Introduction to Formal Methods for SW and HW Development

10: SAT Based Abstraction/Refinement in Model-Checking

Roberto Sebastiani

Based on work and slides by E. Clarke, A. Gupta, J. Kukula, O. Strichman (CAV’02) revisions by M. Roveri, R. Sebastiani and S. Tonetta
Outline

- Preliminaries
- Notion of Abstraction
- Over and under-approximation, simulation, bisimulation
- Counter-example based abstraction refinement
Model Checking

Given a:
- Finite transition system $M (S, I, R, L)$
- A temporal property $p$

The model checking problem:
- Does $M$ satisfy $p$?

$M \models p$
Model Checking

Temporal properties:

- “Always when a train arrives the bar is not up”
  \( G (\text{train\_arriving} \rightarrow \neg \text{bar\_up}) \)
- “Every Send is followed immediately by Ack”
  \( G(\text{Send} \rightarrow X \text{ Ack}) \)
- “Reset can always be reached”
  \( GF \text{ Reset} \)
- “From some point on, always switch\_on”
  \( FG \text{ switch\_on} \)

“Safety” properties

“Liveness” properties
Model Checking (safety)

Add reachable states until reaching a fixed-point

= bad state
Model Checking (safety)

PROBLEM: Too many states to handle!

= bad state
Abstraction

Abstraction Function \( h : S \rightarrow S' \)
Abstraction Function

- Partition variables into \textit{visible}(\mathcal{V}) and \textit{invisible}(\mathcal{I}) variables.

- The abstract model consists of \mathcal{V} variables. \mathcal{I} variables are made inputs.

- The abstraction function maps each state to its projection over \mathcal{V}.
Abstraction Function

<p>| | | | | |</p>
<table>
<thead>
<tr>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>0</td>
<td>0</td>
<td>0</td>
<td>1</td>
<td>0</td>
</tr>
<tr>
<td>0</td>
<td>0</td>
<td>0</td>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>0</td>
<td>0</td>
<td>0</td>
<td>1</td>
<td>1</td>
</tr>
</tbody>
</table>

Group concrete states with identical visible part to a single abstract state.
Building an Abstract Model

$M'$ can be computed efficiently if $M$ is in functional form, e.g. sequential circuits.
Building an Abstract Model (cont.)

next(x1) =
    f1(x1,x2,x3,x4,i1,i2);
next(x2) =
    f2(x1,x2,x3,x4,i1,i2);
next(x3) =
    f3(x1,x2,x3,x4,i1,i2);
next(x4) =
    f4(x1,x2,x3,x4,i1,i2);

next(x1) =
    f1(x1,x2,x3,x4,i1,i2);
next(x2) =
    f2(x1,x2,x3,x4,i1,i2);
abstract
next(x3) =
    f3(x1,x2,x3,x4,i1,i2);
next(x4) =
    f4(x1,x2,x3,x4,i1,i2);
Computing Abstractions

- $S$ – concrete state space
- $S'$ – abstract state space
- $\alpha: S \rightarrow S'$ - abstraction
- $\gamma: S' \rightarrow 2^S: \gamma(s') = \{s \mid s' = \alpha(s)\}$ - concretization (refinement)

Properties of $\alpha$ and $\gamma$:
- $\alpha(\gamma(A)) = A$, for $A$ in $S'$
- $\gamma(\alpha(B)) \supseteq B$, for $B$ in $S$

The above properties mean that $\alpha$ and $\gamma$ are Galois-connected
Aside: simulations

\[ M = (s_0, S, R, L) \]
\[ M' = (t_0, S', R', L') \]

Definition: \( p \) is a simulation between \( M \) and \( M' \) if

1. \( (s_0, t_0) \in p \)
2. \( \forall (s, t) \in p \) \( \forall (t, t_1) \in R' \), \( \exists (s, s_1) \in R \) s.t.
   \( (s_1, t_1) \in p \)

(We say that \( M \) simulates \( M' \).)

Intuitively, for every transition in \( M' \) there is a corresponding transition in \( M \)
Aside: bisimulation

\( M = (s_0, S, R, L) \)
\( M' = (t_0, S', R', L') \)

Definition: \( p \) is a **bisimulation** between \( M \) and \( M' \) if

1. \( p \) is a simulation between \( M \) and \( M' \)
2. \( p \) is a simulation between \( M' \) and \( M \)
Existential Abstraction (Over-Approximation): $M'$ simulates $M$
Model Checking Abstract Model

Let $\varphi$ be a universally-quantified property (i.e., expressed in LTL or ACTL) and $M'$ simulates $M$

Preservation Theorem

$M' \models \varphi \rightarrow M \models \varphi$

Intuition: if $M$ has a countermodel, $M'$ simulates it

The converse does not hold

$M' \not\models \varphi \nrightarrow M \not\models \varphi$

The counterexample may be spurious
Universal Abstraction (Under-Approximation): $M$ simulates $M'$
Model Checking Abstract Model

Let $\varphi$ be a existential-quantified property (i.e., expressed in ECTL) and $M$ simulates $M'$

Preservation Theorem

$M' \models \varphi \rightarrow M \models \varphi$

Intuition: if $M'$ has a model, $M$ simulates it

Converse does not hold

$M' \not\models \varphi \not\rightarrow M \not\models \varphi$
Model Checking Abstract Model

\[ M = (s_0, S, R, L) \] and \[ M' = (t_0, S', R', L') \] related by bisimulation

Then, for every CTL/LTL property \( \phi \):

\[ M' \models \phi \rightarrow M \models \phi \]
\[ M' \not\models \phi \rightarrow M \not\models \phi \]
Our specific problem

Let \( \varphi \) be a universally-quantified property (i.e., expressed in LTL or ACTL) and \( M' \) simulates \( M \)

- Preservation Theorem
  \[ M' \models \varphi \rightarrow M \models \varphi \]

- Converse does not hold
  \[ M' \not\models \varphi \rightarrow M \not\models \varphi \]

- Counter-examples may be spurious
Checking the Counterexample

Counterexample: \((c_1, \ldots, c_m)\)
- Each \(c_i\) is an assignment to \(\mathcal{V}\).

Simulate the counterexample on the concrete model.
Checking the Counterexample

Concrete traces corresponding to the counterexample \((c_1, \ldots, c_m)\):

\[
\phi \quad = \quad I(s_1) \quad \land \\
\land_{i=1}^{m-1} R(s_i, s_{i+1}) \quad \land \\
\land_{i=1}^{m} \text{visible}(s_i) \quad = \quad c_i
\]

(Initial State)

(Unrolled Transition Relation)

(Restriction of \(\forall\) to Counterexample)
Abstraction-Refinement Loop

Abstract → Model Check → Refine → Abstract

M, p, h → M', p → h' → h

Pass → No Bug
Fail → Spurious
Real → Bug

Check Counterexample
Refinement methods...

Localization
(R. Kurshan, 80’s)
Refinement methods...

**Intel’s refinement heuristic**  
(Glusman et al., 2002)

- Generate all counterexamples.
- Prioritize variables according to their consistency in the counterexamples.

![Diagram of counterexamples]

X1 x2 x3 x4
Refinement methods...

Abstraction/refinement with conflict analysis

(Chauhan, Clarke, Kukula, Sapra, Veith, Wang, FMCAD 2002)

- Simulate counterexample on concrete model with SAT
- If the instance is unsatisfiable, analyze conflict
- Make visible one of the variables in the clauses that lead to the conflict
Why spurious counterexample?

- Deadend states
- Bad States
- Failure State

The diagram illustrates the concept of spurious counterexamples in a decision-making process, highlighting the transition from bad states to failure states, potentially leading to incorrect conclusions.
Problem: Deadend and Bad States are in the same abstract state.

Solution: Refine abstraction function.

- The sets of Deadend and Bad states should be separated into different abstract states.
  - $\phi_D$ to represent the set of Deadend states
  - $\phi_B$ to represent the set of Bad states
Refinement

Refinement : $h'$
Let $f$ be the maximum value by which the following formula is satisfiable:

$$
\phi_D = I(s_1) \land \bigwedge_{i=1}^{f-1} R(s_i, s_{i+1}) \land \bigwedge_{i=1}^{f} \text{visible}(s_i) = c_i
$$
Refinement

$$\phi_B = R(s_f, s_{f+1}) \land$$
$$\text{visible}(s_f) = c_f \land \text{visible}(s_{f+1}) = c_{f+1}$$
Refinement as Separation

The state separation problem

Input: Sets $D$, $B$ of states (assignments)

Output: $\mathcal{U}$ (minimal) subset of $\mathcal{I}$ s.t.:

$$\forall \, d \in D, \forall \, b \in B, \exists \, u \in \mathcal{U}. \quad d(u) \neq b(u)$$

The refinement $h'$ is obtained by adding $\mathcal{U}$ to $\mathcal{V}$. 
Refinement as Separation

Refinement: Find subset $U$ of $I$ that separates between all pairs of deadend and bad states. Make them visible.

Keep $U$ small!
Refinement as Separation

Refinement: Find subset $U$ of $I$ that separates between all pairs of deadend and bad states. Make them visible.

Keep $U$ small!
Two separation methods

- **ILP-based separation**
  - Minimal separating set.
  - Computationally expensive.

- **Decision Tree Learning based separation.**
  - Not optimal.
  - Polynomial.
Separation with Decision Tree learning (Example)

Classification:

<table>
<thead>
<tr>
<th></th>
<th>D</th>
<th>B</th>
</tr>
</thead>
<tbody>
<tr>
<td>$d_1$</td>
<td>$(0, 1, 0, 1)$</td>
<td>$(1, 1, 1, 1)$</td>
</tr>
<tr>
<td>$d_2$</td>
<td>$(1, 1, 1, 0)$</td>
<td>$(0, 0, 0, 1)$</td>
</tr>
</tbody>
</table>

Separating Set: $\{v_1, v_2, v_4\}$
Separation with 0-1 ILP

\[ \text{Min } \sum_{i=1}^{\left| \mathcal{I} \right|} v_i \]

subject to: \((\forall d \in D) (\forall b \in B) \sum_{1 \leq i \leq |\mathcal{I}|, d, b \text{ differ at } v_i} v_i \geq 1 \]

- One constraint per pair of states.
- \(v_i = 1\) iff \(v_i\) is in the separating set.
Separation with 0-1 ILP (Example)

\[ d_1 = (0, 1, 0, 1) \quad b_1 = (1, 1, 1, 1) \]
\[ d_2 = (1, 1, 1, 0) \quad b_2 = (0, 0, 0, 1) \]

\[
\text{Min } \sum_{i=1}^{4} v_i
\]

subject to:

\[
v_1 + v_3 \geq 1 \quad /* \text{ Separating } d_1 \text{ from } b_1 */
\]
\[
v_2 \geq 1 \quad /* \text{ Separating } d_1 \text{ from } b_2 */
\]
\[
v_4 \geq 1 \quad /* \text{ Separating } d_2 \text{ from } b_1 */
\]
\[
v_1 + v_2 + v_3 + v_4 \geq 1 \quad /* \text{ Separating } d_2 \text{ from } b_2 */
\]
Refinement as Learning

- For systems of realistic size
  - Not possible to generate $D$ and $B$.
  - Expensive to separate $D$ and $B$.

- Solution:
  - Sample $D$ and $B$
  - Infer separating variables from the samples.

- The method is still complete:
  - counterexample will eventually be eliminated.
Efficient Sampling

Let $\delta(D, B)$ be the smallest separating set of $D$ and $B$.

Q: Can we find it without deriving $D$ and $B$?

A: Search for smallest $d, b$ such that $\delta(d, b) = \delta(D, B)$. 
Efficient Sampling

- Direct search towards samples that contain more information.

- How? Find samples not separated by the current separating set (Sep).
Efficient Sampling

Recall:
- $\phi_D$ characterizes the deadend states
- $\phi_B$ characterizes the bad states
- $\phi_D \land \phi_B$ is unsatisfiable

Samples that agree on the sep variables:

$$\Omega(Sep) \equiv \phi_D \land \phi_B' \land \bigwedge_{v_i \in Sep} v_i = v_i'$$

Rename all $v_i \in B$ to $v_i'$
Efficient Sampling

Sep = {}
d, b = {}

Run SAT solver on $\Omega(\text{Sep})$

Add samples to d and b

Compute Sep := $\delta(d, b)$

Sep is the minimal separating set of D and B

unsat

STOP
The Tool

- LpSolve
- Dec Tree
- NuSMV
- SAT
- Chaff
- Sep
- Cadence SMV
- MC
## Results

### Property 1

<table>
<thead>
<tr>
<th>Circuit</th>
<th>SMV</th>
<th>ILP</th>
<th>DTL</th>
<th>Eff. Samp./ DTL</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>Time</td>
<td>Time</td>
<td>I</td>
<td>L</td>
</tr>
<tr>
<td>IU30</td>
<td>7.3</td>
<td>8.0</td>
<td>3</td>
<td>20</td>
</tr>
<tr>
<td>IU35</td>
<td>19.1</td>
<td>11.8</td>
<td>4</td>
<td>21</td>
</tr>
<tr>
<td>IU40</td>
<td>53.6</td>
<td>25.9</td>
<td>6</td>
<td>23</td>
</tr>
<tr>
<td>IU45</td>
<td>226.1</td>
<td>28.3</td>
<td>5</td>
<td>22</td>
</tr>
<tr>
<td>IU50</td>
<td>1754</td>
<td>160.4</td>
<td>13</td>
<td>32</td>
</tr>
<tr>
<td>IU75</td>
<td>-</td>
<td>1080</td>
<td>21</td>
<td>38</td>
</tr>
<tr>
<td>IU80</td>
<td>-</td>
<td>1136</td>
<td>21</td>
<td>38</td>
</tr>
<tr>
<td>IU85</td>
<td>-</td>
<td>1162</td>
<td>21</td>
<td>38</td>
</tr>
<tr>
<td>IU90</td>
<td>-</td>
<td>965</td>
<td>20</td>
<td>37</td>
</tr>
</tbody>
</table>
## Results

**Property 2**

<table>
<thead>
<tr>
<th>Circuit</th>
<th>SMV Time</th>
<th>Sampling - ILP</th>
<th>Sampling - DTL</th>
<th>Eff. Samp. - DTL</th>
</tr>
</thead>
<tbody>
<tr>
<td>IU30</td>
<td>0.7</td>
<td>0.1</td>
<td>0.1</td>
<td>0.1</td>
</tr>
<tr>
<td>IU35</td>
<td>0.6</td>
<td>0.1</td>
<td>0.1</td>
<td>0.1</td>
</tr>
<tr>
<td>IU40</td>
<td>1.2</td>
<td>6.3</td>
<td>0.9</td>
<td>0.9</td>
</tr>
<tr>
<td>IU45</td>
<td>37.5</td>
<td>6.1</td>
<td>1.1</td>
<td>1.1</td>
</tr>
<tr>
<td>IU50</td>
<td>23.3</td>
<td>19.7</td>
<td>9.8</td>
<td>24.0</td>
</tr>
<tr>
<td>IU55</td>
<td>-</td>
<td>-</td>
<td>2072</td>
<td>3.0</td>
</tr>
<tr>
<td>IU60</td>
<td>-</td>
<td>7.8</td>
<td>7.8</td>
<td>4.5</td>
</tr>
<tr>
<td>IU65</td>
<td>-</td>
<td>7.9</td>
<td>7.9</td>
<td>3.8</td>
</tr>
<tr>
<td>IU70</td>
<td>-</td>
<td>8.1</td>
<td>8.2</td>
<td>3.8</td>
</tr>
<tr>
<td>IU75</td>
<td>102.9</td>
<td>32.0</td>
<td>24.5</td>
<td>24.1</td>
</tr>
<tr>
<td>IU80</td>
<td>603.7</td>
<td>31.7</td>
<td>44.0</td>
<td>24.1</td>
</tr>
<tr>
<td>IU85</td>
<td>2832</td>
<td>33.1</td>
<td>44.6</td>
<td>25.2</td>
</tr>
<tr>
<td>IU90</td>
<td>-</td>
<td>33.0</td>
<td>44.6</td>
<td>25.4</td>
</tr>
</tbody>
</table>

- Efficient Sampling together with Decision Tree Learning performs best.
- Machine Learning techniques are useful in computing good refinements.