Reachability Analysis for Termination and Confluence of Rewriting

Christian Sternagel[1][0000−0001−9864−1014] and
Akihisa Yamada[2][0000−0001−8872−2240]

[1] University of Innsbruck, Innsbruck, Austria
christian.sternagel@uibk.ac.at
[2] National Institute of Informatics, Tokyo, Japan
akihisayamada@nii.ac.jp

Abstract. In term rewriting, reachability analysis is concerned with the problem of deciding whether or not one term is reachable from another by rewriting. Reachability analysis has several applications in termination and confluence analysis of rewrite systems. We give a unified view on reachability analysis for rewriting with and without conditions by means of what we call reachability constraints. Moreover, we provide several techniques that fit into this general framework and can be efficiently implemented. Our experiments show that these techniques increase the power of existing termination and confluence tools.

Keywords: Reachability analysis · Termination · Confluence · Conditional term rewriting · Infeasibility.

1 Introduction

Reachability analysis for term rewriting [6] is concerned with the problem of, given a rewrite system $\mathcal{R}$, a source term s and a target term t, deciding whether the source reduces to the target by rewriting, which is usually written $s \to_{\mathcal{R}}^* t$. A useful generalization of this problem is the "(un)satisfiability" of the following reachability problem: given terms s and t containing variables, decide whether there is a substitution σ such that $s\sigma \to_{\mathcal{R}}^* t\sigma$ or not. This problem, also called (in)feasibility by Lucas and Guitiérrez [11], has various applications in termination and confluence analysis for plain and conditional rewriting.

This can be understood as a form of safety analysis, as illustrated below.

Example 1. Let $\mathcal{R}$ be a term rewrite system consisting of the following rules for division (where s stands for "successor"):

$$x - \mathsf{0} \to x \qquad \mathsf{s}(x) - \mathsf{s}(y) \to x - y \qquad \mathsf{0} \div \mathsf{s}(y) \to \mathsf{0}$$
$$\mathsf{s}(x) \div \mathsf{s}(y) \to \mathsf{s}((x - y) \div \mathsf{s}(y)) \qquad x \div \mathsf{0} \to \mathsf{err}(\texttt{"division by zero"})$$

The question "Can division yield an error?" is naturally formulated as the satisfiability of reachability from $x \div y$ to $\mathsf{err}(z)$. Unsurprisingly, the solution

$$\sigma = [y \mapsto \mathsf{0}, z \mapsto \texttt{"division by zero"}]$$

shows that it is actually possible to obtain an error.

In termination analysis we are typically interested in unsatisfiability of reachability and can thereby rule out certain recursive calls as potential source of non-termination. For confluence analysis of conditional term rewriting, infeasibility is crucial: some other techniques do not apply before critical pairs are shown infeasible, and removal of infeasible rules simplifies proofs.

In this work we provide a formal framework that allows us to uniformly speak about (un)satisfiability of reachability for plain and conditional rewriting, and give several techniques that are useful in practice.

More specifically, our contributions are as follows:

- We introduce the syntax and semantics of *reachability constraints* (Section 3) and formulate their satisfiability problem. We recast several concrete techniques for reachability analysis in the resulting framework.
- We present a new, simple, and efficient technique for reachability analysis based on what we call the *symbol transition graph* of a rewrite system (Section 4.1) and extend it to conditional rewriting (Section 5.2).
- Additionally, we generalize the prevalent existing technique for term rewriting to what we call *look-ahead reachability* (Section 4.2) and extend it to the conditional case (Section 5.3).
- Then, we present a new result for conditional rewriting that is useful for proving conditional rules infeasible (Section 5.1).
- Finally, we evaluate the impact of our work on existing automated tools NaTT [16] and ConCon [13] (Section 6).

2 Preliminaries

In the remainder, we assume some familiarity with term rewriting. Nevertheless, we recall required concepts and notations below. For further details on term rewriting, we refer to standard textbooks [3,14].

Throughout the paper $\mathcal{F}$ denotes a set of function symbols with associated arities, and $\mathcal{V}$ a countably infinite set of variables (so that fresh variables can always be picked) such that $\mathcal{F} \cap \mathcal{V} = \varnothing$. A *term* is either a variable $x \in \mathcal{V}$ or of the form $f(t_1, \dots, t_n)$, where n is the arity of $f \in \mathcal{F}$ and the arguments $t_1, \dots, t_n$ are terms. The set of all terms over $\mathcal{F}$ and $\mathcal{V}$ is denoted by $\mathcal{T}(\mathcal{F}, \mathcal{V})$. The set of variables occurring in a term t is denoted by $\mathsf{Var}(t)$. The *root symbol* of a term $t = f(t_1, \dots, t_n)$ is f and denoted by $\mathsf{root}(t)$. When we want to indicate that a term is not a variable, we sometimes write $f(\dots)$, where "..." denotes an arbitrary list of terms.

A *substitution* is a mapping $\sigma : \mathcal{V} \to \mathcal{T}(\mathcal{F}, \mathcal{V})$. Given a term t, $t\sigma$ denotes the term obtained by replacing every occurrence of variable x in t by $\sigma(x)$. The *domain* of a substitution σ is $\mathsf{Dom}(\sigma) := \{x \in \mathcal{V} \mid x\sigma \neq x\}$, and σ is *idempotent* if $\mathsf{Var}(x\sigma) \cap \mathsf{Dom}(\sigma) = \varnothing$ for every $x \in \mathcal{V}$. A *renaming* is a bijection $\alpha : \mathcal{V} \to \mathcal{V}$. Two terms s and t are *unifiable* if $s\sigma = t\sigma$ for some substitution σ, which is called a *unifier* of s and t.

A *context* is a term with exactly one occurrence of the special symbol $\square$. We write $C[t]$ for the term resulting from replacing $\square$ in context C by term t.

A *rewrite rule* is a pair of terms, written $l \to r$, such that the *variable conditions* $l \notin \mathcal{V}$ and $\mathsf{Var}(l) \supseteq \mathsf{Var}(r)$ hold. By a *variant* of a rewrite rule we mean a rule that is obtained by consistently renaming variables in the original rule to fresh ones. A *term rewrite system (TRS)* is a set $\mathcal{R}$ of rewrite rules. A function symbol $f \in \mathcal{F}$ is *defined* in $\mathcal{R}$ if $f(...) \to r \in \mathcal{R}$, and the set of defined symbols in $\mathcal{R}$ is $\mathcal{D}_\mathcal{R} := \{f \mid f(...) \to r \in \mathcal{R}\}$. We call $f \in \mathcal{F} \setminus \mathcal{D}_\mathcal{R}$ a *constructor*.

There is an $\mathcal{R}$*-rewrite step* from s to t, written $s \to_\mathcal{R} t$, iff there exist a context C, a substitution σ, and a rule $l \to r \in \mathcal{R}$ such that $s = C[l\sigma]$ and $t = C[r\sigma]$. We write $s \xrightarrow{\epsilon}_\mathcal{R} t$ if $C = \square$ (called a *root step*), and $s \xrightarrow{>\epsilon}_\mathcal{R} t$ (called a *non-root step*), otherwise. We say a term s_0 is $\mathcal{R}$*-terminating* if it starts no infinite rewrite sequence $s_0 \to_\mathcal{R} s_1 \to_\mathcal{R} s_2 \to_\mathcal{R} \cdots$, and say $\mathcal{R}$ is *terminating* if every term is $\mathcal{R}$-terminating.

For a relation $\rightarrowtail \subseteq A \times A$, we denote its transitive closure by $\rightarrowtail^+$ and reflexive transitive closure by $\rightarrowtail^*$. We say that $a_1, \ldots, a_n \in A$ are *joinable* (*meetable*) at $b \in A$ with respect to $\rightarrowtail$ if $a_i \rightarrowtail^* b$ ($b \rightarrowtail^* a_i$) for every $i \in \{1, \ldots, n\}$.

3 Reachability Constraint Satisfaction

In this section we introduce the syntax and semantics of reachability constraints, a framework that allows us to unify several concrete techniques for reachability analysis on an abstract level. Reachability constraints are first order formulas[3] with a single binary predicate symbol whose intended interpretation is reachability by rewriting with respect to a given rewrite system.

Definition 1 (Reachability Constraints). Reachability constraints *are given by the following grammar (where $s, t \in \mathcal{T}(\mathcal{F}, \mathcal{V})$ and $x \in \mathcal{V}$)*

$$\phi, \psi, \ldots ::= \top \mid \bot \mid s \twoheadrightarrow t \mid \phi \vee \psi \mid \phi \wedge \psi \mid \neg\phi \mid \forall x.\, \phi \mid \exists x.\, \phi$$

To save some space, we use conventional notation like $\bigwedge_{i \in I} \phi_i$ and $\exists x_1, \ldots, x_n.\, \phi$.

As mentioned above, the semantics of reachability constraints is defined with respect to a given rewrite system. In the following we define satisfiability of constraints with respect to a TRS. (This definition will be extended to conditional rewrite systems in Section 5.)

Definition 2 (Satisfiability). *We define*[4] *inductively when a substitution σ* satisfies *a reachability constraint ϕ modulo a TRS $\mathcal{R}$, written $\sigma \models_\mathcal{R} \phi$, as follows:*

- $\sigma \models_\mathcal{R} \top$*;*
- $\sigma \models_\mathcal{R} s \twoheadrightarrow t$ *if* $s\sigma \to_\mathcal{R}^* t\sigma$*;*

[3] While in general we allow an arbitrary first-order logical structure for formulas, for the purpose of this paper, negation and universal quantification are not required.

[4] It is also possible to give a model-theoretic account for these notions. However, the required preliminaries are outside the scope of this paper.

- *$\sigma \models_{\mathcal{R}} \phi \vee \psi$ if $\sigma \models_{\mathcal{R}} \phi$ or $\sigma \models_{\mathcal{R}} \psi$;*
- *$\sigma \models_{\mathcal{R}} \phi \wedge \psi$ if $\sigma \models_{\mathcal{R}} \phi$ and $\sigma \models_{\mathcal{R}} \psi$;*
- *$\sigma \models_{\mathcal{R}} \neg\phi$ if $\sigma \models_{\mathcal{R}} \phi$ does not hold;*
- *$\sigma \models_{\mathcal{R}} \forall x.\, \phi$ if $\sigma' \models_{\mathcal{R}} \phi$ for every σ' that coincides with σ on $\mathcal{V} \setminus \{x\}$.*
- *$\sigma \models_{\mathcal{R}} \exists x.\, \phi$ if $\sigma' \models_{\mathcal{R}} \phi$ for some σ' that coincides with σ on $\mathcal{V} \setminus \{x\}$.*

We say ϕ and ψ are equivalent *modulo $\mathcal{R}$, written $\phi \equiv_{\mathcal{R}} \psi$, when $\sigma \models_{\mathcal{R}} \phi$ iff $\sigma \models_{\mathcal{R}} \psi$ for all σ. We say ϕ and ψ are* (logically) equivalent*, written $\phi \equiv \psi$, if they are equivalent modulo any $\mathcal{R}$. We say ϕ is* satisfiable *modulo $\mathcal{R}$, written $\mathsf{SAT}_{\mathcal{R}}(\phi)$, if there is a substitution σ that satisfies ϕ modulo $\mathcal{R}$, and call σ a* solution *of ϕ with respect to $\mathcal{R}$.*

Checking for satisfiability of reachability constraints is for example useful for proving termination of term rewrite systems via the *dependency pair method* [2], or more specifically in *dependency graph* analysis. For the dependency pair method, we assume a fresh *marked* symbol $f^\sharp$ for every $f \in \mathcal{D}_{\mathcal{R}}$, and write $s^\sharp$ to denote the term $f^\sharp(s_1, \dots, s_n)$ for $s = f(s_1, \dots, s_n)$. The set of *dependency pairs* of a TRS $\mathcal{R}$ is $\mathsf{DP}(\mathcal{R}) := \left\{ l^\sharp \to r^\sharp \;\middle|\; l \to C[r] \in \mathcal{R},\ r \notin \mathcal{V},\ \mathsf{root}(r) \in \mathcal{D}_{\mathcal{R}} \right\}$. The standard definition of the dependency graph of a TRS [2] can be recast using reachability constraints as follows:

Definition 3 (Dependency Graph). *Given a TRS $\mathcal{R}$, its* dependency graph *$\mathsf{DG}(\mathcal{R})$ is the directed graph over $\mathsf{DP}(\mathcal{R})$ where there is an edge from $l^\sharp \to s^\sharp$ to $t^\sharp \to r^\sharp$ iff $\mathsf{SAT}_{\mathcal{R}}(s^\sharp \twoheadrightarrow t^\sharp\alpha)$, where α is a renaming of variables such that $\mathsf{Var}(t^\sharp\alpha) \cap \mathsf{Var}(s^\sharp) = \varnothing$.*

The nodes of the dependency graph correspond to the possible recursive calls in a program (represented by a TRS), while its edges encode the information which recursive calls can directly follow each other in arbitrary program executions. This is the reason why dependency graphs are useful for investigating the termination behavior of TRSs, as captured by the following result.

Theorem 1 ([10]). *A TRS $\mathcal{R}$ is terminating iff for every strongly connected component $\mathcal{C}$ of an over approximation of $\mathsf{DG}(\mathcal{R})$, there is no infinite chain $s_0 \xrightarrow{\epsilon}_{\mathcal{C}} t_0 \to_{\mathcal{R}}^* s_1 \xrightarrow{\epsilon}_{\mathcal{C}} t_1 \to_{\mathcal{R}}^* \cdots$ where every t_i is $\mathcal{R}$-terminating.*

Example 2. Consider the TRS $\mathcal{R}$ of Toyama [15] consisting of the single rule $\mathsf{f}(\mathsf{0}, \mathsf{1}, x) \to \mathsf{f}(x, x, x)$. Its dependency graph $\mathsf{DG}(\mathcal{R})$ consists of the single node:

$$\mathsf{f}^\sharp(\mathsf{0}, \mathsf{1}, x) \to \mathsf{f}^\sharp(x, x, x) \tag{1}$$

To show $\mathcal{R}$ terminates it suffices to show that $\mathsf{DG}(\mathcal{R})$ has no edge from (1) back to (1), that is, the unsatisfiability of the constraint (with a fresh variable x')

$$\mathsf{f}^\sharp(x, x, x) \twoheadrightarrow \mathsf{f}^\sharp(\mathsf{0}, \mathsf{1}, x') \tag{2}$$

The most popular method today for checking reachability during dependency graph analysis is unifiability between the target and an approximation of the topmost part of the source (its "cap") that does not change under rewriting, which is computed by the $\mathsf{tcap}_{\mathcal{R}}$ function [9].

Definition 4 (tcap). *Let $\mathcal{R}$ be a TRS. We recursively define $\mathsf{tcap}_{\mathcal{R}}(t)$ for a given term t as follows: $\mathsf{tcap}_{\mathcal{R}}(x)$ is a fresh variable if $x \in \mathcal{V}$; $\mathsf{tcap}_{\mathcal{R}}(f(t_1,\dots,t_n))$ is a fresh variable if $u = f(\mathsf{tcap}_{\mathcal{R}}(t_1),\dots,\mathsf{tcap}_{\mathcal{R}}(t_n))$ unifies with some left-hand side of the rules in $\mathcal{R}$; otherwise, it is u.*

The standard way of checking for nonreachability that is implemented in most tools is captured by of the following proposition.

Proposition 1. *If $\mathsf{tcap}_{\mathcal{R}}(s)$ and t are not unifiable, then $s \twoheadrightarrow t \equiv_{\mathcal{R}} \bot$.*

Example 3. Proposition 1 cannot prove the unsatisfiability of (2) of Example 2, since the term cap of the source $\mathsf{tcap}_{\mathcal{R}}(\mathsf{f}^\sharp(x,x,x)) = \mathsf{f}^\sharp(z,z',z'')$, where z, z', z'' are fresh variables, is unifiable with the target $\mathsf{f}^\sharp(\mathsf{0},\mathsf{1},x')$.

4 Reachability in Term Rewriting

In this section we introduce some techniques for analyzing (un)satisfiability of reachability constraints. The first one described below formulates an obvious observation: no root rewrite step is applicable when starting from a term whose root is a constructor.

Definition 5 (Non-Root Reachability). *For terms $s = f(\dots)$ and $t = g(\dots)$, we define the* non-root *reachability constraint $s \xrightarrow{>\epsilon}\!\!\!\!\!\twoheadrightarrow t$ as follows:*

- *$s \xrightarrow{>\epsilon}\!\!\!\!\!\twoheadrightarrow t = \bot$ if $f \neq g$, and*
- *$f(s_1,\dots,s_n) \xrightarrow{>\epsilon}\!\!\!\!\!\twoheadrightarrow f(t_1,\dots,t_n) = s_1 \twoheadrightarrow t_1 \wedge \cdots \wedge s_n \twoheadrightarrow t_n$.*

The intention of non-root reachability constraints is to encode zero or more steps of non-root rewriting, in the following sense.

Lemma 1. *For $s,t \notin \mathcal{V}$, $s\sigma \xrightarrow{>\epsilon}{}^{*}_{\mathcal{R}} t\sigma$ iff $\sigma \models_{\mathcal{R}} s \xrightarrow{>\epsilon}\!\!\!\!\!\twoheadrightarrow t$.*

Proof. The claim vacuously follows if $\mathsf{root}(s) \neq \mathsf{root}(t)$. So let $s = f(s_1,\dots,s_n)$ and $t = f(t_1,\dots,t_n)$. We have $f(s_1,\dots,s_n)\sigma \xrightarrow{>\epsilon}{}^{*}_{\mathcal{R}} f(t_1,\dots,t_n)\sigma$ iff $s_1\sigma \to^{*}_{\mathcal{R}} t_1\sigma$, $\dots$, $s_n\sigma \to^{*}_{\mathcal{R}} t_n\sigma$ iff $\sigma \models_{\mathcal{R}} s_1 \twoheadrightarrow t_1 \wedge \cdots \wedge s_n \twoheadrightarrow t_n$. ☐

Combined with the observation that no root step is applicable to a term whose root symbol is a constructor, we obtain the following reformulation of a folklore result that reduces reachability to direct subterms.

Proposition 2. *If $s = f(\dots)$ with $f \notin \mathcal{D}_{\mathcal{R}}$ and $t \notin \mathcal{V}$, then $s \twoheadrightarrow t \equiv_{\mathcal{R}} s \xrightarrow{>\epsilon}\!\!\!\!\!\twoheadrightarrow t$.*

Proposition 2 is directly applicable in the analysis of dependency graphs.

Example 4. Consider again the constraint $\mathsf{f}^\sharp(x,x,x) \twoheadrightarrow \mathsf{f}^\sharp(\mathsf{0},\mathsf{1},x')$ from Example 2. Since $\mathsf{f}^\sharp$ is not defined in $\mathcal{R}$, Proposition 2 reduces this constraint to $\mathsf{f}^\sharp(x,x,x) \xrightarrow{>\epsilon}\!\!\!\!\!\twoheadrightarrow \mathsf{f}^\sharp(\mathsf{0},\mathsf{1},x')$, that is,

$$x \twoheadrightarrow \mathsf{0} \wedge x \twoheadrightarrow \mathsf{1} \wedge x \twoheadrightarrow x' \tag{3}$$

$$\begin{tikzcd}
& \mathsf{h} \arrow[loop above] \arrow[dl] \arrow[d] \arrow[dr] & \\
\mathsf{f} \arrow[r] & \mathsf{g} \arrow[r] & \mathsf{c}
\end{tikzcd}$$

(a) Example 6

$$\begin{tikzcd}
\mathsf{f} \arrow[loop left] & \mathsf{0} & \mathsf{1}
\end{tikzcd}$$

(b) Example 7

$$\begin{tikzcd}
& \mathsf{g} \arrow[loop above] \arrow[dl] \arrow[d] \arrow[dr] & \\
\mathsf{f} \arrow[loop left] & \mathsf{0} & \mathsf{1}
\end{tikzcd}$$

(c) Example 8

Fig. 1: Example symbol transition graphs.

4.1 Symbol Transition Graphs

Here we introduce a new, simple and efficient way of overapproximating reachability by tracking the relation of root symbols of terms according to a given set of rewrite rules. We first illustrate the intuition by an example.

Example 5. Consider a TRS $\mathcal{R}$ consisting of rules of the following form:

$$\mathsf{f}(...) \to \mathsf{g}(...) \qquad \mathsf{g}(...) \to \mathsf{c}(...) \qquad \mathsf{h}(...) \to x$$

Moreover, suppose $s \to_{\mathcal{R}}^* t$. Then we can make the following observations:

– If $\mathsf{root}(s) = \mathsf{c}$, then $\mathsf{root}(t) = \mathsf{c}$ since non-root steps preserve the root symbol and no root steps are applicable to terms of the form $\mathsf{c}(...)$.
– If $\mathsf{root}(s) = \mathsf{g}$, then $\mathsf{root}(t) \in \{\mathsf{g}, \mathsf{c}\}$ since non-root steps preserve the root symbol and the only possible root step is $\mathsf{g}(...) \to \mathsf{c}(...)$.
– If $\mathsf{root}(s) = \mathsf{f}$, then $\mathsf{root}(t) \in \{\mathsf{f}, \mathsf{g}, \mathsf{c}\}$ by the same reasoning.
– If $\mathsf{root}(s) = \mathsf{h}$, then t can be any term and $\mathsf{root}(t)$ can be arbitrary.

This informal argument is captured by the following definition.

Definition 6 (Symbol Transition Graphs). *The* symbol transition graph $\mathsf{SG}(\mathcal{R})$ *of a TRS* $\mathcal{R}$ *over signature* $\mathcal{F}$ *is the graph* $\langle \mathcal{F}, \rightarrowtail_{\mathcal{R}} \rangle$, *where* $f \rightarrowtail_{\mathcal{R}} g$ *iff* $\mathcal{R}$ *contains a rule of form* $f(...) \to g(...)$ *or* $f(...) \to x$ *with* $x \in \mathcal{V}$.

The following result tells us that for non-variable terms the symbol transition graph captures the relation between the root symbols of root rewrite steps.

Lemma 2. *If* $s \xrightarrow{\epsilon}_{\mathcal{R}} t$ *then* $t \in \mathcal{V}$ *or* $\mathsf{root}(s) \rightarrowtail_{\mathcal{R}} \mathsf{root}(t)$.

Proof. By assumption there exist $l \to r \in \mathcal{R}$ and σ such that $s = l\sigma$ and $r\sigma = t$. If $r \in \mathcal{V}$ then either $t \in \mathcal{V}$ or $\mathsf{root}(s) = \mathsf{root}(l) \rightarrowtail_{\mathcal{R}} \mathsf{root}(t)$. Otherwise, $\mathsf{root}(s) = \mathsf{root}(l) \rightarrowtail_{\mathcal{R}} \mathsf{root}(r) = \mathsf{root}(t)$. □

Since every rewrite sequence is composed of subsequences that take place entirely below the root (and hence do not change the root symbol) separated by root steps, we can extend the previous result to rewrite sequences.

Lemma 3. *If* $s = f(...) \to_{\mathcal{R}}^* g(...) = t$ *then* $f \rightarrowtail_{\mathcal{R}}^* g$.

Proof. We prove the claim for arbitrary s and f by induction on the derivation length of $s \to_{\mathcal{R}}^* t$. The base case is trivial, so consider $s \to_{\mathcal{R}} s' \to_{\mathcal{R}}^n t\sigma$. Since $t \notin \mathcal{V}$, we have $f' \in \mathcal{F}$ with $s' = f'(...)$. Thus the induction hypothesis yields $f' \rightarrowtail_{\mathcal{R}}^* g$. If $s \xrightarrow{\epsilon}_{\mathcal{R}} s'$ then by Lemma 2 we conclude $f \rightarrowtail_{\mathcal{R}} f' \rightarrowtail_{\mathcal{R}}^* g$, and otherwise $f = f' \rightarrowtail_{\mathcal{R}}^* g$. ◻

It is now straightforward to derive the following from Lemma 3.

Corollary 1. *If $f \rightarrowtail_{\mathcal{R}}^* g$ does not hold, then $f(...) \twoheadrightarrow g(...) \equiv_{\mathcal{R}} \bot$.*

Example 6. The symbol transition graph for Example 5 is depicted in Fig. 1(a). By Corollary 1 we can conclude, for instance, $\mathsf{g}(...) \twoheadrightarrow \mathsf{f}(...)$ is unsatisfiable.

Corollary 1 is useful for checking (un)satisfiability of $s \twoheadrightarrow t$, only if neither s nor t is a variable. However, the symbol transition graph is also useful for unsatisfiability in the case when s and t may be variables.

Proposition 3. *If $\mathsf{SAT}_{\mathcal{R}}(x \twoheadrightarrow t_1 \wedge \cdots \wedge x \twoheadrightarrow t_n)$ for $t_1 = g_1(...)$, ..., $t_n = g_n(...)$, then g_1, ..., g_n are meetable with respect to $\rightarrowtail_{\mathcal{R}}$.*

Proof. By assumption there is a substitution σ such that $x\sigma \to_{\mathcal{R}}^* t_1\sigma$, ..., $x\sigma \to_{\mathcal{R}}^* t_n\sigma$. Clearly $x\sigma \in \mathcal{V}$ is not possible. Thus, suppose $x\sigma = f(...)$ for some f. Finally, from Lemma 3, we have $f \rightarrowtail_{\mathcal{R}}^* g_1$, ..., $f \rightarrowtail_{\mathcal{R}}^* g_n$ and thereby conclude that g_1, ..., g_n are meetable at f. ◻

The dual of Proposition 3 is proved in a similar way, but with some special care to ensure $x\sigma \in \mathcal{V}$.

Proposition 4. *If $\mathsf{SAT}_{\mathcal{R}}(s_1 \twoheadrightarrow x \wedge \cdots \wedge s_n \twoheadrightarrow x)$ for $s_1 = f_1(...)$, ..., $s_n = f_n(...)$, then f_1, ..., f_n are joinable with respect to $\rightarrowtail_{\mathcal{R}}$.*

Example 7 (Continuation of Example 4). Due to Proposition 3, proving (3) unsatisfiable reduces to proving that $\mathsf{0}$ and $\mathsf{1}$ are not meetable with respect to $\rightarrowtail_{\mathcal{R}}$. This is obvious from the symbol transition graph depicted in Fig. 1(b). Hence, we conclude the termination of $\mathcal{R}$.

Example 8. Consider the following extension of $\mathcal{R}$ from Example 2.

$$\mathsf{f}(\mathsf{0}, \mathsf{1}, x) \to \mathsf{f}(x, x, x) \qquad \mathsf{g}(x, y) \to x \qquad \mathsf{g}(x, y) \to y$$

The resulting system is not terminating [15]. The corresponding symbol transition graph is depicted in Fig. 1(c), where $\mathsf{0}$ and $\mathsf{1}$ are meetable, as expected.

4.2 Look-Ahead Reachability

Here we propose another method for overapproximating reachability, which eventually subsumes the tcap-unifiability method when target terms are linear. Note that this condition is satisfied in the dependency graph approximation of left-linear TRSs. Our method is based on the observation that any rewrite sequence either contains at least one root step, or takes place entirely below the root. This observation can be captured using our reachability constraints.

Definition 7 (Root Narrowing Constraints). *Let $l \to r$ be a rewrite rule with $\mathsf{Var}(l) = \{x_1, \dots, x_n\}$. Then for terms s and t not containing $x_1, \dots, x_n$, the* root narrowing constraint *from s to t via $l \to r$ is defined by*

$$s \rightsquigarrow_{l \to r} t := \exists x_1, \dots, x_n.\, s \xrightarrow{>\epsilon}\!\!\!\!\twoheadrightarrow l \wedge r \twoheadrightarrow t$$

We write $s \rightsquigarrow_{\mathcal{R}} t$ for $\bigvee_{l \to r \in \mathcal{R}'} s \rightsquigarrow_{l \to r} t$, where $\mathcal{R}'$ is a variant of $\mathcal{R}$ in which variables occurring in s or t are renamed to fresh ones.

In the definition above, the intuition is that if there are any root steps inside a rewrite sequence then we can pick the first one, which is only preceded by non-root steps. The following theorem justifies this intuition.

Theorem 2. *If $s, t \notin \mathcal{V}$, then $s \twoheadrightarrow t \equiv_{\mathcal{R}} s \xrightarrow{>\epsilon}\!\!\!\!\twoheadrightarrow t \vee s \rightsquigarrow_{\mathcal{R}} t$.*

Proof. Let $s = f(s_1, \dots, s_n)$ and σ be a substitution. We show $\sigma \models_{\mathcal{R}} s \twoheadrightarrow t$ iff $\sigma \models_{\mathcal{R}} s \xrightarrow{>\epsilon}\!\!\!\!\twoheadrightarrow t \vee s \rightsquigarrow_{\mathcal{R}} t$. For the "if" direction suppose the latter. If $\sigma \models_{\mathcal{R}} s \xrightarrow{>\epsilon}\!\!\!\!\twoheadrightarrow t$, then t is of form $f(t_1, \dots, t_n)$ and $s_i\sigma \to_{\mathcal{R}}^* t_i\sigma$ for every $i \in \{1, \dots, n\}$, and thus $s\sigma \to_{\mathcal{R}}^* t\sigma$. If $\sigma \models_{\mathcal{R}} s \rightsquigarrow_{\mathcal{R}} t$, then we have a renamed variant $l \to r$ of a rule in $\mathcal{R}$ such that $\sigma \models_{\mathcal{R}} s \rightsquigarrow_{l \to r} t$. This indicates that there exists a substitution σ' that coincides with σ on $\mathcal{V} \setminus \mathsf{Var}(l)$, and satisfies

– $\sigma' \models_{\mathcal{R}} s \xrightarrow{>\epsilon}\!\!\!\!\twoheadrightarrow l$, that is, $l = f(l_1, \dots, l_n)$ and $s_i\sigma' \to_{\mathcal{R}}^* l_i\sigma'$;
– $\sigma' \models_{\mathcal{R}} r \twoheadrightarrow t$, that is, $r\sigma' \to_{\mathcal{R}}^* t\sigma'$.

In combination, we have $s\sigma = s\sigma' \xrightarrow{>\epsilon}{}_{\mathcal{R}}^{*} l\sigma' \xrightarrow{\epsilon}_{\mathcal{R}} r\sigma' \to_{\mathcal{R}}^* t\sigma' = t\sigma$.

Now consider the "only if" direction. Suppose that σ is an idempotent substitution such that $s\sigma \to_{\mathcal{R}}^* t\sigma$. We may assume idempotence, since from any solution σ' of $s \twoheadrightarrow t$, we obtain idempotent solution σ by renaming variables in $\mathsf{Var}(s) \cup \mathsf{Var}(t)$ to fresh ones. We proceed by the following case analysis:

– *No root step is involved:* $s\sigma \xrightarrow{>\epsilon}{}_{\mathcal{R}}^{*} t\sigma$. Then Lemma 1 implies $\sigma \models_{\mathcal{R}} s \xrightarrow{>\epsilon}\!\!\!\!\twoheadrightarrow t$.
– *At least one root step is involved:* there is a rule $l \to r \in \mathcal{R}$ and a substitution θ such that $s\sigma \xrightarrow{>\epsilon}{}_{\mathcal{R}}^{*} l\theta$ and $r\theta \to_{\mathcal{R}}^* t\sigma$. Since variables in $l\theta$ must occur in $s\sigma$ (due to our assumptions on rewrite rules), we have $l\theta = l\theta\sigma$ since σ is idempotent. Thus from Lemma 1 we have $\sigma \models_{\mathcal{R}} s \xrightarrow{>\epsilon}\!\!\!\!\twoheadrightarrow l\theta$. Further, variables in $r\theta$ must occur in $l\theta$ and thus in $s\theta$, we also have $r\theta\sigma = r\theta \to_{\mathcal{R}}^* t\sigma$, and hence $\sigma \models_{\mathcal{R}} r\theta \twoheadrightarrow t$. This concludes $\sigma \models_{\mathcal{R}} s \rightsquigarrow_{l \to r} t$. □

Proposition 2 is a corollary of Theorem 2 together with the following easy lemma, stating that if the root symbol of the source term is not a defined symbol, then no root step can occur.

Lemma 4. *If $c \notin \mathcal{D}_{\mathcal{R}}$ then $c(\dots) \rightsquigarrow_{\mathcal{R}} t \equiv \bot$.*

Example 9. Consider the TRS $\mathcal{R}$ consisting of the following rules:

$$\mathsf{0} > x \to \mathsf{false} \qquad \mathsf{s}(x) > \mathsf{0} \to \mathsf{true} \qquad \mathsf{s}(x) > \mathsf{s}(y) \to x > y$$

Applying Theorem 2 once reduces the reachability constraint $\mathsf{0} > z \twoheadrightarrow \mathsf{true}$ to the disjunction of

1. $\mathsf{0} > z \xrightarrow{>\epsilon}\!\!\!\!\twoheadrightarrow \mathsf{true}$,
2. $\exists x.\ \mathsf{0} > z \xrightarrow{>\epsilon}\!\!\!\!\twoheadrightarrow \mathsf{0} > x\ \wedge\ \mathsf{false} \twoheadrightarrow \mathsf{true}$
3. $\exists x.\ \mathsf{0} > z \xrightarrow{>\epsilon}\!\!\!\!\twoheadrightarrow \mathsf{s}(x) > \mathsf{0}\ \wedge\ \mathsf{true} \twoheadrightarrow \mathsf{true}$
4. $\exists x, y.\ \mathsf{0} > z \xrightarrow{>\epsilon}\!\!\!\!\twoheadrightarrow \mathsf{s}(x) > \mathsf{s}(y)\ \wedge\ x > y \twoheadrightarrow \mathsf{true}$

Disjuncts 1, 3, and 4 expand to $\bot$ by definition of $\xrightarrow{>\epsilon}\!\!\!\!\twoheadrightarrow$. For disjunct 2, applying Theorem 2 or Proposition 2 to $\mathsf{false} \twoheadrightarrow \mathsf{true}$ yields $\bot$.

Note that Theorem 2 can be applied arbitrarily often. Thus, to avoid nontermination in an implementation, we need to control how often it is applied. For this purpose we introduce the following definition.

Definition 8 (k-Fold Look-Ahead). *We define the* k-fold look-ahead transformation *with respect to a TRS $\mathcal{R}$ as follows:*

$$\mathsf{L}^k_{\mathcal{R}}(s \twoheadrightarrow t) := \begin{cases} \mathsf{L}^k_{\mathcal{R}}(s \xrightarrow{>\epsilon}\!\!\!\!\twoheadrightarrow t) \vee s \rightsquigarrow^k_{\mathcal{R}} t & \text{if } k \geq 1 \text{ and } s, t \notin \mathcal{V} \\ s \twoheadrightarrow t & \text{otherwise} \end{cases}$$

which is homomorphically extended to reachability constraints. Here, $\rightsquigarrow^k_{\mathcal{R}}$ is defined as in Definition 7, but k controls the number of root steps to be expanded:

$$s \rightsquigarrow^k_{l \to r} t := \exists x_1, \ldots, x_n.\ \mathsf{L}^k_{\mathcal{R}}(s \xrightarrow{>\epsilon}\!\!\!\!\twoheadrightarrow l) \wedge \mathsf{L}^{k-1}_{\mathcal{R}}(r \twoheadrightarrow t)$$

It easily follows from Theorem 2 and induction on k that the k-fold look-ahead preserves the semantics of reachability constraints.

Corollary 2. $\mathsf{L}^k_{\mathcal{R}}(\phi) \equiv_{\mathcal{R}} \phi$.

The following results indicate that, whenever $\mathsf{tcap}_{\mathcal{R}}$-unifiability (Proposition 1) proves $s \twoheadrightarrow t$ unsatisfiable for linear t, $\mathsf{L}^1_{\mathcal{R}}$ can also conclude it.

Lemma 5. *Let $s = f(s_1, \ldots, s_n)$ and $t \notin \mathcal{V}$ be a linear term, and suppose that $f(\mathsf{tcap}_{\mathcal{R}}(s_1), \ldots, \mathsf{tcap}_{\mathcal{R}}(s_n))$ does not unify with t or any left-hand side in $\mathcal{R}$. Then $\mathsf{L}^1_{\mathcal{R}}(s \twoheadrightarrow t) \equiv \bot$.*

Proof. By structural induction on s. First, we show $\mathsf{L}^1_{\mathcal{R}}(s \xrightarrow{>\epsilon}\!\!\!\!\twoheadrightarrow t) \equiv \bot$. This is trivial if $\mathsf{root}(t) \neq f$. So let $t = f(t_1, \ldots, t_n)$. By assumption there is an $i \in \{1, \ldots, n\}$ such that $\mathsf{tcap}_{\mathcal{R}}(s_i)$ does not unify with t_i. Hence $\mathsf{tcap}_{\mathcal{R}}(s_i)$ cannot be a fresh variable, and thus s_i is of the form $g(u_1, \ldots, u_m)$ and $\mathsf{tcap}_{\mathcal{R}}(s_i) = g(\mathsf{tcap}_{\mathcal{R}}(u_1), \ldots, \mathsf{tcap}_{\mathcal{R}}(u_m))$ is not unifiable with any left-hand side in $\mathcal{R}$. Therefore, the induction hypothesis applies to s_i, yielding $\mathsf{L}^1_{\mathcal{R}}(s_i \twoheadrightarrow t_i) \equiv \bot$. This concludes $\mathsf{L}^1_{\mathcal{R}}(s \xrightarrow{>\epsilon}\!\!\!\!\twoheadrightarrow t) = \mathsf{L}^1_{\mathcal{R}}(s_1 \twoheadrightarrow t_1) \wedge \cdots \wedge \mathsf{L}^1_{\mathcal{R}}(s_n \twoheadrightarrow t_n) \equiv \bot$.

Second, we show $\mathsf{L}^1_{\mathcal{R}}(s \rightsquigarrow^1_{\mathcal{R}} t) \equiv \bot$. To this end, we show for an arbitrary variant $l \to r$ of a rule in $\mathcal{R}$ that $\mathsf{L}^1_{\mathcal{R}}(s \xrightarrow{>\epsilon}\!\!\!\!\twoheadrightarrow l) \equiv \bot$. This is clear if $\mathsf{root}(l) \neq f$. So let $l = f(l_1, \ldots, l_n)$. By assumption there is an $i \in \{1, \ldots, n\}$ such that $\mathsf{tcap}_{\mathcal{R}}(s_i)$ and l_i are not unifiable. By a similar reasoning as above the induction hypothesis applies to s_i and yields $\mathsf{L}^1_{\mathcal{R}}(s_i \twoheadrightarrow l_i) \equiv \bot$. This concludes $\mathsf{L}^1_{\mathcal{R}}(s \xrightarrow{>\epsilon}\!\!\!\!\twoheadrightarrow l) \equiv \bot$. □

Corollary 3. *If $\mathsf{tcap}_{\mathcal{R}}(s)$ and t are not unifiable, then $\mathsf{L}^1_{\mathcal{R}}(s \twoheadrightarrow t) \equiv \bot$.*

5 Conditional Rewriting

Conditional rewriting is a flavor of rewriting where rules are guarded by conditions. On the one hand, this gives us a boost in expressiveness in the sense that it is often possible to directly express equations with preconditions and that it is easier to directly express programming constructs like the where-clauses of Haskell. On the other hand, the analysis of conditional rewrite systems is typically more involved than for plain rewriting.

In this section we first recall the basics of conditional term rewriting. Then, we motivate the importance of reachability analysis for the conditional case. Finally, we extend the techniques of Section 4 to conditional rewrite systems.

Preliminaries. A *conditional rewrite rule* $l \to r \Leftarrow \phi$ consists of two terms $l \notin \mathcal{V}$ and r (the left-hand side and right-hand side, respectively) and a list ϕ of pairs of terms (its *conditions*). A *conditional term rewrite system* (CTRS for short) is a set of conditional rewrite rules. Depending on the interpretation of conditions, conditional rewriting can be separated into several classes. For the purposes of this paper we are interested in *oriented* CTRSs, where conditions are interpreted as reachability constraints with respect to conditional rewriting. Hence, from now on we identify conditions $\langle s_1, t_1\rangle, \ldots, \langle s_n, t_n\rangle$ with the reachability constraint $s_1 \twoheadrightarrow t_1 \wedge \cdots \wedge s_n \twoheadrightarrow t_n$, and the empty list with $\top$ (omitting "$\Leftarrow \top$" from rules).

The rewrite relation of a CTRS is layered into *levels*: given a CTRS $\mathcal{R}$ and level $i \in \mathbb{N}$, the corresponding (unconditional) TRS $\mathcal{R}_i$ is defined recursively:

$$\begin{aligned}\mathcal{R}_0 &:= \varnothing\\ \mathcal{R}_{i+1} &:= \{l\sigma \to r\sigma \mid l \to r \Leftarrow \phi \in \mathcal{R},\ \sigma \models_{\mathcal{R}_i} \phi\}\end{aligned}$$

Then the *(conditional) rewrite relation at level* i, written $\to_{\mathcal{R},i}$ (or $\to_i$ whenever $\mathcal{R}$ is clear from the context), is the plain rewrite relation $\to_{\mathcal{R}_i}$ induced by the TRS $\mathcal{R}_i$. Finally, the *induced (conditional) rewrite relation* of a CTRS $\mathcal{R}$ is defined by $\to_{\mathcal{R}} := \bigcup\{\to_i \mid i \geq 0\}$. At this point Definition 2 is extended to the conditional case in a straightforward manner.

Definition 9 (Level Satisfiability). *Let $\mathcal{R}$ be a CTRS and ϕ a reachability constraint. We say that a substitution σ* satisfies ϕ modulo $\mathcal{R}$ at level i*, whenever $\sigma \models_{\mathcal{R},i} \phi$. If we are not interested in a specific satisfying substitution we say that ϕ* is satisfiable modulo $\mathcal{R}$ at level i *and write $\mathsf{SAT}_{\mathcal{R},i}(\phi)$ (or just $\mathsf{SAT}_i(\phi)$ whenever $\mathcal{R}$ is clear from the context).*

5.1 Infeasibility

The main area of interest for reachability analysis in the conditional case is checking for *infeasibility*. While a formal definition of this concept follows below, for the moment, think of it as unsatisfiability of conditions. The two predominant applications of infeasibility are: (1) if the conditions of a rule are unsatisfiable, the rule can never be applied and thus safely be removed without changing the induced rewrite relation; (2) if the conditions of a conditional critical pair (which arises from confluence analysis of CTRSs) are unsatisfiable, then it poses no problem to confluence and can safely be ignored.

Definition 10 (Infeasibility). *We say that a conditional rewrite rule $l \to r \Leftarrow \phi$ is* applicable at level i with respect to a CTRS $\mathcal{R}$ *iff* $\mathsf{SAT}_{\mathcal{R},i-1}(\phi)$. *A set $\mathcal{S}$ of rules is* infeasible with respect to $\mathcal{R}$ *when no rule in $\mathcal{S}$ is applicable at any level.*

The next theorem allows us to remove some rules from a CTRS while checking for infeasibility of rules.

Theorem 3. *A set $\mathcal{S}$ of rules is infeasible with respect to a CTRS $\mathcal{R}$ iff it is infeasible with respect to $\mathcal{R} \setminus \mathcal{S}$.*

Proof. The 'only if' direction is trivial. Thus we concentrate on the 'if' direction. To this end, assume that $\mathcal{S}$ is infeasible with respect to $\mathcal{R} \setminus \mathcal{S}$, but not infeasible with respect to $\mathcal{R}$. That is, at least one rule in $\mathcal{S}$ is applicable at some level with respect to $\mathcal{R}$. Let m be the minimum level such that there is a rule $l \to r \Leftarrow \phi \in \mathcal{S}$ that is applicable at level m with respect to $\mathcal{R}$. Now if $m = 0$ then $l \to r \Leftarrow \phi$ is applicable at level 0 and thus $\mathsf{SAT}_{\mathcal{R},0}(\phi)$, which trivially implies $\mathsf{SAT}_{\mathcal{R}\setminus\mathcal{S},0}(\phi)$, contradicting the assumption that all rules in $\mathcal{S}$ are infeasible with respect to $\mathcal{R} \setminus \mathcal{S}$. Otherwise, $m = k + 1$ for some $k \geq 0$ and since $l \to r \Leftarrow \phi$ is applicable at level m we have $\mathsf{SAT}_{\mathcal{R},k}(\phi)$. Moreover, the rewrite relations $\to_{\mathcal{R},k}$ and $\to_{\mathcal{R}\setminus\mathcal{S},k}$ coincide (since all rules in $\mathcal{S}$ are infeasible at levels smaller than m by our choice of m). Thus we also have $\mathsf{SAT}_{\mathcal{R}\setminus\mathcal{S},k}(\phi)$, again contradicting the assumption that all rules in $\mathcal{S}$ are infeasible with respect to $\mathcal{R} \setminus \mathcal{S}$. □

The following example from *the confluence problems data base* (Cops)[5] shows that Theorem 3 is beneficial for showing infeasibility of conditional rewrite rules.

Example 10 (Cops 794). Consider the CTRS $\mathcal{R}$ consisting of the two rules:

$$\mathsf{a} \to \mathsf{c} \Leftarrow \mathsf{f}(\mathsf{a}) \twoheadrightarrow \mathsf{f}(\mathsf{b}) \qquad\qquad \mathsf{f}(\mathsf{b}) \to \mathsf{b}$$

The tcap-method does not manage to conclude infeasibility of the first rule, since $\mathsf{tcap}_{\mathcal{R}}(\mathsf{f}(\mathsf{a})) = x$ for some fresh variable x and thus unifies with $\mathsf{f}(\mathsf{b})$. The reason for this result was that for computing $\mathsf{tcap}_{\mathcal{R}}$ we had to recursively (in a bottom-up fashion) try to unify arguments of functions with left-hand sides of rules, which succeeded for the left-hand side of the first rule and the argument a of $\mathsf{f}(\mathsf{a})$, thereby obtaining $\mathsf{f}(x)$ which, in turn, unifies with the left-hand side of the second rule. But by Theorem 3 we do not need to consider the first rule for computing the term cap and thus obtain $\mathsf{tcap}_{\{\mathsf{f}(\mathsf{b})\to\mathsf{b}\}}(\mathsf{f}(\mathsf{a})) = \mathsf{f}(\mathsf{a})$ which does not unify with $\mathsf{f}(\mathsf{b})$ and thereby shows that the first rule is infeasible.

[5] `http://cops.uibk.ac.at/?q=ctrs+oriented`

$$\begin{tikzcd} & \mathsf{a} \arrow[dl] \arrow[dr] & \\ \mathsf{b} & & \mathsf{c} \end{tikzcd}$$

(a)

$$\begin{tikzcd} & \mathsf{a} \arrow[dl] \arrow[dr] & \\ \mathsf{b} \arrow[rr] & & \mathsf{c} \end{tikzcd}$$

(b)

Fig. 2: Inductive and plain symbol transition graph of Example 11.

5.2 Symbol Transition Graphs in the Presence of Conditions

In the presence of conditions in rules we replace Definition 6 by the following inductive definition:

Definition 11 (Inductive Symbol Transition Graphs). *The* symbol transition graph $\mathsf{SG}(\mathcal{R})$ *of a CTRS* $\mathcal{R}$ *over a signature* $\mathcal{F}$ *is the graph* $\langle \mathcal{F}, \rightarrowtail_{\mathcal{R}} \rangle$ *where* $\rightarrowtail_{\mathcal{R}}$ *is defined inductively by the following two inference rules:*

$$\frac{f(...) \to x \Leftarrow \phi \in \mathcal{R} \quad \forall \langle s,t \rangle \in \phi.\, s \in \mathcal{V} \lor t \in \mathcal{V} \lor \mathsf{root}(s) \rightarrowtail_{\mathcal{R}}^{*} \mathsf{root}(t)}{f \rightarrowtail_{\mathcal{R}} g}\; g \in \mathcal{F}$$

$$\frac{f(...) \to g(...) \Leftarrow \phi \in \mathcal{R} \quad \forall \langle s,t \rangle \in \phi.\, s \in \mathcal{V} \lor t \in \mathcal{V} \lor \mathsf{root}(s) \rightarrowtail_{\mathcal{R}}^{*} \mathsf{root}(t)}{f \rightarrowtail_{\mathcal{R}} g}$$

The example below shows the difference between the symbol transition graph for TRSs (which can be applied as a crude overapproximation also to CTRSs by dropping all conditions) and the inductive symbol transition graph for CTRSs.

Example 11 (Cops 293). Consider the CTRS consisting of the three rules:

$$\mathsf{a} \to \mathsf{b} \qquad\qquad \mathsf{a} \to \mathsf{c} \qquad\qquad \mathsf{b} \to \mathsf{c} \Leftarrow \mathsf{b} \twoheadrightarrow \mathsf{c}$$

The corresponding inductive symbol transition graph is depicted in Fig. 2(a) and implies unsatisfiability of $\mathsf{b} \twoheadrightarrow \mathsf{c}$. Note that this conclusion cannot be drawn from the plain symbol transition graph of the TRS obtained by dropping the condition of the third rule, shown in Fig. 2(b).

The inductive symbol transition graph gives us a sufficient criterion for concluding nonreachability with respect to a given CTRS, as shown in the following.

Lemma 6. *If* $f(...) \to_{\mathcal{R}}^{*} g(...)$ *then* $f \rightarrowtail_{\mathcal{R}}^{*} g$.

Proof. Let $s = f(...)$ and $u = g(...)$ and assume that s rewrites to u at level i, that is, $s \to_{i}^{*} u$. We prove the statement by induction on the level i. If $i = 0$ then we are done, since $\to_0$ is empty and therefore $f(...) = s = u = g(...)$, which trivially implies $f \rightarrowtail_{\mathcal{R}}^{*} g$. Otherwise, $i = j + 1$ and we obtain the induction hypothesis (IH) that $s \to_{j}^{*} t$ implies $\mathsf{root}(s) \rightarrowtail_{\mathcal{R}}^{*} \mathsf{root}(t)$ for arbitrary non-variable terms s and t. We proceed to show that $s \to_{i}^{*} u$ implies $f \rightarrowtail_{\mathcal{R}}^{*} g$ by an inner induction on the length of this derivation. If the derivation is empty, then $f(...) = s =$

$u = g(\ldots)$ and therefore trivially $f \rightarrowtail^*_{\mathcal{R}} g$. Otherwise, the derivation is of the shape $s \to^*_i t \to_i u$ for some non-variable term $t = h(\ldots)$ and we obtain the inner induction hypothesis that $f \rightarrowtail^*_{\mathcal{R}} h$. It remains to show $h \rightarrowtail^*_{\mathcal{R}} g$ in order to conclude the proof. To this end, consider the step $t = C[l\sigma] \to_i C[r\sigma] = u$ for some context C, substitution σ, and rule $l \to r \Leftarrow \phi \in \mathcal{R}$ such that $\sigma \models_j \phi$. Now, by IH, we obtain that $s' \in \mathcal{V}$ or $t' \in \mathcal{V}$ or $\mathsf{root}(s') \rightarrowtail^*_{\mathcal{R}} \mathsf{root}(t')$ for all $\langle s', t' \rangle \in \phi$. Thus, by Definition 11, we obtain that $\mathsf{root}(l\sigma) \rightarrowtail_{\mathcal{R}} \mathsf{root}(r\sigma)$. We conclude by a case analysis on the structure of the context C. If C is empty, that is $C = \square$, then $h = \mathsf{root}(l\sigma) \rightarrowtail^*_{\mathcal{R}} \mathsf{root}(r\sigma) = g$ and we are done. Otherwise, $h = \mathsf{root}(t) = \mathsf{root}(u) = g$ and therefore trivially $h \rightarrowtail^*_{\mathcal{R}} g$. □

Corollary 4. *If $f \rightarrowtail^*_{\mathcal{R}} g$ does not hold, then $f(\ldots) \twoheadrightarrow g(\ldots) \equiv_{\mathcal{R}} \bot$.*

5.3 Look-Ahead Reachability in the Presence of Conditions

In the following definition we extend our look-ahead technique from plain rewriting to conditional rewriting.

Definition 12 (Conditional Root Narrowing Constraints). *Let $l \to r \Leftarrow \phi$ be a conditional rewrite rule with $\mathsf{Var}(l) = \{x_1, \ldots, x_n\}$. Then for terms s and t not containing $x_1, \ldots, x_n$, the* conditional root narrowing constraint *from s to t via $l \to r \Leftarrow \phi$ is defined by*

$$s \rightsquigarrow_{l \to r \Leftarrow \phi} t := \exists x_1, \ldots, x_n.\, s \xrightarrow{>\epsilon}\!\!\!\!\twoheadrightarrow l \wedge r \twoheadrightarrow t \wedge \phi$$

We write $s \rightsquigarrow_{\mathcal{R}} t$ for $\bigvee_{l \to r \Leftarrow \phi \in \mathcal{R}'} s \rightsquigarrow_{l \to r \Leftarrow \phi} t$, where $\mathcal{R}'$ is a variant of $\mathcal{R}$ in which variables occurring in s or t are renamed to fresh ones.

And we obtain a result similar to Theorem 2.

Lemma 7. *If $s, t \notin \mathcal{V}$, then $s \twoheadrightarrow t \equiv_{\mathcal{R}} s \xrightarrow{>\epsilon}\!\!\!\!\twoheadrightarrow t \vee s \rightsquigarrow_{\mathcal{R}} t$.*

Example 12 (Cops 793). Consider the CTRS $\mathcal{R}$ consisting of the two rules:

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

To show infeasibility of the first rule we can safely remove it from $\mathcal{R}$ by Theorem 3, resulting in the modified CTRS $\mathcal{R}'$. Then we have to check $\mathsf{SAT}_{\mathcal{R}'}(\mathsf{f}(\mathsf{a}) \twoheadrightarrow \mathsf{a})$ which is made easier by the following chain of equivalences:

$$\begin{aligned}
\mathsf{f}(\mathsf{a}) \twoheadrightarrow \mathsf{a} &\equiv_{\mathcal{R}'} \mathsf{f}(\mathsf{a}) \xrightarrow{>\epsilon}\!\!\!\!\twoheadrightarrow \mathsf{a} \vee \mathsf{f}(\mathsf{a}) \rightsquigarrow_{\mathsf{f}(x) \to \mathsf{a} \Leftarrow x \twoheadrightarrow \mathsf{b}} \mathsf{a} && \text{(by Lemma 7)}\\
&\equiv_{\mathcal{R}'} \mathsf{f}(\mathsf{a}) \rightsquigarrow_{\mathsf{f}(x) \to \mathsf{a} \Leftarrow x \twoheadrightarrow \mathsf{b}} \mathsf{a} && \text{(by Definition 5)}\\
&\equiv_{\mathcal{R}'} \exists x.\, \mathsf{f}(\mathsf{a}) \xrightarrow{>\epsilon}\!\!\!\!\twoheadrightarrow \mathsf{f}(x) \wedge \mathsf{a} \twoheadrightarrow \mathsf{a} \wedge x \twoheadrightarrow \mathsf{b} && \text{(by Definition 12)}\\
&\equiv_{\mathcal{R}'} \exists x.\, \mathsf{a} \twoheadrightarrow x \wedge \mathsf{a} \twoheadrightarrow \mathsf{a} \wedge x \twoheadrightarrow \mathsf{b} && \text{(by Definition 5)}
\end{aligned}$$

Since satisfiability of the final constraint above implies $\mathsf{SAT}_{\mathcal{R}'}(\mathsf{a} \twoheadrightarrow \mathsf{b})$ and we also have $\mathsf{a} \not\rightarrowtail^*_{\mathcal{R}} \mathsf{b}$, we can conclude unsatisfiability of the original constraint by Corollary 4 and hence that the first rule of $\mathcal{R}$ is infeasible.

Table 1: Experimental results for dependency graph analysis (TRSs).

		Look-ahead				
		$\mathsf{L}^0_\mathcal{R}$	$\mathsf{L}^1_\mathcal{R}$	$\mathsf{L}^2_\mathcal{R}$	$\mathsf{L}^3_\mathcal{R}$	$\mathsf{L}^8_\mathcal{R}$
None	UNSAT	0	104 050	105 574	105 875	105 993
	time (s)	33.96	38.98	38.13	39.15	116.52
Corollary 1	UNSAT	307 207	328 216	328 430	328 499	328 636
	time (s)	38.50	42.71	42.72	43.00	66.82

6 Assessment

We implemented our techniques in the TRS termination prover NaTT [16][6] version 1.8 for dependency graph analysis, and the CTRS confluence prover ConCon [13][7] version 1.7 for infeasibility analysis. In both cases we only need a complete satisfiability checker, or equivalently, a sound unsatisfiability checker. Hence, to conclude unsatisfiability of given reachability constraints, we apply Corollary 2 with appropriate k together with a complete approximation of constraints. One such approximation is the symbol transition graph (Corollary 1). In the following we describe the experimental results on TRS termination and CTRS confluence. Further details of our experiments can be found at `http://cl-informatik.uibk.ac.at/experiments/reachability/`.

TRS Termination. For plain rewriting, we take all the 1498 TRSs from the TRS standard category of the *termination problem data base* version 10.6,[8] the benchmark used in the annual *Termination Competition* [8], and over-approximate their dependency graphs. This results in 1 133 963 reachability constraints, which we call "edges" here. Many of these edges are actually satisfiable, but we do not know the exact number (the problem is undecidable in general).

For checking unsatisfiability of edges, we combine Corollary 2 for various values of k (0, 1, 2, 3, and 8), and either Corollary 1 or 'None'. Here 'None' concludes unsatisfiability only for constraints that are logically equivalent to $\bot$. In Table 1 we give the number of edges that could be shown unsatisfiable. Here, the 'UNSAT' row indicates the number of detected unsatisfiable edges and the 'time' row indicates the total runtime in seconds. (We ran our experiments on an Amazon EC2 instance model `c5.xlarge`: 4 virtual 3.0 GHz Intel Xeon Platinum CPUs on 8GB of memory.)

The starting point is $\mathsf{L}^1_\mathcal{R}$ + None, which corresponds to the `tcap` technique, the method that was already implemented in NaTT before. The benefit of symbol transition graphs turns out to be quite significant, while the overhead in runtime seems acceptable. Moreover, increasing k of the look-ahead reasonably improves the power of unsatisfiability checks, both with and without the symbol transition

[6] `https://www.trs.css.i.nagoya-u.ac.jp/NaTT/`
[7] `http://cl-informatik.uibk.ac.at/software/concon/`
[8] `http://www.termination-portal.org/wiki/TPDB`

graph technique. In terms of the overall termination proving power, NaTT using only tcap solves 1039 out of the 1498 termination problems, while using $\mathsf{L}^8_{\mathcal{R}}$ and Corollary 1, it proves termination of 18 additional problems.

CTRS Confluence. For conditional rewriting, we take the 148 oriented CTRSs of Cops,[9] a benchmark of confluence problems used in the annual *Confluence Competition* [1]. Compared to version 1.5 of ConCon (the winner of the CTRS category in the last competition in 2018) our new version (1.7) can solve five more systems (that is a gain of roughly 3%) by incorporating a combination of Theorem 3, inductive symbol transition graphs (Corollary 4), and k-fold look-ahead (Lemma 7), where for the latter we fixed $k = 1$ since we additionally have to control the level of conditional rewriting.

7 Related Work

Reachability is a classical topic in term rewriting; cf. Genet [7] for a survey. Some modern techniques include the tree-automata-completion approach [6,5] and a Knuth-Bendix completion-like approach [4]. Compared to these lines of work, first of all our interest is not directly in reachability problems but their (un)satisfiability. Middeldorp [12] proposed tree-automata techniques to approximate dependency graphs and made a theoretical comparison to an early term-cap-unifiability method [2], a predecessor of the tcap-based method. It is indeed possible (after some approximations of input TRSs) to encode our satisfiability problems into reachability problems between regular tree languages. However, our main motivation is to efficiently test reachability when analyzing other properties like termination and confluence. In that setting, constructing tree automata often leads to excessive overhead.

Our work is inspired by the work of Lucas and Gutiérrez [11]. Their *feasibility sequences* serve the same purpose as our reachability constraints, but are limited to atoms and conjunctions. Our formulation, allowing other constructions of logic formulas, is essential for introducing look-ahead reachability.

8 Conclusion

We introduced reachability constraints and their satisfiability problem. Such problems appear in termination and confluence analysis of plain and conditional rewriting. Moreover, we proposed two efficient techniques to prove (un)satisfiability of reachability constraints, first for plain and then for conditional rewriting. Finally, we implemented these techniques in the termination prover NaTT and the confluence prover ConCon, and experimentally verified their significance.

Acknowledgments. We thank Aart Middeldorp and the anonymous reviewers for their insightful comments. This work is supported by the Austrian Science Fund (FWF) project P27502 and ERATO HASUO Metamathematics for Systems Design Project (No. JPMJER1603), JST.

[9] `http://cops.uibk.ac.at/?q=oriented+ctrs`

References

1. Aoto, T., Hirokawa, N., Nagele, J., Nishida, N., Zankl, H.: Confluence competition 2015. In: CADE-25. pp. 101–104. Springer (2015). `doi:10.1007/978-3-319-21401-6_5`
2. Arts, T., Giesl, J.: Termination of term rewriting using dependency pairs. Theor. Compt. Sci. **236**(1-2), 133–178 (2000). `doi:0.1016/S0304-3975(99)00207-8`
3. Baader, F., Nipkow, T.: Term Rewriting and All That. Cambridge University Press (1998)
4. Burel, G., Dowek, G., Jiang, Y.: A completion method to decide reachability in rewrite systems. In: FroCoS 2015. pp. 205–219 (2015). `doi:10.1007/978-3-319-24246-0_13`
5. Felgenhauer, B., Thiemann, R.: Reachability, confluence, and termination analysis with state-compatible automata. Information and Computation **253**, 467–483 (2017). `doi:10.1016/j.ic.2016.06.011`
6. Feuillade, G., Genet, T., Viet Triem Tong, V.: Reachability analysis over term rewriting systems. J. Autom. Reasoning **33**(341) (2004). `doi:10.1007/s10817-004-6246-0`
7. Genet, T.: Reachability analysis of rewriting for software verification. Habilitation à diriger des recherches, Université de Rennes 1 (2009)
8. Giesl, J., Mesnard, F., Rubio, A., Thiemann, R., Waldmann, J.: Termination competition (termcomp 2015). In: CADE-25. pp. 105–108. Springer (2015). `doi:10.1007/978-3-319-21401-6_6`
9. Giesl, J., Thiemann, R., Schneider-Kamp, P.: Proving and Disproving Termination of Higher-Order Functions. In: Proceedings of the 5th International Workshop on Frontiers of Combining Systems (FroCoS). Lecture Notes in Artificial Intelligence, vol. 3717, pp. 216–231. Springer (2005). `doi:10.1007/11559306_12`
10. Hirokawa, N., Middeldorp, A.: Dependency pairs revisited. In: RTA 2004. LNCS, vol. 3091, pp. 249–268 (2004). `doi:10.1007/978-3-540-25979-4_18`
11. Lucas, S., Gutiérrez, R.: Use of logical models for proving infeasibility in term rewriting. Inf. Process. Lett. **136**, 90–95 (2018). `doi:10.1016/j.ipl.2018.04.002`
12. Middeldorp, A.: Approximating dependency graphs using tree automata techniques. In: IJCAR 2001. LNCS, vol. 2083, pp. 593–610 (2001). `doi:10.1007/3-540-45744-5_49`
13. Sternagel, T., Middeldorp, A.: Conditional confluence (system description). In: Proceedings of the Joint 25th International Conference on Rewriting Techniques and Applications (RTA) and 12th International Conference on Typed Lambda Calculi and Applications (TLCA). Lecture Notes in Computer Science, vol. 8560, pp. 456–465. Springer (2014). `doi:10.1145/2676724.2693171`
14. TeReSe: Term Rewriting Systems, Cambridge Tracts in Theoretical Computer Science, vol. 55. Cambridge University Press (2003)
15. Toyama, Y.: Counterexamples to termination for the direct sum of term rewriting systems. Information Processing Letters **25**(3), 141–143 (1987). `doi:10.1016/0020-0190(87)90122-0`
16. Yamada, A., Kusakari, K., Sakabe, T.: Nagoya termination tool. In: Proceedings of the Joint 25th International Conference on Rewriting Techniques and Applications (RTA) and 12th International Conference on Typed Lambda Calculi and Applications (TLCA). Lecture Notes in Computer Science, vol. 8560, pp. 466–475. Springer (2014). `doi:10.1007/978-3-319-08918-8_32`