Interpolation Algorithms
and their
Applications in Model Checking

Georg Weissenbacher
www.georg.weissenbacher.name
Literature (Open Access)

Boolean Satisfiability Solvers and Their Applications in Model Checking
with Yakir Vizel and Sharad Malik
Proceedings of the IEEE, Nov. 2015
http://dx.doi.org/10.1109/JPROC.2015.2455034

Labelled Interpolation Systems for Hyper-Resolution, Clausal, and Local Proofs
with Matthias Schlaipfer
http://dx.doi.org/10.1007/s10817-016-9364-6

www.georg.weissenbacher.name/publications.html
Part I: Model Checking

Boolean Satisfiability Solvers and Their Applications in Model Checking

with Yakir Vizel and Sharad Malik
Proceedings of the IEEE, Nov. 2015
Software and integrated circuits are everywhere
Software and integrated circuits are everywhere

$10^6$ lines of code 70 micro-processors
Basics of
Symbolic Model Checking
Logic
T
(transitions)
\( T: \) operational semantics of program or circuit

The Model Checking problem:

- "starting states"
- "bad states"
The Model Checking problem:

"starting states" $\neg P$ "bad states"
The Model Checking problem:

- "starting states": $\langle pc \mapsto 2, x \mapsto 1 \rangle$
- "bad states": $\langle pc \mapsto 3, x \mapsto 2 \rangle$

$T$: operational semantics of program or circuit
The Model Checking problem:

I "starting states" ¬ P "bad states"

\[ \langle pc \mapsto 2, x \mapsto 1 \rangle \quad \langle pc \mapsto 3, x \mapsto 2 \rangle \]

(\( T \): operational semantics of program or circuit)
(\(T\): operational semantics of program or circuit)

The **Model Checking** problem:

- "starting states"  
- "bad states"
\[ T(\langle pc \mapsto 2, x \mapsto 1 \rangle, \langle pc \mapsto 3, x \mapsto 2 \rangle) \]

(The Model Checking problem:

\[ I \xrightarrow{T} \neg P \]

“starting states”

“bad states”

\( T \): operational semantics of program or circuit)
The Model Checking problem:

$T$ (operational semantics of program or circuit)

"starting states" \[ \langle pc \mapsto 2, x \mapsto 1 \rangle \]

"bad states" \[ \langle pc \mapsto 3, x \mapsto 2 \rangle \]
The Model Checking problem:

(T: operational semantics of program or circuit)

"starting states"  "bad states"
The **Model Checking** problem:
The Model Checking problem:

\[ \langle pc \mapsto 2, x \mapsto 1 \rangle \quad \langle pc \mapsto 3, x \mapsto 2 \rangle \]

\((T: \text{operational semantics of program or circuit})\)
State Space Explosion
Why explore states one by one?
Why explore states one by one?

Set of states $S$ and post-image $S'$ related by $T$.
Why explore states one by one?

\[ S' = T(S) \overset{\text{def}}{=} \{ s' \mid T(s, s') \land s \in S \}\]
How do we efficiently represent sets of states?

Logical Formulas!

$\forall$

program variables, registers, latches, signals, ...
How do we efficiently represent sets of states?

Logical Formulas!

$$F(V)$$

program variables, registers, latches, signals, ...
How do we efficiently represent sets of states?

**Logical Formulas!**

\[(x > 0) \text{ represents } \{s \mid s(x) > 0\}\]
And what about transitions?

Binary Relations!

\[ T(V, V') \]

target states
And what about transitions?

**Binary Relations!**

\[(x' = x + 1) \text{ represents } \{ \langle s, s' \rangle | s'(x) = s(x) + 1 \}\]
And what about transitions?

Binary Relations!

\[ (x' = x + 1) \quad \iff \quad \{ \langle s, s' \rangle \mid s'(x) = s(x) + 1 \} \]
\[ R \]

\[ R' \]

\[ T - 1 \]

\[ R' \]

\[ V' \]

\[ \exists V. R(V) \land T(V, V') \]

\[ \exists V'. T(V, V') \land R'(V') \]
\[ R'(V') \overset{\text{def}}{=} \exists V. \quad R(V) \land T(V, V') \]
\[ R'(V') \overset{\text{def}}{=} \exists V \cdot R(V) \land T(V, V') \]
\[ R(V) \overset{\text{def}}{=} \exists V' \cdot T(V, V') \land R'(V') \]
T

(transition relation)
if (x>0) {
    x = x - 1;
} else {
    x = x + 1;
}

T

(transition relation)
1: if (x>0) {
2:     x = x - 1;
3: } else {
4:     x = x + 1;
5: }

(transition relation)
\[
T(V, V') = Q' \iff (x \land Q) \land (z \iff (y \lor Q))
\]

\[
P(V) = z
\]

\[
I(V) = Q
\]
$$T(V, V') \overset{\text{def}}{=} (Q' \leftrightarrow (x \land Q)) \land (z \leftrightarrow (y \lor Q))$$
\[
T(V, V') \overset{\text{def}}{=} (Q' \leftrightarrow (x \land Q)) \land (z \leftrightarrow (y \lor Q))
\]

\[
P(V) \overset{\text{def}}{=} z
\]

\[
l(V) \overset{\text{def}}{=} Q
\]
Can property $P$ be violated in $k$ steps?
(here, property = assertion over variables)
$T$ cannot escape \( R_1 \) if $T$ cannot escape \( R_1 \) if $T$ cannot escape \( R_1 \) if $T$ cannot escape $R_1$.
\( T(R_{i-1}) = R_i \) \((1 \leq i \leq k, \text{with } R_0 \overset{\text{def}}{=} I)\)
$T(R_{i-1}) = R_i \quad (1 \leq i \leq k, \text{with } R_0 \overset{\text{def}}{=} I)$
\[ T(R_{i-1}) = R_i \quad (1 \leq i \leq k, \text{with } R_0 \overset{\text{def}}{=} I) \]

\[ R_{\leq k} = \bigcup_{i=0}^{k} R_i \]
$T(R_{i-1}) = R_i \quad (1 \leq i \leq k, \text{with } R_0 \overset{\text{def}}{=} I)$

$$R_{\leq k} = \bigcup_{i=0}^{k} R_i$$

- “Fixed point” if $T$ cannot escape $R_{\leq k}$
System is safe if:

\[ R_{\leq k} \]
System is safe if:

- $R_{\leq k}$ contains $I$
System is safe if:

- $R_{\leq k}$ contains $I$
- $T$ cannot leave $R_{\leq k}$
System is safe if:

- $R_{\leq k}$ contains $I$
- $T$ cannot leave $R_{\leq k}$
- $R_{\leq k}$ does not overlap with $\neg P$
System is safe if:

- $R_{\leq k}$ contains $I$
- $T$ cannot leave $R_{\leq k}$
- $R_{\leq k}$ does not overlap with $\neg P$

$R_{\leq k}$ challenging to compute (see next slide)
Convergence and Fixpoint Detection

- Exact reachability retards convergence:

\[ R_{k+1} \subseteq R_{\leq k} \]
Convergence and Fixpoint Detection

- Exact reachability retards convergence:
  \[ R_{k+1} \subseteq R_{\leq k} \]

- Boolean encoding requires quantifier alternation (\(\forall \exists\)):
  \[ R_{i+1}(V') \overset{\text{def}}{=} \exists V . R_i(V) \land T(V, V'), \quad R_{\leq k} = \bigvee_{i=0}^{k} R_i(V) \]
Convergence and Fixpoint Detection

- Exact reachability retards convergence:
  \[ R_{k+1} \supseteq R_{\leq k} \]

- Boolean encoding requires quantifier alternation (\(\forall \exists\)):
  
  \[ R_{i+1}(V') \overset{\text{def}}{=} \exists V . R_i(V) \land T(V, V') \]
  
  \[ R_{\leq k} = \bigvee_{i=0}^{k} R_i(V) \]

\[ \forall V . R_{k+1}(V) \Rightarrow R_{\leq k}(V) \]
Exact reachability retards convergence

Boolean encoding requires quantifier alternation
Convergence and Fixpoint Detection

- Exact reachability retards convergence
- Boolean encoding requires quantifier alternation
- **Over-approximate** $R_i$ instead?
Approximate inductive frames $F_i$:

- $F_0(V) = I(V)$
- $F_{i-1}(V) \land T(V, V') \Rightarrow F_i(V')$, for $1 \leq i \leq k$
Approximate inductive frames $F_i$: 

- $F_0(V) = I(V)$
- $F_{i-1}(V) \land T(V, V') \Rightarrow F_i(V')$, for $1 \leq i \leq k$

Fixpoint:

- $F_{k+1}(V) \Rightarrow \bigvee_{i=0}^{k} F_i(V)$
- Ideally, $F_i$ should be *quantifier-free*
Interpolation-based Model Checking
Craig’s Interpolation Theorem

If $A(V, V') \land B(V', V'') \Rightarrow \bot$ then there exists $C(V')$ such that $A(V, V') \Rightarrow C(V')$ and $B(V', V'') \Rightarrow \neg C(V')$. 
Craig’s Interpolation Theorem

\[ \text{Given } A(R,S) \Rightarrow B(S) \]

\[ \text{To find: } B(S), \quad \text{if } A(R,S) = B(S) \Rightarrow C \]

\[ \text{then } \exists C(V'') \text{ s.t. } A(V', V'') \Rightarrow C(V'') \quad \text{and } B(V', V'') \Rightarrow \neg C(V'') \]

C “simpler” than A
Craig’s Interpolation Theorem

if \((A(V, V') \land B(V', V'') \Rightarrow \bot)\)
then
\(\exists C(V')\)

s.t.
\(A(V, V') \Rightarrow C(V')\)
\(B(V', V'') \Rightarrow \neg C(V')\)
Interpolation-based Over-Approximation
Interpolation-based Over-Approximation

\[ I(V) \land T(V, V') \land \neg P(V') \]
Interpolation-based Over-Approximation

\[ I(V) \wedge T(V, V') \quad \neg P(V') \]

\[ I(V) \wedge T(V, V') \quad \neg P(V') \]

\[ A(V, V') \quad B(V') \]
Interpolation-based Over-Approximation

\[ I(V) \land T(V, V') \land \neg P(V') \]

\[ I(V) \land T(V, V') \land \neg P(V') \]

\[ F_1(V') \]

\[ A(V, V') \]

\[ B(V') \]
Interpolation-based Over-Approximation

\[ I(V) \land T(V, V') \land \neg P(V') \]

\[ F_1(V') \]
Interpolation-based Over-Approximation

\[ I(V) \land T(V, V') \Rightarrow F_1(V') \]

\[ F_1(V') \text{ over-approximates } R_1(V') \]

\[ F_1(V') \Rightarrow P(V) \]

safe approximation
Interpolation-based Over-Approximation

\[ I(V) \land T(V, V') \Rightarrow F_1(V') \]

\[ F_1(V') \text{ over-approximates } R_1(V') \]

\[ F_1(V') \Rightarrow P(V) \]

safe approximation

- Craig’s theorem guarantees existence
- Part I of tutorial: algorithms to construct (quantifier-free) \( F_1 \)
Interpolation-based Over-Approximation

Craig’s theorem guarantees existence

- Part I of tutorial: algorithms to construct (quantifier-free) $F_1$
  - Note: $R_1$ is strongest interpolant, but not quantifier-free
Interpolation-based Over-Approximation

\[ F(V) \overset{?}{\Rightarrow} I(V) \]

fixpoint check

If the fixpoint check succeeds, the system is safe. Otherwise, the first step is safe.
Interpolation-based Over-Approximation

System is safe if fixpoint check succeeds

Otherwise: first step is safe

\[ (\bigvee_{i=0}^{1} F_i(V)) \Rightarrow P(V) \]
Interpolation-based Over-Approximation

\[
\left( \bigvee_{i=0}^{1} F_i(V) \right) \Rightarrow P(V)
\]

- What about the next step?
Interpolation-based Over-Approximation

\[ I \lor F_1 \]

\[ \neg P \]

\[
\left( \bigvee_{i=0}^{1} F_i(V) \right) \Rightarrow P(V)
\]

What about the next step?

\[
\left( \bigvee_{i=0}^{1} F_i(V) \right) \land T(V, V')
\]
Interpolation-based Over-Approximation

- $(I \lor F_1)$ over-approximates states reachable in up to 1 step
Interpolation-based Over-Approximation

- $(I \lor F_1)$ over-approximates states reachable in up to 1 step
- Check whether $P$ can be violated within 1 additional step
Interpolation-based Over-Approximation

- \((I \lor F_1)\) over-approximates states reachable in up to 1 step
- Check whether \(P\) can be violated within 1 additional step
  - If yes, we get new interpolant \(F_2\)
If $F_2 \Rightarrow (I \lor F_1)$ then we have found a fixpoint!
- The system is safe!
Interpolation-based Over-Approximation

- If $F_2 \Rightarrow (I \lor F_1)$ then we have found a fixpoint!
  - The system is safe!
- If not, merge $I \lor F_1$ with $F_2$ and start over
Interpolation-based Over-Approximation

\[ I \subseteq I \lor F_1 \subseteq I \lor F_1 \lor F_2 \]

over-approximates states reachable in up to 2 steps

- The sequence \( I, (I \lor F_1), (I \lor F_1 \lor F_2), \ldots \) is monotonic
Interpolation-based Over-Approximation

The sequence $I, (I \lor F_1), (I \lor F_1 \lor F_2), \ldots$ is monotonic over-approximates states reachable in up to 2 steps.
Interpolation-based Over-Approximation

The sequence \( I, (I \lor F_1), (I \lor F_1 \lor F_2), \ldots \) is monotonic

Note:

\[
\begin{align*}
\underbrace{(I(V) \lor \cdots \lor F_i(V))}_{i^{th} \text{ step}} \land T(V, V') & \implies \\
\underbrace{(I(V') \lor \cdots \lor F_i(V') \lor F_{i+1}(V'))}_{(i+1)^{st} \text{ step}}
\end{align*}
\]
Interpolation-based Over-Approximation

System is safe if

$$F_{k+1} \Rightarrow (I \lor \cdots \lor F_k)$$
Interpolation-based Over-Approximation

System is safe if

\[ F_{k+1} \Rightarrow (I \lor \cdots \lor F_k) \]
What if $\neg P$ is reachable from current approximation?
Interpolation-based Refinement

What if $\neg P$ is reachable from current approximation?

$$( I(V) \lor F_1(V) \lor F_2(V) ) \land T(V, V') \land \neg P(V')$$

satisfiable!
Interpolation-based Refinement

Accuracy of approximation needs to be *improved*
Accuracy of approximation needs to be improved

Let’s rewind and focus on the B-side of our interpolation problem a bit
“Bad states” no not intersect with $F_1$

$F_1(V')$ constrained by $\neg P(V')$
  - $R_1$ is lower bound for $F_1$
  - $P$ is upper bound for $F_1$
Interpolation-based Refinement

Add an additional step on “B-side”
Add an additional step on “B-side”

New “forbidden zone” for $F_1$:

- $\neg P(V)$ and states one step away from $\neg P(V)$
For a large enough $k$, upper bound for $F_1$ becomes:

“all safe states from which $\neg P$ is unreachable”
Interpolation-based Refinement: Drawbacks

\[
I(V) \land T(V, V') \quad \begin{array}{c}
\left( \bigwedge_{i=1}^{k-1} T(V^i, V^{i+1}) \right) \land \left( \bigvee_{i=1}^{k} \neg P(V^i) \right)
\end{array}
\]

- Requires non-incremental restarts (discarding information)
- Requires costly unwinding of transition relation
Interpolation-based Refinement: Drawbacks

\[
I(V) \land T(V, V') \\
\underbrace{\left( \bigwedge_{i=1}^{k-1} T(V^i, V^{i+1}) \right)}_{A\text{–partition}} \land \underbrace{\left( \bigvee_{i=1}^{k} \neg P(V^i) \right)}_{B\text{–partition}}
\]

- Requires non-incremental restarts (discarding information)
- Requires costly unwinding of transition relation

Can we avoid that?
IC3
Incremental Refinement with *Inductive* Clauses

Recall our problem:

$I \lor F_1 \lor F_2 \Rightarrow \neg P$

Can we refine over-approximation to exclude $s$?

Note that previous approximation doesn't contain $s$ otherwise we would have reached $\neg P$ earlier.
Incremental Refinement with *Inductive* Clauses

Recall our problem:

\[ I \lor F_1 \lor F_2 \]

\[ \neg P \]

Can we *refine* over-approximation to exclude \( s \)?
Incremental Refinement with *Inductive* Clauses

Recall our problem:

\[ I \lor F_1 \]

Can we *refine* over-approximation to exclude \( s \)?

- Note that previous approximation doesn’t contain \( s \)
  - otherwise we would have reached \( \neg P \) earlier
Incremental Refinement with *Inductive* Clauses

\[ S \subseteq I \subseteq I \lor F_1 \subseteq I \lor F_1 \lor F_2 \]

**IC3 maintains monotonic sequence of k frames:**

- \[ G_0(V) = I(V) \]
- \[ G_i(V) \land T(V, V') \Rightarrow G_{i+1}(V') \] (0 ≤ i < k)
- \[ G_i(V) \Rightarrow P(V) \] (0 ≤ i ≤ k)
Incremental Refinement with *Inductive* Clauses

\[ G_0 \subseteq G_1 \subseteq G_2 \]

- IC3 maintains monotonic sequence of \( k \) frames:
Incremental Refinement with *Inductive* Clauses

$G_0 \subseteq G_1 \subseteq G_2$

- IC3 maintains monotonic sequence of $k$ frames:

\[ G_0(V) = I(V) \]
Incremental Refinement with *Inductive* Clauses

IC3 maintains monotonic sequence of $k$ frames:

\[
G_0(V) = I(V) \\
G_i(V) \land T(V, V') \Rightarrow G_{i+1}(V') \quad (0 \leq i < k)
\]
IC3 maintains monotonic sequence of $k$ frames:

\[
\begin{align*}
G_0(V) &= I(V) \\
G_i(V) \land T(V, V') &\Rightarrow G_{i+1}(V') \quad (0 \leq i < k) \\
G_i(V) &\Rightarrow G_{i+1}(V) \quad (0 \leq i < k)
\end{align*}
\]
IC3 maintains monotonic sequence of $k$ frames:

\[
G_0(V) = I(V) \\
G_i(V) \land T(V, V') \Rightarrow G_{i+1}(V') \quad (0 \leq i < k) \\
G_i(V) \Rightarrow G_{i+1}(V) \quad (0 \leq i < k) \\
G_i(V) \Rightarrow P(V) \quad (0 \leq i \leq k)
\]
Incremental Refinement with *Inductive* Clauses

\[ G_0 \subseteq G_1 \subseteq G_2 \]

IC3 checks whether \( s \in G_2 \) is reachable from \( G_1 \) via \( T \):

\[ G_1(V) \land T(V, V') \land s(V') \text{ satisfiable?} \]

If not, \( s \) can be removed from \( G_2 \).
Incremental Refinement with *Inductive* Clauses

IC3 checks whether \( s \in G_2 \) is reachable from \( G_1 \) via \( T \):

\[
G_1(V) \land T(V, V') \land s(V') \text{ satisfiable?}
\]
Incremental Refinement with *Inductive* Clauses

IC3 checks whether \( s \in G_2 \) is reachable from \( G_1 \) via \( T \):

\[
G_1(V) \land T(V, V') \land s(V')
\]

satisfiable?

If not, \( s \) can be removed from \( G_2 \).
Incremental Refinement with *Inductive* Clauses

- Otherwise, $s$ has predecessor $t$ in $G_1$ (with $t \not\in G_0$)
Incremental Refinement with *Inductive* Clauses

- Otherwise, $s$ has predecessor $t$ in $G_1$ (with $t \not\in G_0$)
- If $t$ reachable from $G_0$, we have a *counterexample*
Incremental Refinement with \textit{Inductive} Clauses

- Otherwise, \( s \) has predecessor \( t \) in \( G_1 \) (with \( t \not\in G_0 \))
- If \( t \) reachable from \( G_0 \), we have a \textit{counterexample}
- Otherwise, \( t \) can be removed from \( G_1 \)
Incremental Refinement with *Inductive*Clauses

- $\neg t$ is *inductive* relative to $G_0$:

  $$G_0(V) \land \neg t(V) \land T(V, V') \Rightarrow \neg t(V')$$

- In propositional setting, $\neg t$ is a clause
  - IC3’s frames $G_0, \ldots, G_k$ are in CNF
Incremental Refinement with *Inductive* Clauses: Generalization

For efficiency, we want to cut away more than just $t$.

Generalization finds clause $c$ such that state $t \in \neg c$ and:

- $G_0(V) \land c(V) \land T(V, V') \Rightarrow c(V')$ (consecution)
- $I(V) \Rightarrow c(V)$ (initiation)

Heuristic drops literals from $\neg t$ to obtain $c$.
Incremental Refinement with *Inductive* Clauses: Generalization

For efficiency, we want to cut away more than just $t$. 

- $G_0 \subseteq G_1 \land c \subseteq G_2$
For efficiency, we want to cut away more than just $t$

- Generalization finds clause $c$ such that state $t \in \neg c$ and:
  
  \[
  G_0(V) \land c(V) \land T(V, V') \Rightarrow c(V') \quad \text{(consecution)}
  \]
  
  \[
  I(V) \Rightarrow c(V) \quad \text{(initiation)}
  \]
For efficiency, we want to cut away more than just $t$.

- Generalization finds clause $c$ such that state $t \in \neg c$ and:
  
  $G_0(V) \land c(V) \land T(V, V') \Rightarrow c(V')$ (consecution)
  
  $I(V) \Rightarrow c(V)$ (initiation)

- Heuristic drops literals from $\neg t$ to obtain $c$
Incremental Refinement with *Inductive* Clauses: Generalization

\[ G_i(V) \land c(V) \land T(V, V') \Rightarrow c(V') \]  
\[ I(V) \Rightarrow c(V) \]

- \(c\) is inductive relative to \(G_i\)
  - therefore also for all \(G_j\) with \(j < i\) (since \(G_j \Rightarrow G_i\))
  - \(c\) is added to all \(G_j, j \leq i\)
Incremental Refinement with *Inductive* Clauses: Generalization

\[ G_i(V) \land c(V) \land T(V, V') \implies c(V') \]  
\[ l(V) \implies c(V) \]  
(consecution)  
(initiation)

- \( c \) is inductive relative to \( G_i \)
  - therefore also for all \( G_j \) with \( j < i \) (since \( G_j \implies G_i \))
  - \( c \) is added to all \( G_j, j \leq i \)
- consequently, clauses of \( G_i \subseteq \) clauses of \( G_{i-1} \) (for all \( i \leq k \))
  - fixpoint check is *syntactic*!
Incremental Refinement with *Inductive* Clauses: Generalization

\[ G_i(V) \land c(V) \land T(V, V') \Rightarrow c(V') \]  
(conssecution)

\[ I(V) \Rightarrow c(V) \]  
(initiation)

- \( c \) is inductive relative to \( G_i \)
  - therefore also for all \( G_j \) with \( j < i \) (since \( G_j \Rightarrow G_i \))
  - \( c \) is added to all \( G_j, j \leq i \)
- consequently, clauses of \( G_i \subseteq \) clauses of \( G_{i-1} \) (for all \( i \leq k \))
  - fixpoint check is *syntactic*!
- IC3 also tries to push \( c \) forward (to \( G_l, l > i \))
  - can be added to \( G_i \) if inductive relative to \( G_{(l-1)} \)
  - prevents re-encountering same states over and over again
IC3 + Interpolation
IC3 adds frame $G_{k+1}$ once $\neg P$ unreachable from $G_k$
IC3 adds frame $G_{k+1}$ once $\neg P$ unreachable from $G_k$
- Initialized with $P$
- Does not require interpolation at all
IC3 adds frame $G_{k+1}$ once $\neg P$ unreachable from $G_k$

- Initialized with $P$
- Does not require interpolation at all

Is interpolation obsolete?
Combining IC3 and Interpolation

We’ve encountered this before:

\[ G_1(V) \land T(V, V') \]

\[ \neg P(V') \]

\( A \)–partition

\( B \)–partition
Combining IC3 and Interpolation

\[ G_2 \overset{\text{def}}{=} (G_1 \lor F_2) \text{ satisfies all conditions of IC3:} \]

\[ G_1(V) \land T(V, V') \Rightarrow G_2(V') \]

\[ G_1(V) \Rightarrow G_2(V) \]

\[ G_2(V) \Rightarrow P(V) \]
Combining IC3 and Interpolation

$G_2 \overset{\text{def}}{=} (G_1 \lor F_2)$ satisfies all conditions of IC3:

\[ G_1(V) \land T(V, V') \Rightarrow G_2(V') \]
\[ G_1(V) \Rightarrow G_2(V) \]
\[ G_2(V) \Rightarrow P(V) \]

...but it’s not in CNF!
Using IC3 to Convert Formula into CNF

Execute a *new* IC3 instance:

- Let $I \overset{\text{def}}{=} G_1$
- Let $P \overset{\text{def}}{=} G_2 \overset{\text{def}}{=} (G_1 \lor F_2)$
Using IC3 to Convert Formula into CNF

Execute a new IC3 instance:

- Let $I \overset{\text{def}}{=} G_1$
- Let $P \overset{\text{def}}{=} G_2 \overset{\text{def}}{=} (G_1 \lor F_2)$

IC3 constructs $H_2$ in CNF such that:

- $G_1 \Rightarrow H_2$ and clauses of $H_2 \subseteq$ clauses of $G_1$
- $H_2 \Rightarrow (G_1 \lor F_2)$ and thus $H_2 \Rightarrow P$

Therefore, $H_2$ can be used to initialize new frame!
IC3 and Interpolation

- IC3 is the new kid on the block
- Interpolation is still a frequently used workhorse
  - Most successful model checkers use portfolio approach
  - IC3 and interpolation can be combined
Part II: Interpolation Systems

Labelled Interpolation Systems for Hyper-Resolution, Clausal, and Local Proofs
with Matthias Schlaipfer

Craig’s Interpolation Theorem

if \((A(V, V') \land B(V', V'')) \Rightarrow \bot\)

then

\[\exists C(V')\]

s.t.

\[A(V, V') \Rightarrow C(V')\]

\[B(V', V'') \Rightarrow \neg C(V')\]
Craig’s Interpolation Theorem

if \( (A(V, V') \land B(V', V'') \Rightarrow \bot) \)
then
\[ \exists C(V') \]
s.t.
\[ A(V, V') \Rightarrow C(V') \]
\[ B(V', V'') \Rightarrow \neg C(V') \]

...but where do interpolants come from?
3 Interpolation Systems
Learned Clauses in CDCL SAT Solvers

\[
\begin{align*}
C_1 & \triangleq (\overline{x}_4 \, x_9 \, x_6) \\
C_2 & \triangleq (\overline{x}_4 \, x_2 \, x_5) \\
C_3 & \triangleq (\overline{x}_5 \, \overline{x}_6 \, \overline{x}_7) \\
C_4 & \triangleq (\overline{x}_6 \, x_7)
\end{align*}
\]
Learned Clauses in CDCL SAT Solvers

$C_5 = \text{Res}(C_4, C_3, x_7) = (\overline{x}_5 \overline{x}_6)$

$C_6 = \text{Res}(C_5, C_1, x_6) = (\overline{x}_4 \overline{x}_5 x_9)$

$C_7 = \text{Res}(C_6, C_2, x_5) = (x_2 \overline{x}_4 x_9)$
A–partition

\[ C \lor x \quad \bar{x} \lor D \]

\[
\begin{array}{c}
C \lor D \\
[\text{Res}]
\end{array}
\]

B–partition
\[
\frac{C \lor x \quad \bar{x} \lor D}{C \lor D} \quad \text{[Res]}
\]
\[ C \lor x \quad \overline{x} \lor D \quad [\text{Res}] \]
\[
\frac{C \lor x \quad \overline{x} \lor D}{C \lor D} \quad [\text{Res}]
\]
\[
\frac{C \lor x \quad \overline{x} \lor D}{C \lor D} \quad \text{[Res]}
\]
What’s the *strongest* consequence of $A$ over *shared* variables?

\[
\begin{align*}
& I \overset{\text{def}}{=} \exists x \in (\text{Var}(A) \setminus \text{Var}(B)) \cdot A \\
\end{align*}
\]

We drop all literals “local” to $A$ upfront.
What’s the strongest consequence of $A$ over shared variables?

$$ I \quad \overset{\text{def}}{=} \quad \exists x \in (\text{Var}(A) \setminus \text{Var}(B)) . A $$

We drop all literals “local” to $A$ upfront.

Resolution with $x \in (\text{Var}(A) \setminus \text{Var}(B))$:

\[
\begin{array}{c}
(C \lor x) [l_1] \quad \neg x \lor D [l_2] \\
\hline
(C \lor D) [l_3]
\end{array}
\quad \text{[Res]}
\]

“simulate” resolution elimination of $x$ using disjunction

$$ l_2 \quad \overset{\text{def}}{=} \quad l_1 \lor l_2 $$
Annotate each clause $C$ in the proof with a *partial interpolant* $I$

- **Base case (initial clause $C$):**
  - $I$ = “keep all literals $\ell \in C$ s.t. $\text{var}(\ell) \in \text{Var}(B)”$
  - $I = \text{true}$

- **Induction step (internal clauses $C_1$, $C_2$):**

\[
\begin{array}{c}
  C_1 \lor x & \underset{[l_1]}{\rightarrow} & C_2 \lor \overline{x} & \underset{[l_2]}{\rightarrow} & C_1 \lor C_2 & \underset{[l_3]}{\rightarrow}
\end{array}
\]

if $x \notin \text{Var}(B)$, \quad $l_3 \overset{\text{def}}{=} l_1 \lor l_2$ \quad $\xrightarrow{\longrightarrow} \quad l_3$

if $x \in \text{Var}(B)$, \quad $l_3 \overset{\text{def}}{=} l_1 \land l_2$ \quad $\xrightarrow{\longrightarrow} \quad l_3$
Interpolants from Proofs: Example

\[ A \equiv (x_1 \lor \overline{x}_2) \land (\overline{x}_1 \lor \overline{x}_3) \land x_2 \quad \text{and} \quad B \equiv (\overline{x}_2 \lor x_3) \land (x_2 \lor x_4) \land \overline{x}_4. \]
Projection in McMillan’s Interpolation System

Given clause $C = \{a_1, \ldots, a_n\}$,

- $C|_A =$”all literals $\ell \in C$ s.t. $\text{var}(\ell) \in \text{Var}(A)$”
- $C|_B =$”all literals $\ell \in C$ s.t. $\text{var}(\ell) \in \text{Var}(B)$”

Invariant for partial interpolant $I$:

- $A' \land \neg(C|_{A'}) \Rightarrow I$
- $B' \land \neg(C|_{B'}) \Rightarrow \neg I$
- $\text{Var}(I) \subseteq \text{Var}(A') \cap \text{Var}(B')$
What is a Partial Interpolant?

- Subproofs, intermediate conclusions:

  \[ \begin{array}{c}
  A' \\
  B' \\
  \hline
  C
  \end{array} \quad \begin{array}{c}
  C |_{A'} \\
  C |_{B'}
  \end{array} \]

- Assumption: \( C = (C_{A'} \lor C_{B'}) \), where \( \text{Var}(C_{A'}) \subseteq \text{Var}(A') \)

- *Annotated* inference steps:

\[
\frac{A' \ [I_A] \quad B' \ [I_B]}{C \ [I]},
\]
What is a Partial Interpolant? (continued)

\[ A' \land \neg (C |_{A'}) \Rightarrow I \]
\[ B' \land \neg (C |_{B'}) \Rightarrow \neg I \]
\[ \text{Var}(I) \subseteq \text{Var}(A') \cap \text{Var}(B') \]

\[ A \Rightarrow I \]
\[ B \Rightarrow \neg I \]
\[ \text{Var}(I) \subseteq \text{Var}(A) \cap \text{Var}(B) \]
Symmetry in Interpolation Systems

Note how the invariant is “symmetric”:

- $A' \land \neg(C|_{A'}) \Rightarrow I$
- $B' \land \neg(C|_{B'}) \Rightarrow \neg I$
- $\text{Var}(I) \subseteq \text{Var}(A') \cap \text{Var}(B')$

... but McMillan’s system is not?

- $I = \text{“keep all literals } \ell \in C \text{ s.t. } \text{var}(\ell) \in \text{Var}(B)\text{”}$
- $I = \text{true}$
Symmetry in Interpolation Systems

Note how the invariant is “symmetric”:

- \( A' \land \neg (C|_{A'}) \Rightarrow I \)
- \( B' \land \neg (C|_{B'}) \Rightarrow \neg I \)
- \( \text{Var}(I) \subseteq \text{Var}(A') \cap \text{Var}(B') \)

... but McMillan’s system is not?

- \( A \quad I = \text{“keep all literals } \ell \in C \text{ s.t. } \text{var}(\ell) \in \text{Var}(B)\” \)
- \( B \quad I = \text{true} \)

That’s because there’s another side to it!
Weakest formula over shared variables inconsistent with $B$?

\[ I \overset{\text{def}}{=} \neg ( \exists x \in (\text{Var}(B) \setminus \text{Var}(A)).B ) \]
Weakest formula over shared variables inconsistent with $B$?

$\forall x \in (\text{Var}(B) \setminus \text{Var}(A)). \neg B$

Again, we drop all $B$-local literals upfront
Weakest formula over shared variables inconsistent with $B$?

\[ I \overset{\text{def}}{=} \forall x \in (\text{Var}(B) \setminus \text{Var}(A)). \neg B \]

Again, we drop all $B$-local literals upfront.

Resolution with $x \in (\text{Var}(B) \setminus \text{Var}(A))$:

\[
\begin{array}{c}
\frac{(C \lor x) \ [I_1] \quad (\bar{x} \lor D) \ [I_2]}{(C \lor D) \ [I_3]} \\
\end{array}
\]

[Res]

\[ I_2 \overset{\text{def}}{=} I_1 \land I_2 \]
Inverse McMillan’s Interpolation System

Annotate each clause $C$ in the proof with a partial interpolant $I$

- **Base case (initial clause $C$):**
  - $l = \text{false}$
  - $l = \neg (\text{“keep all literals } \ell \in C \text{ s.t. } \text{var}(\ell) \in \text{Var}(A)\text{”})$

- **Induction step (internal clauses $C_1$, $C_2$):**

  \[
  \frac{C_1 \lor \chi \quad [l_1] \quad C_2 \lor \overline{\chi} \quad [l_2]}{C_1 \lor C_2 \quad [l_3]}
  \]

  if $x \in \text{Var}(A)$, \quad $l_3 \overset{\text{def}}{=} l_1 \lor l_2 \quad \frac{l_1}{l_2} \longrightarrow l_3$

  if $x \notin \text{Var}(A)$, \quad $l_3 \overset{\text{def}}{=} l_1 \land l_2 \quad \frac{l_1}{l_2} \longrightarrow l_3$
Strong and Weak Interpolation Systems

- McMillan’s interpolants *imply* inverse McMillan’s interpolants
- Is there middle ground?
Strong and Weak Interpolation Systems

- McMillan’s interpolants *imply* inverse McMillan’s interpolants
- Is there middle ground?
Interpolants as Separators

\[(\overline{x_1} \lor \overline{x_2}) \land \overline{x_0} \land (x_0 \lor x_2) \land \overline{x_2} \land (x_1 \lor x_2)\]
Interpolants as Separators

\[(\overline{x}_1 \lor \overline{x}_2) \land \overline{x}_0 \land (x_0 \lor x_2) \land (\overline{x}_2 \land (x_1 \lor x_2))\]

\[
\begin{align*}
A & = (\overline{x}_1 \lor \overline{x}_2) \land \overline{x}_0 \land (x_0 \lor x_2) \\
B & = (\overline{x}_2 \land (x_1 \lor x_2))
\end{align*}
\]
Interpolants as Separators

\[(\overline{x_1} \lor \overline{x_2}) \land \overline{x_0} \land (x_0 \lor x_2) \land (\overline{x_2} \land (x_1 \lor x_2))\]

\[A \Rightarrow \overline{x_1} \quad B \Rightarrow x_1 \quad x_1 \in \text{Var}(A) \cap \text{Var}(B)\]
Interpolants as Separators

\[
(\overline{x}_1 \lor \overline{x}_2) \land \overline{x}_0 \land (x_0 \lor x_2) \quad \land \quad \overline{x}_2 \land (x_1 \lor x_2)
\]

\[A \implies \overline{x}_1 \quad B \implies x_1 \quad x_1 \in \text{Var}(A) \cap \text{Var}(B)\]

\begin{align*}
\text{l is false} & \quad (\overline{x}_1 \mapsto 0) \quad \implies \quad A[\overline{x}_1 \mapsto 0] \quad \text{unsatisfiable} \\
\text{l is true} & \quad (\overline{x}_1 \mapsto 1) \quad \implies \quad B[\overline{x}_1 \mapsto 1] \quad \text{unsatisfiable}
\end{align*}
Interpolants separate Resolution Proofs

Annotate each clause $C$ in proof with partial interpolant $I$

$A \land \neg I_C \Rightarrow C\{\ell \in C | \ell \text{ is}\}$

$B \land I_C \Rightarrow C\{\ell \in C | \ell \text{ is}\}$

$\text{Var}(I_C) \subseteq \text{Var}(A) \cap \text{Var}(B)$
Interpolants separate Resolution Proofs

Annotate each clause $C$ in proof with partial interpolant $I$

$A \land \neg I \rightarrow C\{\ell \in C | \ell \text{ is}\}$

$B \land I \rightarrow C\{\ell \in C | \ell \text{ is}\}$

$\text{Var}(I_C) \subseteq \text{Var}(A) \cap \text{Var}(B)$
Interpolants separate Resolution Proofs

Annotate each clause $C$ in proof with partial interpolant $I$

$A \land \neg I \Rightarrow C \{\ell \in C | \ell \text{ is }\}$

$B \land I \Rightarrow C \{\ell \in C | \ell \text{ is }\}$

$\text{Var}(I_C) \subseteq \text{Var}(A) \cap \text{Var}(B)$
Interpolants *separate* Resolution Proofs

Annotate each clause \( C \) in proof with partial interpolant \( I \)

\[
A \land \neg I \Rightarrow C \{ \ell \in C | \ell \text{ is } \}
\]

\[
B \land I \Rightarrow C \{ \ell \in C | \ell \text{ is } \}
\]

\( \text{Var}(I) \subseteq \text{Var}(A) \cap \text{Var}(B) \)
Interpolants *separate* Resolution Proofs

- Annotate each clause $C$ in proof with *partial interpolant* $I_C$

  - $A \land \neg I_C \Rightarrow C \setminus \{\ell \in C \mid \ell \text{ is } \llcorner\}
  
  - $B \land I_C \Rightarrow C \setminus \{\ell \in C \mid \ell \text{ is } \llcorner\}
  
  - $\text{Var}(I_C) \subseteq \text{Var}(A) \cap \text{Var}(B)$
Pudlák’s Interpolation System

- **Base case (initial vertices):**
  - If $C \in A$: $I \overset{\text{def}}{=} \text{false}$
  - If $C \in B$: $I \overset{\text{def}}{=} \text{true}$

- **Induction step (internal vertices):**
  
  \[
  \begin{array}{c}
  \frac{C_1 \lor x \quad [I_1] \quad C_2 \lor \overline{x} \quad [I_2]}{C_1 \lor C_2 \quad [I_3]}
  \end{array}
  \]

  - if $x$ is $I_3 \overset{\text{def}}{=} I_1 \lor I_2$
  - if $x$ is $I_3 \overset{\text{def}}{=} (x \lor I_1) \land (I_2 \lor \overline{x})$
  - if $x$ is $I_3 \overset{\text{def}}{=} I_1 \land I_2$
Interpolants from Proofs: Example Revisited

\[ A \equiv (x_1 \lor \overline{x}_2) \land (\overline{x}_1 \lor \overline{x}_3) \land x_2 \quad \text{and} \quad B \equiv (\overline{x}_2 \lor x_3) \land (x_2 \lor a_4) \land \overline{x}_4. \]
Interpolants from Proofs: Example Revisited

\[ A \equiv (x_1 \lor \overline{x}_2) \land (\overline{x}_1 \lor \overline{x}_3) \land x_2 \quad \text{and} \quad B \equiv (\overline{x}_2 \lor x_3) \land (x_2 \lor a_4) \land \overline{x}_4 . \]

- \( x_1 \overline{x}_2 \ [\bot] \overline{x}_1 \overline{x}_3 \ [\bot] \overline{x}_2 x_3 \ [\top] \ x_2 x_4 \ [\top] \ \overline{x}_4 \ [\top] \\
  \overline{x}_2 x_3 \ [\bot] \ x_2 \ [\bot] \quad \overline{x}_2 \ [\top] \\
  \overline{x}_3 \ [\bot] \quad \ x_3 \ [\top] \\
  \square \ [\overline{x}_3] \\

- \( \overline{x}_3 \) differs from \( \overline{x}_3 \land x_2 \) (obtained using McMillan’s technique)
  - Contains fewer variables
  - Is weaker (but implies inverse McMillan)
Generalizing Interpolation Systems
Generalizing Interpolation

So far, we have 3 interpolation systems:

McMillan $\Rightarrow$ Pudlák $\Rightarrow$ inverse McMillan

We will generalize them in 2 ways:

- allow more general proof systems
- allow more flexible labeling of literals
Interpolants from Clausal Proofs
Resolution Chains Generated By CDCL Solvers

input clauses

conclusion
Hyper-Resolution

\[ \text{satellites} (C_1 \lor x_1) \cdots (C_n \lor x_n) \]
\[ \text{nucleus} (x_1 \lor \cdots \lor x_n \lor D) \]

Summary of a derivation consisting of several resolutions.
Hyper-Resolution

\[ \overline{x_1} \overline{x_2} \quad \overline{x_0} \quad x_0 \cdot x_2 \quad \overline{x_2} \quad x_1 \cdot x_2 \]

\[ \text{nucleus} \quad \text{satellites} \]

\[ \text{[HyRes]} \]

Summary of a derivation consisting of several resolutions.
Hyper-Resolution

\[ \begin{align*}
\text{nucleus} & : \overline{x}_0 \quad x_0 x_2 \\
\text{satellites} & : \overline{x}_2 \quad x_1 x_2
\end{align*} \]

\[ \begin{align*}
\text{[HyRes]} & \\
\text{summary of a derivation consisting of several resolutions}
\end{align*} \]
Hyper-Resolution

\[
\begin{array}{c}
\bar{x}_0 & x_0 & x_2 & \bar{x}_2 & x_1 & x_2 \\
\bar{x}_1 & \bar{x}_2 & x_2 & x_1 & ?
\end{array}
\]

\[
(C_1 \lor x_1) \cdots (C_n \lor x_n) \lor \bigvee_{i=1}^{n} C_i \lor D
\]

- **summary** of a derivation consisting of several resolutions
Interpolation for Hyper-Resolution Steps

\[
\begin{align*}
(C_1 \lor x_1) & \quad [l_1] \\
\vdots & \quad \vdots \\
(C_n \lor x_n) & \quad [l_n] \\
\left(\bar{x}_1 \lor \cdots \lor \bar{x}_n \lor D\right) & \quad [l_{n+1}]
\end{align*}
\]

\[
\bigvee_{i=1}^{n} C_i \lor D \quad [l]
\]

if \( x \) is

\[
I \overset{\text{def}}{=} (x \lor l_1) \land (l_2 \lor \bar{x})
\]

(previously)
Interpolation for Hyper-Resolution Steps

\[
\begin{align*}
(C_1 \lor x_1) & \quad [l_1] \\
\cdots & \\
(C_n \lor x_n) & \quad [l_n] \\
(\overline{x}_1 \lor \cdots \lor \overline{x}_n \lor D) & \quad [l_{n+1}]
\end{align*}
\]

\[
\bigvee_{i=1}^{n} C_i \lor D \quad [l]
\]

if \( x_1, \ldots, x_n \) are

\[
I \overset{\text{def}}{=} \bigwedge_{i=1}^{n} (x_i \lor l_i) \land (l_{n+1} \lor \bigvee_{i=1}^{n} \overline{x}_i)
\]
Interpolation for Hyper-Resolution Steps

\[
\begin{array}{c}
(C_1 \lor x_1) [I_1] \cdots (C_n \lor x_n) [I_n] (\bar{x}_1 \lor \cdots \lor \bar{x}_n \lor D) [I_{n+1}] \\
\bigvee_{i=1}^{n} C_i \lor D [I]
\end{array}
\]

if \( x_1, \ldots, x_n \) are

\[
I \overset{\text{def}}{=} \bigvee_{i=1}^{n+1} I_i
\]

if \( x_1, \ldots, x_n \) are

\[
I \overset{\text{def}}{=} \bigwedge_{i=1}^{n} (x_i \lor I_i) \land (I_{n+1} \lor \bigvee_{i=1}^{n} \bar{x}_i)
\]

if \( x_1, \ldots, x_n \) are

\[
I \overset{\text{def}}{=} \bigwedge_{i=1}^{n+1} I_i
\]
Interpolation for Hyper-Resolution Steps

\[ \begin{align*}
(C_1 \vee x_1) & \quad [l_1] \\
\cdots & \\
(C_n \vee x_n) & \quad [l_n] \\
(\bar{x}_1 \vee \cdots \vee \bar{x}_n \vee D) & \quad [l_{n+1}] \\
\bigvee_{i=1}^{n} C_i \vee D & \quad [l]
\end{align*} \]

- If \( x_1, \ldots, x_n \) are
  \[ I \equiv \bigvee_{i=1}^{n+1} l_i \]
- If \( x_1, \ldots, x_n \) are
  \[ I \equiv \bigwedge_{i=1}^{n} (x_i \vee l_i) \land (l_{n+1} \vee \bigvee_{i=1}^{n} \bar{x}_i) \]
- If \( x_1, \ldots, x_n \) are
  \[ I \equiv \bigwedge_{i=1}^{n+1} l_i \]

not total! (can be fixed)
Interpolation for Hyper-Resolution: Simple Example Revisited

If \( x_1, \ldots, x_n \) are

\[
I \overset{\text{def}}{=} \bigwedge_{i=1}^{n} (x_i \lor i) \land (l_{n+1} \lor \bigvee_{i=1}^{n} \neg x_i)
\]
Interpolation for Hyper-Resolution: Simple Example Revisited

If \( x_1, \ldots, x_n \) are

\[
I \overset{\text{def}}{=} \bigwedge_{i=1}^{n} (x_i \lor l_i) \land (l_{n+1} \lor \bigvee_{i=1}^{n} \overline{x_i})
\]

\[ I \overset{\text{def}}{=} (x_1 \lor \text{true}) \land (x_2 \lor \text{false}) \land (\overline{x_1} \lor \overline{x_2} \lor \text{false}) \equiv \]

\[
x_2 \land (\overline{x_1} \lor \overline{x_2})
\]
Interpolation for Hyper-Resolution: Simple Example Revisited

\[ I \overset{\text{def}}{=} \bigwedge_{i=1}^{n} (x_i \lor l_i) \land (l_{n+1} \lor \bigvee_{i=1}^{n} \bar{x}_i) \]

- \[ I \overset{\text{def}}{=} (x_1 \lor \text{true}) \land (x_2 \lor \text{false}) \land (\bar{x}_1 \lor \bar{x}_2 \lor \text{false}) \]
  \[ \equiv x_2 \land (\bar{x}_1 \lor \bar{x}_2) \]
- \( I \) implies \( \bar{x}_1 \) but is \emph{not equivalent} (try \( x_1 = x_2 = \text{false} \))
Interpolation for Hyper-Resolution provides us with a choice for each *inner* node of the proof:

$$I \overset{\text{def}}{=} \begin{cases} \bigwedge_{i=1}^{n}(x_i \lor l_i) \land (l_{n+1} \lor \bigvee_{i=1}^{n} \overline{x_i}) \\
\text{or} \\
\bigvee_{i=1}^{n}(\overline{x_i} \land l_i) \lor (l_{n+1} \land \bigwedge_{i=1}^{n} x_i) \end{cases}$$
Parametrised Interpolation System

- Interpolation for Hyper-Resolution provides us with a choice for each *inner* node of the proof:

  \[
  I \overset{\text{def}}{=} \begin{cases} 
  \bigwedge_{i=1}^{n} (x_i \lor I_i) \land (I_{n+1} \lor \bigvee_{i=1}^{n} \overline{x_i}) & \text{or} \\
  \bigvee_{i=1}^{n} (\overline{x_i} \land I_i) \lor (I_{n+1} \land \bigwedge_{i=1}^{n} x_i) 
  \end{cases}
  \]

- Choice determines *strength* of interpolant

  \[
  \bigwedge_{i=1}^{n} (x_i \lor I_i) \land (I_{n+1} \lor \bigvee_{i=1}^{n} \overline{x_i}) \\
  \Downarrow \\
  \bigvee_{i=1}^{n} (\overline{x_i} \land I_i) \lor (I_{n+1} \land \bigwedge_{i=1}^{n} x_i)
  \]
Interpolation for Hyper-Resolution: Example Revisited (Again)

If $x_1, \ldots, x_n$ are

\[
I \overset{\text{def}}{=} \bigvee_{i=1}^n (\overline{x}_i \land l_i) \lor (l_{n+1} \land \bigwedge_{i=1}^n x_i)
\]
Interpolation for Hyper-Resolution: Example Revisited (Again)

If \( x_1, \ldots, x_n \) are

\[
I \overset{\text{def}}{=} \bigvee_{i=1}^{n} (\overline{x}_i \land l_i) \lor (l_{n+1} \land \bigwedge_{i=1}^{n} x_i)
\]

\( \square [I] \)

\[
\neg I \overset{\text{def}}{=} (\overline{x}_1 \land \text{true}) \lor (\overline{x}_2 \land \text{false}) \land (x_1 \land x_2 \land \text{false}) \equiv \overline{x}_1
\]
Interpolation for Hyper-Resolution

- **Base case (initial vertices):**
  - If $C \in A$: $I \overset{\text{def}}{=} \text{false}
  - If $C \in B$: $I \overset{\text{def}}{=} \text{true}

- **Induction step (internal vertices):**

$$
\frac{(C_1 \lor x_1) [l_1] \cdots (C_n \lor x_n) [l_n] (\overline{x}_1 \lor \cdots \lor \overline{x}_n \lor D) [l_{n+1}]}{\lor_{i=1}^{n} C_i \lor D [l]}
$$

if $x_1, \ldots, x_n$ are $I \overset{\text{def}}{=} \lor_{i=1}^{n+1} l_i$

if $x_1, \ldots, x_n$ are $I \overset{\text{def}}{=} \left\{ \begin{array}{ll}
\land_{i=1}^{n} (x_i \lor l_i) \land (l_{n+1} \lor \lor_{i=1}^{n} \overline{x}_i) \\
\text{or}
\lor_{i=1}^{n} (\overline{x}_i \land l_i) \lor (l_{n+1} \land \land_{i=1}^{n} x_i)
\end{array} \right.$

if $x_1, \ldots, x_n$ are $I \overset{\text{def}}{=} \land_{i=1}^{n+1} l_i$
Labelled Interpolation Systems
Labelled Interpolation Systems

- Colouring scheme can be relaxed!

(colour lattice)
Locality Preserving Colouring

Each literal $\ell$ in each clause coloured separately!

- Literals from $A \setminus B$ must be coloured
- Literals from $B \setminus A$ must be coloured
- Literals from $A$ and $B$: Any colour $\in \{\text{blue}, \text{red}, \text{blue}, \text{red}\}$
## Strength of Interpolants (Using Labelled Interpolation)

<table>
<thead>
<tr>
<th></th>
<th>A-local</th>
<th>A/B-shared</th>
<th>B-local</th>
</tr>
</thead>
<tbody>
<tr>
<td>strongest</td>
<td><img src="image1.png" alt="Image" /></td>
<td><img src="image2.png" alt="Image" /></td>
<td><img src="image3.png" alt="Image" /></td>
</tr>
<tr>
<td>weakest</td>
<td><img src="image4.png" alt="Image" /></td>
<td><img src="image5.png" alt="Image" /></td>
<td><img src="image6.png" alt="Image" /></td>
</tr>
</tbody>
</table>
Labelled Interpolation System for Hyper-Resolution

- **Base case (initial vertices):**
  - If $C \in A$: $I \overset{\text{def}}{=} \text{"all literals } \ell \in C \text{ s.t. } L(\ell, v) = 0\text{ "}$
  - If $C \in B$: $I \overset{\text{def}}{=} \neg(\text{"all literals } \ell \in C \text{ s.t. } L(\ell, v) = 0\text{ "})$

- **Induction step (internal vertices):**

\[
\begin{align*}
(C_1 \lor x_1) & \ [I_1] \quad \ldots \quad (C_n \lor x_n) & \ [I_n] \quad (\bar{x}_1 \lor \cdots \lor \bar{x}_n \lor D) & \ [I_{n+1}] \\
\lor_{i=1}^{n} C_i \lor D & \ [I]
\end{align*}
\]

if $\bigwedge_{i=1}^{n} L(x_i) \cup L(\bar{x}_i) = 0$

$I \overset{\text{def}}{=} \bigvee_{i=1}^{n+1} I_i$

if $\bigwedge_{i=1}^{n} L(x_i) \cup L(\bar{x}_i) = 0$

$I \overset{\text{def}}{=} \bigg\{ \bigwedge_{i=1}^{n} (x_i \lor I_i) \land (I_{n+1} \lor \bigvee_{i=1}^{n} \bar{x}_i) \bigg\} \lor
\bigg\{ \bigvee_{i=1}^{n} (\bar{x}_i \land I_i) \lor (I_{n+1} \land \bigwedge_{i=1}^{n} x_i) \bigg\}$

if $\bigwedge_{i=1}^{n} L(x_i) \cup L(\bar{x}_i) = 0$

$I \overset{\text{def}}{=} \bigwedge_{i=1}^{n+1} I_i$
Recap

- Interpolants from resolution proofs
- Plenty of choices of interpolation systems
  - related by structure and strength
  - subsumed by *labelled interpolation*
  - but none of them is “best”
Recap

- Interpolants from resolution proofs
- Plenty of choices of interpolation systems
  - related by structure and strength
  - subsumed by labelled interpolation
  - but none of them is “best”

More about our current research:

Interpolants from SAT Solving Certificates
Adrián Rebola Pardo, 4:30pm