Synthesis of models for order-sorted first-order theories using linear algebra and constraint solving*

Salvador Lucas

DSIC, Universitat Politècnica de València, Spain

http://users.dsic.upv.es/~slucas/

Recent developments in termination analysis for declarative programs emphasize the use of appropriate models for the logical theory representing the program at stake as a generic approach to prove termination of declarative programs. In this setting, Order-Sorted First-Order Logic provides a powerful framework to represent declarative programs. It also provides a target logic to obtain models for other logics via transformations. We investigate the automatic generation of numerical models for order-sorted first-order logics and its use in program analysis, in particular in termination analysis of declarative programs. We use convex domains to give domains to the different sorts of an order-sorted signature; we 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 and arithmetic constraint solving to synthesize the models.

Keywords: Linear algebra, Logical models, Order-sorted first-order logic, Program Termination.

1 Introduction

In the logical approach to programming, (declarative) programs are theories of a given logic $\mathcal{L}$ and computation is deduction in the inference system of $\mathcal{L}$ [14]. The corresponding notion of termination of declarative programs is the absence of infinite trees in any proof of a computation [10]. Recently, a framework to prove termination of declarative programs has been developed [12]. In this framework, we obtain the proof jumps associated to the inference system $\mathcal{I}(\mathcal{S})$ which is derived from the logic $\mathcal{S}$ which is used to describe the program $\mathcal{S}$. Proof jumps are structures $A \uparrow B_1, \ldots, B_n$ where $n > 0$ and $A, B_1, \ldots, B_n$ are formulas in the inference rules $(B_1 \cdots B_n \leftarrow A)$ in $\mathcal{I}(\mathcal{S})$ (for $p \geq 0$). Proof jumps are used to capture (infinite) paths in a proof tree $T$ using the rules in $\mathcal{I}(\mathcal{S})$ 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 instances of $B_1, \ldots, B_{n-1}$ were proved, i.e., $\mathcal{S} \vdash \sigma(B_i)$ for all $i, 1 \leq i < n$. A set of proof jumps $\tau$ is an OT problem. The initial OT problem $\tau_0$ consists of all proof jumps for $\mathcal{I}(\mathcal{S})$. Then, we apply an incremental proof methodology which simplifies OT problems $\tau$ in a divide-and-conquer style to eventually prove (or disprove) termination of $\mathcal{S}$. In particular, proof jumps $\psi : A \uparrow B_1, \ldots, B_n$ can be removed from an OT problem $\tau$ by using well-founded relations $\sqsubseteq$ as follows: if, for all substitutions $\sigma$, whenever $\mathcal{S} \vdash \sigma(B_i)$ holds for all $i, 1 \leq i < n$, we have that $\sigma(A) \sqsubseteq \sigma(B_n)$ holds, then we can remove $\psi$ from $\tau$. In [12] we show that logical models are useful for this purpose. Any model $\mathcal{A}$ of $\mathcal{S}$ satisfies the provable formulas, i.e., if $\mathcal{S} \vdash \sigma(B_i)$ holds, then, $\mathcal{A} \models \sigma(B_i)$ holds. The point is using this fact to define the well-founded relation $\sqsubseteq$. This idea is developed in [11] for a systematic treatment of proofs of termination using logical models.

A sufficiently general and expressive framework to represent declarative programs, semantics of programming languages, and program properties is Order-Sorted First-Order Logic (OS-FOL), where the signature consists of a set $S$ of sorts (i.e., names representing sets of values) which are ordered by

---

*Partially supported by the EU (FEDER), Spanish MINECO TIN 2013-45732-C4-1-P and GV PROMETEOII/2015/013.
a **subsort relation** \( \leq \) meaning subset inclusion, and sets \( \Sigma_{w,\ell} \) and \( \Pi_{w} \) of function and predicate symbols, where \( s \in S \) and \( w \) is a sequence \( s_1 \cdots s_k \) of sorts from \( S \) [6]. For instance, in our **running example** (Section 1.1) we develop a termination analysis for an **Order-Sorted Term Rewriting System** (OS-TRS) [6, 7], viewed as a particular case of OS-FOL theory with predicate symbols \( \to \) and \( \to^* \) describing the one-step rewrite relation \( \to \) and the zero-or-more-steps relation \( \to^* \), see Figure 1.

In this paper we consider the automatic generation of models for OS-FOL theories. This can be used in program analysis, in particular to mechanize the termination analysis of declarative programs as explained above. **Semantic structures** \( \mathcal{A} \) [8] leading to **decidable theories** \( \text{Th}(\mathcal{A}) \) [15] can be used to provide an effective way to **find** logical models \( \mathcal{A} \) for a program or specification \( \mathcal{R} \). This is often possible by using **theory transformations** \( \kappa \) from the language of \( \mathcal{R} \) into the language of \( \text{Th}(\mathcal{A}) \) to obtain a set of sentences \( \mathcal{A} = \kappa(\mathcal{R}) \) which is then **decidable**. We formalize this view by extending the notion of **derived algebra** [5] to logical structures. Targeted languages usually involve symbols (e.g., \( + \), \( \times \),...) with an **intended meaning** in the structures \( \mathcal{A} \) that define the decidable theory \( \text{Th}(\mathcal{A}) \). We also show how to transform an OS-FOL theory \( \mathcal{R} \) into a derived **parametric** theory \( \mathcal{R}' \) of linear arithmetic where appropriate algorithms and constraint solving techniques can be used to give value to the parameters thus **synthesizing** a model of \( \mathcal{R} \). The **convex domains** introduced in [11] provide appropriate means for this. They can be used to define bounded and unbounded domains for the sorts in the OS signature. Indeed, the use of different (in particular bounded) domains for some sorts is essential to obtain a simple model which can be used to prove termination of our running example.

Section 2 summarizes the basics of OS-FOL. Section 3 develops the notion of **derived model** and shows how to use it to deal with our running examples. Section 4 describes our automation approach using linear algebra techniques and constraint solving. Section 5 explains the generation of OS-FOL structures based on the **convex domains** and **convex matrix interpretations** introduced in [11]. Section 6 shows how to apply the technique to obtain an **automatic** solution to our case study. Section 7 concludes.

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

The OS-TRS Toyama0S in Figure 2 is based on Toyama's example [18]. It is given as a (hopefully self-explanatory) module of Maude [3]. The unsorted version of this module is nonterminating [18]. Furthermore, if \( S_2 \) and \( S_3 \) are confused into a single sort then Toyama0S is nonterminating too:

\[
\begin{align*}
&f(g(0,1), g(0,1), g(0,1)) \to f(0, g(0,1), g(0,1)) \to f(0, 1, g(0,1)) \to f(0, 0, g(0,1)) \to f(g(0,1), g(0,1), g(0,1)) \to \cdots
\end{align*}
\]

But with all sort information we can prove it **terminating**. For instance, variable \( x \) (of sort \( S_2 \)) cannot be bound to terms of sort \( S_1 \) which is a supersort of \( S_2 \). Thus, the third step, which requires a binding \( x \mapsto g(0,1) \), is not possible because the sort of \( g(0,1) \) is \( S_1 \). 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

\[
\begin{align*}
\text{(Rf)} \quad & t \rightarrow^* t \\
\text{(T)} \quad & t \rightarrow t', t' \rightarrow^* u \\
\text{(C)} \quad & t_i \rightarrow t_i' \\
\text{(Re)} \quad & \ell \rightarrow r
\end{align*}
\]

\[ R = \{ f(t_1, \ldots, t_i, \ldots, t_k) \rightarrow f(t_1, \ldots, t_i', \ldots, t_k) \} \text{ where } f \in \Sigma_{w,\ell}, w = s_1 \cdots s_k, \text{ and } 1 \leq i \leq k \]

\[ \ell \rightarrow r \in \mathcal{R} \]
of the inference rules by taking into account the sorts in the signature and the subset ordering. In particular, the only quantification over $S_2$ 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 $\mathcal{M}$ for the theory in Figure 2 such that $\rightarrow$ is interpreted as a well-founded relation $\ast$. 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 $f \in \Sigma_{w_{1},\ldots,w_{n},s}$, then we display $f$ as $f : s_{1} \cdots s_{n} \rightarrow s$. This is called a rank declaration for symbol $f$. Constant symbols $c$ (taking no argument) have rank declaration $c : \lambda \rightarrow s$ for some sort $s$ (where $\lambda$ 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 $(S, \Sigma)$. 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 $\Sigma$ is sensible, meaning that if $f : s_{1} \cdots s_{n} \rightarrow s$ and $f' : s_{1}' \cdots s_{n}' \rightarrow s'$ are such that $[s_{i}] = [s_{i}']$, $1 \leq i \leq n$, then $[s] = [s']$. Throughout this paper, $\Sigma$ will always be assumed sensible. An order-sorted signature $\Sigma$ 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 $w_{0} < w$. 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 $\mathcal{X} = \{\mathcal{X}_{s} | s \in S\}$ of mutually disjoint sets of variables (which are also disjoint from the signature $\Sigma$), the set $\mathcal{R}(\mathcal{X})_{s}$ of terms of sort $s$ is the least set such that (i) $\mathcal{X}_{s} \subseteq \mathcal{R}(\mathcal{X})_{s}$, (ii) If $s' \leq s$, then $\mathcal{R}(\mathcal{X})_{s'} \subseteq \mathcal{R}(\mathcal{X})_{s}$; and (iii) for each $f : s_{1} \cdots s_{n} \rightarrow s$ and $t_{i} \in \mathcal{R}(\mathcal{X})_{s_{i}}$, $1 \leq i \leq n$, $f(t_{1}, \ldots, t_{n}) \in \mathcal{R}(\mathcal{X})_{s}$. If $\mathcal{X} = \emptyset$, we write $\mathcal{R}$ rather than $\mathcal{R}(\mathcal{X})$ 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 appropriate sort, i.e., \( \Sigma(\mathcal{X})_\lambda, s = \Sigma_\lambda, s \cup \mathcal{X}_s \).

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

1. Set of sorts \( S = \{S, S_1, S_2\} \).
2. The subset relation is the least ordering \( \leq \) on \( S \) satisfying \( S_2 \leq S_1 \).
3. Thus, \((S, \leq)\) (or \( S/_{\equiv_\Sigma} \)) consists of two connected components: \( [S] = \{S\} \) and \([S_1] = \{S_2, S_1\} \).
4. Note that \( S \) is the top sort \( T[S] \) of \([S]\), and \( S_1 \) is the top sort \( T[S_1] \) of \([S_1]\).
5. The signature is \( \Sigma = \Sigma_{S_1} \cup \Sigma_{S_2} \cup \Sigma_{S_1S_1S_1} \cup \Sigma_{S_1S_1S_2} \), with \( \Sigma_{S_1} = \{1\} \), \( \Sigma_{S_2} = \{0\} \), \( \Sigma_{S_1S_1} = \{g\} \), and \( \Sigma_{S_1S_1S_2} = \{f\} \).
6. There is no overloaded function symbol, i.e., \( \Sigma \) is trivially regular. Furthermore, since every connected component has a top sort (see item 2), \((S, \leq, \Sigma)\) is a coherent signature.

The set of variables is \( \mathcal{X} = \mathcal{X}_{S_1} \cup \mathcal{X}_{S_2} \), with \( \mathcal{X}_{S_1} = \{y, z\} \), and \( \mathcal{X}_{S_2} = \{x\} \).

The assumption that \( \Sigma \) is sensible ensures that if \( [s] \neq [s'] \), then \( \mathcal{R}_2(\mathcal{X})_{[s]} \cap \mathcal{R}_2(\mathcal{X})_{[s']} \equiv \emptyset \). The set \( \mathcal{R}_2(\mathcal{X}) \) of order-sorted terms is \( \mathcal{R}_2(\mathcal{X}) = \bigcup_{s \in S} \mathcal{R}_2(\mathcal{X})_s \). An element of any set \( \mathcal{R}_2(\mathcal{X})_s \) is called a well-formed term.

**Order-Sorted Algebras.** Given a many-sorted signature \((S, \Sigma)\), an \((S, \Sigma)\)-algebra \( \mathcal{A} \) (or just a \( \Sigma \)-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 \( f_{w, s}^{\mathcal{A}} : \mathcal{A}_w \to \mathcal{A}_s \) for each \( f \in \Sigma_{w, s} \) where \( \mathcal{A}_w = \mathcal{A}_{w_1} \times \cdots \times \mathcal{A}_{w_n} \) if \( w = s_1 \cdots s_n \), and \( \mathcal{A}_{w_0} \) is a one point set when \( w = \lambda \). Given an order-sorted signature \((S, \leq, \Sigma)\), an \((S, \leq, \Sigma)\)-algebra (or \( \Sigma \)-algebra if \((S, \leq)\) is clear from the context) is an \((S, \Sigma)\)-algebra such that

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

**Remark 1** Note that overloaded symbols \( f \) may be given different functions \( f_{w_1, s_1}^{\mathcal{A}}, \ldots, f_{w_n, s_n}^{\mathcal{A}} \) 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 \((S, \Sigma)\)-homomorphism between \((S, \Sigma)\)-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 \( w = s_1, \ldots, s_k \),

\[
\begin{align*}
    h_s(f_{w, s}^{\mathcal{A}}(a_1, \ldots, a_k) &= h_{s_1}(a_1), \ldots, h_{s_k}(a_k)) \quad \text{if } w = \lambda, \\
    h_s(f_{w, s}^{\mathcal{A}}(a_1, \ldots, a_k) &= f_{w, s}^{\mathcal{A}'}(h_{s_1}(a_1), \ldots, h_{s_k}(a_k)) \quad \text{if } w \neq \lambda.
\end{align*}
\]

If \( w = \lambda \), we have \( h_s(f_{w}^{\mathcal{A}}) = f_{w}^{\mathcal{A}'} \). 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 \((S, \Sigma)\)-homomorphism that satisfies the following additional condition: if \( s \leq s' \) and \( a \in \mathcal{A}_s \), then \( h_s(a) = h_{s'}(a) \).

The family of domains \( \{\mathcal{R}_2(\mathcal{X})_s \mid s \in S\} \) together with the operations \( f : (t_1, \ldots, t_n) \mapsto f(t_1, \ldots, t_n) \) define an order-sorted \( \Sigma \)-algebra called the free \( \Sigma \)-algebra on \( \mathcal{X} \) and denoted \( \mathcal{R}_2(\mathcal{X}) \). When \( \mathcal{X} = \emptyset \), \( \mathcal{R}_2 = \mathcal{R}_2(\emptyset) \) denotes the initial \( \Sigma \)-algebra, i.e., an algebra having a unique homomorphism \( h_{\mathcal{X}} : \mathcal{R}_2 \to \mathcal{A} \) to each \( \Sigma \)-algebra \( \mathcal{A} \). Similarly, \( \mathcal{R}_2(\mathcal{X}) \) (itself a \( \Sigma \)-algebra) is initial in the class of all \( \Sigma(\mathcal{X}) \)-algebras.
Predicates and connectives. Following [6], an order-sorted signature with predicates is a quadruple \((S, \leq, \Sigma, \Pi)\) such that \((S, \leq, \Sigma)\) is a 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 [6, Definition 11]:

1. There is an equality predicate symbol \(= \in \Pi_s\) 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 \(\Sigma, \Pi\) instead of \((S, \leq, \Sigma, \Pi)\) if \(S\) and \(\leq\) are clear from the context. The formulas \(\phi\) of an order-sorted signature with predicates \(\Sigma, \Pi\) are built up from atoms \(P(t_1, \ldots, t_n)\) with \(P \in \Pi_w\) and \(t_1, \ldots, t_n \in \mathcal{T}_S(X)_w\), logic connectives (e.g., \(\land, \neg\)) and quantifiers (\(\forall\)) as follows: (i) if \(P \in \Pi_w\), \(w = s_1 \cdot \cdot \cdot s_n\), and \(t_i \in \mathcal{T}_S(X)_{s_i}\) for all \(i, 1 \leq i \leq n\), then \(P(t_1, \ldots, t_n) \in \text{Form}_{\Sigma, \Pi}\). (ii) if \(\phi \in \text{Form}_{\Sigma, \Pi}\), then \(\neg \phi \in \text{Form}_{\Sigma, \Pi}\); (iii) if \(\phi, \phi' \in \text{Form}_{\Sigma, \Pi}\), then \(\phi \land \phi' \in \text{Form}_{\Sigma, \Pi}\); (iv) if \(s \in S\), \(x \in \mathcal{T}_S\), and \(\phi \in \text{Form}_{\Sigma, \Pi}\), then \((\forall x : s)\phi \in \text{Form}_{\Sigma, \Pi}\). As usual, we can consider formulas involving other logic connectives and quantifiers (e.g., \(\forall, \Rightarrow, \Leftrightarrow, \exists, \ldots\)) by using their standard definitions in terms of \(\land, \neg, \forall\). 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 \(\rightarrow^*\) as connected component \([s]\) in \(S/\equiv_c\): due to axiom (Rf), OS-TRSs are expected to rewrite with \(\rightarrow^*\) any of the classes \(\mathcal{T}_S(X)_{[s]}\) for every connected component \([s]\). By coherence of the signature, we can just let \(\rightarrow^* \in \Pi_{\top_{[s]}}\Pi_{[s]}\) for all \(s \in S\). Then, rule (T) requires a corresponding overload for \(\rightarrow\) as well. By coherence of the signature, we can just let \(\Pi_{\top_{[s]}}\Pi_{[s]}\) for all \(s \in S\). This will be compatible with any possible instance of rule (Re) because terms \(\ell\) and \(r\) in rewrite rules \(\ell \rightarrow r\) of OS-TRSs must be terms belonging to \(\mathcal{T}_S(X)_{[s]}\) for some \(s \in S\). By coherence, we know that \(\ell, r \in \mathcal{T}_S(X)_{\top_{[s]}}\) for some \(s \in S\).

Example 2 The order-sorted signature \((S, \leq, \Sigma)\) described in Example [4] is extended into an order-sorted signature with predicates \((S, \leq, \Sigma, \Pi)\) where \(\Pi = \Pi_{SS} \cup \Pi_{S1S1}\) for \(\Pi_{SS} = \Pi_{S1S1} = \{\rightarrow, \rightarrow^*\}\), which are the only nonempty sets of predicate symbols. They satisfy the regularity condition.

Theories, specifications and programs. A theory \(\mathcal{J}\) of \(\Sigma, \Pi\) is a set of formulas, \(\mathcal{J} \subseteq \text{Form}_{\Sigma, \Pi}\), and its theorems are the formulas \(\phi \in \text{Form}_{\Sigma, \Pi}\) for which we can derive a proof using an appropriate inference system \(\mathcal{J}(\mathcal{L})\) of a logic \(\mathcal{L}\) in the usual way (written \(\mathcal{J} \vdash \phi\)). Given a logic \(\mathcal{L}\) describing computations in a (declarative) programming language, programs are viewed as theories \(\mathcal{J}\) of \(\mathcal{L}\).

Example 3 In the logic of OS-TRSs, with binary (overloaded) predicates \(\rightarrow\) and \(\rightarrow^*\), the theory for an OS-TRS \(\mathcal{R} = (S, \leq, \Sigma, R)\) with set of rules \(R\) (for instance, our running example) is obtained from the schematic inference rules in Figure 7 after specializing them as \((C)_{f,i}\) for each \(f \in \mathcal{F}\) and \(i, 1 \leq i \leq \text{ar}(f)\) and \((Re)_f\) for all \(\rho : \ell \rightarrow r \in R\). Then, inference rules \(\frac{B_1 \cdots B_n \Rightarrow A}{\mathcal{R}}\) become implications \(B_1 \land \cdots \land B_n \Rightarrow A\). For instance, with regard to the sentences for ToyamaOS in Figure 2:

- Sentences [1] and [2] specialize (Rf) in Figure 7 for the two overloads of \(\rightarrow^*\) in \(\Pi_{SS}\) and \(\Pi_{S1S1}\), respectively.
- Sentences [3] and [4] specialize (T) for the overloads of \(\rightarrow^*\) and \(\rightarrow\) in \(\Pi_{SS}\) and \(\Pi_{S1S1}\), respectively.
- Sentences [5], [6], and [7] specialize (C) for symbol \(\ell\) using the appropriate overloads of \(\rightarrow\) in \(\Pi_{SS}\) and \(\Pi_{S1S1}\) according to the rank of \(\ell\). 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 \( S_2 \) only.

**Structures, Satisfaction, Models.** Given an order-sorted signature with predicates \((S, \leq, \Sigma, \Pi)\), an \((S, \leq, \Sigma, \Pi)\)-structure \((\mathcal{A}, \mathcal{I})\) (or just a \((\Sigma, \Pi)\)-structure) is an order-sorted \((S, \leq, \Sigma)\)-algebra \( \mathcal{A} \) together with an assignment to each \( P \in \Pi_w \) of a subset \( P_w^{\mathcal{A}} \subseteq \mathcal{A}_w \) such that \[ \text{if } \mathcal{A} \text{ is an } \mathcal{A} \text{-structure for each } P : w \text{ in } \Pi, \text{ then } h(a_1, \ldots, a_n) \in P_w^{\mathcal{A}} \text{ if and only if } P(a_1, \ldots, a_n) \in P_w^{\mathcal{A}}. \]

Let \((S, \leq, \Sigma, \Pi)\) be an order-sorted signature with predicates \( \mathcal{A}, \mathcal{A}' \) 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 \( \Pi \), if \( h(a_1, \ldots, a_n) \in P_w^{\mathcal{A}} \), then \( h(a_1, \ldots, a_n) \in P_w^{\mathcal{A}'} \).

Given an \( S \)-sorted valuation mapping \( \alpha : \mathcal{X} \to \mathcal{A} \), the evaluation mapping \( [\cdot]^{\alpha}_{\mathcal{A}} : \mathcal{X}(\mathcal{X}) \to \mathcal{A} \) is the unique \((S, \leq, \Sigma)\)-homomorphism extending \( \alpha \).

Finally, \([\cdot]^{\alpha}_{\mathcal{A}} : \mathcal{X}(\mathcal{X}) \to \mathcal{A} \) is given by:

1. \([P(t_1, \ldots, t_k)]^{\alpha}_{\mathcal{A}} = \text{true} \) for \( P : w \) and terms \( t_1, \ldots, t_k \) if and only if \( ([t_1]^{\alpha}_{\mathcal{A}}, \ldots, [t_k]^{\alpha}_{\mathcal{A}}) \in P_w^{\mathcal{A}} \);
2. \([- \psi]^{\alpha}_{\mathcal{A}} = \text{true} \) if and only if \( [\psi]^{\alpha}_{\mathcal{A}} = \text{false} \);
3. \([\psi \land \psi']^{\alpha}_{\mathcal{A}} = \text{true} \) if and only if \( [\psi]^{\alpha}_{\mathcal{A}} = \text{true} \) and \( [\psi']^{\alpha}_{\mathcal{A}} = \text{true} \);
4. \([\forall x \cdot s]^{\alpha}_{\mathcal{A}} = \text{true} \) if and only if \( [\alpha]^{\alpha}_{\mathcal{A}} = \text{true} \).

We say that \( \mathcal{A} \) satisfies \( \phi \in \mathcal{X}(\mathcal{X}) \) if there is \( \alpha \in \mathcal{X} \to \mathcal{A} \) such that \( [\phi]^{\alpha}_{\mathcal{A}} = \text{true} \). If \( [\phi]^{\alpha}_{\mathcal{A}} = \text{true} \) for all valuations \( \alpha \), we write \( \mathcal{A} \models \phi \) and say that \( \mathcal{A} \) is a model of \( \phi \).

We say that \( \mathcal{A} \) is a model of a set of sentences \( \mathcal{S} \subseteq \mathcal{X}(\mathcal{X}) \) (written \( \mathcal{A} \models \mathcal{S} \)) if for all \( \phi \in \mathcal{S}, \mathcal{A} \models \phi \).

We say that \( \mathcal{A} \) is a derived \( \Sigma, \Pi \)-structure \( \mathcal{A}' \) so that \( \mathcal{A} \models \mathcal{S} \). Sound logics guarantee that every provable sentence \( \phi \) is true in every model of \( \mathcal{A} \), i.e., \( \mathcal{A} \models \phi \) implies \( \mathcal{A} \models \phi \).

### 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 \( \phi \) belongs to \( T \). In some cases such theories can be presented as axiomatizations of algebraic structures \( \mathcal{A} \) so that \( T = Th(\mathcal{A}) = \{ \phi \mid \mathcal{A} \models \phi \} \).

**Example 4** Presburger’s arithmetic (or arithmetic without multiplication) can be seen as the set of sentences of the language \( L_P = \{ 0, 1, +, > \} \) which are true in the standard interpretation \( \mathcal{N} \) of the natural numbers [2] page 295]. It is well-known that \( P = Th(\mathcal{N}) \) is decidable.

Assume that \((S', \leq', \Sigma', \Pi')\) is an order-sorted signature with predicates and \( \mathcal{A}' \) is a \( \Sigma', \Pi' \)-structure such that \( T = Th(\mathcal{A}') \) is decidable. We can define an \((S, \leq, \Sigma, \Pi)\)-model for \( \mathcal{S} \subseteq \mathcal{X}(\mathcal{X}) \) by means of a map (theory transformation) \( \kappa : \mathcal{S}(\mathcal{X}) \to \mathcal{X}(\mathcal{X}) \).

If \( \mathcal{S} \) is finite, then, it is decidable whether \( \kappa(\mathcal{S}) \subseteq T \).

If \( \kappa(\mathcal{S}) \subseteq T \), then \( \mathcal{A}' \models \kappa(\mathcal{S}) \), i.e., the \( \Sigma', \Pi' \)-structure \( \mathcal{A}' \) is a model of \( \kappa(\mathcal{S}) \).

If we can define \( \kappa \) on a purely syntactic basis, i.e., as homomorphic extensions of maps from the syntactic components \( S, \Sigma, \) and \( \Pi \) in \((S, \leq, \Sigma, \Pi)\), then we are able to make \( \mathcal{A}' \) into a derived \( \Sigma, \Pi \)-structure \( \mathcal{A}' \) so that \( \mathcal{A}' \) is a model of \( \mathcal{S} \), i.e., \( \mathcal{A}' \models \mathcal{S} \), 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)** [5] Definition 11] Let Σ = (S, ≤, Σ) and Σ′ = (S, ≤′, Σ′) be order-sorted signatures. A derivor from Σ to Σ′ is a monotone function τ : S → S’ (i.e., such that for all s, s’ ∈ S, s ≤ s’ implies τ(s) ≤ τ(s’)) and a family d_{w,s} : Σ_{w,s} → (FΣ)_{τ(w),τ(s)}, where τ(s1, ..., sk) = τ(s1), ..., τ(sk) and where (FΣ)_{τ(w),τ(s)} denotes the set of all Σ′-terms using variables {y1, ..., yk} with y_i of sort τ(s_i). Each operation symbol f ∈ Σ_{w,s} is expressed using a derived operation d_{w,s}(f) of the appropriate arity. We often use d to denote a derivor (τ, d). Now, let ω′ be an Σ′-algebra. Then, the d-derived algebra dω′ of ω′ is the Σ-algebra with carriers (dω′)_{s} = ω′_{τ(s)} for all s ∈ S; and mappings f^{dω′} for each f ∈ Σ defined to be (d(f))^{dω′}, the derived operator of the Σ′-term d(f).

Note that d in Definition 1 is homomorphically extended into a mapping d : FΣ(X) → FΣ′(X′).

**Example 5** Let (S, ≤, Σ) as in Example 1. Let Σ = (zero, nat) with subset relation ≤ given by zero ≤ nat, and Σ′ = Σ′_{zero} ⊔ Σ′_{one} ⊔ Σ′_{nat} where Σ′_{zero} = {0}, Σ′_{one} = {1}, and Σ′_{nat} = {+}. We define a derivor from (S, ≤, Σ) to (S’, ≤′, Σ′) by τ(S) = nat and τ(S1) = zero; also, d(0) = 0, d(1) = 1, d(f) = x + y + z, and d(g) = x + y + 1. Let ω′ be the (S’, ≤′, Σ′) algebra given by ω′_{zero} = {0} and ω′_{nat} = N together with the standard interpretations for 0, 1, and +. The derived (S, ≤, Σ)-algebra ω = dω′ is given by ω_{z0} = ω′_{zero} = {0} and ω_{z1} = ω′_{1} = ω′_{nat} = 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 Σ = (S, ≤, Σ, Π) and Σ′ = (S, ≤′, Σ′, Π′) be order-sorted signatures with predicates and ⟨τ, d⟩ be a derivor from (S, ≤, Σ) to (S′, ≤′, Σ′). We extend d to predicate symbols by adding a component d : Π → FormΣΠ′ such that for all P ∈ Π_w, with w = s1 · · · s_n, d(P) is an atom P′(t′_1, ..., t′_m) with P′ ∈ Π’_w, and terms t′_1, ..., t′_m ∈ FΣ(X) only use variables {y1, ..., y_n} with y_i of sort τ(s_i). In this new context we also call ⟨τ, d⟩ a derivor. Let ω′ = (ω′, Σ′ω′, Π′_ω′) be an (S, ≤, Σ, Π)-structure and ω′ = (ω′, Σ′ω′) be the underlying (S′, ≤′, Σ′)-algebra. Then, the ⟨τ, d⟩-derived structure dω′ of ω′ is the (S, ≤, Σ, Π)-structure that consists of the Σ-algebra dω′_w with S-sorted set of carriers S′ together with interpretations P_w^{dω′} (for P ∈ Π_w) defined to be

\[ P_w^{dω′} = \{(t_1, ..., t_n) | \langle t_1, ..., t_n \rangle \in FΣ(X) \}, \alpha \in X \rightarrow S′, \]

\[ d(P) = P'(t'_1, ..., t'_m), \varphi = \varphi[t'_1, ..., t'_m], \sigma[y_i] = t_i, 1 \leq i \leq n \]

\[ \exists \alpha' : Y \rightarrow \omega'[[\sigma[t'_1]]^{\alpha'}, ..., [\sigma[t'_m]]^{\alpha'}] \in (P')^{\omega'} \]

Note that ⟨τ, d⟩ can be seen now as a transformation d : FormΣΠ → FormΣΠ':

\[ d(P[t_1, ..., t_n]) = d(P)[y_1 \mapsto d(t_1), ..., y_n \mapsto d(t_n)] \]

\[ d(\neg \varphi) = \neg d(\varphi) \]

\[ d(\varphi \land \varphi') = d(\varphi) \land d(\varphi') \]

\[ d((\forall x : s) \varphi) = (\forall x : \tau(s))d(\varphi) \]

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

---

2 Monotonicity is not required in [5] where only many-sorted signatures are considered.
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 \( \mathcal{A} \) be an \( (S', \leq', \Sigma', \Pi') \)-structure and \( \varphi \in \text{Form}_{\Sigma, \Pi} \). If \( \mathcal{A}' \models d(\varphi) \), then \( d\mathcal{A}' \models \varphi \).

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 \( \mathcal{A}' \) be an \( (S', \leq', \Sigma', \Pi') \)-structure and \( \mathcal{I} \subseteq \text{Form}_{\Sigma, \Pi} \) be a theory. If for all \( \varphi \in \mathcal{I} \), \( \mathcal{A}' \models d(\varphi) \), then \( d\mathcal{A}' \models \mathcal{I} \).

The following example shows how to use Corollary 1 together with an appropriate derived model for proving termination of the OS-TRS Toyama0S 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 \( \rightarrow \) and \( \rightarrow' \) that are interpreted by \( > \) and \( \geq \) (over the naturals), respectively. This model satisfies the sentences in Figure 3 that translate the sentences (13)-(12) in Figure 2. The validity of (13)-(17) and (23)-(24) is obvious because \( \mathcal{A}_S = \mathcal{A}_{S_1} = \mathbb{N} \) and by reflexivity and transitivity of \( \geq \) and the fact that \( > \subseteq \geq \). With regard to (18), it holds due to our specific choice for \( \mathcal{A}_{S_2} \): since \( \mathcal{A}_{S_2} = \{0\} \), \( x \) 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 \( \mathcal{A}_S \) and \( \mathcal{A}_{S_1} \), termination of Toyama0S 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 \( \rightarrow \) has been chosen to be a well-founded ordering, which is essential to conclude termination of Toyama0S from the fact that \( \mathcal{A}' \) 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 \( \mathcal{I} \) is a bottom-up process where things remain ‘unspecified’ until an attempt to solve some constraints obtained from \( \mathcal{I} \) 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 $\mathcal{A}_i$ for $1 \leq i \leq k$.

2. Sentences $\varphi \in \mathcal{S}$ are used to obtain a new set $\mathcal{S}^\varphi$ of parametric sentences $\exists \varphi^\varphi$ with existentially quantified parameters $a_1, \ldots, a_n$. Such parameters range over appropriate (constraint solving) domains $D_1, \ldots, D_n$.

3. Then, $\mathcal{S}^\varphi$ 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 $\sigma(\varphi^\varphi)$ (an instantiation of the parameters in $\varphi^\varphi$) 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 $\mathcal{S}^\varphi$. In this paper $\mathcal{S}^\varphi$ consists of arithmetic sentences, using numeric orderings as predicates. Actually, an important issue is handling parametric formulas containing implications of the form

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

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 $\sum a_kx_k$ 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 $\vec{\lambda} \in \mathbb{R}_+^k$ 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. 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 $\mathcal{A}_i$ 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 $C \in \mathbb{R}^{m \times n}$, and $\vec{b} \in \mathbb{R}^m$, the set $D(C, \vec{b}) = \{\vec{x} \in \mathbb{R}^n \mid C\vec{x} \geq \vec{b}\}$ is called a convex polytopic domain.
In Definition 3, vectors \( \vec{x}, \vec{y} \in \mathbb{R}^n \) are compared using the coordinate-wise extension of the ordering \( \geq \) among numbers (by abuse, we use the same symbol): \( \vec{x} = (x_1, \ldots, x_n)^T \geq (y_1, \ldots, y_n)^T = \vec{y} \) if and only if \( x_1 \geq y_1 \land \cdots \land x_n \geq y_n \). Convex domains can be parameterized by considering a subset \( N \subseteq \mathbb{R} \) (e.g., \( \mathbb{N}, \mathbb{Z}, \mathbb{Q}, \) etc.) with \( C \in \mathbb{R}^{m \times n} \), and \( \vec{b} \in \mathbb{R}^m \) and defining \( D_N(C, \vec{b}) = \{ \vec{x} \in \mathbb{R}^n \mid C \vec{x} \geq \vec{b} \} \).

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

<table>
<thead>
<tr>
<th>Sort</th>
<th>( C^s )</th>
<th>( \vec{b}^s )</th>
<th>( \mathcal{A}_s = D(C^s, \vec{b}^s) )</th>
</tr>
</thead>
<tbody>
<tr>
<td>( \emptyset )</td>
<td>(0)</td>
<td>(1)</td>
<td>( \emptyset )</td>
</tr>
<tr>
<td>Nat</td>
<td>(1)</td>
<td>(1)</td>
<td>( [0, +\infty) )</td>
</tr>
<tr>
<td>NzNat</td>
<td>(1)</td>
<td>(1)</td>
<td>( [1, +\infty) )</td>
</tr>
<tr>
<td>Zero</td>
<td>(1, -1)(^t)</td>
<td>(0, 0)(^t)</td>
<td>( {0} )</td>
</tr>
<tr>
<td>Bool</td>
<td>(1, -1)(^t)</td>
<td>(0, -1)(^t)</td>
<td>( [0, 1] )</td>
</tr>
<tr>
<td>Char</td>
<td>(1, -1)(^t)</td>
<td>(0, -255)(^t)</td>
<td>( [0, 255] )</td>
</tr>
</tbody>
</table>

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

### 5.1 Domains

We interpret sorts \( s \in S \) as convex domains \( \mathcal{A}_s = D(C^s, \vec{b}^s) \), where \( C^s \in \mathbb{R}^{m_s \times n_s} \) is an \( m_s \times n_s \)-matrix and \( \vec{b}^s \in \mathbb{R}^{m_s} \). Thus, \( \mathcal{A}_s \subseteq \mathbb{R}^{n_s} \). 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 \( D(C^s, \vec{b}^s) \) will be settled by the subsequent constraint solving process.

**Remark 3** For 1-dimensional convex domains \( D(C^s, \vec{b}^s) \subseteq \mathbb{R} \) (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 \( C^s = \begin{pmatrix} C^s_1 & C^s_2 \end{pmatrix} \) and \( \vec{b}^s = (b^s_1, b^s_2)^T \) means that \( C^s_1 \vec{x} \geq b^s_1 \) and \( C^s_2 \vec{x} \geq b^s_2 \). As shown in Example 4, fixing \( m_s = 2 \) and using \( \mathbb{Z} \) as domain for parameters \( b_1 \) and \( c_1 \) 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 \( n_s = 1 \).

#### 5.1.1 Non-empty convex domains

An important requirement in termination analysis is that the domain \( D(C, \vec{b}) \subseteq \mathbb{R}^n \) 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 \( \xi \) of the appropriate sort \( S \) (to be interpreted by \( D(C, \vec{b}) \)) in the signature: \( \xi : S \). 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 \( C \xi^T \geq \vec{b} \) to be satisfied by a dummy constant \( \xi \in \mathbb{R}^n \).

#### 5.1.2 Convex domains which are bounded from below

In some applications, it is useful to guarantee that a semantic domain \( \mathcal{A} \) is bounded from below. In our setting, the following sentence (which is universally quantified on variable \( x \)):

\[
C \vec{x} \geq \vec{b} \Rightarrow \vec{x} \geq \vec{a}
\]

\(^4\)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.
guarantees that \( \mathcal{A} = D(C, \vec{b}) \) is bounded from below; here \( \alpha \) is a fresh constant whose value will be established by the constraint solving process.

5.1.3 Compatibility with the subsort relation

Regarding the subsort relation, if \( s \leq s' \), then \( \mathcal{A}_s = D(C^s, \vec{b}^s) \subseteq D(C^{s'}, \vec{b}^{s'}) = \mathcal{A}_{s'} \) must hold. Such a condition is expressed by the universally quantified formula:

\[
C^s x \geq \vec{b}^s \Rightarrow C^{s'} x \geq \vec{b}^{s'}
\]

(26)

5.2 Functions

A many-sorted convex matrix interpretation for \( f : s_1 \cdots s_k \rightarrow s \) is a linear expression \( F_1 x_1 + \cdots + F_k x_k + F_0 \) such that (1) for all \( i, 1 \leq i \leq k, F_i \in \mathbb{R}^{n_i \times n_i} \) are matrices and \( x_i \) are variables ranging on \( \mathbb{R}^{n_i} \), (2) \( F_0 \in \mathbb{R}^{n_s} \), and (3) it ranges on \( D(C^s, \vec{b}^s) \) whenever variables \( x_i \) take value on the corresponding domain \( D(C^{s_i}, \vec{b}^{s_i}) \), i.e., that satisfies the following algebraicity condition:

\[
\forall x_1 \in \mathbb{R}^{n_{s_1}}, \ldots, x_k \in \mathbb{R}^{n_k} \left( \bigwedge_{i=1}^{k} C^{s_i} x_i \geq \vec{b}^{s_i} \Rightarrow C^{s_i}(F_1 x_1 + \cdots + F_k x_k + F_0) \geq \vec{b}^{s_i} \right)
\]

For overloaded symbols \( f \in \Sigma_{w,s} \cap \Sigma_{w',s'} \) with \( w \leq w' \), we must have \( s \leq s' \) as well. We have to guarantee that the interpretations \( f_{w,s}^{\alpha} \) and \( f_{w',s'}^{\alpha} \) coincide on \( \mathcal{A}_w \) (see Section 5.1). 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 \( n_s = n_s' \) for all \( i, 1 \leq i \leq k \). Furthermore, \( n_s = n_{s'} \) as well. Therefore, if \( f_{w,s}^{\alpha} = \sum_{i=1}^{k} F_i x_i + F_0 \) and \( f_{w',s'}^{\alpha} = \sum_{i=1}^{k} F'_i x_i + F'_0 \), the desired condition can be written as follows:

\[
\forall x_1 \in \mathcal{A}_{s_1}, \ldots, \forall x_k \in \mathcal{A}_{s_k}, \sum_{i=1}^{k} (F_i - F'_i) x_i + F_0 - F'_0 = 0 \quad \text{or, equivalently:}
\]

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

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. According to [9], \( >_\delta \) is well-founded over subsets \( A \subseteq \mathbb{R} \) that are bounded from below.

6 Automatic treatment of the running example

Since we deal with three different sorts \( S, S_1, \) and \( S_2 \), we consider three convex domains:

\[
\mathcal{A}_S = D(C^S, \vec{b}^S) \quad \mathcal{A}_{S_1} = D(C^{S_1}, \vec{b}^{S_1}) \quad \mathcal{A}_{S_2} = D(C^{S_2}, \vec{b}^{S_2})
\]
where $C^S, C^{S_1}, C^{S_2} \in \mathbb{R}^{2 \times 1}$ and $\vec{b}^S, \vec{b}^{S_1}, \vec{b}^{S_2} \in \mathbb{R}^2$. By requiring non-emptiness, we obtain the first constraints for our running example:

\begin{align}
C^S_1 k &\geq b^S_1 \land C^S_2 k \geq b^S_2 \\
C^{S_1}_1 k' &\geq b^{S_1}_1 \land C^{S_1}_2 k' \geq b^{S_1}_2
\end{align}

(27) (28)

where $k$ and $k'$ are dummy elements $k, k' \in \mathbb{R}$ for $S$ and $S_1$. However, since Toyama0S already includes a symbol 1 of sort $S_1$, constraint (28) 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 $S_1$ (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, (27) could be avoided too.

We guarantee that $\alpha_2$ and $\alpha_{S_1}$ are both bounded from below with the following constraints:

\begin{align}
C^S x &\geq b^S_1 \land C^S x \geq b^S_2 \implies x \geq \alpha \\
C^{S_1} x &\geq b^{S_1}_1 \land C^{S_1} x \geq b^{S_1}_2 \implies x \geq \alpha'
\end{align}

(29) (30)

for constants $\alpha$ and $\alpha'$, where $x$ is universally quantified (but $\alpha$ and $\alpha'$ are treated as new, existentially quantified, parameters). Since $S_2 \leq S_1$, we add the following sentence (universally quantified in $x$):

\begin{align}
C^{S_2} x &\geq b^{S_2}_1 \land C^{S_2} x \geq b^{S_2}_2 \implies C^{S_1} x \geq b^{S_1}_1 \land C^{S_1} x \geq b^{S_1}_2
\end{align}

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:

\begin{align}
C^{S_2} x &\geq b^{S_2}_1 \land C^{S_2} x \geq b^{S_2}_2 \implies C^{S_1} x \geq b^{S_1}_1 \\
C^{S_2} x &\geq b^{S_2}_1 \land C^{S_2} x \geq b^{S_2}_2 \implies C^{S_1} x \geq b^{S_1}_2
\end{align}

(31) (32)

With regard to function symbols, since $n_f = 1$ 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{align}
[0] = z_0 \\
[f] \langle x, y, z \rangle = f_1 x + f_2 y + f_3 z + f_0 &
[1] = u_0 \\
[g] \langle x, y \rangle = g_1 x + g_2 y + g_0
\end{align}

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

\begin{align}
C^{S_2} z_0 &\geq b^{S_2}_1 \land C^{S_2} z_0 \geq b^{S_2}_2 \\
C^{S_1} u_0 &\geq b^{S_1}_1 \land C^{S_1} u_0 \geq b^{S_1}_2
\end{align}

(33) (34)

\begin{align}
\bigwedge_{i=1}^2 C^{S_1} x \geq b^{S_1}_i \land \bigwedge_{i=1}^2 C^{S_1} y \geq b^{S_1}_i \land \bigwedge_{i=1}^2 C^{S_1} z \geq b^{S_1}_i &\implies C^{S}_1 (f_1 x + f_2 y + f_3 z + f_0) \geq b^{S}_1 \\
\bigwedge_{i=1}^2 C^{S_1} x \geq b^{S_1}_i \land \bigwedge_{i=1}^2 C^{S_1} y \geq b^{S_1}_i \land \bigwedge_{i=1}^2 C^{S_1} z \geq b^{S_1}_i &\implies C^{S}_2 (f_1 x + f_2 y + f_3 z + f_0) \geq b^{S}_2
\end{align}

(35) (36)

\begin{align}
\bigwedge_{i=1}^2 C^{S_1} x \geq b^{S_1}_i \land \bigwedge_{i=1}^2 C^{S_1} y \geq b^{S_1}_i \land \bigwedge_{i=1}^2 C^{S_1} z \geq b^{S_1}_i &\implies C^{S_1} (g_1 x + g_2 y + g_0) \geq b^{S_1}_1 \\
\bigwedge_{i=1}^2 C^{S_1} x \geq b^{S_1}_i \land \bigwedge_{i=1}^2 C^{S_1} y \geq b^{S_1}_i \land \bigwedge_{i=1}^2 C^{S_1} z \geq b^{S_1}_i &\implies C^{S_2} (g_1 x + g_2 y + g_0) \geq b^{S_2}_1
\end{align}

(37) (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).

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

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

   $$ C_i S t \geq b_i^S \land C_i S t \geq b_i^S \Rightarrow t \geq t \quad (39) $$
   
   $$ C_i S t \geq b_i^{S1} \land C_i S t \geq b_i^{S1} \Rightarrow t \geq t \quad (40) $$

   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):

   $$ \forall i \leq 2 \left[ C_i S t \geq b_i^S \land C_i S t \geq b_i^S \land C_i S t \geq b_i^S \land C_i S t \geq b_i^S \Rightarrow t \geq t + \delta \land t' \geq u \Rightarrow t \geq u \right] \quad (41)$$
   
   $$ \forall i \leq 2 \left[ C_i S t \geq b_i^{S1} \land C_i S t \geq b_i^{S1} \land C_i S t \geq b_i^{S1} \land C_i S t \geq b_i^{S1} \Rightarrow t \geq t + \delta \land t' \geq u \Rightarrow t \geq u \right] \quad (42)$$

3. Instances of the congruence rule (C), corresponding to (5)-(9), where we use $t \in S_{S1}$ instead of $C_i S t \geq b_i^{S1} \land C_i S t \geq b_i^{S1}$:

   $$ \forall i \leq 3 \left[ t_i \in S_{S1} \land t'_i \in S_{S1} \land t_1 \geq t'_1 + \delta \Rightarrow f_1 t_1 + f_2 t_2 + f_3 t_3 + f_0 \geq f_1 t'_1 + f_2 t'_2 + f_3 t'_3 + f_0 + \delta \right] \quad (43)$$
   
   $$ \forall i \leq 3 \left[ t_i \in S_{S1} \land t'_i \in S_{S1} \land t_2 \geq t'_2 + \delta \Rightarrow f_1 t_1 + f_2 t_2 + f_3 t_3 + f_0 \geq f_1 t'_1 + f_2 t'_2 + f_3 t'_3 + f_0 + \delta \right] \quad (44)$$
   
   $$ \forall i \leq 3 \left[ t_i \in S_{S1} \land t'_i \in S_{S1} \land t_3 \geq t'_3 + \delta \Rightarrow f_1 t_1 + f_2 t_2 + f_3 t_3 + f_0 \geq f_1 t'_1 + f_2 t'_2 + f_3 t'_3 + f_0 + \delta \right] \quad (45)$$
   
   $$ \forall i \leq 3 \left[ t_i \in S_{S1} \land t'_i \in S_{S1} \land t_4 \geq t'_4 + \delta \Rightarrow g_1 t_1 + g_2 t_2 + g_0 \geq g_1 t'_1 + g_2 t'_2 + g_0 + \delta \right] \quad (46)$$
   
   $$ \forall i \leq 3 \left[ t_i \in S_{S1} \land t'_i \in S_{S1} \land t_5 \geq t'_5 + \delta \Rightarrow g_1 t_1 + g_2 t_2 + g_0 \geq g_1 t'_1 + g_2 t'_2 + g_0 + \delta \right] \quad (47)$$

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

   $$ C_i S x \geq b_i^S \land C_i S x \geq b_i^S \Rightarrow f_1 x_0 + f_2 u_0 + f_3 x + f_0 \geq f_1 x + f_2 x + f_3 x + f_0 + \delta \quad (48)$$
   
   $$ C_i S x \geq b_i^{S1} \land C_i S x \geq b_i^{S1} \land C_i S y \geq b_i^{S1} \land C_i S y \geq b_i^{S1} \Rightarrow g_1 x + g_2 y + g_0 \geq x + \delta \quad (49)$$
   
   $$ C_i S x \geq b_i^{S1} \land C_i S x \geq b_i^{S1} \land C_i S y \geq b_i^{S1} \land C_i S y \geq b_i^{S1} \Rightarrow g_1 x + g_2 y + g_0 \geq y + \delta \quad (50)$$
6.1 Synthesis of the model

The conjunction of all previous sentences (27)–(50) (perhaps dropping some of them, as suggested in previous sections) yields an $\exists \forall$-sentence (the $\exists$ concerns existential quantification of $k$, $k'$, $\alpha$, $\alpha'$, $\delta$, 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, $\tilde{b} \in \mathbb{R}^k$, $\tilde{c} \in \mathbb{R}^n$ and $\beta \in \mathbb{R}$, the application of the affine form of Farkas’ Lemma to prove that the universally quantified sentence $Ax \geq \tilde{b} \implies \tilde{c}^T x \geq \beta$ holds tries to find a vector $\tilde{\lambda}$ of $k$ non-negative numbers $\tilde{\lambda} \in \mathbb{R}_0^k$ such that the constraints $\tilde{c} = A^T \tilde{\lambda}$ and $\tilde{\lambda}^T \tilde{b} \geq \beta$ hold.

Example 8 We apply the Affine form of Farkas’ Lemma to sentence (29) as follows: the associated matrix $A$ is actually a vector $(C_1^S, C_2^S)^T$ and $\tilde{b} = (b_1^S, b_2^S)^T$; we have that $\tilde{c} = (1)^T$ is a one-dimensional vector and finally $\beta = \alpha$. Then, we seek a vector $\tilde{\lambda} = (\lambda_1, \lambda_2)^T$ with $\lambda_1, \lambda_2 \geq 0$ that satisfies the (in)equalities:

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

The satisfiability of these inequations (a constraint solving problem for parameters $C_1^S, C_2^S, b_1^S, b_2^S, \lambda_1, \lambda_2$ and $\alpha$), is equivalent to the satisfiability of (29).

Example 9 Sentence (50) is not in affine form, but we can easily fix it as follows:

$$C_1^{S1} x \geq b_1^{S1} \land C_2^{S1} x \geq b_2^{S1} \land C_1^{S1} y \geq b_1^{S1} \land C_2^{S1} y \geq b_2^{S1} \implies g_1 x + (g_2 - 1) y \geq \delta - g_0 \quad (51)$$

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 $\tilde{b}$ has four components: $\tilde{b} = (b_1^{S1}, b_2^{S1}, b_1^{S1}, b_2^{S1})^T$. Now, $\tilde{c} = (g_1, g_2 - 1)^T$ and $\beta = \delta - g_0$. Thus, we want now a vector $\tilde{\lambda} = (\lambda_1, \lambda_2, \lambda_3, \lambda_4)^T$ that satisfies:

$$\lambda_1 b_1^{S1} + \lambda_2 b_2^{S1} + \lambda_3 b_1^{S1} + \lambda_4 b_2^{S1} \geq \delta - g_0 \quad \lambda_1, \lambda_2, \lambda_3, \lambda_4 \geq 0$$

for some values of the parameters.

Remark 4 Note that each implication processed using Farkas’ Lemma can use a different vector $\tilde{\lambda}$, 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:

$$C_1^S = 1 \quad C_2^S = 1 \quad C_1^{S1} = 1 \quad C_2^{S1} = 1 \quad C_1^{S2} = 1 \quad C_2^{S2} = -1$$ $$b_1^S = 0 \quad b_2^S = 0 \quad b_1^{S1} = 0 \quad b_2^{S1} = 0 \quad b_1^{S2} = 0 \quad b_2^{S2} = 0$$ $$f_1 = 1 \quad f_2 = 1 \quad f_3 = 1 \quad f_0 = 0 \quad g_1 = 1 \quad g_2 = 1 \quad g_0 = 0 \quad z_0 = 0 \quad u_0 = 1$$ $$k = 0 \quad k' = 0 \quad \alpha = 0 \quad \alpha' = 0 \quad \delta = 1$$
(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 \( \mathcal{A} \) for the theory \( \mathcal{S} \) in our running example:

\[
\begin{align*}
\mathcal{A}_S &= [0, +\infty) \\
\mathcal{A}_{S1} &= [0, +\infty) \\
\mathcal{A}_{S2} &= \{0\} \\
\end{align*}
\]

\[
\begin{align*}
f^\mathcal{A}_{S1S1S1}(x, y, z) &= x + y + z \\
f^\mathcal{A}_{S1S1}(x, y) &= x + y + 1 \\
1^\mathcal{A}_{\lambda, S1} &= 1 \\
\end{align*}
\]

\[
\begin{align*}
t \rightarrow^\mathcal{A}_{SS} t' &\iff t > t' \\
t (\rightarrow^*)^\mathcal{A}_{SS} t' &\iff t \geq t' \\
t \rightarrow^\mathcal{A}_{S1S1} t' &\iff t > t' \\
t (\rightarrow^*)^\mathcal{A}_{S1S1} t' &\iff t \geq t'
\end{align*}
\]

7 Related work and conclusions

Our extension of derived algebras \[5\] to derived models for order-sorted first-order theories follows some of the ideas in \[6\]. 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 \[17\]. 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 \[16\], 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 \textsc{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.

References


