Ground Interpolation for the Theory of Equality

Alexander Fuchs[1], Amit Goel[2], Jim Grundy[2], Sava Krstić[2], and Cesare Tinelli[1]

[1] Department of Computer Science, The University of Iowa
[2] Strategic CAD Labs, Intel Corporation

Abstract. Given a theory $\mathcal{T}$ and two formulas A and B jointly unsatisfiable in $\mathcal{T}$, a *theory interpolant* of A and B is a formula I such that (i) its non-theory symbols are shared by A and B, (ii) it is entailed by A in $\mathcal{T}$, and (iii) it is unsatisfiable with B in $\mathcal{T}$. Theory interpolants are used in model checking to accelerate the computation of reachability relations. We present a novel method for computing ground interpolants for ground formulas in the theory of equality. Our algorithm computes interpolants from colored congruence graphs representing derivations in the theory of equality. 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.

1 Introduction

The *Craig Interpolation Theorem* [4] 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 [7], involve computation of interpolants in propositional logic or in quantifier-free logics with (combinations of) theories such as the theory of equality, linear real arithmetic, arrays, and finite sets [8, 14, 6, 3]. There are now techniques that use interpolants to obtain property-driven approximate reachability sets or transition relations, and also to compute refinements for predicate abstraction. Experimental results show that interpolation-based techniques are often superior to previous ones.

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 [8]. It derives interpolants from proofs in a formal system that contains rules for the basic properties of equality.

In this paper, 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. Working with graphs makes it possible to exploit the global structure of proofs in order to streamline the interpolant generation. Our 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.

Our interpolation algorithm is described and proved correct in §5. In §4, we give a series of examples to highlight important aspects of the algorithm. Its inherent game-like structure is explained at a high level in §3. A detailed comparison with McMillan's method is given in §6, together with experimental data on a set of benchmarks derived from the SMT-LIB suite.

2 Ground Theory Interpolation

Interpolation is a property of fragments of logical theories. To state it for a fragment $\mathcal{F}$, we need know only the following:

- a partition of symbols used to build formulas in $\mathcal{F}$ into *logical* and *non-logical*
- the *entailment relation* $A \models B$ between formulas in $\mathcal{F}$

Let $\mathcal{F}(X)$ be the set of all formulas in $\mathcal{F}$ whose non-logical symbols belong to 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 $A, B \models \mathsf{false}$, there exists $I \in \mathcal{F}(X \cap Y)$ such that $A \models I$ and $B, I \models \mathsf{false}$. The formula I will be called the INTERPOLANT for the pair A, B. Note the asymmetry: $\neg I$ is an interpolant for the pair B, A.

The classic Craig's theorem says that the set of first-order logic formulas has the interpolation property. (The non-logical symbols are the predicate and function symbols, and variables.) It also implies a "modulo theory" generalization, where, for a given first-order theory $\mathcal{T}$ over a signature Σ, $\mathcal{F}$ is the set of all Σ-formulas, $\models$ is the $\mathcal{T}$-entailment, and the symbols of Σ are treated as logical. The case where $\mathcal{T}$ and Σ are empty is Craig's original theorem.

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

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 [6].

If we want an efficient algorithm for ground $\mathcal{T}$-interpolation, it suffices to have one that works for inputs A and B that are *conjunctions of ground literals*. To see this, refer to [8, 3] for a description of a general mechanism to combine such a restricted interpolation procedure with a method for computing interpolants in propositional logic [12, 7].

Example 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 mutually inconsistent, as

witnessed by the linear combination with *positive* coefficients $2 \cdot (3x - z - 2 \leq 0) + 3 \cdot (-2x + z - 1 \leq 0) + 1 \cdot (3y - 4z + 12 \leq 0) + 3 \cdot (-y + z - 1 \leq 0)$ that 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., [3].

3 Interpolation as a Cooperative Game

There is a way to compute interpolants in a theory $\mathcal{T}$ as a cooperative game between two deductive provers for $\mathcal{T}$—possibly two copies of the same prover. We give an informal description of the *ground interpolation game*. The same idea applies with minor modifications to other interpolation problems, but the success of the method is not guaranteed in general.

Let A and B be two sets of ground formulas such that $A \cup B$ is $\mathcal{T}$-unsatisfiable. We allow A and B to contain *free symbols*, i.e., predicate and function symbols not in the signature Σ of $\mathcal{T}$. Let Σ_I be the *shared signature*, the expansion of Σ with the free symbols occurring in both A and B.

The interpolation game. The participants are the *A-prover* and the *B-prover*. The game starts with $S_A = S_B = \varnothing$ and proceeds so that at each turn one of the following happens:

- the A-prover adds to S_A a new ground Σ_I-clause C such that $A, S_B \models_{\mathcal{T}} C$
- the B-prover adds to S_B a new ground Σ_I-clause C such that $B, S_A \models_{\mathcal{T}} C$

The game ends when the B-prover could add false to S_B. An interpolant for A and B can then be computed as follows.

Let C be a clause of S_A and let $C_1, \ldots, C_n$ be the clauses of S_B actually used by the A-prover to derive C. Let us call $C_1, \ldots, C_n$ the B-PREMISES of C and call the formula $C_1 \wedge \cdots \wedge C_n \Rightarrow C$ the A-JUSTIFICATION of C. Similarly, we can define the set A-PREMISES of each clause $C \in S_B$. For $C \in S_B$, define the CUMULATIVE SET OF PREMISES $\mathcal{P}(C)$ recursively by

$$\mathcal{P}(C) = \{C\} \cup \bigcup \{\mathcal{P}(D) \mid D \text{ is a } B\text{-premise of an } A\text{-premise of } C\}$$

It can be shown that the conjunction of A-justifications of A-premises of clauses in $\mathcal{P}(\mathsf{false})$ is an interpolant for A, B.

For some theories with the ground interpolation property, the game admits *complete strategies*, guaranteed to end the game when A and B are jointly unsatisfiable in the theory. Depending on the theory, these strategies can be considerably more restrictive in the choice of clauses to propagate from one prover to the other. For instance, by results of [13], when the theory is *Σ-convex*,[3] it is enough to propagate just positive unit clauses in all rounds of the game except

[3] 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.

possibly the last. In that case, all interpolants computed will be conjunctions of Horn clauses.

The interpolation method we describe in §5.5 for the theory of equality can be understood as an implementation of this interpolation game, with clause propagation restricted to (positive) equalities. The method does not literally use two provers; it rather takes a proof of unsatisfiability of $A \cup B$, treats it as if cooperatively generated by an A-prover and a B-prover, and extracts an interpolant from it as sketched above.

Remark 1. Nelson and Oppen's method for combining decision procedures [9] is similar to the interpolation game. The main differences are that for the Nelson-Oppen procedure (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 witness 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 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 can be found in [5].

4 *EUF* Interpolation Examples

Let us fix the terminology first. A ground *EUF* term is either a free constant or (inductively) an application $f(t_1, \ldots, t_n)$ of an n-ary function symbol f to terms $t_1, \ldots, t_n$. An *EUF* literal is an equality $s = t$ between terms, or the negation $s \neq t$ of it (a disequality). We use $=$ also to write equalities at the meta-level, relying on the context to disambiguate. For convenience, we treat all equalities modulo symmetry: an equality $s = t$ will stand indifferently for $s = t$ or $t = s$.

Example 2. The picture in Figure 1(a) demonstrates the inconsistency of $A = \{z_1 = x_1, x_1 = z_2, z_2 = x_2, x_2 = f(z_3), f(z_3) = x_3, x_3 = z_4\}$ and $B = \{z_1 = y_1, y_1 = f(z_2), f(z_2) = y_2, y_2 = z_3, z_3 = y_3, y_2 \neq z_4\}$ that follows by transitivity of equality. The interpolant is the equality $z_1 = z_4$ that summarizes the A-chain. For the variation in Figure 1(b), the 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, modify Figure 1(b) by moving the disequality sign to the edge $\langle x_3, z_4 \rangle$. The interpolant changes to $z_1 = z_2 \wedge f(z_3) \neq z_4 \wedge f(z_2) = z_3$. In light of the interpolation game (§3) where the A-prover and the B-prover alternate in deriving sets of equalities, we can see that in all these example the game can end after the second round: A generates some implied literals in the shared language, and B is inconsistent with their conjunction. (Here and elsewhere in this section, a *round* of the game is a sequence of steps done by the same prover.)

Example 3. When the inconsistency of $A \cup B$ requires congruence reasoning, an interpolant in the form of a conjunction of equalities need not exist. Let

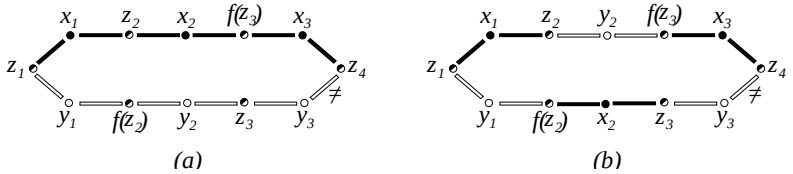

Fig. 1: Transitivity chains with dark (light) edges representing A-literals (B-literals). For the vertex coloring convention, see Example 9.

$A = \{u_1 = x \cdot u_0,\ v_1 = x \cdot v_0\}$ and $B = \{u_0 = v_0,\ u_1 \neq v_1\}$. (The dot is an infix function symbol.) There are no equalities implied 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 implied by A. However, A does imply it under the condition $u_0 = v_0$ that B provides. That gives us the interpolant $u_0 = v_0 \Rightarrow u_1 = v_1$. The game can take 3 rounds in which $u_0 = v_0$ is derived first (by B), then $u_1 = v_1$ (by A), then false (by B).

Example 4. Generalizing the previous example, consider this matrix-organized set of literals:

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

Let A be the set of equalities occurring in the odd-numbered columns (count columns starting from *one*!) of this matrix, and B be the set of the remaining equalities; see Figure 2. 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 $x_1, x_3, \ldots$.

The 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

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

Example 5. With $A = \{x = z_1,\ x \cdot z_2 = z_3\}$ and $B = \{y = z_2,\ z_1 \cdot y \neq z_3\}$, pictured in Figure 3, we can derive inconsistency 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 uses an equality from A and an equality from B ($x = z_1$ and $z_2 = y$), and cannot be derived by 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 our interpolant. The upshot is that creating an interpolant may require terms (in this case $z_1 \cdot z_2$) that do not occur in either A or B. See Lemma 1 below.

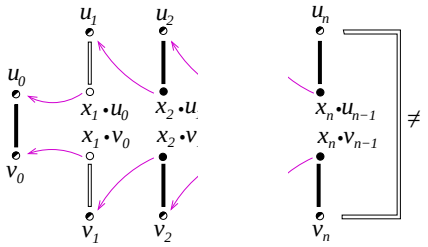

Fig. 2: A long derivation. (Example 4)

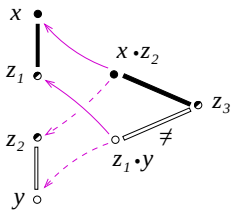

Fig. 3: Who derives $x \cdot x_2 = z_1 \cdot y$? (Example 5)

5 Interpolants From Congruence Closure

5.1 Congruence Closure

Satisfiability of sets of *EUF* literals can be determined by the CONGRUENCE CLOSURE ALGORITHM. The algorithm takes as inputs a finite subterm-closed set T of ground terms and a finite set E of ground equalities. Its state is an *undirected* graph Γ, 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 Γ. The algorithm proceeds as follows.

(**cc1**) Choose $s, t \in T$ 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 $s_1 \sim t_1$, $\ldots$, $s_k \sim t_k$.
  Then add the edge $\langle s, t \rangle$ to Γ.
(**cc2**) Repeat (**cc1**) for as long as possible.

Theorem 1. **[10, 11]** *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.* □

Let L be an arbitrary set of ground *EUF* literals. We have $L = L_= \cup L_\neq$, where the literals in $L_=$ and $L_\neq$ are equalities and disequalities respectively. To check whether L is satisfiable, it suffices to run the congruence closure algorithm with $E = L_=$ and T being the set of all terms occurring in L. By Theorem 1, L is satisfiable if and only if $s \not\sim t$ holds for every disequality $s \neq t$ in $L_\neq$. Note that L is unsatisfiable if and only if $L_= \cup \{\delta\}$ is unsatisfiable for some $\delta \in L_\neq$—the convexity property of *EUF*.

5.2 Congruence Graphs

Define a CONGRUENCE GRAPH over E to be any intermediate graph Γ obtainable by the congruence closure algorithm above. The assumption $s \not\sim t$ in (**cc1**) ensures that every congruence graph is acyclic. Thus, if $u \sim v$ in a congruence graph Γ, then there is a unique PATH connecting them. This path is denoted $\overline{uv}$. EMPTY PATHS are those of the form $\overline{uu}$.

The edges of Γ introduced by step (**cc1-a**) will be called BASIC; those introduced in step (**cc1-b**) are DERIVED. A derived edge $\langle f(u_1, \ldots, u_k), f(v_1, \ldots, v_k)\rangle$ has k PARENT PATHS $\overline{u_1v_1}$,...,$\overline{u_kv_k}$, some (but not all) of which may be empty.

Example 6. Each of the graphs in Figures 1–3, *when we delete from it the edge marked with the* $\neq$ *symbol*, is a congruence graph over the appropriate set of equalities $(A \cup B)_=$. All edges in these graphs are basic; the arrows indicate where derived edged can be added.

Example 7. Let $A = \{x_1 = z_1, z_2 = x_2, z_3 = f(x_1), f(x_2) = z_4, x_3 = z_5, z_6 = x_4, z_7 = f(x_3), f(x_4) = z_8\}$, $B = \{z_1 = z_2, z_5 = f(z_3), f(z_4) = z_6, y_1 = z_7, z_8 = y_2, y_1 \neq y_2\}$, and $E = (A \cup B)_=$. Figure 4(b) depicts a congruence graph over E. The basic edges are shown in Figure 4(a); each corresponds to an equality in E. Since f is unary, each of the three derived edges has one parent path.

5.3 Colorable Congruence Graphs

Let A and B be sets of literals and let Σ_A and Σ_B be the sets of symbols that occur in A and B respectively. Terms, literals, and formulas over Σ_A will be called A-COLORABLE. Define B-COLORABLE analogously, and then define AB-COLORABLE to mean both A-colorable and B-colorable. COLORABLE entities are those that are either A-colorable or B-colorable.

Example 8. In Example 5, $\Sigma_A = \{x, z_1, z_2, z_3, \cdot\}$ 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.

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 $s = t$. Finally, define a congruence graph to be COLORABLE if all its edges are colorable. Note that basic edges are always colorable.

Example 9. The congruence graphs derived from graphs in Figures 1–3 by removing their disequality edges are colorable. Half-filled vertices are AB-colorable, the dark ones are A-colorable but not B-colorable, and the light ones are B- but not A-colorable. If we add the derived edges $\langle x_i \cdot u_{i-1}, x_i \cdot v_{i-1}\rangle$ to the graph in Figure 2, they will be colorable; but if we add the derived edge $\langle x \cdot z_2, z_1 \cdot y\rangle$ to the graph in Figure 3, it will not be colorable.

Lemma 1. *If s and t are colorable terms and if $A, B \models s = t$, then there exists a colorable congruence graph over $(A \cup B)_=$ in which $s \sim t$.*

Proof. (Sketch) This is essentially Lemma 2 of [14], and the proof is constructive. Start with any congruence graph Γ with colorable vertices in which $s \sim t$ holds. If there are uncolorable edges, let $e = \langle f(u_1, \ldots, u_k), f(v_1, \ldots, v_k)\rangle$ be a minimal such edge in the derivation order. Thus, the parent paths $\overline{u_iv_i}$ 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 $\overline{u_iv_i}$

(which may be one of its endpoints). The term $f(w_1, \ldots, w_k)$ is AB-colorable, so we can replace e in Γ with two edges $\langle f(u_1, \ldots, u_k), f(w_1, \ldots, w_k)\rangle$ and $\langle f(w_1, \ldots, w_k), f(v_1, \ldots, v_k)\rangle$, both of which are colorable. Now repeat the process until all uncolorable edges of Γ are eliminated. □

5.4 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 of §3, this means choosing which prover derives an AB-equality in a situation when either of them could do it. In Figure 4(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.

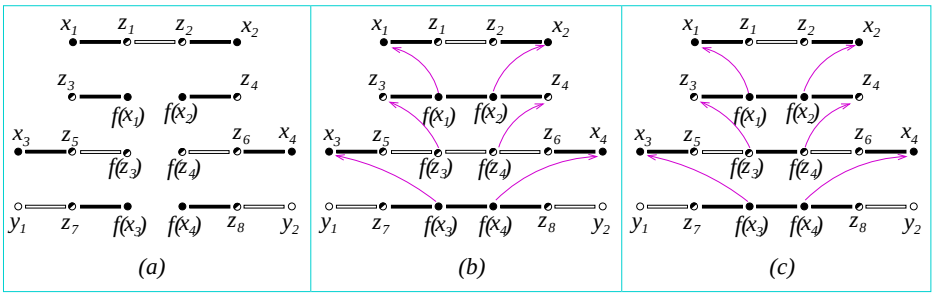

Fig. 4: Congruence graphs over $(A \cup B)_=$, with A and B from Example 7. The connection between a derived edge and its parent is indicated by a pair of arrows.

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 is a maximal subpath consisting of equally colored edges. Clearly, every path can be uniquely represented as a concatenation of its factors, the consecutive factors having distinct colors.

5.5 The Interpolation Algorithm

A path $\overline{uv}$ in a congruence graph represents the equality $u = v$ between its endpoints. We will write $[\![\pi]\!]$ to denote the equality represented by the path π. More generally, if P is a set of paths, $[\![P]\!]$ is the corresponding set of equalities.

For every path π in a colored congruence graph, define the associated B-PREMISE SET $\mathcal{B}(\pi)$, the A-JUSTIFICATION $J(\pi)$, and the INTERPOLANT $I(\pi)$:

$$\mathcal{B}(\pi) = \begin{cases} \bigcup\{\mathcal{B}(\sigma) \,|\, \sigma \text{ is a factor of } \pi\} & \text{if } \pi \text{ has} \geq 2 \text{ factors} \\ \{\pi\} & \text{if } \pi \text{ is a } B\text{-path} \\ \bigcup\{\mathcal{B}(\sigma) \,|\, \sigma \text{ is a parent of an edge of } \pi\} & \text{if } \pi \text{ is an } A\text{-path} \end{cases} \quad (1)$$

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

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

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

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

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

This is well defined because π_1, θ, π_2 are uniquely determined by π if θ is not empty, and if θ is empty, the way we write π 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 inputs jointly inconsistent (disjoint) sets A, B of literals, proceeds as follows.

- (**i1**) Run the congruence closure algorithm to find a congruence graph Γ over $(A \cup B)_=$ and a disequality $(s \neq t) \in A \cup B$ such that $s \sim t$ in Γ [§§5.1,5.2].
- (**i2**) Modify Γ if necessary so that it is colorable [§5.3], then color it [§5.4].
- (**i3**) If $(s \neq t) \in B$, return $I(\overline{st})$; if $(s \neq t) \in A$, return $I'(\overline{st})$.

Example 10. Let us run the algorithm for A, B in Example 7, using the colored congruence graph in Figure 4(b). Since $y_1 \neq y_2 \in B$, the interpolant is $I(\overline{y_1y_2}) = I(\overline{y_1z_7}) \wedge I(\overline{z_7z_8}) \wedge I(\overline{z_8y_2})$, and the first and third conjuncts are true. Thus, $I(\overline{y_1y_2}) = I(\overline{z_7z_8}) = J(\overline{z_7z_8}) \wedge \bigwedge\{I(\sigma) \,|\, \sigma \in \mathcal{B}(\overline{z_7z_8})\}$, so we need to compute $\mathcal{B}(\overline{z_7z_8})$. We have $\mathcal{B}(\overline{z_7z_8}) = \mathcal{B}(\overline{x_3x_4}) = \mathcal{B}(\overline{x_3z_5}) \cup \mathcal{B}(\overline{z_5z_6}) \cup \mathcal{B}(\overline{z_6x_4}) = \varnothing \cup \{\overline{z_5z_6}\} \cup \varnothing = \{\overline{z_5z_6}\}$. Thus, $J(\overline{z_7z_8}) = (z_5 = z_6 \Rightarrow z_7 = z_8)$, which we denote ϕ_1. Continue the main computation: $I(\overline{y_1y_2}) = \phi_1 \wedge I(\overline{z_5z_6}) = \phi_1 \wedge I(\overline{z_3z_4}) = J(\overline{z_3z_4}) \wedge \bigwedge\{I(\sigma) \,|\, \sigma \in \mathcal{B}(\overline{z_3z_4})\}$. 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, $J(\overline{z_3z_4}) = (z_1 = z_2 \Rightarrow z_3 = z_4)$, which we denote ϕ_2. Back to the main computation, $I(\overline{y_1y_2}) = \phi_1 \wedge \phi_2 \wedge I(\overline{z_1z_2}) = \phi_1 \wedge \phi_2 \wedge \mathsf{true} = \phi_1 \wedge \phi_2$. *Exercise:* Using the graph in Figure 4(c) results in a different interpolant: $I(\overline{y_1y_2}) = (z_5 = f(z_3) \wedge z_6 = f(z_4) \wedge z_1 = z_2 \Rightarrow z_7 = z_8)$.

5.6 Correctness

Theorem 2. *With any jointly inconsistent sets A,B of EUF literals as inputs, the EUF ground interpolation algorithm (§5.5) terminates, returning an interpolant for A,B that is a conjunction of Horn clauses.*

Termination of our recursive definitions and other inductive arguments are proved using a well-founded relation $\prec$ over paths. Define $\sigma \prec' \pi$ to hold when one of the following holds:

- π has more than one factor, and σ is one of them
- σ is a parent path of an edge of π

Define $\prec$ as the transitive closure of $\prec'$. 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.

$$\mathcal{A}(\pi) \;=\; \{A\text{-factors of } \pi\} \;\cup\; \mathcal{A}(\{\text{parent paths of } B\text{-edges of } \pi\}) \qquad (2)$$

$$\mathcal{B}(\pi) \;=\; \{B\text{-factors of } \pi\} \;\cup\; \mathcal{B}(\{\text{parent paths of } A\text{-edges of } \pi\}) \qquad (3)$$

Here and in the sequel, we use the convention $f(P) = \bigcup\{f(\sigma) \,|\, \sigma \in P\}$ for extending a set-valued function f defined on paths to a function defined on sets of paths. Observe that (3) is just a restatement of (1). Also, the arguments in the recursive calls are smaller than π 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 2. *Let π be an arbitrary non-empty path in a congruence graph Γ.*

(i) If π is an A-path, then $\mathcal{A}(\pi) = \{\pi\}$; otherwise, $\sigma \prec \pi$ for every $\sigma \in \mathcal{A}(\pi)$.
(ii) If $\sigma \in \mathcal{A}(\pi)$, then $\mathcal{A}(\sigma) \subseteq \mathcal{A}(\pi)$.
(iii) If the endpoints of π 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.

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

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

(iii) 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 π 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 π is either also an endpoint of a B-factor of π, or an endpoint of π itself. □

The following lemma justifies the names *A-premises* and *B-premises*. In accordance with the notation of §3, B-premises are the B-paths whose summaries, if added to A, make the derivation of $[\![\pi]\!]$ possible.

Lemma 3. *A, $[\![\mathcal{B}(\pi)]\!] \models [\![\pi]\!]$ and B, $[\![\mathcal{A}(\pi)]\!] \models [\![\pi]\!]$, for every path π in Γ.*

Proof. We prove the first claim only, by well-founded induction based on $\prec$. Viewing π 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 σ that is either a B-factor of π or a parent of an A-edge of π. In the first case, the claim clearly 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 $\mathcal{B}(\sigma) \subseteq \mathcal{B}(\pi)$, which is true by Lemma 2(ii). □

Define the CUMULATIVE SET OF PREMISES (cf. §3)

$$\mathcal{P}(\pi) \ = \ \{\pi\} \cup \mathcal{P}(\mathcal{B}(\mathcal{A}(\pi))) \tag{4}$$

Termination of this recursive definition follows from Lemma 2(i).

Lemma 4. $I(\pi) \ = \ \bigwedge \{J(\sigma) \,|\, \sigma \in \mathcal{A}(\mathcal{P}(\pi))\}$.

Proof. Let $\mathcal{P}'(\pi) = \mathcal{A}(\mathcal{P}(\pi))$. From (4), we have

$$\mathcal{P}'(\pi) \ = \ \mathcal{A}(\pi) \cup \mathcal{P}'(\mathcal{B}(\mathcal{A}(\pi))) \tag{5}$$

It suffices to check that

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

For the first case, suppose $\pi = \pi_1 \cdots \pi_k$ is the factorization of π. By definition of $\mathcal{A}$, we have $\mathcal{A}(\pi) = \mathcal{A}(\pi_1) \cup \cdots \cup \mathcal{A}(\pi_k)$. The desired equation $\mathcal{P}'(\pi) = \mathcal{P}'(\pi_1) \cup \cdots \cup \mathcal{P}'(\pi_k)$ then follows from (5). Assume now $\pi = e_1 \cdots e_k$ 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. Again, the desired equation $\mathcal{P}'(\pi) = \mathcal{P}'(E_1) \cup \cdots \cup \mathcal{P}'(E_k)$ follows from (5). Finally, assume that π is an A-path. Now $\mathcal{A}(\pi) = \{\pi\}$ and so $\mathcal{P}'(\pi) = \{\pi\} \cup \mathcal{P}'(\mathcal{B}(\pi))$, again by (5). □

Lemma 5. *$B, I(\pi) \models [\![\pi]\!]$, for every path π in Γ with B-colorable endpoints.*

Proof. We argue by induction along $\prec$. Let σ be an arbitrary A-premise of π and τ an arbitrary B-premise of σ. The endpoints of τ 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 τ and we have $B, I(\tau) \models [\![\tau]\!]$. From equation (5) we have $\mathcal{P}'(\tau) \subseteq \mathcal{P}'(\pi)$, so we can derive $I(\pi) \models I(\tau)$ using Lemma 4. Thus, $B, I(\pi) \models [\![\tau]\!]$ for every $\tau \in \mathcal{B}(\sigma)$. By Lemma 4, $I(\pi)$ contains $J(\sigma)$ as a conjunct; therefore, $B, I(\pi) \models [\![\sigma]\!]$. Since σ here is an arbitrary element of $\mathcal{A}(\pi)$, the second claim of Lemma 3 finishes the proof. □

Proof of Theorem 2 The algorithm terminates because all pertinent functions have been proven terminating.

Let $s \neq t$ be the critical disequality obtained in the step (**i1**) of the algorithm. Let π 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 ϕ be the returned formula—$I(\pi)$ in Case 1; $I'(\pi)$ in Case 2.

(i) ϕ is an AB-colorable conjunction of Horn clauses. For any σ with AB-colorable endpoints, $J(\sigma)$ is an AB-colorable Horn clause. If π has B-colorable endpoints, then so do all paths in $\mathcal{P}(\pi)$ and so, by Lemma 2(iii), all paths in $\mathcal{A}(\mathcal{P}(\pi))$ have AB-colorable endpoints. With Lemma 4, this proves Case 1. For Case 2, observe that if θ is empty, then $I(\theta) = [\![\theta]\!] = \mathsf{true}$; otherwise, θ has AB-colorable endpoints. Also, π_1 and π_2 are A-paths, so by the dual of Lemma 2(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) $A \models \phi$. By the first claim of Lemma 3, $A \models J(\sigma)$ holds for every σ. 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 3, the transitivity entailment $[\![\pi_1]\!], [\![\theta]\!], [\![\pi_2]\!] \models [\![\pi]\!]$, and the assumption $\neg[\![\pi]\!] \in A$.

(iii) $B, \phi \models \mathsf{false}$. In Case 1, we have $\neg[\![\pi]\!] \in B$, so Lemma 5 finishes the proof. In Case 2, Lemma 5 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 $I'(\pi)$. □

6 Comparison with McMillan's Algorithm

Our *EUF* ground interpolation algorithm is, as far as we know, the only alternative to McMillan's algorithm [8]. 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 false from an equality and its negation). The algorithm proceeds top down by annotating each intermediate derived equality $u = v$ (or false in the final step) with a quadruple of the

form $[u', v', \rho, \gamma]$, where u', v' are terms and ρ, γ 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 false, then $\rho \Rightarrow \gamma$ is an interpolant for A, B. It can be shown that ρ is always a conjunction of Horn clauses, and γ 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 2 (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 7 (Figure 4(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 10. In general, McMillan's algorithm accumulates B-justifications (duals of our $J(\sigma)$) in the ρ-part of the annotation and keeps them past their one-time use to derive a particular conjunct of γ.

The third difference is in creating auxiliary AB-terms ("equality interpolants", in the terminology of [14]) to split derivations of equalities in which one side is not A-colorable and the other is not B-colorable, as in Example 5. We introduce an absolute minimum of such terms in the preliminary step (**i2**) of our algorithm, where these terms are added to make the congruence graph colorable. In contrast, McMillan's algorithm introduces these terms "on-the-fly", as in the example illustrated in Figure 5. 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 $z_1 = z_2 \Rightarrow z_3 = z_4$.

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 Γ, 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.

In general, our algorithm produces smaller and simpler interpolants. For experimental confirmation, we used the state-of-the-art implementation of McMil-

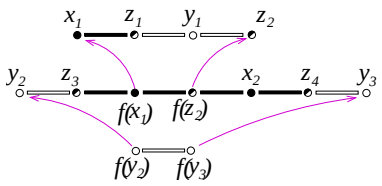

Fig. 5: A colored congruence graph for $A = \{x_1 = z_1, z_3 = f(x_1), f(z_2) = x_2, x_2 = z_4\}$ and $B = \{z_1 = y_1, y_1 = z_2, y_2 = z_3, z_4 = y_3, f(y_2) \neq f(y_3)\}$ with two derived edges $\langle f(x_1), f(z_2)\rangle$ and $\langle f(y_2), f(y_3)\rangle$.

lan's algorithm in *MathSAT* [3] and compared it against our interpolation-generating extension of the *DPT* solver [1]. Two other relevant components—the propositional interpolation algorithm, and the algorithm for combining propositional and theory interpolation in a DPLL($\mathcal{T}$) framework [8, 3]—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. These sizes being comparable, we can eliminate the differences in propositional reasoning as a cause for *DPT* producing smaller interpolants.

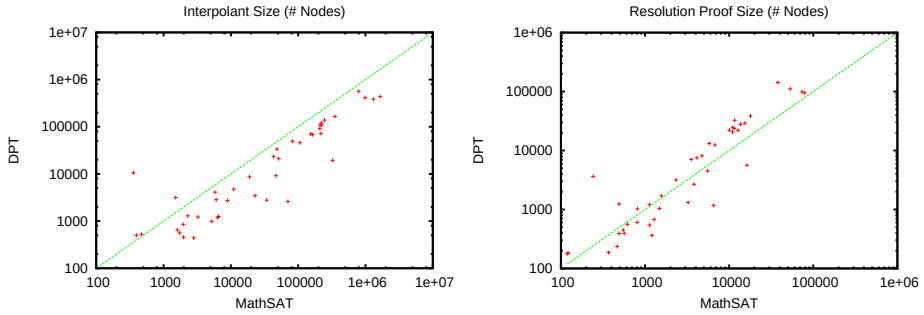

Fig. 6: *DPT vs. MathSAT* on 45 benchmarks from the *MathSAT* library derived by partitioning unsatisfiable SMT-LIB benchmarks [2].

We ran both solvers on 45 *EUF* interpolation benchmarks selected from the set of 100 that are used in [3]. (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 6 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.

7 Conclusion

Our analysis of the 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. The algorithm we obtained 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.

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.

Acknowledgment. We thank Alberto Griggio for providing interpolation benchmarks used in [3], and a *MathSAT* executable for benchmarking.

References

1. Decision Procedure Toolkit. `www.sourceforge.net/projects/DPT`, 2008.
2. C. Barrett, S. Ranise, A. Stump, and C. Tinelli. The Satisfiability Modulo Theories Library (SMT-LIB). `www.SMT-LIB.org`, 2008.
3. A. Cimatti, A. Griggio, and R. Sebastiani. Efficient interpolant generation in satisfiability modulo theories. In C. R. Ramakrishnan and J. Rehof, editors, *TACAS*, volume 4963 of *LNCS*, pages 397–412. Springer, 2008.
4. W. Craig. Three uses of the Herbrand-Gentzen theorem in relating model theory and proof theory. *Journal of Symbolic Logic*, 22(3):269–285, 1957.
5. S. Ghilardi. Model-theoretic methods in combined constraint satisfiability. *Journal of Automated Reasoning*, 33(3–4):221–249, 2005.
6. D. Kapur, R. Majumdar, and C. G. Zarba. Interpolation for data structures. In M. Young and P. T. Devanbu, editors, *SIGSOFT FSE*, pages 105–116. ACM, 2006.
7. K. 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.
8. K. L. McMillan. An interpolating theorem prover. *Theoretical Computer Science*, 345(1):101–121, 2005.
9. G. Nelson and D. C. Oppen. Simplification by cooperating decision procedures. *ACM Transactions on Programming Languages and Systems*, 1(2):245–257, 1979.
10. G. Nelson and D. C. Oppen. Fast decision procedures based on congruence closure. *Journal of the ACM*, 27(2):356–364, 1980.
11. R. Nieuwenhuis and A. Oliveras. Proof-producing congruence closure. In J. Giesl, editor, *RTA*, volume 3467 of *LNCS*, pages 453–468. Springer, 2005.
12. P. Pudlák. Lower bounds for resolution and cutting planes proofs and monotone computations. *Journal of Symbolic Logic*, 62(3), 1997.
13. C. Tinelli. Cooperation of background reasoners in theory reasoning by residue sharing. *Journal of Automated Reasoning*, 30(1):1–31, 2003.
14. G. Yorsh and M. Musuvathi. A combination method for generating interpolants. In R. Nieuwenhuis, editor, *CADE*, volume 3632 of *LNCS*, pages 353–368. Springer, 2005.