Universitat Politècnica de València
Departamento de Sistemas Informáticos y Computación

Máster Universitario en Ingeniería y Tecnología de Sistemas Software

Master Thesis

CONFident: a tool for the analysis of confluence of rewrite systems

Candidate:
Miguel Vítores

Supervisor:
Salvador Lucas, Raúl Gutiérrez

Academic Year 2021/2022

UNIVERSITAT POLITÈCNICA DE VALÈNCIA

Departamento de Sistemas Informáticos y Computación
Universitat Politècnica de València
Camino de Vera, s/n
46022 Valencia
España

Contents

Abstract

CONFident is a tool that proves confluence, one of the most important properties in rewriting systems' analysis. For this purpose it makes use of some tools such as infChecker or MU-TERM. For instance, infChecker is crucial for proving joinability of critical pairs and feasibility of conditions in CTRSs, among other tasks; MU-TERM is the termination prover. Termination plays an important role in rewrite systems. CONFident supports several types of systems, including TRSs, CS-TRSs and JOIN, ORIENTED and SEMI-EQUATIONAL CTRSs. Specifically for CTRSs, CONFident has obtained very good results.

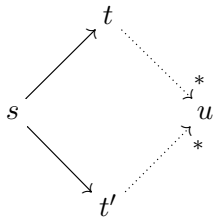
CONFident es una herramienta que demuestra la confluencia, una de las propiedades más importantes en el análisis de sistemas de reescritura. Para ello hace uso de algunas herramientas como infChecker o MU-TERM. Por ejemplo, infChecker es crucial para demostrar la joinability de pares críticos y la feasibility de condiciones en CTRSs, entre otras tareas; MU-TERM es la herramienta que demuestra la terminación. La terminación tiene gran importancia en sistemas de reescritura. CONFident es compatible con diversos tipos de sistemas, como pueden ser los TRSs, CS-TRSs y CTRSs de tipo JOIN, ORIENTED y SEMI-EQUATIONAL. Ha quedado demostrado que los problemas más apropiados para CONFident son los de tipo CTRS para los cuales ha obtenido unos resultados competitivos.

CONFident és una eina que demostra la confluència, una de les propietats més importants en l'anàlisi de sistemes de reescriptura. Per a això fa ús d'algunes eines com infChecker o MU-TERM. Per exemple, infChecker és crucial per a demostrar la joinability de parells crítics i la feasibility de condicions en CTRSs, entre altres tasques; MU-TERM és l'eina que demostra la terminació. La terminació té gran importància en sistemes de reescriptura. CONFident és compatible amb diversos tipus de sistemes, com poden ser els TRSs, CS-TRSs i CTRSs de tipus JOIN, ORIENTED i SEMI-EQUATIONAL. Ha quedat demostrat que els problemes més apropiats per a CONFident són els de tipus CTRS, aconseguint uns resultats competitius.

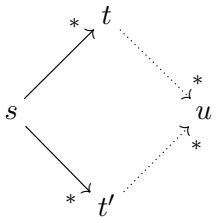
1

Introduction

Confluence is an abstract property of binary relations $\rightarrow$ (often called *reduction relations*) which guarantees that whenever an abstract object s can be reduced in zero or more steps to either t or t' (written $s \rightarrow^* t$ and $s \rightarrow^* t'$), both t and t' can be reduced to a common expression u (written $t \rightarrow^* u$ and $t' \rightarrow^* u$; then, we often say that t and t' are *joinable*). There also is a weaker property related to confluence, which is called *local* confluence, where only a *single* reduction is allowed on s to reduce it into t or t' (i.e. $s \rightarrow t$ and $s \rightarrow t'$) but the same joinability condition on t and t' is required. Confluence and local confluence are often defined by the commutation of the following diagrams:



Local confluence



confluence

If $\rightarrow$ is terminating, then local confluence and confluence coincide (this result is often called *Newman's Lemma*, see, e.g., [2, Lemma 2.7.2]).

Confluence is one of the most important properties of term rewriting systems [2] and that is because (see [2]):

1. It ensures that a computation-of-normal-forms semantics can be adopted, which means that, given a term s, *at most* a normal form t (i.e., a term which cannot be further reduced) of s can be obtained.

2. It ensures that two divergent computations can always join in the future. This makes the implementation of rewriting-based languages easier, or less dependent on specific strategies to implement reductions.

3. It is also essential for implementing decision procedures for equational theories when dealing with equational reasoning.

Confluence has been investigated in several formalisms of rewriting such as first-order rewriting [2], lambda-calculi [11], higher-order rewriting [31], constrained rewriting [1], and conditional rewriting [26]. Confluence is *undecidable*, which means that, in general, no algorithm is able to prove or disprove confluence of a reduction relation associated to a (variant of a) rewrite system. In some special cases, though, this can be done. Indeed, during the last years, a lot of research in confluence has focused on the development of automatable techniques and tools for proving confluence. Interesting consequences of this trend have been the development of the *Confluence Competition* (CoCo [23]) and the platform CoCoWeb [12], where tools participating in CoCo can be freely used and compared in a unified framework.

In this master thesis we describe CONFident, a new confluence tool which can be used through the web application available here

`http://zenon.dsic.upv.es/confident/`

The tool implements the logical approach to prove confluence developed in [10, 21], which transforms proofs of confluence problems into a collection of other problems which are then treated by specialized tools which are able to deal with them. For instance, the analysis of local confluence of conditional rewrite systems involves the analysis of joinability of conditional critical pairs, which amounts at proving whether two terms are joinable (see above), possibly requiring further checks on the conditional part of the critical pair to prove or disprove its infeasibility [8].

Thus, for the purpose of checking confluence, besides implementing specific results guaranteeing confluence of different variants of rewriting (e.g., term rewriting, conditional rewriting, and context-sensitive rewriting [17]), CONFident also implements the necessary interconnection of tools so that specialized tools like infChecker [8], MU-TERM [9], Prover9 [22], and Fort [28] are used to solve auxiliary proof conditions. For instance,

- The use of infChecker is crucial as it is able to prove infeasibility of conditional rules (which are then discarded from the analysis) and critical pairs (which can be used to prove local confluence) among other interesting tasks discussed below.

- With MU-TERM, CONFident can use Newman's Lemma to prove confluence if local confluence has been previously proved and the result of the termination problem is YES.
- Prover9 is used by infChecker in some cases, but CONFident also makes direct calls to it for particular uses, for instance to obtain direct proofs of joinability of conditional critical pairs.
- Fort is used by CONFident to check the *weak-normalization* property (ensuring that all terms have a normal form), which is important to use some specific results for proving confluence of conditional rewrite systems.

CONFident is the first tool which is able to deal with join and semi-equational CTRSs. Furthermore, it was the winner of the CTRS category of the 2021 edition of the Confluence Competition (CoCo 2021).[1] It is also the first tool which can be used to prove confluence of CS-TRSs.

The structure of the thesis is as follows:

- Section 2 introduces the general main definitions and notations used along the thesis.
- Section 3 summarizes the theoretical results about confluence of different variants of rewrite systems which are implemented in CONFident.
- Section 4 provides the details about the mechanization of confluence proofs implemented in our tool.
- Section 5 provides an experimental evaluation of the tool.
- Section 6 concludes and discusses future work.

[1] `http://project-coco.uibk.ac.at/2021/results.php`

2

Preliminaries

Abstract reduction relations Given a binary relation $\mathsf{R} \subseteq A \times A$ on a set A, we often write $a \,\mathsf{R}\, b$ instead of $(a, b) \in \mathsf{R}$. (and say a R-reduces to b or that b is a direct R-successor of a) Given two relations $\mathsf{R}, \mathsf{R}' \subseteq A \times A$, their *composition* is defined by $\mathsf{R} \circ \mathsf{R}' = \{(a, b) \in A \times A \mid (\exists c) a \,\mathsf{R}\, c \wedge c \,\mathsf{R}'\, b\}$. Also, for all $n \in \mathbb{N}$, the n-fold composition R^n of R is defined by $\mathsf{R}^0 = \{(x, x) \mid x \in A\}$ and $\mathsf{R}^n = \mathsf{R} \circ \mathsf{R}^{n-1}$ if $n > 0$. The *transitive* closure of R is denoted by R^+, and its *reflexive and transitive* closure by R^*. The relation R is *finitely branching* if for all $a \in A$, the set $\{a \,\mathsf{R}\, b \mid b \in A\}$ of direct successors of a is finite. An element $a \in A$ is *irreducible* (or an R-*normal form*), if there exists no b such that $a \,\mathsf{R}\, b$. We say that b is an R-normal form of a (written $a \,\mathsf{R}^!\, b$), if $a \,\mathsf{R}^* b$ and b is an R-normal form. We also say that a is R-normalizing, i.e., a has an R-normal form. Also, R is *weakly normalizing* if every $a \in A$ has an R-normal form. Given $a \in A$, if there is no infinite sequence $a = a_1 \;\mathsf{R}\; a_2 \;\mathsf{R}\; \cdots \;\mathsf{R}\; a_n \;\mathsf{R} \cdots$, then a is R-*terminating* (or *well-founded*; [1]); R is *terminating* if a is R-terminating for all $a \in A$. We say that $a, b \in A$ are R-joinable if there is $c \in A$ such that $a \,\mathsf{R}^* c$ and $b \,\mathsf{R}^* c$. We say that R is (locally) *confluent* if, for every $a, b, c \in A$, whenever $a \,\mathsf{R}^* b$ and $a \,\mathsf{R}^* c$ (resp. $a \,\mathsf{R}\, b$ and $a \,\mathsf{R}\, c$), there exists $d \in A$ such that $b \,\mathsf{R}^* d$ and $c \,\mathsf{R}^* d$.

Theorem 1 (Newman´s Lemma) *A terminating* R *is locally confluent if and only if it is confluent. [2, Lemma 2.7.2]*

Signatures, terms, positions We use the standard notations in term rewriting (see, e.g., [26]). In this paper, $\mathcal{X}$ denotes a countable set of *variables* and $\mathcal{F}$ denotes a *signature*, i.e., a set of *function symbols* $\{f, g, \ldots\}$ (disjoint from $\mathcal{X}$), each with a fixed *arity* given by a mapping $ar : \mathcal{F} \to \mathbb{N}$. The set of terms built from $\mathcal{F}$ and $\mathcal{X}$ is $\mathcal{T}(\mathcal{F}, \mathcal{X})$. The set of *ground* terms (i.e., terms without variable occurrences)

[1] See [26, Definition 2.1.1] and the paragraph below this definition for a clarifying discussion about the use of 'well-founded' and 'terminating' in Mathematics and Computer Science.

is denoted $\mathcal{T}(\mathcal{F})$. The symbol labeling the root of t is denoted as $root(t)$. The set of variables occurring in t is $\mathcal{V}ar(t)$. By abuse of notation, we use $\mathcal{V}ar$ also with sequences of terms or other expressions to denote the set of variables occurring in them. Terms are viewed as labeled trees in the usual way. *Positions* $p, q, \ldots$ are represented by chains of positive natural numbers used to address subterms $t|_p$ of t. The *set of positions* of a term t is $\mathcal{P}os(t)$ with $p \in \mathcal{P}os = \{\Lambda\} \cup \{i.q \mid i \in \mathbb{N}_{>0} \wedge q \in \mathcal{P}os\}$ and $\mathcal{P}os(t) = \{\Lambda\}$ if $t \in \mathcal{X}$ and $\mathcal{P}os(t) = \{\Lambda\} \cup \bigcup_{1 \leq i \leq k} i.\mathcal{P}os(t_i)$ if $t = f(t_1, \ldots, t_k)$. The length of a position p is $|p|$. The subterm of t at position $p \in \mathcal{P}os(t)$ is $t|_p$: $t|_\Lambda = t$ and $f(t_1, \ldots, t_k)|_{i.p} = t_i|_p$. The depth of subterm $s = t|_p$ is the length $|p|$ of p. $\mathcal{P}os_{\mathcal{F}}(t)$ is the set of positions of *nonvariable subterms* in t: $\mathcal{P}os_{\mathcal{F}}(t) = \{p \in \mathcal{P}os(t) \mid root(t|_p) \in \mathcal{F}\}$. A subterm replacement $t[s]_p$ means that the term t where $t|_p$ has been replaced by the term s: $t[s]_\Lambda = s$ and $f(t_1, \ldots, t_i, \ldots, t_k)[s]_{i.p} = f(t_1, \ldots, t_i[s]_p, \ldots, t_k)$

Linearity A term t is linear if it has no repetition in variables, i.e., the term $t_1 = f(x, y)$ with $\mathcal{V}ar(t_1) = \{x, y\}$ is linear, and the term $t_2 = g(x, h(x), y)$ with $\mathcal{V}ar(t_2) = \{x, y\}$ is not linear.

Unification A substitution σ is a mapping from variables into terms which is homomorphically extended to a mapping from terms to terms. A term l matches t if there is a substitution σ such that $t = \sigma(l)$. Two terms s and t unify if there is a substitution σ such that $\sigma(s) = \sigma(t)$. If $\sigma(s) = \sigma(t)$ for a substitution σ, then σ is called a *unifier* of s and t. A *most general unifier* (or just *mgu*) of s and t is a unifier σ such that for every unifier τ there exists a substitution χ with $\chi \circ \sigma = \tau$. If two terms are unifiable, then they have a most general unifier that is unique except for variable renaming.

3

Confluence

In this section a summary of the definitions and theorems used and implemented in the tool is presented.

3.1 Term Rewriting Systems

A rewrite rule is an ordered pair, written $\ell \to r$, where $\ell, r \in \mathcal{T}(\mathcal{F}, \mathcal{X})$ are called the left- and right-hand sides (lhs and rhs for short), respectively, such that $\ell \notin \mathcal{X}$ (the left-hand-side is not a variable), and $\mathcal{V}ar(r) \subseteq \mathcal{V}ar(\ell)$ (there is no extra variable in the right-hand rule).

A rewrite rule $\ell \to r$ is *left-linear* if ℓ is a linear term, *right-linear* if r is a linear term, *linear* if both ℓ and r are linear terms, *collapsing* if $r \in \mathcal{X}$ (right-hand side is a variable).

A *Term Rewriting System (TRS)* is a pair $\mathcal{R} = (\mathcal{F}, R)$ such that $\mathcal{F}$ is a signature and R is a set of rewrite rules over the signature $\mathcal{F}$.

An instance $\sigma(\ell)$ of the left-hand side ℓ of a rule $\ell \to r$ is called a *redex* or *reducible expression* of the rule. Given a TRS $\mathcal{R} = (\mathcal{F}, R)$ and $t \in \mathcal{T}(\mathcal{F}, \mathcal{X})$, the set of *redex positions* $\mathcal{P}os_{\mathcal{R}}(t)$ of t w.r.t. $\mathcal{R}$ is $\mathcal{P}os_{\mathcal{R}}(t) = \{p \in \mathcal{P}os(t) \mid \exists \ell \to r \in R, \sigma \in \mathit{Subst}(\mathcal{T}(\mathcal{F}, \mathcal{X})), t|_p = \sigma(\ell)\}$.

A TRS $\mathcal{R} = (\mathcal{F}, R)$ is called left-linear, right-linear or linear if each rule $\ell \to r$ has the corresponding property. On the other hand, $\mathcal{R}$ is called collapsing if at least one of the rules $\ell \to r \in R$ has the corresponding property.

Given a TRS $\mathcal{R} = (\mathcal{F}, R)$ we define the *one-step rewrite relation* $\to_{\mathcal{R}}$ (or just $\to$ if $\mathcal{R}$ is clear from the context) as the set of all possible rewriting steps over terms $\mathcal{T}(\mathcal{F}, \mathcal{X})$: A term s *rewrites* to t at position p (written $s \overset{p}{\hookrightarrow} t$ and called a rewriting step) if there is a position p such that $s|_p = \sigma(\ell)$ for some rule $\ell \to r$ and substitution σ, and $t = s[\sigma(r)]_p$. The *reflexive* and *transitive* closure $\to^*_{\mathcal{R}}$ (or just

$\to^*$) of $\to_\mathcal{R}$ is called the *rewrite relation* associated to $\mathcal{R}$. Let $\mathcal{R} = (\mathcal{F}, R)$ be a TRS and $t \in \mathcal{T}(\mathcal{F}, \mathcal{X})$. We say that t is an $\mathcal{R}$-*normal form* (or just a *normal form* if $\mathcal{R}$ is clear from the context) if it is a $\to$-normal form, i.e., no rewriting step is possible on t (equivalently, if t contains no redex of $\mathcal{R}$)

Symbols $f \in \mathcal{D} = \{root(\ell) \mid \ell \to r\}$ are called *defined symbols* and symbols $c \in \mathcal{C} = \{\mathcal{F} \setminus \mathcal{D}\}$ are called *constructor symbols*. Terms in $\mathcal{T}(\mathcal{C}, \mathcal{X})$ are called *constructor terms*; clearly, they are normal forms. Two terms are called joinable if they are $\to_\mathcal{R}$-joinable.

3.1.1 Confluence of TRSs

Confluence, as an undecidable property of TRSs, has some challenges. In this subsection, we will address some results concerning confluence, along with examples, which will try to clarify the concepts using systems that CONFident is able to give a response to, sometimes when other tools can not.

A TRS $\mathcal{R}$ is (locally) confluent if $\to_\mathcal{R}$ is (locally) confluent.

In the following, we assume that every variable in the involved rules, if any, was transformed into a fresh variable before trying to calculate possible critical pairs. In order to analyze *(local-)confluence* of TRSs an essential notion is the following.

Definition 2 (Critical pair) *With the rules $rl_1 = \ell_1 \to r_1$ and $rl_2 = \ell_2 \to r_2$ of a TRS $\mathcal{R}$ we define a critical pair*

$$\langle \sigma(\ell_1)[\sigma(r_2)]_p, \sigma(r_1) \rangle$$

if there is a non-variable position $p \in \mathcal{P}os_\mathcal{F}(\ell_1)$ (called critical or overlapping position) such that $\ell_1|_p$ and ℓ_2 unify with mgu σ. A case is excluded and it is when $rl_1 = rl_2$ and $p = \Lambda$.

A CP $\langle \sigma(l_1)[\sigma(r_2)]_p, \sigma(r_1) \rangle$ *is an overlay if* $p = \Lambda$

A CP $\langle s, t \rangle$ *is trivial if* $s = t$. *[16]*

It is also worth it to mention that a critical pair represents a *divergence point* in the rewrite one-step relation of the TRS $\mathcal{R}$ in the sense that the same expression can be rewritten in different and overlapping ways.

Example 3

Consider the one-rule TRS $\mathcal{R}$

$$f(f(x,a),a) \quad \to \quad b \tag{3.1}$$

Note that $\ell = f(f(x,a),a)$ unifies with $\ell'|_1 = f(x',a)$, with mgu $\sigma = \{x' \to f(x,a)\}$. Then we have a critical pair $\langle f(b,a), b\rangle$. That is because $\langle \sigma(f(f(x,a)))[\sigma(b)]_1, \sigma(b)\rangle = \langle f(f(f(x,a),a))[b]_1, b\rangle = \langle f(b,a), b\rangle$.

Definition 4 (Convergence of critical pairs) *A critical pair $\langle s,t\rangle$ of a TRS $\mathcal{R}$ is convergent if $s \to^*_{\mathcal{R}} u, t \to^*_{\mathcal{R}} u$ for some term u.*

Example 5 (COPS 111.trs)

$$a \quad \to \quad b \tag{3.2}$$
$$a \quad \to \quad c \tag{3.3}$$
$$a \quad \to \quad e \tag{3.4}$$
$$b \quad \to \quad d \tag{3.5}$$
$$c \quad \to \quad a \tag{3.6}$$
$$d \quad \to \quad a \tag{3.7}$$
$$d \quad \to \quad e \tag{3.8}$$
$$g(x) \quad \to \quad h(a) \tag{3.9}$$
$$h(x) \quad \to \quad e \tag{3.10}$$

This example has 4 joinable critical pairs:

$$\begin{aligned}
\langle c,e\rangle \quad &, \quad c \to_{3.6} a \to_{3.4} e \\
\langle b,e\rangle \quad &, \quad b \to_{3.5} d \to_{3.7} a \to_{3.4} e \\
\langle b,c\rangle \quad &, \quad c \to_{3.6} a \to_{3.2} b \\
\langle a,e\rangle \quad &, \quad a \to_{3.4} e
\end{aligned}$$

Definition 6 (Orthogonality of TRSs) *A left-linear TRS $\mathcal{R}$ is orthogonal if it has no critical pairs; almost orthogonal if all critical pairs are trivial overlays; and weakly orthogonal if all critical pairs are trivial.*

3.1.2 Confluence criteria

Some well-known results of confluence for TRSs are now presented.

Theorem 7 (Huet & Lévy's Theorem) [14, 15] *Weakly orthogonal TRSs are confluent.*

This means that if all the critical pairs of $\mathcal{R}$ are trivial (or there are no critical pairs, which is part of the definition of orthogonality 6) and $\mathcal{R}$ is *left-linear* , then $\mathcal{R}$ is confluent. It is a syntactical condition which does not depend on termination, so it can be used to solve a lot of problems very fast.

Example 8 (First three rules of COPS 55.trs)

$$Ap(Ap(Ap(S,x),y),z) \rightarrow Ap(Ap(x,z),Ap(y,z)) \quad (3.11)$$

$$Ap(Ap(K,x),y) \rightarrow x \quad (3.12)$$

$$Ap(I,x) \rightarrow x \quad (3.13)$$

Example 8 has no critical pairs and is left-linear, then it is orthogonal, consequently, weakly orthogonal, therefore we can say that it is confluent without a termination proof needed by Huet & Lévy's Theorem.

Theorem 9 (Huet's Theorem) [13, Lemma 3.1] *A TRS is locally confluent if and only if all its critical pairs are convergent.*

Huet's Theorem, together with Newman´s Lemma can be used to prove confluence of a TRS $\mathcal{R}$ whenever the critical pairs of the TRS are all joinable and $\mathcal{R}$ is terminating. Trivial CPs converge by definition.

Example 10 (Last rule of COPS 55.trs)

$$Dh(z,z) \rightarrow z \quad (3.14)$$

The TRS $\mathcal{R}$ in example 10 has no critical pairs. However, it is not left-linear, thus it is not weakly orthogonal. Huet & Lévy's result cannot be used to prove confluence of $\mathcal{R}$. However, by Theorem 9, it is locally confluent. We can prove $\mathcal{R}$ terminating with MU-TERM. Thus, it is confluent by Newman´s Lemma.

Another interesting property is *strong confluence*, which implies *confluence* without termination.

Definition 11 (Strong joinability) *Two terms t_1 and t_2 are strongly joinable if, for some terms u_1 and u_2, we have: $t_1 \to^= u_1, t_2 \to^* u_1$ and $t_2 \to^= u_2, t_1 \to^* u_2$*

Definition 12 (Strong confluence) [13] *A TRS $\mathcal{R}$ is strongly confluent if it is linear and all critical pairs $\langle s, t\rangle$ are strongly joinable.*

Theorem 13 (Use of strong confluence to demonstrate confluence) [13] *Strongly confluent TRSs are confluent.*

Example 14 (COPS 163.trs)

$$+(+(x,y),z) \quad \to \quad +(x,+(y,z)) \tag{3.15}$$
$$+(x,+(y,z)) \quad \to \quad +(+(x,y),z) \tag{3.16}$$

This TRS is not orthogonal as it has 5 CPs:

$$\langle s_1, t_1\rangle \quad = \quad \langle +(x,+(x,+(y,z))), +(+(x,+(x,y)),z)\rangle \tag{3.17}$$
$$\langle s_2, t_2\rangle \quad = \quad \langle +(x,+(+(x,y),z)), +(+(x,x),+(y,z)\rangle \tag{3.18}$$
$$\langle s_3, t_3\rangle \quad = \quad \langle +(x,+(y,+(y,z))), +(+(+(x,y),y),z)\rangle \tag{3.19}$$
$$\langle s_4, t_4\rangle \quad = \quad \langle +(+(x,+(y,z)),z), +(+(x,y),+(z,z))\rangle \tag{3.20}$$
$$\langle s_5, t_5\rangle \quad = \quad \langle +(+(+(x,y),z),z), +(x,+(+(y,z),z))\rangle \tag{3.21}$$

It is not terminating:

$$+(+(x,y),z) \to_{3.15} +(x,+(y,z)) \to_{3.16} +(+(x,y),z) \to \ldots$$

Thus, neither Huet's, nor Huet & Levy's results apply. However, it is strongly confluent, as we have:

$\mathsf{CP}_{3.17}$

$$
\begin{aligned}
+(x,+(x,+(y,z))) &\to^{=} +(x,+(x,+(y,z)))\\
+(+(x,+(x,y)),z) &\to_{3.15} +(x,\underline{+(+(x,y),z)}) \to_{3.15} +(x,+(x,+(y,z)))\\
+(+(x,+(x,y)),z) &\to^{=} +(+(x,+(x,y)),z)\\
+(x,\underline{+(x,+(y,z))}) &\to_{3.16} +(x,+(+(x,y),z)) \to_{3.16} +(+(x,+(x,y)),z)
\end{aligned}
$$

$\mathsf{CP}_{3.18}$

$$
\begin{aligned}
+(x,+(+(x,y),z)) &\to^{=} +(x,+(+(x,y),z))\\
+(+(x,x),+(y,z)) &\to_{3.15} +(x,\underline{+(x,+(y,z))}) \to_{3.16} +(x,+(x,+(y,z)))\\
+(+(x,x),+(y,z)) &\to^{=} +(+(x,x),+(y,z))\\
+(x,\underline{+(+(x,y),z)}) &\to_{3.15} +(x,+(+(x,y),z)) \to_{3.16} +(+(x,x),+(y,z))
\end{aligned}
$$

$\mathsf{CP}_{3.19}$

$$
\begin{aligned}
+(x,+(y,+(y,z))) &\to^{=} +(x,+(y,+(y,z)))\\
+(+(+(x,y),y),z) &\to_{3.15} +(+(x,y),+(y,z)) \to_{3.15} +(x,+(y,+(y,z)))\\
+(+(+(x,y),y),z) &\to^{=} +(+(+(x,y),y),z)\\
+(x,+(y,+(y,z))) &\to_{3.16} +(+(x,y),+(y,z)) \to_{3.16} +(+(+(x,y),y),z)
\end{aligned}
$$

$\mathsf{CP}_{3.20}$

$$
\begin{aligned}
+(+(x,+(y,z)),z) &\to^{=} +(+(x,+(y,z)),z)\\
+(+(x,y),+(z,z)) &\to_{3.16} +(\underline{+(+(x,y),z)},z) \to_{3.15} +(+(x,+(y,z)),z)\\
+(+(x,y),+(z,z)) &\to^{=} +(+(x,y),+(z,z))\\
+(\underline{+(x,+(y,z))},z) &\to_{3.16} +(+(+(x,y),z),z) \to_{3.15} +(+(x,y),+(z,z))
\end{aligned}
$$

$\mathsf{CP}_{3.21}$

$$
\begin{aligned}
+(+(+(x,y),z),z) &\to^{=} +(+(+(x,y),z),z)\\
+(x,+(+(y,z),z)) &\to_{3.16} +(\underline{+(x,+(y,z))},z) \to_{3.16} +(+(+(x,y),z),z)\\
+(x,+(+(y,z),z)) &\to^{=} +(x,+(+(y,z),z))\\
+(\underline{+(+(x,y),z)},z) &\to_{3.15} +(+(x,+(y,z)),z) \to_{3.15} +(x,+(+(y,z),z))
\end{aligned}
$$

Since $\mathcal{R}$ is *linear* and it has *strongly joinable* critical pairs, by Definition 12 it is strongly confluent. By Theorem 13, it is confluent.

3.1.3 Modularity of confluence

By using *modularity* we want to know what conditions a combination of confluent systems needs to achieve in order to *inherit* confluence. Some definitions are compulsory to understand modularity. Those are presented in the following.

Let $(\mathcal{F}, \mathcal{R})$ be the union of $(\mathcal{F}_1, \mathcal{R}_1)$ and $(\mathcal{F}_2, \mathcal{R}_2)$, i.e., $\mathcal{F} = \mathcal{F}_1 \cup \mathcal{F}_2$ and $\mathcal{R} = \mathcal{R}_1 \cup \mathcal{R}_2$. It is easy to see that the set of defined symbols in $(\mathcal{F}, \mathcal{R})$ is $\mathcal{D} = \mathcal{D}_1 \cup \mathcal{D}_2$ and the set of constructors is $\mathcal{C} = \mathcal{F} \setminus \mathcal{D}$, where $\mathcal{D}_i$ ($\mathcal{C}_i$) denotes the set of defined symbols (constructors) in $(\mathcal{F}_i, \mathcal{R}_i)$, $i \in \{1, 2\}$:

Definition 15 (Disjoint combination) [26, Section 8.4] *$(\mathcal{F}_1, \mathcal{R}_1)$ and $(\mathcal{F}_2, \mathcal{R}_2)$ are disjoint if they do not share function symbols, i.e., $\mathcal{F}_1 \cap \mathcal{F}_2 = \emptyset$ (or equivalently $\mathcal{C}_1 \cap \mathcal{C}_2 = \mathcal{C}_1 \cap \mathcal{D}_2 = \mathcal{D}_1 \cap \mathcal{C}_2 = \mathcal{D}_1 \cap \mathcal{D}_2 = \emptyset$). The disjoint union $(\mathcal{F}, \mathcal{R}) = (\mathcal{F}_1 \uplus \mathcal{F}_2, \mathcal{R}_1 \uplus \mathcal{R}_2)$ is sometimes also called the direct sum of $(\mathcal{F}_1, \mathcal{R}_1)$ and $(\mathcal{F}_2, \mathcal{R}_2)$.*

Example 16 (COPS 55.trs)

The following TRS, in COPS (55.trs), is a disjoint combination of TRSs in Examples 8 and 10.

$$Ap(Ap(Ap(S, x), y), z) \rightarrow Ap(Ap(x, z), Ap(y, z)) \tag{3.22}$$
$$Ap(Ap(K, x), y) \rightarrow x \tag{3.23}$$
$$Ap(I, x) \rightarrow x \tag{3.24}$$
$$Dh(z, z) \rightarrow z \tag{3.25}$$

Indeed, $\mathcal{R}$ is a combination of two *disjoint* TRSs, because $\mathcal{F}_1 = \{Ap, S, K, I\}$ and $\mathcal{F}_2 = \{Dh\}$ satisfy $\mathcal{F}_1 \cap \mathcal{F}_2 = \emptyset$, as required by Definition 15.

Definition 17 (Constructor-sharing combination) [26, Section 8.5] *$(\mathcal{F}_1, \mathcal{R}_1)$ and $(\mathcal{F}_2, \mathcal{R}_2)$ are constructor-sharing if they share at most constructors, i.e., $\mathcal{F}_1 \cap \mathcal{F}_2 = \mathcal{C}$ (or equivalently $\mathcal{C}_1 \cap \mathcal{D}_2 = \mathcal{D}_1 \cap \mathcal{C}_2 = \mathcal{D}_1 \cap \mathcal{D}_2 = \emptyset$). [26]*

Example 18

The following TRS, made up for this example, has a constructor-sharing combination of TRSs.

$$f(x) \rightarrow f(g(f(a), f(b))) \tag{3.26}$$
$$f(g(a, x)) \rightarrow f(g(f(x), g(a, b))) \tag{3.27}$$

$$h(x,x) \rightarrow g(x,a) \tag{3.28}$$

Indeed, $\mathcal{R}$ is a combination of two *constructor-sharing* TRSs:

$$f(x) \rightarrow f(g(f(a),f(b))) \tag{3.29}$$

$$f(g(a,x)) \rightarrow f(g(f(x),g(a,b))) \tag{3.30}$$

and

$$h(x,x) \rightarrow g(x,a) \tag{3.31}$$

That is because $\mathcal{F}_1 = \{f,g,a,b\}$ and $\mathcal{F}_2 = \{h,g,a,b\}$ satisfy $\mathcal{F}_1 \cap \mathcal{F}_2 = \mathcal{C} = \{g,a,b\}$, as required by Definition 17.

Definition 19 (Constructor-lifting rule) [26, into Definition 8.5.7] *A rule $\ell \rightarrow r$ is constructor-lifting if $root(r)$ is a shared constructor in the combination $(\mathcal{F}_1,\mathcal{R}_1)$ and $(\mathcal{F}_2,\mathcal{R}_2)$.*

Definition 20 (Layer-preserving TRS) [26, into Section 8.2.1] *A disjoint combination of TRSs $\mathcal{R}$ is layer-preserving if and only if is noncollapsing. A constructor-sharing combination of TRSs $\mathcal{R}$ is layer-preserving if and only if it contains neither collapsing nor constructor-lifting rules. [26]*

Example 18 is not layer-preserving, as it is a constructor-sharing combination of TRSs, noncollapsing, however, the rule 3.28 it is a constructor-lifting rule, because $root(r) = g$ and g is a shared constructor in the combination. Example 16 is not layer-preserving either, as it is a disjoint combination of TRSs, but it is collapsing because every rule, excluding the first one, is collapsing, i.e., their right-hand-sides are variables.

Example 21

Adapting the Example 18, we have the following layer-preserving TRS:

$$f(x) \rightarrow f(g(f(a),f(b))) \tag{3.32}$$

$$f(g(a,x)) \rightarrow f(g(f(x),g(a,b))) \tag{3.33}$$

$$h(x,x) \rightarrow h(g(x,x),a) \tag{3.34}$$

Indeed, $\mathcal{R}$ is a layer-preserving TRS, because it is a constructor-sharing combination of TRSs, noncollapsing and without any constructor-lifting rule: with the modification in rule 3, now $root(r) = h$, which is not a shared constructor in the combination.

Theorem 22 (Modularity in the confluence hierarchy) *Some studies have compiled a list of properties which guarantee modularity of confluence:*

- *For disjoint TRSs, confluence is modular. [32]*
- *For constructor-sharing and left-linear TRSs, confluence is modular. [27]*
- *For constructor-sharing and layer-preserving TRSs, confluence is modular. [25]*

Example 23

The TRS $\mathcal{R}$ in Example 16 is a disjoint union of $\mathcal{R}_1$ in Example 8 and $\mathcal{R}_2$ in Example 10. Since $\mathcal{R}_1$ is confluent by Huet & Lévy´s Theorem 7 and $\mathcal{R}_2$ is confluent due to Huet´s Theorem 9 and Newman´s Lemma 1. By Theorem 22, $\mathcal{R}$ is confluent.

Example 24

The TRS $\mathcal{R}$ in Example 21 is a constructor-sharing and layer-preserving. We can decomposed it into:

a TRS $\mathcal{R}_1$

$$f(x) \rightarrow f(g(f(a), f(b))) \tag{3.35}$$
$$f(g(a, x)) \rightarrow f(g(f(x), g(a, b))) \tag{3.36}$$

and a TRS $\mathcal{R}_2$

$$h(x, x) \rightarrow h(g(x, x), a) \tag{3.37}$$

The TRS $\mathcal{R}_1$ is strongly confluent (and hence confluent) because it is linear and it has one strongly joinable critical pair $\langle f(g(f(x), g(a, b))), f(g(f(a), f(b)))\rangle$:

$$f(g(f(x), g(a, b))) \rightarrow^{=}_{3.35} f(g(f(a), f(b)))$$
$$f(g(f(a), f(b))) \rightarrow^{*} f(g(f(a), f(b)))$$
$$f(g(f(a), f(b))) \rightarrow^{=} f(g(f(a), f(b)))$$
$$f(g(f(x), g(a, b))) \rightarrow_{3.35} f(g(f(a), f(b)))$$

The TRS $\mathcal{R}_2$ is confluent due to Huet´s Theorem 9 and Newman´s Lemma 1. By Theorem 22, $\mathcal{R}$ is confluent.

3.1.4 Use of feasibility

Definition 25 (Feasibility) [21, Definition 39] *A condition $s \bowtie t$ is* $(\mathbb{T}, \sigma)$-feasible *if* $\mathsf{Th}_{\bowtie} \vdash \sigma(s) \bowtie \sigma(t)$ *holds; otherwise, it is* $(\mathbb{T}, \sigma)$-infeasible. *We also say that $s \bowtie t$ is* $\mathbb{T}$-feasible *(or* $\mathsf{Th}_{\bowtie}$*-feasible, or just feasible if no confusion arises) if it is* $(\mathbb{T}, \sigma)$*-feasible for some substitution σ; otherwise, we call it* infeasible.

A sequence F *is* $\mathbb{T}$*-feasible (or just* feasible*) iff there is a substitution σ such that, for all $\gamma \in \mathsf{F}$, γ is* $(\mathbb{T}, \sigma)$*-feasible. Note that* () *is trivially feasible.*

As TRSs can be seen as specializations of CS-TRSs and CTRSs, the underlying logic theory could be that one used in [17] or in [10].

For any term t, $t^{\downarrow}$ means that every $x \in \mathcal{V}ar$ in the term is replaced by a constant $c_x \in \mathcal{F}, \mathrm{ar}(c_x) = 0$.

In TRSs, feasibility is used in order to prove the joinability of each $\mathsf{CP}\langle s, t\rangle$ of the system, as the feasibility of the sentence $s^{\downarrow} \to^* u, t^{\downarrow} \to^* u$, being u a fresh variable. We use infChecker for this task. infChecker is a tool that can verify infeasibility conditions of variants of TRSs. It is powered by the theorem prover Prover9, and the model generators Mace4 and AGES, powered by the SMT solver Barcelogic.

Example 26 (COPS 669.trs)

Consider the TRS:

$$a \to c \tag{3.38}$$
$$f(a) \to b \tag{3.39}$$
$$b \to b \tag{3.40}$$
$$b \to h(b, h(c, a)) \tag{3.41}$$

We can remove the rule 3.40 because it is redundant for confluence analysis. The system has only one critical pair $\langle s, t\rangle = \langle f(c), b\rangle$ obtained from rules 3.39 and 3.38. This critical pair consists of a term in normal form, $f(c)$, and b which starts an infinite rewriting sequence: $\underline{b} \to_{3.41} h(\underline{b}, h(c, a)) \to_{3.41} h(h(\underline{b}, h(c, a)), h(c, a)) \to_{3.41}$ .... The non-joinability of $f(c)$ and b could be difficult or impossible to prove for a basic graph-based rewriting processor. However, we can easily see that term b could never be rewritten into $f(c)$. That is because it only creates terms which begin with the function symbol h, which cannot be removed by any rewriting rule.

In contrast, infChecker can prove non-joinability of the critical pair if we try the following infeasibility problem:

```
(PROBLEM INFEASIBILITY)
(RULES
a -> c
f(a) -> b
b -> h(b,h(c,a))
)
(VAR x)
(CONDITION f(c) ->* x, b ->* x)
```

which infChecker proves infeasible.

3.2 Context-Sensitive Term Rewriting Systems

Given a signature $\mathcal{F}$ a *replacement map* is a mapping μ satisfying that, for all symbols $f \in \mathcal{F}, \mu(f) \subseteq \{1, ..., ar(f)\}$ [17]. The set of replacement maps for the signature $\mathcal{F}$ is $M_{\mathcal{F}}$ (or $M_{\mathcal{R}}$ for a TRS with signature $\mathcal{F}$). Replacement maps are compared as follows: $\mu \sqsubseteq \mu'$ if for all $f \in \mathcal{F}, \mu(f) \subseteq \mu'(f)$; we often say that μ is *more restrictive* than μ'. Extreme cases are $\mu_\bot$, which disallows replacements in all arguments: $\mu_\bot(f) = \emptyset$ for all $f \in \mathcal{F}$; and $\mu_\top$, which restricts no replacement: $\mu_\top(f) = \{1, ..., k\}$ for all k-ary symbols $f \in \mathcal{F}$.

The set $\mathcal{P}os^\mu(t)$ of *μ-replacing* (or *active*) *positions* of t is $\mathcal{P}os^\mu(t) = \{\Lambda\}$, if $t \in \mathcal{X}$ and $\mathcal{P}os^\mu(t) = \{\Lambda\} \cup \bigcup_{i \in \mu(f)} i.\mathcal{P}os^\mu(t_i)$, if $t = f(t_1, ..., t_k)$. The set of *non-μ-replacing* (or *frozen*) positions of t is $\overline{\mathcal{P}os^\mu}(t) = \mathcal{P}os(t) - \mathcal{P}os^\mu(t)$. Given terms s, t, we let $\overline{\mathcal{P}os^\mu_s}(t) = \overline{\mathcal{P}os^\mu}(t) \cap \mathcal{P}os_s(t)$, i.e., $\overline{\mathcal{P}os^\mu_s}(t)$ denotes the set of frozen positions of subterm s in t. Given a term t, $\mathcal{V}ar^\mu(t)$ (resp. $\mathcal{V}ar^{\not\mu}(t)$) is the set of variables occurring at active (resp. frozen) positions in t: $\mathcal{V}ar^\mu(t) = \{x \in \mathcal{V}ar(t) \mid \exists p \in \mathcal{P}os^\mu(t), x = t|_p\}$ and $\mathcal{V}ar^{\not\mu}(t) = \{x \in \mathcal{V}ar(t) \mid \exists p \in \overline{\mathcal{P}os^\mu}(t), x = t|_p\}$. Variables in $\mathcal{V}ar^\mu(t)$ could also be in $\mathcal{V}ar^{\not\mu}(t)$ and vice versa.

A pair $(\mathcal{R}, \mu)$ where $\mathcal{R}$ is a TRS and $\mu \in M_{\mathcal{R}}$ is often called a *CS-TRS*. *Context-sensitive rewriting* (CSR) is the restriction of rewriting obtained when a replacement map μ is used to specify the redex positions that can be contracted. A term s μ-rewrites to t written $s \overset{p}{\hookrightarrow}_{\mathcal{R},\mu} t$ (or $s \hookrightarrow_{\mathcal{R},\mu} t$, $s \hookrightarrow_\mu t$, or even $s \hookrightarrow t$), if $s \to^p_{\mathcal{R}} t$ and p is active in s, i.e., $p \in \mathcal{P}os^\mu(s)$.

If $\hookrightarrow_{\mathcal{R},\mu}$ is (locally) confluent (resp. terminating), we say that $\mathcal{R}$ is (locally) μ-confluent (resp. μ-terminating). The $\hookrightarrow_{\mathcal{R},\mu}$-normal forms are called μ-normal forms (of $\mathcal{R}$). Two terms s and t are μ-joinable (written $s \downarrow_\mu t$) if they are $\hookrightarrow_{\mathcal{R}}$-joinable.

3.2.1 Confluence of CSR

In order to prove local confluence of CSR, the concept of μ-critical pair is the *standard* definition, only *active* critical positions $p_i \in \mathcal{P}os^{\mu}_{\mathcal{F}}(\ell_j)$ are considered to find overlaps.

Definition 27 (μ-critical pair [21]) *Let $\mathcal{R}$ be a TRS and $\mu \in M_{\mathcal{R}}$. A critical pair, with the usual definition $\langle \sigma(l_1)[\sigma(r_2)]_p, \sigma(r_1)\rangle \in \mathsf{CP}(\mathcal{R})$ is a μ-critical pair of $\mathcal{R}$ if $p \in \mathcal{P}os^{\mu}_{\mathcal{F}}(l)$. The set of μ-critical pairs is $\mathsf{CP}(\mathcal{R}, \mu)$. A μ-critical pair $\langle s, t\rangle$ is μ-joinable if $s \downarrow_{\mu} t$.*

Example 28

The following TRS

$$\begin{aligned} f(x,y,z) &\to g(g(c,y),x) && (3.42)\\ f(a,y,z) &\to a && (3.43)\\ g(g(x,z),y) &\to y && (3.44)\\ g(c,g(c,z)) &\to f(z,g(a,b),c) && (3.45) \end{aligned}$$

has four critical pairs:

$$\langle a, g(g(c,y),a)\rangle \quad (3.46)$$

with 3.42 *and* 3.43 *with critical position* $p = \Lambda$

$$\langle g(c, f(z,g(a,b),c)), f(g(c,z),g(a,b),c)\rangle \quad (3.47)$$

with 3.45 *and* 3.45 *with critical position* $p = 2$

$$\langle g(z,y), y\rangle \quad (3.48)$$

with 3.44 *and* 3.44 *with critical position* $p = 1$

$$\langle g(f(z,g(a,b),c),y), y\rangle \quad (3.49)$$

with 3.44 *and* 3.45 *with critical position* $p = 1$

However, with $\mu(f) = \{1\}$ and $\mu(g) = \{2\}$, only 2 μ-critical pairs remain: 3.46 and 3.47. This is because 3.48 and 3.49 needs $p = 1$ to be active in g. In fact, $\mathcal{R}$

as a TRS is not confluent because the critical pair 3.48 is not convergent, but as a CS-TRS $(\mathcal{R}, \mu)$ is confluent, as we will in Example 33.

With CS-TRSs we cannot use Huet´s Theorem: having no μ-critical-pair, by its own, does *not* mean that $\mathcal{R}$ is confluent.

Example 29 (Example 9 [21])

Consider the left linear TRS $\mathcal{R}$

$$g(x, a) \rightarrow c(x) \tag{3.50}$$

$$a \rightarrow b \tag{3.51}$$

with $\mu(c) = 0$ and $\mu(g) = 1$. Although the left-hand side a of rule 3.51 overlaps the left-hand side $g(x, a)$ of rule 3.50 at position 2, this position is frozen in $g(x, a)$. Hence, $\mathcal{R}$ has no μ-critical pair.

However, the following peak

$$\mathsf{g(b, a)} \hookleftarrow \mathsf{g(a, a)} \hookrightarrow \mathsf{c(a)} \tag{3.52}$$

is *not* μ-joinable, as $\mathsf{c(a)}$ is a μ-normal form and the only μ-reduction step on $\mathsf{g(b, a)}$ is $\underline{\mathsf{g(b, a)}} \hookrightarrow_\mu \mathsf{c(b)}$, leading to a *different* μ-normal form.

Thus, the definition of left-homogeneous μ-replacing variables is needed.

Definition 30 (LHRV-condition [21]) *Let μ be a replacement map. A rule $\alpha : l \rightarrow r$ has left-homogeneous μ-replacing variables (written $\mathsf{LHRV}(\ell \rightarrow r, \mu)$), if active variables in the left-hand side l have no frozen occurrence neither in l nor in r, i.e., $\mathcal{V}ar^{\mu}(\ell) \cap \mathcal{V}ar^{\not\mu}(\alpha) = \emptyset$. A CSR $\mathcal{R}$ has left-homogeneous μ-replacing variables, written $\mathsf{LHRV}(\mathcal{R}, \mu)$, if $\mathsf{LHRV}(\ell \rightarrow r, \mu)$ holds for all rules $\ell \rightarrow r \in \mathcal{R}$.*

We often say that a rule is a LH_μ-negative rule if the LHRV*-condition does not hold for it.*

Example 31

For $\mathcal{R}$ and μ in example 28, the LHRV condition holds:

- $\mathsf{LHRV}(\alpha_{\ 3.42}, \mu)$ holds because $\mathcal{V}ar^{\mu}(\ell_{\ 3.42}) = \{x\}$ and $\mathcal{V}ar^{\not\mu}(\alpha_{\ 3.42}) = \{y, z\}$.
- $\mathsf{LHRV}(\alpha_{\ 3.43}, \mu)$ holds because $\mathcal{V}ar^{\mu}(\ell_{\ 3.43}) = \emptyset$ and $\mathcal{V}ar^{\not\mu}(\alpha_{\ 3.43}) = \{y, z\}$.

- $\mathsf{LHRV}(\alpha_{\ 3.44}), \mu)$ holds because $\mathcal{V}ar^{\mu}(\ell_{\ 3.44}) = \{y\}$ and $\mathcal{V}ar^{\not\mu}(\alpha_{\ 3.44}) = \{x\}$.
- $\mathsf{LHRV}(\alpha_{\ 3.45}, \mu)$ holds because $\mathcal{V}ar^{\mu}(\ell_{\ 3.45}) = \{z\}$ and $\mathcal{V}ar^{\not\mu}(\alpha_{\ 3.45}) = \emptyset$.

Theorem 32 (Confluence of CSR [17]) *Let $\mathcal{R}$ be a TRS and $\mu \in M_{\mathcal{R}}$.*

- *If $\mathcal{R}$ contains a non-μ-joinable μ-critical pair, then $\mathcal{R}$ is not (locally) μ-confluent.*
- *If $\mathcal{R}$ is left-linear, $\mathsf{LHRV}(\mathcal{R}, \mu)$ holds and $\mathcal{R}$ has no μ-critical pairs, then $\mathcal{R}$ is μ-confluent.*
- *If $\mathcal{R}$ is μ-terminating, $\mathsf{LHRV}(\mathcal{R}, \mu)$ holds, and all μ-critical pairs are μ-joinable, then $\mathcal{R}$ is μ-confluent.*

Example 33

The TRS $\mathcal{R}$ in Example 28 is terminating (thus μ-terminating) and, with $\mu(f) = \{1\}, \mu(g) = \{2\}$, the LHRV condition holds and all μ-critical pairs (3.46, 3.47) are μ-joinable. Therefore, $\mathcal{R}$ is μ-confluent by Theorem 32.

A new concept which appears in [21] is the LH_μ-critical pairs. They are powerful to prove non-μ-confluence when $\mathsf{LHRV}(l \to r, \mu)$ does not hold.

Definition 34 (LH_μ-critical pair) [21, Definition 20] *Let $\mathcal{R}$ be a TRS and $\mu \in M_{\mathcal{R}}$. Let $\ell \to r \in \mathcal{R}$ be a LH_μ-negative rule and $p \in \mathcal{P}os^{\mu}_{x}(\ell)$ for some variable x of ℓ such that $x \in \mathcal{V}ar^{\not\mu}(\ell) \cup \mathcal{V}ar^{\not\mu}(r)$. Let y be a fresh variable, not occurring in ℓ or r. Then $\pi : \langle \ell[y]_p, r\rangle \Leftarrow x \hookrightarrow y$ is called and LH_μ-critical pair. Borrowing from the usual terminology of (ordinary) critical pairs, position p is called the critical position of π. $\mathsf{LHCP}(\mathcal{R}, \mu)$ contains all LH_μ-critical pairs of $\mathcal{R}$.*

With this definition we obtain a set of *extended μ-critical pairs* which is used to provide a characterization of local confluence of CSR [21, Definition 29]:

$$\mathsf{ECP}(\mathcal{R}, \mu) = \mathsf{CP}(\mathcal{R}, \mu) \cup \mathsf{LHCP}(\mathcal{R}, \mu)$$

Theorem 35 (Local Confluence of CSR [21]) *Let $\mathcal{R}$ be a TRS and $\mu \in M_{\mathcal{R}}$. Then, $\mathcal{R}$ is locally μ-confluent if and only if all pairs in $\mathsf{ECP}(\mathcal{R}, \mu)$ are μ-joinable.*

Example 36

Consider the TRS with $\mu(h) = \{1\}$

$$a \rightarrow h(a, h(a, a)) \tag{3.53}$$
$$h(h(x, a), h(a, z)) \rightarrow h(b, b) \tag{3.54}$$
$$h(y, b) \rightarrow h(h(b, y), h(y, a)) \tag{3.55}$$

The TRS in Example 36 has no μ-critical pairs and. However, MU-TERM proves it non-μ-terminating. Therefore, we cannot say anything about μ-confluence. However, helped by this new concept of LHCPs 34, we can prove the non-μ-confluence of the CS-TRS:

Example 37

Consider the TRS $\mathcal{R}$ of example 36. An LH_μ-critical pair is obtained:

$$\langle h(y_1, b), h(h(b, y), h(y, a))\rangle \Leftarrow y \hookrightarrow y_1 \tag{3.56}$$

We obtain 3.56 from the LH_μ-negative rule $\alpha_{\ 3.55}$. There, the variable y is frozen in the right-hand side $h(h(b, y), h(y, a))$ and active in the left-hand side $h(y, b)$ at position 1. Therefore, we have here an LH_μ-critical pair $\pi : \langle \ell[y_1]_1, r\rangle \Leftarrow y \hookrightarrow y_1$, where y_1 is a fresh new variable.

And 3.56 is non-convergent, which can be proved with infChecker. Thus $\mathcal{R}$ is non-confluent.

Definition 38 (Canonical replacement map) [17, into Section 5] *The canonical replacement map $\mu_{\mathcal{R}}^{can}$ of $\mathcal{R}$ is the most restrictive replacement map ensuring that the non-variable subterms of the left-hand sides of the rules of $\mathcal{R}$ are active.*

The set of replacement maps which are less restrictive than $\mu_{\mathcal{R}}^{can}$ is denoted $CM_{\mathcal{R}} = \{\mu \in M_{\mathcal{R}} \mid \mu_{\mathcal{R}}^{can} \sqsubseteq \mu\}$.

Gramlich and Lucas found some connections between μ-confluence of canonical CSR and confluence properties of unrestricted rewriting. First one is that *confluence does not imply canonical μ-confluence*, e.g. [17, Example 8.2]. Also, the following generalization of Newman's Lemma was proved by Gramlich and Lucas in [6]:

Theorem 39 (Gramlich and Lucas) [6, Theorem 2] *Let $\mathcal{R}$ be a left-linear TRS and $\mu \in CM_{\mathcal{R}}$ (38). If all μ-critical pairs are μ-joinable, and $\mathcal{R}$ is μ-terminating and level-decreasing (w.r.t. μ), then $\mathcal{R}$ is confluent.*

Here, a TRS $\mathcal{R}$ is said to be *level-decreasing* if for all rules $\ell \to r$ in $\mathcal{R}$, the level of each variable in r does not exceed its level in ℓ; the level $lv_\mu(t, x)$ of a variable x in a term t is obtained by adding the number of frozen arguments that are traversed from the root to the variable. [17, into Section 8.5]

Related to this theorem is the following corollary, because in those non-μ-terminating systems, weakly normalizing TRSs are enough.

Corollary 40 (Lucas) [17, Corollary 8.23] *Let $\mathcal{R}$ be a left-linear, normalizing TRS and $\mu \in CM_{\mathcal{R}}$ (38). If $\mathcal{R}$ is μ-confluent, then $\mathcal{R}$ is confluent.*

A good example of this is the 111.trs from COPS [12] that was used before as example of joinability of critical pairs (Example 5).

Example 41 (COPS 111.trs)

Consider the non-terminating TRS $\mathcal{R}$ in Example 5. Note that $\mathcal{R}$ is left-linear and its critical pairs are convergent. However, they are not strongly-joinable.

The second component e of the critical pair $\langle c, e\rangle$ is a normal from. However, c cannot be rewritten into e in at most one step. This can be proved by using infChecker:

```
(PROBLEM INFEASIBILITY)
(VAR x)
(RULES
a -> b
a -> c
a -> e
b -> d
c -> a
d -> a
d -> e
g(x) -> h(a)
h(x) -> e
)
```

```
(VAR z_1 z_2)
(CONDITION c ->= z_1, e ->* z1, e ->= z_2, c ->* z_2)
```

Thus, no result in Section 3.1 can be ued to prove confluence of $\mathcal{R}$. So we try another approach: compute its canonical replacement map, which is $\mu(g) = \mu(h) = \emptyset$. However, MU-TERM can prove that with this replacement map, $\mathcal{R}$ is still not μ-terminating (infinite loop between rules 3.3 and 3.6) and Gramlich and Lucas' [6] Theorem 2 is not an option. With Fort [28] we can prove that $\mathcal{R}$ is WN (weakly-normalizing), so we achieved the conditions to apply Corollary 40 to conclude confluence of $\mathcal{R}$.

3.2.2 Joinability of extended μ-critical pairs

Proposition 42 () *[21, Proposition 52] Let $\mathcal{R}$ be a TRS, $\mu \in M_{\mathcal{R}}$, and $\pi : \langle \ell[y]_p, r\rangle \Leftarrow x \hookrightarrow y \in \mathsf{LHCP}(\mathcal{R}, \mu)$ be an LH_μ-critical pair. Being $\overline{\mathcal{R}^\mu}$ the first-order theory associated to $(\mathcal{R}, \mu)$. If $\overline{\mathcal{R}^\mu} \vdash (\forall x)(\forall y)(\exists z)x \to y \Rightarrow \ell^{\downarrow_{\overline{\{x\}}}}[y]_p \to^* z \wedge r^{\downarrow_{\overline{\{x\}}}} \to^* z$ holds, then π is μ-joinable. Here we can ground every variable v in π unless $v \in \{x, y\}$.*

Example 43

Consider the TRS $\mathcal{R}$ "Maude_06/MYNAT_nosorts-noand-peanoSimple.xml" from TermCOMP 2021 TRS Context Sensitive [1] :

$$U11(tt, m, n) \to U12(tt, m, n) \tag{3.57}$$
$$U12(tt, m, n) \to s(plus(n, m)) \tag{3.58}$$
$$plus(n, 0) \to n \tag{3.59}$$
$$plus(n, s(m)) \to U11(tt, m, n) \tag{3.60}$$

where m and n are variables and $\mu(U11) = \{1\}, \mu(U12) = \{1\}$. Its ECPs are:

$$\langle plus(n, s(y)), U11(tt, m, n)\rangle \Leftarrow m \hookrightarrow y \tag{3.61}$$
$$\langle plus(y, s(m)), U11(tt, m, n)\rangle \Leftarrow n \hookrightarrow y \tag{3.62}$$

We can prove with Prover9 that these LH_μ-critical pairs are μ-joinable by using Proposition 42. The goals to prove here are:

$$\overline{\mathcal{R}^\mu} \vdash (\forall M)(\forall y)(\exists z)M \to y \Rightarrow plus(c_n, s(y)) \hookrightarrow^* z \wedge U11(tt, n, c_n) \hookrightarrow^* z$$
$$\overline{\mathcal{R}^\mu} \vdash (\forall N)(\forall y)(\exists z)N \to y \Rightarrow plus(y, s(c_n)) \hookrightarrow^* z \wedge U11(tt, c_m, n) \hookrightarrow^* z$$

[1] see https://termcomp.github.io/Y2021/job_47882.html

Prover9 obtains a proof of them. Thus by Proposition 42, 3.61 and 3.62 are μ-joinable. Since $\mathcal{R}$ is μ-terminating , by Theorem 32, $\mathcal{R}$ is μ-confluent.

3.2.3 Use of feasibility

The underlying logic theory is that one used in [17].

The use of infChecker in CS-TRS problems is the same as in TRSs, but slightly different due to the replacement map μ. However, infChecker handles it and the joinability of each μ-critical pair $\langle s, t\rangle$ is proved as the feasibility of the sentence $s^{\downarrow} \hookrightarrow^* u, t^{\downarrow} \hookrightarrow^* u$, being u a fresh variable.

Example 44

Consider the TRS $\mathcal{R}$:

$$
\begin{aligned}
c &\to e && (3.63)\\
b &\to c && (3.64)\\
e &\to h_1(b, c, e, f(b, c)) && (3.65)\\
f(b, h(b, x_1, c, x_1)) &\to h(b, c, c, e) && (3.66)\\
f(e, y_1) &\to h(e, c, y_1, e) && (3.67)\\
h(y_1, y_1, y, x_1) &\to b && (3.68)\\
h_1(f(c, x), x, e, y) &\to h_1(b, e, y, y) && (3.69)\\
h_1(f(b, e), y, f(e, b), x) &\to b && (3.70)\\
h_1(y, x, y_1, b) &\to h(e, c, y_1, c) && (3.71)\\
h_1(x, c, c, y) &\to c && (3.72)
\end{aligned}
$$

The confluence of $\mathcal{R}$ not be proved or disproved by any of the current CR for TRS solvers of CoCoWeb [2] .

With $\mu(f) = \mu(h_1) = \emptyset$ and $\mu(h) = \{2, 3, 4\}$, MU-TERM can prove the μ-termination of $\mathcal{R}$. Besides, $\mathcal{R}$ satisfies the LHRV condition. Finally, we check the μ-joinability of its critical pairs, which are the following:

- $\langle c, h(e, c, c, c)\rangle$, which is μ-joinable because $h(e, c, c, c) \hookrightarrow_{3.68} b \hookrightarrow_{3.64} c$.
- $\langle h_1(b, e, b, b), h(e, c, e, c)\rangle$, which is μ-joinable because $h_1(b, e, b, b) \hookrightarrow_{3.71} h(e, c, b, c) \hookrightarrow_{3.68}$ and $h(e, c, e, c) \hookrightarrow_{3.68} b$.

[2] see `http://cocoweb.uibk.ac.at/`

- $\langle b, h(e, c, f(e, b), c)\rangle$, whose μ-joinability can be proved as the feasibility of the condition $b \hookrightarrow^* z, h(e, c, f(e, b), c) \hookrightarrow^* z$ with infChecker.

Therefore $\mathcal{R}$ is μ-confluent.

3.3 Conditional Term Rewriting Systems

A *conditional rule* (with label α) is written $\alpha : \ell \to r \Leftarrow c$, where c is a sequence $s_1 \approx t_1, \cdots, s_n \approx t_n$ with $\ell, r, s_1, t_1, \ldots, s_n, t_n \in \mathcal{T}(\mathcal{F}, \mathcal{X})$ and $\ell \notin \mathcal{X}$. As usual, ℓ and r are called the left- and right-hand sides of the rule, and C is the *conditional part* of the rule. A Conditional Term Rerwiting System (CTRS) $\mathcal{R}$ is a set of conditional rules; if all rules in $\mathcal{R}$ have an empty conditional part, then $\mathcal{R}$ is just a TRS.

Example 45

Consider the CTRS $\mathcal{R}$, Example 7.2.10 from [26]

$$a \to d \Leftarrow b \approx c \tag{3.73}$$

$$a \to c \tag{3.74}$$

If we remove the condition of rule 3.73, then we have a TRS.

A CTRS $\mathcal{R}$ is called *deterministic* if for each rule $\ell \to r \Leftarrow s_1 \approx t_1, \ldots, s_n \approx t_n$ in $\mathcal{R}$ and each $1 \leq i \leq n$, we have $\mathcal{V}ar(s_i) \subseteq \mathcal{V}ar(\ell) \cup \bigcup_{j=1}^{i-1} \mathcal{V}ar(t_j)$. Conditions $s \approx t$ in conditional rules admit several *semantics*, i.e., forms to evaluate them see, e.g., [26, Definition 7.1.3]: *Oriented* CTRSs are those whose conditions $s \approx t$ are handled as *reachability* tests $\sigma(s) \to^* \sigma(t)$ for an appropriate substitution σ. *Join* CTRSs use *joinability* tests $\sigma(s) \downarrow \sigma(t)$ instead. *Semiequational* CTRSs use *convertibility* tests $\sigma(s) \leftrightarrow^* \sigma(t)$.

Definition 46 (Quasi-decreasing CTRS) [26, Definition 7.2.39] *A deterministic 3-CTRS $(\mathcal{F}, \mathcal{R})$ is said to be* quasi-decreasing *if there is a well-founded partial ordering $\succ$ on $\mathcal{T}(\mathcal{F}, \mathcal{V})$, satisfying:*

1. $\to_{\mathcal{R}} \subseteq \succ$,
2. $\succ$ *has the subterm property (hence* $\succ = \succ_{st}$*), and*

3. *for every rule* $\ell \to r \Leftarrow s_1 \approx t_1, \ldots s_k \approx t_k \in \mathcal{R}$, *every substitution* σ *and* $0 \leq i < k$: *if* $\sigma(s_j) \to^*_{\mathcal{R}} \sigma(t_j)$ *for every* $1 \leq j \leq i$, *then* $\sigma(\ell) \succ \sigma(s_{i+1})$.

Example 47

Consider the CTRS $\mathcal{R}$ from 45.

The partial ordering $\succ$ defined by $a \succ b$ shows that $\mathcal{R}$ is quasi-decreasing ($\to_{\mathcal{R}_2} = \emptyset$ and b can not be rewritten into c).

Termination of a CTRS can be proved in some ways. In CONFident we call MU-TERM with different flags in order to prove termination or operational termination (which implies quasi-decreasing), however, we can also prove the termination of the underlying TRS $\mathcal{R}_u$.

Definition 48 (Underlying TRS $\mathcal{R}_u$) *Let* $\mathcal{R}$ *be a CTRS, we call its underlying TRS* $\mathcal{R}_u$ *that one with the same rules* $\ell \to r$, *but erasing the conditional part* c, *i.e.,* $\mathcal{R}_u = \{\ell \to r \Leftarrow c \in \mathcal{R}\}$.

For all terms s, t, we write (i) $s \to_{\mathcal{R}} t$ (resp. $s \to^*_{\mathcal{R}} t$) if and only if there is a (well-formed)[3] proof tree for $s \to t$ (resp. $s \to^* t$) using $\mathcal{I}(\mathcal{R})$. Equivalently, we have (ii) $s \to_{\mathcal{R}} t$ (resp. $s \to^*_{\mathcal{R}} t$) if and only if the first-order theory associated to $\mathcal{R}$, $\overline{\mathcal{R}} \vdash s \to t$ (resp. $\overline{\mathcal{R}} \vdash s \to^* t$) holds. The first presentation (i) is well-suited for the analysis of the termination behavior of CTRSs: we say that $\mathcal{R}$ is *operationally terminating* if there is no (well-formed) infinite proof trees for goals $s \to t$ and $s \to^* t$ in $\mathcal{I}(\mathcal{R})$ [18]. However, the proof theoretic presentation (ii) is more important in the analysis of (in)feasibility of rewriting goals. It also suffices to define *termination* of CTRSs: a CTRS $\mathcal{R}$ is terminating if $\to_{\mathcal{R}}$ is terminating. Termination and operational termination of CTRSs differ, see [19, Section 3] for a deeper discussion about differences and connections between both notions.

We use termination and operational termination in some confluence results for CTRSs. The tool MU-TERM [9] can be used for automatically proving and disproving termination and operational termination of (oriented) CTRSs.[4]

[3] By a *well-formed* proof tree we mean a proof tree where proof conditions introduced by inference rules are developed from left to right, see [18].

[4] Although the version of MU-TERM described in [9] did not allow proofs of termination of CTRSs, for the purpose of serving as a backbone for CONFident, we recently modified MU-TERM as to provide explicit use of the techniques described in [20], which can be used to prove and disprove termination of CTRSs. Thus, MU-TERM users can prove and disprove termination of CTRSs by following the instructions in `http://zenon.dsic.upv.es/muterm/?name=documentation#CTRSs`.

The U and U_{opt} transformations are used to prove confluence. They are based on the transformation proposed on [26] and the optimization of [3]:

Definition 49 (U and U_{opt} transformations) *Let $\mathcal{R}$ be a deterministic 3-CTRS. For each conditional rule as $\ell \to r \Leftarrow s_1 \approx t_1, ..., s_n \approx t_n$ we introduce $n+1$ unconditional rules*

$$\ell \to U_1(s_1, \overrightarrow{x}_1) \tag{3.75}$$

$$U_{i-1}(t_{i-1}, \overrightarrow{x}_{i-1}) \to U_i(s_i, \overrightarrow{x}_i) \tag{3.76}$$

$$U_n(t_n, \overrightarrow{x}_n) \to r \tag{3.77}$$

where the U_i are fresh new symbols added to $\mathcal{F}$ and the $\overrightarrow{x}_i$ are vectors of variables occurring in $\mathcal{V}ar(\ell) \cup \mathcal{V}ar(t_1) \cup \cdots \cup \mathcal{V}ar(t_{i-1})$ for all $1 \leq i \leq n$. Let $U(\mathcal{R})$ be the set of unconditional rules obtained in this way.

The optimized version $U_{opt}(\mathcal{R})$ is obtained if the $\overrightarrow{x}_i$ above are defined as follows, for all $1 \leq i \leq n$:

$$\overrightarrow{x}_i = (\mathcal{V}ar(\ell) \cup \mathcal{V}ar(t_1) \cup ... \cup \mathcal{V}ar(t_{i-1})) \cap (\mathcal{V}ar(t_i) \cup \mathcal{V}ar(s_{i+1}) \cup \mathcal{V}ar(t_{i+1}) \cup ... \cup \mathcal{V}ar(s_n) \cup \mathcal{V}ar(t_n) \cup \mathcal{V}ar(r))$$

Example 50

Consider the CTRS $\mathcal{R}$ (793.trs from COPS [12]):

$$a \to a \Leftarrow f(a) \approx a \tag{3.78}$$

$$f(x) \to a \Leftarrow x \approx b \tag{3.79}$$

We could transform $\mathcal{R}$ into the $U_{opt}(\mathcal{R})$ as follows:

$$a \to U_1(f(a)) \tag{3.80}$$

$$U_1(a) \to a \tag{3.81}$$

$$f(x) \to U_2(x, x) \tag{3.82}$$

$$U_2(b, x) \to a \tag{3.83}$$

Example 51

Consider the CTRS $\mathcal{R}$ (553.trs from COPS [12]):

$$a \to b \tag{3.84}$$

$$c \rightarrow k(f(a)) \tag{3.85}$$
$$c \rightarrow k(g(b)) \tag{3.86}$$
$$f(x) \rightarrow g(x) \Leftarrow h(f(x)) \approx k(g(b)) \tag{3.87}$$
$$h(f(a)) \rightarrow c \tag{3.88}$$
$$h(x) \rightarrow k(x) \tag{3.89}$$

$\mathcal{R}$ is a 1-CTRS and its underlying TRS $\mathcal{R}_u$ is terminating. CONFident transforms $\mathcal{R}$ into $U(\mathcal{R})$ by deleting the conditional rule 3.87 and adding the fresh new symbol U_1 and two unconditional rules:

$$f(x) \rightarrow U_1(h(f(x)), x) \tag{3.90}$$
$$U_1(k(g(b)), x) \rightarrow g(x) \tag{3.91}$$

And then CONFident tries TRS methods for confluence, and it ends proving confluence with Theorem 39. The replacement map used is $\mu(f) = \mu(g) = \mu(h) = \mu(k) = \mu(U_1) = \{1\}$. Its critical pairs

$$\langle k(f(a)), k(g(b))\rangle \tag{3.92}$$
$$\langle h(U_1(h(f(a)), a)), c\rangle \tag{3.93}$$
$$\langle h(f(b)), c\rangle \tag{3.94}$$
$$\langle c, k(f(a))\rangle \tag{3.95}$$

are μ-joinable.

Definition 52 (Types of CTRS) *As in [26] and [24] before, rewrite rules are presented as $\ell \rightarrow r \Leftarrow c$ and classified according to the distribution of variables among ℓ, r and c.*

An n-CTRS consists of rule of type n (or n-rules) only.

Type	Requirement
1	$\mathcal{V}ar(r) \cup \mathcal{V}ar(c) \subseteq \mathcal{V}ar(\ell)$
2	$\mathcal{V}ar(r) \subseteq \mathcal{V}ar(\ell)$
3	$\mathcal{V}ar(r) \subseteq \mathcal{V}ar(\ell) \cup \mathcal{V}ar(c)$
4	No restrictions

In a 1-CTRS has no extra variables (all variables in the rule only occur in the left-hand side), a 2-CTRS has no extra variables on the right-hand sides of the

rules, and 3-CTRS may contain extra-variables on the right-hand sides of the rules provided that these also occur in the conditions. More restrictive types are subsets of less restrictive ones, i.e., 1-CTRS are 2,3 and 4-CTRS too.

Definition 53 (Inlining of conditions) [29, Definition 4] *Given a conditional rule* $\ell \to r \Leftarrow s_1 \approx t_1, \cdots, s_n \approx t_n$ *and an index* $1 \leq i \leq n$ *such that* $t_i = x$ *for some variable* x*, let the inlining of the* i*th condition of the rule be* $\ell \to \sigma(r) \Leftarrow \sigma(s_1) \approx t_1, \cdots, \sigma(s_{i-1}) \approx t_{i-1}, \sigma(s_{i+1}) \approx t_{i+1}, \cdots, s_n \approx t_n$ *with* $\sigma = \{x \to s_i\}$.

Example 54

Consider the CTRS $\mathcal{R}$ (351.trs from COPS [12]):

$$ssp(nil, 0) \to nil \tag{3.96}$$
$$ssp(cons(y, ys'), v) \to xs \Leftarrow ssp(ys', v) \approx xs \tag{3.97}$$
$$ssp(cons(y, ys'), v) \to cons(y, xs') \Leftarrow sub(v, y) \approx w, ssp(ys', w) \approx xs' \tag{3.98}$$
$$sub(z, 0) \to z \tag{3.99}$$
$$sub(s(v), s(w)) \to z \Leftarrow sub(v, w) \approx z \tag{3.100}$$

Here we can apply inlining of conditions for every conditional rule, e.g., rule 3.98, beginning from the last condition, $ssp(ys', w) \approx xs'$, we have a variable xs' in its right-hand side, so we can do $\sigma = \{xs' \to ssp(ys', w)\}$ and the inlining of the second condition of the rule is $\ell \to \sigma(r) \Leftarrow \sigma(s_1) \approx t_1$ or $\alpha_{3.98} : ssp(cons(y, ys'), v) \to cons(y, ssp(ys', w)) \Leftarrow sub(v, y) \approx w$. The same happens in the last condition with variable w, now $\sigma = \{w \to sub(v, y)\}$ and the result rule is $\alpha_{3.98} : ssp(cons(y, ys'), v) \to cons(y, ssp(ys', sub(v, y)))$.

Resulting $\mathcal{R}$ after inlining of conditions is:

$$ssp(nil, 0) \to nil \tag{3.101}$$
$$ssp(cons(y, ys'), v) \to ssp(ys', v) \tag{3.102}$$
$$ssp(cons(y, ys'), v) \to cons(y, ssp(ys', sub(v, y))) \tag{3.103}$$
$$sub(z, 0) \to z \tag{3.104}$$
$$sub(s(v), s(w)) \to sub(v, w) \tag{3.105}$$

3.3.1 Confluence of conditional rewriting

A CTRS $\mathcal{R}$ is confluent if $\to_{\mathcal{R}}$ is confluent.

Definition 55 (Conditional Critical Pair) [26] *Let* $\ell_1 \to r_1 \Leftarrow c_1$ *and* $\ell_2 \to r_2 \Leftarrow c_2$ *be renamed versions of rewrite rules of* $\mathcal{R}$ *such that they have no variables in common. Suppose* $\ell_1 = C[t]$ *with* $t \notin \mathcal{V}$ *such that* $\sigma(t) = \sigma(\ell_2)$ *for a most general unifier* σ*.*

We call

$$\langle \sigma(C[r_2]), \sigma(r_1) \rangle \Leftarrow \sigma(c_1), \sigma(c_2)$$

a conditional critical pair CCP *of* $\mathcal{R}$*. It the two rules are renamed versions of the same rule, we do not consider the case where* $p = \Lambda$*. A conditional critical pair* $\langle t, t \rangle \Leftarrow c$ *is called trivial.*

Definition 56 (Feasible CCP) *A conditional critical pair* $\langle s, t \rangle \Leftarrow s_1 \approx t_1, ..., s_n \approx t_n$ *is feasible if there is a substitution* σ *such that* $\sigma(s_i) \approx \sigma(t_i)$ *holds for all* $1 \leq i \leq n$*. Otherwise it is called infeasible.*

Note that infeasible conditional critical pairs are trivially joinable. Thus, in order to prove local-confluence, we can remove infeasible CCPs. That is why a powerful tool for checking the feasibility of the conditions is almost compulsory in confluence tools for CTRS.

Theorem 57 *Every quasi-decreasing strongly-deterministic 3-CTRS with joinable critical pairs is confluent. [26]*

Definition 58 (Strongly deterministic CTRS) [26] *Let* $\mathcal{R}$ *be a deterministic 3-CTRS,* $\mathcal{R}$ *is called strongly deterministic if, for every rule* $\ell \to r \Leftarrow s_1 \approx t_1, ..., s_n \approx t_n$ *in* $\mathcal{R}$*, every term* t_i *is strongly irreducible. A term* t *is called strongly irreducible w.r.t.* $\mathcal{R}$ *if* $\sigma(t)$ *is a normal form for every normalized substitution* σ*.* $\mathcal{R}$ *is called syntactically deterministic if, for every rule* $\ell \to r \Leftarrow s_1 \approx t_1, ..., s_n \approx t_n$ *in* $\mathcal{R}$*, every term* t_i*, is a constructor term or a ground* $\mathcal{R}_u$ *normal form.*

It is undecidable whether a term t is strongly irreducible w.r.t. $\mathcal{R}$. However, there is an important syntactic criterion: every syntactically deterministic CTRS is strongly deterministic.

CONFident also works for join type CTRS, not only oriented, so the following result is also interesting, and related to Theorem 57.

Corollary 59 [26] *A decreasing join 1-CTRS is confluent if and only if all its conditional critical pairs are joinable.*

Definition 60 (Right-stable CTRS) [26] *A CTRS is called right stable if every rewrite rule* $\ell \to r \Leftarrow s_1 \approx t_1, ..., s_n \approx t_n$ *in* $\mathcal{R}$ *satisfies the following conditions for all* $i \in \{1, ..., k\}$*:*

$$(\mathcal{V}ar(l) \cup \bigcup_{j=1}^{i-1} \mathcal{V}ar(s_j = t_j) \cup \mathcal{V}ar(s_i)) \cap \mathcal{V}ar(t_i) = \emptyset$$

and t_i *is either a linear constructor term or a ground* $\mathcal{R}_u$ *normal form. Every variable* $y \in \bigcup_{i=1}^{k} \mathcal{V}ar(t_i)$ *is an extra variable ,i.e., y does not occur in* ℓ*.*

Definition 61 (Normal CTRS) [26] *A normal CTRS* $(\mathcal{F}, \mathcal{R})$ *is an oriented CTRS in which the rewrite rules* $\ell \to r \Leftarrow s_1 \approx t_1, \ldots, s_n \approx t_n$ *are subject to the additional constraint that for all* $1 \leq i \leq n$*,* t_i *is a ground normal form with respect to* $\mathcal{R}_u$*.*

Definition 62 (Almost normal CTRS) [26] *A CTRS is called almost normal if it is normal or right-stable and oriented.*

Definition 63 (Properly oriented CTRS) [30] *An arbitrary 3-CTRS is called properly oriented if every rule* $\ell \to r \Leftarrow s_1 \approx t_1, ..., s_n \approx t_n$ *in* $\mathcal{R}$ *satisfies: If* $\mathcal{V}ar(r) \nsubseteq \mathcal{V}ar(\ell)$*, then* $\mathcal{V}ar(s_i) \subseteq \mathcal{V}ar(\ell) \cup \bigcup_{j=1}^{i-1} \mathcal{V}ar(t_j)$ *holds for all* $1 \leq i \leq k$ *when the right-hand side r of a rule does not contain extra variables. Every deterministic 3-CTRS is properly oriented, but not vice versa.*

Definition 64 (Level-confluence) [26] *A CTRS* $\mathcal{R}$ *is called level-confluent if, for every* $n \in \mathbb{N}$*, the TRS* $\mathcal{R}_n$*, meaning that every rule, independently and without conditions, is confluent.*

Level-confluence implies confluence. However, not vice versa.

Corollary 65 [26] *Every almost orthogonal almost normal 2-CTRS is level-confluent.*

Theorem 66 [30] *Every orthogonal properly oriented right-stable 3-CTRS is level-confluent.*

3.3.2 Use of feasibility

The underlying logic theory is that one used in [10].

In CTRS problems, CONFident uses feasibility calling to infChecker in this cases:

- To prove if a conditional rule $\ell \to r \Leftarrow c$ has a feasible conditional part c.

- To prove if a CCP is feasible (see Definition 56).
- To prove the convergence of a CCP (see [10, Section 6])
- To check (ground) normal forms in order to prove normality or right-stability of the CTRS. A term t is a normal form if and only if $t^{\downarrow} \to x$, where x is a variable, is infeasible.

4

Mechanization

CONFident is written in Haskell and it has more than 80 Haskell files with more than 9000 lines of pure code (blanks and comments not included). Besides, several shell scripts are used for the analysis of the results, some PHP code has been developed for the automation of benchmarks, and also a Python script for generating random examples.

4.1 Divide and Conquer Framework

Because proving confluence is, in general, undecidable, existing techniques for proving and disproving confluence are successful on some kind of systems and fail on others. However, the combination of different techniques in a certain order or the use of auxiliar properties can help to achieve a positive or negative answer about the confluence property.

We describe a framework that follows a divide and conquer strategy, where the input system is described as a *problem* in our framework and the different confluence techniques and the auxiliar property checkings are encapsulated as *processors* and also as other (possibly different) *problems*. The application order of the different processors is defined as an application strategy. Choosing the right strategy to an input problem is not a trivial task. A thoughtful experimental analysis has been done to obtain a general strategy.

This type of frameworks is commonly used when dealing with undecidable properties. Originally developed for proving termination of TRSs [4], several adaptations and extensions have been recently made, for instance for proving infeasibility [7]. In such frameworks, there is a single definition of "problem" and the defined processors return a hopefully simpler set of problems of the same kind. In this thesis, though,

1. There is a main class of problems (namely, the *confluence problems*) but we

also consider and use other subsidiary (termination, operational termination, infeasibility,...) problems, which are introduced depending on the considered phase of the confluence proof.

2. Our processors take a problem and return a hopefully simpler set of new problems (possibly of different types). They can also return "no".

There are processors that call to other frameworks internally, e.g., by issuing external calls to tools like MU-TERM and infChecker, which implement specific proof frameworks for particular problems.

4.1.1 Problems and Processors

Some new problems are introduced and considered in CONFident: *confluence* and *joinability* problems. Those problems naturally appear when proving confluence.

Definition 67 (Confluence Problem) *A* confluence (resp. μ-confluence) problem *is denoted* $CR(\mathcal{R})$ *(resp.* $CR(\mathcal{R},\mu)$*) where* $\mathcal{R}$ *is a CTRS and* μ *is a replacement-map. A* confluence (resp. μ-confluence) problem τ *is* positive *if* $\mathcal{R}$ *is* confluent (resp. μ-confluent)*; otherwise it is* negative*.*

In the following definition, we use conditional pairs $\langle s,t\rangle \Leftarrow c$ where c is a sequence of atoms $s_i \approx t_i$ and/or $s_1 \to t_i$ for $1 \leq i \leq n$. If c is empty we just write $\langle s,t\rangle$. A conditional pair is joinable in $\mathcal{R}$ for all substitutions σ, whenever $\sigma(c)$ holds in the considered theory for $\mathcal{R}$, then there is a term u such that both $\sigma(s) \to^*_{\mathcal{R}} u$ and $\sigma(t) \to^*_{\mathcal{R}} u$ hold.

Definition 68 (Joinability Problem) *A* joinability (resp. μ-joinability) problem *is denoted* $JO(\mathcal{R},\pi)$ *(resp.* $JO(\mathcal{R},\mu,\pi)$*) where* $\mathcal{R}$ *is a possibly conditional rewrite system,* μ *a replacement-map and* π *is a conditional pair. We say that* τ *is positive if* π *is joinable (resp.* μ*-joinable) in* $\mathcal{R}$*; otherwise it is negative.*

Besides these new problems, $T(\mathcal{R})$, $T(\mathcal{R},\mu)$ and $OT(\mathcal{R})$ are used to describe termination, μ-termination, and operational termination problems, respectively. In order to keep our presentation consistent, $T(\mathcal{R})$, $T(\mathcal{R},\mu)$ (resp. $OT(\mathcal{R})$) are said *positive* if $\mathcal{R}$ is (μ-)terminating (resp. operationally terminating); and *negative* otherwise. In order to prove these problems positive or negative, we just translate them into the corresponding problems of the aforementioned existing frameworks to prove

termination, termination of CSR, termination of CTRSs, or operational termination of CTRSs. In such frameworks, *positiveness* is actually proved as *finiteness* (the standard notion which they use, which is equivalent to our notion of positiveness).

For feasibility problems $FE(\mathcal{R}, \mu, F)$, where $\mathcal{R}$ is a CTRS, μ is a replacement map, and F is a *feasibility sequence* [8], we proceed similarly.

Remark 69 (Formal use of infeasiblity problems) In order to keep our presentation consistent, we also consider *infeasiblity problems* $IN(\mathcal{R}, \mu, F)$ as a formal class of problems, although $IN(\mathcal{R}, \mu, F)$ is positive if and only if $FE(\mathcal{R}, \mu, F)$ is negative. This is formally important to have a single definition of positive or negative proof from a proof tree when both feasibility and infeasibility problems are used. Internally, we just use feasibility problems $FE(\mathcal{R}, \mu, F)$ and realize the appropriate translations of the outcomes. ■

The following definition intentionally uses a generic, not formalized notion of *problem*. In practice, we mean any of the problems considered above.

Definition 70 (Generalized processor) *A* generalized processor P *is a partial function from problems into sets of problems; alternatively it can return "no". The domain of* P *(i.e., the set of problems for which* P *returns some answer) is denoted* $\mathcal{D}om(\mathsf{P})$.

A generalized processor P *is*

- sound *if for all* $\tau \in \mathcal{D}om(\mathsf{P})$, τ *is positive whenever* $\mathsf{P}(\tau) \neq$ *"no" and all* $\tau' \in \mathsf{P}(\tau)$ *are positive.*
- complete *if for all* $\tau \in \mathcal{D}om(\mathsf{P})$, τ *is negative whenever* $\mathsf{P}(\tau) =$ *"no" or* τ' *is negative for some* $\tau' \in \mathsf{P}(\tau)$.

4.1.2 Confluence Proof Tree

Confluence problems can be proved or disproved by using a proof tree as follows. The following definition is given for CTRSs $\mathcal{R}$. For CS-TRSs it is similar.

Definition 71 (Confluence Proof Tree) *Let* τ *be an confluence problem CR(*$\mathcal{R}$*) for some CTRS* $\mathcal{R}$*. A confluence proof tree* $\mathcal{T}$ *for* τ *is a tree whose inner nodes are labeled with problems and the leaves are labeled either with problems,* "yes" *or* "no"*. The root of* $\mathcal{T}$ *is labeled with* τ *and for every inner node* n *labeled with* τ'*, there is a processor* P *such that* $\tau' \in \mathcal{D}om(\mathsf{P})$ *and:*

1. *if* $\mathsf{P}(\tau') =$ "no" *then* n *has just one child, labeled with* "no".
2. *if* $\mathsf{P}(\tau') = \varnothing$ *then* n *has just one child, labeled with* "yes".
3. *if* $\mathsf{P}(\tau') = \{\tau_1, \ldots, \tau_k\}$ *with* $k > 0$, *then* n *has* k *children labeled with the problems* $\tau_1, \ldots, \tau_k$.

In this way, a confluence proof tree is obtain by the combination of different confluence processors.

Theorem 72 (Confluence Framework) *Let* $\mathcal{R}$ *be a CTRS and* $\mathcal{T}$ *be a confluence proof tree for CR(*$\mathcal{R}$*). Then:*

1. *if all leaves in* $\mathcal{T}$ *are labeled with* "yes" *and all involved processors are sound for the problems they are applied to, then* τ *is confluent.*
2. *if* $\mathcal{T}$ *has a leaf labeled with* "no" *and all processors in the path from* τ *to the leaf are complete for the problems they are applied to, then* τ *is non-confluent.*

Proof. We analyze the two cases:

1. Since all leaves in $\mathcal{T}$ are labeled with "yes", by definition of confluence proof tree the nodes n that immediately precede the leaves are labeled with problems τ that are *positive* (because $\mathsf{P}(\tau) = \varnothing$ for some sound confluence processor P). By definition of soundness of the processors involved in the construction of $\mathcal{T}$, the root of $\mathcal{T}$ is labeled with a *positive* confluence problem. Therefore, τ is *confluent*.
2. If $\mathcal{T}$ contains a leaf L labeled with "no", then there is a *complete* processor P such that, for the node n (with label τ_f) that immediately precedes L, we have $\mathsf{P}(\tau_f) =$ "no". Hence, τ_f is *negative*. Since all processors used on the path from the root to L are also complete, the confluence problem τ in the root of $\mathcal{T}$ is negative. Thus, $\mathcal{R}$ is not confluent.

■

After some description of preprocessing operations in Section 4.2, Sections 4.3–4.5 provide a summary of processors implemented in CONFident for use in Theorem 72.

4.2 Pre-processing

Before attempting a proof of confluence, some preprocessing is made on the input system to simplify it by removing or simplifying some rules.

4.2.1 Removing redundant rules

It is in charge of erasing rules $t \to t$.

4.2.2 Removing infeasible rules

Analogous to CleanTRSProcessor, but for CTRS: it removes rules $t \to t \Leftarrow c$. It also removes rules with infeasible conditions c using infChecker.

4.2.3 Inlining conditional rules

As explained in [29], we can often reduce the number of conditions of a rule, by using *inlining*, see Definition 53.

4.3 Solving Termination Problems

When invoking processor P_T on a termination problem $T(\mathcal{R})$ for a CTRS $\mathcal{R}$, i.e., $\mathsf{P}_\mathsf{T}(T(\mathcal{R}))$ we just calls MU-TERM to check whether the input CTRS $\mathcal{R}$ is terminating or not. Termination of CTRSs $\mathcal{R}$ is often checked by means of a preprocessing as termination of the underlying TRS $\mathcal{R}_u$.

Similarly, using P_{OT} to check an operational termination problem $OT(\mathcal{R})$ for a CTRS $\mathcal{R}$, i.e., $\mathsf{P}_{\mathsf{OT}}(OT(\mathcal{R}))$ calls MU-TERM to check whether the $\mathcal{R}$ is operationally terminating or not.

4.4 Solving joinability problems

4.4.1 Joinability processor

Checks joinability of a critical pair. $\mathsf{P}_{JO}(JO(\mathcal{R}, \pi))$ returns

- $\emptyset$, if π is joinable
- no, if π is not joinable
- $\{JO(\mathcal{R}, \pi)\}$, otherwise

CONFident can prove (non-)joinability of a CP with two different approaches.

First one is a *graph-based-rewriting*, which will create a directed graph with the possible rewritings of a term, linking nodes and permitting loops. Every time a non-explored node is analised, the *depth* of the algorithm is increased, in order to keep an eye on the number of operations, i.e., when we achieve several rewritings of a single node, only those which creates new nodes increase its depth value. The power of this approach is that is very fast and loops are ignored.

The second procedure is using *logic-based-rewriting* with the aid of infChecker, see Sections 3.1.4, 3.2.3 and 3.3.2. This has the downgrade of performance, because it usually takes some time. However, there are problems which can only be solved this way.

4.4.2 Strong joinability processor

It tries to check strong joinability of a CP $\langle s, t\rangle$ (Definition 12). $\mathsf{P}_{SJO}(JO(\mathcal{R}, \pi))$ returns

- $\emptyset$, if π is strongly-joinable
- no, if π is not strongly-joinable
- $\{JO(\mathcal{R}, \pi)\}$, otherwise

Proofs of strong joinability of specific pairs π are achieved following the ideas expressed in Section 4.4.1 (graph-based rewriting and (in)feasibility proofs with infChecker).

4.5 Solving confluence problems

4.5.1 Modular decomposition

If the system is a TRS, then this processor tries to find a *two-partition* of rules, dividing the original problem in two separated and *confluence-modular* problems $\mathcal{R}_1$ and $\mathcal{R}_2$, only if these *modularity conditions for confluence* are achieved (see Theorem 22). Either way, the original problem is returned: $\mathsf{P}_{MC}(CR(\mathcal{R}))$ returns

- $\{CR(\mathcal{R}_1), CR(\mathcal{R}_2)\}$, if $\mathcal{R}$ is a TRS and can be decomposed into $\mathcal{R}_1$ and $\mathcal{R}_2$.
- $\{CR(\mathcal{R})\}$, otherwise.

4.5.2 Weak orthogonality processor

This processor calculates the *critical pairs* and tries to prove confluence with Huet-Levy's Theorem 7 (weakly orthogonal TRSs are confluent). $\mathsf{P}_{HL}(CR(\mathcal{R},\pi))$: returns

- $\emptyset$ if $\mathcal{R}$ is a left-linear TRS and all critical pairs are trivial
- $\{CR(\mathcal{R},\pi)\}$ otherwise

4.5.3 Extended Huet processor

It checks joinability of (conditional) (μ-)critical pairs. $\mathsf{P}_{Huet}(CR(\mathcal{R},\mu))$ returns

$$\{JO(\mathcal{R},\mu,\pi_1),\ldots,JO(\mathcal{R},\mu,\pi_n)\}$$

if $\pi_1,\ldots,\pi_n$ are the (possibly conditional, or extended μ-)critical pairs of a TRS, CS-TRS, or CTRS $\mathcal{R}$. This processor is *not* sound, although it is *complete*. The next variant of P_{Huet} provides an appropriate complement.

4.5.4 Extended Huet-Newman processor

It tries to apply joinability checkings of (possibly conditional, or extended μ-)critical pairs plus Newman's lemma to TRSs (Theorem 9), CS-TRSs (Theorem 35) and CTRSs (Section 3.3.1). $\mathsf{P}_{HN}(CR(\mathcal{R},\mu))$ returns

$$\{JO(\mathcal{R},\mu,\pi_1),\ldots,JO(\mathcal{R},\mu,\pi_n),OT(\mathcal{R},\mu)\}$$

if $\pi_1,\ldots,\pi_n$ are the (possibly conditional, or extended μ-)critical pairs of a TRS, CS-TRS or CTRS $\mathcal{R}$. This processor is sound (provided that the conditions of the corresponding aforementioned results hold), although it is *not* complete. Note that, since operational termination and termination of of TRSs and CS-TRSs coincide, the problems $OT(\mathcal{R},\mu)$ are treated as termination problems $T(\mathcal{R},\mu)$ in these particular cases.

4.5.5 Confluence as canonical μ-confluence

The processor converts a TRS into a CS-TRS with the most restrictive $\mu^{can}_{\mathcal{R}}$ if the conditions in Theorem 39 are achieved. $\mathsf{P}_{GL}(CR(\mathcal{R}))$ returns

- $\{CR(\mathcal{R},\mu^{can}_{\mathcal{R}})\}$, if the conditions in Theorem 39 are achieved.
- $\{CR(\mathcal{R})\}$, otherwise

4.5.6 Confluence of CTRS as confluence of TRSs

The next processor transforms a confluence problem for a CTRS $\mathcal{R}$ into a confluence problem for a TRS $U(R)$ see Definition 49:

$$\mathsf{P}_U(CR(\mathcal{R})) = \{CR(U(\mathcal{R}))\}$$

This processor is sound but not complete. Similarly, our last processor uses $U_{opt}(\mathcal{R})$:

$$\mathsf{P}_{U_{opt}}(CR(\mathcal{R})) = \{CR(U_{opt}(\mathcal{R}))\}$$

This processor is sound but not complete.

5 Results

CONFident participated in CoCo 2021 in the categories of TRS and CTRS, achieving a first place in the confluence of CTRSs subcategory. The results are summarized in Tables 5.1–5.3:

status	ACP	CONFident	CSI	CSI_2020	CoLL-Saigawa	FORT-h+FORTify
YES	45	14	49	49	36	4
NO	24	14	26	26	19	13
MAYBE	31	72	25	25	45	83
total	100	100	100	100	100	100

Table 5.1: TRS results

status	ACP	CONFident	CSI	CSI_2020	CoLL-Saigawa
YES	24	7	28	27	13
NO	16	15	29	30	12
MAYBE	60	78	43	43	75
total	100	100	100	100	100

Table 5.2: SRS results

status	ACP	CO3	CONFident	ConCon_2020
YES	29	28	37	40
NO	15	19	24	28
MAYBE	56	53	39	31
ERRONEOUS	0	0	0	1
total	100	100	100	100

Table 5.3: CTRS results

The timeout in every subcategory was 60 seconds, with a battery of 100 problems from COPS of the corresponding category type.

As Tables 5.1, 5.2, and 5.3 show, CONFident should improve its performance in the categories of TRS and SRS. However, in the CTRS category, it was the winner of the competition.

6

Conclusions and Future Work

The main goal of this work was to develop a tool for proving confluence of TRSs, CS-TRSs, and (join, oriented, and semi-equational) CTRSs. By means of a logic-based modeling of reduction with such systems, we are able to treat them homogeneously. We translate (non-)joinability problems into (in)feasibility problems and solve them with the aid of infChecker. Although there are several tools which are able to prove confluence of TRSs and oriented CTRSs, to the best of our knowledge, CONFident is the first tool which is able to deal with join and semi-equational CTRSs. It is also the first tool which can be used to prove confluence of CS-TRSs.

We have obtained good results in the CTRS subcategory of the last International Confluence Competition (CoCo 2021) and, as a (conditional) CS-TRS subcategory is planned to be host by CoCo this year, we will whether the good results obtained on a collection of problems from the International Termination Competition, termCOMP [5, Section 3.1], see [21, Section 7.1] can be reproduced on the CoCo 2022 collection of problems (still to be defined).

CONFident achievements regarding CTRSs and CS-TRSs are good, although a lot of work is still pending on TRS/SRS to improve the functionalities of the tool. CONFident is also *in progress* regarding CTRSs and CS-TRSs. For semiequational CTRSs we plan to implement new results in the future. Also, modularity of CS-TRSs, which is underexplored at the moment, is a good subject of study.

Bibliography

[1] Nirina Andrianarivelo and Pierre Réty. Confluence of prefix-constrained rewrite systems. In Hélène Kirchner, editor, *3rd International Conference on Formal Structures for Computation and Deduction, FSCD 2018, July 9-12, 2018, Oxford, UK*, volume 108 of *LIPIcs*, pages 6:1–6:15. Schloss Dagstuhl - Leibniz-Zentrum für Informatik, 2018. doi:10.4230/LIPIcs.FSCD.2018.6.

[2] Franz Baader and Tobias Nipkow. *Term rewriting and all that.* Cambridge University Press, 1998.

[3] Francisco Durán, Steven Eker, Santiago Escobar, Narciso Martí-Oliet, José Meseguer, Rubén Rubio, and Carolyn L. Talcott. Programming and symbolic computation in Maude. *J. Log. Algebr. Meth. Program.*, 110, 2020. doi:10.1016/j.jlamp.2019.100497.

[4] J. Giesl, R. Thiemann, P. Schneider-Kamp, and S. Falke. Mechanizing and Improving Dependency Pairs. *Journal of Automatic Reasoning*, 37(3):155–203, 2006. doi:https://doi.org/10.1007/s10817-006-9057-7.

[5] Jürgen Giesl, Albert Rubio, Christian Sternagel, Johannes Waldmann, and Akihisa Yamada. The termination and complexity competition. In Dirk Beyer, Marieke Huisman, Fabrice Kordon, and Bernhard Steffen, editors, *Tools and Algorithms for the Construction and Analysis of Systems - 25 Years of TACAS: TOOLympics, Held as Part of ETAPS 2019, Prague, Czech Republic, April 6-11, 2019, Proceedings, Part III*, volume 11429 of *Lecture Notes in Computer Science*, pages 156–166. Springer, 2019. doi:10.1007/978-3-030-17502-3_10.

[6] Bernhard Gramlich and Salvador Lucas. Generalizing newman's lemma for left-linear rewrite systems. In Frank Pfenning, editor, *Term Rewriting and Applications, 17th International Conference, RTA 2006, Seattle, WA, USA, August 12-14, 2006, Proceedings*, volume 4098 of *Lecture Notes in Computer Science*, pages 66–80. Springer, 2006. doi:10.1007/11805618_6.

[7] R. Gutiérrez and S. Lucas. Automatically Proving and Disproving Feasibility Conditions. In N. Peltier and V. Sofronie-Stokkermans, editors, *Automated Reasoning*, pages 416–435. Springer International Publishing, 2020. doi:https://doi.org/10.1007/978-3-030-51054-1_27.

[8] Raúl Gutiérrez and Salvador Lucas. Automatically proving and disproving feasibility conditions. In Nicolas Peltier and Viorica Sofronie-Stokkermans, editors, *Automated Reasoning - 10th International Joint Conference, IJCAR 2020, Paris, France, July 1-4, 2020, Proceedings, Part II*, volume 12167 of *Lecture Notes in Computer Science*, pages 416–435. Springer, 2020. doi:10.1007/978-3-030-51054-1_27.

[9] Raúl Gutiérrez and Salvador Lucas. MU-TERM: Verify termination properties automatically (system description). In Nicolas Peltier and Viorica Sofronie-Stokkermans, editors, *Automated Reasoning - 10th International Joint Conference, IJCAR 2020, Paris, France, July 1-4, 2020, Proceedings, Part II*, volume 12167 of *Lecture Notes in Computer Science*, pages 436–447. Springer, 2020. doi:10.1007/978-3-030-51054-1_28.

[10] Raúl Gutiérrez, Salvador Lucas, and Miguel Vítorres. Confluence Of Conditional Rewriting In Logic Form. In Chandra Chekuri and Mikolaj Bojanczyk, editors, *41th IARCS Annual Conference on Foundations of Software Technology and Theoretical Computer Science, FSTTCS 2021, December 15-17, 2021, Virtual conference*, LIPIcs, page submitted, 2021.

[11] J. Roger Hindley and Jonathan P. Seldin. *Introduction to Combinators and Lambda-Calculus*. Cambridge University Press, 1986.

[12] Nao Hirokawa, Julian Nagele, and Aart Middeldorp. Cops and CoCoWeb: Infrastructure for Confluence Tools. In Didier Galmiche, Stephan Schulz, and Roberto Sebastiani, editors, *Automated Reasoning - 9th International Joint Conference, IJCAR 2018, Held as Part of the Federated Logic Conference, FloC 2018, Oxford, UK, July 14-17, 2018, Proceedings*, volume 10900 of *Lecture Notes in Computer Science*, pages 346–353. Springer, 2018. doi:10.1007/978-3-319-94205-6_23.

[13] Gérard P. Huet. Confluent reductions: Abstract properties and applications to term rewriting systems: Abstract properties and applications to term rewriting systems. *J. ACM*, 27(4):797–821, 1980. doi:10.1145/322217.322230.

[14] Gérard P. Huet and Jean-Jacques Lévy. Computations in orthogonal rewriting systems, I. In Jean-Louis Lassez and Gordon D. Plotkin, editors, *Computational Logic - Essays in Honor of Alan Robinson*, pages 395–414. The MIT Press, 1991.

[15] Gérard P. Huet and Jean-Jacques Lévy. Computations in orthogonal rewriting systems, II. In Jean-Louis Lassez and Gordon D. Plotkin, editors, *Computational Logic - Essays in Honor of Alan Robinson*, pages 415–443. The MIT Press, 1991.

[16] Donald E. Knuth and Peter E. Bendix. Simple word problems in universal algebra. In J. Leech, editor, *Computational Problems in Abstract Algebra*, pages 263–297. Pergamon Press, 1970.

[17] Salvador Lucas. Context-sensitive rewriting. *ACM Comput. Surv.*, 53(4):78:1–78:36, 2020. doi:10.1145/3397677.

[18] Salvador Lucas, Claude Marché, and José Meseguer. Operational termination of conditional term rewriting systems. *Inf. Process. Lett.*, 95(4):446–453, 2005.

[19] Salvador Lucas and José Meseguer. Dependency pairs for proving termination properties of conditional term rewriting systems. *J. Log. Algebr. Meth. Program.*, 86(1):236–268, 2017. doi:10.1016/j.jlamp.2016.03.003.

[20] Salvador Lucas, José Meseguer, and Raúl Gutiérrez. The 2D Dependency Pair Framework for conditional rewrite systems. Part I: Definition and basic processors. *J. Comput. Syst. Sci.*, 96:74–106, 2018. doi:10.1016/j.jcss.2018.04.002.

[21] Salvador Lucas, Miguel Vítores, and Raúl Gutiérrez. Proving and disproving confluence of context-sensitive rewriting. *Journal of Logical and Algebraic Methods in Programming*, 126:100749, 2022. doi:10.1016/j.jlamp.2022.100749.

[22] William McCune. Prover9 & Mace4. Technical report, 2005–2010. URL: http://www.cs.unm.edu/~mccune/prover9/.

[23] A. Middeldorp, J. Nagele, and K. Shintani. Confluence Competition 2019. In D. Beyer, M. Huisman, F. Kordon, and B. Steffen, editors, *Proc. of Tools and Algorithms for the Construction and Analysis of Systems, TACAS'19*, pages 25–40. Springer, 2019. doi:10.1007/978-3-030-17502-3_2.

[24] Aart Middeldorp and Erik Hamoen. Completeness results for basic narrowing. *Appl. Algebra Eng. Commun. Comput.*, 5:213–253, 1994. doi:10.1007/BF01190830.

[25] Enno Ohlebusch. On the modularity of confluence of constructor-sharing term rewriting systems. In Sophie Tison, editor, *Trees in Algebra and Programming - CAAP'94, 19th International Colloquium, Edinburgh, UK, April 11-13, 1994, Proceedings*, volume 787 of *Lecture Notes in Computer Science*, pages 261–275. Springer, 1994. doi:10.1007/BFb0017487.

[26] Enno Ohlebusch. *Advanced topics in term rewriting*. Springer, 2002.

[27] Jean-Claude Raoult and Jean Vuillemin. Operational and semantic equivalence between recursive programs. *J. ACM*, 27(4):772–796, 1980. doi:10.1145/322217.322229.

[28] Franziska Rapp and Aart Middeldorp. FORT 2.0. In Didier Galmiche, Stephan Schulz, and Roberto Sebastiani, editors, *Automated Reasoning - 9th International Joint Conference, IJCAR 2018, Held as Part of the Federated Logic Conference, FloC 2018, Oxford, UK, July 14-17, 2018, Proceedings*, volume 10900 of *Lecture Notes in Computer Science*, pages 81–88. Springer, 2018. doi:10.1007/978-3-319-94205-6_6.

[29] Christian Sternagel and Thomas Sternagel. Certifying confluence of quasi-decreasing strongly deterministic conditional term rewrite systems. In Leonardo de Moura, editor, *Automated Deduction - CADE 26 - 26th International Conference on Automated Deduction, Gothenburg, Sweden, August 6-11, 2017, Proceedings*, volume 10395 of *Lecture Notes in Computer Science*, pages 413–431. Springer, 2017. doi:10.1007/978-3-319-63046-5_26.

[30] Taro Suzuki, Aart Middeldorp, and Tetsuo Ida. Level-confluence of conditional rewrite systems with extra variables in right-hand sides. In Jieh Hsiang, editor, *Rewriting Techniques and Applications, 6th International Conference, RTA-95, Kaiserslautern, Germany, April 5-7, 1995, Proceedings*, volume 914 of *Lecture Notes in Computer Science*, pages 179–193. Springer, 1995. doi:10.1007/3-540-59200-8_56.

[31] Terese. *Term rewriting systems*, volume 55 of *Cambridge tracts in theoretical computer science*. Cambridge University Press, 2003.

[32] Yoshihito Toyama. On the church-rosser property for the direct sum of term rewriting systems. *J. ACM*, 34(1):128–143, 1987. doi:10.1145/7531.7534.