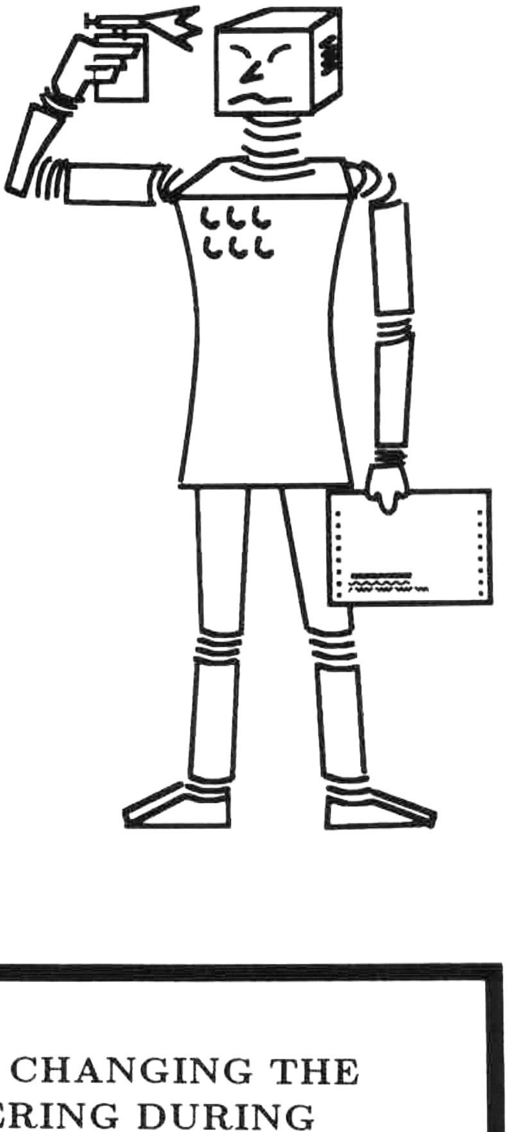
SEKI - REPORT

Fachbereich Informatik
Universität Kaiserslautern
D-67663 Kaiserslautern



ABOUT CHANGING THE ORDERING DURING KNUTH-BENDIX COMPLETION

Andrea Sattler-Klein

SEKI Report SR-93-19 (SFB)

ABOUT CHANGING THE ORDERING DURING KNUTH-BENDIX COMPLETION

Andrea Sattler-Klein
Fachbereich Informatik, Universität Kaiserslautern
D-67653 Kaiserslautern, Germany

email: sattler@informatik.uni-kl.de

Abstract. We will answer a question posed in [DJK91], and will show that Huet's completion algorithm [Hu81] becomes incomplete, i.e. it may generate a term rewriting system that is not confluent, if it is modified in a way that the reduction ordering used for completion can be changed during completion provided that the new ordering is compatible with the actual rules. In particular, we will show that this problem may not only arise if the modified completion algorithm does not terminate: Even if the algorithm terminates without failure, the generated finite noetherian term rewriting system may be non-confluent. Most existing implementations of the Knuth-Bendix algorithm provide the user with help in choosing a reduction ordering: If an unorientable equation is encountered, then the user has many options, especially, the one to orient the equation manually. The integration of this feature is based on the widespread assumption that, if equations are oriented by hand during completion and the completion process terminates with success, then the generated finite system is a maybe nonterminating but locally confluent system (see e.g. [KZ89]). Our examples will show that this assumption is not true.

1 Introduction

The Knuth-Bendix completion procedure is an important deduction tool for term rewriting systems. Given a (finite) set of equations $\mathcal{E}$ and a reduction ordering $>$ as input, the Knuth-Bendix completion procedure tries to generate a complete (confluent and terminating) term rewriting system $\mathcal{R}$ that presents the same equational theory as $\mathcal{E}$. The basic steps of the completion procedure are the computation of certain equational consequences and the generation of rewrite rules by orienting equations according to the given reduction ordering. The completion procedure may either terminate with success, i.e. it generates a finite complete term rewriting system $\mathcal{R}$ equivalent to $\mathcal{E}$, or with failure, or it may not terminate. In the latter case it computes successive approximations $\mathcal{R}_0, \mathcal{R}_1, \mathcal{R}_2, \ldots$ of an infinite complete system $\mathcal{R}$ which is equivalent to $\mathcal{E}$. If the completion procedure terminates with success, then the generated finite complete system $\mathcal{R}$ can be used to decide the word problem of $\mathcal{E}$, since then two terms are equivalent if and only if their normal forms w.r.t. $\mathcal{R}$ are the same.
Correctness of a specific version of the Knuth-Bendix completion procedure was first proved by Huet [Hu81]. In [BDH86] Bachmair et al. introduced a more abstract approach: They formalized the notion of completion within the framework of an equational inference system and introduced the notion of proof orderings for proving correctness of a completion procedure. Moreover, they proved the correctness of a large class of completion procedures. The proof of Huet as well as the one of Bachmair et al. is based essentially on the fact that all rules generated during a completion process are oriented according to the same reduction ordering $>$: Huet's proof is based mainly on noetherian induction using the reduction ordering $>$. Bachmair et al. used a proof ordering that is an extension of $>$ for their proof.

The requirement to use a fixed reduction ordering during completion guarantees that the successively generated systems $\mathcal{R}_0, \mathcal{R}_1, \mathcal{R}_2, \ldots$ are noetherian. One may wonder if a completion procedure remains correct if it is only required that the systems $\mathcal{R}_0, \mathcal{R}_1, \mathcal{R}_2, \ldots$ are noetherian, instead of requiring that the termination of these systems can be proved using the same reduction ordering. From a practical point of view this would be a desirable property, since it would allow to change the reduction ordering during completion. In general, it is not easy to choose an appropriate reduction ordering for a set $\mathcal{E}$ of equations. A completion procedure will fail if it tries to orient an equation and the corresponding terms are incomparable w.r.t. the given reduction ordering. Sometimes failure cannot be avoided, e.g. if $\mathcal{E}$ cannot be presented by a complete term rewriting system. But even if failure can be avoided, completion may fail. If an equation cannot be oriented w.r.t. the given ordering, then in many cases this problem could be circumvented by choosing another ordering. But instead restarting the completion process for $\mathcal{E}$ with a new reduction ordering, one would prefer to carry out the completion process in an incremental fashion, i.e. to continue completion with the new ordering without recomputing critical pairs between rules that have been previously considered. Which requirements are needed to ensure that completion remains correct under these modifications? Obviously, the new ordering should be compatible with the actual term rewriting system in order to guarantee that the system is terminating. Is this requirement strong enough to guarantee correctness of this procedure?
In practice, the Knuth-Bendix algorithm is usually used interactively. One reason for human interaction is to specify incrementally the reduction ordering during completion, i.e. to stepwise refine the reduction ordering given as input if needed. In current implementations of completion based methods, like for example in the system RRL [KZ89], the user cannot only

refine the actual reduction ordering during completion, but also orient equations that are not comparable w.r.t. the actual ordering by hand. This feature allows to delay testing for termination until all critical pairs have been considered as proposed e.g. in [De89]. In that case it is no longer guaranteed that the resulting system as well as the intermediately generated systems are terminating, and hence, a completion process may not terminate due to the computation of an infinite reduction sequence. Methods that can be used to detect certain kinds of non-termination in rewriting have been proposed by Plaisted [Pl86] and Purdom [Pu87]. However, what about a successful computation in case that the termination test is delayed? Is the resulting term rewriting system confluent if it is noetherian, i.e. is it locally confluent, as often implicitly used in the literature (see e.g. [De89], [Pu87]) and explicitly stated for example in [KZ89]? Of course, this is true if interreduction is not used during completion, since then a critical pair that is joinable in an intermediate system will be joinable in the resulting system as well. In practice, interreduction is essential for reasons of efficiency. But, if interreduction is used, then a rule that is used to resolve a critical pair during completion may not exist in the final system. Will the final system yet be confluent?

In this paper we will consider these questions and analyse which problems may arise if a completion algorithm is modified in the ways described. Doing this we will focus our attention on string rewriting systems. String rewriting systems can be viewed as special term rewriting systems, namely such term rewriting systems where only unary function symbols occur. Usually, in order to complete a string rewriting system, a total reduction ordering is used. Hence, in this case failure cannot arise, and a completion procedure will generate a (maybe infinite) complete string rewriting system. But also if a string rewriting system is completed, it would be desirable to have the possibility to change the ordering during completion in an incremental fashion, since in this way divergence of completion, i.e. non-termination of completion, may sometimes be avoided too [He88].

It will turn out that a lot of problems may arise, if it is allowed to change the ordering during completion, as mentioned above, even if the input is restricted to string rewriting systems. Even if interreduction is not being used, this modified completion algorithm is not correct: If the algorithm does not terminate, i.e. if it enumerates an infinite system, then this infinite system can be non-confluent. This is due to the fact that the generated system can be non-noetherian, and hence, local confluence and confluence may not coincide. Concerning interreduction we will prove the following result: If interreduction is used and a corresponding modified completion process does not terminate, then the generated system may not even be equivalent to the initial set of equations. Moreover, the corresponding modified completion algorithm is not even partially correct in the sense that the generated system will be complete whenever the algorithm terminates with success. Obviously, the generated system is noetherian in that case. But, as we will show, it does not need to be locally confluent, since rules that have been used to resolve critical pairs during completion may have been deleted by interreduction. The example that we will construct to illustrate this phenomenon is not only interesting from a theoretical point of view, because neither artificial reduction orderings nor an artificial completion strategy are used within: The example is based on only two recursive path orderings [De82] and Huet's completion algorithm [Hu81].
These results give an answer to one of the 'open problems' listed in [DJK91] (which is still open (see [DJK93])), asking for an example showing that Huet's completion algorithm [Hu81] becomes incomplete, if one allows to change the reduction ordering during completion, provided the new ordering is compatible with the actual rules. Moreover, one of our examples disproves

the widespread assumption which says that even if termination is not guaranteed during completion, local confluence will be assured as soon as all critical pairs have been considered (see e.g. [KZ89]). Hence our results may affect the correctness of existing implementations of completion that provide the option to orient equations by hand.

2 Basic Definitions and Notations

Here we recall the basic definitions and notation that we will use in the following in brief. For further reading concerning string rewriting systems we refer to [Bo87] , [BO93] and [Ja88].

Let Σ be a finite alphabet. Then Σ^* denotes the set of all strings over Σ including the empty string ε. A *string rewriting system* (SRS in short) $\mathcal{R}$ over Σ is a subset of $\Sigma^* \times \Sigma^*$. Its elements are called *(rewrite) rules* and are also written as $l \rightarrow r$ instead of (l, r). A SRS $\mathcal{R}$ induces a *one-step reduction relation* $\longrightarrow_{\mathcal{R}}$ on Σ^* which is defined in the following way: For $x, y \in \Sigma^*$, $x \longrightarrow_{\mathcal{R}} y$ if and only if there exist two strings $u, v \in \Sigma^*$ and a rule $l \rightarrow r \in \mathcal{R}$ such that $x = ulv$ and $y = urv$. The reflexive and transitive closure of $\longrightarrow_{\mathcal{R}}$ is denoted by $\xrightarrow{*}_{\mathcal{R}}$. If $x \xrightarrow{*}_{\mathcal{R}} y$, then we say that x *reduces* to y and that y is a *descendant* of x. A string x is called *irreducible* (modulo $\mathcal{R}$) if $x \longrightarrow_{\mathcal{R}} y$ does not hold for any string y. If x reduces to y and y'is irreducible then y is called a *normal form* of x.
Analogously to $\xrightarrow{*}_{\mathcal{R}}$, $\overset{*}{\leftrightarrow}_{\mathcal{R}}$ denotes the reflexive, symmetric and transitive closure of $\longrightarrow_{\mathcal{R}}$. This relation is called the *Thue-congruence* induced by $\mathcal{R}$. Two SRSs are called *equivalent* if they induce the same Thue-congruence.

Given a SRS $\mathcal{R}$ an important problem is its *word problem*, that is to decide whether or not two arbitrary strings x, y are *congruent* (modulo $\mathcal{R}$), i.e. $x \overset{*}{\leftrightarrow}_{\mathcal{R}} y$. The word problem for a SRS $\mathcal{R}$ can be undecidable even if $\mathcal{R}$ is finite. But it is decidable if $\mathcal{R}$ is finite, noetherian and confluent: Here a string rewriting system $\mathcal{R}$ is called *noetherian* if no infinite chain of the form $x_0 \longrightarrow_{\mathcal{R}} x_1 \longrightarrow_{\mathcal{R}} x_2 \longrightarrow_{\mathcal{R}} \ldots$ exists, and it is called *confluent* if, for all $x, y, z \in \Sigma^*$, the following holds: if y and z are descendants of x then they are *joinable* (i.e., y and z have a common descendant). String rewriting systems that are both, noetherian and confluent, are called *complete* or *convergent*. If a SRS $\mathcal{R}$ is complete, then each string has a unique normal form, and it holds that two arbitrary strings x and y are congruent (modulo $\mathcal{R}$) if and only if their normal forms (modulo $\mathcal{R}$) are the same. Thus, if a SRS $\mathcal{R}$ is complete and in addition finite, the word problem for $\mathcal{R}$ can simply be solved by reduction.

In [KB70] Knuth and Bendix have shown that a noetherian SRS $\mathcal{R}$ is confluent if and only if the so called *critical pairs* of $\mathcal{R}$ are joinable. Thereby a pair of strings (c_1, c_2) is called a critical pair of $\mathcal{R}$, if there exist two rules $l_1 \rightarrow r_1$ and $l_2 \rightarrow r_2$ in $\mathcal{R}$ such that one of the following conditions is satisfied: 1. $l_1 = ul_2v$ for some u, $v \in \Sigma^*$, and $c_1 = r_1$ and $c_2 = ur_2v$, 2. $l_1u = vl_2$ for some u, $v \in \Sigma^*$ with $| u |<| l_2 |$, and $c_1 = r_1v$ and $c_2 = vr_2$. Thus, it is decidable whether or not a finite and noetherian SRS $\mathcal{R}$ is confluent.
There remains the problem to decide if a finite SRS $\mathcal{R}$ is noetherian. While in general this problem is undecidable, it is possible to transform $\mathcal{R}$ into an equivalent finite and noetherian SRS $\mathcal{R}'$ according to the following idea: A SRS $\mathcal{R}$ is noetherian if and only if there exists an ordering $>$ on Σ^* that is *admissible* (i.e., $u > v$ implies $xuy > xvy$ for all $u, v, x, y \in \Sigma^*$),

wellfounded (i.e., there is no infinite descending chain $u_0 > u_1 > u_2 > \ldots$) and *compatible* with $\mathcal{R}$ (i.e., $l > r$ holds for all rules $l \rightarrow r$ of $\mathcal{R}$). Thus, given a total, wellfounded and admissible ordering $>$ on Σ^*, a finite SRS $\mathcal{R}$ can be transformed to an equivalent, finite and noetherian SRS $\mathcal{R}'$ by orienting the rules of $\mathcal{R}$ according to $>$.
If $\mathcal{R}'$ is not confluent, then there exists a critical pair (c_1, c_2) such that corresponding normal forms n_1, n_2 are not identical. Thus, by adding the rule $n_1 \rightarrow n_2$ ($n_2 \rightarrow n_1$), if $n_1 > n_2$ ($n_2 > n_1$) to $\mathcal{R}'$, we obtain an equivalent system $\mathcal{R}''$ that is noetherian too. By repeating this process for the system $\mathcal{R}''$ if $\mathcal{R}''$ is not confluent, $\mathcal{R}'$ can be transformed to a (maybe infinite) equivalent complete system. The algorithm sketched here is due to Knuth and Bendix [KB70] and known as the *Knuth-Bendix completion algorithm (for SRSs).*

One class of admissible and wellfounded orderings is the class of *syllable orderings*: Let $>$ be a total ordering on Σ called *precedence* and let for $u \in \Sigma^*$, $max(u)$ denote the largest letter with respect to the precedence $>$ that occurs in u. Then the induced *syllable ordering* $>_{syl}$ is defined as follows:

$$\begin{gathered} u >_{syl} v \\ \text{iff} \\ |\, u \,|_{max(uv)} > |\, v \,|_{max(uv)} \text{ or} \\ (max(uv) = a,\ |\, u \,|_a = |\, v \,|_a = n,\ u = u_1 a \ldots u_n a u_{n+1},\ v = v_1 a \ldots v_n a v_{n+1}, \\ \text{and } \exists i \in \{1, \ldots, n+1\} : u_i >_{syl} v_i \text{ and } u_j = v_j \text{ for all } j \in \{i+1, \ldots, n+1\}). \end{gathered}$$

This syllable ordering corresponds to the well-known recursive path ordering for monadic terms [St89].
Already in their seminal paper [KB70], Knuth and Bendix have suggested to keep all the rules as small as possible by *interreduction* during the execution of their algorithm. A SRS $\mathcal{R}$ is called *interreduced* if, for each rule $l \rightarrow r$, r is irreducible w.r.t. $\mathcal{R}$ and l is irreducible w.r.t. $\mathcal{R} - \{l \rightarrow r\}$. If $>$ is an admissible well-ordering, then there exists a unique (possibly infinite) interreduced complete system that is compatible with this ordering and equivalent to $\mathcal{R}$.

The original Knuth-Bendix completion algorithm given in [KB70] is defined for general term rewriting systems. (String rewriting systems may be viewed as special term rewriting systems, namely such term rewriting systems where only monadic terms, but no constants, occur.) Given a set of equations and a reduction ordering $>$ as input, the algorithm generates (provided that it does not stop with failure) a (maybe infinite) sequence $(\mathcal{R}_0, \mathcal{E}_0), (\mathcal{R}_1, \mathcal{E}_1), (\mathcal{R}_2, \mathcal{E}_2) \ldots$, where $\mathcal{R}_0, \mathcal{R}_1, \mathcal{R}_2, \ldots$ are term rewriting systems and $\mathcal{E}_0, \mathcal{E}_1, \mathcal{E}_2, \ldots$ are sets of equations, satisfying: 1. $\mathcal{R}_i$ is compatible with $>$, 2. $\overset{*}{\leftrightarrow}_{\mathcal{R}_i \cup \mathcal{E}_i} = \overset{*}{\leftrightarrow}_{\mathcal{R}_{i+1} \cup \mathcal{E}_{i+1}}$, 3. the set $\mathcal{E}_\infty$ of persisting equations is empty and 4. the set $\mathcal{R}_\infty$ of persisting rules (a rule is called *persistent* if for some $j \in \mathbb{N}$ it belongs to any $\mathcal{R}_k$ with $k \geq j$) is complete. In the following we will call $\mathcal{R}_\infty$ the *limit system* or the *system generated* (by the Knuth-Bendix completion algorithm).
For more information about completion of term rewriting systems we refer to the literature (see e.g. [De89]).

3 Modified Completion

In order to complete a term rewriting system a fixed reduction ordering is used. This ensures that any of the successively generated term rewriting systems $\mathcal{R}_0, \mathcal{R}_1, \mathcal{R}_2, \ldots$ is terminating.
As already pointed out above, an important question is whether or not a Knuth-Bendix completion procedure remains correct, if it is modified by requiring only that the generated systems $\mathcal{R}_i$ are noetherian. In the following we will call a corresponding algorithm *modified completion algorithm* and a corresponding process *modified completion* in short.
In the present paper we will consider this problem by studying Huet's completion algorithm, and analyse whether or not it remains correct, if it is allowed to change the reduction ordering during a completion process provided that the new ordering is compatible with the actual term rewriting system. More precisely, we will analyse the correctness of the following algorithm.

```
MODIFIED COMPLETION ALGORITHM CA_MOD1 :
Initial data: a (finite) set of equations $\mathcal{E}$, and
              a family of (recursive) reduction orderings $(>_i)_{i\in\mathbb{N}}$.
$\mathcal{E}_0 := \mathcal{E}$ ; $\mathcal{R}_0 := \emptyset$ ; $i := 0$ ; $p := 0$ ;
loop
   while $\mathcal{E}_i \neq \emptyset$ do
         Reduce equation : Select equation $M = N$ in $\mathcal{E}_i$.
         Let $M\downarrow$ (resp. $N\downarrow$) be an $\mathcal{R}_i$-normal form of $M$ (resp. $N$) obtained
         by applying rules of $\mathcal{R}_i$ in any order, until none applies.
         if $M\downarrow = N\downarrow$ then $\mathcal{E}_{i+1} := \mathcal{E}_i - \{M = N\}$ ; $\mathcal{R}_{i+1} := \mathcal{R}_i$ ; $i := i + 1$ ;
         else if $>_i$ is compatible with $\mathcal{R}_i$ then
                  if ($M\downarrow >_i N\downarrow$) or ($N\downarrow >_i M\downarrow$) then
                     begin
                     if $M\downarrow >_i N\downarrow$ then $\lambda := M\downarrow$ ; $\rho := N\downarrow$
                                      else $\lambda := N\downarrow$ ; $\rho := M\downarrow$
              (*)    Add new rule: Let $K$ be the set of labels $k$ of rules of $\mathcal{R}_i$ whose left-
              (*)    hand side $\lambda_k$ is reducible by $\lambda \to \rho$, say to $\lambda'_k$.
              (*)    $\mathcal{E}_{i+1} := \mathcal{E}_i - \{M = N\} \cup \{\lambda'_k = \rho_k \mid k : \lambda_k \to \rho_k \in \mathcal{R}_i$ with $k \in K\}$;
              (*)    $p := p + 1$ ;
              (*)    $\mathcal{R}_{i+1} := \{j : \lambda_j \to \rho'_j \mid j : \lambda_j \to \rho_j \in \mathcal{R}_i$ with $j \notin K\} \cup \{p : \lambda \to \rho\}$ ,
              (*)      where $\rho'_j$ is a normal form of $\rho_j$ , using rules from $\mathcal{R}_i \cup \{\lambda \to \rho\}$ .
                     The rules coming from $\mathcal{R}_i$ are marked or unmarked as they were in
                       $\mathcal{R}_i$, the new rule $\lambda \to \rho$ is unmarked.
                     $i := i + 1$
                     end
                  else exitloop (failure) endif
              else exitloop (failure) endif
   endwhile ;
Compute critical pairs: If all rules in $\mathcal{R}_i$ are marked, exitloop ($\mathcal{R}_i$ complete).
Otherwise, select an unmarked rule in $\mathcal{R}_i$, say with label $k$. Let $\mathcal{E}_{i+1}$ be the set of
all critical pairs computed between rule $k$ and any rule of $\mathcal{R}_i$ of label not greater
than $k$. Let $\mathcal{R}_{i+1}$ be the same as $\mathcal{R}_i$, except that rule $k$ is now marked.
$i := i + 1$
endloop
```

The starting point of this work was the following problem which is stated in the list of 'open problems' collected by Dershowitz et al. in [DJK91]:

Problem 35. *Huet's proof [Hu81] of the "completeness" of completion is predicated on the assumption that the ordering supplied to completion does not change during the process. Assume that at step i of completion, the ordering used is able to order the current rewriting relation $\rightarrow_{R_i}$, but not necessarily $\rightarrow_{R_k}$ for $k < i$ (since old rules may have been deleted by completion). Is there an example showing that completion is then incomplete (the persisting rules are not confluent)?*

Analysing the correctness of the algorithm CA_MOD1 we will particularly direct our attention to this problem.
The formulation of problem 35 points out that the use of interreduction might play an essential role in that context. Hence, the question arises whether or not the algorithm CA_MOD1 is correct, if interreduction is not used, i.e. if we replace the lines marked with $(*)$ by the following ones:

$$\begin{aligned}&\mathcal{E}_{i+1} := \mathcal{E}_i - \{M = N\};\\ &p := p + 1\ ;\\ &\mathcal{R}_{i+1} := \mathcal{R}_i \cup \{p : \lambda \rightarrow \rho\}\end{aligned}$$

In the present paper we also will analyse the correctness of this algorithm which we will denote CA_MOD2.
For both algorithms we will adopt the fairness of selection hypothesis given in [Hu81]. This hypothesis states that for every rule label k, there is an iteration i such that either the rule of label k is deleted from $\mathcal{R}_i$, or the rule of label k is selected at "compute critical pairs".
In the following we will assume, until otherwise stated, that a modified completion algorithm always is fair in that sense. Moreover, we will assume that the algorithms CA_MOD1 and CA_MOD2 use the following simple strategies:
In order to compute critical pairs, the unmarked rule with the least label is selected. Furthermore, the sets $\mathcal{E}_i$ are implemented as queues. If a rule is overlapped with a set of rules, this also will be done according to the labels of the rules, i.e. in the set of rules to be considered the rule with least label has highest priority.
In case that the algorithm CA_MOD1 (CA_MOD2) terminates with success, say with the pair $(\mathcal{R}_n, \mathcal{E}_n)$, we define for any $j > n$, $\mathcal{R}_j := \mathcal{R}_n$ and $\mathcal{E}_j := \mathcal{E}_n$.

3.1 Modified Completion without Interreduction

If interreduction is not used during completion, then the generated term rewriting systems $\mathcal{R}_0, \mathcal{R}_1, \mathcal{R}_2, \ldots$ form an increasing chain, i.e. $\mathcal{R}_0 \subseteq \mathcal{R}_1 \subseteq \mathcal{R}_2 \subseteq \ldots$ holds, and we have $\mathcal{R}_\infty = \bigcup_{i \in \mathbb{N}} \mathcal{R}_i$. Moreover, since only equational consequences are added during completion, $\mathcal{R}_\infty$ is equivalent to the input system $\mathcal{E}$.
Obviously, these properties are independent of the fact that a fixed reduction ordering is used during completion. Thus, they also hold for modified completion. In modified completion it only is allowed to change the reduction ordering if the new ordering is compatible with the actual rewrite system. Hence, it is guaranteed that any of the intermediately generated

systems $\mathcal{R}_i$ is noetherian if modified completion is used. Moreover, since the successive term rewriting systems $\mathcal{R}_0, \mathcal{R}_1, \mathcal{R}_2, \ldots$ generated by modified completion form an increasing chain, a critical pair that is joinable w.r.t. $\mathcal{R}_i$ for some i is also joinable w.r.t. any $\mathcal{R}_j$ with $j \geq i$. Hence, since it is assumed that a fair strategy is used, $\mathcal{R}_\infty$ is locally confluent. Now consider the case that modified completion stops with success. In that case we have that the generated finite system $\mathcal{R}_\infty$ is noetherian in addition. Thus, modified completion is partially correct in that it generates a complete system equivalent to the input system $\mathcal{E}$ whenever it terminates with success. But, what about those cases when modified completion does not terminate? Is the generated infinite system $\mathcal{R}_\infty$ complete in those cases too? We will consider this case in the following.

The union of a family of noetherian term rewriting systems that form an increasing chain need not be noetherian, and in fact, the systems $\mathcal{R}_\infty$ generated by modified completion without interreduction can be non-noetherian.

Example 1. Let $\mathcal{R} = \{wa \rightarrow ab, ac \rightarrow abc\}$.
Obviously, $\mathcal{R}$ is noetherian and there is an overlap between the first rule and the second one. The corresponding critical pair is: $(abc, wabc)$. While abc is irreducible, $wabc$ can be reduced to the irreducible string $abbc$. Hence, $\mathcal{R}$ is not confluent and a Knuth-Bendix completion procedure will generate either the rule $abc \rightarrow abbc$ or the rule $abbc \rightarrow abc$, depending on the ordering used for completion. Consider the first case: If the rule $abc \rightarrow abbc$ is added, then the resulting system $\mathcal{R}_0 = \{wa \rightarrow ab, ac \rightarrow abc, abc \rightarrow abbc\}$ will be noetherian, too. But, there will be a new overlap between this new rule and the first one: We have $wabc \rightarrow abbc$ and $wabc \rightarrow wabbc \rightarrow abbbc$. Thus, a further rule has to be added. If we add the rule $abbc \rightarrow abbbc$, then the situation will be similar to the one before: The resulting system $\mathcal{R}_1 = \{wa \rightarrow ab, ac \rightarrow abc, abc \rightarrow abbc, abbc \rightarrow abbbc\}$ will be noetherian, but there will be a new overlap between the rule added and the first one. Going on in the way described, we will generate an infinite sequence of noetherian string rewriting systems $\mathcal{R}_0$, $\mathcal{R}_1$, $\mathcal{R}_2$, ... satisfying $\mathcal{R}_i = \{wa \rightarrow ab\} \cup \{ab^nc \rightarrow ab^{n+1}c \mid 0 \leq n \leq i+1\}$ ($i \in \mathbb{N}$). Since interreduction has not been used during the described process, we have $\mathcal{R}_\infty = \cup_{i \in \mathbb{N}} \mathcal{R}_i = \{wa \rightarrow ab\} \cup \{ab^nc \rightarrow ab^{n+1}c \mid n \in \mathbb{N}\}$. Hence, $\mathcal{R}_\infty$ is not noetherian.

Thus, in general modified completion is not correct, since it may generate a non-noetherian system. As mentioned above, the generated systems $\mathcal{R}_\infty$ are always locally confluent. Are they confluent too? Note that the system $\mathcal{R}_\infty$ of example 1 is confluent since it is strongly confluent [Hu80]. For non-noetherian SRSs local confluence and confluence do not coincide. Hence, modified completion might generate also non-confluent systems.
We know that the reduction induced by $\mathcal{R}_\infty$ is acyclic, since all the intermediate systems $\mathcal{R}_i$ are noetherian. Nevertheless, as the following example will show, $\mathcal{R}_\infty$ can indeed be non-confluent.

Example 2. Let $\mathcal{R} = \{1 : uv \rightarrow xA, 2 : vbc \rightarrow W, 3 : uW \rightarrow o, 4 : Abc \rightarrow abbc, 5 : wa \rightarrow Ab, 6 : wA \rightarrow ab, 7 : xa \rightarrow o, 8 : ob \rightarrow o, 9 : oc \rightarrow o, 10 : xA \rightarrow O, 11 : Ob \rightarrow O, 12 : Oc \rightarrow O\}$.
$\mathcal{R}$ is noetherian. There are 3 overlaps: Rule 1 overlaps with rule 2, and rule 4 overlaps with rule 6 and 10. Overlapping rule 1 with rule 2 yields the critical pair $(xAbc, uW)$, which is joinable in the following way: $xAbc \rightarrow xabbc \rightarrow obbc \rightarrow obc \rightarrow oc \rightarrow o \leftarrow uW$. Overlapping rule 4 with rule 6 yields the critical pair $(abbc, wabbc)$. While $abbc$ is irreducible, $wabbc$ will be reduced to the irreducible string $Abbbc$ using rule 5. Adding the rule $abbc \rightarrow Abbbc$ will

result in the noetherian system $\mathcal{R}_0 = \mathcal{R} \cup \{abbc \rightarrow Abbbc\}$. Overlapping rule 4 with rule 10 results in the critical pair $(Obc, xabbc)$, which is joinable, since $Obc \leftarrow Obbc \leftarrow Obbbc \leftarrow xAbbbc \leftarrow xabbc$. The rule added overlaps with rule 5 and with rule 7. The corresponding critical pairs are $(Abbbc, wAbbbc)$ and $(obbc, xAbbbc)$. $Abbbc$ is irreducible and $wAbbbc$ can be reduced to $abbbbc$, which is irreducible too. Adding the rule $Abbbc \rightarrow abbbbc$ will result in the system $\mathcal{R}_1 = \mathcal{R} \cup \{abbc \rightarrow Abbbc, Abbbc \rightarrow abbbbc\}$, which is noetherian. In $\mathcal{R}_1$ the critical pair $(obbc, xAbbbc)$ is joinable: $obbc \leftarrow obbbc \leftarrow obbbbc \leftarrow xabbbbc \leftarrow xAbbbc$. Thus in the next step the new rule $Abbbc \rightarrow abbbbc$ will be overlapped with the other rules. In this way the infinite, locally confluent system $\mathcal{R}_\infty = \mathcal{R} \cup \{ab^nc \rightarrow Ab^{n+1}c \mid n \geq 2, n \text{ even}\} \cup \{Ab^nc \rightarrow ab^{n+1}c \mid n \geq 3, n \text{ odd}\}$ will be generated. Since $o \leftarrow uW \leftarrow uvbc \rightarrow xAbc \rightarrow Obc \rightarrow Oc \rightarrow O$, and o and O are $\mathcal{R}_\infty$-irreducible, $\mathcal{R}_\infty$ is not confluent.

In the examples 1 and 2 we have used a very simple modified completion algorithm. It can be easily checked that the algorithm CA_MOD2 will generate the same infinite systems in these cases if appropriate reduction orderings are chosen. Thus, the algorithm CA_MOD2 is not correct in general.
Usually, the reduction orderings used for completion belong to the class of simplification orderings [De82]. Termination of the systems $\mathcal{R}_i$ that have been constructed in the previous examples cannot be proved using simplification orderings: In example 1 the initial system $\mathcal{R}$, and hence, any of the successively generated systems $\mathcal{R}_i$, is self-embedding and thus not compatible with a simplification ordering. In example 2 any of the systems $\mathcal{R}_i$ contains the set $\{Abc \rightarrow abbc, abbc \rightarrow Abbbc\}$. Since any simplification ordering contains the homeomorphic embedding relation, we have that the string $Abbbc$ is greater than the string Abc w.r.t. any simplification ordering. Hence any SRS containing the rules $Abc \rightarrow abbc$, $abbc \rightarrow Abbbc$ is not compatible w.r.t. a simplification ordering.
As mentioned before, one class of orderings often used to complete SRSs is the class of syllable orderings. Since syllable orderings are simplification orderings, they cannot be used to prove termination in the previous examples. Thus, the question arises whether or not similar phenomena may occur, if we restrict the reduction orderings that may be used during modified completion to the class of syllable orderings.
If $\mathcal{R}$ is a finite SRS and Σ the underlying alphabet, then there are only finitely many, namely $|\Sigma|!$, different syllable orderings on Σ^*. But, if the family $(>_i)_{i\in\mathbb{N}}$ of reduction orderings used during a modified completion process is restricted to a finite set and if in addition interreduction is not used, then one of these orderings is compatible with any of the successively generated systems $\mathcal{R}_i$ and thus with the set $\mathcal{R}_\infty$. Hence, modified completion without interreduction is correct if the reduction orderings $>_i$ ($i \in \mathbb{N}$) given as input belong to a finite set.

We conclude this section with the following theorem that summarizes the main results obtained so far.

Theorem 3.1 *For the algorithm CA_MOD2 holds:*

1. *The algorithm CA_MOD2 is not correct in general: If it terminates on input $(\mathcal{E},(>_i)_{i\in\mathbb{N}})$, then the generated finite system $\mathcal{R}_\infty$ is complete and equivalent to $\mathcal{E}$, but otherwise it may generate an equivalent infinite system $\mathcal{R}_\infty$ that is neither noetherian nor confluent.*

2. *The algorithm CA_MOD2 is correct for string rewriting systems and the class of syllable orderings: If it is started on input $(\mathcal{E},(>_i)_{i\in\mathbb{N}})$ where $\mathcal{E}$ is a string rewriting system and $(>_i)_{i\in\mathbb{N}}$ is a family of syllable orderings, then the generated system $\mathcal{R}_\infty$ is noetherian, confluent and equivalent to $\mathcal{E}$.*

3.2 Modified Completion with Interreduction

As shown in the previous section Huet's completion algorithm remains correct if it is allowed to change the reduction ordering during completion (provided that the new ordering is compatible with the actual set of rules) if interreduction is not used and in addition, the orderings used belong to the class of syllable orderings. Example 2 has illustrated that the second condition, i.e. the restriction of the reduction orderings to the class of syllable orderings, is essential for the correctness of this modified completion algorithm. What about the first condition not to use interreduction during completion? Is this requirement essential for the correctness of the algorithm, too?

In this section we will consider this question and analyse the correctness of the algorithm CA_MOD1. But before investigating this special algorithm, let us first consider example 1 again and analyse what will happen if interreduction is incorporated in the simple algorithm used there.

Example 3. Let $\mathcal{R} = \{wa \rightarrow ab, ac \rightarrow abc\}$.
As mentioned in example 1 a Knuth-Bendix algorithm may generate the rule $abc \rightarrow abbc$ by overlapping. Now, this new rule could be used to reduce the right hand side of the second rule. In this way we obtain the noetherian system $\mathcal{R}_0 = \{wa \rightarrow ab, ac \rightarrow abbc, abc \rightarrow abbc\}$. Again, there is an overlap between the new rule, and the first one and the rule $abbc \rightarrow abbbc$ may be generated. If interreduction is used, then this rule will be used to reduce the right hand sides of the second and the third rule. This yields $\mathcal{R}_1 = \{wa \rightarrow ab, ac \rightarrow abbbc, abc \rightarrow abbbc, abbc \rightarrow abbbc\}$. The new rule overlaps with the first rule too, and this overlap may result in the rule $abbbc \rightarrow abbbbc$, which could be used for interreduction. Using the strategy described, we may generate an infinite sequence of noetherian string rewriting systems $\mathcal{R}_0$, $\mathcal{R}_1$, $\mathcal{R}_2$, ... satisfying $\mathcal{R}_i = \{wa \rightarrow ab\} \cup \{ab^nc \rightarrow ab^{i+2}c \mid 0 \leq n \leq i+1\}$ $(i \in \mathbb{N})$. Since the right hand side of any rule different from $wa \rightarrow ab$ will be modified infinitely many times by interreduction, we have $\mathcal{R}_\infty = \{wa \rightarrow ab\}$. Hence, in this case $\mathcal{R}_\infty$ is noetherian and confluent, but it is not equivalent to $\mathcal{R}$.

Thus, if interreduction is used during modified completion, then the system $\mathcal{E}$ that has been given as input and the limit system $\mathcal{R}_\infty$ that will be generated can be non-equivalent. This phenomenon is due to the facts that the set $\cup_{i \in \mathbb{N}} \mathcal{R}_i$ may be non-noetherian and that the interreduction process in some sense simulates the computation of certain reduction sequences with respect to $\cup_{i \in \mathbb{N}} \mathcal{R}_i$. Of course, any of the intermediate systems $\mathcal{R}_i$ $(i \in \mathbb{N})$ is noetherian, and hence, any reduction process that will be performed will terminate. But, if $\cup_{i \in \mathbb{N}} \mathcal{R}_i$ is not noetherian, then the computation of a certain infinite reduction sequence w.r.t. $\cup_{i \in \mathbb{N}} \mathcal{R}_i$ may be simulated stepwise by interreduction in the following way: A rule $l \rightarrow r$ may be simplified to another rule which will be simplified to another one later on, and so on. Hence, neither the original rule $l \rightarrow r$ nor one of its simplified forms will belong to the limit system $\mathcal{R}_\infty$. Therefore, $\mathcal{R}_\infty$ may be non-equivalent to $\mathcal{R}$.

Example 3 differs from our intended one in the way that no syllable ordering is compatible with $\mathcal{R}$. But, the next example, which is based on a simple modified completion strategy different from CA_MOD1, shows that even if syllable orderings are used during modified completion, it is no longer guaranteed that the initial system and the generated limit system are equivalent.

Example 4. Let $\mathcal{R} = \{a \rightarrow b, b \rightarrow c\}$.
Moreover, let $\succ_1$, $\succ_2$ and $\succ_3$ be the syllable orderings induced by the precedence $a > b > c$, $b > c > a$ and $c > a > b$, respectively. $\mathcal{R}$ is compatible with $\succ_1$. Interreduction of $\mathcal{R}$ may result in $\mathcal{R}_0 = \{b \rightarrow c\}$ and $\mathcal{E}_0 = \{a = c\}$. Since $\mathcal{R}_0$ is compatible with $\succ_2$, we may use $\succ_2$ for the next step, and thus, the rule $c \rightarrow a$ will be added. In this way we obtain $\mathcal{R}_1 = \{b \rightarrow c, c \rightarrow a\}$ and $\mathcal{E}_1 = \emptyset$. Interreduction of $\mathcal{R}_1$ may result in $\mathcal{R}_2 = \{c \rightarrow a\}$ and $\mathcal{E}_2 = \{b = a\}$. Since $\mathcal{R}_2$ is compatible with $\succ_3$, we may use $\succ_3$ for further computations. Hence, the rule $a \rightarrow b$ will be added, and we have $\mathcal{R}_3 = \{c \rightarrow a, a \rightarrow b\}$, $\mathcal{E}_3 = \emptyset$. Again, the new rule can be used to reduce the right hand side of the other rule. Interreduction of $\mathcal{R}_3$ results in $\mathcal{R}_4 = \{a \rightarrow b\}$ and $\mathcal{E}_4 = \{c = b\}$. Now we may again change the ordering and use $\succ_1$ instead of $\succ_3$. In this way we obtain $\mathcal{R}_5 = \mathcal{R}$ and $\mathcal{E}_5 = \emptyset$. Thus, using the strategy described, an infinite sequence $(\mathcal{R}_0, \mathcal{E}_0)$, $(\mathcal{R}_1, \mathcal{E}_1)$, $(\mathcal{R}_2, \mathcal{E}_2)$, ... will be generated. Since there are no persisting rules, the corresponding limit system $\mathcal{R}_\infty$ is empty. Hence, $\mathcal{R}_\infty$ is not equivalent to $\mathcal{R}$.

The main difference between the algorithm CA_MOD1 and the one used in example 4 is the way how right hand sides of rules are simplified: Huet's completion procedure [Hu81] is a standard completion procedure in the sense of Bachmair et al. [BDH86]. In a standard completion procedure, and hence in the algorithm CA_MOD1, the simplification of a right hand side of a rule results in a new rule. In contrast to this, the above used algorithm has generated a new equation each time when a right hand side of a rule could be simplified.
If the algorithm CA_MOD1 is applied on the input system given in the last example and a family $(>_i)_{i \in \mathbb{N}}$ of reduction orderings satisfying $>_0 = \succ_1$ and $>_1 = \succ_1$, the following steps will be performed: First the sets $\mathcal{R}_1 = \{a \rightarrow b\}$, $\mathcal{E}_1 = \{b = c\}$ will be generated. Then the equation $b = c$ will be oriented according to the ordering $\succ_1$. Thus, the rule $b \rightarrow c$ will be added. In addition, the right hand side of the rule $a \rightarrow b$ will be simplified. In this way we obtain $\mathcal{R}_2 = \{a \rightarrow c, b \rightarrow c\}$, $\mathcal{E}_2 = \emptyset$. Since there are no overlaps between rules of $\mathcal{R}_2$, the algorithm CA_MOD1 will stop with the sets $\mathcal{R}_4 = \mathcal{R}_2$ and $\mathcal{E}_4 = \emptyset$. Hence, the generated system $\mathcal{R}_4$ is complete and interreduced and it is equivalent to $\mathcal{R}$.

An important fact illustrated by example 4 is that a modified completion algorithm may not be fair, although the corresponding original completion algorithm is. If a fixed reduction ordering is used, then the interreduction process will always terminate. Example 4 illustrates that this is no longer true if we allow to change the reduction ordering during the interreduction process. Hence, it is possible in that case that certain overlaps between persisting rules are not considered. For example, consider what happens if we slightly modify the input of example 4. If we add the rules $de \rightarrow g$ and $ef \rightarrow g$ and extend the orderings appropriately, then the algorithm used may generate the sequence $(\{de \rightarrow g, ef \rightarrow g\} \cup \mathcal{R}_i)_{i \in \mathbb{N}}$ instead of $(\mathcal{R}_i)_{i \in \mathbb{N}}$. Hence the overlap between the rules $de \rightarrow g, ef \rightarrow g$ will never be considered. This problem can be circumvented by requiring that the ordering only may be changed during modified completion if the actual set of equations is empty. In that case it is guaranteed that any interreduction process will terminate.

Another striking point in example 4 is the fact that the process described does not terminate, although the set $\cup_{i \in \mathbb{N}} \mathcal{R}_i$ is finite. Such a phenomenon cannot arise if Huet's completion algorithm or one of its modified versions CA_MOD1 or CA_MOD2 is used, since then a string that is reducible at some step i of the process is reducible with respect to any of the systems $\mathcal{R}_j$ with $j \geq i$. Since in addition new rules are built only from normal forms, it cannot happen that a rule is generated twice if Huet's completion algorithm or the modified versions CA_MOD1 or CA_MOD2 are used.

But even if the algorithm CA_MOD1 is used and in addition, the changes of the ordering are restricted to those cases where the corresponding sets of equations are empty, a system that is not equivalent to the input system can be generated.

Example 5. Let $\mathcal{R} = \{1 : X \to egabc, 2 : QH \to ga, 3 : QA \to Eega, 4 : Qh \to ga, 5 : qH \to a, 6 : qA \to q, 7 : qd \to q, 8 : qc \to o, 9 : qh \to a, 10 : qb \to q, 11 : WH \to Hbb, 12 : WA \to Add, 13 : wh \to hdd, 14 : wA \to Abb, 15 : eE \to \varepsilon, 16 : Eego \to go, 17 : Hbc \to Addc, 18 : hddc \to Abbbc\}$.
Moreover, let $\succ_1$ be the syllable ordering induced by the precedence $X > W > w > Q > q > H > h > b > d > a > c > g > e > A > E > o$ and $\succ_2$ the syllable ordering induced by the precedence $X > W > w > Q > q > H > h > d > b > a > c > g > e > A > E > o$, and let $(>_i)_{i \in \mathbb{N}}$ be defined by: $>_i = \succ_1$ for $0 \leq i \leq 39$, $>_{40+14j+k} = \succ_2$ for $j \in \mathbb{N}$ and $0 \leq k \leq 6$, and $>_{40+14j+k} = \succ_1$ for $j \in \mathbb{N}$ and $7 \leq k \leq 13$.
Claim: Given $\mathcal{R}$ and $(>_i)_{i \in \mathbb{N}}$ as input, the algorithm CA_MOD1 will generate an infinite sequence $(\mathcal{R}_0, \mathcal{E}_0), (\mathcal{R}_1, \mathcal{E}_1), (\mathcal{R}_2, \mathcal{E}_2), \ldots$ such that for all $j \in \mathbb{N}$ the following holds:

$$\begin{array}{lrcl}
1) & \mathcal{E}_{40+14j} & = & \emptyset \\
2) & \mathcal{R}_{40+14j} & = & \bar{\mathcal{R}}_j \text{ where}
\end{array}$$

$$\begin{array}{rl}
\bar{\mathcal{R}}_j = (\mathcal{R} - & \{1 : X \to egabc\}) \\
\cup & \{1 : X \to egad^{2j+2}c\} \\
\cup & \{Hb^n c \to Ad^{n+1}c \mid n \text{ odd and } 1 \leq n \leq 2j+1\} \\
\cup & \{ab^n c \to o \mid n \text{ odd and } 1 \leq n \leq 2j\} \\
\cup & \{hd^n c \to Ab^{n+1}c \mid n \text{ even and } 2 \leq n \leq 2j\} \\
\cup & \{ad^n c \to o \mid n \text{ even and } 2 \leq n \leq 2j\} \\
\cup & \{l_{j,1} : hd^{2j+2}c \to Ab^{2j+3}c\} \\
\cup & \{l_{j,2} : ab^{2j+1}c \to o\} \\
\cup & \{l_{j,3} : Hb^{2j+3}c \to Ad^{2j+4}c\} \\
\cup & \{l_{j,4} : Eegad^{2j+2}c \to go\}
\end{array}$$

where $l_{j,1}, l_{j,2}, l_{j,3}, l_{j,4} \in \mathbb{N}$ with $l_{j,1} < l_{j,2} < l_{j,3} < l_{j,4}$ and all rules except the rules $l_{j,1}$, $l_{j,2}$, $l_{j,3}$ and $l_{j,4}$ are marked.
Proof. The proof can be found in the appendix. □
Analysis of the proof shows that whenever the ordering is changed the corresponding set of equations is empty (i.e. if for some $i \in \mathbb{N}$, $>_i$ and $>_{i+1}$ are different, then $\mathcal{E}_{i+1}$ is empty).
As mentioned before a rule may not be generated twice during the execution of the algorithm CA_MOD1. Hence the above claim implies that for the set $\mathcal{R}_\infty$ of persisting rules the following holds:

$$\begin{array}{rcrl}
\mathcal{R}_\infty & = & (\mathcal{R} - & \{1 : X \to egabc\}) \\
& & \cup & \{Hb^n c \to Ad^{n+1}c \mid n \text{ odd and } 1 \leq n\} \\
& & \cup & \{ab^n c \to o \mid n \text{ odd and } 1 \leq n\} \\
& & \cup & \{hd^n c \to Ab^{n+1}c \mid n \text{ even and } 2 \leq n\} \\
& & \cup & \{ad^n c \to o \mid n \text{ even and } 2 \leq n\}
\end{array}$$

Since the orderings $\succ_1$ and $\succ_2$ are both used infinitely many times during the described process, $\mathcal{R}_\infty$ is compatible with both of them. Thus, $\mathcal{R}_\infty$ is noetherian. But, $\mathcal{R}_\infty$ is not equivalent to the initial system $\mathcal{R}$: $X \to egabc$ is an initial rule, but X and $egabc$ are obviously not congruent modulo $\mathcal{R}_\infty$.

It can easily be checked that the limit system $\mathcal{R}_\infty$ generated in the last example is confluent. Hence, the algorithm CA_MOD1 has generated a noetherian and confluent system in that case. But, as mentioned before $\mathcal{R}_\infty$ is not equivalent to $\mathcal{R}$. This is due to the following facts: The right hand side of rule 1 is simplified infinitely many times during the process described ($egabc \xrightarrow{*} egad^2c \xrightarrow{*} egab^3c \xrightarrow{*} egad^4c \xrightarrow{*} egab^5c \xrightarrow{*} ...$). Hence neither the original form of rule 1 nor one of its simplified forms belong to $\mathcal{R}_\infty$. On the other hand, none of these rules is redundant w.r.t. $\mathcal{R}_\infty$.

Obviously, if a non-redundant rule is simplified infinitely many times, then the generated limit system may also be non-confluent, since the crucial non-redundant rule may have been used to resolve critical pairs. For instance, consider the following modification of example 5. Extend the precedences used by $U > V > Y > Z > F > X$, and let $\succ_1$ and $\succ_2$ be the syllable orderings induced. Moreover, add the rules $UV \rightarrow Y, VZ \rightarrow Fgabc, YZ \rightarrow X, UF \rightarrow e$ in a way that the overlap between the rules $UV \rightarrow Y$ and $VZ \rightarrow Fgabc$ is the first to be considered. If the algorithm CA_MOD1 is started on this input, then the following will happen: At the moment the rules $UV \rightarrow Y$ and $VZ \rightarrow Fgabc$ are overlapped the corresponding critical pair $(YZ, UFgabc)$ is joinable in the following way: $YZ \rightarrow X \rightarrow egabc \leftarrow UFgabc$. Since the symbols of the left hand sides of the rules added are 'new' ones, the new rules will not have any further influence on the execution of the algorithm CA_MOD1, i.e. the limit system that will be generated is the union of the limit system of example 5 and the set $\{UV \rightarrow Y, VZ \rightarrow Fgabc, YZ \rightarrow X, UF \rightarrow e\}$. Thus, the critical pair $(YZ, UFgabc)$ will not be joinable w.r.t. the limit system, i.e. the limit system is not confluent in that case.

This example already gives an answer to the problem 35 of [DJK91], but we can even give an example where the algorithm generates an *equivalent*, noetherian system that is not confluent. For this purpose let us consider example 5 again, and see what will happen if we remove the crucial rule $X \rightarrow egabc$. Since this rule has neither been used for overlapping nor for reduction, the algorithm CA_MOD1 will generate the same limit system as before. Hence, in that case the generated limit system is complete and equivalent to the input system. But nevertheless, there is still a rule in the set $\cup_{i\in\mathbb{N}}\mathcal{R}_i$ that is simplified infinitely many times during this modified completion process: The rule $gabc \rightarrow Eegad^2c$ generated by overlapping is simplified to the equation $go = Eegad^2c$, which will be oriented to the rule $Eegad^2c \rightarrow go$, which will be simplified to the equation $EeEegab^3c = go$, which will yield the rule $Eegab^3c \rightarrow go$, which will be simplified to the equation $go = EeEegad^4c$, which will yield the rule $Eegad^4c \rightarrow go$, and so on. But, in this case this infinite simplification does not affect the equational theory presented by the limit system, since the rules and equations generated during this simplification process are redundant: In $\mathcal{R}_\infty$ the following reduction steps can be performed: $EeEegab^nc \rightarrow Eegab^nc \rightarrow Eego \rightarrow go$ if n is odd, and $EeEegad^nc \rightarrow Eegad^nc \rightarrow Eego \rightarrow go$ if n is even and greater than 1.

The proof of the above claim shows that the rule $Eego \rightarrow go$ will never be used for reductions during the execution of the algorithm CA_MOD1. Moreover, it shows that this rule will be overlapped only once and the corresponding critical pair is trivial in that case. Hence, if we remove the rule $Eego \rightarrow go$ as well as the rule $X \rightarrow egabc$ from the input system of example 5, the algorithm CA_MOD1 will generate the limit system $\mathcal{R}'_\infty = \mathcal{R}_\infty - \{Eego \rightarrow go\}$ where $\mathcal{R}_\infty$ is the limit system generated in example 5. Again the rule $gabc \rightarrow Eegaddc$ will be generated by overlapping and it will be simplified infinitely many times as in example 5. But the limit system $\mathcal{R}'_\infty$ that will be generated is a superset of the input system and hence, both systems are equivalent. In $\mathcal{R}'_\infty$ the following reduction steps can be performed: $go \leftarrow gabc \leftarrow QHbc \rightarrow QAddc \rightarrow Eegaddc \rightarrow Eego$. Hence we have $go \overset{*}{\leftrightarrow}_{\mathcal{R}'_\infty} Eego$. Thus, $\mathcal{R}'_\infty$ is noetherian and equivalent to the input system, but it is not confluent.

In order to examine this phenomenon more closely, let us consider how the relationship between the components of the critical pair $(gabc, QAddc)$ changes during the execution of the algorithm CA_MOD1.
At the moment when the critical pair $(gabc, QAddc)$ is considered, the string $gabc$ is irreducible and the string $QAddc$ can be reduced to the irreducible string $Eegaddc$. Since the ordering $\succ_1$ is used in that step, the rule $gabc \rightarrow Eegad^2c$ is generated, and we have the situation illustrated in figure 5.1. Later on, the rules $abc \rightarrow o$ and $Eegad^2c \rightarrow go$ are generated. Hence then the critical pair $(gabc, QAddc)$ is joinable as illustrated in figure 5.2. Note that the strings $gabc$ and $Eegad^2c$ are not comparable w.r.t. the reduction ordering that is induced by the current set of rules at that moment. Hence, the ordering $\succ_2$ can be used for further steps. Doing this the rule $gad^2c \rightarrow Eegab^3c$ will be added. Then, this new rule is used to simplify the rule $Eegad^2c \rightarrow go$. In this way the equation $EeEegab^3c = go$ is obtained (s. figure 5.3).

$$\begin{tikzcd}
& QHbc \arrow[dl] \arrow[dr] & \\
gabc \arrow[drr] & & QAddc \arrow[d] \\
& & Eegad^2c
\end{tikzcd}$$

fig. 5.1

$$\begin{tikzcd}
& QHbc \arrow[dl] \arrow[dr] & \\
gabc \arrow[d] & & QAddc \arrow[d] \\
go & & Eegad^2c \arrow[ll]
\end{tikzcd}$$

fig. 5.2

$$\begin{tikzcd}
& QHbc \arrow[dl] \arrow[dr] & \\
gabc \arrow[d] & & QAddc \arrow[d] \\
go \arrow[drr, equal] & & Eegad^2c \arrow[d] \\
& & EeEegab^3c
\end{tikzcd}$$

fig. 5.3

Next, the rule $ad^2c \rightarrow o$ is added and hence, the rule $gad^2c \rightarrow Eegab^3c$ is being simplified to the equation $Eegab^3c = go$ (s. figure 5.4). The equation $EeEegab^3c = go$ yields the rule $Eegab^3c \rightarrow go$. Hence, the string $EeEegab^3c$ can be reduced to $Eego$ as well as to go. Now, the critical pair $(gabc, QAddc)$ is not joinable, but it is connected below the string $QHbc$ (s. figure 5.5).

$$\begin{tikzcd}
& QHbc \arrow[dl] \arrow[dr] & & \\
gabc \arrow[d] & & QAddc \arrow[d] & \\
go \arrow[drr, equal] & & Eegad^2c \arrow[dr] & \\
& & EeEegab^3c \arrow[r, equal] & Eego
\end{tikzcd}$$

fig. 5.4

$$\begin{tikzcd}
& QHbc \arrow[dl] \arrow[dr] & & \\
gabc \arrow[d] & & QAddc \arrow[d] & \\
go & & Eegad^2c \arrow[dr] & \\
& & EeEegab^3c \arrow[r] \arrow[d] & Eego \\
& & Eegab^3c \arrow[uull] &
\end{tikzcd}$$

fig. 5.5

For the next steps $\succ_1$ is used again. By overlapping the rule $gab^3c \rightarrow Eegad^4c$ is generated. This new rule is used for simplification and the situation illustrated in figure 5.6 arises. Then, the rule $ab^3c \rightarrow o$ is generated by overlapping and thus, the strings $gabc$ and $QAddc$ are related as illustrated in figure 5.7.

$$\begin{tikzcd}
 & QHbc \arrow[dl] \arrow[dr] & & \\
gabc \arrow[d] & & QAddc \arrow[d] & \\
go \arrow[dddrr, equal] & & Eegad^2c \arrow[dr] & \\
 & & EeEegab^3c \arrow[d] \arrow[dr] & Eego \arrow[d, equal] \\
 & & Eegab^3c \arrow[d] & EeEeEegad^4c \\
 & & EeEegad^4c &
\end{tikzcd}$$

fig. 5.6

$$\begin{tikzcd}
 & QHbc \arrow[dl] \arrow[dr] & & \\
gabc \arrow[d] & & QAddc \arrow[d] & \\
go \arrow[ddddrr, equal] & & Eegad^2c \arrow[dr] & \\
 & & & Eego \\
 & & Eegab^3c \arrow[ur] & \\
 & & EeEegad^4c \arrow[uur, equal] &
\end{tikzcd}$$

fig. 5.7

The equation $EeEegad^4c = go$ is simplified to the rule $Eegad^4c \rightarrow go$. Hence, the situation is similar to the one illustrated in figure 5.5: The critical pair $(gabc, QAddc)$ is connected below the string $QHbc$, but now, via the 'peak' $go \overset{*}{\leftarrow} EeEegad^4c \rightarrow Eego$ (s. figure 5.8). And in fact, at one of the subsequent steps the situation given by figure 5.9 will arise.

$$\begin{tikzcd}
 & QHbc \arrow[dl] \arrow[dr] & & \\
gabc \arrow[d] & & QAddc \arrow[d] & \\
go & & Eegad^2c \arrow[dr] & \\
 & & & Eego \\
 & & Eegab^3c \arrow[ur] & \\
 & & EeEegad^4c \arrow[uur] \arrow[d] & \\
 & & Eegad^4c \arrow[uuuuull] &
\end{tikzcd}$$

fig. 5.8

$$\begin{tikzcd}
 & QHbc \arrow[dl] \arrow[dr] & & \\
gabc \arrow[d] & & QAddc \arrow[d] & \\
go & & Eegad^2c \arrow[dr] & \\
 & & & Eego \\
 & & Eegab^3c \arrow[ur] & \\
 & & Eegad^4c \arrow[uur] & \\
 & & EeEegab^5c \arrow[uuur] \arrow[d] & \\
 & & Eegab^5c \arrow[uuuuuull] &
\end{tikzcd}$$

fig. 5.9

Hence in small steps the critical pair $(gabc, QAddc)$ will be being connected (w.r.t. $\succ_1$ as well as w.r.t. $\succ_2$) below $QHbc$ via the 'peaks' $go \overset{*}{\leftarrow} EeEegab^nc \rightarrow Eego$ where $n \in \mathbb{N}$ is an odd number greater than 2, as well as via the 'peaks' $go \overset{*}{\leftarrow} EeEegad^nc \rightarrow Eego$ where $n \in \mathbb{N}$ is an even number greater than 3. Thus the original rewrite proof for $gabc \overset{*}{\leftrightarrow} QAddc$, illustrated in figure 5.1, will be transformed infinitely many times during the execution of the algorithm CA_MOD1, and in the generated limit system $gabc$ and $QAddc$ will not be joinable.

The last example has shown that even if the algorithm CA_MOD1 generates an infinite, noetherian system presenting the same equational theory as the input system, this limit system may be non-confluent. There remains to check whether the algorithm CA_MOD1 at least is partially correct, i.e. if it always generates correct results whenever it terminates with success. If the algorithm CA_MOD1 terminates with success, then the finite system is noetherian and equivalent to the input system. But even in this case, the generated system may be non-confluent.

Example 6. Let $\mathcal{R} = \{1 : xwef \rightarrow xweg, 2 : egc \rightarrow dgc, 3 : xwd \rightarrow xwi, 4 : ubc \rightarrow o, 5 : xwigc \rightarrow o, 6 : xwa \rightarrow u, 7 : abc \rightarrow efc, 8 : zf \rightarrow \varepsilon, 9 : hz \rightarrow we, 10 : yb \rightarrow g, 11 : iy \rightarrow a\}$. Moreover, let $\succ_1$ be the syllable ordering induced by the precedence $z > x > y > a > f > g > b > w > e > d > c > i > o > h > u$ and $\succ_2$ be the syllable ordering induced by the precedence $z > x > y > g > a > f > b > w > e > d > c > i > o > h > u$, and let $(>_i)_{i \in \mathbb{N}}$ be a family of syllable orderings satisfying: $>_i = \succ_1$ for $0 \leq i \leq 24$ and $>_i = \succ_2$ for $25 \leq i \leq 30$. Since $\mathcal{R}$ is compatible with $\succ_1$ and $\mathcal{R}$ is interreduced, the algorithm CA_MOD1 will generate the sets $\mathcal{R}_{11} = \mathcal{R}$, $\mathcal{E}_{11} = \emptyset$, where all rules in $\mathcal{R}_{11}$ are unmarked. Since there are no overlaps between rules of $\mathcal{R} - \{7 : abc \rightarrow efc, 8 : zf \rightarrow \varepsilon, 9 : hz \rightarrow we, 10 : yb \rightarrow g, 11 : iy \rightarrow a\}$, we have $\mathcal{R}_{17} = \mathcal{R}$, $\mathcal{E}_{17} = \emptyset$, where all rules of $\mathcal{R}_{17}$ except the rules 7, 8, 9, 10 and 11 are marked. Hence, in the next step the rule 7 will be marked and all critical pairs between the rule 7 and the rules 1-7 will be computed. Rule 7 only overlaps with rule 6. The corresponding critical pair is $(ubc, xwefc)$. Thus we obtain $\mathcal{R}_{18} = \mathcal{R}$, where all rules except the rules 8-11 are marked, and $\mathcal{E}_{18} = \{ubc = xwefc\}$. Since the critical pair $(ubc, xwefc)$ is joinable ($ubc \rightarrow o \leftarrow xwigc \leftarrow xwdgc \leftarrow xwegc \leftarrow xwefc$) the following holds: $\mathcal{R}_{19} = \mathcal{R}$ and $\mathcal{E}_{19} = \emptyset$. In the next step rule 8 will be marked and the corresponding critical pairs will be computed. Since there are no overlaps between rule 8 and the rules 1-8, we obtain $\mathcal{R}_{20} = \mathcal{R}$, $\mathcal{E}_{20} = \emptyset$, where all rules except the rules 9-11 are marked. Rule 9 overlaps only with rule 8. The corresponding critical pair is (wef, h). Thus, we have $\mathcal{R}_{21} = \mathcal{R}$ and $\mathcal{E}_{21} = \{wef = h\}$. Since wef and h are irreducible w.r.t. $\mathcal{R}_{21}$ and $wef \succ_1 h$, the rule $12 : wef \rightarrow h$ will be added. Now this new rule will be used to reduce the left hand side of the first rule. This gives $\mathcal{R}_{22} = (\mathcal{R} - \{1 : xwef \rightarrow xweg\}) \cup \{12 : wef \rightarrow h\}$ and $\mathcal{E}_{22} = \{xweg = xh\}$. Since $xweg$ and xh are irreducible w.r.t. $\mathcal{R}_{22}$ and $xweg \succ_1 xh$, the rule $13 : xweg \rightarrow xh$ will be added. This yields: $\mathcal{R}_{23} = (\mathcal{R} - \{1 : xwef \rightarrow xweg\}) \cup \{12 : wef \rightarrow h, 13 : xweg \rightarrow xh\}$ and $\mathcal{E}_{23} = \emptyset$, where all rules except the rules 10-13 are marked. Since there are no overlaps between rule 10 and the rules 1-10, the sets $\mathcal{R}_{24} = \mathcal{R}_{23}$ and $\mathcal{E}_{24} = \emptyset$ will be generated. In the next step rule 11 will be overlapped with rule 10. This gives $\mathcal{R}_{25} = \mathcal{R}_{23}$ and $\mathcal{E}_{25} = \{ab = ig\}$. $\mathcal{R}_{25}$ is compatible with $\succ_2$. Hence, $\succ_2$ can be used for the next step. Since ab and ig are $\mathcal{R}_{25}$-irreducible and $ig \succ_2 ab$, the rule $14 : ig \rightarrow ab$ will be generated. This new rule will be used to reduce the left hand side of rule 5. In this way we obtain $\mathcal{R}_{26} = \{2 : egc \rightarrow dgc, 3 : xwd \rightarrow xwi, 4 : ubc \rightarrow o, 6 : xwa \rightarrow u, 7 : abc \rightarrow efc, 8 : zf \rightarrow \varepsilon, 9 : hz \rightarrow we, 10 : yb \rightarrow g, 11 : iy \rightarrow a, 12 : wef \rightarrow h, 13 : xweg \rightarrow xh, 14 : ig \rightarrow ab\}$ and $\mathcal{E}_{26} = \{xwabc = o\}$. Since $xwabc \rightarrow ubc \rightarrow o$, the sets $\mathcal{R}_{27} = \mathcal{R}_{26}$ and $\mathcal{E}_{27} = \emptyset$ will be generated. Rule 12 does not overlap with any of the rules 1-12. Therefore we obtain $\mathcal{R}_{28} = \mathcal{R}_{26}$ and $\mathcal{E}_{28} = \emptyset$, where all rules except the rules 13 and 14 are marked. Next, rule 13 will be marked. Rule 13 only overlaps with rule 2. The corresponding critical pair $(xhc, xwdgc)$ is joinable in the following way: $xwdgc \rightarrow xwigc \rightarrow xwabc \rightarrow xwefc \rightarrow xhc$. Hence. we have $\mathcal{R}_{29} = \mathcal{R}_{26}$ and $\mathcal{E}_{29} = \emptyset$. Since rule 14 does not overlap with any of the other rules, the algorithm CA_MOD1 will stop with the sets $\mathcal{R}_{30} = \mathcal{R}_{26}$ and $\mathcal{E}_{30} = \emptyset$. Hence, $\mathcal{R}_{\infty} = \mathcal{R}_{26}$. But $\mathcal{R}_{26}$ is not confluent: $o \overset{*}{\leftrightarrow} xhc$ (since $o \leftarrow ubc \leftarrow xwabc \rightarrow xwefc \rightarrow xhc$), and o and xhc are $\mathcal{R}_{26}$-irreducible.

Thus, even if the algorithm CA_MOD1 terminates with success, the generated noetherian system can be non-confluent.
In order to illustrate this phenomenon, let us consider how the relationship between the strings ubc and $xwefc$ (which form a critical pair) changes during the described process: At the moment of the process when this critical pair is considered it is joinable (see figure 6.1). Later on, the rule $1 : xwef \rightarrow xweg$ that has been used to solve the critical pair $(ubc, xwefc)$ is simplified. Therefore, the critical pair is no longer joinable, but it is connected below $xwabc$ with respect to $\succ_1$ as illustrated in figure 6.2.

$$\begin{tikzcd}
& xwabc \arrow[dl] \arrow[dr] & \\
ubc \arrow[dddd] & & xwefc \arrow[d] \\
& & xwegc \arrow[d] \\
& & xwdgc \arrow[d] \\
& & xwigc \arrow[dll] \\
o & &
\end{tikzcd}$$

fig. 6.1

$$\begin{tikzcd}
& xwabc \arrow[dl] \arrow[dr] & & \\
ubc \arrow[dddd] & & xwefc \arrow[ddr] & \\
& & xwegc \arrow[d] \arrow[dr] & \\
& & xwdgc \arrow[d] & xhc \\
& & xwigc \arrow[dll] & \\
o & & &
\end{tikzcd}$$

fig. 6.2

In one of the further steps the ordering used is changed such that ig is greater than ab with respect to the new ordering. In order to illustrate this fact, we have rewritten the graph of figure 6.2 to the one of figure 6.3. During the following steps the rule $ig \rightarrow ab$ will be generated by overlapping. Thus, the rule $5 : xwigc \rightarrow o$ will be deleted, and the situation illustrated in figure 6.4 arises. In the graph of figure 6.4 there are two 'peaks'. For the corresponding critical pairs the following holds: The critical pair $(ubc, xwefc)$ has already been considered and the critical pair $(xwdgc, xhc)$ is joinable. But, nevertheless the critical pair $(ubc, xwefc)$ is not joinable any more.

$$\begin{tikzcd}
& & xwegc \arrow[dll] \arrow[ddddrr] & & \\
xwdgc \arrow[d] & & & & \\
xwigc \arrow[ddddr] & & xwabc \arrow[dl] \arrow[dr] & & \\
& ubc \arrow[ddd] & & xwefc \arrow[dr] & \\
& & & & xhc \\
& & & & \\
& o & & &
\end{tikzcd}$$

fig. 6.3

$$\begin{tikzcd}
& & xwegc \arrow[dll] \arrow[ddddrr] & & \\
xwdgc \arrow[d] & & & & \\
xwigc \arrow[rr] & & xwabc \arrow[dl] \arrow[dr] & & \\
& ubc \arrow[ddd] & & xwefc \arrow[dr] & \\
& & & & xhc \\
& & & & \\
& o & & &
\end{tikzcd}$$

fig. 6.4

Concluding our analysis of the correctness of the algorithm CA_MOD1 we summarize the results obtained.

Theorem 3.2 *The algorithm CA_MOD1 is not correct:*

1. *If it does not terminate on input $(\mathcal{E},(>_i)_{i\in\mathbb{N}})$, then the generated infinite system $\mathcal{R}_\infty$ may be non-equivalent to $\mathcal{E}$. In addition, $\mathcal{R}_\infty$ may or may not be confluent.*
2. *If it terminates on input $(\mathcal{E},(>_i)_{i\in\mathbb{N}})$, then the generated finite system $\mathcal{R}_\infty$ is noetherian and equivalent to $\mathcal{E}$, but it may be non-confluent.*

4 Concluding Remarks

We have analysed whether or not Huet's algorithm remains correct if it is modified in the following way:
Instead of one (recursive) reduction ordering, a family $(>_i)_{i\in\mathbb{N}}$ of (recursive) reduction orderings is required as input. If the equation that is considered at step i of the algorithm is not trivial, the algorithm will proceed as follows. It will stop with failure if the ordering $>_i$ is not compatible with the actual set of rules $\mathcal{R}_i$. Otherwise, the equation under consideration will be oriented w.r.t. $>_i$ if possible.

We have shown that this variant of Huet's completion algorithm is not correct regardless of the fact whether or not interreduction is used within. In particular, we have proved that in case interreduction is used the algorithm is not even partially correct: Even if the algorithm terminates with success, the generated finite, noetherian system may be non-confluent.

Since a finite term rewriting system $\mathcal{R}$ is noetherian if and only if there exists a (recursive) reduction ordering that is compatible with $\mathcal{R}$, the same systems can be generated if we modify Huet's algorithm as follows:
Instead of using a reduction ordering to ensure termination of the successively generated term rewriting systems, we allow to choose an arbitrary orientation of the equations. If the resulting system can be proved to be noetherian (using a certain method), the algorithm will continue in the usual way. Otherwise, the algorithm will terminate with failure.
This variant of the Knuth-Bendix completion algorithm (apart from slight modifications) has been considered several times in the literature (see e.g. [BL82], [HO80], [Pl86], [KKW89]). In these versions usually interreduction is not used and the authors restrict their attention to those cases where the algorithm considered terminates with success. Of course, a system generated in that way is finite, complete and equivalent to the corresponding input.
Our examples show that one must be careful if these restrictions are not included. Even if interreduction is not used, an infinite, non-noetherian and non-confluent term rewriting system may be generated in that way. Hence, in contrast to the usual completion algorithm, the modified algorithm cannot be used as a semidecision procedure for the word problem of the input system. Moreover, if interreduction is incorporated, a lot of problems may arise. The generated limit system may be non-equivalent to the corresponding input, or even noetherian and equivalent to the input system, but not confluent. Example 6 has shown that the latter case even may arise if the algorithm terminates with success.

While Huet and Oppen state in [HO80] that in case that the algorithm terminates with success the generated system is locally confluent, in [Hu81] Huet makes the following brief remark: *"The proof turned out to be more difficult than we had expected, and revealed critical conditions for the justification of rewrite rules simplifications, which may not be met by existing implementations. In particular, it is not enough to require that all the successive term rewriting systems $\mathcal{R}_1, \mathcal{R}_2, \ldots$ constructed by the algorithm be noetherian. They must be terminating for the same reason; i.e. there must exist some uniform reduction ordering $>$ showing the termination of all these sets."*. But unfortunately, Huet does not explain why this restriction is needed.
Apart from this remark we are not aware of any other hints in the literature that Huet's algorithm becomes incorrect if it is modified in the way described. On the contrary, usually it is assumed that the modified completion algorithm is at least partially correct in that it generates a complete system equivalent to the corresponding input system whenever it terminates with success. Example 6 disproves this widespread assumption.

These results also are important from a practical point of view, since most existing implementations of the Knuth-Bendix algorithm provide the option to orient equations by hand. Example 6 shows that in case this option is used during a completion process and the corresponding process terminates with success, the only thing we can conclude for the generated system is that it is equivalent to the input system, nothing more.
This observation may affect the correctness of existing implementations of the completion algorithm. For example, we have run example 6 with the system RRL (version 4.1) [KZ89] using the option to orient equations manually. By choosing the parameters 'option critical pick f', 'option norm m' (which determine the strategy used for computing critical pairs as well as the normalization strategy) we obtained the same result as in example 6 but with the remark: "Your system is locally-confluent".

Acknowledgement: I would like to thank Klaus Madlener and Birgit Reinert for their valuable comments on a previous version of this paper.

References

[BL82] B. Buchberger, R. Loos. Algebraic simplification. In: *Computer Algebra* (Springer, Berlin, 1982), 11-43.

[BDH86] L. Bachmair, N. Dershowitz, J. Hsiang. Orderings for equational proofs. In: *Proc. IEEE Symposium on Logic in Computer Science*, Cambridge, MA, 1986, 346-357.

[Bo87] R.V. Book. Thue systems as rewriting systems. *Journal Symbolic Computation* 3 (1987), 39-68.

[BO93] R.V. Book, F. Otto. *String-Rewriting Systems.* Texts and Monographs in Computer Science (Springer, New York, 1993).

[De82] N. Dershowitz. Orderings for term-rewriting systems. *Theoretical Computer Science* 17(3) (1982), 279-301.

[De89] N. Dershowitz. Completion and its applications. In: H. Ait-Kaci and M. Nivat (eds.): *Resolution of Equations in Algebraic Structures, Vol. II: Rewriting Techniques* (Academic Press, New York, 1989), 31-86.

[DJK91] N. Dershowitz, J.-P. Jouannaud, J.W. Klop. Open Problems in Rewriting. In: *Proc. Fourth International Conference on Rewriting Techniques and Applications*, Como, Italy, Lecture Notes in Computer Science 488 (Springer, Berlin, 1991), 445-456.

[DJK93] N. Dershowitz, J.-P. Jouannaud, J.W. Klop. More Problems in Rewriting. In: *Proc. Fifth International Conference on Rewriting Techniques and Applications*, Montreal, Canada, Lecture Notes in Computer Science 690 (Springer, Berlin, 1993), 468-487.

[He88] M. Hermann. *Vademecum of divergent term rewriting systems.* CRIN Report 88-R-082 (Centre de Recherche en Informatique de Nancy, 1988).

[HO80] G. Huet, D.C. Oppen: Equations and rewrite rules: A survey. In: R. Book (ed.): *Formal Language Theory: Perspectives and Open Problems* (Academic Press, New York, 1980), 349-405.

[Hu80] G. Huet. Confluent reductions: Abstract properties and applications to term rewriting systems. *Journal of the ACM* 27(4) (1980), 797-821.

[Hu81] G. Huet. A complete proof of correctness of the Knuth-Bendix completion algorithm. *Journal Computer and System Science* 23(1) (1981), 11-21.

[Ja88] M. Jantzen. *Confluent String Rewriting.* EATCS Monographs on Theoretical Computer Science Vol. 14 (Springer, Berlin - Heidelberg, 1988).

[KB70] D.E. Knuth, P. Bendix. Simple word problems in universal algebras. In: J. Leech (ed.): *Computational Problems in Abstract Algebra* (Pergamon, New York, 1970), 263-297.

[KKW89] A. Kandri-Rody, D. Kapur, F. Winkler. Knuth-Bendix procedure and Buchberger algorithm - A Synthesis. In: *Proc. International Symposium on Symbolic and Algebraic Computation*, Portland, Oregon, 1989, 55-67.

[KZ89] D. Kapur, H. Zhang. *RRL: Rewrite Rule Laboratory - User's Manual*, GE Corporate Research and Development Report, Schenectady, New York, 1987 (revised version: May 1989).

[Pl86] D.A. Plaisted. A simple non-termination test for the Knuth-Bendix method. In: *Proc. Eighth International Conference on Automated Deduction*, Oxford, England, Lecture Notes in Computer Science 230 (Springer, Berlin, 1986), 79-88.

[Pu87] P. Purdom. Detecting loop simplifications. In: *Proc. Second International Conference on Rewriting Techniques and Applications*, Bordeaux, France, Lecture Notes in Computer Science 256 (Springer, Berlin, 1987), 54-61.

[St89] J. Steinbach. *Comparing on Strings: Iterated Syllable Ordering and Recursive Path Orderings.* SEKI Report SR-89-15 (Universität Kaiserslautern, 1989).

Appendix

Example 5. Let $\mathcal{R} = \{1 : X \to egabc, 2 : QH \to ga, 3 : QA \to Eega, 4 : Qh \to ga, 5 : qH \to a, 6 : qA \to q, 7 : qd \to q, 8 : qc \to o, 9 : qh \to a, 10 : qb \to q, 11 : WH \to Hbb, 12 : WA \to Add, 13 : wh \to hdd, 14 : wA \to Abb, 15 : eE \to \varepsilon, 16 : Eego \to go, 17 : Hbc \to Addc, 18 : hddc \to Abbbc\}$.
Moreover, let $\succ_1$ be the syllable ordering induced by the precedence $X > W > w > Q > q > H > h > b > d > a > c > g > e > A > E > o$ and $\succ_2$ the syllable ordering induced by the precedence $X > W > w > Q > q > H > h > d > b > a > c > g > e > A > E > o$, and let $(>_i)_{i\in\mathbb{N}}$ be defined by: $>_i = \succ_1$ for $0 \le i \le 39$, $>_{40+14j+k} = \succ_2$ for $j \in \mathbb{N}$ and $0 \le k \le 6$, and $>_{40+14j+k} = \succ_1$ for $j \in \mathbb{N}$ and $7 \le k \le 13$.
Claim: Given $\mathcal{R}$ and $(>_i)_{i\in\mathbb{N}}$ as input, the algorithm CA_MOD1 will generate an infinite sequence $(\mathcal{R}_0, \mathcal{E}_0), (\mathcal{R}_1, \mathcal{E}_1), (\mathcal{R}_2, \mathcal{E}_2), \ldots$ such that for all $j \in \mathbb{N}$ the following holds:

$$
\begin{array}{lrcll}
1) & \mathcal{E}_{40+14j} & = & \emptyset & \\
2) & \mathcal{R}_{40+14j} & = & \bar{\mathcal{R}}_j \text{ where} & \\
 & & & \bar{\mathcal{R}}_j = (\mathcal{R} - & \{1 : X \to egabc\}) \\
 & & & \cup & \{1 : X \to egad^{2j+2}c\} \\
 & & & \cup & \{Hb^nc \to Ad^{n+1}c \mid n \text{ odd and } 1 \le n \le 2j+1\} \\
 & & & \cup & \{ab^nc \to o \mid n \text{ odd and } 1 \le n \le 2j\} \\
 & & & \cup & \{hd^nc \to Ab^{n+1}c \mid n \text{ even and } 2 \le n \le 2j\} \\
 & & & \cup & \{ad^nc \to o \mid n \text{ even and } 2 \le n \le 2j\} \\
 & & & \cup & \{l_{j,1} : hd^{2j+2}c \to Ab^{2j+3}c\} \\
 & & & \cup & \{l_{j,2} : ab^{2j+1}c \to o\} \\
 & & & \cup & \{l_{j,3} : Hb^{2j+3}c \to Ad^{2j+4}c\} \\
 & & & \cup & \{l_{j,4} : Eegad^{2j+2}c \to go\}
\end{array}
$$

where $l_{j,1}, l_{j,2}, l_{j,3}, l_{j,4} \in \mathbb{N}$ with $l_{j,1} < l_{j,2} < l_{j,3} < l_{j,4}$ and all rules except the rules $l_{j,1}$, $l_{j,2}$, $l_{j,3}$ and $l_{j,4}$ are marked.

Proof. The proof is done by induction on j.
Let $(\mathcal{R}_0, \mathcal{E}_0), (\mathcal{R}_1, \mathcal{E}_1), (\mathcal{R}_2, \mathcal{E}_2), \ldots$ be the sets that are generated by the algorithm CA_MOD1 if it is started on input $\mathcal{R}$ and $(>_i)_{i\in\mathbb{N}}$.
(Note that for any $j \in \mathbb{N}$, $\bar{\mathcal{R}}_j$ is compatible with $\succ_1$ as well as with $\succ_2$. Therefore, we will check the orientation of a rule generated by the algorithm CA_MOD1 only if the rule does not belong to any of these sets.)
Induction basis: Since $\mathcal{R}$ is interreduced and compatible with $\succ_1$, it holds that

$$\mathcal{R}_{18} = \mathcal{R} \quad ,$$

$$\mathcal{E}_{18} = \emptyset \quad ,$$

where all rules in $\mathcal{R}_{18}$ are unmarked. There are no overlaps between rules of $\mathcal{R} - \{16 : Eego \to go, 17 : Hbc \to Addc, 18 : hddc \to Abbbc\}$. Hence, we have

$$\mathcal{R}_{33} = \mathcal{R} \quad ,$$

$$\mathcal{E}_{33} = \emptyset \quad ,$$

where all rules of $\mathcal{R}_{33}$ except the rules 16, 17 and 18 are marked. In the next step rule 16 will be marked and all critical pairs between rule 16 and the rules 1-16 will be computed. Rule 16 only overlaps with rule 15. This overlap yields the trivial critical pair (ego, ego). Hence, the algorithm CA_MOD1 generates the sets

$$\mathcal{R}_{35} = \mathcal{R} \quad ,$$
$$\mathcal{E}_{35} = \emptyset \quad ,$$

where all rules of $\mathcal{R}_{35}$ except the rules 17 and 18 are marked. In the next step the rule 17 will be considered. Rule 17 only overlaps with the rules 2, 5 and 11. The corresponding set of critical pairs is $\mathcal{E}_{36} = \{gabc = QAddc, abc = qAddc, Hbbbc = WAddc\}$. Consider the critical pair $gabc = QAddc$. While $gabc$ is irreducible, the string $QAddc$ can be reduced to the irreducible string $Eegaddc$. Since $>_{36} = \succ_1$ and $gabc \succ_1 Eegaddc$, the rule $19 : gabc \rightarrow Eegaddc$ will be generated. This new rule will be used to simplify rule 1. In this way we obtain

$$\begin{aligned}\mathcal{R}_{37} &= (\mathcal{R} - \{1 : X \rightarrow egabc\}) \\ &\cup \{19 : gabc \rightarrow Eegaddc,\ 1 : X \rightarrow egaddc\} \quad , \\ \mathcal{E}_{37} &= \{abc = qAddc,\ Hbbbc = WAddc\} \quad .\end{aligned}$$

In the next step the critical pair $abc = qAddc$ will be considered. Since $qAddc \rightarrow qddc \xrightarrow{*} qc \rightarrow o$, the rule $20 : abc \rightarrow o$ will be added. This new rule will be used to simplify rule 19. In this way we obtain

$$\begin{aligned}\mathcal{R}_{38} &= (\mathcal{R} - \{1 : X \rightarrow egabc\}) \\ &\cup \{1 : X \rightarrow egaddc,\ 20 : abc \rightarrow o\} \quad , \\ \mathcal{E}_{38} &= \{Hbbbc = WAddc,\ go = Eegaddc\} \quad .\end{aligned}$$

Normalizing the critical pair $Hbbbc = WAddc$ will result in the pair $Hbbbc = Addddc$ and the rule $21 : Hbbbc \rightarrow Addddc$ will be added. Thus it holds:

$$\begin{aligned}\mathcal{R}_{39} &= (\mathcal{R} - \{1 : X \rightarrow egabc\}) \\ &\cup \{1 : X \rightarrow egaddc,\ 20 : abc \rightarrow o,\ 21 : Hbbbc \rightarrow Addddc\} \quad , \\ \mathcal{E}_{39} &= \{go = Eegaddc\} \quad .\end{aligned}$$

The strings go and $Eegaddc$ are irreducible. Thus, we obtain

$$\begin{aligned}\mathcal{R}_{40} &= (\mathcal{R} - \{1 : X \rightarrow egabc\}) \\ &\cup \{1 : X \rightarrow egaddc,\ 20 : abc \rightarrow o,\ 21 : Hbbbc \rightarrow Addddc,\ 22 : Eegaddc \rightarrow go\} \quad , \\ \mathcal{E}_{40} &= \emptyset \quad ,\end{aligned}$$

where all rules except the rules 18, 20, 21 and 22 are marked. Hence our claim holds for $j = 0$.
Induction step: Assume the claim holds for some $j \in \mathbb{N}$. Suppose that the algorithm CA_MOD1 has just generated the sets $\mathcal{R}_{40+14j}$ and $\mathcal{E}_{40+14j}$, and consider the steps the algorithm will perform next. First the rule $l_{j,1}$ is marked. Rule $l_{j,1}$ only overlaps with the rules 4, 9 and 13. The corresponding set of critical pairs is $\mathcal{E}_{41+14j} = \{gad^{2j+2}c = QAb^{2j+3}c, ad^{2j+2}c = qAb^{2j+3}c, hd^{2(j+1)+2}c = wAb^{2j+3}c\}$. While $gad^{2j+2}c$ is irreducible, the string $QAb^{2j+3}c$ will be reduced to $Eegab^{2j+3}c$ using rule 3. Since $>_{41+14j} = \succ_2$ and $gad^{2j+2}c \succ_2 Eegab^{2j+3}c$, the rule $l_{j,4} + 1 : gad^{2j+2}c \rightarrow Eegab^{2j+3}c$ will be added. This new rule will be used to simplify the rules 1 and $l_{j,4}$. The resulting sets will be

$$\begin{aligned}\mathcal{R}_{42+14j} &= (\bar{\mathcal{R}}_j - \{1 : X \rightarrow egad^{2j+2}c,\ l_{j,4} : Eegad^{2j+2}c \rightarrow go\}) \\ &\cup \{l_{j,4} + 1 : gad^{2j+2}c \rightarrow Eegab^{2j+3}c,\ 1 : X \rightarrow egab^{2j+3}c\} \quad , \\ \mathcal{E}_{42+14j} &= \{ad^{2j+2}c = qAb^{2j+3}c,\ hd^{2(j+1)+2}c = wAb^{2j+3}c,\ EeEegab^{2j+3}c = go\} \quad .\end{aligned}$$

Normalizing the critical pair $ad^{2j+2}c = qAb^{2j+3}c$ will result in the pair $ad^{2j+2}c = o$. Thus the rule $l_{j,4} + 2 : ad^{2j+2}c \rightarrow o$ will be added. Rule $l_{j,4} + 2$ will be used to simplify the rule $l_{j,4} + 1$. In this way the sets

$$\begin{aligned}
\mathcal{R}_{43+14j} &= (\bar{\mathcal{R}}_j - \{1 : X \rightarrow egad^{2j+2}c,\ l_{j,4} : Eegad^{2j+2}c \rightarrow go\}) \\
&\cup\ \{1 : X \rightarrow egab^{2j+3}c, l_{j,4} + 2 : ad^{2j+2}c \rightarrow o\} \quad , \\
\mathcal{E}_{43+14j} &= \{hd^{2(j+1)+2}c = wAb^{2j+3}c,\ EeEegab^{2j+3}c = go,\ go = Eegab^{2j+3}c\}
\end{aligned}$$

are obtained. While $hd^{2(j+1)+2}c$ is irreducible, the string $wAb^{2j+3}c$ will be reduced to $Ab^{2(j+1)+3}c$, and we get

$$\begin{aligned}
\mathcal{R}_{44+14j} &= \mathcal{R}_{43+14j} \\
&\cup\ \{l_{j,4} + 3 : hd^{2(j+1)+2}c \rightarrow Ab^{2(j+1)+3}c\} \quad , \\
\mathcal{E}_{44+14j} &= \{EeEegab^{2j+3}c = go,\ go = Eegab^{2j+3}c\} \ .
\end{aligned}$$

Using rule 15 the string $EeEegab^{2j+3}c$ can be reduced to $Eegab^{2j+3}c$. This gives

$$\begin{aligned}
\mathcal{R}_{45+14j} &= \mathcal{R}_{44+14j} \\
&\cup\ \{l_{j,4} + 4 : Eegab^{2j+3}c \rightarrow go\} \quad , \\
\mathcal{E}_{45+14j} &= \{go = Eegab^{2j+3}c\} \ .
\end{aligned}$$

Since go and $Eegab^{2j+3}c$ are joinable w.r.t. $\mathcal{R}_{45+14j}$, we have

$$\begin{aligned}
\mathcal{R}_{46+14j} &= (\bar{\mathcal{R}}_j - \{1 : X \rightarrow egad^{2j+2}c,\ l_{j,4} : Eegad^{2j+2}c \rightarrow go\}) \\
&\cup\ \{1 : X \rightarrow egab^{2j+3}c,\ l_{j,4} + 2 : ad^{2j+2}c \rightarrow o\} \\
&\cup\ \{l_{j,4} + 3 : hd^{2(j+1)+2}c \rightarrow Ab^{2(j+1)+3}c,\ l_{j,4} + 4 : Eegab^{2j+3}c \rightarrow go\} \quad , \\
\mathcal{E}_{46+14j} &= \emptyset \ .
\end{aligned}$$

Hence in the next step rule $l_{j,2} : ab^{2j+1}c \rightarrow o$ will be marked. Since there are no overlaps between rule $l_{j,2}$ and any of the rules of $\mathcal{R}_{46+14j}$ that has a label smaller than $l_{j,3}$, we obtain

$$\begin{aligned}
\mathcal{R}_{47+14j} &= \mathcal{R}_{46+14j} \quad , \\
\mathcal{E}_{47+14j} &= \emptyset \quad ,
\end{aligned}$$

where now all rules with a number smaller than $l_{j,3}$ are marked. Rule $l_{j,3}$ only overlaps with the rules 2, 5 and 11. The corresponding critical pairs are $\mathcal{E}_{48+14j} = \{gab^{2j+3}c = QAd^{2j+4}c, ab^{2j+3}c = qAd^{2j+4}c, Hb^{2(j+1)+3}c = WAd^{2j+4}c\}$. Normalizing the critical pair $gab^{2j+3}c = QAd^{2j+4}c$ yields the pair $gab^{2j+3}c = Eegad^{2j+4}c$. Since $>_{48+14j} = \succ_1$ and $gab^{2j+3}c \succ_1 Eegad^{2j+4}c$, the rule $l_{j,4} + 5 : gab^{2j+3}c \rightarrow Eegad^{2j+4}c$ will be generated. This new rule will be used to simplify the rules 1 and $l_{j,4} + 4$. In this way the sets

$$\begin{aligned}
\mathcal{R}_{49+14j} &= (\bar{\mathcal{R}}_j - \{1 : X \rightarrow egad^{2j+2}c,\ l_{j,4} : Eegad^{2j+2}c \rightarrow go\}) \\
&\cup\ \{l_{j,4} + 2 : ad^{2j+2}c \rightarrow o,\ l_{j,4} + 3 : hd^{2(j+1)+2}c \rightarrow Ab^{2(j+1)+3}c\} \\
&\cup\ \{l_{j,4} + 5 : gab^{2j+3}c \rightarrow Eegad^{2j+4}c,\ 1 : X \rightarrow egad^{2(j+1)+2}c\} \quad , \\
\mathcal{E}_{49+14j} &= \{ab^{2j+3}c = qAd^{2j+4}c,\ Hb^{2(j+1)+3}c = WAd^{2j+4}c,\ EeEegad^{2j+4}c = go\}
\end{aligned}$$

are obtained. Since $qAd^{2j+4}c \to qd^{2j+4}c \xrightarrow{*} qc \to o$, the rule $l_{j,4}+6 : ab^{2(j+1)+1}c \to o$ will be generated next. Thus, the rule $l_{j,4}+5 : gab^{2j+3}c \to Eegad^{2j+4}c$ will be simplified and we have

$$\begin{aligned}
\mathcal{R}_{50+14j} &= (\bar{\mathcal{R}}_j - \{1 : X \to egad^{2j+2}c,\ l_{j,4} : Eegad^{2j+2}c \to go\}) \\
&\cup\ \{l_{j,4}+2 : ad^{2j+2}c \to o,\ l_{j,4}+3 : hd^{2(j+1)+2}c \to Ab^{2(j+1)+3}c\} \\
&\cup\ \{1 : X \to egad^{2(j+1)+2}c,\ l_{j,4}+6 : ab^{2(j+1)+1}c \to o\} \quad , \\
\mathcal{E}_{50+14j} &= \{Hb^{2(j+1)+3}c = WAd^{2j+4}c,\ EeEegad^{2j+4}c = go,\ go = Eegad^{2j+4}c\} \ .
\end{aligned}$$

Normalizing the equation $Hb^{2(j+1)+3}c = WAd^{2j+4}c$ yields $Hb^{2(j+1)+3}c = Ad^{2(j+1)+4}c$. This gives

$$\begin{aligned}
\mathcal{R}_{51+14j} &= \mathcal{R}_{50+14j} \\
&\cup\ \{l_{j,4}+7 : Hb^{2(j+1)+3}c \to Ad^{2(j+1)+4}c\} \\
\mathcal{E}_{51+14j} &= \{EeEegad^{2j+4}c = go, go = Eegad^{2j+4}c\} \ .
\end{aligned}$$

By applying rule 15 the string $EeEegad^{2j+4}c$ can be reduced to $Eegad^{2j+4}c$. Thus, the rule $l_{j,4}+8 : Eegad^{2(j+1)+2}c \to go$ will be generated, and we have

$$\begin{aligned}
\mathcal{R}_{52+14j} &= (\bar{\mathcal{R}}_j - \{1 : X \to egad^{2j+2}c,\ l_{j,4} : Eegad^{2j+2}c \to go\}) \\
&\cup\ \{l_{j,4}+2 : ad^{2j+2}c \to o,\ l_{j,4}+3 : hd^{2(j+1)+2}c \to Ab^{2(j+1)+3}c\} \\
&\cup\ \{1 : X \to egad^{2(j+1)+2}c,\ l_{j,4}+6 : ab^{2(j+1)+1}c \to o\} \\
&\cup\ \{l_{j,4}+7 : Hb^{2(j+1)+3}c \to Ad^{2(j+1)+4}c,\ l_{j,4}+8 : Eegad^{2(j+1)+2}c \to go\} \quad , \\
\mathcal{E}_{52+14j} &= \{go = Eegad^{2j+4}c\} \ .
\end{aligned}$$

The critical pair $go = Eegad^{2j+4}c$ resolves trivially. Thus it holds:

$$\begin{aligned}
\mathcal{R}_{53+14j} &= \mathcal{R}_{52+14j} \quad , \\
\mathcal{E}_{53+14j} &= \emptyset \ .
\end{aligned}$$

Hence in the next step rule $l_{j,4}+2 : ad^{2j+2}c \to o$ will be marked. Since there are no overlaps between rule $l_{j,4}+2$ and any of the rules of $\mathcal{R}_{53+14j}$ that has a label smaller than $l_{j,4}+3$, we obtain

$$\begin{aligned}
\mathcal{R}_{54+14j} &= \mathcal{R}_{53+14j} \ = \ \bar{\mathcal{R}}_{j+1} \quad , \\
\mathcal{E}_{54+14j} &= \emptyset \quad ,
\end{aligned}$$

where now all rules except the rules $l_{j,4}+3, l_{j,4}+6, l_{j,4}+7$ and $l_{j,4}+8$ are marked. Thus our claim also holds for $j+1$. □