Interpolation-Sequence Based Model Checking

Yakir Vizel and Orna Grumberg
FMCAD 2009

Verification course, June 2015
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:
    \[ S_j \subseteq \bigcup_{i=1}^{j} \bigcup_{j-1} S_i \]
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 \(k\)?

\[
\begin{align*}
\text{INIT}(V_0) \land \neg p(V_0) \\
\text{INIT}(V_0) \land T(V_0, V_1) \land \neg p(V_1) \\
\text{INIT}(V_0) \land T(V_0, V_1) \land T(V_1, V_2) \land \neg p(V_2) \\
\vdots \\
\vdots \\
\vdots \\
\text{INIT}(V_0) \land T(V_0, V_1) \land T(V_1, V_2) \land \ldots \land T(V_{k-1}, V_k) \land \neg p(V_k)
\end{align*}
\]
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
A Bit of Intuition

INIT $\Rightarrow S_1 \Rightarrow S_2 \Rightarrow S_3$

$\Rightarrow I_1 \Rightarrow I_2 \Rightarrow I_3$

BAD $\neg p$
Interpolation

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

$$A \Rightarrow I$$

$$I \land B = \text{false}$$

$I$ refers only to common variables of $A,B$

[Craig 57]
Interpolation in the context of model checking

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

\[ A \Rightarrow I \]
\[ I \land B \equiv false \]

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

- $I$ is over $V_1$
- $A \Rightarrow I$
  - $I$ 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:

\[ \begin{align*}
&\text{INIT}(V_0) \land T(V_0, V_1) \land \ldots \land T(V_{k-1}, V_k) \land \neg p(V_k) \\
&\quad \Downarrow \\
&I_0 = \text{true}, I_{k+1} = \text{false} \\
&I_{j-1} \land A_j \Rightarrow I_j \\
I_j \text{ is over the common variables of } A_1, \ldots, A_j \text{ and } A_{j+1}, \ldots, A_{k+1}, \text{ i.e. } V_j
\end{align*} \]
Interpolation-Sequence

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

• $I_k \wedge A_{k+1} \Rightarrow \text{false}$
  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 $1 \leq j < n$
  - $A(j) = A_1 \land \ldots \land A_j$
  - $B(j) = A_{j+1} \land \ldots \land A_n$
  - $I_j$ 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:
  \[ M \not\models AGp \]

• 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

\[
\text{INIT}(V_0) \land T(V_0, V_{11}) \land \mathcal{E}(V_{11}, V_{22}) \land \mathcal{E}(V_{22}, V_3) \land \neg p(V_3)
\]
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:
  $$\hat{I} = (I_1, I_2, \ldots, I_N)$$
Each $I_j^k$ over-approximates $S_j$
- Their conjunction results in a more accurate over-approximation

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

\[ j = 1 \]

while (j < k) do

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

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

\[ I^k[j] = I_j^k \]

end while

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

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

\[ j = 2 \]

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

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

\[ \alpha = I_j \land \neg R \quad \text{// negation of } I_j \Rightarrow R \]

\[ \text{if } (\text{SAT}(\alpha) = \text{false}) \text{ then return true} \]

\[ \text{end if} \]

\[ j = j + 1 \]

\[ \text{end while} \]

\[ \text{return false} \]

end function
Function ISB(M, f)  // f = AGq
k = 0
result = BMC (M, f, 0)
if (result == cex) then return cex
\( \widehat{I} = \phi \) // the reachability vector
while (true) do
    k = k+1
    result = BMC (M, f, k)
    if (result==cex) then return cex
    \( \widehat{I}^k = ( T, I_1^k, \ldots, I_k^k, F ) \)
    UpdateReachable (\( \widehat{I} \), \( \widehat{I}^k \))
    if ( FixpointReached (\( \widehat{I} \)) == 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

\[\begin{aligned}
A & \quad \text{INIT}(V_0) \land T(V_0, V_1) \land T(V_1, V_2) \land \ldots \land T(V_{k-1}, V_k) \land (\neg q(V_1) \lor \ldots \lor \neg q(V_k)) \\
B & \quad I \\
A \Rightarrow I \\
I \land B & \equiv F
\end{aligned}\]

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

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

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

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

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

\[ \text{INIT} (V_0) \land T(V_0, V_1) \land T(V_1, V_2) \land \left( \neg q \right) \]
- If BMC finds a satisfying assignment the \textbf{counterexample} might be \textbf{spurious}.
  - The set of initial states is over-approximated.
- \textbf{Increase k} and start with the original \textbf{INIT}. 

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 Model Checking [McM03]

- 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
## Experiments Results - Analysis

<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>$F_1$</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>$F_2$</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>$F_3$</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>$F_4$</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>$F_5$</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>$F_6$</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>$F_7$</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>$F_8$</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 vise-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!