Propositional Interpolation and Abstract Interpretation

Vijay D'Silva[⋆]

Computing Laboratory, Oxford University
vijay.dsilva@comlab.ox.ac.uk

Abstract. Algorithms for computing Craig interpolants have several applications in program verification. Though different algorithms exist, the relationship between them and the properties of the interpolants they generate are not well understood. This paper is a study of interpolation algorithms for propositional resolution proofs. We show that existing interpolation algorithms are abstractions of a more general, parametrised algorithm. Further, existing algorithms reside in the coarsest abstraction that admits correct interpolation algorithms. The strength of interpolants constructed by existing interpolation algorithms and the variables they eliminate are analysed. The algorithms and their properties are formulated and analysed using abstract interpretation.

1 Introduction

Interpolation theorems provide insights about what can be expressed in a logic or derived in a proof system. An interpolation theorem states that if A and B are logical formulae such that A implies B, there is a formula I defined only over the symbols occurring in both A and B such that A implies I and I implies B. This statement was proved by Craig [8] for first order logic and has since been shown to hold for several other logics and logical theories. Consult [18] for a survey of the history and consequences of this theorem in mathematical logic. This paper is concerned with constructing interpolants from propositional resolution proofs.

An *interpolation system* is an algorithm for computing interpolants from proofs. We briefly review the use of interpolation systems for propositional resolution proofs in verification. Consider the formulae $S(x)$ encoding a set of states S, $T(x, x')$ encoding a transition relation T and $\varphi(x')$ encoding a correctness property φ. The *image* of S under the relation T is given by the formula $\exists x.S(x) \wedge T(x, x')$. The standard approach to determine if the states reachable from S satisfy the property φ is to iteratively compute images until a fixed point is reached. However, image computation and fixed point detection both involve quantifier elimination and are computationally expensive.

Consider the formula $S(x) \wedge T(x, x') \Rightarrow \varphi(x')$. If this formula is valid, the states reachable from S by a transition in T satisfy φ. Let A be the formula $S(x) \wedge T(x, x')$ and let B be the formula $\varphi(x')$ and I be an interpolant for

[⋆] Supported by Microsoft Research's European PhD Scholarship Programme.

$A \Rightarrow B$. The formula I represents a set of states that contains the image of S and satisfies the property φ. Thus, as shown by McMillan [19], one can implement a property-preserving, approximate image operator with an interpolation system. Contemporary SAT solvers are capable of generating resolution proofs, so an interpolation system for such proofs yields a verification algorithm for finite-state systems that uses only a SAT solver. The efficiency and precision of such a verification algorithm is contingent on the size and logical strength of the interpolants used. Hence, it is important to understand the properties of interpolants generated by different interpolation systems.

We are aware of three interpolation systems for propositional resolution proofs. The first, which we call the HKP-system, was discovered independently by Huang [14], Krajíček [16] and Pudlák [21]. Another was proposed by McMillan [19] and a third *parametrised* system was proposed by the author and his collaborators [10] as a generalisation of the other systems. One may however ask if the HKP-algorithm and McMillan's algorithm have properties that distinguish them from other instances of the parametrised algorithm. We answer this question in this paper and study other properties of these systems.

Contents and Organisation. In this paper, we study the family of propositional interpolation systems proposed in [10]. We ask two questions about these systems: (1) What is the structure of this space of interpolation systems and how does it relate to the HKP-system and McMillan's system? (2) How are the strength and size of interpolants generated by these systems related? Our contributions to answering these questions are the following results.

- The set of interpolation systems forms a lattice. Interpolation systems that partition variables are abstractions of this lattice. The HKP-system and McMillan's system are two of three systems in the coarsest abstraction that admits correct interpolation systems.
- The set of clauses equipped with interpolants (called extended clauses or e-clauses) is a complete lattice. An interpolation system Int defines a concrete interpretation on this lattice. The lattice of CNF formulae is an abstraction of the lattice of e-clauses and the resolution proof system is a complete abstract interpretation of Int.
- Interpolation systems and e-clauses are ordered by logical strength of interpolants giving rise to a precision order on the lattice of interpolation systems and the lattice of e-clauses. Interpolation systems that eliminate the largest and smallest set of variables from a formula are identified and shown to be different from the most abstract interpolation systems.

The paper is organised as follows: The background on propositional logic and resolution is covered in § 2. Existing interpolation systems are formalised and illustrated with examples in § 3. Some background on abstract interpretation is introduced in § 4 and applied to study the space of interpolation systems and its abstractions in § 4.1 and § 4.2. The logical strength of interpolants and the variables they contain are analysed in § 5. We discuss related work in § 6 and conclude in § 7.

2 Propositional Logic and Interpolation

Propositional logic, resolution and interpolation are introduced in this section.

Sets and functions. Let $\wp(X)$ denote the powerset of X, $X \to Y$ be the set of functions from X to Y and $f \circ g$ denote functional composition. Given $f : X \to Y$ and $S \subseteq X$, we write $f(S)$ for the set $\{f(x) \in Y | x \in S\}$.

Propositional Logic. Fix a finite set Prop of variables (propositions) for this paper. Let T and F denote true and false, respectively. The set of propositional formulae, $\mathbb{B}$, is defined as usual over the basis $\{\neg, \wedge, \vee, \Rightarrow\}$. The set of variables occurring in a formula $F \in \mathbb{B}$ is denoted $\mathrm{Var}(F)$. An *assignment* $\sigma : \mathrm{Prop} \to \{\mathsf{T}, \mathsf{F}\}$ is a function that maps variables to truth values. Let F be a formula. The *evaluation* of F under an assignment σ, written $\mathit{eval}(F, \sigma)$, is defined as usual. F is a *tautology* if $\mathit{eval}(F, \sigma) = \mathsf{T}$ for every assignment σ and F is *unsatisfiable* if $\mathit{eval}(F, \sigma) = \mathsf{F}$ for every assignment σ.

Resolution. A *literal* is a variable $x \in \mathrm{Prop}$ or its negation, denoted $\neg x$ or $\overline{x}$. For a literal t being x or $\overline{x}$, we write $\mathrm{var}(t)$ for x. A *clause* is a disjunction of literals $t_1 \vee \cdots t_k$ represented as a set $\{t_1, \ldots, t_k\}$. Let $\mathbb{C}$ be the set of all clauses. The disjunction of two clauses is denoted $C \vee D$, further simplified to $C \vee t$ if D is the singleton $\{t\}$. The *restriction* of a clause C by a formula F, $C|_F \overset{\text{def}}{=} C \cap \{x, \overline{x} | x \in \mathrm{Var}(F)\}$ is the set of literals in C over variables in F. A formula in Conjunctive Normal Form (CNF) is a conjunction of clauses, also represented as a set of clauses. A clause containing t and $\overline{t}$ is a tautology as is the empty formula $\emptyset$. The empty clause, denoted $\Box$, is unsatisfiable.

The *resolution principle* states that an assignment satisfying the clauses $C \vee x$ and $D \vee \overline{x}$ also satisfies $C \vee D$. It is given by the inference rule below.

$$\frac{C \vee x \qquad D \vee \overline{x}}{C \vee D} \quad [\mathsf{Res}]$$

The clauses $C \vee x$ and $D \vee \overline{x}$ are the *antecedents*, x is the *pivot*, and $C \vee D$ is the *resolvent.* A clause C is derived from a CNF formula F by resolution if it is the resolvent of two clauses that either occur in F or have been derived from F by resolution. The resolvent of C and D with a pivot x is denoted $\mathit{Res}(x, C, D)$. A *proof* is a sequence of resolution deductions. A *refutation* is a proof of $\Box$.

Interpolation. Consider two formulae A and B such that A implies B. Take for example $x \wedge y \Rightarrow y \vee z$. Since B does not involve x, whatever A asserts about y should be enough to imply B. Theorem 1 codifies this intuition and the proof (from [1]) gives a simple but infeasible method for interpolant construction.

Theorem 1. *For propositional formulae A and B, if $A \Rightarrow B$ is a tautology, there exists a propositional formula I, called an interpolant, such that (a) $A \Rightarrow I$ and (b) $I \Rightarrow B$ and (c)* $\mathrm{Var}(I) \subseteq \mathrm{Var}(A) \cap \mathrm{Var}(B)$.

Proof. We proceed in two steps. We first construct a formula I from A and then show that I has the requisite properties. For any set $X \subseteq \mathrm{Prop}$ and assignment σ, let $\mathrm{Pos}(X, \sigma) = \{x \in X | \sigma(x) = \mathsf{T}\}$ be the set of variables in X assigned T and let $\mathrm{Neg}(X, \sigma)$ be $X \setminus \mathrm{Pos}(X, \sigma)$. Let Y be $\mathrm{Var}(A) \cap \mathrm{Var}(B)$. Define:

$$I \overset{\text{def}}{=} \bigvee_{\sigma \in \mathsf{Mod}(A)} \left(\bigwedge_{x \in \mathrm{Pos}(Y,\sigma)} x \;\wedge\; \bigwedge_{z \in \mathrm{Neg}(Y,\sigma)} \neg z \right)$$

We show that I is an interpolant.
(a) By construction, if $\sigma \models A$, then $\sigma \models I$, so $A \Rightarrow I$ is a tautology.
(b) If $\sigma \models I$, there exists an assignment σ' such that $\sigma' \models A$ and for all $x \in \mathrm{Var}(B)$, $\sigma(x) = \sigma'(x)$. As σ and σ' agree on $\mathrm{Var}(B)$, $\mathit{eval}(B, \sigma) = \mathsf{T}$ iff $\mathit{eval}(B, \sigma') = \mathsf{T}$. From the assumption that $A \Rightarrow B$, we have that $\sigma \models B$. It follows that $I \Rightarrow B$.
(c) $\mathrm{Var}(I) \subseteq \mathrm{Var}(A) \cap \mathrm{Var}(B)$ by construction. ■

The interpolant in the proof above is constructed by existentially eliminating some variables in A. Another possibility is to universally eliminate some variables in B. A tautology $A \Rightarrow B$ can have several interpolants and the set of all interpolants forms a complete lattice [11]. The construction above examines the models of A, hence it requires time exponential in $|Y|$ and can produce exponentially large interpolants. Complexity issues aside, the design of an interpolation algorithm follows the same steps. One must provide a procedure for constructing a formula and then prove that the formula is an interpolant. In this paper, we generalise these two steps in the context of resolution.

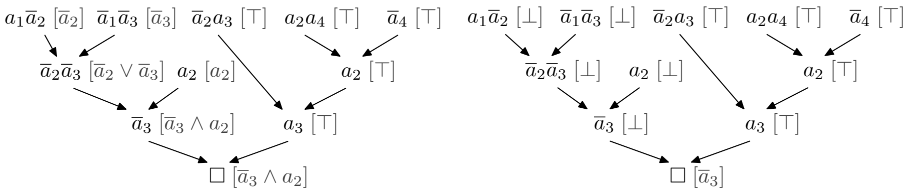
3 Interpolation Systems

Interpolation systems are introduced in this section. No new results are presented but existing systems are formally defined and explained with examples.

A *CNF pair* $\langle A, B\rangle$ is a pair of disjoint CNF formulae (that is, $A \cap B = \emptyset$). A CNF pair $\langle A, B\rangle$ is unsatisfiable if $A \wedge B$ is unsatisfiable. Given $\langle A, B\rangle$, let V_A denote $\mathrm{Var}(A) \setminus \mathrm{Var}(B)$, V_B denote $\mathrm{Var}(B) \setminus \mathrm{Var}(A)$ and $V_{\langle A,B\rangle}$ denote $\mathrm{Var}(A) \cap \mathrm{Var}(B)$. An interpolant for an unsatisfiable CNF pair is defined below.

Definition 1 (Interpolant). *An interpolant for an unsatisfiable CNF pair $\langle A, B\rangle$, is a formula I such that $A \Rightarrow I$, $I \Rightarrow \neg B$, and* $\mathrm{Var}(I) \subseteq \mathrm{Var}(A) \cap \mathrm{Var}(B)$.

An interpolant is not necessarily symmetric with respect to $\langle A, B\rangle$. If I is an interpolant for $\langle A, B\rangle$, then, $\neg I$ is an interpolant for $\langle B, A\rangle$. Interpolants are constructed inductively over the structure of a refutation. Figure 1 illustrates interpolant construction for the CNF pair $\langle A, B\rangle$, where $A = (a_1 \vee \overline{a}_2) \wedge (\overline{a}_1 \vee \overline{a}_3) \wedge a_2$ and $B = (\overline{a}_2 \vee a_3) \wedge (a_2 \vee a_4) \wedge \overline{a}_4$. McMillan's construction [19] is shown on the left and that of Huang [14], Krajíček [16] and Pudlák [21] is on the right. The formula labelling the empty clause is the interpolant for $\langle A, B\rangle$. Observe that the two methods produce different interpolants.


$a_1\overline{a}_2$ [$\overline{a}_2$]
$\overline{a}_1\overline{a}_3$ [$\overline{a}_3$]
$\overline{a}_2 a_3$ [$\top$]
$a_2 a_4$ [$\top$]
$\overline{a}_4$ [$\top$]
$\overline{a}_2\overline{a}_3$ [$\overline{a}_2 \vee \overline{a}_3$]
a_2 [a_2]
a_2 [$\top$]
$\overline{a}_3$ [$\overline{a}_3 \wedge a_2$]
a_3 [$\top$]
$\Box$ [$\overline{a}_3 \wedge a_2$]
$a_1\overline{a}_2$ [$\bot$]
$\overline{a}_1\overline{a}_3$ [$\bot$]
$\overline{a}_2 a_3$ [$\top$]
$a_2 a_4$ [$\top$]
$\overline{a}_4$ [$\top$]
$\overline{a}_2\overline{a}_3$ [$\bot$]
a_2 [$\bot$]
a_2 [$\top$]
$\overline{a}_3$ [$\bot$]
a_3 [$\top$]
$\Box$ [$\overline{a}_3$]

(a) McMillan [19]

(b) Huang [14], Krajíček [16] and Pudlák [21]

Fig. 1. Interpolant construction using systems in the literature.

We formalise these constructions as *interpolation systems*. Recall that $\mathbb{B}$ is the set of all formulae and $\mathbb{C}$ is the set of all clauses. Let $\mathcal{S} \overset{\text{def}}{=} \wp(\{\textsc{a}, \textsc{b}\})$ be a set of symbols. To reduce notation, we write $\textsc{a}$ for $\{\textsc{a}\}$, $\textsc{b}$ for $\{\textsc{b}\}$ and $\textsc{ab}$ for $\{\textsc{a}, \textsc{b}\}$. A *distinction function* is an element of $\mathbb{D} \overset{\text{def}}{=} \text{Prop} \to \mathcal{S}$. An *extended clause* (e-clause) is an element of $\mathbb{C} \times \mathbb{D} \times \mathbb{B}$. In an e-clause $E = \langle C, \Delta, I\rangle$, $cl(E) = C$ is a clause, $df(E) = \Delta$ is a distinction function and $int(E) = I$ is a *partial interpolant*. An interpolation system extends resolution to e-clauses.

Definition 2 (Interpolation System). *Let* $\mathbb{E} = \mathbb{C} \times \mathbb{D} \times \mathbb{B}$ *be a set of* extended clauses. *An interpolation system for* $\mathbb{E}$ *is a tuple* $\mathsf{Int} = \langle T, \mathsf{ERes}\rangle$, *where* $T : \wp(\mathbb{C}) \times \wp(\mathbb{C}) \to \wp(\mathbb{E})$ *is a translation function and* ERes *is an inference rule. The function* T *satisfies that for all disjoint* $A, B \in \wp(\mathbb{C})$, *a clause* $C \in A \cup B$ *iff there exists a unique* $\Delta \in \mathbb{D}$ *and* $I \in \mathbb{B}$ *such that* $\langle C, \Delta, I\rangle \in T(A, B)$. *The inference rule is of the form:*

$$\frac{\langle C_1 \vee x, \Delta_1, I_1\rangle \qquad \langle C_2 \vee \overline{x}, \Delta_2, I_1\rangle}{\langle C_1 \vee C_2, \Delta, I\rangle} \quad [\mathsf{ERes}]$$

The variable x *is called the* pivot.

An e-clause derived from E_1 and E_2 by applying ERes with a pivot x is an *e-resolvent* and is denoted $ERes(x, E_1, E_2)$. For C derived from $\langle A, B\rangle$ by resolution, the *corresponding e-clause* E is defined as:

- If $C \in A \cup B$, then E is the unique e-clause in $T(A, B)$ such that $cl(E) = C$.
- If $C = Res(x, C_1, C_2)$, then $E = ERes(x, E_1, E_2)$, where E_1 and E_2 are the corresponding e-clauses for C_1 and C_2 respectively.

Given a derivation of a clause C, the corresponding e-clause is uniquely defined. In general, there may be multiple derivations of C, and consequently, multiple e-clauses E with $cl(E) = C$. An interpolation system Int is *correct* if for every derivation of the empty clause $\Box$, the corresponding e-clause $E_\Box$ satisfies that $int(E_\Box)$ is an interpolant for $\langle A, B\rangle$. We introduce existing interpolation systems next. The first two systems do not modify the inference rule but the parametrised system does. This difference leads to the abstraction we identify in § 4.2.

Definition 3 (HKP System [14, 16, 21]). *The Huang-Krajíček-Pudlák interpolation system* $\mathsf{Int}_{HKP} = \langle T_{HKP}, \mathsf{HKPRes}\rangle$ *is defined below.*

$$T_{HKP}(A, B) \overset{\text{def}}{=} \{\langle C, \varDelta, \mathsf{F}\rangle | C \in A\} \cup \{\langle C, \varDelta, \mathsf{T}\rangle | C \in B\}$$

$$\frac{\langle C \vee x, \varDelta, I_1\rangle \qquad \langle D \vee \overline{x}, \varDelta, I_2\rangle}{\langle C \vee D, \varDelta, I\rangle}\ [\mathsf{HKPRes}]$$

$$\varDelta(x) \overset{\text{def}}{=} \begin{array}{ll} \text{A} & \textit{if } x \in \ \ V_A \\ \text{AB} & \textit{if } x \in V_{\langle A,B\rangle} \\ \text{B} & \textit{if } x \in \ \ V_B \end{array} \ \textit{ and } I \overset{\text{def}}{=} \begin{array}{ll} I_1 \vee I_2 & \textit{if } \varDelta(x) = \ \text{A} \\ (x \vee I_1) \wedge (\overline{x} \vee I_2) & \textit{if } \varDelta(x) = \text{AB} \\ I_1 \wedge I_2 & \textit{if } \varDelta(x) = \ \text{B} \end{array}$$

The system above distinguishes between variables appearing only in A, variables appearing in A and B and variables appearing only in B. McMillan's system, defined below, has a different translation function and ERes rule.

Definition 4 (McMillan's System [19]). *McMillan's interpolation system* $\mathsf{Int}_M = \langle T_M, \mathsf{MRes}\rangle$ *is defined below with* $\varDelta$ *as in Definition 3.*

$$T_M(A, B) \overset{\text{def}}{=} \{\langle C, \varDelta, C|_B\rangle | C \in A\} \cup \{\langle C, \varDelta, \mathsf{T}\rangle | C \in B\}$$

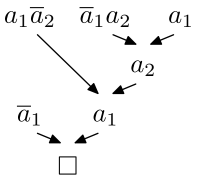
$$\frac{\langle C \vee x, \varDelta, I_1\rangle \qquad \langle D \vee \overline{x}, \varDelta, I_2\rangle}{\langle C \vee D, \varDelta, I\rangle}\ [\mathsf{MRes}]$$

$$I \overset{\text{def}}{=} \begin{array}{ll} I_1 \vee I_2 & \textit{if } \varDelta(x) = \ \text{A} \\ I_1 \wedge I_2 & \textit{if } \varDelta(x) = \text{AB} \\ I_1 \wedge I_2 & \textit{if } \varDelta(x) = \ \text{B} \end{array}$$

Note that the AB and B cases above are identical. Example 1 below shows that the two systems produce different interpolants and that different interpolants can be obtained by interchanging A and B. Example 2 shows that there are interpolants not obtained in either system.

Example 1. Let A be $(a_1 \vee \overline{a}_2) \wedge (\overline{a}_1 \vee \overline{a}_3) \wedge a_2$ and B be $(\overline{a}_2 \vee a_3) \wedge (a_2 \vee a_4) \wedge \overline{a}_4$. The e-clauses in McMillan's system are shown on the left of Figure 1 and those in the other system are on the right. The partial interpolants in both systems are shown in square brackets. The interpolants are different. The interpolant for $\langle B, A\rangle$ in McMillan's system is $a_2 \wedge a_3$. By negating it, we obtain $\overline{a}_2 \vee \overline{a}_3$, which is also an interpolant for $\langle A, B\rangle$ but is not the interpolant obtained from McMillan's system. In contrast, the interpolant for $\langle B, A\rangle$ in the HKP system is a_3, which when negated yields the same interpolant as before. $\lhd$

Example 2. Let A be the formula $\overline{a}_1 \wedge (a_1 \vee \overline{a}_2)$ and B be the formula $(\overline{a}_1 \vee a_2) \wedge a_1$. A refutation for $A \wedge B$ is shown alongside. The interpolant obtained in both systems is $\overline{a}_1 \wedge \overline{a}_2$. The interpolant for $\langle B, A\rangle$ obtained from Int_{HKP} is $a_1 \vee a_2$ and that obtained from Int_M is $a_1 \wedge a_2$. By negating these, we get the additional interpolant $\overline{a}_1 \vee \overline{a}_2$. The pair $\langle A, B\rangle$ has two more interpolants, namely $\overline{a}_1$ and $\overline{a}_2$. These interpolants can be obtained with Int_{HKP} and Int_M from different proofs. ◁



A third, *parametrised* interpolation system that generalises the other two systems was defined in [10]. Unlike Int_{HKP} and Int_M, this system manipulates distinction functions. A parameter to this system associates a distinction function with each clause in a pair $\langle A, B\rangle$. Formally, a *parameter* is a function $\mathcal{D} : \mathbb{C} \to \mathbb{D}$. For simplicity, we write $\mathcal{D}(C)(t)$ for $\mathcal{D}(C)(\mathrm{var}(t))$, where C is a clause and $t \in C$. The resolution of two distinction functions $\Delta_1, \Delta_2 \in \mathbb{D}$ with respect to a pivot x is the distinction function Δ, denoted $DRes(x, \Delta_1, \Delta_2)$, defined as follows: for $y \in \mathrm{Prop}$, $\Delta(y) \overset{\text{def}}{=} \emptyset$ if $y = x$ and $\Delta(y) \overset{\text{def}}{=} \Delta_1(y) \cup \Delta_2(y)$, if $y \neq x$. The parametrised interpolation system is defined below.

Definition 5 (Parametrised Interpolation System [10]). *Let $\mathcal{D}$ be a parameter. The interpolation system $\mathsf{Int}_{\mathcal{D}} \overset{\text{def}}{=} \langle T_{\mathcal{D}}, \mathsf{PRes}\rangle$ is defined below.*

$$T_{\mathcal{D}}(A, B) \overset{\text{def}}{=} \{\langle C, \mathcal{D}(C), I\rangle | C \in A \cup B\},$$

where I is defined below.

For $C \in A$	*For $C \in B$*
$I \overset{\text{def}}{=} \{t \in C \vert \mathcal{D}(C)(t) = \mathrm{B}\}$	$I \overset{\text{def}}{=} \neg\{t \in C \vert \mathcal{D}(C)(t) = \mathrm{A}\}$

$$\frac{\langle C \vee x, \Delta_1, I_1\rangle \qquad \langle D \vee \overline{x}, \Delta_2, I_2\rangle}{\langle C \vee D, DRes(x, \Delta_1, \Delta_2), I\rangle} \ [\mathsf{PRes}]$$

The partial interpolant I in the e-resolvent is defined below.

$$I \overset{\text{def}}{=} \begin{array}{ll} I_1 \vee I_2 & \text{if } \Delta_1(x) \cup \Delta_2(x) = \mathrm{A} \\ (x \vee I_1) \wedge (\overline{x} \vee I_2) & \text{if } \Delta_1(x) \cup \Delta_2(x) = \mathrm{AB} \\ I_1 \wedge I_2 & \text{if } \Delta_1(x) \cup \Delta_2(x) = \mathrm{B} \end{array}$$

Example 3. Recall the CNF pair $\langle A, B\rangle$ from Example 2. Written as sets, A is $\{\{\overline{a}_1\}, \{a_1, \overline{a}_2\}\}$ and B is $\{\{\overline{a}_1, a_2\}, \{a_1\}\}$. Define two distinction functions $\Delta_{\mathrm{A}} \overset{\text{def}}{=} \{a_1 \mapsto \mathrm{A}, a_2 \mapsto \mathrm{A}\}$ and $\Delta_{\mathrm{B}} \overset{\text{def}}{=} \{a_1 \mapsto \mathrm{B}, a_2 \mapsto \mathrm{B}\}$. Three parameters are defined below (all mappings not shown go to the empty set):

- $\mathcal{D}_1(C) \overset{\text{def}}{=} \Delta_{\mathrm{A}}$ for all $C \in A \cup B$.
- $\mathcal{D}_2(C) \overset{\text{def}}{=} \Delta_{\mathrm{A}}$ for all $C \in A$ and is Δ_{B} for $C \in B$.
- $\mathcal{D}_3(C) \overset{\text{def}}{=} \Delta_{\mathrm{B}}$ for all $C \in A \cup B$.

We apply the parametrised interpolation system to the refutation in Example 2. From the systems $\mathsf{Int}_{\mathcal{D}_1}, \mathsf{Int}_{\mathcal{D}_2}$ and $\mathsf{Int}_{\mathcal{D}_3}$, we obtain the interpolants $\overline{a}_1 \vee \overline{a}_2$, $\overline{a}_2$ and $\overline{a}_1 \wedge \overline{a}_2$, respectively. Recall that the interpolant $\overline{a}_2$ could not be obtained from Int_M and Int_{HKP} for the given refutation. The pair $\langle A, B\rangle$ has one more interpolant $\overline{a}_1$. We show in § 5 that this interpolant cannot be obtained from the parametrised system. $\lhd$

The set of parameters defines a set of interpolation systems. However, not all of these interpolation systems are correct. An interpolant I for $\langle A, B\rangle$ must satisfy that $\mathrm{Var}(I) \subseteq V_{\langle A,B\rangle}$. Specifically, if $x \notin V_{\langle A,B\rangle}$, x must not be added to the interpolant by $T_{\mathcal{D}}$ or the PRes rule. Observe that if for every clause $C \in A$ and literal $t \in C$ with $\mathrm{var}(t) \in V_A$, it holds that $\mathcal{D}(C)(t) = \textsc{a}$, then t will not appear in the interpolant. The same applies for $C \in B$ and $\mathrm{var}(t) \in V_B$. *Locality preserving parameters* make this intuition precise and yield correct interpolation systems. Let $\Lambda_{\langle A,B\rangle}$ be the set of locality preserving parameters for $\langle A, B\rangle$.

Definition 6 (Locality [10]). *A parameter $\mathcal{D}$ is* locality preserving *for a CNF pair $\langle A, B\rangle$ if it satisfies the following conditions.*

- *For all $C \in A \cup B$ and $x \in \mathrm{Var}(C)$, $\mathcal{D}(C)(x) \neq \emptyset$.*
- *For any $C \in \mathbb{C}$ and $x \in V_A$, $\mathcal{D}(C)(x) \subseteq \textsc{a}$.*
- *For any $C \in \mathbb{C}$ and $x \in V_B$, $\mathcal{D}(C)(x) \subseteq \textsc{b}$.*

Theorem 2 ([10]). *Let $\mathcal{D}$ be locality preserving for a CNF pair $\langle A, B\rangle$. If $\Box$ is derived from $\langle A, B\rangle$ by resolution and $E_\Box$ is the corresponding e-clause derived with $\mathsf{Int}_{\mathcal{D}}$, then $int(E_\Box)$ is an interpolant for $\langle A, B\rangle$.*

The theorem is proved by showing that for every clause C derived by resolution, the corresponding e-clause $E = \langle C, \Delta, I\rangle$ satisfies the following conditions:

- $A \wedge \neg\{t \in C | \{\textsc{a}\} \subseteq \Delta(\mathrm{var}(t))\} \Rightarrow I$
- $B \wedge \neg\{t \in C | \{\textsc{b}\} \subseteq \Delta(\mathrm{var}(t))\} \Rightarrow \neg I$
- $\mathrm{Var}(I) \subseteq \mathrm{Var}(A) \cap \mathrm{Var}(B)$.

4 Interpolation Systems and Abstract Interpretation

In this section, the parametrised interpolation system is related to the other systems and the resolution proof system by abstract interpretation.

Lattices. A *lattice*, $\langle S, \sqsubseteq, \sqcup, \sqcap\rangle$ (abbreviated to $\langle S, \sqsubseteq\rangle$), is a set S equipped with a partial order $\sqsubseteq$ and two binary operators; a least upper bound, $\sqcup$, called the *join*, and a greatest lower bound, $\sqcap$, called the *meet*. A lattice is complete if for every $X \subseteq S$, the join $\bigsqcup X$ and meet $\bigsqcap X$ are defined and exist in S. A function $F : S \to S$ is *monotone* if for any $x, y \in S$, $x \sqsubseteq y$ implies that $F(x) \sqsubseteq F(y)$. It follows from the Knaster-Tarski theorem that a monotone function on a complete lattice has unique least fixed point, denoted $\mu x.F(x)$.

Consider a set P. A *powerset lattice* is the complete lattice $\langle \wp(P), \subseteq, \cup, \cap\rangle$. Given the set $P \to S$, where S is the lattice above, the structure of S can be *lifted pointwise* to obtain the lattice $\langle P \to S, \dot{\sqsubseteq}, \dot{\sqcup}, \dot{\sqcap}\rangle$ defined below.

- For $f, g \in P \to S$, $f \dot{\sqsubseteq} g$ iff for all $x \in S$, $f(x) \sqsubseteq g(x)$.
- For $f, g \in P \to S$, $f \dot{\sqcup} g$ is the function that maps $x \in S$ to $f(x) \sqcup g(x)$. The pointwise meet operation is similarly defined.

Consult [9] for more details on lattice theory.

Abstract Interpretation. Abstract interpretation is a framework for reasoning about approximation. Only limited aspects of the framework required for the paper are covered here. See [3, 4] for an in-depth treatment.

Elements in one lattice, $\langle C, \sqsubseteq_C \rangle$ called the *concrete domain*, are approximated by elements in another $\langle A, \sqsubseteq_A \rangle$, called the *abstract domain*. The notion of approximation is formalised by an *abstraction function* $\alpha : C \to A$ and a *concretisation function* $\gamma : A \to C$ which form a *Galois connection*. The functions satisfy that for all $c \in C, a \in A$, $c \sqsubseteq_C \gamma(\alpha(c))$ and $\alpha(\gamma(a)) \sqsubseteq_A a$. If in addition $\alpha \circ \gamma$ is the identity map on A, the pair is called a *Galois insertion*. A monotone function $F : C \to C$ is approximated in A by the function $F^A : A \to A$, defined as $(\alpha \circ F \circ \gamma)$ and called the *best approximation*. The structure $\langle C, \sqsubseteq_C, F \rangle$ is the *concrete interpretation* and $\langle A, \sqsubseteq_A, F^A \rangle$ is the *abstract interpretation*. In general, the concrete and abstract interpretations may involve several functions.

The approximation F^A is *sound*, meaning that for any $c \in C$ and $a \in A$, $F(\gamma(a)) \sqsubseteq_C \gamma(F^A(a))$ and $\alpha(F(c)) \sqsubseteq_A F^A(\alpha(c))$. Soundness further implies fixed point soundness. That is, $\mu X.F(X) \sqsubseteq_C \gamma(\mu Y.F^A(Y))$ and $\alpha(\mu X.F(X)) \sqsubseteq_A \mu Y.F^A(Y)$. Thus, to compute sound approximations of concrete fixed points it suffices to compute abstract fixed points. The approximation is *complete* if $\alpha(F(c)) = F^A(\alpha(C)$. An abstract interpretation is not necessarily complete [12].

Domains connected by Galois insertions can be formalised in several other ways, in particular by closure operators [5]. An *upper closure operator* is a function $\rho : C \to C$ that is (a) *extensive*: $c \sqsubseteq_C \rho(c)$, (b) *idempotent*: $\rho(c) = \rho(\rho(c))$, and (c) *monotone*: if $c_1 \sqsubseteq_C c_2$, then $\rho(c_1) \sqsubseteq_C \rho(c_2)$. To show that an operator on a lattice defines an abstraction, it suffices to show that it is a closure operator. Closure operators are convenient because one can deal with abstractions without introducing two different lattices. Both Galois insertions and closure operators are used in this paper, as per convenience.

4.1 The Concrete Domain of Parameters

We introduce the lattice of parameters and show that locality preserving parameters are closed under certain operations on this lattice. Recall from § 3 that $\mathcal{S}$ is the powerset lattice $\langle \wp(\{\text{A}, \text{B}\}), \subseteq, \cup, \cap \rangle$. Further, define the *dual* of an element of $\mathcal{S}$ as follows: $\widehat{\text{A}} \overset{\text{def}}{=} \text{B}$, $\widehat{\text{B}} \overset{\text{def}}{=} \text{A}$, $\widehat{\text{AB}} \overset{\text{def}}{=} \text{AB}$ and $\widehat{\emptyset} \overset{\text{def}}{=} \emptyset$. That is, the dual of A is B and vice versa, but AB and $\emptyset$ are self-duals. The term *dual* is due to Huang [14] who defined the dual of Int_{HKP}. The lattice of distinction functions, $\langle \mathbb{D}, \sqsubseteq^{\mathbb{D}}, \sqcup^{\mathbb{D}}, \sqcap^{\mathbb{D}} \rangle$, where $\mathbb{D} = \text{Prop} \to \mathcal{S}$, is derived from $\mathcal{S}$ by pointwise lifting. The lattice of parameters, $\langle \mathbb{C} \to \mathbb{D}, \sqsubseteq, \sqcup, \sqcap \rangle$, is derived from $\mathbb{D}$, also by pointwise lifting. The dual of a distinction function and a parameter are similarly defined by pointwise lifting. In addition, define the function $\delta_{\langle A,B\rangle}$ that maps a parameter $\mathcal{D}$ to one that agrees with $\mathcal{D}$ on $x \in V_A \cup V_B$ but maps all other variables to their duals. Formally, $\delta_{\langle A,B\rangle}(\mathcal{D}) \overset{\text{def}}{=} \mathcal{D}'$, where for $C \in \mathbb{C}$ and $x \in \text{Prop}$, $\mathcal{D}'(C)(x)$ is $\mathcal{D}(C)(x)$ if $x \in V_A \cup V_B$ and is $\widehat{\mathcal{D}}(C)(x)$ if $x \in V_{\langle A,B\rangle}$.

Locality preserving parameters define correct interpolation systems, so operations on parameters that preserve locality are of particular interest. Such operations are illustrated in Example 4 and formally identified in Lemma 1.

Example 4. Consider again the CNF pair $\langle A, B\rangle$ in Example 1, where $A = \{\{a_1, \overline{a}_2\}, \{\overline{a}_1, \overline{a}_3\}, \{a_2\}\}$ and $B = \{\{\overline{a}_2, \overline{a}_3\}, \{a_2, a_4\}, \{\overline{a}_4\}\}$. Define the $\mathcal{D}_4, \mathcal{D}_5$ and $\mathcal{D}_6$ as below. Let $C \in \mathbb{C}$ be a clause.

- $\mathcal{D}_4(C)(x)$ is $\textsc{a}$ for $x \in V_A$, and is $\textsc{b}$ for $x \notin V_A$.
- $\mathcal{D}_5(C)(x)$ is $\textsc{a}$ for $x \notin V_B$, and is $\textsc{b}$ for $x \in V_B$.
- $\mathcal{D}_6(C)(x)$ is $\textsc{a}$ for $x \in V_A$, is $\textsc{ab}$ for $x \in V_{\langle A,B\rangle}$, and is $\textsc{b}$ for $x \in V_B$.

These parameters are locality preserving for $\langle A, B\rangle$ and that their duals are locality preserving for $\langle B, A\rangle$. Further, we have that $\delta_{\langle A,B\rangle}(\mathcal{D}_4) = \mathcal{D}_5$ and $\mathcal{D}_4 \sqcup \mathcal{D}_5 = \mathcal{D}_6$, so $\delta_{\langle A,B\rangle}$ and $\sqcup$ preserve locality. In contrast, $\mathcal{D}_4 \sqcap \mathcal{D}_5$ is not locality preserving for $\langle A, B\rangle$. $\lhd$

Lemma 1. *Let $\langle A, B\rangle$ be a CNF pair.*

1. *If $\mathcal{D}_1$ and $\mathcal{D}_2$ are locality preserving for $\langle A, B\rangle$, then so is $\mathcal{D}_1 \sqcup \mathcal{D}_2$.*
2. *If $\mathcal{D}$ is locality preserving for $\langle A, B\rangle$, then $\widehat{\mathcal{D}}$ is locality preserving for $\langle B, A\rangle$. Further, if C is derived by resolution and E and F are the corresponding clauses in $\mathsf{Int}_{\mathcal{D}}$ and $\mathsf{Int}_{\widehat{\mathcal{D}}}$ respectively, then $int(E) = \neg int(F)$.*
3. *If $\mathcal{D}$ is locality preserving for $\langle A, B\rangle$, so is $\delta_{\langle A,B\rangle}(\mathcal{D})$.*

Proof. (1) Consider each condition in Definition 6. Observe that $\mathcal{D}_1 \sqcup \mathcal{D}_2$ is the pointwise join of the two parameters. It follows that for any $C \in \mathbb{C}$ and $x \in \text{Var}(C)$, if $\mathcal{D}_1(C)(t) \neq \emptyset$ and $\mathcal{D}_2(C)(t) \neq \emptyset$, then $(\mathcal{D}_1 \sqcup \mathcal{D}_2)(C)(t) \neq \emptyset$. The same argument applies for the other two locality conditions.
(2) The sets $\text{Var}(A) \setminus \text{Var}(B)$ and $\text{Var}(B) \setminus \text{Var}(A)$ are identical in both $\langle A, B\rangle$ and $\langle B, A\rangle$. To preserve locality, any $x \in \text{Var}(A) \setminus \text{Var}(B)$ must be labelled $\textsc{b}$ by $\widehat{\mathcal{D}}$. As $\mathcal{D}$ is locality preserving, these variables are labelled $\textsc{a}$ and by the definition of $\widehat{\mathcal{D}}$, will be labelled $\textsc{b}$. A symmetric argument applies for $x \in \text{Var}(B) \setminus \text{Var}(A)$

The second property is shown by structural induction.
Base case. Consider $C \in A \cup B$, and the corresponding e-clauses $E \in T_{\mathcal{D}}(A, B)$ and $F \in T_{\widehat{\mathcal{D}}}(B, A)$. For any $t \in C$, if $\mathcal{D}(C)(x) = \textsc{a}$, then $\widehat{\mathcal{D}}(C)(x) = \textsc{b}$. It follows from the definition of $T_{\mathcal{D}}$ and $T_{\widehat{\mathcal{D}}}$ that $int(E) = \neg int(F)$. Observe in addition that $df(E) = \widehat{df(F)}$.
Induction step. For a derived clause $C = Res(x, C_1, C_2)$ and consider the corresponding e-clauses $E = ERes(x, E_1, E_2)$ and $F = ERes(x, F_1, F_2)$ derived in $\mathsf{Int}_{\mathcal{D}}$ and $\mathsf{Int}_{\widehat{\mathcal{D}}}$, respectively. For the induction hypothesis, assume that $int(E_1) = \neg int(F_1)$ and $df(E_1) = \widehat{df(F_1)}$ and likewise for E_2 and F_2. For the induction

step, consider the PRes rule in Definition 5. There are three cases for defining $int(E)$. If case A applies in $\mathsf{Int}_{\mathcal{D}}$, then, by the induction hypothesis, case B applies for $\mathsf{Int}_{\widehat{\mathcal{D}}}$. That is, $int(E) = I_1 \vee I_2$ and $int(F) = \neg I_1 \wedge \neg I_2$, so $int(E) = \neg(int(F))$ as required. The other cases are similar.
(3) Holds as $\mathcal{D}(C)(x) = (\delta_{\langle A,B\rangle}(\mathcal{D}))(C)(x)$ for $C \in A \cup B$ and $x \in V_A \cup V_B$. ■

4.2 Abstract Domains of Parameters

Algorithms derived from $\mathsf{Int}_{HKP}, \mathsf{Int}_M$ and the parametrised system have a running time that is linear in proof size, however Int_{HKP} and Int_M are more space efficient because they do not modify the distinction function. Intuitively, an interpolation system is space efficient if the value of the distinction function at a pivot variable does not change in a proof. Formally, a parameter $\mathcal{D}$ is *derivation invariant* with respect to $\langle A, B\rangle$ if for any e-clause E derived from $\langle A, B\rangle$ in $\mathsf{Int}_{\mathcal{D}}$ and any $C \in A \cup B$, if $x \in \mathrm{Var}(cl(E)) \cap \mathrm{Var}(C)$, then $df(E)(x) = \mathcal{D}(C)(x)$.

Example 5. Consider the pair $\langle A, B\rangle$ and the parameters $\mathcal{D}_1$ and $\mathcal{D}_3$ in Example 2. For any clause C derived from $\langle A, B\rangle$ and corresponding e-clause E in the example, $df(E)(x)$ is the same as $\mathcal{D}(C')(x)$, where $C' \in \langle A, B\rangle$. The parameters in Example 4 are also derivation invariant. In contrast, the parameter $\mathcal{D}_2$ in Example 2 is not derivation invariant because the value of the distinction function at a_2 changes in the proof. ◁

We identify a family of abstractions that give rise to derivation invariant parameters. These abstractions are defined over partitions of Prop. A *partition* π of a set S is a set of disjoint subsets of S, called *blocks*, that are pairwise disjoint and whose disjoint union is S. Let $[x]_\pi$ denote the block containing $x \in S$. A partition π is *coarser than* a partition π', denoted $\pi \preceq \pi'$, if for every block $\beta \in \pi$, there is a block $\beta' \in \pi'$ such that $\beta \subseteq \beta'$. It is known that the set of partitions forms a complete lattice. Let $\langle \mathrm{Part(Prop)}, \preceq, \sqcup, \sqcap\rangle$ be the lattice of partitions of Prop. For a CNF pair $\langle A, B\rangle$, define the partition $\pi^{\langle A,B\rangle} \overset{\text{def}}{=} \{\{x|x \in V_A\}, \{x|x \in V_{\langle A,B\rangle}\}, \{x|x \in V_B\}, \{x|x \notin \mathrm{Var}(A) \cup \mathrm{Var}(B)\}\}$. Given a partition $\pi \in \mathrm{Part(Prop)}$ we define a function Υ_π that maps a parameter to another one, assigning the same symbol in $\mathcal{S}$ to variables in the same block.

$$\Upsilon_\pi(\mathcal{D}) \overset{\text{def}}{=} \mathcal{D}' \text{ where } \mathcal{D}'(C)(x) \overset{\text{def}}{=} \bigcup_{C' \in \mathbb{C}} \bigcup_{y \in [x]_\pi} \mathcal{D}(C')(y) \text{ for } C \in \mathbb{C} \text{ and } x \in \mathrm{Prop}.$$

A parameter $\mathcal{D}$ is *partitioning* if $\Upsilon_\pi(\mathcal{D}) = \mathcal{D}$ for some $\pi \in \mathrm{Part(Prop)}$. In Theorem 3, we show that each function Υ_π defines an abstract domain of parameters and relate such parameters to derivation invariance and locality preservation.

Example 6. Consider the CNF pair $\langle A, B\rangle$ in Example 4 and the partitions $\pi_A = \{\{x|x \in V_A\}, \{x|x \notin V_A\}\}$, $\pi_B = \{\{x|x \in V_B\}, \{x|x \notin V_B\}\}$, and $\pi^{\langle A,B\rangle}$. Assume that $\mathrm{Var}(A \cup B) = \mathrm{Prop}$. The parameters $\mathcal{D}_4, \mathcal{D}_5$ and $\mathcal{D}_6$ are partitioning, as witnessed by the partitions π_A, π_B and $\pi^{\langle A,B\rangle}$ respectively.

Consider the CNF pair $\langle A, B\rangle$, the parameters $\mathcal{D}_1, \mathcal{D}_2$ and $\mathcal{D}_3$ in Example 3 and the partition $\pi = \{\text{Prop}\}$. Observe that $V_A = V_B = \emptyset$, so $\mathcal{D}_1$ and $\mathcal{D}_3$ are partitioning with respect to π. However, $\mathcal{D}_2$ is not partitioning. $\lhd$

Theorem 3. *1. The function Υ_π is a closure operator.*
2. A partitioning parameter is derivation invariant.
3. If $\mathcal{D}_1$ and $\mathcal{D}_2$ are partitioning, so are $\mathcal{D}_1 \sqcup \mathcal{D}_2$, $\widehat{\mathcal{D}_1}$ and $\delta_{\langle A,B\rangle}(\mathcal{D}_1)$
4. If $\mathcal{D}$ is locality preserving and $\pi \preceq \pi^{\langle A,B\rangle}$, then $\Upsilon_\pi(\mathcal{D})$ is locality preserving.
5. The coarsest π for which $\Upsilon_\pi(\Lambda_{\langle A,B\rangle}) \subseteq \Lambda_{\langle A,B\rangle}$, for any $\langle A, B\rangle$, is $\pi = \pi^{\langle A,B\rangle}$.

Proof. (1) We show that Υ_π is a closure operator. The function is extensive because for all $C \in \mathbb{C}$ and $x \in \text{Prop}$, $\mathcal{D}(C)(x) \subseteq \Upsilon_\pi(\mathcal{D})(C)(x)$. For any $C \in \mathbb{C}$ and $y \in [x]_\pi$, $\Upsilon_\pi(\mathcal{D})(C)(x) = \Upsilon_\pi(\mathcal{D})(C)(y)$, so the function is idempotent. If $\mathcal{D}_1 \sqsubseteq \mathcal{D}_2$, then for all $C \in \mathbb{C}$ and $x \in \text{Prop}$, $\mathcal{D}_1(C)(x) \subseteq \mathcal{D}_2(C)(x)$. The values $\Upsilon_\pi(\mathcal{D}_1)(C)(x)$ and $\Upsilon_\pi(\mathcal{D}_2)(C)(x)$ are defined as the union over a set of variables of $\mathcal{D}_1$ and $\mathcal{D}_2$ respectively. Monotonicity follows because union is monotone.
(2) Let E be an e-clause derived with $\mathsf{Int}_{\mathcal{D}}$ from $\langle A, B\rangle$. We show that $\mathcal{D}$ is derivation invariant by induction on the structure of the derivation.
Base Case. If $E \in T_{\mathcal{D}}(A, B)$, as $\mathcal{D}$ is partitioning, $\mathcal{D}(C)(x) = \mathit{df}(E)(x)$ for any clause $C \in \mathbb{C}$ and variable $x \in \text{Prop}$.
Induction Step. Consider $E = \mathit{ERes}(x, E_1, E_2)$ for e-clauses E_1 and E_2. For the induction hypothesis, assume that for any $C \in A \cup B$ and $x \in \text{Var}(\mathit{cl}(E_1)) \cap \text{Var}(C)$, $\mathit{df}(E_1)(x) = \mathcal{D}(C)(x)$ and the same for E_2. Consider $C \in A \cup B$ and $x \in \text{Var}(\mathit{cl}(E)) \cap \text{Var}(C)$. Now, x must be in $\text{Var}(\mathit{cl}(E_1))$ only, $\text{Var}(\mathit{cl}(E_2))$ only or both. If $x \in \text{Var}(\mathit{cl}(E_1))$ only, $\mathit{df}(E)(x) = \mathit{df}(E_1)(x)$ and by the induction hypothesis, $\mathit{df}(E)(x) = \mathcal{D}(C)(x)$. The remaining cases are similar.
(3) Consider $\mathcal{D}_1$ and $\mathcal{D}_2$ which are partitioning. That is, there exist π_1 and π_2 such that $\Upsilon_{\pi_1}(\mathcal{D}_1) = \mathcal{D}_1$ and $\Upsilon_{\pi_2}(\mathcal{D}_2) = \mathcal{D}_2$. Let $\mathcal{D} = \mathcal{D}_1 \sqcup \mathcal{D}_2$ and $\pi = \pi_1 \sqcap \pi_2$. and $\mathcal{D} = \mathcal{D}_1 \sqcup \mathcal{D}_2$. Because $\mathcal{D}_1$ and $\mathcal{D}_2$ are partitioning, it follows that for all x and $y \in [x]_\pi$, $\mathcal{D}(C)(x) = \mathcal{D}(C)(y)$. Thus, $\Upsilon_\pi(\mathcal{D}) = \mathcal{D}$ and $\mathcal{D}$ is partitioning. The other cases hold because the dual and $\delta_{\langle A,B\rangle}$ are defined pointwise on variables, so the partition for $\mathcal{D}_1$ is the partition for $\widehat{\mathcal{D}_1}$ and $\delta_{\langle A,B\rangle}(\mathcal{D})$.
(4) If $\pi \preceq \pi^{\langle A,B\rangle}$, for any $x \in V_A$, if $y \in [x]_\pi$, then $y \in V_A$. For a locality preserving $\mathcal{D}$ and $C \in A \cup B$, it holds that $\mathcal{D}(C)(y) \subseteq \textsc{a}$. Hence, $\bigcup_{C \in \mathbb{C}} \bigcup_{y \in [x]_\pi} \mathcal{D}(C)(y) \subseteq \textsc{a}$. The same applies for $x \in V_B$, so $\Upsilon_\pi(\mathcal{D})$ is locality preserving.
(5) It follows from the previous part that $\Upsilon_{\pi^{\langle A,B\rangle}}(\Lambda_{\langle A,B\rangle}) \subseteq \Lambda_{\langle A,B\rangle}$. It suffices to show that there is no $\pi^{\langle A,B\rangle} \prec \pi$ such that $\Upsilon_\pi(\Lambda_{\langle A,B\rangle}) \subseteq \Lambda_{\langle A,B\rangle}$ for all $\langle A, B\rangle$. We prove it by contradiction. It suffices to find a pair $\langle A, B\rangle$ and $\mathcal{D} \in \Lambda_{\langle A,B\rangle}$ such that $\Upsilon_\pi(\mathcal{D}) \notin \Lambda_{\langle A,B\rangle}$. Consider $\langle A, B\rangle$ with $V_A, V_{\langle A,B\rangle}$ and V_B being non-empty. Let $\mathcal{D}$ map $x \in V_A$ to $\textsc{a}$, $x \in V_B$ to $\textsc{b}$ and $x \in V_{\langle A,B\rangle}$ to $\textsc{ab}$. Consider variables $x \in V_A, y \in V_{\langle A,B\rangle}$ and $z \in V_B$. As $\pi^{\langle A,B\rangle} \prec \pi$, either $[x]_\pi = [y]_\pi$, or $[y]_\pi = [z]_\pi$, or $[x]_\pi = [z]_\pi$. If $[x]_\pi = [y]_\pi$, then $\mathcal{D}(C)(x) = \textsc{ab}$, violating the condition $\mathcal{D}(C)(x) \subseteq \textsc{a}$ in Definition 6. Thus, $\Upsilon_\pi(\Lambda_{\langle A,B\rangle}) \not\subseteq \Lambda_{\langle A,B\rangle}$. The other two cases are similar, leading to a contradiction as required. ∎

We highlight that part 5 of Theorem 3 applies to all $\langle A, B\rangle$ and all parameters $\mathcal{D} \in \Lambda_{\langle A,B\rangle}$. For a specific parameter $\mathcal{D} \in \Lambda_{\langle A,B\rangle}$ and a specific pair $\langle A, B\rangle$, there may exist $\pi^{\langle A,B\rangle} \prec \pi$ such that $\Upsilon_\pi(\mathcal{D})$ is locality preserving.

4.3 Existing Systems as Abstractions

The setting of the previous section is now applied to study existing systems. We define two parameters that were shown in [10] to correspond to McMillan's system and the HKP system. Let $\langle A, B\rangle$ be a CNF pair. Define the value of the parameters $\mathcal{D}_M$ and $\mathcal{D}_{HKP}$ for $C \in \mathbb{C}$ and $x \in \text{Prop}$ as below.

- $\mathcal{D}_M(C)(x)$ is A if $x \in V_A$ and is B otherwise.
- $\mathcal{D}_{HKP}(C)(x)$ is A if $x \in V_A$ B if $x \in V_B$ and is AB for $x \in V_{\langle A,B\rangle}$.

Lemma 2 shows that the parameters above are two of three that exist in the coarsest partitioning abstraction defined by $\pi^{\langle A,B\rangle}$. The third system, $\delta_{\langle A,B\rangle}(\mathcal{D}_M)$, was also identified in [10] but the connections presented here were not.

Lemma 2. *Let $\langle A, B\rangle$ be a CNF pair. The image of $\Lambda_{\langle A,B\rangle}$ under $\Upsilon_{\pi^{\langle A,B\rangle}}$ is $\{\mathcal{D}_M, \mathcal{D}_{HKP}, \delta_{\langle A,B\rangle}(\mathcal{D}_M)\}$.*

Proof. There are two steps. The first step is to show that each parameter in the lemma is a fixed point of $\Upsilon_{\pi^{\langle A,B\rangle}}$. We skip this step. The second is to show that no other such fixed points exist. As only elements of $\Lambda_{\langle A,B\rangle}$ are considered, assume that $\mathcal{D}$ is locality preserving. By definition of the closure operator we have that $\Upsilon_{\pi^{\langle A,B\rangle}}(\mathcal{D}) = \mathcal{D}$ only if for any $C_1, C_2 \in \mathbb{C}$ and $x, y \in V_{\langle A,B\rangle}$, $\mathcal{D}(C_1)(x) = \mathcal{D}(C_2)(y)$. It follows that for all C and $x \in V_{\langle A,B\rangle}$, $\mathcal{D}(C)(x)$ must be either A, AB or B. Thus, the only three possible parameters are the ones above. ■

The parameter $\mathcal{D}_{HKP}$ has several properties. It is the greatest locality preserving parameter with respect to $\sqsubseteq$, is symmetric in the sense that $\delta_{\langle A,B\rangle}(\mathcal{D}_{HKP}) = \mathcal{D}_{HKP}$ and can be derived from McMillan's system. These properties, summarised below, may explain why Int_{HKP} has been repeatedly discovered.

$$\boxed{\mathcal{D}_{HKP} = \bigsqcup_{\mathcal{D} \in \Lambda_{\langle A,B\rangle}} \mathcal{D} \quad \text{and} \quad \mathcal{D}_M \sqcup \delta_{\langle A,B\rangle}(\mathcal{D}_M) = \mathcal{D}_{HKP}}$$

4.4 The Domains of E-Clauses and Clauses

We remarked earlier that an interpolation system is an extension of resolution. This intuition is now made precise using the method in [7]. E-clauses constitute a concrete domain and interpolation systems define concrete interpretations. We show that sets of clauses form an abstract domain and that the resolution rule defines a complete abstract interpretation of an interpolation system.

Recall that $\mathbb{E}$ is the set of e-clauses and that for $E = \langle C, \Delta, I\rangle$, $cl(E) = C$. The powerset of e-clauses forms the concrete domain $\langle \wp(\mathbb{E}), \subseteq, \cup, \cap\rangle$. A parameter $\mathcal{D}$

defines an interpolation system $\mathsf{Int}_{\mathcal{D}} = \langle T_{\mathcal{D}}, \mathsf{PRes}\rangle$, which gives rise to a concrete interpretation consisting of two functions. The translation function $T_{\mathcal{D}} : \wp(\mathbb{C}) \times \wp(\mathbb{C}) \to \wp(\mathbb{E})$ and a function $\mathit{PRes} : \wp(\mathbb{E}) \to \wp(\mathbb{E})$ encoding the effect of the PRes rule. The function PRes is defined in a sequence of steps.

- $\mathit{PRes} : \mathrm{Prop} \times \mathbb{E} \times \mathbb{E} \to \mathbb{E}$ is defined as follows. If $E_1, E_2 \in \mathbb{E}$ with $\mathit{cl}(E_1) = x \vee C$ and $\mathit{cl}(E_2) = D \vee \overline{x}$, then $\mathit{PRes}(x, E_1, E_2)$ is given by the PRes rule in Definition 5. $\mathit{PRes}(x, E_1, E_2)$ is defined as $\langle \emptyset, \emptyset, \mathsf{F}\rangle$ otherwise.
- Let $\mathit{PRes} : \mathbb{E} \times \mathbb{E} \to \mathbb{E}$ be $\mathit{PRes}(E_1, E_2) \stackrel{\text{def}}{=} \{\mathit{PRes}(x, E_1, E_2) | x \in \mathrm{Prop}\}$.
- Finally, $\mathit{PRes} : \wp(\mathbb{E}) \to \wp(\mathbb{E})$ maps $X \in \wp(\mathbb{E})$ to $\bigcup_{E_1, E_2 \in X} \mathit{PRes}(E_1, E_2)$.

The concrete semantic object of interest is the set of e-clauses that can be derived in an interpolation system $\mathsf{Int}_{\mathcal{D}}$ and the interpolants obtained from these e-clauses. These sets are defined below.

$$\mathcal{E}_{\mathcal{D}} \stackrel{\text{def}}{=} \mu X.(T_{\mathcal{D}}(A, B) \cup \mathit{PRes}(X)) \text{ and } \mathcal{I}_{\mathcal{D}} \stackrel{\text{def}}{=} \{\mathit{int}(E) | E \in \mathcal{E}_{\mathcal{D}} \text{ and } \mathit{cl}(E) = \Box\}\,.$$

The set $\mathcal{I}_{\mathcal{D}}$ contains all interpolants that can be derived with $\mathsf{Int}_{\mathcal{D}}$ from $\langle A, B\rangle$. Observe that each interpolation system $\mathsf{Int}_{\mathcal{D}}$ defines a different concrete interpretation and a different set of interpolants $\mathcal{I}_{\mathcal{D}}$. Note also that the definition of PRes is independent of the parameter $\mathcal{D}$. Hence, to analyse the properties of the set $\mathcal{I}_{\mathcal{D}}$, we only have to analyse $T_{\mathcal{D}}$. We exploit this observation in § 5.1.

We now relate resolution with interpolation systems. Define the domain $\langle \wp(\mathbb{C}), \subseteq, \cup, \cap\rangle$ of CNF formulae. The function Res corresponding to the resolution rule is first defined as $\mathit{Res} : \mathrm{Prop} \times \mathbb{C} \times \mathbb{C} \to \mathbb{C}$ and then lifted to a function $\mathit{Res} : \wp(\mathbb{C}) \to \wp(\mathbb{C})$, in a manner similar to PRes.

Abstraction and concretisation functions between $\langle \wp(\mathbb{E}), \subseteq\rangle$ and $\langle \wp(\mathbb{C}), \subseteq\rangle$ are defined next. Let $\alpha : \wp(\mathbb{E}) \to \wp(\mathbb{C})$ be a function that maps $X \in \wp(\mathbb{E})$ to the set of clauses $\mathit{cl}(X)$. The concretisation function $\gamma : \wp(\mathbb{C}) \to \wp(\mathbb{E})$ maps a set of clauses $Y \in \wp(\mathbb{C})$ to the set of e-clauses $\{\langle C, \Delta, I\rangle | C \in Y, \Delta \in \mathbb{D}, I \in \mathbb{B}\}$. Lemma 3 states that α and γ define a Galois insertion and that Res is the best approximation of PRes.

What do soundness and completeness mean in this setting? If $\alpha(\mathit{PRes}(X)) \subseteq \mathit{Res}(\alpha(X))$, every clause that can be derived with the inference rule PRes can also be derived with Res. However, we also want that the interpolation system can derive all clauses that can be derived by resolution. That is, as an inference rule, PRes should be as powerful as Res. In abstract interpretation terms, the function Res should be a complete abstraction of PRes.

Lemma 3. *The functions α and γ define a Galois insertion between $\wp(\mathbb{E})$ and $\wp(\mathbb{C})$. Further, $\mathit{Res} = (\alpha \circ \mathit{PRes} \circ \gamma)$, and $(\mathit{Res} \circ \alpha) = (\alpha \circ \mathit{PRes})$.*

The best approximation of $T_{\mathcal{D}}$ is union: $(\alpha \circ T_{\mathcal{D}} \circ \gamma) = \cup$. The abstract semantic object corresponding to $\mathcal{E}_{\mathcal{D}}$ is the set of clauses that can be derived by resolution from $\langle A, B\rangle$. The viewpoint presented here is summarised below.

$$\mathcal{C} \stackrel{\text{def}}{=} \mu X.((A \cup B) \cup \mathit{Res}(X)) = \alpha(\mathcal{E}_{\mathcal{D}})$$

$\langle \wp(\mathbb{C}), \subseteq, \cup, \mathit{Res}\rangle$ is a complete abstract interpretation of $\langle \wp(\mathbb{E}), \subseteq, T_{\mathcal{D}}, \mathit{PRes}\rangle$.

5 Logical Strength and Variable Elimination

Interpolation systems are used in verification tools. The performance of such a tool depends on the logical strength and size of the interpolants obtained. The influence of interpolant strength on the termination of a verification tool is discussed in [10]. Interpolant size affects the memory requirements of a verification tool. The set of variables in an interpolant gives an upper bound on its size, so we study the smallest and largest sets of variables that can occur in an interpolant. We now analyse the logical strength of and variables occurring in interpolants.

5.1 Logical Strength as a Precision Order

The subset ordering on the domain $\wp(\mathbb{E})$ is a *computational order*. Meaning, it is the order with respect to which fixed points are defined. The elements of $\wp(\mathbb{E})$ can moreover be ordered by *precision*, where the notion of precision is application dependent. Cousot and Cousot have emphasised that though the computational and precision orders often coincide, this is not necessary [6]. To understand the logical strength of interpolants, we use a precision order based on implication.

Given X and Y in $\wp(\mathbb{E})$, the set X is more precise than Y if for every interpolant in Y, there is a logically stronger interpolant in X. Formally, define the relation $\preceq_{\mathbb{E}}$ on $\wp(\mathbb{E}) \times \wp(\mathbb{E})$ as $X \preceq_{\mathbb{E}} Y$ iff for all $E_1 \in Y$ with $cl(E_1) = \Box$, there exists $E_2 \in X$ with $cl(E_2) = \Box$ and $int(E_2) \Rightarrow int(E_1)$. Let $\langle A, B \rangle$ be a CNF pair, $\mathcal{D}_1$ and $\mathcal{D}_2$ be two parameters and $\mathcal{E}_1$ and $\mathcal{E}_2$ be the sets of e-clauses derived in these two systems. The system $\mathsf{Int}_{\mathcal{D}_1}$ is *more precise* or *stronger* than $\mathsf{Int}_{\mathcal{D}_2}$ if $\mathcal{E}_1 \preceq_{\mathbb{E}} \mathcal{E}_2$. If *PRes* is monotone with respect to $\preceq_{\mathbb{E}}$, the problem of computing logically stronger interpolants can be reduced to that of ordering translation functions by precision. However, *PRes* is not monotone with respect to $\preceq_{\mathbb{E}}$ because $\preceq_{\mathbb{E}}$ does not take distinction functions into account.

We now derive an order for $\wp(\mathbb{E})$ that is stronger than $\preceq_{\mathbb{E}}$ and with respect to which *PRes* is monotone. The order from [10] is adapted to our setting. We define an order on $\mathcal{S}$ and lift it pointwise. Define the order $\preceq_{\mathcal{S}}$ on $\mathcal{S}$ as $\textsc{b} \preceq_{\mathcal{S}} \textsc{ab} \preceq_{\mathcal{S}} \textsc{a} \preceq_{\mathcal{S}} \emptyset$. The set $\mathcal{S}$ with this order forms the lattice $\langle \mathcal{S}, \preceq_{\mathcal{S}}, \max, \min \rangle$. By pointwise lifting, we obtain the lattice $\langle \mathbb{D}, \preceq_{\mathbb{D}}, \Uparrow_{\mathbb{D}}, \Downarrow_{\mathbb{D}} \rangle$. We use the symbols $\Uparrow_{\mathbb{D}}$ and $\Downarrow_{\mathbb{D}}$ to distinguish them from the computational meet and join, $\sqcup^{\mathbb{D}}$ and $\sqcap^{\mathbb{D}}$, and to emphasise the connection to logical implication.

Recall from § 2 that $C|_A$ is the restriction of C to variables in A. Define a relation $\sqsubseteq_{\mathbb{E}}$ on $\wp(\mathbb{E}) \times \wp(\mathbb{E})$ as: $X \sqsubseteq_{\mathbb{E}} Y$ if for each $E_1 \in Y$ there is an $E_2 \in X$ such that $cl(E_1) = cl(E_2)$, $df(E_1) \preceq_{\mathbb{D}} df(E_2)$ and $int(E_2) \Rightarrow int(E_1) \vee (cl(E_1)|_A \cap cl(E_1)|_B)$. Intuitively, in a strong interpolant, literals are added to the partial interpolant by the translation function whereas in a weaker interpolant, literals are added in the resolution step. The partial interpolant $int(E_1)$ in the definition of $\sqsubseteq_{\mathbb{E}}$ is weakened with $(cl(E_1)|_A \cap cl(E_1)|_B)$ to account for this difference. Nonetheless, if $X \sqsubseteq_{\mathbb{E}} Y$ and $cl(E_1) = \Box$ for $E_1 \in Y$, there exists $E_2 \in X$ such that $cl(E_2) = \Box$ and $int(E_2) \Rightarrow int(E_1)$. Thus, $X \sqsubseteq_{\mathbb{E}} Y$ implies that $X \preceq_{\mathbb{E}} Y$. Theorem 4 shows that *PRes* is monotone with respect to $\sqsubseteq_{\mathbb{E}}$. To order interpolation systems by precision, the precision order on distinction functions is lifted pointwise to parameters to obtain the lattice $\langle \mathbb{C} \to \mathbb{D}, \preceq, \Uparrow, \Downarrow \rangle$.

Example 7. Revisit the functions $\mathcal{D}_1, \mathcal{D}_2$ and $\mathcal{D}_3$ in Example 3. It holds that $\mathcal{D}_3 \preceq \mathcal{D}_2 \preceq \mathcal{D}_1$ and the corresponding interpolants imply each other. ◁

Theorem 4. *Let $\langle A, B \rangle$ be a CNF pair, and $\mathcal{D}_1$ and $\mathcal{D}_2$ be locality preserving parameters for $\langle A, B \rangle$.*

1. *If $\mathcal{D}_1 \preceq \mathcal{D}_2$, then $T_{\mathcal{D}_1}(A, B) \sqsubseteq_{\mathbb{E}} T_{\mathcal{D}_2}(A, B)$.*
2. *If $X \sqsubseteq_{\mathbb{E}} Y$ for $X, Y \in \wp(\mathbb{E})$, then $PRes(X) \sqsubseteq_{\mathbb{E}} PRes(Y)$.*
3. *The structure $\langle \Lambda_{\langle A,B \rangle}, \preceq, \Uparrow, \Downarrow \rangle$ is a complete lattice [10].*

Proof. (1) Consider $T_{\mathcal{D}_1}(A, B)$, $T_{\mathcal{D}_2}(A, B)$, and $F \in T_{\mathcal{D}_2}(A, B)$. It follows from the definition of a translation function that there exists $E \in T_{\mathcal{D}_1}(A, B)$ such that $cl(E) = cl(F)$. If $C \in A$, we further have that $int(E) \subseteq (cl(F)|_A \cap cl(F)|_B)$, and so $int(E) \Rightarrow int(F) \vee (cl(F)|_A \cap cl(F)|_B)$. If $C \in B$, then by definition, $\neg int(F) = \{t \in cl(F) | \mathcal{D}_2(cl(F))(t) = \text{A}\}$. Because $\mathcal{D}_2$ is locality preserving, $\neg int(F) \subseteq (cl(F)|_A \cap cl(F)|_B)$ and we can conclude that $\neg int(F) \subseteq \neg(int(E)) \vee (cl(F)|_A \cap cl(F)|_B)$ and so $int(E) \subseteq int(F) \vee (cl(F)|_A \cap cl(F)|_B)$.
(2) Consider $X \sqsubseteq_{\mathbb{E}} Y$ and $F \in PRes(Y)$. There exists $x \in \text{Prop}$ and $F_1, F_2 \in X$ such that $F = PRes(x, F_1, F_2)$. By the monotony hypothesis, there exist E_1 and E_2 in X such that $E_1 \sqsubseteq_{\mathbb{E}} F_1$ and $E_2 \sqsubseteq_{\mathbb{E}} F_2$. From the definition of $\sqsubseteq_{\mathbb{E}}$ we conclude that $E = PRes(x, E_1, E_2)$ satisfies that $cl(E) = cl(F)$. It remains to show that $int(E) \Rightarrow int(F) \vee (cl(E)|_A \cap cl(E)|_B)$. This can be shown by a straightforward case analysis. ■

The following corollary of Theorem 4 formally states that if $\mathcal{D}_1 \preceq \mathcal{D}_2$, then the interpolants obtained from $\mathsf{Int}_{\mathcal{D}_1}$ imply the interpolants obtained from $\mathsf{Int}_{\mathcal{D}_2}$.

Corollary 1. *If $\mathcal{D}_1$ and $\mathcal{D}_2$ are locality preserving parameters for the CNF pair $\langle A, B \rangle$, then $\mu X.(T_{\mathcal{D}_1}(A, B) \cup PRes(X)) \preceq_{\mathbb{E}} \mu X.(T_{\mathcal{D}_2}(A, B) \cup PRes(X))$.*

5.2 Variable Elimination

Any interpolant I for an unsatisfiable CNF pair $\langle A, B \rangle$ satisfies that $\text{Var}(I) \subseteq V_{\langle A,B \rangle}$. We ask what the largest and smallest possible sets V are such that $\text{Var}(I) \subseteq V$. To develop some intuition for this question, we visualise the *flow* of literals in a proof. Flow graphs have been used by Carbone to study interpolant size in the sequent calculus [2]. We only use them informally.

Example 8. The flow of literals in the refutation from Example 2 is shown in Figure 2. Dashed edges connect antecedents with resolvents and solid edges depict flows. Each literal is a vertex in the flow graph. Positive literals flow upwards and negative literals flow downwards. Observe that a_1 appears in multiple cycles connecting literals in A, literals in B and literals in A and B. In contrast, a_2 appears in only cycle which connects an A and a B literal. Recall that every interpolant constructed from this refutation contained a_2. ◁

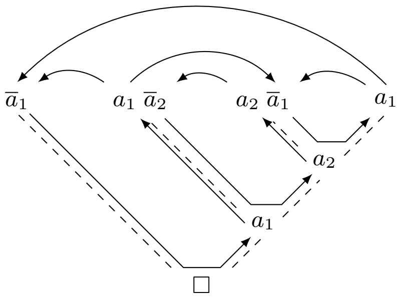


Fig. 2. A resolution proof and its logical flow graph. Dashed edges represent resolution and solid edges represent flows. Every occurrence of a literal is in a cycle.

Informally, a refutation defines a set of may and must variables. Any literal flowing from the A to the B part, like a_1 above, *may* be added to the interpolant. A literal that only flows from an A literal to a B literal, like a_2, *must* be added to the interpolant. To obtain the interpolant with the smallest set of variables, we need a parameter that adds only those literals to the interpolant that flow between A and B. We define two parameters for $\langle A, B\rangle$ as follows.

- $\mathcal{D}_{\min}(C)(x)$ is A for $C \in A$ and $x \in \text{Prop}$ and is B for $C \in B$ and $x \in \text{Prop}$.
- $\mathcal{D}_{\max} \overset{\text{def}}{=} \delta_{\langle A,B\rangle}(\mathcal{D}_{\min})$.

Observe that both these parameters are locality preserving. Lemma 4 states that the parameters above determine the smallest and largest sets of variables that occur syntactically in an interpolant.

Lemma 4. *Let $\Box$ be derived from $\langle A, B\rangle$ and $E_{\min}$ and $E_{\max}$ be the corresponding e-clauses derived in $\mathsf{Int}_{\mathcal{D}_{\min}}$ and $\mathsf{Int}_{\mathcal{D}_{\max}}$ respectively. Let E be the corresponding e-clause in $\mathsf{Int}_{\mathcal{D}}$ for a locality preserving parameter $\mathcal{D}$. It holds that $\text{Var}(\mathit{int}(E_{\min})) \subseteq \text{Var}(\mathit{int}(E)) \subseteq \text{Var}(\mathit{int}(E_{\max}))$.*

Proof. We first show that if $x \in \text{Var}(\mathit{int}(E))$, then $x \in \text{Var}(\mathit{int}(E_{\max}))$. Observe that if $x \in \text{Var}(\mathit{int}(E))$, then $x \in V_{\langle A,B\rangle}$ and either x or $\overline{x}$ must occur in some $C \in A \cup B$. Let F be the clause corresponding to C in $\mathsf{Int}_{\mathcal{D}_{\max}}$. If $C \in A$, $\mathcal{D}_{\max}(C)(x) = $ B and if $C \in B$, $\mathcal{D}_{\max}(C)(x) = $ A. In both cases, by the definition of $T_{\mathcal{D}_{\max}}$ it holds that $x \in \text{Var}(\mathit{int}(F))$.
We show that if $x \in \text{Var}(\mathit{int}(E_{\min}))$, then $x \in \text{Var}(\mathit{int}(E))$. We proceed by induction on the structure of the derivation and consider the step in which x was added to the partial interpolant. Let F be the e-clause derived by the PRes rule in $\mathsf{Int}_{\mathcal{D}_{\min}}$, given as $F = \mathit{PRes}(x, F_1, F_2)$ where F_1 and F_2 are antecedents. It must be that $\mathit{df}(F_1)(x) \cup \mathit{df}(F_2)(x) = $ AB. Further, it must be that $x \in \mathit{cl}(F_1)$ and $\overline{x} \in \mathit{cl}(F_2)$ originated in A and B respectively, or vice versa, or are derived from two literals that originated from these two parts of the formulae. Let G, G_1, G_2 be the corresponding e-clauses derived in $\mathsf{Int}_{\mathcal{D}}$. There are three possibilities for

$df(G_1)(x) \cup df(G_2)(x)$. If the value is AB, then x is added to the interpolant in this derivation step. If the value is A, then the literal that originated from B was added to the interpolant by the translation function. If the value is B, the literal originating from A was added to the interpolant by the translation function. In all cases, $x \in \text{Var}(int(G))$ as required. ■

We draw two further insights from Lemma 4. Observe that $\mathcal{D}_{\min}$ and $\mathcal{D}_{\max}$ are distinct from $\mathcal{D}_M$ and $\mathcal{D}_{HKP}$. A consequence is that McMillan's system and the HKP-system do not necessarily yield the interpolant with the smallest set of variables in an interpolant. This was demonstrated in Example 2, where the interpolants in these systems contained the variables $\{a_1, a_2\}$, but an interpolant over $\{a_2\}$ could be obtained.

A more general insight is a way to determine if specific interpolants cannot be obtained from a refutation. To revisit Example 2 (for the last time), observe that $\text{Var}(int(E_{\min})) = \{a_2\}$ and that $\text{Var}(int(E_{\max})) = \{a_1, a_2\}$. It follows that the interpolant $\overline{a}_1$ for this pair cannot be obtained by any interpolation system $\mathsf{Int}_{\mathcal{D}}$ in the family we consider.

6 Related Work

Though Craig's interpolation theorem was published in 1957 [8], the independent study of interpolation systems is relatively recent. Constructive proofs of Craig's theorem implicitly define interpolation systems. The first such proof is due to Maehara who introduced *split sequents* to capture the contribution of the A and B formulae in a sequent calculus proof [17]. Carbone generalised this construction to flow graphs to study the effect of cut-elimination on interpolant size [2].

Interpolant size was first studied by Mundici [20], Krajíček observed that lower bounds on interpolation systems for propositional proofs have implications for separating complexity classes and gave an interpolation system for resolution [16]. Pudlák published the same system simultaneously [21].

Huang gave an interpolation system for resolution and its dual [14] but his work appears to have gone unnoticed. McMillan proposed an propositional interpolation system and applied it to obtain a purely SAT-based finite-state model checker [19]. These systems were generalised in [10] and the system in that paper was studied here. Yorsh and Musuvathi [24] study interpolation for first-order theories, but also gave a new and elaborate correctness proof for the HKP-system. The invariant for proving Theorem 2 is generalises the induction hypothesis in their proof. The precision order $\sqsubseteq_{\mathbb{E}}$ is a modification of their induction hypothesis to relate interpolants by strength rather than correctness.

The study of variables that can be eliminated from a formula is an issue of gaining interest [13, 15]. Several researchers have noticed that an interpolant can contain fewer variables than $V_{\langle A,B \rangle}$. Related observations have been made by Simmonds and others [23] and have often featured in personal communication. We have shown that studying variables that cannot be eliminated from a proof can provide insights into the limitations of a family of interpolation systems.

Abstract interpretation, due to Cousot and Cousot [4] is a standard framework for reasoning about abstractions of a program's semantics. They have also applied the framework to inference rules in [7]. In program verification, the framework is typically applied to design abstract domains. In contrast, our application of abstract interpretation has been concerned with identifying concrete interpretations corresponding to existing interpolation systems and resolution. Our work was in part inspired by Ranzato and Tapparo's application of abstract interpretation to analyse state minimisation algorithms [22].

7 Conclusion

Interpolation algorithms have several applications in program verification and several interpolation algorithms exist. In this paper, we applied abstract interpretation to study a family of interpolation algorithms for propositional resolution proofs. We showed that existing interpolation algorithms can be derived by abstraction from a general, parametrised algorithm. In abstract interpretation terms, sets of clauses and the resolution proof system define an abstract domain and an abstract interpretation. The set of clauses annotated with interpolants and an interpolation system define a concrete domain and a concrete interpretation. We have also shown analysed these domains gain insights about interpolant strength and about variables that are eliminated by an interpolation system.

However, the analysis in this paper has focused on propositional interpolation systems. Software verification methods based on interpolation require interpolation systems for first order theories. The design and analysis of interpolation algorithms for such theories is the topic of much current research. An open question is whether the kind of analysis in this paper is applicable to these settings. Another question is whether the approach here extends to a comparative analysis of interpolation in different propositional proof systems. Answering these questions is left as future work.

Acknowledgements. Mitra Purandare's observation triggered the logical flows leading to this paper. Leopold Haller interpolated the flow diagrams from my sketches and discussions with Philipp Ruemmer proved useful. A great debt is to my fellow interpolator Georg Weissenbacher; I hope I have refuted his resolution against my abstract interpretation of our propositions. I am grateful to Greta Yorsh for her encouragement and comments.

References

1. S. R. Buss. Propositional proof complexity: An introduction. In U. Berger and H. Schwichtenberg, editors, *Computational Logic*, volume 165 of *NATO ASI Series F: Computer and Systems Sciences*, pages 127–178. Springer, 1999.
2. A. Carbone. Interpolants, cut elimination and flow graphs for the propositional calculus. *Annals of Pure and Applied Logic*, 83(3):249–299, 1997.
3. P. Cousot. Abstract interpretation. MIT course 16.399, Feb–May 2005.

4. P. Cousot and R. Cousot. Abstract interpretation: a unified lattice model for static analysis of programs by construction or approximation of fixpoints. In *Principles of Programming Languages*, pages 238–252. ACM Press, 1977.
5. P. Cousot and R. Cousot. Systematic design of program analysis frameworks. In *Principles of Programming Languages*, pages 269–282. ACM Press, 1979.
6. P. Cousot and R. Cousot. Abstract interpretation frameworks. *Journal of Logic and Computation*, 2(4):511–547, Aug. 1992.
7. P. Cousot and R. Cousot. Inductive definitions, semantics and abstract interpretations. In *Principles of Programming Languages*, pages 83–94. ACM, 1992.
8. W. Craig. Linear reasoning. A new form of the Herbrand-Gentzen theorem. *Journal of Symbolic Logic*, 22(3):250–268, 1957.
9. B. A. Davey and H. A. Priestley. *Introduction to Lattices and Order.* Cambridge University Press, 1990.
10. V. D'Silva, D. Kroening, M. Purandare, and G. Weissenbacher. Interpolant strength. In G. Barthe and M. Hermenegildo, editors, *Verification, Model Checking and Abstract Interpretation*, LNCS. Springer, 2010.
11. J. Esparza, S. Kiefer, and S. Schwoon. Abstraction refinement with Craig interpolation and symbolic pushdown systems. *Journal on Satisfiability, Boolean Modeling and Computation*, 5:27–56, June 2008. Special Issue on Constraints to Formal Verification.
12. R. Giacobazzi, F. Ranzato, and F. Scozzari. Making abstract interpretations complete. *Journal of the ACM*, 47(2):361–416, 2000.
13. S. Gulwani and M. Musuvathi. Cover algorithms and their combination. In *European Symposium on Programming*, volume 4960 of *LNCS*, pages 193–207. Springer, 2008.
14. G. Huang. Constructing Craig interpolation formulas. In *Computing and Combinatorics*, volume 959 of *LNCS*, pages 181–190. Springer, 1995.
15. L. Kovács and A. Voronkov. Interpolation and symbol elimination. In *Computer Aided Deduction*, volume 5663 of *LNCS*, pages 199–213, Berlin, Heidelberg, 2009. Springer.
16. J. Krajíček. Interpolation theorems, lower bounds for proof systems, and independence results for bounded arithmetic. *The Journal of Symbolic Logic*, 62(2):457–486, 1997.
17. S. Maehara. On the interpolation theorem of Craig (in Japanese). *Sûgaku*, 12:235–237, 1961.
18. P. Mancosu, editor. *Interpolations. Essays in Honor of William Craig*, volume 164:3 of *Synthese.* Springer, Oct. 2008.
19. K. L. McMillan. Interpolation and SAT-based model checking. In *Computer Aided Verification*, volume 2725 of *LNCS*, pages 1–13. Springer, 2003.
20. D. Mundici. Complexity of Craig's interpolation. *Fundamenta Informaticae*, 5:261–278, 1982.
21. P. Pudlák. Lower bounds for resolution and cutting plane proofs and monotone computations. *The Journal of Symbolic Logic*, 62(3):981–998, 1997.
22. F. Ranzato and F. Tapparo. Generalizing the Paige-Tarjan algorithm by abstract interpretation. *Information and Computation*, 206(5):620–651, 2008.
23. J. Simmonds, J. Davies, A. Gurfinkel, and M. Chechik. Exploiting resolution proofs to speed up ltl vacuity detection for bmc. In *Formal Methods in Computer-Aided Design*, pages 3–12. IEEE Computer Society, 2007.
24. G. Yorsh and M. Musuvathi. A combination method for generating interpolants. In *Computer Aided Deduction*, volume 3632 of *LNCS*, pages 353–368, 2005.