Interpolant Learning and Reuse in SAT-Based Model Checking

Joao Marques-Silva
Electronics & Computer Science
University of Southampton

BMC Workshop, FLoC, Seattle, August 2006
Motivation

- Remarkable improvements made to SAT solvers over the last decade
  - Clause learning; lazy data structures; adaptive branching heuristics; search restarts

- Very successful application of SAT in model checking
  - Bounded and unbounded model checking

- Existing (industry motivated) challenges
  - Ability to handle ever increasing systems
  - Ability to find deep counterexamples
  - Ability to prove difficult properties

- Lines of research
  - More efficient SAT solvers (?)
  - Better uses of SAT technology in SAT-based model checking
Goals of this talk

- SAT & SAT-based model checking
- Interpolants in SAT-based model checking
- Conditions for interpolant learning and reuse
Outline

- SAT & SAT-based model checking
  - Organization of a modern SAT solver
  - SAT-based bounded model checking (BMC)
  - Interpolant-based unbounded model checking (UMC)

- Improvements to SAT-based model checking

- Results & conclusions
Modern SAT algorithms

- Follow the organization of the DPLL algorithm
  - Backtrack search with unit propagation
  
- Several key techniques are used:
  - Clause learning \[\text{[Marques-Silva&Sakallah’96]}\]
    - Infer new clauses from causes of conflicts
      - Allows implementing non-chronological backtracking
  
  - Exploiting structure of conflicts \[\text{[Marques-Silva&Sakallah’96]}\]
    - Identify Unique Implication Points (UIPs)
      - Dominators in graph of implied assignments
  
  - Optimized data structures \[\text{[Moskewicz et al.’01]}\]
    - Lazy evaluation of clause state
  
  - Adaptive branching heuristics \[\text{[Moskewicz et al.’01]}\]
    - Variable branching metrics are affected by number of conflicts
    - Aging mechanisms for focusing on most recent conflicts
  
  - Search restarts \[\text{[Gomes,Selman&Kautz’98]}\]
    - Opportunistically restart backtrack search
Bounded model checking

- Verification of safety properties: \( F \varphi \)
  
  \[
  \Phi^k = I_0(Y_0) \land \bigwedge_{i=0}^{k-1} T(Y_i, Y_{i+1}) \land \bigvee_{i=r}^k f(Y_i)
  \]

- Characteristic functions for representing initial states and transition relation, respectively \( I_0 \) and \( T \)
  - Resulting Boolean formula: \( \Phi^k = I_0 \land U_k \land F_k \)
    - Where:
      \[
      U_k = \bigwedge_{j=0}^{k-1} T_j \\
      T_i = T(Y_i, Y_{i+1}) \\
      F_k = \left( \bigvee_{i=r}^k f_i \right) \\
      f_i = f(Y_i)
      \]
  - Interpretation:

\[\begin{array}{ccccccc}
I_0 & Y_0 & T_0 & Y_1 & T_1 & \ldots & T_{k-1} & Y_k & F_k
\end{array}\]
Bounded model checking

• A possible BMC algorithm:
  – Given some initial $k$
  – While $k \leq$ user-specified time-bound UB
    • Generate CNF formula $\varphi$ for $I_0 \land U_k \land F_k$
    • Invoke SAT solver on $\varphi$
    • If formula $\varphi$ is satisfiable, then a counterexample within $k$ time steps has been found
      – Return counterexample
    • Otherwise, increase $k$

• The BMC algorithm is incomplete
  – But complete if completeness threshold is known
Towards completeness

- Unbounded model checking
  - Utilization of induction
    - Standard BMC loop
      - Stop BMC loop for some i, if cannot have loop-free path of size i that can be reached from $I_0$ or if cannot have loop-free path of size i that can reach $F_k$
      - Maximum unfolding bounded by largest loop-free path
  - ...  

- Utilization of interpolants
  - BMC and Craig interpolants allow SAT-based computation of abstractions of reachable states
    - Avoid computing exact sets of reachable states
    - One of the most promising approaches in practice
      - Maximum unfolding bounded by largest shortest path between any two states

[Sheeran et al.'00]
[Chauhan et al.'02; Gupta et al.'03]
[McMillan'03]
Interpolants

Given two subsets of clauses A and B, assume $A \land B$ is unsatisfiable. Then, there exists an interpolant $A'$ for the pair $(A, B)$ with the following properties:

- $A$ implies $A'$
- $A' \land B$ is unsatisfiable
- $A'$ refers only to the common variables of $A$ and $B$
- Example:
  - $A = p \land q$, $B = \neg q \land r$
  - $A' = q$

Size of interpolants:

- Given a resolution refutation of $A \land B$, can compute interpolant for the pair $(A, B)$ in linear time on the size of the resolution refutation
  - SAT solvers can be instructed to output resolution refutation!

Computing interpolants:

- Different algorithms can be used
  - Pudlak’97, McMillan’03
Deriving resolution refutations

- For unsatisfiable formulas:
  - Learned clauses capture a resolution refutation from a subset of the original clauses
  - SAT solvers can be instructed to recreate resolution refutation for unsatisfiable formula

\[ \varphi = (a \lor b) \land (\neg a \lor c) \land (\neg b) \land (\neg c) \]

\[ \omega_1 \quad \omega_2 \quad \omega_3 \quad \omega_4 \]

\[ \kappa \]

\[ \omega_1 \quad \omega_2 \quad \omega_3 \quad \omega_4 \]

\[ \begin{align*}
\omega_1 \hspace{1cm} a &= 0 \\
\omega_2 \hspace{1cm} b &= 0 (\omega_3) \\
\omega_3 \hspace{1cm} c &= 0 (\omega_4) \\
\omega_4 \hspace{1cm} (a \lor b) \land (\neg a \lor c) \land (\neg b) \land (\neg c) \\
\end{align*} \]

\[ (b \lor c) \quad (\neg b) \quad (c) \quad (\neg c) \]

\[ \bot \]
Computing interpolants

$A = (r \lor y)(\neg r \lor x)$

$B = (\neg y \lor a)(\neg y \lor \neg a)(\neg x)$

- Interpolant is a Boolean circuit that follows structure of resolution refutation
  - Can map circuit into CNF in linear time and space

$A' = y + x$

A implies $A'$; $A' \land B$ is unsatisfiable

$A'$ with variables common to $A$ and $B$
Abstraction of reachable states

- For each iteration of BMC loop, call to SAT solver returns unsat unless counterexample is found
  - Analysis of resolution refutation yields abstractions of reachable states
    \[ \Phi = I_0 \land T_0 \land T_1 \land \square \land T_{k-1} \land F_k = A \land B \]
    \[ A = I_0 \land T_0 \]
    \[ B = T_1 \land \square \land T_{k-1} \land F_k \]

- Given A and B, and a resolution refutation for \( A \land B \), compute Craig interpolant \( A' \):
  - A = \( I_0 \land T_0 \) implies \( A' \)
  - \( A' \land B \) is unsatisfiable
  - \( A' \) solely represented with state variables
    - If A holds, then \( A' \) holds
      - \( A_1 = A' \) represents abstraction of states reachable from \( I_0 \) in 1 time step!
Fixpoint of reachable states

- Can iterate computation of interpolants:

If $A_i \rightarrow I_0 \lor A_1 \lor A_2 \lor \ldots \lor A_{i-1}$, then a fixpoint is reached; all reachable states identified!
If $F_k$ is satisfied from $I_0$, then we have a counterexample!

If a fixpoint of the reachable states is identified, then no reachable state can satisfy property!

If $A \land B$ is sat, may have abstracted too much; must unfold more time steps

Maximum value of $k$ is bounded by largest shortest path between any two states

**UMC algorithm**

$k = 0$

repeat

if from $I_0$ can satisfy $F_k$ within $k$ steps

return reachable

$R = I_0$

let $A = I_0 \cup T_0$, and $B = T_1 \cup T_2 \cup \ldots \cup T_{k-1} \cup F_k$

while $A \cup B = \text{false}$

$P = \text{unsat\_proof}(A \cup B)$

$A' = \text{interpolant}(P, A, B)$

if $A' \rightarrow R$, return unreachable

$R = A' \cup R$

$A = A' \cup T_0$

end while

increase $k$

end repeat

**BMC loop**
Outline

- SAT & SAT-based model checking

- Conditions for interpolant reuse
  - Interpolants readily available if fixed point condition is based on interpolants
  - Can envision alternative fixpoint conditions

- Results & conclusions
Interpolant reuse

- Boolean formula $N$ is usable for $B$ iff $B \rightarrow N$
  - $B$ satisfiable iff $B \land N$ satisfiable

- Learnt interpolants can be **reused**
  - For requiring states from a set of states
  - For preventing states from a set of states

- A different organization of BMC:
  \[ U_k = \bigwedge_{i=0}^{k-1} T_i \]
  \[ F_k = \bigvee_{i=k}^{k} f_i \]

[Copty et al.’01]
Interpolant reuse

- Different ways for computing interpolants
  - Computed interpolants can be direct or inverse
  - Interpolants can be computed at different time steps

![Diagram]

- Direct interpolants
  - Over-approximation of reachable states
  - Under-approximation of states that do not satisfy failing property

- Inverse interpolants
  - Under-approximation of unreachable states
  - Over-approximation of states that satisfy failing property
Direct interpolants

- $P_{r,t}$:
  - Direct interpolant computed $r$ time steps from $I_0$ and $t$ time steps to $F_k$

- From the initial state, $P_{r,t}$, $t=k-r$:
  \[
  \Phi = I_0 \land T_0 \land T_1 \land \Box \land T_{k-1} \land F_k = A \land B \\
  B = T_r \land \Box \land T_{k-1} \land F_k \\
  A = I_0 \land T_1 \land \Box \land T_{r-1}
  \]

- In general, $P_{r+u,t}$:
  \[
  \Phi = P_{u,v} \land T_0 \land T_1 \land \Box \land T_{k-1} \land F_k = A \land B \\
  B = T_r \land \Box \land T_{k-1} \land F_k \\
  A = P_{u,v} \land T_1 \land \Box \land T_{r-1}
  \]
Conditions for interpolant reuse I

● Conditions on direct interpolants:
  
  – \( P_{r,t}(Y_r) \) is usable for \( \Phi^k \), with \( t \geq 0 \) and \( r \leq k \)
  
  – \( \neg P_{r,t}(Y_{k-t}) \) is usable for \( \Phi^k \), with \( r \geq 0 \) and \( t \leq k \)

\[
\begin{align*}
I_0 & \quad T_0 & \quad T_{r-1} & \quad T_r & \quad T_{k-t-1} & \quad T_{k-t} & \quad T_{k-1} & \quad F_k \\
\quad & & & \quad P_{r,t} & & \neg P_{r,t} & & \quad
\end{align*}
\]
An example I

- Standard UMC model checking, with BMC and fixpoint loops
- Automaton with unfolding of size $k+1$

\[
\begin{array}{c}
I_0 \quad Y_0 \quad T_0 \quad Y_1 \quad T_1 \quad \ldots \quad T_k \quad Y_{k+1} \quad F_{k+1}
\end{array}
\]

- Fixed point checking for $j+1$ iterations
  - Last iteration yields spurious counterexample; $j$ interpolants computed
- Interpolants computed at $Y_1$:
  - $P_{1,k}$, $P_{2,k}$, ..., $P_{j,k}$
- Examples of interpolant reuse:
  - $P_{i,k}(Y_i)$, $1 \leq i \leq j$, is usable for $\Phi^m$, $m \geq k$
    - $P_{i,k}$ represents over-approximation of the states reachable in $i$ time steps
  - With unfolding of size $k+1$, $\neg P_{i,k}(Y_1)$, $1 \leq i \leq j$, is usable for $\Phi^{k+1}$
    - $P_{i,k}$ represents under-approximation of the states that do not satisfy failing property in $k$ time steps
  - With unfolding of size $m \geq k$, $\neg P_{i,k}(Y_{m-k})$, $1 \leq i \leq j$, is usable for $\Phi^m$
Inverse interpolants

- \( Q_{r,t} \):
  - Reverse interpolant computed \( r \) time steps from \( I_0 \) and \( t \) time steps to \( F_k \)

- From the initial state, \( Q_{r,t}, t=k-r \):
  \[
  \Phi = I_0 \land T_0 \land T_1 \land \cdots \land T_{k-1} \land F_k = A \land B \\
  A = T_r \land \cdots \land T_{k-1} \land F_k \\
  B = I_0 \land T_1 \land \cdots \land T_{r-1}
  \]

- In general, \( Q_{r+u,t} \):
  \[
  \Phi = P_{u,v} \land T_0 \land T_1 \land \cdots \land T_{k-1} \land F_k = A \land B \\
  A = T_r \land \cdots \land T_{k-1} \land F_k \\
  B = P_{u,v} \land T_1 \land \cdots \land T_{r-1}
  \]
Conditions for interpolant reuse II

- **Conditions on inverse interpolants:**
  - $Q_{r,t}(Y_{k-t})$ is usable for $\Phi^k$, with $r \geq 0$ and $t \leq k$
  - $\neg Q_{r,t}(Y_r)$ is usable for $\Phi^k$, with $t \geq 0$ and $r \leq k$
An example II

- Standard UMC model checking, with BMC and fixpoint loops
- Automaton with unfolding of size k+1

\[ \text{I}_0 \xrightarrow{Y_0} \text{T}_0 \xrightarrow{Y_1} \text{T}_1 \xrightarrow{\ldots} \text{T}_k \xrightarrow{Y_{k+1}} \text{F}_{k+1} \]

- Fixed point checking for j+1 iterations
  - Last iteration yields spurious counterexample; j interpolants computed
- Interpolants computed at $Y_1$:
  - $Q_{1,k}$, $Q_{2,k}$, ..., $Q_{j,k}$
- Examples of interpolant reuse:
  - $\neg Q_{i,k}(Y_i), 1 \leq i \leq j$, is usable for $\Phi^m$, $m \geq k$
    - $Q_{i,k}$ represents under-approximation of the states unreachable in i time steps
  - With unfolding of size k+1, $Q_{i,k}(Y_1), 1 \leq i \leq j$, is usable for $\Phi^{k+1}$
    - $Q_{i,k}$ represents over-approximation of the states that satisfy failing property in k time steps
  - With unfolding of size $m \geq k$, $Q_{i,k}(Y_{m-k}), 1 \leq i \leq j$, is usable for $\Phi^m$
More on interpolant reuse

- All interpolants computed in standard interpolant-based UMC flow can be reused
  - Easy to integrate with existing interpolant-based UMC flow

- Learning and reusing of interpolants can be integrated into any approach for BMC or UMC
  - Plain BMC algorithm
  - Different approaches for UMC

- Inverse interpolants provide alternative fixpoint condition (from previous slide):
  - If $Q_{k,i} \rightarrow F_{k+1} \lor Q_{k,1} \lor \ldots \lor Q_{k,i-1}$ is satisfiable, then we have a fixpoint
Outline

- SAT & SAT-based model checking
- Improvements to SAT-based model checking
- Results & conclusions
Experience with reuse

- Experimented interpolant reuse on artificial and industrial benchmarks
  - Plain (incomplete) BMC loop
  - Direct interpolants computed at each step (at the last time step)
    - Interpolants not used for checking fixed point condition

- Experience so far:
  - CPU times increase with interpolant reuse

- The problems observed:
  - Large interpolants
    - Naive simplifications
    - Computed solely for search pruning purposes
  - Ineffective representation
    - One Reduced Boolean Circuit (RBC) for each interpolant
## Results

<table>
<thead>
<tr>
<th>Instance</th>
<th>W/o Interpolants</th>
<th>W/ Interpolants</th>
</tr>
</thead>
<tbody>
<tr>
<td>6-bit counter</td>
<td>1.51</td>
<td>5.29</td>
</tr>
<tr>
<td>7-bit counter</td>
<td>16.38</td>
<td>61.03</td>
</tr>
<tr>
<td>8-bit counter</td>
<td>236.90</td>
<td>784.81</td>
</tr>
<tr>
<td>(I_1)</td>
<td>7.08</td>
<td>7.11</td>
</tr>
<tr>
<td>(I_2)</td>
<td>31.36</td>
<td>36.96</td>
</tr>
<tr>
<td>(I_3)</td>
<td>38.36</td>
<td>60.60</td>
</tr>
<tr>
<td>(I_4)</td>
<td>52.45</td>
<td>58.25</td>
</tr>
<tr>
<td>(I_5)</td>
<td>150.54</td>
<td>157.81</td>
</tr>
</tbody>
</table>
Conclusions

- SAT technology has improved dramatically over the last decade
  - Key techniques:
    - Clause learning, optimized data structures, adaptive branching heuristics, search restarts

- SAT has been applied to model checking with success
  - Bounded and unbounded model checking

- Described conditions for interpolant reuse in SAT-based model checking

- Results preliminary
  - Reuse of interpolants increases run times
A few challenges

- Can interpolant reuse yield performance gains?
  - E.g., Negative direct interpolants OR Positive inverse interpolants

- Can we find “good” interpolants to learn and reuse?
  - E.g. size/depth of interpolant (or size of CNF representation)
An example III

- **Inverse** UMC model checking, with BMC and fixpoint loops
- Automaton with unfolding of size \(k+1\)

\[
\begin{array}{c}
I_0 & Y_0 & T_0 & Y_1 & T_1 & \cdots & T_k & Y_{k+1} & F_{k+1}
\end{array}
\]

- Fixed point checking for \(j+1\) iterations
  - Last iteration yields spurious counterexample; \(j\) interpolants computed
- Interpolants computed at \(Y_{k-1}\):
  - \(Q_{k,1}, Q_{k,2}, \ldots, Q_{k,j}\)
- Examples of interpolant reuse:
  - \(\neg Q_{k,i}(Y_k), 1 \leq i \leq j,\) is usable for \(\Phi^m, m \geq k\)
    - \(Q_{k,i}\) represents under-approximation of the states unreachable in \(k\) time steps
  - With unfolding of size \(k+1\), \(Q_{k,i}(Y_{k+1-i}), 1 \leq i \leq j,\) is usable for \(\Phi^{k+1}\)
    - \(Q_{k,i}\) represents over-approximation of the states that satisfy failing property in \(i\) time steps
  - With unfolding of size \(m \geq k\), \(Q_{k,i}(Y_{m-i}), 1 \leq i \leq j,\) is usable for \(\Phi^m\)