Seminar Report

Confluence of Non-Left-Linear TRSs via Relative Termination

Benjamin Höller
benjamin.hoeller@student.uibk.ac.at

22 February 2013

Supervisors: Univ.-Prof. Dr. Aart Middeldorp
Dipl.-Inf. Bertram Felgenhauer

Abstract

We present a method of Klein and Hirokawa which proves confluence of term rewriting systems. In contrast to Newman's famous Lemma, this method works also in absence of termination. The method splits the term rewriting system into two parts, the first part consisting of confluent rules only and the second part being relatively terminating over the first part. Furthermore, the two parts must have no overlaps between each other and the calculation of critical pairs is needed. After presenting the theorem and some examples, we show a step-by-step procedure which deterministically gives an answer about confluence of a given term rewriting system.

Contents

1 Introduction

Term rewriting systems (abbreviated TRSs) are an important formalism in logic, computer science and also in mathematics. By using TRSs one can model a system or a program algebraically and check its correctness. One of the most important properties of TRSs is confluence. Therefore, much effort is spent on developing new techniques which show confluence. Many of these new techniques are integrated in the confluence provers ACP [2], Saigawa [4] and CSI [9].
The most common technique to show this property is Newman's Lemma [6, Theorem 3]. This lemma requires termination, which is also a very important property of TRSs. As not every confluent TRS is terminating, Newman's Lemma is not always applicable. For example the following TRS T' is not terminating:

$$T' = \begin{cases} 1: & \mathsf{eq}(\mathsf{s}(n'), x : xs', x : ys') \to \mathsf{eq}(n', xs', ys') \\ 2: & \mathsf{eq}(n, xs, xs) \to \top \\ 3: & \mathsf{nats} \to \mathsf{0} : \mathsf{inc}(\mathsf{nats}) \\ 4: & \mathsf{inc}(x : xs) \to \mathsf{s}(x) : \mathsf{inc}(xs) \end{cases}$$

The TRS T' is not terminating because of the non-terminating rule 3. In this report we present the confluence criterion of Klein and Hirokawa which shows confluence of TRSs containing non-terminating parts with the help of relative termination.
This report is based mainly on [4] and some definitions are taken from [1]. We first introduce some preliminaries in Section 2, which we need to construct our confluence criterion in Section 3. There we present also a general algorithm for the confluence criterion, which we will improve with some optimizations to a step-by-step procedure. We will show some examples where this procedure gives a suitable answer about the confluence of the input TRS. We also prove confluence of the TRS T' shown above.

2 Preliminaries

In this section we give some basic definitions, which we need for the following sections.

Definition 2.1. We define *terms* inductively over a set $\mathcal{F}$ of *fixed-arity function symbols* and a set $\mathcal{V}$ of *variables*.

Definition 2.2. Let t be a term. We write $Pos_{\mathcal{F}}(t)$ for a set of *function positions* in t.

Note: We use sequences of natural numbers to express positions. The empty sequence stands for the root position ϵ.

Definition 2.3. Let p be a position and t be a term. Then $t(p)$ denotes the symbol of term t at position p and $t|_p$ abbreviates the subterm of t at position p. The replacement of a subterm $t|_p$ by another term s is abbreviated by $t[s]_p$.

Definition 2.4. Let t be a term. We write $\mathcal{V}ar(t)$ for the set of all variables occuring in t.

Definition 2.5. Let $l \approx r$ be an equation. We call $l \approx r$ a rewrite rule if:

1. l is not a variable
2. $\mathcal{V}ar(r) \subseteq \mathcal{V}ar(l)$.

We use the notation $l \to r$ for rewrite rules.

Definition 2.6. Let $l \approx r$ be an equation. We call $l \approx r$ an extended rewrite rule if l is not a variable. We use the notation $l \to r$.

Note: Every rewrite rule is also an extended rewrite rule, but not every extended rewrite rule is a rewrite rule.

Definition 2.7. A collection of rewrite rules is called a TRS and a collection of extended rewrite rules is called an $eTRS$.

Note: If R is a TRS, then R is also an eTRS, but if R is an eTRS, then R does not necessarily have to be a TRS.

Example 2.8. Let

$$R = \begin{cases} 1: & x \to \mathsf{f}(x) \\ 2: & \mathsf{g}(\mathsf{a}) \to \mathsf{a} \end{cases}$$

$$S = \begin{cases} 1: & \mathsf{f}(x,x) \to \mathsf{f}(x,y) \\ 2: & \mathsf{g}(\mathsf{a}) \to \mathsf{a} \end{cases}$$

$$T = \begin{cases} 1: & \mathsf{f}(x,x) \to x \\ 2: & \mathsf{f}(x,\mathsf{g}(y)) \to \mathsf{h}(x,y) \end{cases}$$

R is neither a TRS nor an eTRS because the left-hand side of rule 1 is a variable.

S is an eTRS because the left-hand sides of both rules are not variables. But S is not a TRS as the right-hand side of rule 1 has variables x and y and the left-hand side has only variable x.

T is a TRS and an eTRS, because the left-hand sides of both rules are not variables and the variables occuring in the right-hand sides of the rules are subsets of the variables occuring in their left-hand sides.

Definition 2.9. Let R be a TRS and p a position. We write $\xrightarrow{p}_R$ for a rewrite step on p with rewrite rules from R.

Definition 2.10. Let R be a TRS. We write $\downarrow_R$ for the join relation $\to_R^* \cdot \leftarrow_R^*$.

Definition 2.11. Let R be a TRS. We write R is terminating, if there exists no infinite rewrite sequence using $\to_R$.

Definition 2.12. Let R and S be two TRSs. We say R is relatively terminating over S if $\to_S^* \cdot \to_R \cdot \to_S^*$ is terminating. We use the notation: R/S is terminating.

Note: If TRS $R \cup S$ is terminating then follows that R is relatively terminating over S and S is relatively terminating over R.

Example 2.13. Let

$$R = \Big\{ 1: \quad \mathsf{a} \to \mathsf{b}$$

$$S = \Big\{ 1: \quad \mathsf{b} \to \mathsf{a}$$

$$T = \Big\{ 1: \quad \mathsf{a} \to \mathsf{f}(\mathsf{a})$$

R is not relatively terminating over S because, for example,

$$\mathsf{a} \to_R \mathsf{b} \to_S \mathsf{a}$$

is not terminating.

R is relatively terminating over T because $\to_T^* \cdot \to_R \cdot \to_T^*$ is terminating.

In the examples above it is easy to decide whether the relative termination condition holds, but in general that is not the case. Consider the following example.

Example 2.14. Let

$$R = \left\{ 1: \quad \mathsf{a} \to \mathsf{b} \right.$$

$$S = \left\{ 1: \quad \mathsf{c} \to \mathsf{f}(\mathsf{a}, \mathsf{c}) \right.$$

If we start from term a, we may come to the conclusion that the relative termination condition holds, because

$$\mathsf{a} \to_R \mathsf{b}$$

is terminating.

But, nevertheless, R is not relatively terminating over S, because

$$\underline{\mathsf{c}} \to_S \mathsf{f}(\mathsf{a}, \mathsf{c}) \to_R \mathsf{f}(\mathsf{b}, \underline{\mathsf{c}}) \to_S \mathsf{f}(\mathsf{b}, \mathsf{f}(\mathsf{a}, \mathsf{c}))$$

is not terminating.

Definition 2.15. A set of n equations is written in the form:

$$E = \{s_1 \approx t_1, \ldots, s_n \approx t_n\}$$

We call a set of equations E unifiable if a substition σ exists with $s_i\sigma = t_i\sigma$ for all i. The substitution σ is a unifier of E.

Note: The abbreviation E^{-1} denotes the same set of equations, but in reversed direction: $E^{-1} = \{t_1 \approx s_1, \ldots, t_n \approx s_n\}$.

Definition 2.16. Let E and S be sets of equalities. We call a set of equations E S-unifiable if a substition σ exists with $E\sigma \subseteq \leftrightarrow_S^*$. The substitution σ is called an S-unifier of E.

Definition 2.17. Let s, t be two terms without common variables and p a non-variable position in t. The term s *overlaps* on term t at position p if the subterm $t|_p$ unifies with s.

Definition 2.18. Let S be a eTRS, r, t be two terms without common variables and p a non-variable position in t. The term r *S-overlaps* on term t at position p if the subterm $t|_p$ S-unifies with r.

Note: The substitution σ has to fullfill $l_1\sigma \leftrightarrow_S^* l_2|_p\sigma$ for S-unification.

Definition 2.19. A critical pair is generated when there exists an overlap between the left-hand sides of two rewrite rules. Let $l_1 \to r_1$ (from a TRS S) and $l_2 \to r_2$ (from a TRS T) be two rewrite rules without any common variables and suppose l_1 has an overlap on l_2 at position p. Let σ be a most general unfier for l_1 and $l_2|_p$. The *critical pair* is the following:

$$\langle l_2[r_1]_p\sigma, r_2\sigma \rangle$$

$CP(S,T)$ is the set of all critical pairs originating from overlaps with $l_1 \to r_1$ in S on $l_2 \to r_2$ in T. If no rule in S overlaps on a rule in T, the set of critical pairs is empty.

Note: The definition is not symmetric: $CP(S,T) \neq CP(T,S)$.

Definition 2.20. Let σ and σ' be two S-unifiers of a set of equations E and X be the set of all variables in E. We say σ is more general than σ' if a substitution τ exists with $x\sigma' \leftrightarrow_S^* x\sigma\tau \ \forall x \in X$. We write $\sigma \lesssim_S^X \sigma'$.

Definition 2.21. Let $\mathcal{U}$ be a set of S-unifiers on a set of equations E. We say that $\mathcal{U}$ is *complete* if all S-unifiers of E are contained in $\mathcal{U}$ or there exists a more general element in U.
If the elements in $\mathcal{U}$ are also minimal with respect to $\lesssim_S^X$, then $\mathcal{U}$ is called minimal complete.

Definition 2.22. Let σ be a substitution and E a set of equations. We call σ an S-most general unifier (abbreviated S-mgu) of E if $\{\sigma\}$ is a minimal complete set of S-unifiers of E.

Note: If $S = \emptyset$, then the S-mgu is the same as the mgu.

Definition 2.23. Let S be an eTRS. An S-extended critical pair (abbreviated S-critical pair) exists if there is an S-overlap between the left-hand sides of two extended rewrite rules. Let $l_1 \to r_1$ (from an eTRS R) and $l_2 \to r_2$ (from an eTRS T) be two extended rewrite rules without any common variables and suppose l_1 has an S-overlap on l_2 at position p. Let σ be an S-unifier for l_1 and $l_2|_p$. The *S-critical pair* is the following:

$$\langle l_2[r_1]_p\sigma, r_2\sigma\rangle$$

We write ${}_R{\leftarrow}_S{\infty}{\rightarrow}_T$ for the set of all S-critical pairs originating from S-overlaps with $l_1 \to r_1$ in R on $l_2 \to r_2$ in T. If no rule in R S-overlaps on a rule in T, the set of S-critical pairs is empty.

Definition 2.24. A term t is called *linear* if no variable in t occurs more than once.

Definition 2.25. Let t be a term. $REN(t)$ replaces all variables in t by fresh variables.

Note: $REN(t)$ makes t linear.

Definition 2.26. Let R be a TRS. We write $\hat{R}$ for the eTRS $\{REN(l) \to r \mid l \to r \in R\}$.

Definition 2.27. Let R and S be two TRSs. We say that S is *strongly non-overlapping on* R if for every $l_1 \to r_1 \in \hat{R}$ and every $l_2 \to r_2 \in \hat{S}$, l_1 has no overlap on l_2.

Notation: We write $SNO(R,S)$ if S is strongly non-overlapping on R and R is strongly non-overlapping on S.

Example 2.28. Let

$$R = \left\{ 1 : \ \mathsf{s}(x, y) \to \mathsf{f}(x, y) \right.$$

$$S = \left\{ 1 : \ \mathsf{s}(x, x) \to \mathsf{a} \right.$$

$$T = \left\{ 1 : \ \mathsf{f}(\mathsf{a}) \to \mathsf{a} \right.$$

$SNO(R, S)$ does not hold because, after replacing the variables of both left-hand sides with fresh variables we get an overlap on the root position.

$SNO(R, T)$ holds because, after replacing the variables on the left-hand side of R with fresh variables, there is no overlap.

3 Confluence Criterion

In this section we present first the main theorem.

Theorem 3.1. *[4, Theorem 2]*
Suppose that S is confluent, R/S is terminating, and $SNO(R,S)$. The union $R \cup S$ of the TRSs is confluent if and only if ${}_R\!\leftarrow_S\!\!\times\!\!\rightarrow_R \subseteq \downarrow_{R\cup S}$.

When checking the four conditions of Theorem 3.1 in an automatic way the difficulty lies is verifying the last condition ${}_R\!\leftarrow_S\!\!\times\!\!\rightarrow_R \subseteq \downarrow_{R\cup S}$. The standard approach would be to compute a minimal complete set of S-unifiers for every possible S-critical pair. If the S-critical pairs are joinable, we have a guarantee for the joinability of all S-unifiers. The problem lies in the computation of the minimal complete sets. Depending on S the computation of such minimal complete sets varies and in the worst case they may not even exist for S-unifiable terms.

Example 3.2. Let

$$T = \Big\{ 1: \quad \mathsf{f}(x,y) \to \mathsf{f}(y,x) \qquad\qquad 2: \quad \mathsf{a} \to \mathsf{b}$$

We split the TRS T into two TRSs $R = \{2\}$ and $S = \{1\}$. Confluence of S follows from orthogonality. We can prove termination of R/S by using the termination tool $\mathsf{T_TT_2}$ [5]. $SNO(R,S)$ is established as there are no overlaps between S and R. We get ${}_R\!\leftarrow_S\!\!\times\!\!\rightarrow_R = \emptyset$ which is contained in $\downarrow_{R\cup S}$. Therefore, according to Theorem 3.1, T is confluent.

Note: This splitting is the only useful combination. If we set $R = \{1,2\}$ and $S = \emptyset$, termination of R/S fails, because rule 1 is a commutativity rule which occurs in R and such rules are always non-terminating. Therefore, also the combination $R = \{1\}$ and $S = \{2\}$ fails. To set $R = \emptyset$ and $S = \{1,2\}$ would be of no use as S still has to be confluent and we have no progress. As there are no more possible combinations to split T the suggested solution is the only suitable one.

As in this example there was no S-critical pair we can easily show confluence. But in general that's not the case. Consider the example from the introduction:

Example 3.3. Let

$$T' = \begin{cases} 1: & \mathsf{eq}(\mathsf{s}(n'), x : xs', x : ys') \to \mathsf{eq}(n', xs', ys') \\ 2: & \mathsf{eq}(n, xs, xs) \to \top \\ 3: & \mathsf{nats} \to \mathsf{0} : \mathsf{inc}(\mathsf{nats}) \\ 4: & \mathsf{inc}(x : xs) \to \mathsf{s}(x) : \mathsf{inc}(xs) \end{cases}$$

We split the TRS T' into two TRSs $R = \{1,2\}$ and $S = \{3,4\}$. Confluence of S follows from orthogonality. To show R/S is terminating we use again $\mathsf{T_TT_2}$. $SNO(R,S)$ is established as there are no overlaps between S and R, even if

the left-hand side of the rule in S has fresh variables. The first three conditions of Theorem 3.1 are satisfied, but we are not able to prove the last condition. There could be an S-overlap between rule 1 and rule 2 at the root position and in general it is undecidable to find a minimal complete set of S-unifiers.

Note: This splitting is the only useful combination. If we shift either rule of R to S, the condition $SNO(R, S)$ does not hold. If we shift either or both rules of S to R, the condition of relative termination fails. To set $R = \emptyset$ and $S = \{1, 2, 3, 4\}$ would be of no use as still S has to be confluent and we have no progress. As there are no more possible combinations to split T' the suggested solution is the most suitable.

Therefore, we are looking for a sufficient condition for the joinability of S-critical pairs. This we will demonstrate in Theorem 3.7 below.

Definition 3.4 (Strongly S-stable)**.**
We call a term t strongly S-stable if $\forall p \in Pos_{\mathcal{F}}(t)$ there exist no u and σ such that $t|_p\sigma \xrightarrow{*}_S \cdot \xrightarrow{\epsilon}_S u$.

Example 3.5. Let

$$S = \left\{ 1: \quad \mathsf{a} \to \mathsf{b} \right.$$

$$S' = \begin{cases} 1: & \mathsf{g}(x,x,x) \to \mathsf{f}(x) \\ 2: & \mathsf{a} \to \mathsf{b} \end{cases}$$

be two TRSs and let $t = \mathsf{g}(x, x, x)$ and $u = \mathsf{g}(x, \mathsf{b}, x)$ be two terms. In this example we check if t is strongly S-stable and if u is strongly S'-stable.

- The term t has the root symbol g, which is different from the root symbol a in the rewrite rule 1 of S and, because it is the only rewrite rule, we cannot get an overlap on the root position. Therefore, $\mathsf{g}(x, x, x)$ is strongly S-stable.
- The term u is not strongly S'-stable, because there is a rewrite step on the root position possible with $\sigma = \{x \mapsto \mathsf{a}\}$:

$$\mathsf{g}(x, \mathsf{b}, x)\sigma = \mathsf{g}(\mathsf{a}, \mathsf{b}, \mathsf{a}) \xrightarrow{1}_2 \mathsf{g}(\mathsf{b}, \mathsf{b}, \mathsf{a}) \xrightarrow{3}_2 \mathsf{g}(\mathsf{b}, \mathsf{b}, \mathsf{b}) \xrightarrow{\epsilon}_1 \mathsf{f}(\mathsf{b})$$

Lemma 3.6. *[4, Lemma 12]*
If $SNO(R, S)$, then l is strongly S-stable for all $l \to r \in R$.

Theorem 3.7. *[4, Theorem 15]*
Let S be a confluent TRS. An mgu of strongly S-stable terms s and t is an S-mgu of s and t.

We can use this sufficient condition to make progress in the previous example.

Example 3.8 (continued from Example 3.3)**.** According to the previous lemma we get strong S-stability of the left-hand sides of the two rules in R for free, because we already proved that $SNO(R, S)$ holds. We know that there is an overlap between rule 1 and rule 2 at the root position with the following mgu:

$$\sigma = \{n \mapsto \mathsf{s}(n'), xs \mapsto x : xs', ys' \mapsto xs'\}$$

We already checked confluence of S. According to Theorem 3.7 our mgu, which is derived from two strongly S-stable terms, is also an S-mgu. Thus, we get the following S-critical pair:

$$\langle \mathsf{eq}(n', xs', xs'), \top \rangle$$

This S-critical pair is joinable with the second rule. As there are no overlaps and no further S-overlaps, ${}_R\!\leftarrow_S\!\infty\!\rightarrow_R \subseteq \downarrow_{R \cup S}$ holds. Therefore, the TRS T' is confluent.

In general there is not only one suitable splitting, there could be more useful splittings where we get always the same result. Consider the following example.

Example 3.9. Let

$$L = \begin{cases} 1: & \mathsf{g}(x, x) \to \mathsf{h}(x) \\ 2: & \mathsf{h}(\mathsf{a}) \to \mathsf{d}(\mathsf{a}, \mathsf{a}) \\ 3: & \mathsf{g}(\mathsf{a}, \mathsf{a}) \to \mathsf{d}(\mathsf{a}, \mathsf{a}) \\ 4: & \mathsf{f}(x, y) \to \mathsf{f}(y, x) \end{cases}$$

We get a suitable answer about confluence of L if we split the TRS L into two TRSs $R = \{1, 2, 3\}$ and $S = \{4\}$. Confluence of S follows from orthogonality. To show R/S is terminating we use again $\mathsf{T{\top}T_2}$. $SNO(R, S)$ is established as there are no overlaps between S and R, even if the left-hand sides of the rules in R and S have fresh variables.
According to Lemma 3.6 we have shown already strong S-stability of the left-hand sides of the three rules in R because we proved $SNO(R, S)$. We know that there is an overlap between rules 1 and 2 at the root position with the following mgu:

$$\sigma = \{x \mapsto a\}$$

We already checked confluence of S, therefore, we can apply Theorem 3.7. Our mgu, which is derived from two strongly S-stable terms, is also an S-mgu. Thus, we get the following S-critical pair:

$$\langle \mathsf{h}(\mathsf{a}), \mathsf{d}(\mathsf{a}, \mathsf{a}) \rangle$$

This S-critical pair is joinable with rule 2. As there are no other overlaps and no further S-overlaps, ${}_R\!\leftarrow_S\!\infty\!\rightarrow_R \subseteq \downarrow_{R \cup S}$ holds. Therefore, the TRS L is confluent.

Note: We get also the same answer about confluence of L if we take the splitting

$R = \{1, 3\}$ and $S = \{2, 4\}$. Because confluence of S still follows from orthogonality, R/S is terminating, which can be proved again by $\mathsf{T_T T_2}$. $SNO(R, S)$ holds because we had no overlaps with the previous splitting and there are no overlaps between rule 2 and the rules in R. We get the same S-critical pair as with the previous splitting which is still joinable, because for the joinability of S-critical pairs it makes no difference whether the rule 2 is in R or in S. As there are no other overlaps and no further S-overlaps, we established confluence with this splitting as well.
The combination $R = \{2\}$ and $S = \{1, 3, 4\}$ will give us the same result. But with this splitting we do not gain much progress, because proving confluence of rules 1, 3 and 4 needs nearly the same effort as proving confluence of L. Therefore, it is not the best choice. Thus, when trying all possible splittings to get an answer about confluence, it is more efficient to start with the least number of rules in TRS S.
All other combinations would fail. We cannot shift the commutative rule 4 into R, because it is a non-terminating rule. If we shift rule 1 to S and rule 3 to R, or vice versa, the condition $SNO(R, S)$ does not hold. If we take $R = \emptyset$ and $S = \{1, 2, 3, 4\}$ we have made no progress.

We have considered some examples, but now we present a general algorithm for the fourth condition of the confluence criterion of Theorem 3.1.

Definition 3.10 (General algorithm)**.**
Input: TRSs R and S with conditions:

- $SNO(R, S)$
- R/S is terminating
- S is confluent

Output: An answer (*Yes* or *Maybe*) for confluence of $R \cup S$.

Algorithm: Look at all potential overlaps ($l_1 \to r_1 \in R$, p, $l_2 \to r_2 \in R$) with $p \in Pos_{\mathcal{F}}(l_2)$. We check if $l_1|_p$ and l_2 are unifiable:

1. If the two terms are unifiable, we found an mgu which, according to Lemma 3.6 and Theorem 3.7, is also an S-mgu. Therefore, an overlap by an mgu is also an S-overlap by an S-mgu and thus joinability of critical pairs implies joinability of S-critical pairs. As a consequence we check if the induced critical pair from the overlap is joinable.
   - If the critical pair is joinable, we check the next possible overlap.
   - If we cannot show joinability, return *Maybe*.
2. If the two terms are not unifiable, no mgu exists and therefore the sufficient condition is not fullfilled. We need a theorem prover to test S-unifiability of l_1 and $l_2|_p$. The theorem prover checks $S \models l_1 \approx l_2|_p$ on satisfiability.

- If the theorem prover returns satisfiable or no conclusive answer, then there could be an S-mgu and also an S-overlap, but we cannot be sure. Therefore, return *Maybe*.
- If the theorem prover returns unsatisfiable, then no S-mgu and no S-overlap exists. We check the next possible overlap.

If no overlap is left, return *Yes*.

We want to improve our general algorithm to get some "No" as answer. That can be done with the function $TCAP$. With this function we get a sufficient condition for checking unjoinability of S-critical pairs, which works similarly to check non-confluence of a TRS.

Definition 3.11 (TCAP)**.**
Let t be a term and R be a TRS.

$$TCAP_R(t) = \begin{cases} x, & \text{if } t \text{ is a variable} \\ x, & \text{if } t = f(t_1, \ldots, t_n) \text{ and } u \text{ unifies with lhs of a rule in } R \\ u, & \text{otherwise} \end{cases}$$

where x is a fresh variable and $u = f(TCAP_R(t_1), \ldots, TCAP_R(t_n))$.

Lemma 3.12. *[4, Lemma 19]*
Let $l_1 \to r_1$, $l_2 \to r_2 \in R$ and $p \in Pos_{\mathcal{F}}(l_2)$. If $l_1\sigma \overset{*}{\leftrightarrow}_R l_2|_p\sigma$, and $TCAP_R(r_2)$ and $TCAP_R(l_2[r_1]_p)$ do not unify, R is not confluent.

We can combine Lemma 3.12 and Definition 3.10 to a step-by-step procedure.

Definition 3.13 (Step-by-step procedure)**.**
Input: TRSs R and S with conditions:

- $SNO(R, S)$
- R/S is terminating
- S is confluent

Output: An answer (Yes, No or Maybe) for confluence of $R \cup S$.

Procedure: Let $l_1 \to r_1$ and $l_2 \to r_2$ be rules of TRS R and $p \in Pos_{\mathcal{F}}(l_2)$. We check every tuple $(l_1 \to r_1, p, l_2 \to r_2)$ in the following manner:

1. If $REN(l_1)$ and $REN(l_2|_p)$ are not syntactically unifiable, no S-overlap exists. Check the next tuple.

2.a If l_1 and $l_2|_p$ are syntactically unifiable with σ, we got an S-overlap:

   2.a1 If the S-critical pair is joinable, go to step 1 and check the next tuple.

   2.a2 If we cannot show joinability, check the syntactic unifiability of $TCAP_{R\cup S}(r_2\sigma)$ and $TCAP_{R\cup S}(l_2\sigma[r_1\sigma]_p)$. If they do not unify, return *No*, otherwise, return *Maybe*.

2.b If l_1 and $l_2|_p$ are not syntactically unifiable, check $S \models l_1 \approx l_2|_p$ on satisfiability by a theorem prover.

2.b1 If the formula is unsatisfiable, no S-overlap exists. Go to step 1 and check the next tuple.

2.b2 If the formula is satisfiable, check the syntactic unifiability of $TCAP_{R\cup S}(r_2)$ and $TCAP_{R\cup S}(l_2[r_1]_p)$. If they do not unify, return *No*, otherwise, return *Maybe*.

2.b3 If the theorem prover does not provide an answer, return *Maybe*.

If there are no remaining tuples left, return *Yes*.

Example 3.14. Let

$$U = \begin{cases} 1: & \mathsf{f}(x,x) \to \mathsf{a} \\ 2: & \mathsf{f}(y,\mathsf{g}(y)) \to \mathsf{b} \\ 3: & \mathsf{c} \to \mathsf{g}(\mathsf{c}) \end{cases}$$

We split the TRS U into the TRSs $R = \{1, 2\}$ and $S = \{3\}$. All other splittings are not suitable. If we shift either rule of R to S, the condition $SNO(R, S)$ does not hold. If we shift the rule of S into R, the condition of relative termination fails. To set $R = \emptyset$ is useless as we make no progress.
Considering the suggested splitting, confluence of S follows from orthogonality. We can prove termination of R/S by using the termination tool T$_{\text{T}}$T$_2$. $SNO(R, S)$ is established as there are no overlaps between S and R, even if the left-hand side of the rule in R has fresh variables. With R and S we have a suitable input for our step-by-step procedure.
We get $l_1 = \mathsf{f}(x, x)$ and $l_2 = \mathsf{f}(y, \mathsf{g}(y))$ with $p = \epsilon$.

1. $REN(l_1) = \mathsf{f}(x', x'')$ and $REN(l_2|_p) = \mathsf{f}(y', \mathsf{g}(y''))$ are syntactically unifiable, therefore, an S-overlap may exist.

2.b We have no mgu for the terms $\mathsf{f}(x, x)$ and $\mathsf{f}(y, \mathsf{g}(y))$, therefore, they are not syntactically unifiable. We check with the theorem prover Waldmeister [3] if $S \models l_1 \approx l_2|_p$ is satisfiable.

2.b2 As the formula is satifiable we check syntactic unifiability of b and a. These two terms do not unify, therefore, return *No*.

Thus, U is not confluent.

4 Conclusion

We presented a method to show confluence of TRSs with non-terminating rewrite rules. Other methods, like Newman's Lemma, cannot handle this kinds of TRSs. The key point is to split the TRS into two parts, a terminating part and a potentially non-terminating part.
For the implementation of these method we identified two main challenges. The first one is the calculation of the S-critical pairs and the joinability check. For this a suitable theorem prover is needed, where we suggest Waldmeister [3], E [8] or Vampire [7]. The second main challenge is the efficient splitting of the input term rewriting system. In [4] no possible solution is demonstrated. According to their paper they try all possible combinations. We suggest to make some preprocessing. First, we calculate all possible splitting combinations, but before checking the conditions of our confluence criteria on every splitting, we check our input TRS for commutativity rules. If a rule α is a commutativity rule, we know that α is never terminating and we can remove all the splitting combinations where α is part of S. After filtering we check the conditions on the remaining partition.

References

[1] T. Aoto and Y. Toyama. A reduction-preserving completion for proving confluence of non-terminating term rewriting systems. *Logical Methods in Computer Science*, 8(1):1–29, 2012.

[2] T. Aoto, J. Yoshida, and Y. Toyama. Proving confluence of term rewriting systems automatically. In *Proceedings of the 20th International Conference on Rewriting Techniques and Applications*, RTA '09, pages 93–102, Berlin, Heidelberg, 2009. Springer-Verlag.

[3] T. Hillenbrand, A. Buch, R. Vogt, and B. Löchner. WALDMEISTER - High-Performance Equational Deduction. *J. Autom. Reason.*, 18(2):265–270, 1997.

[4] D. Klein and N. Hirokawa. Confluence of non-left-linear TRSs via relative termination. In *Proceedings of the 18th international conference on Logic for Programming, Artificial Intelligence, and Reasoning*, LPAR'12, pages 258–273, Berlin, Heidelberg, 2012. Springer-Verlag.

[5] M. Korp, C. Sternagel, H. Zankl, and A. Middeldorp. Tyrolean termination tool 2. In *Proceedings of the 20th International Conference on Rewriting Techniques and Applications*, volume 5595 of *Lecture Notes in Computer Science*, pages 295–304, Brasilia, 2009.

[6] M. H. A. Newman. On theories with a combinatorial definition of "Equivalence". In *Annals of Mathematics*, pages 223–243, 1942.

[7] A. Riazanov and A. Voronkov. The design and implementation of VAMPIRE. *AI Commun.*, 15(2,3):91–110, 2002.

[8] S. Schulz. E - A Brainiac Theorem Prover. *AI Commun.*, 15(2,3):111–126, 2002.

[9] H. Zankl, B. Felgenhauer, and A. Middeldorp. CSI – A Confluence Tool. In *Proceedings of the 23rd International Conference on Automated Deduction*, volume 6803 of *Lecture Notes in Artificial Intelligence*, pages 499–505, Wroclaw, 2011.