Towards interpolation in an SMT-solver with integrated superposition[*]

Maria Paola Bonacina **Moa Johansson**
Dipartimento di Informatica
Università degli Studi di Verona
Strada Le Grazie 15, I-39134 Verona, Italy
mariapaola.bonacina@univr.it *moakristin.johansson@univr.it*

Abstract

Interpolation is a technique for extracting intermediate formulæ from a proof. It has applications in formal verification, where interpolation may enable a program analyser to discover information about intermediate program locations and states. We study interpolation in the theorem proving method DPLL($\Gamma+\mathcal{T}$), which integrates tightly a superposition based prover Γ in a DPLL($\mathcal{T}$) based SMT-solver to unite their respective strengths. We show how a first interpolation system for DPLL($\Gamma+\mathcal{T}$) can be obtained from interpolation systems for DPLL, equality sharing and Γ. We describe ongoing work on an interpolation system for Γ, by presenting and proving complete an interpolation system for the ground case, followed by a discussion of ongoing work on an extension to the general case. Thanks to the modular design of DPLL($\Gamma+\mathcal{T}$), its interpolation system can be extended easily beyond the ground case once a general interpolation system for Γ becomes available.

1 Introduction

Interpolation is a theorem proving technique which has recently found several applications in verification. Informally, interpolants are formulæ 'in-between' other fomulæ in a proof: for a proof of $A \vdash B$ with interpolant I, $A \vdash I$ and $I \vdash B$, with I only containing symbols shared between A and B. Interpolation was first proposed for *abstraction refinement* in software model checking, initially for propositional logic and propositional satisfiability [16], and then for quantifier free fragments of first-order theories and their combinations [21, 17, 11, 5, 8, 3]. In the Counter-Example Guided Abstraction Refinement paradigm, interpolants from the proof of unsatisfiability of the formula produced from a spurious counter-example may capture intermediate states in an error trace, and can be used to refine the abstraction by re-introducing predicates from the interpolants to exclude states leading to spurious errors.

Interpolation has also found applications for *invariant generation* in the context of inference systems for first-order logic with equality [18, 13, 9]. Here, one assumes that a k-step unwinding of a loop does not satisfy the post-condition. The formulæ expressing this produces a contradiction if the loop *does* satisfy the post-condition. An interpolant, containing only the symbols occurring in the loop body, can be extracted and used to guide the construction of a loop invariant [18].

A third application of interpolation is to supplement *annotation generation* by a weakest pre-condition calculus [19]. In this context, interpolation allows a static analyser to avoid inserting irrelevant program variables in annotations, such as procedure summaries.

[*]Research supported in part by grant no. 2007-9E5KM8 of the Ministero dell'Istruzione Università e Ricerca, Italy, and by COST Action IC0901 of the European Union.

The aim of this work is to develop an interpolation system for DPLL($\Gamma+\mathcal{T}$) [2], a new theorem proving method which integrates a first-order inference system Γ, based on resolution and superposition, into the DPLL($\mathcal{T}$) framework for satisfiability modulo theories. The motivation for DPLL($\Gamma+\mathcal{T}$) is to unite the strengths of resolution based provers, such as automated treatment of quantifiers, with those of SMT-solvers, such as built-in theories and scalability on large ground problems. All these features are crucial for applications to verification. For instance, formulæ with quantifiers are necessary to state invariants and to axiomatise theories without decision procedures. Heuristic techniques for instantiating variables in SMT-solvers can be used, but they can be fragile and require a lot of user effort to get right [15]. Thus, DPLL($\Gamma+\mathcal{T}$) has properties attractive to the application areas, exemplified above, where also interpolation has uses. Hence an interpolating version of DPLL($\Gamma+\mathcal{T}$) would be of interest for the formal verification community. The work described in this paper is still in progress; we describe how a first interpolation system for DPLL($\Gamma+\mathcal{T}$), thanks to its modular design, is built from interpolation systems for DPLL, equality sharing and Γ.

We will use the propositional interpolation system for DPLL independently discovered by Huang, Krajíček and Pudlàk [10, 20, 14], later reformulated and proved correct in the context of satisfiability modulo theories by Yorsh and Musuvathi [21]. We call this algorithm HKPYM from the initials of the authors. Yorsh and Musuvathi also gave an interpolation system for equality sharing, which we refer to as EQSH [21]. EQSH requires that the satisfiability procedures for the built in theories can produce proofs and interpolants. Then HKPYM and EQSH can be integrated to yield an interpolation system for DPLL($\mathcal{T}$) [21, 5, 8]. What remains for an interpolation system for DPLL($\Gamma+\mathcal{T}$) is an interpolation system for Γ. We present a novel complete interpolation system for Γ in the ground case and give a modular interpolation system for DPLL($\Gamma+\mathcal{T}$). We consider our interpolation system for superposition to be clearer and more general than previous work, because its working is specified for each generative inference, which was not done before. We conclude with a discussion of related work and an overview of ongoing work aiming at extending the ground interpolation system for Γ to proofs involving substitutions, under suitable restrictions. The interpolation system for DPLL(Γ+$\mathcal{T}$) is currently restricted to the ground case, but easily extendable to the non-ground case once such an interpolation system for Γ is available, which is the ultimate goal of this project.

2 Preliminaries

We assume the basic definitions commonly used in theorem proving. Equality in the inference systems will be denoted by $\simeq$ and the symbol $\bowtie$ stands for either $\simeq$ or $\not\simeq$.

Let A and B be two formulæ. We denote by Σ_A, and Σ_B, the set of constant, function and predicate symbols that occur in A, and B, respectively, and we use $\setminus$ for set difference. A non-variable symbol is *A-coloured*, if it is in $\Sigma_A \setminus \Sigma_B$, *B-coloured*, if it is in $\Sigma_B \setminus \Sigma_A$, and *transparent*, if it is in $\Sigma_T = \Sigma_A \cap \Sigma_B$. This extends to terms, literals and clauses:

Definition 2.1 *A term, literal or clause is* transparent *if all its symbols are transparent,* A-coloured *if it contains at least one A-coloured symbol, and the rest are transparent (similarly for* B-coloured*). Otherwise it is AB-mixed.*

A clause is *colourable* if it contains no AB-mixed literals. We use, ambiguously, $\mathcal{L}_A$ for the language of terms, literals or formulæ made of symbols in Σ_A; $\mathcal{L}_B$ and $\mathcal{L}_T$ are defined similarly for Σ_B and Σ_T, respectively. We let $\mathcal{L}_X$ stand for either $\mathcal{L}_A$, $\mathcal{L}_B$ or $\mathcal{L}_T$.

A theory is presented by a set $\mathcal{T}$ of sentences, meaning that the theory is the set of all logical consequences of $\mathcal{T}$. It is customary to call $\mathcal{T}$ itself a theory. Let $\Sigma_{\mathcal{T}}$ be the signature of $\mathcal{T}$, and $\mathcal{L}_{\mathcal{T}}$ the language of terms, literals or formulæ built from $\Sigma_{\mathcal{T}}$. Then, let $\mathcal{L}_T$ be the language of terms, literals or formulæ built from $\Sigma_{\mathcal{T}} \cup \Sigma_T$: in other words, whenever a theory is involved, theory symbols are considered transparent:

Definition 2.2 (Theory Interpolant) *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 is in $\mathcal{L}_T$. A formula I is a* reverse theory interpolant *of formulæ A and B such that $A, B \vdash_{\mathcal{T}} \perp$, if (i) $A \vdash_{\mathcal{T}} I$, (ii) $B, I \vdash_{\mathcal{T}} \perp$ and (iii) I is in $\mathcal{L}_T$.*

Reverse interpolants are more widely used in the context of theorem proving, since practical theorem provers work refutationally. In keeping with most of the literature, in the following we shall write "interpolant" for "reverse interpolant", unless the distinction is relevant. Furthermore, when it is clear from the context, we may omit the "theory" prefix and just write "interpolant". Similarly, we may use $\vdash$ instead of $\vdash_{\mathcal{T}}$.

Definition 2.3 (Projection) *Let C be a disjunction (conjunction) of literals. The* projection *of C on language $\mathcal{L}_X$, denoted $C|_X$, is the disjunction (conjunction) obtained from C by removing any literal whose atom is not in $\mathcal{L}_X$. By convention, if C is a disjunction and $C|_X$ is empty, then $C|_X = \perp$; if C is a conjunction and $C|_X$ is empty, then $C|_X = \top$.*

Many approaches to interpolation work by annotating each clause C in a refutation of A and B with auxiliary formulæ, called *partial interpolants*:

Definition 2.4 (Partial interpolant) *A* partial interpolant *$PI(C)$ of a clause C occurring in a refutation of $A \cup B$ is an interpolant of $A \wedge \neg(C|_A)$ and $B \wedge \neg(C|_B)$.*

By Definition 2.2 applied to Definition 2.4, a partial interpolant needs to satisfy the following requirements:

Proposition 2.1 *A partial interpolant for a clause C have to satisfy:*

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

We now give a brief overview of the DPLL($\mathcal{T}$) and DPLL($\Gamma + \mathcal{T}$) theorem proving methods for satisfiability modulo theories. DPLL($\mathcal{T}$) combines propositional reasoning by DPLL with decision procedures for specific theories. DPLL($\Gamma + \mathcal{T}$) is a further extension which also features an interface to a first-order prover with resolution and superposition. We refer to [2] for a description of DPLL($\Gamma + \mathcal{T}$), which includes DPLL($\mathcal{T}$). DPLL($\Gamma + \mathcal{T}$) works with *hypothetical clauses*, where the hypotheses are the connection between Γ-inferences and the partial model M maintained by DPLL($\mathcal{T}$). Hypothetical clauses have the form $H \triangleright C$, where C is a clause and the hypothesis H is a set of ground literals. The literals in H come from M and are the literals that were used as premises to infer C by a Γ-inference. DPLL(Γ+$\mathcal{T}$) employs model-based theory combination [6], which is a version of equality sharing where only equalities between ground terms are propagated. DPLL(Γ+$\mathcal{T}$) can be described as a transition system with two kinds of states: $M \mid\mid F$ (candidate model and set of clauses) and $M \mid\mid F \mid\mid C$ (candidate model, set of clauses and conflict clause). Let $S = \mathcal{R} \uplus P$ stand for the set of input clauses, where $\mathcal{R}$ is a set of non-ground clauses, without occurrences of $\mathcal{T}$-symbols, while P is a set of ground clauses that typically do contain $\mathcal{T}$-symbols. A transition system derivation for DPLL(Γ+$\mathcal{T}$) is defined as follows:

Definition 2.5 (Transition system derivation) *Let $\mathcal{U}$ stand for DPLL($\Gamma + \mathcal{T}$), and S be the input set $\mathcal{R} \uplus P$. A* transition system derivation*, or* $\mathcal{U}$-derivation*, is a sequence of state transitions:*
$\Delta_0 \Longrightarrow_{\mathcal{U}} \Delta_1 \Longrightarrow_{\mathcal{U}} \ldots \Delta_i \Longrightarrow_{\mathcal{U}} \Delta_{i+1} \Longrightarrow_{\mathcal{U}} \ldots$, 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 $\mathcal{U}$ and $\Delta_0 = \parallel F_0$, where $F_0 = \{\emptyset \triangleright C \mid C \in S\}$.

A transition system derivation is characterised by the sets $F^* = \bigcup_{i \geq 0} F_i$ of all generated clauses and $C^* = \{C_i | i > 0\}$ of all conflict clauses. A DPLL(Γ+$\mathcal{T}$) refutation is a refutation by propositional resolution plus $\mathcal{T}$-conflict clauses, which are derived when one of the theory solvers discovers an inconsistency with the current model, and inferences performed by Γ. We denote the proof tree produced by the $\mathcal{T}$-solver for a $\mathcal{T}$-conflict clause C, by $\Pi_{\mathcal{T}}(C)$.

Definition 2.6 (DPLL($\Gamma+\mathcal{T}$)-proof tree) *Given a DPLL($\Gamma+\mathcal{T}$)-derivation,*

$$\Delta_0 \underset{\mathcal{U}}{\Longrightarrow} \Delta_1 \underset{\mathcal{U}}{\Longrightarrow} \ldots \Delta_i \underset{\mathcal{U}}{\Longrightarrow} \Delta_{i+1} \underset{\mathcal{U}}{\Longrightarrow} \ldots,$$

for all $C \in C^*$ and $H \triangleright C \in F^*$, the DPLL($\Gamma+\mathcal{T}$)-proof tree $\Pi_{\mathcal{U}}(C)$ *of C is defined as follows:*

- *If $C \in F_0$, $\Pi_{\mathcal{U}}(C)$ consists of a node labelled by C;*
- *If C is generated by resolving conflict clause C_1 with justification C_2, $\Pi_{\mathcal{U}}(C)$ consists of a node labelled by C with sub-trees $\Pi_{\mathcal{U}}(C_1)$ and $\Pi_{\mathcal{U}}(C_2)$;*
- *If C is a $\mathcal{T}$-conflict clause, $\Pi_{\mathcal{U}}(C) = \Pi_{\mathcal{T}}(C)$;*
- *If $H \triangleright C$ is inferred by a Γ-based transition from hypothetical clauses $\{H_1 \triangleright C_1, \ldots, H_m \triangleright C_m\}$ and literals $\{l_{m+1}, \ldots, l_k\}$, $\Pi_{\mathcal{U}}(H \triangleright C)$ consists of a node labelled by $H \triangleright C$ with m sub-trees $\Pi_{\mathcal{U}}(H_1 \triangleright C_1), \ldots, \Pi_{\mathcal{U}}(H_m \triangleright C_m)$.*

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

Hypotheses need to be discharged when the hypothetical clause $H \triangleright \square$ is generated. The system then switches to conflict resolution mode, with $\neg H$ as the conflict clause. A refutation is reached only when $\neg H$ is reduced to $\square$. Thus, a DPLL($\Gamma+\mathcal{T}$)-refutation is obtained by attaching a non-ground proof tree with $H \triangleright \square$, or $\neg H$, as root, to a ground proof tree with $\neg H$ among its leaves and $\square$ as root.

An *interpolation system* is a mechanism to annotate each clause C in a refutation of A and B with a partial interpolant. To define an interpolation system, one needs to define the partial interpolants that it associates to the clauses in a proof. Since each clause in a proof is generated by some inference rule, the definition of an interpolation system needs to cover all possibilities. The fundamental property of an interpolation system is *completeness*:

Definition 2.7 (Complete interpolation system) *An interpolation system is* complete *for inference system Γ, or transition system $\mathcal{U}$, if for all sets of clauses A and B, such that $A \cup B$ is unsatisfiable, and for all refutations of $A \cup B$ by Γ, or $\mathcal{U}$, respectively, it generates an interpolant of (A, B).*

The key property of partial interpolants is that $PI(\square)$ is an interpolant of A and B. Thus, 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 indeed partial interpolants.

3 An Interpolation System for DPLL($\Gamma + \mathcal{T}$)

A complete interpolation system for DPLL($\Gamma + \mathcal{T}$) must be able to compute partial interpolants for each clause in the proof tree in Def. 2.6, that is: propositional resolvents, $\mathcal{T}$-conflict clauses, and clauses derived by Γ. The latter are covered by the new interpolation system given in this section. Propositional resolvents are dealt with by a propositional interpolation system such as HKPYM [10, 20, 14, 21]. Since $\mathcal{T}$ is a combination, $\mathcal{T}$-conflict clauses are handled by EQSH [21], which requires that the component theories are *equality interpolating*:

Definition 3.1 (Equality Interpolating Theory) *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-coloured ground term and t_b is a B-coloured ground term, then $A \wedge B \models_{\mathcal{T}} t_a \simeq t \wedge t_b \simeq t$ for some transparent ground term t.*

Several theories used in practice are indeed equality interpolating, for example quantifier-free theories of uninterpreted functions and linear arithmetic [21]. Without this requirement, the notion of transparency is not stable: if $t_a \simeq t_b$ without any transparent t such that $t_a \simeq t$ and $t_b \simeq t$, the congruence class of t_a and t_b includes no transparent term, which means a coloured term should "become" transparent, to serve as a representative for terms of both colours. This is clearly undesirable as transparent terms are those used to build interpolants. A similar issue arises for Γ, which also reasons about equalities. If an AB-mixed equality $t_a \simeq t_b$ is derived, it can be used to simplify clauses in A, introducing B-coloured symbols, which now should be considered transparent as they have become shared between A and B. Proofs without AB-mixed equalities were termed *colourable* in [8]:

Definition 3.2 (Colourable proof) *A proof is* colourable *if it contains no AB-mixed literals.*

We proceed to connect the notion of equality-interpolating theory with the following requirement, that appeared in [18], under the name *AB-oriented ordering*, and then in [12]:

Definition 3.3 (Separating ordering) *An ordering $\succ$ is* separating *if $t \succ s$ whenever s is transparent and t is not, for all ground terms, or literals, s and t.*

If the theory is equality-interpolating, whenever $t_a \simeq t_b$ holds, $t_a \simeq t$ and $t_b \simeq t$ also hold, and a separating ordering ensures that, if $t_a \simeq t$ and $t_b \simeq t$ are derived, t replaces t_a and t_b, or becomes the representative of the congruence class of t_a and t_b.

Lemma 3.1 *If the ordering is separating, all ground Γ-proof-trees are colourable.*

The proof is by induction on the structure of the proof tree (see [1]). To get a superposition based theorem prover to produce ground colourable proofs, it is thus sufficient to adopt a separating ordering. Separating orderings exist, and were implemented, for instance, in Vampire [9]. From now on, we assume that *the built-in theories $\mathcal{T}_i$, $1 \leq i \leq n$, are equality-interpolating*, and that *the ordering $\succ$ for Γ-inferences is separating*, so that all ground proofs are colourable. Under these assumptions, we present a complete interpolation system for Γ in the ground case, where the inferences rules, with premises and consequences labelled for later reference, are as follows (see [2] for full details):

Resolution:

$$\frac{p_1\colon (C \vee l) \quad p_2\colon (D \vee \neg l)}{c\colon (C \vee D)} \qquad \forall m \in C\colon l \succ m \quad \forall m \in D\colon \neg l \succ m$$

Paramodulation:

$$\frac{p_1\colon (C \vee s \simeq r) \quad p_2\colon (D \vee l[s])}{c\colon (C \vee l[r] \vee D)} \qquad (i) \quad (ii) \quad (iii)$$

Superposition:

$$\frac{p_1\colon (C \vee s \simeq r) \quad p_2\colon (D \vee l[s] \bowtie t)}{c : (C \vee l[r] \bowtie t \vee D)} \qquad (i) \quad (ii) \quad (iv) \quad (v)$$

where (i) $s \succ r$, (ii) $\forall m \in C\colon (s \simeq r) \succ m$, (iii) $\forall m \in D\colon l[s] \succ m$, (iv) $l[s] \succ t$, (v) $\forall m \in D\colon (l[s] \bowtie t) \succ m$; and Simplification inferences are instances of Paramodulation/Superposition, where c replaces p_2, C is empty, and (i) is the only side condition.

Definition 3.4 (GΓI interpolation system) *Let $c\colon C$ be a clause that appears in a ground Γ-refutation of $A \cup B$:*

- *If $c\colon C \in A$, then $PI(c) = \perp$, if $c\colon C \in B$, then $PI(c) = \top$.*
- *If $c\colon C$ is generated from premises p_1 and p_2 by a Γ-inference, $PI(c)$ is defined as follows:*
  - *Resolution: $c\colon C \vee D$ generated from $p_1 : C \vee l$ and $p_2\colon D \vee \neg l$*
    - *l is A-coloured: $PI(c) = PI(p_1) \vee PI(p_2)$*
    - *l is B-coloured: $PI(c) = PI(p_1) \wedge PI(p_2)$*
    - *l is transparent: $PI(c) = (l \vee PI(p_1)) \wedge (\neg l \vee PI(p_2))$*
  - *Paramodulation/Superposition/Simplification: $c\colon C \vee l[r] \vee D$ generated from $p_1 : C \vee s \simeq r$ and $p_2\colon D \vee l[s]$*
    - *$s \simeq r$ is A-coloured: $PI(c) = PI(p_1) \vee PI(p_2)$*
    - *$s \simeq r$ is B-coloured: $PI(c) = PI(p_1) \wedge PI(p_2)$*

- $*$ *$s \simeq r$, $l[s]$ are transparent:* $PI(c) = (s \simeq r \vee PI(p_1)) \wedge (l[s] \vee PI(p_2))$
- $*$ *$s \simeq r$ is transparent, $l[s]$ is not:* $PI(c) = (s \simeq r \vee PI(p_1)) \wedge (s \not\simeq r \vee PI(p_2))$.

Superposition is treated like Paramodulation, with $l[s]$ replaced by $l[s] \bowtie t$, and the case for Simplification is subsumed by those for Paramodulation and Superposition. As we assume a separating ordering, transparent terms are smaller than coloured ones and we do not need to consider the case where $s \simeq r$ is coloured and $l[s]$ is transparent for paramodulation inferences. In such a case, s must be transparent, as it also occurs in the transparent literal $l[s]$, and r must be coloured. The separating ordering would thus re-orient such an equation to $r \simeq s$, and only inferences rewriting a coloured term are allowed.

Theorem 1 *If the ordering is separating, $G\Gamma I$ is a complete interpolation system for all ground Γ-refutations.*

Proof: By induction on the structure of the proof. We need to prove that for all clauses $c\colon C$ in the refutation, the partial interpolants satisfy the requirements in Proposition 2.1.
Base cases: *$c : C$ is an input clause*. Trivial.
Inductive cases:
Inductive hypothesis: for $k \in \{1, 2\}$ it holds that:

1. $A \wedge \neg(p_k|_A) \vdash PI(p_k)$ or, equivalently, $A \vdash p_k|_A \vee PI(p_k)$
2. $B \wedge \neg(p_k|_B) \wedge PI(p_k) \vdash \perp$ or, equivalently, $B \wedge PI(p_k) \vdash p_k|_B$
3. $PI(p_k)$ is transparent.

Resolution: $c\colon C \vee D$ generated from $p_1\colon C \vee l$ and $p_2\colon D \vee \neg l$

- l is A-coloured: $l|_A = l$, $(\neg l)|_A = \neg l$, $l|_B = \perp = (\neg l)|_B$
  1. $A \vdash (C \vee D)|_A \vee PI(p_1) \vee PI(p_2)$. From inductive hypothesis (1) we have $A \vdash (C \vee l)|_A \vee PI(p_1)$ and $A \vdash (D \vee \neg l)|_A \vee PI(p_2)$. A resolution step gives $A \vdash (C \vee D)|_A \vee PI(p_1) \vee PI(p_2)$ as desired.
  2. $B \wedge (PI(p_1) \vee PI(p_2)) \vdash (C \vee D)|_B$. From inductive hypothesis (2) we have $B \wedge PI(p_1) \vdash C|_B$ and $B \wedge PI(p_2) \vdash D|_B$ from which the inductive conclusion follows.
  3. Transparency of the partial interpolant follows from the inductive hypothesis.
- l is B-coloured: Symmetric to the previous case.
- l is transparent: $l|_A = l = l|_B$, $(\neg l)|_A = \neg l = (\neg l)|_B$
  1. $A \wedge \neg(C \vee D)|_A \vdash (l \vee PI(p_1)) \wedge (\neg l \vee PI(p_2))$ or, equivalently, $A \wedge \neg C|_A \wedge \neg D|_A \vdash (l \vee PI(p_1)) \wedge (\neg l \vee PI(p_2))$. From inductive hypothesis (1) we have $A \wedge \neg C|_A \vdash l \vee PI(p_1)$ and $A \wedge \neg D|_A \vdash \neg l \vee PI(p_2)$, which together give the desired result.
  2. $B \wedge (l \vee PI(p_1)) \wedge (\neg l \vee PI(p_2)) \vdash (C \vee D)|_B$. By case analysis on l in $PI(c)$: 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 $PI(p_1)$; so that we need to establish:
     (a) $B \wedge l \wedge PI(p_2) \vdash (C \vee D)|_B$. From inductive hypothesis (2) we have $B \wedge PI(p_2) \vdash D|_B \vee \neg l$ whence $B \wedge l \wedge PI(p_2) \vdash D|_B$.
     (b) $B \wedge \neg l \wedge PI(p_1) \vdash (C \vee D)|_B$. From inductive hypothesis (2) we have $B \wedge PI(p_1) \vdash C|_B \vee l$ whence $B \wedge \neg l \wedge PI(p_1) \vdash C|_B$.
  3. Transparency of the partial interpolant follows from the inductive hypothesis and the assumption that l is transparent.

Paramodulation/Superposition/Simplification: $c\colon C \vee l[r] \vee D$ generated from $p_1\colon C \vee s \simeq r$ and $p_2\colon D \vee l[s]$

- $s \simeq r$ is A-coloured: either s and r are both A-coloured, or, since $s \succ r$, s is A-coloured and r is transparent; since there are no AB-mixed literals, either $l[s]$ and $l[r]$ are both A-coloured, or $l[s]$ is A-coloured and $l[r]$ is transparent; thus, we have: $(s \simeq r)|_A = (s \simeq r)$, $l[s]|_A = l[s]$, $l[r]|_A = l[r]$, $(s \simeq r)|_B = \perp$, $l[s]|_B = \perp$

   1. $A \vdash (C \vee l[r] \vee D)|_A \vee PI(p_1) \vee PI(p_2)$. Inductive hypothesis (1) gives $A \vdash C|_A \vee s \simeq r \vee PI(p_1)$ and $A \vdash D|_A \vee l[s] \vee PI(p_2)$; Thus, the inductive conclusion follows by a paramodulation step.
   2. $B \wedge (PI(p_1) \vee PI(p_2)) \vdash (C \vee l[r] \vee D)|_B$. From inductive hypothesis (2) we have $B \wedge PI(p_1) \vdash C|_B$ and $B \wedge PI(p_2) \vdash D|_B$, which proves the inductive conclusion.
   3. The partial interpolant is transparent by inductive hypothesis.

- $s \simeq r$ is B-coloured: Symmetric to the previous case.
- $s \simeq r$ and $l[s]$ are transparent: $l[r]$ is also transparent, and all three literals are unaffected by projections.
   1. $A \wedge \neg(C|_A) \wedge \neg l[r] \wedge \neg(D|_A) \vdash (s \simeq r \vee PI(p_1)) \wedge (l[s] \vee PI(p_2))$. From inductive hypothesis (1) we have $A \wedge \neg C|_A \vdash s \simeq r \vee PI(p_1)$ and $A \wedge \neg D|_A \vdash l[s] \vee PI(p_2)$, which together give the desired result.
   2. $B \wedge (s \simeq r \vee PI(p_1)) \wedge (l[s] \vee PI(p_2))) \vdash (C \vee l[r] \vee D)|_B$. We do a case analysis on $s \simeq r$ and $l[s]$:
      (a) If $s \simeq r$ and $l[s]$ are both true, then $l[r]$ is true.
      (b) If $s \simeq r$ is true and $l[s]$ is false, then $PI(p_2)$ must be true and $D \vee l[s]$ is equivalent to D, so that induction hypothesis (2) gives $B \wedge PI(p_2) \vdash D|_B$.
      (c) If $s \simeq r$ is false and $l[s]$ is true, then $PI(p_1)$ must be true and $C \vee s \simeq r$ is equivalent to C, so that induction hypothesis (2) gives $B \wedge PI(p_1) \vdash C|_B$.
      (d) If $s \simeq r$ and $l[s]$ are both false, then $PI(p_1)$ and $PI(p_2)$ must be true and induction hypothesis (2) gives $B \wedge PI(p_1) \vdash C|_B$ and $B \wedge PI(p_2) \vdash D|_B$.
   3. Transparency of the partial interpolant follows from the inductive hypothesis and the assumption that $s \simeq r$ and $l[s]$ are transparent.
- $s \simeq r$ is transparent, $l[s]$ is not:
   1. $A \vdash (C \vee l[r] \vee D)|_A \vee (s \simeq r \vee PI(p_1)) \wedge (s \not\simeq r \vee PI(p_2))$. This is equivalent to: $A \wedge ((s \not\simeq r \wedge \neg PI(p_1)) \vee (s \simeq r \wedge \neg PI(p_2))) \vdash (C \vee l[r] \vee D)|_A$. We perform a case analysis on $s \simeq r$:
      (a) If $s \simeq r$ is false, $s \simeq r \wedge \neg PI(p_2)$ is false, and it suffices to establish $A \wedge s \not\simeq r \wedge \neg PI(p_1) \vdash (C \vee l[r] \vee D)|_A$. By induction hypothesis (1) we have $A \wedge s \not\simeq r \wedge \neg PI(p_1) \vdash C|_A$, which suffices.
      (b) If $s \simeq r$ is true, $s \not\simeq r \wedge \neg PI(p_1)$ is false, and it suffices to establish $A \wedge s \simeq r \wedge \neg PI(p_2) \vdash (C \vee l[r] \vee D)|_A$ or, equivalently, $A \wedge s \simeq r \wedge \neg PI(p_2) \vdash (C \vee l[s] \vee D)|_A$ since $s \simeq r$ holds. By induction hypothesis (1) we have $A \wedge \neg PI(p_2) \vdash (l[s] \vee D)|_A$, and we are done.
   2. $B \wedge (s \simeq r \vee PI(p_1)) \wedge (s \not\simeq r \vee PI(p_2))) \vdash (C \vee l[r] \vee D)|_B$. By case analysis on $s \simeq r$:
      (a) If $s \simeq r$ is true, $s \simeq r \vee PI(p_1)$ is subsumed, $s \not\simeq r$ is false and $PI(p_2)$ must be true. Thus, it suffices to establish $B \wedge s \simeq r \wedge PI(p_2) \vdash (C \vee l[r] \vee D)|_B$, which is equivalent to $B \wedge s \simeq r \wedge PI(p_2) \vdash (C \vee l[s] \vee D)|_B$, since $s \simeq r$ holds. By induction hypothesis (2) we have $B \wedge PI(p_2) \vdash l[s]|_B \vee D|_B$, which closes this case.
      (b) If $s \not\simeq r$ is true, $s \not\simeq r \vee PI(p_2)$ is subsumed, $s \simeq r$ is false and $PI(p_1)$ must be true. Thus, we need to establish $B \wedge s \not\simeq r \wedge PI(p_1) \vdash (C \vee l[r] \vee D)|_B$. By induction hypothesis (2) we have $B \wedge s \not\simeq r \wedge PI(p_1) \vdash C|_B$, which suffices.
   3. Transparency follows from the transparency of $s \simeq r$ and the inductive hypothesis.

□

Having obtained a complete interpolation system for Γ, we can now define an interpolation system for DPLL($\Gamma+\mathcal{T}$):

Definition 3.5 (**I^* interpolation system**) *Let c: C be a clause that appears in a DPLL($\Gamma+\mathcal{T}$)-refutation of $A \cup B$:*

- *If* $c\colon C \in A$, *then* $PI(c) = \perp$, *if* $c\colon C \in B$, *then* $PI(c) = \top$.
- *If* $c\colon C$ *is a* $\mathcal{T}$*-conflict clause,* $PI(c)$ *is the* $\mathcal{T}$*-interpolant of* $((\neg C)|_A, (\neg C)|_B)$ *produced by EQSH from the refutation* $\neg C \vdash_{\mathcal{T}} \perp$*;*
- *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*-coloured, then* $PI(c) = PI(p_1) \vee PI(p_2)$*,*
  - *If* l *is* B*-coloured, then* $PI(c) = PI(p_1) \wedge PI(p_2)$ *and*
  - *If* l *is transparent, then* $PI(c) = (l \vee PI(p_1)) \wedge (\neg l \vee PI(p_2))$*.*
- *If* $c\colon C$ *is a hypothetical clause* $H \rhd C$ *inferred by a generative* Γ*-based transition from premises* $\{H_1 \rhd C_1, \ldots, H_m \rhd C_m\}$ *and* $\{l_{m+1}, \ldots, l_k\}$*, then* $PI(c)$ *is the partial interpolant produced by the interpolation system* $G\Gamma I$ *for the* Γ*-inference inferring* C *from premises* $C_1, \ldots, C_m, l_{m+1}, \ldots, l_k$*.*

The partial interpolant for a hypothetical clause $H \rhd C$ is given by the partial interpolant for the corresponding regular clause C, because the Γ-inference embedded in a Γ-based transition ignores hypotheses, and, when $H \rhd \square$ is generated, the hypotheses in H are discharged by propositional resolution steps, whose partial interpolant is computed as in HKPYM. In summary, the modular construction of DPLL($\Gamma + \mathcal{T}$) allows us to define its interpolation system from the interpolation systems of its components. Furthermore, this allows us to simply replace $G\Gamma I$ by a general interpolation system for Γ once available. 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, and the completeness of I^* thus follows from the completeness of its component interpolation systems.

4 Related Work

Interpolation for *coloured* superposition proofs was first considered by McMillan [18], and further studied, with some criticism that restricted it to ground proofs, by Kovàcs and Voronkov [13]. *Coloured* is a stronger requirement than *colourable*: each inference may involve at most one colour, so that not only AB-mixed literals, but also AB-mixed clauses are forbidden. The main similarity between our work and these is the adoption of a separating ordering, where transparent literals are smaller than coloured ones. However, our approach differs is several ways: Firstly, in the ground case, we relax the requirement of coloured proofs, and only require the notion of colourable proofs from [8]. We showed that when a separating ordering is used, every ground Γ-refutation is colourable. Secondly, the target inference system in [13] is LASCA (Linear Arithmetic Superposition CAlculus), which is superposition with linear arithmetic built in. We do not consider arithmetic within Γ, because in DPLL($\Gamma + \mathcal{T}$) arithmetic is handled by the DPLL($\mathcal{T}$) part, and therefore by an interpolating decision procedure for arithmetic (e.g. [17]). Thirdly, our notion of partial interpolant is different from [13], which focused on proving existence of partial interpolants[1] only for transparent ground clauses in coloured proof-trees. No explicit interpolation system is given in either [13] or [18]. We define explicitly the partial interpolants for every generative rule in Γ. Thus, we consider the interpolation system I^* to be more concrete and representing a more direct generalisation of propositional interpolation systems to ground first-order logic.

Christ and Hoenicke considers interpolation in the presence of quantifiers in the context of DPLL($\mathcal{T}$) [4]. They assume instantiations are found by heuristic methods, such as triggering, rather than by unification as in superposition. The interpolation method is based on McMillan's ground interpolation system for resolution [17], extended to introduce quantifiers in interpolants, when instantiations introduce coloured terms. This approach thus goes beyond colourable proofs for resolution. Equality reasoning is assumed to be handled by an interpolating decision procedure.

Our interpolation system for ground superposition also covers proofs in EUF (Equality with Uninterpreted Functions). McMillan's interpolation system for EUF in [17] consists of inference rules for reflexivity, symmetry, congruence, transitivity and contradiction, instrumented to compute formulas

[1]Referred to as *C-interpolants* in [13].

akin to partial interpolants for each inference step. There are several versions of each rule, depending on side conditions relating to the colour of the inferences leading up to the conclusion. The interpolation system by Fuchs et al. [7], on the other hand, works on colourable congruence graphs, generated by the congruence closure algorithm. While we rely on the separating ordering to ensure that no AB-mixed literals are present in the proof, Fuchs et al. use the fact that EUF is equality interpolating and perform some modifications on the congruence graph, introducing transparent constants to separate A-coloured and B-coloured terms as needed, essentially implementing the requirement that the theory be equality-interpolating. While our interpolation system will include all transparent literals derived from A, the specialised congruence closure method can summarise chains originating only from A, and only consider adding the last transparent term in such a chain. This means that it tend to produce shorter interpolants, which for some applications may be desirable. At this stage of research, we have focused on completeness of the interpolation system, with analysis of the properties of interpolants for various applications left as further work. Last, the algorithm in [7] is restricted to EUF only, while our aim is a much more general interpolation system.

5 Current and Future Work

We reported on ongoing work on interpolation for the theorem proving method DPLL($\Gamma+\mathcal{T}$). We showed how an interpolation system for DPLL($\Gamma+\mathcal{T}$) can be constructed modularly from interpolation systems for DPLL, equality sharing and for Γ, a first-order resolution and superposition based prover. We presented and proved correct a novel interpolation system for Γ in the ground case for colourable proofs. Current work in progress aims at extending the interpolation system to general proofs with substitution under suitable restrictions. The interpolation system for general Γ-proofs can then simply be plugged into the interpolation system for DPLL($\Gamma+\mathcal{T}$), to extend it beyond ground proofs. In order to generalise the ground interpolation system for Γ to some class of proofs in full first order logic, we need to extend our approach to handle variables and substitutions. In the ground case, we can ensure colours are stable by imposing a separating ordering, which prevents equations between A-coloured and B-coloured terms from being generated. In the general case, a separating ordering is no longer sufficient to ensure that proofs are colourable, as we may unify two literals of different colour, which may have the side effect of generating AB-mixed literals by substitution. One way of avoiding AB-mixed literals is to impose the restriction that the proof is coloured. For coloured proofs, we have that substitution and projection commute, allowing a straightforward extension from the ground interpolation system of the cases where both pivots, or literals paramodulated from and into, have the same colour. However, non-ground coloured proofs also have to deal with the cases where one of the premises is transparent and the other coloured. Thus, excluding AB-mixed literals is not enough to ensure colours are stable, as substitutions may also paint transparent literals as a side effect. Our current work is concerned with extending the interpolation system to these inferences, in which the partial interpolants may contain quantifiers. One of the approaches we are studying is *procrastination*, suggested by McMillan [18], which involves the addition of a special inference rule that delays superposition steps and record restrictions on variable instantiations. We are also considering *instance purification* [4], where coloured literals occurring in partial interpolants are replaced by quantified variables.

References

[1] M.P. Bonacina and M. Johansson. On theorem proving with interpolation for program checking. Technical report, Università degli Studi di Verona, April 2011.

[2] M.P. Bonacina, C.A. Lynch, and L. de Moura. On deciding satisfiability by theorem proving with speculative inferences. *Journal of Automated Reasoning*, In press:1–29. Published online 22 December 2010 (doi: 10.1007/s10817-010-9213-y).

[3] A. Brillout, D. Kroening, P. Rümmer, and T. Wahl. An interpolating sequent calculus for quantifier-free Presburger arithmetic. In *Proceedings of the International Joint Conference on Automated Reasoning*, volume 6173 of *LNAI*, pages 384–399. Springer, 2010.

[4] J. Christ and J. Hoenicke. Instantiation-based interpolation for quantified formulae. Satisfiability Modulo Theories Workshop, 2010.

[5] A. Cimatti, A. Griggio, and R. Sebastiani. Efficient interpolant generation in satisfiability modulo a theory. In *Proceedings of the Conference on Tools and Algorithms for Construction and Analysis of Systems*, volume 4963 of *LNCS*, pages 397–412. Springer, 2008.

[6] L. de Moura and N. Bjørner. Model-based theory combination. In *Proceedings of the Workshop on Satisfiability Modulo Theories 2007*, volume 198(2) of *ENTCS*, pages 37–49. Elsevier, 2008.

[7] A. Fuchs, A. Goel, J. Grundy, S. Krstić, and C. Tinelli. Ground interpolation for the theory of equality. In *Proceedings of the International Conference on Tools and Algorithms for the Construction and Analysis of Systems*, pages 413–427. Springer, 2009.

[8] A. Goel, S. Krstić, and C. Tinelli. Ground interpolation for combined theories. In *Proceedings of the Conference on Automated Deduction*, volume 5663 of *LNAI*, pages 183–198. Springer, 2009.

[9] K. Hoder, L. Kovàcs, and A. Voronkov. Interpolation and symbol elimination in Vampire. In *Proceedings of the International Joint Conference on Automated Reasoning*, volume 6173 of *LNAI*, pages 188–195, 2010.

[10] G. Huang. Constructing Craig interpolation formulas. In *Proceedings of the First Annual International Conference on Computing and Combinatorics*, pages 181–190. Springer, 1995.

[11] D. Kapur, R. Majumdar, and C.G. Zarba. Interpolation for data structures. In Premkumar Devambu, editor, *Proceedings of the ACM SIGSOFT Symposium on the Foundations of Software Engineering*. ACM Press, 2006.

[12] L. Kovàcs and A. Voronkov. Finding loop invariants for programs over arrays using a theorem prover. In *Proceedings of the Conference on Fundamental Approaches to Software Engineering*, pages 470–485. Springer, 2009.

[13] L. Kovàcs and A. Voronkov. Interpolation and symbol elimination. In *Proceedings of the Conference on Automated Deduction*, volume 5663 of *LNAI*, pages 199–213. Springer, 2009.

[14] J. Krajíček. Interpolation theorems, lower bounds for proof systems, and independence results for bounded arithmetic. *Journal of Symbolic Logic*, 62(2):457–486, 1997.

[15] D. Leinenbach and T. Santen. Verifying the Microsoft Hyper–V hypervisor with VCC. In *Proceedings of the Second World Congress on Formal Methods*, volume 5850 of *LNCS*, pages 806–809. Springer, 2009.

[16] K.L. McMillan. Interpolation and SAT-based model checking. In *Proceedings of the Conference on Computer Aided Verification*, volume 2725 of *LNCS*, pages 1–13. Springer, 2003.

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

[18] K.L. McMillan. Quantified invariant generation using an interpolating saturation prover. In *Proceedings of the Conference on Tools and Algorithms for Construction and Analysis of Systems*, volume 4963 of *LNCS*, pages 413–427. Springer, 2008.

[19] K.L. McMillan. Lazy annotation for program testing and verification. In *Proceedings of the Conference on Computer Aided Verification*, volume 6174 of *LNCS*, pages 104–118. Springer, 2010.

[20] P. Pudlàk. Lower bounds for resolution and cutting plane proofs and monotone computations. *Journal of Symbolic Logic*, 62(3):981–998, 1997.

[21] G. Yorsh and M. Musuvathi. A combination method for generating interpolants. In *Proceedings of the Conference on Automated Deduction*, volume 3632 of *LNAI*, pages 353–368, 2005. Early version in MSR-TR-2004-108, October 2004.