Joint work with Pankaj Chaunhan, Jim Kukula, Samir Sapat, Helmut Veith, Dong Wang

Computer Science Department, Carnegie Mellon University

Edmund M. Clarke

by SAT Based Conflict Analyses
Automated Abstraction Refinement
\[ (M, I, R) \] Transition system

\[ S \times S \supseteq R \] Set of transitions

\[ S \supseteq I \] Set of initial states

\[ u_x D \times \cdots \times x_1 D = S \] Set of states

\[ \{ u_x, \cdots, x_1 \} = \Lambda \] Set of variables

Abstract in Model Checking
\{ z^2 = (z^2)h \lor z = (z)h \lor (z^2)R \cdot z \cdot s_E \cdot s_E \mid (s^2, z) \} = y

\{ s = (s)h \lor (s)I \cdot s_E \mid s \} = I

\{ s = (s)h \lor s \ni s \cdot s_E \mid s \} = S

\( (y, I') = I \quad S \leftarrow S : h \)

Abstract Model

Abstract Abstraction Refinement by SAT Based Conflict Analysis
Propositional formula that respects $\eta$. Then let $\mathcal{A}$ be an abstraction of $\mathcal{M}$ corresponding to the abstraction function $\eta$, and be a $d$.

**Preservation Theorem**
Counterexample is spurious. Abstraction is too coarse.

\[ d \not\equiv M \not\equiv d \mathrm{ACP} \not\equiv M \]

**Converse of Preservation Theorem**

Automated Abstraction Refinement by SAT Based Conflict Analysis
\( \forall y \neq (Is)_I y \text{ and } (Iz)_I y = (Is)_I y \text{ such that } S \in I. \forall z \in I, S \in I. \forall y \in S. I. \forall z \in \text{ implies } (Iz)_I y = (Is)_I y \Rightarrow y \) is a refinement of \( I. \)
\[ ((S_2) \cup \neg (S_2), y) \neq (S_1, y) \text{ and } (S_2) \setminus y = (S_1), y \text{ such that } S \in S_2 \in S, \text{ s.t. } \exists v \in S_2, S_1, y \in v \implies (y) = (S_1, y) = (S_2, y) \text{ implies } y \text{ is a refinement of } y \]

Refinement
4. Refine \( h \), and go to step 2.

\( \phi \not\models \mathcal{M} \). Return FALSE.

3. If \( \mathcal{M} \) is real, \( \phi \not\models \mathcal{M} \), check the counterexample on the concrete model. If the
counterexample is real, \( \mathcal{M} \). Return TRUE.

2. Build abstract machine \( \mathcal{M} \). If \( \mathcal{M} \) is not satisfied, then \( \phi \models \mathcal{M} \). Return TRUE.

1. Generate an initial abstraction function \( h \).

**Abstract-Refinement**
• ATPG+SAT [WHLK + DAC'01', ...
• Sequential refinement of approximations [dd LIC'S01, Gd ICC'00],
• Prequantification of variables
  – No expensive separation required, so one SAT check
  – Use SAT, not expensive BDD based validation of counterexamples
• Related closely to [CGLE CAV'00] and [CGKS CAV'02]
• Earliest work on localiziation reduction [Kurschun, BS-V CAV'99, L-NA CAV'99]

Related Work
Refinement : Move variables from \( \mathcal{I} \) to \( \mathcal{L} \).

\[
\begin{array}{c}
\begin{array}{cccc}
0 & 0 & 0 & 1 \\
1 & 1 & 0 & 0 \\
0 & 1 & 0 & 0 \\
1 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 \\
\end{array}
\end{array}
\]

and the abstraction function is

\[
(\forall \alpha)s \cdots (\exists \alpha)s = (s)\eta
\]

The set of abstract states is

\[
\{\forall \alpha, \cdot \cdot \cdot , \exists \alpha\} = \mathcal{L}
\]

Partition variables \( \mathcal{L} \) into visible and invisible variables.

Abstraction Function
Building Abstract Model

Automated Abstraction Refinement by SAT Based Conflict Analysis
\[ (\forall \mathcal{S} \subseteq \mathcal{S}, \forall f) \ni (\forall \mathcal{S} \subseteq \mathcal{S}) \quad \exists \mathcal{S} \in \mathcal{S} \quad \mathcal{S} = (\mathcal{S}, \mathcal{S}) \]

- Theorems: Keeps correlations between variables, no approximation
- Each variable is made a non-deterministic input
- abstraction by making inputs
\[(\forall x \leftrightarrow \exists x) \lor (\forall x \leftrightarrow \exists x) \lor (\exists x \lor \exists x \leftrightarrow Ix) = (\forall s, s)H\]
\[\{\forall x\} = Is\]
\[\{\exists x, \exists x, \exists x\} = \Lambda s\]
\[\forall x \lor \exists x \leftrightarrow \forall x\]
\[\forall x \leftrightarrow \exists x\]
\[\forall x \leftrightarrow \exists x\]
\[\exists x \lor \exists x \leftrightarrow Ix\]

Example:

Abstracting by Making is Inputs
Abstraction by Making $S_I$ Inputs

Problems:

- Too many variables in image computation
- Backward image computation especially expensive [Wang et al., DAC'04]

To alleviate the problems, remove the invisible variables from $T$ entirely by quantification.
\[
(\exists_s s') (\forall s' s'') (\exists x) \text{ s.t. } H \leftrightarrow (s' s'' s) (x) = f \in \mathcal{L} \in \mathcal{E} \land \exists x \forall \mathcal{E} (H(s' s'))
\]

Theorem: \( \mathcal{E} \approx \mathcal{L} \in \mathcal{E} \)

Approximate: \( \mathcal{E} \approx \mathcal{L} \in \mathcal{E} \)

Too expensive to compute!

(\exists_s s') \exists x f = f \in \mathcal{L} \in \mathcal{E} \land \exists x \forall \mathcal{E} (H(s' s'))

Exact existential abstraction: \( \mathcal{E} \approx \mathcal{L} \in \mathcal{E} \)

Quantity away invisible variables from \( \mathcal{L} \) in the beginning

**Existential Abstraction**
Still not an exact abstraction — preserve some correlations.

We propose \textsc{VarScore} based [CCKS+17, ICCAD’01] approximation algorithm for existential abstraction:

- Should keep correlated conjunctions together.
- Solution: Use smarter quantification scheduling for deriving this approximation.

Problem: By existentially quantifying away variables, we may lose correlations.

Existental Abstraction
Upon the case

5. If the BDDs are smaller than threshold, do \texttt{BDDand} or \texttt{BDDandExists} depending

and go back to step 2.

4. If any BDD is larger than the size threshold, quantity the variable from BDD(s)

3. Pick two smallest BDDs for the variable with the smallest score

2. Score the variables by summing up the sizes of BDDs

1. Quantity away variables appearing in only one conjunct \texttt{Is} are variables appearing in only one conjunct

\[ \phi \neq \texttt{Is} \]

Repeat until

\[ \texttt{Is} \]

Given a set of conjuncts \texttt{Ry} and invisible variables \texttt{Is}.

\[ \texttt{VarScore Based Existential Abstraction} \]
Solve \( \phi \) with a SAT solver. 

\( \phi \subseteq \text{empty} \) is empty. 

\( \phi \) is empty if and only if the counterexample 

\[ \{ s = (s)_{\forall} \mid \begin{array}{c} I = i \wedge (I + s)_{\forall} \vee (I - s)_{\forall} \mid \langle w \cdots \rangle \end{array} = \phi \} = \text{counterexample} \]

Set of concrete paths for counterexample: 

\[ \langle s_1, s_2, \cdots, s_m \rangle \]

Checking the Counterexample
Checking the Counterexample

If \( \phi_m \) is unsatisfiable, refine.

If \( \phi_m \) is satisfiable we found a real bug.

- Also restrict values of (original) inputs that are assigned by counterexample.
- Path restricted to counterexample.

Similar to BMC formulas, except
No concrete transition from $A$ to a concrete state in the next abstract state.

- \( f \phi \in \langle f p \ldots f p \rangle \) in $A$.
- The set of all states such that $f p$ is a concrete path is called the set of deadend states.
- The set $D$ of all states such that $f p$ is satisfiable.
- Find largest index $f$ (failure index) such that $m > f$. 

\[ \text{Refinement} \]
\[
\{1 + f_s = (1 + f_s) \eta \lor f_s = (f_s) \eta \lor (1 + f_s, f_s) \eta \mid \langle 1 + f_s, f_s \rangle \} = f \phi
\]

Transitions from \( \eta \) to \( 1 + f_s \) to \( f_s \) to \( f_s \), there is a non-empty set of abstract transitions from \( f_s \) to \( 1 + f_s \) to \( f_s \). Since there is an abstract transition from \( f_s \) to \( 1 + f_s \), there is a non-empty set of abstract transitions from \( f_s \) to \( 1 + f_s \) to \( f_s \).
\[(q, y) \neq (p, y) \text{ and } B \in \mathbb{D}, \forall p \in \mathbb{D} \]

- Refinement: Put \( B \) and \( D \) is separate abstract states.
- Spurious transition because \( B \) and \( D \) lie in the same abstract state.
- There is a spurious transition from \( f \) to \( f +1 \).
check by Boolean constraint propagation and implication graphs.

SAT solvers record the important reasons for the unsatisfiability during the SAT

For a spurious counterexample, \( \forall m \text{ unsatisfiable.} \)

We solve \( \forall m \) with a SAT solver.

describing the set of paths corresponding to the abstract counterexample

\[
\{ \exists !I = \exists I^+ \exists I^{-1} \exists I^{-2} \ldots \exists I^{-m} \mid (\forall I) I | (\forall m \ldots I^{-1}) I^{-m} \} = \forall m
\]

Recall the formula

Refinement Using SAT Conflict Analysis
We propose two methods to identify important variables by analyzing conflicts generated during the SAT check. These methods do not need potentially expensive sampling, thus incurring minimal additional overhead.

1. Heuristically score the variables during the SAT check.

2. Identify important variables by conflict dependency graphs.
\[ w_1 \cdot \text{backtrack-score} + w_2 \cdot \text{conflict-score} \]

Variables are ranked according to the weighted average of the two scores.

- Variables are ranked according to the weighted average of the two scores.
- Variables are ranked according to the weighted average of the two scores.
- Variables are ranked according to the weighted average of the two scores.
- Variables are ranked according to the weighted average of the two scores.

We count the number of backtracks to a variables (backtrack-score) and the number of times a variable appears in conflict clauses (conflict-score).

For unsatisfiability:

Moreover, the variables appearing in conflict clauses collectively record the reason.

Each backtrack variable records a reason for the unsatisfiability of the formula.
Conflict Graph $B$ directly depends on conflict Graph $A$, since the conflict clause $m$ generated by Graph $A$ is used to imply the conflict $B$.

We propose Conflict Dependency Graphs to prune unnecessary conflicts.

Not every conflict is important for the unsatisfiability of the formula.
conflict graph and all the conflict graphs on which the last one depends. This graph is pruned to get the conflict dependency graph. It includes the last edges of the unpruned dependency graph are direct dependencies.

• Vertices of the unpruned dependency graph are all conflict graphs created by the

Given the set of conflict graphs generated during satisfiability checking, we construct

Conflict Dependency Graphs

Automated Abstraction Refinement by SAT Based Conflict Analyses
Only the variables in the pruned conflict dependency graph are considered for refinement:

includes conflict graphs $A, C, D, E$. Conflict graph $B$ is pruned away. Conflict graph $E$ is the last conflict graph, hence the conflict dependency graph

**Conflict Dependency Graphs**
### Comparison between Cadence SMV (CSMV), heuristic score based refinement and dependence analysis based refinement for larger circuits

<table>
<thead>
<tr>
<th>Circuit</th>
<th># Iers</th>
<th>Time</th>
<th>Heuristic Score</th>
<th>CSMV</th>
<th># Regs</th>
<th>Length</th>
<th>Corex</th>
</tr>
</thead>
<tbody>
<tr>
<td>D2</td>
<td>12</td>
<td>29</td>
<td>3.82</td>
<td>32</td>
<td>1.192</td>
<td>105</td>
<td>15</td>
</tr>
<tr>
<td>D5</td>
<td>35</td>
<td>22</td>
<td>69</td>
<td>4.95</td>
<td>29</td>
<td>105</td>
<td>15</td>
</tr>
<tr>
<td>D6</td>
<td>45</td>
<td>26</td>
<td>12.08</td>
<td>20</td>
<td>32</td>
<td>105</td>
<td>15</td>
</tr>
<tr>
<td>D7</td>
<td>56</td>
<td>14</td>
<td>1.493</td>
<td>28</td>
<td>745</td>
<td>28</td>
<td>14</td>
</tr>
<tr>
<td>D24</td>
<td>10</td>
<td>7</td>
<td>1.947</td>
<td>7</td>
<td>265</td>
<td>7</td>
<td>4</td>
</tr>
<tr>
<td>D50</td>
<td>100</td>
<td>3.6</td>
<td>0.995</td>
<td>28</td>
<td>346</td>
<td>3.6</td>
<td>7</td>
</tr>
<tr>
<td>D70</td>
<td>200</td>
<td>3.6</td>
<td>0.997</td>
<td>28</td>
<td>346</td>
<td>3.6</td>
<td>7</td>
</tr>
<tr>
<td>D80</td>
<td>300</td>
<td>3.6</td>
<td>0.998</td>
<td>28</td>
<td>346</td>
<td>3.6</td>
<td>7</td>
</tr>
<tr>
<td>D90</td>
<td>400</td>
<td>3.6</td>
<td>0.999</td>
<td>28</td>
<td>346</td>
<td>3.6</td>
<td>7</td>
</tr>
<tr>
<td>D100</td>
<td>500</td>
<td>3.6</td>
<td>0.999</td>
<td>28</td>
<td>346</td>
<td>3.6</td>
<td>7</td>
</tr>
</tbody>
</table>

### Experimental Results for Refinement by Conflict Analysis

Apart from the smaller IU circuits [CKS52], these techniques are able to handle large D series circuits and an IU circuit with 5000 latches.
Comparison between separation based, heuristic score based refinement and dependency analysis based refinement for smaller circuits.

| Circuit | # Reqs | Heuristic Score | Dependency | # Reqs | Time | Heuristic Score | Dependency | # Reqs | Time | Heuristic Score | Dependency | # Reqs | Time | Heuristic Score | Dependency | # Reqs | Time | Heuristic Score | Dependency | # Reqs | Time | Heuristic Score | Dependency | # Reqs | Time | Heuristic Score | Dependency | # Reqs | Time | Heuristic Score | Dependency | # Reqs | Time | Heuristic Score | Dependency |
|---------|--------|-----------------|------------|--------|------|-----------------|------------|--------|------|-----------------|------------|--------|------|-----------------|------------|--------|------|-----------------|------------|--------|------|-----------------|------------|--------|------|-----------------|------------|--------|------|-----------------|------------|--------|------|-----------------|------------|--------|------|-----------------|------------|
| ST       | 2      | 6               | 3          | 3      | 85.1| 2               | 6          | 85.1| 2               | 6          | 85.1| 2               | 6          | 85.1| 2               | 6          | 85.1| 2               | 6          | 85.1| 2               | 6          | 85.1| 2               | 6          |
| ST       | 2      | 6               | 3          | 3      | 85.1| 2               | 6          | 85.1| 2               | 6          | 85.1| 2               | 6          | 85.1| 2               | 6          | 85.1| 2               | 6          | 85.1| 2               | 6          | 85.1| 2               | 6          | 85.1| 2               | 6          | 85.1| 2               | 6          |

Comparing separation based refinement and conflict based refinement on smaller LUTs.

Experimental Results for Refinement by Conflict Analysis.
\textbf{SAT based fixed points.}

- Explore SAT based abstraction refinement for (G)SPE and software verification for further improving refinement.

Investigate the use of circuit structure information for improving SAT checking and conflict analysis based refinement is the best technique.

**memory requirements.**

Our algorithms outperform standard model checking in both execution time and scale circuits.

We propose a SAT based abstraction refinement framework to handle industrial.

\begin{center}
\textbf{Conclusions and Future Work}
\end{center}