Constructing Craig Interpolation Formulas

Guoxiang Huang

Mathematics Department, University of Hawaii at Manoa
Honolulu, HI 96822 (E-mail: huang@math.hawaii.edu)

Abstract. A Craig interpolant of two inconsistent theories is a formula which is true in one and false in the other. This paper gives an efficient method for constructing a Craig interpolant from a refutation proof which involves binary resolution, paramodulation, and factoring. This method can solve the machine learning problem of discovering a first order concept from given examples. It can also be used to find sentences which distinguish pairs of nonisomorphic finite structures.

1 Background and Introduction

Let Σ and Π be two inconsistent first order theories. Then by Craig's Interpolation Theorem, there is a sentence θ, called a *Craig interpolant*, such that θ is true in Σ and false in Π and every nonlogical symbol occurring in θ occurs in both Σ and Π. Craig interpolants can be used to solve the problem of learning a first order concept by letting Σ and Π be the lists of positive and negative examples of the concept to be learned.

The standard nonconstructive model-theoretic proof of Craig's Theorem is in [3]. Lyndon showed how to construct an interpolant from a special form of natural deduction (see [1]). We show how to construct an interpolant from a refutation proof which uses binary resolution, factoring and paramodulation. In our examples, we use OTTER (the standard text on OTTER is [4]) to generate such proofs.

Craig interpolants can be used to find a sentence which distinguishes two nonisomorphic finite structures. Let Σ and Π be the atomic diagrams of the two structures. Then they are inconsistent and any Craig interpolant for them is a sentence which is true in one structure and false in the other.

2 Constructing Interpolation Formulas from Refutations

Let L_Σ and L_Π be two languages, Σ a theory in L_Σ, and Π a theory in L_Π such that $\Sigma \cup \Pi$ is not consistent. In this paper we use $\Diamond$ to represent contradiction, use $\Box$ to indicate the end of a proof, and suppose P is a refutation of $\Sigma \cup \Pi \models \Diamond$ involving only binary resolutions, paramodulations, and factorings. The input clauses (clauses at the top of the refutation) are required to be instances of clauses from Σ and Π. For convenience, we will assume that different input clauses have disjoint sets of variables.

For any occurrence L in the proof P of a relational symbol in $L_\Sigma \cup L_\Pi$, we define L *is from* Σ recursively by:

(i). If the occurrence L is in an input clause from Σ, we say it is from Σ; otherwise, it is not.

(ii). If the occurrence L is in a non-input clause C, then it is from Σ if the corresponding occurrence in some parent clause is from Σ.

Similarly, we can define L *is from* Π. Since factoring is allowed in the proof, several occurrences of some literal may be factored into a single one. So it is possible that a literal in some clause may be from both Σ and Π.

Let T and F be the truth values of "truth" and "falsehood". For a binary resolution proof P we use the following recursive procedure to assign formulas to the clauses in P:

Interpolation Algorithm

(i). If C is an input clause from Σ, its formula is F; if C is an input clause from Π, its formula is T.

(ii). If ϕ is assigned to $L \vee C$ and ψ is assigned to $\neg L' \vee D$, and if $(C \vee D)\pi$ is the resolvent of $L \vee C$ and $D \vee \neg L'$ resolving against $L\pi(= L'\pi)$, then the formula assigned to $(C \vee D)\pi$ is:

(a). $(\phi \vee \psi)\pi$ if the occurrences of both L and $\neg L'$ are from Σ alone;

(b). $(\phi \wedge \psi)\pi$ if the occurrences of both L and $\neg L'$ are from Π alone;

(c). $((\neg L' \wedge \phi) \vee (L \wedge \psi))\pi$ if neither (a) nor (b).

Definition 1. A formula θ is a *relational interpolant* of Σ and Π relative to a clause C iff

(1). all relational symbols of θ are in $L_\Sigma \cap L_\Pi$,

(2). $\Sigma \models \theta \vee C$, and

(3). $\Pi \models \neg\theta \vee C$.

Theorem 2. *For each clause C of a binary resolution proof P of $\Sigma \cup \Pi \models \Diamond$, the formula assigned by the above algorithm is a relational interpolant of Σ and Π relative to C. In particular, the formula θ assigned to the final empty clause of the proof P is a relational interpolant between Σ and $\neg\Pi$.*

Proof. It is obvious that any assigned formula contains only relation symbols from $L_\Sigma \cap L_\Pi$. So condition (1) of the definition holds.

For any occurrence of a clause or subclause C in the proof P, let C_Σ (let C_Π) be C with all occurrences of literals not from Σ (not from Π) deleted. Then $C_\Sigma \models C$, $C_\Pi \models C$, and $C_\Sigma \vee C_\Pi \iff C$ are valid and $(C \vee D)_\Sigma = (C_\Sigma \vee D_\Sigma)$ and $(C_\Sigma)\pi = (C\pi)_\Sigma$ for any unifier π.

We prove by induction on the depth of C in P the following strengthenings of (2) and (3):

(2)'. $\Sigma \models \theta \vee C_\Sigma$,

(3)'. $\Pi \models \neg\theta \vee C_\Sigma$.

Suppose C is an input clause from Σ. Then θ is F and $C_\Sigma = C$. Thus (2)' and (3)' hold since $\Sigma \models F \vee C$ and $\Pi \models T \vee C$. The argument for an input clause from Π is similar.

Suppose (2)′ and (3)′ are true for clauses $L \vee C$ and $\neg L' \vee D$ of P whose resolvent in P is $(C \vee D)\pi$ where π is a unifier such that $L\pi = L'\pi$. Assume $L \vee C$ is assigned the formula ϕ and $\neg L' \vee D$ is assigned ψ. Thus we have

$\Sigma \models \phi \vee (L \vee C)_\Sigma$, $\quad \Sigma \models \psi \vee (\neg L' \vee D)_\Sigma$,

$\Pi \models \neg\phi \vee (L \vee C)_\Pi$, $\quad \Pi \models \neg\psi \vee (\neg L' \vee D)_\Pi$.

Case (a). Suppose the occurrences of L and $\neg L'$ are both from Σ alone. Then $(L \vee C)_\Sigma = L \vee C_\Sigma$ and $(\neg L' \vee D)_\Sigma = \neg L' \vee D_\Sigma$. By resolution we get (2)′: $\Sigma \models ((\phi \vee C_\Sigma) \vee (\psi \vee D_\Sigma))\pi = (\phi \vee \psi)\pi \vee (C \vee D)\pi_\Sigma$. For (3)′ we have $(L \vee C)_\Pi = C_\Pi$ and $(\neg L' \vee D)_\Pi = D_\Pi$ and so $\Pi \models (\neg\phi \vee C_\Pi) \wedge (\neg\psi \vee D_\Pi)$ and $\Pi \models \neg(\phi \vee \psi)\pi \vee (C \vee D)\pi_\Pi$.

Case (b) for L and $\neg L'$ from Π alone is similar.

Case (c). In any model of Σ with any assignment of variables, if both $C_\Sigma\pi$ and $D_\Sigma\pi$ are false, then $(\phi \vee L)\pi$ and $(\psi \vee \neg L')\pi$ are true. So if $L\pi = L'\pi$ is true, then so is $\psi\pi$; if $L\pi$ is false, then $\phi\pi$ is true. Either way,

$((\neg L' \wedge \phi) \vee (L \wedge \psi) \vee (C \vee D)_\Sigma)$ is always true.

Similarly, $\Pi \models (((L \vee \neg\phi) \wedge (\neg L' \vee \psi)) \vee (C \vee D)_\Pi)\pi$.

Hence, by induction, the theorem holds. □

Resolution provers often use paramodulation to handle equality. Given clauses $C(r)$ and $s = t \vee D$ with no variables in common and a unifier π such that $r\pi = s\pi$ or $r\pi = t\pi$, paramodulation infers the paramodulant $(C(t) \vee D)\pi$ or $(C(s) \vee D)\pi$ respectively.

Definition 3. For a deduction P in $L_\Sigma \cup L_\Pi$, a *noncommon term* is a term which begins with a symbol not in $L_\Sigma \cap L_\Pi$. Such a term is called a *Σ-term* if its initial symbol is from Σ, a *Π-term* if its initial symbol is from Π. An occurrence of a Σ (Π)-term is *maximal* if this occurrence is not a subterm of a larger Σ (Π)-term.

Now we extend the Interpolation Algorithm to proofs with paramodulation as follows:

(iii). If ϕ is assigned to $C(r)$ and ψ is assigned to $s = t \vee D$ and if π is a unifier such that $r\pi = s\pi$, then the formula assigned to the paramodulant is:

(d). $[(\phi \wedge s = t) \vee (\psi \wedge s \neq t)]\pi \vee (s = t \wedge h(s) \neq h(t))\pi$ provided r occurs in $C(r)$ as a subterm of a maximal Π-term $h(r)$ and there is more than one occurrence of $h(r)$ in $C(r) \vee \phi$.

(e). $[(\phi \wedge s = t) \vee (\psi \wedge s \neq t)]\pi \wedge (s \neq t \vee h(s) = h(t))\pi$ provided r occurs in $C(r)$ as a subterm of a maximal Σ-term $h(r)$ and there is more than one occurrence of $h(r)$ in $C(r) \vee \phi$.

(f). $((\phi \wedge s = t) \vee (\psi \wedge s \neq t))\pi$ if neither (d) nor (e).

Lemma 4. *If ϕ, ψ are the interpolants relative to $C(r)$ and $s = t \vee D$, respectively, then the above formula is an interpolant relative to the paramodulant $(C(t) \vee D)\pi$.*

Proof. We prove case (f). Since $\Sigma \models (C(r) \vee \phi)\pi$ and $\Sigma \models (s = t \vee D \vee \psi)\pi$, for any model A of Σ, if $s\pi = t\pi$ in A, then $A \models (C(t) \vee \phi)\pi$; otherwise if $s\pi \neq t\pi$ in A, then $A \models D\pi \vee \psi\pi$. Either way, we have $A \models (C(t) \vee D \vee \theta)\pi$.

Similarly, $\Pi \models (C(t) \vee D \vee \neg\theta)\pi$ in the two cases: For $s\pi = t\pi, \Pi \models (\neg\phi \vee C(t))\pi$; for $s\pi \neq t\pi, \Pi \models (\neg\psi \vee D)\pi$. Thus the assigned formula satisfies the requirement. □

The final rule of inference we need is factoring. Given a clause $L \vee L' \vee C$ and a unifier π such that $L\pi = L'\pi$, factoring infers the clause $(L \vee C)\pi$. We extend the Interpolation Algorithm to proofs with factoring as follows:

(iv). If ϕ is assigned to $L \vee L' \vee C$ and π is a unifier as above, then we assign $\phi\pi$ to the factor clause $(L \vee C)\pi$.

Clearly $\Sigma \models L \vee L' \vee C \vee \phi$ and $\Pi \models L \vee L' \vee C \vee \neg\phi$ imply $\Sigma \models (L \vee C)\pi \vee \phi\pi$ and $\Pi \models (L \vee C)\pi \vee \neg\phi\pi$.

Thus for a refutation proof P by a series of binary resolutions, factorings, and paramodulations, applying the above extended algorithm gives a formula, say θ, for the empty clause. Since $\Sigma \models \theta$ and $\Pi \models \neg\theta$, θ is a relational interpolant between Σ and $\neg\Pi$. Though θ does not contain any non-common relational symbol, it may contain noncommon terms with constants or function symbols which are not in $L_\Sigma \cap L_\Pi$. We now show how to get a Craig interpolant by replacing all noncommon terms in θ with appropriately quantified variables.

First we define a *binary tree deduction* to be a deduction in which any clause is used at most once. Such a deduction involving only binary resolutions, factorings, and paramodulations forms a binary tree.

Lemma 5. *Any refutation P using only binary resolutions, paramodulations, and factorings, lifts to a binary tree deduction P_b with the same conclusion.*

Proof. We prove this lemma by induction on the number $k(P)$ of clauses which are used more than once in the deduction P. If $k(P) = 0$, P is a binary tree deduction. Assume the lemma holds for all deductions with $k(P) \leq n$ and suppose $k(P) = n + 1$. Let C be a clause such that C is used $m \geq 2$ times in P but all the ancestors of C are used only once. We construct a new deduction P' from P such that P' has m copies of C and its ancestors, and each copy of C and its ancestors is used exactly once in P'. Finally, variables may be renamed if necessary, so that different input clauses have disjoint sets of variables. Otherwise P is the same as P' and has the same conclusion. Since P' is a deduction with $k(P') \leq n$, by induction, P' lifts to a binary tree deduction P_b with the same conclusion.

Suppose P is a binary tree deduction whose input clauses have disjoint sets of variables and whose substitutions are generated by the usual unification algorithm, then the following properties hold in P:

1. Every variable of any noninput clause in P occurs in exactly one parent clause and thus traces back to a unique ancestral input clause.
2. Any two incomparable (neither is the ancestor of the other) clauses have disjoint sets of variables.
3. For any substitution π of P and any variable x, either π is trivial on x, i.e., $\pi(x) = x$, or x does not occur in the term $\pi(x)$.
4. If π is nontrivial on x, x never appears in any clause below π.

Definition 6. Given a binary tree deduction P as above, for any variable x occurring in P, let π_p, the *composite substitution for* P, be the substitution such that $\pi_p(x)$ is the term resulting from applying to x the composition of all the substitutions along the path from the unique input clause which contains x to the bottom of P.

Lemma 7. *For any clause C in such a binary tree deduction P, $C\pi_p$ is the clause obtained by applying to C the composition of all the substitutions along the path from C to the bottom of P.*

Proof. Suppose x is a variable of C. Then x traces back to a unique ancestral input clause D. All of the substitutions along the path from D to C are trivial on x since otherwise, x would not occur in C. Hence $\pi(x)$ = the composition of all substitutions from D to the bottom. □

We say a deduction is *propositional* if there are no nontrivial unifying substitutions involved in the deduction.

Lemma 8. *The Boolean operations $\vee, \wedge, \neg$ and propositional binary resolution, factoring, and paramodulation commute with substitution. That is, if π is a substitution, then $(A \vee B)\pi = A\pi \vee B\pi$, $(A \wedge B)\pi = A\pi \wedge B\pi$, $(\neg A)\pi = \neg(A\pi)$; and for any propositional binary resolution $\{A \vee L, B \vee \neg L\} \models A \vee B$, we have $\{A\pi \vee L\pi, B\pi \vee \neg L\pi\} \models A\pi \vee B\pi$; and for any propositional paramodulation $\{C(s),\ s = t \vee D\} \models C(t) \vee D$, we have $\{C(s)\pi,\ (s = t)\pi \vee D\pi\} \models C(t)\pi \vee D\pi$.*

Lemma 9. *Every binary tree proof P_b projects to a propositional proof P_p.*

Proof. Given a binary tree proof P_b, rename the variables if necessary so that the above four properties hold and let π_p be the composite substitution for P. Let P_p be the result of replacing each clause C of P_b with $C\pi_p$ and replacing each substitution with the trivial identity substitution. Then by Lemma 8 P_p is a projection of P_b and P_p is a propositional binary tree deduction. □

Lemma 10. *Assume P_b projects to P_p as in Lemma 9. If we apply the Interpolation Algorithm to the propositional deduction P_p, and if a clause C' in P_p is assigned formula ϕ', and if its corresponding clause C in P_b is assigned formula ϕ, then $\phi' = \phi\pi_p$. In particular, the assignments to $\Diamond$ from both deductions are the same.*

Proof. Any occurrence of a literal L in a clause C' of P_p is from Σ or Π or both iff its corresponding occurrence in P_b is from Σ, Π, or both. So the corresponding clauses of P_b and P_p are assigned interpolants by the same case of the Interpolation Algorithm. Lemma 8 gives the result. □

Let P_p be a propositional deduction in $L_\Sigma \cup L_\Pi$, and $t_1, ..., t_n$ be all the Π-terms with maximal occurrences in P_p. Let $x_1, ..., x_n$ be a set of new variables which do not occur in P_p. For any term or formula θ in P_p, define $\overline{\theta}(x_1, ..., x_n)$ to be the term or formula obtained by simultaneously replacing all maximal occurrences of the Π-terms t_j's by the new variables x_j's. We call $\overline{\theta}$ the *lifted* formula of θ from Π-terms.

Lemma 11.

$$\begin{aligned}
\overline{(A \vee B)}(x_1, ..., x_n) &\iff \overline{A}(x_1, ..., x_n) \vee \overline{B}(x_1, ..., x_n) \\
\overline{(A \wedge B)}(x_n, ..., x_n) &\iff \overline{A}(x_1, ..., x_n) \wedge \overline{B}(x_1, ..., x_n) \\
\overline{(s = t)}(x_1, ..., x_n) &\iff \overline{s}(x_1, ..., x_n) = \overline{t}(x_1, ..., x_n) \\
\overline{(\neg A)}(x_1, ..., x_n) &\iff \neg \overline{A}(x_1, ..., x_n) \\
\theta &= \overline{\theta}(t_1, ..., t_n)
\end{aligned}$$

Lemma 12. *If θ is the relational interpolant of Σ and Π relative to C by the Interpolation Algorithm for the propositional deduction P_p, then we have* $\Sigma \models \overline{(C \vee \theta)}(x_1, ..., x_n)$.

Proof. We prove this lemma by induction on P_p. If $\bar{C}(t_1, ..., t_n)$ is an instance of an input clause from Σ, then all the Π-terms in C come from free variables by the unifying substitutions in the original deduction. So by the construction of P_p we know that $\overline{C}(x_1, ..., x_n)$ is an instance of some input clause in Σ, and F is assigned to this clause. Thus $\Sigma \models \overline{C}(x_1, .., x_n) \vee F$. If $\overline{C}(t_1, .., t_n)$ is an instance of input clause from Π, then it has assigned formula T and $\Sigma \models \overline{C}(x_1, .., x_n) \vee T$ holds.

Now assume that $\Sigma \models \overline{(C \vee L \vee \phi)}(x_1, .., x_n)$ and $\Sigma \models \overline{(D \vee \neg L \vee \psi)}(x_1, .., x_n)$ and that $C \vee L$ and $D \vee \neg L$ resolving against L gives $C \vee D$ with interpolant θ. We show that $\Sigma \models \overline{(C \vee D \vee \theta)}(x_1, .., x_n)$.

Notice that by propositional deduction and Lemma 11 we have
$\{\overline{C} \vee \overline{L} \vee \overline{\phi}, \quad \overline{D} \vee \neg \overline{L} \vee \overline{\psi}\} \models \overline{C} \vee \overline{D} \vee (\overline{\phi} \vee \overline{\psi})$, and
$\{\overline{C} \vee \overline{L} \vee \overline{\phi}, \quad \overline{D} \vee \neg \overline{L} \vee \overline{\psi}\} \models \overline{C} \vee \overline{D} \vee (\neg \overline{L} \wedge \overline{\phi}) \vee (\overline{L} \wedge \overline{\psi})$.

Using Lemma 11 again proves the Lemma for case (a) and case (c) of the Interpolation Algorithm definition of θ. For case (b), $\theta = \phi \wedge \psi$, and the occurrences of L and $\neg L$ are not from Σ. By the proof of Theorem 2 we know that $\Sigma \models \overline{C} \vee \overline{\phi}$ and $\Sigma \models \overline{D} \vee \overline{\psi}$ respectively. Thus we have $\Sigma \models \overline{C} \vee \overline{D} \vee (\overline{\phi} \wedge \overline{\psi})$.

Next assume $C(s)$ and $s = t \vee D$ gives $C(t) \vee D$ by paramodulation. Assume $\Sigma \models \overline{(C(s) \vee \phi)}(x_1, ..., x_n)$ and $\Sigma \models \overline{(s = t \vee D \vee \psi)}(x_1, ..., x_n)$. Here we consider case (d) of the assignment for paramodulation in which s occurs in $C(s)$ as a subterm of a maximal Π-term $h(s)$ which occurs more than once in $C(s) \vee \phi$. Then since $h(s), h(t)$ are distinct Π-terms, they will be replaced by distinct new variables $\overline{h(s)}$, $\overline{h(t)}$ in $\overline{C(t) \vee \phi}$. For any model of Σ and any assignment of all free variables in the lifted paramodulant and its assigned formula, if $\overline{C}(s)$ and $\bar{s} = \bar{t}$ are true but $\overline{C}(t)$ is false, then we must have $\overline{h(s)} \neq \overline{h(t)}$. So in this case we have: $\Sigma \models \overline{(s = t \wedge h(s) \neq h(t))}(x_1, ..., x_n)$. And hence, $\Sigma \models \overline{(C(t) \vee D \vee \theta)}(x_1, ..., x_n)$.

The arguments for the other cases are straightforward. □

We now assign a *dual* formula to each clause in the proof P as follows:

(i). If C is an input clause from Σ, its formula is T; if C is an input clause from Π, its formula is F.

(ii). If ϕ is assigned to $L \vee C$ and ψ is assigned to $\neg L' \vee D$, and if $(C \vee D)\pi$ is the resolvent of $L \vee C$ and $D \vee \neg L'$ against $L\pi = L'\pi$, then the formula assigned to $(C \vee D)\pi$ is:

(a). $(\phi \wedge \psi)\pi$ if the occurrences of both L and $\neg L'$ are from Σ alone;

(b). $(\phi \vee \psi)\pi$ if the occurrences of both L and $\neg L'$ are from Π alone;

(c). $((L \wedge \phi) \vee (\neg L' \wedge \psi))\pi$ if neither (a) nor (b).

(iii). If $(C(t) \vee D)\pi$ is the paramodulant from above described paramodulation, its formula is

(d). $[(\phi \vee s \neq t) \wedge (\psi \vee s = t)]\pi \wedge (s \neq t \vee h(s) = h(t))\pi$ provided the r is a subterm of a maximal Π-term $h(r)$ and there is more than one occurrence of $h(r)$ in $C(r) \vee \phi$.

(e). $[(\phi \vee s \neq t) \wedge (\psi \vee s = t)]\pi \vee (s = t \vee h(s) \neq h(t))\pi$ provided the r is a subterm of a maximal Σ-term $h(r)$ and there is more than one occurrence of $h(r)$ in $C(r) \vee \phi$.

(f). $((\phi \wedge s = t) \vee (\psi \wedge s \neq t))\pi$ if neither (d) nor (e).

(iv). If ϕ is assigned to $L \vee L' \vee C$ and π is a unifier such that $L\pi = L'\pi$, then we assign $\phi\pi$ to the factor clause $(L \vee C)\pi$.

By induction on the depth of a clause in the deduction we can show

Lemma 13. *The formula assigned by the dual method is the logical negation of that assigned by the original Interpolation Algorithm.*

Corollary 14. *Assume that $\bar{\theta}(s_1, ..., s_k)$ is the dual formula assigned to C in P_p by the dual assignment algorithm, where $s_1, ..., s_k$ are all the Σ-terms with maximal occurrences in P_p. If we let $\hat{\theta}(y_1, ..., y_k)$ be the formula obtained by simultaneously replacing all maximal occurrences of the Σ-terms $s_1, ..., s_k$ by the new variables $y_1, ..., y_k$, then we have $\Pi \models \hat{\theta}(y_1, ..., y_k)$.*

Now we are ready to quantify all the variables for noncommon terms in the relational interpolant θ of Σ and Π relative to the empty clause. Assume that all the maximal Σ-terms and Π-terms are $\{t_1, ..., t_n\}$, ordered by their lengths. Assume $\{t_1, ..., t_n\} = \{r_1, ..., r_k\} \cup \{s_{k+1}, ..., s_n\}$ where the r_i's are the maximal Π-terms and the s_j's are the maximal Σ-terms. If lifting θ from Π-terms gives $\bar{\theta}(x_1, ..., x_k)$, and lifting $\bar{\theta}(x_1, ..., x_k)$ from the Σ-terms gives $\theta^*(z_1, ..., z_n)$ where the z_i's are new variables for the t_i's, then we have

Theorem 15. *$Q_1 z_1 ... Q_n z_n \theta^*(z_1, ..., z_n)$ is a Craig interpolant separating Σ and Π, where Q_i is $\forall$ if t_i is a Π-term, otherwise Q_i is $\exists$.*

Proof. Clearly $Q_1 z_1 ... Q_n z_n \theta^*(z_1, ..., z_n)$ is a formula in $L_\Sigma \cap L_\Pi$. By Lemma 12 we have $\Sigma \models \forall x_1 ... x_k \bar{\theta}(x_1 ... x_k)$.

Each maximal Σ-term of $\bar{\theta}(x_1,...,x_k)$ is a lifting $\bar{s}_j(x_1,...,x_k)$ of one of the maximal Σ-terms $s_{k+1},...,s_n$ of θ. If x_i occurs in $\bar{s}_j(x_1,...,x_k)$ then the term r_i which x_i replaces is a subterm of s_j and thus r_i occurs before s_j in the list $\{t_1,...,t_n\}$ and the variable for r_i occurs before the variable for s_j in the prefix $Q_1z_1...Q_nz_n$. Hence $\bar{s}_j$ is a witness for the quantifier $\exists y_j$ in $Q_1z_1...Q_nz_n\theta^*(z_1,...,z_n)$. Hence, $\Sigma \models Q_1z_1...Q_nz_n\theta^*(z_1,...,z_n)$.

On the other side, for the dual formula $\neg\theta$, using the same order among the noncommon terms and by the corollary above and the fact that θ^* is also the lifting from Π-terms of the lifting $\hat{\theta}$ of θ from Σ-terms, we also have

$\Pi \models \overline{Q}_1z_1...\overline{Q}_nz_n\neg\theta^*(z_1,...,z_n)$ where $\overline{Q}_j = \forall\ (\exists)$ iff $Q_j = \exists\ (\forall)$. Moving the negation symbol out we finally have $\Pi \models \neg Q_1z_1...Q_nz_n\theta^*(z_1,...,z_n)$. □

The formula θ^* may contain free variables other than $z_1,...,z_n$. We get a Craig interpolant sentence by quantifying these extra variables with the quantifier Q_1 or any other sequence of quantifiers.

Example 1. Let $\Sigma = \{R(x,a) \vee R(x,b)\}$, $\Pi = \{\neg R(c,y)\}$, where a, b and c are distinct constants. An OTTER resolution refutation for $\Sigma \cup \Pi \models \Diamond$ is:

1 $R(x,a) \vee R(x,b)$.
2 $-R(c,y)$.
3 [binary,1,2] R(c,b).
4 [binary,3,2] .

The Interpolation Algorithm gives the formula $\theta = R(c,a) \vee R(c,b)$. Since a and b are Σ-terms and c is a Π-term, we replace a and b by the existentially quantified variables x and y, and replace c by the universally quantified variable z. Since the lengths of a, b, c are all 1, the order among the quantifiers does not matter. Thus the following three formulas are all Craig interpolants between Σ and Π:

$\forall z\exists xy(R(z,x) \vee R(z,y))$, $\exists x\forall z\exists y(R(z,x) \vee R(z,y))$, $\exists xy\forall z(R(z,x) \vee R(z,y))$

Example 2. Let $\Sigma = \{x \neq f(x), x \neq f(f(x))\}$, $\Pi = \{y = x \vee y = g(x)\}$, where both f and g are functions. Any model of Σ has a universe of size at least 3, while any model of Π has a universe of size at most 2. So Σ and Π are inconsistent. An OTTER resolution refutation for $\Sigma \cup \Pi \models \Diamond$ is:

1 x≠f(x).
2 x≠f(f(x).
3 y=x∨y=g(x).
4 [binary,3.1,2.1] x=g(f(f(x))).
5 [binary,3.1,1.1] x=g(f(x)).
10 [para-into,4.1.2,5.1.2] x=f(x).
11 [binary,10.1,1.1] .

The formula θ the Interpolation Algorithm gives is

$[(x \neq f(f(x)) \wedge x \neq g(f(f(x)))) \vee (x = g(f(f(x))) \wedge f(x) \neq f(f(x)))] \wedge x \neq f(x)$.

The noncommon terms, when sorted according to lengths, are $f(x), f(f(x))$, $g(f(f(x)))$, where $g(f(f(x)))$ is a Π-term and the others are Σ-terms. Replacing these terms with the variables u, v, w and quantifying them gives the formula

$\theta^* = \exists u v \forall w[(x \neq v \wedge x \neq w) \vee (x = w \wedge u \neq v)] \wedge (x \neq u)$.

Note that any model for θ^* contains at least three elements: It can not contain only one or two elements, for x, u, v must be distinct. Thus θ^* is a Craig interpolant separating Σ and Π.

3 Applications of Craig Interpolants

Given two finite structures, to show they are isomorphic, one finds an isomorphism. To show they are not, one gives a statement that separates them, i.e., a sentence which is true in one structure but false in the other. The Interpolation Algorithm can be used to find such a sentence.

For structures S_1 with elements $\{a_1, .., a_n\}$ and S_2 with elements $\{b_1, ..., b_n\}$, assume that the universes for S_1 and S_2 are disjoint, and all the elements a_i, b_j are named by new distinct constant symbols. Furthermore assume the diagrams (the collection of all atomic sentences and negations of atomic sentences which hold in the structure) for the structures are Δ_1 and Δ_2 respectively. Then each of the diagrams is a theory in some language. If the two structures are not isomorphic, then $\Sigma = \Delta_1 \cup \forall x(x = a_1 \vee ... \vee x = a_n)$ and $\Pi = \Delta_2 \cup \forall y(y = b_1 \vee ... \vee y = b_n)$ are inconsistent, and by completeness there exists a refutation proof for $\Sigma \cup \Pi \models \Diamond$. Applying the Interpolation Algorithm to this proof gives a first order sentence which separates the structures.

For example, let S_1 and S_2 be directed graphs. S_1 has vertices $\{a, b, c\}$ with edges $\{(a, b), (a, c)\}$. S_2 has vertices $\{a', b', c'\}$ with edges $\{(a', b'), (c', a')\}$. We use binary relation p to represent the edges of the graphs. Then the diagram for S_1 is

$$\Delta_1 = \{p(a,b), p(a,c), \neg p(b,a), \neg p(b,c), \neg p(c,a), \neg p(c,b), a \neq b, a \neq c, b \neq c\}.$$

And the diagram for S_2 is

$\Delta_2 = \{p(a', b'), p(c', a'), \neg p(a', c'), \neg p(b', a'), \neg p(b', c'), \neg p(c', b'),$
$a' \neq b', a' \neq c', b' \neq c'\}$.

So $\Sigma = \Delta_1 \cup \forall x(x = a \vee x = b \vee x = c)$ and $\Pi = \Delta_2 \cup \forall x(x = a' \vee x = b' \vee x = c')$.

A refutation proof by OTTER is the following:

1 p(a,b).
2 p(a,c).
7 (a ≠ b).
8 (a ≠ c).
10 (x = a) ∨ (x = b) ∨ (x = c).
14 -p(b1,c1).

15 -p(c1,b1).
16 -p(a1,c1).
19 (x = a1) ∨ (x = b1) ∨ (x = c1).
22 (b1 ≠ c1).
30 [para-from,10,7] (a ≠ x) ∨ (x = a) ∨ (x = c).
34 [para-from,10,1] p(a,x) ∨ (x = a) ∨ (x = c).
44 [para-from,19,16] -p(x,c1) ∨ (x = b1) ∨ (x = c1).
274 [para-from,30,8] (a ≠ x) ∨ (x = a).
507 [para-from,34,2] p(a,x) ∨ (x = a).
699 [para-from,44,14] -p(x,c1) ∨ (x = c1).
711 [binary,699,507] (a = c1) ∨ (c1 = a).
783 [binary,711,274] (c1 = a).
794 [para-from,783,22] (b1 ≠ a).
797 [para-from,783,15] -p(a,b1).
816 [binary,794,507] p(a,b1).
817 [binary,816,797] .

Applying the Interpolation Algorithm, and using some trivial logic rules such as $A \vee \neg A \iff T$, $(A \wedge \neg B) \vee B \iff A \vee B$, $A \wedge A \iff A$, $A \wedge \neg A \iff F$, $A \vee \neg A \iff T$, to simplify the formula, we get the following relational interpolant

θ : $(c' = a \vee p(a,c')) \wedge (b' = a \vee p(a,b'))$.

Note that a is a Σ-term, while b' and c' are Π-terms. If we replace a, b', c' by x, y, z, respectively, by Theorem 15, we get the following formulas. They all separate the two graphs:

$\exists x \forall y z[(z = x \vee p(x,z)) \wedge (y = x \vee p(x,y))]$,
$\forall y \exists x \forall z[(z = x \vee p(x,z)) \wedge (y = x \vee p(x,y))]$,
$\forall y z \exists x[(z = x \vee p(x,z)) \wedge (y = x \vee p(x,y))]$.

Note that there is a shorter formula $\exists x \forall y(x = y \vee p(x,y))$ which separates the two given structures. The generation of minimal length separating sentences is an open problem. We also need more efficient proof strategies for such problems since current resolution provers can not find refutation proof for pairs of structures with more than 6 elements.

Acknowledgements: The author thanks Dale Myers for his numerous suggestions and corrections to earlier versions of this paper.

References

1. Roger Lyndon, *Notes on Logic*, D. Van Nostrand Company, Princeton, 1966.
2. Chin-Liang Chang, Richard Char-Tung Lee, *Symbolic Logic and Mechanical Theorem Proving*, Academic Press, 1973.
3. C C. Chang, H. Jerome Keisler, *Model Theory*, North Holland, 1990.
4. Larry Wos, Overbeek, Lusk, Boyle, *Automated Reasoning: Introduction and Applications*, McGraw-Hill, 1992.