Joint Proceedings of HOR 2019 and IWC 2019 (with system descriptions from CoCo 2019)

Contents

Preface

This report contains the joint proceedings of the 10th Workshop on Higher-Order Rewriting (HOR) and 7th International Workshop on Confluence (IWC), held in Dortmund, Germany on June 28th, 2019. In addition, the proceedings include the system descriptions of the 8th Confluence Competition (CoCo 2019). The workshops were part of the International Conference on Formal Structures for Computation and Deduction (FSCD 2019).

HOR is a forum to present work concerning all aspects of higher-order rewriting. The aim is to provide an informal and friendly setting to discuss recent work and work in progress concerning higher-order rewriting, broadly construed. This includes various topics of interest that range from foundations (pattern matching, unification, strategies, narrowing, termination, syntactic properties, type theory), frameworks (term rewriting, conditional rewriting, graph rewriting, net rewriting), semantics (operational semantics, denotational semantics, separability, higher-order abstract syntax) to implementation (graphs, nets, abstract machines, explicit substitution, rewriting tools, compilation techniques) and application (proof checking, theorem proving, generic programming, declarative programming, program transformation, certification).

Confluence provides a general notion of determinism and has been conceived as one of the central properties of rewriting. Confluence relates to the many topics of rewriting (completion, termination, commutation, coherence, etc.) and has been investigated in many formalisms of rewriting such as first-order rewriting, lambda-calculi, higher-order rewriting, higher-dimensional rewriting, constrained rewriting, conditional rewriting, etc. Recently, there is a renewed interest in confluence research, resulting in new techniques, tool support, certification, as well as applications. IWC promotes and stimulates research and collaboration on confluence and related properties.

The joint program contains 5 contributed talks as well as invited talks by Martin Avanzini, Fransico Duran, Jörg Endrullis, Cynthia Kop, and Giulio Manzonetto. In addition, the program contains the system descriptions from the 8th Confluence Competition (CoCo 2019) held in conjunction with the International Conference on Tools and Algorithms for the Construction and Analysis of Systems (TACAS 2019).

Many people contributed to the preparations of HOR and IWC. Hard work by the program commitees, steering committees, and subreviewers made an exciting program of contributed and invited talks possible. In addition, we are greatful to the organizing committee and workshop chairs of FSCD for hosting the workshops in Dortmund.

June 14, 2019 Brasilia/Novi Sad/Copenhagen
Mauricio Ayala-Rincón
Silvia Ghilezan
Jakob Grue Simonsen

HOR Program Committee

Silvia Ghilezan (chair)	University of Novi Sad, Serbia
Stefano Guerrini	Université Paris 13, France
Masahito Hasegawa	Kyoto University, Japan
Cynthia Kop	Radboud University, The Netherlands
Pierre Lescanne	École Normale Superieure de Lyon, France
Julian Nagele	Queen Mary University of London, United Kingdom
Vincent van Oostrom	University of Innsbruck, Austria

HOR Steering committee

Delia Kesner	Université Paris 7, France
Femke van Raamsdonk	Vrije Universiteit Amsterdam, The Netherlands

IWC Program Committee

Sandra Alves	University of Porto, Protugal
Mauricio Ayala-Rincón (co-chair)	University of Brasilia, Brazil
Cyrille Chenavier	INRIA Lille, France
Alejandro Díaz-Caro	University of Quilmes and ICC/UBA-CONICET, Argentina
Jörg Endrullis	Free University of Amsterdam, The Netherlands
Jakob Grue Simonsen (co-chair)	University of Copenhagen, Denmark
Raúl Guttiérez	Technical University of València, Spain
Camilo Rocha	Pontifical Xavierian University, Colombia
Masahiko Sakai	University of Nagoya, Japan
Sarah Winkler	University of Innsbruck, Austria

IWC subreviewer: Guido Martinez.

IWC Steering Committee

Takahito Aoto	Niigata University, Japan
Nao Hirokawa	Japan Advanced Institute of Science and Technology, Japan

infChecker
A Tool for Checking Infeasibility*

Raúl Gutiérrez[1]†and Salvador Lucas[1]

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

Given a Conditional Term Rewriting System (CTRS) $\mathcal{R}$ and terms s and t, we say that the reachability condition $s \to^* t$ is *feasible* if there is a substitution σ instantiating the variables in s and t such that the *reachability test* $\sigma(s) \to^*_{\mathcal{R}} \sigma(t)$ succeeds; otherwise, we call it *infeasible*. Checking infeasibility of such (sequences of) reachability conditions is important in the analysis of computational properties of CTRSs, like confluence or operational termination. Recently, a logic-based approach to prove and disprove infeasibility has been introduced. In this paper we present infChecker, a new tool for checking infeasibility which is based on such an approach.

1 Introduction

When analyzing the computational behaviour of CTRSs $\mathcal{R}$, consisting of rules $\ell \to r \Leftarrow s_1 \approx t_1, \ldots, s_n \approx t_n$, we need to consider two kinds of computations: (1) the reduction of expressions in the usual way, i.e., by replacing an instance $\sigma(\ell)$ of the left-hand side ℓ by the instance $\sigma(r)$ of the right-hand side r using a matching substitution σ and (2) the evaluation of the conditions $s_i \approx t_i$ in the rules, which (for *oriented* CTRSs) are treated as reachability tests $\sigma(s_i) \to^* \sigma(t_i)$. In this setting, representing rewriting steps in CTRSs as proofs of goals in the logic of (oriented) CTRSs with inference system in Figure 1 becomes a natural way to represent computations [6]. Given a CTRS $\mathcal{R}$, an inference system $\mathcal{I}(\mathcal{R})$ is obtained from the inference rules in Figure 1 by *specializing* $(C)_{f,i}$ for each k-ary symbol f in the signature $\mathcal{F}$ and $1 \leq i \leq k$ and $(Rl)_\rho$ for all conditional rules $\rho : \ell \to r \Leftarrow c$ in $\mathcal{R}$. 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 the inference system $\mathcal{I}(\mathcal{R})$, whose rules are *schematic* in the sense that 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.

$$\text{(R)} \quad \frac{}{x \to^* x} \qquad\qquad \text{(T)} \quad \frac{x \to y \qquad y \to^* z}{x \to^* z}$$

$$\text{(C)}_{f,i} \quad \frac{x_i \to y_i}{f(x_1, \ldots, x_i, \ldots, x_k) \to f(x_1, \ldots, y_i, \ldots, x_k)} \qquad\qquad \text{(Rl)}_\rho \quad \frac{s_1 \to^* t_1 \quad \cdots \quad s_n \to^* t_n}{\ell \to r}$$

$$\text{for all } f \in \mathcal{F}^{(k)} \text{ and } 1 \leq i \leq k \qquad\qquad \text{for } \rho : \ell \to r \Leftarrow s_1 \approx t_1, \ldots, s_n \approx t_n \in \mathcal{R}$$

Figure 1: Inference rules for conditional rewriting with an oriented CTRS $\mathcal{R}$ with signature $\mathcal{F}$

*Partially supported by the EU (FEDER), and projects RTI2018-094403-B-C32, PROMETEO/2019/098, and SP20180225.

†Raúl Gutiérrez was also supported by INCIBE program "Ayudas para la excelencia de los equipos de investigación avanzada en ciberseguridad".

A first-order theory $\overline{\mathcal{R}}$ associated to $\mathcal{R}$, where $\rightarrow$ and $\rightarrow^*$ are seen as predicates, is obtained from $\mathcal{I}(\mathcal{R})$: the inference rules $\frac{B_1 \cdots B_n}{A}$ in $\mathcal{I}(\mathcal{R})$ are considered as *sentences* $(\forall x_1,\ldots,x_m)\, B_1 \wedge \cdots \wedge B_n \Rightarrow A$, where $\{x_1,\ldots,x_m\}$ is the (possibly empty) set of variables occurring in the atoms $B_1,\ldots,B_n$ and A. If such a set is empty, we write $B_1 \wedge \cdots \wedge B_n \Rightarrow A$.

Example 1. *Consider the following CTRS $\mathcal{R}$ (903.trs[1]):*

$$\begin{array}{rcl@{\qquad}rcl}
le(0,s(y)) & \rightarrow & true & min(cons(x,nil)) & \rightarrow & x\\
le(s(x),s(y)) & \rightarrow & le(x,y) & min(cons(x,xs)) & \rightarrow & x \Leftarrow min(xs) \approx y, le(x,y) \approx true\\
le(x,0) & \rightarrow & false & min(cons(x,xs)) & \rightarrow & y \Leftarrow min(xs) \approx y, le(x,y) \approx false
\end{array}$$

The first-order theory $\overline{\mathcal{R}}$ for $\mathcal{R}$ is:

$$\begin{array}{rr@{\qquad}rr}
(\forall x)\ x \rightarrow^* x & (1) & (\forall x,y,z)\ x \rightarrow y \Rightarrow le(z,x) \rightarrow le(z,y) & (7)\\
(\forall x,y,z)\ x \rightarrow y \wedge y \rightarrow^* z \Rightarrow x \rightarrow^* z & (2) & (\forall x,y)\ x \rightarrow y \Rightarrow min(x) \rightarrow min(y) & (8)\\
(\forall x,y)\ x \rightarrow y \Rightarrow s(x) \rightarrow s(y) & (3) & (\forall y)\ le(0,s(y)) \rightarrow true & (9)\\
(\forall x,y,z)\ x \rightarrow y \Rightarrow cons(x,z) \rightarrow cons(y,z) & (4) & (\forall x,y)\ le(s(x),s(y)) \rightarrow le(x,y) & (10)\\
(\forall x,y,z)\ x \rightarrow y \Rightarrow cons(z,x) \rightarrow cons(z,y) & (5) & (\forall x)\ le(x,0) \rightarrow false & (11)\\
(\forall x,y,z)\ x \rightarrow y \Rightarrow le(x,z) \rightarrow le(y,z) & (6) & (\forall x)\ min(cons(x,nil)) \rightarrow x & (12)
\end{array}$$

$$(\forall x,y,xs)\ min(xs) \rightarrow^* y \wedge le(x,y) \rightarrow^* true \Rightarrow min(cons(x,xs)) \rightarrow x \qquad (13)$$

$$(\forall x,xs)\ min(xs) \rightarrow^* y \wedge le(x,y) \rightarrow^* false \Rightarrow min(cons(x,xs)) \rightarrow y \qquad (14)$$

2 Feasibility Sequences

Given a CTRS $\mathcal{R}$ and terms s and t, we say that the atom $s \rightarrow^* t$ is *$\mathcal{R}$-feasible* (or just *feasible* if no confusion arises) if there is a substitution σ such that the *reachability test* $\sigma(s) \rightarrow^*_{\mathcal{R}} \sigma(t)$ succeeds. As in [5, Definition 2], sequences $\mathcal{G} = (s_i \rightarrow^* t_i)_{i=1}^n$, where $n > 0$, are called *feasibility sequences.*[2] We say that $\mathcal{G}$ is *$\mathcal{R}$-feasible* if there is a substitution such that $\sigma(s_i) \rightarrow^*_{\mathcal{R}} \sigma(t_i)$ holds for all $1 \leq i \leq n$; we call $\mathcal{G}$ *infeasible* otherwise. In [4, 5], we presented an approach to deal with infeasibility using a satisfiability criterion: a sequence $\mathcal{G}$ as above is *infeasible* if the first-order theory $\overline{\mathcal{R}}$ together with the *negation* of the sentence

$$(\exists \vec{x}) \bigwedge_{i=1}^{n} s_i \rightarrow^* t_i \qquad (15)$$

where $\vec{x}$ are the variables occurring in terms s_i and t_i for $1 \leq i \leq n$, is *satisfiable* by any interpretation $\mathcal{A}$ of the function and predicate symbols, i.e., $\mathcal{A} \models \overline{\mathcal{R}} \cup \{\neg(15)\}$ holds [5, Theorem 6]. Actually, as showed in [3], if (15) is a logical consequence of $\overline{\mathcal{R}}$ (i.e., $\overline{\mathcal{R}} \vdash (15)$ holds), then the feasibility of $\mathcal{G}$ is *proved.* Thus, this logical approach provides a sound and complete method to (dis)prove feasibility.

Example 2. *Continuing with Example 1, the sequence: $min(nil) \rightarrow^* x, le(y,x) \rightarrow^* true$ corresponds to the following first-order formula (15): $(\exists x,y)\ min(nil) \rightarrow^* x \wedge le(y,x) \rightarrow^* true$*

In this paper, we present infChecker, a tool for proving and disproving feasibility conditions taking advantage of this logical approach:

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

[1] This problem belongs to the database COPS of confluence problems in http://cops.uibk.ac.at/

[2] In [5, Definition 2] atoms $s \rightarrow t$ are also considered in feasibility sequences.

3 Feasibility Framework

In order to automatically analyze whether a sequence $\mathcal{G}$ is *feasible* or *infeasible*, we describe a framework similar to the one presented in [1] for termination purposes. We define appropriate notions of (feasibility) *problem* and *processor* and show how to apply processors in order to prove or disprove feasibility.

Definition 3 (fProblem and fProcessor)**.** *An* fProblem τ *is a pair* $\tau = (\mathcal{R}, \mathcal{G})$*, where* $\mathcal{R}$ *is a CTRS and* $\mathcal{G}$ *is a sequence* $(s_i \to^* t_i)_{i=1}^n$*. The* fProblem τ *is* feasible *if* $\mathcal{G}$ *is* $\mathcal{R}$*-feasible; otherwise it is* infeasible*.*

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

An fProcessor P *is* sound *if for all* $\tau \in \mathcal{D}om(\mathsf{P})$*,* τ *is feasible whenever either* $\mathsf{P}(\tau) =$"yes" *or* $\exists \tau' \in \mathsf{P}(\tau)$*, such that* τ' *is feasible.*

An fProcessor P *is* complete *if for all* $\tau \in \mathcal{D}om(\mathsf{P})$*,* τ *is infeasible whenever* $\forall \tau' \in \mathsf{P}(\tau)$*,* τ' *is infeasible.*

Feasibility problems can be proved or disproved by using a proof tree as follows.

Theorem 4 (Feasibility Proof Tree)**.** *Let* τ *be an fProblem. A feasibility proof tree* $\mathcal{T}$ *for* τ *is a tree whose inner nodes are labeled with fProblems and the leaves may also be labeled with either* "yes" *or* "no"*. The root of* $\mathcal{T}$ *is labeled with* τ *and for every inner node* n *labeled with* τ'*, there is a processor* P *such that* $\mathsf{P} \in \mathcal{D}om(\mathsf{P})$ *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"*; and (3) if* $\mathsf{P}(\tau') = \{\tau_1, \ldots, \tau_k\}$ *with* $k > 0$*, then* n *has* k *children labeled with the fProblems* $\tau_1, \ldots, \tau_k$*.*

Theorem 5 (Feasibility Framework)**.** *Let* $\mathcal{R}$ *be an oriented CTRS,* $\mathcal{G}$ *be a feasibility sequence, and* $\mathcal{T}$ *be a feasibility proof tree for* $\tau_I = (\mathcal{R}, \mathcal{G})$*. Then: (1) if all leaves in* $\mathcal{T}$ *are labeled with* "no" *and all involved fProcessors are complete for the fProblems they are applied to, then* $\mathcal{G}$ *is* $\mathcal{R}$*-infeasible; and (2) if* $\mathcal{T}$ *has a leaf labeled with* "yes" *and all fProcessors in the path from* τ_I *to the leaf are sound for the fProblems they are applied to, then* $\mathcal{G}$ *is* $\mathcal{R}$*-feasible.*

In the following subsections we describe a number of sound and complete fProcessors.

3.1 Satisfiability Processor

The following processor integrates the satisfiability approach described in [5] to prove infeasibility in our framework. When dealing with reachability, i.e., all the left-hand sides of the feasibility conditions in $\mathcal{G}$ are ground, we can restrict our theory to the set of usable rules. Given an fProblem $(\mathcal{R}, \mathcal{G})$, we let

$$\mathcal{U}(\mathcal{R}, \mathcal{G}) = \begin{cases} \bigcup_{s_i \to^* t_i \in \mathcal{G}} \mathcal{U}(\mathcal{R}, s_i) & \text{if all } s_i \text{ in } \mathcal{G} \text{ are ground} \\ R & \text{otherwise} \end{cases}$$

where, given a CTRS $\mathcal{R}$ and a term t, $\mathcal{U}(\mathcal{R}, t)$ are the usable rules $\mathcal{R}$ regarding t [5, Section 2].

Theorem 6 (Satisfiability Processor)**.** *Let* $\tau = (\mathcal{R}, \mathcal{G})$ *be an fProblem with* $\mathcal{G} = (s_i \to^* t_i)_{i=1}^n$*. Let* $\mathcal{A}$ *be a structure such that* $\mathcal{A} \neq \emptyset$ *and* $\mathcal{A} \models \overline{\mathcal{U}(\mathcal{R}, \mathcal{G})} \cup \{\neg(\exists \vec{x}) \bigwedge_{i=1}^n s_i \to^* t_i\}$*. The processor* $\mathsf{P}^{\mathsf{Sat}}$ *given by* $\mathsf{P}^{\mathsf{Sat}}(\tau) = \emptyset$ *is sound and complete.*

In infChecker, we use the model generators AGES [2] and Mace4 [8] to find suitable structures $\mathcal{A}$ to be used in the implementation of $\mathsf{P}^{\mathsf{Sat}}$.

Example 7. *For* $\mathcal{R}$ *in Example 1 and* $\mathcal{G}$ *in Example 2, we obtain* $\mathsf{P}^{\mathsf{Sat}}(\tau_I) = \emptyset$ *using* AGES*.*

3.2 Provability Processor

The following processor integrates the logic-based approach to program analysis described in [3] to prove feasibility by theorem proving.

Theorem 8 (Provability Processor)**.** *Let $\tau = (\mathcal{R}, \mathcal{G})$ be an fProblem with $\mathcal{G} = (s_i \to^* t_i)_{i=1}^n$ such that $\overline{\mathcal{R}} \vdash (\exists \vec{x}) \bigwedge_{i=1}^n s_i \to^* t_i$ holds. The processor $\mathsf{P}^{\mathsf{Prov}}$ given by $\mathsf{P}^{\mathsf{Prov}}(\tau) =$ "yes" is sound and complete.*

In infChecker, we use the theorem prover Prover9 [8] as a backend to implement $\mathsf{P}^{\mathsf{Prov}}$.

Example 9. *For $\mathcal{R}$ in Example 1 and $\mathcal{G} = le(x, min(y)) \to^* false, min(y) \to^* x$ (836.trs) with associated first-order formula (15) as follows:*

$$(\exists x, y)\ le(x, min(y)) \to^* false \wedge min(y) \to^* x \tag{16}$$

we have $\mathsf{P}^{\mathsf{Prov}}(\tau_I) =$ "yes" by using Prover9*.*

3.3 Narrowing on Feasibility Conditions Processor

In the context of the 2D DP framework [7], there are powerful processors that can be applied to the conditions of the rules in order to simplify those conditions. We adapt the processor that narrow conditions to be used on fProblems.

Let $N_1(\mathcal{S}, s) = \{(t, \theta\downarrow_{\mathcal{V}ar(s)}) \mid s \leadsto_{\ell \to r \Leftarrow c, \theta} t, \ell \to r \Leftarrow c \in \mathit{NRules}(\mathcal{S}, s)\}$ represents the set of one-step $\mathcal{S}$-narrowings issued from s [7, Definition 79], where $\mathit{NRules}(\mathcal{S}, s)$ is the set of rules $\alpha : \ell \to r \Leftarrow c \in \mathcal{S}$ such that a nonvariable subterm t of s is a *narrex* of α, and $\theta\downarrow_{\mathcal{V}ar(s)}$ is a 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. As discussed in [7, Section 7.5], the set $N_1(\mathcal{S}, s)$ can be *infinite* if $\mathit{NRules}(\mathcal{S}, s)$ is *not* a TRS, i.e., it contains 'proper' conditional rules. In [7, Proposition 87] some sufficient conditions for finiteness of $N_1(\mathcal{S}, s)$ are given. Accordingly, we define a narrowing processor on fProblems. Given a feasibility sequence $\mathcal{G} = (s_i \to^* t_i)_{i=1}^n$ we let

$$\overline{\mathcal{N}}(\mathcal{S}, \mathcal{G}, i) = \{\mathcal{G}[\vec{\theta}, w \to^* t_i]_i \mid s_i \to^* t_i \in \mathcal{G}, (w, \theta) \in N_1(\mathcal{S}, s_i)\}$$

where $\vec{\theta}$ 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 10 (Narrowing on Feasibility Conditions Processor)**.** *Let $\tau = (\mathcal{R}, \mathcal{G})$ be an fProblem, $s_i \to^* t_i \in \mathcal{G}$, and $\mathcal{N} \subseteq \overline{\mathcal{N}}(\mathcal{R}, \mathcal{G}, i)$ finite. P^{NC} is given by $\mathsf{P}^{\mathsf{NC}}(\tau) = \{(\mathcal{R}, \mathcal{N})\}$.*

Theorem 11. *P^{NC} is sound. If $\mathcal{N} = \overline{\mathcal{N}}(\mathcal{R}, \mathcal{G}, i)$ and $s_i \to^* t_i \in \mathcal{G}$ is such that s_i and t_i do* not *unify and either s_i is* ground *or (1) $\mathit{NRules}(\mathcal{R}, s_i)$ is a TRS and (2) s_i is linear, then P^{NC} is complete.*

4 Experimental Evaluation

We participated in the *Infeasibility* (INF) category of the 2019 Confluence Competition (CoCo)[3]:

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

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

Note that answers *Yes/No* in the table refer to *infeasibility* problems (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 negative answers, there are 7 more examples that can be proved positively using infChecker only. Furthermore, there are 10 examples that can be proved by other tools and cannot be proved by infChecker.

5 Conclusions and Future Work

In this paper we present infChecker, a new tool for checking feasibility conditions of CTRSs that takes advantage of the logic-based approach presented in [3, 4, 5]. We succesfully participated in the 2019 Confluence Competition in the INF (infeasibility) category, being the most powerful tool for checking both infeasibility and feasibility.

Currently, the tool has only three processors. As a subject for future work, we would like to increase the power of the tool with new processors focused on feasibility conditions, analyze the examples that cannot be proved by infChecker but can be proved by other tools and extend the tool to deal with more involved rewrite systems (order-sorted, axioms. . .).

References

[1] J. Giesl, R. Thiemann, P. Schneider-Kamp, and S. Falke. Mechanizing and Improving Dependency Pairs. *Journal of Automatic Reasoning*, 37(3):155–203, 2006.

[2] R. Gutiérrez and S. Lucas. Automatic Generation of Logical Models with AGES (System Description). In P. Fontaine, editor, *Proc. of the CADE 2019*, LNCS, to appear. Springer, 2019.

[3] S. Lucas. Proving Program Properties as First-Order Satisfiability. In F. Mesnard and P. J. Stuckey, editors, *LOPSTR 2018*, volume 11408 of *LNCS*, pages 3–21. Springer, 2019.

[4] S. Lucas and R. Gutiérrez. A Semantic Criterion for Proving Infeasibility in Conditional Rewriting. In B. Accattoli and B. Felgenhauer, editors, *Proc. of the IWC'17*, pages 15–19, 2017.

[5] S. Lucas and R. Gutiérrez. Use of Logical Models for Proving Infeasibility in Term Rewriting. *Information Processing Letters*, 136:90–95, 2018.

[6] S. Lucas, C. Marché, and J. Meseguer. Operational termination of conditional term rewriting systems. *Information Processing Letters*, 95(4):446–453, 2005.

[7] S. Lucas, J. Meseguer, and R. Gutiérrez. The 2D Dependency Pair Framework for conditional rewrite systems. Part I: Definition and basic processors. *Journal of Computer and System Sciences*, 96:74–106, 2018.

[8] W. McCune. Prover9 and Mace4. [online]. Available at https://www.cs.unm.edu/~mccune/mace4/.