Use of Logical Models for Proving Operational Termination in General Logics*

Salvador Lucas

DSIC, Universitat Politècnica de València, Valencia, Spain
`slucas@dsic.upv.es`
`http://users.dsic.upv.es/~slucas/`

Abstract. A *declarative programming language* is based on some logic $\mathcal{L}$ and its operational semantics is given by a proof calculus which is often presented in a *natural deduction style* by means of inference rules. *Declarative programs* are theories $\mathcal{S}$ of $\mathcal{L}$ and *executing* a program is proving goals φ in the inference system $\mathcal{I}(\mathcal{S})$ associated to $\mathcal{S}$ as a particularization of the inference system of the logic. The usual soundness assumption for $\mathcal{L}$ implies that every *model* $\mathcal{A}$ of $\mathcal{S}$ also satisfies φ. In this setting, the *operational termination* of a declarative program is quite naturally defined as *the absence of infinite proof trees* in the inference system $\mathcal{I}(\mathcal{S})$. Proving operational termination of declarative programs often involves two main ingredients: (i) the generation of logical models $\mathcal{A}$ to *abstract* the program execution (i.e., the provability of specific goals in $\mathcal{I}(\mathcal{S})$), and (ii) the use of *well-founded relations* to guarantee the absence of *infinite branches* in proof trees and hence of infinite proof trees, possibly taking into account the information about provability encoded by $\mathcal{A}$. In this paper we show how to deal with (i) and (ii) in a uniform way. The main point is the synthesis of logical models where *well-foundedness* is a side requirement for some specific predicate symbols.

Keywords: Abstraction, Logical models, Operational Termination.

1 Introduction

A recent survey defines the *program termination problem* as follows [4]: "using only a *finite* amount of time, *determine* whether a given *program* will always finish *running* or could *execute* forever." Being an intuitively clear definition, some questions should be answered before *using* it: (Q1) What is a *program*? (Q2) What is *running/executing* a program? (Q3) How to *determine* the property (in *practice*!)? In *declarative programming*, early proposals about the use of logic as a *programming framework* provide answers to the first two questions: (A1) programs are *theories* $\mathcal{S}$ of a given logic $\mathcal{L}$; and (A2) executing a program $\mathcal{S}$ is proving a goal φ as a deduction in the inference system $\mathcal{I}(\mathcal{L})$ of $\mathcal{L}$, written $\mathcal{S} \vdash \varphi$ [15, Section 6].

* Partially supported by the EU (FEDER), Spanish MINECO TIN 2013-45732-C4-1-P and TIN2015-69175-C4-1-R, and GV PROMETEOII/2015/013.

Example 1. The following Maude program is a *Membership Equational Logic* specification [16] somehow *sugared*, as explained in [13]. Sort Node represents the nodes in a graph and sorts Edge and Path are intended to classify paths consisting of a single edge or many of them, respectively [3, pages 561-562]:

```
fmod PATH is
  sorts Node Edge Path .
  subsorts Edge < Path .
  ops source target : Edge -> Node .
  ops source target : Path -> Node .
  op _;_ : [Path] [Path] -> [Path] .
  var E : Edge .
  vars P Q R S : Path .
  cmb E ; P : Path if target(E) = source(P) .
  ceq (P ; Q) ; R = P ; (Q ; R)
     if target(P) = source(Q) /\ target(Q) = source(R) .
  ceq source(P) = source(E) if E ; S := P .
  ceq target(P) = target(S) if E ; S := P .
endfm
```

The *execution* of PATH is described as *deduction* of goals $t \to_{[s]} u$ (one-step *rewriting* for terms t, u with sorts in the *kind* $[s]$), $t \to^*_{[s]} u$ (many-step rewriting), or $t : s$ (*membership*: claims that term t is of sort s) using the inference system of the *Context-Sensitive Membership Rewriting Logic* [5] in Figure 1 (see also [13]). Here, a new kind [Truth] with a constant tt and a function symbol eq : [Node] [Node] -> [Truth] are added to deal with *equalities* like $\mathtt{target(E)} = \mathtt{source(P)}$ as *reachability conditions* $\mathtt{eq(target(E), source(P))} \to^* \mathtt{tt}$. And a new membership predicate $t :: s$ arises where terms t are not rewritten before checking its sort s. Also note that the *overloaded* functions source and target (which are used to describe edges in a graph by establishing their *source* and *target* nodes, respectively) receive a *single* rank [Path] -> [Node] and the different overloads are modeled as rules ($M1^E_{\mathsf{src}}$), ($M1^E_{\mathsf{tgt}}$), ($M1^P_{\mathsf{src}}$), and ($M1^P_{\mathsf{tgt}}$).

The notion of *operational termination* [11] (often abbreviated OT in the subsequent related notions and definitions) provides an appropriate *definition* of termination of declarative programs: a program $\mathcal{S}$ is operationally terminating if *there is no infinite proof tree* for any goal in $\mathcal{S}$. We have recently developed a *practical* framework for proving operational termination of declarative programs [14]. In our method, we first obtain the *proof jumps* $A \Uparrow B_1, \ldots, B_n$ associated to inference rules $\frac{B_1 \cdots B_n \cdots B_{n+p}}{A}$ in $\mathcal{I}(\mathcal{S})$ (where A, $B_1, \ldots, B_n, \ldots, B_{n+p}$ are logic formulas, $n > 0$, and $p \geq 0$). Proof jumps capture (infinite) paths in a proof tree T as sequences (*chains*) of proof jumps. A set of proof jumps τ is called an *OT problem*. We call it *finite* if there is no infinite chain of proof jumps taken from τ. The *initial* OT problem τ_I consists of all proof jumps obtained from the inference rules in $\mathcal{I}(\mathcal{S})$ as explained above. Thus, (A3) *determining* that $\mathcal{S}$ is operationally terminating is equivalent to proving τ_I *finite*. This answers Q3.

$$(SR_N)\quad \frac{t \to_{[Node]} u \quad u : Node}{t : Node} \qquad\qquad (SR_E)\quad \frac{t \to_{[Path]} u \quad u : Edge}{t : Edge}$$

$$(SR_P)\quad \frac{t \to_{[Path]} u \quad u : Path}{t : Path} \qquad\qquad (M1_P)\quad \frac{X :: Edge}{X :: Path}$$

$$(M1^{E}_{\mathsf{src}})\quad \frac{X :: Edge}{\mathsf{source}(X) :: Node} \qquad\qquad (M1^{E}_{\mathsf{tgt}})\quad \frac{X :: Edge}{\mathsf{target}(X) :: Node}$$

$$(M1^{P}_{\mathsf{src}})\quad \frac{X :: Path}{\mathsf{source}(X) :: Node} \qquad\qquad (M1^{P}_{\mathsf{tgt}})\quad \frac{X :: Path}{\mathsf{target}(X) :: Node}$$

$$(M2_N)\quad \frac{t :: Node}{t : Node} \qquad\qquad (M2_E)\quad \frac{t :: Edge}{t : Edge}$$

$$(M2_P)\quad \frac{t :: Path}{t : Path}$$

$$(R^*_N)\quad \frac{}{t \to^*_{[Node]} t} \qquad\qquad (R^*_P)\quad \frac{}{t \to^*_{[Path]} t}$$

$$(R^*_T)\quad \frac{}{t \to^*_{[Truth]} t} \qquad\qquad (T_N)\quad \frac{t \to_{[Node]} u \quad u \to^*_{[Node]} v}{t \to^*_{[Node]} v}$$

$$(T_P)\quad \frac{t \to_{[Path]} u \quad u \to^*_{[Path]} v}{t \to^*_{[Path]} v} \qquad\qquad (T_T)\quad \frac{t \to_{[Truth]} u \quad u \to^*_{[Truth]} v}{t \to^*_{[Truth]} v}$$

$$(C_{\mathsf{src}})\quad \frac{t \to_{[Path]} u}{\mathsf{source}(t) \to_{[Node]} \mathsf{source}(u)} \qquad\qquad (C_{\mathsf{tgt}})\quad \frac{t \to_{[Path]} u}{\mathsf{target}(t) \to_{[Node]} \mathsf{target}(u)}$$

$$(C_{\mathsf{sq}_1})\quad \frac{t \to_{[Path]} u}{t\, ;\, v \to_{[Path]} u\, ;\, v} \qquad\qquad (C_{\mathsf{sq}_2})\quad \frac{t \to_{[Path]} u}{v\, ;\, t \to_{[Path]} v\, ;\, u}$$

$$(C^{N}_{\mathsf{eq}_1})\quad \frac{t \to_{[Node]} u}{\mathsf{eq}(t, v) \to_{[Truth]} \mathsf{eq}(u, v)} \qquad\qquad (C^{N}_{\mathsf{eq}_2})\quad \frac{t \to_{[Node]} u}{\mathsf{eq}(v, t) \to_{[Truth]} \mathsf{eq}(v, u)}$$

$$(M1_{\_;\_})\quad \frac{E :: Edge \quad P :: Path \quad \mathsf{eq}(\mathsf{target}(E), \mathsf{source}(P)) \to^*_{[Truth]} \mathsf{tt}}{E; P :: Path}$$

$$(Re_1)\quad \frac{P :: Path \quad Q :: Path \quad R :: Path \quad \mathsf{eq}(\mathsf{target}(P), \mathsf{source}(Q)) \to^*_{[Truth]} \mathsf{tt} \quad \mathsf{eq}(\mathsf{target}(Q), \mathsf{source}(R)) \to^*_{[Truth]} \mathsf{tt}}{(P; Q); R \to_{[Path]} P; (Q; R)}$$

$$(Re_2)\quad \frac{E :: Edge \quad P :: Path \quad S :: Path \quad P \to^*_{[Path]} E; S}{\mathsf{source}(P) \to_{[Node]} \mathsf{source}(E)}$$

$$(Re_3)\quad \frac{E :: Edge \quad P :: Path \quad S :: Path \quad P \to^*_{[Path]} E; S}{\mathsf{target}(P) \to_{[Node]} \mathsf{target}(S)}$$

$$(Re_4)\quad \frac{N :: Node}{\mathsf{eq}(N, N) \to_{[Truth]} \mathsf{tt}}$$

Fig. 1. Inference rules $\mathcal{I}(\texttt{PATH})$ for `PATH`

The OT Framework provides an incremental proof methodology to *simplify* OT problems τ in a divide-and-conquer style to eventually prove termination of the program (Section 2). In order to *remove* proof jumps $\psi : A \Uparrow B_1, \ldots, B_n$ from τ we often use *well-founded relations*: if there is a well-founded relation $\sqsupset$ on *formulas* of the language of $\mathcal{S}$ such that, for all substitutions σ,

$$\text{if } \mathcal{S} \vdash \sigma(B_i) \text{ for all } i,\ 1 \leq i < n, \text{ then } \sigma(A) \sqsupset \sigma(B_n), \tag{1}$$

then we can *remove* ψ from τ to obtain a new OT problem τ' *whose finiteness implies that of* τ [14]. For the sake of *automation*, recasting (1) as follows:

$$\forall \boldsymbol{x}(B_1 \wedge \cdots \wedge B_{n-1} \Rightarrow A \sqsupset B_n) \tag{2}$$

would be interesting to apply theorem proving or semantic methods to prove (1). In [14] we anticipated that *logical models* are useful for this purpose.

In order to provide a general treatment of the aforementioned problems which is well-suited for automation, we need to focus on a sufficiently simple but still powerful logic which can serve to our purposes. In [6] *Order-Sorted First-Order Logic* (OS-FOL) is proposed as a sufficiently general and expressive framework to represent declarative programs, semantics of programming languages, and program properties (see Section 3). In [10] we show how to systematically generate models for OS-FOL theories by using the *convex polytopic domains* introduced in [12]. In Section 4 we extend the work in [10] to generate appropriate interpretations of *predicate symbols* that can be then used to synthesize a model for a given OS-FOL theory $\mathcal{S}$.

Unfortunately, even with $\mathcal{S}$ an OS-FOL theory, (2) *is not* a formula of the theory $\mathcal{S}$: the *new* predicate symbol $\sqsupset$ is *not* in the language of $\mathcal{S}$. And (2) is not *well-formed* because predicate $\sqsupset$ is applied to *formulas* A and B_n rather than *terms* as required in any first-order language. Section 5 shows how to solve this problem by using *theory* transformations. It also shows how to obtain *well-founded* relations when the general approach to generate interpretations of predicate symbols described in Section 4 is used. Section 6 illustrates the use of the new developments to prove operational termination of `PATH` in the OT Framework. Automation of the analysis is achieved by using AGES [8], a web-based tool that implements the techniques in [10] and also in this paper. Section 7 concludes.

2 The OT Framework for General Logics

A *logic* $\mathcal{L}$ is a quadruple $\mathcal{L} = (\mathit{Th}(\mathcal{L}), \mathit{Form}, \mathit{Sub}, \mathcal{I})$, where: $\mathit{Th}(\mathcal{L})$ is the class of *theories* of $\mathcal{L}$, *Form* maps each theory $\mathcal{S} \in \mathit{Th}(\mathcal{L})$ into a set $\mathit{Form}(\mathcal{S})$ of *formulas* of $\mathcal{S}$, *Sub* is a mapping sending each $\mathcal{S} \in \mathit{Th}(\mathcal{L})$ to its set $\mathit{Sub}(\mathcal{S})$ of *substitutions*, with a containment $\mathit{Sub}(\mathcal{S}) \subseteq [\mathit{Form}(\mathcal{S}) \to \mathit{Form}(\mathcal{S})]$.

Remark 1. In [14, Section 2] we further develop the generic notion of substitution we are dealing with. In this paper we focus on first-order theories where the notion of substitution is the usual one: a mapping from variables into terms which is extended to a mapping from terms (formulas) into terms (formulas) in the usual way.

Finally, $\mathcal{I}$ maps each $\mathcal{S} \in \mathit{Th}(\mathcal{L})$ into a subset $\mathcal{I}(\mathcal{S}) \subseteq \mathit{Form}(\mathcal{S}) \times \mathit{Form}(\mathcal{S})^*$, where each $(A, B_1 \ldots B_n) \in \mathcal{I}(\mathcal{S})$ is called an *inference rule* for $\mathcal{S}$ and denoted $\frac{B_1 \ldots B_n}{A}$. In the following we often use $\boldsymbol{B}_n$ to refer a sequence $B_1, \ldots, B_n$ of n formulas. A proof tree T is either

1. an *open goal*, simply denoted as G, where $G \in \mathit{Form}(\mathcal{S})$. Then, we denote $\mathit{root}(T) = G$. Or
2. a *derivation tree* with root G, denoted as $\frac{T_1 \quad \cdots \quad T_n}{G}(\rho)$ where $G \in \mathit{Form}(\mathcal{S})$, $T_1, \ldots, T_n$ are proof trees (for $n \geq 0$), and $\rho : \frac{B_1 \ldots B_n}{A}$ is an inference rule in $\mathcal{I}(\mathcal{S})$, such that $G = \sigma(A)$, and $\mathit{root}(T_1) = \sigma(B_1), \ldots, \mathit{root}(T_n) = \sigma(B_n)$ for some substitution $\sigma \in \mathit{Sub}(\mathcal{S})$. We write $\mathit{root}(T) = G$.

A finite proof tree without open goals is called a *closed* proof tree for $\mathcal{S}$. If there is a closed proof tree T for $\varphi \in \mathit{Form}(\mathcal{S})$ using $\mathcal{I}(\mathcal{S})$ (i.e., such that $\mathit{root}(T) = \varphi$), we often denote this by writing $\mathcal{S} \vdash \varphi$.

A proof tree T for $\mathcal{S}$ is a *proper prefix* of a proof tree T' (denoted $T \subset T'$) 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 derivation tree T_i with root G_i. A proof tree T for $\mathcal{S}$ is *well-formed* if it is either an open goal, or a closed proof tree, or a tree $\frac{T_1 \quad \cdots \quad T_n}{G}(\rho)$ where there is i, $1 \leq i \leq n$ such that $T_1, \ldots, T_{i-1}$ are *closed*, T_i is *well-formed but not closed*, and $T_{i+1}, \ldots, T_n$ are *open* goals. An *infinite proof tree* T for $\mathcal{S}$ is an infinite sequence $\{T_i\}_{i \in \mathbb{N}}$ of finite trees such that for all i, $T_i \subset T_{i+1}$. We write $root(T) = root(T_0)$.

Definition 1. [11] *A theory $\mathcal{S}$ in a logic $\mathcal{L}$ is called* operationally terminating *iff no infinite well-formed proof tree for $\mathcal{S}$ exists.*

A *proof jump* ψ for $\mathcal{S}$ is a pair $(A \Uparrow \boldsymbol{B}_n)$, where $n \geq 1$ and $A, B_1, \ldots, B_n \in$ *Form*$(\mathcal{S})$; A and B_n are called the *head* and *hook* of ψ, respectively. The *proof jumps of* $\mathcal{I}(\mathcal{S})$ are $\mathcal{J}_{\mathcal{S}} = \{(A \Uparrow \boldsymbol{B}_i) \mid \frac{\boldsymbol{B}_n}{A} \in \mathcal{I}(\mathcal{S}), 1 \leq i \leq n\}$.

Remark 2. Given an inference rule $\frac{B_1, \ldots, B_n}{A}$ with label ρ and $1 \leq i \leq n$, $[\rho]^i$ denotes the i-th proof jump $A \Uparrow B_1, \ldots, B_i$ which is obtained from ρ.

An $(\mathcal{S}, \mathcal{J})$*-chain* is a sequence $(\psi_i)_{i \geq 1}$ of proof jumps $\psi_i : (A^i \Uparrow \boldsymbol{B}^i_{n_i}) \in \mathcal{J}$ together with a substitution σ such that for all $i \geq 1$, $\sigma(B^i_{n_i}) = \sigma(A^{i+1})$ and for all j, $1 \leq j < n_i$, $\mathcal{S} \vdash \sigma(B^i_j)$. An *OT problem* τ in $\mathcal{L}$ is a pair $(\mathcal{S}, \mathcal{J})$ with $\mathcal{S} \in Th(\mathcal{L})$ and $\mathcal{J} \subseteq Jumps(\mathcal{S})$; τ is *finite* if there is no infinite $(\mathcal{S}, \mathcal{J})$-chain; τ is called *infinite* if it is *not* finite. The set of all OT problems in $\mathcal{L}$ is $OTP(\mathcal{L})$. The *initial OT problem* τ_I of a theory $\mathcal{S}$ is $(\mathcal{S}, \mathcal{J}_{\mathcal{S}})$.

Theorem 1. [14] *A theory $\mathcal{S}$ is* operationally terminating *iff $(\mathcal{S}, \mathcal{J}_{\mathcal{S}})$ is* finite.

An *OT processor* $\mathsf{P} : OTP(\mathcal{L}) \to \mathcal{P}(OTP(\mathcal{L})) \cup \{\mathsf{no}\}$ 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. A processor P is *complete* if for all OT problems τ, if $\mathsf{P}(\tau) = \mathsf{no}$ or $\mathsf{P}(\tau)$ contains an infinite OT problem, then τ is infinite. By repeatedly applying processors, we can construct a tree (called *OT-tree*) for an OT-problem $(\mathcal{S}, \mathcal{J})$ whose nodes are labeled with OT problems or *"*yes*"* or *"*no*"*, and whose root is labeled with $(\mathcal{S}, \mathcal{J})$. For every inner node labeled with τ, there is a processor P satisfying one of the following: (i) $\mathsf{P}(\tau) = \mathsf{no}$ and the node has just one child that is labeled with *"*no*"*. (ii) $\mathsf{P}(\tau) = \varnothing$ and the node has just one child that is labeled with *"*yes*"*. (iii) $\mathsf{P}(\tau) \neq \mathsf{no}$, $\mathsf{P}(\tau) \neq \varnothing$, and the children of the node are labeled with the OT problems in $\mathsf{P}(\tau)$.

Theorem 2 (OT-Framework). *Let $(\mathcal{S}, \mathcal{J}) \in OTP(\mathcal{L})$. If all leaves of an OT-tree for $(\mathcal{S}, \mathcal{J})$ are labeled with* "yes" *and all used processors are* sound*, then $(\mathcal{S}, \mathcal{J})$ is finite. If there is a leaf labeled with* "no" *and all processors used on the path from the root to this leaf are* complete*, then $(\mathcal{S}, \mathcal{J})$ is infinite.*

3 Order-Sorted First-Order Logic

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 [7]. 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 order $\leq$ on S to strings of equal length in S^* by $s_1 \cdots s_n \leq s'_1 \cdots s'_n$ iff $s_i \leq s'_i$ 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 [7]. Constant symbols, however, have only one rank declaration. Besides, the following *monotonicity condition* must be satisfied: $f \in \Sigma_{w_1,s_1} \cap \Sigma_{w_2,s_2}$ and $w_1 \leq w_2$ imply $s_1 \leq s_2$. We assume that Σ is *sensible*, meaning that if $f : s_1 \cdots s_n \to s$ and $f : s'_1 \cdots s'_n \to s'$ are such that $[s_i] = [s'_i]$, $1 \leq i \leq n$, then $[s] = [s']$. An order-sorted signature Σ is *regular* iff given $w_0 \leq w_1$ in S^* and $f \in \Sigma_{w_1,s_1}$, there is a least $(w, s) \in S^* \times S$ such that $f \in \Sigma_{w,s}$ and $w_0 \leq w$. If, in addition, each connected component $[s]$ of the sort poset has a top element $\top_{[s]} \in [s]$, then the regular signature is called *coherent*.

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 $t_i \in \mathcal{T}_\Sigma(\mathcal{X})_{s_i}$, $1 \leq i \leq n$, $f(t_1, \ldots, t_n) \in \mathcal{T}_\Sigma(\mathcal{X})_s$. If $\mathcal{X} = \varnothing$, we write $\mathcal{T}_\Sigma$ rather than $\mathcal{T}_\Sigma(\varnothing)$ for the set of *ground* terms. Terms with variables can also be seen as a special case of ground terms of the *extended* signature $\Sigma(\mathcal{X})$ where variables are considered as *constant* symbols of the apporpriate sort, i.e., $\Sigma(\mathcal{X})_{\lambda,s} = \Sigma_{\lambda,s} \cup \mathcal{X}_s$. The assumption that Σ is sensible ensures that if $[s] \neq [s']$, then $\mathcal{T}_\Sigma(\mathcal{X})_{[s]} \cap \mathcal{T}_\Sigma(\mathcal{X})_{[s']} = \varnothing$. The set $\mathcal{T}_\Sigma(\mathcal{X})$ of order-sorted terms is $\mathcal{T}_\Sigma(\mathcal{X}) = \cup_{s\in S}\mathcal{T}_\Sigma(\mathcal{X})_s$.

Following [6], an order-sorted signature *with predicates* Ω is a quadruple $\Omega = (S, \leq, \Sigma, \Pi)$ such that $(S, \leq, \Sigma)$ is an coherent order-sorted signature, and $\Pi = \{\Pi_w \mid w \in S^+\}$ is a family of *predicate symbols* $P, Q, \ldots$ We write $P : w$ for $P \in \Pi_w$. Overloading is also allowed on predicates with the following conditions:

1. There is an equality predicate symbol $= \in \Pi_{ss}$ iff s is the top of a connected component of the sort poset S.
2. *Regularity*: For each w_0 such that there is $P \in \Pi_{w_1}$ with $w_0 \leq w_1$, there is a least w such that $P \in \Pi_w$ and $w_0 \leq w$.

We often write Σ, Π instead of $(S, \leq, \Sigma, \Pi)$ if S and $\leq$ are clear from the context. The formulas φ of an order-sorted signature with predicates Σ, Π 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 $P(t_1, \ldots, t_n) \in \mathit{Form}_{\Sigma,\Pi}$ (we often call it an *atom*); (ii) if $\varphi \in \mathit{Form}_{\Sigma,\Pi}$, then $\neg\varphi \in \mathit{Form}_{\Sigma,\Pi}$; (iii) if $\varphi, \varphi' \in \mathit{Form}_{\Sigma,\Pi}$, then $\varphi \wedge \varphi' \in \mathit{Form}_{\Sigma,\Pi}$; (iv) if $s \in S$, $x \in \mathcal{X}_s$, and $\varphi \in \mathit{Form}_{\Sigma,\Pi}$, then $(\forall x : s)\varphi \in \mathit{Form}_{\Sigma,\Pi}$. 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*.

Order-Sorted Algebras and Structures. 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* 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 $\mathcal{A}_w = \mathcal{A}_{s_1} \times \cdots \times \mathcal{A}_{s_n}$ if $w = s_1 \cdots s_n$, and $\mathcal{A}_w$ is a one point set when $w = \lambda$. 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 (i) If $s, s' \in S$ are such that $s \leq s'$, then $\mathcal{A}_s \subseteq \mathcal{A}_{s'}$, and (ii) If $f \in \Sigma_{w_1,s_1} \cap \Sigma_{w_2,s_2}$ and $w_1 \leq w_2$, then $f^{\mathcal{A}}_{w_1,s_1} \in \mathcal{A}_{w_1} \to A_{s_1}$ equals $f^{\mathcal{A}}_{w_2,s_2} \in \mathcal{A}_{w_2} \to A_{s_2}$ on $\mathcal{A}_{w_1}$. With regard to many sorted signatures and algebras, an (S, Σ)-homomorphism between (S, Σ)-algebras $\mathcal{A}$ and $\mathcal{A}'$ is an S-sorted function $h = \{h_s : \mathcal{A}_s \to \mathcal{A}'_s \mid s \in S\}$ such that for each $f \in \Sigma_{w,s}$ with $w = s_1, \ldots, s_k$, $h_s(f^{\mathcal{A}}_{w,s}(a_1, \ldots, a_k)) = f^{\mathcal{A}'}_{w,s}(h_{s_1}(a_1), \ldots, h_{s_k}(a_k))$. If $w = \lambda$, we have $h_s(f^{\mathcal{A}}) = f^{\mathcal{A}'}$. Now, for the order-sorted case, an $(S, \leq, \Sigma)$-homomorphism $h : \mathcal{A} \to \mathcal{A}'$ between $(S, \leq, \Sigma)$-algebras $\mathcal{A}$ and $\mathcal{A}'$ is an (S, Σ)-homomorphism that satisfies the following additional condition: if $s \leq s'$ and $a \in \mathcal{A}_s$, then $h_s(a) = h_{s'}(a)$.

Given an order-sorted signature with predicates $(S, \leq, \Sigma, \Pi)$, an $(S, \leq, \Sigma, \Pi)$-*structure* (or just a Σ, Π-structure) 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 [6]: (i) for P the identity predicate $\_ = \_ : ss$, the assignment is the identity relation, i.e., $(=)_{\mathcal{A}} = \{(a, a) \mid a \in \mathcal{A}_s\}$; and (ii) whenever $P : w_1$ and $P : w_2$ and $w_1 \leq w_2$, then $P^{\mathcal{A}}_{w_1} = \mathcal{A}_{w_1} \cap P^{\mathcal{A}}_{w_2}$.

Let $(S, \leq, \Sigma, \Pi)$ be an order-sorted signature with predicates and $\mathcal{A}, \mathcal{A}'$ be $(S, \leq, \Sigma, \Pi)$-structures. Then, an $(S, \leq, \Sigma, \Pi)$-*homomorphism* $h : \mathcal{A} \to \mathcal{A}'$ is an $(S, \leq, \Sigma)$-homomorphism such that, for each $P : w$ in Π, if $(a_1, \ldots, a_n) \in P^{\mathcal{A}}_w$, then $h(a_1, \ldots, a_n) \in P^{\mathcal{A}'}_w$. Given an S-sorted *valuation mapping* $\alpha : \mathcal{X} \to \mathcal{A}$, the evaluation mapping $[\_]^{\alpha}_{\mathcal{A}} : \mathcal{T}_{\Sigma}(\mathcal{X}) \to \mathcal{A}$ is the unique $(S, \leq, \Sigma)$-homomorphism extending α [7]. Finally, $[\_]^{\alpha}_{\mathcal{A}} : \mathit{Form}_{\Sigma,\Pi} \to \mathit{Bool}$ is given by:

1. $[P(t_1, \ldots, t_k)]^{\alpha}_{\mathcal{A}} = \mathsf{true}$ for $P : w$ and terms $t_1, \ldots, t_k$ if and only if $([t_1]^{\alpha}_{\mathcal{A}}, \ldots, [t_k]^{\alpha}_{\mathcal{A}}) \in P^{\mathcal{A}}_w$;
2. $[\neg\varphi]^{\alpha}_{\mathcal{A}} = \mathsf{true}$ if and only if $[\varphi]^{\alpha}_{\mathcal{A}} = \mathsf{false}$;
3. $[\varphi \wedge \psi]^{\alpha}_{\mathcal{A}} = \mathsf{true}$ if and only if $[\varphi]^{\alpha}_{\mathcal{A}} = \mathsf{true}$ and $[\psi]^{\alpha}_{\mathcal{A}} = \mathsf{true}$;
4. $[(\forall x : s)\, \varphi]^{\alpha}_{\mathcal{A}} = \mathsf{true}$ if and only if for all $a \in \mathcal{A}_s$, $[\varphi]^{\alpha[x \mapsto a]}_{\mathcal{A}} = \mathsf{true}$;

We say that $\mathcal{A}$ *satisfies* $\varphi \in \mathit{Form}_{\Sigma,\Pi}$ if there is $\alpha \in \mathcal{X} \to \mathcal{A}$ such that $[\varphi]^{\alpha}_{\mathcal{A}} = \mathsf{true}$. If $[\varphi]^{\alpha}_{\mathcal{A}} = \mathsf{true}$ for *all* valuations α, we write $\mathcal{A} \models \varphi$ and say that $\mathcal{A}$ is a *model* of φ. Initial valuations are not relevant for establishing the satisfiability of *sentences*; thus, both notions coincide on them. We say that $\mathcal{A}$ is *a model of a set of sentences* $\mathcal{S} \subseteq \mathit{Form}_{\Sigma,\Pi}$ (written $\mathcal{A} \models \mathcal{S}$) if for all $\varphi \in \mathcal{S}$, $\mathcal{A} \models \varphi$. And, given a sentence φ, we write $\mathcal{S} \models \varphi$ if and only if for *all models* $\mathcal{A}$ of $\mathcal{S}$, $\mathcal{A} \models \varphi$.

Sound logics guarantee that every provable sentence φ is true in *every model* of $\mathcal{S}$, i.e., $\mathcal{S} \vdash \varphi$ implies $\mathcal{S} \models \varphi$.

4 Interpreting Predicates Using Convex Domains

In [10] we have shown that convex domains [12] provide an appropriate basis to the *automatic* definition of algebras and structures that can be used in program analysis with order-sorted first-order specifications. In the following definition, vectors $\boldsymbol{x}, \boldsymbol{y} \in \mathbb{R}^n$ are *compared* using the *coordinate-wise* extension of the ordering $\geq$ among *numbers* which, by abuse, we denote using $\geq$ as well:

$$\boldsymbol{x} = (x_1, \ldots, x_n)^T \geq (y_1, \ldots, y_n)^T = \boldsymbol{y} \text{ iff } x_1 \geq y_1 \wedge \cdots \wedge x_n \geq y_n \tag{3}$$

Definition 2. [12, Definition 1] *Given a matrix* $\mathsf{C} \in \mathbb{R}^{m\times n}$*, and* $\boldsymbol{b} \in \mathbb{R}^m$*, the set* $D(\mathsf{C}, \boldsymbol{b}) = \{\boldsymbol{x} \in \mathbb{R}^n \mid \mathsf{C}\boldsymbol{x} \geq \boldsymbol{b}\}$ *is called a* convex polytopic domain.

Sorts $s \in S$ are interpreted as convex domains $\mathcal{A}_s = D(\mathsf{C}^s, \boldsymbol{b}^s)$, where $\mathsf{C}^s \in \mathbb{R}^{m_s\times n_s}$ and $\boldsymbol{b}^s \in \mathbb{R}^{m_s}$ for some $m_s, n_s \in \mathbb{N}$. Thus, $\mathcal{A}_s \subseteq \mathbb{R}^{n_s}$. Function symbols $f : s_1 \cdots s_k \to s$ are interpreted by $F_1x_1 + \cdots + F_kx_k + F_0$ where (1) for all i, $1 \leq i \leq k$, $F_i \in \mathbb{R}^{n_s\times n_{s_i}}$ are $n_s \times n_{s_i}$-matrices and x_i are variables ranging on $\mathbb{R}^{n_{s_i}}$, (2) $F_0 \in \mathbb{R}^{n_s}$, and (3) the following *algebraicity condition* holds:

$$\forall x_1 \in \mathbb{R}^{n_{s_1}}, \ldots \forall x_k \in \mathbb{R}^{n_{s_k}} \left(\bigwedge_{i=1}^{k} \mathsf{C}^{s_i} x_i \geq \boldsymbol{b}^{s_i} \Rightarrow \mathsf{C}^s(F_1x_1 + \cdots + F_kx_k + F_0) \geq \boldsymbol{b}^s \right)$$

In [10] no procedure for the *automatic* generation of predicate interpretations was given. We solve this problem by providing (parametric) interpretations for predicate symbols P of *any* rank $w \in S^+$. Each predicate symbol $P \in \Pi_w$ with $w = s_1 \cdots s_k$ with $k > 0$ is given an expression

$$R_1x_1 + \cdots + R_kx_k + R_0 \qquad (\text{or } \sum_{i=1}^{k} R_ix_i + R_0 \text{ for short})$$

where (i) for all i, $1 \leq i \leq k$, $R_i \in \mathbb{R}^{m_P\times n_{s_i}}$ are $m_P \times n_{s_i}$-matrices for some $m_P > 0$ and x_i are variables ranging on $\mathbb{R}^{n_{s_i}}$ and (ii) $R_0 \in \mathbb{R}^{m_P}$. Then,

$$P_w^{\mathcal{A}} = \{(\boldsymbol{x}_1, \ldots, \boldsymbol{x}_k) \in \mathcal{A}_{s_1} \times \cdots \times \mathcal{A}_{s_k} \mid \sum_{i=1}^{k} R_i\boldsymbol{x}_i + R_0 \geq \boldsymbol{0}\}$$

or, in our specific setting,

$$P_w^{\mathcal{A}} = \{(\boldsymbol{x}_1, \ldots, \boldsymbol{x}_k) \in \mathbb{R}^{n_{s_1}} \times \cdots \times \mathbb{R}^{n_{s_k}} \mid \bigwedge_{i=1}^{k} \mathsf{C}^{s_i}\boldsymbol{x}_i \geq \boldsymbol{b}^{s_i} \wedge \sum_{i=1}^{k} R_i\boldsymbol{x}_i + R_0 \geq \boldsymbol{0}\}$$

Note that $P_w^{\mathcal{A}} \subseteq \mathcal{A}_w$, as required. As explained in [10, Section 4], the automatic generation of predicate interpretations is treated as done for sorts s and function symbols, i.e., by using *parametric entries* in the involved matrices and vectors that are given numeric values through constraint solving processes.

Example 2. 'Extreme' relations $P_w^{\mathcal{A}}$ associated to a predicate $P \in \Pi_w$ are obtained as follows: if $w = s_1 \cdots s_k$, let R_i be *null* $m_P \times n_{s_i}$-matrices for $i = 1, \ldots, k$.

– If $R_0 = (1, 0, \ldots, 0)^T$, then $P_w^{\mathcal{A}} = \varnothing$ (empty relation).
– If R_0 is a null vector, then $P_w^{\mathcal{A}} = \mathcal{A}_w$ (full relation).

Example 3 (Equality). Equality cannot be defined as such at the (first-order) logical level[1]. For this reason, the interpretation of an equality predicate $= \in \Pi_{s\,s}$ is explicitly required to be the *equality* relation $\{(x, x) \mid x \in \mathcal{A}_s\}$ in the domain $\mathcal{A}_s$ of sort s. Fortunately, we can easily obtain such an interpretation by using the generic method above. With $m_P = 2n_s$, $R_1, R_2 \in \mathbb{R}^{m_P \times n_s}$ given by $R_1 = \begin{bmatrix} I_{n_s} \\ -I_{n_s} \end{bmatrix}$ (for I_{n_s} the *identity* matrix of $n_s \times n_s$ entries) and $R_2 = -R_1$, respectively, and $R_0 = (0, \ldots, 0)^T \in \mathbb{R}^{m_P}$, we obtain the equality predicate on $\mathbb{R}^{n_s}$.

Example 4 (Orderings). The coordinate-wise extension (3) of $\geq$ to n-tuples $\boldsymbol{x}, \boldsymbol{y} \in \mathbb{R}^n$ is obtained if $R_1 = I_n$, $R_2 = -I_n$ and $R_0 = \mathbf{0}$. In particular, if $n = 1$, we obtain the usual ordering $\geq$ over the reals.

Definition 3 (Well-Founded Relation). *Consider a binary relation R on a set A, i.e., $R \subseteq A \times A$. We say that R is* well-founded *if there is no infinite sequence $a_1, a_2, \ldots$ such that for all $i \geq 1$, $a_i \in A$ and $a_i \, R \, a_{i+1}$.*

In the following, given $\delta > 0$, and $x, y \in \mathbb{R}$, we write $x >_\delta y$ iff $x - y \geq \delta$.

Example 5 (Well-founded strict ordering). Borrowing [2], the following *strict* ordering on vectors in $\mathbb{R}^n$:

$$(x_1, \ldots, x_n)^T >_\delta (y_1, \ldots, y_n)^T \text{ iff } x_1 >_\delta y_1 \wedge (x_2, \ldots, x_n)^T \geq (y_2, \ldots, y_n)^T$$

is obtained if $R_1 = I_n$, $R_2 = -I_n$ and $R_0 = (-\delta, 0, \ldots, 0)^T$. In particular, if $n = 1$, we obtain the ordering $>_\delta$ over the reals which is well-founded on subsets A of real numbers which are *bounded from below*, i.e., such that $A \subseteq [\alpha, \infty)$ for some $\alpha \in \mathbb{R}$.

Example 6. For tuples of natural numbers the following *strict* ordering on vectors in $\mathbb{R}^n$ $\boldsymbol{x} >_{\Sigma}^{w} \boldsymbol{y}$ iff $\boldsymbol{x} \geq \boldsymbol{y} \wedge \sum_{i=1}^n x_i >_1 \sum_{i=1}^n y_i$, borrowed from the "*weak decrease + strict decrease in sum of components*" ordering over tuples of natural numbers in [17, Definition 3.1] is obtained if $m_P = n + 1$ (hence R_1, R_2 are $(n+1) \times n$-matrices and $R_0 \in \mathbb{R}^{n+1}$) and we let

$$R_1 = \begin{bmatrix} \mathbf{1}^T \\ I_n \end{bmatrix} \qquad R_2 = -R_1 \qquad R_0 = (-\delta, 0, \ldots, 0)^T$$

for some $\delta > 0$, where $\mathbf{1}$ is the constant vector $(1, \ldots, 1)^T \in \mathbb{R}^n$.

[1] It is well-known that equality $x = y$ can be defined by the *second-order* expression $\forall P(P(x) \Leftrightarrow P(y))$.

5 Using the Removal Pair Processor

We can remove proof jumps ($A \Uparrow \boldsymbol{B}_n$) from OT problems $(\mathcal{S},\mathcal{J})$ by using *removal pairs* $(\gtrsim, \sqsupset)$, where $\gtrsim$ and $\sqsupset$ are binary relations on $\mathit{Form}(\mathcal{S})$ such that $\sqsupset$ is *well-founded* and $\gtrsim \circ \sqsupset \subseteq \sqsupset$ or $\sqsupset \circ \gtrsim \subseteq \sqsupset$ (we say that $\gtrsim$ is *compatible* with $\sqsupset$) provided that the *hook* B_n is '*smaller*' (w.r.t. $\sqsupset$) than the *head* A.

Definition 4. [14] *Let* $(\mathcal{S},\mathcal{J}) \in \mathit{OTP}(\mathcal{L})$*,* $\psi : A \Uparrow \boldsymbol{B}_n \in \mathcal{J}$*, and* $(\gtrsim, \sqsupset)$ *be a* removal pair*. Then,* $\mathsf{P}_{RP}(\mathcal{S},\mathcal{J}) = \{(\mathcal{S},\mathcal{J} - \{\psi\})\}$ *if and only if*

1. *for all* $C \Uparrow \boldsymbol{D}_m \in \mathcal{J} - \{\psi\}$ *and substitutions* σ*, if* $\mathcal{S} \vdash \sigma(D_i)$ *for all* $1 \leq i < m$*, then* $\sigma(C) \gtrsim \sigma(D_m)$ *or* $\sigma(C) \sqsupset \sigma(D_m)$*, and*
2. *for all substitutions* σ*, if* $\mathcal{S} \vdash \sigma(B_i)$ for all $1 \leq i < n$*, then* $\sigma(A) \sqsupset \sigma(B_n)$*.*

In order to *use* P_{RP}, we need to check conditions (1) and (2) in Definition 4. That is, given a proof jump $F \Uparrow \boldsymbol{E}_p$ with $E_1, \ldots, E_p, F \in \mathit{Form}(\mathcal{S})$, and $\bowtie \in \{\gtrsim, \sqsupset\}$, we have to prove statements of the following form: for all substitutions σ,

$$\text{if } \mathcal{S} \vdash \sigma(F_i) \text{ for all } i,\ 1 \leq i < p, \text{ then } \sigma(E) \bowtie \sigma(F_p) \tag{4}$$

Although (4) is an "implication", the *provability statements* $\mathcal{S} \vdash \sigma(F_i)$, and the presence of symbols $\gtrsim$ and $\sqsupset$ (in statements $\sigma(E) \bowtie \sigma(F_p)$) which do *not* belong to the language of $\mathcal{S}$, prevents (4) from being an implication of the language of $\mathcal{S}$. We use theory transformations to overcome this problem.

Remark 3. Our approach leads to implementing P_{RP} when applied to an OT problem $\tau = (\mathcal{S},\mathcal{J})$ as a *satisfiability* problem, i.e., the problem of finding a model $\mathcal{A}$ for a theory $\mathcal{S}_\tau$ which is obtained by extending $\mathcal{S}$ with appropriate sentences to represent the application of P_{RP} to τ (see Section 5.2).

5.1 Transforming Order-Sorted First-Order Theories

We define a transformation of order-sorted signatures with predicates as follows: given $\Omega = (S, \leq, \Sigma, \Pi)$, an Ω-theory $\mathcal{S}$ and an OT problem $\tau = (\mathcal{S}, \{A^i \Uparrow \boldsymbol{B}^i_{n_i} \mid 1 \leq i \leq m\})$ where for all i, $1 \leq i \leq m$, A^i and $B^i_{n_i}$ are Ω-atoms, a new order-sorted signature with predicates $\Omega_\tau = (S_\tau, \leq_\tau, \Sigma_\tau, \Pi_\tau)$ is defined, where, if we let $\Psi_\tau = \{\mathit{pred}(A^i) \mid 1 \leq i \leq m\} \cup \{\mathit{pred}(B^i_{n_i}) \mid 1 \leq i \leq m\}$, then

- $S_\tau = S \cup \{s_\tau\}$ where s_τ is a fresh sort symbol.
- $\leq_\tau$ extends $\leq$ by defining $s_\tau \leq_\tau s_\tau$, and for all $s, s' \in S$, $s \leq_\tau s'$ iff $s \leq s'$. Note that we do *not* assume any subsort relation between s_τ and sorts $s \in S$.
- $\Sigma_\tau = \Sigma \cup \{f_P : w \to s_\tau \mid w \in S^+, P \in \Psi_\tau \cap \Pi_w\}$, i.e., each (overloaded version of a) predicate symbol P in Ψ_τ with input sorts w is given a new function symbol $f_P : w \to s_\tau$ with input sorts w and output sort s_τ.
- $\Pi_\tau = \Pi \cup \Pi_{s_\tau s_\tau}$ where $\Pi_{s_\tau s_\tau} = \{\pi_{\gtrsim}, \pi_{\sqsupset}\}$ for new binary (infix) predicate symbols $\pi_{\gtrsim}$ and $\pi_{\sqsupset}$.

Since Ω_τ is an extension of Ω, every Σ_τ, Π_τ-structure $\mathcal{A}$ is also a Σ, Π-structure. Given an atom $P(t_1, \ldots, t_n)$ with $P \in \Psi_\tau \cap \Pi_{s_1 \cdots s_n}$ and terms $t_i \in \mathcal{T}_\Sigma(\mathcal{X})_{s_i}$, for $1 \leq i \leq n$, the transformation $\_^{\downarrow}$ from atoms in Ω to *terms* in Ω_τ is obtained by replacing P by $f_P \in \Sigma_\tau$: $P(t_1, \ldots, t_n)^{\downarrow} = f_P(t_1, \ldots, t_n)$. We can use Ω_τ-structures $\mathcal{A}$ to define binary relations on Ω-formulas.

Definition 5. *Let Ω be an order-sorted signature with predicates, τ be an OT-problem, and $\mathcal{A}$ be an Ω_τ-structure. Given $\pi_{\bowtie} \in \Pi_{s_\tau s_\tau}$, we define a relation $\bowtie$ on Ω-formulas as follows: for all Ω-formulas A and B $A \bowtie B$ iff $\mathcal{A} \models A^{\downarrow} \pi_{\bowtie} B^{\downarrow}$.*

Now, we can recast (4) as a logic formula:

$$\forall \boldsymbol{x}(F_1 \wedge \cdots \wedge F_{p-1} \Rightarrow E^{\downarrow}\, \pi_{\bowtie}\, F_p^{\downarrow}) \tag{5}$$

Theorem 3. *Let Ω be an order-sorted signature with predicates, $\tau = E \Uparrow \boldsymbol{F}_p$ be an OT-problem, $\mathcal{A}$ be an Ω_τ-structure such that $\mathcal{A} \models \mathcal{S}$, $\pi_{\bowtie} \in \Pi_{s_\tau s_\tau}$, and σ be a substitution. If for all i, $1 \leq i < p$, $\mathcal{S} \vdash \sigma(F_i)$ holds and $\mathcal{A} \models \forall \boldsymbol{x}(F_1 \wedge \cdots \wedge F_{p-1} \Rightarrow E^{\downarrow}\, \pi_{\bowtie}\, F_p^{\downarrow})$, then (4) holds for $\bowtie$ as in Definition 5.*

Proof. Since for all i, $1 \leq i < p$, $\mathcal{S} \vdash \sigma(F_i)$ holds and $\mathcal{A} \models \mathcal{S}$, by soundness we have $\mathcal{A} \models \sigma(F_i)$ for all i, $1 \leq i < p$. Now, since $\mathcal{A} \models \forall \boldsymbol{x}(F_1 \wedge \cdots \wedge F_{p-1} \Rightarrow E^{\downarrow}\, \pi_{\bowtie}\, F_p^{\downarrow})$, we have that $\mathcal{A} \models \sigma(E^{\downarrow}\, \pi_{\bowtie}\, F_p^{\downarrow})$ holds, i.e., $\mathcal{A} \models \sigma(E)^{\downarrow}\, \pi_{\bowtie}\, \sigma(F_p)^{\downarrow}$ holds. Thus, by Definition 5, we have $\sigma(E) \bowtie \sigma(F_p)$ as desired.

Compatibility. Component $\gtrsim$ of a removal pair $(\gtrsim, \sqsupset)$ must be *compatible* with $\sqsupset$. This can be guaranteed at the *logical level* by the following Ω_τ-sentence:

$$\big(\forall xyz : s_\tau(x\, \pi_{\gtrsim}\, y \wedge y\, \pi_{\sqsupset}\, z \Rightarrow x\, \pi_{\sqsupset}\, z)\big) \vee \big(\forall xyz : s_\tau(x\, \pi_{\sqsupset}\, y \wedge y\, \pi_{\gtrsim}\, z \Rightarrow x\, \pi_{\sqsupset}\, z)\big)$$

Well-foundedness. We also need to guarantee *well-foundedness* of $\sqsupset$. Unfortunately, the well-foundedness of a relation $P^{\mathcal{A}}$ interpreting a binary predicate symbol P can *not* be characterized at once in first-order logic [18, Section 5.1.4]. We can guarantee well-foundedness of $\sqsupset$, though, at the *semantic* level by interpreting $\pi_{\sqsupset}$ as a well-founded relation $\pi_{\sqsupset}^{\mathcal{A}}$ in the Ω_τ-structure $\mathcal{A}$.

Proposition 1. *Let Ω be an order-sorted signature with predicates, τ be an OT problem, and $\mathcal{A}$ be a Ω_τ-structure. If $\pi_{\sqsupset}^{\mathcal{A}}$ is a well-founded relation on $\mathcal{A}_{s_\tau}$, then $\sqsupset$ as in Definition 5 is a well-founded relation on Ω-formulas.*

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 5, for all $i \geq 1$ we have $\mathcal{A} \models A_i^{\downarrow}\, \pi_{\sqsupset}\, A_{i+1}^{\downarrow}$, i.e., for all valuations α, $([A_i^{\downarrow}]_{\mathcal{A}}^{\alpha}, [A_{i+1}^{\downarrow}]_{\mathcal{A}}^{\alpha}) \in \pi_{\sqsupset}^{\mathcal{A}}$. Therefore, there is an infinite sequence $([A_i^{\downarrow}]_{\mathcal{A}}^{\alpha})_{i \geq 1}$ for some valuation α that contradicts well-foundedness of $\pi_{\sqsupset}^{\mathcal{A}}$.

5.2 A Semantic Version of the Removal Pair Processor

We can provide the following *semantic version* of the removal pair processor.

Definition 6 (Semantic version of P_{RP}). *Let $\mathcal{L}$ be an OS-FOL with order-sorted signature with predicates Ω, $\tau = (\mathcal{S}, \mathcal{J}) \in OTP(\mathcal{L})$, $\mathcal{A}$ be an Ω_τ-structure, and $\psi : A \Uparrow \boldsymbol{B}_n \in \mathcal{J}$. Then, $\mathsf{P}_{RP}(\mathcal{S}, \mathcal{J}) = \{(\mathcal{S}, \mathcal{J} - \{\psi\})\}$ if $\mathcal{A} \models \mathcal{S}$, and the following conditions hold:*

1. *if* $\mathcal{J} - \{\psi\} \neq \varnothing$*, then*

   $\mathcal{A} \models \big(\forall xyz : s_\tau(x\,\pi_{\gtrsim}\,y \wedge y\,\pi_{\sqsupset}\,z \Rightarrow x\,\pi_{\sqsupset}\,z)\big) \vee \big(\forall xyz : s_\tau(x\,\pi_{\sqsupset}\,y \wedge y\,\pi_{\gtrsim}\,z \Rightarrow x\,\pi_{\sqsupset}\,z)\big)$
2. *for each* $C \Uparrow \boldsymbol{D}_m \in \mathcal{J} - \{\psi\}$*, there is* $\pi_{\bowtie} \in \{\pi_{\gtrsim}, \pi_{\sqsupset}\}$ *such that* $\mathcal{A} \models \bigwedge_{i=1}^{m-1} D_i \Rightarrow C^{\downarrow}\,\pi_{\bowtie}\,D_m^{\downarrow}$.
3. $\pi_{\sqsupset}^{\mathcal{A}}$ *is well-founded and* $\mathcal{A} \models \bigwedge_{i=1}^{n-1} B_i \Rightarrow A^{\downarrow}\,\pi_{\sqsupset}\,B_n^{\downarrow}$

Definition 6 transforms the application of P_{RP} to $(\mathcal{S}, \mathcal{J})$ into the problem of *finding a model* $\mathcal{A}$ of $\mathcal{S}$ which satisfies the following formulas (where J is the number of proof jumps in $\mathcal{J}$):

1. φ^1 (for the modeling condition (1) in Definition 6; only required if $J > 1$),
2. $\varphi_1^2, \ldots, \varphi_{J-1}^2$ (where, for all j, $1 \leq j < J$, φ_j^2 is a disjunction of two formulas due to condition (2)) and
3. φ^3 (the formula in the *removal* condition (3)).

Remark 4 (Finding models to implement P_{RP}*).* Let $\mathcal{S}_\tau = \mathcal{S} \cup \{\varphi^1, \varphi_1^2, \ldots, \varphi_{J-1}^2, \varphi^3\}$. We can use the theory in [10] and Section 4 to obtain a model $\mathcal{A}$ such that $\mathcal{A} \models \mathcal{S}_\tau$ holds. Then, if $\pi_{\sqsupset}^{\mathcal{A}}$ is *well-founded*, we can remove the targetted proof jump ψ from $\mathcal{J}$ in τ.

We still need to envisage a method to guarantee that $\pi_{\sqsupset}^{\mathcal{A}}$ is *well-founded.* In the following section, we show how to guarantee that binary relations synthesized as part of a model as explained in Section 4 are well-founded.

5.3 Well-Foundedness of Relations Defined on Convex Domains

The following result provides a sufficient condition to guarantee *well-foundedness* of a binary relation R on a subset $A \subseteq \mathbb{R}^n$ defined as explained in Section 4. It is based on generalizing the fact that the relation $>_\delta$ over real numbers given by $x >_\delta y$ iff $x - y \geq \delta$ is *well-founded* on subsets $A \subseteq \mathbb{R}$ of real numbers which are *bounded from below* (i.e., $A \subseteq [\alpha, +\infty)$ for some $\alpha \in \mathbb{R}$) whenever $\delta > 0$ [9].

Theorem 4. *Let* $R_1, R_2 \in \mathbb{R}^{m \times n}$ *and* $R_0 \in \mathbb{R}^m$ *for some* $m, n > 0$*, and* R *be a binary relation on* $A \subseteq \mathbb{R}^n$ *as follows: for all* $\boldsymbol{x}, \boldsymbol{y} \in A$*,* $\boldsymbol{x}\,R\,\boldsymbol{y}$ *if and only if* $R_1\boldsymbol{x} + R_2\boldsymbol{y} + R_0 \geq \boldsymbol{0}$*. If there is* $i \in \{1, \ldots, n\}$ *such that*

1. $(R_2)_{i\cdot} = -(R_1)_{i\cdot}$*, i.e., the* i*-th row of* R_2 *is obtained from the* i*-th row of* R_1 *by negating all components,*
2. *There is* $\alpha \in \mathbb{R}$ *such that for all* $\boldsymbol{x} \in A$*,* $(R_1)_{i\cdot}\boldsymbol{x} \geq \alpha$*, and*
3. $(R_0)_i < 0$*,*

then R *is well-founded.*

Proof. By contradiction. If R is not well-founded, then there is an infinite sequence $\boldsymbol{x}_1, \ldots, \boldsymbol{x}_n, \ldots$ of vectors in $\mathbb{R}^n$ such that, for all $j \geq 1$, $x_j\,R\,x_{j+1}$. By (1), we have that, for all $j \geq 1$, $(R_1)_{i\cdot}\boldsymbol{x}_j - (R_1)_{i\cdot}\boldsymbol{x}_{j+1} + (R_0)_i \geq 0$. For all $p > 0$,

$$\sum_{j=1}^{p}(R_1)_{i\cdot}\boldsymbol{x}_j - (R_1)_{i\cdot}\boldsymbol{x}_{j+1} + (R_0)_i = (R_1)_{i\cdot}\boldsymbol{x}_1 - (R_1)_{i\cdot}\boldsymbol{x}_{p+1} + p(R_0)_i \geq 0$$

By (2), there is $\alpha \in \mathbb{R}$ such that for all $p > 0$, $(R_1)_{i\cdot}\boldsymbol{x}_p \geq \alpha$. Therefore, for all $p > 0$, $(R_1)_{i\cdot}\boldsymbol{x}_1 - \alpha \geq (R_1)_{i\cdot}\boldsymbol{x}_1 - (R_1)_{i\cdot}x_{p+1}$, and then $(R_1)_{i\cdot}\boldsymbol{x}_1 - \alpha + p(R_0)_i \geq 0$. By (3), $(R_0)_i < 0$; let $r = -(R_0)_i$. Note that $r > 0$. Then, for all $p > 0$, $(R_1)_{i\cdot}\boldsymbol{x}_1 \geq \alpha + pr$, leading to a contradiction because $\alpha + pr$ tends to infinite as p grows to infinite, but $(R_1)_{i\cdot}\boldsymbol{x}_1 \in \mathbb{R}$ is fixed.

Example 7. Theorem 4 applies to $>_\delta$ and $>^w_\Sigma$ defined on $\mathcal{A}_s$ as follows:

1. for $>_\delta$, take $A \subseteq [\alpha, +\infty) \times \mathbb{R}^{n-1}$, for some $\alpha \in \mathbb{R}$ and $i = 1$ in Theorem 4 with the corresponding R_1, R_2, and R_0 to prove $>_\delta$ well-founded on A.
2. for $>^w_\Sigma$, take $A \subseteq [\alpha, +\infty)^n$, for some $\alpha \geq 0$ and $i = 1$ with the corresponding R_1, R_2, and R_0 to prove $>^w_\Sigma$ well-founded on A.

Note that we can use Theorem 4 to prove well-foundedness of relations R defined on domains $\mathcal{A}$ which are *not* bounded from below.

Example 8. Consider $\mathsf{C} = \begin{bmatrix} 1 & 0 \\ -1 & 0 \end{bmatrix}$ and $\boldsymbol{b} = (0, -2)^T$. Then, $\mathcal{A} = D(\mathsf{C}, \boldsymbol{b}) = [0, 2] \times \mathbb{R}$ is *not* bounded from below in the sense that there is no $\alpha \in \mathbb{R}$ such that $\mathcal{A} \subseteq [\alpha, +\infty)^2$. The relation R on $\mathcal{A}$ defined by $R_1 = \begin{bmatrix} 1 & 0 \\ 1 & 1 \end{bmatrix}$, $R_2 = \begin{bmatrix} -1 & 0 \\ 0 & 1 \end{bmatrix}$ and $R_0 = (-1, 0)$ is well-founded as it satisfies the conditions of Theorem 4.

6 Operational Termination of PATH in the OT-Framework

The set $\mathcal{J}_{\texttt{PATH}}$ of proof jumps for $\mathcal{I}(\texttt{PATH})$ has 43 elements. A powerful processor to *reduce* the size of an OT problem $(\mathcal{S}, \mathcal{J})$ is the *SCC processor* [14]. The so-called *estimated proof graph* $\mathsf{EPG}(\mathcal{S}, \mathcal{J})$ for $(\mathcal{S}, \mathcal{J})$ has $\mathcal{J}$ as set of *nodes*; and there is an *arc* from $\psi : (A \Uparrow \boldsymbol{B}_m)$ to $\psi' : (A' \Uparrow \boldsymbol{B}'_n)$ iff $\sigma(B_m) = \sigma(A')$ for some substitution σ. The *Strongly Connected Components* (SCCs) of a graph are its *maximal* cycles, i.e., those cycles that are not part of other cycles. The *SCC Processor* (P_{SCC}) is given by

$$\mathsf{P}_{SCC}(\mathcal{S}, \mathcal{J}) = \{(\mathcal{S}, \mathcal{J}') \mid \mathcal{J}' \text{ is an SCC in } \mathsf{EPG}(\mathcal{S}, \mathcal{J})\}$$

This is a sound and complete processor.

Example 9. Although $\mathsf{EPG}(\texttt{PATH}, \mathcal{J}_{\texttt{PATH}})$ is huge and we do not display it here, the SCCs are displayed in Figure 2. The involved proof jumps are made explicit in Figure 3 to ease our further developments. We use P_{SCC} to transform the *initial* OT problem $\tau_{\texttt{PATH}} = (\texttt{PATH}, \mathcal{J}_{\texttt{PATH}})$ by $\mathsf{P}_{SCC}(\tau_{\texttt{PATH}}) = \{\tau_1, \ldots, \tau_9\}$ where

$\tau_1 = (\texttt{PATH}, \{[SR_N]^2\})$ $\quad\tau_2 = (\texttt{PATH}, \{[SR_E]^2\})$ $\quad\tau_3 = (\texttt{PATH}, \{[SR_P]^2\})$

$\tau_4 = (\texttt{PATH}, \{[T_N]^2\})$ $\quad\tau_5 = (\texttt{PATH}, \{[T_P]^2\})$ $\quad\tau_6 = (\texttt{PATH}, \{[C_{\mathsf{sq}_1}]^1\})$

$\tau_7 = (\texttt{PATH}, \{[C_{\mathsf{sq}_2}]^1\})$ $\quad\tau_8 = (\texttt{PATH}, \{[M1_{\_;\_}]^2\})$ $\quad\tau_9 = (\texttt{PATH}, \{[T_T]^2\})$

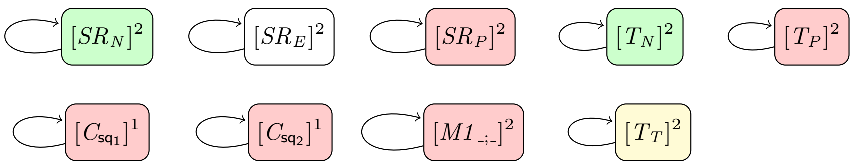


Fig. 2. SCCs of the estimated dependency graph of PATH

$[SR_N]^2$	$t : Node \Uparrow t \to_{[Node]} u \quad u : Node$	
$[SR_E]^2$	$t : Edge \Uparrow t \to_{[Path]} u \quad u : Edge$	
$[SR_P]^2$	$t : Path \Uparrow t \to_{[Path]} u \quad u : Path$	
$[T_N]^2$	$t \to^*_{[Node]} v \Uparrow t \to_{[Node]} u \quad u \to^*_{[Node]} v$	
$[T_P]^2$	$t \to^*_{[Path]} v \Uparrow t \to_{[Path]} u \quad u \to^*_{[Path]} v$	
$[T_T]^2$	$t \to^*_{[Truth]} v \Uparrow t \to_{[Truth]} u \quad u \to^*_{[Truth]} v$	
$[C_{\mathsf{sq}_1}]^1$	$t ; v \to_{[Path]} u ; v \Uparrow t \to_{[Path]} u$	
$[C_{\mathsf{sq}_2}]^1$	$v ; t \to_{[Path]} v ; u \Uparrow t \to_{[Path]} u$	
$[M1_{\_;\_}]^2$	$E; P :: Path \Uparrow E :: Edge \quad P :: Path$	

Fig. 3. Proof jumps of the SCCs in Figure 2

Any further use of P_{SCC} on $\tau_1, \ldots, \tau_9$ is hopeless. Note that $\tau_1, \ldots, \tau_9$ all consist of a *single* proof jump, i.e., $\tau_i = (\mathtt{PATH}, \{\psi_i\})$ for $1 \leq i \leq 9$. With P_{RP} we prove them finite, thus obtaining a proof of operational termination of PATH.

6.1 Using P_{RP} to Prove $\tau_{\mathtt{PATH}}$ finite

Following the approach in Section 5.2 (see Remark 4), for each OT problem τ_i we need to find a appropriate model $\mathcal{A}_i$ to *remove* ψ_i from τ_i thus obtaining the *empty* OT problem $(\mathtt{PATH}, \varnothing)$ which is trivially finite. For this purpose, we use the tool AGES to automatically generate models for order-sorted first-order theories [8]. The tool provides an implementation of the techniques introduced in [10] and also in this paper (Sections 4 and 5.3).

First we express the *order-sorted first-order signature with predicates* that corresponds to PATH as a Maude module as follows:

```
mod PATH_OSSig is
  sorts KTruth .
  sorts Node KNode .
  sorts Edge Path KPath .
  subsorts Node < KNode .
```

```
  subsorts Edge < Path < KPath .
  op tt : -> KTruth .
  op eq : KNode KNode -> KTruth .
  ops source target : KPath -> KNode .
  op seq : KPath KPath -> KPath .
  op mbEdge : KPath -> Bool .
  op mbNode : KNode -> Bool .
  op mbPath : KPath -> Bool .
  op redN : KNode KNode -> Bool .
  op redsN : KNode KNode -> Bool .
  op redP : KPath KPath -> Bool .
  op redsP : KPath KPath -> Bool .
  op redT : KTruth KTruth -> Bool .
  op redsT : KTruth KTruth -> Bool .
endm
```

where

1. `KNode`, `KPath`, and `KTruth` represent *kinds* [`Node`], [`Path`], and [`Truth`] of the MEL specification of `PATH` and have the expected *subsort* relation with the corresponding sorts in the kind.
2. We use the function `seq` instead of the infix operator `_;_`.
3. We are using *predicates* (encoded here as *boolean functions*, as Maude has no specific notation for predicates) `mbEdge`, `mbNode`, and `mbEdge` instead of $\_ : Edge$, $\_ : Node$ and $\_ : Path$.
4. Similarly, we use `redN`, `redsN`, `redP`, `redsP`, `redT`, and `redsT` instead of $\rightarrow_{[Node]}$, $\rightarrow^*_{[Node]}$, $\rightarrow_{[Path]}$, $\rightarrow^*_{[Path]}$, $\rightarrow_{[Truth]}$, and $\rightarrow^*_{[Truth]}$, respectively.

The OS-FOL *theory* $\mathcal{S}^{\texttt{PATH}}$ consists of the sentences obtained from $\mathcal{I}(\texttt{PATH})$ in Figure 1 when each rule $\frac{B_1 \cdots B_n}{A}$ (with variables $x_1, \ldots, x_m$ of sorts $s_1, \ldots, s_m$) is interpreted as a sentence $\forall x_1 : s_1 \cdots x_m : s_m (B_1 \wedge \cdots \wedge B_n \Rightarrow A)$ and written by using the symbols in `PATH_OSSig`. For instance, rule (SR_N) becomes

```
redN(t:KNode,u:KNode) /\ mbNode(u:KNode) => mbNode(t:KNode)
```

in the notation used in AGES, where each variable bears its sort, and universal quantification is assumed.

For the sake of brevity, rather than computing a model $\mathcal{A}_i$ for each OT problem τ_i, $1 \leq i \leq 9$, we proceed in *three* steps by computing models for different *clusters* of OT Problems.

– For OT problems $\tau_1, \ldots, \tau_5$, we compute a model $\mathcal{A}$ of $\mathcal{S} \cup \{\varphi^3_1, \ldots, \varphi^3_5\}$ being φ^3_i for $1 \leq i \leq 5$ the specific formula φ^3 in Section 5.2 particularized to ψ_i.
– For OT problems $\tau_6, \ldots, \tau_8$, we compute a model $\mathcal{A}'$ of $\mathcal{S} \cup \{\varphi^3_6, \ldots, \varphi^3_8\}$.
– For τ_9, we compute a model $\mathcal{A}''$ of $\mathcal{S} \cup \{\varphi^3_9\}$.

Obviously, each computed structure can be used with each *individual* OT problem τ_i in its cluster to remove the corresponding proof jump. Note that, since each OT problem τ_i contains a single proof jump, we do not pay attention to the component $\gtrsim_i$ of the removal pair. Hence, no instance of formulas φ^1 and φ^2 in Section 5.2 is required in the extensions of $\mathcal{S}$.

OT Problems $\tau_1, \ldots, \tau_5$. We extend PATH_OSSig with new sorts, functions and predicate symbols due to the transformation described in Section 5.1:

```
mod PATH-tau1to5 is
  sorts Top1 Top2 Top3 Top4 Top5 .
  op fmbNode : KNode -> Top1 .
  op wfr1 : Top1 Top1 -> Bool [wellfounded] .
  op fisEdge : KPath -> Top2 .
  op wfr2 : Top2 Top2 -> Bool [wellfounded] .
  op fisPath : KPath -> Top3 .
  op wfr3 : Top3 Top3 -> Bool [wellfounded] .
  op fredsN : KNode KNode -> Top4 .
  op wfr4 : Top4 Top4 -> Bool [wellfounded] .
  op fredsP : KPath KPath -> Top5 .
  op wfr5 : Top5 Top5 -> Bool [wellfounded] .
endm
```

In AGES we can impose that the relations interpreting binary predicates wfr1, ..., wfr5 (representing the well-founded components $\sqsupset_i$ of the removal pair which is used in the application of P_{RP} to τ_i for $1 \leq i \leq 5$) be *well-founded*[2]. AGES uses Theorem 4 to ensure this. Then, we obtain a new theory $\mathcal{S}^{\mathtt{PATH}}_{1..5}$ by adding new sentences $\varphi^3_1, \ldots, \varphi^3_5$ corresponding to the proof jumps in $\tau_1, \ldots, \tau_5$ to $\mathcal{S}^{\mathtt{PATH}}$; in AGES notation:

```
redN(tN:KNode,uN:KNode) =>
   wfr1(fmbNode(tN:KNode),fmbNode(uN:KNode))
redP(tP:KPath,uP:KPath) =>
   wfr2(fisEdge(tP:KPath),fisEdge(uP:KPath))
redP(tP:KPath,uP:KPath) =>
   wfr3(fisPath(tP:KPath),fisPath(uP:KPath))
redN(tN:KNode,uN:KNode) =>
   wfr4(fredsN(tN:KNode,vN:KNode),fredsN(uN:KNode,vN:KNode))
redP(tP:KPath,uP:KPath) =>
   wfr5(fredsP(tP:KPath,vP:KPath),fredsP(uP:KPath,vP:KPath))
```

AGES obtains the following model $\mathcal{A}$ for $\mathcal{S}^{\mathtt{PATH}}_{1..5}$:

1. Interpretation of sorts:

$$\mathcal{A}_{\mathtt{KTruth}} = [-1, +\infty) \quad \mathcal{A}_{\mathtt{Node}} = [-1, 0] \quad \mathcal{A}_{\mathtt{KNode}} = [-1, 0]$$
$$\mathcal{A}_{\mathtt{Edge}} = \{-1\} \quad \mathcal{A}_{\mathtt{Path}} = \{-1\} \quad \mathcal{A}_{\mathtt{KPath}} = [-1, 0]$$

$$\mathcal{A}_{\mathtt{Top1}} = [0, +\infty) \quad \mathcal{A}_{\mathtt{Top2}} = [-1, +\infty) \quad \mathcal{A}_{\mathtt{Top3}} = [0, +\infty)$$
$$\mathcal{A}_{\mathtt{Top4}} = [0, +\infty) \quad \mathcal{A}_{\mathtt{Top5}} = [-1, 0]$$

[2] We have enriched the syntax of Maude modules to specifiy this requirement.

2. Interpretation of function symbols (with argument variables taking values in the corresponding sort):

$$\begin{array}{rcr}
\texttt{eq}^{\mathcal{A}}(x,y) = y - x & \texttt{seq}^{\mathcal{A}}(x,y) = -1 - y & \texttt{source}^{\mathcal{A}}(x) = 0 \\
\texttt{target}^{\mathcal{A}}(x) = -1 & \texttt{tt}^{\mathcal{A}} = 0 &
\end{array}$$

$$\begin{array}{rcr}
\texttt{fisEdge}^{\mathcal{A}}(x) = 1 + x & \texttt{fisPath}^{\mathcal{A}}(x) = 2 + x & \texttt{fmbNode}^{\mathcal{A}}(x) = 2 + x \\
\texttt{fredsN}^{\mathcal{A}}(x,y) = 4 + x + y & \texttt{fredsP}^{\mathcal{A}}(x,y) = 0 &
\end{array}$$

3. Interpretation of predicate symbols (as characteristic predicates):

$$\begin{array}{rr}
\texttt{mbEdge}^{\mathcal{A}}(x) \Leftrightarrow x \in [-1,0] & \texttt{mbNode}^{\mathcal{A}}(x) \Leftrightarrow x \in [-1,0] \\
\texttt{mbPath}^{\mathcal{A}}(x) \Leftrightarrow x \in [-1,0] & \texttt{redN}^{\mathcal{A}}(x,y) \Leftrightarrow \mathit{false} \\
\texttt{redP}^{\mathcal{A}}(x,y) \Leftrightarrow \mathit{false} & \texttt{redT}^{\mathcal{A}}(x,y) \Leftrightarrow x,y \in [-1,+\infty) \wedge y \geq x \\
\texttt{redsN}^{\mathcal{A}}(x,y) \Leftrightarrow x,y \in [-1,0] & \texttt{redsP}^{\mathcal{A}}(x,y) \Leftrightarrow x,y \in [-1,0] \wedge x \geq y \\
 & \texttt{redsT}^{\mathcal{A}}(x,y) \Leftrightarrow x,y \in [-1,+\infty) \wedge y \geq x
\end{array}$$

$$\begin{array}{l}
\texttt{wfr1}^{\mathcal{A}}(x,y) \Leftrightarrow x,y \in [0,+\infty) \wedge x >_1 y \\
\texttt{wfr2}^{\mathcal{A}}(x,y) \Leftrightarrow x,y \in [0,+\infty) \wedge x >_1 y \\
\texttt{wfr3}^{\mathcal{A}}(x,y) \Leftrightarrow x,y \in [0,+\infty) \wedge x >_1 y \\
\texttt{wfr4}^{\mathcal{A}}(x,y) \Leftrightarrow \mathit{false} \\
\texttt{wfr5}^{\mathcal{A}}(x,y) \Leftrightarrow x,y \in [-1,0] \wedge y >_1 x
\end{array}$$

Note that $\texttt{redN}^{\mathcal{A}}$ and $\texttt{redP}^{\mathcal{A}}$ are *empty relations*. Actually, this is enough to guarantee that conditions $\varphi_1^3, \ldots, \varphi_5^3$ for the proof jumps at stake hold, thus enabling their removal from the corresponding OT problem.

OT Problems $\tau_6, \ldots, \tau_8$. We extend now `PATH_OSSig` with the following:

```
mod PATH-tau6to8 is
  sorts Top6 Top7 Top8 .
  op fredP : KPath KPath -> Top6 .
  op wfr6 : Top6 Top6 -> Bool [wellfounded] .
  op fredP : KPath KPath -> Top7 .
  op wfr7 : Top7 Top7 -> Bool [wellfounded] .
  op fisPath : KPath -> Top8 .
  op wfr8 : Top8 Top8 -> Bool [wellfounded] .
endm
```

The new theory $\mathcal{S}^{\texttt{PATH}}_{6..8}$ extends $\mathcal{S}^{\texttt{PATH}}$ with $\varphi_6^3, \ldots, \varphi_6^3$, i.e.,

```
wfr6(fredP(seq(tP:KPath,vP:KPath),seq(uP:KPath,vP:KPath)),
     fredP(tP:KPath,uP:KPath))
wfr7(fredP(seq(vP:KPath,tP:KPath),seq(vP:KPath,uP:KPath)),
     fredP(tP:KPath,uP:KPath))
EP:KPath :: Edge =>
   wfr8(fisPath(seq(EP:KPath,PP:KPath)),fisPath(PP:KPath))
```

AGES computes the following model $\mathcal{A}'$ of $\mathcal{S}^{\texttt{PATH}}_{6..8}$:

1. Interpretation of sorts:

$$\begin{array}{ccc} \mathcal{A}'_{\texttt{KTruth}} = [-1,+\infty) & \mathcal{A}'_{\texttt{Node}} = [0,+\infty) & \mathcal{A}'_{\texttt{KNode}} = [0,+\infty) \\ \mathcal{A}'_{\texttt{Edge}} = \{1\} & \mathcal{A}'_{\texttt{Path}} = [1,+\infty) & \mathcal{A}'_{\texttt{KPath}} = [1,+\infty) \end{array}$$

$$\mathcal{A}'_{\texttt{Top6}} = [0,+\infty) \qquad \mathcal{A}'_{\texttt{Top7}} = [0,+\infty) \qquad \mathcal{A}'_{\texttt{Top8}} = [0,+\infty)$$

2. Interpretation of function symbols:

$$\begin{array}{ccc} \texttt{eq}^{\mathcal{A}'}(x,y) = x+y-1 & \texttt{seq}^{\mathcal{A}'}(x,y) = x+y & \texttt{source}^{\mathcal{A}'}(x) = x-1 \\ \texttt{target}^{\mathcal{A}'}(x) = 0 & \texttt{tt}^{\mathcal{A}'} = 0 & \end{array}$$

$$\texttt{fisPath}^{\mathcal{A}'}(x) = 1+x \qquad \texttt{fredP}^{\mathcal{A}'}(x,y) = y-1$$

3. Interpretation of predicate symbols:

$$\begin{array}{cc} \texttt{mbEdge}^{\mathcal{A}'}(x) \Leftrightarrow x \in [1,+\infty) & \texttt{mbNode}^{\mathcal{A}'}(x) \Leftrightarrow x \in [0,+\infty) \\ \texttt{mbPath}^{\mathcal{A}'}(x) \Leftrightarrow x \in [1,+\infty) & \texttt{redN}^{\mathcal{A}'}(x,y) \Leftrightarrow x,y \in [0,+\infty) \wedge x \geq y \\ \texttt{redT}^{\mathcal{A}'}(x,y) \Leftrightarrow x,y \in [-1,+\infty) & \texttt{redP}^{\mathcal{A}'}(x,y) \Leftrightarrow x,y \in [1,+\infty) \wedge x \geq y \\ \texttt{redsN}^{\mathcal{A}'}(x,y) \Leftrightarrow x,y \in [0,+\infty) & \texttt{redsP}^{\mathcal{A}'}(x,y) \Leftrightarrow x,y \in [1,+\infty) \\ & \texttt{redsT}^{\mathcal{A}'}(x,y) \Leftrightarrow x,y \in [-1,+\infty) \end{array}$$

$$\begin{array}{c} \texttt{wfr6}^{\mathcal{A}'}(x,y) \Leftrightarrow x,y \in [0,+\infty) \wedge x >_1 y \\ \texttt{wfr7}^{\mathcal{A}'}(x,y) \Leftrightarrow x,y \in [0,+\infty) \wedge x >_1 y \\ \texttt{wfr8}^{\mathcal{A}'}(x,y) \Leftrightarrow x,y \in [0,+\infty) \wedge x >_1 y \end{array}$$

Note that $\texttt{wfr6}^{\mathcal{A}'}$, $\texttt{wfr7}^{\mathcal{A}'}$, and $\texttt{wfr8}^{\mathcal{A}'}$ coincide with the ordering $>_1$ on $[0,+\infty)$ which is clearly well-founded.

OT Problem τ_9. We extend PATH_OSSig with:

```
mod PATH-tau9 is
  sorts Top9 .
  op fredsT : KTruth KTruth -> Top9 .
  op wfr9 : Top9 Top9 -> Bool [wellfounded] .
endm
```

We obtain a new theory $\mathcal{S}_9^{\texttt{PATH}}$ by adding the sentence φ_9^3:

```
wfr9(fredsT(tT:KTruth,vT:KTruth),fredsT(uT:KTruth,vT:KTruth))
```

corresponding to the proof jumps in τ_9 to $\mathcal{S}^{\texttt{PATH}}$. We obtain a model $\mathcal{A}''$ of $\mathcal{S}_9^{\texttt{PATH}}$:

1. Interpretation of sorts:

$$\begin{array}{cccc} \mathcal{A}''_{\texttt{KTruth}} = [-1,+\infty) & \mathcal{A}''_{\texttt{Node}} = [-1,1] & \mathcal{A}''_{\texttt{KNode}} = [-1,1] & \\ \mathcal{A}''_{\texttt{Edge}} = \{-1\} & \mathcal{A}''_{\texttt{Path}} = \{-1\} & \mathcal{A}''_{\texttt{KPath}} = [-1,0] & \mathcal{A}''_{\texttt{Top9}} = [-1,+\infty) \end{array}$$

2. Interpretation of function symbols:

$$\begin{array}{ccc} \texttt{eq}^{\mathcal{A}''}(x,y) = x-y+1 & \texttt{seq}^{\mathcal{A}''}(x,y) = 0 & \texttt{source}^{\mathcal{A}''}(x) = -x \\ \texttt{target}^{\mathcal{A}''}(x) = -1 & \texttt{tt}^{\mathcal{A}''} = 0 & \texttt{fredsT}^{\mathcal{A}''}(x,y) = x \end{array}$$

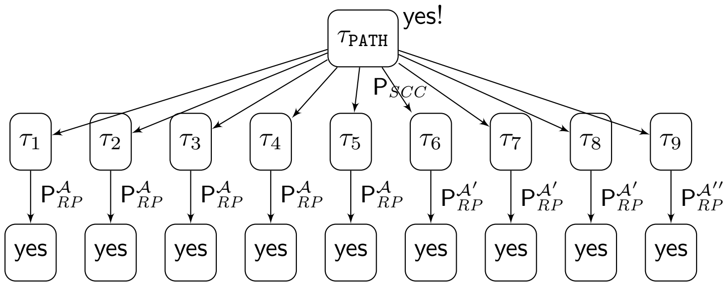
3. Interpretation of predicate symbols:

$$\begin{array}{rlrl}
\texttt{mbEdge}^{\mathcal{A}''}(x) \Leftrightarrow x \in [-1, 0] & & \texttt{mbNode}^{\mathcal{A}''}(x) \Leftrightarrow x \in [-1, 1] & \\
\texttt{mbPath}^{\mathcal{A}''}(x) \Leftrightarrow x \in [-1, 0] & & \texttt{redN}^{\mathcal{A}''}(x, y) \Leftrightarrow \textit{false} & \\
\texttt{redP}^{\mathcal{A}''}(x, y) \Leftrightarrow \textit{false} & & \texttt{redT}^{\mathcal{A}''}(x, y) \Leftrightarrow x, y \in [-1, +\infty) \wedge x >_1 y & \\
\texttt{redsN}^{\mathcal{A}''}(x, y) \Leftrightarrow x, y \in [-1, 1] & & \texttt{redsP}^{\mathcal{A}''}(x, y) \Leftrightarrow x, y \in [-1, 0] \wedge x \geq y & \\
& & \texttt{redsT}^{\mathcal{A}''}(x, y) \Leftrightarrow x, y \in [-1, +\infty) \wedge x \geq y &
\end{array}$$

$$\texttt{wfr9}^{\mathcal{A}''}(x, y) \Leftrightarrow x, y \in [-1, +\infty) \wedge x >_1 y$$

6.2 Proof of Operational Termination of PATH

Putting all together, we have the following OT-Tree for the proof:



We label the application of P_{RP} with symbols $\mathcal{A}$, $\mathcal{A}'$, and $\mathcal{A}''$ to highlight the *different ways* to apply it. By Theorem 2, PATH is operationally terminating.

7 Conclusions

The use of logical models in proofs of operational termination in the OT Framework was suggested in [14] as an possible approach to implement the new processor P_{RP} introduced in the paper. This observation was a main motivation to develop the idea of convex polytopic domain [12] as a sufficiently simple but flexible approach to obtain a variety of domains that can be used in proofs of termination and which are amenable for automation [10]. The research in this paper closes some gaps left during these developments and provides a basis for the implementation of P_{RP} in the OT Framework by means of the automatic generation of logical models for order-sorted first-order theories.

We have extended the work in [10] to achieve the automatic generation of interpretations for *predicate symbols* using *convex polytopic domains*. These results are the basis of the implementation of the tool AGES for the automatic generation of models for OS-FOL theories. To our knowledge, no systematic treatment of the generation of (homogeneous or *heterogeneous*, i.e., with arguments in different sorts) *predicate interpretations* has been attempted to date. We have also shown how to *mechanize* the use of P_{RP} in the OT Framework for proving operational termination of declarative programs by recasting it as the problem of *finding a model* through appropriate transformations.

We believe that the research in this paper is an important step towards the practical use of logical models in proofs of operational termination of programs and hence towards the implementation of a tool for automatically proving operational termination of declarative programs based on the OT Framework in [14]. This is a subject for future work.

Acknowledgments. I thank Raúl Gutiérrez for implementing the results of Sections 4 and 5.3 in AGES.

References

1. B. Alarcón, R. Gutiérrez, S. Lucas, R. Navarro-Marset. Proving Termination Properties with MU-TERM. In M. Johnson and D. Pavlovic, editors, *Proc. of the 13th International Conference on Algebraic Methodology and Software Technology, AMAST'10*, LNCS 6486:201-208, Springer-Verlag, 2011.
2. B. Alarcón, S. Lucas, R. Navarro-Marset. Using Matrix Interpretations over the Reals in Proofs of Termination. In F. Lucio, G. Moreno, R. Peña, editors, *Proc. of IX Jornadas sobre Programación y Lenguajes, PROLE'09.* pages 255-264, September 2009.
3. 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.
4. B. Cook, A. Rybalchenko, and A Podelski. Proving Program Termination. *Communications of the ACM* 54(5):88-98, 2011.
5. 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.
6. J. Goguen and J. Meseguer. Models and Equality for Logical Programming. In H. Ehrig, R.A. Kowalsky, G. Levi, and U. Montanari, editors, *Proc. of the International Joint Conference on Theory and Practice of Software Development, TAPSOFT'87*, vol. 2: Advanced Seminar on Foundations of Innovative Software Development II and Colloquium on Functional and Logic Programming and Specifications (CFLP) LNCS 250:1-22, Springer-Verlag, 1987.
7. 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.
8. R. Gutiérrez, S. Lucas, and P. Reinoso. A tool for the automatic generation of logical models of order-sorted first-order theories. *Submitted*; tool available at `http://zenon.dsic.upv.es/ages/`.
9. S. Lucas. Polynomials over the Reals in Proofs of Termination: from Theory to Practice. *RAIRO Theoretical Informatics and Applications*, 39(3):547–586, 2005.
10. S. Lucas. Synthesis of models for order-sorted first-order theories using linear algebra and constraint solving. *Electronic Proceedings in Theoretical Computer Science* 200:32-47, 2015.
11. S. Lucas, C. Marché, and J. Meseguer. Operational termination of conditional term rewriting systems. *Information Processing Letters*, 95:446–453, 2005.
12. S. Lucas and J. Meseguer. Models for Logics and Conditional Constraints in Automated Proofs of Termination. In G.A. Aranda-Corral and F.J. Martín-Mateos, editors, Proc. of *the 12th International Conference on Artificial Intelligence and Symbolic Computation, AISC'14*, LNAI 8884:7-18, Springer-Verlag, 2014.

13. S. Lucas and J. Meseguer. Operational Termination of Membership Equational Programs: the Order-Sorted Way. In G. Rosu, editor, *Proc. of the 7th International Workshop on Rewriting Logic and its Applications, WRLA'08*, *Electronic Notes in Theoretical Computer Science*, 238:207-225, 2009.
14. S. Lucas and J. Meseguer. Proving Operational Termination Of Declarative Programs In General Logics. In O. Danvy, editor, *Proc. of the 16th International Symposium on Principles and Practice of Declarative Programming, PPDP'14*, pages 111-122, ACM Digital Library, 2014.
15. J. Meseguer. General Logics. In H.-D. Ebbinghaus et al., editors, *Logic Colloquium'87*, pages 275-329, North-Holland, 1989.
16. J. Meseguer. Membership algebra as a logical framework for equational specification. In F. Parisi-Presicce, editor, *Proc. of the 12th International Workshop on Recent Trends in Algebraic Development Techniques, WADT'97*, LNCS 1376:18–61, Springer-Verlag, 1998.
17. F. Neurauter and A. Middeldorp. Revisiting Matrix Interpretations for Proving Termination of Term Rewriting. In M. Schmidt-Schauss, editor, *Proc. of the 22nd International Conference on Rewriting Techniques and Applications, RTA'11*, LIPICS 10:251-266 , 2011.
18. S. Shapiro. Foundations without Foundationalism: A Case for Second-Order Logic. Clarendon Press, 1991.