Bang for the Buck: Improvising and Scheduling Verification Engines for Effective Resource Utilization

HWVV 2010*
July 15, 2010

Malay Ganai and Weihong Li
System Analysis & Verification Group
NEC Labs America
Princeton, U. S. A.

*Originally presented in Memocode’09
Embedded System Verification

• Many properties to be solved in a tight time budget
  – 10-12 min per function modules (NEC Verification services)
• Need good verification procedures
  – Without proofs, falsification engine consumes resources fruitlessly
  – Without falsification, proof engine consumes resources fruitlessly
• **Goal:** How to combine these engines to get "bang for the buck"?
ACE Overview: Bang for the Buck

Augmentation and Combination of Verification Engines
Example

1. void foo(void) {
2.     i=0;
3.     while(1){
4.         if (i==0){
5.             x = ND();
6.             assume (x > 0);
7.         } else if (i>0)
8.             x++;
9.         if (x==10 && i>35)
10.            break;
11.         i++;
12.         assert(i<35); /* P1 */
13.     }
14.     while(1){
15.         i++;
16.         assert(i<40); /* P2 */
17.     }
18. }
C program to CDFG

1. void foo(void) {
2.     i=0;
3.     while(1){
4.         if (i==0){
5.             x = ND();
6.             assume (x > 0);
7.         } else if (i>0)
8.             x++;
9.         if (x==10 && i>35)
10.            break;
11.     i++;
12.     assert(i<35); /* P1 */
13. }
14. while(1){
15.     i++;
16.     assert(i<40); /* P2 */
17. }
18.}
Transition Relation

- Control state variable, $PC$
  - Boolean Predicate, $B_r \equiv (PC = r)$
  - $domain(PC) = \{S_1, \ldots, S_{12}\}$
- Next state functions
  - $next (PC) = B_{S_1} ? S_2 : B_{S_2} && T_{23} ? S_2 : B_{S_2} && T_{24} : S_4 : \ldots : PC$
  - $next (x) = (B_{S_3})? ND() : (B_{S_4}) ? x++ : x$
Forward Control State Reachability

Error blocks: P1, P2

k=0
FR(0) = {1}

k=1
FR(1) = {2}

k=2
FR(2) = {3, 4}

k=3
FR(3) = {5}

k=4
FR(4) = {6}

k=5
FR(5) = {10, 7}

k=6
FR(6) = {11, 8, 9}

k=7
FR(7) = {12, 10, 2}
Transition Relation

- Control state variable, $PC$
  - Boolean Predicate, $B_r \equiv (PC = r)$
  - $domain(PC) = \{S1,\ldots,S12\}$

- Next state functions
  - $next\ (PC) = B_{S1} \ ? \ S2 : B_{S2} \& \& T23 \ ? \ S2 : B_{S2} \& \& T24 : S4 : \ldots : PC$
  - $next\ (x) = (B_{S3})? \ ND() : (B_{S4}) \ ? \ x++ : x$

\[
next(x) = x \text{ if } S3, S4 \not\in FR(k)
\]
BMC simplification using CSR

Ganai et al ICCAD 2006

\[
\begin{align*}
k=0 & : FR(0) = \{1\} \\
k=1 & : FR(1) = \{2\} \\
k=2 & : FR(2) = \{3, 4\} \\
k=3 & : FR(3) = \{5\} \\
k=4 & : FR(4) = \{6\} \\
k=5 & : FR(5) = \{10, 7\} \\
k=6 & : FR(6) = \{11, 8, 9\} \\
k=7 & : FR(7) = \{12, 10, 2\}
\end{align*}
\]
ACE Overview: Bang for Buck

Model, M Error Blocks, E → Model Builder CFG → EFSM → Static Program Analysis → 'C' Program → Checkers

Light-weight Proof Engine (Induction) → Reduction: M → M' → Pruning: E → E' → Light-weight Falsify Engines (BMC, SIM)

ACE

Augmentation and Combination of Verification Engines
SAT-based Induction

- Proof by Induction with increasing depth (G p)

Base Step: If \( \text{Sat}(\neg p_k) \), then property is false

Inductive Step: If \( \text{Unsat}(\neg p_{k+1}) \), then property is true

Else \( k++ \) (until resources are exhausted)
Backward Control State Reachability

Error blocks: P1, P2

- k=7: BR(7) = \{7\}
- k=6: BR(6) = \{9, 1\}
- k=5: BR(5) = \{2\}
- k=4: BR(4) = \{3, 4\}
- k=3: BR(3) = \{5\}
- k=2: BR(2) = \{6\}
- k=1: BR(1) = \{7\}
- k=0: BR(0) = \{8\}
Induction Simplify using CSR

<table>
<thead>
<tr>
<th>k=7</th>
<th>BR(7) = {7}</th>
</tr>
</thead>
<tbody>
<tr>
<td>k=6</td>
<td>BR(6) = {9,1}</td>
</tr>
<tr>
<td>k=5</td>
<td>BR(5) = {2}</td>
</tr>
<tr>
<td>k=4</td>
<td>BR(4) = {3,4}</td>
</tr>
<tr>
<td>k=3</td>
<td>BR(3) = {5}</td>
</tr>
<tr>
<td>k=2</td>
<td>BR(2) = {6}</td>
</tr>
<tr>
<td>k=1</td>
<td>BR(1) = {7}</td>
</tr>
<tr>
<td>k=0</td>
<td>BR(0) = {8}</td>
</tr>
</tbody>
</table>
**Light-weight Proof Engine**

**Goal:** Induction Strengthening using BWD CSR

**IND at depth** $k$, $0 \leq k \leq M$

- **Pick** $e \in E$
- **Control State Reachability**
  - BWD CSR: $BR(k)$, $0 \leq k \leq N$
- **Induction simplified and strengthened using BWD CSR**
  - Update: $E$, $CFG M$

- **Unroll using High level Simplifier**
- **Check:** $SAT(\varphi)$?
  - SMT formula $\varphi$:
    - Reachability of $e$ from $somedc \in BR(k)$
- **Check:** $c_0 \in BR(k)$ & $SAT(\varphi \land I)$?

- **Unroll using High level Simplifier**
- **Check:** $c_0 \in BR(k)$ & $SAT(\varphi \land I)$?

**BR(k):** Set of control states statically reachable at backward depth $k$ from $e \in E$

$c_0$: initial control state (entry block)
ACE Overview: Bang for Buck

Augmentation and Combination of Verification Engines
Lighthouses

“Intermediate state predicates that guide the search”

Eg: Dominators ($D_i$)

Observation: A fewer paths (due to less branching conditions) but longer (due to loops) from $D_i$ to $D_{i+1}$ make simulation engine to reach $D_{i+1}$ more likely.
Lighthouses: Control Dominators
Lighthouses: Error Blocks

“Removing the out edge (8→2) of error block P1, we obtain conditional, inductive proof for P2”

“P1 dominates P2”
Proofs on M and M’

<table>
<thead>
<tr>
<th>Ex (# functions)</th>
<th>#P</th>
<th>M</th>
<th>M’</th>
</tr>
</thead>
<tbody>
<tr>
<td>S1 (14)</td>
<td>166</td>
<td>14</td>
<td>48</td>
</tr>
<tr>
<td>S2 (19)</td>
<td>127</td>
<td>4</td>
<td>80</td>
</tr>
<tr>
<td>S3 (36)</td>
<td>96</td>
<td>16</td>
<td>40</td>
</tr>
<tr>
<td>S4 (8)</td>
<td>82</td>
<td>43</td>
<td>59</td>
</tr>
<tr>
<td>S5 (4)</td>
<td>39</td>
<td>10</td>
<td>12</td>
</tr>
<tr>
<td>S6 (4)</td>
<td>34</td>
<td>4</td>
<td>23</td>
</tr>
<tr>
<td>S7 (4)</td>
<td>4</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td><strong>Total</strong></td>
<td>548</td>
<td>91</td>
<td>262</td>
</tr>
</tbody>
</table>

M’ obtained from M by removing “error-out” edge

Upshot: Error blocks are good candidates for lighthouses

Intel Xeon 2.8Ghz, 4Gb RAM 1 min per functions
Proof Engine used LPE(smt)
CSR Refinement (dynamic)

k=0  FR(0) = {1}

k=1  FR(1) = {2}

k=2  FR(2) = {3, 4}

k=3  FR(3) = {5}

k=4  FR(4) = {6}

k=5  FR(5) = {7}

k=6  FR(6) = {8, 9}

k=7  FR(7) = {2}

Pruning by CSR Refinement
ACE: LPE, BMC, Guided Simulation

**Pre-processing**

1. **Model Transformation**
2. **Lighthouse Discovery**
   - Set of Lighthouses, \( L \)
3. **Control State Reachability**
   - FWD CSR: \( FR(k) \) \( 0 \leq k \leq N \)

**BMC at depth \( k \), \( 0 \leq k \leq N \)**

4. \( k = k + 1 \);
   - Generate Simplification Constraints @\( k \)
40

5. \( \text{Unroll using High level Simplifier} \)
41

6. \( \text{Check: SAT(} \varphi \text{) ?} \)
7. SMT formula \( \varphi \): Reachability of \( b \) from \( c_0 \)
8. \( b \in E \)
45

9. **Store witness state, \( w \)\n10. \( W = W \cup \{w\}; \)
11. Update \( E, L \)
46

12. **CSR Refinement**
13. **Falsify Engines**
47

**Simulate**

14. Pick a seed state \( w \in W \)
15. Simulate and check reachability of \( E, L \)
16. Update \( E, L \)

\( c_0 \): Initial control state (entry block)

\( FR(k) \): Set of control states statically reachable at forward depth \( k \) from \( c_0 \)
BMC + Guided Sim

\[ b1: x > 23 \]
\[ b2: y < 3 \]
\[ b3: y \geq 5 \]
\[ b4: z \neq 0 \]

- **Initial block**
- **Error blocks \((e_i)\)**
- **Lighthouses \((l_i)\)**
- **Simulation traces**
- **BMC traces**
Experiments (1/2)

• 7 Benchmarks C programs (1K-3k LoC)
  – information management system utilities, ftp utilities, network applications, embedded apps.
  – array bound violations, pointer validity, ...
  – # functions: 4 to 36
    • each associated with multiple properties
Experiments (2/2)

• Model Checkers (for comparison)
  – SAT-based Induction (Sheeran et al. FMCAD 2000)
  – SAT-based UMC (Ganai et al. ICCAD 2004)
  – MIX [Composite Model Checking] (Yang, MEMOCODE 2006)
  – SMT-based BMC (Ganai et al. ICCAD 2006)
  – MIX-SA [MIX combined with widening] (Wang, CAV 2007)

• Solvers
  – Hybrid SAT Solver (Ganai et al. DAC 2002)
  – Yices SMT solver (Dutertre et al. CAV 2006)
  – Omega [Presburger Solver] (Pugh et al. AC 1991)
Results: SAT-based Proof Engines

<table>
<thead>
<tr>
<th>Ex (# functions)</th>
<th>#P</th>
<th>UMC</th>
<th>IND</th>
<th>IND (No LFP)</th>
<th>LPE</th>
</tr>
</thead>
<tbody>
<tr>
<td>S1 (14)</td>
<td>166</td>
<td>8</td>
<td>13</td>
<td>13</td>
<td>14</td>
</tr>
<tr>
<td>S2 (19)</td>
<td>127</td>
<td>2</td>
<td>2</td>
<td>2</td>
<td>4</td>
</tr>
<tr>
<td>S3 (36)</td>
<td>96</td>
<td>4</td>
<td>4</td>
<td>7</td>
<td>15</td>
</tr>
<tr>
<td>S4 (8)</td>
<td>82</td>
<td>26</td>
<td>28</td>
<td>31</td>
<td>41</td>
</tr>
<tr>
<td>S5 (4)</td>
<td>39</td>
<td>5</td>
<td>4</td>
<td>6</td>
<td>10</td>
</tr>
<tr>
<td>S6 (4)</td>
<td>34</td>
<td>4</td>
<td>4</td>
<td>4</td>
<td>4</td>
</tr>
<tr>
<td>S7 (4)</td>
<td>4</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td><strong>Total</strong></td>
<td><strong>548</strong></td>
<td><strong>49</strong></td>
<td><strong>55</strong></td>
<td><strong>63</strong></td>
<td><strong>88</strong></td>
</tr>
</tbody>
</table>

Upshot: LPE proves more than any other engine

Intel Xeon 2.8Ghz, 4Gb RAM 1 min per function
All uses Hybrid SAT Solver (Ganai et al. DAC 2002)
## Results: Proof Engines

<table>
<thead>
<tr>
<th>Ex (# functions)</th>
<th>#P</th>
<th>LPE (SMT)</th>
<th>MIX-SA (Presb.)</th>
</tr>
</thead>
<tbody>
<tr>
<td>S1 (14)</td>
<td>166</td>
<td>14</td>
<td>0</td>
</tr>
<tr>
<td>S2 (19)</td>
<td>127</td>
<td>4</td>
<td>6</td>
</tr>
<tr>
<td>S3 (36)</td>
<td>96</td>
<td>16</td>
<td>12</td>
</tr>
<tr>
<td>S4 (8)</td>
<td>82</td>
<td>43</td>
<td>11</td>
</tr>
<tr>
<td>S5 (4)</td>
<td>39</td>
<td>10</td>
<td>0</td>
</tr>
<tr>
<td>S6 (4)</td>
<td>34</td>
<td>4</td>
<td>4</td>
</tr>
<tr>
<td>S7 (4)</td>
<td>4</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td><strong>Total</strong></td>
<td>548</td>
<td>91</td>
<td>33</td>
</tr>
<tr>
<td><strong>Exclusive</strong></td>
<td></td>
<td>62</td>
<td>4</td>
</tr>
</tbody>
</table>

**Upshot: LPE proves more than MIX**

Intel Xeon 2.8Ghz, 4Gb RAM 1 min per function
SMT solver: yices,
Presburger Solver: Omega library
## Falsify Engines Configurations

<table>
<thead>
<tr>
<th>Feature Code</th>
<th>Description</th>
<th>FE_ALL</th>
<th>FE_NoLH</th>
<th>FE_NoSim</th>
<th>FE</th>
</tr>
</thead>
<tbody>
<tr>
<td>a</td>
<td>Static context-sensitive fwd CSR Model Transformation CSR learning and simplification</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>X</td>
</tr>
<tr>
<td>b</td>
<td>Lighthouse</td>
<td>X</td>
<td></td>
<td>X</td>
<td></td>
</tr>
<tr>
<td>c</td>
<td>CSR Refinement</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td></td>
</tr>
<tr>
<td>e</td>
<td>Guided-Simulation</td>
<td>X</td>
<td>X</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
### Results: Comparing ACE and MIX

<table>
<thead>
<tr>
<th>Ex (#func)</th>
<th>#P</th>
<th>ACE (ALL)</th>
<th>ACE(NoLH)</th>
<th>ACE(NoSim)</th>
<th>ACE()</th>
<th>MIX</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
<td>#P</td>
<td>#W</td>
<td>#P</td>
<td>#W</td>
<td>#P</td>
</tr>
<tr>
<td>S1 (14)</td>
<td>166</td>
<td>14</td>
<td>122</td>
<td>14</td>
<td>103</td>
<td>14</td>
</tr>
<tr>
<td>S2 (19)</td>
<td>127</td>
<td>4</td>
<td>33</td>
<td>4</td>
<td>38</td>
<td>4</td>
</tr>
<tr>
<td>S3 (36)</td>
<td>96</td>
<td>16</td>
<td>77</td>
<td>16</td>
<td>71</td>
<td>16</td>
</tr>
<tr>
<td>S4 (8)</td>
<td>82</td>
<td>43</td>
<td>16</td>
<td>43</td>
<td>16</td>
<td>43</td>
</tr>
<tr>
<td>S5 (4)</td>
<td>39</td>
<td>10</td>
<td>17</td>
<td>10</td>
<td>2</td>
<td>10</td>
</tr>
<tr>
<td>S6 (4)</td>
<td>34</td>
<td>4</td>
<td>30</td>
<td>4</td>
<td>30</td>
<td>4</td>
</tr>
<tr>
<td>S7 (4)</td>
<td>4</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>Total</td>
<td>548</td>
<td>91</td>
<td>295</td>
<td>91</td>
<td>260</td>
<td>91</td>
</tr>
<tr>
<td>Excl.</td>
<td></td>
<td>45</td>
<td>47</td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

**Upshot:** ACE(all) solves more than MIX

Intel Xeon 2.8Ghz, 4Gb RAM 10 min per function
SMT solver: yices, Presburger Solver: Omega library
Summary/Conclusions

• Presented an improved verification flow ACE combining
  – light weight proof engines
  – bmc engines
  – guided simulation using lighthouses

• Geared towards checking many properties in a tight time budget

• Integrated in industry verification environment
  – provided control experimentation

• In progress: distribute ACE on network of workstations
Thank you!