Automatically Proving and Disproving Feasibility Conditions*

Raúl Gutiérrez[0000−0002−3984−2868] and Salvador Lucas[0000−0001−9923−2108]

Valencian Research Institute for Artificial Intelligence
Universitat Politècnica de València
Camino de Vera s/n, E-46022 Valencia, Spain
{rgutierrez,slucas}@dsic.upv.es

Abstract. In the realm of term rewriting, given terms s and t, a reachability condition $s \to^* t$ is called *feasible* if there is a substitution σ such that $\sigma(s)$ rewrites into $\sigma(t)$ in zero or more steps; otherwise, it is called *infeasible*. Checking infeasibility of (sequences of) reachability conditions is important in the analysis of computational properties of rewrite systems like confluence or (operational) termination. In this paper, we generalize this notion of feasibility to arbitrary n-ary relations on terms defined by first-order theories. In this way, properties of computational systems whose operational semantics can be given as a first-order theory can be investigated. We introduce a framework for proving feasibility/infeasibility, and a new tool, infChecker, which implements it.

Keywords: Conditional rewriting · Feasibility · Program analysis.

1 Introduction

The *(in)feasibility* of sequences of goals $s \to^* t$ representing many step rewritings in Conditional Term Rewriting Systems (CTRSs, see [22, Section 7]) has been investigated by several authors. The word "feasibility" refers to the possibility of applying a substitution σ as part of the desired test, i.e., checking whether $\sigma(s) \to^*_{\mathcal{R}} \sigma(t)$ holds for some substitution σ, rather than just checking $s \to^*_{\mathcal{R}} t$ (reachability test). The use of (in)feasibility tests in confluence and (operational) termination analysis of CTRSs has been investigated elsewhere (see, e.g., [13,25] and the references therein). We generalize "feasibility of a reachability problem" by defining *feasibility conditions, sequences and goals* without any specific reference to rewriting systems or rewriting goals. Instead, we rely on first-order logic and use (two layered) sequences of *atoms* headed with a predicate $\bowtie$ as feasibility goals. The meaning of predicates $\bowtie$ is given by using first-order theories $\mathsf{Th}_{\bowtie}$ by provability of the corresponding atoms. New properties (also of CTRSs) can be investigated in this way.

* Supported by EU (FEDER), and projects RTI2018-094403-B-C32, PROMETEO/2019/098, and SP20180225. Also by INCIBE program "Ayudas para la excelencia de los equipos de investigación avanzada en ciberseguridad" (Raúl Gutiérrez).

Example 1. Given a CTRS $\mathcal{R}$, a term t *loops* if $t = t_1 \to_{\mathcal{R}} \cdots \to_{\mathcal{R}} t_n$ for some $n > 1$ such that t is a subterm of t_n, written $t_n \trianglerighteq t$, cf. [2, Def. 3]. Provided that $\to$, $\to^*$ and $\trianglerighteq$ are given appropriate theories (see Example 3 below), *non-loopingness* of ground terms t is the *infeasibility* of the sequence $t \to x, x \to^* y, y \trianglerighteq t$.

Now, looping CTRSs can be defined as those having looping terms. Thus, loopingness of CTRSs can be defined as the *feasibility* of $x \to y, y \to^* z, z \trianglerighteq x$.

In order to automatically analyze such (in)feasibility goals, we describe a framework similar to the Dependency Pair (DP) Framework for proving termination of TRSs [3]. After some preliminaries in Section 2, Section 3 presents the notion of feasibility goal. Section 4 describes the feasibility framework for proving and disproving feasibility goals. Section 5 describes our tool infChecker which provides a (partial) implementation of the framework introduced here. Section 6 provides an experimental evaluation and discusses some related work. Section 8 concludes.

2 Preliminaries

We use the standard notations in term rewriting (see, e.g., [22]). In this paper, $\mathcal{X}$ denotes a countable set of *variables* and $\mathcal{F}$ denotes a *signature*, i.e., a set of *function symbols* $\{f, g, \ldots\}$, 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 symbol labeling the root of t is denoted as $root(t)$. The set of variables occurring in t is $\mathcal{V}ar(t)$. 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 $\mathcal{P}os(t)$. A substitution is a mapping from variables into terms which is homomorphically extended to a mapping from terms to terms. A conditional rule is written $\ell \to r \Leftarrow s_1 \approx t_1, \cdots, s_n \approx t_n$, where $\ell, r, s_1, t_1, \ldots, s_n, t_n \in \mathcal{T}(\mathcal{F}, \mathcal{X})$ and $\ell \notin \mathcal{X}$. As usual, ℓ and r are called the left- and right-hand sides of the rule, and the sequence $s_1 \approx t_1, \cdots, s_n \approx t_n$ (often abbreviated to c) is the *conditional part* of the rule. We often write $s_i \approx t_i \in c$ to refer to the i-th atomic condition in c or $s \to t \in c$ if the position of the atomic condition in c does not matter. Rules $\ell \to r \Leftarrow c$ are classified according to the distribution of variables as follows: type 1 (or 1-rules), if $\mathcal{V}ar(r) \cup \mathcal{V}ar(c) \subseteq \mathcal{V}ar(\ell)$; type 2, if $\mathcal{V}ar(r) \subseteq \mathcal{V}ar(\ell)$; type 3, if $\mathcal{V}ar(r) \subseteq \mathcal{V}ar(\ell) \cup \mathcal{V}ar(c)$; and type 4, if no restriction is given. A CTRS $\mathcal{R}$ is a set of conditional rules; $\mathcal{R}$ is called an n-CTRS if it contains only n-rules; A 3-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 $1 \le i \le n$, we have $\mathcal{V}ar(s_i) \subseteq \mathcal{V}ar(\ell) \cup \bigcup_{j=1}^{i-1} \mathcal{V}ar(t_j)$. *Oriented* CTRSs are those whose conditions $s \approx t$ are handled as *reachability* tests $\sigma(s) \to^* \sigma(t)$ for an appropriate substitution σ. For oriented CTRSs $\mathcal{R}$, an inference system $\mathcal{I}(\mathcal{R})$ is obtained from the following generic inference system $\mathfrak{I}_{\mathrm{CTRS}}$:

$$\text{(Rf)}\quad \frac{}{x \to^* x} \qquad \text{(C)}_{f,i}\quad \frac{x_i \to y_i}{f(x_1, \ldots, x_i, \ldots, x_k) \to f(x_1, \ldots, y_i, \ldots, x_k)} \text{ for all } f \in \mathcal{F}^{(k)} \text{ and } 1 \le i \le k$$

$$\text{(T)}\quad \frac{x \to y \qquad y \to^* z}{x \to^* z} \qquad \text{(Rl)}_{\alpha}\quad \frac{s_1 \to^* t_1 \quad \cdots \quad s_n \to^* t_n}{\ell \to r} \text{ for } \alpha : \ell \to r \Leftarrow s_1 \approx t_1, \ldots, s_n \approx t_n \in \mathcal{R}$$

$$le(0, s(y)) \to true \tag{1}$$
$$le(s(x), s(y)) \to le(x, y) \tag{2}$$
$$le(x, 0) \to false \tag{3}$$
$$min(cons(x, nil)) \to x \tag{4}$$
$$min(cons(x, xs)) \to x \Leftarrow min(xs) \approx y, le(x, y) \approx true \tag{5}$$
$$min(cons(x, xs)) \to y \Leftarrow min(xs) \approx y, le(x, y) \approx false \tag{6}$$

Fig. 1. CTRS 551.trs in COPS database of confluence problems.

by *specializing* $(\mathrm{C})_{f,i}$ for each k-ary symbol f in the signature $\mathcal{F}$ and $1 \leq i \leq k$ and $(\mathrm{Rl})_\alpha$ for all conditional rules $\alpha : \ell \to r \Leftarrow c$ in $\mathcal{R}$. Rules in $\mathcal{I}(\mathcal{R})$ are *schematic*: each inference rule $\frac{B_1 \ \cdots \ B_n}{A}$ can be used under any *instance* $\frac{\sigma(B_1) \ \cdots \ \sigma(B_n)}{\sigma(A)}$ of the rule by a substitution σ. We write $s \to_{\mathcal{R}} t$ (resp. $s \to^*_{\mathcal{R}} t$) iff there is a proof tree for $s \to t$ (resp. $s \to^* t$) using $\mathcal{I}(\mathcal{R})$. Operational termination of $\mathcal{R}$ is defined as the absence of infinite proof trees for goals $s \to t$ and $s \to^* t$ in $\mathcal{I}(\mathcal{R})$ [14].

A *structure* $\mathcal{A}$ for a first-order language is an interpretation of the function and predicate symbols ($f, g, \ldots$ and $P, Q, \ldots$, respectively) as mappings $f^{\mathcal{A}}, g^{\mathcal{A}}, \ldots$ and relations $P^{\mathcal{A}}, Q^{\mathcal{A}}, \ldots$ on a given set (carrier) also denoted $\mathcal{A}$. Then, the usual interpretation of first-order formulas with respect to $\mathcal{A}$ is considered. A model for a theory Th, i.e., a set of first-order sentences (formulas whose variables are all *quantified*), is just a structure $\mathcal{A}$ that makes them all true, written $\mathcal{A} \models \mathsf{Th}$. In the following, $\mathsf{Th} \vdash \varphi$ means that formula φ is a *logical consequence* of Th. We assume the use of a sound and complete proof method, in particular Gentzen's natural deduction, see [23]. In this setting, we often assume the use of the inference rules of natural deduction [23, p. 20] to deal with logical connectives and quantifiers when necessary.

3 Feasibility of Sequences and Goals

Consider a signature Σ of function symbols and a set Π of predicate symbols. As in [4], (Σ, Π) is often called a *signature with predicates*. Let $\mathcal{F} \subseteq \Sigma$ be a signature and $\mathbb{P} \subseteq \Pi$ be a set of predicates (e.g., $\mathbb{P} = \{\to, \to^*, \downarrow, \leftrightarrow, \leftrightarrow^*, \trianglerighteq, \ldots\}$).[1] Let $\mathbb{T} = \{\mathsf{Th}_{\bowtie} \mid {\bowtie} \in \mathbb{P}\}$ be a $\mathbb{P}$-indexed set of first-order theories $\mathsf{Th}_{\bowtie}$ defining the predicates $\bowtie$ in $\mathbb{P}$, possibly involving predicate symbols which are *not* in $\mathbb{P}$.

Example 2. For the CTRS $\mathcal{R}$ in Figure 1, we obtain a theory $\overline{\mathcal{R}}$ from $\mathcal{I}(\mathcal{R})$ as follows [11, Section 4.5]: the inference rules $(\rho)\frac{B_1 \ \cdots \ B_n}{A}$ in $\mathcal{I}(\mathcal{R})$ are considered as *sentences* $\overline{\rho}$ of the form $(\forall \boldsymbol{x})\ B_1 \wedge \cdots \wedge B_n \Rightarrow A$, where $\boldsymbol{x}$ is the sequence of variables occurring in atoms $B_1, \ldots, B_n$ and A; if empty, we just write $B_1 \wedge \cdots \wedge B_n \Rightarrow A$ (see Figure 2). For $\mathbb{P} = \{\to, \to^*\}$, we let $\mathsf{Th}_{\to} = \mathsf{Th}_{\to^*} = \overline{\mathcal{R}}$.

Examples 6 and 7 illustrate and motivate the use of theories involving predicates not in $\mathbb{P}$.

[1] For simplicity, in our exposition we restrict the attention to *binary* predicates, but the techniques and results in this paper easily generalize to n-ary predicates.

$$(\forall x)\ x \to^* x$$
$$(\forall x, y, z)\ x \to y \wedge y \to^* z \Rightarrow x \to^* z$$
$$(\forall x, y)\ x \to y \Rightarrow s(x) \to s(y)$$
$$(\forall x, y, z)\ x \to y \Rightarrow cons(x, z) \to cons(y, z)$$
$$(\forall x, y, z)\ x \to y \Rightarrow cons(z, x) \to cons(z, y)$$
$$(\forall x, y, z)\ x \to y \Rightarrow le(x, z) \to le(y, z)$$
$$(\forall x, y, z)\ x \to y \Rightarrow le(z, x) \to le(z, y)$$
$$(\forall x, y)\ x \to y \Rightarrow min(x) \to min(y)$$
$$(\forall y)\ le(0, s(y)) \to true$$
$$(\forall x, y)\ le(s(x), s(y)) \to le(x, y)$$
$$(\forall x)\ le(x, 0) \to false$$
$$(\forall x)\ min(cons(x, nil)) \to x$$
$$(\forall x, y, xs)\ min(xs) \to^* y \wedge le(x, y) \to^* true \Rightarrow min(cons(x, xs)) \to x$$
$$(\forall x, xs)\ min(xs) \to^* y \wedge le(x, y) \to^* false \Rightarrow min(cons(x, xs)) \to y$$

Fig. 2. Theory $\overline{\mathcal{R}}$ for the CTRS $\mathcal{R}$ in Example 2

An $(\mathcal{F}, \mathbb{P})$-*f-condition* γ (or just *f-condition* if $\mathcal{F}$ and $\mathbb{P}$ are clear from the context) is an atom $s \bowtie t$ where $\bowtie \in \mathbb{P}$ and $s, t \in \mathcal{T}(\mathcal{F}, \mathcal{X})$. Sequences $\mathsf{F} = (\gamma_i)_{i=1}^n = (\gamma_1, \ldots, \gamma_n)$ of f-conditions are called *f-sequences*. A set $\mathcal{G} = \{\mathsf{F}_1; \ldots; \mathsf{F}_m\}$ of f-sequences is called an *f-goal*; we use ';' intead of ',' which is already considered in f-sequences. We often drop 'f-' when no confusion arises. Empty sequences and goals are written () and {}.

Remark 1 (Notation). In the following, we often use '$\in$' to denote membership of components in both sequences and goals.

Definition 1 (Feasibility). *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. A* goal $\mathcal{G}$ *is* feasible *iff it contains a feasible sequence $\mathsf{F} \in \mathcal{G}$. Now,* {} *is trivially* infeasible.

Example 3. (continuing Example 1) We can prove a ground term t *non-looping* as the $\mathbb{T}$-infeasibility of $\mathcal{G} = \{(t \to y, y \to^* z, z \trianglerighteq t)\}$, with x, y, and z variables, and $\mathbb{T} = \{\mathsf{Th}_{\to}, \mathsf{Th}_{\to^*}, \mathsf{Th}_{\trianglerighteq}\}$ such that $\mathsf{Th}_{\to} = \mathsf{Th}_{\to^*} = \overline{\mathcal{R}}$ and $\mathsf{Th}_{\trianglerighteq}$ is given by:

$$(\forall x)\ x \trianglerighteq x \qquad (7) \qquad\qquad (\forall x_1, \ldots, x_k)\ f(x_1, \ldots, x_k) \trianglerighteq x_i \qquad (9)$$
$$(\forall x, y, z)\ x \trianglerighteq y \wedge y \trianglerighteq z \Rightarrow x \trianglerighteq z \qquad (8) \qquad\qquad \text{for each } f \in \mathcal{F} \text{ and } 1 \le i \le k$$

Example 4. A term t is *root-stable* (with respect to a TRS $\mathcal{R}$) if t cannot be reduced to a redex, i.e., there is no rule $\ell \to r \in \mathcal{R}$ such that $t \to^* \sigma(\ell)$ for some substitution σ. If $\mathcal{R}$ consists of rules $\ell_1 \to r_1, \ldots, \ell_p \to r_p$ (assume that different rules in $\mathcal{R}$ share no variable), we can prove a *ground* term t root-stable by showing the $\{\mathsf{Th}_{\to^*}\}$-infeasibility of $\mathcal{G} = \{(t \to^* \ell_1); \cdots; (t \to^* \ell_p)\}$ with $\mathsf{Th}_{\to^*} = \overline{\mathcal{R}}$.

More examples of theories $\mathsf{Th}_{\bowtie}$ can be found in [9, Sections 5.3 and 5.4] and [11, Sections 8.1 and 8.2].

Given theories $\mathsf{Th}, \mathsf{Th}'$ and a set of atoms $\mathbb{A}$, we write $\mathsf{Th} \equiv_{\mathbb{A}} \mathsf{Th}'$ if for all $A \in \mathbb{A}$, $\mathsf{Th} \vdash A$ if and only if $\mathsf{Th}' \vdash A$. Also, given a set of atoms $\mathbb{A}$ and a predicate symbol $\bowtie$, $\mathbb{A}_{\bowtie}$ is the subset of atoms in $\mathbb{A}$ with root $\bowtie$.

Definition 2. *Given a set of predicates $\mathbb{P}$ and a $\mathbb{P}$-indexed set of theories $\mathbb{T}$, we say that a theory Th* preserves *a feasibility sequence F (in $\mathbb{T}$) if for all predicates $\bowtie$ occurring in F, $\mathsf{Th} \equiv_{\mathbb{A}_{\bowtie}} \mathsf{Th}_{\bowtie}$ holds. Thus, Th cannot prove more atoms rooted with $\bowtie$ than $\mathsf{Th}_{\bowtie}$. Similarly, Th preserves a goal $\mathcal{G} = \{\mathsf{F}_i\}_{i=1}^m$ if it preserves F_i for all $1 \leq i \leq m$.*

In the following, when no confusion arises, we do not explicitly mention the underlying set of theories $\mathbb{T}$. Given $\mathcal{G} = \{\mathsf{F}_i\}_{i=1}^m$ and $\mathsf{F}_i = (s_{ij} \bowtie_{ij} t_{ij})_{j=1}^{n_i}$ for $1 \leq i \leq m$, we let $\mathsf{Th}_{\mathsf{F}_i} = \bigcup_{j=1}^{n_i} \mathsf{Th}_{\bowtie_{ij}}$ and $\mathsf{Th}_{\mathcal{G}} = \bigcup_{i=1}^{m} \mathsf{Th}_{\mathsf{F}_i}$.

Example 5. It is not difficult to see that $\mathsf{Th}_{\mathcal{G}}$ preserves $\mathcal{G}$ in Example 3.

The following result provides a (first-order) provability perspective of feasibility.

Theorem 1. *1. A condition $\gamma = s \bowtie t$ is feasible iff $\mathsf{Th}_{\bowtie} \vdash (\exists \boldsymbol{x}) s \bowtie t$ holds.*

2. *If $\mathsf{F} = (s_i \bowtie_i t_i)_{i=1}^n$ is feasible, then $\mathsf{Th}_{\mathsf{F}} \vdash (\exists \boldsymbol{x}) \bigwedge_{i=1}^n s_i \bowtie_i t_i$ holds. If Th_{F} preserves F and $\mathsf{Th}_{\mathsf{F}} \vdash (\exists \boldsymbol{x}) \bigwedge_{i=1}^n s_i \bowtie_i t_i$ holds, then F is feasible.*
3. *If $\mathcal{G} = \{\mathsf{F}_i\}_{i=1}^m$, where $\mathsf{F}_i = (s_{ij} \bowtie_{ij} t_{ij})_{j=1}^{n_i}$ for some n_i, is feasible, then we have that $\mathsf{Th}_{\mathcal{G}} \vdash (\exists \boldsymbol{x}) \bigvee_{i=1}^m \bigwedge_{j=1}^{n_i} s_{ij} \bowtie_{ij} t_{ij}$ holds. If $\mathsf{Th}_{\mathcal{G}} \vdash (\exists \boldsymbol{x}) \bigvee_{i=1}^m \bigwedge_{j=1}^{n_i} s_{ij} \bowtie_{ij} t_{ij}$ holds and $\mathsf{Th}_{\mathcal{G}}$ preserves $\mathcal{G}$, then $\mathcal{G}$ is feasible. Also, if there is $1 \leq i \leq m$ such that $\mathsf{Th}_{\mathsf{F}_i}$ preserves F_i and $\mathsf{Th}_{\mathsf{F}_i} \vdash (\exists \boldsymbol{x}) \bigwedge_{j=1}^{n_i} s_{ij} \bowtie_{ij} t_{ij}$ holds, then $\mathcal{G}$ is feasible.*

Sentences in Theorem 1 are *Existentially Closed Boolean Combinations of Atoms* (ECBCAs), i.e., formulas of the form $(\exists \boldsymbol{x}) \bigvee_{i=1}^m \bigwedge_{j=1}^{n_i} A_{ij}$, where A_{ij} are atoms and $\boldsymbol{x}$ is the sequence of variables occurring in such atoms. We have investigated them in [9,11]. Requiring preservation is necessary for items (2) and (3) in Theorem 1.

Example 6. Let $\mathcal{R}_1 = \{a \to b \Leftarrow b \approx a\}$ and $\mathcal{R}_2 = \{b \to a\}$. With $\mathbb{P} = \{\to, \to^*\}$ and $\mathbb{T} = \{\mathsf{Th}_{\to}, \mathsf{Th}_{\to^*}\}$, where $\mathsf{Th}_{\to} = \overline{\mathcal{R}}_1 = \{(\forall x)\, x \to^* x,\ (\forall x, y, z)\, x \to y \wedge y \to^* z \Rightarrow x \to^* z, b \to^* a \Rightarrow a \to b\}$ and $\mathsf{Th}_{\to^*} = \overline{\mathcal{R}}_2 = \{(\forall x)\, x \to^* x,\ (\forall x, y, z)\, x \to y \wedge y \to^* z \Rightarrow x \to^* z, b \to a\}$, we have $\mathsf{Th} = \overline{\mathcal{R}}_1 \cup \overline{\mathcal{R}}_2$ and $\mathsf{Th} \vdash a \to b$. However, $a \to b$ is not $\mathbb{T}$-feasible because $\overline{\mathcal{R}}_1 \nvdash a \to b$. Note that Th does not preserve $(a \to b)$.

Preservation is often achieved by distinguishing predicates describing different computations.

Example 7. Let $\overline{\mathcal{R}}_1'$ and $\overline{\mathcal{R}}_2'$ be theories for $\mathcal{R}_1$ and $\mathcal{R}_2$ in Example 6, where $\to_1^*$ is used instead of $\to^*$ in $\overline{\mathcal{R}}_1$ (but $\to$ remains as it is) to yield $\overline{\mathcal{R}}_1' = \{(\forall x) x \to_1^* x,\ (\forall x, y, z) x \to y \wedge y \to_1^* z \Rightarrow x \to_1^* z, b \to_1^* a \Rightarrow a \to b\}$ and $\to_2$ is used instead of $\to$ in $\overline{\mathcal{R}}_2$ (and $\to^*$ is still used) to yield $\overline{\mathcal{R}}_2' = \{(\forall x)\, x \to^* x,\ (\forall x, y, z)\, x \to_2 y \wedge y \to^* z \Rightarrow x \to^* z, b \to_2 a\}$. With $\mathsf{Th}'_{\to} = \overline{\mathcal{R}}_1'$ and $\mathsf{Th}'_{\to^*} = \overline{\mathcal{R}}_2'$ (and $\mathbb{T}' = \{\mathsf{Th}'_{\to}, \mathsf{Th}'_{\to^*}\}$), we have that $\mathsf{Th}' = \overline{\mathcal{R}}_1' \cup \overline{\mathcal{R}}_2'$ preserves $\mathcal{G} = \{(a \to b)\}$ and $\mathsf{Th}' \nvdash a \to b$. By Theorem 1 $a \to b$, is not $\mathbb{T}'$-feasible, as expected.

Theorem 1 characterizes feasibility of a goal $\mathcal{G}$ as provability of an ECBCA $\varphi_{\mathcal{G}}$. If $\varphi_{\mathcal{G}}$ is shown *unprovable*, we conclude infeasibility of $\mathcal{G}$. And, under appropriate preservation conditions, theorem proving can be used to conclude feasibility of a goal $\mathcal{G}$. However,

(un)provability of atoms is often undecidable. For instance, for TRSs $\mathcal{R}$, it is well-known that given ground terms s and t, it is in general undecidable whether s rewrites into t; i.e., whether $\overline{\mathcal{R}} \vdash s \to^* t$ holds (as Post's correspondence problem is a particular case, see, e.g., [22, Section 4.1]). Hence, feasibility of conditions, sequences and goals remains, in general, undecidable. In order to obtain automatic proofs of feasibility, it is often useful to proceed using a 'divide-and-conquer' strategy. In the following, we exploit this idea to define a practical framework to prove/disprove feasibility of goals.

4 Feasibility Framework

In [3], proofs of termination of TRSs proceed by transforming the so-called *DP problems* τ. A divide-and-conquer approach is applied by means of *processors* P mapping a DP problem τ into a (possibly empty) set $\mathsf{P}(\tau)$ of DP problems $\{\tau_1, \ldots, \tau_n\}$. DP problems τ_i returned by P can now be treated independently by using other processors. In this way, a *DP proof tree* is built.

In our setting, we first define notions of *f-problem* and *f-processor*, and then show how to use them to (dis)prove feasibility.

Definition 3 (f-Problem and f-Processor). *Given a set of predicates $\mathbb{P}$, a $\mathbb{P}$-indexed theory $\mathbb{T}$, and a goal $\mathcal{G}$, a pair $\tau = (\mathbb{T}, \mathcal{G})$ is called an* f-Problem. *We say that τ is* feasible *if $\mathcal{G}$ is $\mathbb{T}$-feasible; otherwise it is* infeasible.

An f-Processor P *is a partial function from f-Problems into sets of f-Problems. Alternatively, it can return* "yes". *$\mathcal{D}om(\mathsf{P})$ represents the domain of P, i.e., the set of f-Problems τ that P is defined for.*

Definition 4 (Soundness and Completness). *Let P be an f-Processor and $\tau \in \mathcal{D}om(\mathsf{P})$. We say that P is*

- sound *iff τ is feasible whenever either $\mathsf{P}(\tau) =$ "yes" or $\exists \tau' \in \mathsf{P}(\tau)$, such that τ' is feasible.*
- complete *iff τ is infeasible whenever $\mathsf{P}(\tau) \neq$ "yes" and $\forall \tau' \in \mathsf{P}(\tau)$, τ' is infeasible.*

Feasibility problems can be proved or disproved by using a proof tree as follows (where inner nodes include the root of the tree unless it consists of a single node).

Definition 5 (Feasibility Proof Tree). *Let $\tau = (\mathbb{T}, \mathcal{G})$ be an f-Problem. A feasibility proof tree (FP Tree) $\mathcal{T}$ for τ is a tree whose inner nodes are labeled with f-Problems and the leaves are labeled either with f-Problems,* "yes" *or* "no". *The root of $\mathcal{T}$ is labeled with τ and for every inner node n labeled with τ', there is an f-Processor P such that $\tau' \in \mathcal{D}om(\mathsf{P})$ and:*

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

Theorem 2 (Feasibility Framework). *Let $\mathcal{T}$ be a feasibility proof tree for $\tau_I = (\mathbb{T}, \mathcal{G})$. Then:*

1. *if all leaves in $\mathcal{T}$ are labeled with* "no" *and all involved f-Processors are complete for the f-Problems they are applied to, then $\mathcal{G}$ is $\mathbb{T}$-infeasible.*
2. *if $\mathcal{T}$ has a leaf labeled with* "yes" *and all f-Processors in the path from τ_I to the leaf are sound for the f-Problems they are applied to, then $\mathcal{G}$ is $\mathbb{T}$-feasible.*

In the following, we describe some sound and complete f-Processors. If no confusion arises, we use *processor* instead of *f-Processor*.

4.1 Splitting Processor

Our first processor decomposes a feasibility goal into its feasibility sequences.

Definition 6 (Splitting Processor). *Let $\tau = (\mathbb{T}, \mathcal{G})$ be an f-Problem. The processor $\mathsf{P}^{\mathsf{Spl}}$ is given by $\mathsf{P}^{\mathsf{Spl}}(\tau) = \{(\mathbb{T}, \{\mathsf{F}\}) \mid \mathsf{F} \in \mathcal{G}\}$.*

The proof of the following result is immediate by using Definitions 3 and 1.

Theorem 3. *Processor $\mathsf{P}^{\mathsf{Spl}}$ is sound and complete.*

Example 8. Consider the following TRS $\mathcal{R}$ [13, Example 9]:

$$a \to b \qquad (10) \qquad\qquad f(x,x) \to c \qquad (12)$$
$$b \to a \qquad (11)$$

Following Example 4, we prove root-stability of $f(a,c)$ as the $\{\overline{\mathcal{R}}\}$-infeasibility of $\mathcal{G} = \{((f(a,c) \to^* a)\,; (f(a,c) \to^* b); (f(a,c) \to^* f(x,x))\}$. With $\mathsf{P}^{\mathsf{Spl}}$ we start the proof of infeasibility of $\tau = (\{\overline{\mathcal{R}}\}, \mathcal{G})$ as follows: $\mathsf{P}^{\mathsf{Spl}}(\tau) = \{\tau_1, \tau_2, \tau_3\}$, where $\tau_1 = (\{\overline{\mathcal{R}}\}, \{(f(a,c) \to^* a)\})$, $\tau_2 = (\{\overline{\mathcal{R}}\}, \{(f(a,c) \to^* b)\})$, and $\tau_3 = ((\{\overline{\mathcal{R}}\}, \{(f(a,c) \to^* f(x,x))\})$.

4.2 Provability Processor

Our next processor exploits Theorem 1 to use theorem proving in proofs of feasibility.

Definition 7 (Provability processor). *Let $\tau = (\mathbb{T}, \mathcal{G})$ be an f-Problem with $\mathcal{G} = \{\mathsf{F}\} \uplus \mathcal{G}'$ where $\mathsf{F} = (s_i \bowtie_i t_i)_{i=1}^n$. Processor $\mathsf{P}^{\mathsf{Prov}}$ is given by*

$$\mathsf{P}^{\mathsf{Prov}}(\tau) = \text{“}\mathsf{yes}\text{”} \text{ iff } \mathsf{Th}_{\mathsf{F}} \vdash (\exists \boldsymbol{x}) \bigwedge_{i=1}^{n} s_i \bowtie_i t_i \text{ holds.}$$

Note that, whenever $n = 0$, i.e., $\mathsf{F} = ()$, then $\bigwedge_{i=1}^n s_i \bowtie_i t_i$ is true and $\mathsf{P}^{\mathsf{Prov}}(\tau) = \text{“}\mathsf{yes}\text{”}$.

Theorem 4. *Processor $\mathsf{P}^{\mathsf{Prov}}$ is complete. If Th_{F} preserves F, then it is sound.*

In infChecker, we use Prover9 [18] as a backend to implement $\mathsf{P}^{\mathsf{Prov}}$.

Example 9. For $\mathcal{R}$ in Figure 1, the feasibility goal $\mathcal{G}$ (see file 903 in COPS):

$$\{(le(x, min(y)) \to^* false, min(y) \to^* x)\}$$

and the corresponding first-order formula:

$$(\exists x, y)\ le(x, min(y)) \to^* false \wedge min(y) \to^* x \qquad (13)$$

with $\tau_I = (\{\overline{\mathcal{R}}\}, \mathcal{G})$, we have $\mathsf{P}^{\mathsf{Prov}}(\tau_I) = \text{“}\mathsf{yes}\text{”}$ by using Prover9 to prove (13) by resolution as follows[2]:

[2] For readability, the output is slightly pretty printed.

```
(11) exists x y (le(x,min(y)) ->* false) & (min(y) ->* x) [goal]
(12) x ->* x [assumption]
(13) -(x -> y) | -(y ->* z) | (x ->* z) [assumption]
(15) -(x -> y) | (le(z,x) -> le(z,y)) [assumption]
(22) le(x,0) -> false [assumption]
(23) min(cons(x,nil)) -> x [assumption]
(27) -(le(x,min(y)) ->* false) | -(min(y) ->* x) [deny(11)]
(48) le(x,0) ->* false [ur(13,22,12)]
(59) -(le(min(x),min(x)) ->* false) [resolve(27,12)]
(67) -(le(min(x),min(x)) -> y) | -(y ->* false) [resolve(59,13)]
(69) -(le(min(x),y) ->* false) | -(min(x) -> y) [resolve(67,15)]
(76) -(le(min(cons(x,nil)),x) ->* false) [resolve(69,23)]
(77) $F [resolve(76,48)]
```

Example 10. Consider the two rules TRS $\mathcal{R} = \{a \to c(b), b \to c(b)\}$. For $\overline{\mathcal{R}} = \{(14) - (18)\}$ and $\mathsf{Th}_{\trianglerighteq} = \{(19) - (21)\}$:

$$(\forall x)\ x \to^* x \quad (14) \qquad (\forall x)\ x \trianglerighteq x \quad (19)$$

$$(\forall x, y, z)\ (x \to y \wedge y \to^* z \Rightarrow x \to^* z) \quad (15) \qquad (\forall x, y, z)\ x \trianglerighteq y \wedge y \trianglerighteq z \Rightarrow x \trianglerighteq z \quad (20)$$

$$(\forall x, y)\ (x \to y \Rightarrow c(x) \to c(y)) \quad (16) \qquad (\forall x)\ c(x) \trianglerighteq x \quad (21)$$

$$a \to c(b) \quad (17)$$

$$b \to c(b) \quad (18)$$

$\mathsf{infChecker}$ can prove loopingness of $\mathcal{R}$ as the feasibility of $(\{\overline{\mathcal{R}}, \mathsf{Th}_{\trianglerighteq}\}, \mathcal{G})$ by relying on $\mathsf{Prover9}$ with $\mathcal{G} = \{(x \to y, y \to^* z, z \trianglerighteq x)\}$ (see Example 1). Note that the union of $\overline{\mathcal{R}}$ and $\mathsf{Th}_{\trianglerighteq}$ preserves both $\overline{\mathcal{R}}$ and $\mathsf{Th}_{\trianglerighteq}$, as required for soundness of $\mathsf{P}^{\mathsf{Prov}}$.

If no proof of $\varphi_{\mathsf{F}} = (\exists \boldsymbol{x})\ \bigwedge_{i=1}^{n} s_i \bowtie_i t_i$ is found, then $\mathsf{P}^{\mathsf{Prov}}$ does *not* apply. In this case, it is still possible that F is feasible, but the proof system failed to prove it. Also, it is possible that F is infeasible. In this case, our next processor, which tries to prove infeasibility as satisfiability [9], can be useful.

4.3 Satisfiability Processors

The next processor implements the satisfiability approach in [9].

Definition 8 (Satisfiability Processor). *Let $\tau = (\mathbb{T}, \mathcal{G})$ be an f-Problem with $\mathcal{G} = \{\mathsf{F}\} \uplus \mathcal{G}'$ for $\mathsf{F} = (s_i \bowtie_i t_i)_{i=1}^{n}$ and $\mathcal{A}$ be a structure. Processor $\mathsf{P}^{\mathsf{Sat}}$ is given by $\mathsf{P}^{\mathsf{Sat}}(\tau) = (\mathbb{T}, \mathcal{G}')$ iff $\mathcal{A} \models\ \mathsf{Th}_{\mathsf{F}} \cup \{\neg(\exists \boldsymbol{x})\ \bigwedge_{i=1}^{n} s_i \bowtie_i t_i\}$.*

Remark 2. In the following, the soundness and completeness theorems given for the different introduced processors assume the notations previously introduced in the corresponding definitions.

In the following, we say that a theory Th is *stable* if for all terms s, t and substitutions σ, if $\mathsf{Th} \vdash s \bowtie t$, then $\mathsf{Th} \vdash \sigma(s) \bowtie \sigma(t)$.

Theorem 5. *Processor $\mathsf{P}^{\mathsf{Sat}}$ is sound. If $\mathcal{T}(\mathcal{F}) \neq \emptyset$ and Th_{F} is stable, then it is complete.*

In infChecker, we use the model generators AGES [5] and Mace4 [18] to find suitable structures $\mathcal{A}$ to be used in the implementation of $\mathsf{P}^{\mathsf{Sat}}$.

Example 11. For $\mathcal{R}$, $\overline{\mathcal{R}}$ and $\mathsf{Th}_{\trianglerighteq}$ as in Example 10, we can prove term a non-looping. The following structure over $\mathbb{N} \cup \{-1\}$:

$$\begin{array}{ccc} a^{\mathcal{A}} = -1 & b^{\mathcal{A}} = 1 & c^{\mathcal{A}}(x) = x \\ x \to^{\mathcal{A}} y \Leftrightarrow x \leq 1 \wedge y \geq 1 & x\ (\to^*)^{\mathcal{A}}\ y \Leftrightarrow x \leq y & x \trianglerighteq^{\mathcal{A}} y \Leftrightarrow x \leq y \end{array}$$

satisfies $\overline{\mathcal{R}} \cup \mathsf{Th}_{\trianglerighteq} \cup \{\neg(\exists x, y)\ (a \to x \wedge x \to^* y \wedge y \trianglerighteq a)\}$. Thus, a is non-looping.

The following version of $\mathsf{P}^{\mathsf{Sat}}$ often provides a direct answer about infeasibility of a goal.

Definition 9 (One-Step Satisfiability Processor). *Let $\tau = (\mathbb{T}, \mathcal{G})$ be an f-Problem with $\mathcal{G} = \{\mathsf{F}_1; \cdots; \mathsf{F}_m\}$, where, for all $1 \leq i \leq m$, F_i is $(s_{i1} \bowtie_{i1} t_{i1}, \ldots, s_{in_i} \bowtie_{in_i} t_{in_i})$ for some $n_i > 0$. Let $\mathcal{A}$ be a structure. Processor $\mathsf{P}^{\mathsf{SatAll}}$ is given by $\mathsf{P}^{\mathsf{SatAll}}(\tau) = \emptyset$ iff $\mathcal{A} \models \mathsf{Th}_{\mathcal{G}} \cup \{\neg(\exists \boldsymbol{x})\ \bigvee_{i=1}^{m} \bigwedge_{j=1}^{n_i} s_{ij} \bowtie_{ij} t_{ij}\}$.*

Theorem 6. *Processor $\mathsf{P}^{\mathsf{SatAll}}$ is sound. If $\mathcal{T}(\mathcal{F}) \neq \emptyset$ and $\mathsf{Th}_{\mathcal{G}}$ is stable, then it is complete.*

Example 12. (continuing Example 8) With Mace4 we obtain a model of

$$\overline{\mathcal{R}} \cup \{\neg(\exists x)\ (f(a,c) \to^* a \vee f(a,c) \to^* b \vee f(a,c) \to^* f(x,x))\}$$

Since $\mathsf{P}^{\mathsf{SatAll}}(\tau) = \emptyset$, $\mathcal{G}$ is $\mathbb{T}$-infeasible. This is an alternative proof without $\mathsf{P}^{\mathsf{Spl}}$.

Although $\mathsf{P}^{\mathsf{SatAll}}$ can be simulated by a single step of $\mathsf{P}^{\mathsf{Spl}}$ followed by the application of $\mathsf{P}^{\mathsf{Sat}}$ to the obtained f-Problems, from a practical point of view $\mathsf{P}^{\mathsf{SatAll}}$ has the advantage of avoiding the overloading due to the split the initial goal into a set of sequences with the generation of several models by means of calls to (external) model generators. Instead, $\mathsf{P}^{\mathsf{SatAll}}$ makes a single call to the model generator(s).

4.4 Usable Rules in CTRS Theories

As discussed in [13, Section 2], dealing with CTRSs $\mathcal{R} = (\mathcal{F}, R)$, we may often drop some rules in R before establishing (in)feasibility of conditions $s \to^* t$. First, consider the following overapproximation of the set of rules that can be applied to a term t:

$$RULES(\mathcal{R}, t) = \{\ell \to r \Leftarrow c \in R \mid \exists p \in \mathcal{P}os(t), root(\ell) = root(t|_p)\}$$

The set of *usable rules* for t is defined as follows:

$$\mathcal{U}(\mathcal{R}, t) = RULES(\mathcal{R}, t) \cup \bigcup_{l \to r \Leftarrow c \in RULES(\mathcal{R},t)} \left(\mathcal{U}(\mathcal{R}^\sharp, r) \cup \bigcup_{s \approx t \in c} \mathcal{U}(\mathcal{R}^\sharp, s) \right)$$

where $\mathcal{R}^\sharp = \mathcal{R} - RULES(\mathcal{R}, t)$.[3] Given a sequence F, we let

$$\mathcal{U}_{\to^*}(\mathcal{R}, \mathsf{F}) = \bigcup_{s \to^* t \in \mathsf{F}} \mathcal{U}(\mathcal{R}, s) \tag{22}$$

[3] The use of $\mathcal{R}^\sharp$ instead of $\mathcal{R}$ is important for implementing the computation of usable rules. By decreasing (from $\mathcal{R}$ to $\mathcal{R}^\sharp$) the set of considered rules, the recursive definition is shown to be terminating.

Example 13. For $\mathcal{R}$ in Figure 1 and $\mathsf{F} = (le(0, s(0)) \to^* x), \mathcal{U}_{\to^*}(\mathcal{R}, \mathsf{F}) = \{(1), (2), (3)\}$.

Let $\overline{\mathcal{R}}|_{\mathcal{U}_{\to^*}(\mathcal{R},\mathsf{F})}$ be the first-order theory for the CTRS $(\mathcal{F}, \mathcal{U}_{\to^*}(\mathcal{R}, \mathsf{F}))$, which keeps the original signature of $\mathcal{R}$.

Definition 10. *Let $\tau = (\mathbb{T}, \mathcal{G})$ be an f-Problem such that $\mathbb{T} = \{\mathsf{Th}_{\to^*}\} \uplus \mathbb{T}'$ with $\mathsf{Th}_{\to^*} = \overline{\mathcal{R}}$ for a CTRS $\mathcal{R}$ and $\mathcal{G} = \{\mathsf{F}\} \uplus \mathcal{G}'$. Let $\mathsf{Th}'_{\to^*} = \overline{\mathcal{R}}|_{\mathcal{U}_{\to^*}(\mathcal{R},\mathsf{F})}$. Processor P^{UR} is given by $\mathsf{P}^{\mathsf{UR}}(\tau) = \{(\{\mathsf{Th}'_{\to^*}\} \uplus \mathbb{T}', \{\mathsf{F}\}),\ (\mathbb{T}, \mathcal{G}')\}$.*

P^{UR} distributes the sequences in $\mathcal{G}$ in two new f-Problems: the first one consists of a goal with a single sequence F together with a refined version of $\mathbb{T}$ where $\overline{\mathcal{R}}$ is simplified into $\overline{\mathcal{R}}|_{\mathcal{U}_{\to^*}(\mathcal{R},\mathsf{F})}$; the second f-Problem consists of $\mathcal{G}'$ but keeps the original set of theories $\mathbb{T}$. By using [13, Proposition 4], we can see that the $(\{\overline{\mathcal{R}}|_{\mathcal{U}_{\to^*}(\mathcal{R},\mathcal{G})}\} \uplus \mathbb{T})$-feasibility of a sequence F implies its $(\{\overline{\mathcal{R}}\} \uplus \mathbb{T})$-feasibility. Similarly, $(\{\overline{\mathcal{R}}\} \uplus \mathbb{T})$-infeasibility of F can be proved as $(\{\overline{\mathcal{R}}|_{\mathcal{U}_{\to^*}(\mathcal{R},\mathsf{F})}\} \uplus \mathbb{T})$-infeasibility provided that all terms s in feasibility conditions $s \to^* t$ in F are ground. Thus, we have the following:

Theorem 7. *P^{UR} is sound. If for all $s \to^* t \in \mathsf{F}$, s is ground, then P^{UR} is complete.*

As discussed in the last paragraph of [13, Section 2], the groundness requirement cannot be dropped, in general (even for TRSs).

Example 14. For $\mathcal{R} = \{a \to b\}$, the sequence $\mathsf{F} = (x \to^* a, x \to^* b)$ is $\{\mathsf{Th}_{\to^*}\}$-feasible (just instantiate variable x to a and use the rule in $\mathcal{R}$). However, it is *not* $\{\mathsf{Th}'_{\to^*}\}$-feasible for $\mathsf{Th}'_{\to^*} = \overline{\mathcal{R}}|_{\mathcal{U}_{\to^*}(\mathcal{R},\mathsf{F})}$ because $\mathcal{U}(\mathcal{R}, x)$ is empty. Hence, $\mathcal{U}_{\to^*}(\mathcal{R}, \mathsf{F}) = \mathcal{U}(\mathcal{R}, x) \cup \mathcal{U}(\mathcal{R}, x)$ is also empty.

P^{UR} deals with many-step conditions $s \to^* t$ only. Furthermore, note that no change (simplification) in $\mathsf{Th}_{\to}$ (if used in $\mathbb{T}$) is introduced. For one-step conditions $s \to t$, we can use a similar (sound and complete) processor $\mathsf{P}^{\mathsf{UR1}}$ as follows. Let $\mathcal{U}_{\to}(\mathcal{R}, \mathsf{F}) = \bigcup_{s \to t \in \mathsf{F}} \mathcal{U}(\mathcal{R}, s)$ and $\overline{\mathcal{R}}|_{\mathcal{U}_{\to}(\mathcal{R},\mathsf{F})}$ be the first-order theory for $(\mathcal{F}, \mathcal{U}_{\to}(\mathcal{R}, \mathsf{F}))$.

Definition 11. *Let $\tau = (\mathbb{T}, \mathcal{G})$ be an f-Problem such that $\mathbb{T} = \{\mathsf{Th}_{\to}\} \uplus \mathbb{T}'$ with $\mathsf{Th}_{\to} = \overline{\mathcal{R}}$ for a CTRS $\mathcal{R}$ and $\mathcal{G} = \{\mathsf{F}\} \uplus \mathcal{G}'$. Let $\mathsf{Th}'_{\to} = \overline{\mathcal{R}}|_{\mathcal{U}_{\to}(\mathcal{R},\mathsf{F})}$. Processor $\mathsf{P}^{\mathsf{UR1}}$ is given by $\mathsf{P}^{\mathsf{UR1}}(\tau) = \{(\{\mathsf{Th}'_{\to}\} \uplus \mathbb{T}', \{\mathsf{F}\}),\ (\mathbb{T}, \mathcal{G}')\}$.*

Theorem 8. *$\mathsf{P}^{\mathsf{UR1}}$ is sound. If for all $s \to t \in \mathsf{F}$, s is ground, then $\mathsf{P}^{\mathsf{UR1}}$ is complete.*

Starting from the f-Problem $(\mathbb{T}, \mathcal{G})$, where $\mathcal{G} = \{\mathsf{F}\} \uplus \mathcal{G}'$, both P^{UR} and $\mathsf{P}^{\mathsf{UR1}}$ return two f-Problems $(\mathbb{T}_1, \mathcal{G}_1)$ and $(\mathbb{T}_2, \mathcal{G}_2)$. For both P^{UR} and $\mathsf{P}^{\mathsf{UR1}}$, we have $\mathcal{G}_1 = \{\mathsf{F}\}$, $\mathcal{G}_2 = \mathcal{G}'$, and $\mathbb{T}_2 = \mathbb{T}$. As for $\mathbb{T}_1$, P^{UR} changes the component $\mathsf{Th}_{\to^*}$ of $\mathbb{T}$. On the other hand, $\mathsf{P}^{\mathsf{UR1}}$ changes $\mathsf{Th}_{\to}$. With regard to the preservation property, which is relative to the goal and theory in a given f-Problem, whenever it holds for $(\mathbb{T}, \mathcal{G})$, it is not difficult to see (from the definition of preservation and usable rules) that it also remains true for $(\mathbb{T}_1, \mathcal{G}_1)$ and $(\mathbb{T}_2, \mathcal{G}_2)$.

4.5 Narrowing on Rewriting Conditions Processor

Reachability problems $\sigma(s) \to^* \sigma(t)$ are often investigated using *narrowing* and unification conditions directly over terms s and t, thus avoiding the 'generation' of the required substitution σ. In this section, we use narrowing to simplify feasibility conditions in $\mathcal{G}$. Definition 12 describes how narrowing is defined in the context of CTRSs. In the following, we write $s =^?_\theta t$ if s and t unify with *mgu* θ.

Definition 12. [16, Definition 79] *Let $\mathcal{R}$ be a CTRS. A term s* narrows *to a term t (written $s \rightsquigarrow_{\mathcal{R},\theta,p} t$ or just $s \rightsquigarrow_{\mathcal{R},\theta} t$ or even $s \rightsquigarrow t$), iff there is a nonvariable position $p \in \mathcal{P}os_{\mathcal{F}}(s)$, a* renamed *rule $\ell \to r \Leftarrow s_1 \to t_1, \ldots, s_n \to t_n$ in $\mathcal{R}$, substitutions $\theta_0, \ldots, \theta_n, \tau_1, \ldots, \tau_n$, and terms $t'_1, \ldots, t'_n$ such that:*

1. *$s|_p =^?_{\theta_0} \ell$,*
2. *for all i, $1 \leq i \leq n$, $\eta_{i-1}(s_i) \rightsquigarrow^*_{\mathcal{R},\theta_i} t'_i$ and $t'_i =^?_{\tau_i} \theta_i(\eta_{i-1}(t_i))$, where $\eta_0 = \theta_0$ and for all $i > 0$, $\eta_i = \tau_i \circ \theta_i \circ \eta_{i-1}$, and*
3. *$t = \theta(s[r]_p)$, where $\theta = \eta_n$.*

We write $u \rightsquigarrow^*_{\mathcal{R},\beta} v$ for terms u, v and substitution β iff there are terms $u_1, \ldots, u_{m+1}$ and substitutions $\beta_1, \ldots, \beta_m$ for some $m \geq 0$ such that

$$u = u_1 \rightsquigarrow_{\mathcal{R},\beta_1} u_2 \rightsquigarrow_{\mathcal{R},\beta_2} \cdots \rightsquigarrow_{\mathcal{R},\beta_m} u_{m+1} = v$$

and $\beta = \beta_m \circ \cdots \circ \beta_1$ (or $\beta = \varepsilon$ if $m = 0$).

Given a term u, the set $N_1(\mathcal{R}, u)$ represents the set of one-step $\mathcal{R}$-narrowings issued from u:

$$N_1(\mathcal{R}, u) = \{(v, \theta{\downarrow_{\mathcal{V}ar(u)}}) \mid u \rightsquigarrow_{\ell \to r \Leftarrow c,\theta} v, \ell \to r \Leftarrow c \in \mathcal{R}\} \tag{23}$$

where $\theta{\downarrow_{\mathcal{V}ar(u)}}$ is a substitution defined by $\theta{\downarrow_{\mathcal{V}ar(u)}}(x) = \theta(x)$ if $x \in \mathcal{V}ar(u)$ and $\theta{\downarrow_{\mathcal{V}ar(u)}}(x) = x$ otherwise.

As discussed in [16, Section 7.5], the set $N_1(\mathcal{R}, u)$ can be *infinite*. In [16, Proposition 87] some sufficient conditions for finiteness of $N_1(\mathcal{R}, u)$ are given. Knowing these restrictions, in Theorem 9 we define a narrowing processor on feasibility conditions.

Given a sequence $\mathsf{F}_k = (s_j \bowtie_j t_j)_{j=1}^n$ in a goal $\mathcal{G}$, and $1 \leq i \leq n$ such that $\bowtie_i = \to^*$, $\overline{\mathcal{N}}(\mathcal{R}, \mathcal{G}, k, i)$ returns a new set of feasibility goals where each element of the set corresponds to a possible narrowing on the condition i:

$$\overline{\mathcal{N}}(\mathcal{R}, \mathcal{G}, k, i) = \{\mathcal{G}[\mathsf{F}_k[\boldsymbol{\theta}, w \to^* t_i]_i]_k \mid s_i \to^* t_i \in \mathsf{F}_k, (w, \theta) \in N_1(\mathcal{R}, s_i)\}$$

where $\boldsymbol{\theta}$ consists of new conditions $x_1 \to^* \theta(x_1), \ldots, x_m \to^* \theta(x_m)$ obtained from the bindings in θ for variables in $\mathcal{V}ar(s_i) = \{x_1, \ldots, x_m\}$.

Definition 13 (Narrowing on f-Conditions Processor). *Let $\tau = (\{\overline{\mathcal{R}}\}, \mathcal{G})$ be an f-Problem, $s_i \to^* t_i \in \mathsf{F}_k$ for some F_k in $\mathcal{G}$, and $\mathcal{N} \subseteq \overline{\mathcal{N}}(\mathcal{R}, \mathcal{G}, k, i)$ finite.* P^{NC} *is given by $\mathsf{P}^{\mathsf{NC}}(\tau) = \{(\{\overline{\mathcal{R}}\}, \mathcal{G}') \mid \mathcal{G}' \in \mathcal{N}\}$.*

Given a term s, we let $NRules(\mathcal{R}, s)$ be the set of rules $\alpha : \ell \to r \Leftarrow c \in \mathcal{R}$ such that a nonvariable subterm t of s is a *narrex* of α,[4] and, given a substitution θ, we denote as $\theta{\downarrow_{\mathcal{V}ar(s)}}$ the substitution defined by $\theta{\downarrow_{\mathcal{V}ar(s)}}(x) = \theta(x)$ if $x \in \mathcal{V}ar(s)$ and $\theta{\downarrow_{\mathcal{V}ar(s)}}(x) = x$ otherwise.

Theorem 9. P^{NC} *is sound. If $\mathcal{N} = \overline{\mathcal{N}}(\mathcal{R}, \mathcal{G}, k, i)$ and $s_i \to^* t_i \in \mathsf{F}_k$ is such that s_i and t_i do* not *unify and either s_i is* ground *and $\mathcal{R}$ is a 2-CTRS or (1) $NRules(\mathcal{R}, s_i)$ is a TRS, (2) s_i is linear, and (3) $\mathcal{V}ar(s_i) \cap \mathcal{V}ar(t_i) = \emptyset$, then* P^{NC} *is complete.*[5]

[4] Given a CTRS $\mathcal{S}$, a non-variable term t is a *narrowing redex* (or a *narrex*, for short) of a rule $\ell \to r \Leftarrow s_1 \approx t_1, \ldots, s_n \approx t_n \in \mathcal{S}$ if t and ℓ unify with *mgu* θ (we assume $\mathcal{V}ar(t) \cap \mathcal{V}ar(\ell) = \emptyset$). However, if $(\theta(s_1) \approx \theta(t_1), \ldots, \theta(s_n) \approx \theta(t_n))$ can be proved $\{\overline{\mathcal{S}}\}$-infeasible, we can discard t, as no narrowing step is possible on it. In our current implementation, though, only the unification test is used.

[5] This processor is inspired by the processor defined in [17, Section 4.1]. A justification for the completeness conditions can be obtained from [17, Examples 18 and 19].

$$
\begin{aligned}
add(0,x) &\to x\\
add(s(x),y) &\to s(add(x,y))\\
div(0,s(x)) &\to 0\\
div(s(x),s(y)) &\to 0 \Leftarrow lte(s(x),y) \approx true\\
div(s(x),s(y)) &\to s(q) \Leftarrow lte(s(x),y) \approx false, div(minus(x,y),s(y)) \approx q\\
lte(0,y) &\to true\\
lte(s(x),0) &\to false\\
lte(s(x),s(y)) &\to lte(x,y)\\
minus(0,s(y)) &\to 0\\
minus(s(x),s(y)) &\to minus(x,y)\\
minus(x,0) &\to x\\
mod(0,y) &\to 0\\
mod(x,0) &\to x\\
mod(x,s(y)) &\to mod(minus(x,s(y)),s(y)) \Leftarrow lte(s(y),x) \approx true\\
mod(x,s(y)) &\to x \Leftarrow lte(s(y),x) \approx false\\
mult(0,y) &\to 0\\
mult(s(x),y) &\to add(mult(x,y),y)\\
power(x,0) &\to s(0)\\
power(x,n) &\to mult(mult(y,y),s(0)) \Leftarrow n \approx s(n'),\\
&\qquad mod(n,s(s(0))) \approx 0, power(x,div(n,s(s(0)))) \approx y\\
power(x,n) &\to mult(mult(y,y),x) \Leftarrow n \approx s(n'),\\
&\qquad mod(n,s(s(0))) \approx s(z), power(x,div(n,s(s(0)))) \approx y
\end{aligned}
$$

Fig. 3. CTRS 529.trs in COPS

Even with $\overline{\mathcal{N}}(\mathcal{R},\mathcal{G},k,i)$ infinite, a subset $\mathcal{N}$ of $\overline{\mathcal{N}}(\mathcal{R},\mathcal{G},k,i)$ can be sufficient to prove feasibility. However, to prove infeasibility we need to consider all possible narrowings.

Example 15. Consider the CTRS $\mathcal{R}$ in Figure 3, $\mathcal{G} = \{(lte(s(x),0) \to^* true)\}$, and $\tau_I = (\{\overline{\mathcal{R}}\},\mathcal{G})$. Since $NRules(\mathcal{R}, lte(s(x),0))$ contains the *lte* rules only, $\overline{\mathcal{N}}(\mathcal{R},\mathcal{G},1,1) = \{(x \to^* x, false \to^* true)\}$. Therefore, $\mathsf{P}^{\mathsf{NC}}(\tau_I) = \{\tau_1\}$ with $\tau_1 = (\{\overline{\mathcal{R}}\}, (x \to^* x, false \to^* true))$. Since $NRules(\mathcal{R}, lte(s(x),0))$ is a TRS, $lte(s(x),0)$ is linear, and $\mathcal{V}ar(lte(s(x),0)) \cap \mathcal{V}ar(true) = \emptyset$, P^{NC} is complete. Now, we apply $\mathsf{P}^{\mathsf{Sat}}$ to τ_1 to obtain $\mathsf{P}^{\mathsf{Sat}}(\tau_1) = \emptyset$ by using Mace4. The obtained FP tree is in displayed in Figure 4.

5 Implementation and Web Interface

infChecker 1.0 is written in Haskell and consists of 30 Haskell modules with more than 4500 lines of code. The tool can be used through its web interface here:

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

The input format is an extended version of the Confluence Competition (CoCo) format [20], which is the official format used in the *infeasibility* (INF) category.[6] The input has two components:

[6] See `http://project-coco.uibk.ac.at/2019/categories/infeasibility.php`

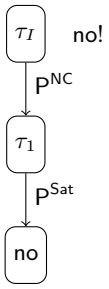

Fig. 4. Proof tree obtained from Example 15

1. A CTRS $\mathcal{R}$ in TPDB format[7] which can specify a *replacement map* μ for *context-sensitive rewriting* (*CSR* [10]) establishing the arguments $\mu(f) \subseteq \{1, \ldots, k\}$ which *can be rewritten* for each k-ary symbol f. This is top-down propagated to *positions* of terms, which are then called *active*. We write $s \hookrightarrow t$ if an *active* subterm of s can be rewritten so that $s \to t$. Then, $\to^*$, $\downarrow$, $\leftrightarrow$, etc. are generalized to *CSR* as $\hookrightarrow^*$, $\downarrow$, $\leftrightarrow$, etc., by using $\hookrightarrow$ instead of $\to$.
2. An f-goal built using the set

   $\mathbb{P}_{\mathsf{iCh}} = \{\texttt{|>}, \texttt{|>=}\} \cup \{\texttt{->}, \texttt{->*}, \texttt{-><-}, \texttt{<-->}, (\texttt{<-->*})\} \cup \{\texttt{\->}, \texttt{\->*}, \texttt{\-><-/}, \texttt{<-/\->}, \texttt{<-/\->*}\} \cup \{\texttt{==}\}$

   of (binary) predicates for (strict) subterm (`|>` and `|>=`), one or many rewriting steps (`->` and `->*`), joinability (`-><-`), symmetric closure of `->` (`<-->`), conversion (`<-->*`) and their context-sensitive versions `\->`, `\->*`, `\-><-/`, `<-/\->`, and `<-/\->*`.

Theories $\mathsf{Th}_{\bowtie}$ for each $\bowtie \in \mathbb{P}_{\mathsf{iCh}}$ are automatically obtained from the components of $\mathcal{R}$ (signature, replacement map, conditional rules). Symbol `==` is borrowed from the COPS syntax, where it is used to specify the conditional part of rewrite rules (see $\approx$ in Figure 1). Both in the conditional part of rules and in f-goals, its meaning depends on the `CONDITIONTYPE` section of the input specifying how the conditions of rules are evaluated [22, Definition 7.1.3] according to:

`CONDITIONTYPE`	replace `==` by
`ORIENTED`	`->*`
`JOIN`	`-><-`
`SEMI-EQUATIONAL`	`<-->*`

In this respect, when using $\hookrightarrow$ or $\hookrightarrow^*$ (i.e., `\->` or `\->*`) in f-goals, the associated theories $\mathsf{Th}_{\hookrightarrow}$ and $\mathsf{Th}_{\hookrightarrow^*}$ are those obtained by evaluating the conditions $s_i \approx t_i$ in rules using $\hookrightarrow^*$, $\downarrow$, or $\leftrightarrow^*$, depending on the label `ORIENTED`, `JOIN`, or `SEMI-EQUATIONAL` specified in `CONDITIONTYPE`. If no replacement map has been specified (i.e., no `STRATEGY` section with `CONTEXTSENSITIVE` label is given, the *trivial* replacement map $\mu_\top(f) = \{1, \ldots, k\}$, establishing no replacement restrictions, is automatically assumed for each k-ary symbol f.

[7] See `http://zenon.dsic.upv.es/muterm/?page_id=31`

When the problem is introduced, a model generator for infeasibility (`AGES`, `Mace4` or `Automatic`) can be selected. Then, pressing button `Prove automatically` initiates procedure to check whether the problem is feasible or infeasible in the given timeout.

Currently, infChecker implements the construction of the FP tree in Definition 5 with the processors presented in Section 4 by depth-first generation of the nodes and orderly attempting the following sequence of processors to develop each node:[8] $\mathsf{P}^{\mathsf{Spl}}$, $\mathsf{P}^{\mathsf{Prov}}$, $\mathsf{P}^{\mathsf{Sat}}$, and P^{NC}. If the final answer is YES or NO, the tool displays a report in plain text. Otherwise, MAYBE is returned.

6 Experimental Evaluation

We participated in the INF category of the 2019 Confluence Competition (CoCo),[9] with a limit of 60 seconds to return a proof of feasibility or infeasibility (or a *don't know* answer). infChecker obtained the following results:

INF Tool	Yes	No	Total
infChecker	40	32	72
ConCon	31	0	31
nonreach	30	0	30
Moca	26	0	26
maedmax	15	0	15
CO3	12	0	12

Answers *Yes/No* in the table refer to *infeasibility* (which is the focus of the competition). In our setting, given a CTRS $\mathcal{R}$ and an infeasibility problem given as a feasibility sequence $\mathcal{G}$, we just return "*Yes*" if τ_I is proved infeasible, and "*No*" if τ_I is proved feasible. Apart from the 32 "*No*" answers, there are 7 more examples that can be proved positively ("*Yes*") using infChecker only. There also are 10 examples that can be proved by other tools and cannot be proved by infChecker.

In the experiments P^{UR} was used 11 times and P^{NC} was used twice. We required a combination of processors in 13 examples: the sum of uses of P^{UR} and P^{NC}. Being unable to provide a definite (YES/NO) answer, their use always requires another processor to finish the proof. According to the strategy described at the end of Section 5, such a combination is necessary. Thus, we need both P^{UR} and P^{NC} to solve the examples.

7 Related Work

The notion of (in)feasibility of a logic formula has been investigated in [12, Section 4.1], in the context of the analysis of operational termination of programs in general logics [15]. A satisfiability approach to prove infeasibility of first-order formulas with respect to an order-sorted first-order theory [4] is described in [12, Section 4.1.1]. No attempt to

[8] We use a Haskell library for parallelism. However, due to the *Breadth First Search* evaluation strategy of the library, in a parallel execution $\mathsf{P}_1 \parallel \mathsf{P}_2 \parallel \cdots \parallel \mathsf{P}_n$ of several processors we wait until the leftmost processor (P_1) is completely evaluated (returns a solution or reaches a timeout) before continuing with $\mathsf{P}_2 \parallel \cdots \parallel \mathsf{P}_n$. Thus, there is a kind of 'restricted' parallelism in our implementation.

[9] http://project-coco.uibk.ac.at/2019/

decompose such proofs by taking into account the structure of the logic formula (as done our feasibility framework) is made. No technique for proving feasibility is proposed. Actually, our feasibility framework could be advantageously used to implement proofs of operational termination of programs in general logics.

Sternagel and Yamada [25] define a framework to prove reachability constraints ϕ for TRSs $\mathcal{R}$ as first-order formulas where only reachability atoms $s \twoheadrightarrow t$ (instead of $s \to^* t$) are allowed. As remarked in [25, footnote 1], negation and universal quantification are not considered, i.e., only ECBCAs with atoms $s \twoheadrightarrow t$ are (ultimately) considered. A constraint $s \twoheadrightarrow t$ is satisfied by a substitution σ with respect to $\mathcal{R}$ if $\sigma(s) \to^*_{\mathcal{R}} \sigma(t)$. Reachability constraints ϕ are called satisfiable if there is a substitution σ such that $\sigma(\phi)$[10] is satisfied in the usual first-order logic sense. Our approach is more flexible as more predicates can be defined by appropriate theories (including CTRSs). For instance, *non-root reachability* constraints $s \overset{>\Lambda}{\twoheadrightarrow} t$ (given in [25, Section 4] in terms of reachability constraints) can be defined by $\mathsf{Th}_{\overset{>\Lambda}{\to}*}$ consisting of

$$(\forall x)\ \ x \overset{>\Lambda}{\longrightarrow}{}^* x \tag{24}$$

$$(\forall x)(\forall y)(\forall z)\ \ x \overset{>\Lambda}{\to} y \wedge y \overset{>\Lambda}{\longrightarrow}{}^* z \Rightarrow x \overset{>\Lambda}{\longrightarrow}{}^* z \tag{25}$$

plus sentences $\overline{(\mathrm{Rl})_\alpha}$ for each rewrite rule α, sentences $\overline{(\mathrm{C})_{f,i}}$ for each k-ary symbol f and $1 \le i \le k$, and $(\forall \boldsymbol{x})(\forall y_i)x_i \to y_i \Rightarrow f(x_1,\ldots,x_i,\ldots,x_k) \overset{>\Lambda}{\to} f(x_1,\ldots,y_i,\ldots,x_k)$. We could also cover CTRSs by also adding $\overline{(\mathrm{T})}$, for the transitivity rule (T), necessary for the evaluation of the conditional part of conditional rules α (which may require root steps). Then, the non-reachability problems considered in [25, Section 4] for TRSs $\mathcal{R}$ could be treated in our framework using $\mathbb{P} = \{\to, \to^*, \overset{>\Lambda}{\longrightarrow}{}^*\}$ and $\mathbb{T} = \{\mathsf{Th}_\to, \mathsf{Th}_{\to^*}, \mathsf{Th}_{\overset{>\Lambda}{\to}*}\}$, where $\mathsf{Th}_\to = \mathsf{Th}_{\to^*} = \overline{\mathcal{R}}$. Actually, we can 'import' the decomposition treatment for non-root reachability goals in [25, Definition 5] as a transformation processor (like P^{NC}) specific for non-root reachability conditions of TRSs as follows: let $\tau = (\mathbb{T}, \mathcal{G})$ and $\mathsf{F}_i \in \mathcal{G}$ be such that $\mathsf{F}_i = (\gamma_1,\ldots,\gamma_j,\ldots,\gamma_n)$ with $\gamma_j = f(s_1,\ldots,s_k) \overset{>\Lambda}{\longrightarrow}{}^* f(t_1,\ldots,t_k) \in \mathsf{F}_i$ for some terms s_i, t_i, $1 \le i \le k$. Then,

$$\mathsf{P}^{\mathsf{non\text{-}root\text{-}r}}(\tau) = \{(\mathbb{T}, \mathcal{G}[\mathsf{F}'_i]_i)\}$$

where $\mathsf{F}'_i = (\gamma_1,\ldots,\gamma_{j-1}, s_1 \to^* t_1,\ldots,s_k \to^* t_k \ldots,\gamma_{j+1},\ldots,\gamma_n)$ and $\mathcal{G}[\mathsf{F}'_i]_i$ is the goal obtained by replacing the i-th sequence of $\mathcal{G}$ by F'_i. This example also shows how techniques developed elsewhere could be smoothly integrated in our framework.

Decidability of reachability problems for *CSR* in TRSs (i.e., does $s \hookrightarrow^* t$ hold?) has been investigated using tree-automata techniques [1,6,7,8]. infChecker is able to (try to) prove and disprove reachability conditions as f-goals using predicate `\->*` without any specific restriction on the TRSs (e.g., left-linearity), as done in these papers.

Regarding the automation of proofs of infeasibility in (conditional) rewriting, 2019 was the first year the infeasibility category was included in the International Confluence Competition. The new category had a good reception, with 6 participating tools (summary of results in Section 6), and provided a good picture of the state of the art:

- CO3 tries to prove confluence and if it fails linearizes the condition and tries to compute a narrowing tree for the linearized condition. The applicability of narrowing trees in this context is restricted to *syntactically deterministic conditional*

[10] obtained by (i) *renaming* all bounded variables in ϕ using variables not occurring in bindings of σ to obtain ϕ', and then (ii) *replacing* each free variable x of ϕ' by $\sigma(x)$.

term rewriting systems (right-hand sides of conditions must be constructor terms or ground normal forms) that are constructor systems [21].
- ConCon uses a variety of methods to check for infeasibility of conditional critical pairs, ranging from a simple technique based on unification, via symbol transition graph analysis, reachability problem decomposition, the exploitation of certain equalities in the conditions, and tree automata completion to equational reasoning [24].
- Moca implements *maximal ordered completion* similar to maedmax [26] together with some *approximation techniques* not yet published.
- maedmax implements *maximal order completion* [26].
- nonreach uses two approaches: *transformations* based on decomposition and narrowing and *nonreachability checks* based on unification, symbol transition graphs, equational reasoning and tree automata completion [19,25].

Thanks to representing CTRSs $\mathcal{R}$ as first-order theories $\overline{\mathcal{R}}$, infChecker was not only the most successful tool for checking infeasibility, but also the only tool currently able to *disprove* it (by proving feasibility). There also is room, however, for improvements, as witnessed by the 10 examples mentioned in the summary of experiments in Section 6. In order to deal with these examples (not handled by infChecker), nonreach uses *narrowing* and Moca uses satisfiability with LPO. Regarding ConCon, it is unclear from the report provided by the tool, which specific technique was used to solve the examples.

8 Conclusions and Future Work

We have extended and generalized the notion of *feasibility sequence* introduced in [13] by considering goals which are sets of sequences of conditions $s \bowtie t$ for arbitrary predicates $\bowtie$. Each predicate symbol $\bowtie$ is 'defined' by a first-order theory $\mathsf{Th}_{\bowtie}$. Such conditions, sequences, and goals have a precise logical characterization as ECBCAs, and its feasibility can be investigated as provability of such formulas (Theorem 1). We have shown some examples of properties (of CTRSs) which can be proved by using this approach. We have introduced a framework for proving and disproving feasibility of such goals. To the best of our knowledge, our logic-based notion of feasibility goal and the framework to prove and disprove them are new in the literature.

We have developed a new tool implementing our framework: infChecker. Currently, infChecker provides a first implementation of the framework introduced in this paper (restricted to CTRSs, but extended with context-sensitivity, subterm, etc.), and supports predicates like $\to$ (one-step rewriting), $\to^*$ (many-step rewriting), $\downarrow$ (joinability), $\leftrightarrow^*$ (conversion), and the analogous for context-sensitive rewriting. We also give support to $\trianglerighteq$ (subterm) and $\triangleright$ (strict subterm). As far as we know, infChecker is the first tool dealing with (in)feasibility problems supporting this set of predicates. Also, the use of provability/satisfiability techniques in proofs of (in)feasibility seems to be new in the literature. We participated in the 2019 Confluence Competition [20] in the INF (infeasibility) category, being the most powerful tool for checking both infeasibility and feasibility. In the near future, we plan to extend infChecker to provide full support to our framework, by allowing the explicit definition of (not necessarily binary) predicates and independent first-order theories associated to such predicates besides the built-in set of predicates $\mathbb{P}_{\mathsf{iCh}}$ and associated theories which are available now.

Acknowledgments We thank the anonymous referees for many remarks and suggestions that led to improve the paper.

References

1. Andrianarivelo, N., Réty, P.: Over-approximating terms reachable by context-sensitive rewriting. In: Boja'nczyk, M., Lasota, S., Potapov, I. (eds.) Reachability Problems - 9th International Workshop, RP 2015, Warsaw, Poland, September 21-23, 2015, Proceedings. Lecture Notes in Computer Science, vol. 9328, pp. 128–139. Springer (2015). https://doi.org/10.1007/978-3-319-24537-9_12
2. Dershowitz, N.: Termination of rewriting. J. Symb. Comput. **3**(1/2), 69–116 (1987). https://doi.org/10.1016/S0747-7171(87)80022-6, `https://doi.org/10.1016/S0747-7171(87)80022-6`
3. Giesl, J., Thiemann, R., Schneider-Kamp, P., Falke, S.: Mechanizing and improving dependency pairs. J. Autom. Reasoning **37**(3), 155–203 (2006). https://doi.org/10.1007/s10817-006-9057-7
4. Goguen, J.A., Meseguer, J.: Models and equality for logical programming. In: Ehrig, H., Kowalski, R.A., Levi, G., Montanari, U. (eds.) TAPSOFT'87: Proceedings of the International Joint Conference on Theory and Practice of Software Development, Pisa, Italy, March 23-27, 1987, Volume 2: Advanced Seminar on Foundations of Innovative Software Development II and Colloquium on Functional and Logic Programming and Specifications (CFLP). Lecture Notes in Computer Science, vol. 250, pp. 1–22. Springer (1987). https://doi.org/10.1007/BFb0014969
5. Gutiérrez, R., Lucas, S.: Automatic generation of logical models with AGES. In: Fontaine, P. (ed.) Automated Deduction - CADE 27 - 27th International Conference on Automated Deduction, Natal, Brazil, August 27-30, 2019, Proceedings. Lecture Notes in Computer Science, vol. 11716, pp. 287–299. Springer (2019). https://doi.org/10.1007/978-3-030-29436-6_17
6. Kojima, Y., Sakai, M.: Innermost reachability and context sensitive reachability properties are decidable for linear right-shallow term rewriting systems. In: Voronkov, A. (ed.) RTA. Lecture Notes in Computer Science, vol. 5117, pp. 187–201. Springer (2008). https://doi.org/10.1007/978-3-540-70590-1_13
7. Kojima, Y., Sakai, M., Nishida, N., Kusakari, K., Sakabe, T.: Context-sensitive innermost reachability is decidable for linear right-shallow term rewriting systems. Information and Media Technologies **4**(4), 802–814 (2009)
8. Kojima, Y., Sakai, M., Nishida, N., Kusakari, K., Sakabe, T.: Decidability of reachability for right-shallow context-sensitive term rewriting systems. IPSJ Online Transactions **4**, 192–216 (2011)
9. Lucas, S.: Analysis of Rewriting-Based Systems as First-Order Theories. In: Fioravanti, F., Gallagher, J.P. (eds.) LOPSTR 2017: Logic-Based Program Synthesis and Transformation. LNCS, vol. 10855, pp. 180–197 (2018). https://doi.org/10.1007/978-3-319-94460-9_11
10. Lucas, S.: Context-sensitive computations in functional and functional logic programs. J. Funct. Log. Program. **1998**(1) (1998), `http://danae.uni-muenster.de/lehre/kuchen/JFLP/articles/1998/A98-01/A98-01.html`
11. Lucas, S.: Proving semantic properties as first-order satisfiability. Artif. Intell. **277** (2019). https://doi.org/10.1016/j.artint.2019.103174
12. Lucas, S.: Using well-founded relations for proving operational termination. J. Autom. Reasoning **64**(2), 167–195 (2020). https://doi.org/10.1007/s10817-019-09514-2
13. Lucas, S., Gutiérrez, R.: Use of logical models for proving infeasibility in term rewriting. Inf. Process. Lett. **136**, 90–95 (2018). https://doi.org/10.1016/j.ipl.2018.04.002

14. Lucas, S., Marché, C., Meseguer, J.: Operational termination of conditional term rewriting systems. Inf. Process. Lett. **95**(4), 446–453 (2005). https://doi.org/10.1016/j.ipl.2005.05.002
15. Lucas, S., Meseguer, J.: Proving operational termination of declarative programs in general logics. In: Chitil, O., King, A., Danvy, O. (eds.) Proceedings of the 16th International Symposium on Principles and Practice of Declarative Programming, Kent, Canterbury, United Kingdom, September 8-10, 2014. pp. 111–122. ACM (2014). https://doi.org/10.1145/2643135.2643152
16. Lucas, S., Meseguer, J., Gutiérrez, R.: The 2D Dependency Pair Framework for conditional rewrite systems. Part I: Definition and basic processors. J. Comput. Syst. Sci. **96**, 74–106 (2018). https://doi.org/10.1016/j.jcss.2018.04.002
17. Lucas, S., Meseguer, J., Gutiérrez, R.: The 2D Dependency Pair Framework for Conditional Rewrite Systems—Part II: Advanced Processors and Implementation Techniques. Journal of Automated Reasoning **in press** (2020). https://doi.org/10.1007/s10817-020-09542-3
18. McCune, W.: Prover9 and Mace4. [online], available at `https://www.cs.unm.edu/~mccune/mace4/`
19. Meßner, F., Sternagel, C.: nonreach – A Tool for Nonreachability Analysis. In: Vojnar, T., Zhang, L. (eds.) Proc. of Tools and Algorithms for the Construction and Analysis of Systems, TACAS'19. pp. 337–343. Springer (2019). https://doi.org/10.1007/978-3-030-17462-0_19
20. Middeldorp, A., Nagele, J., Shintani, K.: Confluence Competition 2019. In: Beyer, D., Huisman, M., Kordon, F., Steffen, B. (eds.) Proc. of Tools and Algorithms for the Construction and Analysis of Systems, TACAS'19. pp. 25–40. Springer (2019). https://doi.org/10.1007/978-3-030-17502-3_2
21. Nishida, N., Maeda, Y.: Narrowing Trees for Syntactically Deterministic Conditional Term Rewriting Systems. In: Kirchner, H. (ed.) Proc. on 3rd International Conference on Formal Structures for Computation and Deduction, FSCD'18. Leibniz International Proceedings in Informatics (LIPIcs), vol. 108, pp. 26:1–26:20. Schloss Dagstuhl–Leibniz-Zentrum fuer Informatik (2018). https://doi.org/10.4230/LIPIcs.FSCD.2018.26
22. Ohlebusch, E.: Advanced topics in term rewriting. Springer (2002), `http://www.springer.com/computer/swe/book/978-0-387-95250-5`
23. Prawitz, D.: Natural Deduction. A proof-theoretical study. Dover (2006)
24. Sternagel, C., Sternagel, T., Middeldorp, A.: CoCo 2018 Participant: ConCon 1.5. In: Felgenhauer, B., Simonsen, J. (eds.) Proc. of the 7th International Workshop on Confluence, IWC'18. p. 66 (2018), `http://cl-informatik.uibk.ac.at/events/iwc-2018/`
25. Sternagel, C., Yamada, A.: Reachability Analysis for Termination and Confluence of Rewriting. In: Vojnar, T., Zhang, L. (eds.) Tools and Algorithms for the Construction and Analysis of Systems. pp. 262–278. Springer (2019). https://doi.org/10.1007/978-3-030-17462-0_15
26. Winkler, S., Moser, G.: MædMax: A Maximal Ordered Completion Tool. In: Galmiche, D., Schulz, S., Sebastiani, R. (eds.) IJCAR 2018: Automated Reasoning. pp. 472–480. Springer (2018). https://doi.org/10.1007/978-3-319-94205-6_31