $IRR(R_\infty) \subseteq \bigcap_{i \geq 0} IRR(R_i)$. In addition, the following properties of R_∞ can be established in much the same way as in the proof of Lemma 3.5:
1. $\to_{R_i} \subseteq \to^*_{R_\infty}$ for all $i \geq 0$, and therewith R_∞ is equivalent to R,
2. R_∞ is normalized, and
3. R_∞ satisfies condition (2.) of Theorem 4.1.

Finally $\nabla^*_{R_\infty}(e)$ is closed under cyclic permutation, since if $u, v \in \Sigma^*$ are such that $(uv, e) \in R_\infty$, then $(uv, e) \in R_j$ for some $j \geq 0$. By Lemma 4.4(c) $\nabla^*_{R_j}(e)$ is closed under cyclic permutation, and so $vu \to^*_{R_j} e$. Hence, by (1.) above $vu \to^*_{R_\infty} e$. Thus, $\nabla^*_{R_\infty}(e)$ is closed under cyclic permutation by Lemma 2.1.

Now Theorem 4.1 implies that R_∞ is e-confluent. This completes the proof of Lemma 4.6. □

According to Lemma 3.7 $S(R) = \{(l, e) | l \in [e]_R$, but no proper factor of l belongs to $[e]_R\}$ is the only special system that is normalized, e-confluent, and equivalent to R. Thus, Corollary 4.5 and Lemma 4.6 yield the following result.

Theorem 4.7. *Let R be a finite special string-rewriting system on Σ such that the monoid $\mathfrak{M}_R$ is a group. Then on input R, Procedure 4.2 computes a normalized special string-rewriting system $S(R)$ on Σ such that $S(R)$ is e-confluent and equivalent to R. Procedure 4.2 terminates if and only if the system $S(R)$ is finite, which in turn happens if and only if there exists at all a finite special string-rewriting system S on Σ that is e-confluent and equivalent to R.*

Again termination of Procedure 4.2 is undecidable, since the following problem is undecidable by [9, Theorem 5.1.3]:

INSTANCE: A finite special string-rewriting system R on Σ such that the monoid $\mathfrak{M}_R$ is a group.

QUESTION: Is the corresponding system $S(R)$ finite?

We close this section by presenting two examples to illustrate the way Procedure 4.2 works. Each time we start with a finite special string-rewriting system R presenting a group. By stepping through Procedure 4.2 applied to input R we construct a special string-rewriting system S that is e-confluent and equivalent to R.

Example 4.8. (Example 3.9 revisited). Let $\Sigma = \{a, b, c, d\}$ and $R = \{ad \to e,\ da \to e,\ b^2 \to e,\ c^2 \to e,\ bcbc \to e\}$. Then $\mathfrak{M}_R$ is the group $\mathbb{Z} * (\mathbb{Z}_2 \times \mathbb{Z}_2)$.

On input $(\Sigma; R)$ Procedure 4.2 first computes the string-rewriting system $R_0 := R \cup \{cbcb \to e\}$ using the subroutines NORMALIZATION and SYMMETRIZATION. This system is equivalent to R, it is normalized, and $\nabla^*_{R_0}(e)$ is closed under cyclic permutation. Then the subroutine CONTEXT_RESOLVING is entered, and the following computations take place:

$$UCP(R_0) = \{(cbc, b), (bcb, c)\},$$
$$RF_{R_0}(b) = \{b, cbc\},\ RF_{R_0}(c) = \{c, bcb\},$$
$$RF_{R_0}(cbc) = \{b, cbc, bc^2, c^2b, bcbcb, c^2bc^2, bcb^2c, cb^2cb, c^2bcbcb, bcbcbc^2, bcbcbcbcb, bcb^3cb\}, \text{ and}$$
$$RF_{R_0}(bcb) = \{c, bcb, cb^2, b^2c, cbcbc, b^2cb^2, cbc^2b, bc^2bc, b^2cbcbc, cbcbcb^2, cbcbcbcbc, cbc^3bc\}.$$

Further, $\triangle^*_{R_0}(b \cdot RF_{R_0}(cbc)) \cap IRR(R_0) = \{e\}$,
$\triangle^*_{R_0}(cbc \cdot RF_{R_0}(b)) \cap IRR(R_0) = \{e\} = \triangle^*_{R_0}(c \cdot RF_{R_0}(bcb)) \cap IRR(R_0)$, and
$\triangle^*_{R_0}(bcb \cdot RF_{R_0}(c)) \cap IRR(R_0) = \{e\}$, and hence, $R'_0 = \varnothing$. Thus, R_0 is a finite special string-rewriting system on Σ that is equivalent to R, and that is e-confluent. □

Example 4.9. Let $\Sigma = \{a, b, c, f, g\}$ and $R = \{ab \to e, ba \to e, c^3 \to e,\ fg \to e,\ gf \to e,\ bfbf \to e,\ fc^2g \to e\}$. Then R is normalized, and the monoid $\mathfrak{M}_R$ is obviously a group. Actually, $\mathfrak{M}_R \cong \mathbb{Z} * \mathbb{Z}_2$, and so this monoid can be presented by a finite special and confluent string-rewriting system on some alphabet Γ [2]. However, no finite special and confluent string-rewriting system is equivalent to R, since $a \leftrightarrow^*_R fbf$, but no factor of fbf is congruent to e mod R. On input R, Procedure 4.2 performs the following computations.

First the subroutine SYMMETRIZATION is applied to R. It yields the system $S_1 := R \cup \{fbfb \to e,\ c^2gf \to e,\ cgfc \to e,\ gfc^2 \to e\}$. Since S_1 is not normalized, the subroutine NORMALIZATION is called. It disposes of the rules $c^2gf \to e$,

$cgfc \to e$, and $gfc^2 \to e$ while introducing the rule $c^2 \to e$. Because of this new rule, the rules $c^3 \to e$ and $fc^2g \to e$ are then also deleted, and the rule $c \to e$ is introduced. Finally, the rule $c^2 \to e$ is deleted, i.e., the system $R_0 = \{ab \to e, ba \to e, c \to e, fg \to e, gf \to e, bfbf \to e, fbfb \to e\}$ is obtained. This system is normalized, and $\nabla^*_{R_0}$ is closed under cyclic permutation. Now $UCP(R_0) = \{(a, fbf), (g, bfb)\}$, i.e., the sets $RF_{R_0}(a)$, $RF_{R_0}(fbf)$, $RF_{R_0}(g)$, and $RF_{R_0}(bfb)$ must be determined. As can be checked easily, $RF_{R_0}(a) = \{b\}$, $RF_{R_0}(g) = \{f\}$,

$$RF_{R_0}(fbf) = \{b, bfg, bfbfb, gfb, gag, gabfb, gfbfg, gfbfbfb, bfbag, bfbabfb, bfbfbfg, bfbfbfbfb\}, \text{ and}$$
$$RF_{R_0}(bfb) = \{f, fba, fbfbf, abf, aga, agfbf, abfba, abfbfbf, fbfga, fbfbfba, fbfgfbf, fbfbfbfbf\}.$$

While $\triangle^*_{R_0}(fbf \cdot RF_{R_0}(a)) \cap IRR(R_0) = \{e\} = \triangle^*_{R_0}(bfb \cdot RF_{R_0}(g)) \cap IRR(R_0)$, we have $\triangle^*_{R_0}(a \cdot RF_{R_0}(fbf)) \cap IRR(R_0) = \{e, agag\}$ and $\triangle^*_{R_0}(g \cdot RF_{R_0}(bfb)) \cap IRR(R_0) = \{e, gaga\}$. This gives the string-rewriting system $R_1 := R_0 \cup \{agag \to e, gaga \to e\}$, which is normalized. Further, $\nabla^*_{R_1}(e)$ is closed under cyclic permutation, and $UCP(R_1) = \{(a, fbf), (g, bfb), (b, gag), (f, aga)\}$. Now $RF_{R_1}(a) = \{b, gag\}$, $RF_{R_1}(b) = \{a, fbf\}$, $RF_{R_1}(g) = \{f, aga\}$, and $RF_{R_1}(f) = \{g, bfb\}$. Further, $RF_{R_1}(fbf) = RF_{R_0}(fbf)$ and $RF_{R_1}(bfb) = RF_{R_0}(bfb)$, while $RF_{R_1}(gag) = \{a, agf, agaga, fga, fbf, fbaga, fgagf, fgagaga, agabf, agabaga, agagagf, agagagaga\}$ and $RF_1(aga) = \{g, gab, gagag, bag, bfb, bfgag, bagab, bagagag, gagfb, gagagab, gagfgag, gagagagag\}$. Finally,

$$\triangle^*_{R_1}(a \cdot RF_{R_1}(fbf)) \cap IRR(R_1) = \{e\} = \triangle^*_{R_1}(fbf \cdot RF_{R_1}(a)) \cap IRR(R_1),$$

and the same is true for the other three critical pairs. Thus, R_1 is a finite special string-rewriting system on Σ that is equivalent to R, and that is e-confluent. □

5. Concluding Remarks

The examples presented at the end of the previous sections are fairly simple ones. The reason for this is the fact that they have been done by hand calculations. Naturally it would be interesting to investigate the behavior of our completion procedures for some more complex examples. For example, one of the questions one would like to answer by constructing appropriate examples is the following: for each $k > 1$, does there exist a finite special string-rewriting system R such that, on input R, Procedure 3.1 (respectively, Procedure 4.2) halts after executing the body of the (outermost) goto-loop exactly $k + 1$ times, i.e. the system R_k is e-confluent, but the system R_{k-1} is not? Due to the number and size of the sets $LF_{R_i}(u)$ and $RF_{R_i}(v)$ involved, this will be possible only by using an actual implementation of these completion procedures. Such an implementation is currently under way.

A part of Procedure 4.2 is the subroutine SYMMETRIZATION. Given a finite special string-rewriting system R presenting a group, this subroutine adds rules of the form $l_2l_1 \to e$, if $(l_1l_2, e) \in R$ and $l_2l_1 \not\to^*_R e$. In this way a finite special system S is obtained that is equivalent to R such that $\nabla^*_S(e)$ is closed under cyclic permutation. This subroutine was motivated by the notion of a **symmetrized group presentation** [5]. Let $\Sigma = \{a_1, \ldots, a_n\}$ be a finite alphabet, let $\bar{\Sigma} = \{\bar{a}_1, \ldots, \bar{a}_n\}$ be an alphabet in one-to-one correspondence to Σ such that $\Sigma \cap \bar{\Sigma} = \emptyset$, and let $^- : \Sigma \to \bar{\Sigma}$ denote the

obvious bijection. For a subset $L \subseteq (\Sigma \cup \bar{\Sigma})^*$, $R(L)$ denotes the Thue system $R(L) := \{(u, e) | u \in L\} \cup \{(a_i \bar{a}_i, e), (\bar{a}_i a_i, e) | i = 1, \ldots, n\}$. Then the monoid $\mathfrak{M}_{R(L)}$ presented by $(\Sigma \cup \bar{\Sigma}; R(L))$ is a group, and the ordered pair $\langle \Sigma; L \rangle$ is called a **group-presentation** for this group. Define a mapping $^{-1}: (\Sigma \cup \bar{\Sigma})^* \to (\Sigma \cup \bar{\Sigma})^*$ through: $e^{-1} := e$, $(wa_i)^{-1} := \bar{a}_i(w^{-1})$, and $(w\bar{a}_i)^{-1} := a_i(w^{-1})$ for all $w \in (\Sigma \cup \bar{\Sigma})^*$, and $i = 1, \ldots, n$. Then, for all $w \in (\Sigma \cup \bar{\Sigma})^*$, $ww^{-1} \leftrightarrow^*_{R(L)} e \leftrightarrow^*_{R(L)} w^{-1}w$, i.e., w^{-1} is a **formal inverse** of w. Observe that $\bar{a}_i$ is a formal inverse of a_i, i.e., each generator has a formal inverse of length 1 in the setting of group presentations. This is not true in general for finite special string-rewriting systems presenting groups.

A word $w \in (\Sigma \cup \bar{\Sigma})^*$ is called **freely reduced** if it does not contain a factor of the form $a_i \bar{a}_i$ or $\bar{a}_i a_i$; it is called **cyclically reduced** if it is freely reduced, and if it is not of the form $w = a_i u \bar{a}_i$ or $w = \bar{a}_i u a_i$. Obviously, if a word w is cyclically reduced, then so is each cyclic permutation of w. Now a subset $L \subseteq (\Sigma \cup \bar{\Sigma})^*$ is called **symmetrized**, if the following holds for each word $w \in L$: w is cyclically reduced, and each cyclic permutation of w as well as of w^{-1} belongs to L. If L is symmetrized, then the set $\nabla^*_{R(L)}(e)$ is closed under cyclic permutation by Lemma 2.1. Thus, when applied to a finite group-presentation the subroutine SYMMETRIZATION essentially constructs a symmetrized group-presentation equivalent to the given one.

If a finite symmetrized set $L \subseteq (\Sigma \cup \bar{\Sigma})^*$ satisfies certain combinatorial conditions (called **small cancellation conditions** [4, 5]), then the word problem for $\langle \Sigma; L \rangle$ can be solved by Dehn's algorithm. This algorithm essentially consists in computing normal forms modulo a finite length-reducing string-rewriting system S on $(\Sigma \cup \bar{\Sigma})$ that is equivalent to $R(L)$ and that is e-confluent. LeChenadic [4] presents a process he calls the group symmetrization algorithm that on input a finite symmetrized group presentation $\langle \Sigma; L \rangle$ satisfying certain small cancellation conditions generates the finite length-reducing system S mentioned above.

Procedures 3.1 and 4.2 only deal with finite special string-rewriting systems. For which classes of less restricted string-rewriting systems can corresponding completion procedures be developed? A specialized completion procedure for finite monadic string-rewriting systems presenting groups has been proposed in [7]. Recall that a string-rewriting system R on Σ is **monadic**, if each rule $(l, r) \in R$ satisfies $|l| > |r|$ and $|r| \leqq 1$. It is shown in [7] that is decidable in polynomial time whether a finite monadic string-rewriting system R presenting a group is e-confluent. On the other hand, for finite monadic string-rewriting systems in general the problem of deciding confluence on a given congruence class seems to be very hard. It has been shown to be decidable, but the algorithm given in [10] uses doubly exponential time. Thus, as a first step towards generalizing Procedure 3.1 to the class of all finite monadic string-rewriting systems, a much more efficient algorithm for testing confluence on a given congruence class must be developed for finite monadic systems.

6. References

1. Book, R. V.: Decidable sentences of Church–Rosser congruences. Theor. Comput. Sci. **23**, 301–312 (1983)
2. Cochet, Y.: Church–Rosser congruences on free semigroups. In: Algebraic Theory of Semigroups. Colloquia Mathematica Societatis Janos Bolyai **20**, pp. 51–60. Amsterdam: North-Holland 1976

3. Kapur, D., Narendran, P.: The Knuth–Bendix completion procedure and Thue systems. SIAM J. Comput. **14**, 1052–1072 (1985)
4. LeChenadec, Ph.: Canonical Forms in Finitely Presented Algebras. London: Pitman, New York, Toronto: Wiley 1986
5. Lyndon, R. C., Schupp, P. E.: Combinatorial group theory. Berlin, Heidelberg, New York: Springer 1977
6. Madlener, K., Otto, F.: About the descriptive power of certain classes of finite string-rewriting systems. Theor. Comput. Sci. **67**, 143–172 (1989)
7. Madlener, K., Narendran, P., Otto, F.: A specialized completion procedure for monadic string-rewriting systems presenting groups. In: Albert, J. L., Monien, B., Artalejo, M. R. (eds.) Automata, Languages, and Programming. Proceedings of the 18th Int. Coll., Lecture Notes Computer Science, Vol. 510, pp. 279–290. Berlin, Heidelberg, New York: Springer 1991
8. Narendran, P., O'Dunlaing, C., Otto, F.: It is Undecidable Whether a Finite Special String-Rewriting System Presents a Group. Discrete Math. (to appear)
9. O'Dunlaing, C.: Finite and Infinite Regular Thue Systems; Ph.D. Dissertation, Department of Mathematics, University of California at Santa Barbara (1981)
10. Otto, F.: On deciding the confluence of a finite string-rewriting system on a given congruence class. J. Comp. Sys. Sci. **35**, 285–310 (1987)
11. Otto, F.: The problem of deciding confluence on a given congruence class is tractable for finite special string-rewriting systems. Preprint No. 4/90, FB Math., GhK Kassel, West Germany (1990); also: Math. Systems Theory (to appear)
12. Otto, F., Zhang, L.: Decision problems for finite special string-rewriting systems that are confluent on a given congruence class. Acta Informatica **28**, 477–510 (1991)
13. Zhang, L.: Conjugacy in special monoids. J. Algebra **143**, 487–497 (1991)
14. Zhang, L.: The word problem and Markov properties for finitely presented special monoids. Submitted for publication