On Interpolation in Decision Procedures*

Maria Paola Bonacina and Moa Johansson

Dipartimento di Informatica, Università degli Studi di Verona
Strada Le Grazie 15, I-37134 Verona, Italy
mariapaola.bonacina@univr.it, moakristin.johansson@univr.it

Abstract. Interpolation means finding intermediate formulae between given formulae. When formulae decorate program locations, and describe sets of program states, interpolation may enable a program analyzer to discover information about intermediate locations and states. This mechanism has an increasing number of applications, that are relevant to program analysis and synthesis. We study interpolation in theorem proving decision procedures based on the DPLL($\mathcal{T}$) paradigm. We survey interpolation systems for DPLL, equality sharing and DPLL($\mathcal{T}$), reconstructing from the literature their completeness proofs, and clarifying the requirements for interpolation in the presence of equality.

1 Introduction

Automated deduction and program verification have always been connected, as described, for instance, in [34] and [1]. A theorem proving technique that has recently found application in verification is *interpolation*. Informally, interpolants are formula 'in between' other formulæ in a proof, containing only their shared symbols. Interpolation was proposed for *abstraction refinement* in software model checking, first for propositional logic and propositional satisfiability [25], and then for quantifier-free fragments of first-order theories, their combinations, and satisfiability modulo theories [18,26,35,21,11,12,17,6,7,9]. Considered theories include equality [26,16], linear rational arithmetic [26,21], Presburger or linear integer arithmetic [21,6], or fragments thereof [12], and arrays without extensionality [21,7,8]. In these papers the theory reasoning is done either by specialized sequent-style inference systems [18,26,6,7] or by satisfiability procedures, such as congruence closure for equality [16], integrated in a DPLL($\mathcal{T}$) framework [35,11,12,17]. Subsequently, interpolation was suggested for *invariant generation*, and in the context of inference systems for first-order logic with equality, based on resolution and superposition [27,23,19]. An early lead towards this application can be traced back to [10]. More recently, interpolation was related to *abstract interpretation* [14] and applied to improve the quality of annotations [28].

The aim of this paper is to present the core of the state of the art in interpolation for the proofs generated by theorem provers for satisfiability modulo

[*] Research supported in part by MIUR grant no. 2007-9E5KM8 and EU COST Action IC0901.

K. Brünnler and G. Metcalfe (Eds.): TABLEAUX 2011, LNAI 6793, pp. 1–16, 2011.

theories, also known as SMT-solvers, based on the DPLL($\mathcal{T}$) paradigm (e.g., [31,33]), where DPLL refers to the Davis-Putnam-Logemann-Loveland procedure for propositional satisfiability, and $\mathcal{T} = \bigcup_{i=1}^{n} \mathcal{T}_i$ is a union of theories.

In this paper we present:

- A framework of definitions for interpolation, including that of *completeness* of an interpolation system;
- Two interpolation systems for propositional logic, HKPYM [20,24,32] and MM [25,26], already surveyed in [15], with a reconstruction of the proof of completeness of HKPYM in [35];
- An analysis of interpolation in the presence of equality, which explains the relation between the notion of *equality-interpolating theory* of [35] and that of *separating ordering* of [27,23,19] and provides a proof of the result that the quantifier-free fragment of the theory of equality is equality-interpolating, which was sketched in [35];
- The interpolation system of [35] for equality sharing [29], which we name EQSH, with a reconstruction of its proof of completeness in [35] and the observation that it applies also to *model-based theory combination* [13];
- The interpolation system for DPLL($\mathcal{T}$) obtained in [35] by uniting HKPYM and EQSH, which we name HKPYM–T.

We emphasize the contributions of [35], because it is where the crucial notion of equality-interpolating theory appeared, and because all subsequent papers that we are aware of refer to [35] for the proofs of completeness of HKPYM and EQSH, whence HKPYM–T. However, those proofs appeared only in the technical report companion to [35], and with discrepancies in definitions and notations between [35] and the technical report. Thus, we choose to present them here.

2 A Framework of Definitions for Interpolation

We assume the basic definitions commonly used in theorem proving. Let A and B be two formulæ with respective signatures Σ_A and Σ_B:

Definition 1. *A non-variable symbol is* A-colored*, if it is in $\Sigma_A \setminus \Sigma_B$;* B-colored*, if it is in $\Sigma_B \setminus \Sigma_A$; and* transparent*, if it is in $\Sigma_T = \Sigma_A \cap \Sigma_B$.*

Let $\mathcal{L}_X$ denote, ambiguously, the language of Σ_X-terms, Σ_X-literals or Σ_X-formulæ, where X stands for either A, B or T.

Definition 2. *A formula I is an* interpolant *of formulæ A and B such that $A \vdash B$, or an interpolant of (A, B), if (i) $A \vdash I$, (ii) $I \vdash B$ and (iii) $I \in \mathcal{L}_T$.*

The following classical result is known as Craig's Interpolation Lemma:

Lemma 1. *Let A and B be closed formulæ such that $A \vdash B$. If Σ_T contains at least one predicate symbol, an interpolant I of (A, B) exists and it is a closed formula; otherwise, either B is valid and I is $\top$, or A is unsatisfiable and $I$is $\bot$.*

From now on we consider only closed formulæ. Since most theorem provers work refutationally, it is useful to adopt the following:

Definition 3. *A formula I is a* reverse interpolant *of formulæ A and B such that $A, B \vdash \bot$, if (i) $A \vdash I$, (ii) $B, I \vdash \bot$ and (iii) $I \in \mathcal{L}_T$.*

A reverse interpolant of (A, B) is an interpolant of $(A, \neg B)$. A theory is presented by a set $\mathcal{T}$ of sentences, meaning that the theory is the set of all logical consequences of $\mathcal{T}$. If $\Sigma_{\mathcal{T}}$ is the signature of $\mathcal{T}$, $\mathcal{L}_T$ is redefined to be the language built from $\Sigma_{\mathcal{T}} \cup \Sigma_T$, so that theory symbols are transparent:

Definition 4. *A formula I is a* theory interpolant *of formulæ A and B such that $A \vdash_{\mathcal{T}} B$, if (i) $A \vdash_{\mathcal{T}} I$, (ii) $I \vdash_{\mathcal{T}} B$ and (iii) $I \in \mathcal{L}_T$. A formula I is a* reverse theory interpolant *of formulæ A and B such that $A, B \vdash_{\mathcal{T}} \bot$, if (i) $A \vdash_{\mathcal{T}} I$, (ii) $B, I \vdash_{\mathcal{T}} \bot$ and (iii) $I \in \mathcal{L}_T$.*

The distinction between interpolant and reverse interpolant appeared in [23]. Since we consider refutational systems, and in keeping with most of the literature, we write "interpolant" for "reverse interpolant" and omit "theory," unless relevant. Because most systems work with clauses, that are disjunctions of literals, from now on A and B are sets of clauses.

Definition 5. *A ground term, literal, or clause is* transparent*, if all its symbols are;* A-colored*, if it contains at least one A-colored symbol, and the others are transparent;* B-colored*, if it contains at least one B-colored symbol, and the others are transparent; and* AB-mixed*, otherwise. A clause is* colorable *if it contains no AB-mixed literals.*

Some authors use *A-local* in place of A-colored, *B-local* in place of B-colored, and *AB-common*, or *global*, in place of transparent. In the following "colored" may mean non-transparent.

Definition 6. *Let C be a disjunction (conjunction) of literals. The* projection *$C|_X$ of C on $\mathcal{L}_X$ is the disjunction (conjunction) obtained by removing from C any literal whose atom is not in $\mathcal{L}_X$. If C is a disjunction and $C|_X$ is empty, then $C|_X = \bot$; if C is a conjunction and $C|_X$ is empty, then $C|_X = \top$.*

Projection commutes with negation $(\neg(C|_X) = (\neg C)|_X)$ and distributes over conjunction and disjunction: $(C \vee D)|_X = C|_X \vee D|_X$ and $(C \wedge D)|_X = C|_X \wedge D|_X$. Since transparent literals of C belong to both $C|_A$ and $C|_B$, if C is a conjunction, $C|_A \Rightarrow C|_T$ and $C|_B \Rightarrow C|_T$; if C is a disjunction, $C|_T \Rightarrow C|_A$ and $C|_T \Rightarrow C|_B$. Alternatively, transparent literals may be put only in the projection on $\mathcal{L}_B$:

Definition 7. *Let C be a disjunction (conjunction) of literals. The* asymmetric projections *of C are $C \backslash_B = C|_A \setminus C|_T$ and $C \downarrow_B = C|_B$.*

Since [25], approaches to interpolation work by annotating each clause C in a refutation of A and B with auxiliary formulæ, unnamed in [25,26], named *C-interpolants* in [23], and *partial interpolants* in [35,15,6]:

Definition 8. *A* partial interpolant $PI(C)$ *of a clause C occurring in a refutation of $A \cup B$ is an interpolant of $g_A(C) = A \wedge \neg(C|_A)$ and $g_B(C) = B \wedge \neg(C|_B)$.*

If C is the empty clause, which is written $\Box$ and represents a contradiction, $PI(C)$ is an interpolant of (A, B). Since $A \wedge B \vdash C$, $g_A(C) \wedge g_B(C) = A \wedge B \wedge \neg C \vdash\perp$, and it makes sense to seek an interpolant of $g_A(C)$ and $g_B(C)$, which may be seen as an interpolant of (A, B) in a proof of C. We also write $c|_X$, $PI(c)$, $g_A(c)$ and $g_B(c)$ if c is the label of clause C, written $c\colon C$. By Definition 3 applied to Definition 8, a partial interpolant needs to satisfy the following requirements:

1. $g_A(C) \vdash PI(C)$ or $A \wedge \neg(C|_A) \vdash PI(C)$ or $A \vdash C|_A \vee PI(C)$
2. $g_B(C) \wedge PI(C) \vdash\perp$ or $B \wedge \neg(C|_B) \wedge PI(C) \vdash\perp$ or $B \wedge PI(C) \vdash C|_B$, and
3. $PI(C)$ is transparent.

Indeed, since the signatures of $g_A(C)$ and $g_B(C)$ are Σ_A and Σ_B, transparency is always with respect to A and B.

3 Transition Systems, Proofs and Interpolation Systems

DPLL and DPLL($\mathcal{T}$) are presented as *transition systems* (e.g., [31,4]) that operate in two modes, *search mode* and *conflict resolution mode.* In search mode, the state has the form $M \parallel F$, where F is a set of clauses and M is a sequence of *assigned literals*, that represents a partial assignment to ground literals, possibly with a justification, and therefore a partial model, or a set of candidate models. An assigned literal can be either a *decided literal* or an *implied literal.* A decided literal represents a guess, and has no justification (*decision* or *splitting*). An implied literal l_C is a literal l justified by a clause C: all other literals of C are false in M so that l needs to be true (*unit propagation*). If there is a clause C whose literals are all false in M, C is *in conflict*, it is called *conflict clause*, and the system switches to conflict resolution mode, where the state has the form $M \parallel F \parallel C$. In conflict resolution mode, a conflict clause $C \vee \neg l$ may be resolved with the justification $D \vee l$ of l in M to yield a new conflict clause $C \vee D$ (*explanation*). Any clause thus derived can be added to F (*learning*). Backjumping unassigns at least one decided literal and drives the system back from conflict resolution mode to search mode. In state $M \parallel F \parallel \Box$, unsatisfiability is detected.

Definition 9. *Let $\mathcal{U}_1$ stand for DPLL and $\mathcal{U}_2$ for DPLL($\mathcal{T}$), and S be the input set of clauses. A* transition system derivation, *or $\mathcal{U}_j$-*derivation, *where $j \in \{1, 2\}$, is a sequence of state transitions $\Delta_0 \Longrightarrow_{\mathcal{U}_j} \Delta_1 \Longrightarrow_{\mathcal{U}_j} \ldots \Delta_i \Longrightarrow_{\mathcal{U}_j} \Delta_{i+1} \Longrightarrow_{\mathcal{U}_j} \ldots$, where $\forall i \geq 0$, Δ_i is of the form $M_i \parallel F_i$ or $M_i \parallel F_i \parallel C_i$, each transition is determined by a transition rule in $\mathcal{U}_j$, $\Delta_0 = \parallel F_0$ and $F_0 = S$.*

As noticed first in [36], according to [34], a proof produced by DPLL is made of propositional resolution steps between conflict clauses (*explanations*). Let $C^* = \{C_i | i > 0\}$ be the set of all conflict clauses in a derivation:

Definition 10. *For DPLL-derivation $\Delta_0 \Longrightarrow_{\mathcal{U}_1} \ldots \Delta_i \Longrightarrow_{\mathcal{U}_1} \Delta_{i+1} \Longrightarrow_{\mathcal{U}_1} \ldots$, for all $C \in C^*$ the* DPLL-proof tree $\Pi_{\mathcal{U}_1}(C)$ *of C is defined as follows:*

- *If $C \in F_0$, $\Pi_{\mathcal{U}_1}(C)$ consists of a node labelled by C;*
- *If C is generated by resolving conflict clause C_1 with justification C_2, $\Pi_{\mathcal{U}_1}(C)$ consists of a node labelled by C with subtrees $\Pi_{\mathcal{U}_1}(C_1)$ and $\Pi_{\mathcal{U}_1}(C_2)$.*

If the derivation halts reporting unsatisfiable, $\Pi_{\mathcal{U}_1}(\Box)$ is a DPLL-refutation.

Since a justification C_2 is either an input clause or a learnt clause, which was once a conflict clause, $\Pi_{\mathcal{U}_1}(C_2)$ is defined.

DPLL($\mathcal{T}$) builds into DPLL a *$\mathcal{T}$-satisfiability procedure*, that decides whether a set of ground $\mathcal{T}$-literals has a $\mathcal{T}$-model. In most cases, $\mathcal{T}$ is a union of theories $\bigcup_{i=1}^{n} \mathcal{T}_i$, and a $\mathcal{T}$-satisfiability procedure is obtained by combining n $\mathcal{T}_i$-satisfiability procedures, that we name $\mathcal{Q}_i$, for $1 \leq i \leq n$, according to *equality sharing* [29] (see Chapter 10 of [5] for a modern presentation). The $\mathcal{T}_i$'s are quantifier-free fragments of first-order theories such as the theory of equality, linear arithmetic or theories of common data structures. For the theory of equality, also known as equality with uninterpreted, or free, symbols (EUF), the satisfiability procedure is based on *congruence closure* (e.g., [30], and Chapter 9 of [5] for a modern presentation).

Equality sharing requires that the $\mathcal{T}_i$'s are pairwise *disjoint*, which means the only shared symbol, beside constants, is equality, and *stably infinite*, which means that any quantifier-free $\mathcal{T}_i$-formula has a $\mathcal{T}_i$-model if and only if it has an infinite one. Equality sharing separates occurrences of function symbols from different signatures to ensure that each $\mathcal{Q}_i$'s deals with $\mathcal{T}_i$-literals: for example, $f(g(a)) \simeq b$, where f and g belong to different signatures, becomes $f(c) \simeq b \wedge g(a) \simeq c$, where c is a new constant. Each $\mathcal{Q}_i$ propagates all disjunctions of equalities between shared constants that are $\mathcal{T}_i$-entailed by the problem. If a theory is *convex*, whenever a disjunction is entailed, a disjunct is also entailed, and therefore it is sufficient to exchange equalities. *Model-based theory combination* [13] is a version of equality sharing that assumes that each $\mathcal{Q}_i$ maintains a candidate $\mathcal{T}_i$-model, and replaces propagation of entailments by addition to M of equalities that are true in the current candidate $\mathcal{T}_i$-model. These equalities are *guesses*, because it is not known whether they are true in all $\mathcal{T}_i$-models consistent with M. If one of them turns out to be inconsistent, backjumping will withdraw it and update the $\mathcal{T}_i$-model.

In DPLL($\mathcal{T}$), the $\mathcal{T}$-satisfiability procedure propagates $\mathcal{T}$-consequences of M to the DPLL engine: if whenever literals $l_1, \ldots, l_n$ are true, some literal l must also be true in $\mathcal{T}$, l is added to M with the $\mathcal{T}$-lemma $\neg l_1 \vee \ldots \vee \neg l_n \vee l$ as justification (*theory propagation*); if a subset $l_1, \ldots, l_n$ of M is $\mathcal{T}$-inconsistent, the system switches to conflict resolution mode with *$\mathcal{T}$-conflict clause* $\neg l_1 \vee \ldots \vee \neg l_n$. Proofs produced by DPLL($\mathcal{T}$) are ground, but not propositional, and include also $\mathcal{T}$-conflict clauses. Thus, we need to assume that the $\mathcal{Q}_i$'s and their combination produce proofs, that we denote by $\Pi_{\mathcal{T}}(C)$:

Definition 11. *For DPLL($\mathcal{T}$)-derivation $\Delta_0 \Longrightarrow_{\mathcal{U}_2} \ldots \Delta_i \Longrightarrow_{\mathcal{U}_2} \Delta_{i+1} \Longrightarrow_{\mathcal{U}_2} \ldots$, for all $C \in C^*$ the* DPLL($\mathcal{T}$)-proof tree *$\Pi_{\mathcal{U}_2}(C)$ of C is defined as follows:*

- *If $C \in F_0$, $\Pi_{\mathcal{U}_2}(C)$ consists of a node labelled by C;*

- *If C is generated by resolving conflict clause C_1 with justification C_2, $\Pi_{\mathcal{U}_2}(C)$ consists of a node labelled by C with subtrees $\Pi_{\mathcal{U}_2}(C_1)$ and $\Pi_{\mathcal{U}_2}(C_2)$;*
- *If C is a $\mathcal{T}$-conflict clause, $\Pi_{\mathcal{U}_2}(C) = \Pi_{\mathcal{T}}(C)$.*

If the derivation halts reporting unsatisfiable, $\Pi_{\mathcal{U}_2}(\Box)$ is a DPLL($\mathcal{T}$)-refutation.

An *interpolation system* is a mechanism to annotate clauses in a proof with partial interpolants; its most important property is completeness:

Definition 12. *An interpolation system is* complete *for transition system $\mathcal{U}$, if for all sets of clauses A and B, such that $A \cup B$ is unsatisfiable, and for all $\mathcal{U}$-refutations of $A \cup B$, it generates an interpolant of (A, B).*

Since $PI(\Box)$ is an interpolant of (A, B), in order to prove that an interpolation system is complete, it is sufficient to show that it annotates the clauses in any refutation with clauses that are partial interpolants.

We conclude recalling that stable infiniteness is connected to interpolation: the set of disjunctions of equalities exchanged by $\mathcal{T}_1$ and $\mathcal{T}_2$ in equality sharing is a reverse interpolant of (F_1, F_2), where F_i, for $i \in \{1, 2\}$, is the conjunction of input $\mathcal{T}_i$-literals after separation. Stable-infiniteness ensures that quantifier-free interpolants suffice (cf. Chapter 10 of [5] for details). This is obviously not sufficient to make a combination of interpolating satisfiability procedures interpolating, because (F_1, F_2) is a partition of the input based on the signatures of $\mathcal{T}_1$ and $\mathcal{T}_2$, whereas we need to generate interpolants of an arbitrary partition (A, B) of the input where both A and B may mix $\mathcal{T}_1$-symbols and $\mathcal{T}_2$-symbols.

4 Propositional Interpolation Systems

If the input $S = A \cup B$ is a set of propositional clauses, the set of literals that may appear in a refutation is determined once and for all by the set of literals that occur in S. Since input literals are either A-colored or B-colored or transparent, *there are no AB-mixed literals* in proofs. Interpolation systems attach a partial interpolant to every resolution step, distinguishing whether the literal resolved upon is A-colored or B-colored or transparent. The first interpolation system for propositional resolution, called HKP in [15] from the initials of three independent authors, appeared in [20,24,32]. We call it HKPYM, because Yorsh and Musuvathi reformulated it and reproved it complete in the context of DPLL($\mathcal{T}$):

Definition 13 (HKPYM interpolation system). *Let $c\colon C$ be a clause that appears in a refutation of $A \cup B$ by propositional resolution:*

- *If $c\colon C \in A$, then $PI(c) = \bot$,*
- *If $c\colon C \in B$, then $PI(c) = \top$,*
- *If $c\colon C \vee D$ is a propositional resolvent of $p_1\colon C \vee l$ and $p_2\colon D \vee \neg l$ then:*
  - *If l is A-colored, then $PI(c) = PI(p_1) \vee PI(p_2)$,*
  - *If l is B-colored, then $PI(c) = PI(p_1) \wedge PI(p_2)$ and*
  - *If l is transparent, then $PI(c) = (l \vee PI(p_1)) \wedge (\neg l \vee PI(p_2))$.*

The system of [25,26], that we call MM, allows a more informative interpolant for clauses in A and treats transparent literals resolved upon like B-colored ones:

Definition 14 (MM interpolation system). *Let $c\colon C$ be a clause that appears in a refutation of $A \cup B$ by propositional resolution:*

- *If $c\colon C \in A$, then $PI(c) = C|_T$,*
- *If $c\colon C \in B$, then $PI(c) = \top$,*
- *If $c\colon C \vee D$ is a propositional resolvent of $p_1\colon C \vee l$ and $p_2\colon D \vee \neg l$ then:*
  - *If l is A-colored, then $PI(c) = PI(p_1) \vee PI(p_2)$,*
  - *If l is B-colored or transparent, then $PI(c) = PI(p_1) \wedge PI(p_2)$.*

Intuitively, HKPYM is symmetric with respect to A and B, and assumes the symmetric notion of projection (cf. Definition 6), while MM is slanted towards B, and assumes the asymmetric notion of projection (cf. Definition 7). In [35], projection is defined to be asymmetric; however, the proof of completeness of HKPYM in the companion technical report, requires projection to be symmetric. The following proof fixes this discrepancy:

Theorem 1 (Yorsh and Musuvathi, 2005). *HKPYM is a complete interpolation system for propositional resolution.*

Proof: We need to prove that for all clauses $c\colon C$ in the refutation, $PI(c)$ satifies Requirements (1), (2) and (3) listed at the end of Section 2. The proof is by induction on the structure of the refutation.
Base case:

- If $c\colon C \in A$, then $\neg(C|_A) = \neg C$; and $g_A(c) = A \wedge \neg C =\perp$, since $C \in A$. Since $PI(c) =\perp$, both (1) and (2) reduce to $\perp \vdash \perp$, which is trivially true, and $PI(c)$ is trivially transparent.
- If $c\colon C \in B$, then $\neg(C|_B) = \neg C$; and $g_B(c) = B \wedge \neg C = \perp$, since $C \in B$. Since $PI(c) = \top$, (1) is trivial, (2) reduces to $\perp \vdash \perp$, which is trivially true, and $PI(c)$ is trivially transparent.

Induction hypothesis: for $k \in \{1,2\}$ $PI(p_k)$ satifies Requirements (1), (2) and (3).
Induction case:

(a) l is A-colored: $PI(c) = PI(p_1) \vee PI(p_2)$.
First we observe that $p_1|_A \wedge p_2|_A \Rightarrow C|_A \vee D|_A$ (*). Indeed, since l is A-colored, $p_1|_A = (l \vee C)|_A = l \vee C|_A$ and $p_2|_A = (\neg l \vee D)|_A = \neg l \vee D|_A$. Then, $p_1|_A \wedge p_2|_A = (l \vee C|_A) \wedge (\neg l \vee D|_A) \Rightarrow C|_A \vee D|_A$ by resolution.
We show (1) $g_A(c) \Rightarrow PI(c)$:
$g_A(c) = A \wedge \neg((C \vee D)|_A) = A \wedge \neg(C|_A \vee D|_A)$
$A \wedge \neg(C|_A \vee D|_A) \Rightarrow A \wedge \neg(p_1|_A \wedge p_2|_A)$ by (*)
$A \wedge \neg(p_1|_A \wedge p_2|_A) = A \wedge(\neg p_1|_A \vee \neg p_2|_A) = (A \wedge \neg p_1|_A) \vee (A \wedge \neg p_2|_A) = g_A(p_1) \vee g_A(p_2)$ and $g_A(p_1) \vee g_A(p_2) \Rightarrow PI(p_1) \vee PI(p_2)$ by induction hypothesis.
We show (2) $g_B(c) \wedge PI(c) \Rightarrow\perp$:
$g_B(c) \wedge PI(c) = B \wedge \neg((C \vee D)|_B) \wedge PI(c) = B \wedge \neg(C|_B \vee D|_B) \wedge PI(c) =$

$B \wedge \neg(C|_B) \wedge \neg(D|_B) \wedge (PI(p_1) \vee PI(p_2)) \Rightarrow$
$(B \wedge \neg(C|_B) \wedge PI(p_1)) \vee (B \wedge \neg(D|_B) \wedge PI(p_2)) =$
$(B \wedge \neg((l \vee C)|_B) \wedge PI(p_1)) \vee (B \wedge \neg((\neg l \vee D)|_B) \wedge PI(p_2)) =$
(because $l \notin \mathcal{L}_B$ and, therefore, $C|_B = (l \vee C)|_B$ and $D|_B = (\neg l \vee D)|_B$)
$= (B \wedge \neg(p_1|_B) \wedge PI(p_1)) \vee (B \wedge \neg(p_2|_B) \wedge PI(p_2)) =$
$(g_B(p_1) \wedge PI(p_1)) \vee (g_B(p_2) \wedge PI(p_2)) \Rightarrow \bot \vee \bot = \bot$ by induction hypothesis.
Requirement (3) follows by induction hypothesis.

(b) l is B-colored: $PI(c) = PI(p_1) \wedge PI(p_2)$.
Similar to Case (a), we have that $p_1|_B \wedge p_2|_B \Rightarrow C|_B \vee D|_B$ (**).
We show (1) $g_A(c) \Rightarrow PI(c)$:
$g_A(c) = A \wedge \neg((C \vee D)|_A) = A \wedge \neg(C|_A \vee D|_A) = A \wedge \neg(C|_A) \wedge \neg(D|_A) =$
$A \wedge \neg((l \vee C)|_A) \wedge \neg((\neg l \vee D)|_A) =$
(because $l \notin \mathcal{L}_A$ and, therefore, $C|_A = (l \vee C)|_A$ and $D|_A = (\neg l \vee D)|_A$)
$= A \wedge \neg(p_1|_A) \wedge \neg(p_2|_A) = (A \wedge \neg(p_1|_A)) \wedge (A \wedge \neg(p_2|_A)) = g_A(p_1) \wedge g_A(p_2)$
and $g_A(p_1) \wedge g_A(p_2) \Rightarrow PI(p_1) \wedge PI(p_2)$ by induction hypothesis.
We show (2) $g_B(c) \wedge PI(c) \Rightarrow \bot$:
$g_B(c) \wedge PI(c) = B \wedge (\neg((C \vee D)|_B)) \wedge PI(p_1) \wedge PI(p_2) =$
$B \wedge (\neg(C|_B \vee D|_B)) \wedge PI(p_1) \wedge PI(p_2) \Rightarrow$ by (**)
$B \wedge (\neg(p_1|_B \wedge p_2|_B)) \wedge PI(p_1) \wedge PI(p_2) = B \wedge (\neg(p_1|_B) \vee \neg(p_2|_B)) \wedge PI(p_1) \wedge$
$PI(p_2) = [(B \wedge \neg(p_1|_B)) \vee (B \wedge \neg(p_2|_B))] \wedge PI(p_1) \wedge PI(p_2) =$
$(g_B(p_1) \wedge PI(p_1) \wedge PI(p_2)) \vee (g_B(p_2) \wedge PI(p_1) \wedge PI(p_2)) \Rightarrow$
$(g_B(p_1) \wedge PI(p_1)) \vee (g_B(p_2) \wedge PI(p_2)) \Rightarrow \bot \vee \bot = \bot$ by induction hypothesis.
Requirement (3) follows by induction hypothesis.

(c) l is transparent: $PI(c) = (l \vee PI(p_1)) \wedge (\neg l \vee PI(p_2))$.
We show (1) $g_A(c) \Rightarrow PI(c)$, or, equivalently, $g_A(c) \wedge \neg PI(c) \Rightarrow \bot$:
$g_A(c) \wedge \neg PI(c) = A \wedge (\neg(C|_A \vee D|_A)) \wedge \neg[(l \vee PI(p_1)) \wedge (\neg l \vee PI(p_2))] =$
$A \wedge \neg(C|_A) \wedge \neg(D|_A) \wedge [\neg(l \vee PI(p_1)) \vee \neg(\neg l \vee PI(p_2))] =$
$[A \wedge \neg(C|_A) \wedge \neg(D|_A) \wedge \neg(l \vee PI(p_1))] \vee [A \wedge \neg(C|_A) \wedge \neg(D|_A) \wedge \neg(\neg l \vee PI(p_2))] \Rightarrow$
$[A \wedge \neg(C|_A) \wedge \neg(l \vee PI(p_1))] \vee [A \wedge \neg(D|_A) \wedge \neg(\neg l \vee PI(p_2))] =$
$[A \wedge \neg(C|_A) \wedge \neg l \wedge \neg PI(p_1)] \vee [A \wedge \neg(D|_A) \wedge l \wedge \neg PI(p_2)] =$ (l is transparent)
$= [A \wedge \neg((l \vee C)|_A) \wedge \neg PI(p_1)] \vee [A \wedge \neg((\neg l \vee D)|_A) \wedge \neg PI(p_2)] =$
$(g_A(p_1) \wedge \neg PI(p_1)) \vee (g_A(p_2) \wedge \neg PI(p_2)) \Rightarrow \bot \vee \bot = \bot$ by induction hypothesis.
We show (2) $g_B(c) \wedge PI(c) \Rightarrow \bot$:
$g_B(c) \wedge PI(c) = B \wedge \neg(C|_B \vee D|_B) \wedge [(l \vee PI(p_1)) \wedge (\neg l \vee PI(p_2))] =$
$B \wedge \neg(C|_B) \wedge \neg(D|_B) \wedge [(l \vee PI(p_1)) \wedge (\neg l \vee PI(p_2))]$ (***)
at this point we reason that l is either true or false; if l is true, l holds, l subsumes $l \vee PI(p_1)$ and simplifies $\neg l \vee PI(p_2)$ to $PI(p_2)$; if l is false, $\neg l$ holds, $\neg l$ subsumes $\neg l \vee PI(p_2)$ and simplifies $l \vee PI(p_1)$ to $PI(p_1)$; thus, (***) implies $[B \wedge \neg(C|_B) \wedge \neg(D|_B) \wedge l \wedge PI(p_2)] \vee [B \wedge \neg(C|_B) \wedge \neg(D|_B) \wedge \neg l \wedge PI(p_1)]$
which implies $[B \wedge \neg(D|_B) \wedge l \wedge PI(p_2)] \vee [B \wedge \neg(C|_B) \wedge \neg l \wedge PI(p_1)] =$
$[B \wedge \neg(D|_B \vee \neg l) \wedge PI(p_2)] \vee [B \wedge \neg(C|_B \vee l) \wedge PI(p_1)] =$ (l is transparent)
$= (B \wedge \neg(p_2|_B) \wedge PI(p_2)) \vee (B \wedge \neg(p_1|_B) \wedge PI(p_1)) =$
$(g_B(p_2) \wedge PI(p_2)) \vee (g_B(p_1) \wedge PI(p_1)) \Rightarrow \bot \vee \bot = \bot$ by induction hypothesis.
Requirement (3) holds by induction hypothesis and l being transparent. □

5 Interpolation and Equality

In propositional logic the notion of being A-colored, B-colored and transparent is stable: if a literal is transparent in the initial state of a derivation, it will be transparent in all. Once equality is added, even in the ground case, this is no longer obvious. Assume that t_a is an A-colored ground term, t_b is a B-colored ground term, and the AB-mixed equation $t_a \simeq t_b$ is derived. The congruence classes of t_a and t_b have to be merged. Assume that the congruence class of t_a only contains A-colored terms, that of t_b only contains B-colored terms, and t_a and t_b are the representatives of their congruence classes. If t_b were chosen as the representative of the new class, it should become transparent. If either one of the two classes already contains a transparent term t, the AB-mixed equation $t_a \simeq t_b$ is not problematic, because t can be the representative of the new class. The issue is the same if we reason about equality by rewriting: if an AB-mixed equation $t_a \simeq t_b$ is generated, where both t_a and t_b are in normal form with respect to the current set of equations, and $t_a \succ t_b$ in the ordering, all occurrences of t_a, including those in A, should be replaced by t_b, which should become transparent. In order to prevent such instability of transparency, one needs to require that all theories are *equality-interpolating*, a property that appeared first in [35]:

Definition 15. *A theory $\mathcal{T}$ is* equality-interpolating *if for all $\mathcal{T}$-formulæ A and B, whenever $A \wedge B \models_{\mathcal{T}} t_a \simeq t_b$, where t_a is an A-colored ground term and t_b is a B-colored ground term, then $A \wedge B \models_{\mathcal{T}} t_a \simeq t \wedge t_b \simeq t$ for some transparent ground term t.*

Then, in congruence closure, it is sufficient to adopt t as the representative of the new congruence class. Operationally, as suggested in [35], if $t_a \simeq t_b$ is generated before $t_a \simeq t$ and $t_b \simeq t$, one may add a new transparent constant c and the equations $t_a \simeq c$ and $t_b \simeq c$: if the theory is equality-interpolating, $c \simeq t$ will be generated eventually, so that c is only a name for t. In a rewrite-based setting, one needs to assume the following:

Definition 16. *An ordering $\succ$ is* separating *if $t \succ s$ whenever s is transparent and t is not, for all terms, or literals, s and t.*

Thus, both t_a and t_b will be rewritten to t. This requirement on the ordering appeared in [27], under the name *AB-oriented ordering*, and then in [22], where it was justified intuitively in terms of symbol elimination: since interpolants have to be transparent, the ordering should orient equations in such a way that rewriting eliminates colored symbols. However, it was not related to the notion of equality-interpolating theory.

Ground proofs made only of equalities and containing no AB-mixed equality were termed *colorable* in [17]. We adopt the name and apply it to any proof made of clauses:

Definition 17. *A proof is* colorable *if all its clauses are.*

If A-local, B-local and AB-common, or global, are used in place of A-colored, B-colored and transparent, respectively, acceptable proofs are called *local.*

It was proved in [35] that the quantifier-free fragments of the theories of *equality*, *linear arithmetic* and *non-empty, possibly cyclic lists* are equality-interpolating. The proof that the theory of lists is equality-interpolating relies on that for the theory of equality. However, the proof for the latter was only sketched in [35] (cf. Lemma 2 in [35]), and therefore we reconstruct it here using the notion of separating ordering:

Lemma 2. *If the ordering is separating, all ground proofs by resolution and rewriting are colorable.*

Proof: By induction on the structure of the proof:
Base Case: By definition, there are no AB-mixed literals in the input.
Induction Hypothesis: The premises do not contain AB-mixed literals.
Induction Case:

- *Resolution:* Since a ground resolvent is made of literals inherited from its parents, it does not contain AB-mixed literals by induction hypothesis.
- *Rewriting:* let $s \simeq r$ be a ground equation such that $s \succ r$, and $l[s]$ be the literal it applies to. By induction hypothesis, neither $s \simeq r$ nor $l[s]$ are AB-mixed. Thus, since the ordering is separating and $s \succ r$, either r has the same color as s or it is transparent. If s and r have the same color, also $l[s]$ and $l[r]$ have the same color. If s is colored and r is transparent, then $l[r]$ either has the same color as $l[s]$ or it is transparent, the latter if s was its only colored term. In either case, $l[r]$ is not AB-mixed. □

To show the following we only need to consider purely equational proofs, that are represented as equational *chains. Unfailing*, or *ordered*, *completion* reduces any ground equational proof $s \overset{*}{\leftrightarrow} t$ to a *rewrite proof*, or *valley proof* in the form $s \overset{*}{\rightarrow} \circ \overset{*}{\leftarrow} t$ (see [2] for a recent treatment with ample references):

Theorem 2. *The quantifier-free fragment of the theory of equality is equality-interpolating.*

Proof: Assume ground completion employs a separating ordering. If $A \wedge B \models t_a \simeq t_b$, where t_a is an A-colored ground term and t_b is a B-colored ground term, then, since unfailing completion is refutationally complete, it generates a contradiction from $A \cup B \cup \{t_a \not\simeq t_b\}$. The only rôle of $t_a \not\simeq t_b$ in the derivation is to be rewritten, until a contradiction is generated in the form $t \not\simeq t$. The resulting proof is made only of rewriting steps and therefore it is a rewrite proof $t_a \overset{*}{\rightarrow} t \overset{*}{\leftarrow} t_b$. Since the ordering is separating, this proof contains no AB-mixed equations by Lemma 2, which means that it must contain at least a transparent term. Since the ordering is separating, the smallest term t is transparent. It follows that $A \wedge B \models t_a \simeq t \wedge t_b \simeq t$, because the inferences are sound. □

From now on, we assume that *the built-in theories $\mathcal{T}_i$, $1 \leq i \leq n$, are equality-interpolating*, so that there are no AB-mixed literals, and all proofs are colorable.

6 Interpolation for Equality Sharing and DPLL($\mathcal{T}$)

The treatment of interpolation for equality sharing in [35] temporarily assumes that the theories in $\mathcal{T} = \bigcup_{i=1}^{n} \mathcal{T}_i$ are convex, so that it is sufficient to propagate equalities. We do not need to assume convexity, not even as a temporary assumption, neither we need to deal with propagation of disjunctions, because we adopt model-based theory combination, where only equalities are propagated. For equality sharing, the input $A \cup B$ is a set of ground $\mathcal{T}$-literals, or unit $\mathcal{T}$-clauses. Separation applies in such a way to respect colors: for example, if $f(g(a)) \simeq b$, where f and g belong to the signatures of different theories, becomes $f(c) \simeq b \wedge g(a) \simeq c$, the new constant c is stipulated to be A-colored, B-colored or transparent, depending on whether $g(a)$ is A-colored, B-colored or transparent, respectively.

Every $\mathcal{Q}_i$ deals with a set $A_i \cup B_i \cup K$, where A_i contains the $\mathcal{T}_i$-literals in A, B_i contains the $\mathcal{T}_i$-literals in B, and K is the set of propagated equalities. Although for equality sharing it suffices to propagate equalities between shared constants, in DPLL($\mathcal{T}$) the propagation is done by adding the literal to M, and model-based theory combination may propagate equalities between ground terms. Thus, we assume that K is a set of equalities between ground terms.

We assume that each $\mathcal{Q}_i$ is capable of generating $\mathcal{T}_i$-interpolants for its proofs. A crucial observation in [35] is that these $\mathcal{T}_i$-interpolants cannot be $\mathcal{T}_i$-interpolants of (A, B), since the input to $\mathcal{Q}_i$ is $A_i \cup B_i \cup K$. They are $\mathcal{T}_i$-interpolants of some partition (A', B') of $A_i \cup B_i \cup K$. This is where the assumption that theories are equality-interpolating plays a rôle: K contains no AB-mixed literals. Therefore, it is possible to define A' and B' based on colors as defined by the original (A, B) partition, using projections with respect to A and B: let A' be $(A_i \cup B_i \cup K)|_A = A_i \cup K|_A$ and B' be $(A_i \cup B_i \cup K)|_B = B_i \cup K|_B$. It follows that $\mathcal{L}_{A'} = \mathcal{L}_A$, $\mathcal{L}_{B'} = \mathcal{L}_B$, and what is transparent with respect to (A', B') is transparent with respect to (A, B), so that $\mathcal{T}_i$-interpolants of (A', B') can be used to build the $\mathcal{T}$-interpolant of (A, B):

Definition 18. *For all ground literals l, such that $A_i \cup B_i \cup K \vdash_{\mathcal{T}_i} l$, where A_i is the set of $\mathcal{T}_i$-literals in A, B_i is the set of $\mathcal{T}_i$-literals in B, and K is a set of propagated equalities between ground terms, the* theory-specific partial interpolant *of l, denoted by $PI^i_{(A',B')}(l)$, is the $\mathcal{T}_i$-interpolant of $(A' \wedge \neg(l|_A), B' \wedge \neg(l|_B))$ generated by $\mathcal{Q}_i$, where $A' = A_i \cup K|_A$ and $B' = B_i \cup K|_B$.*

Note how there is no need to require that only equalities between shared constants are propagated.

In equality sharing the refutation is found by one of the theories, and therefore has the form $A_i \cup B_i \cup K \vdash_{\mathcal{T}_i} \bot$ for some i. The method of [35] shows how to extract a $\mathcal{T}$-interpolant of (A, B) from such a refutation, by combining the theory-specific partial interpolants computed by the $\mathcal{Q}_i$'s for the propagated equalities in K. We state it as an *interpolation system for equality sharing*:

Definition 19 (EQSH interpolation system). *Let C be a literal, or unit clause, that appears in a refutation of $A \cup B$ by equality sharing:*

- *If $C \in A$, then $PI(C) = \bot$,*
- *If $C \in B$, then $PI(C) = \top$,*
- *If $A_i \cup B_i \cup K \vdash_{\mathcal{T}_i} C$ for some i, $1 \leq i \leq n$, then*

$$PI(C) = (PI^i_{(A',B')}(C) \vee \bigvee_{l \in A'} PI(l)) \wedge \bigwedge_{l \in B'} PI(l),$$

where $A' = A_i \cup K|_A$ and $B' = B_i \cup K|_B$.

If there were only one theory, K would be empty, and the partial interpolant in the inductive case of Definition 19 would be equal to the theory-specific partial interpolant in that theory.

Theorem 3 (Yorsh and Musuvathi, 2005). *EQSH is a complete interpolation system for equality sharing.*

Proof: We need to prove that for all unit clauses C in the refutation, $PI(C)$ satifies Requirements (1), (2) and (3), listed at the end of Section 2, in theory $\mathcal{T}$. The base case is the same as for Theorem 1. The inductive case, for a C such that $A_i \cup B_i \cup K \vdash_{\mathcal{T}_i} C$ for some i, $1 \leq i \leq n$, requires another induction on K:
Base case: if $K = \emptyset$, then $A' = A_i$, $B' = B_i$, and $PI(C) = PI^i_{(A',B')}(C)$. By Definition 18, $PI(C)$ is a $\mathcal{T}_i$-interpolant of $(A_i \wedge \neg(C|_A), B_i \wedge \neg(C|_B))$; since $A_i \subseteq A$ and $B_i \subseteq B$, $PI(C)$ is also a $\mathcal{T}$-interpolant of $A \wedge \neg(C|_A)$ and $B \wedge \neg(C|_B)$.
Induction case: if $K \neq \emptyset$, then $A' = A_i \cup K|_A$, $B' = B_i \cup K|_B$ and $PI(C) = (PI^i_{(A',B')}(C) \vee \bigvee_{l \in K|_A} PI(l)) \wedge \bigwedge_{l \in K|_B} PI(l)$, because $PI(l) = \bot$ for all $l \in A_i$ and $PI(l) = \top$ for all $l \in B_i$.
We continue with the main claim:
Induction hypothesis: for all $l \in K|_A$, $PI(l)$ is a $\mathcal{T}$-interpolant of $(A \wedge \neg(l|_A), B \wedge \neg(l|_B))$, that is, a $\mathcal{T}$-interpolant of $(A \wedge \neg l, B)$, because $l|_A = l$ and $l|_B = \bot$, since $l \in K|_A$; for all $l \in K|_B$, $PI(l)$ is a $\mathcal{T}$-interpolant of $(A \wedge \neg(l|_A), B \wedge \neg(l|_B))$, that is, a $\mathcal{T}$-interpolant of $(A, B \wedge \neg l)$, because $l|_A = \bot$ and $l|_B = l$, since $l \in K|_B$; so that the inductive hypothesis is:

- For all $l \in K|_A$,
  (1A) $A \wedge \neg l \vdash_{\mathcal{T}} PI(l)$ or, equivalently, $A \vdash_{\mathcal{T}} l \vee PI(l)$,
  (2A) $B \wedge PI(l) \vdash_{\mathcal{T}} \bot$,
  (3A) $PI(l)$ is transparent; and
- For all $l \in K|_B$,
  (1B) $A \vdash_{\mathcal{T}} PI(l)$,
  (2B) $B \wedge \neg l \wedge PI(l) \vdash_{\mathcal{T}} \bot$, or, equivalently, $B \wedge PI(l) \vdash_{\mathcal{T}} l$,
  (3B) $PI(l)$ is transparent.

Induction case:

1. $A \wedge \neg(C|_A) \vdash_{\mathcal{T}} PI(C)$:
   By Definition 18, $A_i \wedge K|_A \wedge \neg(C|_A) \vdash_{\mathcal{T}_i} PI^i_{(A',B')}(C)$, or, equivalently, $A_i \wedge \neg(C|_A) \vdash_{\mathcal{T}_i} \neg K|_A \vee PI^i_{(A',B')}(C)$ (*), where $\neg K|_A$ is the disjunction $\neg l_1 \vee \ldots \vee \neg l_q$, if $K|_A$ is the conjunction $l_1 \wedge \ldots \wedge l_q$. By induction hypothesis

(1A), we have $A \vdash_{\mathcal{T}} l_j \vee PI(l_j)$ for $1 \leq j \leq q$ (**). By q resolution steps between (*) and (**), and since $A \Rightarrow A_i$, it follows that $A \wedge \neg(C|_A) \vdash_{\mathcal{T}} PI^i_{(A',B')}(C) \vee \bigvee_{l \in K|_A} PI(l)$. By induction hypothesis (1B), $A \vdash_{\mathcal{T}} PI(l)$ for all $l \in K|_B$. Therefore, we conclude that $A \wedge \neg(C|_A) \vdash_{\mathcal{T}} (PI^i_{(A',B')}(C) \vee \bigvee_{l \in K|_A} PI(l)) \wedge \bigwedge_{l \in K|_B} PI(l)$.

2. $B \wedge \neg(C|_B) \wedge PI(C) \vdash_{\mathcal{T}} \bot$:
By Definition 18, $B_i \wedge K|_B \wedge \neg(C|_B) \wedge PI^i_{(A',B')}(C) \vdash_{\mathcal{T}_i} \bot$. Since $B \Rightarrow B_i$, we have $B \wedge K|_B \wedge \neg(C|_B) \wedge PI^i_{(A',B')}(C) \vdash_{\mathcal{T}_i} \bot$ (*). By induction hypothesis (2A), $B \wedge PI(l) \vdash_{\mathcal{T}} \bot$ for all $l \in K|_A$, and thus $B \wedge \bigvee_{l \in K|_A} PI(l) \vdash_{\mathcal{T}} \bot$, and $B \wedge K|_B \wedge \neg(C|_B) \wedge \bigvee_{l \in K|_A} PI(l) \vdash_{\mathcal{T}} \bot$ (**). Combining (*) and (**) gives $B \wedge K|_B \wedge \neg(C|_B) \wedge (PI^i_{(A',B')}(C) \vee \bigvee_{l \in K|_A} PI(l)) \vdash_{\mathcal{T}} \bot$, or, equivalently, $B \wedge \neg(C|_B) \wedge (PI^i_{(A',B')}(C) \vee \bigvee_{l \in K|_A} PI(l)) \vdash_{\mathcal{T}} \neg K|_B$ (†), where $\neg K|_B$ is the disjunction $\neg l_1 \vee \ldots \vee \neg l_q$, if $K|_B$ is the conjunction $l_1 \wedge \ldots \wedge l_q$. By induction hypothesis (2B), $B \wedge PI(l_j) \vdash_{\mathcal{T}} l_j$ for $1 \leq j \leq q$ (‡). By q resolution steps between (†) and (‡), we get $B \wedge \neg(C|_B) \wedge (PI^i_{(A',B')}(C) \vee \bigvee_{l \in K|_A} PI(l)) \wedge \bigwedge_{l \in K|_B} PI(l) \vdash_{\mathcal{T}} \bot$, that is, $B \wedge \neg(C|_B) \wedge PI(C) \vdash_{\mathcal{T}} \bot$.
3. $PI(C)$ is transparent, because $PI^i_{(A',B')}(C)$ is transparent by Definition 18, and the $PI(l)$'s are transparent by induction hypotheses (3A) and (3B). □

This proof does not depend on assuming that K contains only equalities between shared constants, neither does it depend on assuming that K contains all such equalities entailed by the theories. Thus, EQSH is a complete interpolation system for equality sharing, *regardless of whether equality sharing is implemented in its original form, or by model-based theory combination.*

Having an interpolation system for DPLL and an interpolation system for equality sharing, we have all the ingredients for an interpolation system for DPLL($\mathcal{T}$). Let A and B be two sets of ground $\mathcal{T}$-clauses, for which we need to find a $\mathcal{T}$-interpolant. Let $CP_{\mathcal{T}}$ be the set of the $\mathcal{T}$-conflict clauses that appear in the DPLL($\mathcal{T}$)-refutation of $A \cup B$ (cf. Definition 11). Such a refutation shows that $A \cup B$ is $\mathcal{T}$-unsatisfiable, by showing that $A \cup B \cup CP_{\mathcal{T}}$ is propositionally unsatisfiable. An interpolation system for DPLL($\mathcal{T}$) will be given by an interpolation system for propositional resolution plus partial interpolants for the $\mathcal{T}$-conflict clauses. A clause C is a $\mathcal{T}$-conflict clause, because its negation $\neg C$, which is a set, or conjunction, of literals, was found $\mathcal{T}$-unsatisfiable. Then, $(\neg C)|_A \wedge (\neg C)|_B$ is $\mathcal{T}$-unsatisfiable, and we can compute a $\mathcal{T}$-interpolant of $((\neg C)|_A, (\neg C)|_B)$ by EQSH. This $\mathcal{T}$-interpolant provides the partial interpolant for C.

We call the resulting interpolation system HKPYM–T, because it is obtained by adding to HKPYM (cf. Definition 13) another case for $\mathcal{T}$-conflict clauses. The case for $\mathcal{T}$-conflict clauses is a third base case, because they are sort of input clauses from the point of view of the propositional engine:

Definition 20 (HKPYM–T interpolation system). *Let $c\colon C$ be a clause that appears in a DPLL($\mathcal{T}$)-refutation of $A \cup B$:*

– *If $c\colon C \in A$, then $PI(c) = \bot$,*

- *If* $c\colon C \in B$, *then* $PI(c) = \top$,
- *If* $c\colon C$ *is generated by* T-Conflict, $PI(c)$ *is the* $\mathcal{T}$*-interpolant of* $((\neg C)|_A, (\neg C)|_B)$ *produced by EQSH from the refutation* $\neg C \vdash_{\mathcal{T}} \bot$;
- *If* $c\colon C \vee D$ *is a propositional resolvent of* $p_1\colon C \vee l$ *and* $p_2\colon D \vee \neg l$ *then:*
  - *If* l *is A-colored, then* $PI(c) = PI(p_1) \vee PI(p_2)$,
  - *If* l *is B-colored, then* $PI(c) = PI(p_1) \wedge PI(p_2)$ *and*
  - *If* l *is transparent, then* $PI(c) = (l \vee PI(p_1)) \wedge (\neg l \vee PI(p_2))$.

The case analysis on literals resolved upon remains unchanged, because the requirement that all theories in $\mathcal{T}$ are equality-interpolating guarantees that the $\mathcal{T}$-conflict clauses do not introduce in the proof AB-mixed literals. The completeness of HKPYM–T follows from the completeness of HKPYM and EQSH.

7 Future Work

We are working on interpolation for superposition [3] in order to obtain an interpolation system for the theorem proving method DPLL($\Gamma + \mathcal{T}$) [4], which integrates a superposition-based inference system Γ into DPLL($\mathcal{T}$). Interpolation for ground superposition proofs was approached in [27] and then explored further in [23], using a notion of *colored proof*, which is stronger than colorable, since it excludes AB-mixed clauses. The observation that a separating ordering makes proofs colorable (cf. Lemma 2) generalizes easily to ground proofs in a full-fledged superposition-based inference system [3]. The analysis of interpolation and equality reported here means that for DPLL(Γ+$\mathcal{T}$) we need to assume that the built-in theories in $\mathcal{T}$ are equality-interpolating and the ordering used by Γ is separating. The remark that the interpolation system of [35] for equality sharing works also for model-based theory combination is another step towards interpolation in DPLL(Γ+$\mathcal{T}$), since DPLL(Γ+$\mathcal{T}$) uses model-based theory combination.

References

1. Bonacina, M.P.: On theorem proving for program checking – Historical perspective and recent developments. In: Fernandez, M. (ed.) Proc. of the 12th Int. Symp. on Principles and Practice of Declarative Programming, pp. 1–11. ACM Press, New York (2010)
2. Bonacina, M.P., Dershowitz, N.: Abstract canonical inference. ACM Trans. on Computational Logic 8(1), 180–208 (2007)
3. Bonacina, M.P., Johansson, M.: On theorem proving with interpolation for program checking. In: Technical report, Dipartimento di Informatica. Università degli Studi di Verona (April 2011)
4. Bonacina, M.P., Lynch, C.A., de Moura, L.: On deciding satisfiability by theorem proving with speculative inferences. Journal of Automated Reasoning, 1–29 (in press); (Published online December 22, 2010) doi:10.1007/s10817-010-9213-y
5. Bradley, A.R., Manna, Z. (eds.): The Calculus of Computation – Decision Procedures with Applications to Verification. Springer, Heidelberg (2007)

6. Brillout, A., Kroening, D., Rümmer, P., Wahl, T.: An Interpolating Sequent Calculus for Quantifier-Free Presburger Arithmetic. In: Giesl, J., Hähnle, R. (eds.) IJCAR 2010. LNCS, vol. 6173, pp. 384–399. Springer, Heidelberg (2010)
7. Brillout, A., Kroening, D., Rümmer, P., Wahl, T.: Program verification via Craig interpolation for Presburger arithmetic with arrays. Notes of the 6th Int. Verification Workshop (2010), `http://www.philipp.ruemmer.org/`
8. Bruttomesso, R., Ghilardi, S., Ranise, S.: Rewriting-based quantifier-free interpolation for a theory of arrays. In: Proc. of the 22nd Int. Conf. on Rewriting Techniques and Applications, LIPICS. Leibniz-Zentrum für Informatik, Dagsthul Publishing (2011)
9. Bruttomesso, R., Rollini, S., Sharygina, N., Tsitovich, A.: Flexible interpolation generation in satisfiability modulo theories. In: Proc. of the 14th Int. Conf. on Computer-Aided Design, pp. 770–777. IEEE Computer Society Press, Los Alamitos (2010)
10. Chadha, R., Plaisted, D.A.: On the mechanical derivation of loop invariants. Journal of Symbolic Computation 15(5-6), 705–744 (1993)
11. Cimatti, A., Griggio, A., Sebastiani, R.: Efficient Interpolant Generation in Satisfiability Modulo Theories. In: Ramakrishnan, C.R., Rehof, J. (eds.) TACAS 2008. LNCS, vol. 4963, pp. 397–412. Springer, Heidelberg (2008)
12. Cimatti, A., Griggio, A., Sebastiani, R.: Interpolant Generation for UTVPI. In: Schmidt, R. (ed.) CADE-22. LNCS (LNAI), vol. 5663, pp. 167–182. Springer, Heidelberg (2009)
13. de Moura, L., Bjørner, N.: Model-based theory combination. In: Krstić, S., Oliveras, A. (eds.) CAV 2007. ENTCS, vol. 198(2), pp. 37–49. Elsevier, Amsterdam (2008)
14. D'Silva, V.: Propositional Interpolation and Abstract Interpretation. In: Gordon, A.D. (ed.) ESOP 2010. LNCS, vol. 6012, pp. 185–204. Springer, Heidelberg (2010)
15. D'Silva, V., Kroening, D., Purandare, M., Weissenbacher, G.: Interpolant Strength. In: Barthe, G., Hermenegildo, M. (eds.) VMCAI 2010. LNCS, vol. 5944, pp. 129–145. Springer, Heidelberg (2010)
16. Fuchs, A., Goel, A., Grundy, J., Krstić, S., Tinelli, C.: Ground Interpolation for the Theory of Equality. In: Kowalewski, S., Philippou, A. (eds.) TACAS 2009. LNCS, vol. 5505, pp. 413–427. Springer, Heidelberg (2009)
17. Goel, A., Krstić, S., Tinelli, C.: Ground Interpolation for Combined Theories. In: Schmidt, R. (ed.) CADE-22. LNCS, vol. 5663, pp. 183–198. Springer, Heidelberg (2009)
18. Henzinger, T.A., Jhala, R., Majumdar, R., McMillan, K.L.: Abstractions from proofs. In: Leroy, X. (ed.) Proc. of the 31st ACM SIGACT-SIGPLAN Symp. on Principles of Programming Languages, pp. 232–244. ACM Press, New York (2004)
19. Hoder, K., Kovács, L., Voronkov, A.: Interpolation and Symbol Elimination in Vampire. In: Giesl, J., Hähnle, R. (eds.) IJCAR 2010. LNCS (LNAI), vol. 6173, pp. 188–195. Springer, Heidelberg (2010)
20. Huang, G.: Constructing Craig interpolation formulas. In: Proc. of the 1st Annual Int. Conf. on Computing and Combinatorics, pp. 181–190. Springer, Heidelberg (1995)
21. Kapur, D., Majumdar, R., Zarba, C.G.: Interpolation for data structures. In: Devambu, P. (ed.) Proc. of the 14th ACM SIGSOFT Symp. on the Foundations of Software Engineering, ACM Press, New York (2006)
22. Kovács, L., Voronkov, A.: Finding loop invariants for programs over arrays using a theorem prover. In: Chechik, M., Wirsing, M. (eds.) FASE 2009. LNCS, vol. 5503, pp. 470–485. Springer, Heidelberg (2009)

23. Kovács, L., Voronkov, A.: Interpolation and Symbol Elimination. In: Schmidt, R. (ed.) CADE-22. LNCS (LNAI), vol. 5663, pp. 199–213. Springer, Heidelberg (2009)
24. Krajíček, J.: Interpolation theorems, lower bounds for proof systems, and independence results for bounded arithmetic. Journal of Symbolic Logic 62(2), 457–486 (1997)
25. McMillan, K.L.: Interpolation and SAT-Based Model Checking. In: Hunt, W.J., Somenzi, F. (eds.) CAV 2003. LNCS, vol. 2725, pp. 1–13. Springer, Heidelberg (2003)
26. McMillan, K.L.: An interpolating theorem prover. Theoretical Computer Science 345(1), 101–121 (2005)
27. McMillan, K.L.: Quantified Invariant Generation Using an Interpolating Saturation Prover. In: Ramakrishnan, C.R., Rehof, J. (eds.) TACAS 2008. LNCS, vol. 4963, pp. 413–427. Springer, Heidelberg (2008)
28. McMillan, K.L.: Lazy Annotation for Program Testing and Verification. In: Cook, B., Jackson, P., Touili, T. (eds.) CAV 2010. LNCS, vol. 6174, pp. 104–118. Springer, Heidelberg (2010)
29. Nelson, G., Oppen, D.C.: Simplification by cooperating decision procedures. ACM Trans. on Programming Languages and Systems 1(2), 245–257 (1979)
30. Nelson, G., Oppen, D.C.: Fast decision procedures based on congruence closure. Journal of the ACM 27(2), 356–364 (1980)
31. Nieuwenhuis, R., Oliveras, A., Tinelli, C.: Solving SAT and SAT modulo theories: From an abstract Davis–Putnam–Logemann–Loveland procedure to DPLL(T). Journal of the ACM 53(6), 937–977 (2006)
32. Pudlàk, P.: Lower bounds for resolution and cutting plane proofs and monotone computations. Journal of Symbolic Logic 62(3), 981–998 (1997)
33. Sebastiani, R.: Lazy satisfiability modulo theory. Journal on Satisfiability, Boolean Modelling and Computation 3, 141–224 (2006)
34. Shankar, N.: Automated deduction for verification. ACM Computing Surveys 41(4), 40–96 (2009)
35. Yorsh, G., Musuvathi, M.: A Combination Method for Generating Interpolants. In: Nieuwenhuis, R. (ed.) CADE 2005. LNCS (LNAI), vol. 3632, pp. 353–368. Springer, Heidelberg (2005); Early version in MSR-TR-2004-108 (October 2004)
36. Zhang, L., Malik, S.: Validating SAT solvers using an independent resolution-based checker: practical implementations and other applications. In: Proc. of the Conf. on Design Automation and Test in Europe, pp. 10880–10885. IEEE Computer Society Press, Los Alamitos (2003)