Document downloaded from:

http://hdl.handle.net/10251/169643

This paper must be cited as:

Lucas Alba, S. (2020). Using Well-Founded Relations for Proving Operational Termination. Journal of Automated Reasoning. 64(2):167-195. https://doi.org/10.1007/s10817-019-09514-2

The final publication is available at

https://doi.org/10.1007/s10817-019-09514-2

Copyright Springer-Verlag

Additional Information

jar manuscript No.
(will be inserted by the editor)

Using Well-Founded Relations for Proving Operational Termination

Salvador Lucas

Abstract In this paper, we study *operational termination*, a proof theoretical notion for capturing the termination behavior of computational systems. We prove that operational termination can be characterized at different levels by means of well-founded relations on specific formulas which can be obtained from the considered system. We show how to obtain such well-founded relations from logical models which can be automatically generated using existing tools.

Keywords Declarative languages · Logical models · Operational termination · Program analysis · Well-foundedness.

1 Introduction

Computations are often defined as provability of goals in an appropriate inference system. Following a *Natural Deduction* approach *à la Gentzen* [34], provability of goals is naturally understood as the construction of a proof tree for each considered goal. The proof tree is built up using the inference rules to expand goals in nodes. Each rule introduces a new (possibly empty) list of goals to be proved in the next level of the tree. In an implementation, we usually assume some fixed criterion to prove such new goals in any *well-formed* proof tree. For instance, by considering them from left to right. Systems whose computations are defined in this way are called *operationally terminating* if no goal originates an infinite (well-formed) proof tree [26, 8]. Some researchers have used operational termination as a basis for the definition and implementation of techniques and tools for analyzing the termination behavior of (declarative) programs in various formalisms, including Conditional Term Rewriting Systems (CTRSs) [27,37,38], Membership Equational Theories [8], and Generalized Rewriting Theories [7], among others (see also [2,9,35]).

In order to illustrate the need of operational termination in termination analysis we consider a simple Maude program below. Maude [6] is a sophisticated language whose termination behavior is determined by the interaction of several features.

Partially supported by the EU (FEDER), projects TIN2015-69175-C4-1-R, and GV PROMETEOII/2015/013

DSIC, Universitat Politècnica de València, Spain

```
fmod INF is
  sorts S T .
  subsorts S < T .
  ops a b : -> T .
  ceq a = b if a : S .
endfm
```

Fig. 1 Example of Maude program

$$(SR)\ \frac{x \to y \quad y : \mathtt{S}}{x : \mathtt{S}} \qquad (M1)_{\mathtt{S<T}}\ \frac{x :: \mathtt{S}}{x :: \mathtt{T}} \qquad (M1)_{\mathtt{a}}\ \frac{}{\mathtt{a} :: \mathtt{T}} \qquad (M1)_{\mathtt{b}}\ \frac{}{\mathtt{b} :: \mathtt{T}}$$

$$(M2)_{\mathtt{S}}\ \frac{x :: \mathtt{S}}{x : \mathtt{S}} \qquad (M2)_{\mathtt{T}}\ \frac{x :: \mathtt{T}}{x : \mathtt{T}} \qquad (Rl)\ \frac{\mathtt{a} : \mathtt{S}}{\mathtt{a} \to \mathtt{b}} \qquad (Rf)\ \frac{}{x \to^* x}$$

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

Fig. 2 The inference system $\mathcal{I}(\mathtt{INF})$

Some of them are considered here to illustrate our techniques. No previous knowledge about Maude is required, though. Furthermore, the results in this paper are largely independent from Maude, as they actually rely on the abstract notion of a *general logic* [31].

Consider the Maude program INF in Figure 1. The program declares **sorts** S and T. Sort S is a **subsort** of T, i.e., terms of sort S also have sort T (symbol **<** plays the role of set inclusion $\subseteq$). Operation symbols are declared by using the keyword **op** (or **ops** for several operators). Two constants a and b (of sort T) are declared in the program. The *conditional equation* **ceq** a **=** b **if** a **:** S specifies the reduction of constant a into b provided that the membership of constant a to sort S (written a **:** S) can be proved. After loading INF in the Maude interpreter[1] the attempt to reduce constant a by using Maude's command red, leads to an error:

```
Maude> red a .
reduce in INF : a .
Segmentation fault: 11
```

Typically, the last message reveals an unbounded consumption of memory due to a nonterminating computation. How do we explain this? The following question arises:

How to *define* the termination behavior of program INF?

Operational termination provides an appropriate answer. The inference system $\mathcal{I}(\mathtt{INF})$ in Figure 2 describes the execution of program INF in Figure 1. The inference rules are obtained from the generic inference system in [8, Figure 4].[2] Rule (SR) formalizes the idea that, in Maude, terms can be given a sort S if they can be reduced (with $\to$) into a term which is proved to be of sort S. For this reason, for a given sort s, we use $\_ : s$ for such memberships in contrast to $\_ :: s$, which specifies the sort of a constant or variable symbol which is directly obtained from the specification (by using an operation or variable declaration). Rule $(M1)_{\mathtt{S<T}}$ describes the subsort relation between sorts S and T. Rules $(M1)_{\mathtt{a}}$ and $(M1)_{\mathtt{b}}$ associate a sort to constants a and b. Rules $(M2)_{\mathtt{S}}$ and $(M2)_{\mathtt{T}}$ connect the two previous memberships for sorts S and T in

[1] see http://maude.cs.illinois.edu/

[2] The labels of the rules refer to such a system: *SR* stands for *subject reduction*, *M1* and *M2* for *membership-1/-2*, *Rf* for *reflexivity*, *Rl* for *replacement*, and *T* for *transitivity*.

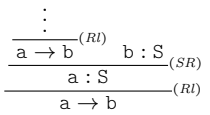


Fig. 3 Infinite proof tree for program INF

the obvious way. Rule (Rl) encodes the conditional equation in the program. Finally, rules (Rf) and (T) define the behavior of the many-step reduction relation $\rightarrow^*$ in the usual way. By using these inference rules, we build the infinite well-formed proof tree in Figure 3 witnessing that INF is *not* operationally terminating.

In termination analysis (back to an early 1949 paper by Turing [39] and also in Floyd [10]), *well-founded relations* have been paramount in modeling, analyzing, and (automatically) proving termination. The following questions arise: (i) Is there a characterization of operational termination in terms of well-founded relations? (ii) Can we use well-founded relations in mechanized proofs of operational termination? In this paper we give positive answers to these questions. After some preliminaries in Section 2, the contributions of the paper are presented as follows:

- In Section 3, operational termination of (the *specification* of) a theory $\mathcal{S}$ is characterized as the well-foundedness of a binary relation *on formulas* which we call the *proof progress* relation (Theorem 1, which gives a positive answer to (i) above). Such a relation is defined by considering all well-formed proof trees associated to $\mathcal{S}$. There usually are infinitely many well-formed proof trees. Then,
- Section 4 shows how to use *proof jumps* [28] instead. For a finite inference system $\mathcal{I}(\mathcal{S})$ for $\mathcal{S}$, proof jumps provide a finite description of how (infinite) well-formed proof trees are built. Proof jumps are obtained from the inference rules to keep track of where proof trees are expanded and which are the provable assumptions for such an expansion. In this way, we obtain a second characterization of operational termination by using well-founded relations that are used to compare specific components of the proof jumps (Theorem 3).
- Section 5 explains how to *synthesize* well-founded relations that can be used to implement the desired comparisons between components of proof jumps to prove operational termination (Theorem 4, which gives a positive answer to (ii)).
- Section 6 shows the practical use of our methods within the OT Framework for proving operational termination developed in [28].

Section 7 concludes. The material in Sections 5 and 6 extends and generalizes previous results in [22, Sections 5.1 and 5.2]. Sections 3 and 4 are completely new.

2 Preliminaries

As in [28], we rely on the notion of a *general logic* [31] which is made more expressive by supporting inference systems that are parametric on theories. The central notion of a general logic $\mathcal{L}$ is that of a (*specification* of a) theory $\mathcal{S}$. A theory $\mathcal{S}$ is given (i) a set of formulas $Form(\mathcal{S})$ that can be used with the theory, (ii) a set of substitutions $Sub(\mathcal{S})$ which are viewed as mappings (or transformations) of formulas in $Form(\mathcal{S})$, and (iii) an inference system $\mathcal{I}(\mathcal{S})$ which is a set of sequences of formulas

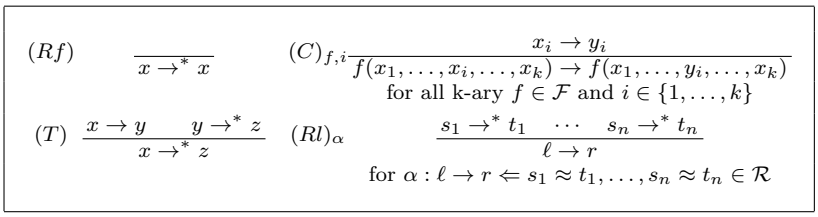


Fig. 4 Schemes of inference rules for (oriented) CTRSs

$A, B_1, \ldots, B_n$ that are interpreted as inference rules $\frac{B_1 \cdots B_n}{A}$. Then, we consider (iv) a notion of deduction which uses the inference rules in $\mathcal{I}(\mathcal{S})$ to build proof trees for goals G which are formulas in $Form(\mathcal{S})$. We first illustrate our approach with some examples. Then, Sections 2.2 and 2.3 provide full definitions for (i)-(iv) above. Finally, Section 2.4 provides a brief introduction to Order-Sorted First-Order Logic.

2.1 Running Examples

Our first example concerns *oriented* CTRSs [33, Chapter 7] which extend Term Rewriting Systems (TRSs [5]). In a *conditional rewrite rule* $\ell \to r \Leftarrow c$, ℓ and r are *terms* of a signature of symbols $\mathcal{F}$; the conditional part c is a (possibly empty) sequence $s_1 \approx t_1, \ldots, s_n \approx t_n$ of conditions. These conditions must be *satisfied* before being allowed to apply a rewriting step with ℓ and r in the usual way. Dealing with oriented CTRSs, conditions $s_i \approx t_i$ are treated as reachability conditions $\sigma(s_i) \to^* \sigma(t_i)$, after applying an appropriate (matching) substitution σ [33, Definition 7.1.3]. The generic inference system for oriented CTRSs is in Figure 4.

Example 1 Consider the CTRS $\mathcal{R} = \{\mathsf{a} \to \mathsf{b}, \mathsf{f}(\mathsf{a}) \to \mathsf{b}, \mathsf{g}(x) \to \mathsf{g}(\mathsf{a}) \Leftarrow \mathsf{f}(x) \approx x\}$ in [11, page 46]. When the inference system in Figure 4 is specialized to $\mathcal{R}$, we obtain an inference system $\mathcal{I}(\mathcal{R})$ as follows:

$$(Rf)\ \frac{}{x \to^* x} \quad (T)\ \frac{x \to y \quad y \to^* z}{x \to^* z} \quad (C)_{\mathsf{f}}\ \frac{x \to y}{\mathsf{f}(x) \to \mathsf{f}(y)} \quad (C)_{\mathsf{g}}\ \frac{x \to y}{\mathsf{g}(x) \to \mathsf{g}(y)}$$

$$(Rl)_1\ \frac{}{\mathsf{a} \to \mathsf{b}} \quad (Rl)_2\ \frac{}{\mathsf{f}(\mathsf{a}) \to \mathsf{b}} \quad (Rl)_3\ \frac{\mathsf{f}(x) \to^* x}{\mathsf{g}(x) \to \mathsf{g}(\mathsf{a})}$$

Now, $s \to_{\mathcal{R}} t$ (resp. $s \to^*_{\mathcal{R}} t$), read "term s rewrites to t in $\mathcal{R}$" (resp. "in 0 or more steps"), if $s \to t$ (resp. $s \to^* t$) is proved in $\mathcal{I}(\mathcal{R})$.

Let us consider an additional example, now written in Maude, where rewriting is not used, but sorts, subsorts, and memberships are essential.

Example 2 The Maude program in Figure 5 exemplifies memberships in Maude [6, Sect. 4.2]. The program uses a sort `Zero` to hold a constant `zero` only; sort `Nat` is intended to represent natural numbers in Peano's notation, where `zero` represents 0, term `s(zero)` represents 1, term `s(s(zero))` represents 2, etc.; finally, sort `3*Nat` is intended to collect natural numbers which are *multiples of* 3. The natural inclusion relationship between these three sorts is made explicit by appropriately defining them as **`subsorts`**. Operation symbols are declared by using keyword **`op`** and giving the symbol a rank $s_1 \cdots s_k \to s$ where $s_1, \ldots, s_k$ are the sorts of the input arguments

```
fmod 3*NAT is
  sorts Zero Nat 3*Nat .
  subsorts Zero < 3*Nat < Nat .
  op zero : -> Zero .
  op s : Nat -> Nat .
  var M3 : 3*Nat .
  mb s(s(s(M3))) : 3*Nat . *** membership axiom
endfm
```

Fig. 5 Use of memberships in Maude

$$(SR)_Z\ \frac{x \to y \quad y : \mathtt{Zero}}{x : \mathtt{Zero}} \qquad (SR)_{3N}\ \frac{x \to y \quad y : 3 * \mathtt{Nat}}{x : 3 * \mathtt{Nat}} \qquad (SR)_N\ \frac{x \to y \quad y : \mathtt{Nat}}{x : \mathtt{Nat}}$$

$$(M1)_{\mathtt{Z<3N}}\ \frac{x :: \mathtt{Zero}}{x :: 3 * \mathtt{Nat}} \qquad (M1)_{\mathtt{3N<N}}\ \frac{x :: 3 * \mathtt{Nat}}{x :: \mathtt{Nat}} \qquad (M1)_{\mathtt{zero}}\ \frac{}{\mathtt{zero} :: \mathtt{Zero}}$$

$$(M1)_{\mathtt{s}}\ \frac{x :: \mathtt{Nat}}{\mathtt{s}(x) :: \mathtt{Nat}} \qquad (M1)_{\mathtt{mb}}\ \frac{\mathtt{M3} :: 3 * \mathtt{Nat}}{\mathtt{s}(\mathtt{s}(\mathtt{s}(\mathtt{M3}))) :: 3 * \mathtt{Nat}}$$

$$(M2)_Z\ \frac{x :: \mathtt{Zero}}{x : \mathtt{Zero}} \qquad (M2)_{3N}\ \frac{x :: 3 * \mathtt{Nat}}{x : 3 * \mathtt{Nat}} \qquad (M2)_N\ \frac{x :: \mathtt{Nat}}{x : \mathtt{Nat}}$$

$$(C)_{\mathtt{s},1}\ \frac{x \to y}{\mathtt{s}(x) \to \mathtt{s}(y)} \qquad (Rf)_N\ \frac{}{x \to^* x} \qquad (T)_N\ \frac{x \to y \qquad y \to^* z}{x \to^* z}$$

Fig. 6 Inference rules for $3 * \mathtt{NAT}$ program

of the operator and s is the output sort. The program consists of a membership axiom `mb s(s(s(M3))) : 3*Nat`, establishing that prefixing an expression `M3` of sort `3*Nat` (hence intended to hold a multiple of 3) with three applications of `s` yields a multiple of 3. The computations intended for this program are membership queries of the form e `::3*Nat` for some expression e. After loading the program in the Maude interpreter, we check this by using Maude's command `red` that evaluates the expression (e `::3*Nat` in this case) which is given as argument. For instance,

```
Maude> red s(s(s(s(s(zero))))) :: 3*Nat .
reduce in 3*NAT : s(s(s(s(s(zero))))) :: 3*Nat .
rewrites: 3 in 0ms cpu (0ms real) (3000000 rewrites/second)
result Bool: true
```

shows that, as expected, `s(s(s(s(s(zero)))))`, i.e., 6, is a multiple of 3. The inference system $\mathcal{I}(3 * \mathtt{NAT})$ in Figure 6 describes the operational semantics of `3*NAT` in Figure 5. It is also obtained from the generic inference system in [8, Figure 4].[3]

2.2 General Logics

A *logic* is a quadruple $\mathcal{L} = (Th(\mathcal{L}), Form, Sub, \mathcal{I})$, where: (i) $Th(\mathcal{L})$ is the class of *theories* of $\mathcal{L}$, (ii) $Form$ is a mapping sending each theory $\mathcal{S} \in Th(\mathcal{L})$ to a set $Form(\mathcal{S})$ of *formulas* of $\mathcal{S}$, (iii) Sub is a mapping sending each $\mathcal{S} \in Th(\mathcal{L})$ to its set $Sub(\mathcal{S})$ of *substitutions*, with the containment $Sub(\mathcal{S}) \subseteq (Form(\mathcal{S}) \to Form(\mathcal{S}))$, and (iv) $\mathcal{I}$ is a mapping sending each $\mathcal{S} \in Th(\mathcal{L})$ to a subset $\mathcal{I}(\mathcal{S}) \subseteq Form(\mathcal{S}) \times Form(\mathcal{S})^*$, where each $(A, B_1 \dots B_n) \in \mathcal{I}(\mathcal{S})$ is called an *inference rule* for $\mathcal{S}$ and denoted $\frac{B_1 \dots B_n}{A}$. In the following, sequences $B_1 \dots B_n$ of formulas are often written $\mathbf{B}_n$ for short.

[3] Some new labels referring to such a system are used now: *M1* for *membership-1* and *C* for *congruence*.

Example 3 For $\mathcal{L} = CTRS$, (i) $Th(CTRS)$ is the class of CTRSs $\mathcal{R} = (\mathcal{F}, R)$ with $\mathcal{F}$ a signature and R a set of conditional rules (over $\mathcal{F}$); (ii) $Form(\mathcal{R}) = \{s \to t, s \to^* t \mid s, t \in \mathcal{T}(\mathcal{F}, \mathcal{X})\}$, where $\mathcal{T}(\mathcal{F}, \mathcal{X})$ is the set of terms over $\mathcal{F}$ with variables from $\mathcal{X}$; (iii) $Sub(\mathcal{R})$ is the set of substitutions defined as usual with the following extension to formulas: $\sigma(s \to t) = \sigma(s) \to \sigma(t)$, and $\sigma(s \to^* t) = \sigma(s) \to^* \sigma(t)$; and (iv) $\mathcal{I}(\mathcal{R})$ is the instantiation of the generic inference system of Figure 4.

Example 4 For $\mathcal{L} = MRT$ (the logic of *Membership Rewrite Theories*, cf. [8, Section 3]), (i) $Th(MRT)$ is the class of membership rewrite theories $\mathcal{R}$ over a signature $(K, \Sigma, S \cup S')$ where K is a set of kinds, Σ is an indexed family of sets $\Sigma = \{\Sigma_{w,k}\}_{(w,k)\in K^*,K}$ of function symbols, and $S = \{S_k\}_{k\in K}$ is a disjoint family of unary predicates; each $s \in S_k$ is called a sort, and is understood as a unary (membership) predicate on k, written $\_ : s$; similarly, $S' = \{S'_k\}_{k\in K}$ is a disjoint family of membership predicates $\_ :: s$, each of them understood as a subrelation of the corresponding $\_ : s$. The theory $\mathcal{R}$ consists of a set of rules of the form $(t \to t' \text{ if } A_1, \ldots, A_n)$ or $(t : s \text{ if } A_1, \ldots, A_n)$, where t and t' are terms over the signature Σ and possibly including variables from a set $\mathcal{X} = \{\mathcal{X}_k\}_{k\in K}$, s is a sort, and for all $1 \leq i \leq n$, A_i is either a rewrite condition $u_i \to^* v_i$ for terms u_i and v_i, or a membership $t_i : s_i$ or $t_i :: s_i$ for a term t_i and sort s_i. (ii) $Form(\mathcal{R})$ consists of formulas $u \to v$, $u \to^* v$, $u : s$, and $u :: s$ for terms u, v of a given kind k and $s \in S_k$. (iii) $Sub(\mathcal{R})$ is the set of (kind-preserving) substitutions defined as usual with the following extension to formulas: for all terms t, u, v and sorts s, $\sigma(u \to v) = \sigma(u) \to \sigma(v)$, $\sigma(u \to^* v) = \sigma(u) \to^* \sigma(v)$, $\sigma(t : s) = \sigma(t) : s$, and $\sigma(t :: s) = \sigma(t) :: s$; finally, (iv) $\mathcal{I}(\mathcal{R})$ is the instantiation of the generic inference system in [8, Figure 4]. Note that, according to the previous presentation of the MRT logic, variables x, y, z, and `M3` occurring in $\mathcal{I}$(`3*NAT`) in Figure 6 all have the single *kind* [`Nat`].

We assume some standard properties about substitutions $\sigma \in Sub(\mathcal{S})$ [28, Section 2]. Given $F, F' \in Form(\mathcal{S})$, $mgu_{\mathcal{S}}(F, F') \subseteq Sub(\mathcal{S})$ denotes a set such that: (i) $\forall \sigma \in mgu_{\mathcal{S}}(F, F')$, $\sigma(F) = \sigma(F')$; and (ii) $\forall \tau \in Sub(\mathcal{S})$ such that $\tau(F) = \tau(F')$, there is $\sigma \in mgu_{\mathcal{S}}(F, F'), \theta \in Sub(\mathcal{S})$ such that $\tau = \theta \circ \sigma$, i.e., for all formulas F, $\tau(F) = \theta(\sigma(F))$.

2.3 Proof Trees and Operational Termination

Given a logic $\mathcal{L}$, a theory $\mathcal{S} \in Th(\mathcal{L})$, and a formula $G \in Form(\mathcal{S})$, a finite proof tree T with root G (denoted $root(T) = G$) is either: (i) an *open goal*, simply denoted as G; or (ii) a *derivation tree* denoted as $\frac{T_1 \quad \cdots \quad T_n}{G}(\rho)$, where $T_1, \ldots, T_n$ are finite proof trees (for $n \geq 0$), and $\rho : \frac{B_1 \cdots B_n}{A}$ is an inference rule such that $G = \sigma(A)$, and $root(T_1) = \sigma(B_1), \ldots, root(T_n) = \sigma(B_n)$ for some substitution σ. A finite proof tree T is *closed* if it contains no open goals. We write $\mathcal{S} \vdash F$ for a formula F if there is a closed proof tree T with $root(T) = F$ using $\mathcal{I}(\mathcal{S})$. The *theorems* of $\mathcal{S}$ are the formulas $F \in Form(\mathcal{S})$ for which we can derive a closed proof tree.

A finite proof tree T is a proper prefix of a finite proof tree T' (written $T \subset T'$) if there are one or more open goals $G_1, \ldots, G_n$ in T such that T' is obtained from T by replacing each G_i by a finite derivation tree T_i with root G_i. An *infinite proof tree* T is an infinite increasing chain of finite proof trees, i.e., a sequence $(T_i)_{i\in\mathbb{N}}$ such that for all i, $T_i \subset T_{i+1}$. Since for all $i \in \mathbb{N}$, $root(T_i) = root(T_{i+1})$, we write

$root(T) = root(T_0)$. There can be different (equivalent) ways to represent an infinite proof tree T [27].

A finite proof tree T is *well-formed* if it is either an open goal, or a closed proof tree, or a derivation tree $\frac{T_1 \quad \cdots \quad T_n}{G}(\rho)$, where $T_1, \ldots, T_{i-1}$ are closed for some $1 \leq i \leq n$, T_i is a well-formed but not closed finite proof tree, and $T_{i+1}, \ldots, T_n$ are open goals. Note the *left-to-right* construction of the proof tree. An infinite proof tree is well-formed if it is an increasing chain of well-formed finite proof trees.

Definition 1 (Operational termination [26]) A theory $\mathcal{S}$ in a logic $\mathcal{L}$ is called *operationally terminating* if no infinite well-formed proof tree for $\mathcal{I}(\mathcal{S})$ exists.

2.3.1 The Spine of an Infinite Well-Formed Proof Tree

Infinite well-formed proof trees have the following form, see [28] and [27, Sect. 2.4]:

$$\dfrac{\mathbf{T}_1 \qquad \dfrac{\mathbf{T}_{n-1} \qquad \dfrac{\vdots}{G_n}(\rho_n) \qquad \mathbf{O}_{n-1}}{\vdots}(\rho_2) \qquad \mathbf{O}_1}{G_1}(\rho_1) \tag{1}$$

with a single infinite branch where there is a substitution σ such that, for all $i \geq 1$, a (possibly renamed) rule ρ_i of the form $\frac{B_1^i \cdots B_{n_i}^i}{A^i}$ for some $n_i > 0$ has been applied, $G_i = \sigma(A^i)$, $\mathbf{T}_i$ are sequences of $m_i - 1$ closed proof trees for some $0 < m_i \leq n_i$, and $\mathbf{O}_i$ are sequences of $n_i - m_i$ open goals. Note that we also have $G_{i+1} = \sigma(A^{i+1}) = \sigma(B_{m_i}^i)$ for all $i \geq 1$. The inference rules ρ_i which are used to build such an infinite branch together with the indices m_i of the formulas $B_{m_i}^i$ matching the goals G_{i+1} are collected in a sequence of pairs $(\langle \rho_i, m_i \rangle)_{i \geq 1}$ which we call the *spine*[4] of (1).

Definition 2 (Spine) Let T be an infinite well-formed proof tree of the form (1). We call the infinite sequence $(\langle \rho_i, m_i \rangle)_{i \geq 1}$ the *spine* of T, denoted *spine*(T).

2.4 Order-Sorted First-Order Logic

This section provides a brief introduction to Order-Sorted First-Order Logic (*OSFOL* for short). Missing details can be found in [15,16]. Given a set of *sorts* S, a many-sorted signature is an $S^* \times S$-indexed family of sets $\Sigma = \{\Sigma_{w,s}\}_{(w,s) \in S^* \times S}$ containing *function symbols* with a given string of argument sorts and a result sort. If $f \in \Sigma_{s_1 \cdots s_n, s}$, then we display f as $f : s_1 \cdots s_n \to s$. This is called a *rank* declaration for symbol f. Constant symbols c (taking no argument) have rank declaration $c : \lambda \to s$ for some sort s (where λ denotes the *empty* sequence). An order-sorted signature $(S, \leq, \Sigma)$ consists of a poset of sorts $(S, \leq)$ together with a many-sorted signature (S, Σ). The *connected components* of $(S, \leq)$ are the equivalence classes $[s]$ corresponding to the least equivalence relation $\equiv_{\leq}$ containing $\leq$. We extend the

[4] Since the drawing of the tree in (1) suggests the back of a skeleton, we use 'spine' for the central part, or backbone, of the tree.

order $\leq$ on S to strings of equal length in S^* by $s_1 \cdots s_n \leq s'_1 \cdots s'_n$ iff[5] $s_i \leq s'_i$ for all i, $1 \leq i \leq n$. Symbols f can be *subsort-overloaded*, i.e., they can have several rank declarations related in the $\leq$ ordering [16]. Constant symbols, however, have only one rank declaration. Given an S-sorted set $\mathcal{X} = \{\mathcal{X}_s \mid s \in S\}$ of mutually disjoint sets of variables (which are also disjoint from the signature Σ), the set $\mathcal{T}_\Sigma(\mathcal{X})_s$ of terms of sort s is the least set such that (i) $\mathcal{X}_s \subseteq \mathcal{T}_\Sigma(\mathcal{X})_s$, (ii) if $s' \leq s$, then $\mathcal{T}_\Sigma(\mathcal{X})_{s'} \subseteq \mathcal{T}_\Sigma(\mathcal{X})_s$; and (iii) for each $f : s_1 \cdots s_n \to s$ and $t_i \in \mathcal{T}_\Sigma(\mathcal{X})_{s_i}$, $1 \leq i \leq n$, $f(t_1, \ldots, t_n) \in \mathcal{T}_\Sigma(\mathcal{X})_s$. The set $\mathcal{T}_\Sigma(\mathcal{X})$ of order-sorted terms is $\mathcal{T}_\Sigma(\mathcal{X}) = \cup_{s \in S} \mathcal{T}_\Sigma(\mathcal{X})_s$. If $\mathcal{X} = \varnothing$, we write $\mathcal{T}_\Sigma$ rather than $\mathcal{T}_\Sigma(\varnothing)$ for the set of *ground* terms. Substitutions σ are S-sorted mappings $(\sigma_s)_{s \in S}$, where each σ_s maps variables in $\mathcal{X}_s$ into terms in $\mathcal{T}_\Sigma(\mathcal{X})_s$ and are extended to terms in the usual way.

An order-sorted first-order signature *with predicates* (OSFO-signature for short) is a quadruple $\Omega = (S, \leq, \Sigma, \Pi)$ such that $(S, \leq, \Sigma)$ is an order-sorted signature, and $\Pi = (\Pi_w)_{w \in S^+}$ is a family of *predicate symbols* $P, Q, \ldots$. We write $P : w$ for $P \in \Pi_w$. Overloading is also allowed on predicates. The formulas $F \in \mathit{Form}_\Omega$ of an OSFO-signature Ω are built up from atoms $P(t_1, \ldots, t_n)$ with $P \in \Pi_w$ and $t_1, \ldots, t_n \in \mathcal{T}_\Sigma(\mathcal{X})_w$, logic connectives (e.g., $\wedge$, $\neg$) and quantifiers ($\forall$) as follows: (i) if $P \in \Pi_w$, $w = s_1 \cdots s_n$, and $t_i \in \mathcal{T}_\Sigma(\mathcal{X})_{s_i}$ for all i, $1 \leq i \leq n$, then $P(t_1, \ldots, t_n) \in \mathit{Form}_\Omega$. (ii) if $F \in \mathit{Form}_\Omega$, then $\neg F \in \mathit{Form}_\Omega$; (iii) if $F, F' \in \mathit{Form}_\Omega$, then $F \wedge F' \in \mathit{Form}_\Omega$; (iv) if $s \in S$, $x \in \mathcal{X}_s$, and $F \in \mathit{Form}_\Omega$, then $(\forall x : s)\, F \in \mathit{Form}_\Omega$. As usual, we can consider formulas involving other logic connectives and quantifiers (e.g., $\vee$, $\Rightarrow$, $\Leftrightarrow$, $\exists$,...) by using their standard definitions in terms of $\wedge$, $\neg$, $\forall$. A closed formula, i.e., whose variables are all universally or existentially quantified, is called a *sentence*. Substitutions σ apply to formulas F in the usual way, i.e., by replacing free variables x occurring in F by terms $\sigma(x)$ (note that x and $\sigma(x)$ have the same sort), perhaps after renaming bound variables to avoid clashes.

Given a many-sorted signature (S, Σ), an (S, Σ)-algebra $\mathcal{A}$ (or just a Σ-algebra, if S is clear from the context) is a family $\{\mathcal{A}_s \mid s \in S\}$ of sets called the *carriers* or *domains*[6] of $\mathcal{A}$ together with a function $f^{\mathcal{A}}_{w,s} \in \mathcal{A}_w \to \mathcal{A}_s$ for each $f \in \Sigma_{w,s}$ where $\mathcal{A}_w = \mathcal{A}_{s_1} \times \cdots \times \mathcal{A}_{s_k}$ if $w = s_1 \cdots s_k$, and $\mathcal{A}_w$ is a singleton when $w = \lambda$ (in this case, disregarding the 'dummy' element in $\mathcal{A}_w$, the set of mappings $\mathcal{A}_w \to \mathcal{A}_s$ is isomorphic to $\mathcal{A}_s$, as expected). Given an order-sorted signature $(S, \leq, \Sigma)$, an $(S, \leq, \Sigma)$-algebra (or Σ-algebra if $(S, \leq)$ is clear from the context) is an (S, Σ)-algebra such that

1. If $s, s' \in S$ are such that $s \leq s'$, then $\mathcal{A}_s \subseteq \mathcal{A}_{s'}$, and
2. If $f \in \Sigma_{w_1,s_1} \cap \Sigma_{w_2,s_2}$ and $w_1 \leq w_2$, then $f^{\mathcal{A}}_{w_1,s_1} \in \mathcal{A}_{w_1} \to \mathcal{A}_{s_1}$ equals $f^{\mathcal{A}}_{w_2,s_2} \in \mathcal{A}_{w_2} \to \mathcal{A}_{s_2}$ on $\mathcal{A}_{w_1}$.

Let $\Omega = (S, \leq, \Sigma, \Pi)$ be an OSFO-signature. An *Ω-structure*[7] is an order-sorted $(S, \leq, \Sigma)$-algebra $\mathcal{A}$ together with an assignment to each $P \in \Pi_w$ of a subset $P^{\mathcal{A}}_w \subseteq \mathcal{A}_w$ such that [15]: (i) for P the identity predicate $\_ = \_ : ss$, the assignment is the identity relation, i.e., $(=)^{\mathcal{A}}_{s\,s} = \{(a, a) \mid a \in \mathcal{A}_s\}$; and (ii) whenever $P : w_1$ and $P : w_2$ and $w_1 \leq w_2$, then $P^{\mathcal{A}}_{w_1} = \mathcal{A}_{w_1} \cap P^{\mathcal{A}}_{w_2}$.

Given an S-sorted *valuation mapping* $\alpha : \mathcal{X} \to \mathcal{A}$ we obtain the evaluation mapping $[\_]^{\mathcal{A}}_\alpha : \mathcal{T}_\Sigma(\mathcal{X}) \to \mathcal{A}$ in the usual way. For ground terms t, since their

[5] In the following, *iff* means *if and only if*.

[6] Following [18, Section 1.1], these sets can be *empty*.

[7] As in [18], we use 'structure' and reserve the word 'model' to refer to those structures satisfying a given set of sentences (theory).

interpretation do not depend on any valuation α, we often write $[t]^{\mathcal{A}}$ or just $t^{\mathcal{A}}$ to denote their semantic value. Finally, $[\_]^{\mathcal{A}}_{\alpha} : \mathit{Form}_{\Omega} \to \mathit{Bool}$ is given by:

1. $[P(t_1, \ldots, t_n)]^{\mathcal{A}}_{\alpha} = \mathit{true}$ (with $P \in \Pi_w$) if and only if $([t_1]^{\mathcal{A}}_{\alpha}, \ldots, [t_n]^{\mathcal{A}}_{\alpha}) \in P^{\mathcal{A}}_w$;
2. $[\neg F]^{\mathcal{A}}_{\alpha} = \mathit{true}$ if and only if $[F]^{\mathcal{A}}_{\alpha} = \mathit{false}$;
3. $[F \wedge F']^{\mathcal{A}}_{\alpha} = \mathit{true}$ if and only if $[F]^{\mathcal{A}}_{\alpha} = \mathit{true}$ and $[F']^{\mathcal{A}}_{\alpha} = \mathit{true}$; and
4. $[(\forall x : s)\, F]^{\mathcal{A}}_{\alpha} = \mathit{true}$ if and only if for all $a \in \mathcal{A}_s$, $[F]^{\mathcal{A}}_{\alpha[x \mapsto a]} = \mathit{true}$.

The truth value $[F]^{\mathcal{A}}_{\alpha}$ of a sentence F does not depend on the valuation mapping α. Thus, we often write $[F]^{\mathcal{A}}$ or just $F^{\mathcal{A}}$ instead. A valuation α *satisfies* F in $\mathcal{A}$ (written $\mathcal{A} \models F\ [\alpha]$) if $[F]^{\mathcal{A}}_{\alpha} = \mathit{true}$. We then say that F is *satisfiable*. If $\mathcal{A} \models F\ [\alpha]$ for *all* valuations α, we write $\mathcal{A} \models F$ and say that $\mathcal{A}$ is a *model* of F or that F is *true* in $\mathcal{A}$. We say that $\mathcal{A}$ is *a model of a set of sentences* $\mathcal{S}$ (written $\mathcal{A} \models \mathcal{S}$) if for all $F \in \mathcal{S}$, $\mathcal{A} \models F$. Given a sentence F, we write $\mathcal{S} \models F$ iff $\mathcal{A} \models F$ holds for *all models* $\mathcal{A}$ of $\mathcal{S}$.

3 Well-Founded Relations Characterize Operational Termination

In this section we characterize operational termination of a theory $\mathcal{S}$ as the well-foundedness of a binary relation $\Uparrow_{\mathcal{S}}$ on formulas which we call the *proof progress* relation. This relation is defined by considering all possible well-formed proof trees associated to $\mathcal{S}$. Essentially, two formulas F and F' are related by $\Uparrow_{\mathcal{S}}$ (written $F \Uparrow_{\mathcal{S}} F'$) if F' is introduced by an inference rule which has been used in an attempt to prove F by means of a well-formed proof tree.

Definition 3 (Proof Progress Relation) Let $\mathcal{S}$ be a theory. The binary *proof progress* relation $\Uparrow_{\mathcal{S}}$ on $\mathit{Form}(\mathcal{S})$ is given as follows: for all $F, F' \in \mathit{Form}(\mathcal{S})$, we write $F \Uparrow_{\mathcal{S}} F'$ if there is a well-formed proof tree T^F of the form

$$\frac{T_1 \quad \cdots \quad T_{i-1} \quad T_i \quad O_{i+1} \quad \cdots \quad O_n}{F}\,(\rho) \tag{2}$$

for some (renamed variant of an) inference rule $\rho : \frac{B_1 \cdots B_n}{A}$ and substitution σ, where (i) $F = \sigma(A)$, (ii) $\mathit{root}(T_i) = F' = \sigma(B_i)$ for some $1 \leq i \leq n$ (T_i could be just an open goal), (iii) $T_1, \ldots, T_{i-1}$ are closed proof trees rooted by $\sigma(B_1), \ldots, \sigma(B_{i-1})$ respectively, and (iv) $O_{i+1}, \ldots, O_n$ are open goals $\sigma(B_{i+1}), \ldots, \sigma(B_n)$ respectively.

Example 5 For program `INF` in Figure 1 and the well-formed proof tree sketched in Figure 3 we have $\mathtt{a} \to \mathtt{b} \Uparrow \mathtt{a} : \mathtt{S}$ and (considering the leftmost infinite subtree, which is also well-formed) we also have $\mathtt{a} : \mathtt{S} \Uparrow \mathtt{a} \to \mathtt{b}$.

Given a set A, a binary relation $R \subseteq A \times A$ on A is called *well-founded* if there is no infinite sequence $(a_i)_{i \geq 1}$ of elements $a_i \in A$ such that $a_i\, R\, a_{i+1}$ for all $i \geq 1$. We have the following characterization of operational termination of a theory.

Theorem 1 *A theory $\mathcal{S}$ is operationally terminating iff $\Uparrow_{\mathcal{S}}$ is well-founded.*

Proof We proceed by contradiction. For the *if* part, assume that $\Uparrow_{\mathcal{S}}$ is well-founded but $\mathcal{S}$ is not operationally terminating. Then, there is an infinite well-formed proof tree with spine $(\langle \rho_i, m_i \rangle)_{i \geq 1}$ where for all $i \geq 1$ and rules $\rho_i : \frac{B^i_1 \cdots B^i_{n_i}}{A^i}$, we have closed proof trees T^i_j with $\mathit{root}(T^i_j) = \sigma(B^i_j)$ for all $1 \leq j < m_i$. Thus, we obtain an

infinite sequence $(\sigma(B^i_{m_i}))_{i\geq 1}$ where $\sigma(B^i_{m_i}) \Uparrow_{\mathcal{S}} \sigma(B^{i+1}_{m_{i+1}})$ holds for all $i \geq 1$. This contradicts well-foundedness of $\Uparrow_{\mathcal{S}}$.

For the *only if* part, if $\Uparrow_{\mathcal{S}}$ is not well-founded, then there is an infinite sequence $(F^j)_{j\geq 1}$ such that $F^j \Uparrow_{\mathcal{S}} F^{j+1}$ for all $j \geq 1$. Then, for all $i \geq 1$, each comparison $F^j \Uparrow_{\mathcal{S}} F^{j+1}$ implies the existence of trees S^i of the form

$$\frac{T^i_1 \quad \cdots \quad T^i_{m_i-1} \quad T^i_{m_i} \quad O^i_{m_i+1} \quad \cdots \quad O^i_{n_i}}{F^i}\,(\rho_i)$$

with the correspondences between components of the segment and components of the inference rule as in Definition 3. Note that, accordingly, for all $i \geq 1$, $root(T^i_{m_i}) = \sigma(B^i_{m_i}) = \sigma(A^{i+1})$ and $F^i = \sigma(A^i)$. Thus, we obtain an infinite well-formed proof tree T^∞ as a sequence $(U_i)_{i\geq 0}$ of finite well-formed proof trees U_i, where U_0 is G and for all $i \geq 0$, U_{i+1} is obtained from U_i by replacing $T^i_{m_i}$ by S^{i+1} (for the special case $i = 0$, assume that $T^0_{m_0}$ is G) using the fact that $root(T^i_{m_i}) = F^{i+1}$. This contradicts operational termination of $\mathcal{S}$. □

Example 6 (Continuing Example 5) Since $\mathtt{a} \to \mathtt{b} \Uparrow \mathtt{a} : \mathtt{S}$ and $\mathtt{a} : \mathtt{S} \Uparrow \mathtt{a} \to \mathtt{b}$, we conclude that $\Uparrow$ is *not* well-founded. By Theorem 1, program `INF` in Figure 1 is not operationally terminating.

Theorem 1 proves operational termination of a theory $\mathcal{S}$ equivalent to the well-foundedness of $\Uparrow_{\mathcal{S}}$. The relation $\Uparrow_{\mathcal{S}}$ is obtained by examining all well-formed proof trees for $\mathcal{S}$. Since this is not affordable in practice, in Section 4 we use the notion of *proof jump* from [28] to obtain a better approach. The next section, though, briefly discusses the use of well-formed proof trees (that impose a specific order to prove the goals introduced by the inference rules) in the analysis of operational termination of computational systems which do not fit such an evaluation scheme.

3.1 About the Left-to-Right Development of Well-Formed Proof Trees

The notion of a well-formed proof tree implies that whenever an inference rule $\rho : \frac{B_1 \cdots B_n}{A}$ is used to prove a goal G such that $G = \sigma(A)$, the proof obligations $\sigma(B_i)$ for $1 \leq i \leq n$ are tried from left to right, i.e., starting from $\sigma(B_1)$, then $\sigma(B_2)$, and so on, until reaching $\sigma(B_n)$. This is a reasonable order that most implementations of programming languages or computational systems would naturally follow [26]. Other proof schemes could be used though. However, for the purpose of operational termination analysis, our techniques would also apply, possibly after an appropriate transformation of the inference system to cope with the considered proof strategy. We justify this claim with some examples. A proof strategy which nondeterministically selects one of the goals $\sigma(B_i)$ to continue a proof after using rule ρ could be simulated by using well-formed proof trees with an extended inference system where all variants of ρ defined by $\rho_\pi : \frac{B_{\pi(1)} \cdots B_{\pi(n)}}{A}$ for π a permutation of $(1, \ldots, n)$ are considered. More restricted cases, like a right-to-left strategy would be faithfully simulated by using simpler transformations like, e.g., using a rule $\rho' : \frac{B_n \cdots B_1}{A}$ for each rule ρ as above.

The extended inference system for the nondeterministic case sketched above would also cover the use of parallel strategies (simultaneously exploiting several goals $\sigma(B_j)_{j\in J}$ for some $J \subseteq \{1, \ldots, n\}$ with more than one index), provided that the notion of operational termination still requires that all proof trees are finite.

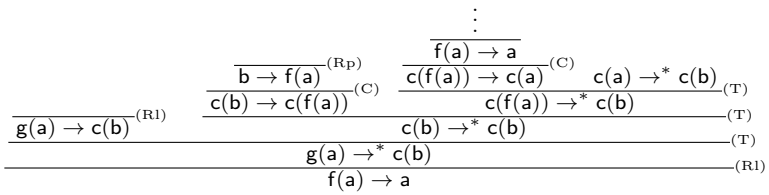


Fig. 7 Infinite well-formed proof tree (see [27, Example 2])

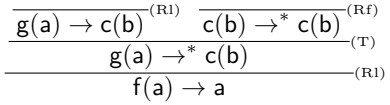
More sophisticated proof strategies could lead to a terminating behavior even for operationally nonterminating systems. For instance, the following CTRS $\mathcal{R}$:

$$\mathsf{g(a)} \to \mathsf{c(b)} \tag{3}$$

$$\mathsf{b} \to \mathsf{f(a)} \tag{4}$$

$$\mathsf{f}(x) \to x \Leftarrow \mathsf{g}(x) \to \mathsf{c}(y) \tag{5}$$

is not operationally terminating due to the infinite well-formed proof tree in Figure 7. However a proof strategy that prioritizes the use of rule (Rf) over (T) would prove $\mathsf{c(b)} \to^* \mathsf{c(b)}$ in the second proof obligation after the first application of (T), thus avoiding the infinite development above:



Under such a strategy, the CTRS would be operationally terminating (modulo the strategy). This is not surprising, as it is well-known that, in term rewriting, and in particular when TRSs are considered, the use of a particular rewriting strategy (e.g., innermost [3], leftmost-outermost [32], context-sensitive [21], etc.) often leads to improve the termination behavior of TRSs with respect to the unrestricted rewriting computations. We expect a similar situation in our setting. Investigating these issues, though, is beyond the scope of this paper.

4 Comparing Proof Jumps with Well-Founded Relations

In [28] we have introduced the *OT Framework* for proving operational termination. The central notion is that of a *proof jump*. Given a theory $\mathcal{S}$, a proof jump ψ for $\mathcal{I}(\mathcal{S})$ (or just $\mathcal{S}$) is a pair $(A \Uparrow \mathbf{B}_m)$, where $m \geq 1$ and $A, B_1, \ldots, B_m$ are formulas. Here, A is called the *head* of the proof jump; B_m is the *hook*; and $B_1, \ldots, B_{m-1}$ is called the *conditional part* of the proof jump. This terminology is motivated below. Given an inference rule $\rho : \frac{B_1 \cdots B_n}{A}$ (or $\frac{\mathbf{B}_n}{A}$ for short) with label ρ and $1 \leq m \leq n$, $[\rho]^m$ denotes the m-th proof jump $A \Uparrow B_1, \ldots, B_m$ which is obtained from ρ. The set of proof jumps of $\mathcal{I}(\mathcal{S})$ is $\mathcal{J}_\mathcal{S} = \{[\rho]^m \mid \rho : \frac{\mathbf{B}_n}{A} \in \mathcal{I}(\mathcal{S}), 1 \leq m \leq n\}$.

Example 7 The proof jumps for $\mathcal{I}(\mathtt{INF})$ in Figure 2 are

$$[SR]^1\ x : \mathtt{S} \Uparrow x \to y \qquad [SR]^2\ x : \mathtt{S} \Uparrow x \to y,\ y : \mathtt{S} \qquad [M1_{\mathtt{S<T}}]^1\ x :: \mathtt{T} \Uparrow x :: \mathtt{S}$$
$$[M2_{\mathtt{S}}]^1\ x : \mathtt{S} \Uparrow x :: \mathtt{S} \qquad [M2_{\mathtt{T}}]^1\ x : \mathtt{T} \Uparrow x :: \mathtt{T} \qquad [Rl]^1\ \mathtt{a} \to \mathtt{b} \Uparrow \mathtt{a} : \mathtt{S}$$
$$[T]^1\ x \to^* z \Uparrow x \to y \qquad [T]^2\ x \to^* z \Uparrow x \to y,\ y \to^* z$$

Example 8 The proof jumps for rules $(SR)_Z$ and $(M1)_{\mathtt{mb}}$ in Figure 6 are:

$$[(SR)_Z]^1\ x : \mathtt{Zero} \Uparrow x \to y \qquad [(SR)_Z]^2\ x : \mathtt{Zero} \Uparrow x \to y,\ y : \mathtt{Zero}$$
$$[(M1)_{\mathtt{mb}}]^1\ \mathtt{s}(\mathtt{s}(\mathtt{s}(\mathtt{M3}))) :: \mathtt{3} * \mathtt{Nat} \Uparrow \mathtt{M3} :: \mathtt{3} * \mathtt{Nat}$$

Proof jumps keep track of the spine $(\langle \rho_i, m_i \rangle)_{i \geq 1}$ of infinite well-formed proof trees. Assume that $\rho : \frac{B_1 \cdots B_n}{A}$ is such that $\rho = \rho_i$ for some $i \in \mathbb{N}$ and let $m = m_i$. Then, $[\rho]^m$ indicates (i) where the progress of the infinite behavior is made (by means of A and B_m, establishing the links with the previous and next inference rules in the spine) and (ii) the provability context which is assumed for such a progress (we can assume $B_1, \ldots, B_{m-1}$ in the conditional part provable, after instantiation). Accordingly, A and B_m are called the *head* and the *hook*[8] of $[\rho]^m$, respectively.

An infinite $(\mathcal{S}, \mathcal{J})$-*chain* is a sequence of (renamed versions of) proof jumps $(A^i \Uparrow \mathbf{B}^i_{m_i}) \in \mathcal{J}$ for $i \geq 1$ together with a substitution σ such that, for all $i \geq 1$, $\sigma(B^i_{m_i}) = \sigma(A^{i+1})$ and for all j, $1 \leq j < m_i$, $\mathcal{S} \vdash \sigma(B^i_j)$. A theory $\mathcal{S}$ is operationally terminating iff there is no infinite $(\mathcal{S}, \mathcal{J}_{\mathcal{S}})$-chain [28, Theorem 1].

4.1 Feasible Rules and Proof Jumps

The use of a proof jump in an $(\mathcal{S}, \mathcal{J})$-chain requires proofs of the goals in the conditional part of the proof jump. In this setting, the following definition is relevant:

Definition 4 (Feasibility) A formula F is $\mathcal{S}$-*feasible* (or just *feasible* if no confusion arises) if there is a substitution σ such that $\mathcal{S} \vdash \sigma(F)$; otherwise, we call it *infeasible*. An inference rule $\rho : \frac{B_1 \cdots B_n}{A}$ is *feasible* if there is a substitution σ that makes each of the $\sigma(B_1), \ldots, \sigma(B_n)$ provable, thus enabling the use of ρ in a proof of $\sigma(A)$; we call ρ infeasible otherwise. A proof jump $A \Uparrow \mathbf{B}_m$ is feasible if there is a substitution σ that makes each of the $\sigma(B_1), \ldots, \sigma(B_{m-1})$ provable (we call it infeasible otherwise).

Infeasibility of a proof jump $[\rho]^m$ implies infeasibility of ρ (but not vice versa). Feasibility is, in general, undecidable. In the following, we introduce a sufficient criterion for proving infeasibility of OSFO-formulas. This can be used to prove infeasibility of inference rules and proof jumps. If such a checking succeeds, then the rule or proof jump is discarded. Otherwise, we consider them feasible.

4.1.1 Proving Infeasibility as First-Order Satisfiability

In proofs of infeasibility we have to deal with provability with respect to $\mathcal{I}(\mathcal{S})$ for a theory $\mathcal{S}$. In our method, we use the correspondence between provability and

[8] We use 'hook' because this formula is intended to 'catch' the head of the next inference rule in the spine.

satisfiability which is provided by the notion of *correctness* of a proof calculus with respect to the semantic interpretation of the logic. We did not provide semantic notions for general logics, but Section 2.4 provides the standard semantic notions for OSFOL. In this section we restrict the attention to OSFOL.

When dealing with inference rules $\frac{B_1 \ \cdots \ B_n}{A}$ where $A, B_1, \ldots, B_n$ are OSFO-formulas, we treat them as *sentences* $(\forall \mathbf{x} : \mathbf{s})\ B_1 \wedge \cdots \wedge B_n \Rightarrow A$, where $\mathbf{x} = x_1, \ldots, x_m$ consists of the free variables occurring in $A, B_1, \ldots, B_n$. The OSFO-theory which is obtained from $\mathcal{I}(\mathcal{S})$ when the inference rules are treated in this way is denoted as $\overline{\mathcal{S}}$.

Example 9 For $\mathcal{I}(\texttt{INF})$ in Figure 2, the theory $\overline{\texttt{INF}}$ is as follows (we use sort KT to represent the kind [T]):

$$\begin{array}{lll}
\texttt{a} :: \texttt{T} & (\forall x, y : KT)\ x \to y \wedge y : \texttt{S} \Rightarrow x : \texttt{S} & (\forall x : KT)\ x :: \texttt{S} \Rightarrow x :: \texttt{T} \\
\texttt{b} :: \texttt{T} & (\forall x : KT)\ x :: \texttt{S} \Rightarrow x : \texttt{S} & (\forall x : KT)\ x :: \texttt{T} \Rightarrow x : \texttt{T} \\
\texttt{a} : \texttt{S} \Rightarrow \texttt{a} \to \texttt{b} & (\forall x, y, z : KT)\ x \to y \wedge y \to^* z \Rightarrow x \to^* z & (\forall x : KT)\ x \to^* x
\end{array}$$

Example 10 For $\mathcal{I}(3 * \texttt{NAT})$ in Figure 6, $\overline{3 * \texttt{NAT}}$ is as follows (KN is the kind [Nat]):

$$\begin{array}{rr}
(\forall x, y : KN)\ x \to y \wedge y : \texttt{Zero} \Rightarrow x : \texttt{Zero} & (\forall x, y : KN)\ x \to y \wedge y : 3 * \texttt{Nat} \Rightarrow x : 3 * \texttt{Nat} \\
(\forall x, y : KN)\ x \to y \wedge y : \texttt{Nat} \Rightarrow x : \texttt{Nat} & (\forall x : KN)\ x :: \texttt{Zero} \Rightarrow x :: 3 * \texttt{Nat} \\
(\forall x : KN)\ x :: 3 * \texttt{Nat} \Rightarrow x :: \texttt{Nat} & \texttt{zero} :: \texttt{Zero} \\
(\forall x : KN)\ x :: \texttt{Nat} \Rightarrow \texttt{s}(x) :: \texttt{Nat} & (\forall \texttt{M3} : KN)\ \texttt{M3} :: 3 * \texttt{Nat} \Rightarrow \texttt{s}(\texttt{s}(\texttt{s}(\texttt{M3}))) :: 3 * \texttt{Nat} \\
(\forall x : KN)\ x :: \texttt{Zero} \Rightarrow x : \texttt{Zero} & (\forall x : KN)\ x :: 3 * \texttt{Nat} \Rightarrow x : 3 * \texttt{Nat} \\
(\forall x : KN)\ x :: \texttt{Nat} \Rightarrow x : \texttt{Nat} & (\forall x, y : KN)\ x \to y \Rightarrow \texttt{s}(x) \to \texttt{s}(y) \\
(\forall x : KN)\ x \to^* x & (\forall x, y, z : KN)\ x \to y \wedge y \to^* z \Rightarrow x \to^* z
\end{array}$$

Deductions with $\overline{\mathcal{S}}$ proceed in the usual predicate calculus *à la Hilbert* by using *modus ponens* and *generalization* as inference rules, the usual set of *logical axioms*, and $\overline{\mathcal{S}}$ as the set of *proper* axioms [30, Section 2.3]. In this way, formulas F that can be proved with $\mathcal{I}(\mathcal{S})$ by using proof trees (which we denote $\mathcal{S} \vdash F$) can also be proved with $\overline{\mathcal{S}}$ using Hilbert's style (which we write $\overline{\mathcal{S}} \vdash_H F$). By *correctness* of the first-order predicate calculus, for all models $\mathcal{A}$ of a theory $\overline{\mathcal{S}}$, whenever $\overline{\mathcal{S}} \vdash_H F$ holds for a sentence F, we have $\mathcal{A} \models F$. We use these facts in the following result.

Theorem 2 *Let $\mathcal{S}$ be an OSFO theory with inference system $\mathcal{I}(\mathcal{S})$, $F(\mathbf{x} : \mathbf{s})$ be a first-order formula, with free variables $x_1, \ldots, x_n$ of sorts $s_1, \ldots, s_n$, respectively, and $\mathcal{A}$ be a structure where $\mathcal{A}_s$ is non-empty for all sorts s. If $\mathcal{A} \models \overline{\mathcal{S}} \cup \{\neg(\exists \mathbf{x} : \mathbf{s})\, F(\mathbf{x} : \mathbf{s})\}$, then F is $\mathcal{S}$-infeasible.*

Proof By contradiction. If F is $\mathcal{S}$-feasible, then there is a substitution σ such that $\mathcal{S} \vdash \sigma(F(\mathbf{x} : \mathbf{s}))$ holds. Thus, $\overline{\mathcal{S}} \vdash_H \sigma(F(\mathbf{x} : \mathbf{s}))$ holds as well. Since $\mathcal{A}$ is a model of $\overline{\mathcal{S}}$, by *correctness* of the predicate calculus we have $\mathcal{A} \models (\forall \mathbf{y} : \mathbf{s'})\ \sigma(F(\mathbf{x} : \mathbf{s}))$. Here, $y_1, \ldots, y_m$ are the variables of sorts $s'_1, \ldots, s'_m$ (written $\mathbf{y} : \mathbf{s'}$ for short) occurring in $\sigma(F(\mathbf{x} : \mathbf{s}))$. Hence, for all valuations ν of the variables $\mathbf{y}$, the interpretation in $\mathcal{A}$ of the universally quantified formula above, i.e., $[\sigma(F)]^{\mathcal{A}}_{\nu}$, is *true*. Since for all sorts s, $\mathcal{A}_s$ is not empty, for each valuation ν of the variables $\mathbf{y}$ there is a valuation ν' of the variables $\mathbf{x}$ given by $\nu'(x) = [\sigma(x)]^{\mathcal{A}}_{\nu}$ for all variables x in $\mathbf{x}$, such that $[F(\mathbf{x} : \mathbf{s})]^{\mathcal{A}}_{\nu'}$ is true. This contradicts the assumption $\mathcal{A} \models \neg(\exists \mathbf{x} : \mathbf{s})\, F(\mathbf{x} : \mathbf{s})$. □

In the following, we use Theorem 2 together with existing tools for the (semi)automatic generation of models (e.g., AGES [17] or Mace4 [29]) in proofs of infeasibility.

Remark 1 Dealing with OSFOL, an inference rule $\frac{B_1 \cdots B_n}{A}$ is feasible iff $B_1 \wedge \cdots \wedge B_n$ is feasible (and similarly for proof jumps). Thus, we can use Theorem 2 to prove infeasibility at once by seeking a model of $\mathcal{S} \cup \{\neg(\exists \mathbf{x} : \mathbf{s})\ B_1 \wedge \cdots \wedge B_n\}$, where $\mathbf{x}$ consists of the free variables occurring in $B_1, \ldots, B_n$ (with the corresponding sorts).

Example 11 We prove rule (Rl) in Figure 2 for program `INF`, i.e.,

$$\frac{\mathtt{a} : \mathtt{S}}{\mathtt{a} \rightarrow \mathtt{b}}$$

infeasible by using Theorem 2. For $\overline{\mathtt{INF}}$ in Example 9, we obtain a model $\mathcal{A}$ of $\overline{\mathtt{INF}} \cup \{\neg\, \mathtt{a} : \mathtt{S}\}$ with AGES (see Appendix A for details). The domain is $\mathcal{A}_{KT} = \{1\}$; function and predicate symbols are interpreted as follows:

$$\mathtt{a}^{\mathcal{A}} = 1 \qquad \mathtt{b}^{\mathcal{A}} = 1 \qquad \_ : \mathtt{S}^{\mathcal{A}}(x) \Leftrightarrow \mathit{false} \qquad \_ :: \mathtt{S}^{\mathcal{A}}(x) \Leftrightarrow \mathit{false}$$
$$\_ : \mathtt{T}^{\mathcal{A}}(x) \Leftrightarrow \mathit{true} \qquad \_ :: \mathtt{T}^{\mathcal{A}}(x) \Leftrightarrow \mathit{true} \qquad x \rightarrow^{\mathcal{A}} y \Leftrightarrow x + y \geq 1 \qquad x(\rightarrow^*)^{\mathcal{A}} y \Leftrightarrow \mathit{true}$$

Note that $[Rl]^1$ in Example 7 is not infeasible! However, infeasibility of (Rl) makes $[T]^2$ infeasible too. This is proved by using Theorem 2. We obtain a model $\mathcal{A}$ of $\overline{\mathtt{INF}} \cup \{\neg(\exists x : T)(\exists y : T)\ x \rightarrow y\}$ with AGES. Now, $\mathcal{A}_{KT} = \mathbb{Z} - \mathbb{N}$ and

$$\mathtt{a}^{\mathcal{A}} = -1 \qquad \mathtt{b}^{\mathcal{A}} = -1 \qquad \_ : \mathtt{S}^{\mathcal{A}}(x) \Leftrightarrow \mathit{false} \qquad \_ :: \mathtt{S}^{\mathcal{A}}(x) \Leftrightarrow \mathit{false}$$
$$\_ : \mathtt{T}^{\mathcal{A}}(x) \Leftrightarrow \mathit{true} \qquad \_ :: \mathtt{T}^{\mathcal{A}}(x) \Leftrightarrow \mathit{true} \qquad x \rightarrow^{\mathcal{A}} y \Leftrightarrow \mathit{false} \qquad x(\rightarrow^*)^{\mathcal{A}} y \Leftrightarrow \mathit{true}$$

Example 12 With regard to the program `3*NAT`, Theorem 2 can be used to prove proof jumps $[(SR)_Z]^2$, $[(SR)_{3N}]^2$, $[(SR)_N]^2$, and $[(T)_N]^2$ infeasible. Since they share the same conditional part $x \rightarrow y$, infeasibility follows if we obtain a model $\mathcal{A}$ of $\overline{\mathtt{3*NAT}} \cup \{\neg(\exists x : KN)(\exists y : KN)\ x \rightarrow y\}$, for $\overline{\mathtt{3*NAT}}$ in Example 10. We obtain such a model with Mace4 (see Appendix B for details). The domain is $\mathcal{A}_{KN} = \{0, 1\}$. Function symbols are interpreted as $\mathtt{zero}^{\mathcal{A}} = \mathtt{s}^{\mathcal{A}}(x) = 0$. Predicates are all interpreted as *true* except $\rightarrow$, which is interpreted as *false*. Therefore, the rules $(SR)_Z$, $(SR)_{3N}$, $(SR)_N$, and $(T)_L$ are infeasible too. Moreover, infeasibility of $x \rightarrow y$ implies infeasibility of $(C)_{\mathtt{s},1}$ as well.

Definition 5 (OT problem/Initial OT problem) A pair $\tau = (\mathcal{S}, \mathcal{J})$ consisting of a theory $\mathcal{S}$ and a set of proof jumps for $\mathcal{S}$ is called an OT problem. The OT problem $\tau_I = (\mathcal{S}, \mathcal{J}_{\mathcal{S}})$ is called the *initial* OT problem.

Remark 2 (Use of feasible rules and proof jumps) According to the definition of an $(\mathcal{S}, \mathcal{J})$-chain, we can restrict the attention to the subsets $\mathcal{S}^+$ and $\mathcal{J}^+$ of $\mathcal{S}$-feasible inference rules and proof jumps. The sets of $(\mathcal{S}, \mathcal{J})$-chains and $(\mathcal{S}^+, \mathcal{J}^+)$-chains coincide. In particular, the initial problem $\tau_I = (\mathcal{S}, \mathcal{J}_{\mathcal{S}})$ can be taken as $\tau_I = (\mathcal{S}^+, \mathcal{J}^+_{\mathcal{S}})$. Note, however, that using the set $\mathcal{J}_{\mathcal{S}^+}$ of proof jumps of the $\mathcal{S}$-feasible inference rules $\mathcal{S}^+$ in the initial problem could lead to an incorrect approach, as the sets of $(\mathcal{S}, \mathcal{J}_{\mathcal{S}})$-chains and $(\mathcal{S}, \mathcal{J}_{\mathcal{S}^+})$-chains may differ. For instance, rule (Rl) in $\mathcal{I}(\mathtt{INF})$ was proved infeasible in Example 11. If we remove it from $\mathcal{I}(\mathtt{INF})$ before computing $\mathcal{J}_{\mathtt{INF}}$, then $[Rl]^1$ would not be obtained and the (spine of the) infinite well-formed proof tree in Figure 3 would not be captured by any $(\mathtt{INF}^+, \mathcal{J}_{\mathtt{INF}^+})$-chain.

Example 13 The initial OT problem for program $3 * \mathtt{NAT}$ is $(3 * \mathtt{NAT}^+, \mathcal{J}^+_{3*\mathtt{NAT}})$, where, according to Example 12, we take:

$$\begin{aligned} 3 * \mathtt{NAT}^+ &= \{(M1)_{\mathtt{Z<3N}}, (M1)_{\mathtt{3N<N}}, (M1)_{\mathtt{zero}}, (M1)_{\mathtt{s}}, (M1)_{\mathtt{mb}}, (M2)_Z, (M2)_{3N}, (M2)_N, \\ &\quad (Rf)_N\} \\ \mathcal{J}^+_{3*\mathtt{NAT}} &= \{[(SR)_Z]^1, [(SR)_{3N}]^1, [(SR)_N]^1, [(M1)_{\mathtt{Z<3N}}]^1, [(M1)_{\mathtt{3N<N}}]^1, [(M1)_{\mathtt{s}}]^1, \\ &\quad [(M1)_{\mathtt{mb}}]^1, [(M2)_Z]^1, [(M2)_{3N}]^1, [(M2)_N]^1 [(C)_{\mathtt{s},1}]^1, [(T)_L]^1\} \end{aligned}$$

Note that rule $(C)_{\mathtt{s},1}$ is not included in $3 * \mathtt{NAT}^+$ (due to its infeasibility), but the proof jump $[(C)_{\mathtt{s},1}]^1$ is included in $\mathcal{J}^+_{3*\mathtt{NAT}}$ as it is not infeasible and we compute the proof jumps with respect to the whole set of inference rules for 3*NAT.

In the following, we often assume the use of $\mathcal{S}^+$ and $\mathcal{J}^+_{\mathcal{S}}$ in the initial OT problem τ_I without making the +-superscript explicit.

4.2 Use of Well-Founded Relations with Proof Jumps

Our next result characterizes operational termination of a theory by comparing (the head and the hook of) proof jumps using a well-founded relation.

Theorem 3 *A theory $\mathcal{S}$ is operationally terminating iff there is a well-founded relation $\sqsupset$ on Form($\mathcal{S}$) such that, for all $A \Uparrow \mathbf{B}_m \in \mathcal{J}_{\mathcal{S}}$,*

for all substitutions σ, if $\mathcal{S} \vdash \sigma(B_i)$ for all i, $1 \leq i < m$, then $\sigma(A) \sqsupset \sigma(B_m)$ (†)

Proof For the *if* part, assume that $\mathcal{S}$ is not operationally terminating. Then, by [28, Theorem 1] there is an infinite $(\mathcal{S}, \mathcal{J}_{\mathcal{S}})$-chain $(A^i \Uparrow \mathbf{B}^i_{m_i})_{i \geq 1}$ for some substitution σ such that, for all $i \geq 1$, if $\mathcal{S} \vdash \sigma(B^i_j)$ holds for all $1 \leq j < m_i$, then $\sigma(B^i_{m_i}) = \sigma(A^{i+1})$. By (†), we have $\sigma(A^i) \sqsupset \sigma(B^i_{m_i})$ for all $i \geq 1$ and, since $\sigma(B^i_{m_i}) = \sigma(A^{i+1})$ for all $i \geq 1$, we obtain an infinite sequence $(\sigma(A^i))_{i \geq 1}$ of formulas satisfying $\sigma(A^i) \sqsupset \sigma(A^{i+1})$ which contradicts well-foundedness of $\sqsupset$.

For the *only if* part, just consider $\Uparrow_{\mathcal{S}}$ in Definition 3. By Theorem 1, it is well-founded. Now consider a proof jump $\psi : A \Uparrow \mathbf{B}_m \in \mathcal{J}_{\mathcal{S}}$ and a substitution σ such that, for all $1 \leq i < m$, $\mathcal{S} \vdash \sigma(B_i)$, i.e., there are closed proof trees T_i with $root(T_i) = \sigma(B_i)$. By definition of proof jump, there is an inference rule $\frac{\mathbf{B}_n}{A} \in \mathcal{I}(\mathcal{S})$ for some $n \geq m$. There is a well-formed proof tree

$$\frac{T_1 \quad \cdots \quad T_{m-1} \quad \sigma(B_m) \quad \sigma(B_{m+1}) \quad \cdots \quad \sigma(B_n)}{\sigma(A)}(\rho)$$

By definition of $\Uparrow_{\mathcal{S}}$, we have $\sigma(A) \Uparrow_{\mathcal{S}} \sigma(B_m)$ as required. □

In order to use Theorem 3, we need to check that, for all proof jumps $A \Uparrow \mathbf{B}_m \in \mathcal{J}_{\mathcal{S}}$, statement (†) holds. The provability statements $\mathcal{S} \vdash \sigma(B_i)$, the presence of symbol $\sqsupset$, and the use of infinitely many substitutions σ prevent (†) from being a formal sentence of the language of $\mathcal{S}$. In the following section, we investigate how to transform (†) into a formal sentence. Then, we provide a new characterization of operational termination as the satisfiability of such transformed sentences.

5 Well-Founded Models and Operational Termination

In the following, we consider OSFO-theories $\mathcal{S}$ over a signature $\Omega = (S, \leq, \Sigma, \Pi)$.

5.1 A Signature Transformation

We define a transformation of OSFO-signatures which is parametric in OT problems $\tau = (\mathcal{S}, \mathcal{J})$. The main goal is implementing the comparisons of (instances of) the head A and hook B_m which are specified in (†) for some proof jump $A \Uparrow \mathbf{B}_m \in \mathcal{J}$. For this purpose, the relation $\sqsupset$ is represented as a new binary predicate $\pi_{\sqsupset}$ on terms of the target signature Ω_τ. In this way, we can recast (†) as a logical formula:

$$(\forall \mathbf{x} : \mathbf{s})\ B_1 \wedge \cdots \wedge B_{m-1} \Rightarrow A^{\downarrow}\, \pi_{\sqsupset}\, B_m^{\downarrow} \tag{6}$$

where $\mathbf{x} : \mathbf{s}$ denotes the sequence $x_1 : s_1, \ldots, x_p : s_p$ of sorted variables referring to all free variables x_i (of sort s_i), for $1 \leq i \leq p$, occurring in $B_1, \ldots, B_m$ and A, and $\_^{\downarrow}$ transforms formulas in $Form_\Omega$ into Ω_τ-terms. Here we just define a requirement for $\_^{\downarrow}$ to ensure the main result in Section 5.4 (Theorem 4). A particular transformation $\_^{\downarrow}$ is defined in Section 5.2. Let us first define the target OSFO-signature Ω_τ.

Definition 6 Let $\Omega = (S, \leq, \Sigma, \Pi)$ be an OSFO-signature and τ be an OT problem. The OSFO-signature $\Omega_\tau = (S^\tau, \leq^\tau, \Sigma^\tau, \Pi^\tau)$ is defined as follows:

- $S_\tau = S \cup \{\varsigma_\tau\}$ where ς_τ is a fresh sort symbol.
- $\leq_\tau$ is $\leq$ viewed as a relation on S_τ, i.e., no new subsort relation is assumed.
- $\Pi_\tau = \Pi \cup \Pi_{\varsigma_\tau \varsigma_\tau}$ where $\Pi_{\varsigma_\tau \varsigma_\tau}$ is a set of new predicate symbols (e.g., $\pi_{\sqsupset}$).

We let the composition of Σ_τ depend on the specific transformation $\_^{\downarrow}$ in use.

Note that (†) only requires the comparison of instances of the head and hook of proof jumps by means of the well-founded relation $\sqsupset$. Given an OT problem $\tau = (\mathcal{S}, \mathcal{J})$, we let $H_\tau = \{A, B_m \mid A \Uparrow \mathbf{B}_m \in \mathcal{J}\}$ be the set of heads and hooks of the proof jumps in τ. The only requirement we impose to a transformation $\_^{\downarrow}$ is the following:

Definition 7 (Stable transformation) Let Ω be an OSFO-signature, τ be an OT problem, and Ω_τ as above. A transformation $\_^{\downarrow} : Form(\Omega) \to (\mathcal{T}_{\Sigma_\tau}(\mathcal{X}))_{\varsigma_\tau}$ is *stable* if for all substitutions σ and formulas $A \in H_\tau$,

$$\sigma(A)^{\downarrow} = \sigma(A^{\downarrow}) \tag{7}$$

5.2 A Stable Transformation

We complement Definition 6 with Σ^τ defined as follows:

$$\Sigma^\tau = \Sigma \cup \bigcup_{s \in S} \Sigma^K_{\lambda,s} \cup \Sigma_{\lambda,\varsigma_\tau} \cup \Sigma_{\varsigma_\tau,\varsigma_\tau} \cup \Sigma_{\varsigma_\tau\varsigma_\tau,\varsigma_\tau} \cup \bigcup_{w \in S^+} \Sigma_{w,\varsigma_\tau}$$

where

- $\Sigma^K_{\lambda,s}$ is a set of new constant symbols of sort s, i.e., for all $s \in S$, $\Sigma_{\lambda,s} \cap \Sigma^K_{\lambda,s} = \varnothing$.

- $\Sigma_{\lambda,\varsigma_\tau} = \{c_x \mid x \in \mathcal{X}\}$ is a set of constants of sort ς_τ associated to the variables in $\mathcal{X}$ (disregarding the sort of such variables).
- $\Sigma_{\varsigma_\tau,\varsigma_\tau} = \{f_\neg\}$, i.e., we add a new function symbol associated to the negation.
- $\Sigma_{\varsigma_\tau\varsigma_\tau,\varsigma_\tau} = \{f_\wedge, f_\vee\}$, i.e., we add new function symbols associated to each binary connective (and, or).
- $\Sigma_{w,\varsigma_\tau} = \{f_P : w \to \varsigma_\tau \mid P \in \Pi_w\}$, i.e., each (overloaded version of a) predicate symbol P with input sorts w is given a new function symbol $f_P : w \to \varsigma_\tau$ with input sorts w and output sort ς_τ.

The amount of symbols in $\Sigma^K_{\lambda,s}$ and $\Sigma_{\lambda,\varsigma_\tau}$ depends on the quantification of the formulas in H_τ, see item 3 in the following.

Definition 8 Let Ω be an OSFO-signature, τ be an OT problem, and Ω_τ be as in Definition 6 with Σ^τ as above. The transformation $\_^\downarrow : \mathit{Form}(\Omega) \to (\mathcal{T}_{\Sigma_\tau}(\mathcal{X}))_{\varsigma_\tau}$ is defined by induction on the structure of F:

1. $P(t_1,\ldots,t_k)^\downarrow = f_P(t_1,\ldots,t_k)$ if $P \in \Pi_w$ for some $w \in S^+$.
2. $(\neg F)^\downarrow = f_\neg(F^\downarrow)$; $(F_1 \wedge F_2)^\downarrow = f_\wedge(F_1^\downarrow, F_2^\downarrow)$; and $(F_1 \vee F_2)^\downarrow = f_\vee(F_1^\downarrow, F_2^\downarrow)$.
3. $((\forall x : s)\ F)^\downarrow = c_x$ for $c_x \in \Sigma_{\lambda,\varsigma_\tau}$ and $((\exists x : s)\ F)^\downarrow = \theta_{x,k}(F^\downarrow)$, where $\theta_{x,k} = \{x \mapsto k\}$ is a substitution for some $k \in \Sigma^K_{\lambda,s}$ which does not occur in F.

In the first two items of Definition 8, variables in logic formulas are kept to determine the final value of the expression. Also, the syntactical structure of boolean combinations of atoms is essentially preserved. Item (3) deserves some further explanation. Universally quantified formulas $(\forall x : s)\ F$ are supposed to have a fixed truth value (either true or false), which does not depend on the valuation of variables in F. For this reason, we transform it into a constant c_x.[9] Regarding existentially quantified formulas $(\exists x : s)\ F$ we give a skolemization-like treatment where the occurrences of the quantified variable x are replaced by a new constant of the same sort. Of course, the usual connection between universal and existential quantification (i.e., $(\forall x)F$ is equivalent to $\neg(\exists x)\ \neg F$) is not preserved by this transformation: we have $((\forall x)F)^\downarrow = c_x$ and $(\neg(\exists x)\ \neg F)^\downarrow = f_\neg(f_\neg(\sigma_k(F^\downarrow)))$. This is not important for our purpose. Note that, if H_τ consists of unquantified formulas only (as it is the case in most examples of this paper), then both $\Sigma^K_{\lambda,s}$ and $\Sigma_{\lambda,\varsigma_\tau}$ can be empty.

Proposition 1 (Stability) *Transformation $\_^\downarrow$ in Definition 8 is stable.*

Proof For the first two items above (atoms and connectives), it is clear. Regarding quantified formulas, we note that the application of a substitution σ to a quantified formula $(Q\ x : s)\ F$, for $Q \in \{\exists, \forall\}$ is naturally defined as $\sigma((Q\ x : s)\ F) = (Q\ x : s)\ \sigma_{\overline{x}}(F)$, where for all variables $y \neq x$, $\sigma_{\overline{x}}(y) = \sigma(y)$, but $\sigma_{\overline{x}}(x) = x$, i.e., no instantiation is allowed on the bound variable x. Then, we have the following:

- With regard to universal quantification, we have:

$$\begin{aligned}\sigma(((\forall x : s)\ F))^\downarrow &= ((\forall x : s)\ \sigma_{\overline{x}}(F))^\downarrow \\ &= c_x \\ &= \sigma(c_x) \\ &= \sigma(((\forall x : s)\ F)^\downarrow)\end{aligned}$$

[9] This transformation keeps no information about the matrix formula F in the universally quantified formula $(\forall x : s)\ F$. This could compromise the success of its use with Theorem 4 below. More precise transformations could be obtained by considering constants $c_{x,F}$ indexed not only by variables x but also by formulas F and envisaging appropriate conditions on such constants so that stability holds.

– With regard to existential quantification, we have:

$$\begin{aligned}
\sigma(((\exists x : s)\ F))^{\downarrow} &= ((\exists x : s)\ \sigma_{\overline{x}}(F))^{\downarrow} \\
&= \theta_{x,k}(\sigma_{\overline{x}}(F)^{\downarrow}) \\
&= \sigma_{\overline{x}}(\theta_{x,k}(F^{\downarrow})) \qquad (8) \\
&= \sigma_{\overline{x}}(((\exists x : s)\ F)^{\downarrow}) \\
&= \sigma(((\exists x : s)\ F)^{\downarrow}) \qquad (9)
\end{aligned}$$

where $\sigma_{\overline{x}}$ and $\theta_{x,k}$ are as above. Note that, since x is not instantiated by $\sigma_{\overline{x}}$ and $\theta_{x,k}$ only instantiates x to a constant symbol k, both substitutions commute, i.e., the order of application of these substitutions does not matter. This has been used in the induction step (8). Also note in (9) that the application of $\sigma_{\overline{x}}$ and σ on the existentially quantified formula coincide by definition of $\sigma_{\overline{x}}$. □

Example 14 The sentences (6) that correspond to proof jumps $[(SR)_Z]^1$ and $[(M1)_{\mathtt{mb}}]^1$ for the inference rules $(SR)_Z$ and $(M1)_{\mathtt{mb}}$ in Figure 6 (see Example 8) are as follows:

$$\begin{array}{ll}
(\forall x, y : KN) & f_{_:\mathtt{Zero}}(x)\ \pi_{\sqsupset}\ f_{\rightarrow}(x, y) \\
(\forall\, \mathtt{M3} : KN) & f_{_::3*\mathtt{Nat}}(\mathtt{s}(\mathtt{s}(\mathtt{s}(\mathtt{M3}))))\ \pi_{\sqsupset}\ f_{_::3*\mathtt{Nat}}(\mathtt{M3})
\end{array}$$

where $f_{\_:\mathtt{Zero}} : KN \rightarrow \varsigma_\tau$ and $f_{\rightarrow} : KN\ KN \rightarrow \varsigma_\tau$ are the new function symbols associated to predicates $\_ : \mathtt{Zero} \in \Pi_{KN}$ and $\rightarrow \in \Pi_{KN\,KN}$, respectively. Finally, $\pi_{\sqsupset} : \varsigma_\tau\varsigma_\tau$ is a new predicate symbol.

The following example illustrates the use of the transformation with quantified formulas.

Example 15 Consider the following instances of the (schemata of) the *elimination* inference rules $(\forall E)$ and $(\Rightarrow E)$ of the natural deduction system in [34, page 20] for a well-known example of reasoning:

$$\frac{(\forall x)\ H(x) \Rightarrow R(x)}{H(aristotle) \Rightarrow R(aristotle)} \qquad \frac{H(aristotle) \quad H(aristotle) \Rightarrow R(aristotle)}{R(aristotle)}$$

If the following axioms are added

$$H(aristotle) \qquad \text{and} \qquad (\forall x)\ H(x) \Rightarrow R(x)$$

we can prove $R(aristotle)$ as expected. The proof jumps for the inference rules are:

$$\begin{gathered}
H(aristotle) \Rightarrow R(aristotle) \Uparrow (\forall x)\ H(x) \Rightarrow R(x) \\
R(aristotle) \Uparrow H(aristotle) \\
R(aristotle) \Uparrow H(aristotle),\ H(aristotle) \Rightarrow R(aristotle)
\end{gathered}$$

The sentences (6) for these proof jumps (using $\neg A \vee B$ instead of $A \Rightarrow B$) are:

$$\begin{gathered}
f_{\vee}(f_{\neg}(f_H(aristotle)), f_R(aristotle))\ \pi_{\sqsupset}\ c_x \\
f_R(aristotle)\ \pi_{\sqsupset}\ f_H(aristotle) \\
H(aristotle) \Rightarrow f_R(aristotle)\ \pi_{\sqsupset}\ f_{\vee}(f_{\neg}(f_H(aristotle)), f_R(aristotle))
\end{gathered}$$

5.3 Interpretations for OSFOL and the Generation of Well-Founded Relations

Since Ω_τ is an extension of Ω, every Ω_τ-structure $\mathcal{A}$ is an Ω-structure. We use Ω_τ-structures $\mathcal{A}$ to define binary relations $\bowtie$ on Ω-*formulas* by associating a predicate symbol $\pi_\bowtie$ to $\bowtie$.

Definition 9 Let Ω be an OSFO-signature, τ be an OT-problem, and $\mathcal{A}$ be an Ω_τ-structure. Given $\pi_\bowtie \in \Pi_{\varsigma_\tau \varsigma_\tau}$, we define a relation $\bowtie$ on Ω-formulas as follows: for all Ω-formulas A and B, we write $A \bowtie B$ iff $\mathcal{A} \models A^\downarrow \, \pi_\bowtie \, B^\downarrow$ (equivalenty, $\mathcal{A} \models (\forall \mathbf{x} : \mathbf{s}) \; A^\downarrow \, \pi_\bowtie \, B^\downarrow$ where $\mathbf{x}$ consists of all free variables in A and B, of sorts $\mathbf{s}$).

The well-foundedness of $\sqsupset$ in (†) cannot be characterized at once in first-order logic [36, Section 5.1.4]. When using Definition 9 to define $\sqsupset$, we can guarantee its well-foundedness if $\pi_\sqsupset^{\mathcal{A}}$ is a well-founded relation.

Proposition 2 *Let Ω be an OSFO-signature, $\tau = (\mathcal{S}, \mathcal{J})$ be an OT problem, T_τ be the set of sorts of the free variables occurring in $\mathcal{J}$, and $\mathcal{A}$ be an Ω_τ-structure such that for all $s \in T_\tau$, $\mathcal{A}_s \neq \varnothing$. If $\pi_\sqsupset^{\mathcal{A}}$ is a well-founded relation on $\mathcal{A}_{\varsigma_\tau}$, then $\sqsupset$ as in Definition 9 is a well-founded relation on Form($\mathcal{S}$).*

Proof By contradiction. If there is an infinite sequence $(A_i)_{i \geq 1}$ of Ω-formulas such that for all $i \geq 1$ $A_i \sqsupset A_{i+1}$, then, by Definition 9, we have $\mathcal{A} \models (\forall \mathbf{x}) \; A_i^\downarrow \, \pi_\sqsupset \, A_{i+1}^\downarrow$ for all $i \geq 1$, i.e., for all valuations α, $([A_i^\downarrow]_\alpha^{\mathcal{A}}, [A_{i+1}^\downarrow]_\alpha^{\mathcal{A}}) \in \pi_\sqsupset^{\mathcal{A}}$. Since the set of valuations α is not empty (because $\mathcal{A}_s \neq \varnothing$ for all sorts $s \in T_\tau$ of the variables occurring in τ), there is an infinite sequence $([A_i^\downarrow]_\alpha^{\mathcal{A}})_{i \geq 1}$ for some valuation α that contradicts well-foundedness of $\pi_\sqsupset^{\mathcal{A}}$. □

5.4 A Characterization of Operational Termination Using Interpretations

Now we are ready to provide a characterization of operational termination by interpretation with well-founded models, i.e., structures $\mathcal{A}$ where some binary predicates $\pi \in \Pi_{s\,s}$ are required to be interpreted as well-founded relations $\pi^{\mathcal{A}}$ on $\mathcal{A}_s$. In the following result, $\overline{\mathcal{S}}$ is the theory obtained from $\mathcal{I}(\mathcal{S})$ as explained in Section 4.1.1. Also note that τ_I is the initial OT problem $(\mathcal{S}, \mathcal{J}_\mathcal{S})$ (Definition 5).

Theorem 4 *Let $\Omega = (S, \leq, \Sigma, \Pi)$ be an OSFO-signature such that for all $s \in S$, $\mathcal{T}_{\Sigma s} \neq \varnothing$. A theory $\mathcal{S}$ is operationally terminating if and only if there is an Ω_{τ_I}-structure $\mathcal{A}$ with no empty domain and a stable transformation $\_^\downarrow$ such that (i) $\mathcal{A} \models \overline{\mathcal{S}}$, (ii) for all $\psi : A \Uparrow \mathbf{B}_m \in \mathcal{J}_\mathcal{S}$, $\mathcal{A} \models (\forall \mathbf{x} : \mathbf{s}) \; B_1 \wedge \cdots \wedge B_{m-1} \Rightarrow A^\downarrow \, \pi_\sqsupset \, B_m^\downarrow$ and (iii) $\pi_\sqsupset^{\mathcal{A}}$ is a well-founded relation.*

Proof For the *if* part, we use Theorem 3 to prove $\mathcal{S}$ operationally terminating. Consider an arbitrary proof jump $A \Uparrow \mathbf{B}_m \in \mathcal{J}_\mathcal{S}$ and a substitution σ such that $\mathcal{S} \vdash \sigma(B_i)$ holds for all $1 \leq i < n$. By (i) and by correctness of the first-order calculus, we have $\mathcal{A} \models \sigma(B_i)$ for all $1 \leq i < n$. By (*ii*), we have $\mathcal{A} \models \sigma(A^\downarrow) \, \pi_\sqsupset \, \sigma(B_m^\downarrow)$. By stability of $\_^\downarrow$, we have $\mathcal{A} \models \sigma(A)^\downarrow \, \pi_\sqsupset \, \sigma(B_m)^\downarrow$. Thus, for $\sqsupset$ given as in Definition 9, we have $\sigma(A) \sqsupset \sigma(B_m)$. Since the domains of the interpretation are not empty and due to (iii), by Proposition 2, $\sqsupset$ is well-founded. Thus, by Theorem 3, $\mathcal{S}$ is operationally terminating.

For the *only if* part, assume $\mathcal{S}$ operationally terminating and consider the stable transformation $\_^\downarrow$ in Section 5.2. Consider the Ω_{τ_I}-structure $\mathcal{A}$ defined as follows:

1. For each sort $s \in S_{\tau_I}$, $\mathcal{A}_s$ is the (nonempty) set $\mathcal{T}_{\Sigma s}$ of ground terms of sort s; for all $w \in S^*_{\tau_I}$ and $s \in S_{\tau_I}$, each function symbol $f : w \to s$ is interpreted as the function $f^{\mathcal{A}}$ mapping $\mathbf{t} \in \mathcal{T}_{\Sigma w}$ into $f(\mathbf{t}) \in \mathcal{T}_{\Sigma s}$ (note that this also includes the interpretations of the new symbols $f_P, f_\wedge, \ldots$, introduced in Σ^τ), and each predicate symbol $P : w$ with $w \in S^+$ (i.e., taking arguments from the 'old' sorts only) is interpreted as the relation $P^{\mathcal{A}} = \{\mathbf{t} \mid \mathbf{t} \in \mathcal{T}_{\Sigma w}, \mathcal{S} \vdash P(\mathbf{t})\}$ containing the tuples $\mathbf{t}$ of ground terms in $\mathcal{T}_{\Sigma w}$ such that $P(\mathbf{t})$ can be proved in $\mathcal{S}$.
2. Now we define the interpretation of $\pi_\sqsupset$, the only predicate symbol which does not take arguments from the old sorts. By Theorem 3, operational termination of $\mathcal{S}$ implies the existence of a well-founded relation $\sqsupset$ among formulas such that (†) holds. Then, $\pi_\sqsupset : s_{\tau_I} s_{\tau_I}$ is interpreted by using $\sqsupset$ and $\_^\downarrow$ as follows:

   $$\pi^{\mathcal{A}}_\sqsupset = \{(\sigma(A)^\downarrow, \sigma(B_m)^\downarrow) \mid A \Uparrow \mathbf{B}_m \in \mathcal{J}_{\mathcal{S}} \wedge \sigma(A) \sqsupset \sigma(B_m) \text{ for the substitution } \sigma\}$$

   Since $\sqsupset$ is well-founded, $\pi^{\mathcal{A}}_\sqsupset$ is also well-founded on $\mathcal{A}_{s_{\tau_I}}$, i.e., (iii) holds.

Now, (i) holds by construction of $\mathcal{A}$. With regard to (ii), consider a proof jump $\psi : A \Uparrow \mathbf{B}_m \in \mathcal{J}_{\mathcal{S}}$ and a valuation in the structure $\mathcal{A}$ (i.e., a ground (S-sorted) substitution σ of the variables in ψ). If $\mathcal{A} \models \sigma(B_i)$ for all $1 \leq i < n$, then, by definition of $\mathcal{A}$, we have $\mathcal{S} \vdash \sigma(B_i)$. Therefore, by (†) and by definition of $\sqsupset$ we have $\sigma(A) \sqsupset \sigma(B_{n_i})$. Hence, by definition of $\pi^{\mathcal{A}}_\sqsupset$, we have $(\sigma(A)^\downarrow, \sigma(B_n)^\downarrow) \in \pi^{\mathcal{A}}_\sqsupset$ and, by stability of $\_^\downarrow$, $(\sigma(A^\downarrow), \sigma(B^\downarrow_m)) \in \pi^{\mathcal{A}}_\sqsupset$. Thus, we have (ii), as required. □

5.5 Automatic Generation of Well-Founded Models

In practice, when using Theorem 4 to prove operational termination of $\mathcal{S}$, we will equivalently seek a structure $\mathcal{A}$ with no empty domain which is a model of

$$\overline{\mathcal{S}} \cup \{(\forall \mathbf{x} : \mathbf{s})\, B_1 \wedge \cdots \wedge B_{m-1} \Rightarrow A^\downarrow \,\pi_\sqsupset\, B^\downarrow_m \mid A \Uparrow \mathbf{B}_m \in \mathcal{J}_{\mathcal{S}}\} \tag{10}$$

and such that $\pi^{\mathcal{A}}_\sqsupset$ is a well-founded relation. Such models can be obtained from model generators like AGES or Mace4. AGES interprets the sort, function, and predicate symbols of an OSFO-theory as parametric domains, functions and predicates. Some binary predicates can be required to be interpreted by a well-founded relation. Sentences in the theory are transformed into constraints over the parameters which are then solved by using standard constraint solving methods and tools [24].

Example 16 Consider the program 3*NAT in Figure 5. Operational termination of 3*NAT can be proved using AGES to find a model of the corresponding set of sentences (10). A model $\mathcal{A}$ is automatically obtained with domains $\mathcal{A}_{KN} = \mathbb{N} \cup \{-1\}$ and $\mathcal{A}_{\varsigma_\tau} = \mathbb{N}$, function symbols are interpreted as follows:

$$\begin{array}{lll} \mathtt{zero}^{\mathcal{A}} = -1 & \mathtt{s}^{\mathcal{A}}(x) = x + 1 & \\ f^{\mathcal{A}}_{\_:\mathtt{Zero}}(x) = 4x + 5 & f^{\mathcal{A}}_{\_:\mathtt{3*Nat}}(x) = 4x + 6 & f^{\mathcal{A}}_{\_:\mathtt{Nat}}(x) = 4x + 7 \\ f^{\mathcal{A}}_{\_::\mathtt{Zero}}(x) = x + 1 & f^{\mathcal{A}}_{\_::\mathtt{3*Nat}}(x) = 2x + 3 & f^{\mathcal{A}}_{\_::\mathtt{Nat}}(x) = 3x + 5 \\ f^{\mathcal{A}}_{\to}(x, y) = 3x + 3 & f^{\mathcal{A}}_{\to^*}(x, y) = 4x + y + 6 & \end{array}$$

with variables ranging on the domains interpreting the corresponding sorts. Note the new functions $f_{\_:\mathtt{Zero}}$ $f_{\_:\mathtt{3*Nat}}$, $f_{\_:\mathtt{Nat}}$, $f_{\_::\mathtt{Zero}}$, $f_{\_::\mathtt{3*Nat}}$, $f_{\_::\mathtt{Nat}}$, $f_\to$, and $f_{\to^*}$ introduced

by the transformation in Section 5.2. Finally, predicates are interpreted as follows:

$$\begin{array}{rrr} \_ : \mathtt{Zero}^{\mathcal{A}}(x) \Leftrightarrow \mathit{true} & \_ : \mathtt{Pal}^{\mathcal{A}}(x) \Leftrightarrow \mathit{true} & \_ : \mathtt{List}^{\mathcal{A}}(x) \Leftrightarrow \mathit{true} \\ \_ :: \mathtt{Zero}^{\mathcal{A}}(x) \Leftrightarrow \mathit{true} & \_ :: \mathtt{Pal}^{\mathcal{A}}(x) \Leftrightarrow \mathit{true} & \_ :: \mathtt{List}^{\mathcal{A}}(x) \Leftrightarrow \mathit{true} \\ x \rightarrow^{\mathcal{A}} y \Leftrightarrow \mathit{true} & x (\rightarrow^*)^{\mathcal{A}} y \Leftrightarrow \mathit{true} & x \, \pi_{\sqsupset}^{\mathcal{A}} \, y \Leftrightarrow x > y \end{array}$$

In order to illustrate the proof technique, consider the proof jumps $[(SR)_Z]^1$ and $[(M1)_{\mathtt{mb}}]^1$ in Example 8 (note that $[(SR)_Z]^2$ in Example 8 was discarded from τ_I by infeasibility, see Example 12). Their transformed versions, obtained in Example 14, are, respectively, as follows:

$$(\forall x, y : KN) \quad f_{\_:\mathtt{Zero}}(x) \; \pi_{\sqsupset} \; f_{\rightarrow}(x, y) \tag{11}$$

$$(\forall \, \mathtt{M3} : KN) \quad f_{\_::3*\mathtt{Nat}}(\mathtt{s}(\mathtt{s}(\mathtt{s}(\mathtt{M3})))) \; \pi_{\sqsupset} \; f_{\_::3*\mathtt{Nat}}(\mathtt{M3}) \tag{12}$$

We have the following:

- For $[(SR)_Z]^1$, transformed into (11), we have

  $$[f_{\_:\mathtt{Zero}}(x)]^{\mathcal{A}} = 4x + 5 \qquad \textit{and} \qquad [f_{\rightarrow}(x, y)]^{\mathcal{A}} = 3x + 3$$

  By using the interpretation of $\pi_{\sqsupset}$, $[(\forall x, y : KN) \; f_{\_:\mathtt{Zero}}(x) \; \pi_{\sqsupset} \; f_{\rightarrow}(x, y)]^{\mathcal{A}}$ is

  $$(\forall x, y \in \mathbb{N} \cup \{-1\}) \; 4x + 5 > 3x + 3$$

  which is obviously true.
- For $[(M1)_{\mathtt{mb}}]^1$, transformed into (12), we have

  $$[f_{\_::3*\mathtt{Nat}}(\mathtt{s}(\mathtt{s}(\mathtt{s}(\mathtt{M3}))))]^{\mathcal{A}} = 2(\mathtt{M3} + 3) + 3 \quad \textit{and} \quad [f_{\_::3*\mathtt{Nat}}(\mathtt{M3})]^{\mathcal{A}} = 2\,\mathtt{M3} + 3$$

  Therefore, $[(\forall \, \mathtt{M3} : KN) \; f_{\_::3*\mathtt{Nat}}(\mathtt{s}(\mathtt{s}(\mathtt{s}(\mathtt{M3})))) \; \pi_{\sqsupset} \; f_{\_::3*\mathtt{Nat}}(\mathtt{M3})]^{\mathcal{A}}$ is

  $$(\forall \mathtt{M3} \in \mathbb{N} \cup \{-1\}) \; 2\,\mathtt{M3} + 9 > 2\,\mathtt{M3} + 3$$

  which, again, is true.

5.5.1 Use of Finite Domains

Mace4 computes (one-sorted) finite models only, but is very fast. There is no support for well-foundedness, though. However, we can use the fact that a finite relation R on a set A is well-founded iff R is not cyclic, i.e., there is no $a \in A$ such that $a \, R^+ \, a$. We instruct Mace4 to obtain a well-founded interpretation for $\pi_{\sqsupset}$ by adding the following sentences to (10):

$$(\forall x : \varsigma_\tau)(\forall y : \varsigma_\tau) \;\; x \, \pi_{\sqsupset} \, y \Rightarrow x \, \pi_{\sqsupset}^{+} \, y \tag{13}$$

$$(\forall x : \varsigma_\tau)(\forall y : \varsigma_\tau)(\forall z : \varsigma_\tau) \;\; x \, \pi_{\sqsupset} \, y \wedge y \, \pi_{\sqsupset}^{+} \, z \Rightarrow x \, \pi_{\sqsupset}^{+} \, z \tag{14}$$

$$\neg(\exists x : \varsigma_\tau) \;\; x \, \pi_{\sqsupset}^{+} \, x \tag{15}$$

where $\pi_{\sqsupset}^{+} : \varsigma_\tau \; \varsigma_\tau$ is a new predicate symbol.

Example 17 Consider the inference rules in Example 15. Operational termination can be proved using Mace4 to find a model of the corresponding set of sentences (10) plus (13)-(15). A model $\mathcal{A}$ is automatically obtained with domains $\mathcal{A}_{\mathrm{S}} = \mathcal{A}_{\varsigma_\tau} = \{0, 1, 2\}$ (for a *dummy* sort S) and function symbols interpreted as follows:

$$aristotle^{\mathcal{A}} = 0 \qquad c_x^{\mathcal{A}} = 0 \qquad f_H^{\mathcal{A}}(x) = 0$$

$$f_R^{\mathcal{A}}(x) = \begin{cases} 1 \text{ if } x = 0 \\ 0 \text{ otherwise} \end{cases} \qquad f_\vee^{\mathcal{A}}(x, y) = \begin{cases} 2 \text{ if } x = 0 \text{ and } y = 1 \\ 0 \text{ otherwise} \end{cases} \qquad f_\neg^{\mathcal{A}}(x) = 0$$

with variables ranging on the domains interpreting the corresponding sorts. Finally, predicates are interpreted as follows:

$$H^{\mathcal{A}}(x) \Leftrightarrow R^{\mathcal{A}}(x) \Leftrightarrow x = 0 \qquad x \, \pi_\sqsupset^{\mathcal{A}} \, y \Leftrightarrow ((x = 1 \vee x = 2) \wedge y = 0) \vee (x = 1 \wedge y = 2)$$

Note that $\pi_\sqsupset^{\mathcal{A}}$ is well-founded. Consider the last proof jump in Example 15, i.e.,

$$R(aristotle) \Uparrow H(aristotle), \; H(aristotle) \Rightarrow R(aristotle)$$

Let us show that $\mathcal{A}$ actually satisfies the corresponding (non-atomic) formula

$$H(aristotle) \Rightarrow f_R(aristotle) \; \pi_\sqsupset \; f_\vee(f_\neg(f_H(aristotle)), f_R(aristotle))$$

Since $[H(aristotle)]^{\mathcal{A}}$ is *true*, we need to prove that

$$[f_R(aristotle) \; \pi_\sqsupset \; f_\vee(f_\neg(f_H(aristotle)), f_R(aristotle))]^{\mathcal{A}}$$

is true. Since $[f_R(aristotle)]^{\mathcal{A}} = 1$ and $[f_\vee(f_\neg(f_H(aristotle)), f_R(aristotle))]^{\mathcal{A}} = 2$, and $(1, 2) \in \pi_\sqsupset^{\mathcal{A}}$, we obtain the desired conclusion.

6 The OT Framework and the Removal Pair Processor

As in the DP Framework for TRSs [12], proofs of operational termination in the OT Framework for general logics [28] are successively decomposed or simplified into smaller or simpler problems until (hopefully) reaching trivial problems to stop the process. In the DP Framework for TRSs, the notion of Dependency Pair [4] is the most crucial. Dependency pairs are just rewrite rules $u \to v$ that are associated to the rules $\ell \to r$ of a TRS $\mathcal{R}$. Each infinite rewrite sequence in $\mathcal{R}$ can be mapped into an infinite sequence $(u_i \to v_i)_{i \geq 1}$ (called a *DP chain*) of dependency pairs. Since dependency pairs focus on those parts of the rules $\ell \to r$ of $\mathcal{R}$ which may contribute to the nontermination behavior, the use of dependency pairs in termination analysis of TRSs usually leads to more efficient proofs. This is reflected in the fact that most termination tools for proving termination of TRSs (e.g., AProVE [14], MU-TERM [1], T$_{\mathsf{T}}$T$_2$ [19], etc.) use dependency pairs. In this automation process, the use of the DP Framework [13,12] is also essential. In the DP Framework for TRSs, proofs of termination proceed by transforming DP problems $\tau = (\mathcal{P}, \mathcal{R})$ where $\mathcal{P}$ and $\mathcal{R}$ are TRSs. A proof of termination starts with an *initial DP Problem* $(\mathsf{DP}(\mathcal{R}), \mathcal{R})$ whose first component $\mathsf{DP}(\mathcal{R})$ consists of all the dependency pairs of $\mathcal{R}$. Then, 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 $\{\tau_1, \ldots, \tau_n\}$ (alternatively, they can return "no"). 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 the OT Framework [28], an *OT problem* $\tau = (\mathcal{S}, \mathcal{J})$ is called *finite* if there is no infinite $(\mathcal{S}, \mathcal{J})$-chain. A theory $\mathcal{S}$ is operationally terminating iff the initial OT problem τ_I is finite. An *OT processor* P maps an OT problem into either a set of OT problems or the answer "no". A processor P is *sound* if for all OT problems τ, if $\mathsf{P}(\tau) \neq \mathsf{no}$ and all OT problems in $\mathsf{P}(\tau)$ are finite, then τ is finite. By repeatedly applying processors, we can construct a tree (called *OT tree*) for an OT problem τ whose nodes are labeled with OT problems or *"*yes*"* or *"*no*"*, and whose root is labeled with τ. For every inner node with label τ', there is a processor P satisfying one of the following: (i) $\mathsf{P}(\tau') = \mathsf{no}$ and τ' has just one child that is labeled with *"*no*"*; (ii) $\mathsf{P}(\tau') = \varnothing$ and τ' has just one child that is labeled with *"*yes*"*; or (iii) $\mathsf{P}(\tau') \neq \mathsf{no}$, $\mathsf{P}(\tau') \neq \varnothing$, and the children of τ' are labeled with the OT problems in $\mathsf{P}(\tau')$. If all leaves of an OT tree for τ are labeled with *"*yes*"* and all used processors are sound, then τ is finite [28, Theorem 3].

Remark 3 (OT Framework and DP Framework) The OT and DP Frameworks are similar in how proofs are organized (use of processors, construction of a proof tree, etc.). However, there is no notion of dependency pair for general logics (yet). In [23, Section 4.3] we show how proof jumps can be used to 'simulate' dependency pairs for TRSs by using a theory transformation. In this way, the improvements that dependency pairs usually bring are also available in the OT Framework. It is unclear, though, how to generalize such a transformational approach to arbitrary logics.

6.1 The Removal Pair Processor Revisited

In [28], six processors were proposed for their use in the OT Framework. One of them is the removal pair processor [28, Section 5.4]. A *removal pair* $(\gtrsim, \sqsupset)$, consists of binary relations $\gtrsim$ and $\sqsupset$ on formulas such that $\sqsupset$ is well-founded and $\gtrsim \circ \sqsupset \subseteq \sqsupset$ or $\sqsupset \circ \gtrsim \subseteq \sqsupset$. We can remove a proof jump $A \Uparrow \mathbf{B}_m \in \mathcal{J}$ from an OT problem $(\mathcal{S}, \mathcal{J})$ provided that the hook B_m is 'smaller' (w.r.t. $\sqsupset$) than the head A.

Definition 10 [28] Let $(\mathcal{S}, \mathcal{J})$ be an OT problem, $\psi : A \Uparrow \mathbf{B}_m \in \mathcal{J}$, and $(\gtrsim, \sqsupset)$ be a removal pair. Then, $\mathsf{P}_{RP}(\mathcal{S}, \mathcal{J}) = \{(\mathcal{S}, \mathcal{J} - \{\psi\})\}$ if and only if (i) for all $C \Uparrow \mathbf{D}_m \in \mathcal{J} - \{\psi\}$ and substitutions σ, if $\mathcal{S} \vdash \sigma(D_i)$ for all $1 \leq i < m$, then (i.1) $\sigma(C) \gtrsim \sigma(D_m)$ or (i.2) $\sigma(C) \sqsupset \sigma(D_m)$, and (ii) for all substitutions σ, if $\mathcal{S} \vdash \sigma(B_i)$ for all $1 \leq i < n$, then $\sigma(A) \sqsupset \sigma(B_m)$.

The practical use of P_{RP} poses the same problems discussed for Theorem 3. As done in Section 5, we transform the application of P_{RP} into a satisfiability problem.

Definition 11 (A semantic version of P_{RP}) Let $(\mathcal{S}, \mathcal{J})$ be an OT problem, $\mathcal{A}$ be an interpretation with no empty domain, and $\mathcal{J}_\sqsupset \subseteq \mathcal{J}$. Then, $\mathsf{P}_{RP}(\mathcal{S}, \mathcal{J}) = \{(\mathcal{S}, \mathcal{J} - \mathcal{J}_\sqsupset)\}$ if $\mathcal{A} \models \overline{\mathcal{S}}$, and the following conditions hold:

1. if $\mathcal{J} - \mathcal{J}_\sqsupset \neq \varnothing$, then $\mathcal{A}$ is a model of

$$\big((\forall xyz : \varsigma_\tau)(x\ \pi_\gtrsim\ y \wedge y\ \pi_\sqsupset\ z \Rightarrow x\ \pi_\sqsupset\ z)\big) \vee \big((\forall xyz : \varsigma_\tau)(x\ \pi_\sqsupset\ y \wedge y\ \pi_\gtrsim\ z \Rightarrow x\ \pi_\sqsupset\ z)\big)$$

2. for each $C \Uparrow \mathbf{D}_m \in \mathcal{J} - \mathcal{J}_\sqsupset$, there is $\pi_\bowtie \in \{\pi_\gtrsim, \pi_\sqsupset\}$ such that

$$\mathcal{A} \models (\forall \mathbf{x} : \mathbf{s}) \bigwedge_{i=1}^{m-1} D_i \Rightarrow C^\downarrow\ \pi_\bowtie\ D_m^\downarrow$$

3. $\pi_{\sqsupset}^{\mathcal{A}}$ is well-founded and for all $A \Uparrow \mathbf{B}_p \in \mathcal{J}_{\sqsupset}$,

$$\mathcal{A} \models (\forall \mathbf{x} : \mathbf{s})\ \bigwedge_{i=1}^{p-1} B_i \Rightarrow A^{\downarrow}\ \pi_{\sqsupset}\ B_p^{\downarrow}.$$

Example 18 Consider the CTRS $\mathcal{R}$ and associated inference system $\mathcal{I}(\mathcal{R})$ in Example 1. The theory $\overline{\mathcal{R}}$ is as follows (where S is a 'dummy' sort):

$$\begin{array}{ll} (\forall x : \mathrm{S})\ x \to^* x & (\forall x, y, z : \mathrm{S})\ x \to y \wedge y \to^* z \Rightarrow x \to^* z \\ (\forall x, y : \mathrm{S})\ x \to y \Rightarrow \mathsf{f}(x) \to \mathsf{f}(y) & (\forall x, y : \mathrm{S})\ x \to y \Rightarrow \mathsf{g}(x) \to \mathsf{g}(y) \\ \mathsf{a} \to \mathsf{b} & \mathsf{f}(\mathsf{a}) \to \mathsf{b} \\ (\forall x : \mathrm{S})\ \mathsf{f}(x) \to^* x \Rightarrow \mathsf{g}(x) \to \mathsf{g}(\mathsf{a}) & \end{array}$$

The initial OT problem is $\tau_I = (\mathcal{R}, \mathcal{J}_{\mathcal{R}}) = (\mathcal{R}, \{[T]^1, [T]^2, [C_{\mathsf{f}}]^1, [C_{\mathsf{g}}]^1, [Rl_3]^1\})$ for the proof jumps $[T]^1$, $[T]^2$, $[C_{\mathsf{f}}]^1$, $[C_{\mathsf{g}}]^1$, and $[Rl_3]^1$ obtained from the inference rules in Example 1. We use Definition 11 to remove $[T]^2$, i.e.,

$$x \to^* z \Uparrow x \to y,\ y \to^* z$$

from τ_I with Mace4 (we were unable to find a model to remove $[T]^2$ with AGES). Accordingly, we let $\mathcal{J}_{\sqsupset} = \{[T]^2\}$ in Definition 11 and thus we need to find a model of $\overline{\mathcal{R}}$ plus (i) the sentence in item 1 of Definition 11, together with (ii) the sentences for the proof jumps in $\mathcal{J} - \mathcal{J}_{\sqsupset} = \{[T]^1, [C_{\mathsf{f}}]^1, [C_{\mathsf{g}}]^1, [Rl_3]^1\}$ after applying the transformation in Section 5, i.e.,

$$\begin{array}{ll} (\forall x, y, z : \mathrm{S})\ f_{\to^*}(x, z)\ \pi_{\gtrsim}\ f_{\to}(x, y) & (\forall x, y : \mathrm{S})\ f_{\to}(\mathsf{f}(x), \mathsf{f}(y))\ \pi_{\gtrsim}\ f_{\to}(x, y) \\ (\forall x, y : \mathrm{S})\ f_{\to}(\mathsf{g}(x), \mathsf{g}(y))\ \pi_{\gtrsim}\ f_{\to}(x, y) & (\forall x : \mathrm{S})\ f_{\to}(\mathsf{g}(x), \mathsf{g}(\mathsf{a}))\ \pi_{\gtrsim}\ f_{\to^*}(\mathsf{f}(x), x) \end{array}$$

and, finally, (iii) the sentence corresponding to $[T]^2$:

$$(\forall x, y, z : \mathrm{S})\ x \to y \Rightarrow f_{\to^*}(x, z)\ \pi_{\sqsupset}\ f_{\to^*}(y, z)$$

In order to guarantee that $\pi_{\sqsupset}^{\mathcal{A}}$ is well-founded (so that we can actually remove $[T]^2$), we add the sentences (13)-(15) as explained in Section 5.5.1. We found a model $\mathcal{A}$ with domains $\mathcal{A}_{\mathrm{S}} = \mathcal{A}_{\varsigma_{\tau_I}} = \{0, 1, 2, 3\}$. Function symbols are interpreted as follows:

$$\begin{array}{lll} \mathsf{a}^{\mathcal{A}} = 0 & \mathsf{b}^{\mathcal{A}} = 1 & \mathsf{f}^{\mathcal{A}}(x) = (x + 2) \mod 4 \\ \mathsf{g}^{\mathcal{A}}(x) = x \mod 2 & f_{\to}^{\mathcal{A}}(x, y) = x \mod 2 & f_{\to^*}^{\mathcal{A}}(x, y) = x \mod 2 \end{array}$$

Predicate symbols are interpreted as follows:

$$\begin{array}{ll} \to^{\mathcal{A}} = \{(0,1), (0,3), (2,1), (2,3)\} & (\to^*)^{\mathcal{A}} = \{(x, x) \mid x \in \mathcal{A}\} \cup \to^{\mathcal{A}} \\ \pi_{\sqsupset}^{\mathcal{A}} = \{(0,1)\} & (\pi_{\gtrsim})^{\mathcal{A}} = \{(0,0), (1,1)\} \end{array}$$

Thus, $\mathsf{P}_{RP}(\tau_I) = \{\tau_1\}$ where $\tau_1 = (\overline{\mathcal{R}}, \{[T]^1, [C_{\mathsf{f}}]^1, [C_{\mathsf{g}}]^1, [Rl_3]^1\})$. Successive applications of P_{RP} (using Definition 11 with AGES), remove $[T^1]$, then $[Rl_3]^1$, followed by $[C_{\mathsf{f}}]^1$, and finally $[C_{\mathsf{g}}]^1$ to finish the proof of operational termination of $\mathcal{R}$.

7 Conclusions

We have characterized operational termination of a theory $\mathcal{S}$ by the existence of a well-founded relation $\uparrow\uparrow_{\mathcal{S}}$ on formulas which are obtained from all possible well-formed proof trees associated to a theory $\mathcal{S}$ (Theorem 1). This fundamental result is difficult to use in practice due to the need of considering infinitely many proof trees. Then, we have characterized operational termination of a theory $\mathcal{S}$ by the existence of a well-founded relation $\sqsupset$ on formulas which introduces a decrease from (any instance of) the head to the hook of the proof jumps associated to $\mathcal{S}$ (Theorem 3). Restricting the attention to feasible inference rules and proof jumps leads to a simpler treatment, as we deal with fewer inference rules and proof jumps. Feasibility is undecidable, but we provide a sufficient criterion for infeasibility of inference rules and proof jumps for first-order theories (Theorem 2). Rules and proof jumps that cannot be proved infeasible are safely considered feasible in proofs of operational termination. When dealing with first-order (specifications of) theories $\mathcal{S}$, we have shown how to use Theorem 3 by means of models of the first-order theory $\overline{\mathcal{S}}$ associated to the inference rules in $\mathcal{I}(\mathcal{S})$ which satisfy some additional conditions. This leads to a third characterization of operational termination of first-order theories $\mathcal{S}$ based on finding well-founded models, of a transformed theory extending $\overline{\mathcal{S}}$ (Theorem 4). Such well-founded models are just logical models where some binary predicates are required to be well-founded. The use of tools for the automatic generation of such models like AGES and Mace4 permits the automation of the proofs. Interestingly, both finite and infinite models were useful for this purpose.

The results in this paper are completely new (Theorem 2 generalizes a technique introduced in [25] for CTRSs). In [22, Section 5.2] the semantic version of P_{RP} was already proposed but the transformation $\_^{\downarrow}$ used there was limited to atomic formulas (see [22, Section 5.1]). Furthermore, such a notion of transformation was not used to characterize operational termination as done in Theorem 4. In this paper we have introduced a completely general and abstract notion of transformation which suffices to guarantee its practical use provided that the new property of stability (Definition 7) is fulfilled. Then, we have extended the transformation introduced in [22] to cope with arbitrary first-order formulas and prove it stable (Proposition 1). Finally, the examples in this paper, and their management with AGES and Mace4 are also new.

The results in this paper can also be used to develop a fully automatic tool for proving operational termination of computational systems that can be described by means of first-order theories (e.g., Maude programs) based on the OT Framework. The implementation of such a system is an important subject for future work. However, more research is necessary to obtain a more precise description of the termination behavior of sophisticated programming languages. For instance, rewriting modulo equational theories E is possible in Maude. The analysis of the termination behavior is challenging when modeling such kind of computations by means of an inference system because it may potentially include an inference system $\mathcal{I}_{EL}(E)$ for equational logic, cf. [20, Table 6.1]:

$$(E_{=})_{s,t}\ \frac{}{s = t} \qquad (Rf_{=})\ \frac{}{x = x} \qquad (S_{=})\ \frac{y = x}{x = y}$$

$$(T_{=})\ \frac{x = y \qquad y = z}{x = z} \qquad (C_{=})_{f}\ \frac{x_1 = y_1 \quad \cdots \quad x_k = y_k}{f(x_1, \ldots, x_k) = f(y_1, \ldots, y_k)}$$

with rules $(E_=)_{s,t}$ for each $(s,t) \in E$ and rules $(C_=)_f$ for each $f \in \Sigma$. Unfortunately, $\mathcal{I}_{EL}(E)$ is operationally nonterminating: by a repeated use of $(T_=)$ we easily obtain an infinite well-formed proof tree. Possible solutions to this problem could be obtained by restricting the attention to specific classes of well-formed proof trees as suggested in [23] (directed operational termination).

Acknowledgements. I thank the anonymous referees for their comments and suggestions, leading to many improvements in the paper.

References

1. B. Alarcón, R. Gutiérrez, S. Lucas, R. Navarro-Marset. Proving Termination Properties with MU-TERM. In *Proc. of AMAST'10*, LNCS 6486:201-208, Springer-Verlag, 2011.
2. L. Aguirre, N. Martí-Oliet, M. Palomino, and I. Pita. Sentence-Normalized Conditional Narrowing Modulo in Rewriting Logic and Maude. *Journal of Automated Reasoning*, 60(4):421-463, 2018.
3. T. Arts and J. Giesl. Proving Innermost Normalisation Automatically. In *Proc. of RTA'97*, LNCS 1232:157-171, Springer-Verlag, Berlin, 1997.
4. T. Arts and J. Giesl. Termination of Term Rewriting Using Dependency Pairs. *Theoretical Computer Science* 236(1–2):133–178, 2000.
5. F. Baader and T. Nipkow. Term Rewriting and All That. Cambridge University Press, 1998.
6. M. Clavel, F. Durán, S. Eker, P. Lincoln, N. Martí-Oliet, J. Meseguer, and C. Talcott. All About Maude – A High-Performance Logical Framework. LNCS 4350, Springer-Verlag, 2007.
7. F. Durán, S. Lucas, J. Meseguer. Methods for Proving Termination of Rewriting-based Programming Languages by Transformation. *Electronic Notes in Theoretical Computer Science*, 248:93-113, 2009.
8. F. Durán, S. Lucas, C. Marché, J. Meseguer, X. Urbain, Proving Operational Termination of Membership Equational Programs, *Higher-Order and Symbolic Computation* 21(1-2):59–88, 2008.
9. S. Falke and D. Kapur Operational Termination of Conditional Rewriting with Built-in Numbers and Semantic Data Structures. *Electronic Notes in Theoretical Computer Science*, 237:75–90, 2009.
10. R.W. Floyd. Assigning meanings to programs. *Mathematical aspects of computer science* 19:19-32, 1967.
11. J. Giesl and T. Arts. Verification of Erlang Processes by Dependency Pairs. *Applicable Algebra in Engineering, Communication and Computing* 12:39–72, 2001.
12. J. Giesl, R. Thiemann, P. Schneider-Kamp, and S. Falke. Mechanizing and Improving Dependency Pairs. *Journal of Automatic Reasoning* 37(3):155–203, 2006.
13. J. Giesl, R. Thiemann, and P. Schneider-Kamp. The Dependency Pair Framework: Combining Techniques for Automated Termination Proofs. In *Proc. of LPAR'04*, LNAI 3452:301–331, 2004.
14. J. Giesl, P. Schneider-Kamp, and R. Thiemann. AProVE 1.2: Automatic Termination Proofs in the Dependency Pair Framework. In *Proc. of IJCAR'06*, LNAI 4130:281–286, 2006.
15. J. Goguen and J. Meseguer. Models and Equality for Logical Programming. In *Proc. of TAPSOFT'87*, LNCS 250:1–22, 1987.
16. J. Goguen and J. Meseguer. Order-sorted algebra I: Equational deduction for multiple inheritance, overloading, exceptions and partial operations. *Theoretical Computer Science*, 105:217–273, 1992.
17. R. Gutiérrez, S. Lucas, and P. Reinoso. A tool for the automatic generation of logical models of order-sorted first-order theories. In *Proc. of PROLE'16*, pages 215–230, 2016.
18. W. Hodges. Model Theory. Cambridge University Press, 1993.
19. M. Korp, C. Sternagel, H. Zankl, and A. Middeldorp. Tyrolean Termination Tool 2. In *Proc. of RTA 2009*, LNCS 5595:295-304, 2009.
20. R. Lalement. Computation as Logic. Masson-Prentice Hall International, 1993.

21. S. Lucas. Context-sensitive rewriting strategies. *Information and Computation*, 178(1):294–343, 2002.
22. S. Lucas. Use Of Logical Models For Proving Operational Termination In General Logics. In *Selected papers from WRLA'16*, LNCS 9942:1–21, 2016.
23. S. Lucas. Directions of Operational Termination. In *Proc. of PROLE'18*, `http://hdl.handle.net/11705/PROLE/2018/009`, 2018.
24. S. Lucas and R. Gutiérrez. Automatic Synthesis of Logical Models for Order-Sorted First-Order Theories. *Journal of Automated Reasoning* 60(4):465–501, 2018.
25. S. Lucas and R. Gutiérrez. Use of logical models for proving infeasibility in term rewriting. *Information Processing Letters*, 136:90-95, 2018.
26. S. Lucas, C. Marché, and J. Meseguer. Operational termination of conditional term rewriting systems. *Information Processing Letters* 95:446–453, 2005.
27. S. Lucas and J. Meseguer. Dependency pairs for proving termination properties of conditional term rewriting systems. *Journal of Logical and Algebraic Methods in Programming*, 86:236-268, 2017.
28. S. Lucas and J. Meseguer. Proving Operational Termination Of Declarative Programs In General Logics. In *Proc. of PPDP'14*, pages 111–122, ACM Digital Library, 2014.
29. W. McCune Prover9 & Mace4. `http://www.cs.unm.edu/~mccune/prover9/`, 2005–2010.
30. E. Mendelson. Introduction to Mathematical Logic. Fourth edition. Chapman & Hall, 1997.
31. J. Meseguer. General Logics. In *Logic Colloquium'87*, pages 275-329, 1989.
32. M.J. O'Donnell. Equational Logic as a Programming Language. The MIT Press, Cambridge, Massachusetts, 1985.
33. E. Ohlebusch. *Advanced Topics in Term Rewriting*. Springer-Verlag, 2002.
34. D. Prawitz. Natural Deduction. A Proof Theoretical Study. Almqvist & Wiksell, 1965. Reprinted by Dover Publications (2006).
35. G. Rosu, A. Stefanescu, S. Ciobaca, and B.M. Moore. One-Path Reachability Logic. In *Proc. of LICS 2013*, pages 358-367, IEEE Press, 2013.
36. S. Shapiro. Foundations without Foundationalism: A Case for Second-Order Logic. Clarendon Press, 1991.
37. F. Schernhammer and B. Gramlich. Characterizing and proving operational termination of deterministic conditional term rewriting systems. *Journal of Logic and Algebraic Programming* 79:659-688, 2010.
38. T. Serbanuta and G. Rosu. Computationally Equivalent Elimination of Conditions. In *Proc. of RTA'06*, LNCS 4098:19-34, Springer-Verlag, Berlin, 2006.
39. A.M. Turing, Checking a Large Routine. In *Report of a Conference on High Speed Automatic Calculating Machines*, Univ. Math. Lab., Cambridge, pp. 67-69, 1949.

A Infeasibility of Proof Jumps for `INF` Proved with AGES (Example 11)

Example 11 claims for infeasibility of rule (*Rl*) in Figure 2, i.e., of

$$\frac{\mathtt{a} : \mathtt{S}}{\mathtt{a} \rightarrow \mathtt{b}}$$

We use AGES to find a model of $\overline{\mathtt{INF}} \cup \{\neg\, \mathtt{a} : \mathtt{S}\}$.

AGES *specification.*

```
mod INF is
  sort KT .
  ops a b : -> KT .

  op isT : KT -> Bool . *** Predicate _:T
  op mbT : KT -> Bool . *** Predicate _::T
  op isS : KT -> Bool . *** Predicate _:S
  op mbS : KT -> Bool . *** Predicate _::S
  op redKT : KT KT -> Bool . *** ->[T]
  op redsKT : KT KT -> Bool . *** ->*[T]
endm
```

AGES *goal.* Note: universal quantification is implicit in AGES.

```
***
*** Inference rules
***

*** SR
redKT(x:KT,y:KT) /\ isS(y:KT) => isS(x:KT)

*** M1_S<T
mbS(x:KT) => mbT(x:KT)

*** M1_a

mbT(a)

*** M1_b

mbT(b)

*** M2_S
mbS(x:KT) => isS(x:KT)

*** M2_T
mbT(x:KT) => isT(x:KT)

*** Rf
```

```
redsKT(x:KT,x:KT)

*** T
redKT(x:KT,y:KT) /\ redsKT(y:KT,z:KT) => redsKT(x:KT,z:KT)

*** Rl
isS(a) => redKT(a,b)

*** GOAL:

~isS(a)
```

AGES *output.*

```
KT: {1}

Function Interpretations:
|[a]| = 1
|[b]| = 1

Predicate Interpretations:
isS(x_1_1:KT) <=> (0 >= x_1_1:KT)
isT(x_1_1:KT) <=> (0 >= 0)
mbS(x_1_1:KT) <=> (0 >= 1 + x_1_1:KT)
mbT(x_1_1:KT) <=> (1 + x_1_1:KT >= 0)
redKT(x_1_1:KT,x_2_1:KT) <=> (x_1_1:KT + x_2_1:KT >= 1)
redsKT(x_1_1:KT,x_2_1:KT) <=> (1 + x_1_1:KT + x_2_1:KT >= 0)
```

Example 11 also claims infeasibility of $[T]^2$, i.e., of

$$x \to^* z \Uparrow x \to y,\ y \to^* z$$

We use AGES to find a model of $\overline{\mathtt{INF}} \cup \{\neg\,(\exists x)(\exists y)\, x \to y\}$. The specification is the same as for the previous example. The goal is also the same except for

```
*** GOAL: We equivalently use (\forall x, y) ~(x -> y)

~(redKT(x:KT,y:KT))
```

AGES *output.*

```
Domains:
KT: -|N \ {0}

Function Interpretations:
|[a]| =  - 1
|[b]| =  - 1

Predicate Interpretations:
isS(x_1_1:KT) <=> (x_1_1:KT >= 1)
isT(x_1_1:KT) <=> (0 >= 2+2.x_1_1:KT)
```

```
mbS(x_1_1:KT) <=> (x_1_1:KT >= 1)
mbT(x_1_1:KT) <=> (0 >= 0)
redKT(x_1_1:KT,x_2_1:KT) <=> (1 + x_1_1:KT + x_2_1:KT >= 0)
redsKT(x_1_1:KT,x_2_1:KT) <=> (1+2.x_2_1:KT >= 4.x_1_1:KT)
```

B Infeasibility of Proof Jumps of $3 * \mathtt{NAT}$ Proved with Mace4 (Example 12)

Example 12 claims for infeasibility of the proof jump $[(SR)_Z]^2$, i.e., of

$$x : \mathtt{Zero} \Uparrow x \to y,\ y : \mathtt{Zero}$$

We use Mace4 to find a model of $\overline{3 * \mathtt{NAT}} \cup \{\neg\, (\exists x, y)\, x \to y\}$. We use the following symbols:

```
% isZ as predicate '_:Zero'
% isN3 as predicate '_:KNat3'
% isN as predicate '_:KNat'
% mbZ as predicate '_::Zero'
% mbN3 as predicate '_::KNat3'
% mbN as predicate '_::KNat'
% redN as ->KNat
% redsN as ->*KNat
```

Mace4 *assumptions*

```
% Inference rules

% SRZ
redN(x,y) & isZ(y) -> isZ(x).
% SRN3
redN(x,y) & isN3(y) -> isN3(x).
% SRN
redN(x,y) & isN(y) -> isN(x).

% M1ZN3
mbZ(x) -> mbN3(x).
% M1N3N
mbN3(x) -> mbN(x).

% M1zero
mbZ(zero).

% M1s
mbN(x) -> mbN(s(x)).

% M1mb
mbN3(x) -> mbN3(s(s(s(x)))).
```

```
% M2Zero
mbZ(x) -> isZ(x).
% M2Nat3
mbN3(x) -> isN3(x).
% M2L
mbN(x) -> isN(x).

% Cs
redN(x,y) -> redN(s(x),s(y)).

% RL*
redsN(x,y).
% TL
redN(x,y) & redsN(y,z) -> redsN(x,z).
```

Mace4 *goal*

```
%
% Refutation goal (negated by default when written in the goal part)
%
exists x exists y redN(x,y).
```

Mace4 *output*

```
                    isN(0).          mbN(0).     - redN(0,0).
                    isN(1).          mbN(1).     - redN(0,1).
                                                 - redN(1,0).
      s(0) = 0.                      mbN3(0).    - redN(1,1).
      s(1) = 0.     isN3(0).         mbN3(1).
                    isN3(1).                       redsN(0,0).
                                     mbZ(0).       redsN(0,1).
                    isZ(0).          mbZ(1).       redsN(1,0).
                    isZ(1).                        redsN(1,1).
```