A Parametric Interpolation Framework for First-Order Theories

Laura Kovács[1], Simone Fulvio Rollini[2], and Natasha Sharygina[2]

[1] Chalmers University of Technology
[2] USI

Abstract. Craig interpolation is successfully used in both hardware and software model checking. Generating good interpolants, and hence automatically determining the quality of interpolants is however a very hard problem, requiring non-trivial reasoning in first-order theories. An important class of state-of-the-art interpolation algorithms is based on recursive procedures that generate interpolants from refutations of unsatisfiable conjunctions of formulas. We analyze this type of algorithms and develop a theoretical framework, called a parametric interpolation framework, for arbitrary first-order theories and inference systems. As interpolation-based verification approaches depend on the quality of interpolants, our method can be used to derive interpolants of different structure and strength, with or without quantifiers, from the same proof. We show that some well-known interpolation algorithms are instantiations of our framework.

1 Introduction

Craig interpolation [3] provides powerful heuristics for verifying software and hardware. In particular, interpolants extracted from proofs of various properties are used in invariant generation and bounded model checking, see e.g. [5, 9, 14].

There exist various methods to compute interpolants from proofs. The work of [12] introduces an interpolation algorithm for propositional logic, and is generalized in [13] to generate interpolants in the combined theory of uninterpreted functions and linear arithmetic. The approaches of [7, 11, 15] propose another interpolation algorithm for propositional logic which is later extended in [17] to address a class of first-order theories. More recently, [4] introduces a framework that generalizes [12, 15], by analyzing the logical strength of interpolants. The work of [4] has been extended in [16] to interpolation in the hyper-resolution system. The methods described in [6, 10] give a general interpolation algorithm that can be used with arbitrary first-order calculi and inference systems. This algorithm is, however, restricted to local proofs [10] or split proofs [9].

While interpolation-based verification techniques crucially depend onto which extent "good" interpolants can be generated, there is no general criterion for defining the notion of "good". Finding a good interpolant is a hard problem, whose solution justifies spurious program paths or helps proving correctness of

program properties. Given a computer program to be verified, a natural question to ask is therefore "what makes some interpolants better than others?", or "what are the essential program properties for which a good interpolant can be derived"? If the program is small or manipulates only a restricted class of data structures, one can expect a programmer to identify possible program errors and hence characterize the set of interpolants justifying these errors. For example, if a program implements linear arithmetic operations over integers, a good interpolant might be a conjunction of linear integer inequalities over program properties [2]. The works of [4, 8] remark that interpolants of different strength can be beneficial in different verification frameworks. When dealing with properties over quantifier-free linear arithmetic and uninterpreted functions, [8] emphasizes the need for logically strong interpolants in model checking and predicate abstraction; however, when restricting interpolation to propositional logic, [4] suggests instead that logically weaker interpolants are more useful in verification.

While [2, 4, 8] answer our question of what a good interpolant can be, these methods are restricted to specific logic fragments and the qualitative measure of their interpolants cannot be easily extended to more general logics. For example, when a program implements also array operations, like array initializations, a good interpolant should summarize that all array elements have been initialized. This interpolant is a first-order property and can be expressed in first-order logic with quantifiers and array symbols. Computing and describing the quality of first-order interpolants is however a non-trivial problem. One could come up with various syntactic measures of quality, e.g. a small amount of quantifiers, as discussed in [6]; nevertheless, such a syntactic approach is limited, since it cannot be used to characterize semantic features like logical strength.

In this paper we introduce a new theoretical framework, called *parametric interpolation framework*, for arbitrary theories and inference systems. We show that the aforementioned interpolation procedures can be considered elements of a class of algorithms characterized by specific structural properties. Our method supports the generation of multiple interpolants of different strength and structure. For example, our approach can generate quantifier-free interpolants on examples where current methods are only able to compute quantified interpolants. Our approach also provides flexibility in adjusting the logical expressiveness of the computed interpolants, and hence can yield interpolants, even quantifier-free ones, that are stronger/weaker than the ones generated by current methods. We therefore believe that our contribution helps to answer the fundamental AI problem delineated above: by comparing structure and strength of interpolants in first-order theories we propose a theoretical framework to explore "good" interpolants. Investigating the impact of our approach on concrete verification problems is a very interesting task which needs extensive experimentation, and we leave it as future work.

Contributions. The main contribution of this paper comes with the *theoretical formalization of a new parametric interpolation framework* (§4). We show that this framework generalizes existing interpolation algorithms for first-order theories and, as a consequence, also for propositional logic (§5). We illustrate the

kind of interpolants we produce (§3) and show how the interpolation algorithms of [4, 10, 16] can be regarded as special cases of our method in the context of first-order and hyper-resolution inference systems.

When compared to [10], the differences and benefits of our approach can be summarized as follows. We derive an algorithm for arbitrary first-order theories and inference systems, which extracts interpolants as boolean combinations of formulas from a refutation proof. Our algorithm can be applied to a class of proofs strictly larger than the class of local proofs in [10]; it can also produce a family of interpolants which contains the interpolants of [10]. Within this family, we relate and compare the interpolants by their logical strength. The results of [10] about the existence of local proofs in the superposition calculus and turning non-local proofs into local ones in the style of [6] can be naturally extended to our framework. Remarkably, our method allows to compute *quantifier-free interpolants* for problems on which [10] can only derive quantified interpolants.

Referring to [4, 16], our approach is different in the following aspects. We integrate the hyper-resolution system into our first-order interpolation algorithm, and discuss the applicability of the family of interpolants proposed there. We then extend the class of proofs from first-order theories to arbitrary hyper-resolution refutations, and show how the structure of the formulas and inference rules allows to obtain additional interpolants, containing those generated by [16]. Finally, we also compare the produced interpolants by their logical strength.

2 Preliminaries

This section fixes our notation and recalls some required terminology by adapting the material of [10] to our setting.

We consider the language of standard first-order logic with equality. We assume that the language contains boolean connectives and quantifiers, as well as the logical constants $\top$ and $\bot$ respectively denoting the *always true* and *always false* formulas. For a formula A we write $\overline{A}$ to mean $\neg A$, that is the negation of A. We write $A_1, \ldots, A_n \vdash A$ to denote that $A_1 \wedge \cdots \wedge A_n \rightarrow A$ is valid.

We call a *symbol* a predicate symbol, a function symbol or a constant. Individual (logical) variables are thus not symbols. We use capital letters A, B, C, D, I, R, possibly with indices, to denote formulas. Terms are denoted by s, t, variables by x, y, z, constants by a, b, c, and functions by f, g, all possibly with indices. A *signature* Σ is a finite set of symbols. The signature of a formula A, denoted by Σ_A, is the set of all symbols occurring in A. For example, the signature of $g(a, x)$ is $\{g, a\}$. The language of a formula A, denoted by $\mathcal{L}_A$, is the set of all formulas built from Σ_A.

Consider a formula A whose free variables are $x_1, \ldots, x_m$. Then $\forall A$ denotes the formula $(\forall x_1, \ldots, x_m)A$; similarly, $\exists A$ is the formula $(\exists x_1, \ldots, x_m)A$.

Inference Systems and Derivations. An *inference rule*, or simply *inference*, is an $n+1$-ary relation on formulas, where $n \geq 0$. It is usually written as: $\frac{A_1 \quad \cdots \quad A_n}{A}$ where $A_1, \ldots, A_n$ are the *premises* and A the *conclusion*. An *inference system* is a set of inference rules. An *axiom* is the conclusion of an inference

with 0 premises. An inference with 0 premises and conclusion A will be written without the bar line as A. A *derivation*, or a *proof*, of a formula A is a finite tree built from inferences in the inference system, such that the root of the tree is A and all leaves are axioms; nodes correspond to formulas. A node A with parents $A_1, \ldots, A_n$ represents the conclusion A of an inference with premises $A_1, \ldots, A_n$. A derivation of A is *from assumptions* $A_1, \ldots, A_n$ if every leaf is either an axiom or one of the formulas $A_1, \ldots, A_n$. A *refutation* is a derivation of $\bot$. A sub-tree of a derivation is called a *sub-derivation*.

Colored Symbols and Formulas. Let us now fix two sentences R and B and give all definitions relative to them. We define $\Sigma_{RB} = \Sigma_R \cap \Sigma_B$ as the set of symbols occurring both in R and B and take $\mathcal{L}_{RB} = \mathcal{L}_R \cap \mathcal{L}_B$. The signature symbols from Σ_{RB} are called *grey* symbols. Signature symbols occurring only in $\Sigma_R \setminus \Sigma_{RB}$ will be called *red*, and symbols occurring only in $\Sigma_B \setminus \Sigma_{RB}$ are *blue*. A symbol that is not *grey* is also called *colored*. A formula A is called *grey* if it contains only grey symbols. Grey formulas are thus in $\mathcal{L}_{RB}$. A formula A that is not grey is called *colored*. A formula A is called *red* if it contains only red and grey symbols, but at least one red symbol. Similarly, A is said to be *blue* if it only contains blue and grey symbols, but at least one blue symbol. Red formulas will be denoted by R and blue formulas by B, possibly with indices.

An *RB-derivation* is any derivation Π satisfying the following conditions:

(RB1) for every leaf C, we have: $\vdash \forall C$ and $C \in \mathcal{L}_R$ or $B \vdash \forall C$ and $C \in \mathcal{L}_B$;

(RB2) for every inference $\frac{C_1 \quad \cdots \quad C_n}{C}$ of Π, we have: $\forall C_1, \ldots, \forall C_n \vdash \forall C$.

We call *RB-refutation* an *RB-derivation* of $\bot$.

Craig Interpolation. Given two formulas R and B such that their conjunction is unsatisfiable, that is $R \wedge B \vdash \bot$, an *(Craig) interpolant* of R and B is any grey formula I such that $A \vdash I$ and $B \wedge I \vdash \bot$. Hence, $I \in \mathcal{L}_{RB}$. Note that we are interested in interpolants I of red R and blue B formulas. For simplicity, in this paper we assume that neither R or B are trivially unsatisfiable, that is $R \nvdash \bot$ and $B \nvdash \bot$. As proved in [10], Craig interpolation can also be defined modulo theories. Symbols occurring in a theory are called *interpreted*, while all other symbols are *uninterpreted*.

3 Example

We start with an example showing what kind of interpolants we can compute.

Example 1. Let us take $\forall z(z = c) \wedge a = c \wedge g(b) = g(h)$ as R, and $f(a) \neq f(h) \wedge h = b$ as B. Then, c, g are red symbols, a, b, h are grey, and f is blue. Clearly, $R \wedge B$ is unsatisfiable. A refutation Π of $R \wedge B$ is given in Fig. 1. A possible interpolant of R and B is the quantified formula $\forall z(z = a)$, which would be computed, for example, by the interpolation algorithm of [10].

However, when applying our method on Fig. 1, besides $\forall z(z = a)$ we are able to compute $a = b$ and $h \neq b \vee (a = b \wedge h = b)$ as interpolants of R and B.

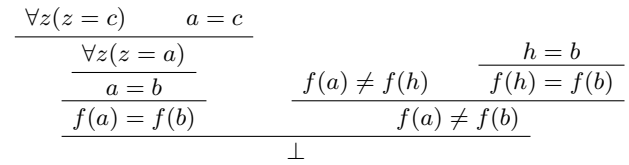

Fig. 1. Local refutation Π of $R \wedge B$.

Note that these two additional interpolants are quantifier-free, and of different strength. Our method thus offers the possibility of computing *quantifier-free interpolants* for problems on which [10] could only derive quantified interpolants. When applying our method to quantifier-free inference systems, for example to the propositional hyper-resolution system, our approach also generates a range of quantifier-free interpolants, including those coming from [16]. The main advantage of our approach hence comes with the flexibility of *choosing between more than one "'good" interpolant* and *generating interpolants of different boolean structure and strength, with or without quantifiers, from the same proof.*

4 A Parametric Interpolation Framework

We now present a new interpolation framework that describes a class of recursive interpolation procedures computing so-called *partial interpolants* from refutation proofs. These procedures start by deriving partial interpolants for the leaves; then, they derive partial interpolants for (some of) the inner nodes, by relying on the previously computed partial interpolants. In what follows, we first define the notion of partial interpolants. Then our *parametric interpolation algorithm* is given (Alg. 1), and the soundness of our approach is discussed. The algorithm will be later instantiated into a specific interpolation algorithm in first-order, as well as propositional systems. (§5).

Let Π be an RB-refutation, corresponding to the unsatisfiability proof of $R \wedge B$. Following [10], we generate an interpolant I of R and B such that I is a boolean combination of formulas of Π. Recall that $R \vdash I$, $B \vdash \overline{I}$ and $I \in \mathcal{L}_{RB}$. Our framework is parametric in a chosen *partition* of Π, i.e. a set of derivations $\mathcal{P} = \{\Pi'_i\}$ such that (i) each Π'_i is a sub-derivation of Π, (ii) a leaf of a sub-derivation Π'_i represents the root of another sub-derivation Π'_j or a leaf of Π, (iii) each inference of Π belongs to some $\Pi'_i \in \mathcal{P}$. We call the leaves of a sub-derivation $\Pi'_i \in \mathcal{P}$ *sub-leaves* of Π'_i; note that a sub-leaf might also be a leaf of Π. Similarly, the root of a sub-derivation Π'_i is called a *sub-root* of Π'_i. The aim of our algorithm is to build an interpolant from Π, by using the partition $\mathcal{P}$ of Π. To this end, we first define the notion of a *partial interpolant* of a formula C.

Definition 1. *[Partial Interpolant] Let C be a formula, and let f and g denote functions over formulas such that $f(\bot) = g(\bot) = \bot$. A formula I_C is called a* partial interpolant of C *with respect to R and B if it satisfies:*

$$R \vdash I_C \vee f(C), \quad B \vdash \overline{I_C} \vee g(C), \quad I_C \in \mathcal{L}_{RB}. \tag{1}$$

When C is $\bot$, a partial interpolant I_C is an interpolant of R and B, since we have $R \vdash I_C$ and $B \vdash \overline{I_C}$. Note also that Def. 1 generalizes the notion of C-interpolants from [10]. Namely, by taking $f(C) = C$ and $g(C) = C$ a partial interpolant I_C is just a C-interpolant in the sense of [10], when C is grey.

Let us emphasize that in Def. 1 we are not restricted to a particular choice of f and g, which can be arbitrary functions over formulas. For example, the value of $f(C)$ and $g(C)$ might not even depend on C, or f and g can be defined using $\mathcal{P}$; the only restriction we impose is that eq. (1) holds. Such a generality allows us to build various (partial) interpolants, as presented later in §5.

Using partial interpolants, our framework is summarized as follows. Given a partition $\mathcal{P}$ of Π, we first compute partial interpolants of the leaves of Π. Next, for each sub-derivation $\Pi'_i \in \mathcal{P}$ with root C and leaves $C_1, \ldots, C_n$, we build a partial interpolant I_C of C, proceeding inductively. We use the sub-leaves $C_1, \ldots, C_n$, and respectively compute their partial interpolants $I_{C_1}, \ldots, I_{C_n}$. I_C is then obtained as a boolean combination of (some of) C, $C_1, \ldots, C_n$, and $I_{C_1}, \ldots, I_{C_n}$ As a consequence, a partial interpolant of the root $\bot$ of Π is an interpolant I of R and B.

When computing partial interpolants of a formula C, we make a case distinction whether C is a leaf (base case) or a sub-root of Π (induction step). We now address each case separately and formulate requirements over a formula to be a partial interpolant of C (see eq. (2) and (5)).

Partial Interpolants of Leaves. Let C be a leaf of Π. Then, by the property (RB1) of RB-derivations, we need to distinguish between $R \vdash C$ and $B \vdash C$. The following *conditions over a partial interpolant I_C of C* are therefore imposed in order to satisfy (1):

$$\begin{array}{lll} R \vdash C \wedge \overline{f(C)} \rightarrow I_C, & B \vdash I_C \rightarrow g(C), \quad I_C \in \mathcal{L}_{RB}, & \text{if } R \vdash C; \\ R \vdash \overline{f(C)} \rightarrow I_C, & B \vdash I_C \rightarrow \overline{C} \vee g(C), \quad I_C \in \mathcal{L}_{RB}, & \text{if } B \vdash C. \end{array} \quad (2)$$

Partial Interpolants of Sub-Roots. Let C be the root of a sub-derivation Π' of Π. We assume that Π' consists of more than one formula (otherwise, we are in Case 1) and that the leaves of Π' are $C_1, \ldots, C_n$. By the property (RB2), we conclude $\bigwedge C_i \vdash C$. By the induction hypothesis over $C_1, \ldots, C_n$, we assume that the partial interpolants $I_{C_1}, \ldots, I_{C_n}$ of the sub-leaves C_i are already computed. Using eq. (1), we have:

$$R \vdash I_{C_i} \vee f(C_i), \quad B \vdash \overline{I_{C_i}} \vee g(C_i), \quad I_{C_i} \in \mathcal{L}_{RB}. \quad (3)$$

From a simple combination of $\bigwedge C_i \vdash C$ and eq. (3), we have:

$$R \vdash \bigwedge (I_{C_i} \vee f(C_i)) \wedge (\bigvee \overline{C_i} \vee C), \;\; B \vdash \bigwedge (\overline{I_{C_i}} \vee g(C_i)) \wedge (\bigvee \overline{C_i} \vee C). \quad (4)$$

Using (1) in conjunction with (4), we derive the *following constraints over a partial interpolant I_C of C*:

$$R \vdash \bigwedge(I_{C_i} \vee f(C_i)) \wedge (\bigvee \overline{C_i} \vee C) \wedge \overline{f(C)} \rightarrow I_C, \qquad I_C \in \mathcal{L}_{RB},$$
$$B \vdash I_C \rightarrow \bigvee(I_{C_i} \wedge \overline{g(C_i)}) \vee (\bigwedge C_i \wedge \overline{C}) \vee g(C). \tag{5}$$

Parametric Interpolation Algorithm. Our interpolation algorithm is given in Alg. 1. It takes as input an RB-derivation Π, a partition $\mathcal{P}$ of Π, and the functions f and g. In addition, Alg. 1 depends on a *construct* function which builds partial interpolants of leaves and sub-roots of Π, by using f and g. That is, for a formula C, *construct* returns a *set Φ of partial interpolants* I_C by making a case distinction whether C is a leaf or a sub-root of Π. Hence, setting $f_C = f(C), g_C = g(C), f_i = f(C_i), g_i = g(C_i), I_i = I(C_i)$, *construct* is defined as:

$$construct(C, C_i, I_i, f_C, g_C, f_i, g_i) = \begin{cases} \Phi_1, & \text{if } C \text{ is a leaf} \\ \Phi_2, & \text{if } C \text{ is a sub-root} \end{cases} \tag{6}$$

where each $I_C \in \Phi_1$ satisfies (2) and each $I_C \in \Phi_2$ satisfies (5). Note that the arguments $C_i, I_{C_i}, f(C_i), g(C_i)$ of *construct* become trivially empty whenever C is a leaf. For simplicity of notation, *we therefore write construct$(C, f(C), g(C))$ whenever C is a leaf.* The behavior of *construct*, in particular the choice of Φ_1 and Φ_2, is specific to the inference system in which Π was produced. We will address one choice of Φ_1 and Φ_2 in §5.

Assuming *construct*, f, g are fixed, Alg. 1 returns an interpolant I of R and B. First, the leaves of Π are identified (line 2). For each leaf C of Π, a set Φ_1 of partial interpolants satisfying (2) is constructed. Then, the partial interpolant of C is selected from Φ_1 (line 5). Next, partial interpolants of the sub-roots C of Π are recursively computed (lines 9-18). To this end, each sub-derivation $\Pi' \in \mathcal{P}$ with root C and leaves $C_1, \ldots, C_n$ is analyzed. A set Φ_2 of partial interpolants of C is built by using the partial interpolants of $C_1, \ldots, C_n$ (line 13). The partial interpolant of C is selected from Φ_2 (line 14). Finally, the partial interpolant of $\bot$ is returned as the interpolant of R and B (line 19).

Algorithm 1 ***Parametric Interpolation Algorithm***
Input: *Formulas R and B such that $R \wedge B \rightarrow \bot$, an RB-refutation Π of $R \wedge B$, a partition $\mathcal{P}$ of Π, and functions $f, g,$ construct.*
Output: *Interpolant I of R and B*
Assumption: *f and g satisfy* (1)*, construct produces grey formulas*

1 **begin**
Compute Partial Interpolants of Leaves
2 $L :=$ *leaves*(Π);
3 **for** *each formula C in L* **do**
4 $\quad \Phi_1 :=$ construct$(C, f(C), g(C))$;
5 $\quad I_C :=$ *select*(Φ_1);
6 **endfor** ;
Compute Partial Interpolants of Sub-Roots
7 $\mathcal{I} := \bigcup_{C \in L} I_C$, *where* $\mathcal{I}[C] := I_C$;
8 $\mathcal{P}_* = \{\}$;

```
repeat
  for each Π′ in P such that leaves(Π′) ⊆ L do
    C := root(Π′);
    for each C_i in leaves(Π′) do I_{C_i} := I[C_i] endfor ;
    Φ_2 := construct(C, C_i, I_{C_i}, f(C), g(C), f(C_i), g(C_i));
    I_C := select(Φ_2);
    I := I ∪ {I_C};  L := L ∪ {C};
  endfor ;
  P_* := P_* ∪ {Π′};
until P_* = P;
Compute Interpolant
return I[⊥]
```

Alg. 1 depends on the choice of f, g, and *construct*, as well as of the partition $\mathcal{P}$; *select* denotes a function that picks and returns a formula from a set of formulas. A *parametric interpolation framework* is thus implicitly defined by $f, g, construct$, and $\mathcal{P}$, yielding different interpolation algorithms based on Alg. 1.

In the sequel we present a concrete choice of f, g and *construct*, together with $\mathcal{P}$, yielding an interpolation procedure for arbitrary first-order inference systems. It is then not hard to argue that our method also yields an interpolation procedure in the propositional hyper-resolution system. Further, we show that Alg. 1 generalizes the interpolation algorithms of [10, 16].

5 Interpolation in First-Order Systems

We present an interpolation procedure for arbitrary first-order inference systems, by fixing the definition of $f, g, construct$ and $\mathcal{P}$ in Alg. 1 as follows.

Definition of f and g. We take f and g such that $f(C) = g(C) = C$, for every formula C. Clearly, the condition $f(\bot) = g(\bot) = \bot$ from Def. 1 is satisfied.

Definition of $\mathcal{P}$. We are interested in a special kind of partition, which we call *RB-partition* and define below.

Definition 2. *[RB-partition] Let Π be an RB-derivation and consider a partition $\mathcal{P} = \{\Pi'_j\}$ of Π into a set of sub-derivations Π'_j. The partition $\mathcal{P}$ of Π is called an RB-*partition *if the following conditions hold:*

- *the sub-root C of each Π'_j is grey;*
- *the sub-leaves C_i of each Π'_j satisfy one of the following conditions: (a) every C_i is grey, or (b) if some of the C_i are colored, then the colored sub-leaves C_j are also leaves of Π and C_j are either all red or all blue. Hence, a colored sub-leaf C_j cannot contain both red and blue symbols.*

In this section we fix $\mathcal{P}$ *to be an RB-partition.* We are now left with defining the input function *construct* of Alg. 1. We make a case distinction on the sub-roots of the proof, and define the sets Φ_1 and Φ_2 of partial interpolants as follows.

Definition of *construct* for Partial Interpolants of Leaves. Let C be a leaf of Π. Since $f(C) = g(C) = C$, eq. (2) yields the following constraints over I_C:

$$R \vdash \bot \to I_C, \qquad B \vdash I_C \to C, \qquad I_C \in \mathcal{L}_{RB}, \qquad \text{if } R \vdash C;$$
$$R \vdash \overline{C} \to I_C, \qquad B \vdash I_C \to \top, \qquad I_B \in \mathcal{L}_{RB}, \qquad \text{if } B \vdash C.$$

In principle, any formula $I_C \in \mathcal{L}_{RB}$ such that $\overline{C} \to \overline{I_C}$ if $R \vdash C$, and $\overline{C} \to I_C$ if $B \vdash C$ can be chosen as partial interpolant. Depending on whether C is grey or not, we define the set Φ_1 of partial interpolants as follows:

- If C is grey, we take: $\Phi_1 = \{C, \bot\}$, if $R \vdash C$; $\{\overline{C}, \top\}$, if $B \vdash C$.
- If C is colored, we take: $\Phi_1 = \{\bot\}$, if $R \vdash C$; $\{\top\}$, if $B \vdash C$.

Definition of *construct* for Partial Interpolants of Sub-Roots. Let C be the root of a sub-derivation $\Pi' \in \mathcal{P}$, and let $C_1, \ldots, C_n$ denote the sub-leaves of Π'. As $f(C) = g(C) = C$ and $f(C_i) = g(C_i) = C_i$, eq. (5) yields the following constraints over $I_C \in \mathcal{L}_{RB}$:

$$R \vdash \bigwedge (I_{C_i} \vee C_i) \wedge (\bigvee \overline{C_i} \vee C) \wedge \overline{C} \to I_C,$$
$$B \vdash I_C \to \bigvee (I_{C_i} \wedge \overline{C_i}) \vee (\bigwedge C_i \wedge \overline{C}) \vee C. \tag{7}$$

Any formula $I_C \in \Phi_2$ needs to satisfy eq. (7). A potential set Φ_2 of partial interpolants consists of the following ten formulas (annotated from (a) to (j)):

$$\begin{array}{ll}
\text{(a) } \bigwedge (I_{C_i} \vee C_i) \wedge (\bigvee \overline{C_i} \vee C) \wedge \overline{C} & \text{(f) } \bigvee (I_{C_i} \wedge \overline{C_i}) \\
\text{(b) } \bigwedge (I_{C_i} \vee C_i) \wedge (\bigvee \overline{C_i}) & \text{(g) } \bigvee (I_{C_i} \wedge \overline{C_i}) \vee C \\
\text{(c) } \bigwedge (I_{C_i} \vee C_i) \wedge (\bigvee \overline{C_i} \vee C) & \text{(h) } \bigvee (I_{C_i} \wedge \overline{C_i}) \vee (\bigwedge C_i \wedge \overline{C}) \\
\text{(d) } \bigwedge (I_{C_i} \vee C_i) \wedge \overline{C} & \text{(i) } \bigvee (I_{C_i} \wedge \overline{C_i}) \vee (\bigwedge C_i) \\
\text{(e) } \bigwedge (I_{C_i} \vee C_i) & \text{(j) } \bigvee (I_{C_i} \wedge \overline{C_i}) \vee (\bigwedge C_i \wedge \overline{C}) \vee C
\end{array} \tag{8}$$

It is not hard to argue that every formula from (8) satisfies eq. (7). However, not any formula from (8) could be used as a partial interpolant I_C, as partial interpolants need to be grey. Note however that $\mathcal{P}$ is an RB-partition; this means that the root of Π' is grey, yielding that $f(C) = g(C) = C$ are grey formulas. Hence, whether a formula from (8) is grey depends only on whether the leaves of Π' are also grey. To define the set Φ_2 of partial interpolants, we therefore exploit the definition of RB-partitions and adjust (8) to the following three cases. In the sequel we refer by (a), ..., (j) to the formulas denoted by (a), ..., (j) in (8).
Case (i). All leaves C_i of Π' are grey. Any formula from (8) is a partial interpolant and:

$$\Phi_2 = \{\text{(a)}, \text{(b)}, \text{(c)}, \text{(d)}, \text{(e)}, \text{(f)}, \text{(g)}, \text{(h)}, \text{(i)}, \text{(j)}\}.$$

Case (ii). Some leaves of Π' are red. Let us write $\{C_i\} = \{D_k\} \cup \{C_j\}$, where C_j are the grey leaves and D_k denote the red leaves of Π'. Using the definition of RB-partitions, D_k are also leaves of Π. From property (RB1), we conclude $R \vdash \bigwedge D_k$ and take $I_{D_k} = \bot$ as the partial interpolants of D_k. From (RB2), we have $\vdash \bigvee \overline{C_i} \vee C$. Then from $R \vdash \bigwedge D_k$ and $\vdash \bigvee \overline{C_i} \vee C$, we derive $R \vdash \bigvee \overline{C_j} \vee C$.

Thus, restricting ourselves to the grey leaves C_j, the constraints (5) become:

$$R \vdash \bigwedge(I_{C_j} \vee C_j) \wedge (\bigvee \overline{C_j} \vee C) \wedge \overline{C} \rightarrow I_C, \qquad B \vdash I_C \rightarrow \bigvee(I_{C_j} \wedge \overline{C_j}) \vee C.$$

Let (a′),(b′),(c′),(f′),(g′) denote the formulas obtained from (a),(b),(c),(f),(g), by replacing C_i with C_j. It is not difficult to prove that any formula (a′),(b′),(c′),(f′),(g′) can be taken as a partial interpolant I_C of C. Hence:

$$\Phi_2 = \{(\mathrm{a}'), (\mathrm{b}'), (\mathrm{c}'), (\mathrm{f}'), (\mathrm{g}')\}.$$

Case (iii). Some leaves of Π' are blue. Using the notation of Case (ii), eq. (5) imposes the following constraints over I_C:

$$R \vdash \bigwedge(I_{C_j} \vee C_j) \wedge \overline{C} \rightarrow I_C, \qquad B \vdash I_{C_j} \rightarrow \bigvee(I_{C_j} \wedge \overline{C_j}) \vee (\bigwedge C_j \wedge \overline{C}) \vee C.$$

Let (d′),(e′),(h′),(i′),(j′) denote the formulas obtained from (d),(e),(h),(i),(j), by replacing C_i with C_j. Then, (d′),(e′),(h′),(i′),(j′) are partial interpolant I_C of C. Hence:

$$\Phi_2 = \{(\mathrm{d}'),(\mathrm{e}'),(\mathrm{h}'),(\mathrm{i}'),(\mathrm{j}')\}.$$

Interpolation Algorithm for First-Order Inference Systems. Alg. 1 yields *a new interpolation procedure for arbitrary first-order inference systems*, as follows. It takes as input an RB-refutation Π and an RB-partition $\mathcal{P}$ of Π. The input functions f, g of Alg. 1 satisfy the condition $f(C) = g(C) = C$, for every C, whereas the *construct* function is defined by using the above given sets Φ_1 and Φ_2 in (6). With these considerations on its inputs, Alg. 1 returns an interpolant I of R and B by recursively computing the partial interpolants of leaves and sub-roots of Π. The (partial) interpolants derived by Alg. 1 are of different strength and are computed from the same proof. We next discuss the strength of our partial interpolants, and relate them to other methods, in particular to the local derivation framework of [10].

Logical Relations among Partial Interpolants. Fig. 2 shows the relationship among the formulas from (8) in terms of logical strength. An arrow is drawn between two formulas denoted by (x) and (y) if (x)→(y). All implications in Fig. 2 are valid, which can be shown by simply applying resolution on (x)∧$\overline{(\mathrm{y})}$. The logical relations of Fig. 2 correspond to Case (i) above; the relations corresponding to Cases (ii) and (iii) are special cases of Fig. 2.

Based on the partial interpolants of (8), we believe that we are now ready to answer the questions raised in §1 about interpolants quality. As illustrated in Fig. 2, *our partial interpolants differ in their logical strength.* Different choices of partial interpolants yield weaker or stronger interpolants in Alg. 1. While previous methods, e.g. [4,8,16], also address logical strength, they are restricted to quantifier-free interpolants. Our approach instead characterizes interpolants strength in full first-order logic, and can be used to derive interpolants in arbitrary first-order theories. The quality of interpolants clearly depends on their ap-

plication to program verification, and weaker/stronger interpolants might yield better scalability of interpolation-based verification approaches.

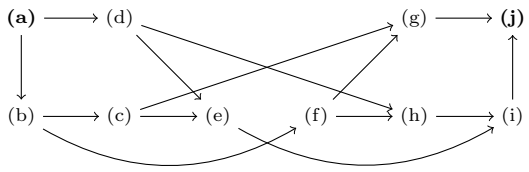

Fig. 2. Implication graph of partial interpolants in first-order inference systems.

The Local Derivations Framework. The interpolation algorithm of [10] extracts interpolants from so-called *local derivations*, also called *split derivations* in [9]. An inference in a local derivation cannot use both red and blue symbols; inferences of local derivations are called local inferences. It is easy to see that local proofs are special cases of RB-derivations.

Given a local RB-derivation Π, by making use of our notation, the algorithm of [10] can be summarized as follows. A partition $\mathcal{P}$ of Π is first created such that each sub-derivation Π' of Π is a maximal red or a maximal blue sub-derivation. Next, partial interpolants are constructed as given below:

- If C is a grey sub-leaf of Π, then: $\Phi_1 = \{C\}$, if $R \vdash C$; $\{\overline{C}\}$, if $B \vdash C$.
- If C is a grey sub-root of a sub-derivation Π' with leaves $C_1, \ldots, C_n$, then $C_1, \ldots, C_n \vdash C$. Let $\{C_j\}$ denote the set of grey leaves of Π'. Hence, $\{C_j\} \subseteq \{C_1, \ldots, C_n\}$ and:

$$\Phi_2 = \begin{cases} \{\bigwedge_j (C_j \vee I_{C_j}) \wedge \bigvee_j \overline{C_j}\}, & \text{if } \Pi' \text{ is a red sub-derivation;} \\ \{\bigwedge_j (C_j \vee I_{C_j})\}, & \text{if } \Pi' \text{ is a blue sub-derivation.} \end{cases}$$

It is therefore not hard to argue that the algorithm of [10] is a special case of Alg. 1. The partial interpolants generated by [10] are a subset of the partial interpolants we compute. In particular, if Π' is a red (respectively, blue) sub-derivation, then the partial interpolant of the sub-root C of Π' in [10] corresponds to our formula (b′) (respectively, (e′)) defined before.

Note that the sets Φ_1 and Φ_2 computed by [10] contain exactly one formula, giving thus exactly one interpolant, while the cardinality of Φ_1 and Φ_2 in our method can be greater than 1 (lines 4 and 13 of Alg. 1). Moreover, some of our interpolants cannot be obtained by other methods – see Example 1.

Example 2. We illustrate our first-order interpolation procedure by using the formulas R and B of Example 1. Consider the RB-refutation Π given in Fig. 1 and take the RB-partition $\mathcal{P} = \{\Pi', \Pi''\}$, where Π' and Π'' are respectively given in Fig. 3 and Fig. 4.
By applying Alg. 1, we first visit the sub-derivation Π'' and compute $I_{a=b}$. Since Π'' has red leaves, the set of partial interpolants corresponding to the root $a = b$

of Π'' is: {(a′), (b′), (c′), (f′), (g′)}. Since all the sub-leaves of Π'' are colored leaves, (a′),(b′),(c′),(f′),(g′) respectively reduce to $a = b \wedge a \neq b$, $\bot$, $a = b$, $\bot$, $a = b$. The set of partial interpolants $I_{a=b}$ is thus given by: $\{a = b, \bot\}$. Next, we

$$\dfrac{\dfrac{a = b}{f(a) = f(b)} \qquad \dfrac{f(a) \neq f(h) \qquad \dfrac{h = b}{f(h) = f(b)}}{f(a) \neq f(b)}}{\bot}$$

Fig. 3. Sub-derivation Π'.

$$\dfrac{\dfrac{\forall z(z = c) \qquad a = c}{\forall z(z = a)}}{a = b}$$

Fig. 4. Sub-derivation Π''.

visit the sub-derivation Π'. As Π' has blue leaves, the set of partial interpolants corresponding to the root $\bot$ of Π' is $\{(\mathrm{d}'), (\mathrm{e}'), (\mathrm{h}'), (\mathrm{i}'), (\mathrm{j}')\}$. Since Π' has two grey sub-leaves, namely $a = b$ and $h = b$, the formulas (d′),(e′),(h′),(i′),(j′) are simplified, yielding the following set of partial interpolants $I_\bot$: $\{(I_{a=b} \vee a = b) \wedge (I_{h=b} \vee h = b), (I_{a=b} \wedge a \neq b) \vee (I_{h=b} \wedge h \neq b) \vee (a = b \wedge h = b)\}$. To derive the partial interpolant $I_{h=b}$, note that $h = b$ is the only grey leaf of B. Therefore, the set of partial interpolants $I_{h=b}$ is given by $\{\top, h \neq b\}$. Using these results, the set of (partial) interpolants $I_\bot$ is finally given by $\{a = b, h \neq b \vee (a = b \wedge h = b)\}$.

The RB-partition we used here is different from the one used in [10]. The flexibility in choosing RB-partitions in Alg. 1 allows us to derive *quantifier-free interpolants* from Fig. 1.

Summarizing, Fig. 2 characterizes the logical strength of the interpolants derived by our method, and hence offers a theoretical approach in finding a good interpolant when one is interested in deriving a logically strong/weak interpolant. A natural question to be further studied is whether a given refutation admits an RB-partition $\mathcal{P}$. It is even more interesting to understand which inference systems yield always an RB-partition of an RB-refutation. To some extent, the works of [6,10] answer these questions by considering *local derivations*. In [6], it is shown that non-local derivations in some cases can be translated into local ones, by existentially quantifying away colored uninterpreted constants; such a transformation comes thus at the price of introducing quantifiers. Further, [10] proves that an extension of the quantifier-free superposition calculus with quantifier-free linear rational arithmetic always guarantees local derivations. Since local derivations are special cases of RB-derivations, the results of [6, 10] also apply to our framework. Deriving sufficient and/or necessary conditions over RB-partitions of RB-derivations is an interesting task to be further investigated.

5.1 Interpolation in the Labeled Hyper-Resolution Framework

Alg. 1 yields a new interpolation procedure for arbitrary first-order inference systems and generalizes existing first-order interpolation methods [6, 10]. It is not hard to prove that, by appropriately choosing $f, g, construct$ and $\mathcal{P}$ in Alg. 1, our approach also yields a new interpolation procedure for the propositional hyper-resolution inference system. Moreover, when adjusting Fig. 2 to that system, our method is able to generate the interpolants produced by the labeled

hyper-resolution framework of [16], as well as additional interpolants that are logically weaker/stronger. In the following, we briefly sketch how Alg. 1 can be instantiated in the hyper-resolution system; for a detailed presentation we refer the reader to [1].

The *hyper-resolution (HR) system* is an inference system that uses a single inference rule, called the *hyper-resolution rule*:

$$\frac{\overline{p_1} \vee \cdots \vee \overline{p_{n-1}} \vee E \qquad D_1 \vee p_1 \qquad \ldots \qquad D_{n-1} \vee p_{n-1}}{\bigvee D_i \vee E}$$

where $p_1, \ldots, p_n$ are literals, called *pivots*, and $D_1, \ldots, D_n, E$ are clauses.

Let Σ be a signature. We introduce a *restriction operator* $|_\Sigma$ over clauses C: $C|_\Sigma$ is the disjunction of the literals l_j of C such that l_j are in Σ. We denote $C|_R = C|_{\Sigma_R \setminus \Sigma_{RB}}$, $C|_B = C|_{\Sigma_B \setminus \Sigma_{RB}}$, $C|_{RB} = C|_{\Sigma_{RB}}$ and say that C is respectively restricted to its red, blue, grey symbols. For each clause C and inference in Π, we define two arbitrary subsets $\Delta_R^C, \Delta_B^C \subseteq \Sigma_{RB}$ of grey symbols, where $\Delta_R^C \cup \Delta_B^C = \Sigma_{RB}$, and write $C|_{R\Delta_R^C} = C|_R \vee C|_{\Delta_R^C}$, $C|_{B\Delta_B^C} = C|_B \vee C|_{\Delta_B^C}$. Δ_R^C, Δ_B^C need not to be the same for the inferences where C is involved: for example, a grey symbol of C can be treated as red in the inference where C is the conclusion, and as blue in an inference where C is a premise. We now define $f, g, \mathcal{P}, \mathit{construct}$ of Alg. 1, by adapting the notation of §5 to the HR system.

Definition of f and g. We take f and g such that, for every C:

$$f(C) = C|_{R\Delta_R^C}, \quad g(C) = C|_{B\Delta_B^C}. \tag{9}$$

Definition of $\mathcal{P}$. We fix the partition $\mathcal{P}$ of an RB-derivation to a so-called *HR-partition*, so that for each sub-derivation with root C and leaves $C_0, \ldots, C_{n-1}$, the inference $\frac{C_0 \ \ldots \ C_{n-1}}{C}$ is an application of the hyper-resolution rule.

Definition of *construct* for Partial Interpolants of Leaves. The constraints of (2) over the partial interpolants $I_C \in \mathcal{L}_{RB}$, for a leaf C, reduce to:

$$R \vdash C \wedge \overline{C|_{R\Delta_R^C}} \rightarrow I_C, \qquad B \vdash I_C \rightarrow C|_{\Delta_B^C}, \qquad \text{if } R \vdash C;$$
$$R \vdash \overline{C|_{\Delta_R^C}} \rightarrow I_C, \qquad B \vdash I_B \rightarrow \overline{C} \vee C|_{B\Delta_B^C}, \qquad \text{if } B \vdash C.$$

A set Φ_1 of partial interpolants is defined as:

$$\Phi_1 = \{C|_{\Delta_B^C}\}, \ \text{if } R \vdash C; \quad \{\overline{C|_{\Delta_R^C}}\}, \ \text{if } B \vdash C.$$

Definition of *construct* for Partial Interpolants of Sub-Roots. Let C be the root of a sub-derivation $\Pi' \in \mathcal{P}$, and let $C_0, \ldots, C_{n-1}$ denote the leaves of Π'. Further, denote by $\Delta_R^i = \Delta_R^{C_i}$ and $\Delta_B^i = \Delta_B^{C_i}$. The constraints of eq. (5) over the partial interpolants $I_C \in \mathcal{L}_{RB}$ are simplified to:

$$\begin{aligned} &R \vdash \textstyle\bigwedge (I_{C_i} \vee C_i|_{R\Delta_R^i}) \wedge (\bigvee \overline{C_i} \vee C) \wedge \overline{C|_{R\Delta_R^C}} \rightarrow I_C, \\ &B \vdash I_C \rightarrow \textstyle\bigvee (I_{C_i} \wedge \overline{C_i|_{B\Delta_B^i}}) \vee (\bigwedge C_i \wedge \overline{C}) \vee C|_{B\Delta_B^C} \end{aligned} \tag{10}$$

Any formula $I_C \in \Phi_2$ thus satisfies eq. (10). A potential set Φ_2 of partial interpolants can be built as in eq. (8), with the logical relationship among them similar to Fig. 2. For brevity, we only mention here:

$$\text{(d)} \quad (I_{C_0} \vee (E \vee \bigvee \overline{p_i})|_{R\Delta^0_R}) \wedge \bigwedge (I_{C_i} \vee (D_i \vee p_i)|_{R\Delta^i_R}) \wedge \overline{(\bigvee D_i \vee E)}|_{R\Delta^C_R};$$

$$\text{(g)} \quad (I_{C_0} \wedge \overline{(E \vee \bigvee \overline{p_i})}|_{B\Delta^0_B}) \vee \bigvee (I_{C_i} \wedge \overline{(D_i \vee p_i)}|_{B\Delta^i_B}) \vee (\bigvee D_i \vee E)|_{B\Delta^C_B}.$$

where (d)$\rightarrow$ (g). Similarly to §5, we conclude that our parametric interpolation framework derives various interpolants in the HR system. The logical relations among these interpolants allow one to explore the strength and structure of interpolants, deriving hence interpolants which are good in the sense of their logical expressiveness.

The Labeled Hyper-Resolution Framework. We now relate our method to the labeled hyper-resolution framework of [16].

The algorithm of [16] relies on a *labeling function* L, which assigns a label $L(l, C) \in \{r, b, rb, \bot\}$ to each clause C in a derivation Π. The leaves are first labeled. Then, the label $L(l, C)$ of a literal l in the conclusion C of an inference with premises $C_0, \ldots, C_{n-1}$ is computed as $L(l, C) = L(l, C_0) \sqcup \cdots \sqcup L(l, C_{n-1})$, where $\sqcup$ is the join operator of the lattice defined by Fig. 5. Labels for the pivots are also computed in this way.

Given an RB-refutation Π, the algorithm of [16] can be summarized as follows:

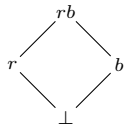

Fig. 5. The Hasse Diagram of $\sqcup$.

- If C is a leaf of Π, then: $\Phi_1 = \{C|_b\}$, if $R \vdash C$; $\{\overline{C|_r}\}$, if $B \vdash C$, where $C|_b$ and $C|_r$ denote the restriction of C to literals with label b and r.
- If C is the conclusion of the HR-rule with premises $C_0, \ldots, C_{n-1}$, then, for some literals $p_1, \ldots, p_{n-1}$ and clauses $D_1, \ldots, D_{n-1}, E$, we have $C_0 = \overline{p_1} \vee \cdots \vee \overline{p_{n-1}} \vee E$, $C_1 = D_1 \vee p_1$, $\ldots$, $C_{n-1} = D_{n-1} \vee p_{n-1}$, and $C = \bigvee D_i \vee E$. The pivots p_i are assumed to have the same label in [16]. Then:

$$\Phi_2 = \begin{cases} I_{C_0} \vee \bigvee_{i=1}^{n-1} I_{C_i}, & \text{if } L(p_i, D_i \vee p_i) \sqcup L(\overline{p_i}, \bigvee \overline{p_i} \vee E) = r; \\ I_{C_0} \wedge \bigwedge_{i=1}^{n-1} I_{C_i}, & \text{if } L(p_i, D_i \vee p_i) \sqcup L(\overline{p_i}, \bigvee \overline{p_i} \vee E) = b; \\ \begin{cases} (I_{C_0} \vee \bigvee \overline{p_i}) \wedge \bigwedge_{i=1}^{n-1} (p_i \vee I_{C_i}), \\ (I_{C_0} \wedge \bigwedge p_i) \vee \bigvee_{i=1}^{n-1} (\overline{p_i} \wedge I_{C_i}) \end{cases}, & \text{if } L(p_i, D_i \vee p_i) \sqcup L(\overline{p_i}, \bigvee \overline{p_i} \vee E) = rb. \end{cases}$$

We argue that Alg. 1 in the HR system generalizes the method of [16]; the behavior of the labeling function on the shared literals can in fact be simulated in our framework, by assigning appropriate sets Δ^C_R, Δ^C_B to every clause C in every inference.

Consider a leaf C of the RB-refutation Π, such that $C \in \mathcal{L}_R$. Using [16], the red literals of C are labeled with r, and the grey literals with one of the labels r, b, rb. The partial interpolant $C|_b$ is thus a sub-clause of $C|_{RB}$. We then fix

Δ_B^C such that $C|_b = C|_{\Delta_B^C}$, and hence our partial interpolant is also a partial interpolant of [16]. A similar argument holds when $C \in \mathcal{L}_B$.

Consider now an arbitrary HR inference in Π, with root C and leaves C_i. The relation (d)→(g) can be further exploited to obtain new partial interpolants by removing the colored literals of D_i and E in (d) and $\overline{\text{(g)}}$ using the HR rule:

$$\text{(m)} \quad (I_{C_0} \vee E|_{\Delta_R^0} \vee \bigvee \overline{p_i}|_{R\Delta_R^0}) \wedge \bigwedge (I_{C_i} \vee D_i|_{\Delta_R^i} \vee p_i|_{R\Delta_R^i}) \wedge \bigwedge \overline{D_i}|_{\Delta_R^C} \wedge \overline{E}|_{\Delta_R^C};$$

$$\text{(n)} \quad (I_{C_0} \wedge \overline{E}|_{\Delta_B^0} \wedge \bigwedge p_i|_{B\Delta_B^0}) \vee \bigvee (I_{C_i} \wedge \overline{D_i}|_{\Delta_B^i} \wedge \overline{p_i}|_{B\Delta_B^i}) \vee \bigvee D_i|_{\Delta_B^C} \vee E|_{\Delta_B^C},$$

It is always possible to split a HR inference in a sequence of HR inferences so that the labeling of the pivots is achieved as in [16]. In turn, the labeling of [16] allows us to further simplify the formulas (m) and (n) above and deriving the partial interpolants of [16] as special cases of these formulas. For example, if if the label is r, then the Δ_R, Δ_B sets are chosen so that, if p_i is grey, then $p_i \in \Delta_R^i \setminus \Delta_B^i$ and $\overline{p_i} \in \Delta_R^0 \setminus \Delta_B^0$. In this case, (n) reduces to the formula

$$(I_{C_0} \wedge \overline{E}|_{\Delta_B^0}) \vee \bigvee (I_{C_i} \wedge \overline{D_i}|_{\Delta_B^i}) \vee \bigvee D_i|_{\Delta_B^C} \vee E|_{\Delta_B^C},$$

which generalizes the partial interpolant

$$\bigvee_{i=0}^{n-1} I_{C_i}$$

of [16]. In a similar way, the partial interpolants generalizing the results of [16] are also obtained when considering the label b, respectively rb, in our framework.

Summarizing, when instantiating Alg. 1 in the HR system, we explore various good interpolants w.r.t. to their logical strength, some of these interpolants generalizing the ones derived in [16].

6 Conclusions

In this paper we proposed a new parametric interpolation framework for arbitrary first-order theories and inference systems. The main advantage of our framework is its ability to compute various interpolants of different structure and strength, with or without quantifiers, from the same proof. We described our method in relation with well-known interpolation algorithms, that respectively address local derivations in first-order logic and the propositional hyper-resolution system, and showed that they can be regarded as instantiations of our method.

Our work makes the first step towards a theoretical formalization of a generic interpolation approach, characterizing the notion of a "good" interpolant in terms of structure and logical strength. We believe our parametric interpolation algorithm can be adjusted and instantiated to cover a range of previously proposed systems (some being discussed in the paper) as well as new ones. As future work, we will study the relationships among specific inference systems

and theories, and features of the derivations that can be produced; the goal is to obtain efficient interpolation algorithms specialized to various theories. On the practical side, we intend to apply our work on examples coming from bounded model checking and/or invariant discovery, in order to assess the practical impact on concrete verification problems.

Acknowledgements. We acknowledge funding from the Austrian FWF grants S11410-N23 and T425-N23, the Austrian WWTF grant ICT C-050, and ICT COST Action IC0901.

References

1. A Parametric Interpolation Framework for First-Order Theories - Extended Version. `http://www.inf.usi.ch/phd/rollini/KRS13ext.pdf`.
2. A. Albarghouthi and K. L. McMillan. Beautiful Interpolants. In *CAV*, pages 313–329, 2013.
3. W. Craig. Three Uses of the Herbrand-Gentzen Theorem in Relating Model Theory and Proof Theory. *Journal of Symbolic Logic*, 22(3):269–285, 1957.
4. V. D'Silva, D. Kroening, M. Purandare, and G. Weissenbacher. Interpolant Strength. In *VMCAI*, pages 129–145, 2010.
5. T. A. Henzinger, R. Jhala, R. Majumdar, and K. L. McMillan. Abstractions from Proofs. In *POPL*, pages 232–244, 2004.
6. K. Hoder, L. Kovács, and A. Voronkov. Playing in the Grey area of Proofs. In *POPL*, pages 259–272, 2012.
7. G. Huang. Constructing Craig Interpolation Formulas. In *COCOON*, pages 181–190, 1995.
8. R. Jhala and K. L. McMillan. Interpolant-Based Transition Relation Approximation. In *CAV*, pages 39–51, 2005.
9. R. Jhala and K. L. McMillan. A Practical and Complete Approach to Predicate Refinement. In *TACAS*, pages 459–473, 2006.
10. L. Kovács and A. Voronkov. Interpolation and Symbol Elimination. In *CADE*, pages 199–213, 2009.
11. J. Krajícek. Interpolation Theorems, Lower Bounds for Proof Systems, and Independence Results for Bounded Arithmetic. *Journal of Symbolic Logic*, 62(2):457–486, 1997.
12. K. 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. K. L. McMillan. Quantified Invariant Generation Using an Interpolating Saturation Prover. In *TACAS*, pages 413–427, 2008.
15. P. Pudlák. Lower Bounds for Resolution and Cutting Plane Proofs and Monotone Computations. *Journal of Symbolic Logic*, 62(3):981–998, 1997.
16. G. Weissenbacher. Interpolant Strength Revisited. In *SAT*, pages 312–326, 2012.
17. G. Yorsh and M. Musuvathi. A Combination Method for Generating Interpolants. In *CADE*, pages 353–368, 2005.