Interpolation: Theory and Applications

Vijay D’Silva
Google Inc., San Francisco

Logic Colloquium, U.C. Berkeley
2016
Interpolation Lemma (1957)

William Craig in 1988
http://sophos.berkeley.edu/interpolations/

**THEOREM.** A NEW FORM OF THE HERBRAND-GENTZEN THEOREM.

WILLIAM CRAIG

1. **Introduction.** In Herbrand’s Theorem [2] or Gentzen’s Extended Hauptsatz [1], a certain relationship is asserted to hold between the structures of $A$ and $A'$, whenever $A$ implies $A'$ (i.e., $A \supset A'$ is valid) and moreover $A$ is a conjunction and $A'$ an alternation of first-order formulas in prenex normal form. Unfortunately, the relationship is described in a roundabout way, by relating $A$ and $A'$ to a quantifier-free tautology. One purpose

**THEOREM.** THREE USES OF THE HERBRAND-GENTZEN THEOREM IN RELATING MODEL THEORY AND PROOF THEORY

WILLIAM CRAIG

1. **Introduction.** One task of metamathematics is to relate suggestive but nonelementary model-theoretic concepts to more elementary proof-theoretic concepts, thereby opening up model-theoretic problems to proof-theoretic methods of attack. Herbrand’s Theorem (see [8] or also [9],
<table>
<thead>
<tr>
<th></th>
<th>A Brief History of Interpolation</th>
</tr>
</thead>
<tbody>
<tr>
<td>2</td>
<td>Analysis with Interpolants</td>
</tr>
<tr>
<td>3</td>
<td>Labelled Interpolation Systems</td>
</tr>
<tr>
<td>4</td>
<td>Current and Future Directions</td>
</tr>
</tbody>
</table>
An interpolant $I$ for a pair of formulae $A$ and $B$, where the validity of $A$ implies the validity of $B$, is a formula satisfying that: (i) $A$ implies $I$, (ii) $I$ implies $B$, and (iii) the *vocabulary condition* that the non-logical symbols in $I$ occur in both $A$ and $B$.

A logic has the interpolation property if every such $A$ and $B$ has an interpolant.

$$P \lor (Q \land R) \quad P \lor Q \quad S \rightarrow (\neg Q \rightarrow P)$$

**Theorem.** (Craig, 1957) First-order logic has the interpolation property.
“In terms of reasoning, this is not at all surprising. If A involves apples and oranges, and B involves apples and bananas and A implies B, then A ought to imply a statement that involves only apples and B ought to follow from a statement that involves only apples. The oranges should not help and the bananas should not hurt.

So what is the mystery then? The Craig statement is trickier to prove than one might think. One has to have the same statement about apples for A and B! ”

-- Alessandra Carbone, Bulletin of the AMS, April ’97
Dear Andreas,

I would like to congratulate Cadence Research Labs on their 15th Anniversary. In these 15 years, Cadence Research Labs has worked at several frontiers of Electronic Design Automation. They focus on hard problems that when solved significantly push the state of the art forward. They found novel solutions to system, synthesis and formal verification problems.

Formal verification is the process of exhaustively validating that a logic entity behaves correctly. In contrast to testing-based approaches, which may expose flaws though generally cannot yield a proof of correctness, the exhaustiveness of formal verification ensures that no flaw will be left unexposed. Formal verification is thus a critical technology in many domains, being essential to safety-critical applications and to enable increased quality and reduced development costs of hardware and software systems. The benefits of formal verification come at a substantial "cost": its exhaustiveness implies that it generally requires computational resources which grow exponentially with the size of the entity being analyzed. Cadence Research Labs has had a fundamental role in the research and development of leading-edge formal verification technologies, which have been critical to increasing the scalability and applicability of formal verification techniques to an industrially relevant level.

CRL made important contributions in satisfiability checking technologies and model checking algorithms. Satisfiability checking is arguably one of the most fundamental algorithms in computer-aided design, with pervasive application domains including verification. Members of Cadence Research Labs are world-recognized experts in the field of high-performance satisfiability solvers, and collectively have developed a set of solvers including MiniSAT, BerkMin, and Forklift which have won numerous competitions, been downloaded and used in thousands of applications, and have integrated novel tricks and ideas which have become the basis of countless other solvers.

Model checking algorithms are widely used for verifying hardware and software models. CRL has pioneered numerous fundamental ideas and algorithms to this field, including "interpolation" as a satisfiability-based proof method which is often dramatically faster and more scalable than prior proof techniques. CRL researchers invented numerous novel methods to automatically reduce the domain of a verification problem through "abstracting" it based upon unsatisfiability proofs. These techniques have substantially increased the scalability of formal verification of complex hardware designs.

CRL researchers have not only used logic optimizations to speed up formal verification algorithms, but are now also applying them to sequential optimization. Sequential synthesis has long been a holy grail in logic optimization. A large part of the design space remains untapped unless one can reliably and effectively optimize and verify in the sequential domain. Recent progress from CRL shows that there is some promise we can tap into this some time in the not too distant future.

Leon
Leon Stok
Director,
Electronic Design Automation
IBM Corporation
Dear Andreas,

I would like to congratulate Cadence Research Labs on their 15th Anniversary. In these 15 years, Cadence Research Labs has worked at several frontiers of Electronic Design Automation. They focus on hard problems that when solved significantly push the state of the art forward. They found novel solutions to system, synthesis and formal verification problems.

Formal verification is the process of exhaustively validating that a logic entity behaves correctly. In contrast to testing-based approaches, which may expose flaws though generally cannot yield a proof of correctness, the exhaustiveness of formal verification ensures that no flaw will be left unexposed. Formal verification is thus a critical technology in many domains, being essential to safety-critical applications and pervasive application domains including verification of complex hardware designs.

Model checking algorithms are widely used for verifying hardware and software models. CRL has pioneered numerous fundamental ideas and algorithms to this field, including "interpolation" as a satisfiability-based proof method which is often dramatically faster and more scalable than prior proof techniques. CBL researchers invented numerous novel methods to automatically reduce the domain of a verification problem through "abstracting" it based upon unsatisfiability proofs. These techniques have substantially increased the scalability of formal verification of complex hardware designs.

Model checking algorithms are widely used for verifying hardware and software models. CRL has pioneered numerous fundamental ideas and algorithms to this field, including "interpolation" as a satisfiability-based proof method which is often dramatically faster and more scalable than prior proof techniques. CBL researchers invented numerous novel methods to automatically reduce the domain of a verification problem through "abstracting" it based upon unsatisfiability proofs. These techniques have substantially increased the scalability of formal verification of complex hardware designs.

CRL researchers have not only used logic optimizations to speed up formal verification algorithms, but are now also applying them to sequential optimization. Sequential synthesis has long been a holy grail in logic optimization. A large part of the design space remains untapped unless one can reliably and effectively optimize and verify in the sequential domain. Recent progress from CRL shows that there is some promise we can tap into this some time in the not too distant future.

Leon

Leon Stok
Director,
Electronic Design Automation
IBM Corporation
Interpolation Within Logic

- Simpler proofs of known properties: Beth definability, Robinson’s theorem.
- Preservation under homomorphisms (connections to finite-model theory).

- Model theoretic characterizations: See Makowsky ’85 for a survey.
- Amalgamation: See Czelakowski and Pigozzi ’95.

- Guarded fragment: Hoogland, Marx, Otto ’00.
- Modal and fixed point logics: Maksimova ’79, ’91, Ten Cate ’05.
- Uniform interpolation: Pitt ’92, Visser ’96, d’Agostino, Hollenberg ’00.
Interpolation and Complexity Theory

<table>
<thead>
<tr>
<th>Year</th>
<th>Event</th>
</tr>
</thead>
<tbody>
<tr>
<td>1971</td>
<td>1971, Cook. The Complexity of Theorem Proving Procedures</td>
</tr>
<tr>
<td>1982</td>
<td>Mundici, NP and Craig’s Interpolation Theorem (pub. 1984)</td>
</tr>
<tr>
<td>1983</td>
<td>Mundici, A Lower bound for the complexity of Craig’s Interpolants in Sentential Logic</td>
</tr>
</tbody>
</table>

**Theorem.** (Mundici, 1982) At least one of the following is true.

1. \( P = \text{NP} \).
2. \( \text{NP} \neq \text{coNP} \).
3. For \( F \) and \( G \) in propositional logic, such that \( F \implies G \), an interpolant is not computable in time polynomial in the size of \( F \) and \( G \).
A proof system $\vdash$ has feasible interpolation if, whenever there is a short refutation of $A \land B$, the interpolant is computable in polynomial time in the size of the proof.

**Lemma** If there is a resolution refutation of size $n$ for a formula $A \land B$, there is an interpolant of circuit size $3n$ that is computable in time $n$. 

Jan Krajíček, Interpolation theorems, lower bounds for proof systems, and independence results for bounded arithmetic.

Pudlák, Lower Bounds for Resolution and Cutting Plane Proofs and Monotone Computations
# Interpolants in Automated Reasoning

<table>
<thead>
<tr>
<th>Year</th>
<th>Paper</th>
</tr>
</thead>
<tbody>
<tr>
<td>1995</td>
<td>Huang, Constructing Craig Interpolation Formulas. (OTTER)</td>
</tr>
<tr>
<td>2001</td>
<td>Amir, McIlraith, Partition-Based Logical Reasoning.</td>
</tr>
<tr>
<td>2003</td>
<td>McMillan, Interpolation and SAT-Based Model Checking.</td>
</tr>
<tr>
<td>2004</td>
<td>Henziger, Jhala, Majumdar, McMillan, Abstractions from Proofs</td>
</tr>
<tr>
<td>2005</td>
<td>McMillan, An Interpolating Theorem Prover</td>
</tr>
<tr>
<td></td>
<td>Section Title</td>
</tr>
<tr>
<td>---</td>
<td>---------------------------------------------------</td>
</tr>
<tr>
<td>1</td>
<td>A Brief History of Interpolation</td>
</tr>
<tr>
<td>2</td>
<td>Analysis with Interpolants</td>
</tr>
<tr>
<td>3</td>
<td>Labelled Interpolation Systems</td>
</tr>
<tr>
<td>4</td>
<td>Current and Future Directions</td>
</tr>
</tbody>
</table>
A Fundamental Problem in Program Verification

```c
int x = i;
int y = j;
while (foo()) {
    // Code that does not modify x, y, i, j.
    x = y + 1;
    y = x + 1;
}
if (i = j && x <= 10)
    assert(y <= 10);
```

- The assertion checking problem.
- More generally, a safety property, of a discrete, state transition system can be reduced to reachability.
- Manual proof would use Hoare logic and invariants.
Bounded Execution as a Formula

```c
int x = i;
int y = j;
while (foo()) {
    // Code that does not modify x,y,i,j.
    x = y + 1;
    y = x + 1;
}
if (i == j && x <= 10)
    assert(y <= 10);
```

```plaintext
x0 = i and
y0 = j and
x1 = y0 + 1 and
y1 = x0 + 1 and
x2 = y1 + 1 and
y2 = x1 + 1 and
x3 = y2 + 1 and
y3 = x2 + 1 and
(i == j and x3 <= 10) implies (y3 > 10)
```
Empirical Progress in SAT Solving

(a) Number of variables

(b) Number of clauses

(c) Clause-to-variable ratio

Katebi, Sakallah, Marques-Silva, 2011
Empirical Progress in SAT Solving

Results of the SAT competition/race winners on the SAT 2009 application benchmarks, 20mn timeout

Biere, 2011
Do these formulas have an interpolant?

```
(declare-const i Int)
(declare-const j Int)
(declare-const x0 Int)
(declare-const y0 Int)
(declare-const x1 Int)
(declare-const y1 Int)
(declare-const x2 Int)
(declare-const y2 Int)
(declare-const x3 Int)
(declare-const y3 Int)
(compute-interpolant
  (and (and (= x0 i) (= y0 j))
       (and (and (= x1 (+ y0 1)) (= y1 (+ x0 1)))
            (and (= x2 (+ y1 1)) (= y2 (+ x1 1)))
       )
     )
  (and (and (= x3 (+ y2 1)) (= y3 (+ x2 1)))
       (and (= i j) (and (<= x3 10) (> y3 10)))
  )
)
```

```
unsat
(let ((a!1 (not (<= 0 (+ (* (- 1) x2) y2)))))
 (not (and a!1 (= i j)))
)
Interpolants from Bounded Executions

\[
\begin{align*}
    x_0 &= i \text{ and } y_0 = j \text{ and } \\
    x_1 &= y_0 + 1 \text{ and } y_1 = x_0 + 1 \text{ and } \\
    x_2 &= y_1 + 1 \text{ and } y_2 = x_1 + 1 \text{ and } \\
    x_3 &= y_2 + 1 \text{ and } y_3 = x_2 + 1 \text{ and } \\
    (i = j \text{ and } x_3 \leq 10) \implies (y_3 > 10)
\end{align*}
\]

- Interpolant is with respect to a theory.
- Computed from a proof produced by solver for the theory.
- After renaming, we have an invariant.
- Invariant generation typically involves a series of quantifier elimination steps, or fixed point computation.

\[
i = j \implies x_2 \leq y_2
\]
Analysis of a System with Interpolants

• A poor person’s quantifier elimination.

• Analysis algorithms involve repeated calls to a solver and repeated computation of invariants.

• Solvers: Efficient in practice contrary to theoretical expectations.

• Proof generation: Arose from theory to explain practice.

• Efficient interpolation: First studied in theory, applied in practice, leading to more theory.
<table>
<thead>
<tr>
<th></th>
<th>A Brief History of Interpolation</th>
</tr>
</thead>
<tbody>
<tr>
<td>2</td>
<td>Analysis with Interpolants</td>
</tr>
<tr>
<td>3</td>
<td>Labelled Interpolation Systems</td>
</tr>
<tr>
<td>4</td>
<td>Current and Future Directions</td>
</tr>
</tbody>
</table>
### Terminology

<table>
<thead>
<tr>
<th><strong>Var</strong></th>
<th><strong>Literal</strong></th>
<th><strong>Clause</strong></th>
<th><strong>CNF Formula</strong></th>
</tr>
</thead>
<tbody>
<tr>
<td>Boolean variables: $a_1, a_2, a_3, \ldots$</td>
<td>Variable or its negation: $a, \overline{a}, \neg a$</td>
<td>Disjunction or set of literals: ${a_1, a_2, a_5}$</td>
<td>Conjunction or set of clauses: ${{a}, {\overline{a}, b}}$</td>
</tr>
</tbody>
</table>

#### Resolution

$$
\frac{C \lor x \quad \overline{x} \lor D}{C \lor D}
$$

[Resolution]
Interpolating Proof Rules

\[ A\text{-Hyp} \quad \frac{C \quad \{\ell \in C \mid \text{var}(\ell) \in B\}}{C \quad \{\ell \in C \mid \text{var}(\ell) \in B\}} \quad [C \in A] \]

\[ B\text{-Hyp} \quad \frac{C \quad \{\top\}}{C \quad \{\top\}} \quad (C \in B) \]

\[ A\text{-Res} \quad \frac{C \lor x \quad [I_1] \quad \overline{x} \lor D \quad [I_2]}{C \lor D \quad [I_1 \lor I_2]} \quad (x \in \text{var}(A) \setminus \text{var}(B)) \]

\[ B\text{-Res} \quad \frac{C \lor x \quad [I_1] \quad \overline{x} \lor D \quad [I_2]}{C \lor D \quad [I_1 \land I_2]} \quad (x \in \text{var}(B)) \]

McMillan, 2003
Annotate formulae with Partial Interpolants

Interpolating Proof Rules

Split rules based on vocabulary

\[\boxed{\text{A-Hyp}}\]

\[
C \quad \{\ell \in C \mid \text{var}(\ell) \in B\} \quad [C \in A]
\]

\[\boxed{\text{B-Hyp}}\]

\[
C \quad [\top] \quad (C \in B)
\]

\[\boxed{\text{A-Res}}\]

\[
\frac{C \lor x [I_1]}{C \lor D} \quad \frac{\overline{x} \lor D [I_2]}{[I_1 \lor I_2]} \quad (x \in \text{var}(A) \setminus \text{var}(B))
\]

\[\boxed{\text{B-Res}}\]

\[
\frac{C \lor x [I_1]}{C \lor D} \quad \frac{\overline{x} \lor D [I_2]}{[I_1 \land I_2]} \quad (x \in \text{var}(B))
\]

McMillan, 2003
Applying Interpolating Proof Rules

\[ A = (a_1 \lor \overline{a}_2) \land (\overline{a}_1 \lor a_3) \land a_2 \]

\[ B = (\overline{a}_2 \lor a_3) \land (a_2 \lor a_4) \land \overline{a}_4 \]

\[ I = \]

\[ a_1 \overline{a}_2 \quad \overline{a}_1 a_3 \quad | \quad \overline{a}_2 a_3 \quad a_2 a_4 \quad \overline{a}_4 \]

\[ \overline{a}_2 \overline{a}_3 \quad a_2 \quad \overline{a}_3 \quad a_3 \]

\[ A - \text{Hyp} \quad \frac{C}{[C]_B} \]

\[ B - \text{Hyp} \quad \frac{C}{[\top]} \]

\[ A - \text{Res} \quad \frac{C \lor x [I_1]}{C \lor D [I_1 \lor I_2]} \quad \frac{\overline{x} \lor D [I_2]}{C \lor D [I_1 \land I_2]} \]

\[ B - \text{Res} \quad \frac{C \lor x [I_1]}{C \lor D [I_1 \land I_2]} \quad \frac{\overline{x} \lor D [I_2]}{C \lor D [I_1 \land I_2]} \]
Applying Interpolating Proof Rules

\[
\begin{align*}
A &= (a_1 \lor a_2) \land (a_1 \lor \overline{a}_3) \land a_2 \\
B &= (\overline{a}_2 \lor a_3) \land (a_2 \lor a_4) \land \overline{a}_4
\end{align*}
\]

\[I = \]
Applying Interpolating Proof Rules

\[ A = (a_1 \lor \overline{a}_2) \land (\overline{a}_1 \lor a_3) \land a_2 \]
\[ B = (\overline{a}_2 \lor a_3) \land (a_2 \lor a_4) \land \overline{a}_4 \]
\[ I = \]

\[ a_1 \overline{a}_2 [\overline{a}_2] \quad a_1 \overline{a}_3 [\overline{a}_3] \quad \overline{a}_2 a_3 \quad a_2 a_4 \quad \overline{a}_4 \]

\[ \overline{a}_2 \overline{a}_3 \quad a_2 [a_2] \quad \overline{a}_3 \quad a_3 \]

\[ A\text{-Hyp} \quad C \quad [C|_B] \]
\[ B\text{-Hyp} \quad C \quad [\top] \]
\[ A\text{-Res} \quad C \lor x [I_1] \quad \overline{x} \lor D [I_2] \quad C \lor D [I_1 \lor I_2] \]
\[ B\text{-Res} \quad C \lor x [I_1] \quad \overline{x} \lor D [I_2] \quad C \lor D [I_1 \land I_2] \]
Example 2.

by these systems and that the interpolants from

Example 2 below shows that there are interpolants that cannot be obtained

Applying Interpolating Proof Rules

For an initial vertex

Let

\[ A = (a_1 \lor \overline{a_2}) \land (\overline{a_1} \lor \overline{a_3}) \land a_2 \]

\[ B = (\overline{a_2} \lor a_3) \land (a_2 \lor a_4) \land \overline{a_4} \]

\[ I = \]

\[ \overline{a_2} \overline{a_3} \]

\[ a_2 [a_2] \]

\[ a_2 \]

\[ a_3 \]

\[ \overline{a_3} \]

\[ a_1 \overline{a_2} \ [a_2] \]
Applying Interpolating Proof Rules

\[ a_1 \overline{a}_2 \quad \overline{a}_2 [a_2] \quad \overline{a}_1 \overline{a}_3 \quad \overline{a}_3 [a_3] \quad \overline{a}_2 a_3 [\top] \quad a_2 a_4 [\top] \quad \overline{a}_4 [\top] \]

\[ \overline{a}_2 \overline{a}_3 [\overline{a}_2 \lor \overline{a}_3] \quad a_2 [a_2] \quad a_3 \]

\[ I = (a_1 \lor \overline{a}_2) \land (\overline{a}_1 \lor \overline{a}_3) \land a_2 \]

\[ B = (\overline{a}_2 \lor a_3) \land (a_2 \lor a_4) \land \overline{a}_4 \]

\[ I = \]

\[ A-Hyp \quad \frac{C}{[C]_B} \]

\[ B-Hyp \quad \frac{C}{[\top]} \]

\[ A-Res \quad \frac{C \lor x [I_1]}{C \lor D [I_1 \lor I_2]} \quad \frac{\overline{x} \lor D [I_2]}{[I_1 \land I_2]} \]

\[ B-Res \quad \frac{C \lor x [I_1]}{C \lor D [I_1 \land I_2]} \quad \frac{\overline{x} \lor D [I_2]}{[I_1 \lor I_2]} \]
Example 2 by these systems and that the interpolants from \( \text{Itp} \) in the symmetric system in Figure 1(b). We have that partial interpolants in McMillan's system are shown in Figure 1(a) and those (Definition 4).

Applying Interpolating Proof Rules

\[ a_1 \overline{a}_2 \quad \overline{a}_1 a_3 \quad \overline{a}_2 a_3 \quad a_2 a_4 \quad \overline{a}_4 \]

\[ a_2 a_3 \quad \overline{a}_2 \overline{a}_3 \quad a_2 \quad \overline{a}_3 \quad a_3 \quad \square \]

\[ A = (a_1 \lor \overline{a}_2) \land (\overline{a}_1 \lor \overline{a}_3) \land a_2 \]

\[ B = (\overline{a}_2 \lor a_3) \land (a_2 \lor a_4) \land \overline{a}_4 \]

\[ I = \overline{a}_3 \land a_2 \]

\[ \begin{align*}
A-Hyp & \quad C \quad [C|_B] \\
B-Hyp & \quad C \quad [\top] \\
A-Res & \quad C \lor x \quad [I_1] \quad \overline{x} \lor D \quad [I_2] \\
B-Res & \quad C \lor x \quad [I_1] \quad \overline{x} \lor D \quad [I_2] 
\end{align*} \]
A Symmetric Construction

\[
\begin{align*}
&\text{A-Hyp} & \frac{C \quad [\bot]}{C \quad [C \in A]} & \quad \text{B-Hyp} & \frac{C \quad [\top]}{C \quad (C \in B)} \\
&\text{A-Res} & \frac{C \lor x \quad [I_1]}{C \lor D \quad [I_1 \lor I_2]} & \quad \text{B-Res} & \frac{C \lor x \quad [I_1]}{C \lor D \quad [I_1 \land I_2]} \\
&\text{AB-Res} & \frac{C \lor x \quad [I_1]}{C \lor D \quad [(x \lor I_1) \land (\overline{x} \lor I_2)]} & \quad (x \in \text{var}(B) \setminus \text{var}(A))
\end{align*}
\]

Huang 1995, Krajíček; Pudlák 1997
An Interpolant from the Symmetric Construction

\[ A = (a_1 \lor \overline{a}_2) \land (\overline{a}_1 \lor \overline{a}_3) \land a_2 \]
\[ B = (\overline{a}_2 \lor a_3) \land (a_2 \lor a_4) \land \overline{a}_4 \]
\[ I = \overline{a}_3 \]
What other constructions are there?

\[
\begin{align*}
A-\text{Hyp} & \quad \frac{C}{C} \quad B-\text{Hyp} \quad \frac{C}{C} \\
A-\text{Res} & \quad \frac{C \lor x \, [I_1]}{C \lor D} \quad \frac{\overline{x} \lor D \, [I_2]}{[I_1 \lor I_2]} \\
AB-\text{Res} & \quad \frac{C \lor x \, [I_1]}{C \lor D} \quad \frac{\overline{x} \lor D \, [I_2]}{[(x \lor I_1) \land (\overline{x} \lor I_2)]} \\
B-\text{Res} & \quad \frac{C \lor x \, [I_1]}{C \lor D} \quad \frac{\overline{x} \lor D \, [I_2]}{[I_1 \land I_2]}
\end{align*}
\]
What other constructions are there?

\[\begin{array}{ll}
A\text{-Hyp} & \frac{C}{\bot} \quad B\text{-Hyp} & \frac{C}{\top} \\
A\text{-Res} & \frac{C \lor x[I_1] \quad \overline{x} \lor D[I_2]}{C \lor D [I_1 \lor I_2]} \\
AB\text{-Res} & \frac{C \lor x[I_1] \quad \overline{x} \lor D[I_2]}{C \lor D [(x \lor I_1) \land (\overline{x} \lor I_2)]} \\
B\text{-Res} & \frac{C \lor x[I_1] \quad \overline{x} \lor D[I_2]}{C \lor D [I_1 \land I_2]} \\
\end{array}\]
Colours: $S \overset{\text{def}}{=} \{\emptyset, A, B, AB\}$

Coloured clauses: $C \rightarrow S$, a lattice under point-wise order.

Coloured CNF: Set of coloured clauses.
Deduction and Interpolation with Labels

Let $\sigma(x)$ be the colour of a literal $x$. 

$$C|_A = \{ x \in C \mid \sigma(x) \sqsubseteq A \}$$

\[\begin{array}{ll}
\text{A-Hyp} & \frac{C \ [C|_B]}{C \ [C|_B]} \quad C \in A \\
\text{B-Hyp} & \frac{C \ [C|_A]}{C \ [C|_A]} \quad C \in B \\
\text{A-Res} & \frac{C \lor x \ [I_1]}{C \lor D} \quad \frac{\overline{x} \lor D \ [I_2]}{[I_1 \lor I_2]} \\
\text{B-Res} & \frac{C \lor x \ [I_1]}{C \lor D} \quad \frac{\overline{x} \lor D \ [I_2]}{[I_1 \land I_2]} \\
\text{AB-Res} & \frac{C \lor x \ [I_1]}{C \lor D} \quad \frac{\overline{x} \lor D \ [I_2]}{[(x \lor I_1) \land (I_2 \lor \overline{x})]} \\
\end{array}\]

($\sigma(x) \sqcup \sigma(\overline{x}) = \mathbf{A}$) 

($\sigma(x) \sqcup \sigma(\overline{x}) = \mathbf{A}\mathbf{B}$) 

($\sigma(x) \sqcup \sigma(\overline{x}) = \mathbf{B}$)

D’Silva, Kroening, Purandare, Weissenbacher, 2010
Applying the Labelled Interpolation System

\[ A = \overline{a}_1 \land (a_1 \lor \overline{a}_2) \]
\[ B = a_1 \land (\overline{a}_1 \lor a_2) \]
\[ I = \overline{a}_2 \]

\[ I = I_1 \lor I_2 \text{ if } \sigma(x) \lor \sigma(\overline{x}) = A \]
\[ (x \lor I_1) \land (I_2 \lor \overline{x}) \text{ if } \sigma(x) \land \sigma(\overline{x}) = AB \]
\[ I_1 \land I_2 \text{ if } \sigma(x) \lor \sigma(\overline{x}) = B \]
Correctness

A colouring of $A \land B$ is locality preserving if
- Every literal in the formula has a non-empty colour,
- every literal occurring only in $A$ is coloured $A$, and
- every literal occurring only in $B$ is coloured $B$.

**Theorem.** If $A \land B$ is unsatisfiable and has a locality preserving colouring, $\Box [I]$ is derivable and $I$ an interpolant for $A$ and $B$.

It’s all in the colour

\[
\begin{align*}
A\text{-Hyp} & \quad \frac{C}{[\bot]} \quad \text{B-Hyp} \quad \frac{C}{[\top]} \\
A\text{-Res} & \quad \frac{C \lor x [I_1]}{C \lor D} \quad \frac{\overline{x} \lor D [I_2]}{[I_1 \lor I_2]} \\
AB\text{-Res} & \quad \frac{C \lor x [I_1]}{C \lor D} \quad \frac{\overline{x} \lor D [I_2]}{[(x \lor I_1) \land (\overline{x} \lor I_2)]} \\
B\text{-Res} & \quad \frac{C \lor x [I_1]}{C \lor D} \quad \frac{\overline{x} \lor D [I_2]}{[I_1 \land I_2]} \\
\end{align*}
\]
But why those constructions?

A colouring is *partitioning* if every instance of a variable has the same colour.

Abstraction: Every colouring is contained in a partitioning one. Different partitions define different abstract domains.
But why those constructions?

A colouring is *partitioning* if every instance of a variable has the same colour.

**Theorem.** There is a unique, coarsest partition that admits exactly three, locality preserving colourings.
The strength order is $B \subseteq AB \subseteq A$.

Coloured clauses and CNF are ordered pointwise by the strength order.

**Theorem.** The set of locality-preserving colourings forms a complete lattice with respect to the strength order.
Additional Analysis

• Colourings can be ordered by variable occurrence, which correlates loosely with interpolant size.

• There is a dual operation on the lattice of colours, which lifts pointwise so that every interpolation construction has a dual.

• Sharygina et al. proved results on labelled interpolation applied in the context of reachability analysis.

• Jhala and McMillan, 2006 and Albarghouthi and McMillan, 2013 study additional restrictions on the vocabulary condition.
<table>
<thead>
<tr>
<th></th>
<th>A Brief History of Interpolation</th>
</tr>
</thead>
<tbody>
<tr>
<td>2</td>
<td>Analysis with Interpolants</td>
</tr>
<tr>
<td>3</td>
<td>Labelled Interpolation Systems</td>
</tr>
<tr>
<td>4</td>
<td>Current and Future Directions</td>
</tr>
</tbody>
</table>
Architecture of a Modern Solver

This talk

Boolean Structure

Theory
Combination
Theory
EUF
Quantifiers
Equality Proofs

\[ f(u, y) = z \quad u = x \quad v = y \quad f(x, v) \neq z \]

- Deduced \textit{literals} may not be in A or in B
- New \textit{terms} may use non-shared symbols
- Interpolant may be over terms not in the proof

\[ A = u = x \land f(u, y) = z \]
\[ B = v = y \land f(x, v) \neq z \]
\[ I = f(x, y) = z \]
Coloured Congruence Graphs

\[ f(u, y) = z \quad u = x \quad v = y \quad f(x, v) \neq z \]

\[ f(x, y) = z \]

\[ f(x, v) = z \]

\[ \quad \]

\[ A = u = x \land f(u, y) = z \]

\[ B = v = y \land f(x, v) \neq z \]

\[ I = f(x, y) = z \]
Propositional Interpolants

<table>
<thead>
<tr>
<th>Year</th>
<th>Author(s)</th>
<th>Title</th>
</tr>
</thead>
<tbody>
<tr>
<td>1995</td>
<td>Huang</td>
<td>Constructing Craig Interpolation Formulas. (OTTER)</td>
</tr>
<tr>
<td>1997</td>
<td>Jan Krajiček</td>
<td>Interpolation theorems, lower bounds for proof systems, and independence results for bounded arithmetic.</td>
</tr>
<tr>
<td>1997</td>
<td>Pudlák</td>
<td>Lower Bounds for Resolution and Cutting Plane Proofs and Monotone Computations</td>
</tr>
<tr>
<td>2003</td>
<td>McMillan</td>
<td>Interpolation and SAT-Based Model Checking.</td>
</tr>
<tr>
<td>2006</td>
<td>Yorsh, Musuvathi</td>
<td>A Combination Method for Generating Interpolants.</td>
</tr>
<tr>
<td>2009</td>
<td>Biere</td>
<td>Bounded Model Checking (in Handbook of Satisfiability).</td>
</tr>
<tr>
<td>2010</td>
<td>D. Kroening, Purandare, Weissenbacher</td>
<td>Interpolant Strength.</td>
</tr>
</tbody>
</table>
## Equality Interpolants

<table>
<thead>
<tr>
<th>Year</th>
<th>Author(s)</th>
<th>Reference</th>
</tr>
</thead>
<tbody>
<tr>
<td>1996</td>
<td>Fitting, First-Order Logic and Automated Theorem Proving</td>
<td></td>
</tr>
<tr>
<td>2005</td>
<td>McMillan, An Interpolating Theorem Prover</td>
<td></td>
</tr>
<tr>
<td>2006</td>
<td>Yorsh, Musuvathi, A Combination Method for Generating Interpolants.</td>
<td></td>
</tr>
<tr>
<td>2009</td>
<td>Fuchs, Goel, Grundy, Krstic, Tinelli, Ground Interpolation for the Theory of Equality.</td>
<td></td>
</tr>
<tr>
<td>2014</td>
<td>Bonacina, Johansson, Interpolation Systems for Ground Proofs in Automated Reasoning</td>
<td></td>
</tr>
</tbody>
</table>
## Interpolation in Theories

<table>
<thead>
<tr>
<th>Year</th>
<th>Authors</th>
<th>Title</th>
<th>Theories</th>
</tr>
</thead>
<tbody>
<tr>
<td>2005</td>
<td>McMillan</td>
<td>Interpolating Theorem Prover</td>
<td>LA(Q)</td>
</tr>
<tr>
<td>2006</td>
<td>Kapur, Majumdar, Zarba</td>
<td>Interpolation for Data Structures</td>
<td>Datatype theories</td>
</tr>
<tr>
<td>2007</td>
<td>Rybalchenko, Sofronie-Stokkermans</td>
<td>Constraint Solving for Interpolation</td>
<td>LA(Q)</td>
</tr>
<tr>
<td>2008</td>
<td>Cimatti, Griggio, Sebastiani</td>
<td>Efficient Interpolant Generation in Satisfiability Modulo Theories</td>
<td>LA(Q), DL(Q), UTVPI</td>
</tr>
<tr>
<td>2008</td>
<td>Jain, Clarke, Grumberg</td>
<td>Efficient Craig Interpolation for Linear Diophantine (dis)Equations and Linear Modular Equations</td>
<td>LDE, LME</td>
</tr>
<tr>
<td>2009</td>
<td>Cimatti, Griggio, Sebastiani</td>
<td>Interpolant Generation for UTVPI</td>
<td>UTVPI</td>
</tr>
<tr>
<td>2011</td>
<td>Griggio</td>
<td>Effective Word-Level Interpolation for Software Verification</td>
<td>Bit-Vectors</td>
</tr>
</tbody>
</table>
## Interpolation in Theory Combinations

<table>
<thead>
<tr>
<th>Year</th>
<th>Paper Title</th>
<th>Method(s)</th>
</tr>
</thead>
<tbody>
<tr>
<td>2005</td>
<td>McMillan. Interpolating Theorem Prover</td>
<td>LA(Q) over EUF over Bool</td>
</tr>
<tr>
<td>2005</td>
<td>Yorsh and Musuvathi, A Combination Method for Generating Interpolants</td>
<td>Nelson-Oppen</td>
</tr>
<tr>
<td>2009</td>
<td>Cimatti, Griggio, Sebastiani, Efficient Generation of Craig Interpolants</td>
<td>Delayed Theory Combination</td>
</tr>
<tr>
<td>2009</td>
<td>Goel, Krstic, Tinelli, Ground Interpolation for Combined Theories</td>
<td>Proof transformation</td>
</tr>
<tr>
<td>2012</td>
<td>Kovacs, Voronkov, Playing in the Gray Area of Proofs</td>
<td>Proof Transformation</td>
</tr>
</tbody>
</table>