Interpolation-Sequence Based Model Checking

Yakir Vizel and Orna Grumberg
FMCAD 2009

Verification course, June 5, 2017
Inspired by:
• forward reachability analysis

Combines:
• Bounded Model Checking
• Interpolation-sequence

Obtains:
• SAT-based model checking algorithm for full verification
Forward Reachability Analysis
Forward reachability analysis

- $S_j$ is the set of states reachable from some initial state in $j$ steps
- termination when
  - either a bad state satisfying $\neg q$ is found
  - or a fixpoint is reached:
  $$S_j \subseteq \bigcup_{i=1}^{j-1} S_i$$
SAT-based model checking:  
A solution for the state explosion problem

Main idea

• Translate the model and the specification to propositional formulas

• Use efficient tools (SAT solvers) for solving the satisfiability problem
Bounded Model Checking (BMC) for checking $\text{AG}p$

- Unwind the model for $k$ levels, i.e., construct all computations of length $k$

- If a state satisfying $\neg p$ is encountered, produce a counterexample; Otherwise, increase $k$

[BCCZ 99]
Bounded Model Checking

- Does the system have a counterexample of length \( k \)?

\[
\text{INIT}(V_0) \land \neg p(V_0)
\]

\[
\text{INIT}(V_0) \land T(V_0, V_1) \land \neg p(V_1)
\]

\[
\text{INIT}(V_0) \land T(V_0, V_1) \land T(V_1, V_2) \land \neg p(V_2)
\]

\[
\ldots
\]

\[
\text{INIT}(V_0) \land T(V_0, V_1) \land T(V_1, V_2) \land \ldots \land T(V_{k-1}, V_k) \land \neg p(V_k)
\]
Bounded Model Checking

Terminates
• with a counterexample or
• with time- or memory-out

The method is suitable for falsification, not verification
Verification with SAT solvers

Two successful methods for SAT-based verification are based on:

- Interpolation [McMillan 03]
- IC3 [Bradley 11]

we present two methods for enhancing interpolation and IC3 model checking
Interpolation

• If $A \land B = \text{false}$, there exists an interpolant $I$ for $(A, B)$ such that:

\[
A \implies I \\
I \land B = \text{false}
\]

$I$ refers only to common variables of $A, B$
Interpolation in the context of model checking

- Given the following BMC formula $\varphi^k$

\[ A \implies I \]
\[ I \land B \equiv \text{false} \]

I is over the common variables of A and B, i.e. $V_1$
Interpolation in the context of model checking

- $I$ is over $V_1$
- $A \implies I$
  - $I$ over-approximates the set $S_1$
- $I \land B \equiv \text{false}$
  - States in $I$ cannot reach a bug in $k-1$ steps
**Interpolation-Sequence**

- The same BMC formula partitioned in a different manner:

\[
\begin{align*}
&\text{INIT}(V_0) \land T(V_0,V_1) \land T(V_1,V_2) \land T(V_2,V_3) \land \ldots \land T(V_{k-1},V_k) \land \neg \phi(V_k) \\
&I_1 \downarrow \quad I_2 \downarrow \quad I_3 \downarrow \quad I_{k-1} \downarrow \quad I_k \downarrow
\end{align*}
\]

\[I_0 = true, I_{k+1} = false\]

\[I_{j-1} \land A_j \Rightarrow I_j\]

I_j is over the common variables of A_1,…,A_j and A_{j+1},…,A_{k+1}, i.e. \(V_j\)
Interpolation-Sequence

- $I_j$ - over-approximation of the set of states reachable in $j$ steps

- $I_k \land A_{k+1} \Rightarrow false$
  the states in $I_k$ do not violate $p$
Interpolation-Sequence

• Can easily be computed in the same way a single interpolation is computed:

• For $1 \leq j < n$
  - $A(j) = A_1 \wedge \ldots \wedge A_j$
  - $B(j) = A_{j+1} \wedge \ldots \wedge A_n$
  - $I_j$ is the interpolant for the pair $(A(j), B(j))$
Combining Interpolation-Sequence and BMC

- Uses BMC for bug finding

- Uses Interpolation-sequence for computing over-approximation of sets $S_j$ of reachable states
Combining Interpolation-Sequence and BMC

Always terminates

• either when BMC finds a bug:
  \( M \models \neg AGp \)

• or when all reachable states has been found:
  \( M \models AGp \)
Using Interpolation-Sequence

\[ \text{INIT}(V_0) \land T(V_0, V_1) \land \neg p(V_1) \]

\[ \text{INIT}(V_0) \land T(V_0, V_1) \land T(V_1, V_2) \land \neg p(V_2) \]
Checking if a “fixpoint” has been reached

- $I_j \Rightarrow V_{k=1,j-1} I_k$

- Similar to checking fixpoint in forward reachability analysis:
  $$S_j \subseteq U_{k=1,j-1} S_k$$

- But here we check inclusion for every $2 \leq j \leq N$
  - No monotonicity because of the approximation

- “Fixpoint” is checked with a SAT solver
The Analogy to Forward Reachability Analysis

\[ \text{INIT}(V_0) \land T(V_0, V_1) \land T_1(V_1, V_2) \land T_2(V_2, V_3) \land \neg p(V_3) \]
Notation:

If no counterexample of length \( N \) or less exists in \( M \), then:

- \( I_j^k \) is the \( j \)-th element in the interpolation-sequence extracted from the BMC-partition of \( \varphi^k \)

- \( I_j = \bigwedge_{k=j,N} I_j^k \ [V_j \leftarrow V] \)

- The reachability vector is:
  \( \hat{I} = (I_1, I_2, \ldots, I_N) \)
$I_j = \bigwedge_{k=j,N} I_j^k [V_j \leftarrow V]$

- Each $I_j^k$ over-approximates $S_j$
  - Their conjunction results in a more accurate over-approximation

- Only $I_j^j$ is guaranteed to satisfy $p$
  - $I_j$ satisfies $p$
function UpdateReachable( \( \hat{I}, \hat{I}^k \) )

\[ j=1 \]

while \((j < k)\) do

\[ I_j = I_j \land I_j^k \]

\[ \hat{I}[j] = I_j \]

end while

\[ \hat{I}[k] = I_k^k \]

end function
function FixpointReached (\( \hat{\mathcal{I}} \))

\[ j=2 \]

while (\( j \leq \hat{\mathcal{I}}.\text{length} \)) do

\[ R = V_{k=1,j-1} I_k \]

\[ \alpha = I_j \land \neg R \] // negation of \( I_j \leftrightarrow R \)

if (SAT(\( \alpha \))==false) then return true

end if

\[ j = j+1 \]

end while

return false

end function
Function ISB(M, f)  // f = AGq
    k = 0
    result = BMC (M, f, 0)
    if (result == cex) then return cex
    \( \hat{I} = \emptyset \)  // the reachability vector
    while (true) do
        k = k+1
        result = BMC (M, f, k)
        if (result==cex) then return cex
        \( \hat{I}^k = ( T, I_1^k, \ldots, I_k^k, F ) \)
        UpdateReachable (\( \hat{I} \), \( \hat{I}^k \))
        if ( FixpointReached (\( \hat{I} \)) == true) then
            return true
        end if
    end while
end function
Interpolation-Based Model Checking [McM03]
Interpolation in The Context of Model Checking

- We can check several bounds with one formula
- Given a BMC formula with possibly several bad states

\[ \text{INIT}(V_0) \land T(V_0, V_1) \land T(V_1, V_2) \land \ldots \land T(V_{k-1}, V_k) \land (\neg q(V_1) \lor \ldots \lor \neg q(V_k)) \]

\[ A \Rightarrow I \]

\[ I \land B \equiv F \]

I is over the common variables of A and B, i.e. \( V_1 \)
Interpolation In The Context of Model Checking

- The interpolant represents an over-approximation of reachable states after one transition.
- Also, there is no path of length $k-1$ or less that can reach a bad state.
Using Interpolation

\[ \text{INIT}(V_0) \land T(V_0, V_1) \land \neg q(V_1) \]

\[ I_1(V_0) \land T(V_0, V_1) \land \neg q(V_1) \]

\[ I_2(V_0) \land T(V_0, V_1) \land \neg q(V_1) \]

BAD \neg q
Using Interpolation

\[ INIT(V_0) \land T(V_0, V_1) \land T(V_1, V_2) \land (\neg q(V_1) \lor \neg q(V_2)) \]

\[ I'_1(V_0) \land T(V_0, V_1) \land T(V_1, V_2) \land (\neg q(V_1) \lor \neg q(V_2)) \]

\[ \vdots \]

\[ I'_k(V_0) \land T(V_0, V_1) \land T(V_1, V_2) \land (\neg q(V_1) \lor \neg q(V_2)) \]
The Analogy to Forward Reachability Analysis

\[ \text{INIT}_2(V_0) \wedge T(V_0, V_1) \wedge T(V_1, V_2) \wedge (\neg q(V_{11}) \wedge T_{22}(V_{22})) \]
• If BMC finds a satisfying assignment the counterexample might be spurious
  – The set of initial states is over-approximated

• Increase k and start with the original INIT
Characteristics

• When calculating the interpolant for the i-th iteration, for bound $k$ the following holds:
  - The interpolant represents an over-approximation of reachable states after $i$ transitions
  - Also, it cannot reach a bad state in $k-1+i$ steps or less
    • It is similar to $I_i$ calculated in ISB after $k+i$ iterations
Interpolation Based [McM03] versus Interpolation-Sequence Based [FMCAD09]

- The computation itself is different
  - Uses interpolation, not interpolation sequence
  - Based on nested loops
  - Not incremental

- The computed over-approximated sets are different.
Experimental Results

• Experiments were conducted on two future CPU designs from Intel (two different architectures)
Experimental Results - Falsification
Experimental Results - Verification

![Graph showing experimental results](image-url)
<table>
<thead>
<tr>
<th>Spec</th>
<th>#Vars</th>
<th>Bound (Ours)</th>
<th>Bound (M)</th>
<th>#Int (Ours)</th>
<th>#Int (M)</th>
<th>#BMC (Ours)</th>
<th>#BMC (M)</th>
<th>Time [s] (Ours)</th>
<th>Time [s] (M)</th>
</tr>
</thead>
<tbody>
<tr>
<td>F₁</td>
<td>3406</td>
<td>16</td>
<td>15</td>
<td>136</td>
<td>80</td>
<td>16</td>
<td>80</td>
<td>970</td>
<td>5518</td>
</tr>
<tr>
<td>F₂</td>
<td>1753</td>
<td>9</td>
<td>8</td>
<td>45</td>
<td>40</td>
<td>9</td>
<td>40</td>
<td>91</td>
<td>388</td>
</tr>
<tr>
<td>F₃</td>
<td>1753</td>
<td>16</td>
<td>15</td>
<td>136</td>
<td>94</td>
<td>16</td>
<td>94</td>
<td>473</td>
<td>1901</td>
</tr>
<tr>
<td>F₄</td>
<td>3406</td>
<td>6</td>
<td>5</td>
<td>21</td>
<td>13</td>
<td>6</td>
<td>13</td>
<td>68</td>
<td>208</td>
</tr>
<tr>
<td>F₅</td>
<td>1761</td>
<td>2</td>
<td>1</td>
<td>3</td>
<td>2</td>
<td>2</td>
<td>2</td>
<td>5</td>
<td>4</td>
</tr>
<tr>
<td>F₆</td>
<td>3972</td>
<td>3</td>
<td>1</td>
<td>6</td>
<td>3</td>
<td>3</td>
<td>3</td>
<td>19</td>
<td>14</td>
</tr>
<tr>
<td>F₇</td>
<td>2197</td>
<td>3</td>
<td>1</td>
<td>6</td>
<td>3</td>
<td>3</td>
<td>3</td>
<td>2544</td>
<td>1340</td>
</tr>
<tr>
<td>F₈</td>
<td>4894</td>
<td>5</td>
<td>1</td>
<td>15</td>
<td>3</td>
<td>5</td>
<td>3</td>
<td>635</td>
<td>101</td>
</tr>
</tbody>
</table>
Analysis

- False properties is always faster.
- True properties – results vary. Heavier properties favor ISB where the easier favor IB.
- Some properties cannot be verified by one method but can be verified by the other and vice-versa.
Conclusions

• A new SAT-based method for **unbounded** model checking.
  - BMC is used for falsification.
  - Simulating forward reachability analysis for verification.

• Method was successfully applied to industrial sized systems.
Additional comments:

- **Interpolation and interpolation sequence:** defined for additional logics, not just propositional logic

- **Interpolation sequence was suggested in** “Lazy abstraction with interpolation”, McMillan, *CAV 2006*
Thank you!