# Infeasible Conditional Critical Pairs*

Thomas Sternagel and Aart Middeldorp

University of Innsbruck, Innsbruck, Austria
{thomas.sternagel, aart.middeldorp}@uibk.ac.at

## 1 Introduction

This paper is concerned with automatically proving (non-)confluence of conditional term rewrite systems (CTRSs). Although confluence of CTRSs has been investigated for decades, the first tools (CO3[1] and ConCon [8]) appeared in 2014. Several of the techniques implemented in these tools would benefit if certain conditional critical pairs (CCPs) are determined to be *infeasible*, which means that their conditional parts are not satisfiable. For instance, the CCP $\mathsf{f}(\mathsf{c}) \approx \mathsf{b} \Leftarrow x \approx \mathsf{a}, x \approx \mathsf{c}$ of the oriented CTRS $\mathcal{R}$

$$\mathsf{f}(\mathsf{g}(x)) \to \mathsf{b} \Leftarrow x \approx \mathsf{a} \qquad\qquad \mathsf{g}(x) \to \mathsf{c} \Leftarrow x \approx \mathsf{c}$$

is infeasible since no term rewrites to both $\mathsf{a}$ and $\mathsf{c}$. Hence $\mathcal{R}$ is orthogonal and thus confluent.

In this paper we present an overview of infeasibility methods for oriented 3-CTRSs, one of the most popular types of conditional rewriting. In such systems extra variables in conditions and right-hand sides of rewrite rules are allowed. Moreover, satisfiability of the conditions amounts to reachability. As a consequence of the latter, establishing infeasibility is similar to the problem of eliminating arrows in dependency graph approximations, a problem which has been investigated extensively in the literature. The difference is that we deal with CTRSs and the terms we test may share variables.

In the sequel we summarize the methods that we have analyzed and adapted for infeasibility. The methods have been implemented in our confluence tool ConCon and experimental data will be presented.

## 2 Preliminaries

We assume familiarity with (conditional) term rewriting and related topics [7]. We very briefly recall important concepts that are used in the sequel.

Given two variants $\ell_1 \to r_1 \Leftarrow c_1$ and $\ell_2 \to r_2 \Leftarrow c_2$ without common variables of rules in a CTRS $\mathcal{R}$, a position $p \in \mathcal{P}\mathsf{os}_{\mathcal{F}}(\ell_2)$, and an mgu $\sigma$ of $\ell_1$ and $\ell_2|_p$, the conditional equation $\ell_2\sigma[r_1\sigma]_p \approx r_2\sigma \Leftarrow c_1\sigma, c_2\sigma$ is a CCP of $\mathcal{R}$. As usual, we exclude the case that $p = \epsilon$ and the rules are variants of the same rule. A CCP $s \approx t \Leftarrow s_1 \approx t_1, \ldots, s_k \approx t_k$ is *infeasible* if there is no substitution $\sigma$ such that $s_i\sigma \to_{\mathcal{R}}^* t_i\sigma$ for all $1 \leqslant i \leqslant k$. We write $\mathsf{cs}(\vec{s})$ and $\mathsf{cs}(\vec{t})$ for the terms $\mathsf{cs}(s_1, \ldots, s_k)$ and $\mathsf{cs}(t_1, \ldots, t_k)$. Here $\mathsf{cs}$ is a fresh function symbol of arity $k$.

The set of ground instances of a term $t$ is denoted by $\Sigma(t)$ and we write $s \nabla t$ to denote that the terms $s$ and $t$ are unifiable. The sets of $\mathcal{R}$-ancestors and $\mathcal{R}$-descendants of a set of terms $T$ are defined as $(\to_{\mathcal{R}}^*)[T] = \{s \mid s \to_{\mathcal{R}}^* t \text{ for some } t \in T\}$ and $[T](\to_{\mathcal{R}}^*) = \{t \mid s \to_{\mathcal{R}}^* t \text{ for some } s \in T\}$, respectively. A TRS $\mathcal{R}$ is called *growing* if variables in $\mathcal{V}\mathsf{ar}(\ell) \cap \mathcal{V}\mathsf{ar}(r)$ occur at depth at most one in $\ell$, for all $\ell \to r \in \mathcal{R}$. The *growing approximation* [6] of a TRS $\mathcal{R}$ is denoted by $\mathsf{g}(\mathcal{R})$. We

*The research described in this paper is supported by FWF (Austrian Science Fund) project I963.

[1] http://www.trs.cm.is.nagoya-u.ac.jp/co3/

call $\mathcal{R}$ *regularity preserving* if $[T](\to_{\mathcal{R}}^*)$ is regular whenever $T$ is regular. We write $\mathsf{ren}(t)$ for a linearization of the term $t$ using fresh variables. As usual $\mathcal{R}^{-1}$ denotes the inverse of a TRS $\mathcal{R}$.

To apply methods developed for unconditional TRSs we need some transformation from CTRSs to TRSs. In its simplest form this means to just forget about the conditions, giving rise to the *underlying* TRS $\mathcal{R}_{\mathsf{u}} = \{\ell \to r \mid \ell \to r \Leftarrow c \in \mathcal{R}\}$ of a CTRS $\mathcal{R}$. More sophisticated transformations modify the signature. For that reason a transformation $\mathsf{T}$ comes equipped with an encoding function $\sharp\colon \mathcal{T}(\mathcal{F},\mathcal{V}) \to \mathcal{T}(\mathcal{F}',\mathcal{V})$ and a partial decoding function $\flat\colon \mathcal{T}(\mathcal{F}',\mathcal{V}) \to \mathcal{T}(\mathcal{F},\mathcal{V})$. Here $\mathcal{F}$ is the signature of the CTRS $\mathcal{R}$ under consideration, $\mathcal{F}'$ is the signature of the transformed TRS $\mathsf{T}(\mathcal{R})$, and we require that $\flat(\sharp(t)) = t$ for all terms $t \in \mathcal{T}(\mathcal{F},\mathcal{V})$. In case of $\mathcal{R}_{\mathsf{u}}$ both $\sharp$ and $\flat$ are just the identity. To be useful for infeasibility checking a transformation $(\mathsf{T},\sharp,\flat)$ has to be *complete*, i.e., if $s \to_{\mathcal{R}}^* t$ then $\sharp(s) \to_{\mathsf{T}(\mathcal{R})}^* \sharp(t)$ for all terms $s,t \in \mathcal{T}(\mathcal{F},\mathcal{V})$. The transformation from $\mathcal{R}$ to $\mathcal{R}_{\mathsf{u}}$ obviously has this property. In our experiments we also employed other transformations which are known to be complete for certain kinds of CTRSs, including the (optimized) unraveling $\mathsf{U}_{\mathsf{opt}}$ [4,7] as well as the structure-preserving transformation $\mathsf{SR}$ [10].

## 3 Unification

In [8] we already reported on the use of the $\mathsf{tcap}$ function in $\mathsf{ConCon}$ for checking infeasibility. Given a TRS $\mathcal{R}$, it is defined as $\mathsf{tcap}_{\mathcal{R}}(t) = u$ if $t = f(t_1,\dots,t_n)$ and $u \mathrel{\not\triangledown} \ell$ for all $\ell \to r \in \mathcal{R}$, and $\mathsf{tcap}_{\mathcal{R}}(t) = y$ otherwise, where $u = f(\mathsf{tcap}_{\mathcal{R}}(t_1),\dots,\mathsf{tcap}_{\mathcal{R}}(t_n))$ and $y$ is a fresh variable.

**Lemma 3.1.** *Let $\mathcal{R}$ be a TRS. For terms $s$ and $t$, if $\mathsf{tcap}_{\mathcal{R}}(s) \mathrel{\not\triangledown} t$ then $s\sigma \not\to_{\mathcal{R}}^* t\tau$ for all substitutions $\sigma$ and $\tau$.*

In [8] we used the above lemma to test the conditions in CCPs separately. Here we combine the conditions to obtain a strictly more powerful criterion for infeasibility. Another difference is that the criterion is now parameterized by an arbitrary complete transformation.

**Corollary 3.2** ($\mathsf{TCAP}$)**.** *Let $\mathcal{R}$ be an oriented 3-CTRS $\mathcal{R}$ and $(\mathsf{T},\sharp,\flat)$ a complete transformation. A CCP $s \approx t \Leftarrow s_1 \approx t_1,\dots,s_k \approx t_k$ of $\mathcal{R}$ is infeasible if $\mathsf{tcap}_{\mathsf{T}(\mathcal{R})}(\mathsf{cs}(\sharp(\vec{s}))) \mathrel{\not\triangledown} \mathsf{cs}(\sharp(\vec{t}))$.*

**Example 3.3.** Consider the CTRS $\mathcal{R}$ consisting of the two rules

$$\mathsf{f}(x) \to \mathsf{a} \;\Leftarrow\; \mathsf{a} \approx x \qquad\qquad \mathsf{f}(x) \to \mathsf{b} \;\Leftarrow\; \mathsf{b} \approx x$$

The two CCPs $\mathsf{a} \approx \mathsf{b} \Leftarrow \mathsf{a} \approx x$, $\mathsf{b} \approx x$ and $\mathsf{b} \approx \mathsf{a} \Leftarrow \mathsf{b} \approx x$, $\mathsf{a} \approx x$ are infeasible by Corollary 3.2 because $\mathsf{cs}(\mathsf{a},\mathsf{b})$ and $\mathsf{cs}(\mathsf{b},\mathsf{a})$ do not unify with $\mathsf{cs}(x,x)$.

## 4 Tree Automata Techniques

Tree automata techniques are another method that were used to approximate dependency graphs in termination analysis. The following result is from [6].

**Lemma 4.1.** *Let $\mathcal{R}$ be a regularity preserving TRS. For terms $s$ and $t$, if $[\Sigma(\mathsf{ren}(s))](\to_{\mathcal{R}}^*) \cap \Sigma(t) = \varnothing$ or $\Sigma(s) \cap (\to_{\mathcal{R}}^*)[\Sigma(\mathsf{ren}(t))] = \varnothing$ then $s\sigma \not\to_{\mathcal{R}}^* t\tau$ for all substitutions $\sigma$ and $\tau$.*

From this result we obtain the following general infeasibility criterion.

**Corollary 4.2** (TAC). *Let $\mathcal{R}$ be an oriented 3-CTRS $\mathcal{R}$ and $(\mathsf{T}, \sharp, \flat)$ a complete transformation. A CCP $s \approx t \Leftarrow s_1 \approx t_1, \ldots, s_k \approx t_k$ of $\mathcal{R}$ is* infeasible *if one of the following intersections is empty:*[2]

$$[\Sigma(\mathsf{ren}(\mathsf{cs}(\sharp(\vec{s}))))](\xrightarrow[\mathsf{T}(\mathcal{R})]{*}) \cap \Sigma(\mathsf{cs}(\sharp(\vec{t}))) \qquad [\Sigma(\mathsf{ren}(\mathsf{cs}(\sharp(\vec{t}))))](\xrightarrow[\mathsf{T}(\mathcal{R})^{-1}]{*}) \cap \Sigma(\mathsf{cs}(\sharp(\vec{s})))$$

To obtain an effective criterion for infeasibility based on Corollary 4.2, we need to construct a tree automaton that over-approximates the ground terms in the sets of descendants or ancestors. There are basically three ways to achieve this.

The first method is to replace $\mathsf{T}(\mathcal{R})$ and $\mathsf{T}(\mathcal{R})^{-1}$ by their *growing approximations*. Adopting a construction of Jacquemard [5], we obtain an exact tree automaton representation of the over-approximation (due to the growing approximation) of the sets of ancestors. Since the growing approximation of $\mathsf{T}(\mathcal{R})^{-1}$ is different from the inverse of the growing approximation of $\mathsf{T}(\mathcal{R})$, we test the emptiness of the following two intersections:

$$\Sigma(\mathsf{cs}(\sharp(\vec{s}))) \cap (\xrightarrow[\mathsf{g}(\mathsf{T}(\mathcal{R}))]{*})[\Sigma(\mathsf{ren}(\mathsf{cs}(\sharp(\vec{t}))))] \qquad \Sigma(\mathsf{cs}(\sharp(\vec{t}))) \cap (\xrightarrow[\mathsf{g}(\mathsf{T}(\mathcal{R})^{-1})]{*})[\Sigma(\mathsf{ren}(\mathsf{cs}(\sharp(\vec{s}))))]$$

This method will be referred to as eTAC in the sequel.

In the second method we attempt to construct a tree automaton for sets of descendants by a process known as tree automata completion, an idea which goes back to Genet [3]. This process is parameterized by an abstraction function which limits the number of newly generated states during completion, thereby providing a trade-off between the termination behavior (and thus runtime) of the process and its accuracy. It takes the tree automaton which represents a regular set (like $\Sigma(\mathsf{ren}(\mathsf{cs}(\sharp(\vec{s}))))$) and a left-linear TRS (like $\mathsf{T}(\mathcal{R})$) as input. Compatibility violations between the tree automaton and the TRS are resolved in an iterative process. Termination depends on the employed abstraction function. We test both $\mathsf{T}(\mathcal{R})$ and $\mathsf{T}(\mathcal{R})^{-1}$:

$$[\Sigma(\mathsf{ren}(\mathsf{cs}(\sharp(\vec{s}))))](\xrightarrow[\mathsf{T}(\mathcal{R})]{*}) \cap \Sigma(\mathsf{cs}(\sharp(\vec{t}))) \qquad [\Sigma(\mathsf{ren}(\mathsf{cs}(\sharp(\vec{t}))))](\xrightarrow[\mathsf{T}(\mathcal{R})^{-1}]{*}) \cap \Sigma(\mathsf{cs}(\sharp(\vec{s})))$$

This method will be referred to as uTAC in the sequel.

In [1] Feuillade and Genet present a version of tree automata completion which operates directly on CTRSs. They showed that this direct approach results in smaller tree automata (thereby reducing the possibility of divergence of the completion process) compared to tree automata completion applied to $\mathcal{R}_{\mathsf{u}}$. We adapted the procedure, which is defined in [1] for join 1-CTRSs with at most one condition per rule, to oriented 3-CTRSs with an arbitrary number of conditions. Here we check the following two intersections for emptiness:

$$[\Sigma(\mathsf{ren}(\mathsf{cs}(\sharp(\vec{s}))))](\xrightarrow[\mathcal{R}]{*}) \cap \Sigma(\mathsf{cs}(\sharp(\vec{t}))) \qquad [\Sigma(\mathsf{ren}(\mathsf{cs}(\sharp(\vec{t}))))](\xrightarrow[\mathcal{R}^{-1}]{*}) \cap \Sigma(\mathsf{cs}(\sharp(\vec{s})))$$

This third method will be referred to as cTAC in the sequel.

# 5 Equational Reasoning

The final method, referred to as ER in Section 6, that we present for infeasibility was also first used for computing dependency graphs [9]. It employs Waldmeister [2], a powerful automatic

[2] Of course we can also use ancestors instead of descendants, provided we interchange $\mathsf{cs}(\sharp(\vec{s}))$ and $\mathsf{cs}(\sharp(\vec{t}))$ in the intersections. Depending on the concrete algorithm, ancestors or descendants might be preferable.

| | TCAP | | | eTAC | | | ER | | | all |
|---|---|---|---|---|---|---|---|---|---|---|
| | $\mathcal{R}_\mathsf{u}$ | $\mathsf{U}_\mathsf{opt}$ | SR | $\mathcal{R}_\mathsf{u}$ | $\mathsf{U}_\mathsf{opt}$ | SR | $\mathcal{R}_\mathsf{u}$ | $\mathsf{U}_\mathsf{opt}$ | SR | |
| confluent | 2 | 2 | 0 | 5 | 4 | 0 | 5 | 0 | 0 | 8 |
| infeasible | 16 | 16 | 2 | 46 | 40 | 0 | 17 | 0 | 0 | 61 |
| maybe | 220 | 220 | 234 | 190 | 196 | 236 | 219 | 236 | 236 | 175 |
| timeout | 2 | 2 | 2 | 2 | 6 | 17 | 2 | 2 | 2 | 1 |
| avg. time | 5.66 | 5.75 | 5.70 | 7.12 | 12.84 | 26.68 | 5.82 | 5.76 | 6.91 | 12.30 |

Table 1: Results for 46 oriented 3-CTRSs with at least one CCP (236 CCPs in total).

theorem prover for equational logic with uninterpreted function symbols. Waldmeister uses a variant of ordered completion to determine for a given set of equations $\mathcal{R}$ and a goal equation (called conclusion) $s \approx t$ whether there exist substitutions $\sigma$ and $\tau$ such that $s\sigma \leftrightarrow^*_\mathcal{R} t\tau$. If Waldmeister refutes the conclusion then surely there are no substitutions $\sigma$ and $\tau$ such that $s\sigma \to^*_\mathcal{R} t\tau$.

**Example 5.1.** Consider system 361 from Cops:[3]

$$\begin{aligned}
0 \leqslant x &\to \mathsf{true} & \mathsf{s}(x) > 0 &\to \mathsf{true} & x - 0 &\to x \\
\mathsf{s}(x) \leqslant \mathsf{s}(y) &\to x \leqslant y & \mathsf{s}(x) > \mathsf{s}(y) &\to x > y & 0 - x &\to 0 \\
x \div y &\to \langle 0, y\rangle \Leftarrow y > x \approx \mathsf{true} & & & \mathsf{s}(x) - \mathsf{s}(y) &\to x - y \\
x \div y &\to \langle \mathsf{s}(q), r\rangle \Leftarrow y \leqslant x \approx \mathsf{true},\ (x - y) \div y \approx \langle q, r\rangle & & &
\end{aligned}$$

This CTRS has two trivial unconditional CPs and one (modulo symmetry) CCP

$$\langle 0, x\rangle \approx \langle \mathsf{s}(y), z\rangle \Leftarrow x \leqslant w \approx \mathsf{true},\ (w - x) \div x \approx \langle y, z\rangle,\ x > w \approx \mathsf{true}$$

which is infeasible because of the contradictory conditions $x \leqslant w \approx \mathsf{true}$ and $x > w \approx \mathsf{true}$. This is confirmed by Waldmeister in conjunction with the $\mathcal{R} \mapsto \mathcal{R}_\mathsf{u}$ transformation.[4]

# 6 Experiments

Our test bed consists of 46 oriented 3-CTRSs from the Cops problem collection which have *at least one* CCP. These 46 CTRSs have 236 CCPs. Our experiments have been conducted on a 64 bit GNU/Linux machine. The time limit was set to 60 seconds. In Table 1 we compare combinations of the various infeasibility methods with different transformations.

The row labeled 'confluent' lists the number of systems which are confluent but could not be shown confluent *without any* infeasibility methods. The next line lists the number of CCPs which could be shown to be infeasible with each method. The rows labeled 'maybe' and 'timeout' give the number of CCPs for which infeasibility could not be shown (within the time limit).

Without any infeasibility checking ConCon could show 11 CTRSs confluent and 1 CTRS non-confluent (with 2 timeouts) with an average time of 5.8 s. Using TCAP yields another 2 confluent systems but these CTRSs are also handled by eTAC and ER. The more involved tree automata methods uTAC and cTAC could not improve upon eTAC and are not listed in the table.

[3] http://cops.uibk.ac.at

[4] The eTAC method together with the unraveling $\mathsf{U}_\mathsf{opt}$ also shows infeasibility of this CCP.

The eTAC and ER methods are incomparable in power, the first one succeeding on systems 288, 292, 330, 336, 361, and 409, the second one on systems 336, 361, 406, 407, and 409. The unraveling $\mathsf{U}_{\mathsf{opt}}$ could improve upon $\mathcal{R}_{\mathsf{u}}$ in one instance (system 361) while SR could not. All in all, ConCon can show 19 of the 46 systems confluent using the infeasibility methods, which handle 61 of the 236 CCPs (with eTAC handling the most).

We conclude with an example where the methods of this paper are not helpful.

**Example 6.1.** Consider system 327 from Cops:

$$\begin{aligned}
\mathsf{gcd}(x,x) &\to x & x < 0 &\to \mathsf{false} & 0 - \mathsf{s}(y) &\to 0\\
\mathsf{gcd}(\mathsf{s}(x),0) &\to \mathsf{s}(x) & 0 < \mathsf{s}(y) &\to \mathsf{true} & x - 0 &\to x\\
\mathsf{gcd}(0,\mathsf{s}(y)) &\to \mathsf{s}(y) & \mathsf{s}(x) < \mathsf{s}(y) &\to x < y & \mathsf{s}(x) - \mathsf{s}(y) &\to x - y\\
\mathsf{gcd}(\mathsf{s}(x),\mathsf{s}(y)) &\to \mathsf{gcd}(x-y,\mathsf{s}(y)) \Leftarrow y < x \approx \mathsf{true}\\
\mathsf{gcd}(\mathsf{s}(x),\mathsf{s}(y)) &\to \mathsf{gcd}(\mathsf{s}(x),y-x) \Leftarrow x < y \approx \mathsf{true}
\end{aligned}$$

This CTRS has six CCPs of which we show two (the conditions of the others are similar):

$$\begin{aligned}
&\mathsf{gcd}(\mathsf{s}(x),y-x) \approx \mathsf{gcd}(x-y,\mathsf{s}(y)) \Leftarrow y < x \approx \mathsf{true},\ x < y \approx \mathsf{true}\\
&\mathsf{gcd}(x-x,\mathsf{s}(x)) \approx \mathsf{s}(x) \Leftarrow x < x \approx \mathsf{true}
\end{aligned}$$

These CCPs are obviously infeasible, but this cannot be shown by the methods of this paper. For instance, when using $\mathcal{R}_{\mathsf{u}}$ we open the door for inconsistencies:

$$\mathsf{s}(0) \mathrel{{}^*\!\leftarrow} \mathsf{gcd}(\mathsf{s}(0),\mathsf{s}(0)) \mathrel{{}^*\!\leftarrow} \mathsf{gcd}(\mathsf{s}(\mathsf{s}(0)),\mathsf{s}(0)) \to^* \mathsf{gcd}(0,\mathsf{s}(\mathsf{s}(0))) \to^* \mathsf{s}(\mathsf{s}(0))$$

and thus $\mathsf{gcd}(\mathsf{s}(\mathsf{s}(0)),\mathsf{s}(0)) < \mathsf{gcd}(\mathsf{s}(\mathsf{s}(0)),\mathsf{s}(0)) \to^* \mathsf{s}(0) < \mathsf{s}(\mathsf{s}(0)) \to^* \mathsf{true}$. Consequently, we may substitute $\mathsf{gcd}(\mathsf{s}(\mathsf{s}(0)),\mathsf{s}(0))$ for both $x$ and $y$ to satisfy the conditions of the CCPs.

# References


[1] G. Feuillade and T. Genet. Reachability in conditional term rewriting systems. In *Proc. 4th FTP*, volume 86 of *ENTCS*, pages 133–146, 2003.

[2] J.-M. Gaillourdet, Th. Hillenbrand, B. Löchner, and H. Spies. The new Waldmeister loop at work. In *Proc. 19th CADE*, volume 2741 of *LNCS*, pages 317–321. Springer, 2003.

[3] T. Genet. Decidable approximations of sets of descendants and sets of normal forms. In *Proc. 9th RTA*, volume 1379 of *LNCS*, pages 151–165, 1998.

[4] K. Gmeiner, N. Nishida, and B. Gramlich. Proving confluence of conditional term rewriting systems via unravelings. In *Proc. 2nd IWC*, pages 35–39, 2013.

[5] F. Jacquemard. Decidable approximations of term rewriting systems. In *Proc. 7th RTA*, volume 1103 of *LNCS*, pages 362–376, 1996.

[6] A. Middeldorp. Approximating dependency graphs using tree automata techniques. In *Proc. 1st IJCAR*, volume 2083 of *LNCS*, pages 593–610. Springer, 2001.

[7] E. Ohlebusch. *Advanced Topics in Term Rewriting*. Springer, 2002.

[8] T. Sternagel and A. Middeldorp. Conditional confluence (system description). In *Proc. Joint 25th RTA and 12th TLCA*, volume 8560 of *LNCS*, pages 456–465, 2014.

[9] H. Zankl and A. Middeldorp. Equational reasoning for termination of rewriting. In *Proc. 10th WST*, pages 112–115, 2009.

[10] T. Şerbănuţă and G Roşu. Computationally equivalent elimination of conditions. In *Proc. 6th RTA*, volume 4098 of *LNCS*, pages 19–34. Springer, 2006.