Ground Interpolation for Combined Theories

Amit Goel[1], Sava Krstić[1], and Cesare Tinelli[2]

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

Abstract. We give a method for modular generation of ground interpolants in modern SMT solvers supporting multiple theories. Our method uses a novel algorithm to modify the proof tree obtained from an unsatifiability run of the solver into a proof tree without occurrences of troublesome "uncolorable" literals. An interpolant can then be readily generated using existing procedures. The principal advantage of our method is that it places few restrictions (none for convex theories) on the search strategy of the solver. Consequently, it is straightforward to implement and enables more efficient interpolating SMT solvers. In the presence of non-convex theories our method is incomplete, but still more general than previous methods.

1 Introduction

Given mutually inconsistent formulas F and G in some logic, an interpolant I is a formula such that: *(i)* $F \models I$; *(ii)* $G, I \models \mathsf{false}$; and *(iii)* the non-logical symbols in I occur in both F and G. In [13], McMillan presented an algorithm for propositional interpolation and described a complete procedure for model-checking finite-state systems. In this method, interpolants are used to derive property-driven overapproximations of reachable state sets from unsatisfiable symbolic traces. This technique has proven to be efficient in practice, and the recipe in [13] is used as a starting point in many finite-state model checkers.

A natural desire is to extend interpolation-based methods to decidable fragments of richer logics, for use in applications such as software model checking. Most solvers for *satisfiability modulo theories* (SMT) employ a propositional SAT solver in cooperation with theory-specific decision procedures (*theory solvers*) to solve queries in the combined language. The promise of interpolating SMT solvers has been demonstrated by the use of FOCI [14] for model-checking C programs [10, 15]. However, their development has not been nearly as widespread as for the propositional case; we know of only two other interpolating SMT solvers [6, 3].

The quick adoption of propositional interpolation is in large part due to the simplicity of the propositional interpolation algorithm. It requires a SAT solver enhanced only with the capability to produce a resolution refutation for unsatisfiable formulas. The interpolant is computed by a simple recursive function on resolution proofs. The published solutions for SMT interpolation, on the other hand, either describe an ad hoc solver for a specific collection of theories, or

require significant modifications in more general SMT solvers to limit them sufficiently for the described method to work. In this paper we present a simple algorithm for interpolant generation from refutations produced by SMT solvers, while placing minimal restrictions on the solvers' search strategy.

Related Work. In the seminal work [14], McMillan produced a proof system for the ground theory of linear arithmetic with uninterpreted functions and showed how to generate interpolants from such proofs. Yorsh and Musuvathi [18] extended the approach to general combinations of theories that are individually interpolant-generating. These authors were the first to isolate the important requirement that the theories be *equality-interpolating*: if a theory solver can derive $x{=}y$ from $F \wedge G$, where x occurs only in F and y occurs only in G ("uncolorable equality"), then it must be able to derive $x{=}t$ and $t{=}y$ for some term t in the language common to F and G. There are two shortcomings to their approach. Firstly, it requires the generation and propagation of equality-interpolating terms on the fly, thus imposing an overhead during the search procedure of the SMT solver. Secondly, it requires theory solvers to be equality-propagating. As noted in [8], equality propagation can take the majority of time in some decision procedures for little gain. Indeed, modern SMT solvers let the SAT solver split on equalities and either forgo equality propagation completely (*delayed theory combination* (DTC) [4]) or use it sparingly (*model-based theory combination* [7]).

The MathSAT [6] and CSIsat [3] tools avoid the problem of on-the-fly creation of equality interpolants; they create only those equality interpolants that are needed for a series of local proof transformations that modify refutations produced by their solvers into the form suitable for deriving interpolants. The authors of [6] identify the class of *ie-local* refutations which are amenable to such transformations. However, the search strategy in both tools is restricted. CSIsat requires equality-propagating decision procedures, while MathSAT simulates equality propagation with heuristics to restrict delayed theory combination.

The interpolation algorithms in all these methods and ours rely on theory-specific interpolation procedures such as those in [14, 17, 6, 9].

Contributions. We define *almost-colorable* refutations and present a two-phase algorithm for the generation of interpolants from such refutations. In the first phase, the almost-colorable refutation is transformed into a *colorable* refutation. The interpolant is then derived from the colorable refutation in the second phase.

There are several advantages to our approach. The class of almost-colorable refutations is more general than the class of ie-local refutations. We show that for the case of convex theories, any search strategy for an SMT solver will produce almost-colorable refutations as long as the theory solvers satisfy the reasonable requirement of not generating lemmas with redundant equalities. In the more general case with non-convex theories, we require the SAT solver not to split on uncolorable equalities. This compromises the completeness of the SMT solver, but enables us to interpolate for a larger set of formulas than [18] since we do allow splitting on colorable equalities. We also show that for a subset of

almost-colorable refutations (including ie-local ones), our colorability algorithm produces refutations whose size, measured by the number of nodes in a tree representation, is at most twice the size of the input refutation.

Outline. In §2, we review and define the necessary material, including the concepts of proof trees modulo a given collection of theories and a given set S of input clauses. We also define colorability of proof trees with respect to a partition $S = A \cup B$ of the input clause set and recall the algorithm that produces an interpolant for A, B from a given colorable proof tree. In §3, we define the class $\mathcal{P}(A, B)$ of almost-colorable proof trees and prove in Theorem 2 that each proof tree from this class can be transformed into a colorable one. We also give a detailed description of the coloring transformation algorithm. In §4, we define $\text{NODPLL}^{\text{pf}}$, a transition system for abstractly describing modern SMT solvers, and prove in Theorem 4 that it produces almost-colorable proof trees when the theories are convex, or if splitting on uncolorable equalities is disallowed.

2 Preliminaries

2.1 Syntax

We will use the standard terminology. A *signature* is a set of *function symbols* plus a set of *predicate symbols*. *Terms* are built using *variables* and *free constants* by recursive application of function symbols. *Atoms* are applications of predicate symbols to terms. Atoms and their negations are *literals*. A *(quantifier-free) formula* is a boolean combination of atoms. A term or a formula is *ground* if it has no occurrences of variables. See [2] for more details.

A *(ground) clause* is a set of ground literals. Clause γ is a *resolvent* of clauses α and β if there is an atom p such that $\alpha = \alpha' \uplus p$, $\beta = \beta' \uplus \neg p$ and $\gamma = \alpha' \cup \beta'$. We call p the *atom resolved upon*. We say that γ is a *merge* [1] of any common literal in α' and β'. We use $\uplus$ to denote disjoint union and, to avoid clutter, we write l for the singleton $\{l\}$. We will not distinguish between the clause $\{l_1, \ldots, l_n\}$ and the disjunction $l_1 \vee \cdots \vee l_n$.

2.2 Resolution Proof Trees

A *tree* is a finite directed graph with a *root* node that is reachable from every other node, and every other node has exactly one outgoing edge. *Leaves* are nodes with no incoming edges. In a *binary tree* every *internal* (i.e. non-leaf) node n has exactly two incoming edges connecting n with its *parents*.

A *resolution proof tree* (or just *proof tree*) is a binary tree together with a mapping that associates with each node n a ground clause $[\![n]\!]$ so that the clause at each internal node of the tree is a resolvent of the clauses of the node's parents. The atom resolved upon at the node n is called the *pivot at* n. If P is a proof tree, we will write $[\![P]\!]$ for the clause associated with the root of P. A *refutation* is any proof tree P such that $[\![P]\!]$ is the empty clause.

We will write $P = \langle P_1, l, P_2 \rangle$ when P_1 and P_2 are the subtrees of P rooted at the parent nodes of the root of P, l is the literal resolved upon at the root of P, and $l \in [\![P_1]\!]$, $\neg l \in [\![P_2]\!]$. Note that $\langle P_1, l, P_2 \rangle$ and $\langle P_2, \neg l, P_1 \rangle$ represent the same proof tree. When using $\langle P_1, l, P_2 \rangle$ to define P_1, P_2 we will assume, without loss of generality, that l is an atom.

Lemma 1. *If $P = \langle P_1, l, P_2 \rangle$, then $[\![P_1]\!] \subseteq [\![P]\!] \cup l$ and $[\![P_2]\!] \subseteq [\![P]\!] \cup \neg l$.*

2.3 Theories

A signature Σ defines the class of *Σ-models*. A *Σ-theory* is a set $\mathcal{T}$ of Σ-models. A ground Σ-formula ϕ is *$\mathcal{T}$-satisfiable* if there is a model of $\mathcal{T}$ and an assignment of elements of the model to free constants that make ϕ true. We write $S \models_{\mathcal{T}} \phi$ when ϕ is true in all $\mathcal{T}$-models that satisfy each formula in the set S, for all assignments to free constants (and abbreviate $\varnothing \models_{\mathcal{T}} \phi$ with $\models_{\mathcal{T}} \phi$). If $\Sigma_1, \ldots, \Sigma_n$ are disjoint signatures, and $\mathcal{T}_i$ is a Σ_i-theory ($i = 1, \ldots, n$), then there is a well-defined $(\Sigma_1 + \cdots + \Sigma_n)$-theory $\mathcal{T}_1 + \cdots + \mathcal{T}_n$. For more details, see [2].

Let S be a finite set of *input clauses* and $\mathcal{T}_1 + \cdots + \mathcal{T}_n$ be a fixed disjoint union of theories. A clause γ such that $\models_{\mathcal{T}_i} \gamma$ is called a *theory lemma*, or a $\mathcal{T}_i$-lemma to be specific. We define a *$(\mathcal{T}_1, \ldots, \mathcal{T}_n)$-proof tree from S* to be any proof tree in which the clause $[\![n]\!]$ for every leaf n is either an input clause or a theory lemma. It is straightforward to show that $S \models_{\mathcal{T}_1 + \cdots + \mathcal{T}_n} [\![P]\!]$, if P is a $(\mathcal{T}_1, \ldots, \mathcal{T}_n)$-proof tree from S.

When the input set of clauses is given as a union $S = A \cup B$, we will use the following coloring terminology. A term or literal will be called *A-colorable* if all non-logical symbols that occur in it also occur in A. We define *B-colorable* similarly. A term or literal that is both A- and B-colorable will be called *AB-colored*. A term or literal that is A-colorable (resp. B-colorable) but not AB-colored is *A-colored* (resp. *B-colored*). A term or literal is *colorable* if it is A- or B-colorable, and is *uncolorable* otherwise. A clause is colorable if every literal occurring in it is colorable. Define the splitting $\gamma = \gamma_{\setminus B} \uplus \gamma_{\downarrow B}$ of any colorable clause γ into subclauses $\gamma_{\setminus B}$ and $\gamma_{\downarrow B}$ consisting of A-colored and B-colorable literals in γ respectively. A $(\mathcal{T}_1, \ldots, \mathcal{T}_n)$-proof tree from $A \cup B$ is colorable if every literal occurring in it is colorable. A node in a proof tree is *critical* if it is an internal node and its pivot is uncolorable.

A theory $\mathcal{T}$ is *ground interpolating* if for every pair of sets A, B of ground clauses such that $A, B \models_{\mathcal{T}} \mathsf{false}$, there exists an AB-colored ground formula ϕ (a *ground $\mathcal{T}$-interpolant* for A, B) such that $A \models_{\mathcal{T}} \phi$ and $B, \phi \models_{\mathcal{T}} \mathsf{false}$. A computable function $\mathsf{itp}_{\mathcal{T}}(A, B)$ that computes a ground $\mathcal{T}$-interpolant for any given input sets A and B of *literals*[3] will be called a *ground interpolation procedure* for $\mathcal{T}$. Such a procedure can be extended to a procedure that computes ground interpolants for arbitrary sets A and B of ground clauses (not just sets of literals); see [14, 6] and the special case $n = 1$ of Theorem 1 below.

A theory $\mathcal{T}$ is *equality interpolating* [18] if for every $\mathcal{T}$-lemma $\gamma \uplus x{=}y$ such that γ is colorable and $x{=}y$ is uncolorable, there exists an AB-colored term

[3] More precisely, A and B are sets of *one-literal clauses*.

z such that $\models_{\mathcal{T}} \gamma \cup x{=}z$ and $\models_{\mathcal{T}} \gamma \cup z{=}y$. The term z is called an *equality interpolant* for the clause $\gamma \uplus x{=}y$. It is shown in [18] that not all theories are equality interpolating, but the commonly used ones are.

2.4 Deriving Interpolants from Colorable Proof Trees

It is possible to produce a ground interpolant for A, B from any colorable $(\mathcal{T}_1, \ldots, \mathcal{T}_n)$-refutation P from $A \cup B$, if each $\mathcal{T}_i$ has a ground interpolation procedure, itp_i. Define I_P by:

$$I_P = \begin{cases} \mathsf{itp}_i(\neg[\![P]\!]_{\setminus B}, \neg[\![P]\!]_{\downarrow B}) & \text{if } [\![P]\!] \text{ is a } \mathcal{T}_i\text{-lemma} \\ [\![P]\!]_{\downarrow B} & \text{if } [\![P]\!] \in A \\ \mathsf{true} & \text{if } [\![P]\!] \in B \\ I_{P_1} \vee I_{P_2} & \text{if } P = \langle P_1, l, P_2 \rangle \text{ and } l \text{ is } A\text{-colored} \\ I_{P_1} \wedge I_{P_2} & \text{if } P = \langle P_1, l, P_2 \rangle \text{ and } l \text{ is } B\text{-colorable} \end{cases}$$

Theorem 1 ([14, 6]). *If P is a colorable $(\mathcal{T}_1, \ldots, \mathcal{T}_n)$-refutation from $A \cup B$, then I_P is a ground interpolant for A, B.*

Proof. By induction on the number of nodes in P, *(i)* $A \models_{\mathcal{T}_1+\cdots+\mathcal{T}_n} I_P \vee [\![P]\!]_{\setminus B}$, *(ii)* $B, I_P \models_{\mathcal{T}_1+\cdots+\mathcal{T}_n} [\![P]\!]_{\downarrow B}$, and *(iii)* I_P is AB-colored. □

Note that I_P as defined here is not unique because the conditions for the cases are not mutually exclusive. For our purposes, this is inconsequential. Note also that this definition is obtained from the propositional interpolation algorithm of [13] by the addition of the first case (for theory lemmas).

2.5 Modifying Proof Trees

When $[\![P']\!] \subseteq [\![P]\!]$, we say that P' is *stronger* than P, and that P is *weaker* than P'. Clearly, any proof tree stronger than a refutation is also a refutation.

We will use a simple, typically unnamed, construction to strengthen a proof, given strengthened subproofs [1]. Let $P = \mathsf{stitch}(P_1, l, P_2)$ be specified as follows: if $l \in [\![P_1]\!]$ and $\neg l \in [\![P_2]\!]$, then $P = \langle P_1, l, P_2 \rangle$; if $l \notin [\![P_1]\!]$ then $P = P_1$; otherwise $P = P_2$. Thus, stitch attempts to resolve two given proof trees over a specified literal, returning one of the input trees when resolution is not possible.

Lemma 2. *Let P_1, P_2 be arbitrary proof trees, l be an arbitrary literal and α, β be arbitrary clauses.*

(i) If $[\![P_1]\!] \subseteq \alpha \cup l$ and $[\![P_2]\!] \subseteq \beta \cup \neg l$, then $[\![\mathsf{stitch}(P_1, l, P_2)]\!] \subseteq \alpha \cup \beta$.
(ii) If $\langle P_1, l, P_2 \rangle$ is defined and the proof trees P_1' and P_2' are stronger than P_1 and P_2 respectively, then $\mathsf{stitch}(P_1', l, P_2')$ is stronger than $\langle P_1, l, P_2 \rangle$.

Another way of strengthening proof trees is by changing the order of pivots. If $P = \langle\langle P_1, l_1, P_2\rangle, l_2, P_3\rangle$ and $P' = \mathsf{stitch}(\mathsf{stitch}(P_1, l_2, P_3), l_1, \mathsf{stitch}(P_2, l_2, P_3))$, we say then that P' is obtained from P by a *raising* the pivot l_2 over l_1 [11]; see also Exchange Lemma 4.1.3 of [5].

Lemma 3. *Let P and P' be as above. Then:*

(i) *P' is stronger than P.*
(ii) *If $l_1 \neq l_2$ and $l_1 \neq \neg l_2$, then $P' = \langle \mathsf{stitch}(P_1, l_2, P_3), l_1, \mathsf{stitch}(P_2, l_2, P_3)\rangle$.*

Inductive proofs based on node counts will rely on the simple facts in the following lemma. Here and in the sequel, $|P|$ denotes the number of nodes in P and $|P|_c$ denotes the number of critical nodes in P.

Lemma 4. *Let P_1 and P_2 be arbitrary proof trees and l be an arbitrary literal. Let ϵ be 1 if l is uncolorable and 0 otherwise.*

(i) *If $\langle P_1, l, P_2\rangle$ is defined then $|\langle P_1, l, P_2\rangle| = |P_1| + |P_2| + 1$ and $|\langle P_1, l, P_2\rangle|_c = |P_1|_c + |P_2|_c + \epsilon$;*
(ii) *$|\mathsf{stitch}(P_1, l, P_2)| \leq |P_1| + |P_2| + 1$ and $|\mathsf{stitch}(P_1, l, P_2)|_c \leq |P_1|_c + |P_2|_c + \epsilon$.*

3 Obtaining Colorable Refutations

Theorem 1 tells us how to derive ground interpolants from colorable refutations. In this section, we show how and under what conditions it is possible to obtain colorable refutations from those produced by an SMT solver.

3.1 Prelude

As argued in §4, the only literals occurring in proof trees produced by SMT solvers, under standard assumptions, are (colorable) literals occurring in the input set $A \cup B$ or (dis)equalities between terms that occur in $A \cup B$. Thus, the only uncolorable atoms are equalities $x{=}y$, where x is A-colored and y is B-colored.

Figure 1 shows the basic transformation that removes one such equality from a proof tree. It uses equality interpolation (§2.3) to replace a lemma $\alpha \vee x{=}y$ containing the uncolorable equality $x{=}y$ with two colorable lemmas $\alpha \vee x{=}z$ and $\alpha \vee z{=}y$. Occurrences of the corresponding disequality $x \neq y$ are then split into $x \neq z \vee z \neq y$. This transformation can be applied repeatedly, under appropriate conditions discussed below, to eliminate all uncolorable equalities.

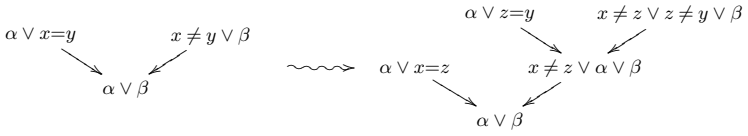

Fig. 1. Basic transformation to eliminate an uncolorable equality $x{=}y$.

Clearly, we have to assume that all theories be equality interpolating. Additionally, the use of equality interpolation imposes a hard constraint on the

proof trees modifiable by the basic transformation above: there must be at most one uncolorable equality in each leaf clause (see Figure 2). This restriction will define the class of almost-colorable refutations. Note that if all the theories are convex then the restriction causes no loss of generality. In a convex theory, if $\alpha \vee x{=}y \vee x'{=}y'$ is a lemma, then either $\alpha \vee x{=}y$ or $\alpha \vee x'{=}y'$ must be a lemma as well.

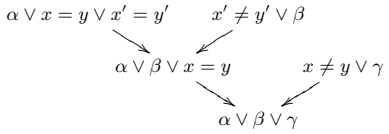

Fig. 2. Equality interpolation cannot be applied to the theory lemma $\alpha \vee x{=}y \vee x'{=}y'$ with two uncolorable equalities.

The method of [6] employs the basic transformation to eliminate all uncolorable equalities from *ie-local* refutations—those in which all uncolorable equalities are resolved before other literals. However, as witnessed by the example in Figure 3, ie-locality is not a necessary condition for the applicability of the basic transformation.

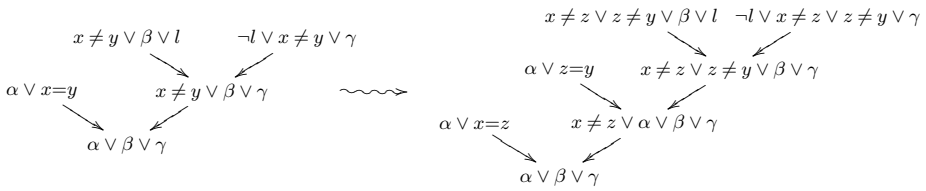

Fig. 3. Basic transformation applied to remove the uncolorable equality $x{=}y$ from a non-ie-local proof. The literal l is assumed colorable.

The real difficulty with producing colorable refutations from uncolorable ones is not the lack of ie-locality, but merges of uncolorable equalities. The example on the left in Figure 4 merges the equality $x{=}y$ from two leaves. If we perform equality interpolation on only one of the two occurrences of this equality in a leaf, we get a strictly weaker proof with the uncolorable equality $x{=}y$ still in the derived clause. If we perform equality interpolation on both occurrences and obtain distinct equality interpolants, then also the modified proof is strictly weaker than the original, irrespective of how we split the disequality $x \neq y$.

It can be shown that refutations that are almost-colorable and ie-local contain no merges of uncolorable equalities. For this reason, the approach of [6] insists on ie-locality. We place no such restriction, prefering to eliminate the problematic merges by changing the order of pivots as shown in Figure 4.

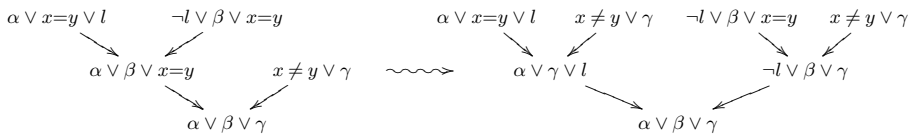

Fig. 4. Raising the merged pivot $x{=}y$ eliminates the merge.

3.2 The Colorability Theorem

Let $\mathcal{P}(A, B)$ be the set of all $(\mathcal{T}_1, \ldots, \mathcal{T}_n)$-proof trees from $A \cup B$ which use only theory lemmas satisfying the following conditions:

- (col_1) every uncolorable literal in the lemma is an equality or a disequality
- (col_2) at most one literal in the lemma is an uncolorable equality

We will call proofs in $\mathcal{P}(A, B)$ *almost-colorable*. Clearly, all colorable proof trees from $A \cup B$ are also almost-colorable.

Theorem 2. *Let the theories $\mathcal{T}_1, \ldots, \mathcal{T}_n$ be equality-interpolating. If $\mathcal{P}(A, B)$ contains a refutation, then it contains a colorable refutation.*

Proof. Since every literal that occurs in a refutation must be resolved upon at some node, the existence of an uncolorable (dis)equality in a refutation implies the existence of a critical node in it. Thus, to prove the theorem, it suffices to show that there exists a refutation with no critical nodes. We will establish this by proving the following more general statement: *If $P \in \mathcal{P}(A, B)$ has no uncolorable disequalities in its clause $[\![P]\!]$, then there exists a stronger proof tree $P' \in \mathcal{P}(A, B)$ with no critical nodes.* We prove this claim by well-founded induction over the relation $\prec$ defined by:

$$P \prec Q \quad \text{iff} \quad |P|_c < |Q|_c \;\text{ or }\; |P|_c = |Q|_c \text{ and } |P| < |Q|.$$

The proof breaks down into five cases. In all cases, it is easily verified that the offered proof tree P' belongs to $\mathcal{P}(A, B)$, either directly or using the simple fact that $\langle P_1, l, P_2 \rangle \in \mathcal{P}(A, B)$ if and only if $P_1, P_2 \in \mathcal{P}(A, B)$. So, we will focus only on verifying that P' is stronger than P and has no critical nodes.

Case 1: P is a single node. We can take P' to be P, which has no internal nodes and, hence, no critical nodes.

Case 2: $P = \langle P_1, l, P_2 \rangle$. We assume, without loss of generality, that l is an atom. Lemma 1 implies that there are no uncolorable disequalities in $[\![P_1]\!]$ and by Lemma 4 we infer that $P_1 \prec P$. Thus, the induction hypothesis applies to P_1 ensuring the existence of a proof tree P_1' that is stronger than P_1 and has no critical nodes. If there are no critical nodes in P_1, then we will let P_1' be P_1.

Case 2.1: $l \notin [\![P_1']\!]$. We can take P' to be P_1', which has no critical nodes. Since P_1' is stronger than P_1 and $l \notin [\![P_1']\!]$, it follows by Lemma 1 that P_1' is stronger than P.

Case 2.2: $l \in [\![P_1']\!]$.

Case 2.2.1: l is colorable. Lemma 1 then implies that there are no uncolorable disequalities in $[\![P_2]\!]$. Since $P_2 \prec P$ by Lemma 4, from the induction hypothesis we obtain a proof tree P_2' that is stronger than P_2 and contains no critical nodes. Let $P' = \mathsf{stitch}(P_1', l, P_2')$. Since l is colorable, it follows from Lemma 4 that P' does not contain any critical nodes, either. It also follows, by Lemma 2, that P' is stronger than P.

Case 2.2.2: l is uncolorable. By property (col_1), we have that l is an uncolorable equality $x{=}y$. We can infer from the absence of uncolorable disequalities in $[\![P]\!]$ and Lemma 1 that $x \neq y$ is the only uncolorable disequality in P_2.

Case 2.2.2.1: P_1' is a single node. Let $[\![P_1']\!] = \gamma \uplus (x{=}y)$. We know that there are no uncolorable disequalities in $[\![P_1']\!]$. This, together with uncolorability of $x{=}y$ and the fact $P_1' \in \mathcal{P}(A, B)$, implies that all the literals in γ are colorable. Now, $[\![P_1']\!]$ must be a theory lemma because $x{=}y$ is not colorable. Since our theories are assumed to be equality-interpolating, there exists an equality interpolant z for the clause $[\![P_1']\!]$. Let Q^x be the single-node proof tree with $[\![Q^x]\!] = \gamma \cup (x{=}z)$ and let Q^y be the single-node proof tree with $[\![Q^y]\!] = \gamma \cup (z{=}y)$. Note that Q^x and Q^y are colorable and, since they have no internal nodes, they have no critical nodes either.

Let $[\![P_2]\!] = \delta \uplus (x \neq y)$. From Lemma 5 below, we obtain a proof tree $P_2^* \in \mathcal{P}(A, B)$ such that $|P_2^*|_c \leq |P_2|_c$ and $[\![P_2^*]\!] \subseteq \delta \cup \{x \neq z, z \neq y\}$. Since $x \neq y$ is the only uncolorable disequality in $[\![P_2]\!]$, and the disequalities $x \neq z$ and $z \neq y$ are colorable, there can be no uncolorable disequalities in $[\![P_2^*]\!]$. Since the root is a critical node in P, we have by Lemma 4 that $|P_2|_c < |P|_c$. Thus, the induction hypothesis applies to P_2^*, yielding a proof tree P_2' stronger than P_2^* and without critical nodes. We take P' to be $\mathsf{stitch}(Q_1^x, x{=}z, \mathsf{stitch}(Q_1^y, z{=}y, P_2'))$. By Lemma 4 there are no critical nodes in P'. Since $[\![P']\!] \subseteq \gamma \cup \delta$, P' is stronger than P.

Case 2.2.2.2: $P_1' = \langle P_{11}, l', P_{12} \rangle$. Since there are no critical nodes in P_1' and no uncolorable disequalities in $[\![P_1']\!]$, it follows that the literal l' is colorable and, from Lemma 1, that there are no uncolorable disequalities in $[\![P_{11}]\!]$ and $[\![P_{12}]\!]$. Let $P^\dagger = \langle P_1', x{=}y, P_2 \rangle$. Note that $P^\dagger$ is well-defined (since $x{=}y \in [\![P_1']\!]$) and stronger than P (by Lemma 2). Note also that $x{=}y$ differs from l' and $\neg l'$ since l' is colorable and $x{=}y$ is uncolorable. We raise the pivot $x{=}y$ over l' in $P^\dagger$ to get $P^\ddagger = \mathsf{stitch}(Q_1, l', Q_2)$, where $Q_i = \mathsf{stitch}(P_{1i}, x{=}y, P_2)$ $(i = 1, 2)$. By Lemma 3, we have $P^\ddagger = \langle Q_1, l', Q_2 \rangle$.

We now show that the induction hypothesis applies to Q_i $(i = 1, 2)$. Since $x \neq y \in [\![P_2]\!]$, we have that Q_i is either P_{1i} or $\langle P_{1i}, x{=}y, P_2 \rangle$. Since $x \neq y$ is the only uncolorable disequality in $[\![P_2]\!]$ and there are no uncolorable disequalities in $[\![P_{1i}]\!]$, we can infer using Lemma 2 that there are no uncolorable disequalities in $[\![Q_i]\!]$. We also have (by Lemma 4) that $|Q_i|_c \leq 1 + |P_1'|_c + |P_2|_c = 1 + |P_2|_c$ and $|P|_c = 1 + |P_1|_c + |P_2|_c$. Thus, $|Q_i|_c \leq |P|_c$. Moreover, if $|Q_i|_c = |P|_c$ then we must have $|P_1|_c = 0$, in which case P_1' is P_1 (see Case 2.2) and by Lemma 4, we have $|Q_i| < |P|$. It follows that $Q_i \prec P$.

Thus, we have proof trees Q'_i that have no critical nodes and are stronger than Q_i. We take P' to be $\mathsf{stitch}(Q'_1, l', Q'_2)$. There are no critical nodes in P' (Lemma 4) and P' is stronger than $P^\ddagger$ (Lemma 2), which in turn is stronger than $P^\dagger$ (Lemma 3), which, as we have already noticed, is stronger than P. □

Lemma 5. *Let P be a proof in $\mathcal{P}(A, B)$, $x \neq y$ be an uncolorable disequality and z be an arbitrary term. Then, there exists $P^* \in \mathcal{P}(A, B)$ such that:*

(i) $[\![P^*]\!] \subseteq [\![P]\!] \cup \{x \neq z, z \neq y\} \setminus \{x \neq y\}$;
(ii) $|P^*|_c \leq |P|_c$.

Proof. We argue by induction on the number of nodes in P.

Case 1: $x \neq y$ does not occur in $[\![P]\!]$. Take P^* to be P.

Case 2: $[\![P]\!] = \delta \uplus (x \neq y)$.

Case 2.1: P is a single node. The uncolorability of $x \neq y$ implies that $[\![P]\!]$ is a theory lemma. Take P^* be the single node with $[\![P^*]\!] = \delta \cup \{x \neq z, z \neq y\}$. By the transitivity of equality, $[\![P^*]\!]$ is also a theory lemma.

Case 2.2: $P = \langle P_1, l, P_2 \rangle$. We know from Lemma 1 that $[\![P_1]\!] \subseteq \delta \cup \{x \neq y, l\}$ and $[\![P_2]\!] \subseteq \delta \cup \{x \neq y, \neg l\}$. By the induction hypothesis, there exist P_1^* and P_2^* such that $[\![P_1^*]\!] \subseteq \delta \cup \{x \neq z, z \neq y, l\}$, $[\![P_2^*]\!] \subseteq \delta \cup \{x \neq z, z \neq y, \neg l\}$ and P_1^*, P_2^* have no more critical nodes than P_1, P_2 respectively. Let $P^* = \mathsf{stitch}(P_1^*, l, P_2^*)$. It follows from Lemma 2 that $[\![P^*]\!] \subseteq \delta \cup \{x \neq z, z \neq y\}$. Finally, by Lemma 4, $|P^*|_c \leq \epsilon + |P_1^*|_c + |P_2^*|_c \leq \epsilon + |P_1|_c + |P_2|_c = |P|_c$, for suitable $\epsilon \in \{0, 1\}$. □

3.3 The Colorability Algorithm

The proofs of Theorem 2 and Lemma 5 are constructive and directly lead to Algorithm 1 and Algorithm 2. The algorithms use the following functions: is_lit_colorable tests if a literal is colorable; eq_interp computes an equality interpolant for the input clause; node creates a single-node proof annotated with the given clause.

Merges of uncolorable equalities have the potential to exponentially blow-up the size of $\mathsf{mk\_colorable}(P)$ because raising an uncolorable-equality pivot doubles the right subproof, as in Figure 4. The following result guarantees linear growth in the absence of these problematic merges.

Theorem 3. *If P is a refutation in $\mathcal{P}(A, B)$ such that there are no merges of uncolorable equalities in P, then $|\mathsf{mk\_colorable}(P)| \leq 2 \cdot |P|$.*

Proof. We will prove the following more general statement: *Let P be a proof in $\mathcal{P}(A, B)$ such that there are no uncolorable equalities in $[\![P]\!]$ and no merges of uncolorable equalities in P. Let $P' = \mathsf{mk\_colorable}(P)$. Then:*

(i) $|P'| \leq 2 \cdot |P|$;
(ii) If $P = \langle P_1, l, P_2 \rangle$ and P_1 has no critical nodes, then $|P'| \leq |P_1| + 2 \cdot |P_2| + 3$.

Algorithm 1 $\mathsf{mk\_colorable}(P)$

1: **if** $|P| = 1$ **then** $(*\,\text{Case } 1\,*)$
2: $\quad P' \leftarrow P$
3: **else** $(*\,\text{Case } 2\,*)$
4: $\quad$ **let** P **be** $\langle P_1, l, P_2 \rangle$
5: $\quad P'_1 \leftarrow \mathsf{mk\_colorable}(P_1)$
6: $\quad$ **if** $l \notin [\![P'_1]\!]$ **then** $(*\,\text{Case } 2.1\,*)$
7: $\quad\quad P' \leftarrow P'_1$
8: $\quad$ **else** $(*\,\text{Case } 2.2\,*)$
9: $\quad\quad$ **if** $\mathsf{is\_lit\_colorable}(l)$ **then** $(*\,\text{Case } 2.2.1\,*)$
10: $\quad\quad\quad P'_2 \leftarrow \mathsf{mk\_colorable}(P_2)$
11: $\quad\quad\quad P' \leftarrow \mathsf{stitch}(P'_1, l, P'_2)$
12: $\quad\quad$ **else** $(*\,\text{Case } 2.2.2\,*)$
13: $\quad\quad\quad$ **let** l **be** $x{=}y$
14: $\quad\quad\quad$ **if** $|P'_1| = 1$ **then** $(*\,\text{Case } 2.2.2.1\,*)$
15: $\quad\quad\quad\quad$ **let** $[\![P'_1]\!]$ **be** $\gamma \uplus x{=}y$
16: $\quad\quad\quad\quad z \leftarrow \mathsf{eq\_interp}(\gamma \uplus x{=}y)$
17: $\quad\quad\quad\quad Q^x \leftarrow \mathsf{node}(\gamma \cup x{=}z)$
18: $\quad\quad\quad\quad Q^y \leftarrow \mathsf{node}(\gamma \cup z{=}y)$
19: $\quad\quad\quad\quad P^*_2 \leftarrow \mathsf{split}(P_2, x \neq y, z)$
20: $\quad\quad\quad\quad P'_2 \leftarrow \mathsf{mk\_colorable}(P^*_2)$
21: $\quad\quad\quad\quad P' \leftarrow \mathsf{stitch}(Q^x, x{=}z, \mathsf{stitch}(Q^y, z{=}y, P'_2))$
22: $\quad\quad\quad$ **else** $(*\,\text{Case } 2.2.2.2\,*)$
23: $\quad\quad\quad\quad$ **let** P'_1 **be** $\langle P_{11}, l', P_{12} \rangle$
24: $\quad\quad\quad\quad Q_1 \leftarrow \mathsf{stitch}(P_{11}, x{=}y, P_2)$
25: $\quad\quad\quad\quad Q_2 \leftarrow \mathsf{stitch}(P_{12}, x{=}y, P_2)$
26: $\quad\quad\quad\quad Q'_1 \leftarrow \mathsf{mk\_colorable}(Q_1)$
27: $\quad\quad\quad\quad Q'_2 \leftarrow \mathsf{mk\_colorable}(Q_2)$
28: $\quad\quad\quad\quad P' \leftarrow \mathsf{stitch}(Q'_1, l', Q'_2)$
29: **return** P'

Algorithm 2 $\mathsf{split}(P, x \neq y, z)$

1: **if** $x \neq y \notin [\![P]\!]$ **then** $(*\,\text{Case } 1\,*)$
2: $\quad P^* \leftarrow P$
3: **else** $(*\,\text{Case } 2\,*)$
4: $\quad$ **let** $[\![P]\!]$ **be** $\delta \uplus (x \neq y)$
5: $\quad$ **if** $|P| = 1$ **then** $(*\,\text{Case } 2.1\,*)$
6: $\quad\quad P^* \leftarrow \mathsf{node}(\delta \cup \{x \neq z, z \neq y\})$
7: $\quad$ **else** $(*\,\text{Case } 2.2\,*)$
8: $\quad\quad$ **let** P **be** $\langle P_1, l, P_2 \rangle$
9: $\quad\quad P^*_1 \leftarrow \mathsf{split}(P_1, x \neq y, z)$
10: $\quad\quad P^*_2 \leftarrow \mathsf{split}(P_2, x \neq y, z)$
11: $\quad\quad P^* \leftarrow \mathsf{stitch}(P^*_1, l, P^*_2)$
12: **return** P^*

We will use the easily proven facts that $|\mathsf{split}(P, x \neq y, z)| \leq |P|$ and that if there are no critical nodes in P, then $\mathsf{mk\_colorable}(P) = P$. The proof will follow the structure of the proof of Theorem 2.
Case 1: Trivial.
Case 2.1: We have $|P'| = |P'_1|$.

(i) By the induction hypothesis, $|P'_1| \leq 2 \cdot |P_1|$. But $|P_1| < |P|$.
(ii) If P_1 has no critical nodes, then $P'_1 = P_1$.

Case 2.2.1:

(i) $|P'| \leq |P'_1| + |P'_2| + 1 \leq 2 \cdot |P_1| + 2 \cdot |P_2| + 1 = 2 \cdot (|P_1| + |P_2|) + 1$. We have $|P| = |P_1| + |P_2| + 1$. Thus, $|P'| \leq 2 \cdot (|P| - 1) + 1 < 2 \cdot |P|$.
(ii) If P_1 has no critical nodes, then $P'_1 = P_1$. Thus, $|P'| \leq |P_1| + |P'_2| + 1 \leq |P_1| + 2 \cdot |P_2| + 1$.

Case 2.2.2.1:

(i) $|P'| \leq |Q^x| + |Q^y| + |P'_2| + 2$. We have $|Q^x| = |Q^y| = 1$, and by induction hypothesis, $|P'_2| \leq 2 \cdot |P^*_2| \leq 2 \cdot |P_2|$. Thus $|P'| \leq 2 \cdot |P_2| + 4$. We also know that $|P_2| \leq |P| - 2$. Thus $|P'| \leq 2 \cdot |P|$.
(ii) If P_1 has no critical nodes, then $P'_1 = P_1$. The assumption for this case is that $|P'_1| = 1$. Thus, $|P'| \leq |P'_2| + 4 \leq |P_1| + 2 \cdot |P_2| + 3$.

Case 2.2.2.2: Without loss of generality, assume $x{=}y \notin [\![P_{11}]\!]$ and $x{=}y \in [\![P_{12}]\!]$. Thus, $Q_1 = P_{11}$ and $Q_2 = \langle P_{12}, l, P_2 \rangle$.

(i) $|P'| \leq |Q'_1| + |Q'_2| + 1$. Note that there are no critical nodes in either P_{11} or in P_{12}. Thus, $Q'_1 = P_{11}$ and $|Q'_2| \leq |P_{12}| + 2 \cdot |P_2| + 3$. Thus, $|P'| \leq |P_{11}| + |P_{12}| + 2 \cdot |P_2| + 3 = |P'_1| + 2 \cdot |P_2| + 2 \leq 2 \cdot (|P_1| + |P_2| + 1) = 2 \cdot |P|$.
(ii) Assume no critical nodes in P_1. Then $P'_1 = P_1 = \langle P_{11}, l, P_{12} \rangle$. Also, P_{11} and P_{12} have no critical nodes and $Q'_1 = P_{11}$, $|Q'_2| \leq |P_{12}| + 2 \cdot |P_2| + 3$. Thus, $|P'| \leq |Q'_1| + |Q'_2| + 1 \leq |P_{11}| + |P_{12}| + 2 \cdot |P_2| + 3 + 1 = |P_1| + 2 \cdot |P_2| + 3$.□

More substantial complexity analysis is left for future work. Our algorithms can be easily modified (by memoization) to operate on proof DAGs instead on proof trees. It would be particularly interesting to understand the complexity of these optimized versions.

4 Almost-Colorable Refutations from SMT Solvers

Modern SMT solvers integrate a SAT solver and several solvers for specific theories. An abstract model of an SMT solver that covers the essentials of the cooperation algorithm is given in [12] in the form of a transition system called NODPLL (Nelson-Oppen with DPLL), which in turn is an elaboration of the abstract system DPLL($\mathcal{T}$) of [16].

In this section, starting with a simplified (more abstract) version of the system NODPLL described in [12], we obtain the system NODPLL$^{\mathrm{pf}}$ which tracks

the derivations of all conflict clauses and thus produces $(\mathcal{T}_1, \ldots, \mathcal{T}_n)$-refutations when it finds that the input set of clauses is inconsistent.

The main parameters of the system NODPLL$^{\mathrm{pf}}$ are theories $\mathcal{T}_1, \ldots, \mathcal{T}_n$ with disjoint signatures $\Sigma_1, \ldots, \Sigma_n$. The union signature and the union theory will be denoted Σ and $\mathcal{T}$ respectively. Additional parameters of NODPLL$^{\mathrm{pf}}$ are a set L of Σ-literals and a set E of equalities between Σ-terms. Intuitively, the set L consists of literals that the SAT solver can decide on, and E is the set of equalities that theory solvers may share without sharing them with the SAT solver. It is not required that L and E be disjoint. (In extensions of the system, one can also promote L and E from parameters to system variables, adding rules to grow them dynamically.)

NODPLL$^{\mathrm{pf}}$ is a transition system over states of the form $\langle \mathsf{P}, \mathsf{M}, \mathsf{C} \rangle$ where *(i)* P is a set of proof trees over Σ-clauses; *(ii)* M is a *checkpointed sequence*, any element of which is either the special symbol $\Box$, or a literal from $\mathsf{L} \cup \mathsf{E}$; *(iii)* C, the state's *conflict proof tree*, is either a proof tree for a clause that is a subset of $\mathsf{L} \cup \mathsf{E}$, or the special symbol none, denoting the absence of conflict.

As before, we use the notation $\mathsf{node}(\gamma)$ for the proof tree with a single node whose associated clause is γ.

The input to NODPLL$^{\mathrm{pf}}$ is a set S of ground $\mathcal{T}$-clauses. With a given S, the initialization procedure specifies the sets L and E, and an initial state of NODPLL$^{\mathrm{pf}}$. The initial state naturally has $\mathsf{P} = \{\mathsf{node}(\gamma) \,|\, \gamma \in S\}$, M equal to the empty sequence, and $\mathsf{C} = \mathsf{none}$. As for the parameter literal sets L and E, there are two main options. To define them, let L_S denote the set of all literals that occur in S, and let E_S be the set of all equalities between distinct terms that occur in S. For *Nelson-Oppen initialization*, we take $\mathsf{L} = L_S^{\pm 1}$ and $\mathsf{E} = E_S$. For *DTC initialization*, we take $\mathsf{L} = L_S^{\pm 1} \cup E_S^{\pm 1}$ and $\mathsf{E} = \varnothing$. (The notation $X^{\pm 1}$ stands for the set that contains the literals of X and their negations.) To be general, we will assume only that $\mathsf{L} \subseteq L_S^{\pm 1} \cup E_S^{\pm 1}$ and $\mathsf{E} \subseteq E_S$.

The transition rules of NODPLL$^{\mathrm{pf}}$ are given in Figure 5. The index i ranges over $\{0, \ldots, n\}$. The symbol $\models_i$ stands for the theory entailment $\models_{\mathcal{T}_i}$ in the case when $i > 0$. For $i = 0$, the symbol stands for the propositional entailment from a single clause of P. More precisely, the condition $\mathsf{M} \models_0 l$ in the rule Infer_0 stands for "there exist a proof tree $P \in \mathsf{P}$ such that $[\![P]\!] = \{\neg l_1, \ldots, \neg l_k, l\}$ and $l_1, \ldots, l_k \in \mathsf{M}$". Similarly, $l_1, \ldots, l_k \models_0 \mathsf{false}$ and $l_1, \ldots, l_k \models_0 l$ in the rules $\mathsf{Conflict}_0$ and $\mathsf{Explain}_0$ stand for the existence of $P \in \mathsf{P}$ satisfying $[\![P]\!] = \{\neg l_1, \ldots, \neg l_k\}$ and $[\![P]\!] = \{\neg l_1, \ldots, \neg l_k, l\}$ respectively.

When $i > 0$, the notation $\mathsf{pf}_i\gamma$ is synonymous with $\mathsf{node}(\gamma)$. As for $\mathsf{pf}_0\gamma$, it is used only in rules $\mathsf{Conflict}_0$ and $\mathsf{Explain}_0$, and it stands for a proof tree $P \in \mathsf{P}$ such that $[\![P]\!] = \gamma$. In view of the definitions in the previous paragraph, such a proof tree P always exists.

The number of occurrences of $\Box$ in M is the *current decision level*. Thus, we can write $\mathsf{M} = \mathsf{M}^{\langle 0 \rangle} \,\Box\, \mathsf{M}^{\langle 1 \rangle} \,\Box \cdots \Box\, \mathsf{M}^{\langle d \rangle}$, where d is the current decision level, and $\Box$ does not occur in any $\mathsf{M}^{\langle k \rangle}$. It is an invariant that for every $k > 0$, $\mathsf{M}^{\langle k \rangle}$ is non-empty. The first element of $\mathsf{M}^{\langle k \rangle}$ $(k > 0)$ is the k^{th} *decision literal* of M. By $\mathsf{M}^{[k]}$, where $0 \leq k \leq d$, we denote the prefix $\mathsf{M}^{\langle 0 \rangle} \,\Box \cdots \Box\, \mathsf{M}^{\langle k \rangle}$ of M.

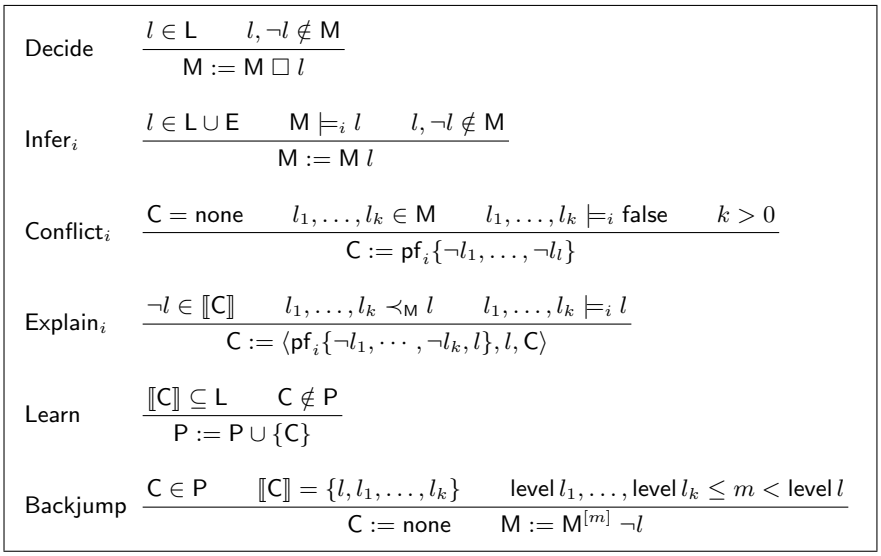

Fig. 5. Rules of NODPLL$^{\mathrm{pf}}$. Above each line is the rule's *guard*, below is its *action*.

The rule $\mathsf{Explain}_i$ uses the notation $l \prec_{\mathsf{M}} l'$; by definition, this means that both literals are in M and the (unique) occurrence of l precedes in M the (unique) occurrence of l'. For correctness of this definition, we need to know that *any literal can occur at most once* in M, which is another easily verified invariant of NODPLL$^{\mathrm{pf}}$. Finally, the function level used in the $\mathsf{Backjump}$ rule is defined only for literals that occur in M; for these literals $\mathsf{level}\, l = k$ holds if l occurs in $\mathsf{M}^{\langle k \rangle}$.

A NODPLL$^{\mathrm{pf}}$ *execution* is a finite or infinite sequence $s_0, s_1, \ldots$ such that s_0 is an initial state and each state s_{i+1} is obtained from s_i by the application of one of the transition rules of the system. We can prove the following lemma by induction on the length of execution sequences.

Lemma 6. *If* NODPLL$^{\mathrm{pf}}$ *is given a clause set S as input, then, in any state,* C *is either* none *or a $(\mathcal{T}_1, \ldots, \mathcal{T}_n)$-proof tree from S.*

The results of [12] for the original NODPLL system apply to NODPLL$^{\mathrm{pf}}$ as well, with straightforward modifications of the proofs. Specifically, one can prove that the system NODPLL$^{\mathrm{pf}}$ is *terminating*: every execution is finite and ends in a state in which $\mathsf{C} = \mathsf{none}$ or $[\![\mathsf{C}]\!] = \varnothing$. The *soundness* of NODPLL$^{\mathrm{pf}}$ is actually a consequence of Lemma 6: if the system reaches a state in which C is a refutation ($[\![\mathsf{C}]\!] = \varnothing$), then S is $\mathcal{T}$-unsatisfiable. There are two *completeness* results: [4] if on an input S the system terminates in a state in which $\mathsf{C} = \mathsf{none}$, then S is $\mathcal{T}$-satisfiable, provided *(i)* the system is given the Nelson-Oppen initialization, and all the $\mathcal{T}_i$ are convex; or *(ii)* the system is given the DTC initialization.

[4] In the context of [12], we assume that the theories are *parametric*; for the classical first-order combination, we need to assume that the theories are stably-infinite [2].

Consider now the colorability of proof trees C of our system. The initialization assumption $\mathsf{L} \subseteq L_S^{\pm 1} \cup E_S^{\pm 1}$ and colorability of all literals in L_S (each of them occurs in A or in B) imply that the only uncolorable literals in $\mathsf{L} \cup \mathsf{E}$ are equalities from E_S or their negations. Thus, proof trees C always satisfy the property (col_1).

One way to satisfy (col_2) is to ensure that all literals in L are colorable; for instance, by initializing the system with L being the union of $L_S^{\pm 1}$ and all colorable (dis)equalities from $E_S^{\pm 1}$. To see that (col_2) holds in this case, note first that (by induction) all uncolorable literals in M are equalities from $\mathsf{E} \setminus \mathsf{L}$. This ensures the clause of C introduced by $\mathsf{Conflict}_i$ contains no uncolorable equalities, and clauses introduced by $\mathsf{Explain}_i$ contain at most one uncolorable equality. Thus, (col_2) is satisfied, but note that the restriction we put on L makes NODPLL$^{\mathrm{pf}}$ potentially incomplete.

Another way to guarantee proof trees C satisfying (col_2) is to run the system NODPLL$^{\mathrm{pf}}$ with the following *convexity restriction*: allow rule $\mathsf{Conflict}_i$ to fire only when at most one of the literals $l_1, \ldots, l_k$ is a disequality and allow rule $\mathsf{Explain}_i$ to fire only when none of the literals $l_1, \ldots, l_k$ is a disequality. It is easy to see that if all theories $\mathcal{T}_i$ are convex, then the convexity restriction does not jeopardize the completeness of NODPLL$^{\mathrm{pf}}$.

Theorem 4. *Suppose $S = A \cup B$ is given as input to* NODPLL$^{\mathrm{pf}}$. *Suppose, in addition, that either* (a) *the system is run with the convexity restriction; or* (b) *the system is initialized so that all literals in L are colorable. Then, in all reachable states, the proof tree C is in $\mathcal{P}(A, B)$.*

Proof. Sketched in the preceding paragraphs. □

5 Conclusion

We have presented a simple approach for the generation of ground interpolants by SMT solvers supporting multiple theories. Our main contribution is an algorithm that transforms any *almost-colorable* refutation into one that is *colorable* and thus suitable for straigthforward interpolant extraction using known algorithms.

The definition of almost-colorable refutations is minimally demanding. We show that modern SMT solvers can produce such refutations with the slightest restrictions on their search strategy. What constitutes a good search strategy for interpolation remains an open question, but by being more general than previous approaches, we enable the design of more efficient interpolating SMT solvers.

The colorability algorithm uses a sequence of elementary proof transformations to convert an almost-colorable refutation into a colorable one. There is some flexibility in the order in which these transformations are applied. Our particular choice of the colorability algorithm ensures that for a subset of almost-colorable refutations—including the class of ie-local refutations that could be used with previous methods for ground interpolation—we at most double the size of the input tree. In practice, however, proofs are represented compactly as DAGs. More work is required to understand the effect of various transformation choices on DAG size.

Acknowledgment. We thank Alexander Fuchs, Jim Grundy and anonymous reviewers for suggestions that helped improve the paper.

References

1. P. B. Andrews. Resolution with merging. *J. ACM*, 15(3):367–381, 1968.
2. C. Barrett et al. Satisfiability Modulo Theories. In A. Biere et al., editors, *Handbook of Satisfiability*, pp. 825–885. IOS Press, 2009.
3. D. Beyer, D. Zufferey, and R. Majumdar. CSIsat: Interpolation for LA+EUF. In *CAV*, vol. 5123 of *LNCS*, pp. 304–308. Springer, 2008.
4. M. Bozzano et al. Efficient theory combination via Boolean search. *Information and Computation*, 204(10):1493–1525, 2006.
5. H. K. Büning and T. Lettmann. *Propositional Logic: Deduction and Algorithms*. Cambridge University Press, New York, NY, USA, 1999.
6. A. Cimatti, A. Griggio, and R. Sebastiani. Efficient interpolant generation in Satisfiability Modulo Theories. In *TACAS*, vol. 4963 of *LNCS*, pp. 397–412. Springer, 2008.
7. L. de Moura and N. Bjørner. Model-based theory combination. *ENTCS*, 198:37–49, 2008.
8. B. Dutertre and L. de Moura. A fast linear-arithmetic solver for DPLL(T). In *CAV*, vol. 4144 of *LNCS*, pp. 81–94. Springer, 2006.
9. A. Fuchs et al. Ground interpolation for the theory of equality. In *TACAS*, volume 5505 of *LNCS*, pp. 413–427. Springer, 2009.
10. T. A. Henzinger et al. Abstractions from proofs. In *POPL*, pp. 232–244. ACM, 2004.
11. R. Jhala and K. L. McMillan. Interpolant-based transition relation approximation. In *CAV*, vol. 3576 of *LNCS*, pp. 39–51. Springer, 2005.
12. S. Krstić and A. Goel. Architecting solvers for SAT Modulo Theories: Nelson-Oppen with DPLL. In *FroCoS*, vol. 4720 of *LNCS*, pp. 1–27. Springer, 2007.
13. K. L. McMillan. Interpolation and SAT-based model checking. In *CAV*, vol. 2725 of *LNCS*, pp. 1–13. Springer, 2003.
14. K. L. McMillan. An interpolating theorem prover. *Theoretical Computer Science*, 345(1):101–121, 2005.
15. K. L. McMillan. Lazy abstraction with interpolants. In *CAV*, vol. 4144 of *LNCS*, pp. 123–136. Springer, 2006.
16. R. Nieuwenhuis, A. Oliveras, and C. Tinelli. Abstract DPLL and abstract DPLL modulo theories. In *LPAR*, vol. 3452 of *LNCS*, pp. 36–50. Springer, 2005.
17. A. Rybalchenko and V. Sofronie-Stokkermans. Constraint solving for interpolation. In *VMCAI*, vol. 4349 of *LNCS*, pp. 346–362. Springer, 2007.
18. G. Yorsh and M. Musuvathi. A combination method for generating interpolants. In *CADE*, vol. 3632 of *LNCS*, pp. 353–368. Springer, 2005.