A Parametric Interpolation Framework for First-Order Theories

Laura Kovács
Chalmers University of Technology, Sweden
laura.kovacs@chalmers.se

Simone Fulvio Rollini
USI, Switzerland
simone.fulvio.rollini@usi.ch

Natasha Sharygina
USI, Switzerland
natasha.sharygina@usi.ch

***Abstract*—Craig interpolation is successfully used in both hardware and software model checking. 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. Our framework is able to compute interpolants of different structure and strength, with or without quantifiers, from the same proof. We show that two classes of well-known interpolation algorithms, that address local proofs in first-order logic and the propositional hyper-resolution system, are instantiations of our framework.**

I. Introduction

Craig interpolation [1] 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. [2]–[4].

There exist various methods to compute interpolants from proofs. The work of [5] introduces an interpolation algorithm for propositional logic, and is generalized in [6] to generate interpolants in the combined theory of uninterpreted functions and linear arithmetic. The approaches of [7]–[9] propose another interpolation algorithm for propositional logic which is later extended in [10] to address a class of first-order theories. More recently, [11] introduces a framework that generalizes [5], [7], by analyzing the logical strength of interpolants. The work of [11] has been extended in [12] to interpolation in the hyper-resolution system. The methods described in [13], [14] 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 [13] or split proofs [3].

While interpolation-based verification techniques crucially depend to which extent nice interpolants can be automatically generated, there is no general criteria for defining the notion of a "good" interpolant. The work of [11] suggests that logically weaker interpolants are more useful in verification, whereas [15] emphasizes the need for logically strong interpolants in model checking and predicate abstraction. Hence, interpolants of different strength are needed in different verification frameworks. However, the interpolants generated by the wide range of existing interpolation algorithms are not always comparable. Even more, existing methods might not allow to derive interpolants of different structure and strength.

In this paper we address some of these issues and introduce a new theoretical framework, called *parametric interpolation framework*, for arbitrary theories and inference systems. We show that the afore-cited interpolant generation methods 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.

Contributions. The main contribution of this paper comes with the *theoretical formalization of a new parametric interpolation framework* (§IV). We show that this framework generalizes existing interpolation algorithms for first-order theories (§V) and propositional logic (§VI). We illustrate the kind of interpolants we produce (§III) and show how the interpolation algorithms of [11]–[13] can be regarded as special cases of our method in the context of first-order (§V) and hyper-resolution inference systems (§VI).

When compared to [13], 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 [13]; it can also produce a family of interpolants which contains the interpolants of [13]. Within this family, we relate and compare the interpolants by their logical strength. The results of [13] about the existence of local proofs in the superposition calculus and turning non-local proofs into local ones in the style of [14] can be naturally extended to our framework. Remarkably, our method allows to compute *quantifier-free interpolants* for problems on which [13] can only derive quantified interpolants (see §III).

Referring to [11], [12], our approach is different in the following aspects. We first 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 [12]. Finally, we also compare the produced interpolants by their logical strength.

II. Preliminaries

This section fixes our notation and recalls some required terminology by adapting the material of [13] 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 $\bot$.

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*. Further, 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*. Finally, 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 a *blue* formula if it only contains blue and grey symbols, but at least one blue symbol. In the sequel, 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:
$$R \vdash \forall C \text{ and } C \in \mathcal{L}_R \quad \text{or} \quad B \vdash \forall C \text{ and } C \in \mathcal{L}_B;$$

(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. As proved in [13], Craig interpolation can also be defined modulo theories. Symbols occurring in a theory are called *interpreted*, while all other symbols are *uninterpreted*.

III. Example

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

Example 1: Let us take the formula $\forall z(z = c) \wedge a = c \wedge g(b) = g(h)$ as R, and $f(a) \neq f(h) \wedge h = b$ as B. Then, c, g are red symbols, a, b, h are grey symbols, and f is a blue symbol. Clearly, $R \wedge B$ is unsatisfiable. A possible refutation

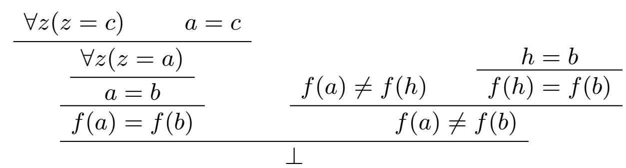

Fig. 1. Local refutation Π of $R \wedge B$.

Π of $R \wedge B$ is given in Fig. 1.

A possible interpolant of R and B is the quantified formula $\forall z(z = a)$. For example, the interpolation algorithm of [13] would compute this quantified interpolant from Fig. 1. However, when applying our method on Fig. 1, besides $\forall z(z = a)$ we are able to compute the formulas $a = b$ and $h \neq b \vee (a = b \wedge h = b)$ as possible interpolants of R and 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 [13] 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 [12]. The main advantage of our approach hence comes with the flexibility of *choosing between more than one interpolant* and *generating interpolants of different boolean structure and strength, with or without quantifiers, from the same proof.*

IV. A Parametric Interpolation Framework

In this section we present a new interpolation framework that describes a class of recursive interpolation procedures computing so-called *partial interpolants* from refutation proofs, as follows. First, partial interpolants of the leaves of the proof are derived. Next, partial interpolants for (some of) the inner nodes are derived, 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. Our parametric interpolation algorithm will be later instantiated into two specific interpolation algorithms (§V and §VI).

Let Π be an RB-refutation, corresponding to the unsatisfiability proof of $R \wedge B$. Following [13], in our approach to interpolation 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 interpolation framework is parametric in a chosen *partition* of Π. By a partition of Π we mean 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, (iii) each inference of Π belongs to some $\Pi'_i \in \mathcal{P}$. We call 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. We are then interested in computing the partial interpolants of the sub-roots C of the sub-derivations in $\mathcal{P}$.

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:

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

Note that when C is $\bot$, a partial interpolant I_C is an interpolant of R and B, since we have $R \vdash I_C$ and $B \vdash \overline{I_C}$. We also note that Def. 1 generalizes the notion of C-interpolants from [13]. Namely, by taking $f(C) = C$ and $g(C) = C$ a partial interpolant I_C is just a C-interpolant in the sense of [13], when C is grey.

Let us emphasize that in Def. 1 we are not restricted to a particular choice of f and g. That is, f and g 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 §V and §VI.

Using partial interpolants, our interpolation 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, inductively proceeding as follows. 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 of this approach, 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 next 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):

$$R \vdash C \wedge \overline{f(C)} \to I_C, \quad B \vdash I_C \to g(C), \quad I_C \in \mathcal{L}_{RB}, \quad \text{if } R \vdash C;$$
$$R \vdash \overline{f(C)} \to I_C, \quad B \vdash I_C \to \overline{C} \vee g(C), \quad I_C \in \mathcal{L}_{RB}, \quad \text{if } B \vdash C. \tag{2}$$

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:

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

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

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

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

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

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 the functions 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 the choice of Φ_1 and Φ_2 in §V and §VI.

Assuming that $construct, f, g$ are fixed, Alg. 1 returns an interpolant I of R and B as follows. 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 then selected from Φ_2 (line 14). Finally, the partial interpolant of the sub-root $\bot$ is returned as the interpolant of R and B (line 19).

Algorithm 1: **Parametric Interpolation Algorithm**
Input: Formulas R and B such that $R \wedge B \to \bot$, 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 $\Phi_1 := construct(C, f(C), g(C))$;
5 $I_C :=$ 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 $\mathcal{P}_* = \{\}$;
9 **repeat**
10 **for** each Π' in $\mathcal{P}$ such that leaves$(\Pi') \subseteq L$ **do**
11 $C := root(\Pi')$;
12 **for** each C_i in leaves(Π') **do** $I_{C_i} := \mathcal{I}[C_i]$ **endfor** ;
13 $\Phi_2 := construct(C, C_i, I_{C_i}, f(C), g(C), f(C_i), g(C_i))$;
14 $I_C :=$ select(Φ_2);
15 $\mathcal{I} := \mathcal{I} \cup \{I_C\}$; $L := L \cup \{C\}$;
16 **endfor** ;

```
    P_* := P_* ∪ {Π'};
  until P_* = P;
  Compute Interpolant
return I[⊥]
```

Alg. 1 depends on the particular choice of f, g, and $construct$, as well as of the proof 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 the values of $f, g, construct$, and $\mathcal{P}$, yielding different interpolation algorithms based on Alg. 1.

In the sequel we present two concrete choices of f, g and $construct$, together with $\mathcal{P}$. The first one, illustrated in §V, yields an interpolation procedure for arbitrary first-order inference systems. The second one, discussed in §VI, addresses the propositional hyper-resolution system. We also show that Alg. 1 generalizes the interpolation algorithms of [12], [13].

V. 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 functions 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 partition $\mathcal{P}$. 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:

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

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 $I_C \in \mathcal{L}_{RB}$:

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

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:

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

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) of RB-derivations, we conclude $R \vdash \bigwedge D_k$ and take $I_{D_k} = \bot$ as the partial interpolants of D_k. From property (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:

$$\begin{aligned} &R \vdash \bigwedge (I_{C_j} \vee C_j) \wedge (\bigvee \overline{C_j} \vee C) \wedge \overline{C} \to I_C, \\ &B \vdash I_C \to \bigvee (I_{C_j} \wedge \overline{C_j}) \vee C. \end{aligned}$$

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:

$$\Phi_2 = \{\text{(a′), (b′), (c′), (f′), (g′)}\}.$$

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

$$\begin{aligned} &R \vdash \bigwedge (I_{C_j} \vee C_j) \wedge \overline{C} \to I_C, \\ &B \vdash I_{C_j} \to \bigvee (I_{C_j} \wedge \overline{C_j}) \vee (\bigwedge C_j \wedge \overline{C}) \vee C. \end{aligned}$$

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

$$\Phi_2 = \{\text{(d′),(e′),(h′),(i′),(j′)}\}.$$

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 [13].

Logical Relations among Partial Interpolants. The logical relations among the formulas from (8) are given in Fig. 2. An arrow is drawn between two formulas denoted by (x) and (y) if (x)$\rightarrow$(y). All implications in Fig. 2 are valid, which can be shown by simply applying resolution on (x)$\wedge\overline{\text{(y)}}$. 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.

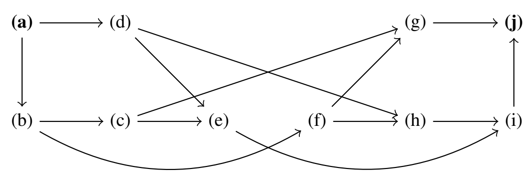

Fig. 2. Implication graph of partial interpolants in first-order inference systems.

The Local Derivations Framework. The interpolation algorithm of [13] extracts interpolants from so-called *local derivations*, also called *split derivations* in [3]. 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 [13] 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 [13] is a special case of Alg. 1. The partial interpolants generated by [13] 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 [13] corresponds to our formula (b$'$) (respectively, (e$'$)) defined before.

Note that the sets Φ_1 and Φ_2 computed by [13] 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 $\{$(d$'$), (e$'$), (h$'$), (i$'$), (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 [13]. The flexibility in choosing RB-partitions in Alg. 1 allows us to derive *quantifier-free interpolants* from Fig. 1.

Summarizing, a natural question to ask about Alg. 1 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 [13], [14] answer these questions by considering so-called *local derivations*. In [14], 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, [13] 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 [13], [14] 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.

VI. Interpolation in the Hyper-Resolution System

In this section, we study our parametric interpolation algorithm in the propositional hyper-resolution system. We start by introducing the hyper-resolution system and some notation. We then turn Alg. 1 into an *interpolation procedure for the propositional hyper-resolution system*, by fixing the choices of $f, g, construct$, and $\mathcal{P}$. We also argue that our approach generalizes the work of [12].

Formulas in the hyper-resolution system are of a special format, called *clauses*. A clause is a finite disjunction of *literals*, where a literal is either an atomic predicate p (positive literal) or a negation of p (negative literal). We assume that R and B, as well as all formulas in an RB-derivation are clauses. The *hyper-resolution 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. An RB-derivation is thus a finite tree built from applications of the hyper-resolution rule. In what follows, we write HR system and HR rule to mean respectively the hyper-resolution system and its rule.

Let us introduce the following notations specific to the clauses of the HR system. Note that the color of a formula in the HR system is defined by the color of its literals. Let Σ be a signature. We introduce a *restriction operator* $|_\Sigma$ over clauses C. The application of $|_\Sigma$ to a clause C yields a clause $C|_\Sigma$, where $C|_\Sigma$ is the disjunction of the literals l_j of C such that l_j are in Σ. We denote $C|_R = C|_{\Sigma_R \setminus \Sigma_{RB}}$ and say that $C|_R$ is restricted to the red symbols of C. Similarly, we write $C|_B = C|_{\Sigma_B \setminus \Sigma_{RB}}$ and $C|_{RB} = C|_{\Sigma_{RB}}$, where $C|_B$ and $C|_{RB}$ are restricted to the blue and grey symbols of C, respectively. Hence, a formula C in the HR system can be written as $C = C|_B \vee C|_R \vee C|_{RB}$. Next, 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 $\Delta_R^C \cup \Delta_B^C = \Sigma_{RB}$. We then write $C|_{R\Delta_R^C} = C|_R \vee C|_{\Delta_R^C}$, $C|_{B\Delta_B^C} = C|_B \vee C|_{\Delta_B^C}$. It is important to remark that 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. With these notations at hand, we now define the input parameters $f, g, construct$ and $\mathcal{P}$ of Alg. 1 in the HR system.

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

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

Note that $f(C) \vee g(C) = C|_R \vee C|_B \vee C|_{RB} = C$ for any C. Similarly to [11], $f(C)$ and $g(C)$ separate the symbols of C into sets of red and blue symbols, where the grey symbols in Σ_{RB} can be treated either as red, blue, or grey.

Definition of partition $\mathcal{P}$. We fix the partition of an RB-derivation to a so-called *HR-partition*, as defined below.

Definition 3: [HR-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 *HR-partition* if the following condition holds:

- for each sub-derivation Π'_j with root C and leaves $C_1, \ldots, C_n$, the inference $\frac{C_1 \ \ldots \ C_n}{C}$ is an application of the hyper-resolution rule. That is, for some clauses $E, D_1, \ldots, D_{n-1}$ and literals $p_1, \ldots, p_{n-1}$, C_1 can be written as $\overline{p_1} \vee \cdots \vee \overline{p_{n-1}} \vee E$, $C_2, \ldots, C_n$ denote respectively $D_1 \vee p_1, \ldots, D_{n-1} \vee p_{n-1}$, and C is $\bigvee D_i \vee E$.

In this section we fix $\mathcal{P}$ *to be an HR-partition*, and proceed to the definition of the $construct$ function for partial interpolants.

Definition of $construct$ for Partial Interpolants of Leaves. Let C be a leaf of Π. Note that, if $R \vdash C$, then we have $C \in \mathcal{L}_R$. Similarly, if $B \vdash C$ then $C \in \mathcal{L}_B$ holds. Therefore, by using the definition of f and g from (9), the constraints of (2) over the partial interpolants $I_C \in \mathcal{L}_{RB}$ reduce to:

$$\begin{array}{lll} R \vdash C \wedge \overline{C|_{R\Delta_R^C}} \to I_C, & B \vdash I_C \to C|_{\Delta_B^C}, & \text{if } R \vdash C; \\ R \vdash \overline{C|_{\Delta_R^C}} \to I_C & B \vdash I_B \to \overline{C} \vee C|_{B\Delta_B^C}, & \text{if } B \vdash C. \end{array}$$

By satisfying the above constraints, 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 $C_1, \ldots, C_n$ denote the leaves of Π'. Further, denote by $\Delta_R^i = \Delta_R^{C_i}$ and $\Delta_B^i = \Delta_B^{C_i}$. Considering the definitions of f and g from (9), the constraints of eq. (5) over the partial interpolants $I_C \in \mathcal{L}_{RB}$ are simplified to:

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

Any formula $I_C \in \Phi_2$ thus satisfies eq. (10). A potential set Φ_2 of partial interpolants therefore consists of the following ten formulas (similarly to §V, annotated from (a) to (j)):

$$\begin{array}{ll} \text{(a)} & \bigwedge(I_{C_i} \vee C_i|_{R\Delta_R^i}) \wedge (\bigvee \overline{C_i} \vee C) \wedge \overline{C|_{R\Delta_R^C}} \\ \text{(b)} & \bigwedge(I_{C_i} \vee C_i|_{R\Delta_R^i}) \wedge (\bigvee \overline{C_i}) \\ \text{(c)} & \bigwedge(I_{C_i} \vee C_i|_{R\Delta_R^i}) \wedge (\bigvee \overline{C_i} \vee C) \\ \text{(d)} & \bigwedge(I_{C_i} \vee C_i|_{R\Delta_R^i}) \wedge \overline{C|_{R\Delta_R^C}} \\ \text{(e)} & \bigwedge(I_{C_i} \vee C_i|_{R\Delta_R^i}) \\ \text{(f)} & \bigvee(I_{C_i} \wedge \overline{C_i|_{B\Delta_B^i}}) \\ \text{(g)} & \bigvee(I_{C_i} \wedge \overline{C_i|_{B\Delta_B^i}}) \vee C|_{B\Delta_B^C} \\ \text{(h)} & \bigvee(I_{C_i} \wedge \overline{C_i|_{B\Delta_B^i}}) \vee (\bigwedge C_i \wedge \overline{C}) \\ \text{(i)} & \bigvee(I_{C_i} \wedge \overline{C_i|_{B\Delta_B^i}}) \vee (\bigwedge C_i) \\ \text{(j)} & \bigvee(I_{C_i} \wedge \overline{C_i|_{B\Delta_B^i}}) \vee (\bigwedge C_i \wedge \overline{C}) \vee C|_{B\Delta_B^C} \end{array} \tag{11}$$

To use the formulas from (11) as partial interpolants we need to ensure that they are grey. Similarly to §V, the definition of the set Φ_2 of partial interpolants comes by considering the following three cases.

Case (i). The root C and all leaves C_i of Π' are grey. As C is grey, we have $C|_{R\Delta_R^C} = C|_{\Delta_R^C}$ and $C|_{B\Delta_B^C} = C|_{\Delta_B^C}$. A similar result for C_i is also derived. The formulas (a),(d),(g),(j) of (11) are therefore partial interpolants I_C. Moreover, if $C = C|_{\Delta_R^C}$, the formulas (b),(f),(h) are also partial interpolants. On the other hand, if $C = C|_{\Delta_B^C}$, (c),(e),(i) also yield partial interpolants. We thus have:

$$\Phi_2 = \begin{cases} \{\text{(a),(b),(d),(f),(g),(h),(j)}\}, & \text{if } C = C|_{\Delta_R^C}; \\ \{\text{(a),(c),(d),(e),(g),(i),(j)}\}, & \text{if } C = C|_{\Delta_B^C}; \\ \{\text{(a),(d),(g),(j)}\}, & \text{otherwise.} \end{cases}$$

Case (ii). Some leaves of Π' are red. Similarly to §V, we write $\{C_i\} = \{D_k\} \vee \{C_j\}$, where C_j are the grey leaves (i.e. disjunction of grey literals) and D_k denote the red leaves of Π'. Let (a′),(b′),(c′),(f′),(g′) denote the formulas obtained from (a),(b),(c),(f),(g), by replacing C_i with C_j. We then have:

$$\Phi_2 = \begin{cases} \{\text{(a}'\text{),(b}'\text{),(f}'\text{),(g}'\text{)}\}, & \text{if } C = C|_{\Delta_R^C}; \\ \{\text{(a}'\text{),(c}'\text{),(g}'\text{)}\}, & \text{if } C = C|_{\Delta_B^C}; \\ \{\text{(a}'\text{),(g}'\text{)}\}, & \text{otherwise.} \end{cases}$$

Case (iii). Some leaves of Π' are blue. Using the notation of Case (ii), let (d′),(e′),(h′),(i′),(j′) denote the formulas obtained from (d),(e),(h),(i),(j), by replacing C_i with C_j. Then:

$$\Phi_2 = \begin{cases} \{(\mathrm{d}'),(\mathrm{h}'),(\mathrm{j}')\}, & \text{if } C = C|_{\Delta_R^C}; \\ \{(\mathrm{d}'),(\mathrm{e}'),(\mathrm{i}'),(\mathrm{j}')\}, & \text{if } C = C|_{\Delta_B^C}; \\ \{(\mathrm{d}'),(\mathrm{j}')\}, & \text{otherwise.} \end{cases}$$

Interpolation Algorithm for the HR System. Alg. 1 yields *a new interpolation algorithm for the HR system*, as follows. It takes as input an RB-refutation Π and an HR-partition $\mathcal{P}$ of Π. The input functions f, g of Alg. 1 satisfy eq. (9), whereas the $construct$ function is defined by using the above specified Φ_1 and Φ_2 in (6). With such specification, Alg. 1 computes an interpolant I of R and B in the HR system.

Our interpolation method for the HR system benefits from the advantage of computing partial interpolants of different strength, from the same proof. We next discuss the strength of these partial interpolants, and relate them to the labeled HR framework of [12].

Logical Relations among Partial Interpolants. The logical relations among the formulas of (11) are given in the implication graph of Fig. 5. Similarly to Fig. 2, an arrow in Fig. 5 is drawn between two formulas denoted by (x) and (y) if (x)→(y) holds. Furthermore, an arrow annotated by $\triangle$ (resp. $\triangledown$) in Fig. 5 is drawn between two formulas (x) and (y) if (x)→(y) holds under the assumption that $C = C|_{\Delta_R^C}$ (resp. $C = C|_{\Delta_B^C}$). Fig. 5 shows that some of the implications do not always hold: (a)→(b) and (d)→(h) hold only if $C = C|_{\Delta_R^C}$, while (c)→(g) and (i)→(j) hold only if $C = C|_{\Delta_B^C}$.

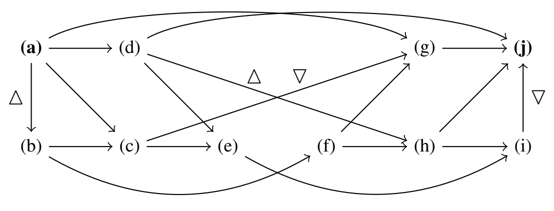

Fig. 5. Implication graph of partial interpolants in the HR system.

The Labeled Hyper-Resolution Framework. We now relate Alg. 1 to the approach of [12] in the HR system. The algorithm of [12] relies on a so-called *labeling function* L. Given a derivation Π, L first assigns labels to the literals l in the leaves C of Π. Labels are denoted by $L(l, C)$ where $L(l, C) \in \{r, b, rb, \bot\}$, with the following meaning in our coloring approach: a red literal has label r, a blue literal b and a grey literal can have label r, b, rb. Next, the label $L(l, C)$ of a literal l in the conclusion C of an inference with premises $C_1, \ldots, C_n$ is computed from the labels $L(l, C_1), \ldots, L(l, C_n)$, under the assumption that $L(l, C_i) = \bot$ if l does not appear in C_i. Namely, $L(l, C) = L(l, C_1) \sqcup \cdots \sqcup L(l, C_n)$, where $\sqcup$ is the join operator of the lattice defined by Fig. 6. For example, $r \sqcup b = rb$ and $b \sqcup \bot = b$. Labels for the pivots of a HR inference are also computed in this way.

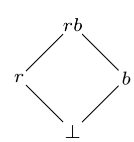

Fig. 6. The Hasse Diagram of $\sqcup$.

Given an RB-refutation Π, using our notation, the algorithm of [12] can be summarized as follows:

- If C is a leaf of Π, then:

$$\Phi_1 = \{C|_b\}, \text{ if } R \vdash C; \quad \{\overline{C|_r}\}, \text{ if } B \vdash C,$$

where $C|_b$ and $C|_r$ denote the restriction of C to its literals with label b and r.

- If C is the conclusion of the HR-rule with premises $C_1, \ldots, C_n$, then, for some literals $p_1, \ldots, p_{n-1}$ and clauses $D_1, \ldots, D_{n-1}, E$, we have $C_1 = \overline{p_1} \vee \cdots \vee \overline{p_{n-1}} \vee E$, $C_2 = D_1 \vee p_1, \ldots, C_n = D_{n-1} \vee p_{n-1}$, and $C = \bigvee D_i \vee E$. The pivots p_i are assumed to have the same label in [12]. Then:

$$\Phi_2 = \begin{cases} I_{C_1} \vee \bigvee_{i=2}^{n} I_{C_i}, & \text{if } L(p_i, D_i \vee p_i) \sqcup L(\overline{p_i}, \bigvee \overline{p_i} \vee E) = r; \\ I_{C_1} \wedge \bigwedge_{i=2}^{n} I_{C_i}, & \text{if } L(p_i, D_i \vee p_i) \sqcup L(\overline{p_i}, \bigvee \overline{p_i} \vee E) = b; \\ \begin{cases} (I_{C_1} \vee \bigvee \overline{p_i}) \wedge \bigwedge_{i=2}^{n} (p_i \vee I_{C_i}), \\ (I_{C_1} \wedge \bigwedge p_i) \vee \bigvee_{i=2}^{n} (\overline{p_i} \wedge I_{C_i}) \end{cases}, & \text{if } L(p_i, D_i \vee p_i) \sqcup L(\overline{p_i}, \bigvee \overline{p_i} \vee E) = rb. \end{cases}$$

We argue that Alg. 1 in the HR system generalizes the method of [12]. To this end, the behavior of the labeling function on the shared literals will be simulated in our framework, by assigning appropriate sets Δ_R^C, Δ_B^C to every clause C in every inference as follows.

Consider a leaf C of the RB-refutation Π, such that $C \in \mathcal{L}_R$. Using [12], 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 [12]. 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. An HR inference is also a sub-derivation, and thus clearly satisfies the restrictions of an HR-partition of Π. Assume that, for all i, $L(p_i, D_i \vee p_i) \sqcup L(\overline{p_i}, \bigvee \overline{p_i} \vee E) = r$. According to the definition of L, we have either $p_i \in \Sigma_{R \setminus RB}$ or $p_i \in \Sigma_{RB}$, and both p_i and $\overline{p_i}$ have label r. We then choose the sets Δ_R, Δ_B such that, if p_i is grey, then $p_i \in \Delta_R^i \setminus \Delta_B^i$ and $\overline{p_i} \in \Delta_R^1 \setminus \Delta_B^1$. A similar argument holds also when $L(p_i, D_i \vee p_i) \sqcup L(\overline{p_i}, \bigvee \overline{p_i} \vee E)$ is b or rb.

We thus conclude that Alg. 1 generalizes and extends the method of [12] in a number of ways. While in [12] the label $L(l, C)$ of a literal l in the conclusion C of an HR inference is derived in a unique way from the labels of the inference premises, in our framework the sets Δ_R^C and Δ_B^C can be chosen *independently for every clause C and every inference*. An important aspect of the labeled system is that it allows to systematically compare the strength of the interpolants resulting from different labelings. In particular, in [11] a total order $\preceq$ is defined over the labels $\{r, b, rb, \bot\}$ as $b \preceq rb \preceq r \preceq \bot$. Then, $\preceq$ is extended to a partial order over labeling functions L, L', as follows: $L \preceq L'$ is defined if, for every clause C and literal l in C, $L(l, C) \preceq L'(l, C)$. If $L \preceq L'$, then the interpolant given by L is stronger than the one given by L'. Our framework also benefits from such a comparison, since in any RB-refutation we are able to simulate the labeling function L by an appropriate choice of Δ_R^C, Δ_B^C for every clause in the refutation. Therefore, for any RB-refutation and labelings L, L' such that $L \preceq L'$, the interpolant obtained with L is stronger than the one obtained with L'.

Extending the Labeled Hyper-Resolution Framework. The main advantage of the HR system is that the HR rule can be applied to remove colored literals in order to make colored formulas grey. This allows us to obtain an additional set of partial interpolants for sub-derivations containing colored sub-leaves/sub-roots. Let $\Pi' \in \mathcal{P}$ be a sub-derivation with root C and leaves $C_1, \ldots, C_n$. As $\mathcal{P}$ is an HR-partition, formulas C and C_i are as given in Def. 3. Consider now the formulas (d) and (g) from eq. (11), and replace C and C_i with the clauses from Def. 3. We thus obtain the new formulas (d) and (g):

$$\text{(d)} \quad (I_{C_1} \vee (E \vee \bigvee \overline{p_i})|_{R\Delta_R^1}) \wedge \bigwedge (I_{C_i} \vee (D_i \vee p_i)|_{R\Delta_R^i}) \wedge \overline{(\bigvee D_i \vee E)}|_{R\Delta_R^C};$$

$$\text{(g)} \quad (I_{C_1} \wedge \overline{(E \vee \bigvee \overline{p_i})}|_{B\Delta_B^1}) \vee \bigvee (I_{C_i} \wedge \overline{(D_i \vee p_i)}|_{B\Delta_B^i}) \vee (\bigvee D_i \vee E)|_{B\Delta_B^C}.$$

From Fig. 5, we have (a)→(d) and (g)→(j). It is also not hard to derive that (d) $\wedge$ $\overline{\text{(g)}}$ $\rightarrow \perp$. Therefore, we conclude that (d)→(g) also holds in the HR system. The relation (d)→(g) can be further exploited to derive intermediate pairs of formulas (x),(y), such that (d)→(x), (x)→(y) and $\overline{\text{(g)}} \rightarrow \overline{\text{(y)}}$. Our goal is 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. This reasoning gives us the following formulas:

$$\text{(m)} \quad (I_{C_1} \vee E|_{\Delta_R^1} \vee \bigvee \overline{p_i}|_{R\Delta_R^1}) \wedge \bigwedge (I_{C_i} \vee D_i|_{\Delta_R^i} \vee p_i|_{R\Delta_R^i}) \wedge \bigwedge \overline{D_i}|_{\Delta_R^C} \wedge \overline{E}|_{\Delta_R^C};$$

$$\text{(n)} \quad (I_{C_1} \wedge \overline{E}|_{\Delta_B^1} \wedge \bigwedge p_i|_{B\Delta_B^1}) \vee \bigvee (I_{C_i} \wedge \overline{D_i}|_{\Delta_B^i} \wedge \overline{p_i}|_{B\Delta_B^i}) \vee \bigvee D_i|_{\Delta_B^C} \vee E|_{\Delta_B^C},$$

It is always possible to split a HR inference in a sequence of HR inferences so that a uniform labeling of the pivots is achieved (see [12] – §4). In turn, the presence of a uniform labeling allows to further simplify (m) and (n), thus deriving the partial interpolants of [12] as special cases of our formulas (see Appendix A).

Example 3: Consider a proof Π with an HR inference, as in Fig. 7. Assume that the following partial interpolants are given: $I_1 = I_{\overline{p_1 p_2} q_1}$, $I_2 = I_{p_1 q_2}$, $I_3 = I_{p_2 q_3}$. We assume that all literals belong to $\Sigma|_{RB}$, and the pivots p_1 and p_2 are both labeled as rb.

$$\dfrac{\begin{matrix} \vdots & \vdots & \vdots \\ \overline{p_1 p_2} q_1 & p_1 q_2 & p_2 q_3 \end{matrix}}{\begin{matrix} q_1 q_2 q_3 \\ \vdots \end{matrix}}$$

Fig. 7. RB-proof Π with HR inferences.

The algorithm of [12] yields the partial interpolant $I_{q_1 q_2 q_3} = (I_1 \vee \overline{p_1} \vee \overline{p_2}) \wedge (I_2 \vee p_1) \wedge (I_3 \vee p_2)$. W.l.o.g., we assume that the sets Δ_R, Δ_B have been chosen such that, for every clause C, all shared literals of C are in $\Delta_R^C \cap \Delta_B^C$[1]. Then, our method generates the partial interpolant $I_{q_1 q_2 q_3}$ as the formula (m), simplified below:

$$I = (I_1 \vee q_1 \vee \overline{p_1} \vee \overline{p_2}) \wedge (I_2 \vee q_2 \vee p_1) \wedge (I_3 \vee q_3 \vee p_2) \wedge \overline{q_1} \wedge \overline{q_2} \wedge \overline{q_3}$$

Our interpolant I, which is stronger than the interpolant I' of [12], cannot be generated in [12]. Moreover, our method can also generate the interpolant I' of [12], by applying the HR rule with pivots q_i.

VII. Conclusions

In this paper we proposed a new parametric interpolation framework for arbitrary first-order theories and inference systems. We discussed two classes of 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. 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.

Our work makes the first step towards a theoretical formalization of a generic interpolation approach. 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 some 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. We believe that such studies will yield 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 characterize the notion of a "good" interpolant. For example, we plan to extend our method with ideas from [16] where interpolants of a certain structure are computed.

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] W. Craig, "Three uses of the Herbrand-Gentzen Theorem in Relating Model Theory and Proof Theory," *J. Symb. Log.*, vol. 22, no. 3, 1957.

[2] T. A. Henzinger, R. Jhala, R. Majumdar, and K. L. McMillan, "Abstractions from Proofs," in *POPL*, 2004, pp. 232–244.

[3] R. Jhala and K. L. McMillan, "A Practical and Complete Approach to Predicate Refinement," in *TACAS*, 2006, pp. 459–473.

[4] K. L. McMillan, "Quantified Invariant Generation Using an Interpolating Saturation Prover," in *TACAS*, 2008, pp. 413–427.

[5] ——, "Interpolation and SAT-Based Model Checking," in *CAV*, 2003.

[6] ——, "An Interpolating Theorem Prover," in *TACAS*, 2004, pp. 16–30.

[7] P. Pudlák, "Lower Bounds for Resolution and Cutting Plane Proofs and Monotone Computations," *J. Symb. Log.*, vol. 62, no. 3, 1997.

[8] J. Krajícek, "Interpolation Theorems, Lower Bounds for Proof Systems, and Independence Results for Bounded Arithmetic," *J. Symb. Log.*, vol. 62, no. 2, pp. 457–486, 1997.

[9] G. Huang, "Constructing Craig Interpolation Formulas," in *COCOON*, 1995.

[10] G. Yorsh and M. Musuvathi, "A Combination Method for Generating Interpolants," in *CADE*, 2005, pp. 353–368.

[11] V. D'Silva, D. Kroening, M. Purandare, and G. Weissenbacher, "Interpolant Strength," in *VMCAI*, 2010, pp. 129–145.

[12] G. Weissenbacher, "Interpolant Strength Revisited," in *SAT*, 2012.

[13] L. Kovács and A. Voronkov, "Interpolation and Symbol Elimination," in *CADE*, 2009, pp. 199–213.

[14] K. Hoder, L. Kovács, and A. Voronkov, "Playing in the Grey area of Proofs," in *POPL*, 2012, pp. 259–272.

[15] R. Jhala and K. L. McMillan, "Interpolant-Based Transition Relation Approximation," in *CAV*, 2005, pp. 39–51.

[16] A. Albarghouthi and K. L. McMillan, "Beautiful Interpolants," in *CAV*, 2013, to appear.

[1] This is case (C) in Appendix A. The other choices of Δ_R, Δ_B yield other generalizations of [12].

Appendix A: Extension of the Labeled Hyper-Resolution Framework

As mentioned in §VI (page 8), it is always possible to split a HR inference in a sequence of HR inferences so that a uniform labeling of the pivots is achieved (see [12] – §4). We show how this affects the choice of sets Δ_R, Δ_B in our setup, by analyzing the three possible cases:

(A) 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 $\overline{p_i} \in \Delta_R^1 \setminus \Delta_B^1$;

(B) if the label is b then, if p_i is grey, then $p_i \in \Delta_B^i \setminus \Delta_R^i$ and $\overline{p_i} \in \Delta_B^1 \setminus \Delta_R^1$;

(C) if the label is rb then $p_i \in \Delta_R^i \cap \Delta_B^i$ and $\overline{p_i} \in \Delta_R^1 \cap \Delta_B^1$.

In case (A), (n) reduces to:

$$\text{(o)} \quad (I_{C_1} \wedge \overline{E}|_{\Delta_B^1}) \vee \bigvee (I_{C_i} \wedge \overline{D_i}|_{\Delta_B^i}) \vee \bigvee D_i|_{\Delta_B^C} \vee E|_{\Delta_B^C}.$$

Formula (o) can be thus also used as an additional partial interpolant in Φ_2. A special case of (o) is the partial interpolant $\bigvee_{i=1}^n I_{C_i}$.

In case (B), (m) reduces to:

$$\text{(p)} \quad (I_{C_1} \vee E|_{\Delta_R^1}) \wedge \bigwedge (I_{C_i} \vee D_i|_{\Delta_R^i}) \wedge \bigwedge \overline{D_i}|_{\Delta_R^C} \wedge E|_{\Delta_R^C}.$$

Formula (p) can then be also used as a partial interpolant in Φ_2. A special case of (p) is $\bigwedge_{i=1}^n I_{C_i}$.

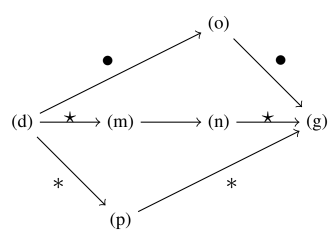

Fig. 8. Additional implication graph.

In case (C), the formulas (m) and (n) can also be used as partial interpolants in Φ_2.

The relations between Fig. 5 and the formulas (m), (n), (p), and (o) are shown in Fig. 8. Similarly to Fig. 2, an arrow in Fig. 8 is drawn between two formulas denoted by (x) and (y) if (x)→(y) holds. In addition, an arrow annotated by $\star$ (respectively, by $*$ and $\bullet$) is drawn between (x) and (y) if (x)→(y) holds under the assumption that $p_i \in (\Delta_R^i \cap \Delta_B^i)$ (respectively, $p_i \in (\Sigma_{B \setminus RB} \cup (\Delta_B^i \setminus \Delta_R^i))$ and $p_i \in (\Sigma_{R \setminus RB} \cup (\Delta_R^i \setminus \Delta_B^i)))$.