Logical Methods in Computer Science
Vol. 8 (1:06) 2012, pp. 1–23
www.lmcs-online.org

Submitted Nov. 21, 2009
Published Feb. 16, 2012

# GROUND INTERPOLATION FOR THE THEORY OF EQUALITY *

ALEXANDER FUCHS [a], AMIT GOEL [b], JIM GRUNDY [c], SAVA KRSTIĆ [d], AND CESARE TINELLI [e]

[a,e] Department of Computer Science, The University of Iowa
*e-mail address*: cesare-tinelli@uiowa.edu

[b,c,d] Strategic CAD Labs, Intel Corporation
*e-mail address*: {amit1.goel,jim.d.grundy,sava.krstic}@intel.com

Abstract. Given a theory $T$ and two formulas $A$ and $B$ jointly unsatisfiable in $T$, a *theory interpolant* of $A$ and $B$ is a formula $I$ such that (i) its non-theory symbols are shared by both $A$ and $B$, (ii) it is entailed by $A$ in $T$, and (iii) it is unsatisfiable with $B$ in $T$. Theory interpolation has found several successful applications in model checking. We present a novel method for computing interpolants for ground formulas in the theory of equality. The method produces interpolants from colored congruence graphs representing derivations in that theory. These graphs can be produced by conventional congruence closure algorithms in a straightforward manner. By working with graphs, rather than at the level of individual proof steps, we are able to derive interpolants that are pleasingly simple (conjunctions of Horn clauses) and smaller than those generated by other tools. Our interpolation method can be seen as a theory-specific implementation of a cooperative interpolation game between two provers. We present a generic version of the interpolation game, parametrized by the theory $T$, and define a general method to extract runs of the game from proofs in $T$ and then generate interpolants from these runs.

## 1. Introduction

The *Craig Interpolation Theorem* [Cra57] asserts, for every inconsistent pair of first-order formulas $A$, $B$, the existence of a formula $I$ that is implied by $A$, inconsistent with $B$, and written using only logical symbols and symbols that occur in both $A$ and $B$. Analogues of this result hold for a variety of logics and logic fragments. Recently, they have found practical use in symbolic model checking. Applications, starting with the work by McMillan [McM03], involve computation of interpolants in propositional logic or in quantifier-free logics with (combinations of) theories such as the theory of equality, linear rational arithmetic, arrays, and finite sets [McM05b, YM05, KMZ06, CGS08]. There are now several techniques that use interpolants to obtain property-driven approximate reachability sets of

*1998 ACM Subject Classification:* D.2.4, F.3.1, F.4.1, I.2.3.

*Key words and phrases:* Logical Interpolation, Satisfiability Modulo Theories.

[*] Revised and extended version of [FGG$^+$09].

[e] Partially supported AFOSR Grant FA9550-09-1-0517.

LOGICAL METHODS IN COMPUTER SCIENCE DOI:10.2168/LMCS-8 (1:06) 2012

transition relations, or compute refinements for predicate abstraction [McM05a, McM06, JM05, JM06].

An important functionality in much of this work is the computation of ground interpolants in the *theory of equality*, also known as the theory of *uninterpreted functions* (*EUF*). The ground interpolation algorithm for this theory used in existing interpolation-based model checkers was developed by McMillan [McM05b]. It derives interpolants from proofs in a formal system that contains rules for the basic properties of equality.

In this paper, which is a revised and expanded version of [FGG$^+$09], we present a novel method for ground *EUF* interpolation. We compute interpolants from *colored congruence graphs* that compactly represent *EUF* derivations from two sets of equalities, and can be produced in a straightforward manner by conventional congruence closure algorithms, as implemented in solvers for Satisfiability Modulo Theories (*e.g.*, [DNS05, NO05]). Working with graphs makes it possible to exploit the global structure of proofs to streamline interpolant generation. The generated interpolants are conjunctions of Horn clauses, the simplest conceivable form for this theory. In most cases, they are smaller and logically simpler than those produced by McMillan's method.

We restrict ourselves to input formulas $A$ and $B$ that are just conjunctions of *literals.* Such a restriction causes no loss of generality because any interpolation procedure for conjunctions of literals can be extended in a uniform way to arbitrary ground formulas—and under the right conditions also combined with interpolation procedures for other theories [McM05b, CGS08, GKT09].

Our interpolation method can be understood as the implementation of a cooperative interpolation game between two provers. The game is not specific to the theory of equality and can be generalized to other theories. We present a general version of the interpolation game for a theory $\mathcal{T}$ and define a generic method to extract runs of the game from *local* refutations in $\mathcal{T}$ and generate interpolants from these runs.

Our interpolation algorithm for *EUF* is described and proved correct in §4. In §3, we give a series of examples to highlight important aspects of the algorithm. A detailed comparison with McMillan's method is given in §5, together with experimental data on a set of benchmarks derived from those in the SMT-LIB repository [BST11]. The general version of the interpolation game is described and proved correct in §6.

### 1.1. **Formal preliminaries.**

We work in the context of first-order logic with equality, and use standard notions of signature, term, literal, formula, clause, Horn clause, entailment, and so on. We use the symbol $=$ to denote the equality predicate in the logic as well as equality at the meta-level, relying on context to disambiguate the two. For convenience, we treat all equations modulo symmetry, that is, an equation of the form $s = t$ will stand indifferently for $s = t$ or $t = s$. For terms or formulas we will use "ground", *i.e.*, variable-free, and "quantifier-free" interchangeably since for our purposes free variables can be always seen as free constants.

If $S, S_1, \ldots, S_n$ are sets of SENTENCES (*i.e.*, closed formulas) and $\varphi$ is a sentence, we write, as usual, $S_1, \ldots, S_n \models \varphi$ if $S_1 \cup \cdots \cup S_n$ logically entails $\varphi$; we write $S_1, \ldots, S_n \models S$ if $S_1, \ldots, S_n \models \psi$ for all $\psi \in S$. If $\mathcal{T}$ is a theory, understood as a set of sentences, we write $S \models_{\mathcal{T}} \varphi$ as an abbreviation of $\mathcal{T} \cup S \models \varphi$. We use the literals $\mathsf{true}$ and $\mathsf{false}$ as logical constants denoting the universally true and the universally false formula. We say that a set of sentences $S$ is $\mathcal{T}$-UNSATISFIABLE if $S \models_{\mathcal{T}} \mathsf{false}$.

In FOL with equality, for any given signature $\Sigma$ the theory $EUF$ is axiomatized by the empty set of sentences. For convenience then, we write $\models$ in place of $\models_{EUF}$ and write "unsatisfiable" instead of "$EUF$-unsatisfiable" when talking about that theory. Also for convenience, we do not distinguish a finite set of sentences from the conjunction of its elements.

## 2. Ground Theory Interpolation

Interpolation is a property of *logical fragments*, *i.e.*, classes of formulas with an associated *entailment* relation over such formulas. To state it for a fragment $\mathcal{F}$ with entailment relation $\models_{\mathcal{F}}$ we need know only a partition of the symbols used to build formulas in $\mathcal{F}$ into *logical* and *non-logical* symbols.

Let $\mathcal{F}(X)$ be the set of all formulas in $\mathcal{F}$ whose non-logical symbols belong to some set $X$. By definition, $\mathcal{F}$ has the INTERPOLATION PROPERTY if for every $A \in \mathcal{F}(X)$ and $B \in \mathcal{F}(Y)$ such that $A, B \models_{\mathcal{F}} \mathsf{false}$, there exists $I \in \mathcal{F}(X \cap Y)$ such that $A \models_{\mathcal{F}} I$ and $B, I \models_{\mathcal{F}} \mathsf{false}$. The formula $I$ is an ($\mathcal{F}$-)INTERPOLANT of $A$ and $B$. Note the asymmetry: $I$ is not an interpolant of $B$ and $A$; however, $\neg I$ is—provided it belongs to $\mathcal{F}$.

A classic theorem by William Craig [Cra57] states that the fragment of all first-order logic formulas with the standard entailment relation has the interpolation property. (The non-logical symbols are predicate and function symbols, and free variables.) The result also implies a *modulo theory* generalization, where, for a given first-order theory $\mathcal{T}$ over a signature $\Sigma$, the fragment $\mathcal{F}$ is the set of all $\Sigma$-formulas together with the entailment relation $\models_{\mathcal{T}}$, and the symbols of $\Sigma$ are treated as logical. The case where $\mathcal{T}$ and $\Sigma$ are empty is Craig's original theorem.

Of particular interest is the interpolation property for quantifier-free fragments of theories. The property may or may not hold, depending on the theory. Take, for example, the quantifier-free fragment of linear integer arithmetic, and let $A = \{x = 2y\}, B = \{x = 2z{+}1\}$. The set $A \cup B$ is unsatisfiable in this theory, and the formula $\exists u.(x = 2u)$ is an interpolant. However, there is no quantifier-free interpolant for $A$ and $B$.

By definition, a theory has the GROUND INTERPOLATION PROPERTY if its quantifier-free fragment has the interpolation property. Aside from $EUF$, several other theories of interest in model checking have this property, including the theory of rational arithmetic among others [KMZ06, JCG08].

**Example 2.1.** The sets of inequalities $A = \{3x - z - 2 \leq 0, -2x + z - 1 \leq 0\}$ and $B = \{3y - 4z + 12 \leq 0, -y + z - 1 \leq 0\}$ are jointly unsatisfiable in the theory of rational arithmetic, as witnessed by the linear combination with positive coefficients

$$2 \cdot (3x - z - 2 \leq 0) + 3 \cdot (-2x + z - 1 \leq 0) + 1 \cdot (3y - 4z + 12 \leq 0) + 3 \cdot (-y + z - 1 \leq 0)$$

which simplifies to $2 \leq 0$. The $A$-part of this linear combination $2 \cdot (3x - z - 2 \leq 0) + 3 \cdot (-2x + z - 1 \leq 0)$ gives us the interpolant $I = (z - 7 \leq 0)$ for $A, B$. Generalizing what goes on in this example, one can obtain a ground interpolation procedure for the linear arithmetic with real coefficients. See, e.g., [CGS08]. □

By the following lemma, if we want an algorithm for ground $\mathcal{T}$-interpolation, it suffices to have one that works for inputs $A$ and $B$ that are *sets of ground literals.*

**Lemma 2.2.** *Let $\mathcal{T}$ be a theory and suppose every pair of jointly $\mathcal{T}$-unsatisfiable sets of literals has a quantifier-free interpolant. Then, $\mathcal{T}$ has the ground interpolation property.* □

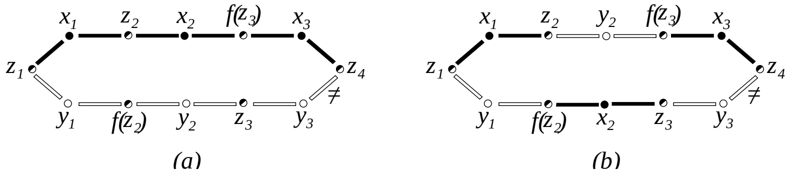

Figure 1: Solid (hollow) edges represent literals from the set $A$ (set $B$) in Example 3.1. For the vertex coloring convention, see Example 4.5.

The reader is referred to [McM05b, CGS08, GKT09] for effective proofs of the lemma—descriptions of a general mechanism to combine interpolation procedures restricted to sets of literals with a method for computing interpolants in propositional logic [Pud97, McM03]. With this justification, our interpolation method for *EUF* focuses on sets of ground literals.

## 3. Interpolation in *EUF*

It is instructive to look first at some examples of interpolants for pairs of literal sets $A$ and $B$ jointly unsatisfiable in *EUF*.

**Example 3.1.** The picture in Figure 1(a) demonstrates the joint unsatisfiability of

$$\begin{array}{rcl} A &=& \{z_1 = x_1,\, x_1 = z_2,\, z_2 = x_2,\, x_2 = f(z_3),\, f(z_3) = x_3,\, x_3 = z_4,\, f(z_2) = x_2,\, x_2 = z_3\}, \\ B &=& \{z_1 = y_1,\, y_1 = f(z_2),\, f(z_2) = y_2,\, y_2 = z_3,\, z_3 = y_3,\, z_2 = y_2,\, y_2 = f(z_3),\, y_3 \neq z_4\} \end{array}$$

which follows by the transitivity of equality. An interpolant is the equality $z_1 = z_4$ that summarizes the transitivity $A$-chain in the figure. For the variation in Figure 1(b), which provides an alternative demonstration of the joint unsatisfiability of $A$ and $B$, an interpolant is the conjunction $z_1 = z_2 \wedge f(z_3) = z_4 \wedge f(z_2) = z_3$ of summaries of $A$-chains.

For yet another variation, this time with slightly different sets $A$ and $B$, modify Figure 1(b) by moving the disequality sign to the edge $\langle x_3, z_4 \rangle$. There, an interpolant is $z_1 = z_2 \wedge f(z_3) \neq z_4 \wedge f(z_2) = z_3$. □

**Example 3.2.** When the unsatisfiability of $A \cup B$ involves the congruence property of $=$, an interpolant in the form of a conjunction of equalities need not exist. Let

$$A = \{u_1 = x \cdot u_0,\, v_1 = x \cdot v_0\} \quad \text{and} \quad B = \{u_0 = v_0,\, u_1 \neq v_1\}$$

where the dot is an infix binary function symbol. There are no equalities entailed by $A$ that do not contain $x$. The transitivity chain $u_1 = x \cdot u_0 = x \cdot v_0 = v_1$ contradicts $u_1 \neq v_1 \in B$, but its middle equality is not entailed by $A$. However, $A$ does entail it under the condition $u_0 = v_0$ that $B$ provides. That gives us the interpolant $u_0 = v_0 \Rightarrow u_1 = v_1$.

**Example 3.3.** With

$$A \;=\; \{x = z_1,\, x \cdot z_2 = z_3\} \quad \text{and} \quad B \;=\; \{y = z_2,\, z_1 \cdot y \neq z_3\}$$

pictured in Figure 2, we can derive $\mathsf{false}$ from the chain $z_3 = x \cdot z_2 = z_1 \cdot y \neq z_3$, where the congruence reasoning that produces the middle equality $x \cdot z_2 = z_1 \cdot y$ uses an equality from $A$ ($x = z_1$) and an equality from $B$ ($z_2 = y$), and cannot be derived from either $A$ or $B$ alone. A simple split of the problematic equality into two produces a chain in which every literal follows from either $A$ or $B$: $z_3 = x \cdot z_2 = z_1 \cdot z_2 = z_1 \cdot y \neq z_3$. The summary $z_3 = z_1 \cdot z_2$ of the $A$-chain is and interpolant of $A$ and $B$. The upshot here is that creating

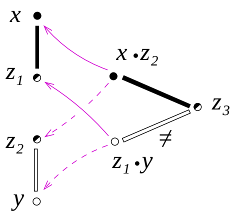

Figure 2: The solid and the dashed arrows point to the two equalities of $A \cup B$ that entail the equality $x \cdot z_2 = z_1 \cdot y$. (Example 3.3)

an interpolant may require terms (in this case, $z_1 \cdot z_2$) that do not occur in either $A$ or $B$. See Lemma 4.6 below. □

## 4. Interpolants From Congruence Closure

Efficient decision procedures for the satisfiability of sets of literals in EUF are typically based on *congruence closure* [NO80, DNS05, NO05]. In this section, we show that one can minimally modify such procedures to produce interpolants as well.

### 4.1. **Congruence Closure.**

The CONGRUENCE CLOSURE ALGORITHM takes as inputs

- a finite set $E$ of ground equalities and
- a finite subterm-closed set $T$ of ground terms.

Its state is an *undirected* graph $G$, initialized so that its vertex set is $T$ and its edge set is empty. We write $u \sim v$ to mean that $u$ and $v$ are connected by a path in $G$. The algorithm proceeds as follows.

(**cc0**) Let $G = (T, \varnothing)$
(**cc1**) Choose distinct $s, t \in T$ such that $s \not\sim t$ and either
  (**a**) $(s = t) \in E$; or
  (**b**) $s$ is $f(s_1, \ldots, s_k)$, $t$ is $f(t_1, \ldots, t_k)$, and $s_1 \sim t_1$, …, $s_k \sim t_k$.
  Then add the edge $\langle s, t\rangle$ to $G$
(**cc2**) Repeat (**cc1**) for as long as possible.

**Theorem 4.1.** [NO80, NO05] *Let $\sim$ be the equivalence relation obtained by running the congruence closure algorithm above. For every $s, t \in T$, one has $E \models s = t$ if and only if $s \sim t$. Moreover, the set $E \cup \{s \neq t \mid s \not\sim t\}$ is satisfiable.* □

If $L$ is an arbitrary set of ground *EUF* literals, let $L = L_= \cup L_{\neq}$, where $L_=$ and $L_{\neq}$ consists respectively of the equalities and disequalities of $L$. To check whether $L$ is satisfiable, it suffices to run the congruence closure algorithm with $E = L_=$ and $T$ consisting of all the terms (and subterms) occurring in $L$. By Theorem 4.1, $L$ is satisfiable if and only if $s \not\sim t$ holds for every disequality $s \neq t$ in $L_{\neq}$. Conversely, $L$ is unsatisfiable if and only if $L_= \cup \{\delta\}$ is unsatisfiable for some $\delta \in L_{\neq}$.

4.2. **Congruence Graphs.** For any finite set $E$ of ground equalities and a finite subterm-closed set $T$ of ground terms, a CONGRUENCE GRAPH over $E$ and $T$ is any intermediate graph $G$ obtainable by the congruence closure algorithm above. We will not mention the term set $T$ when it is understood or unimportant.

The assumption $s \not\sim t$ in Step (**cc1**) ensures that every congruence graph is acyclic. Thus, if $u \sim v$ in a congruence graph $G$, there is a unique PATH connecting them. We denote this path by $\overline{uv}$. EMPTY PATHS are those of the form $\overline{uu}$.

We call an edge of a congruence graph $G$ BASIC or DERIVED depending on whether it has been introduced in $G$ respectively because of Condition (**a**) or Condition (**b**) of Step (**cc1**). A derived edge $\langle f(u_1,\dots,u_k), f(v_1,\dots,v_k)\rangle$ has $k$ PARENT PATHS $\overline{u_1v_1}, \dots, \overline{u_kv_k}$, some (but not all) of which may be empty.

**Example 4.2.** Each of the graphs in Figures 1 and 2, *when we delete from it the edge marked with the* $\neq$ *symbol*, is a congruence graph over the corresponding set of equalities $(A \cup B)_=$. All edges in these graphs are basic; in Figure 2, a derived edge between the nodes $x \cdot z_2$ and $z_1 \cdot y$ could be added as a consequence of the basic edges pointed to by the arrows. □

**Example 4.3.** Let $E = (A \cup B)_=$ where

$$\begin{aligned} A &= \{x_1 = z_1,\, z_2 = x_2,\, z_3 = f\,x_1,\, f\,x_2 = z_4,\, x_3 = z_5,\, z_6 = x_4,\, z_7 = f\,x_3,\, f\,x_4 = z_8\}, \\ B &= \{z_1 = z_2,\, z_5 = f\,z_3,\, f\,z_4 = z_6,\, y_1 = z_7,\, z_8 = y_2,\, y_1 \neq y_2\}\ . \end{aligned}$$

Figure 3(b) depicts a congruence graph over $E$. The basic edges are shown in Figure 3(a); each corresponds to an equality in $E$. Since $f$ is unary, each of the three derived edges has one parent path. □

4.3. **Colorable Congruence Graphs.** Let $A$ and $B$ be sets of ground literals and let $\Sigma_A$ and $\Sigma_B$ be the sets of non-logical symbols that occur in $A$ and $B$, respectively. Terms, literals, and formulas over $\Sigma_A$ will be called $A$-COLORABLE, those over $\Sigma_B$ will be called $B$-COLORABLE. Such expressions will be called COLORABLE if they are either $A$-colorable or $B$-colorable, and $AB$-COLORABLE if they are both.

**Example 4.4.** In Example 3.3, $\Sigma_A = \{x, z_1, z_2, z_3, \cdot\}$ and $\Sigma_B = \{y, z_1, z_2, z_3, \cdot\}$. Terms and equalities without occurrences of either $x$ or $y$ are $AB$-colorable. The term $x \cdot y$ and the equality $x \cdot z_2 = z_1 \cdot y$ are not colorable. □

We extend the above definitions to edges of congruence graphs over $A \cup B$ so that an edge $\langle s, t\rangle$ has the same colorability attributes as the equality $s = t$. Note that basic edges are always colorable. Finally, we define a path in a congruence graph (resp., a congruence graph) to be COLORABLE if all edges in the path (resp., graph) are colorable.

**Example 4.5.** The congruence graphs derived from graphs in Figures 1 and 2 by removing their disequality edges are all colorable. Among the vertices (which are terms), the half-filled ones are $AB$-colorable, the dark ones are $A$-colorable but not $B$-colorable, and the light ones are $B$- but not $A$-colorable; however, if we add the derived edge $\langle x \cdot z_2, z_1 \cdot y\rangle$ to the graph in Figure 2, it will not be colorable. □

For our purposes, the uncolorability of some congruence graphs is not a problem thanks to the following result.

**Lemma 4.6.** *If $s$ and $t$ are colorable terms and if $A, B \models s = t$, then there exist a term set $T$ and a colorable congruence graph over $(A \cup B)_=$ and $T$ in which $s \sim t$.*

*Proof.* This is essentially Lemma 2 of [YM05], and the proof is constructive. Start with any congruence graph $G$ with colorable vertices in which $s \sim t$ holds. If there are uncolorable edges, let $e = \langle f(u_1, \dots, u_k), f(v_1, \dots, v_k)\rangle$ be a minimal such edge in the derivation order. Thus, the parent paths $\overline{u_i v_i}$ are all colorable, and each of them connects an $A$-colorable vertex with a $B$-colorable one. It follows that there exists an $AB$-colorable vertex $w_i$ on each path $\overline{u_i v_i}$ (which may be one of its endpoints). The term $f(w_1, \dots, w_k)$ is $AB$-colorable, so add it to the vertex set of $G$ and replace $e$ in $G$ with the two edges $\langle f(u_1, \dots, u_k), f(w_1, \dots, w_k)\rangle$ and $\langle f(w_1, \dots, w_k), f(v_1, \dots, v_k)\rangle$, both of which are colorable. Now repeat the process until all uncolorable edges of $G$ are eliminated. The set $T$ is the final set of vertices of $G$. □

Note that the proof of Lemma 4.6 provides an effective procedure for turning any uncolorable graph into a colorable one. Using a data structure for the congruence graph that also maintains for each derived edge a pointer to its parent paths allows a linear-time bottom-up implementation of the procedure.

**Example 4.7.** Consider again the uncolorable congruence graph obtained by adding the derived edge $\langle x \cdot z_2, z_1 \cdot y\rangle$ to the graph in Figure 2. Using the procedure in the proof of Lemma 4.6 we can turn it into a colorable congruence graph by replacing the edge $\langle x \cdot z_2, z_1 \cdot y\rangle$ with the edges $\langle x \cdot z_2, z_1 \cdot z_2\rangle$ and $\langle z_1 \cdot z_2, z_1 \cdot y\rangle$. □

### 4.4. Colored Congruence Graphs.

Assume (without loss of generality) that the literal sets $A, B$ are disjoint. A COLORING of a colorable congruence graph over $(A \cup B)_=$ is an assignment of a unique color $A$ or $B$ to each edge of the graph, such that

- basic edges are assigned the color of the set they belong to,
- every edge colored $X$ has both endpoints $X$-colorable ($X \in \{A, B\}$).

Thus, to color a colorable congruence graph, the only choice we have is with $AB$-colorable derived edges, and each of them can be colored arbitrarily. In the terminology of the interpolation game described later in §6, this means choosing which prover derives an $AB$-equality in a situation when either of them could do it. In Figure 3(b,c) we have two colored congruence graphs. They differ only in the coloring of $\langle f(z_3), f(z_4)\rangle$—the only derived edge with $AB$-colorable endpoints.

In a colored graph, we can speak of $A$-PATHS (whose edges are all colored $A$), and $B$-PATHS. There is also a color-induced factorization of arbitrary paths, where a FACTOR of a path $\pi$ is a maximal subpath of $\pi$ consisting of equally colored edges. Clearly, every path can be uniquely represented as a concatenation of its factors, the consecutive factors having distinct colors.

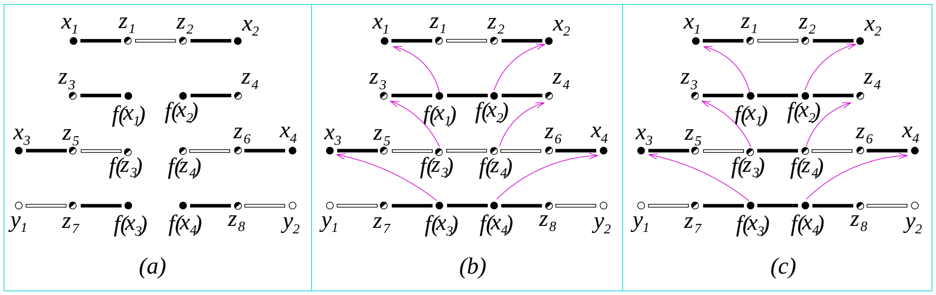

Figure 3: Congruence graphs over $(A \cup B)_=$, with $A$ and $B$ from Example 4.3. The connection between a derived edge and its parent is indicated by a pair of arrows.

4.5. **The Interpolation Algorithm.** Our goal is to construct an interpolant for the pair of sets $A$ and $B$ of ground literals that are jointly inconsistent in $EUF$. The algorithm presented below relies on the results in the previous subsection which guaranteed the existence (and computability) of a disequality $s \neq t$ in $A \cup B$ and a colored congruence graph $G$ over $(A \cup B)_=$ such that $s$ and $t$ are connected in $G$.

A path $\overline{uv}$ in a congruence graph represents the equality $u = v$ between its endpoints, summarizing the reflexivity, symmetry and transitivity inferences encoded by the path. The algorithm presented below builds an interpolant as a conjunction of Horn clauses whose atoms are $AB$-colorable equalities, each summarizing an A-path or a B-path of the graph $G$. The algorithm minimizes the number of such equalities by breaking paths only along their color-induced factorization (as opposed to other, finer partitions).

We will write $[\![\pi]\!]$ to denote the equality represented by the path $\pi$. More generally, if $P$ is a set of paths, $[\![P]\!]$ is the corresponding set of equalities. For convenience, we will take $[\![\overline{uu}]\!]$ to be true, instead of $u = u$, for each empty path $\overline{uu}$. (Similarly for $[\![P]\!]$, when $P = \varnothing$.)

For every path $\pi$ in a colored congruence graph $G$, we define below the associated $B$-premise set $\mathcal{B}(\pi)$, the $A$-justification $J(\pi)$, and the path interpolant $I(\pi)$. Intuitively, for an $A$-path $\pi$, the $B$-premise set collects all the maximal $B$-paths in $G$ that allow the construction of $\pi$ (by connecting ancestors of edges in $\pi$); the $A$-justification is an implication from all the equalities represented by $\pi$'s $B$-premises to $[\![\pi]\!]$, capturing the fact that $[\![\pi]\!]$ is a consequence of $A$ and all those equalities; the path interpolant is the conjunction of $\pi$'s $A$-justification together with the path interpolants for each of its $B$-premises. For a $B$-path $\pi$, the $B$-premise set is simply $\{\pi\}$; the $A$-justification is, trivially, $[\![\pi]\!] \Rightarrow [\![\pi]\!]$ (and actually never used); the path interpolant is the conjunction of all the path interpolants of $\pi$'s parent paths.

For instance, for the congruence graph in Figure 3(b), $\mathcal{B}(\overline{z_3z_4}) = \{\overline{z_1z_2}\}$, $J(\overline{z_3z_4}) = (z_1 = z_2 \Rightarrow z_3 = z_4)$, $I(\overline{z_5z_6}) = I(\overline{z_3z_4}) = (z_1 = z_2 \Rightarrow z_3 = z_4)$, $\mathcal{B}(\overline{z_7z_8}) = \{\overline{z_5z_6}\}$, $J(\overline{z_7z_8}) = (z_5 = z_6 \Rightarrow z_7 = z_8)$, and $I(\overline{z_7z_8}) = (z_5 = z_6 \Rightarrow z_7 = z_8) \wedge (z_1 = z_2 \Rightarrow z_3 = z_4)$.

$$\mathcal{B}(\pi) = \begin{cases} \bigcup\{\mathcal{B}(\sigma) \mid \sigma \text{ is a factor of } \pi\} & \text{if } \pi \text{ has} \geq 2 \text{ factors} \\ \{\pi\} & \text{if } \pi \text{ is a } B\text{-path} \\ \bigcup\{\mathcal{B}(\sigma) \mid \sigma \text{ is a parent of an edge of } \pi\} & \text{if } \pi \text{ is an } A\text{-path} \end{cases} \tag{4.1}$$

$$J(\pi) = (\bigwedge [\![\mathcal{B}(\pi)]\!]) \Rightarrow [\![\pi]\!] \tag{4.2}$$

$$I(\pi) = \begin{cases} \bigwedge\{I(\sigma) \mid \sigma \text{ is a factor of } \pi\} & \text{if } \pi \text{ has} \geq 2 \text{ factors} \\ \bigwedge\{I(\sigma) \mid \sigma \text{ is a parent of an edge of } \pi\} & \text{if } \pi \text{ is a } B\text{-path} \\ J(\pi) \ \wedge \ \bigwedge\{I(\sigma) \mid \sigma \in \mathcal{B}(\pi)\} & \text{if } \pi \text{ is an } A\text{-path} \end{cases} \tag{4.3}$$

Empty parent paths $\sigma$ in the definitions of $\mathcal{B}(\pi)$ and $I(\pi)$ can be ignored because $[\![\sigma]\!] = J(\sigma) = I(\sigma) = \mathsf{true}$ when $\sigma$ is empty.

We also need a modified interpolant function $I'$, expressed in terms of $I$ as follows. The argument path $\pi$ is first decomposed as $\pi = \pi_1\theta\pi_2$, where $\theta$ is the largest subpath with $B$-colorable endpoints, or an empty path if there are no $B$-colorable vertices on $\pi$; then

$$I'(\pi) = I(\theta) \ \wedge \ \bigwedge\{I(\tau) \,|\, \tau \in \mathcal{B}(\pi_1) \cup \mathcal{B}(\pi_2)\} \wedge \big(\bigwedge [\![\mathcal{B}(\pi_1) \cup \mathcal{B}(\pi_2)]\!] \ \Rightarrow \ \neg[\![\theta]\!]\big) \tag{4.4}$$

It is not difficult to see that $\mathcal{B}, J, I$ and $I'$ are all well defined and computable. In particular, $I'$ is well defined because $\pi_1, \theta, \pi_2$ are uniquely determined by $\pi$ if $\theta$ is not empty, and if $\theta$ is empty, the way we write $\pi$ as $\pi_1\pi_2$ is irrelevant. Note that when $\pi = \theta$, we have $I'(\pi) = I(\pi) \wedge \neg[\![\pi]\!]$.

The *EUF* GROUND INTERPOLATION ALGORITHM, given as input two jointly inconsistent (disjoint) sets $A, B$ of literals, proceeds as follows.

**(i1)** Run the congruence closure algorithm to find a congruence graph $G$ over $(A \cup B)_=$ and a disequality $(s \neq t) \in A \cup B$ such that $s \sim t$ in $G$ [§4.1,§4.2].
**(i1)** Modify $G$ as necessary to make it colorable [§4.3], then color it [§4.4].
**(i1)** If $(s \neq t) \in B$, return $I(\overline{st})$; if $(s \neq t) \in A$, return $I'(\overline{st})$.

**Example 4.8.** Let us run the algorithm for $A, B$ in Example 4.3, using the colored congruence graph in Figure 3(b). Since $y_1 \neq y_2 \in B$, the interpolant is computed by applying $I$ to $\overline{y_1y_2}$:

$$\begin{aligned} I(\overline{y_1y_2}) &= I(\overline{y_1z_7}) \wedge I(\overline{z_7z_8}) \wedge I(\overline{z_8y_2}) = \mathsf{true} \wedge I(\overline{z_7z_8}) \wedge \mathsf{true} \\ &= I(\overline{z_7z_8}) = J(\overline{z_7z_8}) \wedge \bigwedge\{I(\sigma) \mid \sigma \in \mathcal{B}(\overline{z_7z_8})\} \end{aligned}$$

In turn, $\mathcal{B}(\overline{z_7z_8}) = \mathcal{B}(\overline{x_3x_4}) = \mathcal{B}(\overline{x_3z_5}) \cup \mathcal{B}(\overline{z_5z_6}) \cup \mathcal{B}(\overline{z_6x_4}) = \varnothing \cup \{\overline{z_5z_6}\} \cup \varnothing = \{\overline{z_5z_6}\}$. Thus, $J(\overline{z_7z_8}) = (z_5 = z_6 \Rightarrow z_7 = z_8)$. Continuing the main computation:

$$\begin{aligned} I(\overline{y_1y_2}) &= J(\overline{z_7z_8}) \wedge I(\overline{z_5z_6}) \\ &= J(\overline{z_7z_8}) \wedge J(\overline{z_3z_4}) \wedge \bigwedge\{I(\sigma) \mid \sigma \in \mathcal{B}(\overline{z_3z_4})\} \end{aligned}$$

Now, $\mathcal{B}(\overline{z_3z_4}) = \mathcal{B}(\overline{x_1x_2}) = \mathcal{B}(\overline{x_1z_1}) \cup \mathcal{B}(\overline{z_1z_2}) \cup \mathcal{B}(\overline{z_2x_2}) = \varnothing \cup \{\overline{z_1z_2}\} \cup \varnothing = \{\overline{z_1z_2}\}$. Thus, $J(\overline{z_3z_4}) = (z_1 = z_2 \Rightarrow z_3 = z_4)$. Back to the main computation again,

$$\begin{aligned} I(\overline{y_1y_2}) &= J(\overline{z_7z_8}) \wedge J(\overline{z_3z_4}) \wedge I(\overline{z_1z_2}) = J(\overline{z_7z_8}) \wedge J(\overline{z_3z_4}) \wedge \mathsf{true} \\ &= (z_5 = z_6 \Rightarrow z_7 = z_8) \wedge (z_1 = z_2 \Rightarrow z_3 = z_4) \end{aligned}$$

The reader can verify that using the graph in Figure 3(c) results in a different interpolant:

$$I(\overline{y_1y_2}) = (z_5 = f(z_3) \wedge z_6 = f(z_4) \wedge z_1 = z_2) \Rightarrow z_7 = z_8.$$

4.6. **Correctness.** Our main correctness results can be expressed as follows.

**Theorem 4.9.** *With any jointly inconsistent sets A,B of EUF literals as inputs, the EUF ground interpolation algorithm (§4.5) terminates and returns an interpolant for A,B that is a conjunction of Horn clauses.* □

To prove the theorem we need to introduce some additional notions and notation. For the rest of the section let $G$ be a colored congruence graph.

The termination of our recursive definitions and other inductive arguments are proved using a well-founded relation $\prec$ over paths of $G$. Define $\sigma \prec_1 \pi$ to hold whenever:

- $\pi$ has more than one factor and $\sigma$ is one of them, or
- $\sigma$ is a parent path of an edge of $\pi$.

Then, define $\prec$ as the transitive closure of $\prec_1$. It is not difficult to see that the relation $\prec$ is well-founded. Note that minimal elements under $\prec$ are the paths all of whose edges are basic and of the same color.

The following equations redefine the set $\mathcal{B}(\pi)$ of $B$-premises and introduce the analogous set $\mathcal{A}(\pi)$ of $A$-PREMISES.

$$\mathcal{A}(\pi) \;=\; \{A\text{-factors of } \pi\} \;\cup\; \mathcal{A}(\{\text{parent paths of } B\text{-edges of } \pi\}) \tag{4.5}$$

$$\mathcal{B}(\pi) \;=\; \{B\text{-factors of } \pi\} \;\cup\; \mathcal{B}(\{\text{parent paths of } A\text{-edges of } \pi\}) \tag{4.6}$$

Here and in the sequel, we use the convention $f(P) = \bigcup\{f(\sigma) \mid \sigma \in P\}$ for extending a set-valued function $f$ defined on paths to a function defined on sets of paths. Observe that (4.6) is just a restatement of (4.1). Also, the arguments in the recursive calls are smaller than $\pi$ under the relation $\prec$, so termination is guaranteed.

The basic properties of $\mathcal{A}$ are collected in the following lemma. The analogous properties of $\mathcal{B}$ follow by symmetry.

**Lemma 4.10.** *Let $\pi$ be an arbitrary non-empty path in $G$.*

(1) *If $\pi$ is an $A$-path, then $\mathcal{A}(\pi) = \{\pi\}$; otherwise, $\sigma \prec \pi$ for every $\sigma \in \mathcal{A}(\pi)$.*
(2) *If $\sigma \in \mathcal{A}(\pi)$, then $\mathcal{A}(\sigma) \subseteq \mathcal{A}(\pi)$.*
(3) *If the endpoints of $\pi$ are $B$-colorable, then the endpoints of all paths in $\mathcal{A}(\pi)$ are $AB$-colorable.*

*Proof.* All three parts are proved by well-founded induction.

*(1)* If $\pi$ is an $A$-colored path, then $\pi$ is the only element of $\mathcal{A}(\pi)$ (by definition). If $\pi$ is not an $A$-colored path and $\tau$ is an element of $\mathcal{A}(\pi)$, then $\tau$ is either an $A$-factor of $\pi$ and so $\tau \prec \pi$ holds, or $\tau \in \mathcal{A}(\sigma)$ for some parent $\sigma$ of a $B$-edge of $\pi$. In the latter case, $\tau \prec \pi$ holds because of $\sigma \prec \pi$ and the consequence $\tau \preceq \sigma$ of the induction hypothesis.

*(2)* If $\sigma$ is an $A$-factor of $\pi$, then $\mathcal{A}(\sigma) = \{\sigma\} \subseteq \mathcal{A}(\pi)$. If $\sigma \in \mathcal{A}(\tau)$ where $\tau$ is a parent path of a $B$-edge of $\pi$, then $\mathcal{A}(\sigma) \subseteq \mathcal{A}(\tau) \subseteq \mathcal{A}(\pi)$, the first inclusion by induction hypothesis, the second from the definition of $\mathcal{A}$.

*(3)* Since parent paths of any $B$-edge must have $B$-colorable endpoints, for the inductive argument we only need to check that every $A$-factor of a path $\pi$ with $B$-colorable endpoints has $AB$-colorable endpoints. Indeed, $A$-colorability of endpoints of $A$-factors is obvious. For $B$-colorability, observe that an endpoint of an $A$-factor of $\pi$ is either also an endpoint of a $B$-factor of $\pi$, or an endpoint of $\pi$ itself. □

The following lemma justifies the names *A-premises* and *B-premises*. Intuitively, $B$-premises are the $B$-paths whose summaries, together with $A$, entail $[\![\pi]\!]$. Dually, $A$-premises are the $A$-paths whose summaries, together with $B$, entail $[\![\pi]\!]$.

**Lemma 4.11.** *$A, [\![\mathcal{B}(\pi)]\!] \models [\![\pi]\!]$ and $B, [\![\mathcal{A}(\pi)]\!] \models [\![\pi]\!]$ for every path $\pi$ in $G$.*

*Proof.* We prove the first claim only, by well-founded induction based on $\prec$. Viewing $\pi$ as the concatenation of its $B$-factors and $A$-edges, we have by transitivity

$$[\![B\text{-factors of } \pi]\!],\ [\![A\text{-edges of } \pi]\!] \models [\![\pi]\!]$$

and then, since $A \models [\![e]\!]$ for every basic $A$-edge $e$ (by definition of edge coloring),

$$A,\ [\![B\text{-factors of } \pi]\!],\ [\![\text{derived } A\text{-edges of } \pi]\!] \models [\![\pi]\!].$$

For every derived edge $e$ we have $[\![\text{parents of } e]\!] \models [\![e]\!]$. Thus,

$$A,\ [\![B\text{-factors of } \pi]\!],\ [\![\text{parents of } A\text{-edges of } \pi]\!] \models [\![\pi]\!],$$

so it suffices to prove $A,\ [\![\mathcal{B}(\pi)]\!] \models [\![\sigma]\!]$ for every $\sigma$ that is either a $B$-factor of $\pi$ or a parent of an $A$-edge of $\pi$.

In the first case, the claim holds since $\sigma \in \mathcal{B}(\pi)$. In the second case, we have $\sigma \prec \pi$, so the induction hypothesis gives us $A,\ [\![\mathcal{B}(\sigma)]\!] \models [\![\sigma]\!]$. To finish the proof, just use the fact $\mathcal{B}(\sigma) \subseteq \mathcal{B}(\pi)$, by Lemma 4.10(ii). □

Define the CUMULATIVE SET OF PREMISES (cf. §6) of a path $\pi$ as

$$\mathcal{P}(\pi) \ = \ \{\pi\} \cup \mathcal{P}(\mathcal{B}(\mathcal{A}(\pi))) \ . \tag{4.7}$$

The termination of this recursive definition follows from Lemma 4.10(i).

**Lemma 4.12.** *For every path $\pi$ in $G$, $I(\pi) \ = \ \bigwedge \{J(\sigma) \mid \sigma \in \mathcal{A}(\mathcal{P}(\pi))\}$.*

*Proof.* Let $\mathcal{P}'(\pi) = \mathcal{A}(\mathcal{P}(\pi))$. From (4.7), we have

$$\mathcal{P}'(\pi) \ = \ \mathcal{A}(\pi) \cup \mathcal{P}'(\mathcal{B}(\mathcal{A}(\pi))) \tag{4.8}$$

It suffices to check that

$$\mathcal{P}'(\pi) \quad = \quad \begin{cases} \bigcup\{\mathcal{P}'(\sigma) \mid \sigma \text{ is a factor of } \pi\} & \text{if } \pi \text{ has} \geq 2 \text{ factors} \\ \bigcup\{\mathcal{P}'(\sigma) \mid \sigma \text{ is a parent of an edge of } \pi\} & \text{if } \pi \text{ is a } B\text{-path} \\ \{\pi\} \ \cup \ \bigcup\{\mathcal{P}'(\sigma) \mid \sigma \in \mathcal{B}(\pi)\} & \text{if } \pi \text{ is an } A\text{-path} \end{cases}$$

For the first case, suppose $\pi = \pi_1 \cdots \pi_k$ is the factorization of $\pi$. By definition of $\mathcal{A}$, we have $\mathcal{A}(\pi) = \mathcal{A}(\pi_1) \cup \cdots \cup \mathcal{A}(\pi_k)$. The desired equality $\mathcal{P}'(\pi) = \mathcal{P}'(\pi_1) \cup \cdots \cup \mathcal{P}'(\pi_k)$ then follows from (4.8).

Assume now that $\pi = e_1 \cdots e_k$ is a $B$-path. By definition of $\mathcal{A}$, we have $\mathcal{A}(\pi) = \mathcal{A}(E_1) \cup \cdots \cup \mathcal{A}(E_k)$, where $E_i$ is the set of parent paths of the edge $e_i$ for $i = 1, \ldots, k$. Again, the desired equality $\mathcal{P}'(\pi) = \mathcal{P}'(E_1) \cup \cdots \cup \mathcal{P}'(E_k)$ follows from (4.8).

Finally, assume that $\pi$ is an $A$-path. Now $\mathcal{A}(\pi) = \{\pi\}$ and so $\mathcal{P}'(\pi) = \{\pi\} \cup \mathcal{P}'(\mathcal{B}(\pi))$, again by (4.8). □

**Lemma 4.13.** *$B, I(\pi) \models [\![\pi]\!]$ for every path $\pi$ in $G$ with $B$-colorable endpoints.*

*Proof.* We argue by induction along $\prec$. Let $\sigma$ be an arbitrary $A$-premise of $\pi$ and $\tau$ an arbitrary $B$-premise of $\sigma$. The endpoints of $\tau$ are $B$-colorable, because in general, every $B$-premise of any path is a $B$-factor of some path, and every $B$-factor of any path has $B$-colorable endpoints. Thus, the induction hypothesis applies to $\tau$ and we have $B, I(\tau) \models [\![\tau]\!]$. From equation (4.8) we have $\mathcal{P}'(\tau) \subseteq \mathcal{P}'(\pi)$, so we can derive $I(\pi) \models I(\tau)$ using Lemma 4.12. Thus, $B, I(\pi) \models [\![\tau]\!]$ for every $\tau \in \mathcal{B}(\sigma)$. By Lemma 4.12, $I(\pi)$ contains $J(\sigma)$ as a conjunct; therefore, $B, I(\pi) \models [\![\sigma]\!]$. Since $\sigma$ here is an arbitrary element of $\mathcal{A}(\pi)$, the second claim of Lemma 4.11 finishes the proof. □

*Proof of Theorem 4.9.* The algorithm terminates because all pertinent functions have been proven terminating.

Let $s \neq t$ be the disequality obtained in the step (**i1**) of the algorithm. Let $\pi$ be the path $\overline{st}$, and let $\overline{st} = \pi_1 \theta \pi_2$, as in the definition of $I'(\pi)$. The two cases to consider, $s \neq t \in B$ and $s \neq t \in A$, will be referred to as Cases 1 and 2 respectively. Let $\varphi$ be the returned formula—$I(\pi)$ in Case 1; $I'(\pi)$ in Case 2.

*(i) $\varphi$ is an $AB$-colorable conjunction of Horn clauses.* For any factor $\sigma$ of $\pi$ with $AB$-colorable endpoints, $J(\sigma)$ is an $AB$-colorable Horn clause. If $\pi$ has $B$-colorable endpoints, then so do all paths in $\mathcal{P}(\pi)$ and so, by Lemma 4.10(iii), all paths in $\mathcal{A}(\mathcal{P}(\pi))$ have $AB$-colorable endpoints. With Lemma 4.12, this proves Case 1. For Case 2, observe that if $\theta$ is empty, then $I(\theta) = [\![\theta]\!] = \mathsf{true}$; otherwise, $\theta$ has $AB$-colorable endpoints. Also, $\pi_1$ and $\pi_2$ are $A$-paths, so by the dual of Lemma 4.10(iii), all paths in $\mathcal{B}(\pi_1) \cup \mathcal{B}(\pi_2)$ have $AB$-colorable endpoints. These facts suffice to derive the proof of Case 2 from the already proved Case 1.

*(ii) $A \models \varphi$.* By the first claim of Lemma 4.11, $A \models J(\sigma)$ holds for every path $\sigma$. This suffices for Case 1. For Case 2 then, we only need to check that the last conjunct of $I'(\pi)$ is implied by $A$, which amounts to showing $A, [\![\mathcal{B}(\pi_1)]\!], [\![\mathcal{B}(\pi_2)]\!] \models \neg[\![\theta]\!]$. This indeed follows from the first claim of Lemma 4.11, the transitivity entailment $[\![\pi_1]\!], [\![\theta]\!], [\![\pi_2]\!] \models [\![\pi]\!]$, and the assumption $\neg[\![\pi]\!] \in A$.

*(iii) $B, \varphi \models \mathsf{false}$.* In Case 1, we have $\neg[\![\pi]\!] \in B$, so Lemma 4.13 finishes the proof. In Case 2, Lemma 4.13 implies $B, I'(\pi) \models [\![\theta]\!]$ and $B, I'(\pi) \models I(\tau)$ for every $\tau \in \mathcal{B}(\pi_1) \cup \mathcal{B}(\pi_2)$. These consequences of $B \cup I'(\pi)$ contradict the last conjunct of $I'(\pi)$. □

## 5. Comparison with McMillan's Algorithm

Our *EUF* ground interpolation algorithm is, as far as we know, the only alternative to McMillan's algorithm [McM05b]. The latter constructs an interpolant for $A, B$ from the proof of $A, B \models \mathsf{false}$ derived in a formal system ($\mathcal{E}$, say) with rules for introducing hypotheses (equalities from $A \cup B$), reflexivity, symmetry, transitivity, congruence, and contradiction (deriving $\mathsf{false}$ from an equality and its negation). The algorithm proceeds top down by

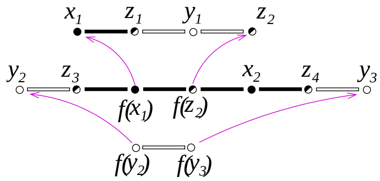

Figure 4: A colored congruence graph for $A = \{x_1 = z_1, z_3 = f(x_1), f(z_2) = x_2, x_2 = z_4\}$ and $B = \{z_1 = y_1, y_1 = z_2, y_2 = z_3, z_4 = y_3, f(y_2) \neq f(y_3)\}$ with two derived edges $\langle f(x_1), f(z_2)\rangle$ and $\langle f(y_2), f(y_3)\rangle$.

annotating each intermediate derived equality $u = v$ (or $\mathsf{false}$ in the final step) with a quadruple of the form $[u', v', \rho, \gamma]$, where $u', v'$ are terms and $\rho, \gamma$ are $AB$-colorable formulas. The annotation of each derived equality is obtained from annotations of the equalities occurring in the premises of the corresponding rule application. The exact computation of annotations is specified by 11 rules, each corresponding to a case (depending on colors of the terms involved) of one of the original six rules. An invariant that relates a derived intermediate equality with its annotation is formulated and all 11 rules are proved to preserve the invariant. The invariant implies that if $[u', v', \rho, \gamma]$ is the annotation of $\mathsf{false}$, then $\rho \Rightarrow \gamma$ is an interpolant for $A, B$. It can be shown that $\rho$ is always a conjunction of Horn clauses, and $\gamma$ is a conjunction of equalities and at most one disequality.

There is a clear relationship between proofs in the formal system $\mathcal{E}$ and congruence graphs from which our interpolants are derived. The main difference is that in congruence graphs, paths condense inferences by reflexivity, symmetry, and transitivity. A congruence graph provides a big-step proof that, if necessary, can be expanded into a proof in the system $\mathcal{E}$.

In Example 3.1 (Figure 1(a)) our algorithm looks at the path $\overline{y_3 z_4}$, summarizes its only $A$-factor, producing the interpolant $z_1 = z_4$. McMillan's algorithm processes the path edge-by-edge, eagerly summarizing $A$-chains with $AB$-colorable endpoints, so that the interpolant it produces is $z_1 = z_2 \wedge z_2 = f(z_3) \wedge f(z_3) = z_4$.

For the second difference, consider Example 4.3 (Figure 3(b)) where McMillan's algorithm produces an entangled version $(z_1 = z_2 \wedge (z_3 = z_4 \Rightarrow z_5 = z_6)) \Rightarrow z_3 = z_4 \wedge z_7 = z_8$ of our interpolant $(z_1 = z_2 \Rightarrow z_3 = z_4) \wedge (z_5 = z_6 \Rightarrow z_7 = z_8)$, computed in Example 4.8. In general, McMillan's algorithm accumulates $B$-justifications (duals of our $J(\sigma)$) in the $\rho$-part of the annotation and keeps them past their one-time use to derive a particular conjunct of $\gamma$.

The third difference is in creating auxiliary $AB$-terms ("equality interpolants", in the terminology of Yorsh and Musuvathi [YM05]) to split derivations of equalities in which one side is not $A$-colorable and the other is not $B$-colorable, as in Example 3.3. We introduce such terms in the preliminary step (**i2**) of our algorithm only when required to make the congruence graph colorable. In contrast, McMillan's algorithm introduces these terms "on-the-fly", as in the example illustrated in Figure 4. When it derives the equality $x_1 = z_2$, its annotation is $[z_1, z_2, \mathsf{true}, \mathsf{true}]$, then when it uses the congruence rule to derive $f(x_1) = f(z_2)$, this equality gets annotated with $[f(z_1), f(z_2), \mathsf{true}, \mathsf{true}]$, and the term $f(z_1)$ becomes part of the final interpolant $z_3 = f(z_1) \wedge f(z_2) = z_4$. On the other hand, our algorithm recognizes the edge $\langle f(x_1), f(z_2)\rangle$ as $A$-colorable and does not split it; the interpolant it produces is $z_1 = z_2 \Rightarrow z_3 = z_4$.

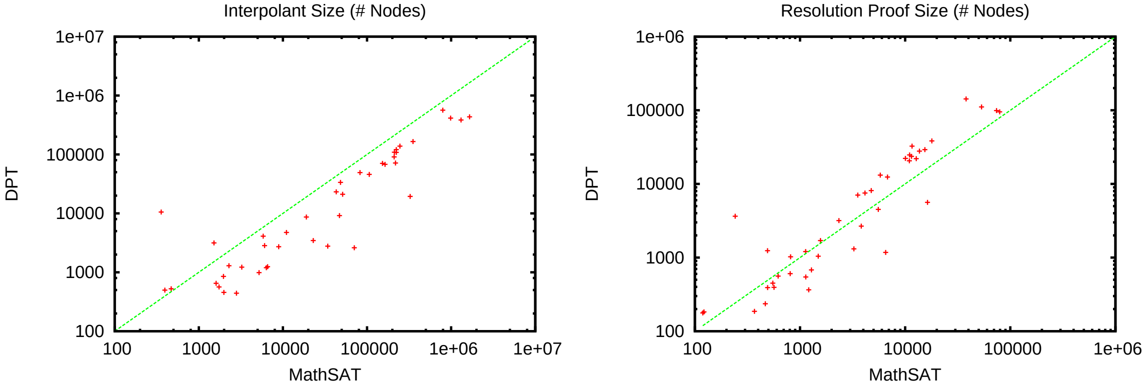

Figure 5: *DPT vs. MathSAT* on 45 benchmarks from the *MathSAT* library derived by partitioning unsatisfiable SMT-LIB benchmarks [BST11].

The final difference is in flexibility. McMillan's algorithm is fully specified and leaves little room for variation. On the other hand, the actions in the step (**i2**) of our algorithm are largely non-deterministic. Our current implementation chooses to minimize the number of vertices in the colorable modification of $\Gamma$, and then colors the graph with a strategy that eagerly minimizes the number of factors in the relevant paths. Other choices are yet to be explored.

5.1. **Experimental evaluation.** In general, our interpolation algorithm produces smaller and simpler interpolants. For experimental confirmation, we used the state-of-the-art implementation of McMillan's algorithm in *MathSAT* [CGS08] and compared it against our interpolation-generating extension of the *DPT* solver [Var08].

Two other relevant components—the propositional interpolation algorithm, and the algorithm for combining propositional and theory interpolation in a DPLL($\mathcal{T}$) framework [McM05b, CGS08]—are the same in *MathSAT* and *DPT*, and therefore unlikely to substantially affect the comparison. The last factor to be accounted for in this comparison is the size of the resolution proofs derived from the DPLL search within each solver. Since these sizes are comparable, we can eliminate differences in propositional reasoning as a cause for *DPT*'s producing smaller interpolants.

We ran both solvers on 45 *EUF* interpolation benchmarks selected from the set of 100 that are used in [CGS08]. (In the remaining 55 benchmarks, either all formulas in $A$ are $B$-colorable, or all formulas in $B$ are $A$-colorable, so one of the formulas $A$, $\neg B$ is an easily obtained interpolant.) Both solvers computed 42 interpolants, timing out in $100s$ on the same three benchmarks. Runtimes were comparable, with *DPT* being slightly faster. Figure 5 shows the sizes of interpolants produced: *DPT* interpolants are, on average, 3.8 times smaller, in spite of *DPT* proofs being, on average, 1.7 times larger.

While these experimental results confirm the claim that our algorithm produces smaller interpolants, we observe that formula size is not necessarily a good metric, given the ability of modern SMT-solvers to process large formulas quickly. It could be argued that some measure of *logical strength* would be better instead. The case for that, however, is not obvious either. To start, the only reasonable way to compare two first-order logic formulas $\varphi_1$ and $\varphi_2$ for logical strength is to check whether one of the two entails the other in the theory (*i.e.*, whether $\varphi_1 \models_{\mathcal{T}} \varphi_2$ or $\varphi_2 \models_{\mathcal{T}} \varphi_1$). Unfortunately, entailment is not a total relation and so it is possible to have incomparable interpolants for the same partition $A, B$ of a set of formulas. Finally, even with comparable invariants, whether the stronger or the weaker one is better depends on the specific application using them; worse still, for other applications, such as interpolation-based predicate abstraction, it is arguable that logical strength (or formula size for that matter) is of any importance, since interpolants are simply mined for useful predicates. Further work is needed to identify useful evaluation metrics for interpolants and then see if the flexibility of our algorithm, or a suitably modified version of it, can be used to produce better interpolants according to some of those metrics.

## 6. Interpolation as a Cooperative Game

Our results about *EUF* interpolation can be generalized to a wider class of theories $\mathcal{T}$ in terms of a cooperative *interpolation game* between two deductive provers for $\mathcal{T}$—possibly two copies of the same prover. The game metaphor suggests a simple and general mechanism for producing interpolants from sets of formulas and theories that satisfy certain requirements. We define this mechanism and prove its properties in §6.2 and §6.3, after giving an informal general description of the interpolation game.

For the rest of the section, let $\mathcal{T}$ be a first-order theory of signature $\Sigma$, and let $A$ and $B$ be two disjoint sets of formulas possibly containing FREE SYMBOLS, *i.e.*, predicate and function symbols not in $\Sigma$. For convenience, and without loss of generality, we consider only formulas with no free variables. Let $\Sigma_I$ be the SHARED SIGNATURE, the expansion of $\Sigma$ with the free symbols occurring in both $A$ and $B$.

### 6.1. The interpolation game.

The participants are an $A$-PROVER and a $B$-PROVER which incrementally construct a set $S_A$ and a set $S_B$ of $\Sigma_I$-formulas. The game starts with $S_A = S_B = \varnothing$ and proceeds in rounds so that at each round one of the following happens:

- the $A$-prover adds to $S_A$ one or more $\Sigma_I$-formulas $\alpha$ such that $A, \beta_1, \ldots, \beta_n \models_{\mathcal{T}} \alpha$ for some $\beta_1, \ldots, \beta_m \in S_B$, the $B$-PREMISES of $\alpha$;
- the $B$-prover adds to $S_B$ one or more $\Sigma_I$-formulas $\beta$ such that $B, \alpha_1, \ldots, \alpha_n \models_{\mathcal{T}} \beta$ for some $\alpha_1, \ldots, \alpha_n \in S_A$, the $A$-PREMISES of $\beta$.

The game ends successfully when the $B$-prover adds $\mathsf{false}$ to $S_B$.

As we discuss below, a $\mathcal{T}$-interpolant for $A$ and $B$ can be generated from a successful run of the game by tracking the $B$-premises of each formula in $S_A$ and the $A$-premises of each formula in $S_B$.

Note that, as described, the interpolation game involves arbitrary theories and input sets $A$ and $B$. Also, the game does not have to use two provers literally. If $A \cup B$ has a *local* refutation (see later) in the theory $\mathcal{T}$, it is possible to extract from that refutation a successful run of the game from which a $\mathcal{T}$-interpolant of $A$ and $B$ can then be generated.

For some theories and classes of input formulas the game admits *complete strategies*, guaranteed to end the game when $A$ and $B$ are jointly $\mathcal{T}$-unsatisfiable. Depending on the theory and the class of input formulas, these strategies can be considerably restrictive in the choice of formulas to propagate from one prover to the other (*i.e.*, formulas to add to $S_A$ and $S_B$). For instance, when $A$ and $B$ are sets of ground literals and the theory is *convex*,[1]

[1] A theory is CONVEX if $L \models_{\mathcal{T}} p_1 \vee \cdots \vee p_k$ implies $L \models_{\mathcal{T}} p_i$ for some $i$, where $L$ is any set of ground literals and the $p_i$ are any positive literals.

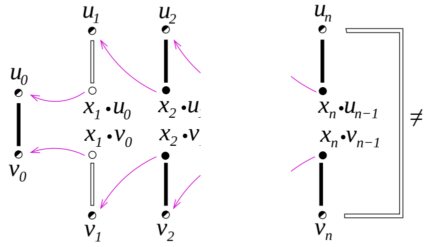

Figure 6: A long derivation. (Example 6.3)

it is enough to propagate just ground atomic formulas in all rounds of the game. In that case, all interpolants computed will be conjunctions of ground Horn clauses.

The interpolation method described in §4.5 for the theory of equality (which is convex) can be seen as a customized implementation of the interpolation game, with formula propagation restricted to (positive) equalities. A colorable congruence graph is a compact representation of a local refutation, and the interpolation function $I$ defined in §4.5 can be understood as generating the interpolant from a successful run of the game extracted from the local refutation.

**Example 6.1.** Looking back at Example 3.1 in terms of the interpolation game above, we can see that in each of the three cases presented in the example there is a successful interpolation game with two rounds. In the first round, the $A$-prover derives a conjunction of literals in the shared signature (respectively, $z_1 = z_4$, $z_1 = z_2 \wedge f(z_3) = z_4 \wedge f(z_2) = z_3$ and $z_1 = z_2 \wedge f(z_3) \neq z_4 \wedge f(z_2) = z_3$) that the $B$-prover uses them to derive false. □

**Example 6.2.** For the sets $A$ and $B$ in Example 3.2, there is a game with three rounds where $u_0 = v_0$ is initially derived by the $B$-prover, $u_1 = v_1$ is derived next by the $A$-prover, and then false is derived by the $B$-prover. □

**Example 6.3.** Generalizing the previous example, consider this matrix—organized set of literals:

$$\begin{array}{cccccc} u_0 = v_0 & \begin{array}{c} x_1 \cdot u_0 = u_1 \\ x_1 \cdot v_0 = v_1 \end{array} & \begin{array}{c} x_2 \cdot u_1 = u_2 \\ x_2 \cdot v_1 = v_2 \end{array} & \cdots & \begin{array}{c} x_n \cdot u_{n-1} = u_n \\ x_n \cdot v_{n-1} = v_n \end{array} & u_n \neq v_n \end{array}$$

Let $A$ be the set of equalities occurring in the odd-numbered columns (with columns counted starting from 1) of this matrix, and $B$ be the set of the remaining equalities; see Figure 6. The shared symbols are $u_0, v_0, \ldots, u_n, v_n$, the symbols local to $A$ are $x_2, x_4, \ldots$, and the symbols local to $B$ are $x_1, x_3, \ldots$

A run of the interpolation game takes $n+2$ rounds. It begins with the $A$-prover adding $u_0 = v_0$ to $S_A$. Then, using the equalities from the second column, the $B$-prover can derive $u_1 = v_1$, and add it to $S_B$. Now, the $A$-prover can use this equality together with equalities from the third column to derive $u_2 = v_2$ and add it to $S_A$. Assuming $n$ is even, the last equality $u_n = v_n$ will be derived by the $A$-prover, after which $B$ derives false. Collecting justifications of all equalities derived by $A$, we obtain the interpolant

$$(u_0 = v_0) \wedge (u_1 = v_1 \Rightarrow u_2 = v_2) \wedge \cdots \wedge (u_{n-1} = v_{n-1} \Rightarrow u_n = v_n) \; .$$

□

**Remark 6.4.** The well-known method for combining decision procedures due to Nelson and Oppen [NO79] is essentially a version of the interpolation game. The main differences are that in the Nelson-Oppen method (i) the input sets of formulas $A_1$ and $A_2$ need not be jointly $\mathcal{T}$-unsatisfiable; (ii) the goal is not to produce interpolants for $A_1$ and $A_2$ but just to check the $\mathcal{T}$-unsatisfiability of $A_1 \cup A_2$; (iii) $\mathcal{T}$ is the union of two signature-disjoint theories $\mathcal{T}_1$ and $\mathcal{T}_2$; (iv) each formula $A_i$ is built from the symbols of $\mathcal{T}_i$ and free constants; (v) each $A_i$-prover works just over $\mathcal{T}_i$ instead of the whole $\mathcal{T}$; (vi) additional restrictions on $\mathcal{T}_1$ and $\mathcal{T}_2$ guarantee termination even when $A_1 \cup A_2$ is $\mathcal{T}$-satisfiable.

A description of a Nelson-Oppen combination framework in terms similar to our interpolation game is given by Ghilardi [Ghi05]. □

6.2. **Extracting interpolants from interpolation runs.** To show how to generate $\mathcal{T}$-interpolants from runs of the interpolation game we start by formalizing the notion of a run.

**Definition 6.5.** A $\mathcal{T}$-INTERPOLATION RUN for $A$ and $B$ is a triple $(S_A, S_B, \sqsubset)$ where $S_A$ and $S_B$ are two disjoint finite sets of $\Sigma_I$-formulas and $\sqsubset$ is a well-founded (partial) ordering on $S_A \cup S_B$ with associated computable functions $\mathsf{P}_B : S_A \to 2^{S_B}$, $\mathsf{P}_A : S_B \to 2^{S_A}$ such that:

(1) $A, \mathsf{P}_B(\alpha) \models_{\mathcal{T}} \alpha$ and $\beta \sqsubset \alpha$ for all $\beta \in \mathsf{P}_B(\alpha)$;
(2) $B, \mathsf{P}_A(\beta) \models_{\mathcal{T}} \beta$ and $\alpha \sqsubset \beta$ for all $\alpha \in \mathsf{P}_A(\beta)$.

A $\mathcal{T}$-interpolation run $(S_A, S_B, \sqsubset)$ is SUCCESSFUL if $\mathsf{false} \in S_B$.

Given a $\mathcal{T}$-interpolation run $(S_A, S_B, \sqsubset)$, we extend $\mathsf{P}_B$ from $S_A$ to $2^{S_A}$ as done in §4.6, that is, for all $S \subseteq S_A$,

$$\mathsf{P}_B(S) = \bigcup \{\mathsf{P}_B(\alpha) \mid \alpha \in S\} \ .$$

We extend $\mathsf{P}_A$ from $S_B$ to $2^{S_B}$ in a similar way. Then, for all $\beta \in S_B$ let

$$\mathsf{P}(\beta) = \{\beta\} \cup \bigcup \{\mathsf{P}(\beta') \mid \beta' \in \mathsf{P}_B(\mathsf{P}_A(\beta))\} \ .$$

Extending $\mathsf{P}$ to $2^{S_B}$ as done with $\mathsf{P}_A$, we can write the definition of $\mathsf{P}$ more compactly as

$$\mathsf{P}(\beta) = \{\beta\} \cup \mathsf{P}(\mathsf{P}_B(\mathsf{P}_A(\beta))) \ .$$

Finally, we define the (computable) function $\mathcal{I}$ from $S_B$ to the set of $\Sigma_I$-formulas such that

$$\mathcal{I}(\beta) = \bigcup \{\mathsf{P}_B(\alpha) \Rightarrow \alpha \mid \alpha \in \mathsf{P}_A(\mathsf{P}(\beta))\} \ .$$

This function returns PARTIAL $\mathcal{T}$-INTERPOLANTS in the following sense.

**Lemma 6.6.** *Let $(S_A, S_B, \sqsubset)$ be a $\mathcal{T}$-interpolation run for $A$ and $B$ and let $\mathcal{I}$ be defined as above. Then, for all $\beta \in S_B$,*

*(1) $A \models_{\mathcal{T}} \mathcal{I}(\beta)$;*
*(2) $B, \mathcal{I}(\beta) \models_{\mathcal{T}} \beta$.*

*Proof.* We prove both claims by well founded induction on $\sqsubset$. By definition, $\mathsf{P}(\beta) = \{\beta\} \cup \mathsf{P}(\beta_1) \cup \cdots \cup \mathsf{P}(\beta_k)$ where $\{\beta_1, \ldots, \beta_k\} = \mathsf{P}_B(\mathsf{P}_A(\beta))$ with $k \geq 0$. Then,

$$\begin{array}{rcl}
\mathcal{I}(\beta) & = & \bigcup\{\mathsf{P}_B(\alpha) \Rightarrow \alpha \mid \alpha \in \mathsf{P}_A(\{\beta\} \cup \mathsf{P}(\beta_1) \cup \cdots \cup \mathsf{P}(\beta_k))\} \\
& = & \bigcup\{\mathsf{P}_B(\alpha) \Rightarrow \alpha \mid \alpha \in \mathsf{P}_A(\beta) \cup \mathsf{P}_A(\mathsf{P}(\beta_1)) \cup \cdots \cup \mathsf{P}_A(\mathsf{P}(\beta_k))\} \\
& = & \bigcup\{\mathsf{P}_B(\alpha) \Rightarrow \alpha \mid \alpha \in \mathsf{P}_A(\beta)\} \cup \bigcup_i \bigcup\{\mathsf{P}_B(\alpha) \Rightarrow \alpha \mid \alpha \in \mathsf{P}_A(\mathsf{P}(\beta_i))\} \\
& = & \bigcup\{\mathsf{P}_B(\alpha) \Rightarrow \alpha \mid \alpha \in \mathsf{P}_A(\beta)\} \cup \bigcup_i \mathcal{I}(\beta_i)
\end{array}$$

To prove Claim (1), we check that every element of $\mathcal{I}(\beta)$ is entailed by $A$. Indeed, $A \models_{\mathcal{T}} \mathcal{I}(\beta_i)$ holds by the induction hypothesis, and $A \models_{\mathcal{T}} \mathsf{P}_B(\alpha) \Rightarrow \alpha$ follows directly from the defining property of $\mathsf{P}_B$.

The defining property $B, \mathsf{P}_A(\beta) \models_{\mathcal{T}} \beta$ of $\mathsf{P}_A$ reduces proving Claim (2) to proving $B, \mathcal{I}(\beta) \models_{\mathcal{T}} \alpha$, for every $\alpha \in \mathsf{P}_A(\beta)$. Since $(\mathsf{P}_B(\alpha) \Rightarrow \alpha)$ is in $\mathcal{I}(\beta)$, it suffices to prove that $B, \mathcal{I}(\beta) \models_{\mathcal{T}} \mathsf{P}_B(\alpha)$. And indeed, $\mathsf{P}_B(\alpha)$ is a subset of $\{\beta_1, \ldots, \beta_k\}$, and $B, \mathcal{I}(\beta) \models_{\mathcal{T}} \beta_i$ holds by induction hypothesis. □

Lemma 6.6 is the induction vehicle for the following main result.

**Theorem 6.7.** *Let $(S_A, S_B, \sqsubset)$ be a successful $\mathcal{T}$-interpolation run for $A$ and $B$ and let $\mathcal{I}$ be defined as above. The formula $\bigwedge \mathcal{I}(\mathsf{false})$ is a $\mathcal{T}$-interpolant of $A$ and $B$.*

*Proof.* Since $\mathsf{false} \in S_B$, we can instantiate Lemma 6.6 with $\beta$ equal to $\mathsf{false}$. The free symbols occurring in $\mathcal{I}(\mathsf{false})$ are shared by $A$ and $B$ because, by construction, $\mathcal{I}$ returns $\Sigma_I$-formulas. □

### 6.3. Interpolation runs from local refutations.

The next question is how to construct successful interpolation runs for $A$ and $B$. One way is to extract them from proofs of $\mathcal{T}$-unsatisfiability of $A \cup B$ in a suitable proof system. We define a fairly general notion of a proof system and show that any refutation of $A \cup B$ in the system that is local in the sense of Jhala and McMillan [JM06] contains a successful interpolation run.

6.3.1. *Proofs and proof systems.* A PROOF RULE is a binary relation between finite sets of formulas and formulas. Any pair $\varphi_1, \ldots, \varphi_n \vdash \varphi$, with $n \geq 0$, in a proof rule, usually written as

$$\frac{\varphi_1 \quad \cdots \quad \varphi_n}{\varphi},$$

is an INFERENCE STEP with PREMISES $\varphi_1, \ldots, \varphi_n$ and CONCLUSION $\varphi$. The conclusion of an inference step with an empty set of premises is an AXIOM. A PROOF SYSTEM is a set of proof rules. A proof rule is SOUND with respect to a theory $\mathcal{T}$ if $\varphi_1, \ldots, \varphi_n \models_{\mathcal{T}} \varphi$ for each inference step $\varphi_1, \ldots, \varphi_n \vdash \varphi$ of the rule.

**Definition 6.8.** For every proof system $\mathcal{R}$, formula $\varphi$ and set of formulas $S$, a PROOF OF $\varphi$ FROM $S$ IN $\mathcal{R}$ is a labelled tree defined inductively as follows.

(1) If $\varphi \in S$, the one-node tree with root labelled $\varphi$ is a proof of $\varphi$ from $S$ in $\mathcal{R}$;
(2) if $\varphi_1, \ldots, \varphi_n \vdash_{\mathcal{R}} \varphi$ is an inference step of $\mathcal{R}$ and $\mathsf{D}_i$ a proof of $\varphi_i$ from $S$ in $\mathcal{R}$ for $i = 1, \ldots, n$, then the tree $\mathsf{D}$ with root $\varphi$ and immediate subtrees $\mathsf{D}_1, \ldots, \mathsf{D}_n$ is a proof of $\varphi$ from $S$ in $\mathcal{R}$. The roots of $\mathsf{D}_1, \ldots, \mathsf{D}_n$ are the PARENTS of the root of $\mathsf{D}$.

A REFUTATION of $S$ in $\mathcal{R}$ is a proof of $\mathsf{false}$ from $S$ in $\mathcal{R}$. □

In the following, we will identify nodes of a proof with their labels when this does not cause confusion. Observe that if all the rules of $\mathcal{R}$ are sound with respect to a theory $\mathcal{T}$, then $S \models_{\mathcal{T}} \varphi$ for each proof in $\mathcal{R}$ of a formula $\varphi$ from a set of formulas $S$. In particular, any set of formulas that has a refutation in $\mathcal{R}$ is $\mathcal{T}$-unsatisfiable.

Extending the terminology introduced in §4.3, we say that an inference step in $\mathcal{R}$ is $A$-COLORABLE (resp., $B$-COLORABLE) if the formulas in the inference step are all $A$-colorable (resp., all $B$-colorable). We define a proof of a formula $\varphi$ from $A \cup B$ in $\mathcal{R}$ to be LOCAL if every inference step in the proof is $A$- or $B$-colorable. An example of local proof is shown in Figure 7.

6.3.2. *Constructing interpolation runs.* Fix any proof system $\mathcal{R}$ that is sound for $\mathcal{T}$. We show that from any local proof $\mathsf{D}$ from $A \cup B$ in $\mathcal{R}$, we can construct a $\mathcal{T}$-interpolation run $(S_A, S_B, \sqsubset)$ so that if $\mathsf{D}$ is a refutation then $\mathsf{false} \in S_B$. (Then, the function $\mathcal{I}$ can be used to produce a $\mathcal{T}$-interpolant of $A$ and $B$ as shown in §6.2.)

Let $\mathsf{D}$ be a local refutation of $A \cup B$ in $\mathcal{R}$. Without loss of generality we can assume that $(i)$ if two nodes of $\mathsf{D}$ have the same label, then they are roots of structurally identical subtrees of $\mathsf{D}$, and $(ii)$ the parents of $\mathsf{false}$ in $\mathsf{D}$ are all $B$-colorable. Local refutations that do not satisfy Requirement $(ii)$ can be modified by replacing $\mathsf{false}$ with a new logical constant $\mathsf{false}'$ interpreted in the same way and then adding a final, $B$-colorable inference, $\mathsf{false}' \vdash \mathsf{false}$.

Define $\sqsubset$ as the relation on the labels of $\mathsf{D}$ such that $\varphi \sqsubset \psi$ iff $\varphi$ is an ancestor of $\psi$ in $\mathsf{D}$. By the assumptions on $\mathsf{D}$, the (finite) relation $\sqsubset$ is acyclic. Hence, both $\sqsubset$ and its inverse are well founded.

If we cut $\mathsf{D}$ at a node $\varphi$, we obtain two local proofs in $\mathcal{R}$: a local proof of $\varphi$ from $A \cup B$ (the tree rooted at $\varphi$), and a local proof of $\mathsf{false}$ from $A \cup B \cup \{\varphi\}$ (the remaining tree, with same root as $\mathsf{D}$ and $\varphi$ as one of its leafs). More generally, we can decompose $\mathsf{D}$ into several smaller local proofs by cutting it repeatedly at different nodes.

**Definition 6.9.** A pair $T_A, T_B$ of sets of nodes in $\mathsf{D}$ is a COLORING CUT of $\mathsf{D}$ if

(1) all nodes in $T_A \cup T_B$ are $AB$-colorable;
(2) $T_A$ and $T_B$ are disjoint, and $\mathsf{false}$ is in $T_B$;
(3) for all $\alpha \in T_A$ and $\psi \in T_A \cup (B \setminus T_B)$ with $\psi \sqsubset \alpha$, there is a $\beta \in T_B$ such that $\psi \sqsubset \beta \sqsubset \alpha$;
(4) for all $\beta \in T_B$ and $\psi \in T_B \cup (A \setminus T_A)$ with $\psi \sqsubset \beta$, there is a $\alpha \in T_A$ such that $\psi \sqsubset \alpha \sqsubset \beta$.

It is simple to verify that cutting $\mathsf{D}$ at the nodes of $T_A \cup T_B$, where $T_A, T_B$ is a coloring cut, decomposes $\mathsf{D}$ into colorable proofs. More precisely, every resulting smaller proof rooted at a node of $T_A$ (resp., $T_B$) consists of $A$-colorable (resp., $B$-colorable) nodes.

**Example 6.10.** Let $\mathcal{T}$ be some arbitrary theory with a signature consisting of the predicate symbols $r, t$ and such that $\models_{\mathcal{T}} \forall x.(r(x) \Rightarrow t(x))$. Then, let

$$\begin{aligned} A &= \{\forall u.(p(u) \wedge q(v,u) \Rightarrow r(u)),\ p(a),\ r(b) \vee q(fa,a)\}, \\ B &= \{\forall v.(s(v) \wedge r(v) \Rightarrow r(fv)),\ \forall x.s(x),\ \neg r(b),\ \neg t(fa)\}\ . \end{aligned}$$

where $p, q, a, b, f$ and $s$ are non-theory symbols. The proof in Figure 7 is a local refutation of $A \cup B$. The exact proof system used to build the refutation is not important here. Simply observe that each inference step is sound with respect to $\mathcal{T}$, which shows that $A \cup B$ is $\mathcal{T}$-unsatisfiable.

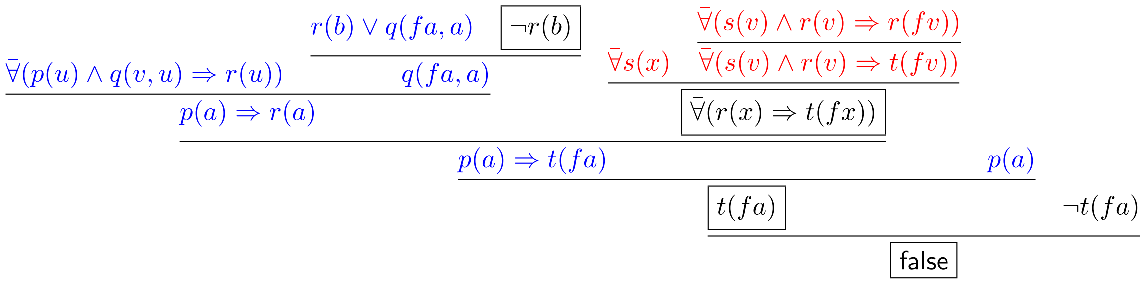
$r(b) \vee q(fa,a)$
$\neg r(b)$
$\bar\forall(s(v) \wedge r(v) \Rightarrow r(fv))$
$\bar\forall(p(u) \wedge q(v,u) \Rightarrow r(u))$
$q(fa,a)$
$\bar\forall s(x)$
$\bar\forall(s(v) \wedge r(v) \Rightarrow t(fv))$
$p(a) \Rightarrow r(a)$
$\bar\forall(r(x) \Rightarrow t(fx))$
$p(a) \Rightarrow t(fa)$
$p(a)$
$t(fa)$
$\neg t(fa)$
false

Figure 7: A local refutation $\mathsf{D}$ of $A \cup B$. The symbols $r$ and $t$ are from the theory's signature. Of the remaining symbols, $p$ and $q$ occur only in $A$, $s$ occurs only in $B$, and $a, b$ and $f$ occur in both. The boxed formulas are those in the coloring cut in Example 6.10.

A coloring cut of $\mathsf{D}$ is given by the sets

$$T_A = \{t(fa)\} \quad \text{and} \quad T_B = \{\neg r(b), \forall x.(r(x) \Rightarrow t(fx)), \mathsf{false}\} .$$

Note that the last inference step (the one with conclusion $\mathsf{false}$) is both $A$- and $B$-colorable. In the cut, however, it is essentially seen as a $B$-colored step. □

Every coloring cut induces a successful interpolation run.

**Theorem 6.11.** *If $S_A, S_B$ is a coloring cut, then $(S_A, S_B, \sqsubset)$ is a successful $\mathcal{T}$-interpolation run for $A$ and $B$.*

*Proof.* It is enough to define functions $\mathsf{P}_A$ and $\mathsf{P}_B$ satisfying Definition 6.5.

For each $\alpha \in S_A$, let $\mathsf{D}_\alpha$ be the proof of $\alpha$ in the decomposition of $\mathsf{D}$ defined by the coloring cut $S_A, S_B$. Define

$$\mathsf{P}_A(\alpha) = \{\beta \in S_B \mid \beta \text{ is a leaf of } \mathsf{D}_\alpha\} .$$

Clearly, $\beta \sqsubset \alpha$ for all $\beta \in \mathsf{P}_A(\alpha)$. To show that $A, \mathsf{P}_A(\alpha) \models_{\mathcal{T}} \alpha$ we show that every leaf of $\mathsf{D}_\alpha$ is in $A \cup \mathsf{P}_A(\alpha)$. Now, every leaf $\varphi$ of $\mathsf{D}_\alpha$ is either a leaf of $\mathsf{D}$ (so an element of $A \cup B$), or a cut node (an element of $S_A \cup S_B$). Since $\varphi \sqsubset \alpha$, it follows from the third defining property of coloring cuts, that $\varphi \notin S_A \cup B$ (otherwise, we would be able to cut $\mathsf{D}_\alpha$ at a node between $\varphi$ and $\alpha$). Thus, $\varphi \in S_B \cup A$.

The function $\mathsf{P}_B$ is defined similarly. □

**Example 6.12.** The $\mathcal{T}$-interpolation run induced by the coloring cut in Example 6.10 can be described informally in terms of the interpolation game as follows. In the first round, the $B$-prover adds to $S_B$ the formulas $\beta_1 = \neg r(b)$ and $\beta_2 = \forall x.(r(x) \Rightarrow t(fx))$, each with an empty set of $A$-premises (*i.e.*, $\mathsf{P}_A(\beta_1) = \mathsf{P}_A(\beta_2) = \varnothing$). In the second round, the $A$-prover adds to $S_A$ the formula $\alpha_1 = t(fa)$, with $\mathsf{P}_B(\alpha_1) = \{\beta_1, \beta_2\}$. In the third and final round, the $B$-prover adds $\mathsf{false}$ to $S_B$, with $\mathsf{P}_A(\mathsf{false}) = \{\alpha_1\}$.

The $\mathcal{T}$-interpolant computed by the function $\mathcal{I}$, defined in §6.2, from this interpolation run is $\beta_1 \wedge \beta_2 \Rightarrow \alpha_1$, as shown below.

$$\begin{array}{lcl}
\mathsf{P}(\beta_1) & = & \{\alpha_1\} \cup \mathsf{P}(\mathsf{P}_B(\mathsf{P}_A(\beta_1))) = \{\alpha_1\} \cup \varnothing \\
 & = & \{\alpha_1\} \\
 & & \\
\mathsf{P}(\beta_2)) & = & \{\beta_2\} \cup \mathsf{P}(\mathsf{P}_B(\mathsf{P}_A(\beta_2)))) = \{\beta_2\} \cup \varnothing \\
 & = & \{\beta_2\} \\
 & & \\
\mathsf{P}(\mathsf{false}) & = & \{\mathsf{false}\} \cup \mathsf{P}(\mathsf{P}_B(\mathsf{P}_A(\mathsf{false}))) \\
 & = & \{\mathsf{false}\} \cup \mathsf{P}(\mathsf{P}_B(\alpha_1)) = \{\mathsf{false}\} \cup \mathsf{P}(\beta_1) \cup \mathsf{P}(\beta_2) \\
 & = & \{\mathsf{false},\, \beta_1,\, \beta_2\} \\
 & & \\
\mathsf{P}_A(\mathsf{P}(\mathsf{false})) & = & \mathsf{P}_A(\mathsf{false}) \cup \mathsf{P}_A(\beta_1) \cup \mathsf{P}_A(\beta_2) = \{\alpha_1\} \cup \varnothing \cup \varnothing \\
 & = & \{\alpha_1\} \\
 & & \\
\mathcal{I}(\mathsf{false}) & = & \bigcup\{\mathsf{P}_B(\alpha) \Rightarrow \alpha \mid \alpha \in \mathsf{P}_A(\mathsf{P}(\mathsf{false}))\} \\
 & = & \{\beta_1 \wedge \beta_2 \Rightarrow \alpha_1\}
\end{array}$$

□

We stress that computing a coloring cut from local refutations is just one way to produce interpolation runs. For specific theories, other mechanisms are possible. A crucial point, however, is that local refutations always admit a coloring cut. In fact, with $\mathsf{D}$ and with $\sqsubset$ as defined in §6.3.2, a coloring cut of $\mathsf{D}$ is provided by the sets $S_A$ and $S_B$ defined inductively as follows over the set of $AB$-colorable nodes $\varphi$ of $\mathsf{D}$:

(1) $\mathsf{false} \in S_B$;
(2) if $\varphi \sqsubset \beta$ for some $\beta \in S_B$, $\varphi$ is a leaf from $A$ or has a non-$B$-colorable parent, and $\varphi$ is $\sqsubset$-maximal with these properties[2], then $\varphi \in S_A$;
(3) if $\varphi \sqsubset \alpha$ for some $\alpha \in S_A$, $\varphi$ is a leaf from $B$ or has a non-$A$-colorable parent, and $\varphi$ is $\sqsubset$-maximal with these properties, then $\varphi \in S_B$.

Different coloring cut algorithms produce different interpolation runs, and therefore different interpolants. The inductive definition above aims at minimizing the cardinality of $S_A \cup S_B$[3] and so is likely to produce smaller interpolants. If there is a need to find interpolants optimal in some other sense, one can hope that the problem will translate into a meaningful optimization problem for coloring cuts.

## 7. Conclusion

Our study of interpolation for the theory of equality was motivated by the central role this theory plays in SMT solving, and by the practical applicability of interpolant-producing SMT solvers in model checking. The algorithm we presented is easy to implement on top of the standard congruence closure procedure. It generates interpolants of a simple logical form and smaller size than those produced by the alternative method.

[2] That is, there is no $AB$-colorable $\varphi'$ with a non-$B$-colorable parent such that $\varphi \sqsubset \varphi' \sqsubset \beta$.

[3] In this sense, it is analogous to the congruence path factorization used in §4.4, where each relevant path is broken into *maximal* subpaths consisting of equally colored edges. In both cases, the intent is to minimize the number of *color switches*, so to speak—the number of factors in one case and the size of the coloring cut in the other.

We identified *congruence graphs* as a convenient structure to represent proofs in *EUF* and to derive interpolants. The possibilities for global analysis and transformations of these graphs go beyond what we have explored. Our algorithm provides a basis for further refinement and multiple implementations. This flexibility may prove useful when the notion of interpolant quality is better understood.

The heart of our algorithm—the generation of an interpolant from a suitably colored congruence graph—is not *EUF*-specific. We showed that behind it is a general *interpolation game* and a general mechanism for deriving interpolants from suitably colored (*local*) proofs.

## Acknowledgement

We thank Alberto Griggio for providing us with the interpolation benchmarks used in [CGS08], and with a *MathSAT* executable for benchmarking. We also thank the anonymous reviewers for their thoughtful comments on a preliminary version of this work, and their suggestions for improving the presentation.

## References

[BST11] Clark Barrett, Aaron Stump, and Cesare Tinelli. The Satisfiability Modulo Theories Library (SMT-LIB). `http://www.SMT-LIB.org`, 2011.

[CGS08] Alessandro Cimatti, Alberto Griggio, and Roberto Sebastiani. Efficient interpolant generation in satisfiability modulo theories. In C. R. Ramakrishnan and Jakob Rehof, editors, *TACAS*, volume 4963 of *LNCS*, pages 397–412. Springer, 2008.

[Cra57] William Craig. Three uses of the Herbrand-Gentzen theorem in relating model theory and proof theory. *Journal of Symbolic Logic*, 22(3):269–285, 1957.

[DNS05] D. Detlefs, G. Nelson, and J. B. Saxe. Simplify: a theorem prover for program checking. *Journal of ACM*, 52(3):365–473, 2005.

[FGG$^+$09] Alexander Fuchs, Amit Goel, Jim Grundy, Sava Krstić, and Cesare Tinelli. Ground interpolation for the theory of equality. In S. Kowalewski and A. Philippou, editors, *Proceedings of the 15th International Conference on Tools and Algorithms for the Construction and Analysis of Systems (York, UK)*, volume 5505 of *Lecture Notes in Computer Science*, pages 413–427. Springer, 2009.

[Ghi05] Silvio Ghilardi. Model-theoretic methods in combined constraint satisfiability. *Journal of Automated Reasoning*, 33(3–4):221–249, 2005.

[GKT09] Amit Goel, Sava Krstić, and Cesare Tinelli. Ground interpolation for combined theories. In R. Schmidt, editor, *Proceedings of the 22nd International Conference on Automated Deduction (Montreal, Canada)*, volume 5663 of *Lecture Notes in Artificial Intelligence*, pages 183–198. Springer, 2009.

[JCG08] Himanshu Jain, Edmund M. Clarke, and Orna Grumberg. Efficient craig interpolation for linear diophantine (dis)equations and linear modular equations. In *Proceedings of the 20th International Conference on Computer-Aided Verification*, pages 254–267, 2008.

[JM05] Ranjit Jhala and Kenneth L. McMillan. Interpolant-based transition relation approximation. In Kousha Etessami and Sriram K. Rajamani, editors, *Proceedings of 17th International Conference on Computer Aided Verification (Edinburgh, Scotland, UK)*, volume 3576 of *Lecture Notes in Computer Science*, pages 39–51. Springer, 2005.

[JM06] Ranjit Jhala and Kenneth L. McMillan. A practical and complete approach to predicate refinement. In *Proceedings of the 12th International Conference on Tools and Algorithms for the Construction and Analysis of Systems*, pages 459–473, 2006.

[KMZ06] Deepak Kapur, Rupak Majumdar, and Calogero G. Zarba. Interpolation for data structures. In Michal Young and Premkumar T. Devanbu, editors, *SIGSOFT FSE*, pages 105–116. ACM, 2006.

[McM03] Ken McMillan. Interpolation and SAT-based model checking. In W. A. Hunt Jr. and F. Somenzi, editors, *CAV*, volume 2725 of *LNCS*, pages 1–13. Springer, 2003.

[McM05a] Kenneth L. McMillan. Applications of Craig interpolants in model checking. In Nicolas Halbwachs and Lenore D. Zuck, editors, *TACAS*, volume 3440 of *LNCS*, pages 1–12. Springer, 2005.

[McM05b] Kenneth L. McMillan. An interpolating theorem prover. *Theoretical Computer Science*, 345(1):101–121, 2005.

[McM06] Kenneth L. McMillan. Lazy abstraction with interpolants. In T. Ball and R. Jones, editors, *CAV*, volume 4144 of *LNCS*, pages 123–136. Springer, 2006.

[NO79] Greg Nelson and Derek C. Oppen. Simplification by cooperating decision procedures. *ACM Transactions on Programming Languages and Systems*, 1(2):245–257, 1979.

[NO80] Greg Nelson and Derek C. Oppen. Fast decision procedures based on congruence closure. *Journal of the ACM*, 27(2):356–364, 1980.

[NO05] Robert Nieuwenhuis and Albert Oliveras. Proof-producing congruence closure. In J. Giesl, editor, *RTA*, volume 3467 of *LNCS*, pages 453–468. Springer, 2005.

[Pud97] Pavel Pudlák. Lower bounds for resolution and cutting planes proofs and monotone computations. *Journal of Symbolic Logic*, 62(3), 1997.

[Var08] Various. Decision Procedure Toolkit. `http://sourceforge.net/projects/dpt`, 2008.

[YM05] Greta Yorsh and Madanlal Musuvathi. A combination method for generating interpolants. In Robert Nieuwenhuis, editor, *CADE*, volume 3632 of *LNCS*, pages 353–368. Springer, 2005.