Tree Interpolants via Localized Proofs

Ashutosh Gupta[1] Alexandre Thevenet-Montagne[1,2]

[1]IST, Austria [2]ENS Ulm

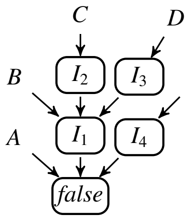
Abstract. Tree interpolation is a generalization of interpolation, in which partitions of an unsat formula are arranged in a tree rather than a sequence. Tree interpolation is needed for verification of programs with complex control flows. In this paper, we present a novel method for computing the tree interpolants in the theory of QF_UFLRA. Our method obtains a proof of unsatisfiability of the unsat formula using some theorem solver. Using a single pass algorithm, we transform the proofs of the theory axioms into a restricted form, which we call *localized proofs*. We further transform the Boolean part of the proof to obtain a fully localized proof. We compute the interpolants from the fully localized proofs in a straight forward way.

1 Introduction

Interpolation is a useful method to find concise explanations of impossibility of certain program behaviors. Interpolation has been leveraged by various formal verification techniques, including abstraction refinement [16], invariant generation [24], and bounded model checking [22]. Let A and B be two formulas such that $A \wedge B$ is unsatisfiable. An interpolant I between A and B is a formula such that $A \rightarrow I$, $I \wedge B \rightarrow \mathit{false}$, and I contains only the symbols that appear both in A and B. If A and B represent two parts of a program then I is an explanation of the infeasibility of any trace of the program that starts from A and ends in B.

Tree interpolation is a generalization of interpolation. Tree interpolation has been applied for verification of the programs with complex control flows, e.g., multi-threaded programs [13], recursive programs [15], and higher-order programs [10]. In tree interpolation, the partitions of an unsat formula are arranged in a tree rather than a sequence. Each leaf of the tree is a formula, the root of the tree is *false*, and each internal node in the tree represents an unknown formula. Tree interpolation finds formulas for the unknown formulas such that the formula for an internal node is entailed by the conjunction of its children and contains the symbols that occur both the inside and outside the subtree rooted at the internal node. In figure 1(a), we present a tree interpolation problem. A, B, C, D, and E are formulas. The goal is to find formulas I_1, I_2, I_3, and I_4 such that the implications shown in figure 1(b) are satisfied along with the symbol occurrence restrictions. For example, I_1 can only contain the symbols that occurs both in $A \wedge E$ and $B \wedge C \wedge D$. In figure 1(c), we present an interpolation problem as a tree interpolation problem.

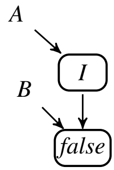
Usually one computes interpolants modulo a theory. A typical interpolation procedure requires a refutation proof of the unsatisfiability of the conjunction of both the parts. By annotating the refutation proof, it computes the interpolant. For each theory



(a)

$$A \wedge I_1 \wedge I_4 \rightarrow \mathit{false}$$
$$B \wedge I_2 \wedge I_3 \rightarrow I_1$$
$$C \rightarrow I_2$$
$$D \rightarrow I_3$$
$$E \rightarrow I_4$$

(b)



(c)

Fig. 1. (a)A tree interpolation problem (b) In logical form (c) Interpolation as tree interpolation

proof rule, one needs a corresponding annotation rule. For example, [23] presents the annotation rules for the theory of QF_UFLRA.

One may compute a tree interpolant using a näive iterative procedure which calls an interpolation procedure in each iteration to compute a solution of one of the unknown formulas [15]. [14] presents a method that annotates the poofs in the theory of QF_UFLRA such that the tree interpolants are computed in a single iteration.

The annotation rules of the proof implies certain restrictions on the proof structure. Therefore, a proof search engine has to restrict the proof search space to produce such proofs. This restriction on the proof search engine may impact its performance. In the past decade, a significant research in *satisfiability modulo theory*(SMT) solvers has produced several efficient tools [7, 1]. Some of these tools can produce proofs of unsatisfiability. Since they are highly optimized, the proofs produced by them may not have the structure as desired by the interpolation procedure.

ɪZ3 [26]–an interpolation tool–avoids this restriction on the proof structure and allows an efficient SMT solver (Z3) to produce arbitrary proofs. ɪZ3 transforms the arbitrary proofs into "localized proofs" [20] and generate the interpolants using theory independent proof annotation rules. Since the transformation rules of ɪZ3 are theory *unaware*, ɪZ3 may fail to transform a part of the proof into a localized proof. In that case, it transforms the proof such that the part becomes an axiom in the proof and a third party interpolation tool is called to compute the annotation for the axiom. [26] also observes that this method of computing interpolation is more efficient than the approach that interferes with the solver. ɪZ3 also computes [25] tree interpolants but the publicly available information does not describe the employed method of computing tree interpolants.

In this paper, we extend the definition of localized proofs for tree interpolation. We also present a theory *aware* proof transformation method for localizing proofs of axioms of the theory of QF_UFLRA. The above transformation traverses a proof only once and due to this transformation the proof size does not blow up. We also present a proof transformation method for localizing Boolean part of the proof and we show a trivial tree interpolant computation from such fully localized proofs.

Related work: [6] proves that an interpolant always exist between any two mutually unsatisfiable formula. However, one may expect the interpolant to satisfy certain logical restrictions. In that case, an interpolant may not exist. There has been significant research to find algorithms that compute interpolants for various theories. Many interpolation procedure for various theories are presented in [23, 2–4, 21, 19, 5].

Multiple interpolants may satisfy an interpolation problem. The verification techniques that uses interpolation are sensitive to the choice between these interpolants. Therefore, there has been a significant focus [2, 8, 11, 20, 21, 18, 24, 17] to compute interpolants that are optimized under some criteria.

We specially note that in [17] and [26], proofs are transformed either to compute optimized interpolants or interpolants efficiently. Both techniques use theory *unaware* transformation rules which leads one to introduce quantifiers and another to call another interpolation tool. Our theory aware transformation rules can help these methods to avoid the drawbacks.

The rest of the paper is organizes as follows. In section 2, we present basic notation of logical formulas, a proof system for the theory of QF_UFLRA, and formally define tree interpolation. In section 3, we present a method of computing tree interpolants via localized proofs. In section 4, we will conclude the paper.

2 Preliminaries

In this section, we will define the language of formulas in the theory of uninterpreted functions with linear rational arithmetic, a proof system for the theory, and formally define tree interpolation.

Theory of linear rational arithmetic with uninterpreted functions(QF_UFLRA): We assume countably many variables X, with $x \in X$, function symbols $\mathcal{F}$, with $f \in \mathcal{F}$, predicate symbols $\mathcal{P}$, with $P \in \mathcal{P}$, and rationals $\mathbb{Q}$, with $c \in \mathbb{Q}$. Let the arity of function and predicate symbols be encoded in their names. The following grammar defines the syntax of the formulas in QF_UFLRA.

$$
\begin{array}{lrl}
\text{terms} & \ni t ::= & v \mid cv \mid f(t, \ldots, t) \mid t + t \mid t - t \mid c \\
\text{atoms} & \ni a ::= & P(t, \ldots, t) \mid t = t \mid t \leq 0 \\
\text{formulas} & \ni \phi ::= & a \mid \neg\phi \mid \phi \vee \phi
\end{array}
$$

Let $\phi_1 \wedge \phi_2$ be summary of $\neg(\neg\phi_1 \vee \neg\phi_2)$. Let $-t$ be summary of $0 - t$ and $s \leq t$ be summary of $s - t \leq 0$. Let *true* be summary of $\phi \vee \neg\phi$ and *false* be summary of $\neg$*true*. Let $\mathbb{B} = \{true, false\}$.

A *literal* is an atom or its negation. Let l be a literal. If $l = \neg a$ then let $\neg l = a$. Let $atom(l)$ be the atom in l. A *clause* is a set of literals. A clause is interpreted as the disjunction of its literals. Naturally, empty clause $\emptyset$ denotes *false*. Let C and D be clauses. Let $C \vee D$ denote union of the clauses, and let $s \vee C$ denote $\{s\} \vee C$. A *conjunctive formula* is negation of a clause. For example, $\neg C$ is a conjunctive formula. A *CNF formula* is a set of clauses. A CNF formula is interpreted as the conjunction of its clauses. Since any formula can be converted into a CNF formula, we will assume that all the formulas in this paper are CNF formulas. Let ϕ and ψ be CNF formulas/clauses/literals. Let $symb(\phi)$ be the set of variables, uninterpreted functions, and uninterpreted predicates occurring in ϕ. Let $\phi \preceq \psi$ iff $symb(\phi) \subseteq symb(\psi)$. For clause C, let $C|_\phi = \{l \in C | l \preceq \phi\}$. Let $Atoms(\phi)$ be the set of atoms that appear in ϕ. Let $Lits(\phi) = \{a, \neg a | a \in Atoms(\phi)\}$.

$$\textsc{HypC}\frac{}{\neg\{\neg a\} \vdash a}\, a \in \text{Atoms}$$

$$\textsc{Sym}\frac{\neg C \vdash s = t}{\neg C \vdash t = s} \qquad \textsc{Tra}\frac{\neg C_1 \vdash r = s \quad \neg C_2 \vdash s = t}{\neg(C_1 \vee C_2) \vdash r = t}$$

$$\textsc{Con}\frac{\neg C_1 \vdash s_1 = t_1 \ \dots \ \neg C_n \vdash s_n = t_n}{\neg(C_1 \vee \cdots \vee C_n) \vdash f(s_1, \dots, s_n) = f(t_1, \dots, t_n)}$$

$$\textsc{PCon}\frac{\neg C \vdash P(s_1, \dots, s_n) \ \ \neg C_1 \vdash s_1 = t_1 \ \dots \ \neg C_n \vdash s_n = t_n}{\neg(C \vee C_1 \vee \cdots \vee C_n) \vdash P(t_1, \dots, t_n)}$$

$$\textsc{Comb}\frac{\neg C_1 \vdash r \leq 0 \quad \neg C_2 \vdash s \leq 0}{\neg(C_1 \vee C_2) \vdash c_1 r + c_2 s \leq 0}\, c_1, c_2 > 0$$

$$\textsc{LeEq}\frac{\neg C_1 \vdash r \leq s \quad \neg C_2 \vdash s \leq r}{\neg(C_1 \vee C_2) \vdash r = s} \qquad \textsc{EqLe}\frac{\neg C \vdash r = s}{\neg C \vdash r \leq s}$$

$$\textsc{AxiGen}\frac{\neg C \vdash a}{\vdash C \vee a}$$

(a) Proof rules for proving axioms in $\mathcal{T}_u$.

$$\textsc{Hyp}\frac{}{\phi \vdash C}\, C \in \phi,\ \phi \in \text{CNF} \qquad \textsc{Axi}\frac{\vdash C}{\phi \vdash C}$$

$$\textsc{Res}\frac{\phi \vdash a \vee C \quad \phi \vdash \neg a \vee D}{\phi \vdash C \vee D}$$

(b) Proof rules for proving formula ϕ unsatisfiable

Fig. 2. Sound and complete proof rules for the theory of linear rational arithmetic with uninterpreted functions

Unsatisfiability proof: We consider the usual semantics of QF_UFLRAformulas. The problem of proving unsatisfiability of the formulas is decidable. In figure 2, we present a set of sound and complete proof rules for proving unsatisfiability of formula ϕ. For formulas/clauses ϕ and ψ, let $\phi \vdash \psi$ means that ϕ implies ψ. We derive this entailment relation by applying the proof rules. In an application of a rule, the entailment relations above the line are called *antecedents* and the entailment relation below the line is called *consequent*.

We expect to obtain a proof from an SMT solver [9, 7]. SMT solvers prove unsatisfiability using *conflict driven clause learning* with the support of theory solvers. This method of proving separates the theory specific reasoning from the reasoning over the Boolean structure of the formula. Therefore, we have divided the proof rules in two sub-figures 2(a) and 2(b).

The rules in figure 2(a) encodes the theory level reasoning. The applications of rules HypC, Sym, Tra, Con, PCon, Comb, LeEq, and EqLe derive the entailment relations between conjunctive formulas and atomic formulas. Using these derived entailment

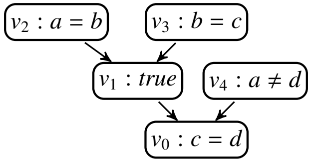


Fig. 3. An example of tree interpolation problem. $\mathcal{I} = (V, \pi, \alpha, v_0)$, where $V = \{v_0, v_1, v_2, v_3, v_4\}$, π is defined by the edges, α is also defined by the labels in the nodes.

relations, an application of rule AxiGen derives valid clauses, which are called *theory axioms*.

The rules in figure 2(b) encodes reasoning over boolean structure. If a clause C appears in formula ϕ then an application of rule Hyp derives $\phi \vdash C$. An application of rule Axi derives $\phi \vdash C$, where C is a theory axiom. Rule Res applies the resolution principle, i.e., clauses $a \vee C$ and $\neg a \vee D$ implies $C \vee D$.

A *proof* is a directed acyclic graph of the derivations of consequences. ϕ is unsatisfiable if applications of these rules can derive $\phi \vdash \emptyset$. The solving architecture of SMT solvers ensures that if a theory axiom C appears in a proof of unsatisfiability of formula ϕ then $Lits(C) \subseteq Lits(\phi)$. Therefore, the atom a appearing in the rules HypC and AxiGen is in $Lits(\phi)$.

2.1 Tree interpolation

A tree interpolation problem is formally defined as follows.

Definition 1 (Tree interpolation problem). *A tree interpolation problem is a labeled tree $\mathcal{I} = (V, \pi, \alpha, v_0)$, where V is a finite set of nodes, π is a map from the nodes to their parent nodes, α is a map from nodes to* QF_UFLRA *formulas (labels), and v_0 is the root of the tree($\pi(v_0) = \bot$). Furthermore, $\bigwedge_{v \in V} \alpha(v)$ is unsatisfiable.*

Example 1. Consider an example of the tree interpolation problem in figure 3.

Let π^* be the transitive closure of π. Let $chidren(v) = \{v' | v = \pi(v')\}$. Let $children^*$ be the transitive closure of $children$. For $v, w \in V$, let $path(v, w)$ be the sequence of nodes that connect v and w in $\mathcal{I}$. Since $\mathcal{I}$ is a tree, $path(v, w)$ is always unique. Let $leastAncestor(v, w)$ be the node $u \in V$ such that $u \in \pi^*(v) \cap \pi^*(w)$ and $\forall u' \in \pi^*(v) \cap \pi^*(w).\ u' \in \pi^*(u)$. Let $\alpha^{in}(v) = \bigwedge_{v' \in children^*(v)} \alpha(v')$ and $\alpha^{out}(v) = \bigwedge_{v' \notin children^*(v)} \alpha(v')$. Let $shared(v) = symb(\alpha^{in}(v)) \cap symb(\alpha^{out}(v))$.

Definition 2 (Tree interpolant). *Let I be a map from V to formulas. I is a* tree interpolant *of $\mathcal{I}$ if*

- $\forall v.\ \alpha(v) \wedge \bigwedge_{v' \in children(v)} I(v') \to I(v)$,
- $I(v) \preceq shared(v)$, *and*
- $I(v_0) = false$.

In the next section, we will present a method of computing tree interpolants for tree interpolation problems.

3 Tree interpolation using localized proofs

In this section, we will present an algorithm to transform a proof of unsatisfiability of $\bigwedge_{v \in V} \alpha(v)$ in the theory of QF_UFLRA into a *localized proof* and computing a tree interpolant using the localized proof. The first objective of the localization is to avoid theory level reasoning during computation of the tree interpolant. To achieve this, the localization adds each theory axiom appearing in the proof to the label of some node of $\mathcal{I}$ without modifying the set of valid tree interpolants for $\mathcal{I}$. The localization rewrites the proof to obtain theory axioms that allow such addition. The second objective of the localization is to rewrite resolution proof in a way such that computing tree interpolants from the localized proof becomes trivial.

Localized proof: For $v \in V$, let $scope(v) = symb(\alpha(v)) \bigcup_{v' \in children(v)} shared(v')$. For a formula ψ, let $\psi \preceq v$ be a shorthand for $\psi \preceq scope(v)$. We say atom a (clause C) is *localized* if there is a node v such that $a \preceq v$ ($C \preceq v$). We say $\phi \vdash \psi$ is *localized* if ψ is localized. We say a proof step is *localized* if there is a node v such that all the antecedents and consequent are localized with respect to v. A (sub-)proof is *localized* if each proof step in the (sub-)proof is localized.

For formulas ψ and ϕ, let $\psi \equiv \phi$ if there exists a $w \in V$ such that $\psi \preceq w$ and $\phi \preceq w$. Let $scope^{-1}(\psi) = \{v | \psi \preceq v\}$. Due to the definition of *scope*, the nodes in $scope^{-1}(\psi)$ will form a tree embedded in $\mathcal{I}$. Let $maxscope(\psi)$ denote the root of $scope^{-1}(\psi)$. Let $\phi \sqsubseteq \psi$ if $maxscope(\psi) \in \pi^*(maxscope(\phi))$.

Lemma 1. *For a formula ψ, $scope^{-1}(\psi)$ forms a tree embedded in $\mathcal{I}$.*

3.1 Localizing theory axiom proofs

In this section, we will present a procedure that localizes the sub-proofs of theory axioms.

In figures 4 and 5, we present the transformation rules to obtain the localized proof steps in the sub-proofs of theory axioms. In the figures, we have abbreviated the entailment relations in the proof rules from figure 2(a), but their expansions should be obvious from the context. These transformations are applicable only if a proof step uses a non-localized antecedent or produces a non-localized consequent. If an non-localized atom $r = t$ is derived in a proof then the transformation rules remember an additional *ghost* proof step

$$* \frac{r = p_1 = \cdots = p_n = t}{r = t},$$

where $n > 0$, $r \equiv p_1$, $p_n \equiv t$, and $\forall i \in 2..n.\ p_{i-1} \equiv p_i$. If a non-localized atom $r \leq 0$ is derived then the transformation rules remember an additional *ghost* proof step

$$* \frac{L}{r \leq 0},$$

where L is a map from V to terms such that $L(v_0) = r$, and for each $v \in V$, $L(v) - \sum_{v' \in children(v)} L(v') \preceq v$. The ghost proof steps for non-localized consequents guide our

Proof steps	Conditions to apply transformation	The consequent ghost step and supporting proof steps
$\textsc{Tra}\,\frac{r = s \quad s = t}{r = t}$	$r \not\equiv t$	$*\,\frac{r = s = t}{r = t}$
	$r \not\equiv s \quad *\,\frac{r = p_1 = \cdots = p_n = s}{r = s}$	$*\,\frac{r = p_1 = \cdots = p_n = s = t}{r = t}$
	$s \not\equiv t \quad *\,\frac{s = p_1 = \cdots = p_n = t}{s = t}$	$*\,\frac{r = s = p_1 = \cdots = p_n = t}{r = t}$
	$r \not\equiv s \quad *\,\frac{r = p_1 = \cdots = p_{n'} = s}{r = s}$ $s \not\equiv t \quad *\,\frac{s = p_{n'+1} = \cdots = p_n = t}{s = t}$	$*\,\frac{r = p_1 = \cdots = p_{n'} = s = p_{n'+1} = \cdots = p_n = t}{r = t}$
$\textsc{Sym}\,\frac{t = r}{r = t}$	$*\,\frac{t = p_1 = \cdots = p_n = r}{t = r}$	$\frac{p_n = r}{r = p_n} \quad \ldots \quad \frac{t = p_1}{p_1 = t} \quad *\,\frac{r = p_n = \cdots = p_1 = t}{r = t}$
$\textsc{Con}\,\frac{\bar{r} = \bar{s}}{f(\bar{r}) = f(\bar{s})}$ $f(\bar{r}) \le v$ $f(\bar{s}) \le w$	$\bar{r} := p_{10}, \ldots, p_{m0} \quad \bar{s} := p_{1n_1}, \ldots, p_{mn_m}$ $\forall k \in 1..m.\ *\,\frac{p_{k0} = \cdots = p_{kn_k}}{p_{k0} = p_{kn_k}}\dagger$ Let $n := \sum_k^m n_k$ and $\bar{r}_1, \ldots \bar{r}_n$ such that $\bar{r}_0 := \bar{r}$ and $\forall i \in 1..n\ \exists k \in 1..m\ \exists j_1 \le n_1 \ldots \exists j_k < n_k \ldots \exists j_m \le n_m.$ $\bar{r}_{i-1} := p_{1j_1}, \ldots, p_{kj_k}, \ldots, p_{mj_m} \wedge \bar{r}_i := p_{1j_1}, \ldots, p_{k(j_k+1)}, \ldots, p_{mj_m} \wedge$ $f(\bar{r}_{i-1}) \equiv f(\bar{r}_i)$	$\forall i \in 1..n\,\frac{\bar{r}_{i-1} = \bar{r}_i}{f(\bar{r}_{i-1}) = f(\bar{r}_i)} \quad *\,\frac{f(\bar{r}_0) = \cdots = f(\bar{r}_n)}{f(\bar{r}_0) = f(\bar{r}_n)}$
$\textsc{PCon}\,\frac{\bar{r} = \bar{s} \quad P(\bar{r})}{P(\bar{s})}$ $P(\bar{r}) \le v \quad P(\bar{s}) \le w$		$\forall i \in 1..n\,\frac{\bar{r}_{i-1} = \bar{r}_i \quad P(\bar{r}_{i-1})}{P(\bar{r}_i)}$
$*\,\frac{p_1 = \cdots = p_n}{p_1 = p_n}$	$p_i \equiv p_{i+2}$	$\frac{p_i = p_{i+1} \quad p_{i+1} = p_{i+2}}{p_i = p_{i+2}} \quad *\,\frac{p_1 = \cdots = p_i = p_{i+2} = \cdots = p_n}{p_1 = p_n}$

Fig. 4. Transformation Rules for cleaning theory axiom sub-proofs. †By abuse of notation, $n_k = 1$ implies that there is no ghost step for $p_{k0} = p_{kn_k}$ and $n_k = 0$ implies that p_{k0} and p_{kn_k} are the same term.

Proof steps	Conditions to apply transformation	The consequent ghost step and supporting proof steps
$\textsc{Comb}\dfrac{r \le 0 \quad t \le 0}{c_1 r + c_2 t \le 0}$	$r \preceq u \quad t \preceq w \quad u \ne w$	$\dfrac{r \le 0}{c_1 r \le 0}$ $\dfrac{t \le 0}{c_2 t \le 0}$ $*\dfrac{\bar{0}[u \mapsto c_1 r] + \bar{0}[w \mapsto c_2 t]}{c_1 r + c_2 t \le 0}$
	$*\dfrac{L}{r \le 0} \quad t \preceq w$	$\dfrac{L(w) \le 0 \quad t \le 0}{c_1 L(w) + c_2 t \le 0}$ $\forall w' \in V \dfrac{L_1(w') \le 0}{c_1 L_1(w') \le 0}$ $*\dfrac{c_1 L + \bar{0}[w \mapsto c_2 t]}{c_1 r + c_2 t \le 0}$
	$*\dfrac{L_1}{r \le 0} \quad *\dfrac{L_2}{t \le 0}$	$\forall w \in V \dfrac{L_1(w) \le 0 \quad L_2(w) \le 0}{c_1 L_1(w) + c_2 L_2(w) \le 0}$ $*\dfrac{c_1 L_1 + c_2 L_2}{c_1 r + c_2 t \le 0}$
$\textsc{EqLe}\dfrac{p_1 = p_n}{p_1 - p_n \le 0}$	$*\dfrac{p_1 = \cdots = p_n}{p_1 = p_n} \quad \forall i \in 2..n.\ p_{i-1} - p_i \preceq v_i$	$\forall i \in 2..n \dfrac{p_{i-1} = p_i}{p_{i-1} - p_i \le 0}$ $*\dfrac{\bar{0}[v_2 \mapsto p_1 - p_2] + \cdots + \bar{0}[v_n \mapsto p_{n-1} - p_n]}{r - t \le 0}$
$\textsc{LeEq}\dfrac{r \le t \quad t \le r}{r = t}$	$*\dfrac{L_1}{r \le t} \quad *\dfrac{L_2}{t \le r}$ Let $v_1, \dots, v_{n+1} \in V$ be such that $v_1 \ne v_{n+1}, r \preceq v_1, t \preceq v_{n+1}$, $v_k := leastAncestor(v_1, v_{n+1})$, $v_1, \dots, v_k := path(v_1, v_k)$, and $v_k, \dots, v_{n+1} := path(v_k, v_{n+1})$. Let terms $p_0, \dots, p_{n+1}, q_0, \dots, q_{n+1}$ be defined as follows. $p_0 := q_0 := r \qquad p_{n+1} := q_{n+1} := t$ $\forall i \in 1..(k-1). \quad p_i := r - L_1(v_i) \quad q_i := r + L_2(v_i)$ $\forall i \in k..n. \quad p_i := t + L_1(v_{i+1}) \quad q_i := t - L_2(v_{i+1})$	$\dfrac{L_1^{v_1}}{p_0 \le p_1}$ $\dfrac{L_2^{v_1}}{q_1 \le q_0}$ $\forall i \in 2..(k-1)$ $\dfrac{L_1^{v_i} - L_1^{v_{i-1}}}{p_{i-1} \le p_i}$ $\dfrac{L_2^{v_i} - L_2^{v_{i-1}}}{q_i \le q_{i-1}}$ $\dfrac{L_1 - L_1^{v_{k-1}} - L_1^{v_{k+1}}}{p_{k-1} \le p_k}$ $\dfrac{L_2 - L_2^{v_{k-1}} - L_2^{v_{k+1}}}{q_k \le q_{k-1}}$ $\dfrac{L_1^{v_{n+1}}}{p_n \le p_{n+1}}$ $\dfrac{L_2^{v_{n+1}}}{q_{n+1} \le q_n}$ $\forall i \in (k+1)..n$ $\dfrac{L_1^{v_i} - L_1^{v_{i-1}}}{p_{i-1} \le p_i}$ $\dfrac{L_2^{v_i} - L_2^{v_{i-1}}}{q_i \le q_{i-1}}$ $\forall i \in 1..n$ $\dfrac{q_i \le q_{i-1} \quad q_{i-1} \le p_{i-1} \quad p_{i-1} \le p_i}{q_i \le p_i}$ $\dfrac{p_i \le p_{i+1} \quad p_{i+1} \le q_{i+1} \quad q_{i+1} \le q_i}{p_i \le q_i}$ $\forall i \in 0..n$ $\dfrac{p_{i+1} \le q_{i+1} \quad q_{i+1} \le q_i \quad q_i \le p_i}{p_{i+1} \le p_i}$ $\dfrac{p_{i+1} \le p_i \quad p_i \le p_{i+1}}{p_i = p_{i+1}}$ $*\dfrac{p_0 = p_1 = \cdots = p_n = p_{n+1}}{p_0 = p_{n+1}}$
If $*\dfrac{L}{r \le 0}$ and $\exists w \in V$ such that $r \preceq w$ then we remove the ghost step and replace the derivation of $r \le 0$ by $\dfrac{L}{r \le 0}$.		

Fig. 5. Transformation Rules for cleaning theory axiom sub-proofs.

transformation rules to obtain a localized sub-proof of theory axioms whose sub-proof contains the non-localized consequents.

Each row in the figures corresponds to a transformation rule. These transformation rules are applied in the topological order of the sub-proof of an axiom. For each row, the first and second columns present the proof step and the condition in which the transformation rule is applied. If a ghost step is to be generated then it is presented in the third column. In some transformations, the proofs of antecedents of the generated ghost step may not already exist in the proof, therefore some additional localized proof steps are added, which are also presented in the third column.

For simplicity of the presentation, the transformation rules do not check immediately if a generated ghost step can be further simplified or removed. In that case, we transform or remove the ghost step according to the last rows of the figures.

In figure 4, the first transformation rule for Tra is applied if both the antecedents are localized with respect to the different nodes. The consequent potentially be non-localized therefore a ghost proof step is created. If any of the antecedents of the Tra is non-localized then the rest of the cases of the Tra are applied. Using the ghost steps of the antecedents, a ghost step corresponding to the consequent is generated. Note that we are using the existence of the ghost steps to detect the non-localized antecedents. For each non-localized antecedent, there must exist a ghost step, since we apply these transformation rules in the topological order.

For Sym, if the consequent is non-localized then a ghost step is generated with the reverse order of the terms with respect to the antecedent ghost step. Since this newly generated ghost step has equalities whose proof may not already exist in the proof therefore the additional proofs are also generated.

For Con, a sequence of vectors of terms $\bar{r}_0, \ldots, \bar{r}_n$ is computed such that $\forall i \in 1..n.\ \bar{r}_{i-1} \equiv \bar{r}_i$. Due to the definition of $\equiv$, such sequence of vectors always exist. Since two consecutive vectors $\bar{r}_{i-1}$ and $\bar{r}_i$ exactly differs at a single index, the proof of $\bar{r}_{i-1} = \bar{r}_i$ must exist. The transformation rule applies Con rule on $\bar{r}_{i-1} = \bar{r}_i$ to obtain a localized proof step for $f(\bar{r}_{i-1}) = f(\bar{r}_i)$. Using $f(\bar{r}_{i-1}) = f(\bar{r}_i)$'s, we obtain a ghost step for the consequent. The transformation rule for PCon is similar.

If the equality ghost steps can be simplified then the simplification is eagerly applied according to the last row of the figure. Due to the definition of localization, if the consequent of a ghost step is localized then the ghost step must get simplified to a single equality in the antecedent. In the case, we remove such trivial ghost steps from the proof. One more thing to note is that in the antecedent of a ghost step the chain can not grow longer than $|V|$. Otherwise, at least two terms in the chain are localized with respect to the same node and the ghost step can be simplified further.

In figure 5, we present transformation rules to localize linear arithmetic derivations. To understand the transformation rules, lets introduce some notations. Let L be a map from V to terms. Let $\bar{0}$ be a map from V to 0s. Let $\bar{0}[w \mapsto t]$ be a map that maps a node v to t if $v \in \pi^*(w)$, otherwise to 0. A ghost step represents $*\frac{L}{r\leq 0}$ a derivation of $r \leq 0$, which we denote simply by $\frac{L}{r\leq 0}$. For each $v \in V$, $\frac{L}{r\leq 0}$ contains a proof step

$$\textsc{Comb}\,\frac{L(v) \leq L(v_1) + \cdots + L(v_k) \quad L(v_1) \leq 0 \ \ldots \ L(v_k) \leq 0}{L(v) \leq 0}\{v_1, \ldots, v_k\} = \mathit{children}(v).$$

Let $L := c_1L_1 + c_2L_2$ if for for each $w \in V$ $L(w) := c_1L_1(w)+c_2L_2(w)$. For a node $v \in V$, let L^v be a map from V to terms such that for each $w \in V$

$$L^v(w) = \begin{cases} L(w) & w \in \mathit{children}^*(v) \\ L(v) & w \in \pi^*(v) \\ 0 & \text{otherwise.} \end{cases}$$

The first transformation rule for Comb is applied if both the antecedents are localized with respect to the different nodes. The consequent linear combination may not be localized therefore a ghost step is created, whose antecedent map L records for each node the contribution of the subtree of the node.

The transformation rule for the EqLe is applied if its consequent is non-localized. For each equality appearing in the equality chain in the ghost step corresponding to the antecedent, we derive a localized inequality. Each derived inequality is added in the ghost step for the consequent such that their contribution is considered at a node with respect to which the inequality is localized.

The transformation rule for the LeEq is also applied if its consequent is non-localized. The rule finds a sequence of terms p_0 to p_{n+1} such that $r := p_0$, $t := p_{n+1}$, and the consecutive terms in the sequence are mutually localized and can be proved equal. The rule finds the sequence in the following way. r is localized with respect to v_1 and t is localized with respect to v_{n+1}. The sequence of nodes $v_1, \ldots, v_{n+1}$ is a path between v_1 and v_{n+1} in $\mathcal{I}$. v_k is the node closet to v_0. Now $p_0, \ldots, p_n$ and $q_0, \ldots, q_n$ are defined in the rule. In the third column, the first three lines of the additional localized proof steps prove $p_0 \leq \cdots \leq p_{n+1}$ and $q_0 \geq \cdots \geq q_{n+1}$. Using these chains of inequalities, the next two lines of the additional localized proof steps prove $p_0 = \cdots = p_{n+1}$. We obtained the desired ghost step.

After exhaustive application of these transformation rules the sub-proofs of the theory axioms are localized.

Theorem 1. *The sub-proofs of theory axioms are localized after applying the transformation rules of figure 4 and 5.*

The proof of the above theorem is riddled with hairy details. We do not provide the proof in this version of the paper.

Example 2. Consider the example from figure 3. The following is a fragment of the proof of the unsatisfiability of the conjunctions of the labels.

$$\textsc{Tra}\frac{a = b \qquad \textsc{Tra}\dfrac{b = c \quad c = d}{b = d}}{a = d}$$

$b = d$ is not localized with respect to any node in the problem. On encounter with the first proof step, our transformation rules will create the following ghost step.

$$*\frac{b = c = d}{b = d}$$

$$
\textsc{Axi}\,\dfrac{\textsc{AxiGen}\,\dfrac{\cdots\dfrac{\neg C_1 \vdash a_1 \ldots \neg C_m \vdash a_m}{\neg(C_1 \lor \cdots \lor C_m) \vdash a}}{\vdash C_1 \lor \cdots \lor C_m \lor a}}{\phi \vdash C_1 \lor \cdots \lor C_m \lor a} \qquad \exists j \in 1..m.\ C_j \neq \{\neg a_j\}
$$

$$
\rightsquigarrow
$$

$$
\textsc{Res}\,\dfrac{\textsc{Axi}\,\dfrac{\textsc{AxiGen}\,\dfrac{\cdots\dfrac{\neg C_1 \vdash a_1 \ldots \neg C_{j-1} \vdash a_{j-1} \quad \neg\{\neg a_j\} \vdash a_j \quad \neg C_{j+1} \vdash a_{j+1} \ldots \neg C_m \vdash a_m}{\neg(C_1 \lor \cdots \lor C_{j-1} \lor \neg a_j \lor C_{j+1} \lor \cdots \lor C_m) \vdash a}}{\vdash C_1 \lor \cdots \lor C_{j-1} \lor \neg a_j \lor C_{j+1} \lor \cdots \lor C_m \lor a}}{\phi \vdash C_1 \lor \cdots \lor C_{j-1} \lor \neg a_j \lor C_{j+1} \lor \cdots \lor C_m \lor a} \quad \textsc{Axi}\,\dfrac{\textsc{AxiGen}\,\dfrac{\neg C_j \vdash a_j}{\vdash C_j \lor a_j}}{\phi \vdash C_j \lor a_j}}{\phi \vdash C_1 \lor \cdots \lor C_m \lor a}
$$

Fig. 6. Transformation rule to convert application of a theory rule into a resolution step.

Next our transformation rules will be applied to the second proof step and it will create the following ghost step.

$$
*\,\dfrac{a = b = c = d}{a = d}
$$

Now observe that $a \equiv c$. Therefore, the ghost step simplification rule will be applied and the following ghost step and proof step are created.

$$
*\,\dfrac{a = c = d}{a = d} \qquad \textsc{Tra}\,\dfrac{a = b \quad b = c}{a = c}
$$

Since $a \equiv d$, the ghost step simplification rule will be applied and the following the proof steps are created.

$$
\textsc{Tra}\,\dfrac{\textsc{Tra}\,\dfrac{a = b \quad b = c}{a = c} \quad c = d}{a = d}
$$

The above localized proof fragment replaces the original proof fragment.

3.2 Localizing theory axioms

In figure 6, we present a proof transformation rule to convert the applications of theory rules into resolution steps in the reverse topological order. This transformation rule is applicable if the theory proof step just before an AxiGen step contains an antecedent $\neg C_j \vdash a_j$ such that $C_j \neq \{\neg a_j\}$. The transformation splits the theory axiom generated in the AxiGen step into two simpler axioms such that a resolution step between them obtains the original axiom. After applying this transformation rule exhaustively, the resulting proof will have theory axioms that are obtained by a single theory proof step.

$$\textsc{Res}\,\dfrac{\textsc{Res}\,\dfrac{a \lor b \lor D \qquad \neg a \lor E}{b \lor D \lor E} \qquad \neg b \lor C}{C \lor D \lor E} \;\rightsquigarrow\; \textsc{Res}\,\dfrac{\textsc{Res}\,\dfrac{a \lor b \lor D \qquad \neg b \lor C}{g \lor C \lor D} \qquad \neg a \lor E}{C \lor D \lor E}$$

$$\textsc{Res}\,\dfrac{\textsc{Res}\,\dfrac{a \lor b \lor D \qquad \neg a \lor l \lor E}{b \lor D \lor E} \qquad \neg b \lor C}{C \lor D \lor E} \;\rightsquigarrow\; \textsc{Res}\,\dfrac{\textsc{Res}\,\dfrac{a \lor b \lor D \quad \neg b \lor C}{g \lor C \lor D} \qquad \textsc{Res}\,\dfrac{\neg a \lor l \lor E \quad \neg b \lor C}{\neg a \lor C \lor E}}{C \lor D \lor E}$$

Fig. 7. If pivot b occurs immediately after pivot a in a proof and $b \sqsubseteq a$ then we can locally rewrite the proofs using one of the above two transformation rules. After the transformation, the proof resolves first using b then using a.

Since we have localized all the theory proof steps, the resulting proof will contain only localized axioms.

Due to the definition of *scope*, a localized axiom with respect of some node v will contain symbols that are either shared with respect to one of v's children or appear in $\alpha(v)$. Therefore, we replace $\alpha(v)$ with the conjunction of the axiom and $\alpha(v)$ without modifying the set of valid tree interpolants of $\mathcal{I}$. After exhaustive addition of the axioms into the labels, we obtain a pure resolution proof without any theory axioms.

3.3 Localize resolution proofs

To obtain a localized resolution proof, we traverse the proof in the topological order. Each time we encounter a consecutive pair of resolution steps with pivots a and b such that $b \sqsubseteq a$, we apply one of the two transformation rules presented in figure 7 depending on the matching pattern. These two transformation rules are the standard pivot reordering rules from [8]. Note that these rules assume that the proof is redundancy free, which can be achieved by the algorithms presented in [12]. Note that the $\sqsubseteq$ is a partial order. Since we apply this transformation in the topological order, a clause from outside of the subtree rooted at $maxscope(a)$ can not be part of the sub-proof that derives $b \lor D \lor E$. Therefore, $b \lor D \lor E$ can only contain symbols from $symb(\alpha^{in}(maxscope(a)))$. Therefore, a and b will be ordered one way or another. After exhaustive application of these transformation rules, we obtain localized resolution proofs.

3.4 Tree interpolant via localized proof

A tree interpolant for $\mathcal{I}$ is trivially encoded in the localized resolution proof obtained in the previous sub-section. For a $v \in V$, let a clause C be v-derived if C is derived only from the clauses that appear in the labels of the nodes from $children^*(v)$. We say C is maximally v-derived if there exist at least one proof step in which C occurs as an antecedent and the other antecedent in the proof step is not v-derived.

Theorem 2. *The tree interpolant $I(v)$ is the conjunction of all the maximally v-derived clauses appearing in the resolution proof.*

Proof sketch. Let C be a maximally v-derived clause. Since there is a proof step in which C is co-antecedent with a clause that is not v-derived, C can not contain a literal that is in $symb(\alpha^{in}(v)) \setminus symb(\alpha^{out}(v))$. Otherwise due to an inductive argument, there exist another resolution step in subsequent derivations after the proof step that removes the literal, which will violate the $\sqsubseteq$ ordering over pivots. Let v' be the parent of v. The v'-derived clauses contains the v-derived clauses. Therefore, a maximally v'-derived clause must be implied by the conjunction of the maximally derived clauses of its children. Hence $I(v)$ satisfies the conditions of tree interpolation. □

Using the above theorem, we obtain the tree interpolants.

4 Conclusion

We presented a novel method of computing tree interpolants using the localized proofs. We presented a three step procedure to obtain the localized proofs. The first step localizes the theory axiom proofs. The second step removes the theory axiom proofs and replaces it with the resolution proofs. The third step reorders the resolution proofs to localize the complete proofs. From these localized proofs, we trivially learn tree interpolants.

Our resolution proof localization follows the similar approach of [4] for interpolation. The resolution proof localization is not necessary if one aims to use ɪZ3 like scheme to compute tree interpolants, because our theory axiom proof localization step will remove the non-localized theory terms that will enable ɪZ3 to compute tree interpolants without the help of external interpolation tools and resolution proof localization.

For the future work, we aim to extend the theory aware localization rules for more theories, e.g., the theory of integer arithmetic and arrays.

References

1. C. Barrett and C. Tinelli. CVC3. In *CAV*, volume 4590, pages 298–302. Springer-Verlag, 2007.
2. A. Brillout, D. Kroening, P. Rümmer, and T. Wahl. Beyond quantifier-free interpolation in extensions of presburger arithmetic. In *VMCAI*. Springer, 2011.
3. R. Bruttomesso, S. Ghilardi, and S. Ranise. Rewriting-based quantifier-free interpolation for a theory of arrays. In *RTA*. Schloss Dagstuhl, 2011.
4. R. Bruttomesso, S. Rollini, N. Sharygina, and A. Tsitovich. Flexible interpolation with local proof transformations. In *ICCAD*, pages 770–777. IEEE, 2010.
5. A. Cimatti, A. Griggio, and R. Sebastiani. Efficient generation of Craig interpolants in satisfiability modulo theories. *ACM Trans. Comput. Logic*, 12, November 2010.
6. W. Craig. Linear reasoning. a new form of the herbrand-gentzen theorem. *The Journal of Symbolic Logic*, 22(3):pp. 250–268, 1957.
7. L. M. de Moura and N. Bjørner. Z3: An efficient SMT solver. In *TACAS*, 2008.
8. V. D'Silva, D. Kroening, M. Purandare, and G. Weissenbacher. Interpolant strength. In *VMCAI*, 2010.
9. H. Ganzinger, G. Hagen, R. Nieuwenhuis, A. Oliveras, and C. Tinelli. DPLL(T): Fast decision procedures. In *CAV*, pages 175–188. Springer, 2004.

10. S. Grebenshchikov, N. P. Lopes, C. Popeea, and A. Rybalchenko. Synthesizing Software Verifiers from Proof Rules. In *PLDI*, June 2012.
11. A. Griggio, T. T. H. Le, and R. Sebastiani. Efficient interpolant generation in satisfiability modulo linear integer arithmetic. In *TACAS*, pages 143–157. Springer, 2011.
12. A. Gupta. Improved single pass algorithms for resolution proof reduction. In *ATVA*. Springer, 2012.
13. A. Gupta, C. Popeea, and A. Rybalchenko. Predicate abstraction and refinement for verifying multi-threaded programs. In *POPL*, 2011.
14. A. Gupta, C. Popeea, and A. Rybalchenko. Solving recursion-free horn clauses over li+uif. In *APLAS*, pages 188–203. Springer, 2011.
15. M. Heizmann, J. Hoenicke, and A. Podelski. Nested interpolants. In *POPL*, 2010.
16. T. A. Henzinger, R. Jhala, R. Majumdar, and K. L. McMillan. Abstractions from proofs. In *POPL*, 2004.
17. K. Hoder, L. Kovács, and A. Voronkov. Playing in the grey area of proofs. In *POPL*, pages 259–272. ACM, 2012.
18. R. Jhala and K. L. McMillan. A practical and complete approach to predicate refinement. In *TACAS*. Springer, 2006.
19. D. Kapur, R. Majumdar, and C. G. Zarba. Interpolation for data structures. In *SIGSOFT FSE*, pages 105–116. ACM, 2006.
20. L. Kovács and A. Voronkov. Interpolation and symbol elimination. In *CADE*, pages 199–213. Springer, 2009.
21. D. Kroening, J. Leroux, and P. Rümmer. Interpolating quantifier-free presburger arithmetic. In *LPAR (Yogyakarta)*, pages 489–503. Springer, 2010.
22. K. L. McMillan. Interpolation and sat-based model checking. In *CAV*, 2003.
23. K. L. McMillan. An interpolating theorem prover. *Theor. Comput. Sci.*, 345(1):101–121, 2005.
24. K. L. McMillan. Quantified invariant generation using an interpolating saturation prover. In *TACAS*. Springer, 2008.
25. K. L. McMillan. iz3 documentation, 2009. `http://research.microsoft.com/en-us/um/`
26. K. L. McMillan. Interpolants from z3 proofs. In *FMCAD*, pages 19–27. FMCAD Inc., 2011.