Journal of Automated Reasoning article No.
DOI 10.1007/s10817-015-9325-5 in 54(4):353–390, April 2015

Interpolation Systems for Ground Proofs in Automated Deduction: a Survey

Maria Paola Bonacina · **Moa Johansson**

Received: 27 March 2014 / Accepted: 6 March 2015 / Published online: 20 March 2015

Abstract Interpolation is a deductive technique applied in program analysis and verification: for example, it is used to compute over-approximations of images or refine abstractions. An *interpolation system* takes a refutation and extracts an interpolant by building it inductively from *partial interpolants*. We survey color-based interpolation systems for ground proofs produced by key inference engines of state-of-the-art solvers: DPLL for propositional logic, *equality sharing* for combination of convex theories, and DPLL($\mathscr{T}$) for SMT-solving. Since color-based interpolation systems use colors to track symbols in proofs, equality is problematic, because replacement of equals by equals mixes symbols and therefore colors. We analyze interpolation in the presence of equality, and we demonstrate the color-based approach by giving a complete interpolation system for ground proofs by superposition.

Keywords Interpolation systems · Satisfiability modulo theories · Decision procedures · Theory combination

1 Introduction

1.1 Motivation and Aim

Automated reasoners play a major rôle in supporting tools for program analysis and verification, as described for instance in [81,32,8,33]. A theorem-proving technique applied in this context is *interpolation*, or the operation of extracting interpolants

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

Maria Paola Bonacina
Dipartimento di Informatica, Università degli Studi di Verona, Strada Le Grazie 15, I-37134 Verona, Italy
E-mail: mariapaola.bonacina@univr.it

Moa Johansson
Department of Computer Science, Chalmers University of Technology, Göteborg, Sweden
E-mail: moa.johansson@chalmers.se

from proofs. Given closed formulæ A and B such that A implies B, an *interpolant* of A and B is a closed formula that is implied by A, implies B, and contains only symbols they share; if A implies $\neg B$, A and B are inconsistent, and an interpolant of A and $\neg B$ is a *reverse interpolant* of A and B. If A and B are written in the language of a theory, so is the interpolant, and implication is implication in the theory.

The first motivation to study interpolation was to allow software model checking to benefit from theorem proving by *abstraction refinement* (e.g., [64,48,66,52,2]). A model checker is applied to determine whether an abstraction of a concrete program P satisfies a property: if it does, so does P; otherwise, the model checker produces a counter-example, representing an execution trace leading to an error state, and a formula, which is satisfiable if and only if the counter-example applies also to P. If the formula is found unsatisfiable by a theorem prover, the counter-example is spurious, and the abstraction should be refined to exclude it. Interpolants of the refutation may capture intermediate states in the spurious error trace, and can be used to refine the abstraction, by re-introducing, for instance, predicate symbols occurring in interpolants, to exclude states that lead to the spurious error.

Interpolation for abstraction refinement was proposed first for propositional logic and propositional satisfiability (SAT) (e.g., [64,52,37,86]), and then for quantifier-free fragments of first-order theories, their combinations (e.g., [88,45,21,23]), and satisfiability modulo theories (SMT) (e.g., [48,53,24,86]). Considered theories include equality [65,58,26,39], linear rational arithmetic [65,26], linear integer arithmetic [17,19], or fragments thereof [26], arrays [18,22,2], and bitvectors [86,46]. In these approaches, the theory reasoning is done either by specialized inference systems [48,65,17–19] or by satisfiability procedures [86,46,22,39,2], built into the DPLL($\mathscr{T}$) framework for the "lazy" approach to SMT (e.g., [88,45,26]). In the "eager" approach, the first-order theory is encoded into propositional logic. Although the "lazy" approach is generally preferred, if the application suggests to be "eager," one may remedy the loss of proof structure caused by the encoding, by lifting the propositional resolution proof produced by the SAT-solver to a proof in the first-order theory, and interpolate the latter [58].

In all these contexts, the formulae A and B to be interpolated and the proofs are *ground*, because the fragments are quantifier-free and the theory axioms are built-in. This does not imply that interpolants are quantifier-free. A theory is quantifier-free interpolating if quantifier-free input formulæ are guaranteed to have quantifier-free interpolants. Given a ground proof, it is obviously desirable to extract a quantifier-free interpolant, and therefore this end of the field is mostly interested in quantifier-free interpolation. Sufficient and necessary conditions to ensure that a union of quantifier-free interpolating theories is quantifier-free interpolating were given in [21,23].

In this article, we study interpolation of *ground proofs*, referring to [15] for the general case. We consider some of the main inference engines at the heart of contemporary solvers and provers: DPLL, equality sharing, DPLL($\mathscr{T}$), and superposition. DPLL stands for the Davis-Putnam-Logemann-Loveland procedure for propositional satisfiability (SAT) originated in [30,29]: the version assumed here is the one with *Conflict-Driven Clause Learning* (DPLL-CDCL) [62,71,61], which is the basis of contemporary SAT-solvers. *Equality sharing* is the original name (cf. pages 383–386 in [35]) of the *Nelson-Oppen method* [75] to combine satisfiability procedures for

theories $\mathscr{T}_1, \ldots, \mathscr{T}_n$ to get a satisfiability procedure for their union $\mathscr{T} = \bigcup_{i=1}^{n} \mathscr{T}_i$ (cf. Chapter 10 of [16] for a textbook presentation). DPLL($\mathscr{T}$) is DPLL with a built-in procedure for satisfiability in theory $\mathscr{T}$ [76], typically obtained by equality sharing. DPLL($\mathscr{T}$) and equality sharing are the foundations of SMT-solvers. Superposition-based inference systems are used in theorem provers for first-order logic with equality, and also yield decision procedures for theories relevant to program verification (e.g., [4,3,10,11]).

1.2 State of the Art

Given a refutation, an *interpolation system* works by associating a *partial interpolant* to every clause, in such a way that the partial interpolant of the empty clause is a reverse interpolant of the input formulæ. We call this approach *inductive*, because an interpolation system is defined by defining how it builds the partial interpolant of the conclusion from those of the premises, for all inference rules. We deem an interpolation system *complete* for an inference system, if for all its refutations it extracts a reverse interpolant. Since it does not require the generation of all possible interpolants of a refutation, this notion of completeness lets an inference system have different complete interpolation systems generating different interpolants.

Complete interpolation systems for propositional resolution, and therefore for DPLL, which produces proofs by propositional resolution, were given in [51,56,77, 64,88] and surveyed in [37]. Since [88], an interpolation system for DPLL($\mathscr{T}$) is obtained by uniting interpolation systems for DPLL and equality sharing. A complete interpolation system for equality sharing was given in [88] for convex theories, assuming that the $\mathscr{T}_i$-satisfiability procedures produce proofs and interpolants. An interpolation algorithm for combinations of theories, that goes beyond the convex case, was obtained in [21,23] by a different method. Interpolation of refutations produced by a DPLL($\mathscr{T}$)-based SMT-solver from inputs including also non-ground clauses was investigated in [25,69]. The non-ground input clauses are instantiated upfront by an instantiation procedure (e.g., [35,31,42,70,43]) invoked by the SMT-solver, so that the refutations to be interpolated are ground. Interpolation for ground superposition was studied in [55] continuing work in [67].

1.3 Overview of Contributions

In this article we survey what we call the *color-based approach* to the interpolation of ground proofs: the interpolation system tracks non-shared symbols, called *local* (e.g., A-local and B-local) or *colored* (e.g., A-colored and B-colored), to exclude them from the interpolant, and determine which literals descend from A or B, in order to ensure that the reverse interpolant is entailed by A and inconsistent with B. We begin with color-based interpolation systems for propositional resolution. Intuitively, the color-based approach requires that A-colored and B-colored symbols do not mix. At the propositional level this is obvious, since no new literals are generated. Ground proofs in first-order logic are like propositional proofs in this regard.

The situation changes as soon as we add equality, already for ground proofs: regardless of whether equality reasoning is done by congruence closure or superposition, equalities where one side is A-colored and the other B-colored are problematic. We clarify why this is the case, and we explain how the restriction to *equality-interpolating convex* theories in [88], and the assumption of a *separating ordering* for ground superposition in [67,55], prevent precisely such *AB-mixed* equalities. We relate these two notions, by giving a superposition-based proof that the quantifier-free fragment of the theory of equality is equality-interpolating. We cover color-based interpolation systems for equality sharing in the case of convex theories, and DPLL($\mathscr{T}$). All papers on color-based interpolation in DPLL($\mathscr{T}$) that we are aware of refer to [88] for the proofs of completeness of the interpolation systems for DPLL, equality sharing, and DPLL($\mathscr{T}$). Since the proofs appeared in [87], and with discrepancies in definitions and notations between [88] and [87], we reconstruct them here.

Proofs without inferences that mix A-colored and B-colored symbols were called *local* (e.g., [67]) — later *colored* — and ground refutations by superposition are colored under a separating ordering [55]. We give a new complete color-based interpolation system for ground refutations by a standard inference system Γ for first-order logic with equality, based on resolution and paramodulation/superposition. In summary, our contributions include:

- A unified framework of definitions for the inductive approach to interpolation;
- A survey of interpolation systems for DPLL, equality sharing and DPLL($\mathscr{T}$); and
- An analysis of interpolation in the presence of equality, resulting in a complete interpolation system for ground superposition.

This article is organized as follows. Section 2 introduces basic concepts and notations. Section 3 surveys interpolation for propositional proofs. Section 4 analyzes interpolation and equality. Section 5 covers interpolation for equality sharing and DPLL($\mathscr{T}$). Section 6 contains the interpolation system for ground superposition. Section 7 closes the article with a discussion summarizing related work and contributions. The contents of Sections 3, 4 and 5 appeared in preliminary form in [13] and those of Section 6 in [14].

2 Background

We assume the basic definitions and notations commonly used in theorem proving, such as $\bot$ for $false$, $\top$ for $true$, $\Box$ for the empty clause, $\models$ for logical consequence from formulæ or truth in a model, and $\vdash$ for derivability, where $\vdash$ without subscript means generic, sound and complete derivability in the logic, while $\vdash$ with subscript (e.g., $\vdash_\Gamma$) will be used for concrete derivation in a specific inference system. Equality is denoted by the symbol $\simeq$, which is symmetric, the symbol $\bowtie$ stands for either $\simeq$ or $\not\simeq$, and $=$ is reserved for identity. We typically use a, b, c for constant symbols, f, g for function symbols, l, r, s, t for terms, l, m, r for literals, and C, D for clauses. A clause C is a disjunction of literals $l_1 \vee \ldots \vee l_n$, and its negation $\neg C$ is the conjunction $\neg l_1 \wedge \ldots \wedge \neg l_n$. Both C and $\neg C$ can be seen as sets of literals. We use "symbol" to mean a constant, function or predicate symbol, that is, a non-variable symbol, and

"variable" for variable symbol. A term s is a subterm of a term, or literal, l, if s appears in l; s is a proper subterm, if it is not l itself. The notation $l[s]$ represents a term, or literal, where s occurs as subterm; in this notation l is called *context*.

2.1 Preliminaries on Interpolation

Given formula A, let Σ_A be the signature of symbols occurring in A. Let A and B be two formulæ to be interpolated, such that $\Sigma_A \not\subseteq \Sigma_B$ and $\Sigma_B \not\subseteq \Sigma_A$, so that their intersection $\Sigma_{A,B} = \Sigma_A \cap \Sigma_B$ is not trivial.

Definition 1 (Interpolant) 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) all symbols in I are in $\Sigma_{A,B}$.

If $\Sigma_A \subseteq \Sigma_B$, then $\Sigma_{A,B} = \Sigma_A$ and A itself would be an interpolant. Symmetrically, if $\Sigma_B \subseteq \Sigma_A$, then $\Sigma_{A,B} = \Sigma_B$ and B itself would be an interpolant. The above assumption on signatures excludes these trivial cases.

The following fundamental result, known as Craig's Interpolation Lemma [28], and presented for instance in [82,38], applies to closed formulæ, that is, formulæ where all variables are quantified:

Theorem 1 *If A and B are closed formulæ such that $A \vdash B$, and $\Sigma_{A,B}$ contains at least one predicate symbol, then an interpolant I of A and B exists, and it is also a closed formula. If $\Sigma_{A,B}$ contains no predicate symbol, then either B is valid (and the interpolant is $\top$), or A is unsatisfiable (and the interpolant is $\bot$).*

From now on we consider only closed formulæ or sentences. Since automated reasoners work refutationally, it is useful to adopt the notion of *reverse interpolant*, thus named in [55]:

Definition 2 (Reverse interpolant) 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) all symbols in I are in $\Sigma_{A,B}$.

It is simple to see that a reverse interpolant of (A,B) is an interpolant of $(A,\neg B)$, and that if I is a reverse interpolant of (A,B), then $\neg I$ is a reverse interpolant of (B,A). Swapping A and B may be relevant to applications, where it means going backward rather than forward, or vice versa, on a given path in a program or model.

A *theory* is presented by a set $\mathscr{T}$ of sentences, meaning that the theory is the set of all logical consequences of $\mathscr{T}$. It is customary to call $\mathscr{T}$ itself a theory. Let $\Sigma_{\mathscr{T}}$ be the signature of $\mathscr{T}$. Its symbols are called *defined*, because they are defined by the axioms in $\mathscr{T}$, or *interpreted*, because they are interpreted in the models of $\mathscr{T}$. The other symbols are called *free* or *uninterpreted*. Interpreted symbols are allowed in interpolants:

Definition 3 (Theory interpolant) A formula I is a *theory interpolant* of formulæ A and B such that $A \vdash_{\mathscr{T}} B$, if (i) $A \vdash_{\mathscr{T}} I$, (ii) $I \vdash_{\mathscr{T}} B$ and (iii) all uninterpreted symbols

in I are in $\Sigma_{A,B}$. 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) all uninterpreted symbols in I are in $\Sigma_{A,B}$.

Interpreted symbols may appear in A and B, so that the intersections $\Sigma_A \cap \Sigma_{\mathcal{T}}$, $\Sigma_B \cap \Sigma_{\mathcal{T}}$, and $\Sigma_{A,B} \cap \Sigma_{\mathcal{T}}$ are not empty in general. The relaxing of the third requirement of Definitions 1 and 2 in Definition 3 means that a symbol that is not shared may appear in the interpolant provided it is interpreted. Equivalently, if the interpolant contains uninterpreted symbols, then they must be shared. Since we consider refutational systems, and keeping with most of the literature, in the following we write "interpolant" for "reverse interpolant," unless the distinction is relevant, and omit "theory" whenever clear from context.

Since most reasoners transform closed formulæ into sets, or conjunctions, of clauses, from now on we assume that A and B are *disjoint* sets of clauses. This includes as special case sets of unit clauses, or, equivalently, conjunctions of literals. Since sets are understood as conjunctions, we may write $A \cup B \vdash \Box$ or, equivalently, $A \wedge B \vdash \bot$ or $A, B \vdash \bot$, depending on context.

A difficulty with interpolation is to ensure that uninterpreted symbols in interpolants are shared. The following definition, where $\setminus$ is set difference, introduces terminology that facilitates the discussion:

Definition 4 An uninterpreted symbol is *transparent*, if it is in $\Sigma_{A,B}$, *A-colored*, if it is in $\Sigma_A \setminus \Sigma_B$, and *$B$-colored*, if it is in $\Sigma_B \setminus \Sigma_A$. It is *colored*, if it is either A-colored or B-colored.

The assumption that $\Sigma_A \not\subseteq \Sigma_B$ and $\Sigma_B \not\subseteq \Sigma_A$ means that both A and B contain at least one colored symbol.

Definition 5 A term, literal, or clause is

- *Transparent*, if all its uninterpreted symbols are transparent,
- *A-colored*, if all its uninterpreted symbols are either A-colored or transparent and at least one is A-colored,
- *B-colored*, if all its uninterpreted symbols are either B-colored or transparent and at least one is B-colored, and
- *AB-mixed*, otherwise.

A term, literal, or clause is *colored*, if it is either A-colored or B-colored.

This terminology, or variants thereof, is widely adopted. Some authors use *A-local* in place of A-colored, *B-local* in place of B-colored, and *AB-common*, or *global*, or *shared*, in place of transparent. Asymmetric definitions, where A-local corresponds to A-colored, and B-local to B-colored or transparent, were given in [65]. We adopt the color-based terminology, because "local" has another (and older) meaning in automated deduction since [44,63], although in verification, "local" and "global" may be connected to the scope of program variables. We hope that this survey will contribute to establish a standard, but readers should always check the meaning of terms in a paper. Following [45] we also use:

Definition 6 A literal is *colorable* if it is not AB-mixed, or, equivalently, if it is either A-colored or B-colored or transparent. A clause is *colorable* if all its literals are.

Colorable is more general than colored: a colorable clause may have both A-colored and B-colored literals, whereas a colored clause cannot.

In the inductive approach to interpolation, interpolants are built by structural induction on a refutation of $A \cup B$. The intermediate interpolants during the construction are called *partial interpolants*. The definition of partial interpolant requires that of *projection*. Thus, we begin by defining projections, for disjunctions of literals, because we work with clauses, and conjunctions of literals, that arise when negating clauses:

Definition 7 (Projection) Let C be a disjunction (conjunction) of literals and let Σ_X stand for either Σ_A, Σ_B or $\Sigma_{A,B}$. The *projection* of C on signature Σ_X, denoted $C|_X$, is the disjunction (conjunction) of literals of C whose uninterpreted symbols are all in Σ_X. By convention, 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 $C|_X = \top$.

Assume that C and D are disjunctions or conjunctions of literals. Definition 7 implies that $(\neg C)|_X = \neg(C|_X)$, and we generalize it slightly by stipulating that $(C \vee D)|_X = C|_X \vee D|_X$ and $(C \wedge D)|_X = C|_X \wedge D|_X$. Transparent literals of C belong to both $C|_A$ and $C|_B$, while AB-mixed literals belong to neither. If C is a conjunction, $C|_A \Rightarrow C|_{A,B}$ and $C|_B \Rightarrow C|_{A,B}$; if it is a disjunction, $C|_{A,B} \Rightarrow C|_A$ and $C|_{A,B} \Rightarrow C|_B$. Clause C is colorable if and only if $C = C|_A \vee C|_B$. Alternatively, transparent literals may be put only in the projection on Σ_B as in [64,65]:

Definition 8 (Asymmetric projection) Let C be a disjunction (conjunction) of literals. The *asymmetric projections* of C are $C\backslash_B = C|_A \setminus C|_{A,B}$ and $C\downarrow_B = C|_B$.

Starting with [64], a partial interpolant is an interpolant *relative to a clause* in a refutation, so that a partial interpolant of the empty clause will be an interpolant:

Definition 9 (Partial interpolant) 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 $g_B(C) = B \wedge \neg(C|_B)$.

Indeed, $PI(\Box)$ is an interpolant of (A,B). If C occurs in a refutation of $A \cup B$, it means that $A \wedge B \vdash C$, or $A \wedge \neg C \vdash \neg B \vee C$. Thus, one could seek an interpolant of $A \wedge \neg C$ and $\neg B \vee C$, or, equivalently, a reverse interpolant of $A \wedge \neg C$ and $B \wedge \neg C$. However, the signatures of $A \wedge \neg C$ and $B \wedge \neg C$ are not necessarily Σ_A and Σ_B, unless C is transparent. Thus, the definition of partial interpolant uses projections and takes $g_A(C)$ and $g_B(C)$, whose signatures are Σ_A and Σ_B, so that to be transparent with respect to $g_A(C)$ and $g_B(C)$ is the same as to be transparent with respect to A and B.

Proposition 1 *For all clauses C occurring in a refutation of $A \cup B$, the partial interpolant $PI(C)$ has the following properties:*

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

3. $PI(C)$ is transparent.

Proof: It follows from Definitions 2 and 9. □

For ease of reference in definitions of interpolation systems, we will write $c\colon C$ to say that c is the identifier of clause C, and then we may use $c|_X$, $PI(c)$, $g_A(c)$ and $g_B(c)$.

An *interpolation system* takes a refutation of $A \cup B$, attaches a partial interpolant to every clause, and returns the partial interpolant of the empty clause as the interpolant of (A,B). In order to define an interpolation system, one has to define its partial interpolants. Since each clause in a refutation is generated by some inference rule, the definition of an interpolation system needs to cover all inference rules that generate clauses. The fundamental property of an interpolation system is *completeness*, that we define with respect to an inference system or a transition system, because in the sequel we consider both:

Definition 10 (Complete interpolation system) Given inference system Γ, or transition system $\mathscr{U}$, an interpolation system is *complete* for Γ, or $\mathscr{U}$, 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 $\mathscr{U}$, respectively, it generates an interpolant of (A,B).

In order to prove that an interpolation system is complete, it is sufficient to show that its partial interpolants satisfy Proposition 1.

2.2 Inference Systems and their Proof Trees in the Ground Case

Let Γ be a resolution and superposition based inference system. Inference systems of this kind feature *expansion inferences*, that expand the existing set by generating clauses, such as resolution and superposition, and *contraction inferences*, that contract the set by removing clauses, such as simplification and subsumption. We are interested only in inferences that appear in proofs because they generate clauses: expansion inferences, and those contraction inferences, such as simplification, that *replace* clauses by clauses; contraction inferences that merely remove clauses do not appear in proofs. We use the name *generative rules* for expansion rules and replacement rules.

Let $\succ$ be a *complete simplification ordering*, that is, a simplification ordering which is *total* on ground terms and literals. The *recursive path orderings* (RPO's) and *Knuth-Bendix orderings* (KBO's), that are commonly implemented in theorem provers, are complete simplification orderings (e.g., [34] for basic definitions about orderings). The generative rules of Γ in the ground case are in Figure 1: substitutions are not needed, and constraints of the form $\not\preceq$ take the form $\succ$, because the ordering is total on ground terms and literals. Paramodulation, superposition, and reflection build equality into the inference system; we use superposition when the literal paramodulated into is equational, and paramodulation otherwise. Duplicate literals in clauses are removed by a standard book-keeping operation known as *merging*, whose usefulness in the context of interpolation was recognized in [45].

Resolution
$$\frac{l \vee C \quad \neg l \vee D}{C \vee D} \qquad \forall m \in C\colon l \succ m;\ \forall m \in D\colon \neg l \succ m$$

Reflection
$$\frac{s \not\simeq s \vee C}{C} \qquad \forall l \in C\colon (s \not\simeq s) \not\prec l$$

Paramodulation
$$\frac{s \simeq r \vee C \quad l[s] \vee D}{C \vee l[r] \vee D} \qquad s \succ r;\ \forall m \in C\colon (s \simeq r) \succ m;\ \forall m \in D\colon l[s] \succ m$$

Superposition
$$\frac{s \simeq r \vee C \quad l[s] \bowtie t \vee D}{C \vee l[r] \bowtie t \vee D} \qquad s \succ r;\ \forall m \in C\colon (s \simeq r) \succ m;\ \forall m \in D\colon (l[s] \bowtie t) \succ m$$

where $s \simeq r$ is the *literal paramodulated/superposed from*, $l[s]$ is the *literal paramodulated/superposed into*, and the same terminology extends to clauses. Simplification is an instance of paramodulation or superposition, where the generated clause replaces the second premise, C is empty, and $s \succ r$ is the only side condition.

Fig. 1 Generative rules of Γ in the ground case

Definition 11 (Γ-derivation) Given an input set of clauses S_0, a *Γ-derivation* is a sequence $S_0 \vdash_\Gamma S_1 \vdash_\Gamma \ldots S_i \vdash_\Gamma S_{i+1} \vdash_\Gamma \ldots$, where for all $i > 0$, S_i is a set of clauses derived from S_{i-1} by a Γ-inference.

Let $Th(S) = \{C \mid S \models C\}$: inferences are *sound* ($S_i \vdash_\Gamma S_{i+1}$ implies $S_{i+1} \subseteq Th(S_i)$) and *adequate* ($S_i \vdash_\Gamma S_{i+1}$ implies $S_i \subseteq Th(S_{i+1})$). $S^* = \bigcup_{i \geq 0} S_i$ is the set of all generated clauses. A Γ-derivation is *successful* if $\Box \in S_k$ for some k, which reveals that the input set S_0 is inconsistent.

Upon success, the theorem prover extracts a *refutation*, or *refutational proof*, or *proof*, for short, which includes only the inferences and clauses involved in the generation of $\Box$. A proof is usually represented as a *proof tree* drawn with the root at the bottom and the leaves at the top:[1]

Definition 12 (Γ-proof tree) Given a Γ-derivation $S_0 \vdash_\Gamma S_1 \vdash_\Gamma \ldots S_i \vdash_\Gamma S_{i+1} \vdash_\Gamma \ldots$, for all $C \in S^*$, the *Γ-proof tree* $\Pi_\Gamma(C)$ of C is defined as follows:

- If $C \in S_0$, $\Pi_\Gamma(C)$ consists of a node labeled by C;
- If C is inferred by a generative Γ-inference from premises $C_1, \ldots, C_k$, $\Pi_\Gamma(C)$ consists of a node labeled by C with k subtrees $\Pi_\Gamma(C_1), \ldots, \Pi_\Gamma(C_k)$.

If the derivation is successful, $\Box \in S^*$ and $\Pi_\Gamma(\Box)$ is a Γ-refutation.

If a proof is made only of equations, it can be also represented as a *chain* of equational replacement steps. A simplification step where $s \simeq r$ simplifies $t[s]$ to $t[r]$ is represented by the proof chain $t[s] \rightarrow_{s \simeq r} t[r]$. The subterm s is called a *redex*, and a term is in *normal form* if it has no redex. A superposition step of $s \simeq r$ into $l[s] \simeq t$ is represented by the proof chain $t \leftarrow_{l[s] \simeq t} l[s] \rightarrow_{s \simeq r} l[r]$. It is well known from the theory

[1] In general, it is a rooted graph, called *ancestor-graph* [12], but it can be put in the form of a tree by allowing different vertices to have the same clause as label.

of completion that a ground equational proof $s \overset{*}{\leftrightarrow} t$ can be reduced to a *rewrite proof*, or *valley proof*, that is a chain in the form $s \overset{*}{\rightarrow} r \overset{*}{\leftarrow} t$ for some term r. The interested reader can find in [9] a recent treatment, historical background, and references for the theory of completion.

2.3 Equality Sharing and its Proof Trees

The Nelson-Oppen scheme is a standard method to combine theories. Its original name is *equality sharing* [73,74,35]. We summarize the relevant elements for what follows, referring the interested reader to (e.g., [73,75,74,35,16]) for a complete presentation and more references. Let $\mathscr{T}$ be a union $\bigcup_{i=1}^{n} \mathscr{T}_i$ of quantifier-free fragments of first-order theories, where each $\mathscr{T}_i$ is assumed to be equipped with a $\mathscr{T}_i$-satisfiability procedure, that we name Q_i, which decides whether a set of ground $\mathscr{T}_i$-literals has a $\mathscr{T}_i$-model. Equality sharing is not concerned with the inner working of these procedures; it combines them to yield a procedure that decides whether a set S of ground $\mathscr{T}$-literals has a $\mathscr{T}$-model, under a few requirements. First, the $\mathscr{T}_i$'s are required to be *disjoint*: their signatures share only uninterpreted constants and equality. Second, they are required to be *stably infinite*:

Definition 13 A theory $\mathscr{T}$ is *stably infinite*, if a quantifier-free $\mathscr{T}$-formula is $\mathscr{T}$-satisfiable if and only if it has a $\mathscr{T}$-model with domain of infinite cardinality.

For instance, the theory of equality, linear arithmetic, and theories of data structures such as lists and arrays, are stably infinite (e.g., [16]). The first phase of equality sharing, called *separation*, transforms S into a collection $S_1, \ldots, S_n$, where S_i, for $1 \leq i \leq n$, is a set of ground $\mathscr{T}_i$-literals. Function symbols from signatures of different theories that occur mixed in the terms are separated by introducing new uninterpreted constant symbols. For example, $f(g(a)) \simeq b$, where f and g belong to the signatures of different theories, becomes $f(c) \simeq b \wedge g(a) \simeq c$, where c is a new constant. Separation ensures that each Q_i deals only with $\mathscr{T}_i$-literals. Since only new constants are introduced, all literals in $\bigcup_{i=1}^{n} S_i$ are ground, and S and $\bigcup_{i=1}^{n} S_i$ are $\mathscr{T}$-equisatisfiable.

In the second phase of equality sharing, each Q_i, for $1 \leq i \leq n$, propagates to all other Q_j's, for $1 \leq j \neq i \leq n$, the *disjunctions of equalities* between shared constants, that it $\mathscr{T}_i$-deduces from its set of $\mathscr{T}_i$-literals. Completeness requires that each Q_i deduces and propagates *all* such disjunctions that are $\mathscr{T}_i$-entailed by the set of $\mathscr{T}_i$-literals that Q_i works with. If the theories are convex, it is sufficient to propagate equalities between shared constants:

Definition 14 A theory $\mathscr{T}$ is *convex*, if whenever $H \models_{\mathscr{T}} \bigvee_{k=1}^{m} s_k \simeq t_k$, then $H \models_{\mathscr{T}} s_j \simeq t_j$, for some j, $1 \leq j \leq m$, where H is a conjunction of literals.

It was proved in [6] that every first-order theory that is convex and has no trivial models is stably-infinite.

For interpolation, we are interested in defining the *proof tree* generated by equality sharing. For uniformity with the other inference or transition systems considered in this article, we view literals as unit clauses, and use $\Box$ for the contradiction generated by the procedure: in equality sharing the contradiction is detected by one of

the Q_i's. We use K for the set of propagated equalities, we stipulate that also $\Box$ gets propagated, and we denote with $\vdash_{\mathscr{T}_i}$ a $\mathscr{T}_i$-deduction by Q_i. Since equality sharing treats each Q_i as a black box, a deduction in Q_i is viewed like a single inference by equality sharing. For the same reason, equality sharing is not concerned with all the literals that a Q_i generates, but only with those that it propagates, namely equalities between shared constants and $\Box$:

Definition 15 (ES-derivation) Given a union $\mathscr{T} = \bigcup_{i=1}^{n} \mathscr{T}_i$ of quantifier-free fragments of disjoint convex first-order theories, with $\mathscr{T}_i$-satisfiability procedures Q_i, $1 \leq i \leq n$, and an input set of ground $\mathscr{T}$-literals (unit clauses) S, a *derivation by equality sharing*, or an *ES-derivation* for short, is a sequence

$$(S_1,\ldots,S_n,K_0) \underset{ES}{\vdash} (S_1,\ldots,S_n,K_1) \underset{ES}{\vdash} \ldots (S_1,\ldots,S_n,K_j) \underset{ES}{\vdash} (S_1,\ldots,S_n,K_{j+1}) \underset{ES}{\vdash} \ldots$$

where $S_1,\ldots,S_n$ is the separation of S, $K_0 = \emptyset$, and for all $j > 0$, $K_j = K_{j-1} \cup \{C\}$, where C is either $\Box$ or an equality between shared constants, $C \notin K_{j-1}$, and $S_i \cup K_{j-1} \vdash_{\mathscr{T}_i} C$, for some i, $1 \leq i \leq n$.

Since equality sharing is a decision procedure, an ES-derivation is guaranteed to halt, and if it generates $\Box$, a proof can be extracted:

Definition 16 (ES-proof tree) Given an ES-derivation from input set S, which halts at stage h, for all $C \in \bigcup_{i=1}^{n} S_i \cup K_h$, the *ES-proof tree* $\Pi_{ES}(C)$ of C is defined as follows:

- If $C \in \bigcup_{i=1}^{n} S_i$, $\Pi_{ES}(C)$ consists of a node labeled by C;
- If $\{C_1,\ldots,C_k\} \vdash_{\mathscr{T}_i} C$, for some i, $1 \leq i \leq n$, $\Pi_{ES}(C)$ consists of a node labeled by C with k subtrees $\Pi_{ES}(C_1),\ldots,\Pi_{ES}(C_k)$.

If a contradiction is found, $\Box \in K_h$ and $\Pi_{ES}(\Box)$ is an ES-refutation.

We gave the definitions of derivation and proof tree under the hypothesis that all combined theories are convex, because in the rest of this article we will be primarily concerned with this case.

2.4 Transition Systems and their Proof Trees

While *inference system* refers to a set of non-deterministic inference rules that yield a proof procedure once coupled with a search plan, *transition system* is used for algorithmic engines such as DPLL-CDCL and DPLL($\mathscr{T}$), where inference and control are separated only to a lesser degree. These transition systems operate in two modes: *search mode* and *conflict resolution mode*. In search mode, the state of the system has the form $M \,\|\, F$, where M is a sequence of *assigned literals*, and F is a set of clauses. Intuitively, M represents a partial assignment to literals, possibly with a justification, and therefore it represents 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. 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. No assigned literal

occurs twice in M nor does it occur negated in M. If neither l nor $\neg l$ appears in M, then l is said to be *undefined*. The set of assigned literals in M is denoted $lits(M)$.

In conflict resolution mode, the state has the form $M \parallel F \parallel C$, where C is a clause in F whose literals are all false in M. Such a clause is *in conflict* and is called *conflict clause*. We could state that C is in conflict by writing $M \models \neg C$. In DPLL($\mathscr{T}$), the DPLL engine accepts only propositional clauses, whereas the Q_i's accept ground first-order literals. To bridge this gap, an *abstraction function* α maps first-order ground atoms to propositional atoms, and then straightforwardly first-order ground clauses to propositional clauses (e.g., [5]). Thus, it is customary to write $M \models_P \neg C$, read M "propositionally satisfies" $\neg C$, to mean $M \models \neg\alpha(C)$.

Definition 17 (Transition system derivation) Given a transition system $\mathscr{U}$ and an input set of clauses F_0, a *transition system derivation*, or *$\mathscr{U}$-derivation*, is a sequence

$$\Delta_0 \underset{\mathscr{U}}{\Longrightarrow} \Delta_1 \underset{\mathscr{U}}{\Longrightarrow} \dots \Delta_i \underset{\mathscr{U}}{\Longrightarrow} \Delta_{i+1} \underset{\mathscr{U}}{\Longrightarrow} \dots$$

where $\Delta_0 = \parallel F_0$, and $\forall i > 0$, Δ_i is of the form $M_i \parallel F_i$ or $M_i \parallel F_i \parallel C_i$, and is produced from Δ_{i-1} by a transition rule in $\mathscr{U}$.

A transition system derivation is characterized by the sets $F^* = \bigcup_{i \geq 0} F_i$ of all generated clauses and $C^* = \{C_i | i > 0\}$ of all conflict clauses. If all conflict clauses were learnt, which is not the case in practice, then $C^* \subseteq F^*$. In DPLL-CDCL and DPLL($\mathscr{T}$) learning a conflict clause is the only way to generate a new clause. In the sequel, we use $\mathscr{U}_1$ for DPLL-CDCL and $\mathscr{U}_2$ for DPLL($\mathscr{T}$) as subscripts.

The transition rules of DPLL-CDCL are in Figure 2. Rules Decide, UnitPropagate and Conflict apply in search mode. Decide guesses the truth value of a literal that occurs in F; UnitPropagate propagates the implications of guesses; Conflict detects the presence of a conflict clause and puts the system in conflict resolution mode. Rules Explain, Learn, Backjump, and Unsat apply in conflict resolution mode. Explain resolves a literal, say $\neg l$, in a conflict clause, with its complement l in the clause that is the justification of l in M; the resolvent is also a conflict clause. Learn adds to F a clause derived by Explain, because it is a logical consequence of the original set of clauses. Backjump unassigns at least one decided literal, named l' in the rule definition, and drives the system back from conflict resolution mode to search mode. A typical choice is that l' be the least recently decided literal that satisfies the conditions of the rule. There is always a way to come back to search mode from conflict resolution mode, unless the conflict clause is empty, in which case the Unsat rule concludes that the input is unsatisfiable.

A refutation by DPLL-CDCL is a refutation by propositional resolution, composed of the resolution steps performed by Explain, as noticed first in [89], according to [81]. The clauses that appear in the refutation are input clauses and conflict clauses:

Definition 18 (DPLL-proof tree) Given a DPLL-derivation,

$$\Delta_0 \underset{\mathscr{U}_1}{\Longrightarrow} \Delta_1 \underset{\mathscr{U}_1}{\Longrightarrow} \dots \Delta_i \underset{\mathscr{U}_1}{\Longrightarrow} \Delta_{i+1} \underset{\mathscr{U}_1}{\Longrightarrow} \dots,$$

for all $C \in F_0 \cup C^*$ the *DPLL-proof tree* $\Pi_{\mathscr{U}_1}(C)$ of C is defined as follows:

Decide

$$M \,\|\, F \Longrightarrow M\, l \,\|\, F \qquad \textbf{if} \begin{cases} l \text{ or } \neg l \text{ occurs in } F, \\ l \text{ is undefined in } M; \end{cases}$$

UnitPropagate

$$M \,\|\, F, C \vee l \Longrightarrow M\, l_{C \vee l} \,\|\, F, C \vee l \qquad \textbf{if} \begin{cases} M \models_P \neg C, \\ l \text{ is undefined in } M; \end{cases}$$

Conflict

$$M \,\|\, F, C \Longrightarrow M \,\|\, F, C \,\|\, C \qquad \textbf{if } M \models_P \neg C$$

Explain

$$M \,\|\, F \,\|\, C \vee \neg l \Longrightarrow M \,\|\, F \,\|\, D \vee C \qquad \textbf{if } l_{D \vee l} \in M$$

Learn

$$M \,\|\, F \,\|\, C \Longrightarrow M \,\|\, F, C \,\|\, C \qquad \textbf{if } C \notin F$$

Backjump

$$M\, l'\, M' \,\|\, F \,\|\, C \vee l \Longrightarrow M\, l_{C \vee l} \,\|\, F \qquad \textbf{if} \begin{cases} M \models_P \neg C, \\ l \text{ is undefined in } M \end{cases}$$

Unsat

$$M \,\|\, F \,\|\, \Box \Longrightarrow \mathit{unsat}$$

Fig. 2 Transition rules of DPLL-CDCL

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

If the derivation terminates in state *unsat*, $\Box \in C^*$ and $\Pi_{\mathscr{U}_1}(\Box)$ is a DPLL-refutation.

A justification C_2 is either an input clause or a learnt clause, which was once a conflict clause, and therefore $\Pi_{\mathscr{U}_1}(C_2)$ is defined. An example illustrates proof generation:

Example 1 Assume $F = \{p \vee q, \neg p \vee q, \neg q \vee p, \neg p \vee \neg q\}$. The derivation starts with a Decide step, followed by a UnitPropagate step, which leads to a conflict:

$$\emptyset \,\|\, F \Longrightarrow p \,\|\, F \Longrightarrow p, q_{\neg p \vee q} \,\|\, F \Longrightarrow p, q_{\neg p \vee q} \,\|\, F \,\|\, \neg p \vee \neg q.$$

An Explain step performs the resolution between conflict clause $\neg p \vee \neg q$ and justification $\neg p \vee q$, the resolvent $\neg p$ is learnt, and Backjump applies with M empty, $l' = p$, $M' = \neg q_{\neg p \vee q}$, C empty, and $l = \neg p$:

$$p, q_{\neg p \vee q} \,\|\, F \,\|\, \neg p \vee \neg q \Longrightarrow p, \neg q_{\neg p \vee q} \,\|\, F \,\|\, \neg p \Longrightarrow p, \neg q_{\neg p \vee q} \,\|\, F, \neg p \,\|\, \neg p$$
$$\Longrightarrow \neg p_{\neg p} \,\|\, F, \neg p,$$

where $\Pi_{\mathscr{U}_1}(\neg p)$ is given by the resolution step between $\neg p \vee \neg q$ and $\neg p \vee q$. The search restarts with a UnitPropagate step, which leads to another conflict:

$$\neg p_{\neg p} \,\|\, F, \neg p \Longrightarrow \neg p_{\neg p}, q_{p \vee q} \,\|\, F, \neg p \Longrightarrow \neg p_{\neg p}, q_{p \vee q} \,\|\, F, \neg p \,\|\, \neg q \vee p.$$

T-Propagate

$$M \,\|\, F \Longrightarrow M\; l_{(\neg l_1 \vee \ldots \vee \neg l_n \vee l)} \,\|\, F \qquad \textbf{if} \begin{cases} l \text{ occurs in } F, \\ l \text{ is undefined in } M, \\ l_1, \ldots, l_n \in \mathit{lits}(M), \\ l_1, \ldots, l_n \models_{\mathscr{T}} l; \end{cases}$$

T-Conflict

$$M \,\|\, F \Longrightarrow M \,\|\, F \,\|\, \neg l_1 \vee \ldots \vee \neg l_n \qquad \textbf{if} \begin{cases} l_1, \ldots, l_n \in \mathit{lits}(M), \\ l_1, \ldots, l_n \models_{\mathscr{T}} \bot\,. \end{cases}$$

Fig. 3 Additional transition rules for DPLL($\mathscr{T}$)

An Explain step performs the resolution between conflict clause $\neg q \vee p$ and justification $p \vee q$, and the resolvent p is learnt:

$$\neg p_{\neg p}, q_{p \vee q} \,\|\, F, \neg p \,\|\, \neg q \vee p \Longrightarrow \neg p_{\neg p}, q_{p \vee q} \,\|\, F, \neg p \,\|\, p \Longrightarrow \neg p_{\neg p}, q_{p \vee q} \,\|\, F, \neg p, p \,\|\, p,$$

where $\Pi_{\mathscr{U}_1}(p)$ is given by the resolution step between $\neg q \vee p$ and $p \vee q$. Another Explain step resolves conflict clause p and justification $\neg p$:

$$\neg p_{\neg p}, q_{p \vee q} \,\|\, F, \neg p, p \,\|\, p \Longrightarrow \neg p_{\neg p}, q_{p \vee q} \,\|\, F, \neg p, p \,\|\, \Box \Longrightarrow \mathit{unsat}.$$

The refutation $\Pi_{\mathscr{U}_1}(\Box)$ is given by the resolution step between p and $\neg p$, with $\Pi_{\mathscr{U}_1}(p)$ and $\Pi_{\mathscr{U}_1}(\neg p)$ as subtrees.

Note that a derivation that simulates truth tables by guessing assignments and backtracking chronologically, without generating a proof by resolution, is not allowed by these transition rules, because Backjump is driven by the conflict clause according to its conditions, and therefore needs one or more Explain steps.

The addition of transition rules T-Propagate and T-Conflict, shown in Figure 3, yields DPLL($\mathscr{T}$). These two rules connect the DPLL engine with the $\mathscr{T}$-satisfiability procedure, by letting it propagate $\mathscr{T}$-consequences of M. Both T-Propagate and T-Conflict apply in search mode, and T-Conflict causes the system to switch to conflict resolution mode. T-Propagate detects that if literals $l_1, \ldots, l_n$ are true, then l must also be true in $\mathscr{T}$, or, equivalently, $\neg l_1 \vee \ldots \vee \neg l_n \vee l$ is a $\mathscr{T}$-lemma. If a disjunction $l \vee l'$ is entailed, it makes no difference, because it is the same as saying that $l_1, \ldots, l_n, \neg l'$ entail l, or $\neg l_1 \vee \ldots \vee \neg l_n \vee l \vee l'$ is a $\mathscr{T}$-lemma. T-Conflict detects that a subset $l_1, \ldots, l_n$ of M is $\mathscr{T}$-inconsistent, or, equivalently, $\neg l_1 \vee \ldots \vee \neg l_n$ is a $\mathscr{T}$-lemma, which is generated as *$\mathscr{T}$-conflict clause*. If the $\mathscr{T}$-satisfiability procedure is a combination of Q_i's by equality sharing, ever since [75], the propagation of disjunctions of equalities between shared constants, is implemented through case analysis and backtracking. In DPLL($\mathscr{T}$), the case analysis and backtracking are those of DPLL, as a disjunction coming from a theory is treated like any other clause.

In a refutation by DPLL($\mathscr{T}$), in addition to conflict clauses generated by Conflict and Explain, there may be $\mathscr{T}$-conflict clauses, and also $\mathscr{T}$-lemmas generated by T-Propagate may be involved as justifications. Since $\mathscr{T}$-conflict clauses are also $\mathscr{T}$-lemmas, we use $\mathscr{T}$-lemmas for both. Thus, we need to assume that the Q_i's and their combination produce proofs of $\mathscr{T}$-lemmas, that we denote by $\Pi_{\mathscr{T}}(C)$. The proof of the $\mathscr{T}$-unsatisfiability of $l_1, \ldots, l_n$ is a proof that $C = \neg l_1 \vee \ldots \vee \neg l_n$ is a $\mathscr{T}$-lemma.

Similarly, the proof that $l_1, \ldots, l_n$ $\mathscr{T}$-entail l is a proof that $\neg l_1 \vee \ldots \vee \neg l_n \vee l$ is a $\mathscr{T}$-lemma. Refutations by DPLL($\mathscr{T}$) are ground, but not propositional, because the inverse α^{-1} of the abstraction function is applied to restore first-order ground atoms.

Definition 19 (DPLL($\mathscr{T}$)-proof tree) Given a DPLL($\mathscr{T}$)-derivation,

$$\Delta_0 \underset{\mathscr{U}_2}{\Longrightarrow} \Delta_1 \underset{\mathscr{U}_2}{\Longrightarrow} \ldots \Delta_i \underset{\mathscr{U}_2}{\Longrightarrow} \Delta_{i+1} \underset{\mathscr{U}_2}{\Longrightarrow} \ldots,$$

for all $C \in F_0 \cup C^*$ and all C that are $\mathscr{T}$-lemmas, the *DPLL($\mathscr{T}$)-proof tree* $\Pi_{\mathscr{U}_2}(C)$ of C is defined as follows:

- If $C \in F_0$, $\Pi_{\mathscr{U}_2}(C)$ consists of a node labeled by C;
- If C is generated by resolving conflict clause C_1 with justification C_2, $\Pi_{\mathscr{U}_2}(C)$ consists of a node labeled by C with subtrees $\Pi_{\mathscr{U}_2}(C_1)$ and $\Pi_{\mathscr{U}_2}(C_2)$;
- If C is a $\mathscr{T}$-lemma, $\Pi_{\mathscr{U}_2}(C) = \Pi_{\mathscr{T}}(C)$.

If the derivation terminates in state *unsat*, $\Box \in C^*$ and $\Pi_{\mathscr{U}_2}(\Box)$ is a DPLL($\mathscr{T}$)-refutation.

Both DPLL-proof trees and DPLL($\mathscr{T}$)-proof trees are made of inferences performed in conflict resolution mode, except for the $\Pi_{\mathscr{T}}(C)$ subtrees.

3 Propositional Interpolation Systems

In this section we see the first instance of the inductive approach to interpolation by covering interpolation systems for resolution refutations in propositional logic. If the input problem is propositional, DPLL($\mathscr{T}$) solves it by using DPLL alone, and Γ proves it by resolution. Thus, the interpolation systems of this section apply to a refutation that could be produced by anyone of DPLL, DPLL($\mathscr{T}$), and Γ, if applied to a propositional problem.

Let A and B be disjoint sets of propositional clauses: a literal is either a propositional variable or its negation, and Definitions 4 and 5 apply to propositional variables, literals and clauses. In propositional logic the set of literals that may appear in a refutation is determined once and for all by the set of literals that occur in the input set $A \cup B$. Since input literals are either A-colored or B-colored or transparent, so is every literal in a proof: in other words, *there are no AB-mixed literals*. Therefore, interpolation systems build inductively the partial interpolant of every resolvent from those of its parents, distinguishing whether the literal resolved upon is A-colored or B-colored or transparent:

Definition 20 (HKPYM interpolation system) Let $c\colon C$ be a clause 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 $p_1\colon l \vee C$ and $p_2\colon \neg l \vee D$ 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 $PI(c) = (l \vee PI(p_1)) \wedge (\neg l \vee PI(p_2))$.

This interpolation system comes from [88], where it is called *Pudlàk algorithm* with [77] as source. The author of [77] refers the reader also to [56]. This interpolation system was analyzed also in [37,36,86], and called HKP in [36], from the initials of Huang [51], Krajíček [56] and Pudlàk [77], its three independent authors. We add the initials of Yorsh and Musuvathi and call it HKPYM, on the account of the proof of completeness in [87]. The system in [51] did not use colors, but a notion of literals coming from A, from B, or from both, and was for first-order resolution and paramodulation, with propositional resolution as a special case: we studied interpolation in the first-order case in [15], including a discussion of [51].

The following system was introduced in [64,65] and studied in [37,86]. We call it MM from the two M in McMillan. While HKPYM treats A and B symmetrically, *MM* is slanted towards B, by the choice of partial interpolant for input clauses in A, and by treating B-colored and transparent literals resolved upon in the same way:

Definition 21 (MM interpolation system) Let $c\colon C$ be a clause in a refutation of $A \cup B$ by propositional resolution:

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

The following example applies both systems to the same input:

Example 2 Assume $A = \{a \vee e,\ \neg a \vee b,\ \neg a \vee c\}$ and $B = \{\neg b \vee \neg c \vee d,\ \neg d,\ \neg e\}$. In the refutation, each clause is decorated with its partial interpolant surrounded by brackets. We apply first HKPYM:

– $a \vee e$ $[\bot]$ resolves with $\neg e$ $[\top]$ to yield a $[e]$: since e is transparent, the partial interpolant of the resolvent is $(e \vee \bot) \wedge (\neg e \vee \top) = e$;
– a $[e]$ resolves with $\neg a \vee c$ $[\bot]$ to yield c $[e]$: since a is A-colored, the partial interpolant of the resolvent is $e \vee \bot = e$;
– a $[e]$ resolves with $\neg a \vee b$ $[\bot]$ to yield b $[e]$: again a is A-colored, and the partial interpolant of the resolvent is $e \vee \bot = e$;
– b $[e]$ resolves with $\neg b \vee \neg c \vee d$ $[\top]$ to yield $\neg c \vee d$ $[b \vee e]$: since b is transparent, the partial interpolant of the resolvent is $(b \vee e) \wedge (\neg b \vee \top) = b \vee e$;
– c $[e]$ resolves with $\neg c \vee d$ $[b \vee e]$ to yield d $[e \vee (c \wedge b)]$: c is also transparent, and the partial interpolant of the resolvent is $(c \vee e) \wedge (\neg c \vee b \vee e) = e \vee (c \wedge b)$;
– d $[e \vee (c \wedge b)]$ resolves with $\neg d$ $[\top]$ to yield $\Box$ $[e \vee (c \wedge b)]$: as d is B-colored, the interpolant is $(e \vee (c \wedge b)) \wedge \top = e \vee (c \wedge b)$.

We apply next MM to the same refutation:

– $a \vee e$ $[e]$ resolves with $\neg e$ $[\top]$ to yield a $[e]$: since e is transparent, the partial interpolant of the resolvent is $e \wedge \top = e$;
– a $[e]$ resolves with $\neg a \vee c$ $[c]$ to yield c $[e \vee c]$: since a is A-colored, the partial interpolant of the resolvent is $e \vee c$;

- a $[e]$ resolves with $\neg a \vee b$ $[b]$ to yield b $[e \vee b]$: again a is A-colored, and the partial interpolant of the resolvent is $e \vee b$;
- b $[e \vee b]$ resolves with $\neg b \vee \neg c \vee d$ $[\top]$ to yield $\neg c \vee d$ $[e \vee b]$: as b is transparent, the partial interpolant of the resolvent is $(e \vee b) \wedge \top = e \vee b$;
- c $[e \vee c]$ resolves with $\neg c \vee d$ $[e \vee b]$ to yield d $[e \vee (c \wedge b)]$: c is also transparent, so that the partial interpolant of the resolvent is $(e \vee c) \wedge (e \vee b) = e \vee (c \wedge b)$;
- d $[e \vee (c \wedge b)]$ resolves with $\neg d$ $[\top]$ to yield $\square$ $[e \vee (c \wedge b)]$: since d is B-colored, the interpolant is $(e \vee (c \wedge b)) \wedge \top = e \vee (c \wedge b)$.

The final result is the same, but some of the intermediate partial interpolants differ: for each step of the proof, the HKPYM partial interpolant implies the MM partial interpolant, but the converse is not true. This is interesting because it was shown in [37] that final MM interpolants imply final HKPYM interpolants. In this example the HKPYM interpolant starts out stronger and sufficiently weak.

Both [64,65] for MM and [88] for HKPYM give the asymmetric definition of projection (cf. Definition 8). However the proof of completeness of HKPYM in [87] requires the symmetric definition (cf. Definition 7):

Theorem 2 (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, $PI(c)$ is an interpolant of $g_A(c) = A \wedge \neg(C|_A)$ and $g_B(c) = B \wedge \neg(C|_B)$, that is, it satifies the requirements:

1. $g_A(c) \vdash PI(c)$,
2. $g_B(c) \wedge PI(c) \vdash \bot$, and
3. $PI(c)$ is transparent.

The proof is by induction on the structure of the refutation by resolution:
Base case:

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

Inductive hypothesis: for $k \in \{1,2\}$ it holds that:

1. $g_A(p_k) \vdash PI(p_k)$,
2. $g_B(p_k) \wedge PI(p_k) \vdash \bot$,
3. $PI(p_k)$ is transparent.

Inductive 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 contrapositive of (*)
$A \wedge \neg(p_1|_A \wedge p_2|_A) = A \wedge (\neg p_1|_A \vee \neg p_2|_A) = (A \wedge \neg p_1|_A) \vee (A \wedge \neg p_2|_A) = g_A(p_1) \vee g_A(p_2)$ and $g_A(p_1) \vee g_A(p_2) \Rightarrow PI(p_1) \vee PI(p_2)$ by induction hypothesis.
We show (2) $g_B(c) \wedge PI(c) \Rightarrow \bot$:
$g_B(c) \wedge PI(c) = B \wedge \neg((C \vee D)|_B) \wedge PI(c) = B \wedge \neg(C|_B \vee D|_B) \wedge PI(c) =$
$B \wedge \neg(C|_B) \wedge \neg(D|_B) \wedge (PI(p_1) \vee PI(p_2)) \Rightarrow$
$(B \wedge \neg(C|_B) \wedge PI(p_1)) \vee (B \wedge \neg(D|_B) \wedge PI(p_2)) =$
$(B \wedge \neg((l \vee C)|_B) \wedge PI(p_1)) \vee (B \wedge \neg((\neg l \vee D)|_B) \wedge PI(p_2)) =$
(because l is A-colored 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 is B-colored 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 contrapositive of (**)
$B \wedge (\neg(p_1|_B \wedge p_2|_B)) \wedge PI(p_1) \wedge PI(p_2) = B \wedge (\neg(p_1|_B) \vee \neg(p_2|_B)) \wedge PI(p_1) \wedge PI(p_2) = [(B \wedge \neg(p_1|_B)) \vee (B \wedge \neg(p_2|_B))] \wedge PI(p_1) \wedge PI(p_2) =$
$(g_B(p_1) \wedge PI(p_1) \wedge PI(p_2)) \vee (g_B(p_2) \wedge PI(p_1) \wedge PI(p_2)) \Rightarrow$
$(g_B(p_1) \wedge PI(p_1)) \vee (g_B(p_2) \wedge PI(p_2)) \Rightarrow \bot \vee \bot = \bot$ 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, we get

$[B \wedge \neg(C|_B) \wedge \neg(D|_B) \wedge l \wedge PI(p_2)] \vee [B \wedge \neg(C|_B) \wedge \neg(D|_B) \wedge \neg l \wedge PI(p_1)] \Rightarrow$
$[B \wedge \neg(D|_B) \wedge l \wedge PI(p_2)] \vee [B \wedge \neg(C|_B) \wedge \neg l \wedge PI(p_1)] =$
$[B \wedge \neg(D|_B \vee \neg l) \wedge PI(p_2)] \vee [B \wedge \neg(C|_B \vee l) \wedge PI(p_1)] = (l$ 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) follows from the assumption that l is transparent and the induction hypothesis. □

Given an interpolation system, its *dual* [51,36], or *inverse* [37], is the interpolation system that associates to every clause C in a refutation of $A \cup B$ the partial interpolant that the original system would associate to C if A and B were exchanged. Applying this to □, the interpolant of (A,B) produced by the inverse system is the interpolant of (B,A) produced by the original system. An interpolation system is *symmetric*, if its interpolant of (A,B) is the negation of the interpolant of (A,B) produced by its inverse, or if the interpolant of (A,B) is the negation of the interpolant of (B,A).

4 Interpolation and Equality

The addition of equality, even in the ground case, changes the picture, because an AB-mixed equation $t_a \simeq t_b$, where t_a is an A-colored ground term and t_b is a B-colored ground term, may be derived. Other AB-mixed literals may appear, if occurrences of t_a in A-colored literals are replaced by t_b or vice versa. Thus, it is no longer true that literals in a refutation are either A-colored or B-colored or transparent, as assumed by the propositional interpolation systems.

Furthermore, in the propositional case, the status of literals with respect to colors is stable: if a literal is A-colored, B-colored, or transparent, in the initial state of a derivation, it will remain such throughout the derivation. In a Γ-derivation, if $t_a \simeq t_b$ is generated, t_a and t_b are in normal form with respect to the current set of equations, and $t_a \succ t_b$, simplification replaces all occurrences of t_a, including those in A, by occurrences of t_b. Thus, t_b should become transparent. In a DPLL($\mathscr{T}$)-derivation, if $t_a \simeq t_b$ is generated, 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, in order to represent both A-colored and B-colored terms. Note that if either one of the two classes already contains a transparent term t, then the equation $t_a \simeq t_b$ is harmless, because t can be the representative of the new class, containing both A-colored and B-colored terms. Regardless of whether equality reasoning is done by rewriting or congruence closure, inferences with AB-mixed equalities cause the status of terms and literals with respect to colors to be unstable.

A proof without AB-mixed equalities was termed *colorable* in [45] referring to DPLL($\mathscr{T}$). Since we do not assume that equality is the only predicate, we give the definition for literals:

Definition 22 (Colorable proof tree) A proof tree is *colorable* if all its clauses are; equivalently, a proof tree is *colorable* if it contains no AB-mixed literals.

A key property to ensure that proofs are colorable was identified in [88]:

Definition 23 (Equality-interpolating convex theory) A convex theory $\mathscr{T}$ is *equality-interpolating* if, for all interpolation problems (A,B), where there exist transparent ground terms, whenever $A \wedge B \models_{\mathscr{T}} t_a \simeq t_b$, where t_a is an A-colored ground term and t_b is a B-colored ground term, then $A \wedge B \models_{\mathscr{T}} t_a \simeq t \wedge t_b \simeq t$ for some transparent ground term t.

The term t is called *equality-interpolating term*. This definition is for convex theories, because it considers only the case where a unit equality is entailed. The theory of equality is convex. If the proof that $\mathscr{T}$ is equality-interpolating is constructive, it yields an algorithm to compute equality-interpolating terms. If an equality-interpolating term t can be computed whenever a $t_a \simeq t_b$ is generated, t can be adopted as the representative of the merged congruence class.

In a Γ-derivation, it suffices to have an ordering that ensures that simplification replaces t_a and t_b by t. An ordering that makes terms in a shared signature smaller than terms in an extended signature was introduced for *hierarchic reasoning* [41]. For interpolation, a notion of *ordering oriented for* (A,B), whereby any A-colored term is larger than any B-colored or transparent term, was introduced in [67]. The definition that we adopt here was given in [55]:

Definition 24 (Separating ordering) An ordering $\succ$ is *separating*, if for all ground terms, or literals, l and r, $l \succ r$ whenever r is transparent and l is not.

For a recursive path ordering (RPO), which is defined based on a *precedence* on function and predicate symbols, it is sufficient to assume a *separating* precedence, namely one where colored symbols are larger than transparent ones, to obtain a separating RPO. In a separating RPO, a term t with a colored symbol will be larger than any term s made of transparent symbols, regardless of the number of symbols in s. For a Knuth-Bendix ordering (KBO), which is defined based on a precedence and a *weight assignment* to symbols, a separating precedence is not sufficient, because of the rôle of weight, which prevents this kind of behavior. As noticed in [49], an *ordinal KBO* [60], where colored symbols have weight ω and transparent symbols have finite weight is a separating ordering. The following lemma shows that a separating ordering excludes AB-mixed literals:

Lemma 1 *If the ordering is separating, all ground Γ-proof trees are colorable.*

Proof: By induction on the structure of the proof tree:
Base Case: By definition, there are no AB-mixed literals in the input clauses $A \cup B$.
Inductive Case:

- *Resolution:* By induction hypothesis, the parent clauses do not contain AB-mixed literals. Since a ground resolvent is made of literals inherited from its parents, it does not contain AB-mixed literals either.
- *Paramodulation/Superposition/Simplification:* let $s \simeq r$ be the equation and $l[s]$ be the paramodulated into or simplified literal. By inductive hypothesis, neither $s \simeq r$ nor $l[s]$ are AB-mixed. Since $s \succ r$ and $\succ$ is separating, 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. □

Based on this lemma we give a different proof of Lemma 2 in [88]. Since our proof uses a separating ordering, it connects the equality-interpolating property proposed for equality sharing with the separating ordering proposed for superposition:

Theorem 3 *The quantifier-free fragment of the theory of equality is equality-interpolating.*

Proof: Assume Γ employs a separating ordering, and (A,B) is an interpolation problem where there exist transparent ground terms. 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 Γ is refutationally complete, there is a successful Γ-derivation from $A \cup B \cup \{t_a \not\simeq t_b\}$. Note that although $t_a \not\simeq t_b$ appears to be part of the input, it plays no rôle in the inferences, except that t_a and t_b get rewritten. Since the ordering is separating, the resulting refutation $\Pi_\Gamma(\Box)$ contains no AB-mixed literals by Lemma 1. $\Pi_\Gamma(\Box)$ is made only of rewriting steps and therefore represents a rewrite proof $t_a \stackrel{*}{\rightarrow} t \stackrel{*}{\leftarrow} t_b$. Since there are no AB-mixed literals in $\Pi_\Gamma(\Box)$, there must be at least a transparent term in this proof chain. Since the ordering is separating, the smallest term t is transparent. Since the inferences are sound, it follows that $A \wedge B \models t_a \simeq t \wedge t_b \simeq t$. □

In [67] an inference is *local* if all its symbols are either in Σ_A or in Σ_B. The definition in [55,50] also requires that if all premises are transparent, so is the conclusion. A proof is *local* if all its inferences are. Since there are no AB-mixed clauses in the input and inferences do not mix colors, a local proof has no AB-mixed clauses:

Definition 25 (Colored proof tree) A proof tree is *colored* if it contains no AB-mixed clauses.

As for literals and clauses, colorable is more general than colored, or local, also for proofs. A separating ordering implies the stronger requirement as well (cf. Theorem 6 in [55]):

Lemma 2 *If the ordering is separating, all ground Γ-proof trees are colored.*

Proof: By Lemma 1 there are no AB-mixed literals and we only need to show by induction that there is no clause with literals of different color.
Base Case: By definition, there are no AB-mixed clauses in $A \cup B$.
Inductive Case:

- *Resolution:* By induction hypothesis, neither $l \vee C$ nor $\neg l \vee D$ is AB-mixed. The only way that the resolvent $C \vee D$ could be AB-mixed is if C contains an A-colored literal, D contains a B-colored literal, or vice versa, and l is transparent. However, this is impossible, because under a separating ordering l and $\neg l$ cannot be transparent and $\succ$-maximal in clauses containing colored literals.

– *Paramodulation/Superposition/Simplification:* assume non-AB-mixed clauses $s \simeq r \vee C$ and $l[s] \vee D$ generate $C \vee l[r] \vee D$ with $s \succ r$. For $C \vee l[r] \vee D$ to be AB-mixed we need that C contains an A-colored literal and $l[r] \vee D$ contains a B-colored literal, or vice versa. If C contains an A-colored literal, since $C \vee s \simeq r$ is not AB-mixed, s is either transparent or A-colored. If s is transparent, since $\succ$ is separating, and $s \succ r$, then r is also transparent. This is impossible, because $s \simeq r$ cannot be $\succ$-maximal and transparent in $C \vee s \simeq r$ which contains an A-colored literal. If s is A-colored, r is either A-colored or transparent. Context l cannot be B-colored, because otherwise $l[s]$ would be AB-mixed. Clause D cannot contain a B-colored literal, because otherwise $l[s] \vee D$ would be AB-mixed. Thus, $C \vee l[r] \vee D$ cannot be AB-mixed. □

5 Interpolation for Equality Sharing and DPLL($\mathscr{T}$)

In this section we see how excluding AB-mixed literals is crucial also when combining theories by equality sharing (or Nelson-Oppen if the reader prefers). Let A and B be disjoint sets of ground $\mathscr{T}$-literals, or unit $\mathscr{T}$-clauses, and $\mathscr{T} = \bigcup_{i=1}^{n} \mathscr{T}_i$ be a combination of equality-interpolating convex disjoint theories, with $\mathscr{T}_i$-satisfiability procedures $Q_1, \ldots, Q_n$, that satisfy the requirements for the completeness of equality sharing[2] (cf. Section 2.3). We assume that each Q_i is capable of generating equality-interpolating terms, proofs, and $\mathscr{T}_i$-interpolants for its proofs.

The interpolation partition of $S = A \cup B$ into A and B, and its separation into $S_1, \ldots, S_n$, based on the theories' signatures, are orthogonal. Constant symbols introduced by separation inherit the status of the term they replace: for example, if $f(g(a)) \simeq b$, where f and g belong to the signatures of different theories, becomes $\{f(c) \simeq b, g(a) \simeq c\}$, the new constant c is A-colored, B-colored or transparent, depending on what $g(a)$ is. This is possible because the input, by definition, contains no AB-mixed terms. For every Q_i the input set of literals is $S_i = A_i \cup B_i$, where A_i contains the $\mathscr{T}_i$-literals in A and B_i contains the $\mathscr{T}_i$-literals in B. By Definition 16, an ES-refutation $\Pi_{ES}(\square)$ is made of input literals and propagated equalities. The restriction to equality-interpolating convex theories (cf. Definition 23), and the assumption that each Q_i generates equality-interpolating terms, ensure that *all* propagated equalities are colorable: if an AB-mixed $t_a \simeq t_b$ is entailed, colorable literals $t_a \simeq t$ and $t_b \simeq t$ are deduced and propagated instead. It follows that under these hypotheses any ES-refutation $\Pi_{ES}(\square)$ is colorable.

In order to define an interpolation system for equality sharing, we need to define partial interpolants for the propagated literals in $\Pi_{ES}(\square)$. Every such literal l is generated by a $\mathscr{T}_i$-deduction $A_i \cup B_i \cup K \vdash_{\mathscr{T}_i} l$, where K is the set of propagated equalities involved in $\mathscr{T}_i$-deducing l, and l may be $\square$ as a special case. A key observation is that a partial $\mathscr{T}_i$-interpolant of l that Q_i may generate cannot be a partial $\mathscr{T}_i$-interpolant

[2] Note however that completeness of an interpolation system for a proof procedure (cf. Definition 10) and completeness of the proof procedure itself are orthogonal concepts: if the proof procedure is incomplete, there will be unsatisfiable $A \cup B$ for which it does not produce a proof, and therefore the interpolation system will not even be invoked to extract an interpolant from the proof.

of l with respect to (A_i, B_i), because the premises of the $\mathscr{T}_i$-deduction of l also include K. It will be a partial $\mathscr{T}_i$-interpolant of l with respect to some partition (A', B') of $A_i \cup B_i \cup K$. Since K contains no AB-mixed literals, it is possible to define A' and B' based on colors as defined by the original (A,B) partition, by using *asymmetric projections* (cf. Definition 8) with respect to A and B: let A' be $A_i \cup K\backslash_B$ and B' be $B_i \cup K \downarrow_B$. It follows that $\Sigma_{A'} = \Sigma_A$, $\Sigma_{B'} = \Sigma_B$, and what is transparent with respect to (A', B') is transparent with respect to (A,B). The absence of AB-mixed equalities is crucial here, because color-based projections cannot handle them.

Definition 26 (Theory-specific partial interpolant) Let $\mathscr{T}_1, \ldots, \mathscr{T}_n$ be equality-interpolating convex disjoint theories, with satisfiability procedures $Q_1, \ldots, Q_n$ capable of generating equality-interpolating terms, proofs, and $\mathscr{T}_i$-interpolants. For all i, $1 \leq i \leq n$, for all $\mathscr{T}_i$-deductions $A_i \cup B_i \cup K \vdash_{\mathscr{T}_i} l$, where A_i is the set of $\mathscr{T}_i$-literals in A, B_i is the set of $\mathscr{T}_i$-literals in B, K is the set of colorable propagated equalities involved in deducing l, and l is a ground colorable propagated literal, the *theory-specific partial interpolant* of l, denoted $PI^i_{(A',B')}(l)$, is the $\mathscr{T}_i$-interpolant of $(A' \wedge \neg(l\backslash_B), B' \wedge \neg(l \downarrow_B))$ generated by Q_i, where $A' = A_i \cup K\backslash_B$ and $B' = B_i \cup K \downarrow_B$.

The following *interpolation system for equality sharing*, that we name YM from the initials of Yorsh and Musuvathi, extracts a $\mathscr{T}$-interpolant of (A,B) from an ES-refutation $\Pi_{ES}(\square)$ of $A \cup B$, by combining the theory-specific partial interpolants computed by the Q_i's for the propagated equalities in $\Pi_{ES}(\square)$. The definition is inductive in the length of the derivation:

Definition 27 (YM interpolation system) Let C be a literal (unit clause) in a refutation of $A \cup B$ by equality sharing:

- Base case (the derivation of C has length 0):
  - If $C \in A$, then $PI(C) = \bot$,
  - If $C \in B$, then $PI(C) = \top$;
- Inductive case (the derivation of C has length $m > 0$):
  if $A_i \cup B_i \cup K \vdash_{\mathscr{T}_i} C$ for some Q_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 $A' = A_i \cup K\backslash_B$, $B' = B_i \cup K \downarrow_B$, and therefore each $l \in A' \cup B'$ was generated by a derivation of length smaller than m.

The base cases of this definition follow HKPYM (cf. Definition 20) rather than MM (cf. Definition 21), so that $PI(C)$ coincides with $PI^i_{(A',B')}(C)$, if C is derived without involving propagated equalities. Indeed, if $K = \emptyset$, we have $K\backslash_B = K \downarrow_B = \emptyset$, $A' = A_i$, $B' = B_i$, $PI(l) = \bot$, which is the unit of disjunction, for all $l \in A_i$, and $PI(l) = \top$, which is the unit of conjunction, for all $l \in B_i$. Similarly, if there were only one theory, K would be empty, and the partial interpolant of C would be equal to its theory-specific partial interpolant. The following example from [88] shows how Definitions 26 and 27 work together:

Example 3 Assume $A = \{f(x_1)+x_2 \simeq x_3,\ f(y_1)+y_2 \simeq y_3,\ y_1 \leq x_1\}$ and $B = \{x_2 \simeq g(b),\ y_2 \simeq g(b),\ x_1 \leq y_1,\ x_3 < y_3\}$. The combined theories are equality ($\mathscr{T}_1$ with procedure Q_1) and linear rational arithmetic ($\mathscr{T}_2$ with procedure Q_2). Separation gives $A_1 = \{a_1 \simeq f(x_1),\ a_2 \simeq f(y_1)\}$, $A_2 = \{a_1+x_2 \simeq x_3,\ a_2+y_2 \simeq y_3,\ y_1 \leq x_1\}$, $B_1 = \{x_2 \simeq g(b),\ y_2 \simeq g(b)\}$, and $B_2 = \{x_1 \leq y_1,\ x_3 < y_3\}$. Thus, $\{f, a_1, a_2\}$ are A-colored, $\{g, b\}$ are B-colored, and $\{x_1, y_1, x_2, y_2, x_3, y_3\}$ are transparent. The set of constant symbols shared by $\mathscr{T}_1$ and $\mathscr{T}_2$ is $V = \{a_1, x_1, a_2, y_1, x_2, y_2\}$. The proof of unsatisfiability of $A \cup B$ generated by equality sharing is made of the following steps, where literals are decorated with their partial interpolants in brackets:

1. Procedure Q_2 deduces $x_1 \simeq y_1$ from input literals $y_1 \leq x_1$ $[\bot]$ and $x_1 \leq y_1$ $[\top]$. Since $x_1, y_1 \in V$, $x_1 \simeq y_1$ is propagated to Q_1. YM computes its partial interpolant as follows. Since no propagated equalities are involved, $A' = A_2$ and $B' = B_2$. Since $x_1 \simeq y_1$ is transparent, $(x_1 \simeq y_1)\backslash_B = \bot$, and $A' \wedge \neg((x_1 \simeq y_1)\backslash_B) = A_2 \wedge \top = A_2$; $(x_1 \simeq y_1) \downarrow_B = x_1 \simeq y_1$, and $B' \wedge \neg((x_1 \simeq y_1) \downarrow_B) = B_2 \cup \{x_1 \not\simeq y_1\}$. The theory-specific partial interpolant of $x_1 \simeq y_1$ is $PI^2_{(A',B')}(x_1 \simeq y_1) = y_1 \leq x_1$, $\mathscr{T}_2$-interpolant of A_2 and $B_2 \cup \{x_1 \not\simeq y_1\}$, because it follows from $y_1 \leq x_1 \in A_2$ and it is $\mathscr{T}_2$-inconsistent with $\{x_1 \leq y_1,\ x_1 \not\simeq y_1\}$, where $x_1 \leq y_1 \in B_2$. The partial interpolant of $x_1 \simeq y_1$ is $PI(x_1 \simeq y_1) = (y_1 \leq x_1 \vee \bot) \wedge \top = y_1 \leq x_1$.
2. Procedure Q_1 deduces $a_1 \simeq a_2$ from input equalities $a_1 \simeq f(x_1)$ $[\bot]$ and $a_2 \simeq f(y_1)$ $[\bot]$, and propagated equality $x_1 \simeq y_1$ $[y_1 \leq x_1]$. Since $a_1, a_2 \in V$, $a_1 \simeq a_2$ is propagated to Q_2. YM computes its partial interpolant as follows. Since the propagated equality $x_1 \simeq y_1$ is involved in deducing $a_1 \simeq a_2$, $K = \{x_1 \simeq y_1\}$; and since $x_1 \simeq y_1$ is transparent, $A' = A_1$ and $B' = B_1 \cup \{x_1 \simeq y_1\}$. Since $a_1 \simeq a_2$ is A-colored, $(a_1 \simeq a_2)\backslash_B = a_1 \simeq a_2$, and $A' \wedge \neg((a_1 \simeq a_2)\backslash_B) = A_1 \cup \{a_1 \not\simeq a_2\}$; $(a_1 \simeq a_2) \downarrow_B = \bot$, and $B' \wedge \neg((a_1 \simeq a_2) \downarrow_B) = B_1 \cup \{x_1 \simeq y_1\}$. The theory-specific partial interpolant of $a_1 \simeq a_2$ is $PI^1_{(A',B')}(a_1 \simeq a_2) = x_1 \not\simeq y_1$, $\mathscr{T}_1$-interpolant of $A_1 \cup \{a_1 \not\simeq a_2\}$ and $B_1 \cup \{x_1 \simeq y_1\}$, because it follows from $\{a_1 \simeq f(x_1),\ a_2 \simeq f(y_1),\ a_1 \not\simeq a_2\}$, and it is inconsistent with $\{x_1 \simeq y_1\}$. The partial interpolant of $a_1 \simeq a_2$ is $PI(a_1 \simeq a_2) = (x_1 \not\simeq y_1 \vee \bot) \wedge y_1 \leq x_1 = y_1 < x_1$.
3. Procedure Q_1 deduces $x_2 \simeq y_2$ from input equalities $x_2 \simeq g(b)$ $[\top]$ and $y_2 \simeq g(b)$ $[\top]$. Since $x_2, y_2 \in V$, $x_2 \simeq y_2$ is propagated to Q_2. YM computes its partial interpolant as follows. Since no propagated equalities are involved, $A' = A_1$ and $B' = B_1$. Since $x_2 \simeq y_2$ is transparent, $(x_2 \simeq y_2)\backslash_B = \bot$, and $A' \wedge \neg((x_2 \simeq y_2)\backslash_B) = A_1 \wedge \top = A_1$; $(x_2 \simeq y_2) \downarrow_B = x_2 \simeq y_2$, and $B' \wedge \neg((x_2 \simeq y_2) \downarrow_B) = B_1 \cup \{x_2 \not\simeq y_2\}$. The theory-specific partial interpolant of $x_2 \simeq y_2$ is $PI^1_{(A',B')}(x_2 \simeq y_2) = \top$, $\mathscr{T}_1$-interpolant of A_1 and $B_1 \cup \{x_2 \not\simeq y_2\}$, because $B_1 \cup \{x_2 \not\simeq y_2\}$ is $\mathscr{T}_1$-inconsistent. The partial interpolant of $x_2 \simeq y_2$ is $PI(x_2 \simeq y_2) = (\top \vee \bot) \wedge \top = \top$.
4. Procedure Q_2 deduces $\Box$ from input literals $a_1 + x_2 \simeq x_3$ $[\bot]$, $a_2 + y_2 \simeq y_3$ $[\bot]$, $x_3 < y_3$ $[\top]$, and propagated equalities $a_1 \simeq a_2$ $[y_1 < x_1]$ and $x_2 \simeq y_2$ $[\top]$. YM computes the partial interpolant of $\Box$, as follows. Since the propagated equalities $a_1 \simeq a_2$ and $x_2 \simeq y_2$ are involved in deducing $\Box$, $K = \{a_1 \simeq a_2,\ x_2 \simeq y_2\}$; and since $a_1 \simeq a_2$ is A-colored and $x_2 \simeq y_2$ is transparent, $A' = A_2 \cup \{a_1 \simeq a_2\}$, and $B' = B_2 \cup \{x_2 \simeq y_2\}$. Then, $(\Box)\backslash_B = \Box = (\Box) \downarrow_B$; $A' \wedge \neg((\Box)\backslash_B) = A_2 \cup \{a_1 \simeq a_2\} \wedge \top = A_2 \cup \{a_1 \simeq a_2\}$; and $B' \wedge \neg((\Box) \downarrow_B) = B_2 \cup \{x_2 \simeq y_2\} \wedge \top = B_2 \cup \{x_2 \simeq y_2\}$. The theory-specific partial interpolant of $\Box$ is $PI^2_{(A',B')}(\Box) = x_3 - x_2 \simeq y_3 - y_2$,

$\mathscr{T}_2$-interpolant of $A_2 \cup \{a_1 \simeq a_2\}$ and $B_2 \cup \{x_2 \simeq y_2\}$, because $\{a_1 + x_2 \simeq x_3,\ a_2 + y_2 \simeq y_3,\ a_1 \simeq a_2\}$ entail $x_3 - x_2 \simeq y_3 - y_2$, which is $\mathscr{T}_2$-inconsistent with $\{x_3 < y_3,\ x_2 \simeq y_2\}$. The partial interpolant of $\Box$, and interpolant of the original problem, is $PI(\Box) = (x_3 - x_2 \simeq y_3 - y_2 \vee y_1 < x_1) \wedge \top = x_3 - x_2 \simeq y_3 - y_2 \vee y_1 < x_1$.

Repeating the same example with symmetric projections shows why asymmetric ones are preferable for Definitions 26 and 27:

Example 4 We assume everything is as in Example 3, except that symmetric projections are applied. We only show the changes that ensue:

1. In the first step, since $x_1 \simeq y_1$ is transparent, $(x_1 \simeq y_1)|_A = x_1 \simeq y_1 = (x_1 \simeq y_1)|_B$; $A' \wedge \neg((x_1 \simeq y_1)|_A) = A_2 \cup \{x_1 \not\simeq y_1\}$, and $B' \wedge \neg((x_1 \simeq y_1)|_B) = B_2 \cup \{x_1 \not\simeq y_1\}$. Then $PI^2_{(A',B')}(x_1 \simeq y_1) = y_1 < x_1$, $\mathscr{T}_2$-interpolant of $A_2 \cup \{x_1 \not\simeq y_1\}$ and $B_2 \cup \{x_1 \not\simeq y_1\}$, because it follows from $\{y_1 \leq x_1,\ x_1 \not\simeq y_1\}$ and is $\mathscr{T}_2$-inconsistent with $\{x_1 \leq y_1,\ x_1 \not\simeq y_1\}$. It follows that $PI(x_1 \simeq y_1) = (y_1 < x_1 \vee \bot) \wedge \top = y_1 < x_1$, so that here symmetric projections give a stronger interpolant than asymmetric ones.
2. In the second step, since $x_1 \simeq y_1$ is transparent, applying symmetric projections to K means that $x_1 \simeq y_1$ ends up in both A' and B': $A' = A_1 \cup \{x_1 \simeq y_1\}$ and $B' = B_1 \cup \{x_1 \simeq y_1\}$. Since $a_1 \simeq a_2$ is A-colored, $(a_1 \simeq a_2)|_A = a_1 \simeq a_2$, and $A' \wedge \neg((a_1 \simeq a_2)|_A) = A_1 \cup \{x_1 \simeq y_1,\ a_1 \not\simeq a_2\}$; $(a_1 \simeq a_2)|_B = \bot$, and $B' \wedge \neg((a_1 \simeq a_2) \downarrow_B) = B_1 \cup \{x_1 \simeq y_1\}$. Then $PI^1_{(A',B')}(a_1 \simeq a_2) = \bot$, $\mathscr{T}_1$-interpolant of $A_1 \cup \{x_1 \simeq y_1,\ a_1 \not\simeq a_2\}$ and $B_1 \cup \{x_1 \simeq y_1\}$, because the first set is $\mathscr{T}_1$-inconsistent. It follows that $PI(a_1 \simeq a_2) = (\bot \vee y_1 < x_1) \wedge y_1 < x_1 = y_1 < x_1$, so that here symmetric projections give the same result as asymmetric ones.
3. In the third step, since $x_2 \simeq y_2$ is transparent, $(x_2 \simeq y_2)|_A = x_2 \simeq y_2 = (x_2 \simeq y_2)|_B$; $A' \wedge \neg((x_2 \simeq y_2)|_A) = A_1 \cup \{x_2 \not\simeq y_2\}$; and $B' \wedge \neg((x_2 \simeq y_2)|_B) = B_1 \cup \{x_2 \not\simeq y_2\}$. Then $PI^1_{(A',B')}(x_2 \simeq y_2) = \top$, $\mathscr{T}_1$-interpolant of $A_1 \cup \{x_2 \not\simeq y_2\}$ and $B_1 \cup \{x_2 \not\simeq y_2\}$, because $B_1 \cup \{x_2 \not\simeq y_2\}$ is $\mathscr{T}_1$-inconsistent. As before, $PI(x_2 \simeq y_2) = (\top \vee \bot) \wedge \top = \top$.
4. In the fourth step, since $a_1 \simeq a_2$ is A-colored and $x_2 \simeq y_2$ is transparent, $A' = A_2 \cup \{a_1 \simeq a_2,\ x_2 \simeq y_2\}$, and $B' = B_2 \cup \{x_2 \simeq y_2\}$. Since $(\Box)|_A = \Box = (\Box)|_B$, we have $A' \wedge \neg((\Box)|_A) = A_2 \cup \{a_1 \simeq a_2,\ x_2 \simeq y_2\} \wedge \top = A_2 \cup \{a_1 \simeq a_2,\ x_2 \simeq y_2\}$; and $B' \wedge \neg((\Box)|_B) = B_2 \cup \{x_2 \simeq y_2\} \wedge \top = B_2 \cup \{x_2 \simeq y_2\}$. Then $PI^2_{(A',B')}(\Box) = x_3 \simeq y_3$, $\mathscr{T}_2$-interpolant of $A_2 \cup \{a_1 \simeq a_2,\ x_2 \simeq y_2\}$ and $B_2 \cup \{x_2 \simeq y_2\}$, because $\{a_1 + x_2 \simeq x_3,\ a_2 + y_2 \simeq y_3,\ a_1 \simeq a_2,\ x_2 \simeq y_2\}$ entail $x_3 \simeq y_3$, which is $\mathscr{T}_2$-inconsistent with $x_3 < y_3$. The partial interpolant of $\Box$ is $PI(\Box) = (x_3 \simeq y_3 \vee y_1 < x_1 \vee \top) \wedge \top = \top$, which is trivial. The glitch is that symmetric projections put the transparent propagated equality $x_2 \simeq y_2$ in both A' and B': since $x_2 \simeq y_2$ descends only from B, this spoils the interpolant.

The following theorem proves the completeness of YM for refutations by equality sharing. Since what matters are the propagated equalities, the inner induction in the proof is on the set K:

Theorem 4 (Yorsh and Musuvathi, 2005) *If $\mathscr{T} = \bigcup_{i=1}^n \mathscr{T}_i$ is a union of equality-interpolating convex disjoint theories, YM is a complete interpolation system for refutations by equality sharing of sets $A \cup B$ of ground $\mathscr{T}$-literals.*

Proof: We prove that for all literals, or unit clauses, C in the refutation, $PI(C)$ is a $\mathscr{T}$-interpolant of $g_A(C) = A \wedge \neg(C\backslash_B)$ and $g_B(C) = B \wedge \neg(C\downarrow_B)$, that is, it satifies the requirements:

1. $g_A(C) \vdash_{\mathscr{T}} PI(C)$,
2. $g_B(C) \wedge PI(C) \vdash_{\mathscr{T}} \bot$, and
3. $PI(C)$ is transparent.

The proof is by induction on the length of the derivation of C. The base case (proof of length 0, $C \in A$ or $C \in B$) is the same as in the proof of Theorem 2. The inductive case, for a C such that $A_i \cup B_i \cup K \vdash_{\mathscr{T}_i} C$ by a derivation of length $m > 0$, in a theory $\mathscr{T}_i$, 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 26, $PI(C)$ is a $\mathscr{T}_i$-interpolant of $(A_i \wedge \neg(C\backslash_B), B_i \wedge \neg(C\downarrow_B))$; since $A_i \subseteq A$ and $B_i \subseteq B$, $PI(C)$ is also a $\mathscr{T}$-interpolant of $A \wedge \neg(C\backslash_B)$ and $B \wedge \neg(C\downarrow_B)$.
Inductive case: if $K \neq \emptyset$, then $A' = A_i \cup K\backslash_B$, $B' = B_i \cup K\downarrow_B$ and $PI(C) = (PI^i_{(A',B')}(C) \vee \bigvee_{l \in K\backslash_B} PI(l)) \wedge \bigwedge_{l \in K\downarrow_B} PI(l)$, because $PI(l) = \bot$ for all $l \in A_i$ and $PI(l) = \top$ for all $l \in B_i$.
Inductive hypothesis: for all $l \in K\backslash_B$, $PI(l)$ is a $\mathscr{T}$-interpolant of $(A \wedge \neg(l\backslash_B), B \wedge \neg(l\downarrow_B))$, that is, a $\mathscr{T}$-interpolant of $(A \wedge \neg l, B)$, because $l\backslash_B = l$ and $l\downarrow_B = \bot$ as $l \in K\backslash_B$; for all $l \in K\downarrow_B$, $PI(l)$ is a $\mathscr{T}$-interpolant of $(A \wedge \neg(l\backslash_B), B \wedge \neg(l\downarrow_B))$, that is, a $\mathscr{T}$-interpolant of $(A, B \wedge \neg l)$, because $l\backslash_B = \bot$ and $l\downarrow_B = l$ as $l \in K\downarrow_B$; so that:

– For all $l \in K\backslash_B$:
(1A) $A \wedge \neg l \vdash_{\mathscr{T}} PI(l)$ or, equivalently, $A \vdash_{\mathscr{T}} l \vee PI(l)$,
(2A) $B \wedge PI(l) \vdash_{\mathscr{T}} \bot$,
(3A) $PI(l)$ is transparent; and
– For all $l \in K\downarrow_B$:
(1B) $A \vdash_{\mathscr{T}} PI(l)$,
(2B) $B \wedge \neg l \wedge PI(l) \vdash_{\mathscr{T}} \bot$, or, equivalently, $B \wedge PI(l) \vdash_{\mathscr{T}} l$,
(3B) $PI(l)$ is transparent.

We show that Requirements 1, 2 and 3 are satisfied:

1. $A \wedge \neg(C\backslash_B) \vdash_{\mathscr{T}} PI(C)$:
   By Definition 26, $A_i \wedge K\backslash_B \wedge \neg(C\backslash_B) \vdash_{\mathscr{T}_i} PI^i_{(A',B')}(C)$, or, equivalently,

$$A_i \wedge \neg(C\backslash_B) \vdash_{\mathscr{T}_i} \neg K\backslash_B \vee PI^i_{(A',B')}(C) \tag{1}$$

   where $\neg K\backslash_B$ is the disjunction $\neg l_1 \vee \ldots \vee \neg l_q$, if $K\backslash_B$ is the conjunction $l_1 \wedge \ldots \wedge l_q$. By induction hypothesis (1A), we have

$$A \vdash_{\mathscr{T}} l_j \vee PI(l_j) \quad for \;\; 1 \leq j \leq q \tag{2}$$

   By q resolution steps between (1) and (2), and since $A \Rightarrow A_i$, it follows that $A \wedge \neg(C\backslash_B) \vdash_{\mathscr{T}} PI^i_{(A',B')}(C) \vee \bigvee_{l \in K\backslash_B} PI(l)$. By induction hypothesis (1B), $A \vdash_{\mathscr{T}} PI(l)$ for all $l \in K\downarrow_B$. Therefore, we conclude that $A \wedge \neg(C\backslash_B) \vdash_{\mathscr{T}} (PI^i_{(A',B')}(C) \vee \bigvee_{l \in K\backslash_B} PI(l)) \wedge \bigwedge_{l \in K\downarrow_B} PI(l)$.

2. $B \wedge \neg(C\downarrow_B) \wedge PI(C) \vdash_{\mathscr{T}} \bot$:
   By Definition 26, $B_i \wedge K\downarrow_B \wedge \neg(C\downarrow_B) \wedge PI^i_{(A',B')}(C) \vdash_{\mathscr{T}_i} \bot$. Since $B \Rightarrow B_i$, we have

$$B \wedge K\downarrow_B \wedge \neg(C\downarrow_B) \wedge PI^i_{(A',B')}(C) \vdash_{\mathscr{T}_i} \bot \tag{3}$$

   By induction hypothesis (2A), we have $B \wedge PI(l) \vdash_{\mathscr{T}} \bot$ for all $l \in K\setminus_B$, and thus $B \wedge \bigvee_{l\in K\setminus_B} PI(l) \vdash_{\mathscr{T}} \bot$, and

$$B \wedge K\downarrow_B \wedge \neg(C\downarrow_B) \wedge \bigvee_{l\in K\setminus_B} PI(l) \vdash_{\mathscr{T}} \bot \tag{4}$$

   Combining (3) and (4) gives

$$B \wedge K\downarrow_B \wedge \neg(C\downarrow_B) \wedge (PI^i_{(A',B')}(C) \vee \bigvee_{l\in K\setminus_B} PI(l)) \vdash_{\mathscr{T}} \bot$$

   or, equivalently,

$$B \wedge \neg(C\downarrow_B) \wedge (PI^i_{(A',B')}(C) \vee \bigvee_{l\in K\setminus_B} PI(l)) \vdash_{\mathscr{T}} \neg K\downarrow_B \tag{5}$$

   where $\neg K\downarrow_B$ is the disjunction $\neg l_1 \vee \ldots \vee \neg l_q$, if $K\downarrow_B$ is the conjunction $l_1 \wedge \ldots \wedge l_q$. By induction hypothesis (2B), we have

$$B \wedge PI(l_j) \vdash_{\mathscr{T}} l_j \quad for \;\; 1 \le j \le q \tag{6}$$

   By q resolution steps between (5) and (6), we get

$$B \wedge \neg(C\downarrow_B) \wedge (PI^i_{(A',B')}(C) \vee \bigvee_{l\in K\setminus_B} PI(l)) \wedge \bigwedge_{l\in K\downarrow_B} PI(l) \vdash_{\mathscr{T}} \bot$$

   that is, $B \wedge \neg(C\downarrow_B) \wedge PI(C) \vdash_{\mathscr{T}} \bot$.
3. $PI(C)$ is transparent, because $PI^i_{(A',B')}(C)$ is transparent by Definition 26, and the $PI(l)$'s are transparent by induction hypotheses (3A) and (3B). □

Having an interpolation system for DPLL and YM, we have all the ingredients for an interpolation system for DPLL($\mathscr{T}$). Let A and B be sets of ground $\mathscr{T}$-clauses. According to Definition 19, a DPLL($\mathscr{T}$)-refutation of $A \cup B$ will be a refutation by propositional resolution plus $\mathscr{T}$-lemmas. Such a refutation is colorable, because all its literals are input literals, since also $\mathscr{T}$-lemmas are made of input literals (cf. rule T-Propagate in Section 2.4). Let $CP_{\mathscr{T}}$ be the set of $\mathscr{T}$-lemmas that appear in the DPLL($\mathscr{T}$)-refutation of $A \cup B$. Such a refutation shows that $A \cup B$ is $\mathscr{T}$-unsatisfiable, by showing that $A \cup B \cup CP_{\mathscr{T}}$ is propositionally unsatisfiable. An interpolation system for DPLL($\mathscr{T}$) will be given by an interpolation system for propositional resolution plus partial interpolants for $\mathscr{T}$-lemmas. A clause C is a $\mathscr{T}$-lemma, if and only if its negation $\neg C$, which is a set, or conjunction, of literals, is $\mathscr{T}$-unsatisfiable. Since C is colorable, $C = C\setminus_B \vee C\downarrow_B$, whence $\neg C = (\neg C)\setminus_B \wedge (\neg C)\downarrow_B$. Then, $(\neg C)\setminus_B \wedge (\neg C)\downarrow_B$ is $\mathscr{T}$-unsatisfiable, and we can compute a $\mathscr{T}$-interpolant of $((\neg C)\setminus_B, (\neg C)\downarrow_B)$ by YM. This $\mathscr{T}$-interpolant provides the partial interpolant for the $\mathscr{T}$-lemma C.

Then, we can define two interpolation systems for DPLL($\mathscr{T}$) by adding a case for $\mathscr{T}$-lemmas to either HKPYM (cf. Definition 20), as done in [88], or MM (cf. Definition 21), as done in [45,26]. The case for $\mathscr{T}$-lemmas is a third base case, because they are sort of input clauses from the point of view of the propositional engine:

Definition 28 (HKPYM($\mathscr{T}$) interpolation system) For $c\colon C$ a clause in a DPLL($\mathscr{T}$)-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 a $\mathscr{T}$-lemma, $PI(c)$ is the $\mathscr{T}$-interpolant of $((\neg C)\backslash_B, (\neg C)\downarrow_B)$ produced by YM from the refutation $\neg C \vdash_{\mathscr{T}} \bot$;
- If $c\colon C \vee D$ is a propositional resolvent of $p_1\colon l \vee C$ and $p_2\colon \neg l \vee D$ 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 $PI(c) = (l \vee PI(p_1)) \wedge (\neg l \vee PI(p_2))$.

Definition 29 (MM($\mathscr{T}$) interpolation system) For $c\colon C$ a clause in a DPLL($\mathscr{T}$)-refutation of $A \cup B$:

- If $c\colon C \in A$, then $PI(c) = C|_{A,B}$,
- If $c\colon C \in B$, then $PI(c) = \top$,
- If $c\colon C$ is a $\mathscr{T}$-lemma, $PI(c)$ is the $\mathscr{T}$-interpolant of $((\neg C)\backslash_B, (\neg C)\downarrow_B)$ produced by YM from the refutation $\neg C \vdash_{\mathscr{T}} \bot$;
- If $c\colon C \vee D$ is a propositional resolvent of $p_1\colon l \vee C$ and $p_2\colon \neg l \vee D$ then:
  - If l is A-colored, then $PI(c) = PI(p_1) \vee PI(p_2)$,
  - If l is B-colored or transparent, then $PI(c) = PI(p_1) \wedge PI(p_2)$.

The completeness of HKPYM($\mathscr{T}$), or MM($\mathscr{T}$), follows from that of HKPYM, or MM, and YM. However, there are theories used in practice that are not convex:

Example 5 By relying on the proof that the theory of equality is equality-interpolating (cf. Theorem 3), it was shown in [88] that the theory of *non-empty, possibly cyclic lists*, which is convex, is also equality-interpolating. The theory of *possibly empty, possibly cyclic lists* is not convex, since $l \simeq nil \vee l \simeq cons(car(l), cdr(l))$ is valid in the theory, but neither disjunct is. Similarly, the theory of *arrays* is not convex, since $select(store(a,i,e),j) \simeq e \vee select(store(a,i,e),j) \simeq select(a,j)$ is valid in the theory, but neither disjunct is. The presentations of these theories can be found, for instance, in [3].

Example 6 *Linear rational arithmetic* is convex (cf. Example 10.12 in [16]), and it was shown to be equality-interpolating in [88]: first, $A \wedge B \Rightarrow a \simeq b$, where a is A-colored and b is B-colored, is written as $A \wedge B \Rightarrow a \leq b \wedge b \leq a$; second, it is shown that if $A \wedge B \Rightarrow a \leq b$, there exists a transparent term t such that $A \wedge B \Rightarrow a \leq t \leq b$ (cf. Lemma 3 in [88]); third, from $A \wedge B \Rightarrow a \leq t_1 \leq b$ and $A \wedge B \Rightarrow b \leq t_2 \leq a$, one gets $A \wedge B \Rightarrow a \simeq t_1 \simeq t_2 \simeq b$, so that t_1, or t_2, is the equality-interpolating term. *Linear integer arithmetic* is not convex: $2 \leq a \wedge a \leq 3$ implies $a \simeq 2 \vee a \simeq 3$, but it does not imply either disjunct. Neither it is equality-interpolating: if A contains $2a \simeq c$ and B contains $2b \simeq c$, where a is A-colored, b is B-colored, and c is transparent, $A \wedge B \Rightarrow a \simeq b$, but

the needed interpolating term $c/2$ is not in the integers. Approaches to interpolation in the quantifier-free fragment of linear integer arithmetic were presented in [17,19].

A generalization of the definition of equality-interpolating theory to non-convex theories was suggested in [87]; however, no theory was shown to satisfy it. A weaker, and therefore more general, definition, which is satisfied by several theories of interest, was given in [21,23], together with a thorough comparison with [87], and a non-deterministic algorithm for interpolation in any combination of disjoint, stably-infinite, quantifier-free interpolating theories. This algorithm is based on the meta-level rules approach of [22], rather than on a color-based interpolation system, and is therefore outside the scope of this survey. We refer the interested reader to [23].

6 Interpolation for Ground Superposition

In this section we push as far as possible the color-based approach to interpolation by giving a color-based interpolation system for ground Γ-refutations of $A \cup B$ containing ground clauses.

Definition 30 (Interpolation system GΓI) Let $c\colon C$ be a clause in a ground colorable Γ-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 a Γ-inference from premises p_1 and p_2, $PI(c)$ is defined as follows:
  - Resolution: $c\colon C \vee D$ is generated from $p_1\colon l \vee C$ and $p_2\colon \neg l \vee D$
    - 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 $PI(c) = (l \vee PI(p_1)) \wedge (\neg l \vee PI(p_2))$;
  - Reflection: $c\colon C$ is generated from $p_1\colon s \not\simeq s \vee C$: $PI(c) = PI(p_1)$;
  - Paramodulation: $c\colon C \vee l[r] \vee D$ is generated from $p_1\colon s \simeq r \vee C$ and $p_2\colon l[s] \vee D$; and Superposition: $c\colon C \vee l[r] \bowtie t \vee D$ is generated from $p_1\colon s \simeq r \vee C$ and $p_2\colon l[s] \bowtie t \vee D$
    - If $s \simeq r$ is A-colored, then $PI(c) = PI(p_1) \vee PI(p_2)$,
    - If $s \simeq r$ is B-colored, then $PI(c) = PI(p_1) \wedge PI(p_2)$, and
    - If $s \simeq r$ is transparent, then $PI(c) = (s \simeq r \vee PI(p_1)) \wedge (s \not\simeq r \vee PI(p_2))$.

GΓI is a generalization of HKPYM (cf. Definition 20) from propositional logic to ground first-order logic with equality.

Example 7 Assume $A = \{Q(f(a)),\ f(a) \simeq c\}$ and $B = \{\neg Q(f(b)),\ f(b) \simeq c\}$, so that a is A-colored, b is B-colored, and all other symbols are transparent. As in previous examples, each clause in the refutation is decorated with its partial interpolant surrounded by brackets.

- $Q(f(a))$ $[\bot]$ is simplified by $f(a) \simeq c$ $[\bot]$ to $Q(c)$ $[\bot]$, as $f(a) \succ c$ in a separating ordering: since $f(a) \simeq c$ is A-colored, the partial interpolant is $\bot \vee \bot = \bot$;

- $\neg Q(f(b))$ $[\top]$ is simplified by $f(b) \simeq c$ $[\top]$ to $\neg Q(c)$ $[\top]$, as $f(b) \succ c$ in a separating ordering: since $f(b) \simeq c$ is B-colored, the partial interpolant is $\top \wedge \top = \top$;
- $Q(c)$ $[\bot]$ resolves with $\neg Q(c)$ $[\top]$ to yield $\Box$ $[Q(c)]$: since $Q(c)$ is transparent, the partial interpolant is $(Q(c) \vee \bot) \wedge (\neg Q(c) \vee \top) = Q(c)$.

Theorem 5 *The interpolation system $G\Gamma I$ is complete for ground colorable Γ-refutations.*

Proof: We need to prove that for all clauses $c\colon C$ in the refutation,

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

The proof is by induction on the structure of the refutation: the base case is the same as for Theorem 2.
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. $B \wedge \neg(p_k|_B) \wedge PI(p_k) \vdash \bot$ or, equivalently, $B \wedge PI(p_k) \vdash p_k|_B$
3. $PI(p_k)$ is transparent.

Inductive cases:
Resolution: $c\colon C \vee D$ is generated from $p_1\colon l \vee C$ and $p_2\colon \neg l \vee D$; there are three cases:

- l is A-colored; then $(l \vee C)|_A = l \vee C|_A$, $(\neg l \vee D)|_A = \neg l \vee D|_A$, $(l \vee C)|_B = C|_B$ and $(\neg l \vee D)|_B = D|_B$:
  1. $A \vdash (C \vee D)|_A \vee PI(p_1) \vee PI(p_2)$
     From inductive hypothesis (1) we have $A \vdash (l \vee C)|_A \vee PI(p_1)$ and $A \vdash (\neg l \vee D)|_A \vee PI(p_2)$. Since l is A-colored and $\succ$-maximal in $l \vee C$, and $PI(p_1)$ is transparent, l is $\succ$-maximal in $(l \vee C)|_A \vee PI(p_1)$. Similarly, $\neg l$ is $\succ$-maximal in $(\neg l \vee D)|_A \vee PI(p_2)$. Thus, 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 follows the inductive conclusion.
  3. Transparency of the partial interpolant follows from the inductive hypothesis.
- l is B-colored; then $(l \vee C)|_A = C|_A$, $(\neg l \vee D)|_A = D|_A$, $(l \vee C)|_B = l \vee C|_B$ and $(\neg l \vee D)|_B = \neg l \vee D|_B$:
  1. $A \wedge \neg(C \vee D)|_A \vdash PI(p_1) \wedge PI(p_2)$ is equivalent to
     $A \wedge \neg(C|_A) \wedge \neg(D|_A) \vdash PI(p_1) \wedge PI(p_2)$ which follows from inductive hypotheses (1) $A \wedge \neg(C|_A) \vdash PI(p_1)$ and $A \wedge \neg(D|_A) \vdash PI(p_2)$.
  2. $B \wedge PI(p_1) \wedge PI(p_2) \vdash (C \vee D)|_B$
     From inductive hypothesis (2) we have $B \wedge PI(p_1) \vdash (l \vee C)|_B$ and $B \wedge PI(p_2) \vdash (\neg l \vee D)|_B$; by resolution this gives $B \wedge PI(p_1) \wedge PI(p_2) \vdash (C \vee D)|_B$.
  3. Transparency of the partial interpolant follows from the inductive hypothesis.
- l is transparent:
  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 $B \wedge PI(p_2) \vdash \neg l \vee D|_B$ 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 $B \wedge PI(p_1) \vdash l \vee C|_B$ 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.

Reflection: $c\colon C$ is generated from $p_1\colon s \not\simeq s \vee C$

- s is A-colored: $(s \not\simeq s \vee C)|_A = s \not\simeq s \vee C|_A$, $(s \not\simeq s \vee C)|_B = C|_B$
  1. $A \wedge \neg(C|_A) \vdash PI(c)$
     Inductive hypothesis (1) is $A \wedge \neg(s \not\simeq s \vee C|_A) \vdash PI(p_1)$, whence $A \wedge s \simeq s \wedge \neg(C|_A) \vdash PI(p_1)$, and $A \wedge \neg(C|_A) \vdash PI(c)$, since $PI(c) = PI(p_1)$.
  2. $B \wedge \neg(C|_B) \wedge PI(c) \vdash \bot$
     Inductive hypothesis (2) gives $B \wedge \neg(C|_B) \wedge PI(c) \vdash \bot$ since $PI(c) = PI(p_1)$.
  3. The partial interpolant is transparent by inductive hypothesis.
- s is B-colored: $(s \not\simeq s \vee C)|_A = C|_A$, $(s \not\simeq s \vee C)|_B = s \not\simeq s \vee C|_B$; the rest of the proof is symmetric to the previous case.
- s is transparent: $(s \not\simeq s \vee C)|_A = s \not\simeq s \vee C|_A$, $(s \not\simeq s \vee C)|_B = s \not\simeq s \vee C|_B$; the rest of the proof is as in the previous cases.

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

- $s \simeq r$ is A-colored: either s and r are both A-colored, or, since $s \succ r$, s is A-colored and r is transparent; since there are no AB-mixed literals, either $l[s]$ and $l[r]$ are both A-colored, or $l[s]$ is A-colored and $l[r]$ is transparent; $(s \simeq r \vee C)|_A = s \simeq r \vee C|_A$, $(l[s] \vee D)|_A = l[s] \vee D|_A$, $(C \vee l[r] \vee D)|_A = C|_A \vee l[r] \vee D|_A$, $(s \simeq r \vee C)|_B = C|_B$, $(l[s] \vee D)|_B = D|_B$:
  1. $A \vdash (C \vee l[r] \vee D)|_A \vee PI(p_1) \vee PI(p_2)$
     Inductive hypothesis (1) gives $A \vdash s \simeq r \vee C|_A \vee PI(p_1)$ and $A \vdash l[s] \vee D|_A \vee PI(p_2)$; since $s \simeq r$ is $\succ$-maximal in $s \simeq r \vee C$ and $PI(p_1)$ is transparent, $s \simeq r$ is $\succ$-maximal also in $s \simeq r \vee C|_A \vee PI(p_1)$; similarly, $l[s]$ is $\succ$-maximal also in $l[s] \vee D|_A \vee PI(p_2)$; 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-colored: similar to the previous case, either $l[s]$ and $l[r]$ are both B-colored, or $l[s]$ is B-colored and $l[r]$ transparent, so that $(s \simeq r \vee C)|_B = s \simeq r \vee C|_B$, $(l[s] \vee D)|_B = l[s] \vee D|_B$, $(C \vee l[r] \vee D)|_B = C|_B \vee l[r] \vee D|_B$, $(s \simeq r \vee C)|_A = C|_A$, $(l[s] \vee D)|_A = D|_A$:

1. $A \wedge \neg((C \vee l[r] \vee D)|_A) \vdash PI(p_1) \wedge PI(p_2)$ is equivalent to $A \wedge \neg(C|_A) \wedge \neg l[r]|_A \wedge \neg(D|_A) \vdash PI(p_1) \wedge PI(p_2)$ which follows from $A \wedge \neg C|_A \vdash PI(p_1)$ and $A \wedge \neg D|_A \vdash PI(p_2)$, that hold by inductive hypothesis (1).
2. $B \wedge PI(p_1) \wedge PI(p_2) \vdash (C \vee l[r] \vee D)|_B$ or, equivalently, $B \wedge PI(p_1) \wedge PI(p_2) \vdash C|_B \vee l[r] \vee D|_B$ (*)
   By inductive hypothesis (2) we have $B \wedge PI(p_1) \vdash s \simeq r \vee C|_B$ and $B \wedge PI(p_2) \vdash l[s] \vee D|_B$ so that (*) follows by a paramodulation step.
3. The partial interpolant is transparent by inductive hypothesis.

– $s \simeq r$ is transparent:
1. $A \vdash (C \vee l[r] \vee D)|_A \vee ((s \simeq r \vee PI(p_1)) \wedge (s \not\simeq r \vee PI(p_2)))$ is equivalent to $A \wedge ((s \not\simeq r \wedge \neg PI(p_1)) \vee (s \simeq r \wedge \neg PI(p_2))) \vdash (C \vee l[r] \vee D)|_A$
   (a) Assume $s \not\simeq r$; then 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 $A \wedge \neg(s \simeq r \vee C)|_A \vdash PI(p_1)$, whence $A \wedge (s \not\simeq r)|_A \wedge \neg(C|_A) \vdash PI(p_1)$, and $A \wedge s \not\simeq r \wedge \neg PI(p_1) \vdash C|_A$, which proves the required.
   (b) Assume $s \simeq r$; then 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, and the literals $l[r]$ and $l[s]$ are treated in the same way by the projection, as s and r are transparent. By induction hypothesis (1) we have $A \wedge \neg(l[s] \vee D)|_A \vdash PI(p_2)$, whence $A \wedge \neg PI(p_2) \vdash (l[s] \vee D)|_A$, and we are done.
2. $B \wedge (s \simeq r \vee PI(p_1)) \wedge (s \not\simeq r \vee PI(p_2)) \vdash (C \vee l[r] \vee D)|_B$
   (a) Assume $s \simeq r$; then $s \simeq r \vee PI(p_1)$ is subsumed, and $s \not\simeq r \vee PI(p_2)$ reduces to $PI(p_2)$. 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, and $l[r]$ and $l[s]$ are treated in the same way by the projection, as s and r are transparent. By induction hypothesis (2) we have $B \wedge PI(p_2) \vdash (l[s] \vee D)|_B$, which closes this case.
   (b) Assume $s \not\simeq r$; then $s \not\simeq r \vee PI(p_2)$ is subsumed, and $s \simeq r \vee PI(p_1)$ reduces to $PI(p_1)$. 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 $B \wedge PI(p_1) \vdash (s \simeq r \vee C)|_B$, or $B \wedge PI(p_1) \vdash (s \simeq r)|_B \vee C|_B$, whence $B \wedge s \not\simeq r \wedge PI(p_1) \vdash C|_B$, because $s \simeq r$ is transparent.
3. Transparency follows from the transparency of $s \simeq r$ and the inductive hypothesis.

Superposition is treated like paramodulation, with $l[s]$ replaced by $l[s] \bowtie t$, and the case analysis for simplification is subsumed by those for paramodulation and superposition. □

Corollary 1 *If the ordering is separating, GΓI is a complete interpolation system for ground Γ-refutations.*

Proof: It follows from Lemma 1 and Theorem 5. □

7 Discussion

The existence of interpolants in first-order logic was established by Craig's Interpolation Lemma in [28]. Constructive proofs based on cut elimination were given in [54, 84,40]. One of the first studies of the complexity of interpolation appeared in [72]. In [56,77,57] interpolation was studied to advance what is known as "Cook's program," which can be summarized as follows. Since the result by Cook and Reckhow in [27] that $NP = co\text{–}NP$ if and only if there is a polynomially bounded proof system for the classical propositional tautologies, the goal of that research line has been to prove that there is no polynomially bounded proof system in order to settle $NP \neq co\text{–}NP$. A proof system as defined in [27] (or [85] for a survey), is a function f such that $f(x) = y$, if x is a string representing a proof, and y is a string representing the tautology proved by x. The proof system f is polynomially bounded if the length of x is polynomially bounded by the length of y. The results of [56,77] are lower bounds on the length of interpolants, that imply lower bounds on the length of proofs and show that the proof system at hand is not polynomially bounded. The investigation in [57] studies limitations to this approach.

The relevance of interpolation for model checking was discovered by Ken McMillan beginning with [64], and since then interpolation has received increasing attention (e.g., [48,65,52,45,17,68,18,86,19,2,22,78,80,23]), and has been implemented in reasoners, such as *Foci* [65,67,69], *MathSAT* [26], *OpenSMT* [24], *Vampire* [49,50], and program analyzers based on model checking, such as *CSIsat* [7], *Wolverine* [59], and *Eldarica* [80]. In model checking one is interested in interpolants that accelerate convergence towards a fixed point in forward, or backward, reachability search. This goal has led to study several techniques, including interpolation systems with *labelling*, or *coloring*, functions, to tune the *strength* of interpolants [37,86,78], and *abstraction* over terms [2] or entire interpolants [80].

There are several approaches to practical interpolation. One is the inductive and color-based approach surveyed here, which is appropriate for generic inference and transition systems. Another one consists of building interpolation into specialized inference systems [48,65,17–19] and satisfiability procedures [86,46,22,39,2] both with the theory built-in. For instance, the interpolation algorithm of [39] is for a congruence-closure-based satisfiability procedure for the quantifier-free fragment of the theory of equality. The interpolation systems for equality sharing and DPLL($\mathscr{T}$), that we covered in this article, are consumers of interpolation algorithms incorporated into satisfiability procedures for specific theories, because they assume that every $\mathscr{T}_i$-satisfiability procedure produces $\mathscr{T}_i$-interpolants. A third one formulates the interpolation problem as a set of Horn clauses with an unknown query, and gives an algorithm to solve it: the solution is a conjunction of constraints that represents the interpolant [47,79].

An approach to combination and interpolation based on the notion of *locality* of theories of [44,63] was pursued in [83]. A theory $\mathscr{T}$ is *local* in this sense, if the $\mathscr{T}$-satisfiability of a set of ground clauses can be decided by involving only finitely many ground instances of the $\mathscr{T}$-axioms. A combination of theories is seen as a series of local extensions of a convex *core theory*, where an extension is local if the resulting theory is local. The notion of equality-interpolating convex theory was generalized in

[83] to that of *P-interpolating* convex theory, where P is the main predicate symbol of the core theory (e.g., an ordering in place of equality).

The methodology of [20,22,21,23] offers a different approach to both interpolation algorithm design and combination of theories. It uses *meta-rules* to obtain interpolation algorithms: it was applied in [22] to the theory of arrays with extensionality, in [20] to a combination of arrays and integer difference constraints, and in [21,23] to any combination of disjoint, stably-infinite, quantifier-free interpolating theories. The latter algorithm relies on a notion of equality-interpolating theories, which is the most general in the literature thus far.

The study of interpolation for superposition in [67] aimed at generalizing to refutations in first-order logic with equality the color-based approach that had been applied to propositional resolution [64] and to quantifier-free fragments of first-order theories [48,65]. The notions of *local inference* and *local proof* were introduced in [67], together with that of *ordering oriented for* (A,B), which was a precursor of the separating ordering. "Local" was replaced by "colored" in [45]. Ground refutations by superposition with a separating ordering were shown to be colored in [55].

Approaches based on *instantiation* and *proof transformation* were developed in [25] and [69] for ground refutations by a DPLL($\mathscr{T}$)-based SMT-solver, equipped with an instantiation procedure (e.g., [35,31,42,70,43]). The instantiation procedure is used to generate ground instances of non-ground input axioms. For instance in [69] this technique is applied to proofs generated by Z3, that are processed and then passed on to Foci for interpolation. The refutations to be interpolated are ground, but not necessarily local, because the instantiation procedure may have introduced AB-mixed literals. Thus, AB-mixed literals are eliminated by proof transformation.

In this article, we surveyed color-based interpolation systems for ground refutations. After covering interpolation systems for propositional resolution, or, equivalently, DPLL, we analyzed *interpolation and equality*. When going from propositional reasoning to equality reasoning, already in the ground case, and regardless of whether equality reasoning is done by rewriting or congruence closure, interpolation becomes more difficult, because equality makes colors unstable as soon as terms of different colors become equal. We clarified how the requirement of *convex equality-interpolating theory* for equality sharing and that of a *separating ordering* for ground superposition address this same issue, as both aim at avoiding AB-mixed literals, and ensuring that the proof is *colorable* (cf. Lemma 1). We connected them by using the separating ordering to prove that the quantifier-free fragment of the theory of equality is equality-interpolating (cf. Theorem 3). Then we surveyed color-based interpolation systems for combinations of convex equality-interpolating theories by equality sharing and for DPLL($\mathscr{T}$). The key point is that propagated equalities are not AB-mixed, so that the proof is colorable. Next we studied interpolation of ground Γ-refutations. Under the assumption of a separating ordering, ground Γ-refutations are colorable (cf. Lemma 1) and colored (cf. Lemma 2). "Colorable" is more general and more precise then "colored," since it captures exactly the absence of AB-mixed literals. We gave a new interpolation system, named GΓI, which is *complete* for ground Γ-refutations (cf. Theorem 5), and generalizes the interpolation systems for propositional resolution. The interested reader may find in [15] an approach to the interpolation of non-ground Γ-refutations.

The state of the art on ground interpolation can be advanced by giving interpolation systems that produce better interpolants than the existing ones, in terms of strength, length, or other features [37,39,78,1,80], or by giving interpolation systems for quantifier-free fragments of theories that do not have them. The latter include nonlinear arithmetic, where interpolation is relevant to hybrid system verification, theories of data structures beyond arrays, sets, and multisets [53], where interpolation is relevant to the verification of heap-manipulating programs, and the theory of bitvectors [86,46], where interpolation is relevant to applications in hardware verification and security. For example, it is not known how to extract theory interpolants from the propositional proofs produced by solving bitvector problems by bit-blasting.

Acknowledgements This work started when also the second author was with the Dipartimento di Informatica of the Università degli Studi di Verona. Early versions of parts of this work were presented at a meeting of COST Action No. IC0109 "Rich-model toolkit: an infrastructure for reliable computer systems" held in Turin, Italy, in October 2011, and at a Z3 Special Interest Group Meeting in Cambridge, UK, in November 2011. We thank the organizers of those meetings, Leonardo de Moura, Mnacho Echenim, and Madan Musuvathi for their comments, and the anonymous reviewers for their suggestions.

References

1. Aws Albarghouthi and Kenneth L. McMillan. Beautiful interpolants. In Natasha Sharygina and Helmut Veith, editors, *Proceedings of the 25th Conference on Computer Aided Verification (CAV)*, volume 8044 of *Lecture Notes in Computer Science*, pages 313–329, Berlin, 2013. Springer.
2. Francesco Alberti, Roberto Bruttomesso, Silvio Ghilardi, Silvio Ranise, and Natasha Sharygina. Lazy abstraction with interpolants for arrays. In Nikolaj Bjørner and Andrei Voronkov, editors, *Proceedings of the 18th Conference on Logic, Programming and Automated Reasoning (LPAR)*, volume 7180 of *Lecture Notes in Artificial Intelligence*, pages 46–61, Berlin, 2012. Springer.
3. Alessandro Armando, Maria Paola Bonacina, Silvio Ranise, and Stephan Schulz. New results on rewrite-based satisfiability procedures. *ACM Transactions on Computational Logic*, 10(1):129–179, 2009.
4. Alessandro Armando, Silvio Ranise, and Michaël Rusinowitch. A rewriting approach to satisfiability procedures. *Information and Computation*, 183(2):140–164, 2003.
5. Clark W. Barrett, David L. Dill, and Aaron Stump. Checking satisfiability of first-order formulas by incremental translation to SAT. In Kim G. Larsen and Ed Brinksma, editors, *Proceedings of the 14th Conference on Computer Aided Verification (CAV)*, volume 2404 of *Lecture Notes in Computer Science*, pages 236–249, Berlin, 2002. Springer.
6. Clark W. Barrett, David L. Dill, and Aaron Stump. A generalization of Shostak's method for combining decision procedures. In Alessandro Armando, editor, *Proceedings of the 4th Workshop on Frontiers of Combining Systems (FroCoS)*, volume 2309 of *Lecture Notes in Computer Science*, Berlin, 2002. Springer.
7. Dirk Beyer, Damien Zufferey, and Rupak Majumdar. CSIsat: Interpolation for LA+EUF. In Aarti Gupta and Sharad Malik, editors, *Proceedings of the 20th Conference on Computer Aided Verification (CAV)*, volume 5123 of *Lecture Notes in Computer Science*, pages 304–308, Berlin, 2008. Springer.
8. Maria Paola Bonacina. On theorem proving for program checking – historical perspective and recent developments. In Maribel Fernández, editor, *Proceedings of the 12th International Symposium on Principles and Practice of Declarative Programming (PPDP)*, pages 1–11, New York, 2010. ACM.
9. Maria Paola Bonacina and Nachum Dershowitz. Abstract canonical inference. *ACM Transactions on Computational Logic*, 8(1):180–208, 2007.
10. Maria Paola Bonacina and Mnacho Echenim. Rewrite-based satisfiability procedures for recursive data structures. In Byron Cook and Roberto Sebastiani, editors, *Proceedings of the 4th Workshop on Pragmatics of Decision Procedures in Automated Reasoning (PDPAR 2006)*, volume 174(8) of *Electronic Notes in Theoretical Computer Science*, pages 55–70, Amsterdam, 2007. Elsevier.

11. Maria Paola Bonacina and Mnacho Echenim. On variable-inactivity and polynomial $\mathscr{T}$-satisfiability procedures. *Journal of Logic and Computation*, 18(1):77–96, 2008.
12. Maria Paola Bonacina and Jieh Hsiang. On the modelling of search in theorem proving – towards a theory of strategy analysis. *Information and Computation*, 147:171–208, 1998.
13. Maria Paola Bonacina and Moa Johansson. On interpolation in decision procedures. In Kai Brünnler and George Metcalfe, editors, *Proceedings of the 20th International Conference on Analytic Tableaux and Related Methods (TABLEAUX)*, volume 6793 of *Lecture Notes in Artificial Intelligence*, pages 1–16, Berlin, 2011. Springer.
14. Maria Paola Bonacina and Moa Johansson. Towards interpolation in an SMT solver with integrated superposition. In Shuvendu Lahiri and Sanjit A. Seshia, editors, *Notes of the 9th International Workshop on Satisfiability Modulo Theories (SMT)*, number UCB/EECS-2011-80 in Technical Reports, pages 9–18, Berkeley, CA, 2011. Department of EECS, University of California at Berkeley.
15. Maria Paola Bonacina and Moa Johansson. On interpolation in automated theorem proving. *Journal of Automated Reasoning*, 54(1):69–97, January 2015.
16. Aaron R. Bradley and Zohar Manna. *The Calculus of Computation – Decision Procedures with Applications to Verification*. Springer, Berlin, 2007.
17. Angelo Brillout, Daniel Kroening, Philipp Rümmer, and Thomas Wahl. An interpolating sequent calculus for quantifier-free Presburger arithmetic. In Jürgen Giesl and Reiner Hähnle, editors, *Proceedings of the 5th International Joint Conference on Automated Reasoning (IJCAR)*, volume 6173 of *Lecture Notes in Artificial Intelligence*, pages 384–399, Berlin, 2010. Springer.
18. Angelo Brillout, Daniel Kroening, Philipp Rümmer, and Thomas Wahl. Program verification via Craig interpolation for Presburger arithmetic with arrays. Notes of the 6th International Verification Workshop (VERIFY), 2010. Available at `http://www.philipp.ruemmer.org/`.
19. Angelo Brillout, Daniel Kroening, Philipp Rümmer, and Thomas Wahl. Beyond quantifier-free interpolation in extensions of Presburger arithmetic. In Ranjit Jhala and David Schmidt, editors, *Proceedings of the 12th International Conference on Verification, Model Checking and Abstract Interpretation (VMCAI)*, volume 6538 of *Lecture Notes in Computer Science*, pages 88–102, Berlin, 2011. Springer.
20. Roberto Bruttomesso, Silvio Ghilardi, and Silvio Ranise. A combination of rewriting and constraint solving for the quantifier-free interpolation of arrays with integer difference constraints. In Cesare Tinelli and Viorica Sofronie-Stokkermans, editors, *Proceedings of the 8th Symposium on Frontiers of Combining Systems (FroCoS)*, volume 6989 of *Lecture Notes in Artificial Intelligence*, pages 103–118, Berlin, 2011. Springer.
21. Roberto Bruttomesso, Silvio Ghilardi, and Silvio Ranise. From strong amalgamability to modularity of quantifier-free interpolation. In Bernhard Gramlich, Dale Miller, and Ulrike Sattler, editors, *Proceedings of the 6th International Joint Conference on Automated Reasoning (IJCAR)*, volume 7364 of *Lecture Notes in Artificial Intelligence*, pages 118–133, Berlin, 2012. Springer.
22. Roberto Bruttomesso, Silvio Ghilardi, and Silvio Ranise. Quantifier-free interpolation for a theory of arrays. *Logical Methods in Computer Science*, 8(2), 2012.
23. Roberto Bruttomesso, Silvio Ghilardi, and Silvio Ranise. Quantifier-free interpolation in combinations of equality interpolating theories. *ACM Transactions on Computational Logic*, 15(1), 2014.
24. Roberto Bruttomesso, Simone Fulvio Rollini, Natasha Sharygina, and Aliaksei Tsitovich. Flexible interpolation generation in satisfiability modulo theories. In *Proceedings of the 14th International Conference on Computer-Aided Design (ICCAD)*, pages 770–777, Los Alamitos, 2010. IEEE.
25. Jürgen Christ and Jochen Hoenicke. Instantiation-based interpolation for quantified formulae. Notes of the 8th International Workshop on Satisfiability Modulo Theories (SMT), 2010.
26. Alessandro Cimatti, Alberto Griggio, and Roberto Sebastiani. Efficient generation of Craig interpolants in satisfiability modulo theories. *ACM Transactions on Computational Logic*, 12(1):Article 7, 2010.
27. Stephen A. Cook and Robert A. Reckhow. The relative efficiency of propositional proof systems. *Journal of Symbolic Logic*, 44(1):36–50, 1979.
28. William Craig. Linear reasoning. A new form of the Herbrand-Gentzen theorem. *Journal of Symbolic Logic*, 22(3):250–268, 1957.
29. Martin Davis, George Logemann, and Donald Loveland. A machine program for theorem-proving. *Communications of the ACM*, 5(7):394–397, 1962.
30. Martin Davis and Hilary Putnam. A computing procedure for quantification theory. *Journal of the ACM*, 7:201–215, 1960.
31. Leonardo de Moura and Nikolaj Bjørner. Efficient E-matching for SMT-solvers. In Frank Pfenning, editor, *Proceedings of the 21st Conference on Automated Deduction (CADE)*, volume 4603 of *Lecture Notes in Artificial Intelligence*, pages 183–198, Berlin, 2007. Springer.

32. Leonardo de Moura and Nikolaj Bjørner. Bugs, moles and skeletons: Symbolic reasoning for software development. In Jürgen Giesl and Reiner Hähnle, editors, *Proceedings of the 5th International Joint Conference on Automated Reasoning (IJCAR)*, volume 6173 of *Lecture Notes in Artificial Intelligence*, pages 400–411, Berlin, 2010. Springer.
33. Leonardo de Moura and Nikolaj Bjørner. Satisfiability modulo theories: introduction and applications. *Communications of the ACM*, 54(9):69–77, 2011.
34. Nachum Dershowitz and David A. Plaisted. Rewriting. In Alan Robinson and Andrei Voronkov, editors, *Handbook of Automated Reasoning*, volume 1, pages 535–610. Elsevier, Amsterdam, 2001.
35. David L. Detlefs, Greg Nelson, and James B. Saxe. Simplify: a theorem prover for program checking. *Journal of the ACM*, 52(3):365–473, 2005.
36. Vijay D'Silva. Propositional interpolation and abstract interpretation. In Andrew D. Gordon, editor, *Proceedings of the 19th European Symposium on Programming (ESOP)*, volume 6012 of *Lecture Notes in Computer Science*, pages 185–204, Berlin, 2010. Springer.
37. Vijay D'Silva, Daniel Kroening, Mitra Purandare, and Georg Weissenbacher. Interpolant strength. In Gilles Barthe and Manuel V. Hermenegildo, editors, *Proceedings of the 11th International Conference on Verification, Model Checking, and Abstract Interpretation (VMCAI)*, volume 5944 of *Lecture Notes in Computer Science*, pages 129–145, Berlin, 2010. Springer.
38. Melvin Fitting. *First-order Logic and Automated Theorem Proving*. Springer, Berlin, 1996.
39. Alexander Fuchs, Amit Goel, Jim Grundy, Sava Krstić, and Cesare Tinelli. Ground interpolation for the theory of equality. *Logical Methods in Computer Science*, 8(1), 2012.
40. Jean Gallier. *Logic for Computer Science – Foundations of Automatic Theorem Proving*. John Wiley & Sons, New York, 1987.
41. Harald Ganzinger, Viorica Sofronie-Stokkermans, and Uwe Waldmann. Modular proof systems for partial functions with Evans equality. *Information and Computation*, 240(10):1453–1492, 2006.
42. Yeting Ge, Clark Barrett, and Cesare Tinelli. Solving quantified verification conditions using satisfiability modulo theories. In Frank Pfenning, editor, *Proceedings of the 21st Conference on Automated Deduction (CADE)*, volume 4603 of *Lecture Notes in Artificial Intelligence*, pages 167–182, Berlin, 2007. Springer.
43. Yeting Ge and Leonardo de Moura. Complete instantiation for quantified formulas in satisfiability modulo theories. In Ahmed Bouajjani and Oded Maler, editors, *Proceedings of the 21st Conference on Computer Aided Verification (CAV)*, volume 5643 of *Lecture Notes in Computer Science*, pages 306–320, Berlin, 2009. Springer.
44. Robert Givan and David McAllester. New results on local inference relations. In Bernhard Nebel, Charles Rich, and William R. Swartout, editors, *Proceedings of the 3rd International Conference on Principles of Knowledge Representation and Reasoning (KR)*, pages 403–412. Morgan Kaufmann, 1992.
45. Amit Goel, Sava Krstić, and Cesare Tinelli. Ground interpolation for combined theories. In Renate Schmidt, editor, *Proceedings of the 22nd Conference on Automated Deduction (CADE)*, volume 5663 of *Lecture Notes in Artificial Intelligence*, pages 183–198, Berlin, 2009. Springer.
46. Alberto Griggio. Effective word-level interpolation for software verification. In Per Bjesse and Anna Slobodova, editors, *Proceedings of the 11th Conference on Formal Methods in Computer Aided Design (FMCAD)*, New York, 2011. ACM and IEEE.
47. Ashutosh Gupta, Corneliu Popeea, and Andrey Rybalchenko. Solving recursion-free Horn clauses over LI+UIF. In Hongseok Yang, editor, *Proceedings of the 9th Asian Symposium on Programming Languages and Systems (APLAS)*, volume 7078 of *Lecture Notes in Computer Science*, Berlin, 2011. Springer.
48. Thomas A. Henzinger, Ranjit Jhala, Rupak Majumdar, and Kenneth L. McMillan. Abstractions from proofs. In Xavier Leroy, editor, *Proceedings of the 31st ACM SIGACT-SIGPLAN Symposium on Principles of Programming Languages (POPL)*, pages 232–244, New York, 2004. ACM.
49. Kryštof Hoder, Laura Kovàcs, and Andrei Voronkov. Interpolation and symbol elimination in Vampire. In Jürgen Giesl and Reiner Hähnle, editors, *Proceedings of the 5th International Joint Conference on Automated Reasoning (IJCAR)*, volume 6173 of *Lecture Notes in Artificial Intelligence*, pages 188–195, Berlin, 2010. Springer.
50. Kryštof Hoder, Laura Kovàcs, and Andrei Voronkov. Playing in the grey area of proofs. In Michael Hicks, editor, *Proceedings of the 39th ACM SIGACT-SIGPLAN Symposium on Principles of Programming Languages (POPL)*, pages 259–272, New York, 2012. ACM.
51. Guoxiang Huang. Constructing Craig interpolation formulas. In Ding-Zhu Du and Ming Li, editors, *Proceedings of the 1st Annual International Conference on Computing and Combinatorics (COCOON)*, volume 959 of *Lecture Notes in Computer Science*, pages 181–190, Berlin, 1995. Springer.

52. Himanshu Jain. *Verification using satisfiability checking, predicate abstraction and Craig interpolation*. PhD thesis, School of Computer Science, Carnegie Mellon University, 2008.
53. Deepak Kapur, Rupak Majumdar, and Calogero G. Zarba. Interpolation for data structures. In Premkumar Devambu et al., editor, *Proceedings of the 14th ACM SIGSOFT Symposium on the Foundations of Software Engineering*. ACM Press, 2006.
54. Stephen C. Kleene. *Mathematical Logic*. Wiley Interscience, New York, 1967.
55. Laura Kovàcs and Andrei Voronkov. Interpolation and symbol elimination. In Renate Schmidt, editor, *Proceedings of the 22nd Conference on Automated Deduction (CADE)*, volume 5663 of *Lecture Notes in Artificial Intelligence*, pages 199–213, Berlin, 2009. Springer.
56. Jan Krajíček. Interpolation theorems, lower bounds for proof systems, and independence results for bounded arithmetic. *Journal of Symbolic Logic*, 62(2):457–486, 1997.
57. Jan Krajíček and Pavel Pudlàk. Some consequences of cryptographical conjectures for s_2^1 and EF. *Information and Computation*, 140:82–94, 1998.
58. Daniel Kroening and Georg Weissenbacher. Lifting propositional interpolants to the word-level. In Jason Baumgartner and Mary Sheeran, editors, *Proceedings of the 7th Conference on Formal Methods in Computer Aided Design (FMCAD)*, pages 85–89, New York, 2007. ACM and IEEE.
59. Daniel Kroening and Georg Weissenbacher. Interpolation-based software verification with Wolverine. In Ganesh Gopalakrishnan and Shaz Qaader, editors, *Proceedings of the 23rd Conference on Computer Aided Verification (CAV)*, volume 6806 of *Lecture Notes in Computer Science*, pages 573–578, Berlin, 2011. Springer.
60. Michel Ludwig and Uwe Waldmann. An extension of the Knuth-Bendix ordering with LPO-like properties. In Nachum Dershowitz and Andrei Voronkov, editors, *Proceedings of the 14th Conference on Logic, Programming and Automated Reasoning (LPAR)*, volume 4790 of *Lecture Notes in Artificial Intelligence*, pages 348–362, Berlin, 2007. Springer.
61. Sharad Malik and Lintao Zhang. Boolean satisfiability: from theoretical hardness to practical success. *Communications of the ACM*, 52(8):76–82, 2009.
62. João P. Marques-Silva and Karem A. Sakallah. GRASP: A new search algorithm for satisfiability. In *Proceedings of the 1996 IEEE/ACM International Conference on Computer Aided Design (ICCAD)*, pages 220–227, 1997.
63. David McAllester. Automatic recognition of tractability in inference relations. *Journal of the ACM*, 40(2):284–303, 1993.
64. Kenneth L. McMillan. Interpolation and SAT-based model checking. In *Proceedings of the 15th Conference on Computer Aided Verification (CAV)*, volume 2725 of *Lecture Notes in Computer Science*, pages 1–13, Berlin, 2003. Springer.
65. Kenneth L. McMillan. An interpolating theorem prover. *Theoretical Computer Science*, 345(1):101–121, 2005.
66. Kenneth L. McMillan. Lazy abstraction with interpolants. In Tom Ball and Robert B. Jones, editors, *Proceedings of the 18th Conference on Computer Aided Verification (CAV)*, volume 4144 of *Lecture Notes in Computer Science*, pages 123–136, Berlin, 2006. Springer.
67. Kenneth L. McMillan. Quantified invariant generation using an interpolating saturation prover. In C. R. Ramakrishnan and Jakob Rehof, editors, *Proceedings of the 14th Conference on Tools and Algorithms for Construction and Analysis of Systems (TACAS)*, volume 4963 of *Lecture Notes in Computer Science*, pages 413–427, Berlin, 2008. Springer.
68. Kenneth L. McMillan. Lazy annotation for program testing and verification. In *Proceedings of the 22nd Conference on Computer Aided Verification (CAV)*, volume 6174 of *Lecture Notes in Computer Science*, pages 104–118, Berlin, 2010. Springer.
69. Kenneth L. McMillan. Interpolants from Z3 proofs. In Per Bjesse and Anna Slobodova, editors, *Proceedings of the 11th Conference on Formal Methods in Computer Aided Design (FMCAD)*, New York, 2011. ACM and IEEE.
70. Michal Moskał. Fx7 or in software, it is all about quantifiers. System Descriptions at the Satisfiability Modulo Theories Competition (SMT-COMP), 2007. Available at `http://research.microsoft.com/en-us/um/people/moskal/`.
71. Matthew W. Moskewicz, Conor F. Madigan, Ying Zhao, Lintao Zhang, and Sharad Malik. Chaff: Engineering an efficient SAT solver. In David Blaauw and Luciano Lavagno, editors, *Proceedings of the 39th Design Automation Conference (DAC)*, pages 530–535, 2001.
72. Daniele Mundici. Complexity of Craig's interpolation. *Fundamenta Informaticae*, 5:261–278, 1982.
73. Greg Nelson. *Techniques for Program Verification*. PhD thesis, Stanford University, 1979. A revised version was published as Xerox PARC Computer Science Laboratory Research Report No. CSL-81-10.

74. Greg Nelson. Combining satisfiability procedures by equality sharing. In Woodrow W. Bledsoe and Don W. Loveland, editors, *Automatic Theorem Proving: After 25 Years*, pages 201–211. American Mathematical Society, 1983.
75. Greg Nelson and Derek C. Oppen. Simplification by cooperating decision procedures. *ACM Transactions on Programming Languages and Systems*, 1(2):245–257, 1979.
76. Robert Nieuwenhuis, Albert Oliveras, and Cesare Tinelli. 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.
77. Pavel Pudlàk. Lower bounds for resolution and cutting plane proofs and monotone computations. *Journal of Symbolic Logic*, 62(3):981–998, 1997.
78. Simone Fulvio Rollini, Ondrej Sery, and Natasha Sharygina. Leveraging interpolant strength in model checking. In Madhusudan Parthasarathy and Sanjit A. Seshia, editors, *Proceedings of the 24th Conference on Computer Aided Verification (CAV)*, volume 7358 of *Lecture Notes in Computer Science*, pages 193–209, Berlin, 2012. Springer.
79. Philipp Rümmer, Hossein Hojjat, and Viktor Kuncak. Disjunctive interpolation for Horn clause verification. In Natasha Sharygina and Helmut Veith, editors, *Proceedings of the 25th Conference on Computer Aided Verification (CAV)*, volume 8044 of *Lecture Notes in Computer Science*, pages 347–363, Berlin, 2013. Springer.
80. Philipp Rümmer and Pavle Subotić. Exploring interpolants. In Barbara Jobstmann and Sandip Ray, editors, *Proceedings of the 13th Conference on Formal Methods in Computer Aided Design (FMCAD)*. FMCAD Inc, 2013.
81. Natarajan Shankar. Automated deduction for verification. *ACM Computing Surveys*, 41(4):40–96, 2009.
82. Raymond M. Smullyan. *First-Order Logic*. Dover Publications, New York, 1995. First published by Springer in 1968.
83. Viorica Sofronie-Stokkermans. Interpolation in local theory extensions. *Logical Methods in Computer Science*, 4(4):Article 1, 2008.
84. Gaisi Takeuti. *Proof Theory*, volume 81 of *Studies in Logic*. North Holland, Amsterdam, 1975.
85. Alasdair Urquhart. The complexity of propositional proofs. *Bulletin of Symbolic Logic*, 1:425–467, 1995.
86. Georg Weissenbacher. *Program Analysis with Interpolants*. PhD thesis, Magdalen College, Oxford University, 2010.
87. Greta Yorsh and Madanlal Musuvathi. A combination method for generating interpolants. Technical Report MSR-TR-2004-108, Microsoft Research, October 2004.
88. Greta Yorsh and Madanlal Musuvathi. A combination method for generating interpolants. In Robert Nieuwenhuis, editor, *Proceedings of the 20th Conference on Automated Deduction (CADE)*, volume 3632 of *Lecture Notes in Artificial Intelligence*, pages 353–368, Berlin, 2005. Springer.
89. Lintao Zhang and Sharad Malik. Validating SAT solvers using an independent resolution-based checker: practical implementations and other applications. In *Proceedings of the Conference on Design Automation and Test in Europe (DATE)*, pages 10880–10885, Los Alamitos, 2003. IEEE.