Solving Recursion-Free Horn Clauses over LI+UIF

Ashutosh Gupta[1,2], Corneliu Popeea[2], and Andrey Rybalchenko[2]

[1]IST Austria [2]Technische Universität München

Abstract. Verification of programs with procedures, multi-threaded programs, and higher-order functional programs can be effectively automated using abstraction and refinement schemes that rely on spurious counterexamples for abstraction discovery. The analysis of counterexamples can be automated by a series of interpolation queries, or, alternatively, as a constraint solving query expressed by a set of recursion free Horn clauses. (A set of interpolation queries can be formulated as a single constraint over Horn clauses with linear dependency structure between the unknown relations.) In this paper we present an algorithm for solving recursion free Horn clauses over a combined theory of linear real/rational arithmetic and uninterpreted functions. Our algorithm performs resolution to deal with the clausal structure and relies on partial solutions to deal with (non-local) instances of functionality axioms.

1 Introduction

Constraint solving is a vehicle of software verification that provides symbolic reasoning techniques for dealing with assertions describing program behaviors. In particular, abstraction and refinement techniques greatly benefit from applying constraint solving, where interpolation techniques [1–3, 5, 6, 10–12, 15–18] play a prominent role today. Roughly, interpolation computes an assertion that separates two mutually unsatisfiable assertions and only refers to their shared symbols.

Certain abstraction refinement tasks cannot be *directly* expressed as an interpolation question. For example, abstraction refinement for imperative programs with procedures [11] and for higher order functional programs [13, 19], require additional pre-processing that splits discovered spurious counterexamples in multiple ways and applies interpolation on each splitting. Alternatively, as exemplified by an abstraction refinement procedure for multi-threaded programs [9], this preprocessing and series of interpolation computations can be expressed using a single constraint that consists of a finite set of recursion-free Horn clauses interpreted over the logical theory that is used to describe program behaviors.

In this paper, we present an algorithm for solving Horn clauses over a combination of linear rational/real arithmetic, uninterpreted functions and queries. Our algorithm opens new possibilities for the development of abstraction refinement schemes by providing the verification method designer an expressive, declarative way to specify what the refinement procedure needs to compute using Horn clauses. Several existing abstraction refinement schemes can directly benefit from our algorithm, e.g., for programs with procedures [10,11], for multi-threaded programs [9], and for higher-order functional programs [13,19,20].

Related work In [9] we presented an algorithm that deals with recursion-free Horn clauses over linear real/rational arithmetic. Here, we present an extension with uninterpreted functions.

Technically, our treatment of uninterpreted functions can be seen as a generalization of partial interpolants [16] to partial solutions for recursion-free Horn clauses, i.e., clauses that do not have cyclic dependencies between the occurring queries. Our algorithm follows a general scheme of combining interpolation procedures for different theories [8,21].

The following example illustrates the relation to interpolation. First, we consider an interpolation question for a pair of mutually unsatisfiable assertions $a(x, y)$ and $b(y, z)$ in a logical theory. An interpolant is an assertion $I(y)$ such that $I(y)$ is a logical consequence of $a(x, y)$, $I(y)$ and $b(y, z)$ are mutually unsatisfiable, and $I(y)$ only contains non-theory constants that are shared by $a(x, y)$ and $b(y, z)$, which is y in our example. Now, we present our re-formulation of interpolation as a constraint solving question for constraints given by recursion-free Horn clauses. We introduce a relation $Q_I(y)$ that represents an interpolant that we want to compute. We represent the interpolation conditions by the following two Horn clauses: $a(x, y) \rightarrow Q_I(y)$ and $Q_I(y) \wedge b(y, z) \rightarrow \mathit{false}$. Any interpretation of $Q_I(y)$ that only refers to y (and theory constants) is an interpolant for $a(x, y)$ and $b(y, z)$. In Section 6, we discuss the relation of this paper with [16] in more detail.

Horn clauses with more that two unknown queries in the body do not directly correspond to interpolation problems. For example, we consider two relations $Q_I(y)$ and $Q_J(y)$ that represent assertions we want to compute together with the Horn clauses $a(x, y) \rightarrow Q_I(y)$, $b(y, z) \rightarrow Q_J(y)$, and $Q_I(y) \wedge Q_J(y) \rightarrow \mathit{false}$. Solving these clauses using interpolation requires two invocations of an interpolation procedure (i.e., interpolation between $a(x, y)$ and $b(y, z)$ determines $Q_I(y)$ interpolation between $b(y, z)$ and $Q_I(y)$ determines $Q_J(y)$) that we would like to avoid for efficiency considerations. Furthermore, by computing $Q_I(y)$ and $Q_J(y)$ one after the other it is not evident how to compute solutions satisfying certain preference conditions, e.g., where all constrants are within predefined bounds (such conditions are useful for abstraction refinement, as shown by the FOCI procedure [14]).

Organization Section 2 illustrates our algorithm. Section 3 provides formal definitions. We present the solving algorithm in Section 4 and discuss its correctness and complexity in Section 5. Section 6 conclude and clarifies the connection of our algorithm to interpolation procedures. Appendix A illustrates how Horn clauses can be used to refine abstraction for procedural and multi-threaded programs. Appendix B contains a complete example execution of our algorithm. Appendix C presents proofs of the key theorems.

2 Illustration

In this section, we shall illustrate our proposed algorithm by solving an example set of Horn clauses. The example set of Horn clauses $\mathcal{HC}$ is presented in Figure 1(a) and consists of three clauses. Our algorithm is looking for solutions to the two symbols $S(\mathtt{t},\mathtt{u},\mathtt{v})$ and $E(\mathtt{t},\mathtt{u})$ that we name *queries*. To obtain solutions over the domain of linear arithmetic and uninterpreted functions, our algorithm proceeds following three steps.

Resolution tree Our solving algorithm starts by constructing from $\mathcal{HC}$ a resolution tree R shown in Figure 1(b). We label nodes of R with indices for easy reference. From the first clause, the algorithm constructs the subtree rooted at label 2. In this subtree, we have edges between the node corresponding to the head of the clause (labeled 2) and the nodes corresponding to the body of the clause (labeled 3–6). A second subtree rooted at the node labelled 7 is constructed from the second clause. With the appearance of the queries $S(\mathtt{t},\mathtt{u},\mathtt{v})$ and $E(\mathtt{t},\mathtt{u})$ in the body of the third clause from $\mathcal{HC}$, the two previously constructed subtrees are extended in a tree with the root labeled corresponding to the clause head, $(1 : \mathit{false})$. The extension of the subtrees leads to the variables occurring in these subtrees to be renamed to a common set of variables $\mathtt{p},\mathtt{q},\mathtt{c}$. Note that, the set of clauses $\mathcal{HC}$ is satisfiable, and, consequently, the conjunction of the predicates from the leaves of the resolution tree is unsatisfiable.

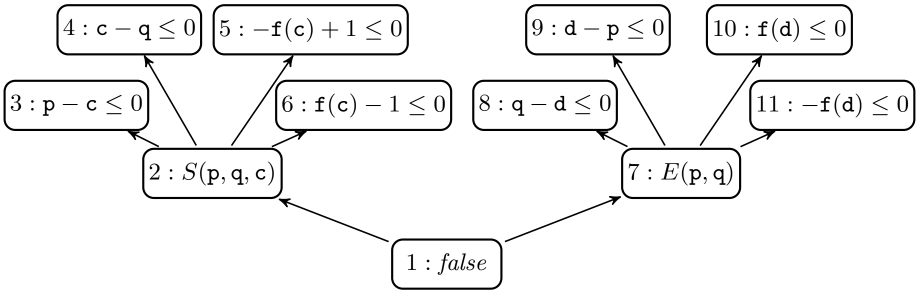
Proof tree Next, our algorithm constructs a proof tree that proves unsatisfiability of the constraints from the leaves of the resolution tree. For the resolution tree from Figure 1(b), our algorithm computes the proof tree P shown in Figure 1(c). A linear combination rule is applied to derive the constraint $(\mathtt{c}-\mathtt{d} \leq 0)$ from the premises $(\mathtt{c}-\mathtt{q} \leq 0)$ and $(\mathtt{q}-\mathtt{d} \leq 0)$. The linear combination rule is also used to derive $(\mathtt{d}-\mathtt{c} \leq 0)$ from the premises $(\mathtt{p}-\mathtt{c} \leq 0)$ and $(\mathtt{d}-\mathtt{p} \leq 0)$. A congruence rule is used to relate function symbols applied to equivalent arguments. This rule derives $(\mathtt{f}(\mathtt{c})-\mathtt{f}(\mathtt{d}) \leq 0)$ from the premises $(\mathtt{c}-\mathtt{d} \leq 0)$ and $(\mathtt{d}-\mathtt{c} \leq 0)$. Lastly, $(1 \leq 0)$ is derived by applying the linear combination rule on three premises, $(\mathtt{f}(\mathtt{d}) \leq 0)$, $(\mathtt{f}(\mathtt{c})-\mathtt{f}(\mathtt{d}) \leq 0)$, and $(-\mathtt{f}(\mathtt{c})+1 \leq 0)$.

Partial and final solutions The proof tree P explicates the inference rules and the order in which to apply them to derive the false constraint $(1 \leq 0)$. The main idea behind our solving algorithm is to apply corresponding inference rules in the same order to derive a solution for the Horn clauses. We obtain an annotated proof tree (see Figure 1(d)) where for each of the premises used in P , our algorithm creates one tree with the same number of nodes as R . We call these trees, which are annotated with formulas that will be explained next, partial-solution trees.

The tree Π_1 corresponds to the premise $(\mathtt{c}-\mathtt{q} \leq 0)$, Π_2 corresponds to the premise $(\mathtt{q}-\mathtt{d} \leq 0)$ and both trees are shown in Figure 2. Two or more premises are used to derive a new fact in the proof tree and, likewise, two or more corresponding partial-solution trees are used to derive a new tree using a

$$\begin{aligned}\mathcal{HC} = \{\ & \forall \mathtt{p},\mathtt{q},\mathtt{c} : \mathtt{p} \le \mathtt{c} \wedge \mathtt{c} \le \mathtt{q} \wedge -\mathtt{f}(\mathtt{c}) + 1 \le 0 \wedge \mathtt{f}(\mathtt{c}) - 1 \le 0 \rightarrow S(\mathtt{p},\mathtt{q},\mathtt{c}),\\ & \forall \mathtt{r},\mathtt{s},\mathtt{d} : \mathtt{s} \le \mathtt{d} \wedge \mathtt{d} \le \mathtt{r} \wedge \mathtt{f}(\mathtt{d}) \le 0 \wedge -\mathtt{f}(\mathtt{d}) \le 0 \rightarrow E(\mathtt{r},\mathtt{s}),\\ & \forall \mathtt{t},\mathtt{u},\mathtt{v} : S(\mathtt{t},\mathtt{u},\mathtt{v}) \wedge E(\mathtt{t},\mathtt{u}) \rightarrow \mathit{false}\ \}\end{aligned}$$

(a)


4 : $\mathtt{c} - \mathtt{q} \le 0$
5 : $-\mathtt{f}(\mathtt{c}) + 1 \le 0$
9 : $\mathtt{d} - \mathtt{p} \le 0$
10 : $\mathtt{f}(\mathtt{d}) \le 0$
3 : $\mathtt{p} - \mathtt{c} \le 0$
6 : $\mathtt{f}(\mathtt{c}) - 1 \le 0$
8 : $\mathtt{q} - \mathtt{d} \le 0$
11 : $-\mathtt{f}(\mathtt{d}) \le 0$
2 : $S(\mathtt{p},\mathtt{q},\mathtt{c})$
7 : $E(\mathtt{p},\mathtt{q})$
1 : *false*

(b)

$$\dfrac{\dfrac{}{\mathtt{f}(\mathtt{d}) \le 0} \qquad \dfrac{\dfrac{\mathtt{c} - \mathtt{q} \le 0 \quad \mathtt{q} - \mathtt{d} \le 0}{\mathtt{c} - \mathtt{d} \le 0} \qquad \dfrac{\mathtt{p} - \mathtt{c} \le 0 \quad \mathtt{d} - \mathtt{p} \le 0}{\mathtt{d} - \mathtt{c} \le 0}}{\mathtt{f}(\mathtt{c}) - \mathtt{f}(\mathtt{d}) \le 0} \qquad \dfrac{}{-\mathtt{f}(\mathtt{c}) + 1 \le 0}}{1 \le 0}$$

(c)

$$\dfrac{\dfrac{}{\mathtt{f}(\mathtt{d}) \le 0[\ldots]} \qquad \dfrac{\dfrac{\mathtt{c} - \mathtt{q} \le 0[\Pi_1] \quad \mathtt{q} - \mathtt{d} \le 0[\Pi_2]}{\mathtt{c} - \mathtt{d} \le 0[\Pi_3]} \qquad \ldots}{\mathtt{f}(\mathtt{c}) - \mathtt{f}(\mathtt{d}) \le 0[\ldots]} \qquad \dfrac{}{-\mathtt{f}(\mathtt{c}) + 1 \le 0[\ldots]}}{1 \le 0[\Pi]}$$

(d)

Fig. 1. (a) A set of Horn clauses $\mathcal{HC}$. (b) Corresponding resolution tree R . (c) Proof of unsatisfiability P for the constraints from the leaves of the resolution tree. For abbreviation, we did not mark nodes of subtree of $\mathtt{f}(\mathtt{c}) - \mathtt{f}(\mathtt{d}) \le 0$ with the applied proof rules. (d) A part of the annotated proof tree. The partial solutions Π_1, Π_2, Π_3, and Π are presented in Figure 2.

specific inference rule. The two trees Π_1 and Π_2 shown in the top part of Figure 2 are combined using a rule corresponding to the arithmetic combination rule. The rule takes a pair of corresponding nodes, one from Π_1 and one from Π_2 , and computes a node in the resulting tree Π_3 . For the node labeled ($2 : \mathtt{c} - \mathtt{q} \le 0$) from Π_1 and the node labeled ($2 : 0 \le 0$) from Π_2, the algorithm adds the two

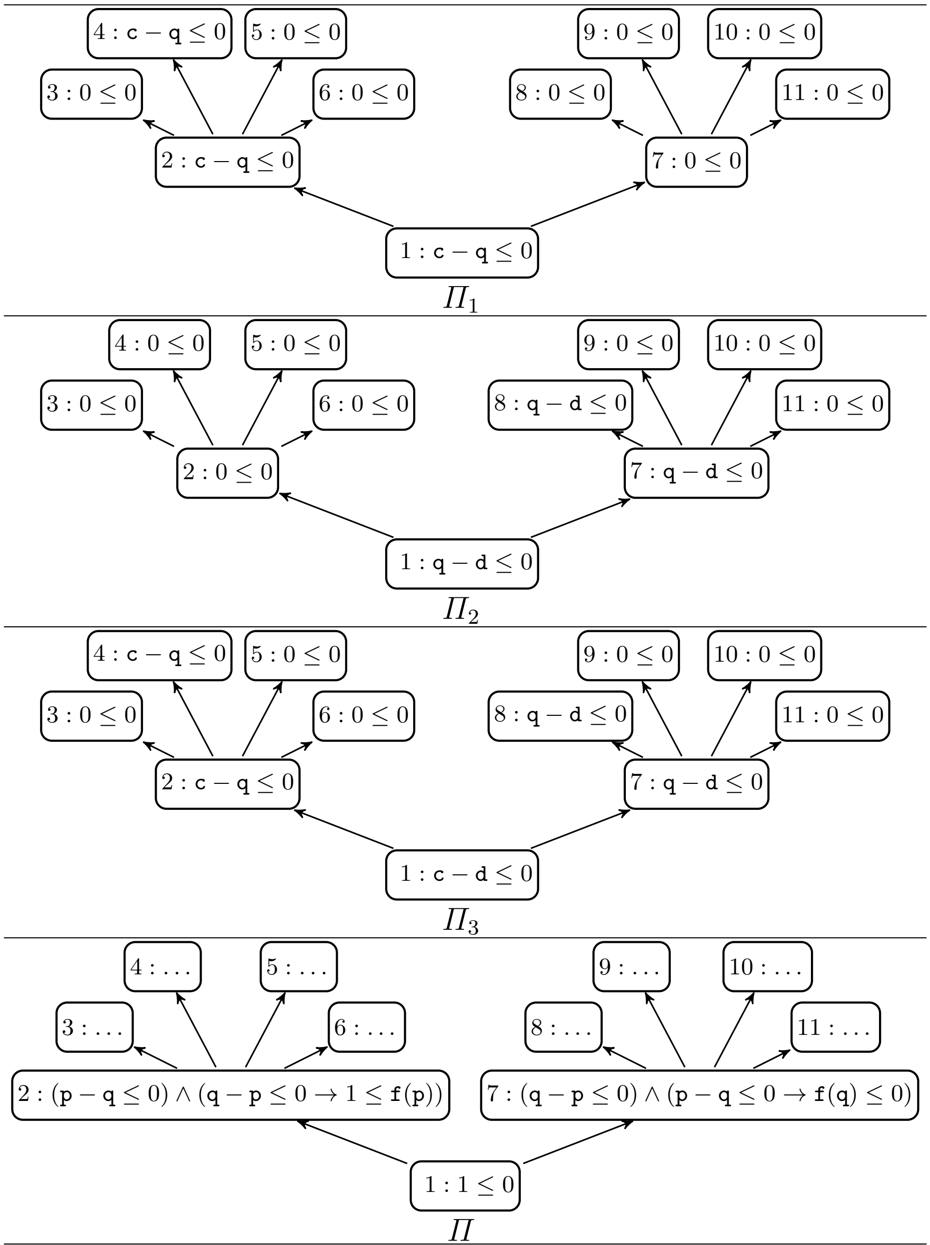

$4 : \mathtt{c} - \mathtt{q} \leq 0$
$5 : 0 \leq 0$
$9 : 0 \leq 0$
$10 : 0 \leq 0$
$3 : 0 \leq 0$
$6 : 0 \leq 0$
$8 : 0 \leq 0$
$11 : 0 \leq 0$
$2 : \mathtt{c} - \mathtt{q} \leq 0$
$7 : 0 \leq 0$
$1 : \mathtt{c} - \mathtt{q} \leq 0$
Π_1
$4 : 0 \leq 0$
$5 : 0 \leq 0$
$9 : 0 \leq 0$
$10 : 0 \leq 0$
$3 : 0 \leq 0$
$6 : 0 \leq 0$
$8 : \mathtt{q} - \mathtt{d} \leq 0$
$11 : 0 \leq 0$
$2 : 0 \leq 0$
$7 : \mathtt{q} - \mathtt{d} \leq 0$
$1 : \mathtt{q} - \mathtt{d} \leq 0$
Π_2
$4 : \mathtt{c} - \mathtt{q} \leq 0$
$5 : 0 \leq 0$
$9 : 0 \leq 0$
$10 : 0 \leq 0$
$3 : 0 \leq 0$
$6 : 0 \leq 0$
$8 : \mathtt{q} - \mathtt{d} \leq 0$
$11 : 0 \leq 0$
$2 : \mathtt{c} - \mathtt{q} \leq 0$
$7 : \mathtt{q} - \mathtt{d} \leq 0$
$1 : \mathtt{c} - \mathtt{d} \leq 0$
Π_3
$4 : \ldots$
$5 : \ldots$
$9 : \ldots$
$10 : \ldots$
$3 : \ldots$
$6 : \ldots$
$8 : \ldots$
$11 : \ldots$
$2 : (\mathtt{p} - \mathtt{q} \leq 0) \wedge (\mathtt{q} - \mathtt{p} \leq 0 \rightarrow 1 \leq \mathtt{f}(\mathtt{p}))$
$7 : (\mathtt{q} - \mathtt{p} \leq 0) \wedge (\mathtt{p} - \mathtt{q} \leq 0 \rightarrow \mathtt{f}(\mathtt{q}) \leq 0)$
$1 : 1 \leq 0$
Π

Fig. 2. Four partial-solution trees Π_1 , Π_2 , Π_3 , and Π . Π_1 and Π_2 are derived from the nodes $(\mathtt{c} - \mathtt{q} \leq 0)$ and $(\mathtt{q} - \mathtt{d} \leq 0)$ from the proof tree P from Figure 1(d). Π_3 is obtained by applying a combination rule to Π_1 and Π_2. Π annotates the false constraint $(1 \leq 0)$ from P and the final solution of $\mathcal{HC}$ can be derived from Π . In particular, the nodes labeled “2” and “7” contain the solutions for $S(\mathtt{p}, \mathtt{q}, \mathtt{c})$ and $E(\mathtt{p}, \mathtt{q})$, respectively.

constraints and creates a node labeled $(2 : \mathtt{c} - \mathtt{q} \leq 0)$ in Π_3 . Similarly, the nodes labeled $(1 : \mathtt{c} - \mathtt{q} \leq 0)$ and $(1 : \mathtt{q} - \mathtt{d} \leq 0)$ are used to obtain a node labeled $(1 : \mathtt{c} - \mathtt{d} \leq 0)$ in Π_3 .

Following the derivation of the proof tree P, inference rules are used to combine partial-solution trees until a final-solution tree corresponding to the rule applied at the bottom of the proof tree. The final-solution tree is Π and is shown in Figure 2. The node labeled "2" contains the solution for $S(\mathtt{p}, \mathtt{q}, \mathtt{c})$ and it can be simplified to $S(\mathtt{p}, \mathtt{q}, \mathtt{c}) = (\mathtt{p} < \mathtt{q} \vee \mathtt{p} \leq \mathtt{q} \wedge \mathtt{f}(\mathtt{p}) \geq 1)$. The solution from the node labeled "7" can be simplified to $E(\mathtt{p}, \mathtt{q}) = (\mathtt{p} > \mathtt{q} \vee \mathtt{p} \geq \mathtt{q} \wedge \mathtt{f}(\mathtt{p}) \leq 0)$. The solutions obtained for $S(\mathtt{p}, \mathtt{q}, \mathtt{c})$ and $E(\mathtt{p}, \mathtt{q})$ indeed satisfy the set of Horn clauses $\mathcal{HC}$ from Figure 1(a).

3 Recursion-free Horn clauses

This section presents auxiliary definitions together with the syntax and semantics of recursion-free Horn clauses over linear arithmetic, uninterpreted functions, and queries.

Syntax We assume countable sets of *variables* $\mathcal{V}$, with $v \in \mathcal{V}$, *function symbols* $\mathcal{F}$, with $f \in \mathcal{F}$, and *predicate symbols* $\mathcal{P}$, with $p \in \mathcal{P}$. Let the arity of function and predicate symbols be encoded in their names. In addition, we assume a set of *number symbols* $\mathcal{N}$, with $\{0, n\} \subseteq \mathcal{N}$, and an inequality symbol $\leq$. Then, we define:

$$\begin{array}{llll} \text{terms} \ni t ::= n \mid nv \mid t + t \mid f(t, \dots, t) & \text{bodies} & \ni b & ::= a \mid q \mid b \wedge b \\ \text{atoms} \ni a ::= t \leq 0 & \text{heads} & \ni h & ::= a \mid q \mid \mathit{false} \\ \text{queries} \ni q ::= p(v, \dots, v) & \text{Horn clauses} & \ni s & ::= b \rightarrow h \end{array}$$

Without loss of generality, as justified later, we assume that all variables that occur in a query are distinct.

A set of Horn clauses defines a binary *dependency* relation on predicate symbols. A predicate symbol $p \in \mathcal{P}$ *depends* on a predicate symbol $p_i \in \mathcal{P}$ if there is a Horn clause $\dots \wedge p_i(\dots) \wedge \dots \rightarrow p(\dots)$, i.e., when p appears in the head of a clause that contains p_i in its body. A set of Horn clauses is *recursion-free* if the corresponding dependency relation does not contain any cycles. A set of Horn clauses is *tree-like* if the corresponding dependency relation defines a tree-like graph, i.e., when 1) each predicate symbol appears at most once in the set of bodies and at most once in the set of heads of the given clauses, 2) there is no clause with an atom in its head, 3) there is one clause whose head is *false*. For example, the set of clauses $\{p(v_1) \wedge p(v_2) \rightarrow q(v_1, v_2), q(v_3, v_4) \rightarrow \mathit{false}\}$ is not tree-like since the predicate symbol p appears more than once in the body of the first clause.

For the rest of the presentation, we consider a finite set of Horn clauses $\mathcal{HC}$ that satisfies the following conditions. First, we assume that each variable occurs in at most one clause and that all variables occurring in a query are distinct.

These assumptions simplify our presentation and can be established by an appropriate variable renaming and additional (in)equality constraints. Furthermore, we assume that $\mathcal{HC}$ is recursion-free and tree-like. The recursion-free assumption is critical for ensuring termination of the solving algorithm presented in this paper. The tree-like assumption simplifies our presentation without imposing any restrictions on the algorithm's applicability. Any finite set of recursion-free clauses can be transformed into the tree-like form. The solution for the computed tree-like form can be translated into the solution for the original set of clauses.

Finally, we define *constraints* together with a *conjunctive constraint* fragment below.

$$\text{constraints} \ni c ::= a \mid \neg c \mid c \wedge c \mid c \vee c \quad \text{conjunctive constraints} \ni \hat{c} ::= a \mid \hat{c} \wedge \hat{c}$$

Auxiliary definitions We assume the following standard functions. For dealing with trees, let $\mathit{nodes}(T)$ be the nodes of a tree T, $\mathit{root}(T)$ be the root node of T, $\mathit{leaves}(T)$ be the leaves of T, and $\mathit{subtree}(o, T)$ be the subtree of T rooted in its node o. Furthermore, let $\mathit{subterms}(C)$ be the subterms occurring in a constraint C and $\mathit{atoms}(C)$ be the atoms occurring in C. Let $\mathit{sym}(t)$ be the variables and uninterpreted function symbols occurring in a term t.

Let $\mathit{match}(p(v_1, \dots, v_n), p'(v'_1, \dots, v'_m))$ return a substitution $\{v_1 \mapsto v'_1, \dots, v_n \mapsto v'_m\}$ if $p = p'$ (and hence $n = m$). Thus, if a substitution σ is the result of $\mathit{match}(p'(v_1, \dots, v_n), p'(v'_1, \dots, v'_m))$ then $p'(v_1, \dots, v_n)\sigma = p'(v'_1, \dots, v'_m)$, i.e., by applying the substitution we equate the queries. For example, $\mathit{match}(p_1(v_1), q(v_2, v_3))$ is not defined, and $\mathit{match}(q(v_1, v_2), q(v_3, v_4)) = \{v_1 \mapsto v_3, v_2 \mapsto v_4\}$. We assume a canonical extension of the unifier application to constraints and their combination into sequences and sets.

Given two substitutions $\sigma_1 = \{v_1 \mapsto v'_1, \dots, v_n \mapsto v'_n\}$ and $\sigma_2 = \{w_1 \mapsto w'_1, \dots, w_m \mapsto w'_m\}$ over disjoint domains, i.e., $\{v_1, \dots, v_n\} \cap \{w_1, \dots, w_m\} = \emptyset$, we define a *combined* substitution $\sigma_1 + \sigma_2 = \{v_1 \mapsto v'_1, \dots, v_n \mapsto v'_n, w_1 \mapsto w'_1, \dots, w_m \mapsto w'_m\}$.

Semantics Let $\models$ be the (logical) satisfaction relation for our constraints in the combined theory of linear real/rational arithmetic and uninterpreted functions. We write $\models c$ when c is a valid constraint.

Let Σ be a function from queries to constraints. We assume that in the domain of Σ no two queries have an equal predicate symbol, all queries have disjoint variables, and each query is mapped to a constraint whose free variables occur in the query. For example, consider $\Sigma = \{p(v_1) \mapsto (v_1 \geq 0), q(v_2, v_3) \mapsto (v_2 \leq f(v_3))\}$.

We use Σ function to transform the set of Horn clauses containing queries into a set of query-free clauses as follows. In each clause $s \in \mathcal{HC}$ we replace each query q in s with the constraint $\Sigma(q')\sigma$ where q' is in the domain of Σ, queries q' and q have an equal predicate symbol, and $\sigma = \mathit{match}((q', q))$. For example, the above Σ transforms the clause $x \leq f(y) \wedge p(x) \wedge q(y, z) \rightarrow \mathit{false}$ into $x \leq f(y) \wedge (x \leq 0) \wedge (y \leq f(z)) \rightarrow \mathit{false}$.

```
    algorithm SolveHorn(li+uif)
    input
      $\mathcal{HC}$ : Horn clauses
    vars
      $R$ : resolution tree
      $C$ : conjunctive constraint
      $P$ : proof tree
      $A$ : annotated proof tree
    output
      $\Sigma$ : solution
    begin
1     $R$ := exhaustively apply RInit and RStep on $\mathcal{HC}$
2     $C$ := $\bigwedge leaves(R)$
3     if exists $P$ inferred from $C$ by PHyp, PComb, and PCong
4        such that $\models (root(P) \rightarrow 1 \leq 0)$
5     then
6        $A$ := exhaustively apply AHyp, AComb, and ACong on $P$
7        $false\ [\ \Pi\ ]$ := $root(A)$
8        $\Sigma$ := $\{(o, \pi) \mid (o, \pi) \in \Pi \wedge o \notin (leaves(R) \cup \{false\})\}$
9        return $\Sigma$
10    else
11       return "no solution exists"
    end.
```

Fig. 3. Solving algorithm SolveHorn(li+uif). Line 7 extracts the partial solution Π annotating the root node of A. Line 8 obtains Σ by restricting the domain of Π to intermediate nodes of R, i.e., to the nodes that are labeled by queries.

Let $\mathcal{HC}_\Sigma$ be the set of query-free clauses obtained by applying Σ. Σ is a *solution* for $\mathcal{HC}$ if each clause $c_\Sigma$in $\mathcal{HC}_\Sigma$ is a valid implication, i.e, $\models c_\Sigma$, and the following condition holds for the uninterpreted function symbols occurring in the range of the solution function. An uninterpreted function symbol f can occur in the solution $\Sigma(q)$ for a query q if f appears in the atoms of a Horn clause from $\mathcal{HC}$ whose head depends on q and in the atoms of a Horn clause from $\mathcal{HC}$, whose head does not depend on q. For example, given the clauses $\{f(v_1) = 0 \rightarrow p(v_1), f(v_2) = 1 \rightarrow q(v_2), v_3 = v_4 \wedge p(v_3) \wedge q(v_4) \rightarrow \mathit{false}\}$ the function symbol f can appear in the solution of each query. A set of clauses is *satisfiable* if it has a solution.

4 Algorithm

Our goal is an algorithm for computing solutions for recursion-free Horn clauses over linear arithmetic, uninterpreted functions, and queries. This section presents our solving algorithm SolveHorn(li+uif).

See Figure 3. The algorithm SolveHorn(li+uif) consists of the following main steps. First, we compute a resolution tree R on the given set of Horn clauses. Next, we take a conjunction C of the leaves of the resolution tree and attempt

$$\textsc{RInit}\ \frac{a_1 \wedge \dots \wedge a_m \rightarrow h}{\{(a_1, \dots, a_m, h)\}}$$

$$\textsc{RStep}\ \frac{\begin{array}{c} R_1 \quad \dots \quad R_n \\ q_1 \wedge \dots \wedge q_n \wedge a_1 \wedge \dots \wedge a_m \rightarrow h \end{array}}{\begin{array}{l} R_1\sigma \cup \dots \cup R_n\sigma \cup \\ \{(q_1, \dots, q_n, a_1, \dots, a_m, h)\}\sigma \end{array}}\ \begin{array}{l} \sigma = (\mathit{match}(q_1, \mathit{root}(R_1)) + \\ \quad \dots + \mathit{match}(q_n, \mathit{root}(R_n))) \end{array}$$

Fig. 4. Resolution tree inference rules RInit and RStep.

to find a proof of its unsatisfiability. If no such proof can be found, then we report that there is no solution for the given set of Horn clauses. Otherwise, we proceed with the given proof by annotating its steps. Each intermediate atom derived by proof is annotated by a function that assigns constraints to nodes of the resolution tree. Finally, the annotation of the root of the proof yields a solution for the given set of Horn clauses.

In the rest of this section we provide a detailed presentation of the main steps of SolveHorn(li+uif).

4.1 Resolution tree

We put together individual Horn clauses from $\mathcal{HC}$ by applying resolution inference. A *resolution tree* keeps the intermediate results of this computation. An edge of a *resolution tree* is a sequence of queries and atoms that is terminated by a query or *false*. Each edge consists of $n > 2$ elements. The first $n - 1$ elements represent the children nodes and the n-th element represents the parent node.

Given the set of Horn clauses $\mathcal{HC}$, we compute the corresponding resolution tree by applying the inference rules shown in Figure 4. Each rule takes as a premise a set of resolution trees and a Horn clause and infers an extended resolution tree.

The rule RInit initiates the resolution tree computation by inferring a tree from each clause that does not have any queries in its body. The atoms $a_1, \dots, a_m$ become the children of the node h. The rule RStep performs the extension of a set of trees computed so far using a Horn clause. The extension is only possible if the root nodes of the respective trees can be unified with the queries occurring in the body of the clause. This condition is formalized by the side condition requiring the existence of the most general unifier σ. The computed unifier is applied on the trees and the clause before they are combined into an extended resolution tree.

The resolution tree computation terminates since $\mathcal{HC}$ is recursion-free. Let R be the resulting tree. We consider the set of leaves of the tree, and take their conjunction $C = \bigwedge \mathit{leaves}(R)$.

For a node o of the resolution tree, we define $\mathit{insym}(o)$ to be variables and uninterpreted function symbols that occur in atoms in the leaves of the subtree of o, and let $\mathit{outsym}(o)$ be variables and uninterpreted function symbols that

$$\textsc{PHyp}\ \frac{}{t \leq 0}\ t \leq 0 \in \mathit{atoms}(C) \qquad\qquad \textsc{PComb}\ \frac{t_1 \leq 0 \quad \dots \quad t_n \leq 0}{\lambda_1 t_1 + \dots + \lambda_n t_n \leq 0}\ \lambda_1, \dots, \lambda_n > 0$$

$$\textsc{PCong}\ \frac{\begin{array}{cc} t_1 - s_1 \leq 0 & s_1 - t_1 \leq 0 \\ \vdots & \vdots \\ t_n - s_n \leq 0 & s_n - t_n \leq 0 \end{array}}{f(t_1, \dots, t_n) - f(s_1, \dots, s_n) \leq 0}\ f(t_1, \dots, t_n), f(s_1, \dots, s_n) \in \mathit{subterms}(C)$$

Fig. 5. Standard, complete proof rules PHyp, PComb, and PCong for combination of linear rational/real arithmetic and uninterpreted functions. C is the conjunction of leaves of the resolution tree R obtained from the Horn clauses $\mathcal{HC}$.

occur in the leaves outside of the subtree of o. Formally, we have

$$\begin{aligned} \mathit{insym}(o) &= \bigcup \{ \mathit{sym}(o') \mid o' \in \mathit{leaves}(\mathit{subtree}(o, R)) \}\ , \\ \mathit{outsym}(o) &= \bigcup \{ \mathit{sym}(o') \mid o' \in (\mathit{leaves}(R) \setminus \mathit{leaves}(\mathit{subtree}(o, R))) \}\ . \end{aligned}$$

The following proposition allows a transition from the clausal structure to the conjunction of atoms.

Proposition 1. *The set of Horn clauses $\mathcal{HC}$ is satisfiable if and only if the conjunction C is not satisfiable.*

The proof of Proposition 1 follows directly by applying induction over the resolution treee and relying on the definitions of RInit and RStep.

4.2 Proof tree

The algorithm SolveHorn(li+uif) relies on unsatisfiability proofs. We use a standard set of proof rules for the combination of linear rational/real arithmetic and uninterpreted functions [16]. The implementation of the corresponding proof search procedure is irrelevant for our algorithm, yet we assume that this procedure is complete and use an existing tool for this task, e.g. [4, 7].

See Figure 5 for the proof rules, which we apply to the conjunction of atoms C. The rule PHyp states that atoms appearing in C are provable from C. The rule PComb infers that a set of inequalities implies a non-negatively weighted sum thereof. The congruence rule PCong represents a form of the functionality axiom, which states that equal inputs to a function lead to equal results. We are only interested in one inequality part of this axiom. The side condition of PCong is taken from the interpolating proof rules of [16], and simplifies the proof tree annotation in a way similar to [16].

We assume that there exists a mechanism that uniquely identifies the nodes of the proof tree, even in the presence of nodes that are labeled by equal inequalities, for example by numbering them. For clarity of exposition, we omit any details of such mechanism and assume that the node label carries all necessary information.

$$\textsc{AHyp}\ \frac{}{t \leq 0\ [\ \textsc{MkHyp}(t \leq 0)\ ]}$$

$$\textsc{AComb}\ \frac{t_1 \leq 0\ [\ \Pi_1\ ] \quad \dots \quad t_n \leq 0\ [\ \Pi_n\ ]}{\lambda_1 t_1 + \dots + \lambda_n t_n \leq 0\ [\ \textsc{MkComb}(\Pi_1, \dots, \Pi_n, \lambda_1, \dots, \lambda_n)\ ]}$$

$$\textsc{ACong}\ \frac{\begin{array}{cc} t_1 - s_1 \leq 0\ [\ \Pi_1\ ] & s_1 - t_1 \leq 0\ [\ \Pi'_1\ ] \\ \vdots & \vdots \\ t_n - s_n \leq 0\ [\ \Pi_n\ ] & s_n - t_n \leq 0\ [\ \Pi'_n\ ] \end{array}}{\begin{array}{r} f(t_1, \dots, t_n) - f(s_1, \dots, s_n) \leq 0\ [\ \textsc{MkCong}(f(t_1, \dots, t_n), f(s_1, \dots, s_n), \\ \Pi_1, \dots, \Pi_n, \Pi'_1, \dots, \Pi'_n)\ ] \end{array}}$$

Fig. 6. Annotation rules. The function MkHyp, MkComb, and MkCong are shown in Figure 7.

If no proof can be found then our algorithm reports that no solution exists. Otherwise, let P be the discovered proof. We assume that P is represented by a tree where nodes are atoms and the children of a node are defined by the rules PHyp, PComb, and PCong. Furthermore, we assume that each edge is labeled by the name of the proof rule that created it.

4.3 Annotated proof tree

We construct a solution for the given Horn clauses through an iterative process, where the intermediate results are called *partial solutions*. Each partial solution is parameterized by a constraint c. A c-partial solution Π for the resolution tree R is a function from nodes of the resolution tree, $\mathit{nodes}(R)$, to constraints that satisfies the following conditions.

$$(\forall o \in \mathit{leaves}(R) :\ (\models o \rightarrow \Pi(o))) \wedge \tag{PS1}$$

$$(\forall (o^1, \dots, o^m, o) \in R : \models \Pi(o^1) \wedge \dots \wedge \Pi(o^m) \rightarrow \Pi(o)) \wedge \tag{PS2}$$

$$(\models \Pi(\mathit{false}) \rightarrow c) \wedge \tag{PS3}$$

$$(\forall o \in \mathit{nodes}(R) :\ \mathit{sym}(\Pi(o)) \subseteq (\mathit{insym}(o) \cap \mathit{outsym}(o)) \cup \mathit{sym}(c)) \tag{PS4}$$

Our annotation uses constraints of the following form, called *solution constraints*.

$$\text{solution constraints} \ni \pi ::= t \leq 0 \mid \hat{c} \wedge (\hat{c} \rightarrow \pi)$$

To simplify the presentation, we represent a solution constraint

$$C_1 \wedge (D_1 \rightarrow (\dots C_r \wedge (D_r \rightarrow p \leq 0)))$$

as a pair consisting of a corresponding sequence and a term $\langle(\,(C_1, D_1), \dots, (C_r, D_r)\,), p\rangle$. A solution constraint $p \leq 0$ is represented by $\langle [], p\rangle$.

Given the proof tree P, we annotate its nodes with partial solutions using the rules shown in Figure 6 and auxiliary functions shown in Figure 7. The rule AHyp annotates each leaf of the proof tree with the result of applying the function MkHyp. The annotation is enclosed by a pair of square brackets. The rule AComb shows how to annotate a parent node when provided with an annotation of its children in case when the parent was obtained by a non-negatively weighted sum. The parent annotation is computed by MkComb. Similarly, the rule ACong annotates parent nodes obtained by the congruence rule.

For each node of R at line 6, ACong has four cases that deal with the difficulty of solving Horn clauses over uninterpreted functions, i.e., a sub term may contain variables that are not allowed to appear in the partial solutions. The proof of theorem 3 explains how these cases avoid such variables in the partial solutions.

We annotate P and obtain an annotated proof tree A. Our algorithm Solve-Horn(li+uif) uses the annotation of the root of A to derive a solution to the Horn clauses $\mathcal{HC}$.

5 Correctness and complexity

This section presents the correctness and complexity properties of our algorithm. The corresponding proofs are in Appendix C.

The correctness of our algorithm follows from Proposition 1 and Theorems 1–3 below. First, we establish that a $(1 \leq 0)$-partial solution, which satisfies Equations (PS1)–(PS4), defines a solution for the given Horn clauses.

Theorem 1. *$(1 \leq 0)$-partial solution defines a solution of the Horn clauses.*

Now, we show that the annotations computed by the rules in Figure 6 satisfy the partial solution conditions in Equations (PS1)–(PS4). This step relies on the following inductive invariant.

Definition 1 ($t \leq 0$-annotation invariant). *Π is $t \leq 0$-annotation invariant for the resolution tree R if there exists $r \geq 0$ such that for each $o \in nodes(R)$ the following conditions hold.*

- *$\Pi(o)$ is a solution constraint such that*

$$\Pi(o) = \langle ((C_1, D_1), \dots, (C_r, D_r)), p \rangle. \tag{AI-1}$$

- *If $o \in leaves(R)$ then*

$$\left(\forall i \in 1..r : \quad \models o \wedge \bigwedge_{k=1}^{i-1} D_k \rightarrow C_i \right) \wedge \tag{AI-2a}$$

$$\left(\models o \wedge \bigwedge_{k=1}^{r} D_k \rightarrow p \leq 0 \right). \tag{AI-2b}$$

```
function MkHyp
input
  $t \le 0$ : inequality term/node in $R$
begin
  for each $o \in nodes(R)$ do
    if $t \le 0 \in leaves(subtree(o, R))$ then
      $\mathit{\Pi}(o) := \langle [], t \rangle$
    else
      $\mathit{\Pi}(o) := \langle [], 0 \rangle$
  return $\mathit{\Pi}$
end
```

```
function MkComb
input
  $\mathit{\Pi}_1, \ldots, \mathit{\Pi}_n$ : partial solutions
  $\lambda_1, \ldots, \lambda_n$ : constants
begin
  for each $o \in nodes(R)$ do
    for each $i \in 1..n$ do
      $\langle L_i, t_i \rangle := \mathit{\Pi}_i(o)$
    $L := L_1 \bullet \cdots \bullet L_n$
    $t := \lambda_1 t_1 + \cdots + \lambda_n t_n$
    $\mathit{\Pi}(o) := \langle L, t \rangle$
  return $\mathit{\Pi}$
end
```

```
function MkCong
input
  $f(t_1, \ldots, t_n)$, $f(s_1, \ldots, s_n)$ : terms
  $\mathit{\Pi}_1, \ldots, \mathit{\Pi}_n$, $\mathit{\Pi}'_1, \ldots, \mathit{\Pi}'_n$ : partial solutions
begin
  for each $o \in nodes(R)$ do
    for each $i \in 1..n$ do
      $\langle L_i, p_i \rangle := \mathit{\Pi}_i(o)$
      $\langle L'_i, p'_i \rangle := \mathit{\Pi}'_i(o)$
    $(C, D, p) :=$
        match $sym(f(t_1, \ldots, t_n)) \subseteq outsym(o)$,
              $sym(f(s_1, \ldots, s_n)) \subseteq outsym(o)$ with
        | $true, true$ -> $(\bigwedge_{i=1}^{n}(p_i \le 0 \wedge p'_i \le 0), true, 0)$
        | $true, false$ -> $(\bigwedge_{i=1}^{n} p_i + p'_i \le 0, \bigwedge_{i=1}^{n} -p_i - p'_i \le 0,$
                         $f(s_1 + p_1, \ldots, s_n + p_n) - f(s_1, \ldots, s_n))$
        | $false, true$ -> $(\bigwedge_{i=1}^{n} p_i + p'_i \le 0, \bigwedge_{i=1}^{n} -p_i - p'_i \le 0,$
                         $f(t_1, \ldots, t_n) - f(t_1 + p'_1, \ldots, t_n + p'_n))$
        | $false, false$ -> $(true, \bigwedge_{i=1}^{n}(t_i - s_i - p_i \le 0 \wedge s_i - t_i - p'_i \le 0),$
                         $f(t_1, \ldots, t_n) - f(s_1, \ldots, s_n))$
    $\mathit{\Pi}(o) := \langle L_1 \bullet \cdots \bullet L_n \bullet L'_1 \bullet \cdots \bullet L'_n \bullet (C, D), p \rangle$
  return $\mathit{\Pi}$
end
```

Fig. 7. Computation of partial solutions to annotate nodes of the proof tree, as shown in Figure 6. We use $\bullet$ to denote concatenation of sequences.

- *If* $(o^1, \dots, o^m, o) \in R$ *and* $\forall j \in 1..m : \Pi(o^j) = \langle((C_1^j, D_1^j), \dots, (C_r^j, D_r^j)), p^j\rangle$ *then*

$$\left(\forall i \in 1..r : \quad \models \left(\bigwedge_{k=1}^{i} \bigwedge_{l=1}^{m} C_k^l\right) \ \wedge \ \bigwedge_{k=1}^{i-1} D_k \to C_i\right) \wedge \tag{AI-3a}$$

$$\left(\begin{array}{l} \forall i \in 1..r \\ \forall j \in 1..m \end{array} : \ \models \begin{array}{l} \left(\bigwedge_{l \in 1..m \setminus \{j\}} C_i^l\right) \wedge \\ \left(\bigwedge_{k=1}^{i-1} \bigwedge_{l=1}^{m} C_k^l\right) \wedge \bigwedge_{k=1}^{i} D_k \to D_i^j \end{array}\right) \wedge \tag{AI-3b}$$

$$\left(\models \left(\bigwedge_{k=1}^{r} \bigwedge_{l=1}^{m} C_k^l\right) \ \wedge \ \bigwedge_{k=1}^{r} D_k \to p - p^1 - \dots - p^m \leq 0\right). \tag{AI-3c}$$

- *If* $o = \mathit{false}$ *then*

$$p = t \wedge \forall i \in 1..r : D_i = C_i = \mathit{true}. \tag{AI-4}$$

- *Conditions on symbol appearance:*

$$\mathit{sym}(\{C_1, \dots, C_r, D_1, \dots, D_r, p \leq 0\}) \subseteq \mathit{insym}(o) \wedge \tag{AI-5}$$

$$\mathit{sym}(\{C_1, \dots, C_r, D_1, \dots, D_r, t - p \leq 0\}) \subseteq \mathit{outsym}(o). \tag{AI-6}$$

The above definition act as an intermediate step. In theorem 2, we show that a $t \leq 0$-annotation invariant satisfies all the conditions for being a $t \leq 0$-partial solution.

Theorem 2. *Each $t \leq 0$-annotation invariant is a $t \leq 0$-partial solution.*

Now, we show that the presented algorithm computes the partial solutions that satisfies the invariant.

Theorem 3. *The annotation rules in Figure 6 compute annotation invariants.*

Theorem 4 (Complexity). *The application of the annotation rules from Figure 6 takes time proportional to the product of the size of the proof tree and the size of the resolution tree. The size of the resolution tree is linear in the size of the corresponding set of recursion-free, tree-like Horn clauses.*

Note that we present the complexity of our algorithm in terms of the size of the proof tree. Since the size of a resolution tree can also be exponential in the size of the set of Horn clauses, the size of a proof tree can be exponential.

6 Conclusion

We presented an algorithm for computing solutions for recursion-free Horn clauses over the combination of linear rational/real arithmetic, uninterpreted functions, and queries.

Connection to interpolation The interpolation algorithm presented in [16] is a special case for the algorithm presented in this paper. As illustrated in the introduction, an interpolation problem can be translated into solving a set of recursion free Horn clauses. The set Horn clauses resulting from an interpolation problem has only one unknown query. Therefore, the corresponding resolution tree obtain from the set of Horn clauses contains only one internal node. The partial solution of this internal node in the $(1 \leq 0)$-partial solution will be the interpolant. In this special case, we only need to track partial solutions of the internal node in the annotated proof tree. We can transform our algorithm for this case such that nodes of the proof tree are annotated with a formula corresponding to the partial solution of this internal node. The resulting algorithm will be the algorithm presented in [16].

Our algorithm can be directly applied to support abstraction and refinement tasks for the verification of programs with procedures, threads and higher order functions.

7 Acknowledgment

Ashutosh Gupta was supported in part by the DFG Graduiertenkolleg 1480 (PUMA), FWF NFN Grant No S11407-N23 (RiSE), and the ERC Advanced Grant QUAREM.

References

1. D. Beyer, A. Cimatti, A. Griggio, M. E. Keremoglu, and R. Sebastiani. Software model checking via large-block encoding. In *FMCAD*, 2009.
2. D. Beyer, D. Zufferey, and R. Majumdar. CSIsat: Interpolation for LA+EUF. In *CAV*, 2008.
3. A. Brillout, D. Kroening, P. Rümmer, and T. Wahl. An interpolating sequent calculus for quantifier-free Presburger arithmetic. In *Proceedings of IJCAR*, LNCS, pages 384–399. Springer, 2010.
4. R. Bruttomesso, A. Cimatti, A. Franzén, A. Griggio, and R. Sebastiani. The MathSAT 4SMT solver. In *CAV*, 2008.
5. A. Cimatti, A. Griggio, and R. Sebastiani. Interpolant generation for UTVPI. In *CADE*, 2009.
6. A. Cimatti, A. Griggio, and R. Sebastiani. Efficient generation of Craig interpolants in satisfiability modulo theories. *ACM Trans. Comput. Logic*, 12, November 2010.
7. L. M. de Moura and N. Bjørner. Z3: An efficient SMT solver. In *TACAS*, 2008.
8. A. Goel, S. Krstic, and C. Tinelli. Ground interpolation for combined theories. In *CADE*, 2009.
9. A. Gupta, C. Popeea, and A. Rybalchenko. Predicate abstraction and refinement for verifying multi-threaded programs. In *POPL*, 2011.
10. M. Heizmann, J. Hoenicke, and A. Podelski. Nested interpolants. In *POPL*, 2010.
11. T. A. Henzinger, R. Jhala, R. Majumdar, and K. L. McMillan. Abstractions from proofs. In *POPL*, 2004.

12. H. Jain, E. M. Clarke, and O. Grumberg. Efficient Craig interpolation for linear Diophantine (dis)equations and linear modular equations. *Formal Methods in System Design*, pages 6–39, 2009.
13. R. Jhala and R. Majumdar. Counterexample refinement for functional programs. available from http://www.cs.ucla.edu/˜rupak/Papers/CEGARFunctional.ps, 2009.
14. R. Jhala and K. L. McMillan. A practical and complete approach to predicate refinement. In *TACAS*, 2006.
15. D. Kroening, J. Leroux, and P. Rümmer. Interpolating quantifier-free Presburger arithmetic. In *Proceedings of LPAR*, LNCS, pages 489–503. Springer, 2010.
16. K. L. McMillan. An interpolating theorem prover. *Theor. Comput. Sci.*, 345(1):101–121, 2005.
17. K. L. McMillan. Lazy abstraction with interpolants. In *CAV*, pages 123–136, 2006.
18. A. Rybalchenko and V. Sofronie-Stokkermans. Constraint solving for interpolation. In *VMCAI*, 2007.
19. T. Terauchi. Dependent types from counterexamples. In *POPL*, 2010.
20. H. Unno and N. Kobayashi. Dependent type inference with interpolants. In *PPDP*, 2009.
21. G. Yorsh and M. Musuvathi. A combination method for generating interpolants. In *CADE*, pages 353–368, 2005.

```
// take_lock : multi-thread program
int f[N];
int p, q;

// Thread1(int c)                  // Thread2(int d)
a1:  assume(p <= c <= q);          b1:  assume(q <= d <= p);
a2:  take_lock(f, c);              b2:  take_lock(f, d);
a3:  // critical                   b3:  // critical
```

(a)

```
int p, q;

int main() {                      void foo() {
m1:   int c = ..;                 n1:   int d = ..;
m2:   assume(p <= c <= q);        n2:   assume(q <= d <= p);
m3:   if (f(c) == 1) { foo(); }   n3:   if (f(d) == 0)
m4:   assert(false);              n4:     return;
}                                 n5:   ...
                                  }
```

(b)

Fig. 8. Two example programs take_lock and main. (a) take_lock illustrates how Horn clauses can represent an abstraction refinement task in presence thread interaction. (b) main illustrates a formalization the abstraction refinement for programs with procedures using Horn clauses.

A Refinement using Horn clauses

We present examples of Horn clauses obtained during the abstraction refinement step when verifying multi-threaded programs and programs with procedures.

Abstraction refinement for multi-threaded programs See Figure 8(a) for a program take_lock that consists of two threads. These threads attempt to access a critical section and synchronize their accesses using a lock stored in the global array f. The two threads receive the identifier of the lock as an integer argument c for the first thread and d for the second thread. The assume statements at labels a1 and b1 ensure that the two integer indices, c and d , are equal. The calls at labels a2 and b2 ensure that the two threads cannot both enter the critical section, i.e., the assertion $\neg(\mathtt{pc}_1 = \mathtt{a3} \wedge \mathtt{pc}_2 = \mathtt{b3})$ holds for all executions of the program. We write $\mathcal{V} = \{\mathtt{f}, \mathtt{p}, \mathtt{q}, \mathtt{c}, \mathtt{d}, \mathtt{pc}_1, \mathtt{pc}_2\}$ for the set of all program variables, where $\mathtt{pc}_1$ and $\mathtt{pc}_2$ are local program counter variables of the first and second thread, respectively. Let $G = \{\mathtt{p}, \mathtt{q}\}$ be the set of global program variables.

To verify the program take_lock, the method described in [9] performs abstract reachability computations for each thread considering both local thread transitions and environment transitions that capture updates of program state done by the other thread. Let us assume that the abstract reachability procedure finds a spurious error state following an interleaving of the statements from the two threads represented by two assertions ρ_1 and ρ_2 . In this case, the results computed by the abstract reachability are an abstract state s and an environment transition e such that:

$$s = \dot{\alpha}(post(\rho_1, true)), \qquad\qquad e = \ddot{\alpha}(\rho_2),$$

where *post* denotes the successor function and $\dot{\alpha}$ and $\ddot{\alpha}$ denote abstraction functions for over-approximation of sets of states and sets of pairs of states, respectively. The constraint ρ_1 represents program statements at location `a1` and `a2` from the first thread, while ρ_2 represents the program statements at location `b1` and `b2` from the second thread. Both transitions are over unprimed and primed program variables. We only show the critical part of these constraints that is relevant to the infeasibility of the interleaving:

$$\begin{aligned}\rho_1 &= (\mathtt{p} \leq \mathtt{c} \wedge \mathtt{c} \leq \mathtt{q} \wedge \mathtt{f(c)} = 1 \wedge \mathtt{p} = \mathtt{p}' \wedge \mathtt{q} = \mathtt{q}' \wedge \mathtt{c} = \mathtt{c}')\ , \\ \rho_2 &= (\mathtt{q} \leq \mathtt{d} \wedge \mathtt{d} \leq \mathtt{p} \wedge \mathtt{f(d)} = 0 \wedge \mathtt{p} = \mathtt{p}' \wedge \mathtt{q} = \mathtt{q}' \wedge \mathtt{d} = \mathtt{d}')\ .\end{aligned}$$

We model the fact that the first thread acquires the lock indexed by `c` using $\mathtt{f(c)} = 1$. The constraint $\mathtt{f(d)} = 0$ from ρ_2 represents the requirement that the lock indexed by `d` must be released in order to complete the call to `take_lock` at program location `b2` .

Following the reachability of an abstract state that intersects the error states $(\mathtt{pc}_1 = \mathtt{a3} \wedge \mathtt{pc}_2 = \mathtt{b3})$, abstraction refinement constraints are derived. We obtain a set of Horn clauses where the unknown query $S(\mathcal{V})$ represents the refined abstract state s and $E(G, G')$ represents the refined environment transition e :

$$\mathcal{HC}_{\mathtt{take\_lock}} = \{\ \rho_1 \to S(\mathcal{V}'),\ \rho_2 \to E(G, G'),\ S(\mathcal{V}) \wedge E(G, G') \to \mathit{false}\ \}\ .$$

The third clause requires that the intersection of the set of states $S(\mathcal{V})$ and the environment transition is empty. In general, solutions for the refined environment transitions can be expressed in terms of the whole set of program variables $\mathcal{V}$. However, an efficient verification procedure relies on using thread-modular solutions (in terms of only global variables) whenever they exist [9]. In our example, preference towards thread-modular solutions is declared using $E(G, G')$ instead of $E(\mathcal{V}, \mathcal{V}')$. Each Horn clause is implicitly universally quantified over the variables that appear in the clause, i.e., $\mathcal{V}$ and $\mathcal{V}'$. The set of clauses $\mathcal{HC}_{\mathtt{take\_lock}}$ is satisfiable if and only if the abstraction can be refined to exclude the spurious interleaving. Note that the solving procedure for Horn clauses proposed in [9] is not applicable here due to the presence of uninterpreted function symbols.

Abstraction refinement for programs with procedures We use the second program in Figure 8(b) to illustrate refinement constraints for proving the infeasibility of an interprocedural path that is expressed using Horn clauses. This program has same set of program variables $\mathcal{V}$ and program global variables G as `take_lock`.

The procedure `main` establishes at line `m2` that the value of the local variable `c` is in a required range of integer values. At line `m3` , `foo` is called if an unspecified function `f` returns the integer value 1. Due to the conditions at lines `n2` and `n3` , the procedure `foo` cannot return at line `n4` from the calling context at line `m3`. However, due to over-approximation, an abstract reachability computation may result in a summary for the `foo` procedure that is too imprecise. Assuming that the constraint ρ_1 represents the calling context of `foo` at line `m3`.

$$\rho_1 = (\mathtt{p} \leq \mathtt{c} \wedge \mathtt{c} \leq \mathtt{q} \wedge \mathtt{f}(\mathtt{c}) = 1 \wedge \mathtt{p} = \mathtt{p}' \wedge \mathtt{q} = \mathtt{q}' \wedge \mathtt{c} = \mathtt{c}') \ ,$$

An abstract state s is computed as follows:

$$s = \dot{\alpha}(\mathit{post}(\rho_1, \mathit{true})) \ .$$

Further, using a transition abstraction function $\ddot{\alpha}$, a summary transition e is computed for the `foo` procedure:

$$\begin{aligned} \rho_2 &= (\mathtt{q} \leq \mathtt{d} \wedge \mathtt{d} \leq \mathtt{p} \wedge \mathtt{f}(\mathtt{d}) = 0 \wedge \mathtt{p} = \mathtt{p}' \wedge \mathtt{q} = \mathtt{q}') \ , \\ e \ &= \ddot{\alpha}(\rho_2) \ . \end{aligned}$$

In order to show the infeasibility of the interprocedural path denoted by the sequence of program labels $\mathtt{m1}, \mathtt{m2}, \mathtt{m3}, \mathtt{n1}, \mathtt{n2}, \mathtt{n3}, \mathtt{n4}, \mathtt{m4}$, abstraction refinement constraints are expressed by the following Horn clauses:

$$\mathcal{HC}_{\mathtt{foo}} = \{ \ \rho_1 \rightarrow S(\mathcal{V}'), \ \rho_2 \rightarrow E(G, G'), \ S(\mathcal{V}) \wedge E(G, G') \rightarrow \mathit{false} \ \} \ .$$

We require that the solution for the procedure summary refers only to global variables `p` and `q`, but not to the local variable `d`. Therefore, $E(G, G')$ refers to only global variables.

We constructed the above examples so that $\mathcal{HC}_{\mathtt{take\_lock}} = \mathcal{HC}_{\mathtt{foo}}$. We further simplify the Horn clauses and drop the variables from the queries that do not contribute to the satisfiability of the set of Horn clauses. After the simplification, we obtain

$$\mathcal{HC} = \{ \ \rho_1 \rightarrow S(\mathtt{p}, \mathtt{q}, \mathtt{c}), \ \rho_2 \rightarrow E(\mathtt{p}, \mathtt{q}), \ S(\mathtt{p}, \mathtt{q}, \mathtt{c}) \wedge E(\mathtt{p}, \mathtt{q}) \rightarrow \mathit{false} \ \} \ .$$

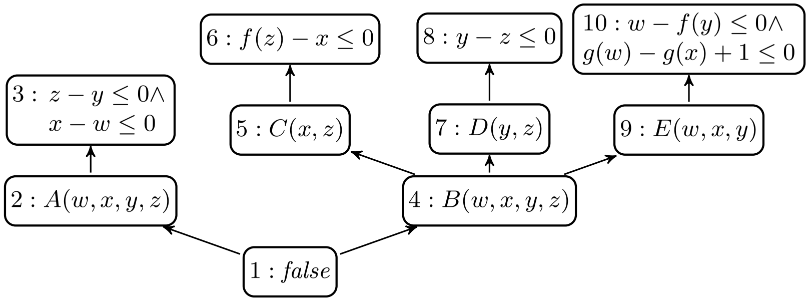
Our algorithm computes a solution to this set of Horn clauses as illustrated in Section 2.

B Example execution of SolveHorn(li+uif)

This section presents an execution example for our algorithm SolveHorn(li+uif). We apply it on the set of Horn clauses shown in Figure 9(a). The corresponding resolution tree is given in in Figure 9(b) and the proof tree for the conjunction of leaves in the resolution tree is shown in Figure 9(c). The annotated tree is shown in Figure 9(d), where the partial solutions $\mathit{\Pi}_1, \ldots, \mathit{\Pi}_7$ are presented below.

$$\begin{aligned}
B(w,x,y,z) \wedge A(w,x,y,z) &\rightarrow \mathit{false}\\
z-y \leq 0 \wedge x-w \leq 0 &\rightarrow A(w,x,y,z)\\
C(x,z) \wedge D(y,z) \wedge E(w,x,y) &\rightarrow B(w,x,y,z)\\
f(z)-x \leq 0 &\rightarrow C(x,z)\\
y-z \leq 0 &\rightarrow D(y,z)\\
w-f(y) \leq 0 \wedge g(w)-g(x)+1 \leq 0 &\rightarrow E(w,x,y)
\end{aligned}$$

(a)



(b)

$$\textsc{PComb}\ \dfrac{\textsc{PCong}\ \dfrac{x-w \leq 0 \quad \textsc{PComb}\ \dfrac{\textsc{PCong}\ \dfrac{y-z \leq 0 \quad z-y \leq 0}{f(y)-f(z) \leq 0} \quad w-f(y) \leq 0 \quad f(z)-x \leq 0}{w-x \leq 0}}{g(x)-g(w) \leq 0} \quad g(w)-g(x)+1 \leq 0}{1 \leq 0}$$

(c)

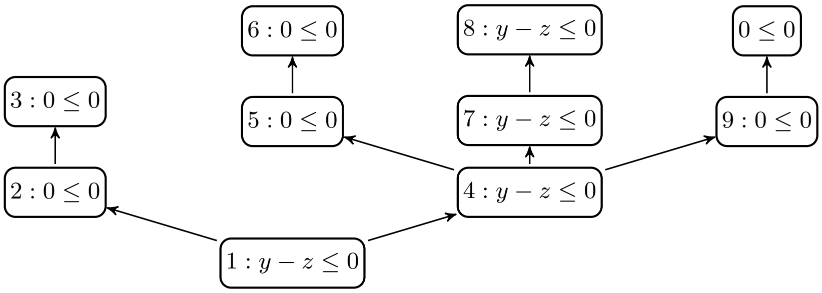
$$\textsc{AComb}\ \dfrac{\textsc{ACong}\ \dfrac{x-w \leq 0[\mathit{\Pi}_5] \quad \textsc{AComb}\ \dfrac{\textsc{ACong}\ \dfrac{y-z \leq 0[\mathit{\Pi}_1] \quad z-y \leq 0[\mathit{\Pi}_2]}{f(y)-f(z) \leq 0[\mathit{\Pi}_3]} \quad w-f(y) \leq 0[\dots] \quad f(z)-x \leq 0[\dots]}{w-x \leq 0[\mathit{\Pi}_4]}}{g(x)-g(w) \leq 0[\mathit{\Pi}_6]} \quad g(w)-g(x)+1 \leq 0[\dots]}{1 \leq 0[\mathit{\Pi}_7]}$$

(d)

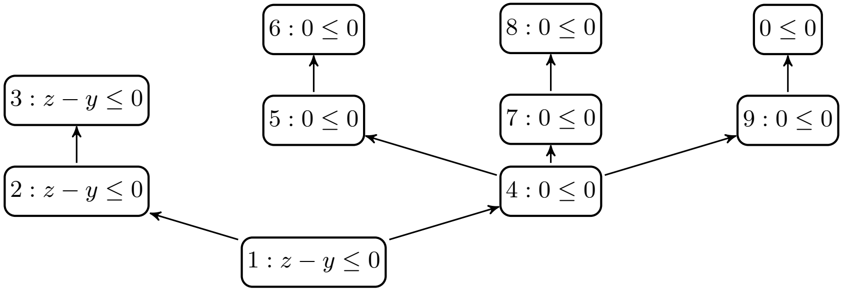
Fig. 9. (a) An example of Horn clauses (b) Resolution tree of the Horn clauses (c) Proof-tree of unsatisfiability (d) Annotate proof-tree of unsatisfiability

$A(w,x,y,z)$	$z-y\leq 0\wedge(0\leq 0\rightarrow x-w\leq 0\wedge(0\leq 0\rightarrow 0\leq 0))$
$B(w,x,y,z)$	$0\leq 0\wedge(z-y\leq 0\rightarrow 0\leq 0\wedge(x-w\leq 0\rightarrow 1\leq 0))$
$C(x,z)$	$0\leq 0\wedge(0\leq 0\rightarrow f(z)-x\leq 0\wedge(0\leq 0\rightarrow 0\leq 0))$
$D(y,z)$	$y-z\leq 0\wedge(0\leq 0\rightarrow 0\leq 0\wedge(0\leq 0\rightarrow 0\leq 0))$
$E(w,x,y)$	$0\leq 0\wedge(0\leq 0\rightarrow 0\leq 0\wedge(f(y)-x\leq 0\wedge x-w\leq 0\rightarrow 1\leq 0))$

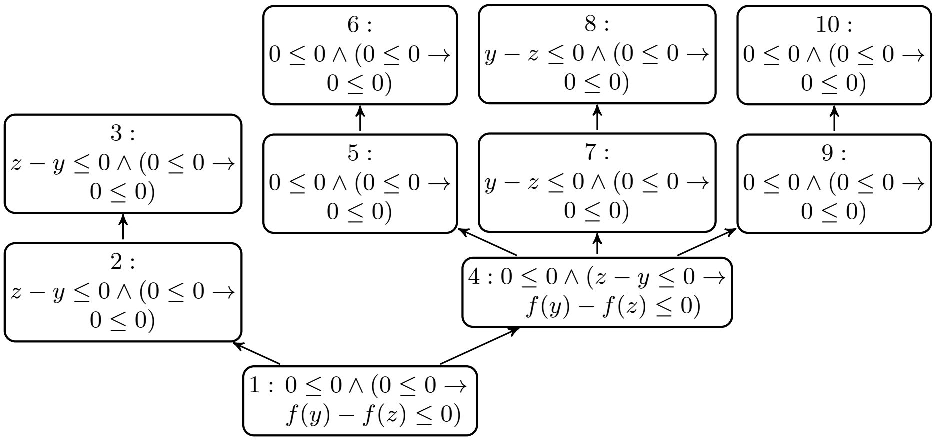
Table 1. Solution for Horn clauses in Figure 9(a).



Π_1

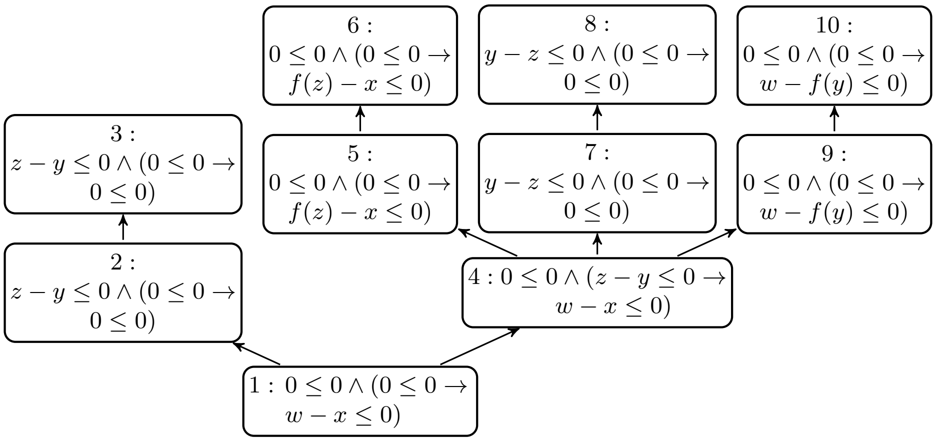


Π_2

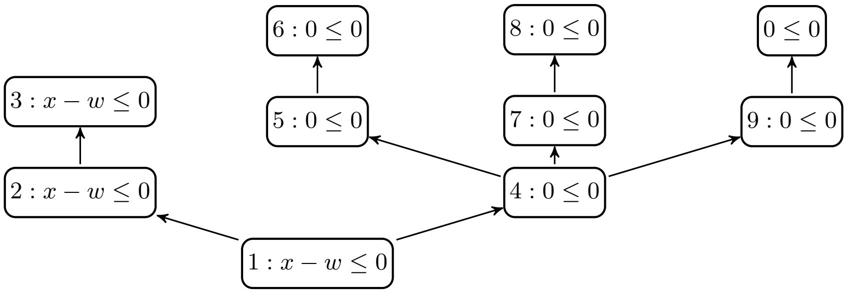


$6:$ $0\leq 0\wedge(0\leq 0\rightarrow$ $0\leq 0)$
$8:$ $y-z\leq 0\wedge(0\leq 0\rightarrow$ $0\leq 0)$
$10:$ $0\leq 0\wedge(0\leq 0\rightarrow$ $0\leq 0)$
$3:$ $z-y\leq 0\wedge(0\leq 0\rightarrow$ $0\leq 0)$
$5:$ $0\leq 0\wedge(0\leq 0\rightarrow$ $0\leq 0)$
$7:$ $y-z\leq 0\wedge(0\leq 0\rightarrow$ $0\leq 0)$
$9:$ $0\leq 0\wedge(0\leq 0\rightarrow$ $0\leq 0)$
$2:$ $z-y\leq 0\wedge(0\leq 0\rightarrow$ $0\leq 0)$
$4:0\leq 0\wedge(z-y\leq 0\rightarrow$ $f(y)-f(z)\leq 0)$
$1:\ 0\leq 0\wedge(0\leq 0\rightarrow$ $f(y)-f(z)\leq 0)$

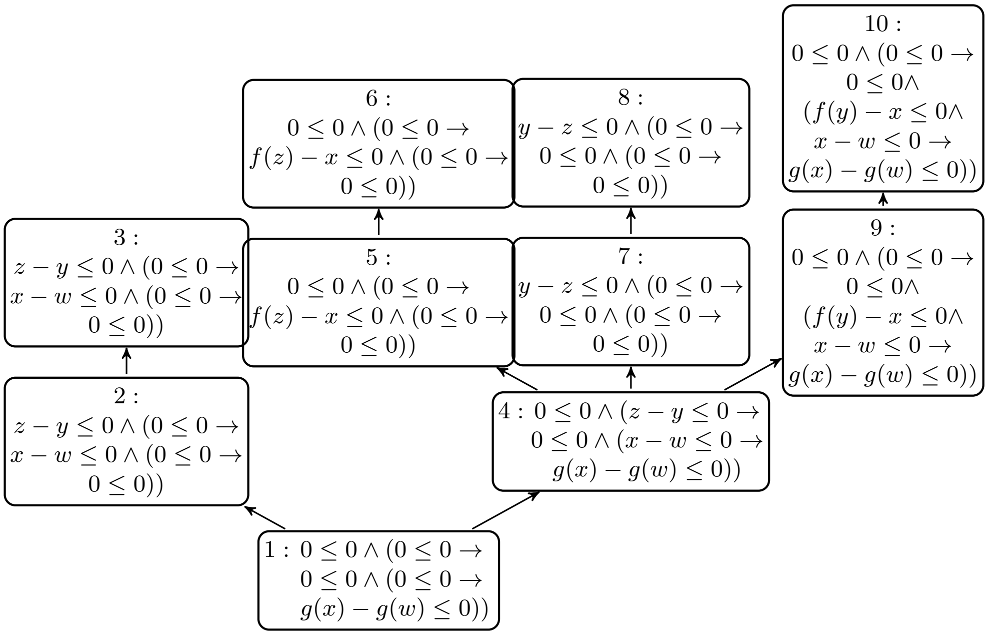
Π_3


$6: 0 \le 0 \wedge (0 \le 0 \to f(z) - x \le 0)$
$8: y - z \le 0 \wedge (0 \le 0 \to 0 \le 0)$
$10: 0 \le 0 \wedge (0 \le 0 \to w - f(y) \le 0)$
$3: z - y \le 0 \wedge (0 \le 0 \to 0 \le 0)$
$5: 0 \le 0 \wedge (0 \le 0 \to f(z) - x \le 0)$
$7: y - z \le 0 \wedge (0 \le 0 \to 0 \le 0)$
$9: 0 \le 0 \wedge (0 \le 0 \to w - f(y) \le 0)$
$2: z - y \le 0 \wedge (0 \le 0 \to 0 \le 0)$
$4: 0 \le 0 \wedge (z - y \le 0 \to w - x \le 0)$
$1: 0 \le 0 \wedge (0 \le 0 \to w - x \le 0)$

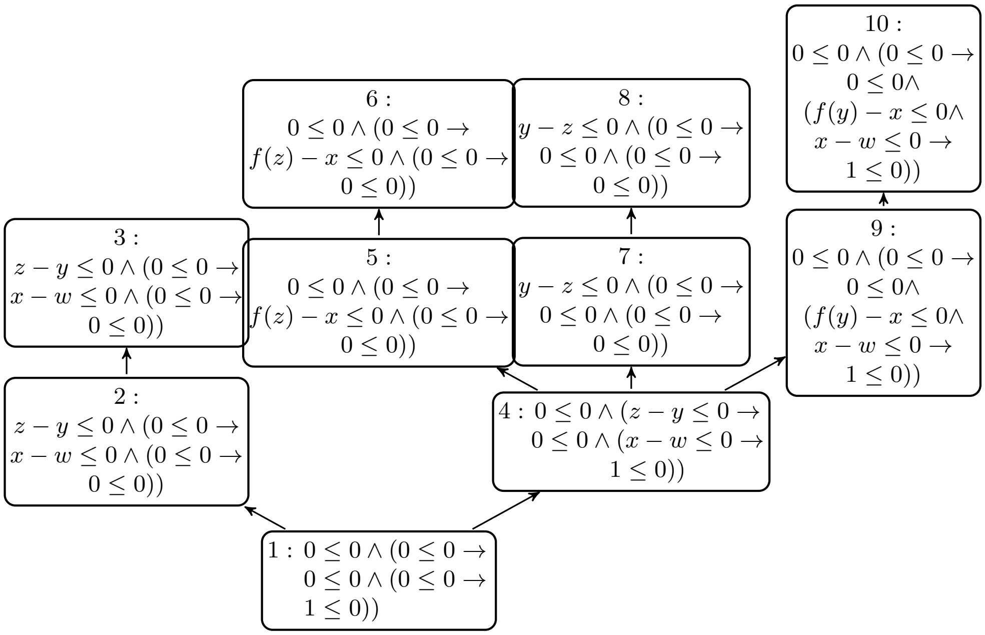
Π_4


$6: 0 \le 0$
$8: 0 \le 0$
$0 \le 0$
$3: x - w \le 0$
$5: 0 \le 0$
$7: 0 \le 0$
$9: 0 \le 0$
$2: x - w \le 0$
$4: 0 \le 0$
$1: x - w \le 0$

Π_5


$10: 0 \le 0 \wedge (0 \le 0 \to 0 \le 0 \wedge (f(y) - x \le 0 \wedge x - w \le 0 \to g(x) - g(w) \le 0))$
$6: 0 \le 0 \wedge (0 \le 0 \to f(z) - x \le 0 \wedge (0 \le 0 \to 0 \le 0))$
$8: y - z \le 0 \wedge (0 \le 0 \to 0 \le 0 \wedge (0 \le 0 \to 0 \le 0))$
$3: z - y \le 0 \wedge (0 \le 0 \to x - w \le 0 \wedge (0 \le 0 \to 0 \le 0))$
$5: 0 \le 0 \wedge (0 \le 0 \to f(z) - x \le 0 \wedge (0 \le 0 \to 0 \le 0))$
$7: y - z \le 0 \wedge (0 \le 0 \to 0 \le 0 \wedge (0 \le 0 \to 0 \le 0))$
$9: 0 \le 0 \wedge (0 \le 0 \to 0 \le 0 \wedge (f(y) - x \le 0 \wedge x - w \le 0 \to g(x) - g(w) \le 0))$
$2: z - y \le 0 \wedge (0 \le 0 \to x - w \le 0 \wedge (0 \le 0 \to 0 \le 0))$
$4: 0 \le 0 \wedge (z - y \le 0 \to 0 \le 0 \wedge (x - w \le 0 \to g(x) - g(w) \le 0))$
$1: 0 \le 0 \wedge (0 \le 0 \to 0 \le 0 \wedge (0 \le 0 \to g(x) - g(w) \le 0))$

Π_6


$1: 0 \leq 0 \wedge (0 \leq 0 \rightarrow 0 \leq 0 \wedge (0 \leq 0 \rightarrow 1 \leq 0))$
$2: z - y \leq 0 \wedge (0 \leq 0 \rightarrow x - w \leq 0 \wedge (0 \leq 0 \rightarrow 0 \leq 0))$
$3: z - y \leq 0 \wedge (0 \leq 0 \rightarrow x - w \leq 0 \wedge (0 \leq 0 \rightarrow 0 \leq 0))$
$4: 0 \leq 0 \wedge (z - y \leq 0 \rightarrow 0 \leq 0 \wedge (x - w \leq 0 \rightarrow 1 \leq 0))$
$5: 0 \leq 0 \wedge (0 \leq 0 \rightarrow f(z) - x \leq 0 \wedge (0 \leq 0 \rightarrow 0 \leq 0))$
$6: 0 \leq 0 \wedge (0 \leq 0 \rightarrow f(z) - x \leq 0 \wedge (0 \leq 0 \rightarrow 0 \leq 0))$
$7: y - z \leq 0 \wedge (0 \leq 0 \rightarrow 0 \leq 0 \wedge (0 \leq 0 \rightarrow 0 \leq 0))$
$8: y - z \leq 0 \wedge (0 \leq 0 \rightarrow 0 \leq 0 \wedge (0 \leq 0 \rightarrow 0 \leq 0))$
$9: 0 \leq 0 \wedge (0 \leq 0 \rightarrow 0 \leq 0 \wedge (f(y) - x \leq 0 \wedge x - w \leq 0 \rightarrow 1 \leq 0))$
$10: 0 \leq 0 \wedge (0 \leq 0 \rightarrow 0 \leq 0 \wedge (f(y) - x \leq 0 \wedge x - w \leq 0 \rightarrow 1 \leq 0))$

Π_7

C Proofs

Proof (Proof for Theorem 1). Due to (PS1)–(PS3), a $1 \leq 0$-partial solution satisfies the Horn clauses. Since, $sym(1 \leq 0)$ is empty, (PS4) is equivalent to the restriction on symbols appearance for a solution of the Horn clauses.

Proof (Proof for Theorem 2). Let Π be a $t \leq 0$-annotation invariant and let $o \in nodes(R)$. Then, $\Pi(o)$ satisfies (AI-1)–(AI-6). We will prove that Π is $t \leq 0$-partial solution by showing (PS3),(PS4), (PS1), and (PS2).

(PS3)*:* If $o = false$ then (AI-4) directly implies (PS3).

(PS4)*:* Due to (AI-5), $sym(\Pi(o)) \subseteq insym(o)$. Due to (AI-6), $sym(t - p \leq 0) \subseteq outsym(o)$. Now, let us assume there is a subterm s in p such that $sym(s) \not\subseteq outsym(o) \cup sym(t \leq 0)$ and s does not have $+$ as the outermost function symbol. Therefore, s must be a subterm of $t - p$. Therefore, $sym(t - p \leq 0) \not\subseteq outsym(o)$. Hence, we obtain a contradiction. Therefore, $sym(p \leq 0) \subseteq outsym(o) \cup sym(t \leq 0)$. So we deduce $sym(\Pi(o)) \subseteq insym(o) \cap (outsym(o) \cup sym(t \leq 0))$. Hence, (PS4) holds.

(PS1)*:* Let $o \in leaves(R)$. First, we will prove the following validity for all $i \in 0..r$ by induction.

$$\models o \wedge \bigwedge_{k=1}^{r-i} D_k \rightarrow \langle (\, (C_{r-i+1}, D_{r-i+1}), \ldots, (C_r, D_r) \,), p \rangle$$

Base case: $i = 0$. (AI-2b) implies $\models o \wedge \bigwedge_{k=1}^{r} D_k \rightarrow \langle (), p \rangle$.
Induction step: $r > i > 0$. By induction hypothesis, we have

$$\models o \wedge \bigwedge_{k=1}^{r-i} D_k \rightarrow \langle (\, (C_{r-i+1}, D_{r-i+1}), \ldots, (C_r, D_r) \,), p \rangle.$$

By separating D_{r-i}, we obtain

$$\models o \wedge \bigwedge_{k=1}^{r-i-1} D_k \rightarrow (D_{r-i} \rightarrow \langle (\ (C_{r-i+1}, D_{r-i+1}), \ldots, (C_r, D_r)\ ), p\rangle).$$

Due to the (AI-2a), $\models o \wedge \bigwedge_{k=1}^{r-i-1} D_k \rightarrow C_{r-i}$. Therefore,

$$\models o \wedge \bigwedge_{k=1}^{r-i} D_k \rightarrow (C_{r-i} \wedge (D_{r-i} \rightarrow \langle (\ (C_{r-i+1}, D_{r-i+1}), \ldots, (C_r, D_r)\ ), p\rangle)),$$

which is equivalent to

$$\models o \wedge \bigwedge_{k=1}^{r-i-1} D_k \rightarrow \langle (\ (C_{r-i}, D_{r-i}), \ldots, (C_r, D_r)\ ), p\rangle.$$

From our proved validity, we obtain for $i = r$:

$$\models o \rightarrow \langle (\ (C_{r-i}, D_{r-i}), \ldots, (C_r, D_r)\ ), p\rangle.$$

Hence, (PS1) holds.

(PS2)*:* Let $(o^1, \ldots, o^m, o) \in R$. First, we will prove the following validity for all $i \in 0..r$ by induction.

$$\models \bigwedge_{j=1}^{m} \langle (\ (C_{r-i+1}^j, D_{r-i+1}^j), \ldots, (C_r^j, D_r^j)\ ), p^j\rangle \wedge \\ \left(\bigwedge_{k=1}^{r-i} \bigwedge_{l=1}^{m} C_k^l\right) \wedge \bigwedge_{k=1}^{r-i} D_k \rightarrow \langle (\ (C_{r-i+1}, D_{r-i+1}), \ldots, (C_r, D_r)\ ), p\rangle$$

Base case: $i = 0$. (AI-3c) implies

$$\models \bigwedge_{j=1}^{m} \langle (), p^j\rangle \ \wedge\ \left(\bigwedge_{k=1}^{r} \bigwedge_{l=1}^{m} C_k^l\right) \wedge \bigwedge_{k=1}^{r} D_k \rightarrow \langle (), p\rangle,$$

which is the base case.
Induction step: $r > i > 0$. Consider the left hand side of induction step $i + 1$,

$$\bigwedge_{j=1}^{m} \langle (\ (C_{r-i}^j, D_{r-i}^j), \ldots, (C_r^j, D_r^j)\ ), p^j\rangle \ \wedge\ \left(\bigwedge_{k=1}^{r-i-1} \bigwedge_{l=1}^{m} C_k^l\right) \wedge \bigwedge_{k=1}^{r-i-1} D_k.$$

By unfolding definition of a solution constraint once,

$$\bigwedge_{j=1}^{m} (D_{r-i}^j \rightarrow \langle (\ (C_{r-i+1}^j, D_{r-i+1}^j), \ldots, (C_r^j, D_r^j)\ ), p^j\rangle) \wedge \\ \left(\bigwedge_{k=1}^{r-i} \bigwedge_{l=1}^{m} C_k^l\right) \wedge \bigwedge_{k=1}^{r-i-1} D_k.$$

Due to (AI-3a), the above formula implies C_{r-i}.
Now lets take conjunction of the above formula and D_{r-i},

$$\bigwedge_{j=1}^{m} (D_{r-i}^j \rightarrow \langle (\ (C_{r-i+1}^j, D_{r-i+1}^j), \ldots, (C_r^j, D_r^j)\ ), p^j\rangle) \wedge \\ \left(\bigwedge_{k=1}^{r-i} \bigwedge_{l=1}^{m} C_k^l\right) \wedge \bigwedge_{k=1}^{r-i} D_k.$$

Due to (AI-3b), the above formula implies

$$\bigwedge_{j=1}^{m} (D_{r-i}^j \rightarrow \langle (\ (C_{r-i+1}^j, D_{r-i+1}^j), \ldots, (C_r^j, D_r^j)\ ), p^j\rangle) \wedge \\ \left(\bigwedge_{k=1}^{r-i} \bigwedge_{l=1}^{m} C_k^l\right) \wedge \bigwedge_{k=1}^{r-i} D_k \wedge \bigwedge_{j=1}^{m} D_{r-i}^j.$$

Therefore,

$$\bigwedge_{j=1}^{m} \langle (\ (C_{r-i+1}^j, D_{r-i+1}^j), \ldots, (C_r^j, D_r^j)\ ), p^j\rangle \wedge \\ \left(\bigwedge_{k=1}^{r-i} \bigwedge_{l=1}^{m} C_k^l\right) \wedge \bigwedge_{k=1}^{r-i} D_k.$$

Due to the induction hypothesis, the above formula implies

$$\langle(\,(C_{r-i+1}, D_{r-i+1}), \ldots, (C_r, D_r)\,), p\rangle.$$

So, we have proven that the left hand side of the induction step at $i+1$ implies

$$C_{r-i} \wedge (D_{r-i} \rightarrow \langle(\,(C_{r-i+1}, D_{r-i+1}), \ldots, (C_r, D_r)\,), p\rangle),$$

which is the right hand side of the induction step at $i+1$.

From our proved validity, we obtain for $i = r$,

$$\models \bigwedge_{j=1}^{m} \langle(\,(C_1^j, D_1^j), \ldots, (C_r^j, D_r^j)\,), p^j\rangle \rightarrow \langle(\,(C_1, D_1), \ldots, (C_r, D_r)\,), p\rangle.$$

Hence, (PS2) holds. □

The following three lemmas will be used to prove Theorem 3.

Lemma 1. *Let Π be $t \leq 0$-annotation invariant and let Π' be $t' \leq 0$-annotation invariant. Let Π_1 and Π_1 be a function from R to constraints such that*

$$\forall o \in \mathit{nodes}(R) : \Pi(o) = \langle L, p\rangle \wedge \Pi'(o) = \langle L', \_\rangle \rightarrow \Pi_1(o) = \langle L \bullet L', p\rangle$$

and

$$\forall o \in \mathit{nodes}(R) : \Pi(o) = \langle L, p\rangle \wedge \Pi'(o) = \langle L', \_\rangle \rightarrow \Pi_2(o) = \langle L' \bullet L, p\rangle.$$

Π_1 and Π_2 are $t \leq 0$-annotation invariants.

Proof. We will only deal with Π_1. The proof for Π_2 is similar.

Let $o \in \mathit{nodes}(R)$, $\Pi(o) = \langle(\,(C_1, D_1), \ldots, (C_n, D_n)\,), p\rangle$, and $\Pi'(o) = \langle(\,(C_{n+1}, D_{n+1}), \ldots, (C_{n+m}, D_{n+m})\,), \_\rangle$. Then, $\Pi_1(o) = \langle(\,(C_1, D_1), \ldots, (C_{n+m}, D_{n+m})\,), p\rangle$. $\Pi_1(o)$ maps to a solution constraint that has prefix sequence of length $n+m$. Therefore, (AI-1) holds. (AI-2a)–(AI-3c) for $\Pi_1(o)$ are satisfied since these conditions have stronger left hand sides compare to the corresponding conditions for $\Pi(o)$ and $\Pi'(o)$. (AI-4)–(AI-6) are directly holds. □

The above lemma can be applied multiple times on a $t \leq 0$-annotation invariant satisfying Π to show that a prefix extension in the above way does not violate $t \leq 0$-annotation invariant.

Lemma 2. *Let $(o^1, \ldots, o^m, o) \in R$. If $\mathit{sym}(f(t_1, \ldots, t_n)) \subseteq \mathit{outsym}(o)$ then $\forall l \in 1..m :\ \mathit{sym}(f(t_1, \ldots, t_n)) \subseteq \mathit{outsym}(o^l)$.*

The proof of above lemma is left for the reader to verify.

Lemma 3. *Let $(o^1, \ldots, o^m, o) \in R$. If $\mathit{sym}(f(t_1, \ldots, t_n)) \not\subseteq \mathit{outsym}(o)$ then either of the following cases is true.*

1. $\forall l \in 1..m :\ \mathit{sym}(f(t_1, \ldots, t_n)) \subseteq \mathit{outsym}(o^l)$
2. $\exists j :\ \mathit{sym}(f(t_1, \ldots, t_n)) \not\subseteq \mathit{outsym}(o^j) \wedge$
   $\forall l \in 1..m \setminus \{j\} :\ \mathit{sym}(f(t_1, \ldots, t_n)) \subseteq \mathit{outsym}(o^l)$.

Proof. Since PComb does not allow introduction of terms that are not present in the input atoms, if $\mathit{sym}(f(t_1, \ldots, t_n)) \not\subseteq \mathit{outsym}(o)$ then $\mathit{sym}(f(t_1, \ldots, t_n)) \subseteq \mathit{insym}(o)$ and there exist at least one child node o^j such that $\mathit{sym}(f(t_1, \ldots, t_n)) \subseteq \mathit{insym}(o^j)$.

If there are at least two children o^{j_1} and o^{j_2} such that $\mathit{sym}(f(t_1, \ldots, t_n)) \subseteq \mathit{insym}(o^{j_1})$ and $\mathit{sym}(f(t_1, \ldots, t_n)) \subseteq \mathit{insym}(o^{j_2})$ then first case will be true.

If there is exactly one child o^j such that $\mathit{sym}(f(t_1, \ldots, t_n)) \subseteq \mathit{insym}(o^{j_1})$ then second case will be true. □

Proof (Proof for Theorem 3). We will proof that AHyp computes annotation invariants as base case and AComb and ACong inductively compute the annotation invariants.

AHYP *rule:* Let $\Pi = \textsc{MkHyp}(t \leq 0)$. For each $o \in R$, $\Pi(o)$ is $\langle [], p \rangle$, which implies $r = 0$ in the Definition 1 with respect to Π. Therefore, (AI-1),(AI-2a), (AI-3a), and (AI-3b) hold, trivially.

Let $o \in \mathit{leaves}(R)$. (AI-2b) holds since if $o = (t \leq 0)$ then $p = t$ else $p = 0$.

Let $(o^1, \ldots, o^m, o) \in R$. If $(t \leq 0)$ is in the subtree of the node o then $p = t$. Since R is a tree, there is $j \in 1..m$ such that the subtree of o^j contains $(t \leq 0)$. Therefore, $p^j = t$ and $\forall l \in 1..m \setminus \{j\} : p^l = 0$. Therefore, the right hand side of (AI-3b) is $0 \leq 0$. In other case,i.e., $t \leq 0$ is not in subtree of node o, $p = 0$ and $\forall j \in 1..m : p^l = 0$. Again the right hand side of (AI-3b) is $0 \leq 0$. Therefore, in both the cases (AI-3b) holds.

Since all leaves are in the subtree rooted at the node *false*, (AI-4) is satisfied.

If $(t \leq 0)$ is in the subtree of o then $p = t$. Hence, $p - t = 0$. Therefore (AI-5) and (AI-6) hold. Otherwise, i.e., if $(t \leq 0)$ is not in the subtree of o, then $p = 0$. Hence, $p - t = -t$. Therefore (AI-5) and (AI-6) holds.

ACOMB *rule:* By the induction hypothesis, Π_i is $t_i \leq 0$-annotation invariant for each $i \in 1..n$. Let $\Pi = \textsc{MkComb}(\Pi_1, \ldots, \Pi_n, \lambda_1, \ldots, \lambda_n)$. We show that Π is $\lambda_1 t_1 + \cdots + \lambda_n t_n \leq 0$-annotation invariant. For each $i \in 1..n$, we first construct $\overline{\Pi}_i$ such that

$$\forall o \in \mathit{nodes}(R) : \begin{pmatrix} \Pi_1(o) = \langle L_1, p_1 \rangle \\ \wedge \\ \vdots \\ \wedge \\ \Pi_n(o) = \langle L_n, p_n \rangle \end{pmatrix} \rightarrow \overline{\Pi}_i(o) = \langle L_1 \bullet \cdots \bullet L_n, p_i \rangle.$$

Due to Lemma 1, $\overline{\Pi}_i$ is $t_i \leq 0$-annotation invariant. MKCOMB constructs Π such that

$$\forall o \in \mathit{nodes}(R) : \begin{pmatrix} \overline{\Pi}_1(o) = \langle L, p_1 \rangle \\ \wedge \\ \vdots \\ \wedge \\ \overline{\Pi}_n(o) = \langle L, p_n \rangle \end{pmatrix} \rightarrow \Pi(o) = \langle L, \lambda_1 p_1 + \cdots + \lambda_n p_n \rangle.$$

(AI-1), (AI-2a), (AI-3a), (AI-3b), and (AI-4) w.r.t. $\lambda_1 t_1 + \cdots + \lambda_n t_n \leq 0$-annotation invariant are trivially satisfied.

Let $o \in \mathit{leaves}(R)$. The left hand sides of (AI-2b) w.r.t. $\overline{\Pi}_1(o), \ldots, \overline{\Pi}_n(o)$ are equal and they also equal to the left hand side of (AI-2b) w.r.t. $\Pi(o)$. The right hand side of (AI-2b) w.r.t. $\Pi(o)$ is a linear combination of the right hand sides of (AI-2b) w.r.t. $\overline{\Pi}_1(o), \ldots, \overline{\Pi}_n(o)$. Therefore, (AI-2b) w.r.t. $\Pi(o)$ holds. A similar argument proves (AI-3c). $\mathit{sym}(\{p_1 \leq 0, \ldots, p_n \leq 0\}) \subseteq \mathit{insym}(o)$, therefore $\mathit{sym}(\lambda_1 p_1 + \cdots + \lambda_n p_n) \subseteq \mathit{insym}(o)$. Hence, (AI-5) holds. A similar argument proves (AI-6).

ACONG *rule:* By the induction hypothesis, Π_i is $t_i - s_i \leq 0$-annotation invariant and Π'_i is $t_i - s_i \leq 0$-annotation invariant for $i \in 1..n$. Let $\Pi = \textsc{MkCong}(f(t_1, \ldots, t_n),\ f(s_1, \ldots, s_n), \Pi_1, \ldots, \Pi_n, \Pi'_1, \ldots, \Pi'_n)$. We prove that Π is $f(t_1, \ldots, t_n) - f(s_1, \ldots, s_n) \leq 0$-annotation invariant. For each $i \in 1..n$, we construct

$\overline{\Pi}_i$ and $\overline{\Pi'}_n$ such that

$$\forall o \in \mathit{nodes}(R) : \begin{pmatrix} \Pi_1(o) = \langle L_1, p_1 \rangle \\ \wedge \\ \vdots \\ \wedge \\ \Pi_n(o) = \langle L_n, p_n \rangle \\ \wedge \\ \Pi'_1(o) = \langle L'_1, p'_1 \rangle \\ \wedge \\ \vdots \\ \wedge \\ \Pi'_n(o) = \langle L'_n, p'_n \rangle \end{pmatrix} \rightarrow \begin{pmatrix} \overline{\Pi}_i(o) = \langle\ L_1 \bullet \cdots \bullet L_n \bullet \\ \qquad L'_1 \bullet \cdots \bullet L'_n,\ p_i\ \rangle \\ \wedge \\ \overline{\Pi'}_i(o) = \langle\ L_1 \bullet \cdots \bullet L_n \bullet \\ \qquad L'_1 \bullet \cdots \bullet L'_n,\ p'_i\ \rangle \end{pmatrix}.$$

Due to Lemma 1, $\overline{\Pi}_i$ satisfies $t_i - s_i \leq 0$-annotation invariant and $\overline{\Pi'}_i$ satisfies $s_i - t_i \leq 0$-annotation invariant for $i \in 1..n$.

Let $o \in \mathit{nodes}(R)$. Let $\overline{\Pi}_i(o) = \langle (\,(C_1, D_1), \ldots, (C_r, D_r)\,), p_i \rangle$ and let $\overline{\Pi'_i}(o) = \langle (\,(C_1, D_1), \ldots, (C_r, D_r)\,), p'_i \rangle$ for each $i \in 1..n$. MkCong returns Π such that $\Pi(o) = \langle (\,(C_1, D_1), \ldots, (C_r, D_r), (C_{r+1}, D_{r+1})\,), p \rangle$, where C_{r+1}, D_{r+1} and p are computed at line 5. At line 6 of function MkCong, match has four cases which we will lead to four or more cases distinction for proving (AI-1)–(AI-6) w.r.t. $f(t_1, \ldots, t_n) - f(s_1, \ldots, s_n) \leq 0$-annotation invariant. Now rest of the proof is divided into proving each of the conditions.

(AI-1): Since Π maps all nodes of R to solution constraints that have prefix sequence of length $r + 1$, (AI-1) holds.

(AI-5) and (AI-6): We show in the following four cases that C_{r+1}, D_{r+1}, and p satisfy (AI-5) and (AI-6).

1. $\mathit{sym}(f(t_1, \ldots, t_n)) \subseteq \mathit{outsym}(o) \wedge \mathit{sym}(f(s_1, \ldots, s_n)) \subseteq \mathit{outsym}(o)$:
   Let $i \in 1..n$. Due to the condition of this case, $\mathit{sym}(t_i - s_i) \subseteq \mathit{outsym}(o)$. (AI-6) w.r.t. $\overline{\Pi}_i(o)$ implies $\mathit{sym}(t_i - s_i - p_i) \subseteq \mathit{outsym}(o)$. Therefore, $\mathit{sym}(p_i) \subseteq \mathit{outsym}(o)$. Due to (AI-5) w.r.t. $\overline{\Pi}_i(o)$, $\mathit{sym}(p_i) \subseteq \mathit{insym}(o)$. A similar argument proves $\mathit{sym}(p'_i) \subseteq \mathit{outsym}(o)$ and $\mathit{sym}(p'_i) \subseteq \mathit{insym}(o)$. Therefore, C_{r+1} satisfies (AI-5) and (AI-6) w.r.t. $\Pi(o)$. Since, $D_{r+1} = \mathit{true}$ and $p = 0$, we do not need to prove anything for them.

2. $\mathit{sym}(f(t_1, \ldots, t_n)) \subseteq \mathit{outsym}(o) \wedge \mathit{sym}(f(s_1, \ldots, s_n)) \not\subseteq \mathit{outsym}(o)$:
   Let $i \in 1..n$. Due to (AI-5) w.r.t. $\overline{\Pi}_i(o)$ and $\overline{\Pi'_i}(o)$, $\mathit{sym}(p_i + p'_i) \in \mathit{insym}(o)$. Due to (AI-6), $\mathit{sym}(t_i - s_i - p_i) \in \mathit{outsym}(o)$ and $\mathit{sym}(s_i - t_i - p'_i) \in \mathit{outsym}(o)$ therefore $\mathit{sym}(-p_i - p'_i) \in \mathit{outsym}(o)$. Therefore, C_{r+1} and D_{r+1} satisfy (AI-5) and (AI-6) of Π.
   $\mathit{sym}(f(s_1, \ldots, s_n)) \not\subseteq \mathit{outsym}(o)$ implies $\mathit{sym}(f(s_1, \ldots, s_n)) \subseteq \mathit{insym}(o)$. Therefore, $\mathit{sym}(s_i) \subseteq \mathit{insym}(o)$. Therefore, $\mathit{sym}(s_i + p_i) \subseteq \mathit{insym}(o)$. Therefore, $\mathit{sym}(f(s_1 + p_1, \ldots, s_n + p_n) - f(s_1, \ldots, s_n)) \subseteq \mathit{insym}(o)$. Hence, (AI-5) w.r.t. $\Pi(o)$ holds. Due to conditions (AI-6) w.r.t. $\overline{\Pi}_i(o)$, $\mathit{sym}(t_i - s_i - p_i) \subseteq \mathit{outsym}(o)$. Since $\mathit{sym}(t_i) \subseteq \mathit{outsym}(o)$, $\mathit{sym}(s_i + p_i) \subseteq \mathit{outsym}(o)$. Therefore, $\mathit{sym}(f(t_1, \ldots, t_n) - f(s_1 + p_1, \ldots, s_n + p_n)) \subseteq \mathit{outsym}(o)$. Hence, (AI-6) w.r.t. $\Pi(o)$ holds.

3. $\mathit{sym}(f(t_1,\ldots,t_n)) \not\subseteq \mathit{outsym}(o) \wedge \mathit{sym}(f(s_1,\ldots,s_n)) \subseteq \mathit{outsym}(o)$:
   A similar argument as in the previous case.

4. $\mathit{sym}(f(t_1,\ldots,t_n)) \not\subseteq \mathit{outsym}(o) \wedge \mathit{sym}(f(s_1,\ldots,s_n)) \not\subseteq \mathit{outsym}(o)$:
   Due to the condition of this case, $\mathit{sym}(f(t_1,\ldots,t_n) - f(s_1,\ldots,s_n)) \subseteq \mathit{insym}(o)$. Hence, p satisfies (AI-5) and (AI-6) w.r.t. $\Pi(o)$. Let $i \in 1..n$. Due to (AI-6) w.r.t. $\overline{\Pi_i}(o)$ and $\overline{\Pi'_i}(o)$, $\mathit{sym}(t_i - s_i - p_i, s_i - t_i - p'_i) \subseteq \mathit{outsym}(o)$. Due to (AI-5) w.r.t. $\overline{\Pi_i}(o)$ and $\overline{\Pi'_i}(o)$, $\mathit{sym}(p_i, p'_i) \subseteq \mathit{insym}(o)$. Due to the condition of this case, $\mathit{sym}(t_i - s_i) \subseteq \mathit{insym}(o)$. Therefore, $\mathit{sym}(t_i - s_i - p_i, s_i - t_i - p'_i) \subseteq \mathit{insym}(o)$. Hence, D_{r+1} satisies (AI-5) and (AI-6) w.r.t. $\Pi(o)$. Since $C_{r+1} = \mathit{true}$, we do not have to prove anything for it.

(AI-2a) and (AI-2b): Let $o \in \mathit{leaves}(R)$. In (AI-2a) w.r.t. $\Pi(o)$, the implications for $i \in 1..r$ are satisfied due to (AI-2a) w.r.t. $\overline{\Pi_1}(o)$ and we only prove $r+1^{\text{th}}$ instantiation of the implications, i.e.,

$$\models o \wedge \ \bigwedge_{k=1}^{r} D_k \rightarrow C_{r+1}. \tag{6}$$

We also prove condition (AI-2b) w.r.t. $\Pi(o)$. There are again four cases.

1. $\mathit{sym}(f(t_1,\ldots,t_n)) \subseteq \mathit{outsym}(o) \wedge \mathit{sym}(f(s_1,\ldots,s_n)) \subseteq \mathit{outsym}(o)$
   Since $C_{r+1} = \bigwedge_{i=1}^{n}(p_i \leq 0 \wedge p'_i \leq 0)$, (AI-2b) w.r.t. $\overline{\Pi_i}(o)$ and $\overline{\Pi'_i}(o)$ imply (6). (AI-2b) w.r.t. $\Pi(o)$ is trivially satisfied.

2. $\mathit{sym}(f(t_1,\ldots,t_n)) \subseteq \mathit{outsym}(o) \wedge \mathit{sym}(f(s_1,\ldots,s_n)) \not\subseteq \mathit{outsym}(o)$
   Since $C_{r+1} = \bigwedge_{i=1}^{n}(p_i + p'_i \leq 0)$, (AI-2b) w.r.t. $\overline{\Pi_i}(o)$ and $\overline{\Pi'_i}(o)$ imply (6). In this case, $D_{r+1} = \bigwedge_{i=1}^{n}(-p_i - p'_i \leq 0)$. Let $i \in 1..n$. The left hand side of (AI-2b) w.r.t. $\Pi(o)$ implies $-p_i - p'_i \leq 0 \wedge p'_i \leq 0 \wedge p_i \leq 0$. So, $p_i = 0$. Therefore, $s_i + p_i = s_i$. Therefore, $f(s_1 + p_1, \ldots, s_n + p_n) - f(s_1,\ldots,s_n) \leq 0$, which is the right hand side of (AI-2b) w.r.t. $\Pi(o)$. Hence, (AI-2b) w.r.t. $\Pi(o)$ holds.

3. $\mathit{sym}(f(t_1,\ldots,t_n)) \not\subseteq \mathit{outsym}(o) \wedge \mathit{sym}(f(s_1,\ldots,s_n)) \subseteq \mathit{outsym}(o)$
   A similar argument as in the previous case.

4. $\mathit{sym}(f(t_1,\ldots,t_n)) \not\subseteq \mathit{outsym}(o) \wedge \mathit{sym}(f(s_1,\ldots,s_n)) \not\subseteq \mathit{outsym}(o)$
   In this case, $C_{r+1} = \mathit{true}$ and $D_{r+1} = \bigwedge_{i=1}^{n}(t_i - s_i - p_i \leq 0 \wedge s_i - t_i - p'_i \leq 0)$. (6) is trivially satisfied. Left hand sides of (AI-2b) w.r.t. $\overline{\Pi_i}$ and $\overline{\Pi'_i}$ are equal, and their conjunction with D_{r+1} is equal to the left hand side of (AI-2b) w.r.t. $\Pi(o)$. Therefore, the left hand side of (AI-2b) w.r.t. $\Pi(o)$ implies $\bigwedge_{i=1}^{n}(t_i - s_i - p_i \leq 0 \wedge s_i - t_i - p'_i \leq 0) \wedge \bigwedge_{i=1}^{n}(p_i \leq 0 \wedge p'_i \leq 0)$. Therefore, $\bigwedge_{i=1}^{n}(t_i - s_i \leq 0 \wedge s_i - t_i \leq 0)$. Therefore, $\bigwedge_{i=1}^{n} t_i = s_i$. Therefore, $f(t_1,\ldots,t_n) - f(s_1,\ldots,s_n) \leq 0$, which is the right hand side of (AI-2b) w.r.t. $\Pi(o)$. Hence, (AI-2b) w.r.t. $\Pi(o)$ holds.

(AI-3a), (AI-3b) and (AI-3c): Let $(o^1,\ldots,o^m,o) \in R$. For each $l \in 1..m$, let $\overline{\Pi_i}(o^l) = \langle((C_1^l, D_1^l),\ldots,(C_r^l, D_r^l)), p_i^l\rangle$, $\overline{\Pi'_i}(o^l) = \langle((C_1^l, D_1^l),\ldots,(C_r^l, D_r^l)), p_i^{l\prime}\rangle$, and $\overline{\Pi}(o^l) = \langle((C_1^l, D_1^l),\ldots,(C_r^l, D_r^l)), p^l\rangle$. In (AI-3a) w.r.t. $\Pi(o)$, the implications for $i \in 1..r$ are satisfied due to (AI-3a) w.r.t. $\overline{\Pi_1}(o)$. We only prove $r+1^{\text{th}}$ instantiation of the implications, i.e.,

$$\left(\bigwedge_{k=1}^{r+1}\bigwedge_{l=1}^{m} C_k^l\right) \ \wedge \ \bigwedge_{k=1}^{r} D_k \rightarrow C_{r+1}.$$

By reorganizing the above formula, we obtain

$$\bigwedge_{l=1}^{m} C_{r+1}^{l} \wedge \left(\left(\bigwedge_{k=1}^{r} \bigwedge_{l=1}^{m} C_{k}^{l}\right) \quad \wedge \quad \bigwedge_{k=1}^{r} D_k\right) \to C_{r+1}.$$

Due to (AI-3c) w.r.t. $\overline{\mathit{\Pi}_1}(o), \ldots, \overline{\mathit{\Pi}_n}(o)$ and $\overline{\mathit{\Pi}'_1}(o), \ldots, \overline{\mathit{\Pi}'_n}(o)$, we need to prove the following formula in order to prove the formula above.

$$\bigwedge_{l=1}^{m} C_{r+1}^{l} \wedge \bigwedge_{i=1}^{n} \begin{pmatrix} p_i - p_i^1 - \cdots - p_i^m \leq 0 \wedge \\ p'_i - p_i^{1\prime} - \cdots - p_i^{m\prime} \leq 0 \end{pmatrix} \to C_{r+1} \tag{7}$$

In (AI-3b) w.r.t. $\mathit{\Pi}(o)$, the implications for $i \in 1..r$ are satisfied due to (AI-3b) w.r.t. $\overline{\mathit{\Pi}_1}(o)$. We only prove $r+1^{\text{th}}$ instantiations of the implications, i.e.,

$$\begin{aligned} \forall j \in 1..m : \models \; & \left(\bigwedge_{l \in 1..m \setminus \{j\}} C_{r+1}^{l}\right) \wedge \\ & \left(\bigwedge_{k=1}^{r} \bigwedge_{l=1}^{m} C_{k}^{l}\right) \wedge \bigwedge_{k=1}^{r+1} D_k \to D_{r+1}^{j}. \end{aligned}$$

By reorganizing the above formula, we obtain

$$\begin{aligned} \forall j \in 1..m : \models \; & \left(\bigwedge_{l \in 1..m \setminus \{j\}} C_{r+1}^{l}\right) \wedge D_{r+1} \wedge \\ & \left(\left(\bigwedge_{k=1}^{r} \bigwedge_{l=1}^{m} C_{k}^{l}\right) \wedge \bigwedge_{k=1}^{r} D_k\right) \to D_{r+1}^{j}. \end{aligned}$$

Due to (AI-3c) w.r.t. $\overline{\mathit{\Pi}_1}(o), \ldots, \overline{\mathit{\Pi}_n}(o)$ and $\overline{\mathit{\Pi}'_1}(o), \ldots, \overline{\mathit{\Pi}'_n}(o)$, we need to prove the following formula in order to prove the formula above.

$$\begin{aligned} \forall j \in 1..m : \models \; & \left(\bigwedge_{l \in 1..m \setminus \{j\}} C_{r+1}^{l}\right) \wedge D_{r+1} \wedge \\ & \bigwedge_{i=1}^{n} \begin{pmatrix} p_i - p_i^1 - \cdots - p_i^m \leq 0 \wedge \\ p'_i - p_i^{1\prime} - \cdots - p_i^{m\prime} \leq 0 \end{pmatrix} \to D_{r+1}^{j} \end{aligned} \tag{8}$$

(AI-3c) w.r.t. $\mathit{\Pi}(o)$ is

$$\models \left(\bigwedge_{k=1}^{r+1} \bigwedge_{l=1}^{m} C_{k}^{l}\right) \quad \wedge \quad \bigwedge_{k=1}^{r+1} D_k \to p - p^1 - \cdots - p^m \leq 0.$$

By reorganizing the above formula, we obtain

$$\models \bigwedge_{l=1}^{m} C_{r+1}^{l} \wedge D_{r+1} \wedge \left(\left(\bigwedge_{k=1}^{r} \bigwedge_{l=1}^{m} C_{k}^{l}\right) \quad \wedge \quad \bigwedge_{k=1}^{r} D_k\right) \to p - p^1 - \cdots - p^m \leq 0.$$

Due to (AI-3c) w.r.t. $\overline{\mathit{\Pi}_1}(o), \ldots, \overline{\mathit{\Pi}_n}(o)$ and $\overline{\mathit{\Pi}'_1}(o), \ldots, \overline{\mathit{\Pi}'_n}(o)$, we need to prove the following formula in order to prove the formula above.

$$\begin{aligned} \models \; & \bigwedge_{l=1}^{m} C_{r+1}^{l} \wedge D_{r+1} \wedge \\ & \bigwedge_{i=1}^{n} \begin{pmatrix} p_i - p_i^1 - \cdots - p_i^m \leq 0 \wedge \\ p'_i - p_i^{1\prime} - \cdots - p_i^{m\prime} \leq 0 \end{pmatrix} \to p - p^1 - \cdots - p^m \leq 0 \end{aligned} \tag{9}$$

We prove (7), (8), and (9) for the following ten cases, which are consequence of Lemmas 2 and 3. In each case, we will present the table of values of C_{r+1}, D_{r+1}, p, and, for each $l \in 1..m$, C_{r+1}^{l}, D_{r+1}^{l} and p^l. Then, provide proves of (7), (8), and (9) for the given values.

1. $sym(f(t_1,\ldots,t_n)) \subseteq outsym(o) \wedge sym(f(s_1,\ldots,s_n)) \subseteq outsym(o)$:

	$\forall l \in 1..m$
$C_{r+1} = \bigwedge_{i=1}^{n}(p_i \leq 0 \wedge p'_i \leq 0)$	$C_{r+1}^{l} = \bigwedge_{i=1}^{n}(p_i^l \leq 0 \wedge p_i^{l\prime} \leq 0)$
$D_{r+1} = true$	$D_{r+1}^{l} = true$
$p = 0$	$p^l = 0$

(8) and (9) are trivially satisfied. Placing values of C_{r+1}^{l} in left hand side of (7), we obtain

$$\bigwedge_{l=1}^{m} \bigwedge_{i=1}^{n}(p_i^l \leq 0 \wedge p_i^{l\prime} \leq 0) \wedge \bigwedge_{i=1}^{n} \begin{pmatrix} p_i - p_i^1 - \cdots - p_i^m \leq 0 \wedge \\ p'_i - p_i^{1\prime} - \cdots - p_i^{m\prime} \leq 0 \end{pmatrix}.$$

By taking linear combination of above atoms, we obtain

$$\bigwedge_{i=1}^{n} \left(p_i \leq 0 \wedge p'_i \leq 0 \right),$$

which is right hand side of (7).

2. $sym(f(t_1,\ldots,t_n)) \subseteq outsym(o) \wedge sym(f(s_1,\ldots,s_n)) \not\subseteq outsym(o) \wedge$
$\left(\forall j \in 1..m : \begin{array}{l} sym(f(t_1,\ldots,t_n)) \subseteq outsym(o^j) \wedge \\ sym(f(s_1,\ldots,s_n)) \subseteq outsym(o^j) \end{array}\right)$:

	$\forall l \in 1..m$
$C_{r+1} = \bigwedge_{i=1}^{n}(p_i + p'_i \leq 0)$	$C_{r+1}^{l} = \bigwedge_{i=1}^{n}(p_i^l \leq 0 \wedge p_i^{l\prime} \leq 0)$
$D_{r+1} = \bigwedge_{i=1}^{n}(-p_i - p'_i \leq 0)$	$D_{r+1}^{l} = true$
$p = f(s_1 + p_1, \ldots, s_n + p_n) - f(s_1, \ldots, s_n)$	$p^l = 0$

(8) is trivially true. The left hand side of (7) is equal to the previous case, therefore, it implies $\bigwedge_{i=1}^{n} \left(p_i \leq 0 \wedge p'_i \leq 0 \right)$. By taking linear combination of inequalities, we obtain $\bigwedge_{i=1}^{n} (p_i + p'_i \leq 0)$, which is the right hand side of (7). In the right hand side of (9), $p - p^1 - \cdots - p^n = f(s_1 + p_1, \ldots, s_n + p_n) - f(s_1, \ldots, s_n)$. Left hand side of (9) implies

$$\bigwedge_{l=1}^{m} \bigwedge_{i=1}^{n}(p_i^l \leq 0 \wedge p_i^{l\prime} \leq 0) \wedge D_{r+1} \wedge \bigwedge_{i=1}^{n} \begin{pmatrix} p_i - p_i^1 - \cdots - p_i^m \leq 0 \wedge \\ p'_i - p_i^{1\prime} - \cdots - p_i^{m\prime} \leq 0 \end{pmatrix}.$$

By taking linear combinations, we obtain

$$D_{r+1} \wedge \bigwedge_{i=1}^{n} (p_i \leq 0 \wedge p'_i \leq 0).$$

After placing value of D_{r+1},

$$\bigwedge_{i=1}^{n}(-p_i - p'_i \leq 0) \wedge \bigwedge_{i=1}^{n} (p_i \leq 0 \wedge p'_i \leq 0).$$

By taking linear combinations, we obtain

$$\bigwedge_{i=1}^{n}(-p_i \leq 0 \wedge p_i \leq 0).$$

So for all $i \in 1..n$, $p_i = 0$. Therefore, $s_i + p_i = s_i$. Therefore, $f(s_1 + p_1, \ldots, s_n + p_n) - f(s_1, \ldots, s_n) \leq 0$, which is right hand side of (9).

3. $sym(f(t_1,\ldots,t_n)) \subseteq outsym(o) \wedge sym(f(s_1,\ldots,s_n)) \not\subseteq outsym(o) \wedge$
$\left(\exists j \in 1..m : \begin{array}{l} sym(f(t_1,\ldots,t_n)) \subseteq outsym(o^j) \wedge \\ sym(f(s_1,\ldots,s_n)) \not\subseteq outsym(o^j) \end{array}\right)$:

	$\forall l \in 1..m \setminus \{j\}$
$C_{r+1} = \bigwedge_{i=1}^{n}(p_i + p'_i \leq 0)$	$C^l_{r+1} = \bigwedge_{i=1}^{n}(p^l_i \leq 0 \wedge p_i^{l\prime} \leq 0)$
$D_{r+1} = \bigwedge_{i=1}^{n}(-p_i - p'_i \leq 0)$	$D^l_{r+1} = \mathit{true}$
$p = f(s_1 + p_1, \dots, s_n + p_n) - f(s_1, \dots, s_n)$	$p^l = 0$

$C^j_{r+1} = \bigwedge_{i=1}^{n}(p^j_i + p_i^{j\prime} \leq 0)$
$D^j_{r+1} = \bigwedge_{i=1}^{n}(-p^j_i - p_i^{j\prime} \leq 0)$
$p^j = f(s_1 + p^j_1, \dots, s_n + p^j_n) - f(s_1, \dots, s_n)$

Left hand side of (7) implies

$$(\textstyle\bigwedge_{l \in 1..m \setminus \{j\}} \bigwedge_{i=1}^{n}(p^l_i \leq 0 \wedge p_i^{l\prime} \leq 0)) \wedge \bigwedge_{i=1}^{n}(p^j_i + p_i^{j\prime} \leq 0) \wedge \\ \textstyle\bigwedge_{i=1}^{n} \begin{pmatrix} p_i - p^1_i - \dots - p^m_i \leq 0 \wedge \\ p'_i - p_i^{1\prime} - \dots - p_i^{m\prime} \leq 0 \end{pmatrix}.$$

By taking linear combinations, we obtain $\bigwedge_{i=1}^{n} p_i + p'_i \leq 0$, which is right hand side of (7).

For (8), we only need to prove the instance of implications in which, D^j_{r+1} is equal to $\bigwedge_{i=1}^{n}(-p^j_i - p_i^{j\prime} \leq 0)$. Lets consider left hand side of (8), which implies

$$\bigwedge_{i}^{n} \left(\begin{array}{l} \left(\bigwedge_{l \in 1..m \setminus \{j\}}(p^l_i \leq 0 \wedge p_i^{l\prime} \leq 0)\right) \wedge \\ \left(p_i - p^1_i - \dots - p^m_i \leq 0 \wedge p'_i - p_i^{1\prime} - \dots - p_i^{m\prime} \leq 0\right) \wedge -p_i - p'_i \leq 0 \end{array} \right).$$

By by adding above linear inequalities, we can obtain $\bigwedge_{i=1}^{n} -p^j_i - p_i^{j\prime} \leq 0$, which is right hand side of (8).

In the right hand side of (9), $p - p^1 - \dots - p^m = f(s_1 + p_1, \dots, s_n + p_n) - f(s_1 + p^j_1, \dots, s_n + p^j_n)$. So for proving (9), we need to show that the left hand side implies $\bigwedge_{i=0}^{n} s_i + p_i = s_i + p^j_i$. By further simplification, $\bigwedge_{i=0}^{n} p_i - p^j_i = 0$. Now, lets consider the left hand side, which implies

$$\bigwedge_{i}^{n} \left(\begin{array}{l} \bigwedge_{l \in 1..m \setminus j} p^l_i \leq 0 \wedge p_i - p^1_i - \dots - p^m_i \leq 0 \wedge \\ \bigwedge_{l \in 1..m \setminus j} p_i^{l\prime} \leq 0 \wedge p'_i - p_i^{1\prime} - \dots - p_i^{m\prime} \leq 0 \wedge \\ (p^j_i + p_i^{j\prime} \leq 0) \wedge -p_i - p'_i \leq 0 \end{array} \right)$$

By adding inequalities of each row, we obtain

$$\bigwedge_{i}^{n} \left(\begin{array}{l} p_i - p^j_i \leq 0 \wedge \\ p'_i - p_i^{j\prime} \leq 0 \wedge \\ p^j_i + p_i^{j\prime} - p_i - p'_i \leq 0 \end{array} \right).$$

By adding 2nd and 3rd row, we obtain

$$\bigwedge_{i}^{n} \left(p_i - p^j_i \leq 0 \wedge p^j_i - p_i \leq 0 \right),$$

which we were aiming to prove.

4. $\mathit{sym}(f(t_1, \dots, t_n)) \nsubseteq \mathit{outsym}(o) \wedge \mathit{sym}(f(s_1, \dots, s_n)) \subseteq \mathit{outsym}(o) \wedge$ $\left(\forall j \in 1..m : \begin{array}{l} \mathit{sym}(f(t_1, \dots, t_n)) \subseteq \mathit{outsym}(o^j) \wedge \\ \mathit{sym}(f(s_1, \dots, s_n)) \subseteq \mathit{outsym}(o^j) \end{array}\right)$:
   Argument is similar to case 2.

5. $sym(f(t_1,\dots,t_n)) \nsubseteq outsym(o) \wedge sym(f(s_1,\dots,s_n)) \subseteq outsym(o) \wedge$
$\left(\exists j \in 1..m : \begin{array}{l} sym(f(t_1,\dots,t_n)) \nsubseteq outsym(o^j)\wedge \\ sym(f(s_1,\dots,s_n)) \subseteq outsym(o^j) \end{array}\right)$:
Argument is similar to case 3.

6. $sym(f(t_1,\dots,t_n)) \nsubseteq outsym(o) \wedge sym(f(s_1,\dots,s_n)) \nsubseteq outsym(o) \wedge$
$\left(\forall j \in 1..m : \begin{array}{l} sym(f(t_1,\dots,t_n)) \subseteq outsym(o^j)\wedge \\ sym(f(s_1,\dots,s_n)) \subseteq outsym(o^j) \end{array}\right)$:

$C_{r+1} = true$
$D_{r+1} = \bigwedge_{i=1}^{n}(t_i - s_i - p_i \leq 0 \wedge s_i - t_i - p'_i \leq 0)$
$p = f(t_1,\dots,t_n) - f(s_1,\dots,s_n)$

For each $l \in 1..m$
$C^l_{r+1} = \bigwedge_{i=1}^{n}(p^l_i \leq 0 \wedge p^{l\prime}_i \leq 0)$
$D^l_{r+1} = true$
$p^l = 0$

(7) and (8) are trivially true. In the right hand side of (9), $p - p^1 - \cdots - p^m = f(t_1,\dots,t_n) - f(s_1,\dots,s_n)$. So, we only need to prove that left hand side of (9) implies $\bigwedge_{i=1}^{n} t_i = s_i$. By placing values of C^l_{r+1} and D_{r+1}, the left hand side implies

$$\bigwedge_{i=1}^{n}(p_i \leq 0 \wedge p'_i \leq 0 \wedge t_i - s_i - p_i \leq 0 \wedge s_i - t_i - p'_i \leq 0).$$

By taking linear combinations, we obtain

$$\bigwedge_{i=1}^{n}(t_i - s_i \leq 0 \wedge s_i - t_i \leq 0),$$

which we were aiming to prove.

7. $sym(f(t_1,\dots,t_n)) \nsubseteq outsym(o) \wedge sym(f(s_1,\dots,s_n)) \nsubseteq outsym(o) \wedge$
$\left(\exists j \in 1..m : \begin{array}{l} sym(f(t_1,\dots,t_n)) \nsubseteq outsym(o^j) \wedge \\ sym(f(s_1,\dots,s_n)) \subseteq outsym(o^j) \end{array}\right) \wedge$
$\left(\forall j' \in 1..m \setminus \{j\} : \begin{array}{l} sym(f(t_1,\dots,t_n)) \subseteq outsym(o^{j'}) \wedge \\ sym(f(s_1,\dots,s_n)) \subseteq outsym(o^{j'}) \end{array}\right)$:

$C_{r+1} = true$
$D_{r+1} = \bigwedge_{i=1}^{n}(t_i - s_i - p_i \leq 0 \wedge s_i - t_i - p'_i \leq 0)$
$p = f(t_1,\dots,t_n) - f(s_1,\dots,s_n)$

For each $l \in 1..m \setminus \{j\}$
$C^l_{r+1} = \bigwedge_{i=1}^{n}(p^l_i \leq 0 \wedge p^{l\prime}_i \leq 0)$
$D^l_{r+1} = true$
$p^l = 0$

$C^j_{r+1} = \bigwedge_{i=1}^{n}(p^j_i + p^{j\prime}_i \leq 0)$
$D^j_{r+1} = \bigwedge_{i=1}^{n}(-p^j_i - p^{j\prime}_i \leq 0)$
$p^j = f(s_1 + p^j_1,\dots,s_n + p^j_n) - f(s_1,\dots,s_n)$

(7) is trivially true. For (8), we only need to prove the instance of implications in which, D^j_{r+1} is equal to $\bigwedge_{i=1}^{n}(-p^j_i - p^{j\prime}_i \leq 0)$. Lets consider left hand side of (8), which is

$$(\textstyle\bigwedge_{l\in 1..m\setminus\{j\}} C^l_{r+1}) \wedge D_{r+1} \wedge \bigwedge_{i=1}^{n} \begin{pmatrix} p_i - p^1_i - \cdots - p^m_i \leq 0 \wedge \\ p'_i - p^{1\prime}_i - \cdots - p^{m\prime}_i \leq 0 \end{pmatrix}.$$

After placing values of C^l_{r+1} and D_{r+1},

$$\bigwedge_{i=1}^{n} \begin{pmatrix} t_i - s_i - p_i \leq 0 \wedge \\ s_i - t_i - p'_i \leq 0 \end{pmatrix} \wedge \bigwedge_{i=1}^{n} \begin{pmatrix} p_i - p^j_i \leq 0 \wedge \\ p'_i - p^{j\prime}_i \leq 0 \end{pmatrix}.$$

After adding all inequalities above, we obtain $\bigwedge_{i=1}^{n}\left(-p^j_i - p^{j\prime}_i \leq 0\right)$, which is right hand side of (8).

In the right hand side of (9), $p - p^1 - \cdots - p^m = f(t_1, \ldots, t_n) - f(s_1 + p^j_1, \ldots, s_n + p^j_n)$. So, we only need to prove that left hand side of (9) implies $\bigwedge_{i=1}^n t_i = s_i + p^j_i$. By placing values of C^l_{r+1} and D_{r+1}, the left hand side implies

$$\bigwedge_{i=1}^n (p^j_i + p^{j\prime}_i \leq 0) \wedge \bigwedge_{i=1}^{n} \begin{pmatrix} t_i - s_i - p_i \leq 0 \wedge \\ s_i - t_i - p'_i \leq 0 \end{pmatrix} \wedge \bigwedge_{i=1}^{n} \begin{pmatrix} p_i - p^j_i \leq 0 \wedge \\ p'_i - p^{j\prime}_i \leq 0 \end{pmatrix}$$

by taking linear combination of above equations,

$$\bigwedge_{i=1}^{n} \begin{pmatrix} t_i - s_i - p^{j\prime}_i \leq 0 \wedge \\ s_i - t_i - p^j_i \leq 0 \end{pmatrix} \wedge \bigwedge_{i=1}^{n} \begin{pmatrix} t_i - s_i - p^j_i \leq 0 \wedge \\ s_i - t_i - p^{j\prime}_i \leq 0 \end{pmatrix}$$

Therefore, $\bigwedge_{i=1}^n t_i = s_i + p^j_i$, which we were aiming to prove.

8. $sym(f(t_1, \ldots, t_n)) \not\subseteq outsym(o) \wedge sym(f(s_1, \ldots, s_n)) \not\subseteq outsym(o) \wedge$
$$\left(\exists j \in 1..m : \begin{array}{l} sym(f(t_1, \ldots, t_n)) \subseteq outsym(o^j) \wedge \\ sym(f(s_1, \ldots, s_n)) \not\subseteq outsym(o^j) \end{array} \right) \wedge$$
$$\left(\forall j' \in 1..m \setminus \{j\} : \begin{array}{l} sym(f(t_1, \ldots, t_n)) \subseteq outsym(o^{j'}) \wedge \\ sym(f(s_1, \ldots, s_n)) \subseteq outsym(o^{j'}) \end{array} \right)$$

   A similar argument as in previous case.

9. $sym(f(t_1, \ldots, t_n)) \not\subseteq outsym(o) \wedge sym(f(s_1, \ldots, s_n)) \not\subseteq outsym(o) \wedge$
$$\left(\exists j^1 \in 1..m : \begin{array}{l} sym(f(t_1, \ldots, t_n)) \subseteq outsym(o^{j^1}) \wedge \\ sym(f(s_1, \ldots, s_n)) \not\subseteq outsym(o^{j^1}) \end{array} \right) \wedge$$
$$\left(\exists j^2 \in 1..m : \begin{array}{l} sym(f(t_1, \ldots, t_n)) \not\subseteq outsym(o^{j^2}) \wedge \\ sym(f(s_1, \ldots, s_n)) \subseteq outsym(o^{j^2}) \end{array} \right) \wedge$$
$$\left(\forall j' \in 1..m \setminus \{j^1, j^2\} : \begin{array}{l} sym(f(t_1, \ldots, t_n)) \subseteq outsym(o^{j'}) \wedge \\ sym(f(s_1, \ldots, s_n)) \subseteq outsym(o^{j'}) \end{array} \right)$$

$C_{r+1} = true$
$D_{r+1} = \bigwedge_{i=1}^n (t_i - s_i - p_i \leq 0 \wedge s_i - t_i - p'_i \leq 0)$
$p = f(t_1, \ldots, t_n) - f(s_1, \ldots, s_n)$

For each $l \in 1..m \setminus \{j^1, j^2\}$
$C^l_{r+1} = \bigwedge_{i=1}^n (p^l_i \leq 0 \wedge p^{l\prime}_i \leq 0)$
$D^l_{r+1} = true$
$p^l = 0$

$C^{j^1}_{r+1} = \bigwedge_{i=1}^n (p^{j^1}_i + p^{j^1\prime}_i \leq 0)$
$D^{j^1}_{r+1} = \bigwedge_{i=1}^n (-p^{j^1}_i - p^{j^1\prime}_i \leq 0)$
$p^{j^1} = f(s_1 + p^{j^1}_1, \ldots, s_n + p^{j^1}_n) - f(s_1, \ldots, s_n)$

$C^{j^2}_{r+1} = \bigwedge_{i=1}^n (p^{j^2}_i + p^{j^2\prime}_i \leq 0)$
$D^{j^2}_{r+1} = \bigwedge_{i=1}^n (-p^{j^2}_i - p^{j^2\prime}_i \leq 0)$
$p^{j^2} = f(t_1, \ldots, t_n) - f(t_1 + p^{j^2\prime}_1, \ldots, t_n + p^{j^2\prime}_n)$

   (7) is trivially true. In (8), there are two non trivial implications, when $j = j^1$ and $j = j^2$. For $j = j^1$, the left hand side of implication is

$$(\bigwedge_{l \in 1..m \setminus \{j^1\}} C^l_{r+1}) \wedge D_{r+1} \wedge \bigwedge_{i=1}^{n} \begin{pmatrix} p_i - p^1_i - \cdots - p^m_i \leq 0 \wedge \\ p'_i - p^{1\prime}_i - \cdots - p^{m\prime}_i \leq 0 \end{pmatrix}.$$

   After placing values of C^l_{r+1} other than $l = j^2$, we obtain

$$C^{j^2}_{r+1} \wedge D_{r+1} \wedge \bigwedge_{i=1}^{n} \begin{pmatrix} p_i - p^{j^1}_i - p^{j^2}_i \leq 0 \wedge \\ p'_i - p^{j^1\prime}_i - p^{j^2\prime}_i \leq 0 \end{pmatrix}.$$

After placing values of $C_{r+1}^{j^2}$ and D_{r+1}, we obtain

$$\begin{array}{l} \bigwedge_{i=1}^{n}(p_i^{j^2} + p_i^{{j^2}\prime} \leq 0) \wedge \bigwedge_{i=1}^{n} \begin{pmatrix} t_i - s_i - p_i \leq 0 \wedge \\ s_i - t_i - p_i' \leq 0 \end{pmatrix} \wedge \\ \bigwedge_{i=1}^{n} \begin{pmatrix} p_i - p_i^{j^1} - p_i^{j^2} \leq 0 \wedge \\ p_i' - p_i^{j^1_\prime} - p_i^{j^2_\prime} \leq 0 \end{pmatrix}. \end{array}$$

By taking linear combinations, we obtain

$$\bigwedge_{i=1}^{n} \left(-p_i^{j^1} - p_i^{j^1_\prime} \leq 0 \right),$$

which is the right hand side. A similar argument proves $j = j^2$ instantiation of (8). The left hand side of (9) is

$$\begin{array}{l} (\bigwedge_{l \in 1..m \setminus \{j^1, j^2\}} C_{r+1}^{l}) \wedge C_{r+1}^{j^1} \wedge C_{r+1}^{j^2} \wedge D_{r+1} \wedge \\ \bigwedge_{i=1}^{n} \begin{pmatrix} p_i - p_i^1 - \cdots - p_i^m \leq 0 \wedge \\ p_i' - p_i^{1\prime} - \cdots - p_i^{m\prime} \leq 0 \end{pmatrix}. \end{array}$$

By placing values of C_{r+1}^{j} for $j \in 1..m \setminus \{j^1, j^2\}$, we obtain

$$C_{r+1}^{j^1} \wedge C_{r+1}^{j^2} \wedge D_{r+1} \wedge \bigwedge_{i=1}^{n} \begin{pmatrix} p_i - p_i^{j^1} - p_i^{j^2} \leq 0 \wedge \\ p_i' - p_i^{j^1_\prime} - p_i^{j^2_\prime} \leq 0 \end{pmatrix}.$$

After placing value of D_{r+1}, we obtain

$$C_{r+1}^{j^1} \wedge C_{r+1}^{j^2} \wedge \bigwedge_{i=1}^{n} \begin{pmatrix} t_i - s_i - p_i^{j^1} - p_i^{j^2} \leq 0 \wedge \\ s_i - t_i - p_i^{j^1_\prime} - p_i^{j^2_\prime} \leq 0 \end{pmatrix}.$$

After placing values of $C_{r+1}^{j^1}$ and $C_{r+1}^{j^2}$, we obtain

$$\bigwedge_{i=1}^{n} \begin{pmatrix} p_i^{j^1} + p_i^{{j^1}\prime} \leq 0 \\ p_i^{j^2} + p_i^{{j^2}\prime} \leq 0 \end{pmatrix} \wedge \bigwedge_{i=1}^{n} \begin{pmatrix} t_i - s_i - p_i^{j^1} - p_i^{j^2} \leq 0 \wedge \\ s_i - t_i - p_i^{j^1_\prime} - p_i^{j^2_\prime} \leq 0 \end{pmatrix}.$$

By taking linear combinations of above inequalities, we obtain

$$\bigwedge_{i=1}^{n} \begin{pmatrix} t_i - s_i + p_i^{j^1_\prime} - p_i^{j^2} \leq 0 \wedge \\ s_i - t_i - p_i^{j^1_\prime} + p_i^{j^2} \leq 0 \end{pmatrix}.$$

Therefore,

$$\bigwedge_{i=1}^{n} \left(t_i + p_i^{j^1_\prime} = s_i + p_i^{j^2} \right)$$

Therefore,

$$f(t_1 + p_1^{{j^2}\prime}, \ldots, t_n + p_n^{{j^2}\prime}) - f(s_1 + p_1^{j^1}, \ldots, s_n + p_n^{j^1}) \leq 0,$$

which is right hand side of (9).

10. $\mathit{sym}(f(t_1,\ldots,t_n)) \nsubseteq \mathit{outsym}(o) \wedge \mathit{sym}(f(s_1,\ldots,s_n)) \nsubseteq \mathit{outsym}(o) \wedge$
$\left(\exists j \in 1..m : \begin{array}{l} \mathit{sym}(f(t_1,\ldots,t_n)) \nsubseteq \mathit{outsym}(o^j) \wedge \\ \mathit{sym}(f(s_1,\ldots,s_n)) \nsubseteq \mathit{outsym}(o^j) \end{array}\right) \wedge$
$\left(\forall j' \in 1..m \setminus \{j\} : \begin{array}{l} \mathit{sym}(f(t_1,\ldots,t_n)) \subseteq \mathit{outsym}(o^{j'}) \wedge \\ \mathit{sym}(f(s_1,\ldots,s_n)) \subseteq \mathit{outsym}(o^{j'}) \end{array}\right) :$

$C_{r+1} = \mathit{true}$
$D_{r+1} = \bigwedge_{i=1}^{n}(t_i - s_i - p_i \leq 0 \wedge s_i - t_i - p'_i \leq 0)$
$p = f(t_1,\ldots,t_n) - f(s_1,\ldots,s_n)$

For each $l \in 1..m \setminus \{j\}$
$C^l_{r+1} = \bigwedge_{i=1}^{n}(p^l_i \leq 0 \wedge p^{l\prime}_i \leq 0)$
$D^l_{r+1} = \mathit{true}$
$p^l = 0$

$C^j_{r+1} = \mathit{true}$
$D^{j^1}_{r+1} = \bigwedge_{i=1}^{n}(t_i - s_i - p^j_i \leq 0 \wedge s_i - t_i - p^{j\prime}_i \leq 0)$
$p^{j^1} = f(t_1,\ldots,t_n) - f(s_1,\ldots,s_n)$

(7) and (9) are trivially true. For (8), we only need to prove the instance of implications in which, D^j_{r+1} is equal to $\bigwedge_{i=1}^{n}(t_i - s_i - p^j_i \leq 0 \wedge s_i - t_i - p^{j\prime}_i \leq 0)$. Lets consider left hand side of (8), which implies

$$\bigwedge_{i=1}^{n} \begin{pmatrix} t_i - s_i - p_i \leq 0 \wedge \\ s_i - t_i - p'_i \leq 0 \end{pmatrix} \wedge \bigwedge_{i=1}^{n} \begin{pmatrix} p_i - p^j_i \leq 0 \wedge \\ p'_i - p^{j\prime}_i \leq 0 \end{pmatrix}.$$

By taking linear combinations, we obtain

$$\bigwedge_{i=1}^{n} \begin{pmatrix} t_i - s_i - p^j_i \leq 0 \wedge \\ s_i - t_i - p^{j\prime}_i \leq 0 \end{pmatrix},$$

which is the right hand side.

(AI-4): Let $o = \mathit{false}$. The node *false* is root of the resolution tree therefore $\mathit{sym}(f(t_1,\ldots,t_n)) \nsubseteq \mathit{outsym}(o)$ and $\mathit{sym}(f(s_1,\ldots,s_n)) \nsubseteq \mathit{outsym}(o)$. Therefore, $C_{r+1} = \mathit{true}$ and $D_{r+1} = \bigwedge_{i=1}^{n}(t_i - s_i - p_i \leq 0 \wedge s_i - t_i - p'_i \leq 0)$. Since, for each $i \in 1..n$, $p_i = t_i - s_i$ and $p'_i = s_i - t_i$, $D_{r+1} = \mathit{true}$. Hence, (AI-4) w.r.t. $\mathit{\Pi}(o)$ holds.

Proof (Proof for Theorem 4). The annotation of the rules are computed in linear pass by depth first traversal of a proof-tree. At each node or the proof tree, the nodes of resolution tree are traversed.