Automatic generation of logical models for order-sorted first-order theories in program analysis*

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

Computations are often viewed as proofs of specific sentences in some *computational logic* describing the operational semantics of the programming language or computational system. Since the semantics of programs (i.e., the set of such specific sentences that are provable in the logic) is usually incomputable, and most program properties undecidable, *abstraction* is essential in *program analysis*. Abstractions can be formalized as *semantic models* which should be *automatically generated* in a *push-the-button-and-wait* style of program analysis and verification. We investigate the *automatic generation* of numerical models for order-sorted first-order logics and its use in program analysis. Our development systematically uses the recently introduced *convex domains* which are well-suited for representing domains for different *sorts*; we use them to interpret the *ranked* symbols of sorted signatures by means of appropriately adapted *convex matrix interpretations*. Such *numerical* interpretations permit the use of existing algorithms and tools from linear algebra (e.g., Farkas' Lemma), real algebraic geometry, and arithmetic constraint solving in the implementation of the analyses.

Keywords: Abstraction, Logical models, Order-sorted first-order logic, Program analysis, Termination.

1 Introduction

The use of logic as a *programming framework* relies on the idea that programs are *theories of a given logic* $\mathscr{L}$ [14, page 40]. A sufficiently general and expressive framework for representing declarative programs, semantics of programming languages, and program properties is *Order-Sorted First-Order Logic* (OS-FOL), where the signature consists of (i) a set S of *sorts* (i.e., names representing sets of values) which are ordered by means of a *subsort relation* $\leq$ meaning subset inclusion, and (ii) sets $\Sigma_{w,s}$ and $\Pi_{s,s}$ of function and predicate symbols, where $s \in S$ and w is a sequence $s_1 \cdots s_k$ of sorts from S [5]. In this setting, *the user can ask questions about his/her program. Such questions, called queries, belong to a specified class of sentences in the language* [14, page 40]. Thus, computation is deduction in the *inference system* of a logic. In 2005, this view led to a general notion of *termination* of programs as the absence of infinite trees in any proof of a computation; this is called *operational termination* [10]. Recently, a framework to automatically prove termination of programs in a wide class of paradigms, including declarative [12] and imperative languages [13] has been developed. In our method, we obtain the *proof jumps* associated to the inference system $\mathscr{I}(\mathscr{S})$ which is derived from the logic $\mathscr{L}$ which is used to describe the program $\mathscr{S}$. Proof jumps are structures $A \Uparrow B_1,\ldots,B_n$ where $n > 0$ and $A, B_1,\ldots,B_n$ are *formulas* which are obtained from inference rules $\frac{B_1 \cdots B_n \cdots B_{n+p}}{A}$ in $\mathscr{I}(\mathscr{S})$ (where $p \geq 0$). Proof jumps are used to capture (infinite) paths in a proof tree T using the rules in $\mathscr{I}(\mathscr{S})$ so that there is a *jump* from an instance $\sigma(A)$ of A to an instance $\sigma(B_n)$ of B_n provided that the corresponding intances of $B_1,\ldots,B_{n-1}$ were *proved*, i.e., $\mathscr{S} \vdash \sigma(B_i)$ for all i, $1 \leq i < n$. A set of proof jumps τ is an *OT problem*. The *initial* OT problem τ_I consists of all proof jumps for $\mathscr{I}(\mathscr{S})$. Then, we apply an incremental proof methodology which

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

Submitted to:
PROLE 2015

$$\text{(Rf)}\quad \frac{}{t \to^* t} \qquad\qquad \text{(T)}\quad \frac{t \to t' \qquad t' \to^* u}{t \to^* u}$$

$$\text{(C)}\quad \frac{t_i \to t_i'}{f(t_1,\ldots,t_i,\ldots,t_k) \to f(t_1,\ldots,t_i',\ldots,t_k)} \qquad\qquad \text{(Re)}\quad \frac{}{\ell \to r}$$

$$\text{where } f \in \Sigma_{w,s},\ w = s_1,\ldots,s_k, \text{ and } 1 \leq i \leq k \qquad\qquad \text{where } \ell \to r \in \mathscr{R}$$

Figure 1: Inference rules for Order-Sorted TRSs $\mathscr{R}$

simplifies OT problems τ in a divide-and-conquer style to eventually prove termination of the program. An important technique which is used to *remove* a proof jump $\psi : A \Uparrow B_1,\ldots,B_n$ from an OT problem τ is using *well-founded orderings* $\sqsupset$: if, for all substitutions σ, whenever $\mathscr{S} \vdash \sigma(B_i)$ holds for all i, $1 \leq i < n$, we have that $\sigma(A) \sqsupset \sigma(B_n)$ holds, then we can *remove* ψ from τ. In [12] we show that *logical models* are useful for this purpose. Any model $\mathscr{A}$ of $\mathscr{S}$ satisfies the provable formulas, i.e., if $\mathscr{S} \vdash \sigma(B_i)$ holds, then, $\mathscr{A} \models \sigma(B_i)$ holds. The point is using this fact to *define* the well-founded relation $\sqsupset$. This idea is developed in [11] for a systematic treatment of proofs of termination using logical models.

In program analysis, the *automatic generation of models* is essential to (i) *simulate* computational relations associated to programs by means of a computational logic: one-step transitions $\to$, big-step transitions $\Downarrow$, etc.; and (ii) use *relations* to *implement* proof obligations in program analysis; for instance, dealing with *proof jumps* as discussed above. *Semantic structures* $\mathscr{A}'$ [8] leading to *decidable theories* $Th(\mathscr{A}')$ [16] can be used to provide an effective way to *find* logical models $\mathscr{A}$ for a program or specification $\mathscr{S}$ and also to *check* whether the properties at stake hold. This is often possible by using *theory transformations* κ from the language of $\mathscr{S}$ into the language of $Th(\mathscr{A}')$ to obtain a set of sentences $\mathscr{S}' = \kappa(\mathscr{S})$ which is then *decidable*. We formalize this view by extending the notion of *derived algebra* [6] to logical structures. Targeted languages of our transformations consist of symbols having an *intended meaning* in the structures $\mathscr{A}'$ that define the decidable theory $Th(\mathscr{A}')$. Below we introduce a running example concerning the termination analysis of an order-sorted term rewriting system. The specification is given as a module of Maude [3]. Computationally, it is viewed as an *Order-Sorted Term Rewriting System* (OS-TRS), a particular case of OS-FOL theory where only two predicate symbols $\to$ and $\to^*$ describing the one-step rewrite relation $\to$ and the zero-or-more-steps relation $\to^*$ are used, see Figure 1. The use of different (in particular *bounded*) domains for some sorts is essential to obtain a simple model which can be used to solve the verification problem. The *convex domains* introduced in [11] provide an appropriate framework to generate them.

Section 2 summarizes the basics of *order-sorted first-order logic*. Section 3 develops the notion of *derived model* and shows how to use it to deal with our running examples. Section 4 describes our approach to automation of program analysis. Section 5 explains the generation of order-sorted first-order structures based on the *convex domains* and *convex matrix interpretations* introduced in [11] and shows how to apply the technique to obtain an *automatic* solution to our case study. Section 6 concludes.

1.1 Running example: termination of an order-sorted rewrite system

The OS-TRS `ToyamaOS` in Figure 2 is based on the famous Toyama's example [19]. It shows that the use of sorts can reinforce termination (see also [20]). The unsorted version of this module is nonterminating [19]. Furthermore, if `S1` and `S2` are confused into a single sort (thus becoming a *many sorted* TRS, with only two sorts without any subsort relation between them), then `ToyamaOS` is nonterminating too:

$$\mathtt{f}(\underline{\mathtt{g(0,1)}},\mathtt{g(0,1)},\mathtt{g(0,1)}) \to \mathtt{f}(\mathtt{0},\underline{\mathtt{g(0,1)}},\mathtt{g(0,1)}) \to \underline{\mathtt{f(0,1,g(0,1))}} \to \mathtt{f}(\underline{\mathtt{g(0,1)}},\mathtt{g(0,1)},\mathtt{g(0,1)}) \to \cdots$$

```
mod ToyamaOS is
  sorts S S1 S2 .
  subsort S2 < S1 .
  op 0 : -> S2 .
  op 1 : -> S1 .
  op f : S1 S1 S1 -> S .
  op g : S1 S1 -> S1 .
  var x : S2 .
  vars  y z : S1 .
  rl f(0,1,x) => f(x,x,x) .
  rl g(y,z) => y .
  rl g(y,z) => z .
endm
```

$$\forall t : \mathtt{S}\ (t \to^* t) \tag{1}$$
$$\forall t : \mathtt{S1}\ (t \to^* t) \tag{2}$$
$$\forall t,t',u : \mathtt{S}\ (t \to t' \wedge t' \to^* u \Rightarrow t \to^* u) \tag{3}$$
$$\forall t,t',u : \mathtt{S1}\ (t \to t' \wedge t' \to^* u \Rightarrow t \to^* u) \tag{4}$$
$$\forall t_1,t_1',t_2,t_3 : \mathtt{S1}\ (t_1 \to t_1' \Rightarrow \mathtt{f}(t_1,t_2,t_3) \to \mathtt{f}(t_1',t_2,t_3)) \tag{5}$$
$$\forall t_1,t_2,t_2',t_3 : \mathtt{S1}\ (t_2 \to t_2' \Rightarrow \mathtt{f}(t_1,t_2,t_3) \to \mathtt{f}(t_1,t_2',t_3)) \tag{6}$$
$$\forall t_1,t_2,t_3,t_3' : \mathtt{S1}\ (t_3 \to t_3' \Rightarrow \mathtt{f}(t_1,t_2,t_3) \to \mathtt{f}(t_1,t_2,t_3')) \tag{7}$$
$$\forall t_1,t_1',t_2 : \mathtt{S1}\ (t_1 \to t_1' \Rightarrow \mathtt{g}(t_1,t_2) \to \mathtt{g}(t_1',t_2)) \tag{8}$$
$$\forall t_1,t_2,t_2',t_3 : \mathtt{S1}\ (t_2 \to t_2' \Rightarrow \mathtt{g}(t_1,t_2) \to \mathtt{g}(t_1,t_2')) \tag{9}$$
$$\forall x : \mathtt{S2}\ (\mathtt{f}(\mathtt{0},\mathtt{1},x) \to \mathtt{f}(x,x,x)) \tag{10}$$
$$\forall x,y : \mathtt{S1}\ (\mathtt{g}(x,y) \to x) \tag{11}$$
$$\forall x,y : \mathtt{S1}\ (\mathtt{g}(x,y) \to y) \tag{12}$$

Figure 2: Order-sorted version of Toyama's example and its associated Order-Sorted First-Order Theory

But with all sort information we can it prove it *terminating*. For instance, variable x (of sort S2) cannot be bound to terms of sort S1 which is a supersort of S2. Thus, the third step, which requires a binding $\mathtt{x} \mapsto \mathtt{g}(\mathtt{0},\mathtt{1})$, is not possible because the sort of $\mathtt{g}(\mathtt{0},\mathtt{1})$ is S1. Thus the infinite sequence is not possible.

The *order-sorted first-order theory* for the OS-TRS is also shown in Figure 2. It is obtained by specializing the inference rules in Figure 1. Sentences in Figure 2 make explicit the implicit quantification of the inference rules by taking into account the sorts in the signature and the subsort ordering. In particular, the only quantification over S2 occurs in (10). It turns out that such a quantification is crucial to obtain a simple proof of termination. In order to prove termination of this OS-TRS we need to find a model $\mathscr{A}$ for the theory in Figure 2 such that $\to$ is interpreted as a *well-founded relation* $>$. Although we do not have space to further justify this claim, it easily follows from the theory in [12].

2 Order-Sorted First-Order Logic

Sorts and Order-Sorted Signatures. 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 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 $s_1 \leq s_2$. To avoid ambiguous terms, 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 $[s] = [s']$. Throughout this paper, Σ will always be assumed *sensible*. 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 $\mathscr{X} = \{\mathscr{X}_s \mid s \in S\}$ of *mutually disjoint* sets of variables (which are also disjoint from the signature Σ), the set $\mathscr{T}_\Sigma(\mathscr{X})_s$ of terms of sort s is the least set such that (i) $\mathscr{X}_s \subseteq \mathscr{T}_\Sigma(\mathscr{X})_s$, (ii) If $s' \leq s$, then $\mathscr{T}_\Sigma(\mathscr{X})_{s'} \subseteq \mathscr{T}_\Sigma(\mathscr{X})_s$; and (iii) for each $f : s_1 \cdots s_n \to s$ and $t_i \in \mathscr{T}_\Sigma(\mathscr{X})_{s_i}$, $1 \leq i \leq n$, $f(t_1,\ldots,t_n) \in \mathscr{T}_\Sigma(\mathscr{X})_s$. If $\mathscr{X} = \varnothing$, we write $\mathscr{T}_\Sigma$ rather than $\mathscr{T}_\Sigma(\varnothing)$ for the set of *ground* terms. Terms with variables can also be seen as a special case of ground terms of the *extended* signature $\Sigma(\mathscr{X})$ where variables are considered as *constant* symbols of the apporpriate sort, i.e., $\Sigma(\mathscr{X})_{\lambda,s} = \Sigma_{\lambda,s} \cup \mathscr{X}_s$.

Example 1 *The order-sorted signature $(S,\leq,\Sigma)$ for program* `ToyamaOS` *consists of the following components:*

1. Set of sorts $S = \{\texttt{S},\texttt{S1},\texttt{S2}\}$.
2. *The* subsort relation *is the least ordering $\leq$ on S satisfying* $\texttt{S2} \leq \texttt{S1}$.
3. *Thus, $(S,\leq)$ (or $S/_{\equiv_\leq}$) consists of two* connected components*:* $[\texttt{S}] = \{\texttt{S}\}$ *and* $[\texttt{S1}] = \{\texttt{S2},\texttt{S1}\}$.
4. *Note that* $\texttt{S}$ *is the* top sort $\top_{[\texttt{S}]}$ *of* $[\texttt{S}]$*, and* $\texttt{S1}$ *is the top sort* $\top_{[\texttt{S1}]}$ *of* $[\texttt{S1}]$.
5. *The signature is* $\Sigma = \Sigma_{\texttt{S1}} \cup \Sigma_{\texttt{S2}} \cup \Sigma_{\texttt{S1}\,\texttt{S1},\texttt{S1}} \cup \Sigma_{\texttt{S1}\,\texttt{S1}\,\texttt{S1},\texttt{S}}$*, with* $\Sigma_{\texttt{S1}} = \{\texttt{1}\}$, $\Sigma_{\texttt{S2}} = \{\texttt{0}\}$, $\Sigma_{\texttt{S1}\,\texttt{S1},\texttt{S1}} = \{\texttt{g}\}$*, and* $\Sigma_{\texttt{S1}\,\texttt{S1}\,\texttt{S1},\texttt{S}} = \{\texttt{f}\}$.
6. *There is no overloaded function symbol, i.e., Σ is trivially* regular*. Furthermore, since every connected component has a top sort (see item 4), $(S,\leq,\Sigma)$ is a* coherent *signature.*

The set of variables is $\mathscr{X} = \mathscr{X}_{\texttt{S1}} \cup \mathscr{X}_{\texttt{S2}}$, with $\mathscr{X}_{\texttt{S1}} = \{\texttt{y},\texttt{z}\}$, and $\mathscr{X}_{\texttt{S2}} = \{\texttt{x}\}$.

The assumption that Σ is sensible ensures that if $[s] \neq [s']$, then $\mathscr{T}_\Sigma(\mathscr{X})_{[s]} \cap \mathscr{T}_\Sigma(\mathscr{X})_{[s']} = \varnothing$. The set $\mathscr{T}_\Sigma(\mathscr{X})$ of order-sorted terms is $\mathscr{T}_\Sigma(\mathscr{X}) = \cup_{s\in S}\mathscr{T}_\Sigma(\mathscr{X})_s$. An element of any set $\mathscr{T}_\Sigma(\mathscr{X})_s$ is called a *well-formed* term.

Order-Sorted Algebras. Given a many-sorted signature (S,Σ), an (S,Σ)-algebra $\mathscr{A}$ (or just a Σ-algebra, if S is clear from the context) is a family $\{\mathscr{A}_s \mid s \in S\}$ of sets called the *carriers* or *domains* of $\mathscr{A}$ together with a function $f^{\mathscr{A}}_{w,s} \in \mathscr{A}_w \to \mathscr{A}_s$ for each $f \in \Sigma_{w,s}$ where $\mathscr{A}_w = \mathscr{A}_{s_1} \times \cdots \times \mathscr{A}_{s_n}$ if $w = s_1 \cdots s_n$, and $\mathscr{A}_w$ 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

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

Remark 1 *Note that overloaded symbols f may be given* different *functions $f^{\mathscr{A}}_{w_1,s_1},\ldots,f^{\mathscr{A}}_{w_n,s_n}$ depending on the specific ranks $w_1 \to s_1,\ldots,w_n \to s_n$ of the overload for symbol f. Of course, such functions must still fulfill condition 2 above.*

With regard to many sorted signatures and algebras, an (S,Σ)-homomorphism between (S,Σ)-algebras $\mathscr{A}$ and $\mathscr{A}'$ is an S-sorted function $h = \{h_s : \mathscr{A}_s \to \mathscr{A}'_s \mid s \in S\}$ such that for each $f \in \Sigma_{w,s}$ with $w = s_1,\ldots,s_k$, $h_s(f^{\mathscr{A}}_{w,s}(a_1,\ldots,a_k)) = f^{\mathscr{A}'}_{w,s}(h_{s_1}(a_1),\ldots,h_{s_k}(a_k))$. If $w = \lambda$, we have $h_s(f^{\mathscr{A}}) = f^{\mathscr{A}'}$. Now, for the order-sorted case, an $(S,\leq,\Sigma)$-homomorphism $h : \mathscr{A} \to \mathscr{A}'$ between $(S,\leq,\Sigma)$-algebras $\mathscr{A}$ and $\mathscr{A}'$ is an (S,Σ)-homomorphism that satisfies the following additional condition: if $s \leq s'$ and $a \in \mathscr{A}_s$, then $h_s(a) = h_{s'}(a)$.

The family of *domains* $\{\mathscr{T}_\Sigma(\mathscr{X})_s\}_{s\in S}$ together with the operations $f : (t_1,\ldots,t_n) \mapsto f(t_1,\ldots,t_n)$ define an order-sorted Σ-algebra called the *free* algebra on $\mathscr{X}$ and denoted $\mathscr{T}_\Sigma(\mathscr{X})$. When $\mathscr{X} = \varnothing$, $\mathscr{T}_\Sigma = \mathscr{T}_\Sigma(\varnothing)$ denotes the *initial* Σ-algebra, i.e., an algebra having a unique homomorphism $h_{\mathscr{A}} : \mathscr{T}_\Sigma \to \mathscr{A}$ to each Σ-algebra $\mathscr{A}$. Similarly, $\mathscr{T}_\Sigma(\mathscr{X})$ (itself a Σ-algebra) is initial in the class of all $\Sigma(\mathscr{X})$-algebras.

Predicates and connectives. Following [5], an order-sorted signature *with predicates* is a quadruple $(S,\leq,\Sigma,\Pi)$ 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, ... We write $P : w$ for $P \in \Pi_w$. Overloading is also allowed on predicates with the following conditions [5, Definition 11]:

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 $t_1,\ldots,t_n \in \mathscr{T}_\Sigma(\mathscr{X})_w$, logic connectives (e.g., $\wedge$, $\neg$) and quantifiers ($\forall$) as follows: (i) if $P \in \Pi_w$, $w = s_1\cdots s_n$, and $t_i \in \mathscr{T}_\Sigma(\mathscr{X})_{s_i}$ for all i, $1 \leq i \leq n$, then $P(t_1,\ldots,t_n) \in Form_{\Sigma,\Pi}$. (ii) if $\varphi \in Form_{\Sigma,\Pi}$, then $\neg\varphi \in Form_{\Sigma,\Pi}$; (iii) if $\varphi,\varphi' \in Form_{\Sigma,\Pi}$, then $\varphi \wedge \varphi' \in Form_{\Sigma,\Pi}$; (iv) if $s \in S$, $x \in \mathscr{X}_s$, and $\varphi \in Form_{\Sigma,\Pi}$, then $(\forall x : s)\varphi \in Form_{\Sigma,\Pi}$. 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*.

Remark 2 *In order to define an order-sorted signature with predicates that can be used to reason about rewritings with OS-TRSs, we have to provide (at least) as many overloads for the computational relation* $\to^*$ *as connected component* $[s]$ *in* $S/_{\equiv_\leq}$*: due to axiom (Rf), OS-TRSs are expected to rewrite with* $\to^*$ *any of the classes* $\mathscr{T}_\Sigma(\mathscr{X})_{[s]}$ *for every connected component* $[s]$*. By coherence of the signature, we can just let* $\to^* \in \Pi_{\top_{[s]}\top_{[s]}}$ *for all* $s \in S$*. Then, rule (T) requires a corresponding overload for* $\to$ *as well. By coherence of the signature, we can just let* $\Pi_{\top_{[s]}\top_{[s]}} = \{\to,\to^*\}$ *for all* $s \in S$*. This will be compatible with any possible instance of rule (Re) because terms* ℓ *and* r *in rewrite rules* $\ell \to r$ *of OS-TRSs must be terms belonging to* $\mathscr{T}_\Sigma(\mathscr{X})_{[s]}$ *for some* $s \in S$*. By coherence, we know that* $\ell, r \in \mathscr{T}_\Sigma(\mathscr{X})_{\top_{[s]}}$ *for some* $s \in S$*.*

Example 2 *The order-sorted signature* $(S,\leq,\Sigma)$ *described in Example 1 is extended into a order-sorted signature with predicates* $(S,\leq,\Sigma,\Pi)$ *where* $\Pi = \Pi_{\mathtt{S}\,\mathtt{S}} \cup \Pi_{\mathtt{S1}\,\mathtt{S1}}$ *for* $\Pi_{\mathtt{S}\,\mathtt{S}} = \Pi_{\mathtt{S1}\,\mathtt{S1}} = \{\to,\to^*\}$*, which are the only nonempty sets of predicate symbols. They satisfy the regularity condition.*

Theories, specifications and programs. A *theory* $\mathscr{S}$ of Σ,Π is a set of formulas, $\mathscr{S} \subseteq Form_{\Sigma,\Pi}$, and its *theorems* are the formulas $\varphi \in Form_{\Sigma,\Pi}$ for which we can derive a proof using an appropriate inference system $\mathscr{I}(\mathscr{L})$ of a logic $\mathscr{L}$ in the usual way (written $\mathscr{S} \vdash \varphi$). Given a logic $\mathscr{L}$ describing computations in a (declarative) programming language, programs are viewed as *theories* $\mathscr{S}$ of $\mathscr{L}$.

Example 3 *In the logic of OS-TRSs, with binary (overloaded)* predicates $\to$ *and* $\to^*$*, the theory for an OS-TRS* $\mathscr{R} = (S,\leq,\Sigma,R)$ *with set of rules R (for instance, our running example) is obtained from the* schematic *inference rules in Figure 1 after* specializing *them as* $(C)_{f,i}$ *for each* $f \in \mathscr{F}$ *and i,* $1 \leq i \leq ar(f)$ *and* $(Re)_\rho$ *for all* $\rho : \ell \to r \in R$*. Then, inference rules* $\frac{B_1,\ldots,B_n}{A}$ *become* implications $B_1 \wedge \cdots \wedge B_n \Rightarrow A$*. For instance, with regard to the sentences for* `ToyamaOS` *in Figure 2:*

- *Sentences (1) and (2) specialize (Rf) in Figure 1 for the two overloads of* $\to^*$ *in* $\Pi_{\mathtt{S}\,\mathtt{S}}$ *and* $\Pi_{\mathtt{S1}\,\mathtt{S1}}$*, respectively.*
- *Sentences (3) and (4) specialize (T) for the overloads of* $\to^*$ *and* $\to$ *in* $\Pi_{\mathtt{S}\,\mathtt{S}}$ *and* $\Pi_{\mathtt{S1}\,\mathtt{S1}}$*, respectively.*
- *Sentences (5), (6), and (7) specialize (C) for symbol* `f` *using the appropriate overloads of* $\to$ *in* $\Pi_{\mathtt{S}\,\mathtt{S}}$ *and* $\Pi_{\mathtt{S1}\,\mathtt{S1}}$ *according to the rank of* `f`*. Similarly, (8) and (9) specialize (C) for symbol* `g`.

- *Sentences (10), (11), and (12) specialize (Re) for each rewrite rule in* `ToyamaOS`.

Note that, according to the variable declaration for `x` *in* `ToyamaOS`*, in sentence (10) variable x ranges on values of sort* `S2` *only.*

Structures, Satisfaction, Models. Given an order-sorted signature with predicates $(S,\leq,\Sigma,\Pi)$, an $(S,\leq,\Sigma,\Pi)$-*structure*[1] (or just a Σ,Π-structure) is an order-sorted $(S,\leq,\Sigma)$-algebra $\mathscr{A}$ together with an assignment to each $P\in\Pi_w$ of a subset $P_w^{\mathscr{A}}\subseteq\mathscr{A}_w$ such that [5]: (i) for P the identity predicate $\_=\_ : ss$, the assignment is the identity relation, i.e., $(=)_{\mathscr{A}}=\{(a,a)\mid a\in\mathscr{A}_s\}$; and (ii) whenever $P:w_1$ and $P:w_2$ and $w_1\leq w_2$, then $P_{w_1}^{\mathscr{A}}=\mathscr{A}_{w_1}\cap P_{w_2}^{\mathscr{A}}$.

Let $(S,\leq,\Sigma,\Pi)$ be an order-sorted signature with predicates and $\mathscr{A},\mathscr{A}'$ be $(S,\leq,\Sigma,\Pi)$-structures. Then, an $(S,\leq,\Sigma,\Pi)$-*homomorphism* $h:\mathscr{A}\to\mathscr{A}'$ is an $(S,\leq,\Sigma)$-homomorphism such that, for each $P:w$ in Π, if $(a_1,\ldots,a_n)\in P_w^{\mathscr{A}}$, then $h(a_1,\ldots,a_n)\in P_w^{\mathscr{A}'}$.

Given an S-sorted *valuation mapping* $\alpha:\mathscr{X}\to\mathscr{A}$, the evaluation mapping $[\_]_{\mathscr{A}}^{\alpha}:\mathscr{T}_{\Sigma}(\mathscr{X})\to\mathscr{A}$ is the unique $(S,\leq,\Sigma)$-homomorphism extending α [7]. Finally, $[\_]_{\mathscr{A}}^{\alpha}:\mathit{Form}_{\Sigma,\Pi}\to\mathit{Bool}$ is given by:

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

We say that $\mathscr{A}$ *satisfies* $\varphi\in\mathit{Form}_{\Sigma,\Pi}$ if there is $\alpha\in\mathscr{X}\to\mathscr{A}$ such that $[\varphi]_{\mathscr{A}}^{\alpha}=\mathsf{true}$. If $[\varphi]_{\mathscr{A}}^{\alpha}=\mathsf{true}$ for *all* valuations α, we write $\mathscr{A}\models\varphi$ and say that $\mathscr{A}$ is a *model* of φ [8, page 12]. Initial valuations are not relevant for establishing the satisfiability of *sentences*; thus, both notions coincide on them. We say that $\mathscr{A}$ is *a model of a set of sentences* $\mathscr{S}\subseteq\mathit{Form}_{\Sigma,\Pi}$ (written $\mathscr{A}\models\mathscr{S}$) if for all $\varphi\in\mathscr{S}$, $\mathscr{A}\models\varphi$. And, given a sentence φ, we write $\mathscr{S}\models\varphi$ if and only if for *all models* $\mathscr{A}$ of $\mathscr{S}$, $\mathscr{A}\models\varphi$. *Sound* logics guarantee that every provable sentence φ is true in *every model* of $\mathscr{S}$, i.e., $\mathscr{S}\vdash\varphi$ implies $\mathscr{S}\models\varphi$.

3 Derived models

By a *decidable theory* T in a given language (often a fragment of first-order logic) we mean one having a *decision procedure* which can be used to establish whether a given formula φ belongs to T [16]. In some cases such theories can be presented as *axiomatizations* of algebraic structures $\mathscr{A}$ so that $T=Th(\mathscr{A})=\{\varphi\mid\mathscr{A}\models\varphi\}$. We often say that $\mathscr{A}$ is the *intended model* of T [8, page 32].

Example 4 *Presburger's arithmetic (or arithmetic without multiplication) can be seen as the set of sentences of the language $L_P=\{\mathsf{0},',+,>\}$ which are true in the standard interpretation $\mathscr{N}$ of the natural numbers [2, page 295]. It is well-known that $P=Th(\mathscr{N})$ is decidable.*

Assume that $(S',\leq',\Sigma',\Pi')$ is an order-sorted signature with predicates and $\mathscr{A}'$ is a Σ',Π'-structure such that $T=Th(\mathscr{A}')$ is *decidable*. We can define an $(S,\leq,\Sigma,\Pi)$-model for $\mathscr{S}\subseteq\mathsf{Form}_{\Sigma,\Pi}$ by means of a map (theory transformation) $\kappa:\mathsf{Form}_{\Sigma,\Pi}\to\mathsf{Form}_{\Sigma',\Pi'}$. If $\mathscr{S}$ is finite, then, it is *decidable* whether $\kappa(\mathscr{S})\subseteq T$. If $\kappa(\mathscr{S})\subseteq T$, then $\mathscr{A}'\models\kappa(\mathscr{S})$, i.e., the Σ',Π'-structure $\mathscr{A}'$ is a model of $\kappa(\mathscr{S})$. If we can define κ on a purely syntactic basis, i.e., as homomorphic extensions of maps from the syntactic components S, Σ, and Π in $(S,\leq,\Sigma,\Pi)$, then we are able to make $\mathscr{A}'$ into a *derived* Σ,Π-structure $\mathscr{A}$ so that $\mathscr{A}$ is a model of $\mathscr{S}$, i.e., $\mathscr{A}\models\mathscr{S}$, as desired. In the following, we further develop this methodology.

[1]As in [8], we use 'structure' and reserve the word 'model' to refer those structures satisfying a given theory.

3.1 Derived algebras and structures

Appropriate Σ-algebras can be obtained as *derived algebras* if we first consider a *new* signature Σ' of symbols with 'intended' (often arithmetic) interpretations.

Definition 1 (Derivor and Derived algebra) [6, Definition 11] *Let $\Sigma = (S, \leq, \Sigma)$ and $\Sigma' = (S', \leq', \Sigma')$ be order-sorted signatures. A* derivor *from Σ to Σ' is a monotone function $\tau : S \to S'$ (i.e., such that for all $s, s' \in S$, $s \leq s'$ implies*[2] *$\tau(s) \leq' \tau(s')$) and a family $d_{w,s} : \Sigma_{w,s} \to (\mathscr{T}_{\Sigma})_{\tau(w),\tau(s)}$, where $\tau(s_1, \ldots, s_k) = \tau(s_1), \ldots, \tau(s_k)$ and where $(\mathscr{T}_{\Sigma})_{\tau(w),\tau(s)}$ denotes the set of all Σ'-terms using variables $\{y_1, \ldots, y_k\}$ with y_i of sort $\tau(s_i)$. Each operation symbol $f \in \Sigma_{w,s}$ is expressed using a derived operation $d_{w,s}(f)$ of the appropriate arity. We often use d to denote a derivor $\langle \tau, d \rangle$. Now, let $\mathscr{A}'$ be an Σ'-algebra. Then, the d-derived algebra $d\mathscr{A}'$ of $\mathscr{A}'$ is the Σ-algebra with carriers $(d\mathscr{A}')_s = \mathscr{A}'_{\tau(s)}$ for all $s \in S$; and mappings $f^{d\mathscr{A}'}$ for each $f \in \Sigma$ defined to be $(d(f))^{\mathscr{A}'}$, the derived operator of the Σ'-term $d(f)$.*

Note that d in Definition 1 is *homomorphically extended* into a mapping $d : \mathscr{T}_{\Sigma}(\mathscr{X}) \to \mathscr{T}_{\Sigma'}(\mathscr{X}')$.

Example 5 *Let $(S, \leq, \Sigma)$ as in Example 1. Let $S' = \{\mathsf{zero}, \mathsf{nat}\}$ with subsort relation $\leq'$ given by $\mathsf{zero} \leq' \mathsf{nat}$, and $\Sigma' = \Sigma'_{\lambda,\mathsf{zero}} \cup \Sigma'_{\lambda,\mathsf{one}} \cup \Sigma'_{\mathsf{nat}^2\mathsf{nat}}$ where $\Sigma_{\lambda,\mathsf{zero}} = \{0\}$, $\Sigma_{\lambda,\mathsf{nat}} = \{1\}$, and $\Sigma_{\mathsf{nat}^2\mathsf{nat}} = \{+\}$. We define a derivor from $(S, \leq, \Sigma)$ to $(S', \leq', \Sigma')$ by $\tau(\mathtt{S}) = \tau(\mathtt{S1}) = \mathsf{nat}$ and $\tau(\mathtt{S2}) = \mathsf{zero}$; also, $d(\mathtt{0}) = 0$, $d(\mathtt{1}) = 1$, $d(\mathtt{f}) = x + y + z$, and $d(g) = x + y + 1$. Let $\mathscr{A}'$ be the $(S', \leq', \Sigma')$ algebra given by $\mathscr{A}'_{\mathsf{zero}} = \{0\}$ and $\mathscr{A}'_{\mathsf{nat}} = \mathbb{N}$ together with the* standard *interpretations for* 0, 1, *and* +. *The derived $(S, \leq, \Sigma)$-algebra $\mathscr{A} = d\mathscr{A}'$ is given by $\mathscr{A}_{\mathtt{S2}} = \mathscr{A}'_{\mathsf{zero}} = \{0\}$ and $\mathscr{A}_{\mathtt{S}} = \mathscr{A}_{\mathtt{S1}} = \mathscr{A}'_{\mathsf{nat}} = \mathbb{N}$, together with the derived interpretations for each symbol in Σ.*

A slight generalization of Definition 1 leads to the notion of *derived structure*.

Definition 2 (Derivor for signatures with predicates / Derived structure) *Let $\Sigma = (S, \leq, \Sigma, \Pi)$ and $\Sigma' = (S', \leq', \Sigma', \Pi')$ be order-sorted signatures with predicates and $\langle \tau, d \rangle$ be a derivor from $(S, \leq, \Sigma)$ to $(S', \leq', \Sigma')$. We extend d to predicate symbols by adding a component $d : \Pi \to \mathsf{Form}_{\Sigma'\Pi'}$ such that for all $P \in \Pi_w$, with $w = s_1 \cdots s_n$, $d(P)$ is an* atom *$P'(t'_1, \ldots, t'_m)$ with $P' \in \Pi'_{w'}$, and terms $t'_1, \ldots, t'_m \in \mathscr{T}_{\Sigma'}(\mathscr{X})$ only use variables $\{y_1, \ldots, y_n\}$ with y_i of sort $\tau(s_i)$. In this new context we also call $\langle \tau, d \rangle$ a derivor. Let $\mathscr{A}' = (\mathscr{A}', \Sigma_{\mathscr{A}'}, \Pi'_{\mathscr{A}'})$ be an $(S', \leq', \Sigma', \Pi')$-structure and $\mathscr{A}'_0 = (\mathscr{A}', \Sigma_{\mathscr{A}'})$ be the underlying $(S', \leq', \Sigma')$-algebra. Then, the $\langle \tau, d \rangle$-derived structure $d\mathscr{A}'$ of $\mathscr{A}'$ is the $(S, \leq, \Sigma, \Pi)$-structure that consists of the Σ-algebra $d\mathscr{A}'_0$ with S-sorted set of carriers $\mathscr{A}$ together with interpretations $P_w^{d\mathscr{A}'}$ (for $P \in \Pi_w$) defined to be*

$$
\begin{array}{rcl}
P_w^{d\mathscr{A}'} & = & \{([t_1]^{\alpha}_{d\mathscr{A}'}, \ldots, [t_n]^{\alpha}_{d\mathscr{A}'}) \mid (t_1, \ldots, t_n) \in \mathscr{T}_{\Sigma}(\mathscr{X})_w, \alpha \in \mathscr{X} \to \mathscr{A}, \\
& & \quad d(P) = P'(t'_1, \ldots, t'_m), \mathscr{Y} = \mathscr{V}ar(t'_1, \ldots, t'_m), \sigma(y_i) = t_i, 1 \leq i \leq n \\
& & \quad \exists \alpha' : \mathscr{Y} \to \mathscr{A}'([\sigma(t'_1)]^{\alpha'}_{\mathscr{A}'}, \ldots, [\sigma(t'_m)]^{\alpha'}_{\mathscr{A}'}) \in (P')^{\mathscr{A}'}\}
\end{array}
$$

Note that $\langle \tau, d \rangle$ can be seen now as a transformation $d : \mathit{Form}_{\Sigma,\Pi} \to \mathit{Form}_{\Sigma',\Pi'}$:

$$
\begin{array}{rcl}
d(P(t_1, \ldots, t_n)) & = & d(P)[y_1 \mapsto d(t_1), \ldots, y_n \mapsto d(t_n)] \\
d(\neg\varphi) & = & \neg d(\varphi) \\
d(\varphi \wedge \varphi') & = & d(\varphi) \wedge d(\varphi') \\
d((\forall x : s)\varphi) & = & (\forall x : \tau(s)) d(\varphi)
\end{array}
$$

The following obvious result formalizes the use of the previous construction.

[2] Monotonicity is *not* required in [6] where only many-sorted signatures are considered.

$$\begin{array}{rcr}
\forall t \in \mathscr{A}_{\mathtt{S}} & (t \geq t) & (13)\\
\forall t \in \mathscr{A}_{\mathtt{S1}} & (t \geq t) & (14)\\
\forall t,t',u \in \mathscr{A}_{\mathtt{S}} & (t > t' \wedge t' \geq s \Rightarrow t \geq s) & (15)\\
\forall t,t',u \in \mathscr{A}_{\mathtt{S1}} & (t > t' \wedge t' \geq s \Rightarrow t \geq s) & (16)\\
\forall t_1,t_1',t_2,t_3 \in \mathscr{A}_{\mathtt{S1}} & (t_1 > t_1' \Rightarrow t_1+t_2+t_3 > t_1'+t_2+t_3) & (17)\\
\forall t_1,t_2,t_2',t_3 \in \mathscr{A}_{\mathtt{S1}} & (t_2 > t_2' \Rightarrow t_1+t_2+t_3 > t_1+t_2'+t_3) & (18)\\
\forall t_1,t_2,t_3,t_3' \in \mathscr{A}_{\mathtt{S1}} & (t_3 > t_3' \Rightarrow t_1+t_2+t_3 > t_1+t_2+t_3') & (19)\\
\forall t_1,t_1',t_2 \in \mathscr{A}_{\mathtt{S1}} & (t_1 > t_1' \Rightarrow t_1+t_2+1 > t_1'+t_2+1) & (20)\\
\forall t_1,t_2,t_2',t_3 \in \mathscr{A}_{\mathtt{S1}} & (t_2 > t_2' \Rightarrow t_1+t_2+1 > t_1+t_2'+1) & (21)\\
\forall x \in \mathscr{A}_{\mathtt{S2}} & (0+1+x > x+x+x) & (22)\\
\forall x,y \in \mathscr{A}_{\mathtt{S1}} & (x+y+1 > x) & (23)\\
\forall x,y \in \mathscr{A}_{\mathtt{S1}} & (x+y+1 > y) & (24)
\end{array}$$

Figure 3: Derived sentences for the sentences in Figure 2

Theorem 1 *Let $\Sigma = (S, \leq, \Sigma, \Pi)$ and $\Sigma' = (S', \leq', \Sigma', \Pi')$ be order-sorted signatures with predicates and $\langle \tau, d \rangle$ be a derivor from $(S, \leq, \Sigma, \Pi)$ to $(S', \leq', \Sigma', \Pi')$. Let $\mathscr{A}'$ be an $(S', \leq', \Sigma', \Pi')$-structure and $\varphi \in \mathsf{Form}_{\Sigma,\Pi}$. If $\mathscr{A}' \models d(\varphi)$, then $d\mathscr{A}' \models \varphi$.*

The following corollary of Theorem 1 formalizes our approach of seeking models of theories through derived structures.

Corollary 1 (Derived model) *Let $\Sigma = (S, \leq, \Sigma, \Pi)$ and $\Sigma' = (S', \leq', \Sigma', \Pi')$ be order-sorted signatures with predicates and $\langle \tau, d \rangle$ be a derivor from $(S, \leq, \Sigma, \Pi)$ to $(S', \leq', \Sigma', \Pi')$. Let $\mathscr{A}'$ be an $(S', \leq', \Sigma', \Pi')$-structure and $\mathscr{S} \subseteq Forms_{\Sigma,\Pi}$ be a theory. If for all $\varphi \in \mathscr{S}$, $\mathscr{A}' \models d(\varphi)$, then $d\mathscr{A}' \models \mathscr{S}$.*

The following example shows how to use Corollary 1 together with an *appropriate* derived model for proving termination of the OS-TRS `ToyamaOS` in our running example.

Example 6 *For the OS-TRS in Figure 2, we use a logical model with the derived algebra in Example 5 and predicates $\to$ and $\to^*$ that are interpreted by $>$ and $\geq$ (over the naturals), respectively. This model satisfies the sentences in Figure 3 that translate the sentences (1)-(12) in Figure 2. The validity of (13)-(21) and (23)-(24) is obvious because $\mathscr{A}_{\mathtt{S}} = \mathscr{A}_{\mathtt{S1}} = \mathbb{N}$ and by reflexivity and transitivity of $\geq$ and the fact that $> \subseteq \geq$. With regard to (22), it holds due to our specific choice for $\mathscr{A}_{\mathtt{S2}}$: since $\mathscr{A}_{\mathtt{S2}} = \{0\}$, x is restricted to take value 0; thus, the condition $0+1+x > x+x+x$ becomes $1 > 0$, which is trivially true. Since $>$ is a well-founded relation over $\mathscr{A}_{\mathtt{S}}$ and $\mathscr{A}_{\mathtt{S1}}$, termination of* `ToyamaOS` *is proved.*

Note that the model in Example 6 is based on a *decidable theory*, namely, Presburger's arithmetic (see Example 4). Also, note that the interpretation of the one-step rewriting predicate $\to$ has been *chosen* to be a well-founded ordering, which is essential to conclude termination of `ToyamaOS` from the fact that $\mathscr{A}$ is a model of the sentences in Figure 3.

4 Constraint-solving and automation of the analyses

The *automatic generation* of models for a theory $\mathscr{S}$ is a *bottom-up* process where things remain 'unspecified' until an attempt to *solve* some constraints obtained from $\mathscr{S}$ succeeds. The *solution* is then used to synthesize a structure which is (by construction) a model of $\mathcal{S}$. This is accomplished as follows:

1. The syntactic objects are given *parametric interpretations* of a given type, usually chosen according to their amenability to automation. For instance, function symbols are given *linear polynomials* $a_1x_1 + a_2x_2 + \cdots + a_kx_k + a_0$, where $a_0, a_1, \ldots, a_k$ are *parameters* which are assumed to be *existentially quantified* in any formula during the generation process and variables $x_1, \ldots, x_k$ (of sorts $s_1, \ldots, s_k$) range on the interpretation domains $\mathcal{A}_{s_i}$ for $1 \leq i \leq k$.
2. *Sentences* $\varphi \in \mathcal{S}$ are used to obtain a new set $\mathcal{S}'$ of *parametric* sentences $\exists\varphi^\sharp$ with existentially quantified parameters $a_1, \ldots, a_n$. Such parameters range over appropriate (constraint solving) domains $D_1, \ldots, D_n$.
3. Then, $\mathcal{S}'$ is treated as a *constraint* whose solutions $\sigma = \{a_i \mapsto d_i \mid 1 \leq i \leq n\}$, with $d_i \in D_i$ for $1 \leq i \leq n$, make $\sigma(\varphi^\sharp)$ (an instantiation of the parameters in $\varphi^\sharp$) *true*.

In the realm of this paper, the *parameterization step* (item (1) above) is part of the definition of *derivors* (Definitions 1 and 2).

Then, as remarked in item (2) above, the original theory $\mathcal{S}$ is transformed into a *derived theory* $\mathcal{S}'$. In this paper $\mathcal{S}'$ consists of arithmetic sentences, using numeric orderings as predicates. Actually, an important issue is handling parametric formulas containing *implications* of the form

$$\bigwedge_{j=1}^{p_i} e_{ij} \geq d_{ij} \Rightarrow e_i \geq d_i \tag{25}$$

where for all $i \in \{1, \ldots, k\}$, $p_i > 0$ and for all j, $1 \leq j \leq p_i$, e_{ij} and e_i are *linear expressions* of the form $\sum a_k x_k$ for numbers a_k and variables x_k, and $d_{ij}, d_i \in \mathbb{R}$. Implications following the format (25) are said to be in *affine form*. They are obtained as derived formulas from the theory at stake (e.g., the theory in Figure 2). In this setting, the Affine form of Farkas' Lemma considered in [11, Section 5.1] is useful. In general, given $\vec{c} \in \mathbb{R}^n$ and $\beta \in \mathbb{R}$, the affine form of Farkas' Lemma can be used to check whether a constraint $\vec{c}^T\vec{x} \geq \beta$ holds whenever $\vec{x}$ ranges on the set S of solutions $\vec{x} \in \mathbb{R}^n$ of a linear system $A\vec{x} \geq \vec{b}$ of k inequalities, i.e., A is a matrix of k rows and n columns and $\vec{b} \in \mathbb{R}^k$. According to Farkas' Lemma, we have to *find* a vector $\vec{\lambda}$ of k non-negative numbers $\vec{\lambda} \in \mathbb{R}^k_0$ such that $\vec{c} = A^T\vec{\lambda}$ and $\vec{\lambda}^T\vec{b} \geq \beta$.

Farkas' Lemma permits the *removal* of all variables $\vec{x}$ and the transformation of the conditional constraint into a set of equalities and inequalities that, as indicated in item (3) above, can be handled by means of tools for arithmetic *constraint solving* like MULTISOLVER[3]. Then, we obtain a model for $\mathcal{S}$. The following section provides a complete account of this process using our running example.

5 Order-sorted structures with convex domains

The resolution of our running example (Example 6) shows that flexibility in the definition of domains $\mathcal{A}_s$ for sorts $s \in S$ is an asset: we have *simultaneously used* (due to the presence of sorts) an infinite domain like $\mathbb{N}$ (which is typical in termination proofs) and the finite domain $\{0\}$. In order to provide an appropriate computational basis to the *automatic* definition of algebras and structures that can be used in program analysis with order-sorted first-order specifications, we follow [11] and focus on domains that are obtained as the solution of polynomial and specially *linear* constraints.

Definition 3 (Convex polytopic domain) [11, Definition 1] *Given a matrix* $\mathsf{C} \in \mathbb{R}^{m\times n}$*, and* $\vec{b} \in \mathbb{R}^m$*, the set* $D(\mathsf{C}, \vec{b}) = \{\vec{x} \in \mathbb{R}^n \mid \mathsf{C}\vec{x} \geq \vec{b}\}$ *is called a* convex polytopic domain.

[3]`http://zenon.dsic.upv.es/multisolver/`

Convex domains in Definition 3 can be parameterized by considering a subset $N \subseteq \mathbb{R}$ (e.g., $\mathbb{N}$, $\mathbb{Z}$, $\mathbb{Q}$, etc.) with $\mathsf{C} \in N^{m\times n}$, and $\vec{b} \in N^m$ and defining $D_N(\mathsf{C},\vec{b}) = \{\vec{x} \in N^n \mid \mathsf{C}\vec{x} \geq \vec{b}\}$.

Example 7 Intended *interpretations $\mathscr{A}^s$ for some usual sorts s as convex domains $\mathscr{A}_s = D(\mathsf{C}^s,\vec{b}^s)$ are:*

Sort	C^s	$\vec{b}^s$	$\mathscr{A}_s = D(\mathsf{C}^s,\vec{b}^s)$
$\varnothing$	(0)	(1)	$\varnothing$
Nat	(1)	(0)	$[0,+\infty)$
NzNat	(1)	(1)	$[1,+\infty)$
Zero	$(1,-1)^T$	$(0,0)^T$	$\{0\}$
Bool	$(1,-1)^T$	$(0,-1)^T$	$[0,1]$
Char	$(1,-1)^T$	$(0,-255)^T$	$[0,255]$

We discuss the automatic generation of structures based on convex polytopic domains according to the general scheme in Section 4. We illustrate the develoment by using our running example.

5.1 Domains

We interpret sorts $s \in S$ as convex domains $\mathscr{A}_s = D(\mathsf{C}^s,\vec{b}^s)$, where[4] $\mathsf{C}^s \in \mathbb{R}^{m_s\times n_s}$ is an $m_s \times n_s$-matrix and $\vec{b}^s \in \mathbb{R}^{m_s}$. Thus, $\mathscr{A}_s \subseteq \mathbb{R}^{n_s}$. Given $s \in S$, we have to *fix* m_s and n_s according to some criterion. Then, matrices C^s and vectors $\vec{b}^s$ can be written *parametrically*. The exact shape of $D(\mathsf{C}^s,\vec{b}^s)$ will be settled by the subsequent *constraint solving process*.

Remark 3 *For 1-dimensional convex domains $D(\mathsf{C}^s,\vec{b}^s) \subseteq \mathbb{R}$ (i.e., intervals, with $n_s = 1$), imposing $0 < m_s \leq 2$ is appropriate because the existence of more than 2 rows in C^s for a given entry in $\vec{b}^s$ is useless: they define the same interval that those producing the least and bigger values when applying them to $\vec{x}$. In general, if $m_s = 2$, then $\mathsf{C}^s = (C^s_1,C^s_2)^T$ and $\vec{b}^s = (b^s_1,b^s_2)^T$ means that $C^s_1x \geq b^s_1$ and $C^s_2x \geq b^s_2$. As shown in Example 7, fixing $m_s = 2$ and using $\mathbb{Z}$ as domain for parameters b_i and c_i is important to gain flexibility in the definition of convex domains, especially if bounded domains are desirable. Our choice, in this 1-dimensional case is $m_s = 2$ and $n_s = 1$.*

Since we deal with three different sorts `S`, `S1`, and `S2`, we have to consider three convex domains:

$$\mathscr{A}_{\mathtt{S}} = D(\mathsf{C}^S,\vec{b}^S) \qquad \mathscr{A}_{\mathtt{S1}} = D(\mathsf{C}^{S1},\vec{b}^{S1}) \qquad \mathscr{A}_{\mathtt{S2}} = D(\mathsf{C}^{S2},\vec{b}^{S2})$$

where $\mathsf{C}^S,\mathsf{C}^{S1},\mathsf{C}^{S2} \in \mathbb{R}^{2\times 1}$ and $\vec{b}^S,\vec{b}^{S1},\vec{b}^{S2} \in \mathbb{R}^2$. In the following, we investigate how to guarantee that convex domains satisfy typical requirements in termination (and other program) analysis.

5.1.1 Non-empty convex domains

An important requirement in termination analysis is that the domain $D(\mathsf{C},\vec{b}) \subseteq \mathbb{R}^n$ where a well-founded relation $>$ is to be defined should *not be empty*. At the syntactic level we guarantee this by just adding a *fresh* constant `k` of the appropriate sort `S` (to be interpreted by $D(\mathsf{C},\vec{b})$) in the signature: `k : S`. Of course, if such a constant is already part of the specification, nothing else is required. At the *derived* level this becomes a (vectorial) constraint $\mathsf{C}k^T \geq \vec{b}$ to be satisfied by a *dummy* constant $k \in \mathbb{R}^n$. Thus, we obtain our first constraints for our running example:

$$C^S_1k \geq b^S_1 \wedge C^S_2k \geq b^S_2 \tag{26}$$

$$C^{S1}_1k' \geq b^{S1}_1 \wedge C^{S1}_2k' \geq b^{S1}_2 \tag{27}$$

[4]In the following, we use write the sort s in the superscript of the matrix and vector components C and $\vec{b}$ of the convex domain. In this way, we can use the subscripts to identify their *components*: rows, columns, etc.

where k and k' are *dummy* elements $k, k' \in \mathbb{R}$ for `S` and `S1`. According to our previous discussion, and since `ToyamaOS` already includes a symbol `1` of sort `S1`, constraint (27) is not really necessary and could be avoided (see constraint (34) below). And, although there is no constant symbol of sort `S`, function `f` takes arguments of sort `S1` (which is not empty) and yields a term of sort `S`. Thus, sort `S` is not empty; this is guaranteed by means of other constraints like (35)-(36) below. Thus, (26) could be avoided too.

5.1.2 Convex domains which are bounded from below

In termination analysis, we need to use non-empty well-founded domains $\mathscr{A}$. Well-foundedness of a set is *not* expressible in first-order logic. However, at our *derived level*, we can guarantee well-foundendness of $\mathscr{A}$ if $\mathscr{A} = D(\mathsf{C}, \vec{b})$ is a convex domain. If $D(\mathsf{C}, \vec{b})$ is a subset of natural numbers, then the usual ordering $>_{\mathbb{N}}$ over the naturals guarantees well-foundedness. If $D(\mathsf{C}, \vec{b})$ is a subset of $\mathbb{R}$, then the use of the ordering $>_\delta$ for some $\delta > 0$ (where, for all $x, y \in \mathbb{R}$, we let $x >_\delta y$ iff $x \geq y + \delta$ [9]) guarantees well-foundedness provided that $D(\mathsf{C}, \vec{b})$ is *bounded from below*, i.e., there is a lower bound α for $D(\mathsf{C}, \vec{b})$, i.e., $D(\mathsf{C}, \vec{b}) \subseteq [\alpha, +\infty)$ see [9]). We can then add the following sentence (which is universally quantified on variable x, although we do not make it explicit):

$$\mathsf{C}x \geq \vec{b} \Rightarrow x \geq \alpha$$

to guarantee that $\mathscr{A}$ is well-founded; here α is a fresh *constant* whose value will be established by the constraint solving process. In our case, we have to guarantee that $\mathscr{A}_{\mathtt{S}}$ and $\mathscr{A}_{\mathtt{S1}}$ are both bounded from below. Thus, we impose the following constraints:

$$C_1^S x \geq b_1^S \wedge C_2^S x \geq b_2^S \quad \Rightarrow \quad x \geq \alpha \tag{28}$$

$$C_1^{S1} x \geq b_1^{S1} \wedge C_2^{S1} x \geq b_2^{S1} \quad \Rightarrow \quad x \geq \alpha' \tag{29}$$

for constants α and α', where x is universally quantified (but α and α' are treated as new, existentially quantified, parameters).

5.1.3 Compatibility with the subsort relation

Regarding the *subsort* relation, if $s \leq s'$, then $\mathscr{A}_s = D(\mathsf{C}^s, \vec{b}^s) \subseteq D(\mathsf{C}^{s'}, \vec{b}^{s'}) = \mathscr{A}_{s'}$ must hold. We can ensure this property *at the syntactic level* by just adding the following (universally quantified) implication:

$$x : s \Rightarrow x : s'$$

to the theory in Figure 2. At the *derived* level, such a formula becomes the universally quantified formula:

$$\mathsf{C}^s x \geq \vec{b}^s \Rightarrow \mathsf{C}^{s'} x \geq \vec{b}^{s'} \tag{30}$$

Since `S2` $\leq$ `S1`, we add the following sentence which is universally quantified in x:

$$C_1^{S2} x \geq b_1^{S2} \wedge C_2^{S2} x \geq b_2^{S2} \Rightarrow C_1^{S1} x \geq b_1^{S1} \wedge C_2^{S1} x \geq b_2^{S1}$$

However, since this sentence is *not* in affine form (due to the conjunction in the consequent of the implication), we decompose it as a conjunction of two implications as follows:

$$C_1^{S2} x \geq b_1^{S2} \wedge C_2^{S2} x \geq b_2^{S2} \Rightarrow C_1^{S1} x \geq b_1^{S1} \tag{31}$$

$$C_1^{S2} x \geq b_1^{S2} \wedge C_2^{S2} x \geq b_2^{S2} \Rightarrow C_2^{S1} x \geq b_2^{S1} \tag{32}$$

5.2 Functions

As a generalization of convex matrix interpretations in [11], a *many-sorted convex matrix intepretation* for $f : s_1 \cdots s_k \to s$ is a linear expression $F_1x_1 + \cdots + F_kx_k + F_0$ such that (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) it ranges on $D(\mathsf{C}^s, \vec{b}^s)$ whenever variables x_i take value on the corresponding domain $D(\mathsf{C}^{s_i}, \vec{b}^{s_i})$, i.e., that satisfies the following *algebraicity condition*:

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

Example 8 *Since $n_s = 1$ for all $s \in S$, components F_i for each symbol $f \in \Sigma$ are* numbers*, actually. We give* parametric interpretations *to each $f \in \Sigma$ as follows:*

$$\begin{array}{rclcrcl} [\mathtt{0}] & = & z_0 & \quad & [\mathtt{1}] & = & u_0 \\ [\mathtt{f}](x,y,z) & = & f_1x + f_2y + f_3z + f_0 & \quad & [\mathtt{g}](x,y) & = & g_1x + g_2y + g_0 \end{array}$$

and the algebraicity conditions *are (with x, y, z universally quantified in all formulas):*

$$C_1^{S2} z_0 \geq b_1^{S2} \wedge C_2^{S2} z_0 \geq b_2^{S2} \tag{33}$$

$$C_1^{S1} u_0 \geq b_1^{S1} \wedge C_2^{S1} u_0 \geq b_2^{S1} \tag{34}$$

$$\bigwedge_{i=1}^{2} C_i^{S1} x \geq b_i^{S1} \wedge \bigwedge_{i=1}^{2} C_i^{S1} y \geq b_i^{S1} \wedge \bigwedge_{i=1}^{2} C_i^{S1} z \geq b_i^{S1} \quad \Rightarrow \quad C_1^{S}(f_1x + f_2y + f_3z + f_0) \geq b_1^{S} \tag{35}$$

$$\bigwedge_{i=1}^{2} C_i^{S1} x \geq b_i^{S1} \wedge \bigwedge_{i=1}^{2} C_i^{S1} y \geq b_i^{S1} \wedge \bigwedge_{i=1}^{2} C_i^{S1} z \geq b_i^{S1} \quad \Rightarrow \quad C_2^{S}(f_1x + f_2y + f_3z + f_0) \geq b_2^{S} \tag{36}$$

$$\bigwedge_{i=1}^{2} C_i^{S1} x \geq b_i^{S1} \wedge \bigwedge_{i=1}^{2} C_i^{S1} y \geq b_i^{S1} \quad \Rightarrow \quad C_1^{S1}(g_1x + g_2y + g_0) \geq b_1^{S1} \tag{37}$$

$$\bigwedge_{i=1}^{2} C_i^{S1} x \geq b_i^{S1} \wedge \bigwedge_{i=1}^{2} C_i^{S1} y \geq b_i^{S1} \quad \Rightarrow \quad C_2^{S1}(g_1x + g_2y + g_0) \geq b_2^{S1} \tag{38}$$

where (35) and (36) are actually obtained from a single algebraicity condition after splitting the conjunction in the consequent of the implication to obtain implications in affine form (as in Section 5.1.3). Similarly for (37) and (38)). Note also that, even though `0` *and* `1` *are constant symbols, (33) and (34) are also necessary to guarantee that they receive a value according to their* sort *(*`S2` *and* `S1`*, respectively).*

We have no overloaded symbol $f \in \Sigma_{w_1,s_1} \cap \Sigma_{w_2,s_2}$. Thus, we do not care about guaranteeing that the interpretations $f^{\mathscr{A}}_{w_1,s_1}$ and $f^{\mathscr{A}}_{w_2,s_2}$ coincide on $\mathscr{A}_{w_1}$ (see Section 2). In general, though, if $f \in \Sigma_{w,s} \cap \Sigma_{w',s'}$ and $w \leq w'$, then we must have $s \leq s'$ as well. As discussed in Section 5.1, this implies that, with $w = s_1 \cdots s_k$ and $w' = s'_1 \cdots s'_k$, we must have $n_{s_i} = n_{s'_i}$ for all i, $1 \leq i \leq k$. Furthermore, $n_s = n_{s'}$ as well. Therefore, if $f^{\mathscr{A}}_{w,s} = \sum_{i=1}^{k} F_i \vec{x}_i + F_0$ and $f^{\mathscr{A}}_{w',s'} = \sum_{i=1}^{k} F'_i \vec{x}_i + F'_0$, the desired condition can be written as follows:

$$\forall x_1 \in \mathscr{A}_{s_1}, \ldots, \forall x_k \in \mathscr{A}_{s_n}, \sum_{i=1}^{k} (F_i - F'_i) x_i + F_0 - F'_0 = 0 \qquad \text{or, equivalently:}$$

$$\forall x_1 \in \mathbb{R}^{n_{s_1}} \ldots, \forall x_k \in \mathbb{R}^{n_{s_k}} (\bigwedge_{i=1}^{k} \mathsf{C}^{s_i} x_i \geq \vec{b}^{s_i} \Rightarrow \sum_{i=1}^{k} (F_i - F'_i) x_i + F_0 - F'_0 = 0)$$

5.3 Predicates

The interpretation of the (universally quantified) rules of the theory for the running example, with overloaded predicates $\rightarrow, \rightarrow^*$ (see Example 2), is given by interpreting the overloads of $\rightarrow^*$ as $\geq$ (the usual ordering on numbers) and the overloads of $\rightarrow$ as $>_\delta$ for some $\delta > 0$. The use of this special ordering over the reals instead of the usual one $>_\mathbb{R}$ is due to the need of interpreting $\rightarrow$ by using a *well-founded ordering* in order to obtain a sound termination analysis. Remember that $>_\delta$ is well-founded over subsets $A \subseteq \mathbb{R}$ that are *bounded from below*, (see Section 5.1.2).

Using the interpretations for sorts, function symbols, and predicates, we obtain the following derived sentences:

1. Instances of the *reflexivity* rule (Rf), corresponding to sentences (1) and (2) in Figure 2, with t universally quantified:

$$C_1^S t \geq b_1^S \wedge C_2^S t \geq b_2^S \Rightarrow t \geq t \tag{39}$$
$$C_1^{S1} t \geq b_1^{S1} \wedge C_2^{S1} t \geq b_2^{S1} \Rightarrow t \geq t \tag{40}$$

   Note that the two sentences above trivially hold *under the current interpretation of* $\geq$ *as a quasi-ordering* (a *reflexive* and *transitive* relation). Thus, (39) and (40) could be *removed*.

2. Instances of the *transitivity* rule (T), corresponding to (3) and (4):

$$\bigwedge_{i=1}^{2} C_i^S t \geq b_i^S \wedge \bigwedge_{i=1}^{2} C_i^S t' \geq b_i^S \wedge \bigwedge_{i=1}^{2} C_i^S u \geq b_i^S \wedge t \geq t' + \delta \wedge t' \geq u \Rightarrow t \geq u \tag{41}$$
$$\bigwedge_{i=1}^{2} C_i^{S1} t \geq b_i^{S1} \wedge \bigwedge_{i=1}^{2} C_i^{S1} t' \geq b_i^{S1} \wedge \bigwedge_{i=1}^{2} C_i^{S1} u \geq b_i^{S1} \wedge t \geq t' + \delta \wedge t' \geq u \Rightarrow t \geq u \tag{42}$$

3. Instances of the *congruence* rule (C), corresponding to (5)-(9), where we use $t \in \mathscr{A}_{\mathsf{S1}}$ instead of $C_1^{S1} t \geq b_1^{S1} \wedge C_2^{S1} t \geq b_2^{S1}$:

$$\bigwedge_{i=1}^{3} t_i \in \mathscr{A}_{\mathsf{S1}} \wedge t_1' \in \mathscr{A}_{\mathsf{S1}} \wedge t_1 \geq t_1' + \delta \Rightarrow f_1 t_1 + f_2 t_2 + f_3 t_3 + f_0 \geq f_1 t_1' + f_2 t_2 + f_3 t_3 + f_0 + \delta \tag{43}$$
$$\bigwedge_{i=1}^{3} t_i \in \mathscr{A}_{\mathsf{S1}} \wedge t_2' \in \mathscr{A}_{\mathsf{S1}} \wedge t_2 \geq t_2' + \delta \Rightarrow f_1 t_1 + f_2 t_2 + f_3 t_3 + f_0 \geq f_1 t_1 + f_2 t_2' + f_3 t_3 + f_0 + \delta \tag{44}$$
$$\bigwedge_{i=1}^{3} t_i \in \mathscr{A}_{\mathsf{S1}} \wedge t_3' \in \mathscr{A}_{\mathsf{S1}} \wedge t_3 \geq t_3' + \delta \Rightarrow f_1 t_1 + f_2 t_2 + f_3 t_3 + f_0 \geq f_1 t_1 + f_2 t_2 + f_3 t_3' + f_0 + \delta \tag{45}$$
$$\bigwedge_{i=1}^{2} t_i \in \mathscr{A}_{\mathsf{S1}} \wedge t_1' \in \mathscr{A}_{\mathsf{S1}} \wedge t_1 \geq t_1' + \delta \Rightarrow g_1 t_1 + g_2 t_2 + g_0 \geq g_1 t_1' + g_2 t_2 + g_0 + \delta \tag{46}$$
$$\bigwedge_{i=1}^{2} t_i \in \mathscr{A}_{\mathsf{S1}} \wedge t_2' \in \mathscr{A}_{\mathsf{S1}} \wedge t_2 \geq t_2' + \delta \Rightarrow g_1 t_1 + g_2 t_2 + g_0 \geq g_1 t_1 + g_2 t_2' + g_0 + \delta \tag{47}$$

4. Instances of the *replacement* rule (Re), corresponding to (10)-(11):

$$C_1^{S2} x \geq b_1^{S2} \wedge C_2^{S2} x \geq b_2^{S2} \Rightarrow f_1 z_0 + f_2 u_0 + f_3 x + f_0 \geq f_1 x + f_2 x + f_3 x + f_0 + \delta \tag{48}$$
$$C_1^{S1} x \geq b_1^{S1} \wedge C_2^{S1} x \geq b_2^{S1} \wedge C_1^{S1} y \geq b_1^{S1} \wedge C_2^{S1} y \geq b_2^{S1} \Rightarrow g_1 x + g_2 y + g_0 \geq x + \delta \tag{49}$$
$$C_1^{S1} x \geq b_1^{S1} \wedge C_2^{S1} x \geq b_2^{S1} \wedge C_1^{S1} y \geq b_1^{S1} \wedge C_2^{S1} y \geq b_2^{S1} \Rightarrow g_1 x + g_2 y + g_0 \geq y + \delta \tag{50}$$

5.4 Synthesis of the model

The conjunction of all previous sentences (26)-(50) (perhaps dropping some of them, as suggested in previous sections) yields an $\exists\forall$-sentence (the $\exists$ concerns existential quantification of k, k', α, α', δ, and all parameters in domain descriptions and algebraic interpretations) where all *introduced parameters* are existentially quantified (on appropriate domains of coefficients, see Section 4) and all *semantic variables* (i.e., those ultimately coming from the description of the problem and required by the semantic interpretation of symbols) are universally quantified (over the reals). As mentioned in Section 4, we can use now the techniques discussed in [11] together with standard constraint solving techniques to obtain an assignment of values to the parameters which defines the desired model. Given a matrix A of k rows and n columns, $\vec{b} \in \mathbb{R}^k$, $\vec{c} \in \mathbb{R}^n$ and $\beta \in \mathbb{R}$, the application of the affine form of Farkas' Lemma to prove that the universally quantified sentence $A\vec{x} \geq \vec{b} \Rightarrow \vec{c}^T\vec{x} \geq \beta$ holds tries to *find* a vector $\vec{\lambda}$ of k non-negative numbers $\vec{\lambda} \in \mathbb{R}_0^k$ such that the *constraints* $\vec{c} = A^T\vec{\lambda}$ and $\vec{\lambda}^T\vec{b} \geq \beta$ hold.

Example 9 *We apply the Affine form of Farkas' Lemma to sentence (28) as follows: the associated matrix A is actually a vector* $(C_1^S, C_2^S)^T$ *and* $\vec{b} = (b_1^S, b_2^S)^T$*; we have that* $\vec{c} = (1)^T$ *is a one-dimensional vector and finally* $\beta = \alpha$*. Then, we seek a vector* $\vec{\lambda} = (\lambda_1, \lambda_2)^T$ *with* $\lambda_1, \lambda_2 \geq 0$ *that satisfies the (in)equations:*

$$1 = C_1^S\lambda_1 + C_2^S\lambda_2 \qquad \lambda_1 b_1^S + \lambda_2 b_2^S \geq \alpha \qquad \lambda_1, \lambda_2 \geq 0$$

The satisfiability of these inequations (a constraint solving problem for parameters C_1^S, C_2^S, b_1^S, b_2^S, λ_1, λ_2 *and* α*), is equivalent to the satisfiability of (28).*

Example 10 *Sentence (50) is* not *in* affine *form, but we can easily fix it as follows:*

$$C_1^{S1}x \geq b_1^{S1} \wedge C_2^{S1}x \geq b_2^{S1} \wedge C_1^{S1}y \geq b_1^{S1} \wedge C_2^{S1}y \geq b_2^{S1} \Rightarrow g_1 x + (g_2 - 1)y \geq \delta - g_0 \tag{51}$$

Now, we apply Farkas' lemma to each of them. The associated matrix A has four rows (corresponding to the four atoms in the conjunction of the antecedent of the implication) and two columns (corresponding to variables x and y): $A = (C_1^{S1}, 0\ ;\ C_2^{S1}, 0\ ;\ 0, C_1^{S1}\ ;\ 0, C_2^{S1})$*. Vector* $\vec{b}$ *has four components:* $\vec{b} = (b_1^{S1}, b_2^{S1}, b_1^{S1}, b_2^{S1})^T$*. Now,* $\vec{c} = (g_1, g_2 - 1)^T$ *and* $\beta = \delta - g_0$*. Thus, we want now a vector* $\vec{\lambda} = (\lambda_1, \lambda_2, \lambda_3, \lambda_4)^T$ *that satisfies:*

$$\begin{array}{rr} g_1 = C_1^{S1}\lambda_1 + C_2^{S1}\lambda_2 & g_2 - 1 = C_1^{S1}\lambda_3 + C_2^{S1}\lambda_4 \\ \lambda_1 b_1^{S1} + \lambda_2 b_2^{S1} + \lambda_3 b_1^{S1} + \lambda_4 b_2^{S1} \geq \delta - g_0 & \lambda_1, \lambda_2, \lambda_3, \lambda_4 \geq 0 \end{array}$$

for some values of the parameters.

Remark 4 *Note that each implication processed using Farkas' Lemma can use a* different *vector* $\vec{\lambda}$*, but we have to* solve a single set of inequations corresponding to a single solution which produces a single model that makes all sentences valid.

The following assignment:

$$\begin{array}{lllllllll} C_1^S = 1 & C_2^S = 1 & C_1^{S1} = 1 & C_2^{S1} = 1 & C_1^{S2} = 1 & C_2^{S2} = -1 \\ b_1^S = 0 & b_2^S = 0 & b_1^{S1} = 0 & b_2^{S1} = 0 & b_1^{S2} = 0 & b_2^{S2} = 0 \\ f_1 = 1 & f_2 = 1 & f_3 = 1 & f_0 = 0 & g_1 = 1 & g_2 = 1 & g_0 = 0 & z_0 = 0 & u_0 = 1 \\ k = 0 & k' = 0 & \alpha = 0 & \alpha' = 0 & \delta = 1 \end{array}$$

(where we disregard the different $\vec{\lambda}$ required by the application of Farkas' Lemma as *administrative* symbols) makes all sentences *true* and *generates* the model $\mathscr{A}$ for the theory $\mathscr{S}$ in our running example:

$$\begin{array}{cccc}
\mathscr{A}_{\mathtt{S}} = [0,+\infty) & \mathscr{A}_{\mathtt{S1}} = [0,+\infty) & \mathscr{A}_{\mathtt{S2}} = \{0\} & \\
\mathtt{f}^{\mathscr{A}}_{\mathtt{S1\,S1\,S1,S}}(x,y,z) = x+y+z & \mathtt{g}^{\mathscr{A}}_{\mathtt{S1\,S1,S1}}(x,y) = x+y+1 & \mathtt{0}^{\mathscr{A}}_{\lambda,\mathtt{S2}} = 0 & \mathtt{1}^{\mathscr{A}}_{\lambda,\mathtt{S1}} = 1 \\
t \to^{\mathscr{A}}_{\mathtt{S\,S}} t' \Leftrightarrow t >_1 t' & t\,(\to^*)^{\mathscr{A}}_{\mathtt{S\,S}} t' \Leftrightarrow t \geq t' & t \to^{\mathscr{A}}_{\mathtt{S1\,S1}} t' \Leftrightarrow t >_1 t' & t\,(\to^*)^{\mathscr{A}}_{\mathtt{S1\,S1}} t' \Leftrightarrow t \geq t'
\end{array}$$

6 Related work and conclusions

Our extension of derived algebras [6] to derived models for order-sorted first-order theories follows some of the ideas in [5]. The generation of *homogeneous algebras* using parametric interpretations followed by a constraint solving process is standard in termination analysis of term rewriting [4]. However, no systematic treatment of the generation of *domains for sorts* and *heterogeneous* functions for *ranked symbols* in many-sorted or order-sorted algebras has been attempted to date. And the generation of predicates as part of the generation of a model is also new. This work is also a step forward in the practical use of logical models in proofs of operational termination of programs. This was a main motivation of [11] after understanding the practical role of using models in proofs of termination in the OT-Framework [12, 13]. This paper also generalizes our previous experience in termination to envisage a generic, logic-oriented approach to abstraction in program analysis, which is based on defining appropriate *models* for the logic which is used to describe the computations. Focusing on an order-sorted first-order logic to describe programs and program properties, we have generalized the *convex domains* and *convex matrix interpretations* introduced in [11] to the order-sorted setting. Such a generalization leads to a flexible framework to define different domains for different sorts whereas it is still amenable for automation by using existing algorithms and techniques from linear algebra [18] and algebraic geometry [15]. Indeed, the use of *bounded* convex domains for some sorts (as $\{0\}$ for sort `S2` in `ToyamaOS`) has been essential to obtain a simple solution of the corresponding problem. A first implementation of the techniques presented in this paper has been reported in [17], including the generation of convex domains and convex interpretations along the lines of Section 5. The use of convex domains in termination analysis is also available as part of the tool MU-TERM [1]. Their usefulness has been recently shown in the 2015 International Termination Competition held in August as part of CADE 2015, where convex domains have been successfully used to prove operational termination of *conditional term rewriting systems*.

Acknowledgments. I thank the anonymous referees for their comments and suggestions. Special thanks go to Marisa Navarro for her kindness during the preparation of the final version of this 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, 2011.

[2] G.S. Boolos, J.P. Burgess, and R.C. Jeffrey. Computability and Logic, fourth edition. Cambridge University Press, 2002.

[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. Lecture Notes in Computer Science 4350, 2007.

[4] E. Contejean, C. Marché, A.-P. Tomás, and X. Urbain. Mechanically proving termination using polynomial interpretations. *J. of Aut. Reas.*, 34(4):325-363, 2006.

[5] J. Goguen and J. Meseguer. Models and Equality for Logical Programming. In *Proc. of TAPSOFT'87*, LNCS 250:1-22, Springer-Verlag, 1987.

[6] J.A. Goguen, J.W. Thatcher, and E.G. Wagner. An initial algebra approach to the specification, correctness and implementation of abstract data types. In *Current trends in Programming Methodology*, pages 80-149, Prentice Hall, 1978.

[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] W. Hodges. A shorter model theory. Cambridge University Press, 1997.

[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, C. Marché, and J. Meseguer. Operational termination of conditional term rewriting systems. *Information Processing Letters*, 95:446–453, 2005.

[11] S. Lucas and J. Meseguer. Models for Logics and Conditional Constraints in Automated Proofs of Termination. In Proc. of *AISC'14*, LNAI 8884:7-18, 2014.

[12] S. Lucas and J. Meseguer. Proving Operational Termination Of Declarative Programs In General Logics. In *Proc. of PPDP'14*, pages 111-122, 2014.

[13] S. Lucas and J. Meseguer. Localized Operational Termination In General Logics. In *Software, Services and Systems*, LNCS 8950:91-114, 2015.

[14] J. Meseguer. General Logics. In H.-D. Ebbinghaus et al., editors, *Logic Colloquium'87*, pages 275-329, North-Holland, 1989.

[15] A. Prestel and C.N. Delzell. Positive Polynomials. From Hilbert's 17th Problem to Real Algebra. Springer-Verlag, Berlin, 2001.

[16] M.O. Rabin. Decidable Theories. In J. Barwise, editor, Handbook of Mathematical Logic, North-Holland, 1977.

[17] E.P. Reinoso. Logical Models For Automated Semantics-Directed Program Analysis. Master Thesis, DSIC, Master in Software Systems Engineering and Technology (MITSS), Universitat Politècnica de València, July 2015.

[18] A. Schrijver. Theory of linear and integer programming. John Wiley & sons, 1986.

[19] Y. Toyama. Counterexamples to termination for the direct sum of term rewriting systems. *Information Processing Letters* 25:141-143, 1987.

[20] H. Zantema. Termination of term rewriting: interpretation and type elimination. *Journal of Symbolic Computation* 17:23-50, 1994.