# COMBINATORIAL REWRITING ON TRACES *

Volker Diekert
Institut für Informatik
Technische Universität München
Arcisstr. 21, D-8000 München 2

**Abstract**

There are two main problems in working with replacement systems over free partially commutative monoids: For finite noetherian systems confluence is undecidable, in general, and the known algorithm to compute irreducible normal forms need time square in the derivation length instead of linear. We first give a decidable and sufficient condition for finite noetherian systems such that confluence becomes decidable. This condition is weaker than the known ones before. Then we give a decidable and sufficient condition such that irreducible normal forms are computable in time linear to the derivation length. Furthermore, we prove that the first condition is implied by the second. We also present a new uniform algorithm for computing normal forms using Zielonka's theory of asynchronous automata.

## 1 Introduction

In Combintorics free partially commutative monoids have been introduced in [CF69]. In computer science they serve today as an algebraic model for concurrent processes. This is mainly due to the work of A. Mazurkiewicz [Maz77] who called the elements of these monoids traces, a notion which is now standard. Since then an intensive study of traces under various aspects has begun, let us mention [Maz87], [AR88] and [Per89] for recent overviews.

In the present paper we continue to consider rewriting on traces. In [Die87] we introduced trace replacement systems in order to have an abstract calculus for transformations of concurrent processes. Trace replacement system generalize (and unify) the notion of semi-Thue systems and vector replacement systems. They could also be viewed as semi-Thue systems together with specified set of symmetric rules, this is the approach, for example, in [BL87], [NO88], [Wra88], or as term rewriting system modulo an equivalence relation [Ott89].

A basic question with respect to trace replacement systems is how to decide their word problem. This leads to the investigation of noetherian and confluent systems. The

*The paper also appears in the proceedings of the 7th Symposium on Theoretical Aspects of Computer Science, Rouen (France) 1990, Lecture Notes in Computer Science, Springer: Berlin.

noetherian property can always be achieved by directing the rules of the systems appropriately. But unfortunately, even for finite length reducing systems, deciding confluence is recursively unsolvable. Therefore we need (good) sufficient and computable conditions for deciding confluence.

Another important question is how fast we can decide the word problem by means of a finite noetherian and confluent system. This is essentially the same question as to ask how fast irreducible descendants are computable. Now, if the number of possible derivation steps is polynomial in the length of input traces, then it is trivial to see that the problem of computing irreducible normal forms rests tractable. But this is a very weak assertion. In order to allow efficient calculations it is desirable (and may be necessary) that we can compute irreducible normal forms of traces for a given finite noetherian system in time linear to the number of derivation steps. (Let us abbreviate this by *relatively linear time complexity*). However, the known algorithm achieve time square in that number only. In [Die89] we gave a sufficient condition on the set of left-hand sides such that the relatively linear time complexity can be obtained. But we left open the question whether confluence is decidable if this condition holds. In the present paper we give a slightly weaker condition which is still decidable and sufficient to ensure the relatively linear time complexity and we also prove that confluence is decidable if this weaker condition holds. This result is shown in the second section and based on the first section where we analyse the structure of the set of traces which are critical for confluence. Our result roughly says that we can decide the confluence of a noetherian system by inspecting the set of traces which are generated by two left-hand sides and where no third rule appears. This is an analogue of the critical pair criterion of Winkler-Buchberger for term rewriting systems. (It is not exactly a special case of this criterion since trace replacement systems are term rewriting systems modulo an equivalence relation.) Our main contribution is to show that the critical set of traces is effectively recognizable. Therefore we have an effective procedure to decide whether this set is finite and if it is finite then we can test confluence on this finite set. This set is, of course, finite for semi-Thue systems or vector replacement systems. It is also finite for systems which are called coherent and convergent in [Ott89]. Thus, the decidability result of [Ott89] can be viewed as a special case of our situation. It is also possible to combine our result with the condition A2) of [Die87] to obtain a even smaller set of critical traces. In the third section we present a new algorithm for computing irreducible normal forms. This algorithm terminates in all cases in time square to the number of derivation steps and its worst-case behaviour is of that complexity for certain systems. However, the interesting point is that whenever the system satisfies the condition of section two, then the same algorithm realizes the relatively linear time complexity. The existence of such a uniform algorithm was not known before and its implementation depends essentially on the Zielonka's theory of asynchronous automata. This gives further evidence for the importance of the notion of asynchronous automata.

## 2 Preliminaries

Throughout this paper $X$ means a fixed finite alphabet with independence relation $I \subseteq X \times X$ which for technical reasons is assumed to be reflexive and symmetric. The comple-

ment of $I$ is denoted by $D = X \times X \backslash I$ and called the dependence relation. We use $M$ to denote the associated free partially commutative monoid $M = X^*/\{ab = ba \mid (a,b) \in I\}$, the elements of $M$ are called *traces*. Each trace $t \in M$ is identified with a labelled partially ordered set as usual: the empty trace $t = 1$ is identified with the empty set, if $t \in M$ is a trace and $a \in X$ is a letter then $ta$ is the disjoint union of $t$ and a new point labelled with $a$. The partial order $\leq$ of $ta$ is induced by $t$ and the requirement that the new point is behind every point of $t$ which has a label depending on $a$. If $t \in M$ is a trace and $a \in X$ is a letter then, by abuse of language, we shall write $a \in t$ if we mean a fixed point of $t$ with label $a$. Similary we proceed with subsets of $t$. Note that every subset $l \subseteq t$ defines a unique trace $l \in M$. A subset $l \subseteq t$ of a trace $t \in M$ is called a *subtrace* if for all $x, y, z \in t$ with $x \leq y \leq z$ and $x, z \in l$ we have $y \in l$. If $l \subseteq t$ is a subtrace then we can write $t = ulv$ for some $u, v \in M$ and vice versa: if we have any factorization $t = ulv$ then the factor $l$ defines a unique subtrace $l \subseteq t$. For any subset $l \subseteq t$ we define the *generated subtrace* of $l$ in $t$ by the smallest subtrace of $t$ containing $l$. We denote this subtrace by $\langle l \rangle$. We can also define the generated subtrace by $\langle l \rangle = \{y \in t \mid \exists x, z \in l : x \leq y \leq z\}$. With a subtrace $l \subseteq t$ we associate the following subtraces:

$$\begin{aligned} \mathrm{pre}(l) &= \{x \in t \backslash l \mid x \leq y \text{ for some } y \in l\} \text{ of elements before } l, \\ \mathrm{suf}(l) &= \{z \in t \backslash l \mid y \leq z \text{ for some } y \in l\} \text{ of elements behind } l, \text{ and} \\ \mathrm{ind}(l) &= t \backslash (l \cup \mathrm{pre}(l) \cup \mathrm{suf}(l)) \text{ of elements which are independent of } l. \end{aligned}$$

The following simple observation is important: let $l \subseteq t$ be a subtrace then a factorization $t = ulv$ defines this subtrace $l \subseteq t$ if and only if we have equations $u = \mathrm{pre}(l)u_1$, $v = v_1 \,\mathrm{suf}(l)$ for some $u_1 v_1 = \mathrm{ind}(l)$.

Two subtraces $l_1 \subseteq t$, $l_2 \subseteq t$ are called *strictly separated* if $l_i \subseteq (\mathrm{pre}(l_j) \cup \mathrm{ind}(l_j))$ for some $\{i, j\} = \{1, 2\}$. Thus, $l_1 \subseteq t$, $l_2 \subseteq t$ are strictly separated if and only if we can write $t = ul_i v l_j w$ for some $u, v, w \in M$ and $\{i, j\} = \{1, 2\}$. (In previous papers we said that $l_1 \subseteq t$, $l_2 \subseteq t$ are not in mixed order.)

For a trace $t \in M$ its length is denoted by $|t|$ and its alphabet by $\mathrm{alph}(t)$. The independence relation $I$ is extended to $I \subseteq M \times M$ by setting $(t, t') \in I$ if $\mathrm{alph}(t) \times \mathrm{alph}(t') \subseteq I \subseteq X \times X$. For a trace $t \in M$ the trace $\min(t)$ ($\max(t)$ respectively) is defined by the set of minimal (maximal respectively) elements of the labelled partial order $t$.

A trace replacement system is a subset $S \subseteq M \times M$. Rules $(l, r) \in S$ are also written in the form $l \Rightarrow r$. A system $S \subseteq M \times M$ defines a reduction relation $\underset{S}{\Longrightarrow}$ on traces by $t \underset{S}{\Longrightarrow} t'$ if $t = ulv$, $t' = urv$ for some $u, v \in M$, $(l, r) \in S$. By $\underset{S}{\overset{*}{\Longrightarrow}}$ ($\underset{S}{\overset{*}{\Longleftrightarrow}}$ respectively) we mean the reflexive, transitive (, and symmetric respectively) closure of the relation $\underset{S}{\Longrightarrow}$. By $\mathrm{Irr}(S)$ we denote the set of irreducible traces. The word problem of $S$ is to decide on input traces $t, t' \in M$ whether or not $t \underset{S}{\overset{*}{\Longleftrightarrow}} t'$ holds. For time complexities we view the replacement system $S$ as fixed, i.e., we measure the non-uniform word problem where the input size is given by the length of the traces $t$ and $t'$.

A trace replacement system $S \subseteq M \times M$ is called noetherian if there are no infinite derivation chains $t_0 \underset{S}{\Longrightarrow} t_1 \underset{S}{\Longrightarrow} \ldots$, and confluent if for all $t_1 \underset{S}{\overset{*}{\Longleftarrow}} t \underset{S}{\overset{*}{\Longrightarrow}} t_2$ there exists a trace $\hat{t}$ such that $t_1 \underset{S}{\overset{*}{\Longrightarrow}} \hat{t} \underset{S}{\overset{*}{\Longleftarrow}} t_2$. If a finite system $S \subseteq M \times M$ is noetherian and confluent then the word problem of $S$ is decidable. But, of course, without further restrictions

the time complexity may become arbitrary high. (This follows since the computation of a deterministic always halting Turing machine can be simulated by a noetherian and confluent semi-Thue system.)

# 3 A sufficient decidable condition for testing confluence

For deciding the word problem of a given trace replacement system it is no restriction to assume that the system is noetherian. We simply direct the rules such that the system becomes noetherian. This is possible since every free partially commutative monoid has a so-called admissible well-ordering, see [Die87, Prop. 1.1]. However, one of the main problems is dealing with trace replacement systems is that the confluence may be undecidable even for finite length-reducing systems, see [NO88]. The general attempt to overcome this difficulty is to find decidable sufficient conditions which guarantee that the confluence of the system can be tested on an effectively calculable finite subset of the monoid. We give such a condition below which is weaker than the ones known before, [Die87], [BL87], [Ott89].

Before we state this condition let us recall another problem which one meets in the replacement of traces. (This problem is not present in the semi-Thue or vectorreplacement case.) Let $(l,r) \in S$ be the rule and $t \underset{(l,r)}{\Longrightarrow} t'$ be a reduction step. Then the result $t'$ is not uniquely by the subtrace $l \subseteq t$ which is replaced and by the rule $(l,r)$. It may depend on the explicit factorization $t = ulv$. Indeed, let $a \in X$ be a letter which is independent of $l$, then there is a unique subtrace $l \subseteq t$ in the trace $t = al = la$ but we have $ar \underset{(l,r)}{\Longleftarrow} t \underset{(l,r)}{\Longrightarrow} ra$. This observation led us in previous papers [Die87], [Die89] to the assumption, called A1), that $a$ and $r$ should commute in all these cases. However, for deciding confluence this assumption is not really necessary. It is enough if the system is confluent on all these pairs $(ar, ra)$. The first lemma is obvious and simply states that the confluence of these pairs is decidable.

**Lemma 3.1** *Let $S \subseteq M \times M$ be a finite noetherian trace replacement system, $(l,r) \in S$ be a rule, and $a \in X$ be a letter such that $(a,l) \in I$. Then it is decidable whether the pair $(ar, ra)$ is confluent.* □

The following considerations are based on a certain partial ordering $\prec$ of $M$ which is canonically associated with any noetherian trace replacement system $S \subseteq M \times M$. For $x, y \in M$ we put $x \preceq y$ if $y \overset{*}{\underset{S}{\Longrightarrow}} uxv$ for some $u, v \in M$, i.e., $x$ is "smaller" than $y$ if and only if $x$ is a subtrace of some descendant of $y$. As usual $x \prec y$ means $x \preceq y$ and $x \neq y$. It is an easy exercise to see that, since $S$ is noetherian, this defines a well-founded ordering of $M$. This means that every non-empty subset of $M$ has minimal elements with respect to $\prec$. The reason that we do not need the assumption A1) here results from the next lemma.

**Lemma 3.2** *Let $S \subseteq M \times M$ be a noetherian trace replacement system, $t \in M$ be a trace, $l \subseteq t$ be a subtrace and $(l,r) \in S$ be rule. Let $u_1 v_1 = u_2 v_2$ be two factorizations*

*of* $\mathrm{ind}(l)$ *and* $t_i = \mathrm{pre}(l)u_irv_i\,\mathrm{suf}(l)$ *for* $i = 1, 2$, *i.e.,* $t_1 \underset{(l,r)}{\Longleftarrow} t \underset{(l,r)}{\Longrightarrow} t_2$. *Then the following implication holds. If the pair* $(ar, ra)$ *is confluent for all* $a \in X$ *such that* $(a, l) \in I$ *and if the system* $S$ *is confluent on all traces* $t' \in M$ *such that* $t' \prec t$ *then the pair* $(t_1, t_2)$ *is confluent, too.*

**Proof:** For simplification of notation it is convenient to observe first that we may assume $\mathrm{pre}(l) = \mathrm{suf}(l) = 1$. Thus, we have $t = \mathrm{ind}(l)l$ and $t_i = u_irv_i$ for $i = 1, 2$. If we have $|u_1u_2| = 0$ then $t_1 = t_2$ and the claim follows. If $|u_1u_2| > 0$ then we may assume $u_1 = ua$ for some $u \in M, a \in X$ with $(a, l) \in I$. Since $(ar, ra)$ is confluent, the pair $(t_1, urav_1)$ is confluent, too. The pair $(urav_1, t_2)$ is confluent by induction since $|uu_2| < |u_1u_2|$. The lemma follows since $urav_1 \prec t$. □

For a noetherian trace replacement system $S \subseteq M \times M$ and rules $(l_1, r_1), (l_2, r_2) \in S$ let us define the *set of critical traces* $CT(l_1, l_2, S)$ by the set of traces $t \in M$ satisfying the following two conditions:

1) The left-hand sides $l_1, l_2$ are subtraces $l_1 \subseteq t, l_2 \subseteq t$ such that $\langle l_1 \cup l_2 \rangle = t$ and $l_1 \subseteq t, l_2 \subseteq t$ are not strictly separated.

2) For all subtraces $l \subseteq t$ such that $(l, r) \in S$ for some $r \in M$ we have for $i = 1$ or for $i = 2$ that $\langle l_i \cup l \rangle = t$ and $l_i \subseteq t, l \subseteq t$ are not strictly separated.

To illustrate the notion of $CT(l_1, l_2, S)$ consider the semi-Thue case $S \subseteq X^* \times X^*$, where for further simplification $S$ is assumed to be normalized, thus every left-hand side of $S$ is irreducible with respect to all other rules. Then a word $w \in X^*$ belongs to $CT(l_1, l_2, S)$ if and only if $w = l_1v = ul_2, l_1, l_2$ have over-lapping, and there is no third left-hand-side occuring in $w$. It is the last property which will become important.

The following theorem shows that the decidability of confluence can be based on these sets $CT(l_1, l_2, S)$. It can be viewed as an analogue to the Winkler-Buchberger criterion for term rewriting systems [WB83], see also [KMN88, section 4,(C1)] and [BD88] for a rather general treatment of critical pair criteria.

**Theorem 3.3** *Let* $S \subseteq M \times M$ *be a noetherian trace replacement system. Then the system* $S$ *is confluent if and only if the following two assertions hold:*

*i) The pair* $(ar, ra)$ *is confluent for all* $(l, r) \in S, a \in X$ *such that* $(a, l) \in I$

*ii) For all* $(l_1, r_1), (l_2, r_2) \in S$ *the system* $S$ *is confluent on the set* $CT(l_1, l_2, S)$ *of critical traces defined above.*

**Remark 3.4** Before we prove the theorem observe that, in general ii) does not imply i), unless $M$ is free or commutative (in which case it does for trivial reasons). Indeed if $M$ is neither free nor commutative then there are three different letters $a, b, c \in X$ such that $(a, c) \in I$ and $(b, c) \in D$. Consider the one-rule system $S = \{a \Longrightarrow b^2\}$. Then $CT(a, a, S) = \{a\}$ and the system is confluent on $\{a\}$ although $(b^2c, cb^2)$ is not confluent.

**Proof of Theorem 3.3:** Since the only-if-part is trivial, it is enough to show that if i) and ii) hold then $S$ is confluent. Let $(l_1, r_1), (l_2, r_2) \in S$ be rules, $t \in M$ be a trace with subtraces $l_1 \subseteq t, l_2 \subseteq t$, and $t_1 \underset{(l_1,r_1)}{\Longleftarrow} t \underset{(l_2,r_2)}{\Longrightarrow} t_2$. We shall prove that $(t_1, t_2)$ is confluent. By noetherian induction we may assume that $S$ is confluent on all traces $t' \in M$ such that $t' \prec t$. Clearly, we may also assume that $t$ does not belong to $CT(l_1, l_2, S)$. Hence, for some rule $(l, r) \in S$ we find a subtrace $l \subseteq t$ such that for $i = 1$ and $i = 2$ we have that $\langle l_i \cup l \rangle \neq t$ or $l_i \subseteq t, l \subseteq t$ are strictly separated. For $i = 1, 2$ we choose any $t_i' \underset{(l_i,r_i)}{\Longleftarrow} t \underset{(l,r)}{\Longrightarrow} t_i''$ such that $(t_i', t_i'')$ is confluent. This is possible: if $l_i \subseteq t, l \subseteq t$ are strictly separated then we may write $t = ul_ivlw$ ($t = ulvl_iw$ respectively) and the pair $(ur_ivlw, ul_ivrw)$ ($(ulvr_iw, urvl_iw)$ respectively) will do, if $\langle l_i \cup l \rangle \neq t$ there exists such a pair since $\langle l_i \cup l \rangle \prec t$. By standard techniques we are reduced to show the confluence of the following pairs $(t_1, t_1'), (t_1', t_1''), (t_1'', t_2''), (t_2'', t_2'), (t_2', t_2)$. The confluence of $(t_1', t_1''), (t_2', t_2'')$ is known by construction. The confluence of the other three pairs follows by Lemma 3.2. □

The key to our decidability result below is the fact that the sets $CT(l_1, l_2, S)$ are effectively calculable recognizable trace languages. To see this we introduce sets $B(p, q, Y)$ which roughly stands for the set of possible traces between $p$ and $q$ with alphabet $Y$. Formally $B(p, q, Y)$ is defined for traces $p, q \in M$ and subsets $Y \subseteq X$ by

$$B(p,q,Y) = \{y \in M \mid \begin{array}{l} \mathrm{alph}(y) = Y \text{ and with respect to} \\ \text{the trace } pyq \text{ it holds} \\ \mathrm{suf}(p) = yq \text{ and } \mathrm{pre}(q) = py\} \end{array}$$

Note that in the free commutative case $M = \mathbf{N}^X$ the set $B(p, q, Y)$ will be empty unless $Y \subseteq \mathrm{alph}(p) = \mathrm{alph}(q)$. More generally, $B(p, q, Y)$ is empty unless for each maximal element $a \in p$ there exists some $b \in Y \cup \mathrm{alph}(q)$ with $(a, b) \in D$ and for each minimal element $c \in q$ there exists $b \in Y \cup \mathrm{alph}(p)$ with $(b, c) \in D$. If $B(p, q, Y)$ is non-empty then it contains those $y \in M$ with $\mathrm{alph}(y) = Y$ such that every minimal letter of $y$ depends on some letter in $p$ and every maximal letter of $y$ depends on some letter in $q$. Thus, in all cases $B(p, q, Y)$ is recognizable.

**Theorem 3.5** *Let $S \subseteq M \times M$ be a noetherian trace replacement system such that the set of left hand sides is recognizable. Then for each $(l_1, r_1), (l_2, r_2) \in S$ the set $CT(l_1, l_2, S)$ is an effectively calculable recognizable subset.*

**Proof:** Let $t \in M$ such that $t \in CT(l_1, l_2, S)$. Then $l_1 \subseteq t$, $l_2 \subseteq t$ are non-strictly separated subtraces and we have $t = \langle l_1 \cup l_2 \rangle$. Define the following nine subtraces of $t$:

$$\begin{array}{rcll} p_i & = & l_i \cap \mathrm{pre}(l_j) & , \{i, j\} = \{1, 2\}, \\ s_i & = & l_i \cap \mathrm{ind}(l_j) & , \{i, j\} = \{1, 2\}, \\ q_i & = & l_i \cap \mathrm{suf}(l_j) & , \{i, j\} = \{1, 2\}, \\ s & = & l_1 \cap l_2, & \\ y_1 & = & \mathrm{suf}(l_1) \cap \mathrm{pre}(l_2) & \\ y_2 & = & \mathrm{suf}(l_2) \cap \mathrm{pre}(l_1) & \end{array}$$

We have a picture as in Figure 1.

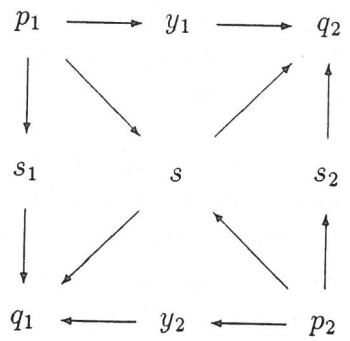


Figure 1: A trace $t = \langle l_1 \cup l_2 \rangle$ devided into nine subtraces.

The following formulae hold:

1) $p_i s_i s q_i = l_i$ for $i = 1, 2$,

2) $(s_i, l_j) \in I$ for $\{i, j\} = \{1, 2\}$

3) $(p_1, p_2) \in I$, $(q_1, q_2) \in I$,

4) $y_i \in B(p_i, q_j, Y_i)$ for some $Y_i \subseteq \{a \in X \mid (a, p_j s_1 s s_2 q_i) \in I\}$, $\{i, j\} = \{1, 2\}$, $Y_1 \times Y_2 \subseteq I$

5) $s \neq 1$ or $p_1, p_2, q_1, q_2 \neq 1$.

Vice versa, if the formulae 1) - 5) hold for some $p_i$, $s_i$, $s$, $q_i$, $y_i$ and $i = 1, 2$ then $t = p_1 p_2 y_1 s_1 s s_2 y_2 q_1 q_2$ yields a trace with non-strictly separated subtraces $l_1 \subseteq t$, $l_2 \subseteq t$ such that $\langle l_1 \cup l_2 \rangle = t$. Thus, the set $CT(l_1, l_2, S)$ is a subset of a finite union of recognizable sets of the form

$$p_1 p_2 B(p_1, q_2, Y_1) s_1 s s_2 B(p_2, q_1, Y_2) q_1 q_2.$$

In the following we may think that the data $p_1, p_2, s_1, s, s_2, q_1, q_2 \in M$ and $Y_1, Y_2 \subseteq X$ are fixed. There are only finitely many traces where $p_1 s_1 q_1 = 1$ or $p_2 s_2 q_2 = 1$, thus we assume $p_1 s_1 q_1 \neq 1 \neq p_2 s_2 q_2$. In the next step one replaces $B(p_i, q_j, Y_i)$ by $B_i = \{y \in B(p_i, q_i, Y_i) \mid p_i s_i y, y s_j q_j \in \mathrm{Irr}(S)\}$ for $\{i, j\} = \{1, 2\}$. This is possible without loosing anything from $CT(l_1, l_2, S)$. In fact, say $p_1 s_1 y_1$ is reducible by some rule $(l, r) \in S$. Then the subtrace $l \subseteq p_1 s_1 y_1 \subset t$ is strictly separated from $l_2 \subseteq t$ and $\langle l_1 \cup l \rangle \neq t$ since $p_2 s_2 q_2 \neq 1$. Note that $B = p_1 p_2 B_1 s_1 s s_2 B_2 q_1 q_2$ is recognizable. But it is still too large. It may contain traces $t$ such that $l \subseteq t$ is a subtrace for some left-hand side where $\langle l_1 \cup l_2 \rangle \neq t \neq \langle l_2 \cup l \rangle$. It is not very difficult to exclude these traces, too, by distinguishing several cases. This is left to the reader, since in our application of the next section the set $B$ is already finite. □

We now state the main result of this section which follows directly from the theorem above together with Lemma 3.1.

**Corollary 3.6** *Let $S$ be a finite noetherian trace replacement system. Then it is decidable whether the set $CT(S) = \bigcup\{CT(l_1, l_2, S) \mid (l_i, r_i) \in S,\ i = 1, 2\}$ is finite. If the set $CT(S)$ is finite then it is decidable whether the system $S$ is confluent.* □

**Remark 3.7** i) The exact calculation of the set $CT(S)$ above seems to be very difficult in general. However, in order to prove that $CT(S)$ is finite it is enough to prove an upper bound on this set. For example we might prove that the length of traces in $CT(S)$ cannot exceed a certain length. Then we can test confluence on all traces up to this length without knowing the explicit description of $CT(S)$.
ii) The reader might convince himself that it is possible to combine the corollary above with [Die87, Thm. 3.1]. Then we obtain an even weaker condition for the decidability of confluence. Since this is rather technical but not very difficult we have not shown it here.
□

# 4 The condition $G_k$ for $k \geq 0$

We are going to measure the time complexity to compute irreducible forms for finite noetherian systems $S \subseteq M \times M$ in terms of the following function

$$d_S : \mathbf{N} \to \mathbf{N}, \quad d_S(n) = \max\{m \in \mathbf{N} \mid t \overset{m}{\underset{S}{\Longrightarrow}} \hat{t},\ |t| = n\}$$

This means $d_S(n)$ is the maximal number of possible reduction steps starting on a trace of length $n$. For applications, one is mainly interested in cases where $d_S$ grows slowly. This is for example the case when $S \subseteq M \times M$ is weight-reducing, then $d_S$ is a linear function. It is easy to see that irreducible normal forms can be computed (on some multi-tape Turing machine) in time $O(d_S^2)$. But we have no idea whether this bound is optimal, and for semi-Thue systems we can achieve a time bound $O(d_S)$, see [Boo82]. In [Die89] we exhibited a sufficient decidable condition for trace replacement systems $S \subseteq M \times M$ such that irreducible normal forms can be computed by a very simple algorithm in time $O(d_S)$. The condition of [Die89] is equivalent to condition $G_0(S)$ below. We left open the question whether confluence is decidable when $G_0(S)$ holds. We will see here that the question has a positive answer. We present in fact a slightly weaker condition.

The informal reason why, so far, we can not prove a better time bound than $O(d_S^2)$ is that for some $c > 0$ there might be irreducible traces $t \in \mathrm{Irr}(S)$ such that if we multiply $t$ by a letter $a \in X$ from the left (or right) then $at$ (or $ta$) becomes reducible by some rule $(l, r) \in S$, but for all factorizations $at = ulv$, we have $|u| \geq c|t|$ and $|v| \geq c|t|$. Even if there would be, at this stage, a fast way (constant time) to compute the reduction step $at = ulv \underset{S}{\Longrightarrow} urv$, we see no fast way to test whether $urv$ is irreducible, it will take time linear to the length of $|t|$. Of course, all these problems vanish if we could bound the length of $|u|$ in the situation above by some constant $k \geq 0$ depending on $S$ only. This is exactly what the following condition says.

**Definition:** *Let $k \geq 0$, $X^{(k)} = \{t \in M \mid |t| \leq k\}$, and $S \subseteq M \times M$ be a noetherian trace replacement system with set of left-hand sides $L = \{l \in M \mid (l, r) \in S$ for some $r \in M\}$.*

*We say that the condition $G_k(S)$ holds if we have*

$$X\,\mathrm{Irr}(S) \subseteq \mathrm{Irr}(S) \cup X^{(k)}LM.$$

**Theorem 4.1** *Let $k \geq 0$ and $S \subseteq M \times M$ be a finite noetherian trace replacement system and $L$ be the set of left-hand sides. Then we have the following assertions.*

*i) It is decidable whether $G_k(S)$ holds.*

*ii) If $G_k(S)$ holds then we can decide whether $S$ is confluent.*

*iii) If $G_k(S)$ holds and if $S$ is confluent then we can decide the word problem of $S$ in time $O(d_S)$.*

**Proof:** i) trivial since all sets involved for deciding $G_k(S)$ are recognizable.
iii) Follows easily by an obvious modification, which depends on $k$, of the very simple algorithm right-reduce presented in [Die89]. Details are omitted since the new (but more complicated) algorithm of the next section yields the same time bound up to constants.
ii) We show that the set $CT(l_1, l_2, S)$ introduced in the previous section is finite for all $l_1, l_2 \in L$. The result then follows by Corollary 3.6.

Let $l_1, l_2 \in L$ and $t \in M$ be a trace with subtraces $l_1 \subseteq t$, $l_2 \subseteq t$ such that $\langle l_1 \cup l_2 \rangle = t$ and $l_1 \subseteq t$, $l_2 \subseteq t$ are not strictly separated. As in the proof of Theorem 3.5 we devide $t$ into nine subtraces:

$$\begin{aligned} p_i &= l_i \cap \mathrm{pre}(l_j) \quad , \quad \{i,j\} = \{1,2\}, \\ s_i &= l_i \cap \mathrm{ind}(l_j) \quad , \quad \{i,j\} = \{1,2\}, \\ q_i &= l_i \cap \mathrm{suf}(l_j) \quad , \quad \{i,j\} = \{1,2\}, \\ s &= l_1 \cap l_2, \\ y_i &= \mathrm{suf}(l_i) \cap \mathrm{pre}(l_j) \quad , \quad \{i,j\} = \{1,2\}. \end{aligned}$$

Then we have $t = p_1 p_2 y_1 s_1 s s_2 q_1 q_2$.

By symmetry we may assume that $|y_1| \geq |y_2|$ and it will be enough to show that if $t \in CT(l_1, l_2, S)$ then the length of $y_1$ is bounded by some constant not depending on $t$. Let $m = \max\{|l| \mid l \in L\}$ and assume that $|y_1| \geq |p_2 s| + k + m$. (Note that $|p_2 s|$ is bounded by $|l_2|$ and this bound is independent of $|t|$).

Now, if the subtrace $y_1 s_2 q_2 \subseteq t$ is reducible by some rule $(l, r) \in S$ then $t$ contains a subtrace $l \subseteq t$ such that $l_1 \subseteq t$, $l \subseteq t$ are strictly separated and $\langle l_2 \cup l \rangle \neq t$. (Note that $p_1 \neq 1$ since $|y_1| \geq 1$). Hence, we have $t \notin CT(l_1, l_2, S)$ in this case. Therefore we may assume $y_1 s_2 q_2 \in \mathrm{Irr}(S)$. On the other hand, $p_2 s y_1 s_2 q_2 = y_1 l_2$ is reducible since $l_2 \in L$. Now, it follows from $G_k(S)$ that $p_2 s y_1 s_2 q_2 = ulv$ for some $(l, r) \in S$, $u, v \in M$ with $|u| \leq |p_2 s| + k - 1$. Since $p_2 s y_1 s_2 q_2 = y_1 l_2$ is a subtrace of $t$, we may identify $u$, $l$, $v$ with subtraces of $t$, too. Since $|y_1| \geq |p_2 s| + k + m$, at least one letter of the subtrace $y_1 \subseteq t$ belongs to $v \subseteq t$, hence $\langle l_1 \cup l \rangle \neq t$. Since $p_1 \neq 1$ we also have $\langle l_2 \cup l \rangle \neq t$, thus we have $t \notin CT(l_1, l_2, S)$. The theorem follows. □

**Open problem** Is it decidable whether for given finite $S \subseteq M \times M$ there exists $k \geq 0$ such that $G_k(S)$ holds? This question would have an affirmative answer if we could decide

for given recognizable trace languages $A, B \subseteq M$ with $A \subseteq MB$ whether there exists a finite set $F \subseteq M$ such that $A \subseteq FB$. This seems to be an interesting question for recognizable trace languages independent of trace rewriting. It can be solved for regular word languages.

The reason to consider the condition $G_k(S)$ for different $k \geq 0$ follows from the next proposition:

**Proposition 4.2** *Let $k \geq 0$ and $M$ be neither free nor commutative then there exists a length-reducing trace replacement system $S \subseteq M \times M$ of at most two rules such that $G_{k+1}(S)$ holds but not $G_k(S)$.*

**Proof:** Since $M$ is neither free nor commutative we find three different letters $a, b, c \in X$ such that $(a, b) \in D$ and $(a, c) \in I$. Consider the system $S = \{cb \Longrightarrow 1, a^{k+2} \Longrightarrow 1\}$. Then $t = ca^{k+1}b \in X \operatorname{Irr}(S)$ but neither $t = a^{k+1}cb \in \operatorname{Irr}(S)$ nor $a^{k+1}cb \in X^{(k)}\{cb, a^{k+2}\}$. Hence $G_k(S)$ does not hold whereas it is easy to see that $G_{k+1}(S)$ is true. □

**Remark 4.3** i) If $M$ is free or commutative then $G_0(S)$ holds for every system $S \subseteq M \times M$. For one-rule systems $G_k(S)$ implies $G_0(S)$ for any $k \geq 0$. Therefore, the proposition above is tight. In the terminology of [Die89], the property $G_0(S)$ for a one-rule system $S = \{(l, r)\}$ is equivalent with the property that $l$ is a cone or a block.
ii) In [Ott89] another decidable and sufficient condition is given such that the confluence of finite noetherian trace replacement systems becomes decidable. The approach of Otto is based on the notion of convergence and coherence for term rewriting systems which was developped by Jouannaud [Jou83]. Inspecting Otto's condition it turns out to be equivalent with confluence in our sense and $G_0(S)$. Since $G_0(S)$ implies $G_k(S)$ for all $k \geq 0$ and any $G_k(S)$ implies the condition given in Corollary 3.6, but non of these implications is reversible, our condition is clearly weaker. Furthermore our approach has the advantage of staying entirely in the theory of traces.

# 5 An efficient algorithm for computing irreducible normal forms

In this section we present an algorithm which computes always irreducible normal forms in time $O(d_S^2)$ but which has the property that whenever the system satisfies $G_k(S)$ for some $k \geq 0$ then it works in time $O(d_S)$. From this viewpoint it is the best known algorithm in this area. The implementation of the algorithm depends essentially on the existence of finite asynchronous automata which were introduced in [Zie87]. The proof that every recognizable trace language is accepted by such automaton seems to be one of the most difficult in the field of trace theory. Unfortunately, the constants which are obtained constructing these automata are extremely high. So, in practice it might be necessary to work with "less optimal" algorithms. But may be a better understanding of asynchronous automata will change the situation.

The algorithm we are going to construct is based on a notion of protocols which is available for asynchronous automata, but not for usual finite $M$-automata. In fact, we

shall use a minor modification of asynchronous automata which we will call asynchronous cellular [1]. This modification is not important, it is done here to have smaller state sets.

A finite $M$-automaton $U = (Z, \delta, q_0, F)$ (where $Z$ denotes the finite state set, $\delta : Z \times M \to Z$ is the (partially defined) transition mapping, $q_0 \in Z$ is the initial state, and $F \subseteq Z$ is the set of final states) is called *asynchronous cellular* if the following two conditions hold:

1) The state set $Z$ is a cartesian product $Z = \prod_{x \in X} Z_x$

2) The partially defined transition mapping $\delta$ is given by a collection of partial mappings

$$\{\delta_a : (\prod_{b \in D(a)} Z_b) \to Z_a \mid a \in X\}$$

where $D(a) = \{b \in X \mid (a, b) \in D\}$ for $a \in X$.

3) For all traces $t \in M$ the state $\delta(q_0, t)$ is defined.

Condition 2) means that for $a \in X$, $(z_x)_{x \in X} \in \prod_{x \in X} Z_x$ the state $\delta((z_x)_{x \in X}, a)$ is defined if and only if $\delta_a((z_b)_{b \in D(a)})$ is defined. In this case we have $\delta((z_x)_{x \in X}, a)_y = z_y$ for $y \neq a$ and $\delta((z_x)_{x \in X}, a)_a = \delta_a((z_b)_{b \in D(a)})$.

Condition 3) is included for technical reasons only.

The deep theorem of Zielonka [Zie87] can be read as follows: For every recognizable trace language $L \subseteq M$ there exists effectively a finite asynchronous cellular automaton which recognizes $L$. In fact, in the known proofs of Zielonka's theorem [Zie87], or [CM87], (implicitely) a asynchronous cellular automaton is constructed first and the asynchronous automaton in the sense of [Zie87] is obtained simply by blowing-up the state space. We avoid this (unnecessary) blow-up. (The reader can also translate the following construction to the "usual" asynchronous automata).

Let $U = (\prod_{a \in X} Z_a, \delta, q_0, F)$ be a asynchronous cellular automaton. Protocols of $U$ are elements of the product space $\prod_{a \in X}(Z_a^+)$ which are inductively defined as follows:

The element $q_0 \in \prod_{x \in X} Z_x \subseteq \prod_{x \in X}(Z_x^+)$ is a protocol. If $p \in \prod_{x \in X}(Z_x^+)$ is a protocol and $a \in X$ is a letter then the protocol $ap$ is defined as follows: Take $q \in \prod_{x \in X} Z_x$ such that $p = qp'$ for some $p' \in \prod_{x \in X} Z_x^*$. Then compute $\delta(q, a)_a \in Z_a$ and multiply this state from the left to the $a$-component of $p$, the other components are unchanged. The reason that we build up protocols from right-to-left is that we view protocols contained in a stack and we follow the convention that the top of a stack is on the left-hand side. It follows from the definition that if $t \in M$ is a trace then $p = tq_0$ denotes a well-defined protocol with $p \in \prod_{x \in X} Z_x^+$. The crucial point is that if $p = tq_0$ is a protocol and $t = at'$ for some $t' \in M$, $a \in X$ then the protocol $t'q_0$ can be computed from $tq_0$ in constant time by erasing the left most state in the $a$-component of the protocol $tq_0$. More generally, if $p = tq_0$ and $t = uv$ then we may compute the protocol $vq_0$ starting from $p$ in $|u|$-steps. We also shall write $u^{-1}p$ to denote the protocol $vq_0$ if $p = uvq_0$. This will also be done for traces: if $t = uv$ then $u^{-1}t$ denotes the trace $v$.

[1] In a previous version of this paper these automata were called "uniform". The new notation is due to W. Zielonka who introduced this modification independently

If $U = (\prod_{x\in X} Z_x, \delta, q_0, F)$ is a finite asynchronous cellular automata then a protocol $p \in \prod_{x\in X} Z_x^+$ is called *final* if $p = zp'$ for some $z \in \prod_{x\in X} Z_x$, $p' \in \prod_{x\in X} Z_x^*$ with $z \in F$. (We could also view $\prod_{x\in X} Z_x^+$ as a state set of an infinite asynchronous cellular automata where $M$ operates on the left). We are now ready to prove the following result.

**Theorem 5.1** *There is a construction giving on input a finite noetherian trace replacement system $S \subseteq M \times M$ an algorithm* right_reduce$_\infty$ *which satisfies the following assertions:*

*i) It holds* right_reduce$_\infty(s) \in \mathrm{Irr}(S)$ *for all $s \in M$ and* right_reduce$_\infty$ *terminates in time $O(d_S^2)$.*

*ii) For some systems $S \subseteq M \times M$ the worst-case behaviour of* right_reduce$_\infty$ *is $\Theta(d_S^2)$.*

*iii) Whenever $S \subseteq M \times M$ satisfies $G_k(S)$ for some $k \geq 0$ then* right_reduce$_\infty$ *terminates in time $O(d_S)$.*

**Proof:** For $S \subseteq M \times M$ let $U = (\prod_{x\in X} Z_x, \delta, q_0, F)$ be a finite asynchronous cellular automaton which recognizes the set of reducible traces if they are read from right-to-left. A protocol means an element $p \in \prod_{x\in X} Z_x^+$ as defined above. Define the algorithm right_reduce$_\infty$ as follows

**function** right_reduce$_\infty$ (s:trace):trace
**var** t:trace:=1;
**var** p:protocol:=$q_0$;
**while** s $\neq$ 1 **do**
choose some $a \in X$ which is maximal in $s$;
$s := sa^{-1}$; $t := at$; $p := ap$;
("time: $O(1)$")
**if** p is a final protocol ("recall that $p$ is final if and only if $t$ is reducible, time: $O(1)$")
**then** compute some $u \in M$ of minimal length such that
$t = ulv$ for some $(l, r) \in S$, $v \in M$;
("It is crucial that $|u|$ is minimal and to note that this can be done in time $O(|u|)$.
Note also that we must have $l \in aM$, hence $v = (ul)^{-1}t \in \mathrm{Irr}(S)$ ")
$s := sur$; $t := (ul)^{-1}t$; $p := (ul)^{-1}p$; ("time: $O(|u|)$")
**endif**
**endwhile**
**return** t
**endfunction.**

It is easy to verify the correctness of the algorithm, i.e., right_reduce$_\infty(s) \in \mathrm{Irr}(S)$ for all $s \in M$, by the following invariants: $st$ is a descendant of the input trace, $t$ is irreducible and $p$ is the protocol $tq_0$. For the time complexity see the comments above. Two points are important: First, the "if-test" can be performed in constant time. This means we try to find a left-hand side inside (the stack) $t$ only if we know that such a left-hand side exists.

This was the only reason to work with asynchronous or asynchronous cellular automata. Second, the factorization $t = ulv$ for some $u, v \in M$, $(l, r) \in M$ with $|u|$ minimal can be performed in $O(|u|)$ steps. This can be seen, for example, from the representation of a trace as a tuple of words. Now, if the system $S$ satisfies $G_k(S)$ for some $k \geq 0$ then we will have $O(|u|) = O(1)$. This proves iii). Assertion i) is obvious since we do not enter the then-part of the while-loop if $t$ is an irreducible trace. Assertion ii) is shown in the following example. □

**Example:** Let $(X, D)$ be given by the graph $a - b - c - d$. Consider the following special trace replacement system $S = \{bc \Longrightarrow 1,\ ad \Longrightarrow 1\}$. This system is confluent and the function $d_S$ is linear. Independently of implementation details the worst-case behaviour of the algorithm right-reduce$_\infty$ above will be $\Theta(n^2)$.

Indeed, consider an input trace of the form $s = (ab)^n c^n d^n$. Of course, $s$ reduces to the empty trace. But the algorithm right-reduce$_\infty$ will perform $\Theta(n^2)$ times the while-loop. Thus, the time complexity of the algorithm can not be better than $\Theta(n^2)$ even if the whole while-loop could always be performed in constant time. □

**Acknowledgement** I would like to thank Friedrich Otto and the anonymous referees of STACS for valuable comments. Special thanks are due to Harald Hadwiger for excellent cooperation.

# References

[AR88] I.J. Aalbersberg and G. Rozenberg. Theory of traces. *Theoret. Comput. Sci.*, 60:1–82, 1988.

[BD88] L. Bachmair and N. Dershowitz. Critical pair criteria for completion. *J. Symbolic Computation*, 6:1–18, 1988.

[BL87] R. Book and H.-N. Liu. Rewriting systems and word problems in a free partially commutative monoid. *Inform. Proc. Letters*, 26:29–32, 1987.

[Boo82] R. Book. Confluent and other types of Thue systems. *J. Assoc. for Comp. Mach.*, 29:171–182, 1982.

[CF69] P. Cartier and D. Foata. *Problèmes combinatoires de commutation et réarrangements.* Number 85 in Lect. Notes in Math. Springer, 1969.

[CM87] R. Cori and Y. Métivier. Approximation d' une trace, automates asynchrones et ordre des evenement dans les systemes repartis. Technical Report 1-8708, UER de Mathematiques et d' Informatique, Université de Bordeaux I, 1987.

[Die87] V. Diekert. On the Knuth-Bendix completion for concurrent processes. In Th. Ottmann, editor, *Proc. of the 14th International Colloquium on Automata Languages and Programming, Karlsruhe 1987, (ICALP'87)*, number 267 in Lect.

Notes in Comp. Sci., pages 42–53. Springer, 1987. Appeared also in a revised version in Theoret. Comp. Science 66:117-136, 1989.

[Die89] V. Diekert. Word problems over traces which are solvable in linear time. In B. Monien et al., editors, *Proceedings of the 6th Annual Symposium on Theoretical Aspects of Computer Science (STACS'89), Paderborn 1989*, number 349 in Lect. Notes in Comp. Sci., pages 168–180. Springer, 1989. To appear in revised version in Theoret. Comp. Science.

[Jou83] J.P. Jouannaud. Confluent and coherent equational term rewriting systems applications to proofs in abstract data types. In Ausiello G. et al., editors, *Proceeding of the conference of Trees in Algebra and Programming (CAAP'83)*, number 159 in Lect. Notes in Comp. Sci., pages 269–283. Springer, 1983.

[KMN88] D. Kapur, D. Musser, and P. Narendran. Only prime superposition need be considered in the Knuth-Bendix completion procedure. *J. Symbolic Computation*, 6:19–36, 1988.

[Maz77] A. Mazurkiewicz. Concurrent program schemes and their interpretations. DAIMI Rep. PB 78, Aarhus University, Aarhus, 1977.

[Maz87] A. Mazurkiewicz. Trace theory. In W. Brauer et al., editors, *Petri Nets, Applications and Relationship to other Models of Concurrency*, number 255 in Lect. Notes in Comp. Sci., pages 279–324. Springer, 1987.

[NO88] P. Narendran and F. Otto. Preperfectness is undecidable for Thue systems containing only length-reducing rules and a single commutation rule. *Information Proc. Letters*, 29:125–130, 1988.

[Ott89] F. Otto. On deciding confluence of finite string rewriting systems modulo partial commutativity. *Theoret. Comput. Sci.*, 67:19–36, 1989.

[Per89] D. Perrin. Partial commutations. In *Proc. of the 16th International Colloquium on Automata, Languages and Programming (ICALP '89), Stresa 1989, Italy*, number 372 in Lect. Notes in Comp. Sci., pages 637–651. Springer, 1989.

[WB83] F. Winkler and B. Buchberger. A criterion for eleminating unnecessary reductions in the Knuth-Bendix algorithm. In *Proc. Coll. on Algebra, Combinatorics and Logic in Computer Science, Györ, Hungary*, 1983.

[Wra88] C. Wrathall. The word problem for free partially commutative groups. *J. Symbolic Computation*, 6:99–104, 1988.

[Zie87] W. Zielonka. Notes on finite asynchronous automata. *R.A.I.R.O.-Informatique théorique et Application*, 21:99–135, 1987.