Journal of Automated Reasoning article No.
DOI 10.1007/s10817-014-9314-0 in 54(1):69–97, January 2015

On Interpolation in Automated Theorem Proving

Maria Paola Bonacina · **Moa Johansson**

Received: 24 August 2013 / Accepted: 7 October 2014 / Published online: 19 October 2014

Abstract Given two inconsistent formulæ, a (reverse) interpolant is a formula implied by one, inconsistent with the other, and only containing symbols they share. Interpolation finds application in program analysis, verification, and synthesis, for example, towards invariant generation. An *interpolation system* takes a refutation of the inconsistent formulæ and extracts an interpolant by building it inductively from *partial interpolants*. Known interpolation systems for ground proofs use colors to track symbols. We show by examples that the color-based approach cannot handle non-ground refutations by resolution and paramodulation/superposition. We present a *two-stage approach* that works by tracking literals, computes a *provisional interpolant*, which may contain non-shared symbols, and applies *lifting* to replace non-shared constants by quantified variables. We obtain an interpolation system for non-ground refutations, and we prove that it is complete, if the only non-shared symbols in provisional interpolants are constants.

Keywords Interpolation Systems · Superposition · Resolution

1 Introduction

1.1 Motivation and Aim

Automated reasoners are increasingly embedded in tools for the analysis, verification and synthesis of programs (e.g., [58,25,6,26]). A theorem-proving technique

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

that is receiving significant attention is *interpolation*, or the operation of producing interpolants 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.

A challenging application of interpolation is *invariant generation* (e.g., [51,45, 61,48]). For instance, one assumes that a k-step unwinding of a loop does not satisfy the post-condition. Formulæ expressing this conjecture are given to a theorem prover. If the loop does satisfy the post-condition up to k-iterations, the conjecture is unsatisfiable. Interpolants of the refutation are used to guide the construction of a loop invariant [51]. The requirement that the interpolant contains only shared symbols means that it contains only symbols occurring in the loop body, and not auxiliary symbols introduced to formulate the conjecture. Thus, interpolation is related to symbol elimination [46]. It was also suggested to let the theorem prover generate formulæ from formulæ specifying the loop, and take as candidate invariants those that only use the desired subset of symbols [45,40]. In approaches based on quantifier elimination, the symbols to be eliminated are quantified, so that quantifier elimination expunges them (e.g., [43,44]). An early lead towards this application can be traced back to [17].

Invariant generation led to the study of interpolation in first-order logic with equality, for proofs produced by superposition [51,46,40] or generic inferences [46, 41]. In first-order logic with equality the proofs to be interpolated in general are not ground, and interpolants naturally contain quantifiers. For invariant generation, the capability of generating interpolants with quantifiers is an advantage, because invariants often need them (e.g., [53,5]).

In this article we focus on interpolation of refutations generated by a standard inference system Γ for first-order logic with equality, based on resolution and paramodulation/superposition. Such inference systems are at the heart of theorem provers for first-order logic with equality (e.g., [60,47,57]), and also yield decision procedures for theories relevant to program verification (e.g., [3,2,8,9]). As a byproduct, we discuss interpolation for DPLL($\Gamma + \mathscr{T}$) [13], a theorem-proving method that integrates Γ in the DPLL($\mathscr{T}$) framework for satisfiability modulo theories (SMT) [55, 26], to unite the strengths of resolution-based theorem provers, such as the automatic treatment of quantifiers, with those of DPLL($\mathscr{T}$)-based SMT-solvers, such as built-in theories and scalability of performance on large sets of very long ground clauses.

1.2 State of the Art and Problem Statement

Given a refutation, an *interpolation system* generates a *partial interpolant* for every clause, in such a way that the partial interpolant of the empty clause is a reverse interpolant of the input formulæ. This approach is *inductive*, because the interpolation system builds the partial interpolant of the conclusion from those of the premises, for each inference rule that may appear in a proof. An interpolation system is *complete* for an inference system, if for all its refutations it extracts a reverse interpolant.

In the more recent literature in automated deduction and verification, the study of interpolation for superposition began in [51], with the aim of extending to refutations

in first-order logic with equality the *color-based approach* applied to propositional resolution [49] and quantifier-free fragments of first-order theories [39,50]. In the color-based approach, given formulæ A and B to be interpolated, the interpolation system traces non-shared symbols, called first *local* (e.g., A-local and B-local), and then *colored* (e.g., A-colored and B-colored), to prevent them from entering the interpolant, and identify which literals descend from A or B, to make sure that the reverse interpolant is entailed by A and inconsistent with B. Intuitively, this requires that the colors do not mix, an observation captured in [51] with the restriction to *local*, or *colored*, proofs, that is, proofs where no inference involves both A-colored and B-colored symbols. The research on interpolation of colored proofs continued in [46], where it was shown that if the ordering is *separating*, meaning that all terms made only of shared symbols are smaller than all other terms, then all *ground* refutations by superposition are colored. The notion of an ordering with this property, called *AB-oriented* ordering, appeared already in [51]. Experiments on interpolating colored proofs generated by the Vampire theorem prover were described in [40].

The restriction to colored proofs can be weakened to *colorable* proofs [38]: a proof is colorable, if it has no *AB-mixed* literals, that is, no literals with both A-colored and B-colored symbols. Proofs by propositional resolution are trivially colorable, since no new literals are generated, and ground proofs in first-order logic are like propositional proofs in this respect. If equality enters the picture, already for ground proofs, equalities where one side is A-colored and the other B-colored are problematic: the assumption of a separating ordering for ground superposition prevents precisely such AB-mixed equalities, as explained in [11,12], where we give a systematic treatment of color-based interpolation systems for ground proofs. Proof transformation techniques to make ground proofs colored or colorable were studied in [38,19,18,52,41].

While all ground refutations by superposition are colorable under a separating ordering, this is not the case in general, and we are not aware of a proof transformation mechanism that can make any non-ground refutation in first-order logic with equality colorable. In non-ground refutations, not only equality, but also substitutions (most general unifiers and matching substitutions applied in inferences) create AB-mixed literals. Thus, the problem is how to go beyond the restriction to colorable refutations, and give an interpolation system for non-ground proofs with universally quantified variables instantiated by substitutions.

1.3 Overview of Contributions

We start with counter-examples showing that it is impossible to generalize to non-ground proofs a color-based approach, not even assuming that the proof is colorable or colored. Interpolation problems where constant symbols are the only non-shared symbols suffice to obtain counter-examples.

Unknown to the above literature on color-based interpolation, an interpolation system for non-ground proofs by resolution and paramodulation was suggested much earlier in [42]. We are indebted to [42] for the idea of a *two-stage approach* to interpolation, which separates the issues of ensuring that the reverse interpolant is entailed

by A and inconsistent with B, and that it contains only shared symbols. The first stage addresses the first issue by computing inductively what we call a *provisional interpolant*, which is entailed by A and inconsistent with B, but may contain colored symbols. The second stage extracts an interpolant from the provisional interpolant.

We present a two-stage interpolation system for Γ. We give and prove complete a *provisional interpolation system*, that computes provisional interpolants for any Γ-refutation. In the second stage our interpolation system replaces colored constants with quantified variables, a technique suggested since [21] and mentioned in [50]. We called it *lifting* as in [42], while it was termed *abstraction* in [4]. We prove that this interpolation system is *complete*, if all function symbols in provisional interpolants are shared (or interpreted and therefore admissible). Function-free fragments such as those in [1,35,29,34,33,56,32] satisfy upfront this restriction. In summary, our contributions include:

- An analysis of interpolation in the presence of substitutions, showing how approaches that suffice at the ground level cannot be generalized to non-ground proofs;
- A new complete provisional interpolation system for a standard inference system Γ for first-order logic with equality, based on resolution and paramodulation/superposition;
- An interpolation system for Γ obtained by combining our provisional interpolation system with the classical technique of replacing colored constants with quantified variables, and a proof that such an interpolation system is complete, if the only colored symbols in provisional interpolants are constants;
- A discussion of the application of the two-stage approach to interpolating refutations by DPLL($\Gamma+\mathscr{T}$).

Our provisional interpolation system significantly clarifies, simplifies, and makes more transparent its ancestor in [42]. To the best of our knowledge, the only paper that cited [42] before us, was [30], where [42] was cited only for the interpolation of proofs by propositional resolution, the topic of [30]. We consider the rediscovery of [42] as an additional scholarly contribution of this article. In what follows, Section 2 introduces basic concepts and notations; Section 3 analyzes the difficulties with extending to non-ground proofs a color-based approach; Section 4 presents the provisional interpolation system and its completeness; Section 5 covers lifting, the interpolation system, and its completeness; Section 6 contains the comparison with related work, including a careful analysis and detailed discussion of Huang's construction; Section 7 discusses results and directions for future work. A one-page abstract of this article appeared in [7]. This work started when the second author was with the Dipartimento di Informatica of the Università degli Studi di Verona.

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 written $\simeq$, which is symmetric, with $\bowtie$ standing for either $\simeq$ or $\not\simeq$, and $=$ reserved for identity. Set difference is denoted by $\setminus$. We use "symbol" to mean a constant, function or predicate symbol, that is, a non-variable symbol, and "variable" for variable symbol. We typically use a, b, c for constant symbols, f, g for function symbols, v, w, x, y, z for variables, with a bar (e.g., $\bar{v}$) for a tuple, l, r, s, t for terms, l, m, r for literals, and C, D for clauses. A clause C is a disjunction of literals $l_1 \vee \ldots \vee l_n$, where all variables are implicitly universally quantified, and its negation $\neg C$ is the conjunction $\neg l_1 \wedge \ldots \wedge \neg l_n$. Both C and $\neg C$ can be viewed as sets of literals. We write $c\colon C$ to say that c is the identifier of clause C in a proof and then we may use c in place of C. A term s is a subterm of a term, or literal, l, if s appears in l, s is a proper subterm, written $s \rhd l$, 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*. A substitution is a function mapping variables to terms, which is not identity for finitely many variables; it is written in the form $\sigma = \{x_1 \leftarrow t_1, \ldots, x_n \leftarrow t_n\}$, where $x_i \neq x_j$, for all i, j, $1 \leq i \neq j \leq n$, and $x_i \neq t_i$, for all i, $1 \leq i \leq n$.

2.1 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 $\Sigma_{A,B}$.

Note that 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 [20], and illustrated for instance in [59], 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 practical theorem provers work refutationally, it is useful to adopt the notion of *reverse interpolant*, thus named in [46]:

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 $\Sigma_{A,B}$.

A reverse interpolant of (A,B) is an interpolant of $(A,\neg B)$, and if I is a reverse interpolant of (A,B), then $\neg I$ is a reverse interpolant of (B,A).

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 may appear in A and B, so that the intersections $\Sigma_A \cap \Sigma_{\mathscr{T}}$, $\Sigma_B \cap \Sigma_{\mathscr{T}}$, and $\Sigma_{A,B} \cap \Sigma_{\mathscr{T}}$ are not empty in general. 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 $A,B \vdash_{\mathscr{T}} \bot$, if (i) $A \vdash_{\mathscr{T}} I$, (ii) $B,I \vdash_{\mathscr{T}} \bot$ and (iii) all uninterpreted symbols in I are in $\Sigma_{A,B}$.

Thus, a symbol that is not shared may appear in the interpolant provided it is interpreted. For most of this article the considered theory will be that of equality, built into an inference system for first-order logic with equality. Since we consider clausal refutational inference systems, we mostly write "interpolant" for "reverse interpolant," omit "theory," and assume that A and B are disjoint sets of clauses; since set is understood as conjunction, 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 terminology is widely adopted:

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. *Local* in place of colored, and *AB-common*, or *global*, or *shared*, in place of transparent, were also used.

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. 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 [51] an inference

is *local* if all its symbols are either in Σ_A or in Σ_B. The definition in [46,41] 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 is *colored* (no AB-mixed clauses) and vice versa:

Definition 6 A proof is *colored* if it contains no AB-mixed clauses; it is *colorable* if all its clauses are colorable, or, equivalently, if it contains no AB-mixed literals.

An *interpolation system* takes a refutation of $A \cup B$, and returns an interpolant of (A,B). A fundamental property of an interpolation system is *completeness*:

Definition 7 (Complete interpolation system) An interpolation system is *complete* for an inference system Γ, if for all sets of clauses A and B, such that $A \cup B$ is unsatisfiable, and for all Γ-refutations of $A \cup B$, it generates an interpolant of (A,B).

We consider next preliminaries for inference systems and their proofs.

2.2 Inference Systems and their Proof Trees

We consider resolution-based clausal refutational inference systems for first-order logic with equality, and use Γ as the name of such a system. These inference systems use a *complete simplification ordering* $\succ$ on terms and literals to restrict *expansion inferences*, that expand the existing set by adding clauses, and to define *contraction inferences*, that contract the set by removing clauses. An ordering is a simplification ordering if it is *stable* (if $s \succ t$ then $s\sigma \succ t\sigma$ for all substitutions σ), *monotone* (if $s \succ t$ then $l[s] \succ l[t]$ for all contexts l), and has the *subterm property* ($s \succ t$ whenever $s \rhd t$). These properties imply that the ordering is *well-founded* (there is no infinite decreasing chain $t_1 \succ t_2 \succ \ldots t_i \succ t_{i+1} \succ \ldots$). A complete simplification ordering is also *total* on ground terms and literals (if $s \neq t$, then either $s \succ t$ or $t \succ s$). An ordering on terms and literals is extended to clauses by multiset extension, which preserves well-foundedness (e.g., [27] for basic definitions and results about orderings).

Since our purpose is interpolation, we are interested only in inferences rules that appear in proofs. These are expansion inferences and those contraction inferences, such as simplification, that *replace* clauses by clauses: we use *generative* rules for expansion and replacement rules. Contraction inferences that merely remove clauses, such as subsumption or tautology deletion, and book-keeping rules such as merging, that removes duplicate literals, are assumed, but do not appear in proofs. The generative rules of Γ are listed in Figure 1, where in expansion what is below the single inference line is added to what is above, and in contraction what is above the double inference line is replaced by what is below. Paramodulation, superposition, and reflection build equality into the inference system; we use superposition when the literal paramodulated into is equational, and paramodulation otherwise.

Definition 8 (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.

Resolution
$$\frac{l \vee C \quad \neg l' \vee D}{(C \vee D)\sigma} \qquad \forall m \in C : l\sigma \not\preceq m\sigma;\ \forall m \in D : \neg l'\sigma \not\preceq m\sigma$$

Factoring
$$\frac{l \vee l' \vee C}{(l \vee C)\sigma} \qquad \forall m \in C : l\sigma \not\preceq m\sigma$$

Paramodulation
$$\frac{s \simeq r \vee C \quad l[s'] \vee D}{(C \vee l[r] \vee D)\sigma} \qquad s\sigma \not\preceq r\sigma;\ \forall m \in C : (s \simeq r)\sigma \not\preceq m\sigma;\ \forall m \in D : l[s']\sigma \not\preceq m\sigma$$

Superposition
$$\frac{s \simeq r \vee C \quad l[s'] \bowtie t \vee D}{(C \vee l[r] \bowtie t \vee D)\sigma} \qquad s\sigma \not\preceq r\sigma;\ \forall m \in C : (s \simeq r)\sigma \not\preceq m\sigma;\ l[s']\sigma \not\preceq t\sigma;\ \forall m \in D : (l[s'] \bowtie t)\sigma \not\preceq m\sigma$$

Reflection
$$\frac{s' \not\simeq s \vee C}{C\sigma} \qquad \forall l \in C : (s' \not\simeq s)\sigma \not\preceq l\sigma$$

Equational Factoring
$$\frac{s \simeq t \vee s' \simeq t' \vee C}{(t \not\simeq t' \vee s \simeq t' \vee C)\sigma} \qquad s\sigma \not\preceq t\sigma;\ \forall l \in \{s' \simeq t'\} \cup C : (s \simeq t)\sigma \not\preceq l\sigma$$

Simplification
$$\frac{\begin{array}{c} s \simeq r \quad D[l] \\ \hline\hline s \simeq r \quad D[r\sigma] \end{array}}{} \qquad l = s\sigma;\ s\sigma \succ r\sigma;\ D[l] \succ (s \simeq r)\sigma$$

where the substitution σ is the most general unifier (mgu) of the *literals resolved upon* l and l', in resolution and factoring, of the terms s and s' in paramodulation, superposition, reflection, and equational factoring. For paramodulation and superposition, s' is not a variable, $s \simeq r$ is the *literal paramodulated/superposed from*, $l[s']$ is the *literal paramodulated/superposed into*, and the same terminology extends to clauses.

Fig. 1 The generative rules of Γ

Every inference $S_i \vdash_\Gamma S_{i+1}$ is required to be *sound* ($S_{i+1} \subseteq Th(S_i)$) and *adequate* ($S_i \subseteq Th(S_{i+1})$), where $Th(S) = \{C \mid S \models C\}$. The set $S^* = \bigcup_{i \geq 0} S_i$ 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 $\Box$:

Definition 9 (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.

A proof tree is drawn with the root at the bottom and the leaves at the top.[1]

[1] In general, it is a rooted graph, called *ancestor-graph* [10], but it can be put in the form of a tree by allowing different vertices to have the same clause as label.

3 The Challenge of Interpolating Non-Ground Proofs

Ground Γ-refutations can be interpolated by an approach that we call *color-based* and cover systematically in [12]: an interpolation system for ground Γ-refutations is defined inductively with base cases for input clauses belonging to A or B, and inductive cases that distinguish whether the literal resolved upon, or superposed from, is A-colored, B-colored, or transparent. AB-mixed literals are excluded because it is sufficient to assume a *separating ordering* (one where transparent terms are smaller than non-transparent ones) to ensure that ground proofs are colorable.

In this section we show by examples that a color-based approach does not extend to non-ground proofs. Non-ground inferences instantiate literals by most general unifiers (mgu's) and matching substitutions. Substitutions may insert Σ_B-symbols in Σ_A-literals and vice versa, easily generating AB-mixed literals. Assume that predicate symbol P and function symbols f and g are transparent, and constant symbols a and b are A-colored and B-colored, respectively. Superposition of $g(y,b) \simeq y$ into $f(g(a,x),x) \simeq f(x,a)$, with mgu $\{y \leftarrow a, x \leftarrow b\}$, generates $f(a,b) \simeq f(b,a)$, where both sides are AB-mixed. This inference is compatible with a separating ordering, and shows that a separating ordering does not guarantee that the proof is colorable in the general case.

One might renounce completeness and restrict attention to colorable non-ground Γ-refutations. Consider resolution between $\neg P(x,b) \vee C$ and $P(a,y) \vee D$, where x does not appear in C, y does not appear in D, and C and D are colorable. Resolvent $(C \vee D)\sigma$, with $\sigma = \{x \leftarrow a, y \leftarrow b\}$, is colorable, because $(C \vee D)\sigma = C \vee D$, since x does not occur in D and y does not occur in C, as each clause has its own variables. However, the *instances* of the literals resolved upon, $\neg P(a,b)$ and $P(a,b)$, are AB-mixed. Thus, the classification of literals as either A-colored, B-colored, or transparent does not suffice for non-ground proofs, even if they happen to be colorable.

Not only the color-based approach does not work for colorable proofs, once substitutions come into the picture: it does not work for colored, or local, proofs either. Assume that predicates L, R, and Q are transparent, and constants a_1 and a_2 are A-colored. Consider the resolution step where parents $p_1\colon L(x,a_1) \vee R(x)$ and $p_2\colon \neg L(a_2,y) \vee Q(y)$ yield resolvent $c\colon R(a_2) \vee Q(a_1)$ with mgu $\sigma = \{x \leftarrow a_2, y \leftarrow a_1\}$. This resolution step is local in the sense of [51,46,41], as only one color appears. An attempt to generalize the color-based approach to non-ground colored proofs would suggest to define the partial interpolant $PI(c)$ for such a step as $(PI(p_1) \vee PI(p_2))\sigma$. However, since σ replaces variables by colored terms, there is no guarantee that $(PI(p_1) \vee PI(p_2))\sigma$ is transparent, even if $PI(p_1)$ and $PI(p_2)$ are.

In all these problematic examples, the only colored symbols are constant symbols. Thus, the color-based approach does not extend to non-ground proofs, not even under the hypothesis that all predicate and function symbols are interpreted or transparent, and the only colored symbols are constant symbols.

These difficulties suggest that approaches based on colors, or, more generally, on classifying expressions and proofs by the presence of symbols from $\Sigma_A \setminus \Sigma_{A,B}$, $\Sigma_B \setminus \Sigma_{A,B}$, and $\Sigma_{A,B}$, do not extend beyond the ground level. In essence, an interpolation system needs to determine whether a literal l should be added to the interpolant. This requires (1) to detect whether l comes from the A side or the B side of the proof,

which is relevant to ensure that the interpolant is implied by A and inconsistent with B, and (2) to check that l is made of shared symbols. Approaches based on colors look at the symbols in l to achieve (1) and (2) simultaneously. This is impossible when substitutions mix the symbols and AB-mixed literals become unavoidable, because the presence of Σ_A-colored or Σ_B-colored symbols cannot be used to record where literals come from.

4 A Two-stage Approach to Interpolation

We design a *two-stage approach* that separates the issues (1) and (2) of the above analysis. In the first stage, the interpolation system is only concerned with making sure that the interpolant is implied by A and inconsistent with B. It uses *labels*, **A** and **B**, to keep track of where a literal in the proof comes from. The resulting interpolant is a *provisional interpolant*, because it may contain colored symbols. Only in the second stage is the interpolation system concerned with excluding colored symbols.

In order to assign labels to literals in refutations, we define a notion of *labeled Γ-proof tree*. Its key properties are that labels are independent of signatures and substitutions, and are attached to all literals, including AB-mixed ones. For instance, literals inherited by resolvents inherit the label of the corresponding literals in the parents, and new literals built by equational replacement inherit the label of the literal paramodulated or superposed into, even if a substitution was applied.

Definition 10 (Labeled Γ-proof tree) A *labeled Γ-proof tree* is a Γ-proof tree where literals in clauses are labeled by a label, denoted $label(l,c)$ for literal l in clause $c\colon C$, as follows:

- If $c\colon C \in A$, then for all $l \in C$, $label(l,c) = \mathbf{A}$;
- If $c\colon C \in B$, then for all $l \in C$, $label(l,c) = \mathbf{B}$;
- If $c\colon (C \vee D)\sigma$ is a resolvent of $p_1\colon l \vee C$ and $p_2\colon \neg l' \vee D$, then for all $m \in C$, $label(m\sigma,c) = label(m,p_1)$; and for all $m \in D$, $label(m\sigma,c) = label(m,p_2)$;
- If $c\colon (l \vee C)\sigma$ is a factor of $p\colon l \vee l' \vee C$, then for all $m \in C$, $label(m\sigma,c) = label(m,p)$, and

$$label(l\sigma,c) = \begin{cases} \mathbf{A} \text{ if } label(l,p) = label(l',p) = \mathbf{A}, \\ \mathbf{B} \text{ otherwise;} \end{cases}$$

- If $c\colon (C \vee l[r] \vee D)\sigma$ is generated by paramodulating $p_1\colon s \simeq r \vee C$ into $p_2\colon l[s'] \vee D$, then for all $m \in C$, $label(m\sigma,c) = label(m,p_1)$; for all $m \in D$, $label(m\sigma,c) = label(m,p_2)$; and $label(l[r]\sigma,c) = label(l[s'],p_2)$;
- If $c\colon (C \vee l[r] \bowtie t \vee D)\sigma$ is generated by superposing $p_1\colon s \simeq r \vee C$ into $p_2\colon l[s'] \bowtie t \vee D$, then for all $m \in C$, $label(m\sigma,c) = label(m,p_1)$; for all $m \in D$, $label(m\sigma,c) = label(m,p_2)$; and $label((l[r] \bowtie t)\sigma,c) = label(l[s'] \bowtie t,p_2)$;
- If $c\colon C\sigma$ is generated by reflection from $p\colon s' \not\simeq s \vee C$, then for all $m \in C$: $label(m\sigma,c) = label(m,p)$;
- If $c\colon (t \not\simeq t' \vee s \simeq t' \vee C)\sigma$ is an equational factor of $p\colon s \simeq t \vee s' \simeq t' \vee C$, then for all $m \in C$, $label(m\sigma,c) = label(m,p)$, and $label((t \not\simeq t')\sigma,c) = label((s \simeq$

$$t')\sigma, c) = \\ = \begin{cases} \mathbf{A} \text{ if } label(s \simeq t, p) = label(s' \simeq t', p) = \mathbf{A}, \\ \mathbf{B} \text{ otherwise.} \end{cases}$$

A clause generated by simplification is treated like a clause generated by paramodulation or superposition. Merging of identical literals is treated like a factoring step with empty unifier.

Note that the cases for the factoring rules treat A and B asymmetrically, favoring B (cf. Section 3 in [12] for a discussion of symmetry in interpolation systems).

Example 1 If $L(x_1, c)_{\mathbf{A}} \vee P(x_1)_{\mathbf{A}} \vee Q(x_1, y_1)_{\mathbf{A}}$ and $\neg L(c, x_2)_{\mathbf{B}} \vee P(x_2)_{\mathbf{B}} \vee R(x_2, y_2)_{\mathbf{B}}$ resolve on their first literals with mgu $\sigma = \{x_1 \leftarrow c, x_2 \leftarrow c\}$, the resolvent is $P(c)_{\mathbf{A}} \vee Q(c, y_1)_{\mathbf{A}} \vee P(c)_{\mathbf{B}} \vee R(c, y_2)_{\mathbf{B}}$, or $Q(c, y_1)_{\mathbf{A}} \vee P(c)_{\mathbf{B}} \vee R(c, y_2)_{\mathbf{B}}$ after merging.

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*. Intuitively, a partial interpolant is an interpolant *relative to a clause* in a refutation, so that a partial interpolant of the empty clause is an interpolant. Indeed, 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 in the color-based approach applies projections with respect to the signatures, or the colors, to the C in $A \wedge \neg C$ and $B \wedge \neg C$. Since our interpolation system works by tracking literals by labels, rather than symbols, or colors, we give a notion of projection based on labels (cf. Section 2 of [12] for a comparison with the color-based definitions). We consider disjunctions of literals, because we work with clauses, and conjunctions of literals, that arise when negating clauses:

Definition 11 (Labeled projection) Let C be a disjunction (conjunction) of labeled literals. The *labeled projection* of C on label $\mathbf{X}$, denoted $C|_{\mathbf{X}}$, is the disjunction (conjunction) of literals of C that are labeled $\mathbf{X}$. By convention, if $C|_{\mathbf{X}}$ is empty, $C|_{\mathbf{X}} = \bot$, if C is a disjunction; and $C|_{\mathbf{X}} = \top$, if C is a conjunction.

Thus, $(\neg C)|_{\mathbf{X}} = \neg(C|_{\mathbf{X}})$, $(C \vee D)|_{\mathbf{X}} = C|_{\mathbf{X}} \vee D|_{\mathbf{X}}$ and $(C \wedge D)|_{\mathbf{X}} = C|_{\mathbf{X}} \wedge D|_{\mathbf{X}}$. Unlike those based on colors, labeled projections also commute with substitutions, because labels are defined in such a way that substitutions have no bearing on them:

Proposition 1 *For all occurrences of clauses $(C \vee D)\sigma$, $(C \vee l[r] \vee D)\sigma$, and $(C \vee l[r] \bowtie t \vee D)\sigma$, generated by resolution, including reflection, paramodulation, and superposition, including simplification, in a labeled Γ-proof tree, $(C \vee D)\sigma|_{\mathbf{X}} = (C|_{\mathbf{X}} \vee D|_{\mathbf{X}})\sigma$. For all occurrences of factors $(l \vee C)\sigma$ and $(t \not\simeq t' \vee s \simeq t' \vee C)\sigma$ in a labeled Γ-proof tree, $(C\sigma)|_{\mathbf{X}} = (C|_{\mathbf{X}})\sigma$.*

Proof: It follows from Definitions 10 and 11. □

The interpolation system computes first a *provisional interpolant*, which is not required to be transparent:

Definition 12 (Provisional interpolant) A formula $\hat{I}$ is a *provisional (reverse) interpolant* of formulæ A and B such that $A, B \vdash \bot$, or a provisional interpolant of (A, B), if (i) $A \vdash \hat{I}$ and (ii) $B, \hat{I} \vdash \bot$.

A provisional theory interpolant is defined analogously. *Provisional partial interpolants* also do not have to be transparent, and are defined using labeled projections:

Definition 13 (Provisional partial interpolant) A *provisional partial interpolant* $\widehat{PI}(C)$ of a clause C occurring in a refutation of $A \cup B$ is a provisional interpolant of $A \wedge \neg(C|_{\mathbf{A}})$ and $B \wedge \neg(C|_{\mathbf{B}})$.

As for partial interpolants, $\widehat{PI}(\Box)$ is a provisional interpolant of (A, B). Thus, a *provisional interpolation system* extracts from a labeled refutation a provisional interpolant, by associating to every clause a provisional partial interpolant.

Definition 14 (Complete provisional interpolation system) A provisional interpolation system is *complete* for an inference system Γ, if for all sets of clauses A and B, such that $A \cup B$ is unsatisfiable, and for all Γ-refutations of $A \cup B$, it generates a provisional interpolant of (A, B).

For completeness, one shows that for all clauses C in the refutation, $\widehat{PI}(C)$ satisfies Definition 13. We define a provisional interpolation system for Γ:

Definition 15 (Provisional interpolation system $\Gamma\hat{I}$) Let $c\colon C$ be a clause in a labeled Γ-refutation of $A \cup B$:

- If $c\colon C \in A$, then $\widehat{PI}(c) = \bot$,
- If $c\colon C \in B$, then $\widehat{PI}(c) = \top$,
- If $c\colon C$ is generated by a Γ-inference from premises p_1 and p_2, $\widehat{PI}(c)$ is defined as follows:
  - Resolution: $c\colon (C \vee D)\sigma$ generated from $p_1\colon l \vee C$ and $p_2\colon \neg l' \vee D$ with σ mgu of l and l':
    - If $label(l, p_1) = label(\neg l', p_2) = \mathbf{A}$, then $\widehat{PI}(c) = (\widehat{PI}(p_1) \vee \widehat{PI}(p_2))\sigma$,
    - If $label(l, p_1) = label(\neg l', p_2) = \mathbf{B}$, then $\widehat{PI}(c) = (\widehat{PI}(p_1) \wedge \widehat{PI}(p_2))\sigma$,
    - If $label(l, p_1) = \mathbf{A}$ and $label(\neg l', p_2) = \mathbf{B}$, then $\widehat{PI}(c) = [(l \vee \widehat{PI}(p_1)) \wedge \widehat{PI}(p_2)]\sigma$, and
    - If $label(l, p_1) = \mathbf{B}$ and $label(\neg l', p_2) = \mathbf{A}$, then $\widehat{PI}(c) = [\widehat{PI}(p_1) \wedge (\neg l' \vee \widehat{PI}(p_2))]\sigma$;
  - Factoring: $c\colon (l \vee C)\sigma$ generated from $p\colon l \vee l' \vee C$, with σ mgu of l and l':

$$\widehat{PI}(c) = \begin{cases} \widehat{PI}(p)\sigma & \text{if } label(l, p) = label(l', p), \\ (l \vee \widehat{PI}(p))\sigma & \text{otherwise;} \end{cases}$$

  - Paramodulation: $c\colon (C \vee l[r] \vee D)\sigma$ generated from $p_1\colon s \simeq r \vee C$ and $p_2\colon l[s'] \vee D$; and Superposition: $c\colon (C \vee l[r] \bowtie t \vee D)\sigma$ generated from $p_1\colon s \simeq r \vee C$ and $p_2\colon l[s'] \bowtie t \vee D$, with σ mgu of s and s':
    - If $label(s \simeq r, p_1) = label(l[s'], p_2) = \mathbf{A}$, then $\widehat{PI}(c) = (\widehat{PI}(p_1) \vee \widehat{PI}(p_2))\sigma$,
    - If $label(s \simeq r, p_1) = label(l[s'], p_2) = \mathbf{B}$, then $\widehat{PI}(c) = (\widehat{PI}(p_1) \wedge \widehat{PI}(p_2))\sigma$,

  - If $label(s \simeq r, p_1) = \mathbf{A}$ and $label(l[s'], p_2) = \mathbf{B}$, then $\widehat{PI}(c) = [(s \simeq r \vee \widehat{PI}(p_1)) \wedge \widehat{PI}(p_2)]\sigma$,
  - If $label(s \simeq r, p_1) = \mathbf{B}$ and $label(l[s'], p_2) = \mathbf{A}$, then $\widehat{PI}(c) = [\widehat{PI}(p_1) \wedge (s \not\simeq r \vee \widehat{PI}(p_2))]\sigma$;
- Reflection: $c\colon C\sigma$ generated from $p\colon s' \not\simeq s \vee C$, with σ mgu of s and s': $\widehat{PI}(c) = \widehat{PI}(p)\sigma$;
- Equational factoring: $c\colon (t \not\simeq t' \vee s \simeq t' \vee C)\sigma$ generated from $p\colon s \simeq t \vee s' \simeq t' \vee C$, with σ mgu of s and s':
  - If $label(s \simeq t, p) = label(s' \simeq t', p)$, then $\widehat{PI}(c) = \widehat{PI}(p)\sigma$,
  - If $label(s \simeq t, p) = \mathbf{A}$ and $label(s' \simeq t', p) = \mathbf{B}$, then $\widehat{PI}(c) = (s \simeq t \vee \widehat{PI}(p))\sigma$,
  - If $label(s \simeq t, p) = \mathbf{B}$ and $label(s' \simeq t', p) = \mathbf{A}$, then $\widehat{PI}(c) = (s' \simeq t' \vee \widehat{PI}(p))\sigma$.

Simplification is treated as a special case of paramodulation and superposition, where C is empty and σ is a matching substitution that modifies only variables in $s \simeq r$ and applies only to term r in the resulting clause. In most cases, $\Gamma\hat{I}$ builds provisional interpolants by adding **A**-labeled literals that were resolved upon, factorized with, or paramodulated into **B**-labeled literals, and applying the most general unifier. The exception is the addition of the negation of those **B**-labeled literals that were paramodulated into **A**-labeled literals. Intuitively, the added literals represent a sort of communication interface between the A side and the B side of the problem. By using Proposition 1, we show that $\Gamma\hat{I}$ is complete:

Theorem 2 *$\Gamma\hat{I}$ is a complete provisional interpolation system for Γ.*

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

1. $A \wedge \neg(C|_{\mathbf{A}}) \vdash \widehat{PI}(c)$ or, equivalently, $A \vdash C|_{\mathbf{A}} \vee \widehat{PI}(c)$ and
2. $B \wedge \neg(C|_{\mathbf{B}}) \wedge \widehat{PI}(c) \vdash \bot$ or, equivalently, $B \wedge \widehat{PI}(c) \vdash C|_{\mathbf{B}}$.

The proof is by induction on the structure of the refutation.
Base cases:

- If $c\colon C \in A$, we have $\widehat{PI}(c) = \bot$, $C|_{\mathbf{A}} = C$, and $C|_{\mathbf{B}} = \bot$, so that (1) reduces to $A \vdash C$, which holds since $C \in A$, and (2) to $\bot \vdash \bot$, which is trivial.
- If $c\colon C \in B$, we have $\widehat{PI}(c) = \top$, $C|_{\mathbf{A}} = \bot$ and $C|_{\mathbf{B}} = C$, so that (1) reduces to $A \vdash \top$, which holds trivially, and (2) to $B \vdash C$, which holds since $C \in B$.

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

1. $A \wedge \neg(p_k|_{\mathbf{A}}) \vdash \widehat{PI}(p_k)$ or, equivalently, $A \vdash p_k|_{\mathbf{A}} \vee \widehat{PI}(p_k)$
2. $B \wedge \neg(p_k|_{\mathbf{B}}) \wedge \widehat{PI}(p_k) \vdash \bot$ or, equivalently, $B \wedge \widehat{PI}(p_k) \vdash p_k|_{\mathbf{B}}$.

Inductive cases:
Resolution: $c\colon (C \vee D)\sigma$ is generated from $p_1\colon (l \vee C)$ and $p_2\colon (\neg l' \vee D)$, with σ mgu of l and l'. There are four cases:

- $label(l, p_1) = label(\neg l', p_2) = \mathbf{A}$; then $(l \vee C)|_{\mathbf{A}} = l \vee C|_{\mathbf{A}}$, $(\neg l' \vee D)|_{\mathbf{A}} = \neg l' \vee D|_{\mathbf{A}}$, $(l \vee C)|_{\mathbf{B}} = C|_{\mathbf{B}}$ and $(\neg l' \vee D)|_{\mathbf{B}} = D|_{\mathbf{B}}$:

1. $A \vdash (C \vee D)\sigma|_{\mathbf{A}} \vee (\widehat{PI}(p_1) \vee \widehat{PI}(p_2))\sigma$
   From inductive hypothesis (1) we have $A \vdash l \vee C|_{\mathbf{A}} \vee \widehat{PI}(p_1)$ and $A \vdash \neg l' \vee D|_{\mathbf{A}} \vee \widehat{PI}(p_2)$. A resolution step yields $A \vdash (C|_{\mathbf{A}} \vee \widehat{PI}(p_1) \vee D|_{\mathbf{A}} \vee \widehat{PI}(p_2))\sigma$, which gives the required by reordering of literals and applying Proposition 1.
2. $B \wedge (\widehat{PI}(p_1) \vee \widehat{PI}(p_2))\sigma \vdash (C \vee D)\sigma|_{\mathbf{B}}$
   From inductive hypothesis (2) we have $B \wedge \widehat{PI}(p_1) \vdash C|_{\mathbf{B}}$ and $B \wedge \widehat{PI}(p_2) \vdash D|_{\mathbf{B}}$ whence the inductive conclusion follows with another application of Proposition 1. For brevity, in all following cases Proposition 1 will be applied silently.

– $label(l, p_1) = label(\neg l', p_2) = \mathbf{B}$; then $(l \vee C)|_{\mathbf{A}} = C|_{\mathbf{A}}$, $(\neg l' \vee D)|_{\mathbf{A}} = D|_{\mathbf{A}}$, $(l \vee C)|_{\mathbf{B}} = l \vee C|_{\mathbf{B}}$ and $(\neg l' \vee D)|_{\mathbf{B}} = \neg l' \vee D|_{\mathbf{B}}$:
  1. $A \vdash (C \vee D)\sigma|_{\mathbf{A}} \vee (\widehat{PI}(p_1) \wedge \widehat{PI}(p_2))\sigma$ is equivalent to
     $A \vdash C\sigma|_{\mathbf{A}} \vee D\sigma|_{\mathbf{A}} \vee (\widehat{PI}(p_1) \wedge \widehat{PI}(p_2))\sigma$ or
     $A \vdash (C\sigma|_{\mathbf{A}} \vee D\sigma|_{\mathbf{A}} \vee \widehat{PI}(p_1)\sigma) \wedge (C\sigma|_{\mathbf{A}} \vee D\sigma|_{\mathbf{A}} \vee \widehat{PI}(p_2)\sigma)$.
     From inductive hypothesis (1) we have $A \vdash C|_{\mathbf{A}} \vee \widehat{PI}(p_1)$ and $A \vdash D|_{\mathbf{A}} \vee \widehat{PI}(p_2)$, whence $A \vdash (C|_{\mathbf{A}} \vee \widehat{PI}(p_1)) \wedge (D|_{\mathbf{A}} \vee \widehat{PI}(p_2))$ which implies the required.
  2. $B \wedge (\widehat{PI}(p_1) \wedge \widehat{PI}(p_2))\sigma \vdash (C \vee D)\sigma|_{\mathbf{B}}$
     From inductive hypothesis (2) we have $B \wedge \widehat{PI}(p_1) \vdash l \vee C|_{\mathbf{B}}$ and $B \wedge \widehat{PI}(p_2) \vdash \neg l' \vee D|_{\mathbf{B}}$, or, equivalently, $B \vdash \neg\widehat{PI}(p_1) \vee l \vee C|_{\mathbf{B}}$ and $B \vdash \neg\widehat{PI}(p_2) \vee \neg l' \vee D|_{\mathbf{B}}$; by resolution these give $B \vdash (\neg\widehat{PI}(p_1) \vee C|_{\mathbf{B}} \vee \neg\widehat{PI}(p_2) \vee D|_{\mathbf{B}})\sigma$ or $B \wedge (\widehat{PI}(p_1) \wedge \widehat{PI}(p_2))\sigma \vdash (C|_{\mathbf{B}} \vee D|_{\mathbf{B}})\sigma$. The required thus follows.

– $label(l, p_1) = \mathbf{A}$ and $label(\neg l', p_2) = \mathbf{B}$; then $(l \vee C)|_{\mathbf{A}} = l \vee C|_{\mathbf{A}}$, $(\neg l' \vee D)|_{\mathbf{A}} = D|_{\mathbf{A}}$, $(l \vee C)|_{\mathbf{B}} = C|_{\mathbf{B}}$ and $(\neg l' \vee D)|_{\mathbf{B}} = \neg l' \vee D|_{\mathbf{B}}$:
  1. $A \vdash (C \vee D)\sigma|_{\mathbf{A}} \vee [(l \vee \widehat{PI}(p_1)) \wedge \widehat{PI}(p_2)]\sigma$ is equivalent to
     $A \vdash [C|_{\mathbf{A}} \vee D|_{\mathbf{A}} \vee l \vee \widehat{PI}(p_1)]\sigma \wedge [C|_{\mathbf{A}} \vee D|_{\mathbf{A}} \vee \widehat{PI}(p_2)]\sigma$.
     From inductive hypothesis (1) we have $A \vdash l \vee C|_{\mathbf{A}} \vee \widehat{PI}(p_1)$ and $A \vdash D|_{\mathbf{A}} \vee \widehat{PI}(p_2)$, which together imply the required.
  2. $B \wedge [(l \vee \widehat{PI}(p_1)) \wedge \widehat{PI}(p_2)]\sigma \vdash (C \vee D)\sigma|_{\mathbf{B}}$
     By case analysis on $(l \vee \widehat{PI}(p_1))\sigma$:
     (a) If $\widehat{PI}(p_1)\sigma$ holds, it suffices to establish $B \wedge (\widehat{PI}(p_1) \wedge \widehat{PI}(p_2))\sigma \vdash (C \vee D)\sigma|_{\mathbf{B}}$. From inductive hypothesis (2) we have $B \wedge \widehat{PI}(p_1) \vdash C|_{\mathbf{B}}$, which proves it.
     (b) If $l\sigma$ holds, it suffices to establish $B \wedge (l \wedge \widehat{PI}(p_2))\sigma \vdash (C \vee D)\sigma|_{\mathbf{B}}$, which is equivalent to $B \wedge \widehat{PI}(p_2)\sigma \vdash \neg l\sigma \vee C\sigma|_{\mathbf{B}} \vee D\sigma|_{\mathbf{B}}$. From inductive hypothesis (2) we have $B \wedge \widehat{PI}(p_2) \vdash \neg l' \vee D|_{\mathbf{B}}$ which proves our target since $l'\sigma = l\sigma$.

– $label(l, p_1) = \mathbf{B}$ and $label(\neg l', p_2) = \mathbf{A}$; then $(l \vee C)|_{\mathbf{A}} = C|_{\mathbf{A}}$, $(\neg l' \vee D)|_{\mathbf{A}} = \neg l' \vee D|_{\mathbf{A}}$, $(l \vee C)|_{\mathbf{B}} = l \vee C|_{\mathbf{B}}$ and $(\neg l' \vee D)|_{\mathbf{B}} = D|_{\mathbf{B}}$:
  1. $A \vdash (C \vee D)\sigma|_{\mathbf{A}} \vee [\widehat{PI}(p_1) \wedge (\neg l' \vee \widehat{PI}(p_2))]\sigma$ is equivalent to
     $A \vdash C|_{\mathbf{A}}\sigma \vee D|_{\mathbf{A}}\sigma \vee [\widehat{PI}(p_1) \wedge (\neg l' \vee \widehat{PI}(p_2))]\sigma$ or $A \vdash (C|_{\mathbf{A}} \vee D|_{\mathbf{A}} \vee \widehat{PI}(p_1))\sigma \wedge (C|_{\mathbf{A}} \vee D|_{\mathbf{A}} \vee \neg l' \vee \widehat{PI}(p_2))\sigma$. From inductive hypothesis (1) we have $A \vdash C|_{\mathbf{A}} \vee \widehat{PI}(p_1)$ and $A \vdash \neg l' \vee D|_{\mathbf{A}} \vee \widehat{PI}(p_2)$, which together give the required.
  2. $B \wedge [\widehat{PI}(p_1) \wedge (\neg l' \vee \widehat{PI}(p_2))]\sigma \vdash (C \vee D)\sigma|_{\mathbf{B}}$
     By case analysis on $\neg l'\sigma \vee \widehat{PI}(p_2)\sigma$:
     (a) If $\neg l'\sigma$ holds, it suffices to establish $B \wedge \widehat{PI}(p_1)\sigma \wedge \neg l'\sigma \vdash (C \vee D)\sigma|_{\mathbf{B}}$ which is equivalent to $B \wedge \widehat{PI}(p_1)\sigma \vdash l'\sigma \vee C|_{\mathbf{B}}\sigma \vee D|_{\mathbf{B}}\sigma$. From inductive

hypothesis (2) we have $B \wedge \widehat{PI}(p_1) \vdash l \vee C|_{\mathbf{B}}$ which implies the required since $l\sigma = l'\sigma$.

(b) If $\widehat{PI}(p_2)\sigma$ holds, it suffices to establish $B \wedge \widehat{PI}(p_1)\sigma \wedge \widehat{PI}(p_2)\sigma \vdash (C \vee D)\sigma|_{\mathbf{B}}$ or $B \wedge \widehat{PI}(p_1)\sigma \wedge \widehat{PI}(p_2)\sigma \vdash C|_{\mathbf{B}}\sigma \vee D|_{\mathbf{B}}\sigma$. From inductive hypothesis (2) we have $B \wedge \widehat{PI}(p_2) \vdash D|_{\mathbf{B}}$, which establishes the required.

Factoring: $c\colon (C \vee l)\sigma$ is generated from $p\colon (l \vee l' \vee C)$, with σ mgu of l and l'. There are again four cases:

- $label(l,p) = label(l',p) = \mathbf{A}$; then $(l \vee l' \vee C)|_{\mathbf{A}} = l \vee l' \vee C|_{\mathbf{A}}$, $(l \vee C)\sigma|_{\mathbf{A}} = l\sigma \vee C|_{\mathbf{A}}\sigma$, $(l \vee l' \vee C)|_{\mathbf{B}} = C|_{\mathbf{B}}$ and $(l \vee C)\sigma|_{\mathbf{B}} = C|_{\mathbf{B}}\sigma$:
  1. $A \vdash l\sigma \vee C|_{\mathbf{A}}\sigma \vee \widehat{PI}(p)\sigma$
     It follows from inductive hypothesis (1) $A \vdash l \vee l' \vee C|_{\mathbf{A}} \vee \widehat{PI}(p)$ and $l\sigma = l'\sigma$.
  2. $B \wedge \widehat{PI}(p)\sigma \vdash C|_{\mathbf{B}}\sigma$
     It follows from inductive hypothesis (2) $B \wedge \widehat{PI}(p) \vdash C|_{\mathbf{B}}$.
- $label(l,p) = label(l',p) = \mathbf{B}$; then $(l \vee l' \vee C)|_{\mathbf{A}} = C|_{\mathbf{A}}$, $(l \vee C)\sigma|_{\mathbf{A}} = C|_{\mathbf{A}}\sigma$, $(l \vee l' \vee C)|_{\mathbf{B}} = l \vee l' \vee C|_{\mathbf{B}}$ and $(l \vee C)\sigma|_{\mathbf{B}} = l\sigma \vee C|_{\mathbf{B}}\sigma$:
  1. $A \vdash C|_{\mathbf{A}}\sigma \vee \widehat{PI}(p)\sigma$
     It follows from inductive hypothesis (1) $A \vdash C|_{\mathbf{A}} \vee \widehat{PI}(p)$.
  2. $B \wedge \widehat{PI}(p)\sigma \vdash l\sigma \vee C|_{\mathbf{B}}\sigma$
     It follows from inductive hypothesis (2) $B \wedge \widehat{PI}(p) \vdash l \vee l' \vee C|_{\mathbf{B}}$ and $l\sigma = l'\sigma$.
- $label(l,p) = \mathbf{A}$ and $label(l',p) = \mathbf{B}$; then $(l \vee l' \vee C)|_{\mathbf{A}} = l \vee C|_{\mathbf{A}}$, $(l \vee C)\sigma|_{\mathbf{A}} = C|_{\mathbf{A}}\sigma$, $(l \vee l' \vee C)|_{\mathbf{B}} = l' \vee C|_{\mathbf{B}}$ and $(l \vee C)\sigma|_{\mathbf{B}} = l\sigma \vee C|_{\mathbf{B}}\sigma$:
  1. $A \vdash C|_{\mathbf{A}}\sigma \vee l\sigma \vee \widehat{PI}(p)\sigma$
     It follows from inductive hypothesis (1) $A \vdash l \vee C|_{\mathbf{A}} \vee \widehat{PI}(p)$.
  2. $B \wedge (l \vee \widehat{PI}(p))\sigma \vdash l\sigma \vee C|_{\mathbf{B}}\sigma$
     which is equivalent to $(B \wedge l\sigma) \vee (B \wedge \widehat{PI}(p)\sigma) \vdash l\sigma \vee C|_{\mathbf{B}}\sigma$.
     $B \wedge l\sigma \vdash l\sigma \vee C|_{\mathbf{B}}\sigma$ is trivial. $B \wedge \widehat{PI}(p)\sigma \vdash l\sigma \vee C|_{\mathbf{B}}\sigma$ follows from inductive hypothesis (2) $B \wedge \widehat{PI}(p) \vdash l' \vee C|_{\mathbf{B}}$ since $l'\sigma = l\sigma$.
- $label(l,p) = \mathbf{B}$ and $label(l',p) = \mathbf{A}$; then $(l \vee l' \vee C)|_{\mathbf{A}} = l' \vee C|_{\mathbf{A}}$, $(l \vee C)\sigma|_{\mathbf{A}} = C|_{\mathbf{A}}\sigma$, $(l \vee l' \vee C)|_{\mathbf{B}} = l \vee C|_{\mathbf{B}}$ and $(l \vee C)\sigma|_{\mathbf{B}} = l\sigma \vee C|_{\mathbf{B}}\sigma$:
  1. $A \vdash C|_{\mathbf{A}}\sigma \vee l\sigma \vee \widehat{PI}(p)\sigma$
     It follows from inductive hypothesis (1) $A \vdash l' \vee C|_{\mathbf{A}} \vee \widehat{PI}(p)$ because $l\sigma = l'\sigma$.
  2. $B \wedge (l \vee \widehat{PI}(p))\sigma \vdash l\sigma \vee C|_{\mathbf{B}}\sigma$
     which is equivalent to $(B \wedge l\sigma) \vee (B \wedge \widehat{PI}(p)\sigma) \vdash C|_{\mathbf{B}}\sigma \vee l\sigma$
     $B \wedge l\sigma \vdash l\sigma \vee C|_{\mathbf{B}}\sigma$ is trivial. $B \wedge \widehat{PI}(p)\sigma \vdash l\sigma \vee C|_{\mathbf{B}}\sigma$ follows from inductive hypothesis (2) $B \wedge \widehat{PI}(p) \vdash l \vee C|_{\mathbf{B}}$.

Paramodulation: $c\colon (C \vee l[r] \vee D)\sigma$ is generated from $p_1\colon (s \simeq r \vee C)$ and $p_2\colon (l[s'] \vee D)$ with mgu σ of s and s'. There are four cases:

- $label(s \simeq r, p_1) = label(l[s'], p_2) = \mathbf{A}$; then $(s \simeq r \vee C)|_{\mathbf{A}} = s \simeq r \vee C|_{\mathbf{A}}$, $(l[s'] \vee D)|_{\mathbf{A}} = l[s'] \vee D|_{\mathbf{A}}$, $(C \vee l[r] \vee D)\sigma|_{\mathbf{A}} = C|_{\mathbf{A}}\sigma \vee l[r]\sigma \vee D|_{\mathbf{A}}\sigma$, $(s \simeq r \vee C)|_{\mathbf{B}} = C|_{\mathbf{B}}$, $(l[s'] \vee D)|_{\mathbf{B}} = D|_{\mathbf{B}}$ and $(C \vee l[r] \vee D)\sigma|_{\mathbf{B}} = C|_{\mathbf{B}}\sigma \vee D|_{\mathbf{B}}\sigma$:
  1. $A \vdash C|_{\mathbf{A}}\sigma \vee l[r]\sigma \vee D|_{\mathbf{A}}\sigma \vee \widehat{PI}(p_1)\sigma \vee \widehat{PI}(p_2)\sigma$
     Inductive hypothesis (1) gives $A \vdash s \simeq r \vee C|_{\mathbf{A}} \vee \widehat{PI}(p_1)$ and $A \vdash l[s'] \vee D|_{\mathbf{A}} \vee \widehat{PI}(p_2)$. By a paramodulation step we get $A \vdash (C|_{\mathbf{A}} \vee l[r] \vee D|_{\mathbf{A}} \vee \widehat{PI}(p_1) \vee \widehat{PI}(p_2))\sigma$ from which the required follows.

2. $B \wedge (\widehat{PI}(p_1) \vee \widehat{PI}(p_2))\sigma \vdash C|_{\mathbf{B}}\sigma \vee D|_{\mathbf{B}}\sigma$
   From inductive hypothesis (2) we have that $B \wedge \widehat{PI}(p_1) \vdash C|_{\mathbf{B}}$ and $B \wedge \widehat{PI}(p_2) \vdash D|_{\mathbf{B}}$, which gives the inductive conclusion.

– $label(s \simeq r, p_1) = label(l[s'], p_2) = \mathbf{B}$; then $(s \simeq r \vee C)|_{\mathbf{A}} = C|_{\mathbf{A}}$, $(l[s'] \vee D)|_{\mathbf{A}} = D|_{\mathbf{A}}$, $(C \vee l[r] \vee D)\sigma|_{\mathbf{A}} = C|_{\mathbf{A}}\sigma \vee D|_{\mathbf{A}}\sigma$, $(s \simeq r \vee C)|_{\mathbf{B}} = s \simeq r \vee C|_{\mathbf{B}}$, $(l[s'] \vee D)|_{\mathbf{B}} = l[s'] \vee D|_{\mathbf{B}}$ and $(C \vee l[r] \vee D)\sigma|_{\mathbf{B}} = C|_{\mathbf{B}}\sigma \vee l[r]\sigma \vee D|_{\mathbf{B}}\sigma$:

1. $A \vdash C|_{\mathbf{A}}\sigma \vee D|_{\mathbf{A}}\sigma \vee (\widehat{PI}(p_1) \wedge \widehat{PI}(p_2))\sigma$ or, equivalently,
   $A \vdash (C|_{\mathbf{A}} \vee D|_{\mathbf{A}} \vee \widehat{PI}(p_1))\sigma \wedge (C|_{\mathbf{A}} \vee D|_{\mathbf{A}} \vee \widehat{PI}(p_2))\sigma$.
   By inductive hypothesis (1) we have $A \vdash C|_{\mathbf{A}} \vee \widehat{PI}(p_1)$ and $A \vdash D|_{\mathbf{A}} \vee \widehat{PI}(p_2)$, which together give the desired conclusion.
2. $B \wedge (\widehat{PI}(p_1) \wedge \widehat{PI}(p_2))\sigma \vdash C|_{\mathbf{B}}\sigma \vee l[r]\sigma \vee D|_{\mathbf{B}}\sigma$.
   By inductive hypothesis (2) we have $B \wedge \widehat{PI}(p_1) \vdash s \simeq r \vee C|_{\mathbf{B}}$ and $B \wedge \widehat{PI}(p_2) \vdash l[s'] \vee D|_{\mathbf{B}}$, or, equivalently, $B \vdash \neg\widehat{PI}(p_1) \vee s \simeq r \vee C|_{\mathbf{B}}$ and $B \vdash \neg\widehat{PI}(p_2) \vee l[s'] \vee D|_{\mathbf{B}}$, which by paramodulation produce $B \vdash (\neg\widehat{PI}(p_1) \vee \neg\widehat{PI}(p_2) \vee C|_{\mathbf{B}} \vee l[r] \vee D|_{\mathbf{B}})\sigma$, so that the inductive conclusion follows.

– $label(s \simeq r, p_1) = \mathbf{A}$ and $label(l[s'], p_2) = \mathbf{B}$; then $(s \simeq r \vee C)|_{\mathbf{A}} = s \simeq r \vee C|_{\mathbf{A}}$, $(l[s'] \vee D)|_{\mathbf{A}} = D|_{\mathbf{A}}$, $(C \vee l[r] \vee D)\sigma|_{\mathbf{A}} = C|_{\mathbf{A}}\sigma \vee D|_{\mathbf{A}}\sigma$, $(s \simeq r \vee C)|_{\mathbf{B}} = C|_{\mathbf{B}}$, $(l[s'] \vee D)|_{\mathbf{B}} = l[s'] \vee D|_{\mathbf{B}}$ and $(C \vee l[r] \vee D)\sigma|_{\mathbf{B}} = C|_{\mathbf{B}}\sigma \vee l[r]\sigma \vee D|_{\mathbf{B}}\sigma$:

1. $A \vdash C|_{\mathbf{A}}\sigma \vee D|_{\mathbf{A}}\sigma \vee [(s \simeq r \vee \widehat{PI}(p_1)) \wedge \widehat{PI}(p_2)]\sigma$ or, equivalently,
   $A \vdash (C|_{\mathbf{A}} \vee D|_{\mathbf{A}} \vee s \simeq r \vee \widehat{PI}(p_1))\sigma \wedge (C|_{\mathbf{A}} \vee D|_{\mathbf{A}} \vee \widehat{PI}(p_2))\sigma$.
   From inductive hypothesis (1) we have $A \vdash s \simeq r \vee C|_{\mathbf{A}} \vee \widehat{PI}(p_1)$ and $A \vdash D|_{\mathbf{A}} \vee \widehat{PI}(p_2)$, which together give the inductive conclusion.
2. $B \wedge [(s \simeq r \vee \widehat{PI}(p_1)) \wedge \widehat{PI}(p_2)]\sigma \vdash C|_{\mathbf{B}}\sigma \vee l[r]\sigma \vee D|_{\mathbf{B}}\sigma$
   By case analysis on $(s \simeq r)\sigma \vee \widehat{PI}(p_1)\sigma$:
   (a) If $\widehat{PI}(p_1)\sigma$ holds, it suffices to establish $B \wedge (\widehat{PI}(p_1) \wedge \widehat{PI}(p_2))\sigma \vdash C|_{\mathbf{B}}\sigma \vee l[r]\sigma \vee D|_{\mathbf{B}}\sigma$ which follows from inductive hypothesis (2) $B \wedge \widehat{PI}(p_1) \vdash C|_B$.
   (b) If $(s \simeq r)\sigma$ holds, it suffices to establish $B \wedge (s \simeq r)\sigma \wedge \widehat{PI}(p_2)\sigma \vdash C|_{\mathbf{B}}\sigma \vee l[r]\sigma \vee D|_{\mathbf{B}}\sigma$, which is equivalent to $B \wedge (s \simeq r)\sigma \wedge \widehat{PI}(p_2)\sigma \vdash C|_{\mathbf{B}}\sigma \vee l[s]\sigma \vee D|_{\mathbf{B}}\sigma$ since $(s \simeq r)\sigma$ holds. By inductive hypothesis (2) we have $B \wedge \widehat{PI}(p_2) \vdash l[s'] \vee D|_B$, which proves the required since $s\sigma = s'\sigma$.

– $label(s \simeq r, p_1) = \mathbf{B}$ and $label(l[s'], p_2) = \mathbf{A}$; then $(s \simeq r \vee C)|_{\mathbf{A}} = C|_{\mathbf{A}}$, $(l[s'] \vee D)|_{\mathbf{A}} = l[s'] \vee D|_{\mathbf{A}}$, $(C \vee l[r] \vee D)\sigma|_{\mathbf{A}} = C|_{\mathbf{A}}\sigma \vee l[r]\sigma \vee D|_{\mathbf{A}}\sigma$, $(s \simeq r \vee C)|_{\mathbf{B}} = s \simeq r \vee C|_{\mathbf{B}}$, $(l[s'] \vee D)|_{\mathbf{B}} = D|_{\mathbf{B}}$ and $(C \vee l[r] \vee D)\sigma|_{\mathbf{B}} = C|_{\mathbf{B}}\sigma \vee D|_{\mathbf{B}}\sigma$:

1. $A \wedge \neg(C \vee l[r] \vee D)\sigma|_{\mathbf{A}} \vdash [\widehat{PI}(p_1) \wedge (s \not\simeq r \vee \widehat{PI}(p_2))]\sigma$
   We need to show:
   (a) $A \wedge \neg(C \vee l[r] \vee D)\sigma|_{\mathbf{A}} \vdash \widehat{PI}(p_1)\sigma$, or, equivalently, $A \vdash C|_{\mathbf{A}}\sigma \vee l[r]\sigma \vee D|_{\mathbf{A}}\sigma \vee \widehat{PI}(p_1)\sigma$, which follows from inductive hypothesis (1) $A \vdash C|_{\mathbf{A}} \vee \widehat{PI}(p_1)$.
   (b) $A \wedge \neg(C \vee l[r] \vee D)\sigma|_{\mathbf{A}} \vdash (s \not\simeq r \vee \widehat{PI}(p_2))\sigma$ which is equivalent to $A \wedge \neg C|_{\mathbf{A}}\sigma \wedge \neg l[r]\sigma \wedge \neg D|_{\mathbf{A}}\sigma \wedge (s \simeq r)\sigma \vdash \widehat{PI}(p_2)\sigma$. By applying the equation $(s \simeq r)\sigma$, this in turn is equivalent to $A \wedge \neg C|_{\mathbf{A}}\sigma \wedge \neg l[s]\sigma \wedge \neg D|_{\mathbf{A}}\sigma \wedge (s \simeq r)\sigma \vdash \widehat{PI}(p_2)\sigma$ and finally to $A \vdash C|_{\mathbf{A}}\sigma \vee l[s]\sigma \vee D|_{\mathbf{A}}\sigma \vee (s \not\simeq r)\sigma \vee \widehat{PI}(p_2)\sigma$. By inductive hypothesis (1) we have that $A \vdash l[s'] \vee D|_{\mathbf{A}} \vee \widehat{PI}(p_2)$, which proves the required since $s\sigma = s'\sigma$.

2. $B \wedge [\widehat{PI}(p_1) \wedge (s \not\simeq r \vee \widehat{PI}(p_2))]\sigma \vdash C|_{\mathbf{B}}\sigma \vee D|_{\mathbf{B}}\sigma$.
   By case analysis on $(s \not\simeq r)\sigma \vee \widehat{PI}(p_2)\sigma$:
   (a) If $(s \not\simeq r)\sigma$ holds, it suffices to show that $B \wedge (\widehat{PI}(p_1) \wedge s \not\simeq r)\sigma \vdash C|_{\mathbf{B}}\sigma \vee D|_{\mathbf{B}}\sigma$, or, equivalently, $B \wedge \widehat{PI}(p_1)\sigma \vdash (s \simeq r)\sigma \vee C|_{\mathbf{B}}\sigma \vee D|_{\mathbf{B}}\sigma$. From inductive hypothesis (2) we have $B \wedge \widehat{PI}(p_1) \vdash s \simeq r \vee C|_{\mathbf{B}}$, which proves the required.
   (b) If $\widehat{PI}(p_2)\sigma$ holds, it suffices to show that $B \wedge (\widehat{PI}(p_1) \wedge \widehat{PI}(p_2))\sigma \vdash C|_{\mathbf{B}}\sigma \vee D|_{\mathbf{B}}\sigma$. From inductive hypothesis (2) we have $B \wedge \widehat{PI}(p_2) \vdash D|_{\mathbf{B}}$, which proves the required.

Reflection: $c\colon C\sigma$ is generated from $p\colon s' \not\simeq s \vee C$, with σ mgu of s and s'. There are two cases:

- $label(s' \not\simeq s, p) = \mathbf{A}$; then $(s' \not\simeq s \vee C)|_{\mathbf{A}} = s' \not\simeq s \vee C|_{\mathbf{A}}$ and $(s' \not\simeq s \vee C)|_{\mathbf{B}} = C|_{\mathbf{B}}$:
  1. $A \vdash C\sigma|_{\mathbf{A}} \vee \widehat{PI}(c)$
     From inductive hypothesis (1) we have $A \vdash s' \not\simeq s \vee C|_{\mathbf{A}} \vee \widehat{PI}(p)$, whence $A \vdash s'\sigma \not\simeq s\sigma \vee C\sigma|_{\mathbf{A}} \vee \widehat{PI}(p)\sigma$, and the inductive conclusion follows because $s'\sigma = s\sigma$ and $\widehat{PI}(c) = \widehat{PI}(p)\sigma$.
  2. $B \wedge \widehat{PI}(c) \vdash C\sigma|_{\mathbf{B}}$
     From inductive hypothesis (2) we have $B \wedge \widehat{PI}(p) \vdash C|_{\mathbf{B}}$ whence $B \vdash \neg\widehat{PI}(p)\sigma \vee \vdash C\sigma|_{\mathbf{B}}$, and the inductive conclusion follows from $\widehat{PI}(c) = \widehat{PI}(p)\sigma$.
- $label(s' \not\simeq s, p) = \mathbf{B}$; then $(s' \not\simeq s \vee C)|_{\mathbf{A}} = C|_{\mathbf{A}}$, and $(s' \not\simeq s \vee C)|_{\mathbf{B}} = s' \not\simeq s \vee C|_{\mathbf{B}}$: the rest is symmetric to the above.

Equational factoring: $c\colon (t \not\simeq t' \vee s \simeq t' \vee C)\sigma$ is generated from $p\colon s \simeq t \vee s' \simeq t' \vee C$ with mgu σ of s and s'. There are three cases, because the two where $s \simeq t$ and $s' \simeq t'$ have the same label can be merged:

- $label(s \simeq t, p) = label(s' \simeq t', p)$; then, by Definition 10, the same label is given to $(t \not\simeq t')\sigma$ and $(s \simeq t')\sigma$ in the factor:
  1. $A \vdash (t \not\simeq t' \vee s \simeq t' \vee C)\sigma|_{\mathbf{A}} \vee \widehat{PI}(p)\sigma$
     It follows from inductive hypothesis (1) $A \vdash (s \simeq t \vee s' \simeq t' \vee C)|_{\mathbf{A}} \vee \widehat{PI}(p)$, with a step of equational factoring, if the label of the equations is **A**, without, if it is **B**.
  2. $B \wedge \widehat{PI}(p)\sigma \vdash (t \not\simeq t' \vee s \simeq t' \vee C)\sigma|_{\mathbf{B}}$
     It follows from inductive hypothesis (2) $B \wedge \widehat{PI}(p) \vdash (s \simeq t \vee s' \simeq t' \vee C)|_{\mathbf{B}}$ through a step of equational factoring, if the label of the equations is **B**, without, if it is **A**.
- $label(s \simeq t, p) = \mathbf{A}$ and $label(s' \simeq t', p) = \mathbf{B}$; then, by Definition 10, $label((s \simeq t')\sigma, c) = label((t \not\simeq t')\sigma, c) = \mathbf{B}$, $(s \simeq t \vee s' \simeq t' \vee C)|_{\mathbf{A}} = s \simeq t \vee C|_{\mathbf{A}}$, $(s \simeq t \vee s' \simeq t' \vee C)|_{\mathbf{B}} = s' \simeq t' \vee C|_{\mathbf{B}}$, $(t \not\simeq t' \vee s \simeq t' \vee C)\sigma|_{\mathbf{A}} = C|_{\mathbf{A}}\sigma$ and $(t \not\simeq t' \vee s \simeq t' \vee C)\sigma|_{\mathbf{B}} = (t \not\simeq t')\sigma \vee (s \simeq t')\sigma \vee C|_{\mathbf{B}}\sigma$:
  1. $A \vdash C|_{\mathbf{A}}\sigma \vee (s \simeq t \vee \widehat{PI}(p))\sigma$
     It follows from inductive hypothesis (1) $A \vdash s \simeq t \vee C|_{\mathbf{A}} \vee \widehat{PI}(p)$.
  2. $B \wedge (s \simeq t \vee \widehat{PI}(p))\sigma \vdash (t \not\simeq t')\sigma \vee (s \simeq t')\sigma \vee C|_B\sigma$
     By case analysis on $(s \simeq t)\sigma \vee \widehat{PI}(p)\sigma$:
     (a) If $(s \simeq t)\sigma$ holds, it suffices to show $B \wedge (s \simeq t)\sigma \vdash (t \not\simeq t')\sigma \vee (s \simeq t')\sigma \vee C|_B\sigma$. Since $(s \simeq t)\sigma$ holds, this is equivalent to $B \wedge (s \simeq t)\sigma \vdash (t \not\simeq t')\sigma \vee (t \simeq t')\sigma \vee C|_B\sigma$, which holds trivially.

(b) If $\widehat{PI}(p)\sigma$ holds, it suffices to show $B \wedge \widehat{PI}(p)\sigma \vdash (t \not\simeq t')\sigma \vee (s \simeq t')\sigma \vee C|_B\sigma$. As $s\sigma = s'\sigma$, this follows from inductive hypothesis (2) $B \wedge \widehat{PI}(p) \vdash s' \simeq t' \vee C|_B$.

– $label(s \simeq t, p) = \mathbf{B}$ and $label(s' \simeq t', p) = \mathbf{A}$; then, by Definition 10, $label((s \simeq t')\sigma, c) = label((t \not\simeq t')\sigma, c) = \mathbf{B}$, $(s \simeq t \vee s' \simeq t' \vee C)|_{\mathbf{A}} = s' \simeq t' \vee C|_{\mathbf{A}}$, $(s \simeq t \vee s' \simeq t' \vee C)|_{\mathbf{B}} = s \simeq t \vee C|_{\mathbf{B}}$, $(t \not\simeq t' \vee s \simeq t' \vee C)\sigma|_{\mathbf{A}} = C|_{\mathbf{A}}\sigma$ and $(t \not\simeq t' \vee s \simeq t' \vee C)\sigma|_{\mathbf{B}} = (t \not\simeq t')\sigma \vee (s \simeq t')\sigma \vee C|_{\mathbf{B}}\sigma$:
  1. $A \vdash C|_{\mathbf{A}}\sigma \vee (s' \simeq t' \vee \widehat{PI}(p))\sigma$
     It follows from inductive hypothesis (1) $A \vdash s' \simeq t' \vee C|_{\mathbf{A}} \vee \widehat{PI}(p)$.
  2. $B \wedge (s' \simeq t' \vee \widehat{PI}(p))\sigma \vdash (t \not\simeq t')\sigma \vee (s \simeq t')\sigma \vee C|_B\sigma$
     By case analysis on $(s' \simeq t')\sigma \vee \widehat{PI}(p)\sigma$:
     (a) If $(s' \simeq t')\sigma$ holds, it suffices to establish $B \wedge (s' \simeq t')\sigma \vdash (t \not\simeq t')\sigma \vee (s \simeq t')\sigma \vee C|_B\sigma$. Since $s\sigma = s'\sigma$, this is equivalent to $B \wedge (s' \simeq t')\sigma \vdash (t \not\simeq t')\sigma \vee (s' \simeq t')\sigma \vee C|_B\sigma$, which holds trivially.
     (b) If $\widehat{PI}(p)\sigma$ holds, it suffices to establish $B \wedge \widehat{PI}(p)\sigma \vdash (t \not\simeq t')\sigma \vee (s \simeq t')\sigma \vee C|_B\sigma$. Consider the disjunct $(t \not\simeq t')\sigma$: if it holds, the conclusion holds; otherwise, $(t \simeq t')\sigma$ holds, and the target reduces to $B \wedge \widehat{PI}(p)\sigma \vdash (s \simeq t)\sigma \vee C|_B\sigma$, which follows from inductive hypothesis (2) $B \wedge \widehat{PI}(p) \vdash s \simeq t \vee C|_B$.

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. □

The following lemma characterizes the provisional interpolants produced by $\Gamma\hat{I}$:

Lemma 1 *A provisional interpolant $\hat{I}$ generated by $\Gamma\hat{I}$ is a formula in negation normal form, where all variables, if any, are implicitly universally quantified and all predicate symbols are either interpreted or transparent.*

Proof: By construction, the only connectives occurring in $\hat{I}$ are $\vee$, $\wedge$ and $\neg$, and $\neg$ applies only to atoms; thus, $\hat{I}$ is in negation normal form. $\Gamma\hat{I}$ constructs $\hat{I}$ by adding instances of literals resolved upon, factorized or paramodulated from. These instances are not necessarily ground, and therefore $\hat{I}$ may contain variables, that are implicitly universally quantified, because they come from clauses where all variables are implicitly universally quantified. For equational literals added to $\hat{I}$, the predicate symbol is equality which is interpreted. Non-equational literals are added if they are **A**-labeled literals that resolved upon or factorized with **B**-labeled literals. Unifiable literals have the same predicate symbol. An **A**-labeled literal and a **B**-labeled literal have the same predicate symbol only if the symbol is transparent. Thus, all literals in $\hat{I}$ have either interpreted or transparent predicate symbols. □

5 A Complete Interpolation System

In the second stage, the interpolation system applies to the provisional interpolant a transformation called *lifting*, which replaces colored constants by quantified vari-

ables. This is an idea that appeared as early as [21]. It is sufficient to obtain a transparent formula if the only colored symbols in the formula are constants:

Definition 16 (Color-flat formula) A closed formula F is *color-flat*, if its only colored symbols are constant symbols.

By Lemma 1, provisional interpolants generated by $\Gamma\hat{I}$ do not contain colored predicate symbols, and therefore they are color-flat if, in addition, all their function symbols are either interpreted or transparent.

Definition 17 (Lifting) Given a color-flat formula $F = \forall\bar{v}.G$ in prenex normal form, where the matrix G is in negation normal form, and either the prefix $\forall\bar{v}$ is empty and G is ground, or $\forall\bar{v}$ contains universal quantifiers for all variables in G, let $CC(G)$ be the set of the colored constants occurring in G. The *lifting* of F, denoted $Lift(F)$, is the formula $Lift(F) = Q\bar{x}.\forall\bar{v}.G'$, where G' is G with all occurrences of each $c \in CC(G)$ replaced by a new quantified variable x, whose quantifier is $\exists$ if c is A-colored, $\forall$ if c is B-colored, and $Q\bar{x}$ is the prefix of quantifiers thus inserted.

By Lemma 1, provisional interpolants generated by $\Gamma\hat{I}$ are in negation normal form with all variables, if any, universally quantified, and therefore they satisfy the requirements of the definition of lifting. Variables introduced by lifting are requested to be new, even if they are quantified, so that variables in the result are standardized apart. The order of quantifiers is immaterial since only constants are lifted. Next, we show that the lifting of a color-flat provisional interpolant gives an interpolant.

Lemma 2 *If $\hat{I}$ is a color-flat provisional interpolant of (A,B), then $B, Lift(\hat{I}) \vdash \bot$.*

Proof: By Definition 12, we have $B, \hat{I} \vdash \bot$. We show that $B, \hat{I} \vdash \bot$ implies $B, Lift(\hat{I}) \vdash \bot$. By way of contradiction assume that $B \wedge Lift(\hat{I})$ is consistent, with a model $M = \langle D, \Phi\rangle$, where D is the domain and Φ the interpretation function, such that $M \models B \wedge Lift(\hat{I})$. We build a model $M' = \langle D, \Phi'\rangle$ such that $M' \models B \wedge \hat{I}$. By Definition 17, $\hat{I}$ and $Lift(\hat{I})$ are in negation normal form, which means that no quantifier falls in the scope of an occurrence of negation, and therefore universal (existential) quantifiers are truly universal (existential). Let $Lift(\hat{I})^\star$ be the instance of $Lift(\hat{I})$, where the universally quantified variables introduced by lifting are replaced by the corresponding B-colored constants originally in $\hat{I}$. Clearly, $M \models Lift(\hat{I})$ implies $M \models Lift(\hat{I})^\star$. Let Φ' be identical to Φ in the interpretation of B-colored and transparent symbols. A-colored symbols do not appear in $B \wedge Lift(\hat{I})$, and therefore are not interpreted by Φ. Since $\hat{I}$ is color-flat, its only A-colored symbols are constants. Let c_a be an A-colored constant that occurs in $\hat{I}$. Since c_a is replaced by an $\exists x$ in $Lift(\hat{I})$ and $Lift(\hat{I})^\star$, let $\Phi'(c_a)$ be the element $d \in D$ used by M for x when satisfying $Lift(\hat{I})$ and $Lift(\hat{I})^\star$. Then, $M' \models B$ follows from $M \models B$, because the two interpretations are identical on B-colored and transparent symbols. $M' \models \hat{I}$ follows from $M \models Lift(\hat{I})^\star$, because the only difference between $\hat{I}$ and $Lift(\hat{I})^\star$ is that where $\hat{I}$ has an A-colored constant c_a, $Lift(\hat{I})^\star$ has an existentially quantified variable x, and M' interprets c_a exactly with the element of D that M assigns to x to satisfy $Lift(\hat{I})^\star$. In conclusion, $M' \models B \wedge \hat{I}$, a contradiction. □

The second lemma works dually on the A side, with the rôles of A and B exchanged:

Lemma 3 *If $\hat{I}$ is a color-flat provisional interpolant of (A,B), then $A \vdash Lift(\hat{I})$.*

Proof: By Definition 12, we have $A \vdash \hat{I}$, or $A \wedge \neg\hat{I} \vdash \bot$. We show that $A \wedge \neg\hat{I} \vdash \bot$ implies $A \wedge \neg Lift(\hat{I}) \vdash \bot$. By way of contradiction, assume that $A \wedge \neg Lift(\hat{I})$ is consistent, with model $M = \langle D, \Phi \rangle$, such that $M \models A \wedge \neg Lift(\hat{I})$. We build a model $M' = \langle D, \Phi' \rangle$ such that $M' \models A \wedge \neg\hat{I}$. We reduce $\neg\hat{I}$ and $\neg Lift(\hat{I})$ to negation normal form by applying De Morgan laws, which push the negation inside, turning $\exists$ into $\forall$ and $\forall$ into $\exists$: let $(\neg\hat{I})^\dagger$ and $(\neg Lift(\hat{I}))^\dagger$ denote the negation normal forms of $\neg\hat{I}$ and $\neg Lift(\hat{I})$. Since reduction to negation normal form preserves logical equivalence, $M \models \neg Lift(\hat{I})$ implies $M \models (\neg Lift(\hat{I}))^\dagger$, and proving $M' \models (\neg\hat{I})^\dagger$ suffices to prove $M' \models \neg\hat{I}$. The variables introduced by lifting to replace the A-colored constants in $\hat{I}$ are existentially quantified in $Lift(\hat{I})$ and universally quantified in $(\neg Lift(\hat{I}))^\dagger$. Let $(\neg Lift(\hat{I}))^{\dagger\star}$ be the instance of $(\neg Lift(\hat{I}))^\dagger$, where these variables are replaced by the corresponding A-colored constants originally in $\hat{I}$. Clearly, $M \models (\neg Lift(\hat{I}))^\dagger$ implies $M \models (\neg Lift(\hat{I}))^{\dagger\star}$. Let Φ' be identical to Φ in the interpretation of A-colored and transparent symbols. B-colored symbols do not appear in $A \wedge \neg Lift(\hat{I})$, and therefore are not interpreted by Φ. Since $\hat{I}$ is color-flat, the only B-colored symbols in $\neg\hat{I}$ and $(\neg\hat{I})^\dagger$ are constants. Let c_b be a B-colored constant that occurs in $\neg\hat{I}$. Since c_b is replaced by a $\forall x$ in $Lift(\hat{I})$, it is replaced by an $\exists x$ in $(\neg Lift(\hat{I}))^\dagger$. Let $\Phi'(c_b)$ be the element $d \in D$ used by M for x when satisfying $(\neg Lift(\hat{I}))^\dagger$ and $(\neg Lift(\hat{I}))^{\dagger\star}$. Then, $M' \models A$ follows from $M \models A$, because the two interpretations are identical on A-colored and transparent symbols. $M' \models (\neg\hat{I})^\dagger$ follows from $M \models (\neg Lift(\hat{I}))^{\dagger\star}$, because the only difference between $(\neg\hat{I})^\dagger$ and $(\neg Lift(\hat{I}))^{\dagger\star}$ is that where $(\neg\hat{I})^\dagger$ has a B-colored constant c_b, $(\neg Lift(\hat{I}))^{\dagger\star}$ has an existentially quantified variable x, and M' interprets c_b exactly with the element of D that M assigns to x to satisfy $(\neg Lift(\hat{I}))^{\dagger\star}$. In conclusion, $M' \models A \wedge \neg\hat{I}$, a contradiction. □

These lemmas explain why lifting replaces A-colored constants by existentially quantified variables, and B-colored constants by universally quantified variables. In Lemma 2, where we reason in a model of B, A-colored constants are new, and their interpretation can be built based on that of existentially quantified variables. When $\hat{I}$ and $Lift(\hat{I})$ are negated, B-colored constants turn out to be replaced by existentially quantified variables. Thus, in Lemma 3, where we reason in a model of A, it is the turn of B-colored constants to be new, and have their interpretation built based on that of existentially quantified variables. The following theorem gives the final result:

Theorem 3 *If $\hat{I}$ is a color-flat provisional interpolant of (A,B), then $Lift(\hat{I})$ is an interpolant of (A,B).*

Proof: $Lift(\hat{I})$ satisfies the definition of interpolant: (i) $A \vdash Lift(\hat{I})$ holds by Lemma 3; (ii) $B, Lift(\hat{I}) \vdash \bot$ holds by Lemma 2; and (iii) $Lift(\hat{I})$ is transparent. □

An interpolation system for Γ is obtained by combining $\Gamma\hat{I}$ and lifting:

Definition 18 (Interpolation system ΓI) For all Γ-refutations of $A \cup B$, the interpolant of (A,B) produced by ΓI is $\mathit{Lift}(\hat{I})$, where $\hat{I}$ is the provisional interpolant of (A,B) computed by $\Gamma\hat{I}$.

Corollary 1 *ΓI is a complete interpolation system for Γ for interpolation problems whose provisional interpolant is color-flat.*

Proof: It follows from Theorems 2 and 3. □

We give next a complete example.

Example 2 Suppose $A = \{f(x) \simeq g(a,x)\}$, $B = \{P(f(b)),\ \neg P(g(y,b))\}$, so that $\Sigma_A = \{f,g,a\}$, $\Sigma_B = \{P,f,g,b\}$, and the ordering $\succ$ used by Γ-inferences is a recursive path ordering based on partial precedence $f > g > a$. Γ produces the refutation:

$$\dfrac{\dfrac{f(x) \simeq g(a,x)_{(\mathbf{A})} \qquad P(f(b))_{(\mathbf{B})}}{P(g(a,b))_{(\mathbf{B})}}\ \{x \leftarrow b\} \qquad \neg P(g(y,b))_{(\mathbf{B})}}{\square}\ \{y \leftarrow a\}$$

where the mgu is written next to the inference line. The first step paramodulates an **A**-labeled equation into a **B**-labeled literal, so that

$$\widehat{PI}(P(g(a,b))) = (f(b) \simeq g(a,b) \vee \bot) \wedge \top = f(b) \simeq g(a,b).$$

The second inference resolves upon two **B**-labeled literals, hence $\hat{I}_1 = \widehat{PI}(\square) = f(b) \simeq g(a,b) \wedge \top = f(b) \simeq g(a,b)$. The interpolant is $I_1 = \mathit{Lift}(\hat{I}_1) = \forall v.\ \exists w.\ f(v) \simeq g(w,v)$, where the universally quantified variable v replaces the B-colored constant b, and the existentially quantified variable w replaces the A-colored constant a. The order of the quantifiers is arbitrary, which means $\exists w.\ \forall v.\ f(v) \simeq g(w,v)$ is also an interpolant. If we swap A and B, and a and b, we have $A = \{P(f(a)),\ \neg P(g(y,a))\}$, $B = \{f(x) \simeq g(b,x)\}$, $\Sigma_A = \{P,f,g,a\}$, and $\Sigma_B = \{f,g,b\}$, with partial precedence $f > g > b$. The Γ-refutation is

$$\dfrac{\dfrac{f(x) \simeq g(b,x)_{(\mathbf{B})} \qquad P(f(a))_{(\mathbf{A})}}{P(g(b,a))_{(\mathbf{A})}}\ \{x \leftarrow a\} \qquad \neg P(g(y,a))_{(\mathbf{A})}}{\square}\ \{y \leftarrow b\}$$

The first inference paramodulates a **B**-labeled equation into an **A**-labeled literal. Thus,

$$\widehat{PI}(P(g(b,a))) = \top \wedge (f(a) \not\simeq g(b,a) \vee \bot) = f(a) \not\simeq g(b,a).$$

The second inference produces $\hat{I}_2 = \widehat{PI}(\square) = f(a) \not\simeq g(b,a) \vee \bot = f(a) \not\simeq g(b,a)$. Then $I_2 = \mathit{Lift}(\hat{I}_2) = \exists v.\ \forall w.\ f(v) \not\simeq g(w,v)$, where the existentially quantified variable v replaces the A-colored constant a, and the universally quantified variable w replaces the B-colored constant b. I_2 is the negation of I_1, as expected from Definition 2.

We conclude this section with a discussion of how the two stage approach can be applied to DPLL($\Gamma+\mathscr{T}$) [13]. A refutation by DPLL-CDCL (the Davis-Putnam-Logemann-Loveland procedure for propositional satisfiability with Conflict-Driven Clause Learning) is a refutation by propositional resolution. A refutation by DPLL($\mathscr{T}$) is a refutation by DPLL-CDCL, where some leaves may be $\mathscr{T}$-lemmas, rather than input clauses. If the $\mathscr{T}$-decision procedure generates proofs, its proofs become subproofs of the DPLL($\mathscr{T}$)-refutation, and $\mathscr{T}$-lemmas appear as roots of subproofs rather than as leaves (cf. Section 2.4 in [12]).

A refutation by DPLL($\Gamma+\mathscr{T}$) is a refutation by DPLL($\mathscr{T}$) where also Γ-proof trees appear as subtrees. Informally, DPLL($\Gamma+\mathscr{T}$) works with *hypothetical clauses* $H \triangleright C$, where C is a clause, and the *hypothesis* H is the set (conjunction) of ground literals, coming from the candidate model built by DPLL($\mathscr{T}$), that were used as premises by Γ to infer C (cf. [13] for details). A hypothetical clause $(l_1 \wedge \ldots \wedge l_n) \triangleright (l'_1 \vee \ldots \vee l'_m)$ is interpreted as $\neg l_1 \vee \ldots \vee \neg l_n \vee l'_1 \vee \ldots \vee l'_m$. When the system detects that a clause $H \triangleright C$ generated by Γ, and such that C is ground, is in conflict with the candidate model built by DPLL($\mathscr{T}$), it generates the ground conflict clause $\neg H \vee C$, and explains the conflict by resolution steps according to the DPLL-CDCL paradigm. This includes as a special case the situation where C is empty. Thus, $\neg H \vee C$ may enter a DPLL($\Gamma+\mathscr{T}$)-refutation, with its Γ-proof tree as subproof. Since a Γ-proof tree is not necessarily ground, even if it generates a $H \triangleright C$ where C is ground, also the refutation by DPLL($\Gamma+\mathscr{T}$) is not necessarily ground.

Thus, one can combine $\Gamma\hat{I}$ with a provisional interpolation system for DPLL($\mathscr{T}$), to obtain a provisional interpolation system for DPLL($\Gamma+\mathscr{T}$), and then apply lifting to obtain interpolants from color-flat provisional interpolants. Because provisional interpolants do not need to be transparent, the two stage approach may allow one to weaken requirements on the DPLL($\mathscr{T}$) side. For instance, the color-based interpolation systems for DPLL($\mathscr{T}$) given in Section 5 of [12] assume that the $\mathscr{T}$-decision procedure generates proofs and interpolants. Assume that $\mathscr{T}$ is a union of theories $\mathscr{T}_1, \ldots, \mathscr{T}_n$, and the $\mathscr{T}$-decision procedure is obtained by combining by *equality sharing* [54] (cf. Chapter 10 of [14] for a more recent presentation) $\mathscr{T}_i$-decision procedures for $1 \leq i \leq n$. The color-based interpolation system for equality sharing in [12] requires that the $\mathscr{T}_i$'s are *convex equality-interpolating* theories. This condition ensures that the proof by equality sharing will not contain AB-mixed equalities, as required by the color-based approach. If all is needed are provisional interpolants, this requirement may be avoided, and AB-mixed equalities remaining in the provisional interpolant will be taken care of by lifting.

As a practical example where this weakening of requirements is useful, we mention *model-based theory combination* (MBTC) [23,13], which is an approach to the implementation of equality sharing. MBTC works for those $\mathscr{T}_i$-decision procedures that build a candidate $\mathscr{T}_i$-model M_i of the $\mathscr{T}_i$-literals. Although this is a strong requirement, there exists, for instance, solvers for *linear arithmetic* that satisfy it (e.g., [31,23]). Then, MBTC lets the $\mathscr{T}_i$-decision procedure propagate equalities true in M_i, rather than $\mathscr{T}_i$-entailed disjunctions of equalities between shared constants, as in standard equality sharing. These equalities are only *guesses*, because it is not known whether they are true in all $\mathscr{T}_i$-models. If one of them turns out to be inconsistent, a conflict occurs. The backjumping mechanism of DPLL-CDCL withdraws the culprit

equality. For those decision procedures for which MBTC works, this also generates enough information to fix M_i. On the other hand, if a disjunction of equalities between shared constants is $\mathscr{T}_i$-entailed, at least one of them is true in M_i.

The proof of refutational completeness of DPLL($\Gamma + \mathscr{T}$) in [13] relies on the standard result of completeness of equality sharing for disjoint and stably infinite theories (cf. Chapter 10 of [14]) with propagation of entailed disjunctions of equalities between shared constants. In practice, an implementation of DPLL($\Gamma + \mathscr{T}$) may use MBTC, for instance for linear arithmetic. If DPLL($\Gamma + \mathscr{T}$) generates a refutation, by soundness, it means that the input is unsatisfiable. Thus, all equalities guessed by MBTC turn out to be inconsistent, are undone by backjumping, and cannot appear in the refutation as unit $\mathscr{T}$-lemmas. However, they may appear as disjuncts in non-unit $\mathscr{T}$-lemmas. These equalities propagated by MBTC may be *AB*-mixed, so that a color-based interpolation system for equality sharing does not apply. On the other hand, a provisional interpolation system tolerates *AB*-mixed equalities, and therefore the two-stage approach allows one to circumvent this practical problem.

6 Related Work

The approach in [42] is the closest to ours, because it also targets resolution and paramodulation in first-order logic with equality, without restrictions to colored or ground proofs, and works in two stages. It employs a notion of *coming from A* or *from B* defined for predicate symbols and applied also to literals. Literals may also come *from both*, which is justified in [42] with factoring. However, also other inference rules may match and mix what comes from A with what comes from B when they unify literals. When the notion is applied to literals, the fact that the classification of a literal may depend on the clause where it occurs does not seem to be considered in [42]. A classification of predicate symbols does not depend on the clause (e.g., P is P in both $C_1 = P(x,y) \vee R(y,x)$ and $C_2 = P(a,b) \vee R(b,a)$), whereas one of literals does (e.g., $P(x,y)$ and $P(a,b)$ are two different literals), as literals get instantiated below the predicate symbol level. When the notion is applied to symbols, and because of the three-way distinction *from A*, *from B*, *from both*, which looks analogous to A-colored, B-colored, transparent, the notion of "coming from" resembles that of color.

We use labels to capture the intuition of a literal descending from A or from B. Every literal is either **A**-labeled or **B**-labeled, not both. We keep colors and labels distinct: colors are based on signature, apply to symbols (cf. Definition 4), and to literals as a consequence (cf. Definition 5); labels are based on inference, apply to literals, and appropriately depend on the clause where the literal occurs (cf. Definition 10).

In the first stage, Huang's system computes *relational interpolants*, that are defined like partial interpolants, but without using projections, and admitting only transparent predicate symbols. Informally, relational interpolants play in Huang's system a rôle analogous to that of provisional partial interpolants in ours. While our definition of provisional interpolant does not require predicate symbols to be transparent, the provisional interpolants produced by $\Gamma\hat{I}$ satisfy this requirement (cf. Lemma 1). Projections are not defined in [42]; however labeled projections and their commuta-

tivity with substitutions seem to be used implicitly in the proof of completeness of the interpolation system for resolution (cf. Proof of Theorem 2 on page 182 of [42]).

$\Gamma\hat{I}$ and Huang's system have the same base cases (cf. Definition 15 and Case (i) on page 182 of [42]). Then, $\Gamma\hat{I}$ has four inductive cases for resolution, because there are two literals resolved upon and two labels. On the other hand, Huang's system has three inductive cases for resolution:

- Both literals resolved upon coming from A (cf. Case (ii)(a) on page 182 of [42]), which is the same as the one in Definition 15 where both literals resolved upon have label **A**;
- Both coming from B (cf. Case (ii)(b) on page 182 of [42]), which is the same as the one in Definition 15 where both literals resolved upon have label **B**; and
- A third catch-all case (cf. Case (ii)(c) on page 182 of [42]), which in our notation is written $\widehat{PI}(c) = [(\neg l' \wedge \widehat{PI}(p_1)) \vee (l \wedge \widehat{PI}(p_2))]\sigma$.

Similarly, in place of the two cases for factoring in Definition 15, Huang's system has one, where $\widehat{PI}(c) = \widehat{PI}(p)\sigma$ (cf. Case (iv) on page 184 of [42]).

For paramodulation, Huang's system has three inductive cases, numbered (iii)(d), (iii)(e), and (iii)(f) on page 183 of [42]. Cases (iii)(d) and (iii)(e) use a notion of *maximal A-term* or *B-term* defined as follows: an A-term is a term whose top symbol comes from A, and an occurrence of an A-term is maximal if it is not a subterm of another A-term. B-terms and their maximal occurrences are defined in the same way with B in place of A. Here the notion of "coming from" is implicitly extended to function symbols, so that it appears to coincide with color, in the sense that an A-term is a term whose top symbol is A-colored. We reproduce the three cases in our notation, assuming that $p_1 \colon s \simeq r \vee C$ paramodulates into $p_2 \colon l[s'] \vee D$ as in Definition 15:

(iii)(d) $\widehat{PI}(c) = [(\widehat{PI}(p_2) \wedge s \simeq r) \vee (\widehat{PI}(p_1) \wedge s \not\simeq r)]\sigma \vee (s \simeq r \wedge h(s) \not\simeq h(r))\sigma$, provided s' occurs in $l[s'] \vee D$ as subterm of a maximal B-term $h(s')$, and there is more than one occurrence of $h(s')$ in $l[s'] \vee D \vee \widehat{PI}(p_2)$;

(iii)(e) $\widehat{PI}(c) = [(\widehat{PI}(p_2) \wedge s \simeq r) \vee (\widehat{PI}(p_1) \wedge s \not\simeq r)]\sigma \wedge (s \not\simeq r \vee h(s) \simeq h(r))\sigma$, provided s' occurs in $l[s'] \vee D$ as subterm of a maximal A-term $h(s')$, and there is more than one occurrence of $h(s')$ in $l[s'] \vee D \vee \widehat{PI}(p_2)$;

(iii)(f) $\widehat{PI}(c) = [(\widehat{PI}(p_2) \wedge s \simeq r) \vee (\widehat{PI}(p_1) \wedge s \not\simeq r)]\sigma$, otherwise.

It is not said whether h is a function symbol or a context. Cases (iii)(d) and (iii)(e) are neither explained nor applied in the examples, while Cases (iii)(e) and (iii)(f) are not covered in the proof of completeness. Resolution and paramodulation are treated separately, so that, for instance, the notion of "literal coming from" is not defined, when the parent of a resolution step is a paramodulant. Our provisional interpolation system $\Gamma\hat{I}$ uses labels systematically, and treats resolution and paramodulation uniformly and in an integrated way.

The second stage in [42] features a lifting mechanism which replaces maximal A-terms with existentially quantified variables and maximal B-terms with universally quantified variables. We borrowed the name "lifting" from [42], however our lifting mechanism only replaces colored constants. The idea of obtaining interpolants by replacing non-shared constant symbols with quantified variables appeared in [21], was mentioned in [50], and under the name of *abstraction*, was used in [4] to interpolate

LK-proofs with only atomic cuts (cf. Chapter 8.2). If the system is allowed to replace compound subterms by quantified variables, alternations of quantifiers must be generated in the right order in order to obtain an interpolant. For instance, for $A = \{P(a,x)\}$, $B = \{\neg P(y,f(y))\}$, and $\hat{I} = P(a,f(a))$, where the first occurrence of a is a maximal A-term, and $f(a)$ is a maximal B-term, the interpolant is $I = \exists z.\forall w.P(z,w)$, because w depends on z, as a is a subterm of $f(a)$, whereas $\forall w.\exists z.P(z,w)$ is not an interpolant. The argument of the proofs of our Lemmas 3 and 2 does not apply if lifting replaces compound terms, because there is not enough information to build an interpretation of $\hat{I}$ from one of $Lift(\hat{I})$ if function symbols are removed. The claim of completeness in [42] rests on a different proof argument, which goes through grounding the non-ground proof: we refer the interested reader to [42] for details.

After [42], the study of interpolation for superposition restarted in [51], with the notions of *local* inference and proof, and a hint that non-local superposition inferences would have to be *procrastinated*, until they become local by further instantiation. In hindsight this may be seen as an early lead towards approaches based on *instantiation* and *proof transformation*, that were later developed in [18] and [52] for ground refutations by a DPLL($\mathscr{T}$)-based SMT-solver, equipped with an instantiation procedure (e.g., [28,22,36,53,37]) to generate ground instances of non-ground input axioms. The refutations to be interpolated are ground, but not necessarily colorable, because instantiation may insert AB-mixed literals. In [18], *purification* replaces critical subterms by new constant symbols considered transparent. In [52], *localization* replaces critical subterms by new constant symbols considered interpreted. Purification visits the literal top-down, while localization proceeds bottom-up. In both approaches, equations defining the new constant symbols are introduced in the proof. Since new constants do not belong to the original signature, they are later replaced by quantified variables. In order to avoid loops in the definitions, and introduce the quantifiers with the correct alternations, both methods use an ordering, defined in [18] as the topological order on the inverse dependency graph of the equations defining the new constants, and captured in [52] by a precedence on new constants.

Another line of research stemming from [51] was pursued in [46,40,41]. A key idea in both [50] and [46] is that an interpolant is a boolean combination of transparent formulæ from the proof. In [46] this result is formulated as a lemma which proves the existence of C-interpolants for transparent clauses in local proof trees built by generic inferences. A *C-interpolant* is defined like a partial interpolant of clause C, for the special case where C is transparent and therefore projections are not needed. The approach is inductive, as the constructed C-interpolant is a boolean combination of selected transparent ancestors of C in the proof and their C-interpolants. The selected ancestors are consequences of *symbol-eliminating inferences*, meaning inferences that deduce a transparent consequence from premises with at least one colored symbol. The report of experiments in [40] discusses restricting inferences in Vampire, to favor the generation of colored proofs, and having the prover output all consequences of symbol-eliminating inferences. The development in [41] adds a notion of *digest* of a proof, to specify which ancestors to pick to construct the C-interpolant. The digest is made of transparent subproofs between colored subproofs. The focus of [41] is on two proof transformations. The first one aims at localizing non-local refutations: it assumes that the only colored symbols in proofs are constants, and it

replaces constants of one color, either A-color, or B-color, by existentially quantified variables. Since the purpose is localizing the proof, eliminating one color suffices. The second one, called *grey slicing*, aims at minimizing the generated interpolant: it works by repeatedly removing intermediate transparent clauses. The experiments in [41] involve Z3 [24] and Vampire [47]: localization is applied to the proofs by Z3, which are ground. Clearly, not all proofs can be localized, and localization fails if there are colored function symbols.

7 Discussion

We studied interpolation of refutations generated by inference systems for first-order logic with equality based on resolution and paramodulation/superposition. The main contribution is a *new interpolation system for non-ground refutations*, which generates first *provisional interpolants*, and obtains then interpolants by replacing colored, or non-shared, constant symbols with quantified variables. This interpolation system is *complete* if the provisional interpolant is *color-flat*, meaning that its only colored symbols are constants. The general question of extracting an interpolant from any refutation by resolution and paramodulation, for any interpolation problem (A,B), without hypotheses or restrictions, remains open. Although the restriction to color-flat provisional interpolants is not a minor one, our interpolation system may be relevant to several fragments without function symbols, that are decidable, have the finite model property (e.g., [34,56]), and in some cases admit superposition-based decision procedures (e.g., [35,33,32]).

We started with examples showing that a color-based approach to interpolation (surveyed in [12] for the ground case) does not generalize to non-ground proofs, not even assuming that they are colorable, colored (also known as local), or that all predicate and function symbols are interpreted or transparent. Counter-examples where the only colored symbols are constants suffice. Thus, we reckoned that in the inductive approach to interpolation, one needs (1) to track the descendance of literals from either the A or the B part of the refutation, to ensure that the interpolant follows from A and is inconsistent with B, and (2) to check that only transparent literals enter in the interpolant. The color-based style employs colors to achieve both goals.

We presented a *two-stage approach*, inspired by [42], which separates the two concerns. In the first stage, a *provisional interpolation system* uses *labels* in place of colors to track descendance of literals, and computes a *provisional interpolant*, which follows from A and is inconsistent with B, but may contain colored symbols. A key property of this labeling mechanism is that labels are independent of substitutions, because they are inherited through inferences, rather than being defined based on signatures like colors. In this way, the obstacle represented by the fact that an instance $l\sigma$ is not guaranteed to have the same color of l, which hinders the color-based approach, disappears. We defined an inductive provisional interpolation system for Γ, named $\Gamma\hat{I}$, and proved its *completeness* (cf. Theorem 2).

In the second stage, *lifting* replaces A-colored constants with existentially quantified variables, and B-colored constants with universally quantified variables. This suffices to obtain a transparent formula, if all function symbols in the provisional

interpolant are either interpreted or transparent, as $\Gamma\hat{I}$ already ensures that predicate symbols will be such. Under this hypothesis, we proved that the lifting of provisional interpolants yields interpolants (cf. Theorem 3), thus establishing the *completeness* of the interpolation system ΓI that combines $\Gamma\hat{I}$ with lifting.

In some contexts this hypothesis is satisfied because there are no function symbols. Examples include Datalog or the recursion-free Horn clauses without uninterpreted function symbols of [56]. Other examples are some decidable fragments of first-order logic studied for instance in [34]: the *Löwenheim class with equality*, also known as first-order relational monadic logic (only unary predicates and constants, no functions); the *Bernays-Schönfinkel-Ramsey (BSR) class with equality* (formulæ of the form $\exists^*\forall^*\varphi$, where φ is quantifier-free and function-free, while constants are allowed); and the FO^2 *class* (first-order formulæ with only two variables, with constants, no functions). A subclass of BSR, the Bernays-Schönfinkel-Horn class, is sufficient to axiomatize *timed automata*, with superposition as a decision procedure [33, 32]. The FO^2 class includes the *modal fragment*, or the fragment of formulæ obtained by translating modal logic into first-order logic. The modal fragment was generalized to the *guarded fragment* in [1], by showing that the key property for decidability is not the number of variables, but the absence of function symbols and the restriction on quantification. A superposition-based decision procedure for the guarded fragment was given in [35]. Model-building decision procedures for the guarded fragment were studied in [29].

We discussed the application of the two-stage approach to the DPLL($\Gamma+\mathscr{T}$) method of [13]. As surveyed in [12], a color-based approach to interpolation of equality sharing requires the built-in theories to be *convex equality-interpolating*, to avoid the propagation of AB-mixed literals. A different approach to interpolation in combination of theories appears in [15,16]. In a two-stage approach AB-mixed literals become harmless, and this requirement is no longer necessary. This may be useful for implementations of equality sharing that employ *model-based theory combination* [23], which propagates equalities true in a model, rather than entailed, and may propagate AB-mixed equalities. Model-based theory combination is used, for instance, for linear integer arithmetic, which is not equality-interpolating.

The state of the art on interpolation can be advanced by giving interpolation systems for logics, theories, or inference systems that did not have them, or by giving interpolation systems that produce better interpolants than the existing ones. Our research is of the first kind, and therefore we aimed at generality and completeness, by developing a two-stage approach that overcomes the limitations of the color-based approach with respect to non-ground proofs. A direction for future work is to aim at a two-stage interpolation system that handles also provisional interpolants that are not color-flat: a possibility is to investigate combining our provisional interpolation system with a reconstruction of the second stage of [42]. Since the two-stage approach may generate interpolants with many quantifiers, other topics for future work are the evaluation of its practical efficiency by implementation and experiments, and the analysis of its complexity. Also important is the study of the characteristics of the generated interpolants, such as strength, length, number of quantifier alternations, and how to bound them. Research on the qualities of interpolants will benefit from

feed-back from experiments and may be more significant if conducted in relation to specific applications.

Acknowledgements Early versions of parts of this work were presented at meetings of COST Action IC0109 in Tallinn (March 2012), and Haifa (November 2012), at a Workshop in Honor of Arnon Avron in Tel Aviv (November 2012), and at the First International Workshop on Interpolation: from Proofs to Applications in St. Petersburg (July 2013). We thank the organizers, and Leonardo de Moura, Swen Jacobs, Laura Kovàcs, Viktor Kuncak, Ken McMillan, Philipp Rümmer, and Andrei Voronkov, for their comments. We thank the anonymous reviewers for their suggestions.

References

1. Hajnal Andréka, Johan van Benthem, and István Nemeti. Modal logics and bounded fragments of predicate logic. *Journal of Philosophical Logic*, 27(3):217–274, 1998.
2. 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.
3. Alessandro Armando, Silvio Ranise, and Michaël Rusinowitch. A rewriting approach to satisfiability procedures. *Information and Computation*, 183(2):140–164, 2003.
4. Matthias Baaz and Alexander Leitsch. *Methods of Cut-Elimination*. Springer, Berlin, 2011.
5. Sascha Böhme and Michal Moskał. Heaps and data structures: a challenge for automated provers. In Nikolaj Bjørner and Viorica Sofronie-Stokkermans, editors, *Proceedings of the 23rd Conference on Automated Deduction (CADE)*, volume 6803 of *Lecture Notes in Artificial Intelligence*, pages 177–191, Berlin, 2011. Springer.
6. Maria Paola Bonacina. On theorem proving for program checking – historical perspective and recent developments. In Maribel Fernandez, editor, *Proceedings of the 12th International Symposium on Principles and Practice of Declarative Programming (PPDP)*, pages 1–11, New York, NY, 2010. ACM.
7. Maria Paola Bonacina. Two-stage interpolation systems. In Laura Kovàcs and Georg Weissenbacher, editors, *Notes of the First International Workshop on Interpolation: from Proofs to Applications (IPrA), Twenty-Fifth International Conference on Computer Aided Verification (CAV)*, Technical Reports. Technische Universität Wien, 2013.
8. 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.
9. Maria Paola Bonacina and Mnacho Echenim. On variable-inactivity and polynomial $\mathscr{T}$-satisfiability procedures. *J. Logic and Computation*, 18(1):77–96, 2008.
10. 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.
11. 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.
12. Maria Paola Bonacina and Moa Johansson. Interpolation of ground proofs: a survey. Submitted for publication, 2014. Available at `http://profs.sci.univr.it/~bonacina/`.
13. Maria Paola Bonacina, Christopher A. Lynch, and Leonardo de Moura. On deciding satisfiability by theorem proving with speculative inferences. *J. Automated Reasoning*, 47:161–189, 2011.
14. Aaron R. Bradley and Zohar Manna. *The Calculus of Computation – Decision Procedures with Applications to Verification*. Springer, Berlin, 2007.
15. 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.
16. Roberto Bruttomesso, Silvio Ghilardi, and Silvio Ranise. Quantifier-free interpolation in combinations of equality interpolating theories. *ACM Transactions on Computational Logic*, 15(1), 2014.

17. Ritu Chadha and David A. Plaisted. On the mechanical derivation of loop invariants. *J. Symbolic Computation*, 15(5-6):705–744, 1993.
18. Jürgen Christ and Jochen Hoenicke. Instantiation-based interpolation for quantified formulae. Notes of the 8th International Workshop on Satisfiability Modulo Theories (SMT), 2010.
19. Alessandro Cimatti, Alberto Griggio, and Roberto Sebastiani. Efficient interpolant generation in satisfiability modulo theory. *ACM Transactions on Computational Logic*, 12(1):7, 2010.
20. William Craig. Linear reasoning. A new form of the Herbrand-Gentzen theorem. *J. Symbolic Logic*, 22(3):250–268, 1957.
21. William Craig. Three uses of the Herbrand-Gentzen theorem in relating model theory and proof theory. *J. Symbolic Logic*, 22(3):269–285, 1957.
22. 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.
23. Leonardo de Moura and Nikolaj Bjørner. Model-based theory combination. In Sava Krstić and Albert Oliveras, editors, *Proceedings of the 5th Workshop on Satisfiability Modulo Theories (SMT 2007)*, volume 198(2) of *Electronic Notes in Theoretical Computer Science*, pages 37–49, Amsterdam, 2008. Elsevier.
24. Leonardo de Moura and Nikolaj Bjørner. Z3: an efficient SMT solver. In C. R. Ramakrishnan and Jakob Rehof, editors, *Proceedings of the 14th Conference on Tools and Algorithms for the Construction and Analysis of Systems (TACAS)*, volume 4963 of *Lecture Notes in Computer Science*, pages 337–340, Berlin, 2008. Springer.
25. 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.
26. Leonardo de Moura and Nikolaj Bjørner. Satisfiability modulo theories: introduction and applications. *Comm. ACM*, 54(9):69–77, 2011.
27. 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.
28. David L. Detlefs, Greg Nelson, and James B. Saxe. Simplify: a theorem prover for program checking. *JACM*, 52(3):365–473, 2005.
29. Michael Dierkes. *Model Building for Sets of Guarded Clauses*. PhD thesis, Institut National Polytechnique de Grenoble, 2001.
30. 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.
31. Bruno Dutertre and Leonardo de Moura. A fast linear arithmetic solver for DPLL(T). In Tom Ball and R. B. Jones, editors, *Proceedings of the 18th Conference on Computer Aided Verification (CAV)*, volume 4144 of *Lecture Notes in Computer Science*, pages 81–94, Berlin, 2006. Springer.
32. Arnaud Fietzke. *Labelled superposition*. PhD thesis, Max Planck Institut für Informatik, Saabrücken, 2013.
33. Arnaud Fietzke and Christoph Weidenbach. Superposition as a decision procedure for timed automata. *Mathematics in Computer Science*, 6(4):409–425, 2012.
34. Pascal Fontaine. Combinations of theories for decidable fragments of first-order logic. In Silvio Ghilardi and Roberto Sebastiani, editors, *Proceedings of the 7th Symposium on Frontiers of Combining Systems (FroCoS)*, volume 5749 of *Lecture Notes in Artificial Intelligence*, pages 263–278. Springer, 2009.
35. Harald Ganzinger and Hans de Nivelle. A superposition decision procedure for the guarded fragment with equality. In *Proceedings of the 14th IEEE Symposium on Logic in Computer Science (LICS)*. IEEE Computer Society Press, 1999.
36. 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.
37. 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.

38. 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.
39. 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, NY, 2004. ACM.
40. 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.
41. 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, NY, 2012. ACM.
42. 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.
43. Deepak Kapur. A quantifier-elimination based heuristic for automatically generating inductive assertions of programs. *J. System Science and Complexity*, 19(3):307–330, 2006.
44. Deepak Kapur, Zhihai Zhang, Matthias Horbach, Hengjun Zhao, Qi Lu, and Thanh Vu Nguyen. Geometric quantifier elimination heuristics for automatically generating octagonal and max-plus invariants. In Maria Paola Bonacina and Mark E. Stickel, editors, *Automated Reasoning and Mathematics: Essays in Memory of William W. McCune*, volume 7788, pages 189–228. Springer, Berlin, 2013.
45. Laura Kovàcs and Andrei Voronkov. Finding loop invariants for programs over arrays using a theorem prover. In *Proc. of the Conf. on Fundamental Approaches to Software Engineering*, number 5503 in LNCS, pages 470–485, Berlin, 2009. Springer.
46. 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.
47. Laura Kovàcs and Andrei Voronkov. First order theorem proving and Vampire. 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 1–35, Berlin, 2013. Springer.
48. 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.
49. 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.
50. Kenneth L. McMillan. An interpolating theorem prover. *Theoretical Computer Science*, 345(1):101–121, 2005.
51. 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.
52. 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, NY, 2011. ACM and IEEE.
53. 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/`.
54. Greg Nelson and Derek C. Oppen. Simplification by cooperating decision procedures. *ACM Transactions on Programming Languages and Systems*, 1(2):245–257, 1979.
55. Robert Nieuwenhuis, Albert Oliveras, and Cesare Tinelli. Solving SAT and SAT modulo theories: from an abstract Davis-Putnam-Logemann-Loveland procedure to DPLL(T). *JACM*, 53(6):937–977, 2006.
56. 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.

57. Stephan Schulz. System description: E 1.8. In Ken McMillan, Aart Middeldorp, and Andrei Voronkov, editors, *Proceedings of the 19th Conference on Logic, Programming and Automated Reasoning (LPAR)*, volume 8312 of *Lecture Notes in Artificial Intelligence*, pages 735–743, Berlin, 2013. Springer.
58. Natarajan Shankar. Automated deduction for verification. *ACM Comput. Surv.*, 41(4):40–96, 2009.
59. Raymond M. Smullyan. *First-Order Logic*. Dover Publications, New York, NY, 1995. First published by Springer in 1968.
60. Christoph Weidenbach, Dylana Dimova, Arnaud Fietzke, Rohit Kumar, Martin Suda, and Patrick Wischnewski. SPASS version 3.5. In Renate Schmidt, editor, *Proceedings of the 22nd Conference on Automated Deduction (CADE)*, volume 5663 of *Lecture Notes in Artificial Intelligence*, pages 140–145, Berlin, 2009. Springer.
61. Georg Weissenbacher. *Program Analysis with Interpolants*. PhD thesis, Magdalen College, Oxford University, 2010.