A Combination Method for Generating Interpolants

Greta Yorsh[1] and Madanlal Musuvathi[2]

[1] School of Comp. Sci., Tel Aviv Univ., gretay@post.tau.ac.il,
[2] Microsoft Research, Redmond, madanm@microsoft.com

Abstract. We present a *combination* method for generating interpolants for a class of first-order theories. Using interpolant-generation procedures for individual theories as black-boxes, our method modularly generates interpolants for the combined theory. Our combination method applies for a broad class of first-order theories, which we characterize as *equality-interpolating* Nelson-Oppen theories. This class includes many useful theories such as the quantifier-free theories of uninterpreted functions, linear inequalities over reals, and Lisp structures. The combination method can be implemented within existing Nelson-Oppen-style decision procedures (such as Simplify, Verifun, ICS, CVC-Lite, and Zap).

1 Introduction

Given two logical formulas A and B such that $A \wedge B$ is unsatisfiable, an interpolant I is a formula such that (i) A implies I, (ii) $I \wedge B$ is unsatisfiable, and (iii) every non-logical symbol that appears in I appears in both A and B. Craig interpolation theorem [2], a classic result in logic, proves the existence of interpolants for all first-order formulas A and B.

Motivation While interpolation theorems are of great theoretical significance, our interest in interpolants is particularly motivated by their use in program analysis and model checking [12, 11, 13]. When A represents the current state of a system, and B represents the error state of the system, an interpolant I for A and B can be used as a *goal-directed* over-approximation of A. [12] uses this technique to achieve faster termination while model checking finite state systems, and [13] explores the possibility of using interpolants for model checking infinite systems. Such applications typically require an efficient procedure to generate an interpolant of A and B from the proof of unsatisfiability of $A \wedge B$. Existing procedures are either too expensive or work only for specific first-order theories (see Sec. 6).

In this paper, we provide a novel *combination* method for generating interpolants for a class of first-order theories. Using interpolant-generation procedures for component theories as black-boxes, this method generates interpolants for formulas in the combined theory. Provided the individual procedures for the

component theories can generate interpolants in polynomial time, our method generates interpolants for the combined theory in polynomial time.

Our combination method relies on the Nelson-Oppen [14] framework for combining decision procedures. In this framework, the decision procedures for component theories communicate by propagating entailed equalities. The crucial idea behind our combination method is to associate a *partial interpolant* (Sec. 3.1) with each propagated equality. Whenever a component theory propagates an equality, the combination method uses the interpolant-generation procedure for that theory to generate the partial interpolant for the equality. When a theory detects a contradiction, the combination method uses the partial interpolants of all propagated equalities to compute the interpolant for the input formulas.

Our method places some restrictions on the theories that can be combined. The Nelson-Oppen combination method requires that the component theories have disjoint signatures and be *stably-infinite* [14, 16]. Our method naturally inherits these restrictions. Additionally, our combination method restricts the form of equalities that can be shared by the component theories. Specifically, if a propagated equality contains a symbol that appears only in the input formula A, then it does not contain symbols that appear only in B, and vice versa. We show that this restricted form of equality propagation is sufficient for a class of theories, which we characterize as *equality-interpolating* theories (Sec. 4). Many useful theories including the quantifier-free theories of uninterpreted functions, linear arithmetic, and Lisp structures are equality-interpolating, and thus can be combined with our method.

Our method handles arbitrary quantifier-free input formulas. To handle Boolean structure in the input formula and to extend the combination to non-convex theories [14], we use an extended version of Pudlák's algorithm for generating interpolants for the propositional part. To show correctness of interpolants generated this way, we give an alternative explanation of Pudlák's algorithm based on partial interpolants.

The combination method has definite advantages over an interpolant generation procedure that is specific to a particular theory [17] or specific to a particular combination of theories [13]. First, by being modular, this method greatly simplifies the exposition, the proof of correctness, and the implementation of interpolant-generation procedures. More importantly, the combination method makes it easy to incrementally extend interpolant generation for additional theories. From a practical perspective, our combination method can be easily integrated with existing Nelson-Oppen-style decision procedures, such as [3, 1, 4, 5], greatly enhancing their utility for program analysis.

In summary, this paper makes the following contributions. First, the paper presents an efficient method to generate interpolants for a general class of first-order theories, namely the union of equality-interpolating theories. Second, this paper shows that the classic Nelson-Oppen framework for combining decision procedures can be extended in a novel way to combine interpolant-generation procedures. Finally, we show that the basic combination algorithm can be generalized to work for both convex and non-convex theories. Due to a lack of space,

this version does not contain proofs and contains only a partial discussion of the results. For a full version, the reader is referred to our technical report [19].

2 Preliminaries

Throughout this paper we use A and B to denote logical formulas of interest, restricting the syntax of A and B in different sections. Free variables in all formulas are implicitly existentially quantified, as we are checking satisfiability. We use *symbol* to refer to non-logical symbols and variables.[3]

Let $\mathcal{V}(\phi)$ be the set of symbols that appear in a formula or a term ϕ. Given formulas A and B, a symbol is A-local if it is in $\mathcal{V}(A) - \mathcal{V}(B)$. Similarly, a symbol is B-local if it is in $\mathcal{V}(B) - \mathcal{V}(A)$. A symbol is AB-common if it is in $\mathcal{V}(A) \cap \mathcal{V}(B)$. A formula or a term ϕ is AB-pure when either $\mathcal{V}(\phi) \subseteq \mathcal{V}(A)$ or $\mathcal{V}(\phi) \subseteq \mathcal{V}(B)$. Otherwise, ϕ is AB-mixed. Note that an AB-mixed formula or term contains at least one A-local symbol and at least one B-local symbol. Throughout this paper, we use a to refer to an A-local variable, b to refer to a B-local variable, and c, x and y to refer to a AB-common variables.

We use the standard notations $\vdash$, $\bot$, and $\top$ for entailment, contradiction, and tautology.

Definition 1. (Craig interpolant) *Given two first-order logical formulas A and B such that $A \wedge B \vdash \bot$, an interpolant for $\langle A, B \rangle$ is a first-order formula I such that (i) $A \vdash I$, (ii) $I \wedge B \vdash \bot$, and (iii) I refers only to AB-common symbols.*

Example 1. A is $(a = c) \wedge (f(c) = a)$ and B is $(c = b) \wedge \neg(b = f(c))$. The variables a, b, c are respectively A-local, B-local, AB-common variables, and f is an AB-common function symbol. The interpolant $f(c) = c$ for $\langle A, B \rangle$ involves only AB-common symbols f and c. By definition, A and B contain only AB-pure terms and formulas, however, AB-mixed terms and formulas may appear in the proof of unsatisfiability of $A \wedge B$. In this example, an AB-mixed equality $a = b$ can be generated in a proof of unsatisfiability of $A \wedge B$, as discussed later.

2.1 Theory-Specific Interpolants

As opposed to a well-known definition of Craig interpolants (Def. 1), interpolants for a specific first-order theory can be defined in several ways. In this section, we provide the definition used in this paper, and discuss alternative definitions in [19]. We adapt Def. 1 to the context of a specific first-order theory T. We use $\vdash_T$ to denote entailment in theory T. A theory T can contain uninterpreted

[3] Like much literature on decision procedures and model-checking, we use the term "variables" to refer to what in logic and theorem proving are "free constants." Similarly, "variables" in quantifier-free formulas refer to the corresponding Skolem-constants, as the formulas are implicitly existentially quantified.

symbols (e.g., uninterpreted functions), as well as a designated set of *interpreted* symbols (with intended meaning in T, or implicitly defined by T). [4]

Definition 2. (theory-specific interpolant) *Let T be a first-order theory of a signature Σ and let $\mathcal{L}$ be the class of quantifier-free Σ-formulas. Let $\Sigma_T \subseteq \Sigma$ denote a designated set of interpreted symbols in T. Let A and B be formulas in $\mathcal{L}$ such that $A \wedge B \vdash_T \bot$, i.e., the formula $A \wedge B$ is unsatisfiable in the theory T. We define a theory-specific interpolant for $\langle A, B\rangle$ in T to be a formula I in $\mathcal{L}$ such that (i) $A \vdash_T I$, (ii) $I \wedge B \vdash_T \bot$, and (iii) I refers only to AB-common symbols, and symbols in Σ_T (interpreted by the theory T).*

This definition differs from the traditional notion of an interpolant in two important ways. First, we require a theory-specific interpolant to be a quantifier-free formula. Second, a theory-specific interpolant can contain a symbol interpreted by the theory, even when if the symbol is A-local, or B-local or does not appear at all in A and B. The following example and discussion motivates these differences.

Example 2. Let A be $c_2 = \texttt{car}(c_1) \wedge c_3 = \texttt{cdr}(c_1) \wedge \neg\texttt{atom}(c_1)$ and B be $\neg c_1 = \texttt{cons}(c_2, c_3)$ in the theory of Lisp structures [15]. This theory interprets all function symbols that appear in this example, i.e., $\Sigma_T = \{\texttt{car}, \texttt{cdr}, \texttt{cons}, \texttt{atom}\}$. The variables c_1, c_2, and c_3 are AB-common, `cons` is B-local, and other function symbols are A-local. $A \wedge B$ is unsatisfiable in the theory of Lisp structures because A entails $c_1 = \texttt{cons}(c_2, c_3)$ using the axiom: $\forall x, y, z : \neg\texttt{atom}(x) \Rightarrow \texttt{cons}(\texttt{car}(x), \texttt{cdr}(x)) = x$. According to Def. 2, $c_1 = \texttt{cons}(c_2, c_3)$ is a theory-specific interpolant for $\langle A, B\rangle$. Note that if we do not allow `cons`, `car`, `cdr`, and `atom` to appear in the interpolant, then there is no first-order formula that is an interpolant for $\langle A, B\rangle$.

A theory T has *no* interpolants (or does not have the interpolation theorem), if there exists a pair of input formulas A and B in $\mathcal{L}$ such that $A \wedge B \vdash_T \bot$ but there exists no formula in $\mathcal{L}$ that satisfies conditions (i)-(iii). If we allow any first-order Σ-formula as a theory-specific interpolant (instead of a quantifier-free Σ-formula), then every theory has theory-specific interpolants, as follows from Craig interpolation theorem. In practice however, we are interested in quantifier-free interpolants to guarantee that the satisfiability checks involving interpolants are complete, say in the subsequent stages of a program analysis. [5]

If we strengthen requirement (iii) from Def. 2 to eliminate interpreted symbols as well, then many interesting theories do not have interpolants (even if the generated interpolants are not restricted to quantifier-free formulas). Example 2 demonstrates that the theory of Lisp structures does not have interpolants under

[4] The details of how T is defined (e.g., set of axioms, set of models) are not essential to our method.

[5] In general, the language $\mathcal{L}$ of input formulas A and B is not necessarily the same as the language of the generated interpolants. For example, one could require quantifier-free formulas as input, but allow that generated interpolants contain quantifiers. Our method applies to this generalized definition of theory-specific interpolants [19].

this stronger requirement. However, a weaker requirement, as stated in Def. 2, is sufficient for our purposes, because in program analysis interpolants are used to eliminate state information, encoded by uninterpreted symbols, whereas interpreted symbols encode persistent semantics of statements, such as arithmetic operations and memory manipulations.

2.2 Interpolants for Combined Theories

In this paper, we address the problem of computing interpolants for a combined theory T. Without loss of generality, let T be a combination of two theories T_1 and T_2.[6] Let T_i be a first-order theory of signature Σ_i, with a set of interpreted symbols $\Sigma_{T_i} \subseteq \Sigma_i$, and $\mathcal{L}_i$ be a class of Σ_i-formulas, for $i = 1, 2$. The signature Σ of the combined theory T is a union of Σ_1 and Σ_2; also, the set of interpreted symbols of T is $\Sigma_T = \Sigma_{T_1} \cup \Sigma_{T_2}$. Let $\mathcal{L}$ be a class of Σ-formulas.

The input of the combination method consists of two $\mathcal{L}$-formulas A and B. Note that the input may contain *mixed* terms from both Σ_1 and Σ_2, but it does not contain AB-mixed terms, by definition. An interpolant for $\langle A, B\rangle$ in the combined theory T is an $\mathcal{L}$-formula which may contain mixed terms from both Σ_1 and Σ_2, but contains only AB-common uninterpreted symbols, or symbols interpreted by T_1 and T_2.

Example 3. Consider a combination of the theory of uninterpreted functions and the theory of linear inequalities (where the symbols $\{+, <, \leq\}$ have the standard interpretation over reals) with the input formulas:

$$A \stackrel{\text{def}}{=} (f(x_1) + x_2 = x_3) \wedge (f(y_1) + y_2 = y_3) \wedge (y_1 \leq x_1)$$
$$B \stackrel{\text{def}}{=} (x_2 = g(b)) \wedge (y_2 = g(b)) \wedge (x_1 \leq y_1) \wedge (x_3 < y_3)$$

The first subformula of A contains a mixed term, with both f from the theory of uninterpreted functions and $+$ from the theory of linear inequalities.

We assume that T_1 and T_2 are stably-infinite theories with disjoint signatures, i.e., the only common symbol for Σ_1 and Σ_2 is equality. Each T_i has a decision procedure for satisfiability of a (quantifier-free) conjunction of Σ_i-literals (a literal is an atomic formula or its negation). These are standard requirements of the component theories in the Nelson-Oppen framework. In addition, we assume that each T_i has an efficient interpolant generation procedure that takes as input a pair of conjunctions of Σ_i-literals A_i and B_i, (i.e., A_i and B_i are pure Σ_i formulas). It returns an $\mathcal{L}_i$-formula as a theory-specific interpolant for A_i and B_i. Finally, we make the assumption that each T_i is an equality-interpolating theory; we explain and justify it in the next sections.

[6] As usual, combination of theories means union of axioms or, equivalently, intersection of sets of models.

3 The Combination Method

This section deals with the simple case in which (i) input formulas are conjunctions of pure literals, and (ii) all theories are convex, i.e., if a disjunction of equalities between variables is entailed, then at least one of the disjuncts is entailed [14]. These restrictions greatly simplify our description, while capturing the intuition behind our algorithm; they are relaxed in the following sections.

Purification Our method uses the Nelson-Oppen framework [14] to compute an interpolant for the combined theory. We assume that the unsatisfiability of input formula ψ was proved by a Nelson-Oppen procedure. The first step of the Nelson-Oppen procedure is *purification*. Given a mixed formula ψ, it constructs an equisatisfiable formula $\psi_1 \wedge \psi_2$, where ψ_i consists only of pure Σ_i-literals. Purification introduces new variables to replace terms of one signature that appear as sub-terms of terms in the other signature. Equalities defining these variables are added to the input formulas.

In our setting, the input ψ is a conjunction $A \wedge B$. We purify A and B separately. This guarantees that the new variables generated by the purification of A do not appear in the interpolant, because these variables are A-local. The result of purification of A is $A_1 \wedge A_2$ such that A_i contains only symbols from Σ_i and $A_1 \wedge A_2$ is satisfiable if and only if A is satisfiable; similarly, for B.

Example 4. After purifying A and B from Example 3 separately, we have that $A = A_{UIF} \wedge A_{LI}$ and $B = B_{UIF} \wedge B_{LI}$, where

$$
\begin{array}{l}
A_{UIF} \stackrel{\text{def}}{=} a_1 = f(x_1) \wedge a_2 = f(y_1) \\
A_{LI} \stackrel{\text{def}}{=} a_1 + x_2 = x_3 \wedge a_2 + y_2 = y_3 \wedge y_1 \leq x_1 \\
B_{UIF} \stackrel{\text{def}}{=} x_2 = g(b) \wedge y_2 = g(b) \\
B_{LI} \stackrel{\text{def}}{=} x_1 \leq y_1 \wedge x_3 < y_3
\end{array}
$$

a_1 and a_2 are A-local variables, b is a B-local variables, and f and g are A-local and B-local function symbols, respectively. We will use this as a running example.

Equality Propagation Let A and B be conjunctions of pure literals in the signature Σ of the combined theory T. A is $A_1 \wedge A_2$ such that A_i contains only symbols from Σ_i; similarly, for B. Let $\psi_i \stackrel{\text{def}}{=} A_i \wedge B_i$, for $i = 1, 2$. Note that ψ_i is a pure formula in Σ_i, but it is not AB-pure. Suppose that a Nelson-Oppen procedure shows the unsatisfiability of $\psi_1 \wedge \psi_2$ in T. It generates the set of equalities between variables, denoted by Eq.[7] Eq is sufficient to show the unsatisfiability of $A \wedge B$ using only one of the theories; assume, without loss of generality, that this theory is T_1. That is, $A \wedge B \vdash_T \bot$ follows from the fact that $Eq \wedge A_1 \wedge B_1 \vdash_{T_1} \bot$.

[7] [14] shows that it is sufficient to propagate only equalities between variables.

Example 5. The input $A \wedge B$ from Example 4 is not satisfiable, because $A \wedge B \vdash_T x_1 = y_1 \wedge a_1 = a_2 \wedge x_2 = y_2 \wedge x_3 = y_3$, which contradicts $x_3 < y_3$ from B. The set of equalities $Eq = \{x_1 = y_1, a_1 = a_2, x_2 = y_2\}$ is sufficient to derive a contradiction using only the theory of linear inequalities: $A_{LI} \wedge B_{LI} \wedge Eq \vdash_{LI} \bot$. An interpolant for $\langle A, B \rangle$ is $y_1 < x_1 \vee x_2 - y_2 = x_3 - y_3$. Note that the symbols g and f are eliminated because they denote local uninterpreted functions, but theory-specific interpreted functions $+$ and $<$ are not eliminated, recall the discussion of Def. 2(iii).

Overview of our combination method The idea is to use an interpolant generated by T_1 from a proof of unsatisfiability of $Eq \wedge A_1 \wedge B_1$, to generate an interpolant for $\langle A, B \rangle$ in the combined theory T.

The interpolant-generation procedure for T_1 takes as input two formulas A' and B' for which the conjunction $A' \wedge B'$ is not satisfiable. In our case, the unsatisfiable conjunction is $Eq \wedge A_1 \wedge B_1$. The question is how to split it into two formulas, A' and B'. The condition for splitting is that the common symbols for $\langle A', B' \rangle$ should be (a subset of) AB-common symbols, because we would like to use the resultant interpolant for $\langle A', B' \rangle$ as a part of an interpolant for the original A and B.

Suppose that Eq contains only AB-pure equalities. We split Eq into an A-part and a B-part: all the equalities from Eq that involve A-local symbols are added to the A-part; the B-part contains the rest of Eq.[8] We define A' to be a conjunction of A_1 and the A-part of Eq, and similarly for B'. Now, we can generate an interpolant for $\langle A', B' \rangle$ in theory T_1, using the interpolant generation procedure for T_1, as we planned.

It is important to note that the theory-specific interpolant for $\langle A', B' \rangle$ in theory T_1 is not an interpolant for the input formula $\langle A, B \rangle$ in the combined theory T. It uses only AB-common symbols, i.e., satisfies property (iii) of Def. 2, but it need not satisfy the properties (i) and (ii). The reason, intuitively, is that A does not imply A', because A' contains equalities which cannot be derived without information from B, as shown in the following example:

Example 6. In Example 5, the theory of linear inequalities derives a contradiction from $A' = A_{LI} \wedge a_1 = a_2$ and $B' = B_{LI} \wedge x_1 = y_1 \wedge x_2 = y_2$. The theory-specific partial interpolant for $\langle A', B' \rangle$ is $x_2 - y_2 = x_3 - y_3$. It is *not* an interpolant for the input $\langle A, B \rangle$, because A does not entail $x_2 - y_2 = x_3 - y_3$ (in the combined theory), as A alone does not entail $a_1 = a_2$.

To address this problem, we attach, for each propagated equality, additional information, called "partial interpolant". This notion is formally defined in

[8] For equalities with only AB-common variables, the question as to whether to add them "to B or not to B" can be answered in either way, as long as the answer is always the same for a given variable, and consistent for all equalities that appear in the proof of unsatisfiability. This gives us some control over how precise the interpolant is (how "close" it is to A or B), but we have not explored this direction yet. In all our examples, we add AB-common equalities to the B-part.

Sec. 3.1, and used along with a theory-specific interpolant for $\langle A', B' \rangle$ in theory T_1 to generate an interpolant for $\langle A, B \rangle$ in T.

Finally, if Eq contains AB-mixed equalities, we construct an equivalent set of AB-pure equalities, as explained in Sec. 4. This is the reason for restricting our method only to equality-interpolating theories.

3.1 Partial Interpolants

In the following definitions we assume that all equalities generated in the Nelson-Oppen framework are AB-pure equalities, as guaranteed by Sec. 4.

A *partial interpolant* is a formula associated with each formula derived by the theories in the proof of unsatisfiability. To simplify the definitions, we associate a (trivial) partial interpolant with each input formula as well. A partial interpolant for $\bot$ is the interpolant for the input formula $\langle A, B \rangle$. The crucial part of our combination method is a way to associate a partial interpolant with each propagated equality. Whenever a decision procedure for a component theory T_i generates an equality e that needs to be propagated, our method provides a partial interpolant for that equality. Our method does not require a special interface for generating partial interpolants, but uses the interpolant-generation procedures of the component theories. A partial interpolant is a boolean combination of the partial interpolants for the inputs and a *theory-specific partial interpolant*. A theory-specific partial interpolant is generated by T_1 using only the input formulas and equalities generated for other theories, without using their partial interpolants.

Definition 3. (projection) *Let Θ be a conjunction of AB-pure literals. Let $\Theta|_A$ be a conjunction of A-local literals of Θ, and $\Theta|_B$ be a conjunction of B-local and AB-common literals of Θ. Note that $\Theta = \Theta|_A \wedge \Theta|_B$.*

Definition 4. (theory-specific partial interpolant) *Let A' and B' be conjunctions of pure literals in Σ_i, and let e be an AB-pure atomic formula generated by the decision procedure for the theory T_i, i.e., $A' \wedge B' \vdash_{T_i} e$ (thus, $A' \wedge B' \wedge \neg e \vdash_{T_i} \bot$).*

An interpolant generated for $\langle A' \wedge \neg(e|_{A'}), B' \wedge \neg(e|_{B'}) \rangle$ by T_i's procedure is a theory-specific partial interpolant for e w.r.t. $\langle A', B' \rangle$, denoted by $\phi^i_{A',B'}(e)$.

Intuitively, we add $\neg e$ to A' if e contains an A-local symbol, otherwise we add it to B', using the assumption that e is AB-pure, i.e., e cannot contain both A-local and B-local symbols. Thus, any theory-specific partial interpolant for e contains only AB-common symbols.

Example 7. The first step of a proof of unsatisfiability in Example 4 uses the theory of linear inequalities to derive the equality $x_1 = y_1$ from $A' \stackrel{\text{def}}{=} A_{LI}$, which contains the literal $(y_1 \leq x_1)$, and $B' \stackrel{\text{def}}{=} B_{LI}$, which contains the literal $(x_1 \leq y_1)$. We use interpolant generation of the theory of linear inequalities to derive an interpolant for $(y_1 \leq x_1)$ and $(x_1 \leq y_1) \wedge \neg(x_1 = y_1)$. Trivially, the interpolant is $y_1 \leq x_1$, which is the theory-specific partial interpolant for $x_1 = y_1$, denoted by $\phi^{LI}_{A',B'}(x_1 = y_1)$.

Let e be an AB-pure equality such that $A \wedge B \vdash e$. We define a *partial interpolant* for e w.r.t. $\langle A, B\rangle$ as follows.

Definition 5. (partial interpolant) *Suppose that e is derived from $A \wedge B$ in the Nelson-Oppen framework by a theory T_i. Suppose that T_i derives e from two conjunctions of pure literals in Σ_i, denoted by A_i and B_i, and a set of AB-pure equalities Eq: $A_i \wedge B_i \wedge Eq \vdash e$.*

A partial interpolant for e w.r.t. $\langle A, B\rangle$ denoted by $\phi_{A,B}(e)$ is defined inductively. The base cases: If $e \in A_i$ then $\phi_{A,B}(e)$ is $\bot$. If $e \in B_i$ then $\phi_{A,B}(e)$ is $\top$. The inductive definition: Let $A' \stackrel{\text{def}}{=} A_i \wedge Eq|_A$ and $B' \stackrel{\text{def}}{=} B_i \wedge Eq|_B$.

$$\phi_{A,B}(e) = (\phi^i_{A',B'}(e) \vee \bigvee_{a \in A'} \phi_{A,B}(a)) \wedge \bigwedge_{b \in B'} \phi_{A,B}(b) \tag{1}$$

Note that this definition includes the special case when e is $\bot$.

Example 8. Table 1 shows the partial interpolants generated by the combination method for the input formulas from Example 4. In the second step of the proof, the decision procedure for the theory of uninterpreted functions generates the equality $a_1 = a_2$ from the input A_{UIF} and B_{UIF} defined in Example 4, and the equality $(x_1 = y_1)$. First, we compute a theory-specific partial interpolant for $a_1 = a_2$, denoted by $\phi^{UIF}_{A',B'}(a_1 = a_2)$, where $A' = A_{UIF}$ and $B' = B_{UIF} \wedge (x_1 = y_1)$, because x_1 and y_1 are AB-common. By Def. 4, we run the interpolant-generation procedure of the theory of uninterpreted functions with the input $A' \wedge \neg(a_1 = a_2)$ and B', and we get $\neg(x_1 = y_1)$, which is $\phi^{UIF}_{A',B'}(a_1 = a_2)$.

We compute a partial interpolant $\phi_{A,B}(a_1 = a_2)$ using $\phi^{UIF}_{A',B'}(a_1 = a_2) = \neg(x_1 = y_1)$ and $\phi_{A,B}(x_1 = y_1) = (y_1 \leq x_1)$ (and the partial interpolant for the input equality $x_1 = y_1$, generated in the previous step). The result $\phi_{A,B}(a_1 = a_2)$ is $(y_1 \leq x_1) \wedge \neg(x_1 = y_1)$.

Theory T	$\mathbf{e}$	$\phi^{\mathbf{T}}_{\mathbf{A'},\mathbf{B'}}(\mathbf{a_1 = a_2})$	$\phi_{\mathbf{A},\mathbf{B}}(\mathbf{e})$
LI	$x_1 = y_1$	$y_1 \leq x_1$	$y_1 \leq x_1$
UIF	$a_1 = a_2$	$\neg(x_1 = y_1)$	$y_1 < x_1$
UIF	$x_2 = y_2$	$\top$	$\top$
LI	$\bot$	$x_2 - y_2 = x_3 - y_3$	$x_2 - y_2 = x_3 - y_3 \vee y_1 < x_1$

Table 1. Partial interpolants generated by the combination method for the input formulas from Example 4. In each step of the process, the decision procedure for the component theory T generates a formula e and the corresponding partial interpolant.

If a theory proves unsatisfiability, the partial interpolant $\phi_{A,B}(\bot)$ is an interpolant for $\langle A, B\rangle$, as shown in the following lemma.

Lemma 1. *The partial interpolant $\phi_{A,B}(e)$ from Def. 5 is an interpolant for $A \wedge \neg(e|_A)$ and $B \wedge \neg(e|_B)$ in the combined theory T. In the special case when e is $\bot$, $\phi_{A,B}(\bot)$ is an interpolant for $\langle A, B\rangle$.*

Example 9. In the last step, $\phi_{A,B}(\bot)$ is $x_2 - y_2 = x_3 - y_3 \vee y_1 < x_1$. It is easy to verify that $\phi_{A,B}(\bot)$ is an interpolant for the input formulas A and B from Example 3.

4 Equality-Interpolating Theories

The combination method in Sec. 3 requires that each component theory only propagates AB-pure equalities. This restriction arose as partial interpolants are not defined for AB-mixed equalities. This section justifies the restriction by showing that for a class of first-order theories, defined as *equality-interpolating* theories, it is sufficient to share AB-pure equalities. Also, this section shows that many interesting theories including the quantifier-free theories of uninterpreted functions, linear arithmetic, and Lisp structures are equality-interpolating.

The basic idea behind equality-interpolating theories is the following. Whenever a decision procedure for a component theory generates an equality $a = b$, where a is an A-local variable and b is a B-local variable the combination method requires that the theory also produce an AB-common term t, such that $a = t$ and $t = b$. Instead of propagating $a = b$, the theory now propagates these two AB-pure equalities separately. For an *equality-interpolating* theory, such an AB-common term t exists for all entailed AB-mixed equalities:

Definition 6. (equality-interpolating theory) *Theory T is equality-interpolating when for all A and B in T, and for all AB-mixed equalities between variables $a = b$ such that $A \wedge B \vdash_T a = b$, there exists a term t such that $A \wedge B \vdash a = t \wedge t = b$ and t contains only AB-common symbols. We say that t is an equality-interpolating term for $a = b$ w.r.t. $\langle A, B\rangle$.*

Example 10. This example shows that not all theories are equality-interpolating. Consider a theory with two relation symbols P and Q, and the axiom $\forall abc\, P(a, c) \wedge Q(c, b) \Rightarrow a = b$. When A contains $P(a, c)$ and B contains $Q(c, b)$, $A \wedge B \vdash a = b$. However, there is no equality-interpolating term for $a = b$.

The proof of correctness of the combination method for equality-interpolating theories follows from Def. 6 and Lem. 1. Whenever a decision procedure for a component theory generates an AB-mixed equality $a = b$, the combination method propagates two equalities $a = v_t$ and $v_t = b$, where v_t is a previously unseen variable representing the equality-interpolating term t. The combination method treats these new variables v_t as AB-common symbols, thus the two equalities propagated instead of $a = b$ are AB-pure and so the correctness of the combination method described in Sec. 3 applies. After the interpolant is generated, occurrences of v_t in it are replaced by the associated term t. By Def. 6, t contains only AB-common symbols, thus the interpolant is an AB-common formula, as required.

The modification mentioned above does not affect the Nelson-Oppen framework in terms of complexity, termination or soundness. All component theories contain equality axioms, thus each theory can infer $a = b$ from the equalities

$a = v_t$ and $v_t = b$. Moreover, as the variable v_t is previously unseen, this is the only inference the theories can make. Note, the number of new variables v_t generated in this process is bounded by the number of AB-mixed equalities used in the proof of unsatisfiability of $A \wedge B$.

In the remainder of this section, we prove that some useful theories are equality-interpolating.

The theory of uninterpreted functions A decision procedure for the theory of uninterpreted functions can be easily modified to generate only AB-pure equalities. The idea is to modify the implementation of the congruence closure algorithm [15] to choose a representative for an equivalence class to be an AB-common term, whenever an equivalence class contains at least one such term. When an equivalence class contains both A-local and B-local terms, it also contains an AB-common term, as follows from:

Lemma 2. *The theory of uninterpreted functions is equality-interpolating.*
Sketch of Proof: We prove a stronger claim: there exists an interpolating term t for all equalities of the form $t_a = t_b$ entailed by $A \wedge B$, where t_a is an A-local term (involves at least one A-local symbols or variable) and t_b is a B-local term.

Note that $t_a = t_b$ is an AB-mixed equality, but it does not contain AB-mixed terms. Every proof of $t_a = t_b$ that uses AB-mixed terms can be transformed into a proof of $t_a = t_b$, in which all derivations involve only AB-pure terms. Assume that a proof of $t_a = t_b$ from $A \wedge B$ uses only AB-pure terms in all derivations. The proof proceeds by induction on the proof tree of $t_a = t_b$ from $A \wedge B$.

The theory of Lisp structures A decision procedure for the theory of Lisp structures based on the congruence closure algorithm is described in [15]. A proof generated by the decision procedure for the theory of Lisp structures can contain proof rules of the theory of uninterpreted functions and an additional rule:

$$\frac{z = cons(x, y)}{x = car(z) \wedge y = cdr(z) \wedge \neg atom(z)} \tag{2}$$

The interpolant-generation procedure for the theory of uninterpreted functions [13] can be adapted to the theory of Lisp structures, as follows. Given a proof P of unsatisfiability of $A \wedge B$ in the theory of Lisp structures, we replace each derivation in P that uses proof rule (2) by the formula it derives, which is treated as an axiom. The result is a proof of unsatisfiability P_{UIF} in the theory of uninterpreted functions, where the symbols `car`, `cdr`, `cons`, and `atom` are treated as uninterpreted function symbols.

Let H be the set of new axioms added to P. Using Lem. 2, we can ensure that all formulas in H are AB-pure. Thus, the interpolant generated for $A \wedge (H|_A)$ and $B \wedge (H|_B)$ by the theory of uninterpreted functions using the proof P_{UIF}, is the interpolant for $\langle A, B \rangle$ in the theory of Lisp structures.

The theory of linear inequalities To show that the theory of linear inequalities is equality-interpolating, we first show that there is an *inequality-interpolating* term for every AB-mixed inequality between variables, entailed by $A \wedge B$.

Definition 7. (inequality-interpolant) *For an AB-mixed inequality $a \leq b$ such that $A \wedge B \vdash a \leq b$, an inequality-interpolant is an AB-common term t such that $A \wedge B \vdash a \leq t \leq b$.*

If $A \wedge B \vdash (a \leq b) \wedge (b \leq a)$ and t_1, t_2 are the inequality-interpolating terms for $a \leq b$ and $b \leq a$ respectively, it follows that $a = b = t_1 = t_2$. Thus, t_1 (or t_2) is an equality-interpolating term for $a = b$.

Lemma 3. *Given conjunctions of linear arithmetic constraints A, B, an inequality-interpolant exists for every AB-mixed inequality $a \leq b$ entailed by $A \wedge B$.*
Proof: Consider an AB-mixed inequality $a \leq b$ derived from $A \wedge B$. Let A and B respectively contain m and n linear constraints. These constraints are of the following form

$$A \equiv \bigwedge_{1 \leq i \leq m} 0 \leq s_i + t_i \quad \text{and} \quad B \equiv \bigwedge_{1 \leq j \leq n} 0 \leq s'_j + t'_j$$

where the terms s_i are A-local, the terms s'_j are B-local, and the terms t_i and t'_j are AB-common. Any linear inequality that can be derived from A and B can be obtained by a linear combination of the constraints in A and B. As $A \wedge B \vdash a \leq b$ (or equivalently $0 \leq -a + b$), there exist non-negative constants $d_1, d_2, \ldots, d_m, d'_1, d'_2, \ldots, d'_n$ such that

$$0 \leq \sum_{i=1}^{m} d_i(s_i + t_i) + \sum_{j=1}^{n} d'_j(s'_j + t'_j) = -a + b \tag{3}$$

Consider the linear combination in Equation 3 restricted to the terms in A. As the terms s'_j and t'_j contain no A-local variables, we have $0 \leq \sum_{i=1}^{m} d_i(s_i + t_i) = -a + t$ for some AB-common term t. Similarly, considering the linear combination restricted to terms in B, we have $0 \leq \sum_{j=1}^{n} d'_j(s'_j + t'_j) = t' + b$ for some AB-common term t'. From Equation 3 it follows that $t = -t'$. Thus, $A \wedge B \vdash (a \leq t) \wedge (t \leq b)$, and t is the inequality-interpolating term for $a \leq b$.

Lemma 4. *Theory of linear arithmetic is equality-interpolating.*
Proof: Directly follows from Lem. 3 and the discussion following Def. 7.

5 Interpolants for Arbitrary Quantifier-Free Formulas

Sec. 3 describes the combination of interpolant-generation procedures for convex theories with disjoint signatures, when the input formulas A and B are conjunctions of literals. This section relaxes these constraints. First, we describe Pudlák's algorithm for generating propositional interpolants [17]. Then, we integrate our method with an extended version of Pudlák's algorithm in the *lazy-proof-explication* framework for checking satisfiability of quantifier free first-order formulas with arbitrary boolean structure.

Pudlák's Algorithm We describe Pudlák's algorithm for generating propositional interpolants and give an alternative correctness argument based on partial interpolants. This algorithm takes as input a proof of unsatisfiability of a propositional formula $A \wedge B$ and generates a propositional interpolant I for $\langle A, B\rangle$. The algorithm generates a partial interpolant $p(c)$ for each clause c derived in the proof, as described below. The partial interpolant generated for the empty clause $p(\bot)$ is an interpolant I for $\langle A, B\rangle$.

Definition 8. *Given two clauses of the form $c_1 \stackrel{\text{def}}{=} x \vee c_1'$ and $c_2 \stackrel{\text{def}}{=} \neg x \vee c_2'$, the resolution of c_1 and c_2 is a clause $c \stackrel{\text{def}}{=} c_1' \vee c_2'$, denoted by $c = resolve_x(c_1, c_2)$, where x is called the pivot variable.*

Let $\langle A, B\rangle$ be a pair of clause sets such that $A \wedge B \vdash \bot$. Let $\mathcal{T}$ be a proof of unsatisfiability of $A \wedge B$. The propositional formula $p(c)$ for a clause c in $\mathcal{T}$ is defined by induction on the proof structure:

(i) if c is one of the input clauses then
  (a) if $c \in A$, then $p(c) := \bot$;
  (b) if $c \in B$, then $p(c) := \top$.
(ii) otherwise, c is a result of resolution, i.e., $c = resolve_x(c_1, c_2)$
  (a) if $x \in A$ and $x \notin B$ (x is A-local), then $p(c) := p(c_1) \vee p(c_2)$
  (b) if $x \notin A$ and $x \in B$ (x is B-local), then $p(c) := p(c_1) \wedge p(c_2)$
  (c) otherwise (x is AB-common), $p(c) := (x \vee p(c_1)) \wedge (\neg x \vee p(c_2))$.

The correctness of the algorithm is guaranteed by the following invariant: for each clause $c \in \mathcal{T}$, the partial interpolant $p(c)$ is an interpolant for $\langle g_A(c), g_B(c)\rangle$ where $g_A(c) \stackrel{\text{def}}{=} A \wedge ((\neg c)|_A)$ and $g_B(c) \stackrel{\text{def}}{=} B \wedge ((\neg c)|_B)$ When c is an empty clause $\bot$, we get that $\langle g_A(\bot), g_B(\bot)\rangle$ is $\langle A, B\rangle$, and the formula $p(\bot)$ is the result.

Lazy Proof-Explication Framework In order to leverage advances in SAT solving, state-of-the-art decision procedures [1, 4, 5] based on the Nelson-Oppen framework use a SAT solver to perform propositional reasoning. Given an input formula, the SAT solver treats all atomic formulas occurring in the input formula as free boolean variables. Suppose that the SAT solver finds an assignment to the boolean variables that satisfies the input formula propositionally. This assignment is a conjunction of (first-order) literals. It is passed to a Nelson-Oppen decision procedure.

The decision procedure attempts to derive a contradiction from this conjunction of literals. If it cannot derive a contradiction, the input formula is declared as satisfiable. If a contradiction is detected, the negation of the current assignment is added to the SAT solver as a new conflict clause. Because this new clause is in conflict with the current assignment, the SAT solver backtracks, searching for a new assignment. If it cannot find another assignment, it has proved that the propositional abstraction is unsatisfiable. Thus, the input formula is unsatisfiable.

Integration with an Extended Pudlák's Algorithm We assume that the unsatisfiability of A and B in theory T is proved by a lazy proof-explication framework. That is, a SAT solver proved propositional unsatisfiability of A and B using a set of conflict clauses C. For each conflict clause c in C, $\neg c$ is a conjunction of (first-order) literals. By construction, we have a proof of unsatisfiability of $\neg c$. Also, it is guaranteed that $\neg c$ contains only AB-pure literals, as it contains only the original literals from A or B (each of which is AB-pure by definition). Therefore, we can use the method described in Sec. 3 to generate an interpolant between the A-part of $\neg c$ and the B-part of $\neg c$, which is also called a partial interpolant for the conflict clause c.

Definition 9. (partial interpolants for clauses) *Let $A \wedge B \vdash_T \bot$ be proved by a decision procedure for T using a corresponding set of conflict clauses C, such that $A \wedge B \wedge C$ is propositionally unsatisfiable. A partial interpolant for a clause c is denoted by $\phi_{A,B}(c)$. If $c \in A$, then $\phi_{A,B}(c) = \bot$, if $c \in B$, then $\phi_{A,B}(c) = \top$, otherwise, for a conflict clause $c \in C$, a partial interpolant $\phi_{A,B}(c)$ is the interpolant for $\langle (\neg c)|_A, (\neg c)|_B \rangle$ in theory T, where the projection operation is given in Def. 3.*

We use partial interpolants $\phi_{A,B}(c)$ defined above as initial values for $p(c)$ in the extended version of Pudlák's algorithm, instead of using the standard initialization of Pudlák's algorithm from Def. 8(i). (To see why this change is necessary, recall that a conflict clause may involve both A-local and B-local literals.) Partial interpolants in Def. 9 satisfy the invariant of Pudlak's algorithm.

There is no change in phase (ii) of Def. 8. The input for the extended algorithm consists of three clause sets, denoted by $\langle A, B; C \rangle$, all three of them are necessary for an unsatisfiability proof. However, in each resolution step, the pivot is guarantee to be in A or B, because all literals in conflict clauses C appear in the original formulas A and B. Note that the result is a first-order interpolant, which is a combination of the original clauses and the interpolants generated for the conflict clauses. The correctness of the interpolant generated by the extended Pudlák's algorithm follows from the correctness of theory-specific interpolants for conflict clauses.

Lemma 5. *The interpolant for $\langle A, B; C \rangle$ generated by the extended Pudlak's algorithm using partial interpolants for clauses as in Def. 9 is an interpolant for the input $\langle A, B \rangle$ in theory T.*

6 Related Work

Interpolants are of great theoretical and practical significance. Our interest in interpolants is particularly motivated by their use in program analysis and model checking. [12] uses interpolants to achieve faster termination while model checking finite state systems, and [13] explores the possibility of using interpolants for model checking infinite systems.

Craig interpolation theorem for first-order logic [2] shows the existence of a first-order interpolant for any pair of formulas in first-order logic. While constructive proofs of Craig interpolation theorem exist [6, 18, 9], these proofs are (to the best of our knowledge) based on cut elimination and result in very expensive interplation-generation procedures. (See [8] for references on the complexity of cut elimination.)

On the other hand, interpolants can be generated efficiently for formulas in a restricted subclass of first-order logic. When the input formulas A and B are propositional, or when they are both conjunctions of linear constraints, Pudlák [17] provides interpolant-generation procedures that are linear in the proof of unsatisfiability of $A \wedge B$. McMillan [13] extends these procedures to compute interpolants for quantifier-free formulas in the combined theory of uninterpreted functions and linear arithmetic.

Finally, the Nelson-Oppen framework is being constantly improved. For example, a recent work by [7] defines sufficient conditions for extending the Nelson-Oppen framework to theories with non-disjoint signatures, e.g., the theory of bit-vectors, presburger arithmetic, a theory of Lists with length operator, or theories of many-sorted logics. In the extended framework, the theories can exchange atomic formulas over the intersection of their signatures, and not only equalities between variables. Our method, by being modular, is well-suited to support such advances in the Nelson-Oppen framework. Given the unsatisfiability proof with only AB-pure equalities, our method can generate interpolants for non-disjoint theories, because partial interpolants in Def. 4 and Def. 5 do not assume that the theories exchange only equalities.

7 Conclusions and Future Work

The combination method for equality-interpolating theories presented in this paper proves existence of quantifier-free interpolants for a combined theory, if all the component theories have quantifier-free interpolants. If some of the theories has quantified interpolants, our method produces correct, quantified interpolant for the combined theory. Currently, our method applies only to quantifier-free input formulas. We believe the method can be extended to handle quantified formulas, because the proof of unsatisfiability contains a finite instantiation of the quantified variables.

Our method shows how to integrate interpolant generation for various theories within the existing satisfiability-checking tools, adding only a small overhead. This provides a practical way to use interpolants for speeding up termination of software model checking and real-time model checking.

Finally, the combination of interpolant-generation procedures demonstrates that *equality propagation* in the Nelson-Oppen framework can be used to combine operations other than satisfiability checking. Recently, [10] have used a similar approach to combine abstract domains. We believe that similar combination methods for operations would enhance program analysis tools while retaining the flexibility of the Nelson-Oppen framework to extend with additional theories.

References

1. C. Barrett and S. Berezin. CVC Lite: A new implementation of the cooperating validity checker. In *CAV*, pages 515–518, 2004.
2. W. Craig. Linear reasoning. a new form of the herbrand-gentzen theorem. *J. Symbolic Logic*, 22:250–268, 1957.
3. D. Detlefs, G. Nelson, and J. Saxe. Simplify theorem prover. http://research.compaq.com/SRC/esc/Simplify.html.
4. J.-C. Filliâtre, S. Owre, H. Rueß, and N. Shankar. ICS: integrated canonizer and solver. In *CAV*, pages 246–249, 2001.
5. C. Flanagan, R. Joshi, X. Ou, and J. B. Saxe. Theorem proving using lazy proof explication. In *CAV*, pages 355–367, 2003.
6. J. H. Gallier. *Logic for Computer Science: Foundations of Automatic Theorem Proving*. John Wiley & Sons, New York, 1987.
7. V. Ganesh, S. Berezin, C. Tinelli, and D. L. Dill. Combination results for many-sorted theories with overlapping signatures. Technical report, Department of Computer Science, Stanford University, 2004.
8. P. Gerhardy. Refined Complexity Analysis of Cut Elimination. In *CSL*, pages 212–225, 2003.
9. G.Takeuti. *Studies in Logic*, volume 81. Elsevier, Amsterdam, North Holland, 1975.
10. Sumit Gulwani and Ashish Tiwari. Unpublished manuscript.
11. T. A. Henzinger, R. Jhala, R. Majumdar, and K. L. McMillan. Abstractions from proofs. In *POPL*, pages 232–244, 2004.
12. K.L. McMillan. Interpolation and sat-based model checking. In *CAV*, pages 1–13, 2003.
13. K.L. McMillan. An interpolating theorem prover. In *TACAS*, pages 16–30, 2004.
14. G. Nelson and D. C. Oppen. Simplification by cooperating decision procedures. *ACM Transactions on Programming Languages and Systems*, 1(2):245–257, October 1979.
15. G. Nelson and D. C. Oppen. Fast decision procedures based on congruence closure. *J. ACM*, 27(2):356–364, 1980.
16. Derek C. Oppen. Complexity, convexity and combinations of theories. In *Theoretical Computer Science*, volume 12, pages 291–302, 1980.
17. P. Pudlák. Lower bounds for resolution and cutting planes proofs and monotone computations. *J. of Symbolic Logic*, 62(3):981–998, 1995.
18. S.C.Kleene. *Mathematical Logic*. Wiley Interscience, New York, 1967.
19. G. Yorsh and M. Musuvathi. A combination method for generating interpolants. Technical Report MSR-TR-2004-108, Microsoft Research, October 2004.