Interpolation: Theory and Applications

Vijay D’Silva
Google Inc., San Francisco

UC Berkeley 2016
<table>
<thead>
<tr>
<th></th>
<th>Section Title</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>A Brief History of Interpolation</td>
</tr>
<tr>
<td>2</td>
<td>Verification with Interpolants</td>
</tr>
<tr>
<td>3</td>
<td>Interpolant Construction</td>
</tr>
<tr>
<td>4</td>
<td>Further Reading and Research</td>
</tr>
</tbody>
</table>
Craig Interpolants

For two formulae $A$ and $B$ such that $A$ implies $B$, a Craig interpolant is a formula $I$ such that

1. $A$ implies $I$, and
2. $I$ implies $B$, and
3. the non-logical symbols in $I$ occur in $A$ and in $B$.

$P \land (P \implies Q) \implies Q \implies R \implies Q$
Reverse Interpolants or “Interpolants”

For a contradiction $A \land B$, a reverse interpolant is a formula $I$ such that

1. $A$ implies $I$, and
2. $I \land B$ is a contradiction, and
3. the non-logical symbols in $I$ occur in $A$ and in $B$.

Note: In classical logic, $A \implies B$ is valid exactly if $A \land \neg B$ is a contradiction.

In the verification literature, “interpolant” usually means “reverse interpolant.”
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 correctness 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
<table>
<thead>
<tr>
<th></th>
<th>A Brief History of Interpolation</th>
</tr>
</thead>
<tbody>
<tr>
<td>2</td>
<td>Verification with Interpolants</td>
</tr>
<tr>
<td>3</td>
<td>Interpolant Construction</td>
</tr>
<tr>
<td>4</td>
<td>Further Reading and Research</td>
</tr>
</tbody>
</table>
Craig’s Interpolation Lemma (1957)

The Journal of Symbolic Logic
Volume 22, Number 3, Sept. 1957

LINEAR REASONING.
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 struc-
tures of A and A’, whenever A implies A’ (i.e., A ⊨ 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

The Journal of Symbolic Logic
Volume 22, Number 3, Sept. 1957

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 modeltheoretic concepts to more elementary proof-
theoretic concepts, thereby opening up modeltheoretic problems to proof-
theoretic methods of attack. Herbrand’s Theorem (see [8] or also [9],

William Craig in 1988
http://sophos.berkeley.edu/interpolations/
Lemma. First-order logic has the interpolation property. That is, if $A \implies B$ is valid, then a Craig interpolant exists.
A High-Level View of the Proof

“The intuitive idea for Craig’s proof of the Interpolation Theorem rests on the completeness theorem for FOL, in the form of the equivalence of validity with provability in a suitable system of axioms and rules of inference. By “suitable” here is meant one in which there is a notion of a direct proof for which if Phi implies Psi is provable then there is a direct proof of Psi from Phi. One would expect that in such a proof, the relation symbols of Phi that are in Psi would not disappear in the middle. Such systems were devised by Herbrand (1930) and Gentzen (1934); Hilbert-style systems enlarged by the axioms and rules of the epsilon-calculus can also serve this purpose.

Feferman, Harmonious Logic: Craig’s Interpolation Theorem and Its Descendants, 2008
Another Perspective on the Interpolation Theorem

“Important results have many things to say. At first sight, the Interpolation Theorem of Craig (1957) seems a rather technical result for connoisseurs inside logical meta-theory. But over the past decades, its broader importance has become clear from many angles. In this paper, I discuss my own current favourite views of interpolation: no attempt is made at being fair or representative. First, I discuss the entanglement of inference and vocabulary that is crucial to interpolation. Next, I move to the role of interpolants in facilitating generalized inference across different models. Then I raise the perhaps surprising issue of ‘what is the right formulation of Craig’s Theorem?’, high-lighting the existence of non-trivial options in formulating meta-theorems.

... 

Finally, I discuss the ‘end of history’. Craig’s Theorem is about the last significant property of first-order logic that has come to light. Is there something deeper going on here, and if so, can we prove it?”

-- Johan van Benthem, The Many Faces of Interpolation, 2008
Interpolation In Mathematical Logic

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

- Model theoretic characterizations: Makowsky ’85 for a survey.
- Amalgamation and algebraic characterization

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


1971 1971, Cook. The Complexity of Theorem Proving Procedures

Proof Content. For any language $A \in \text{NP}$ and $n \in \mathbb{N}$, one can construct in polynomial time a formula

$$F_n(x_1, \ldots, x_n, y_1, \ldots, y_p(n))$$

in propositional logic such that for all $x \in \{0, 1\}^n$

$$x \in A \iff \exists y. F_n(x, y) = \text{true}$$
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 $n$. 
Interpolation and (Proof) Complexity Theory

Combinatorial description of how information flows in a proof. Interpolants eliminate certain flows and preserve others. The “relevant” information is preserved.
## Interpolants in Automated Reasoning

<table>
<thead>
<tr>
<th>Year</th>
<th>Publication</th>
</tr>
</thead>
<tbody>
<tr>
<td>1957</td>
<td>Huang, Constructing Craig Interpolation Formulas. (OTTER)</td>
</tr>
<tr>
<td>1960</td>
<td>[References]</td>
</tr>
<tr>
<td>1970</td>
<td>[References]</td>
</tr>
<tr>
<td>1980</td>
<td>[References]</td>
</tr>
<tr>
<td>1990</td>
<td>[References]</td>
</tr>
<tr>
<td>2000</td>
<td>McMillan, An Interpolating Theorem Prover</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>
</tbody>
</table>

Wednesday, March 30, 16
<table>
<thead>
<tr>
<th></th>
<th>A Brief History of Interpolation</th>
</tr>
</thead>
<tbody>
<tr>
<td>2</td>
<td>Verification with Interpolants</td>
</tr>
<tr>
<td>3</td>
<td>Interpolant Construction</td>
</tr>
<tr>
<td>4</td>
<td>Further Reading and Research</td>
</tr>
</tbody>
</table>
Systems Analysis with SAT/SMT Solvers

- Bounded Model Checking and Symbolic Execution generate formulae encoding bounded executions.
- Can we generate invariants?
- Can we explore deeper executions without running out of memory?
- Can we avoid exploring redundant system behaviours?
A Simple Binary Counter

Initial State

\[ J(a) \overset{\text{def}}{=} (\overline{a}_2 \land \overline{a}_1 \land \overline{a}_0) \]

\[ T(a, a') \overset{\text{def}}{=} (\overline{a}_2 \land \overline{a}_1 \Rightarrow \overline{a}_2' \land a_1') \land \]

\[ (\overline{a}_2 \land a_1 \Rightarrow a_2' \land \overline{a}_1') \land \]

\[ (a_2 \land \overline{a}_1 \Rightarrow \overline{a}_2' \land \overline{a}_1') \land \]

\[ (a_2 \land a_1 \Rightarrow a_2' \land a_1') \land \]

\[ (a_2' \land a_1' \lor \overline{a}_2' \land \overline{a}_1' \lor a_2' \land \overline{a}_1') \Rightarrow \overline{a}_0' \]

Transition Relation

Reachability Condition

\[ F(a) \overset{\text{def}}{=} a_0 \land (\overline{a}_1 \lor \overline{a}_2) \]
A Refutation and Its Interpolant

The interpolant, in this case is the image. In general, interpolants for an appropriately constructed formula are overapproximations of images.
Reachability with Interpolants

Algorithm: Compute images till a fixed point is reached.

Image of $Qx$: $\exists x : Q(x) \land T(x, x')$
Interpolation Slogans

- A poor person’s quantifier elimination
- A separator between two regions of a search space
- A summary of why a bounded-property holds occurs
- An approximate image operator
- A relevance heuristic that articulates the core reason for a proof
Bounded Execution as a Formula

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

```c
int x0 = i;
int y0 = j;
int x1 = y0 + 1;
y1 = x0 + 1;
x2 = y1 + 1;
y2 = x1 + 1;
x3 = y2 + 1;
y3 = x2 + 1;
if (i == j &&
    x3 <= 10) {
    if (y3 > 10)
        Err:// ERROR
    REACHED
}  
```
Bounded Execution and Interpolants

```c
int x0 = i;
int y0 = j;
x1 = y0 + 1;
y1 = x0 + 1;
x2 = y1 + 1;
y2 = x1 + 1;
x3 = y2 + 1;
y3 = x2 + 1;
if (i == j && x3 <= 10) {
    if (y3 > 10)
        Err:// ERROR
    REACHED
}
```

\[ x_2 = i + 2 \land y_2 = j + 2 \]

- Symbolic representation of the states reachable after two iterations
- Image computation for program statements typically requires quantifier elimination in that theory.
Another Interpolant

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

- Potential loop invariant
- Invariant computation often requires fixed point computation, quantifier elimination, or even both.
A Space of Interpolants

Multiple interpolants exist.
• They differ in size, logical strength, symbols, etc.
• The ideal one depends on the problem
Approximation of States with Interpolants

The challenge is to encode these constraints and respect the vocabulary condition
Abstract Reachability Tree Construction

L = 0;
\textbf{do} { \textbf{assert} (L==0); } \textbf{lock}()
L = 1;
old = new;
\textbf{if} (*){
L = 0;
} \textbf{unlock}()
new++;
}\textbf{while} (new!=old);

Example: McMillan 2006, Graphic Ruemmer ’14
Sequence Interpolants from Reachability Tree

Example: McMillan 2006, Graphic Ruemmer '14
Abstract Reachability Construction

• Interpolants Decorate Positions on the Reachability Tree

• They denote state that are reached at those points

• A covering check is used to determine if all states at some location have been visited

• More complicated than predicate abstraction or fixed point computation due to non-monotonicity of interpolant construction
Dynamic System (Circuit, Program, Hybrid System)

Analyzer = Constraint Generation

Solver = Constraint Solving + Interpolant Construction

Property Checking with Interpolants
Property Checking with Interpolants

- Data Types
- Conditionals/Assignments
- Loops
- Recursion
- Path Sharing
- Generalization
- (Relative) Completeness
- Boolean Structure
  - Theory
  - Combination
  - EUF
  - Quantifiers

Wednesday, March 30, 16
Generalizations of Classical Interpolants

(Restricted) DAG interpolation

Disjunctive interpolation

Tree interpolation

Inductive interpolant sequences

Binary interpolation
<table>
<thead>
<tr>
<th></th>
<th>A Brief History of Interpolation</th>
</tr>
</thead>
<tbody>
<tr>
<td>2</td>
<td>Verification with Interpolants</td>
</tr>
<tr>
<td>3</td>
<td>Interpolant Construction</td>
</tr>
<tr>
<td>4</td>
<td>Further Reading and Research</td>
</tr>
</tbody>
</table>
Dynamic System

Analyzer = Constraint Generation

Property Checking with Interpolants
Flows in Resolution Proofs

- Literals flow around in the proof
- Literals with opposite polarity cancel each other
- Propositional interpolant construction can be viewed as controlling initial inputs and gating the flows using a circuit
- Want to let shared variables flow through the A-part and restrict flow from the B-part.
Interpolating Proof Rules

Split rules based on vocabulary

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

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

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

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

Annotate formulae with Partial Interpolants
Applying Interpolating Proof Rules

\[ A = (a_1 \lor \overline{a}_2) \land (\overline{a}_1 \lor \overline{a}_3) \land a_2 \]
\[ B = (\overline{a}_2 \lor a_3) \land (a_2 \lor a_4) \land \overline{a}_4 \]
\[ I = \]

\[ \begin{align*}
A\text{-Hyp} & \quad \frac{C}{[C|_B]} \\
B\text{-Hyp} & \quad \frac{C}{[T]} \\
A\text{-Res} & \quad \frac{C \lor x [I_1]}{C \lor D [I_1 \lor I_2]} \\
B\text{-Res} & \quad \frac{C \lor x [I_1]}{C \lor D [I_1 \land I_2]} 
\end{align*} \]
Applying Interpolating Proof Rules

\[
\begin{align*}
A &= (a_1 \lor \overline{a}_2) \land (\overline{a}_1 \lor \overline{a}_3) \land a_2 \\
B &= (\overline{a}_2 \lor a_3) \land (a_2 \lor a_4) \land \overline{a}_4 \\
I &= \overline{a}_3 \land a_2
\end{align*}
\]

\[
\begin{align*}
a_1 \overline{a}_2 \quad \overline{a}_2 \quad \overline{a}_1 \overline{a}_3 \quad \overline{a}_3 \quad \overline{a}_2 a_3 \quad a_2 a_4 \quad \overline{a}_4
\end{align*}
\]

\[
\begin{align*}
\overline{a}_2 \overline{a}_3 \quad \overline{a}_2 \lor \overline{a}_3 \quad a_2 \quad a_2 \quad a_2 \quad a_3 \quad a_3
\end{align*}
\]

\[
\begin{align*}
\overline{a}_3 \quad \overline{a}_3 \land a_2 \quad a_3 \quad \overline{a}_3 \land a_2
\end{align*}
\]

\[
\begin{align*}
A-Hyp & \quad C \quad [C|_B] \\
B-Hyp & \quad C \quad [T] \\
A-Res & \quad \frac{C \lor x [I_1]}{C \lor D} \quad \frac{\overline{x} \lor D [I_2]}{[I_1 \lor I_2]} \\
B-Res & \quad \frac{C \lor x [I_1]}{C \lor D} \quad \frac{\overline{x} \lor D [I_2]}{[I_1 \lor I_2]}
\end{align*}
\]
McMillan’s Interpolation System

\[
\begin{align*}
A\text{-Hyp} & : C \left[ \{ \ell \mid \text{var}(\ell) \in \text{var}(B) \} \right] (C \in A) \\
B\text{-Hyp} & : C \left[ T \right] (C \in B) \\
A\text{-Res} & : \frac{C \lor x [I_1]}{C \lor D } \frac{\bar{x} \lor D [I_2]}{[I_1 \lor I_2]} (x \in \text{var}(A) \setminus \text{var}(B)) \\
B\text{-Res} & : \frac{C \lor x [I_1]}{C \lor D } \frac{\bar{x} \lor D [I_2]}{[I_1 \land I_2]} (x \in \text{var}(B))
\end{align*}
\]

**Theorem.** The partial interplant labelling the empty clause is an interplant for \( A \) and \( B \).
Program = Control + Data

Analyzer = Constraint Generation

Property Checking with Interpolants

Boolean Structure
- Theory
- Combination
- EUF
- Quantifiers
Equality Proofs

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

- Deduced literals may not be in A or in B
- New 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

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

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

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

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

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

\[ I = f(x, y) = z \]
Interpolation from Coloured Congruence Graphs

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

- Modify graph to be colourable
- Take summaries A-paths by endpoints that are over the shared vocabulary
- Summarize B-paths as premises for Asummaries
Interpolation with B-Premises

- Mixes A and B reasoning
- Endpoints of B-reasoning paths are antecedents of implications for A
- Implication introduced by combination of congruence and shared reasoning

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

\[ B = u = x \land v \neq y \]

\[ I = u = x \implies v = y \]
<table>
<thead>
<tr>
<th></th>
<th>A Brief History of Interpolation</th>
</tr>
</thead>
<tbody>
<tr>
<td>2</td>
<td>Verification with Interpolants</td>
</tr>
<tr>
<td>3</td>
<td>Interpolant Construction</td>
</tr>
<tr>
<td>4</td>
<td>Further Reading and Research</td>
</tr>
</tbody>
</table>
Interpolation and SMT

<table>
<thead>
<tr>
<th>Year</th>
<th>Reference</th>
</tr>
</thead>
<tbody>
<tr>
<td>1957</td>
<td>Huang, Constructing Craig Interpolation Formulas. (OTTER)</td>
</tr>
<tr>
<td>1960</td>
<td></td>
</tr>
<tr>
<td>1970</td>
<td></td>
</tr>
<tr>
<td>1980</td>
<td></td>
</tr>
<tr>
<td>1990</td>
<td></td>
</tr>
<tr>
<td>2000</td>
<td></td>
</tr>
<tr>
<td>2010</td>
<td></td>
</tr>
<tr>
<td>2005 to present</td>
<td>Interpolation for theories: numeric, bit-vectors, strings, arrays, etc.</td>
</tr>
<tr>
<td></td>
<td>Interpolation for equality and theory combinations.</td>
</tr>
<tr>
<td></td>
<td>Quantified interpolants.</td>
</tr>
<tr>
<td></td>
<td>Sequence, tree and DAG interpolants.</td>
</tr>
</tbody>
</table>

Solver = Constraint Solving + Interpolant Construction
Analyzer = Constraint Generation
Program = Control + Data

Wednesday, March 30, 16
## Analysis with Interpolants

<table>
<thead>
<tr>
<th>Year</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>2003</td>
<td>McMillan, Interpolation and SAT-Based Model Checking.</td>
</tr>
<tr>
<td>2004</td>
<td>Henzinger, Jhala, Majumdar, McMillan, Abstraction from Proofs.</td>
</tr>
<tr>
<td>2006</td>
<td>McMillan, Lazy Abstraction with Interpolants</td>
</tr>
<tr>
<td>2009</td>
<td>Vizel, Grumberg, Interpolation-Sequence Based Model Checking</td>
</tr>
<tr>
<td>2010</td>
<td>Heizmann, Hoenicke, Podelski, Nested Interpolants</td>
</tr>
<tr>
<td>2012</td>
<td>Albarghouthi, Gurfinkel, Chechik, Whale: An Interpolation-Based Algorithm for Inter-Procedural Verification</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Frameworks</th>
<th>2006 onwards</th>
</tr>
</thead>
<tbody>
<tr>
<td>Loops</td>
<td>Interpolant Sequence</td>
</tr>
<tr>
<td>Recursion</td>
<td>Tree Interpolant</td>
</tr>
<tr>
<td>Multiple Paths</td>
<td>DAG Interpolant</td>
</tr>
<tr>
<td>Frameworks</td>
<td>Horn Clauses</td>
</tr>
</tbody>
</table>
## Analysis of Interpolants

<table>
<thead>
<tr>
<th>Year</th>
<th>Authors</th>
<th>Title</th>
<th>Logic Type</th>
</tr>
</thead>
<tbody>
<tr>
<td>1957</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>1960</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>1970</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>1980</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>1990</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>2000</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>2010</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>2012</td>
<td>Jhala, McMillan</td>
<td>A Practical and Complete Approach to Predicate Refinement.</td>
<td>Theory Independent</td>
</tr>
<tr>
<td>2012</td>
<td>D. Kroening, Purandare, Weissenbacher</td>
<td>Interpolant Strength.</td>
<td>Propositional Logic</td>
</tr>
<tr>
<td>2012</td>
<td>Rollini, Sery, Sharygina</td>
<td>Leveraging Interpolant Strength in Model Checking.</td>
<td></td>
</tr>
<tr>
<td>2012</td>
<td>Alberti, Brutomesso, Ghilardi, Ranise, Sharygina</td>
<td>Lazy Abstraction with Interpolants for Arrays.</td>
<td></td>
</tr>
<tr>
<td>2013</td>
<td>Albarghouthi, McMillan</td>
<td>Beautiful Interpolants.</td>
<td></td>
</tr>
<tr>
<td>2013</td>
<td>Ruemmer, Subotic</td>
<td>Exploring Interpolants.</td>
<td></td>
</tr>
</tbody>
</table>

**Program = Control + Data**

**Solver = Constraint Solving + Interpolant Construction**

**Analyzer = Constraint Generation**

**Solver = Constraint Solving + Interpolant Construction**
### Further Reading: 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 Krajíček</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>
## Further Reading: Equality Interpolants

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

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