Software Model Checking via Large-Block Encoding

Dirk Beyer\textsuperscript{1}, Alessandro Cimatti\textsuperscript{2}, Alberto Griggio\textsuperscript{3}, Erkan Keremoglu\textsuperscript{1}, Roberto Sebastiani\textsuperscript{3}

\textsuperscript{1}SFU, Vancouver, Canada, \textsuperscript{2}FBK-IRST and \textsuperscript{3}DISI-Univ. of Trento, Italy
Motivations

♦ SMT: very promising technology for verification
  ♦ SMT solvers: efficient, powerful, scalable
  ♦ Several SMT-based verification techniques recently proposed

♦ Software Model Checking: effective technique for software verification (e.g. SLAM, BLAST, verification of device drivers)
  ♦ Popular approach: lazy abstraction with analysis of an abstract reachability tree (ART)

♦ Current ART-based approaches do not take full advantage of SMT solvers
  ♦ Explicit exploration of the ART, SMT only used (mostly) for conjunctions of constraints
Contribution

- **Large-Block Encoding**: (simple) generalization of traditional ART-based approach aimed at better exploiting SMT technology

- **Less explicit search** on the ART, **more symbolic search** within the SMT solver

- **Empirical evidence** of the benefits on a set of standard benchmark C programs
Outline

♦ Background

♦ Large-Block Encoding

♦ Experimental evaluation
A CFA is a pair \((L, G)\), where:

- \(L\): set of program locations
- \(G\): set of edges \(L \times \text{Op} \times L\)
- \(l_0\) entry point of a program,
- \(l_E\) error location
Programs represented as control-flow automata (CFAs)

- A CFA is a pair \((L, G)\), where:
  - \(L\): set of program locations
  - \(G\): set of edges \(L \times Op \times L\)
  - \(l_0\) entry point of a program,
  - \(l_E\) error location

Example:

```python
while (1) {
    if (y > 0) x = 1;
    else x = 0;
    if (x == 0) goto ERROR;
}
```
Background – semantics

♦ Concrete state of a program: \((l, s)\), where
  ♦ \(l\) is a location
  ♦ \(s\) is an assignment to program variables (a formula \(\bigwedge_i x_i = v\))

♦ Concrete semantics of an operation \(op\) given by \(\text{SP}_{\text{op}}\):
  ♦ Assign: \(\text{SP}_{x:=e}(\varphi) = \exists \hat{x} : \varphi[x\mapsto\hat{x}] \land (x = e[x\mapsto\hat{x}])\)
  ♦ Assume: \(\text{SP}_{p}(\varphi) = \varphi \land p\)

♦ Path: seq. \(\sigma = \langle (op_1, l_1)\ldots(op_n, l_n) \rangle\) where \((l_{i\rightarrow}, op_i, l_i)\) is a CFA edge
  ♦ Semantics \(\text{SP}_{\sigma}(\varphi) = \text{SP}_{op_n}(\ldots \text{SP}_{op_1}(\varphi) \ldots)\)
  ♦ Feasible if \(\text{SP}_{\sigma}(\text{true})\) is satisfiable

♦ Location \(l\) reachable iff exists feasible path \(\langle (op_1, l_1)\ldots(op_n, l) \rangle\)

♦ Program safe iff \(l_E\) not reachable
Background – ART-based SW MC

♦ Abstraction of $\varphi : \alpha(\varphi)$ such that $\varphi \models \alpha(\varphi)$

♦ Abstract SP: $\text{SP}^\alpha_{op}(\varphi) = \alpha(\text{SP}_{op}(\varphi))$

♦ Abstract Reachability Tree (ART): unwinding of the CFA in an abstract space

♦ Each node is an abstract state $(l, \varphi)$

♦ children of $(l, \varphi)$: abstract successors: $\{(l_i, \hat{\varphi}_i)\}_i$

♦ $(l, op_i, l_i)$ is an edge in the CFA

♦ $\hat{\varphi}_i = \text{SP}^\alpha_{op_i}(\varphi)$ and $\hat{\varphi}_i \models \bot$

♦ $(l, \varphi)$ has children only if not covered

♦ there is no $(l, \psi)$ in the ART s.t. $\varphi \models \psi$

♦ ART safe if there is no $(l, \cdot)$
On-the-fly ART construction with counterexample-guided abstraction refinement (CEGAR)

♦ Pick node
On-the-fly ART construction with counterexample-guided abstraction refinement (CEGAR)

- Pick node
- Compute abstract successors
On-the-fly ART construction with counterexample-guided abstraction refinement (CEGAR)

- Pick node
- Compute abstract successors
On-the-fly ART construction with counterexample-guided abstraction refinement (CEGAR)

- Pick node
- Compute abstract successors
- If error reached: analyze abstract trace
  - If spurious:
On-the-fly ART construction with counterexample-guided abstraction refinement (CEGAR)

- Pick node
- Compute abstract successors
- If error reached: analyze abstract trace
  - If spurious:
    - refine abstraction
On-the-fly ART construction with counterexample-guided abstraction refinement (CEGAR)

- Pick node
- Compute abstract successors
- If error reached: analyze abstract trace
  - If spurious:
    - refine abstraction
    - undo part of the ART
On-the-fly ART construction with counterexample-guided abstraction refinement (CEGAR)

- Pick node
- Compute abstract successors
- If error reached: analyze abstract trace
  - If spurious:
    - refine abstraction
    - undo part of the ART
    - Rebuild subtree
On-the-fly ART construction with counterexample-guided abstraction refinement (CEGAR)

- Pick node
- Compute abstract successors
- If error reached: analyze abstract trace
  - If spurious:
    - refine abstraction
    - undo part of the ART
    - Rebuild subtree
On-the-fly ART construction with counterexample-guided abstraction refinement (CEGAR)

- Pick node
- Compute abstract successors
- If error reached: analyze abstract trace
  - If spurious:
    - refine abstraction
    - undo part of the ART
    - Rebuild subtree
On-the-fly ART construction with counterexample-guided abstraction refinement (CEGAR)

- Pick node
- Compute abstract successors
- If error reached: analyze abstract trace
  - If spurious:
    - Refine abstraction
    - Undo part of the ART
    - Rebuild subtree
On-the-fly ART construction with counterexample-guided abstraction refinement (CEGAR)

- Pick node
- Compute abstract successors
- If error reached: analyze abstract trace
  - If spurious:
    - refine abstraction
    - undo part of the ART
    - Rebuild subtree
- ART safe $\Rightarrow$ Program safe
Background – Predicate abstraction

- Abstraction of $\varphi$ as a **Boolean combination** of a set of predicates $P$
- **Boolean abstraction**
  - $\alpha^B_P(\varphi)$ strongest Boolean combination of $P$ s.t. $\varphi \models \alpha^B_P(\varphi)$
  - Expensive to compute
    - Traditional approach: $2^{|P|}$ calls to a decision procedure
- **Cartesian abstraction**
  - $\alpha^C_P(\varphi) = \bigwedge\{p \in P \mid \varphi \models p\} \cup \bigwedge\{\neg p \mid p \in P \text{ and } \varphi \models \neg p\}$
  - Much cheaper to compute
  - Much weaker
- **Abstraction refinement**: add more predicates to $P$
Example

\[ \varphi = ((y > 0) \land (x = 1)) \lor (\neg(y > 0) \land (x = 0)) \]

\[ P = \{(x = 0), (y = 0)\} \]
Background – Cartesian vs Boolean abst.

♦ Example

\[ \varphi = ((y > 0) \land (x = 1)) \lor (\neg(y > 0) \land (x = 0)) \]

\[ P = \{(x = 0), (y = 0)\} \]

♦ \[ \alpha_P^C(\varphi) = \top \], since

\[ \varphi \not\models (x = 0) \quad \varphi \not\models \neg(x = 0) \]

\[ \varphi \not\models (y = 0) \quad \varphi \not\models \neg(y = 0) \]
Background – Cartesian vs Boolean abst.

♦ Example

\[ \varphi = ((y > 0) \land (x = 1)) \lor (\neg(y > 0) \land (x = 0)) \]

\[ P = \{(x = 0), (y = 0)\} \]

♦ \( \alpha^C_P(\varphi) = \top \), since

\[ \varphi \not\models (x = 0) \quad \varphi \not\models \neg(x = 0) \]

\[ \varphi \not\models (y = 0) \quad \varphi \not\models \neg(y = 0) \]

♦ \( \alpha^B_P(\varphi) = (y = 0) \rightarrow (x = 0) \)

\[ if\ \ [y > 0] \quad then\ \ x := 1 \]

\[ if\ \ [!(y > 0)] \quad else\ \ x := 0 \]

\[ L \]
Background – Cartesian vs Boolean abst.

♦ Example

\[ \varphi = ((y > 0) \land (x = 1)) \lor (\neg(y > 0) \land (x = 0)) \]

\[ P = \{(x = 0), (y = 0)\} \]

♦ \( \alpha^C_P(\varphi) = \top \), since

\[ \varphi \not\models (x = 0) \quad \varphi \not\models \neg(x = 0) \]

\[ \varphi \not\models (y = 0) \quad \varphi \not\models \neg(y = 0) \]

♦ \( \alpha^B_P(\varphi) = (y = 0) \rightarrow (x = 0) \)

♦ However, e.g. for \( \varphi_{\text{then}} = (y > 0) \land (x = 1) \)

\[ \alpha^C_P(\varphi_{\text{then}}) = \alpha^B_P(\varphi_{\text{then}}) = \neg(x = 0) \land \neg(y = 0) \]
Background – Cartesian vs Boolean abst.

- **Example**

\[ \varphi = ((y > 0) \land (x = 1)) \lor (\neg(y > 0) \land (x = 0)) \]

\[ P = \{(x = 0), (y = 0)\} \]

- \( \alpha^C_P(\varphi) = \top \), since

\[ \varphi \not\models (x = 0) \quad \varphi \not\models \neg(x = 0) \]

\[ \varphi \not\models (y = 0) \quad \varphi \not\models \neg(y = 0) \]

- \( \alpha^B_P(\varphi) = (y = 0) \rightarrow (x = 0) \)

- However, e.g. for \( \varphi_{\text{then}} = (y > 0) \land (x = 1) \)

\[ \alpha^C_P(\varphi_{\text{then}}) = \alpha^B_P(\varphi_{\text{then}}) = \neg(x = 0) \land \neg(y = 0) \]

Cart. abst. often good enough for ART-based SW MC in practice
Outline

♦ Background

♦ Large-Block Encoding

♦ Experimental evaluation
ART and SMT solvers

- Using an ART, reduced cost in computing abstractions
  - Separate $SP^\alpha_{op}$ computation for each edge (*single block*)
  - Cartesian abstraction works well in practice
  - *Consequence:* very simple queries to the SMT solver

- However, up to exponential number of paths to explore *explicitly* in the ART
  - Exponentially-many trivial SMT solver calls

*Power, scalability and features of modern SMT solvers not fully exploited*
Example of exponential ART

if (p1) {x1 = 1;} else {x1 = 0;}
...
if (pn) {xn = 1;} else {xn = 0;}

if (p1) { if (!x1) goto ERROR; }
...
if (pn) { if (!xn) goto ERROR; }
Example of exponential ART

```c
if (p1) {x1 = 1;} else {x1 = 0;}
...
if (pn) {xn = 1;} else {xn = 0;}
if (p1) { if (!x1) goto ERROR; }
...
if (pn) { if (!xn) goto ERROR; }
```

CFA:
Example of exponential ART

if (p1) {x1 = 1;} else {x1 = 0;}
...
if (pn) {xn = 1;} else {xn = 0;}
if (p1) { if (!x1) goto ERROR; }
...
if (pn) { if (!xn) goto ERROR; }

Preds needed: \{(x_1=0),..., (x_n=0), p_1,...,p_n\}
Example of exponential ART

if (p1) {x1 = 1;} else {x1 = 0;}
...
if (pn) {xn = 1;} else {xn = 0;}
if (p1) { if (!x1) goto ERROR; }
...
if (pn) { if (!xn) goto ERROR; }

Preds needed: \{(x_1=0),..., (x_n=0), p_1,...,p_n\}

ART for \(n=5\)

CFA:
Large-Block Encoding

- **Main idea: use a “more coarse-grained” ART**
  - No more 1:1 mapping between ART paths and program paths
  - Each ART edge encodes a loop-free subpart of the program
  - Each ART path encodes a set of program paths

- **Consequences:**
  - reduce size of the ART (up to exponentially)
  - Increase cost of $SP^\alpha_{op}$, path feasibility checks, refinement
Large-Block Encoding

- **Main idea: use a "more coarse-grained" ART**
  - No more 1:1 mapping between ART paths and program paths
  - Each ART edge encodes a loop-free subpart of the program
  - Each ART path encodes a set of program paths

- **Consequences:**
  - Reduce size of the ART (up to exponentially)
  - Increase cost of $\text{SP}_o^\alpha$, path feasibility checks, refinement

Less explicit search, more (symbolic) work for the SMT solver
Large-Block Encoding: implementation

♦ *Do not modify the algorithm, modify the CFA*

♦ *Summarization* of the CFA
  ♦ Fixpoint application of 2 summarization rules
Large-Block Encoding: implementation

- Do not modify the algorithm, modify the CFA
- Summarization of the CFA
  - Fixpoint application of 2 summarization rules

Rule 1 (Sequence)
- Conditions:
  - \( l_1 \neq l_2 \)
  - No other incoming edges to \( l_2 \)
Large-Block Encoding: implementation

- Do not modify the algorithm, modify the CFA
- Summarization of the CFA
  - Fixpoint application of 2 summarization rules

Rule 1 (Sequence)
- Conditions:
  - \( l_1 \neq l_2 \)
  - No other incoming edges to \( l_2 \)
Large-Block Encoding: implementation

- Do not modify the algorithm, modify the CFA
- Summarization of the CFA
  - Fixpoint application of 2 summarization rules

Rule 1 (Sequence)
- Conditions:
  - $l_1 \neq l_2$
  - No other incoming edges to $l_2$

$$\text{SP}_{op_1; op_2}(\varphi) = \text{SP}_{op_2}(\text{SP}_{op_1}(\varphi))$$
Large-Block Encoding: implementation

- Do not modify the algorithm, modify the CFA
- Summarization of the CFA
  - Fixpoint application of 2 summarization rules

Rule 2 (Choice)
Large-Block Encoding: implementation

- Do not modify the algorithm, modify the CFA
- Summarization of the CFA
  - Fixpoint application of 2 summarization rules

Rule 2 (Choice)

\[
SP_{op_1||op_2}(\varphi) = SP_{op_1}(\varphi) \lor SP_{op_2}(\varphi)
\]
CFA summarization – Example

L1: while (i > 0) {
L2:   if (x == 1) {
L3:     z = 0;
L4:   } else {
L5:     z = 1;
L6:   }  
L7:   i = i - 1;
L8: 
L9: ...
CFA summarization – Example

L1: while \( i > 0 \) {
L2:   if \( x == 1 \) {
L3:     z = 0;
L4:   } else {
L5:     z = 1;
L6:   }
L7:   i = i - 1;
L7: ...

CFA summarization – Example

L1: while (i > 0) {
L2:   if (x == 1) {
L3:     z = 0;
L4:   } else {
L5:     z = 1;
L6:   }
L7:   i = i – 1;
L7: ...
CFA summarization – Example

L1: while (i > 0) {
L2:   if (x == 1) {
L3:     z = 0;
L4:   } else {
L5:     z = 1;
L6:   }
L7:   i = i – 1;
L7: ...

Diagram:
- L1: while
- L2: if
- L3: then
  - z = 0
- L4: else
  - z = 1
- L5: i = i - 1
- L7: ...

States:
- [i>0]
- [!(i>0)]
- [x=1]
- [!(x=1)]

Notations:
- L1:while
- L2:if
- L3:then
- L4:else
- L5
CFA summarization – Example

L1: while (i > 0) {
L2:   if (x == 1) {
L3:     z = 0;
L4:   } else {
L5:     z = 1;
L6:   }
L7:   i = i - 1;
L7: ...

CFA summarization – Example

L1: while (i > 0) {
L2:   if (x == 1) {
L3:     z = 0;
L4:   } else {
L5:     z = 1;
L6:   }
L7:   i = i - 1;
L7: ...
CFA summarization – Example

L1: while (i > 0) {
L2:   if (x == 1) {
L3:     z = 0;
L4:   } else {
L5:     z = 1;
L6:   }
L7:   i = i - 1;
L8: }
L9: ...

[i>0] [!(i>0)]
[x==1];z = 0 [!(x==1)];z = 1
i = i-1
CFA summarization – Example

L1: while (i > 0) {
L2:   if (x == 1) {
L3:     z = 0;
L4:   } else {
L5:     z = 1;
L6:   }
L7:   i = i - 1;
L7:   ...
}
CFA summarization – Example

\[
L1: \text{while } (i > 0) \ { \\
L2: \quad \text{if } (x == 1) \ { \\
L3: \quad \quad z = 0; \\
L4: \quad \quad } \text{ else } \ { \\
L5: \quad \quad \quad z = 1; \\
L6: \quad \quad } \} \\
L7: \ \ldots
\]

\[
([x==1]; z = 0) \lor ([!(x==1)]; z = 1)
\]

\[
i = i - 1
\]
CFA summarization – Example

L1: while (i > 0) {
L2:   if (x == 1) {
L3:     z = 0;
L4:   } else {
L5:     z = 1;
L6:   }
L7:   i = i – 1;
L7: }

([x==1]; z = 0) || ([!x==1]; z = 1)

i = i-1
CFA summarization – Example

L1: while (i > 0) {
L2:   if (x == 1) {
L3:     z = 0;
L4:   } else {
L5:     z = 1;
L6:   }
L7:   i = i - 1;
L7: }

(((x==1); z = 0) || (!((x==1)]; z = 1));
(i = i-1)
CFA summarization – Example

L1: while (i > 0) {
L2:   if (x == 1) {
L3:     z = 0;
L4:   } else {
L5:     z = 1;
L6:   }
L7:   i = i – 1;
L7:   ...

((x==1); z = 0) || (! (x==1); z = 1));
(i = i-1)
CFA summarization – Example

L1: while (i > 0) {
L2:   if (x == 1) {
L3:     z = 0;
L4:   } else {
L5:     z = 1;
L6:   }
L7:   i = i - 1;
L8: }
L9: ...

([i>0]);
(([x==1]; z = 0) | | ([!(x==1)]; z = 1));
(i = i-1)
Large blocks and predicate abstraction

- With Large-Block Encoding:
  - Each edge represents a whole loop-free subpart of a program
  - ART paths represent sets of program paths
- Consequence: Cartesian abstraction can't be used anymore
  - Too weak
  - Need to keep track of dependencies among predicates
- In this work: experiment with Boolean abstraction
  - Exponential in the worst case, but can exploit recent work in computation of Boolean abstraction with SMT ([Lahiri et al CAV06], [Cavada et al FMCAD07])
    - Huge progress wrt. previous techniques
  - Cost of the analysis moved from explicit path enumeration to symbolic $SP^\alpha$ computation
One error trace in the ART corresponds to a set of concrete traces

Complexity increased also for counterexample analysis and abstraction-refinement

With interpolation-based refinement [Henzinger et al POPL04], exploit improvements in Craig interpolant computation with SMT ([Cimatti et al TACAS08, CADE09], [Fuchs et al TACAS09], [Goel et al CADE09])
Outline

♦ Background

♦ Large-Block Encoding

♦ Experimental evaluation
Experimental evaluation

- Implementation within CPAchecker
  - http://www.cs.sfu.ca/~dbeyer/cpachecker
  - Not only Large-Block Encoding (LBE), but also traditional “Single-Block” (SBE)
- Use MathSAT as SMT backend
  - Boolean abstraction via All-SMT, efficient interpolation for refinement

- Comparison of CPAchecker-LBE vs
  - CPAchecker-SBE
    - Same language, same infrastructure, ...
  - BLAST
    - State-of-the-art ART-based SW MC
    - 4 different configurations tested
Benchmark instances

- **test_locks**
  - Artificial instances that show exponential blowup of ART with SBE

- **ntdrivers**
  - Windows NT device drivers
  - “standard” instances used in previous work
  - Taken from BLAST distribution

- **SSH**
  - Check properties of SSH client and server protocols
  - Taken from BLAST distribution, used in previous work

- For ntdrivers and SSH, test also programs with artificial bugs
## Results - test_locks

<table>
<thead>
<tr>
<th>Program</th>
<th>Blast Best result</th>
<th>Blast -dfs -predH 7</th>
<th>CPAchecker SBE</th>
<th>CPAchecker LBE</th>
</tr>
</thead>
<tbody>
<tr>
<td>test_locks5.c</td>
<td>4.50</td>
<td>4.96</td>
<td>4.01</td>
<td>0.29</td>
</tr>
<tr>
<td>test_locks6.c</td>
<td>7.81</td>
<td>8.81</td>
<td>7.22</td>
<td>0.34</td>
</tr>
<tr>
<td>test_locks7.c</td>
<td>13.91</td>
<td>15.15</td>
<td>12.63</td>
<td>0.34</td>
</tr>
<tr>
<td>test_locks8.c</td>
<td>25.00</td>
<td>26.49</td>
<td>23.93</td>
<td>0.57</td>
</tr>
<tr>
<td>test_locks9.c</td>
<td>46.84</td>
<td>49.29</td>
<td>52.04</td>
<td>0.38</td>
</tr>
<tr>
<td>test_locks10.c</td>
<td>94.57</td>
<td>97.85</td>
<td>131.39</td>
<td>0.40</td>
</tr>
<tr>
<td>test_locks11.c</td>
<td>204.55</td>
<td>208.78</td>
<td>MO</td>
<td>0.70</td>
</tr>
<tr>
<td>test_locks12.c</td>
<td>529.16</td>
<td>533.97</td>
<td>MO</td>
<td>0.46</td>
</tr>
<tr>
<td>test_locks13.c</td>
<td>1229.27</td>
<td>1232.87</td>
<td>MO</td>
<td>0.49</td>
</tr>
<tr>
<td>test_locks14.c</td>
<td>&gt;1800.00</td>
<td>&gt;1800.00</td>
<td>MO</td>
<td>0.50</td>
</tr>
<tr>
<td>test_locks15.c</td>
<td>&gt;1800.00</td>
<td>&gt;1800.00</td>
<td>MO</td>
<td>0.29</td>
</tr>
<tr>
<td><strong>TOTAL</strong></td>
<td><strong>9 / 2155.61</strong></td>
<td><strong>9 / 2178.17</strong></td>
<td><strong>6 / 231.22</strong></td>
<td><strong>11 / 4.76</strong></td>
</tr>
</tbody>
</table>
## Results - test_locks

<table>
<thead>
<tr>
<th>Program</th>
<th>Best result</th>
<th>Blast</th>
<th>CPAchecker</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
<td>-dfs -predH 7</td>
<td>SBE</td>
</tr>
<tr>
<td>test_locks5.c</td>
<td>4.50</td>
<td>4.96</td>
<td>4.01</td>
</tr>
<tr>
<td>test_locks6.c</td>
<td>7.81</td>
<td>8.81</td>
<td>7.22</td>
</tr>
<tr>
<td>test_locks7.c</td>
<td>13.91</td>
<td>15.15</td>
<td>12.63</td>
</tr>
<tr>
<td>test_locks8.c</td>
<td>25.00</td>
<td>26.49</td>
<td>23.93</td>
</tr>
<tr>
<td>test_locks9.c</td>
<td>19.29</td>
<td>52.04</td>
<td>0.38</td>
</tr>
<tr>
<td>test_locks10.c</td>
<td>529.16</td>
<td>533.97</td>
<td>MO</td>
</tr>
<tr>
<td>test_locks11.c</td>
<td>1229.27</td>
<td>1232.87</td>
<td>MO</td>
</tr>
<tr>
<td>test_locks12.c</td>
<td>&gt;1800.00</td>
<td>&gt;1800.00</td>
<td>MO</td>
</tr>
<tr>
<td>test_locks13.c</td>
<td>&gt;1800.00</td>
<td>&gt;1800.00</td>
<td>MO</td>
</tr>
<tr>
<td>TOTAL</td>
<td>9 / 2155.61</td>
<td>9 / 2178.17</td>
<td>6 / 231.22</td>
</tr>
</tbody>
</table>

2 more instances, >500x faster
## Results – ntdrivers and SSH – safe

<table>
<thead>
<tr>
<th>Program</th>
<th>Best result</th>
<th>Blast -dfs -predH 7</th>
<th>CPAchecker</th>
<th>SBE</th>
<th>LBE</th>
</tr>
</thead>
<tbody>
<tr>
<td>cdaudio</td>
<td>175.76</td>
<td>264.12</td>
<td>MO</td>
<td>53.55</td>
<td></td>
</tr>
<tr>
<td>diskperf</td>
<td>&gt;1800.00</td>
<td>&gt;1800.00</td>
<td>MO</td>
<td>232.00</td>
<td></td>
</tr>
<tr>
<td>floppy</td>
<td>218.26</td>
<td>&gt;1800.00</td>
<td>MO</td>
<td>56.36</td>
<td></td>
</tr>
<tr>
<td>kbfiltr</td>
<td>23.55</td>
<td>32.80</td>
<td>41.12</td>
<td>7.82</td>
<td></td>
</tr>
<tr>
<td>parport</td>
<td>738.82</td>
<td>915.79</td>
<td>MO</td>
<td>378.04</td>
<td></td>
</tr>
<tr>
<td>s3_clnt.01</td>
<td>33.01</td>
<td>1000.41</td>
<td>755.81</td>
<td>19.51</td>
<td></td>
</tr>
<tr>
<td>s3_clnt.02</td>
<td>62.65</td>
<td>312.77</td>
<td>1075.45</td>
<td>16.00</td>
<td></td>
</tr>
<tr>
<td>s3_clnt.03</td>
<td>60.62</td>
<td>314.74</td>
<td>746.31</td>
<td>49.50</td>
<td></td>
</tr>
<tr>
<td>s3_clnt.04</td>
<td>63.96</td>
<td>197.65</td>
<td>730.80</td>
<td>25.45</td>
<td></td>
</tr>
<tr>
<td>s3srvr.01</td>
<td>811.27</td>
<td>1036.89</td>
<td>&gt;1800.00</td>
<td>125.33</td>
<td></td>
</tr>
<tr>
<td>s3srvr.02</td>
<td>360.47</td>
<td>360.47</td>
<td>&gt;1800.00</td>
<td>122.83</td>
<td></td>
</tr>
<tr>
<td>s3srvr.03</td>
<td>276.19</td>
<td>276.19</td>
<td>&gt;1800.00</td>
<td>98.47</td>
<td></td>
</tr>
<tr>
<td>s3srvr.04</td>
<td>175.64</td>
<td>301.85</td>
<td>&gt;1800.00</td>
<td>71.77</td>
<td></td>
</tr>
<tr>
<td>s3srvr.06</td>
<td>304.63</td>
<td>304.63</td>
<td>&gt;1800.00</td>
<td>59.70</td>
<td></td>
</tr>
<tr>
<td>s3srvr.07</td>
<td>478.05</td>
<td>666.53</td>
<td>&gt;1800.00</td>
<td>85.82</td>
<td></td>
</tr>
<tr>
<td>s3srvr.08</td>
<td>115.76</td>
<td>115.76</td>
<td>&gt;1800.00</td>
<td>61.29</td>
<td></td>
</tr>
<tr>
<td>s3srvr.09</td>
<td>445.21</td>
<td>1037.09</td>
<td>&gt;1800.00</td>
<td>126.47</td>
<td></td>
</tr>
<tr>
<td>s3srvr.10</td>
<td>115.10</td>
<td>115.10</td>
<td>&gt;1800.00</td>
<td>63.36</td>
<td></td>
</tr>
<tr>
<td>s3srvr.11</td>
<td>367.98</td>
<td>844.28</td>
<td>&gt;1800.00</td>
<td>162.76</td>
<td></td>
</tr>
<tr>
<td>s3srvr.12</td>
<td>304.05</td>
<td>304.05</td>
<td>&gt;1800.00</td>
<td>170.33</td>
<td></td>
</tr>
<tr>
<td>s3srvr.13</td>
<td>580.33</td>
<td>878.54</td>
<td>&gt;1800.00</td>
<td>74.49</td>
<td></td>
</tr>
<tr>
<td>s3srvr.14</td>
<td>303.21</td>
<td>303.21</td>
<td>&gt;1800.00</td>
<td>50.38</td>
<td></td>
</tr>
<tr>
<td>s3srvr.15</td>
<td>115.88</td>
<td>115.88</td>
<td>&gt;1800.00</td>
<td>21.01</td>
<td></td>
</tr>
<tr>
<td>s3srvr.16</td>
<td>305.11</td>
<td>305.11</td>
<td>&gt;1800.00</td>
<td>127.82</td>
<td></td>
</tr>
</tbody>
</table>

**TOTAL** 23 / 6435.51  22 / 10003.06  5 / 3349.48  24 / 2260.07
## Results – ntdrivers and SSH – safe

<table>
<thead>
<tr>
<th>Program</th>
<th>Best result</th>
<th>Blast</th>
<th>CPAchecker</th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
<td>-dfs</td>
<td>-predH 7</td>
<td>SBE</td>
<td>LBE</td>
</tr>
<tr>
<td>cdaudio</td>
<td>175.76</td>
<td>264.12</td>
<td>MO</td>
<td>53.55</td>
<td></td>
</tr>
<tr>
<td>diskperf</td>
<td>&gt;1800.00</td>
<td>&gt;1800.00</td>
<td>MO</td>
<td>232.00</td>
<td></td>
</tr>
<tr>
<td>floppy</td>
<td>218.26</td>
<td>&gt;1800.00</td>
<td>MO</td>
<td>56.36</td>
<td></td>
</tr>
<tr>
<td>kbfiltr</td>
<td>23.55</td>
<td>32.80</td>
<td>41.12</td>
<td></td>
<td>7.82</td>
</tr>
<tr>
<td>parport</td>
<td>738.82</td>
<td>915.79</td>
<td>MO</td>
<td>378.04</td>
<td></td>
</tr>
<tr>
<td>s3_clnt.01</td>
<td>33.01</td>
<td>1000.41</td>
<td>755.81</td>
<td>19.51</td>
<td></td>
</tr>
<tr>
<td>s3_clnt.02</td>
<td>62.65</td>
<td>312.77</td>
<td>1075.45</td>
<td>16.00</td>
<td></td>
</tr>
<tr>
<td>s3_clnt.03</td>
<td>60.62</td>
<td>314.74</td>
<td>746.31</td>
<td>49.50</td>
<td></td>
</tr>
<tr>
<td>s3_clnt.04</td>
<td>63.96</td>
<td>197.65</td>
<td>730.80</td>
<td>25.45</td>
<td></td>
</tr>
<tr>
<td>s3_srvr.01</td>
<td>811.27</td>
<td>1036.89</td>
<td>&gt;1800.00</td>
<td>125.33</td>
<td></td>
</tr>
<tr>
<td>s3_srvr.02</td>
<td>360.47</td>
<td>360.47</td>
<td>&gt;1800.00</td>
<td>122.83</td>
<td></td>
</tr>
<tr>
<td>s3_srvr.03</td>
<td>276.19</td>
<td>276.19</td>
<td>&gt;1800.00</td>
<td>98.47</td>
<td></td>
</tr>
<tr>
<td>s3_srvr.04</td>
<td>175.64</td>
<td>301.85</td>
<td>&gt;1800.00</td>
<td>71.77</td>
<td></td>
</tr>
<tr>
<td>s3_srvr.06</td>
<td>44.63</td>
<td>4.63</td>
<td>&gt;1800.00</td>
<td>59.70</td>
<td></td>
</tr>
<tr>
<td>s3_srvr.07</td>
<td>16.53</td>
<td>16.53</td>
<td>&gt;1800.00</td>
<td>85.82</td>
<td></td>
</tr>
<tr>
<td>s3_srvr.08</td>
<td>5.76</td>
<td>5.76</td>
<td>&gt;1800.00</td>
<td>61.29</td>
<td></td>
</tr>
<tr>
<td>s3_srvr.09</td>
<td>67.09</td>
<td>67.09</td>
<td>&gt;1800.00</td>
<td>126.47</td>
<td></td>
</tr>
<tr>
<td>s3_srvr.10</td>
<td>115.10</td>
<td>115.10</td>
<td>&gt;1800.00</td>
<td>63.36</td>
<td></td>
</tr>
<tr>
<td>s3_srvr.11</td>
<td>367.98</td>
<td>344.28</td>
<td>&gt;1800.00</td>
<td>162.76</td>
<td></td>
</tr>
<tr>
<td>s3_srvr.12</td>
<td>304.05</td>
<td>304.05</td>
<td>&gt;1800.00</td>
<td>170.33</td>
<td></td>
</tr>
<tr>
<td>s3_srvr.13</td>
<td>580.33</td>
<td>878.54</td>
<td>&gt;1800.00</td>
<td>74.49</td>
<td></td>
</tr>
<tr>
<td>s3_srvr.14</td>
<td>303.21</td>
<td>303.21</td>
<td>&gt;1800.00</td>
<td>50.38</td>
<td></td>
</tr>
<tr>
<td>s3_srvr.15</td>
<td>115.88</td>
<td>115.88</td>
<td>&gt;1800.00</td>
<td>21.01</td>
<td></td>
</tr>
<tr>
<td>s3_srvr.16</td>
<td>305.11</td>
<td>305.11</td>
<td>&gt;1800.00</td>
<td>127.82</td>
<td></td>
</tr>
<tr>
<td>TOTAL</td>
<td>23 / 6435.51</td>
<td>22 / 10003.06</td>
<td>5 / 3349.46</td>
<td>24 / 2260.07</td>
<td></td>
</tr>
</tbody>
</table>

1 more instance, ~3.2x faster
# Results – ntdrivers and SSH – safe

<table>
<thead>
<tr>
<th>Program</th>
<th>Best result</th>
<th>Blast -dfs -predH 7</th>
<th>CPAchecker SBE</th>
<th>LBE</th>
</tr>
</thead>
<tbody>
<tr>
<td>cdaudio</td>
<td>175.76</td>
<td>264.12</td>
<td>MO</td>
<td>53.55</td>
</tr>
<tr>
<td>diskperf</td>
<td>&gt;1800.00</td>
<td>&gt;1800.00</td>
<td>MO</td>
<td>232.00</td>
</tr>
<tr>
<td>floppy</td>
<td>218.26</td>
<td>&gt;1800.00</td>
<td>MO</td>
<td>56.36</td>
</tr>
<tr>
<td>kbfiltr</td>
<td>23.55</td>
<td>32.80</td>
<td>41.12</td>
<td>7.82</td>
</tr>
<tr>
<td>parport</td>
<td>738.82</td>
<td>915.79</td>
<td>MO</td>
<td>378.04</td>
</tr>
<tr>
<td>s3_clnt.01</td>
<td>33.01</td>
<td>1000.41</td>
<td>755.81</td>
<td>19.51</td>
</tr>
<tr>
<td>s3_clnt.02</td>
<td>62.65</td>
<td>312.77</td>
<td>1075.45</td>
<td>16.00</td>
</tr>
<tr>
<td>s3_clnt.03</td>
<td>60.62</td>
<td>314.74</td>
<td>746.31</td>
<td>49.50</td>
</tr>
<tr>
<td>s3_clnt.04</td>
<td>63.96</td>
<td>197.65</td>
<td>730.80</td>
<td>25.45</td>
</tr>
<tr>
<td>s3_srwr.01</td>
<td>811.27</td>
<td>1036.89</td>
<td>&gt;1800.00</td>
<td>125.33</td>
</tr>
<tr>
<td>s3_srwr.02</td>
<td>360.47</td>
<td>360.47</td>
<td>&gt;1800.00</td>
<td>122.83</td>
</tr>
<tr>
<td>s3_srwr.03</td>
<td>276.19</td>
<td>276.19</td>
<td>&gt;1800.00</td>
<td>98.47</td>
</tr>
<tr>
<td>s3_srwr.04</td>
<td>175.64</td>
<td>301.85</td>
<td>&gt;1800.00</td>
<td>71.77</td>
</tr>
<tr>
<td>s3_srwr.06</td>
<td>54.63</td>
<td>&gt;1800.00</td>
<td>59.70</td>
<td>85.82</td>
</tr>
<tr>
<td>s3_srwr.07</td>
<td>76.53</td>
<td>&gt;1800.00</td>
<td>85.82</td>
<td>61.29</td>
</tr>
<tr>
<td>s3_srwr.08</td>
<td>55.76</td>
<td>&gt;1800.00</td>
<td>61.29</td>
<td>126.47</td>
</tr>
<tr>
<td>s3_srwr.09</td>
<td>7.09</td>
<td>&gt;1800.00</td>
<td>126.47</td>
<td>63.36</td>
</tr>
<tr>
<td>s3_srwr.10</td>
<td>415.10</td>
<td>&gt;1800.00</td>
<td>&gt;1800.00</td>
<td>162.76</td>
</tr>
<tr>
<td>s3_srwr.11</td>
<td>367.98</td>
<td>244.28</td>
<td>&gt;1800.00</td>
<td>170.33</td>
</tr>
<tr>
<td>s3_srwr.12</td>
<td>304.05</td>
<td>304.05</td>
<td>&gt;1800.00</td>
<td>74.49</td>
</tr>
<tr>
<td>s3_srwr.13</td>
<td>520.33</td>
<td>878.54</td>
<td>&gt;1800.00</td>
<td>50.38</td>
</tr>
<tr>
<td>s3_srwr.14</td>
<td>303.21</td>
<td>303.21</td>
<td>&gt;1800.00</td>
<td>21.01</td>
</tr>
<tr>
<td>s3_srwr.15</td>
<td>115.88</td>
<td>115.88</td>
<td>&gt;1800.00</td>
<td>127.82</td>
</tr>
<tr>
<td>s3_srwr.16</td>
<td>305.11</td>
<td>305.11</td>
<td>&gt;1800.00</td>
<td></td>
</tr>
<tr>
<td>TOTAL</td>
<td>23 / 6435.51</td>
<td>22 / 1003.06</td>
<td>5 / 3349.48</td>
<td>24 / 2260.07</td>
</tr>
</tbody>
</table>
## Results – ntdrivers and SSH – unsafe

<table>
<thead>
<tr>
<th>Program</th>
<th>Best result</th>
<th>Blast -bfs-predH 7</th>
<th>CPAchecker</th>
<th>LBE</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
<td></td>
<td>SBE</td>
<td></td>
</tr>
<tr>
<td>cdaudio</td>
<td>18.79</td>
<td>99.82</td>
<td>74.39</td>
<td>9.85</td>
</tr>
<tr>
<td>diskperf</td>
<td>889.79</td>
<td>&gt;1800.00</td>
<td>26.53</td>
<td>6.78</td>
</tr>
<tr>
<td>floppy</td>
<td>119.60</td>
<td>&gt;1800.00</td>
<td>36.49</td>
<td>4.30</td>
</tr>
<tr>
<td>kbfiltir</td>
<td>46.80</td>
<td>144.25</td>
<td>75.45</td>
<td>11.52</td>
</tr>
<tr>
<td>parport</td>
<td><strong>1.67</strong></td>
<td>10.95</td>
<td>14.62</td>
<td>2.64</td>
</tr>
<tr>
<td>s3_clnt.01</td>
<td>8.84</td>
<td>28.30</td>
<td>1514.90</td>
<td>3.33</td>
</tr>
<tr>
<td>s3_clnt.02</td>
<td>9.02</td>
<td>9.02</td>
<td>843.42</td>
<td>3.27</td>
</tr>
<tr>
<td>s3_clnt.03</td>
<td>6.64</td>
<td>6.64</td>
<td>780.72</td>
<td>2.61</td>
</tr>
<tr>
<td>s3_clnt.04</td>
<td>9.78</td>
<td>9.78</td>
<td>724.04</td>
<td>3.18</td>
</tr>
<tr>
<td>s3_srbr.01</td>
<td>7.59</td>
<td>7.59</td>
<td>MO</td>
<td>2.09</td>
</tr>
<tr>
<td>s3_srbr.02</td>
<td>7.16</td>
<td>7.16</td>
<td>&gt;1800.00</td>
<td>2.10</td>
</tr>
<tr>
<td>s3_srbr.03</td>
<td>7.42</td>
<td>7.42</td>
<td>&gt;1800.00</td>
<td>2.08</td>
</tr>
<tr>
<td>s3_srbr.04</td>
<td>7.33</td>
<td>7.33</td>
<td>&gt;1800.00</td>
<td>1.93</td>
</tr>
<tr>
<td>s3_srbr.06</td>
<td>39.81</td>
<td>56.11</td>
<td>MO</td>
<td>5.08</td>
</tr>
<tr>
<td>s3_srbr.07</td>
<td>310.84</td>
<td>310.84</td>
<td>&gt;1800.00</td>
<td>28.35</td>
</tr>
<tr>
<td>s3_srbr.08</td>
<td>40.51</td>
<td>73.59</td>
<td>&gt;1800.00</td>
<td>36.47</td>
</tr>
<tr>
<td>s3_srbr.09</td>
<td>265.48</td>
<td>265.48</td>
<td>&gt;1800.00</td>
<td>4.94</td>
</tr>
<tr>
<td>s3_srbr.10</td>
<td>40.24</td>
<td>66.88</td>
<td>&gt;1800.00</td>
<td>12.01</td>
</tr>
<tr>
<td>s3_srbr.11</td>
<td>49.05</td>
<td>49.05</td>
<td>&gt;1800.00</td>
<td>4.80</td>
</tr>
<tr>
<td>s3_srbr.12</td>
<td>38.66</td>
<td>38.66</td>
<td>&gt;1800.00</td>
<td>6.11</td>
</tr>
<tr>
<td>s3_srbr.13</td>
<td>251.56</td>
<td>251.56</td>
<td>&gt;1800.00</td>
<td>15.20</td>
</tr>
<tr>
<td>s3_srbr.14</td>
<td>39.94</td>
<td>53.93</td>
<td>1656.54</td>
<td>4.63</td>
</tr>
<tr>
<td>s3_srbr.15</td>
<td>40.19</td>
<td>77.51</td>
<td>&gt;1800.00</td>
<td>10.19</td>
</tr>
<tr>
<td>s3_srbr.16</td>
<td>39.54</td>
<td>55.97</td>
<td>&gt;1800.00</td>
<td>5.21</td>
</tr>
<tr>
<td><strong>TOTAL</strong></td>
<td><strong>24 / 2296.25</strong></td>
<td><strong>22 / 1637.84</strong></td>
<td><strong>10 / 5747.10</strong></td>
<td><strong>24 / 188.67</strong></td>
</tr>
</tbody>
</table>
## Results – ntdrivers and SSH – unsafe

<table>
<thead>
<tr>
<th>Program</th>
<th>Best result</th>
<th>-bfs -predH 7</th>
<th>CPAchecker</th>
<th>SBE</th>
<th>LBE</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
<td></td>
<td>Blast</td>
<td>CPAchecker</td>
<td>LBE</td>
</tr>
<tr>
<td>cdaudio</td>
<td>18.79</td>
<td>99.82</td>
<td>74.39</td>
<td>9.85</td>
<td></td>
</tr>
<tr>
<td>diskperf</td>
<td>889.79</td>
<td>&gt;1800.00</td>
<td>26.53</td>
<td>6.78</td>
<td></td>
</tr>
<tr>
<td>floppy</td>
<td>119.60</td>
<td>&gt;1800.00</td>
<td>36.49</td>
<td>4.30</td>
<td></td>
</tr>
<tr>
<td>kbfiltir</td>
<td>46.80</td>
<td>144.25</td>
<td>75.45</td>
<td>11.52</td>
<td></td>
</tr>
<tr>
<td>parport</td>
<td>1.67</td>
<td>10.95</td>
<td>14.62</td>
<td>2.64</td>
<td></td>
</tr>
<tr>
<td>s3_clnt.01</td>
<td>8.84</td>
<td>28.30</td>
<td>1514.90</td>
<td>3.33</td>
<td></td>
</tr>
<tr>
<td>s3_clnt.02</td>
<td>9.02</td>
<td>9.02</td>
<td>843.42</td>
<td>3.27</td>
<td></td>
</tr>
<tr>
<td>s3_clnt.03</td>
<td>6.64</td>
<td>6.64</td>
<td>780.72</td>
<td>2.61</td>
<td></td>
</tr>
<tr>
<td>s3_clnt.04</td>
<td>9.78</td>
<td>9.78</td>
<td>724.04</td>
<td>3.18</td>
<td></td>
</tr>
<tr>
<td>s3_srvr.01</td>
<td>7.59</td>
<td>7.59</td>
<td>MO</td>
<td>2.09</td>
<td></td>
</tr>
<tr>
<td>s3_srvr.02</td>
<td>7.16</td>
<td>7.16</td>
<td>&gt;1800.00</td>
<td>2.10</td>
<td></td>
</tr>
<tr>
<td>s3_srvr.03</td>
<td>7.42</td>
<td>7.42</td>
<td>&gt;1800.00</td>
<td>2.08</td>
<td></td>
</tr>
<tr>
<td>s3_srvr.04</td>
<td>7.33</td>
<td>7.33</td>
<td>&gt;1800.00</td>
<td>1.93</td>
<td></td>
</tr>
<tr>
<td>s3_srvr.06</td>
<td>6.11</td>
<td>6.11</td>
<td>MO</td>
<td>5.08</td>
<td></td>
</tr>
<tr>
<td>s3_srvr.07</td>
<td>0.84</td>
<td>0.84</td>
<td>&gt;1800.00</td>
<td>28.35</td>
<td></td>
</tr>
<tr>
<td>s3_srvr.08</td>
<td>3.59</td>
<td>3.59</td>
<td>&gt;1800.00</td>
<td>36.47</td>
<td></td>
</tr>
<tr>
<td>s3_srvr.09</td>
<td>5.48</td>
<td>5.48</td>
<td>&gt;1800.00</td>
<td>4.94</td>
<td></td>
</tr>
<tr>
<td>s3_srvr.10</td>
<td>36.88</td>
<td>36.88</td>
<td>&gt;1800.00</td>
<td>12.01</td>
<td></td>
</tr>
<tr>
<td>s3_srvr.11</td>
<td>49.05</td>
<td>49.05</td>
<td>&gt;1800.00</td>
<td>4.80</td>
<td></td>
</tr>
<tr>
<td>s3_srvr.12</td>
<td>38.66</td>
<td>38.66</td>
<td>&gt;1800.00</td>
<td>6.11</td>
<td></td>
</tr>
<tr>
<td>s3_srvr.13</td>
<td>251.56</td>
<td>251.56</td>
<td>&gt;1800.00</td>
<td>15.20</td>
<td></td>
</tr>
<tr>
<td>s3_srvr.14</td>
<td>39.94</td>
<td>53.93</td>
<td>1656.54</td>
<td>4.63</td>
<td></td>
</tr>
<tr>
<td>s3_srvr.15</td>
<td>40.19</td>
<td>77.51</td>
<td>&gt;1800.00</td>
<td>10.19</td>
<td></td>
</tr>
<tr>
<td>s3_srvr.16</td>
<td>39.54</td>
<td>55.97</td>
<td>&gt;1800.00</td>
<td>5.21</td>
<td></td>
</tr>
<tr>
<td>TOTAL</td>
<td>24 / 2296.25</td>
<td>22 / 1637.84</td>
<td>10 / 5747.10</td>
<td>24 / 188.67</td>
<td></td>
</tr>
</tbody>
</table>
## Results – ntdrivers and SSH – unsafe

| Program       | Best result | -bfs -predH 7 | CPAchecker | |
|---------------|-------------|---------------|------------|
|               | SBE         | LBE           |            | |
| cdadudio      | 18.79       | 99.82         | 74.39      | 9.85 |
| diskperf      | 889.79      | >1800.00      | 26.53      | 6.78 |
| floppy        | 119.60      | >1800.00      | 36.49      | 4.30 |
| kbfiltr       | 46.80       | 144.25        | 75.45      | 11.52 |
| parport       | 1.67        | 10.95         | 14.62      | 2.64 |
| s3_clnt.01    | 8.84        | 28.30         | 1514.90    | 3.33 |
| s3_clnt.02    | 9.02        | 9.02          | 843.42     | 3.27 |
| s3_clnt.03    | 6.64        | 6.64          | 780.72     | 2.61 |
| s3_clnt.04    | 9.78        | 9.78          | 724.04     | 3.18 |
| s3_srvr.01    | 7.59        | 7.59          | MO         | 2.09 |
| s3_srvr.02    | 7.16        | 7.16          | >1800.00   | 2.10 |
| s3_srvr.03    | 7.42        | 7.42          | >1800.00   | 2.08 |
| s3_srvr.04    | 7.33        | 7.33          | >1800.00   | 1.93 |
| s3_srvr.06    | 6.11        | 6.11          | MO         | 5.08 |
| s3_srvr.07    | 0.84        | 0.84          | >1800.00   | 28.35 |
| s3_srvr.08    | 3.59        | 3.59          | >1800.00   | 36.47 |
| s3_srvr.09    | 5.48        | 5.48          | >1800.00   | 4.94 |
| s3_srvr.10    | 45.24       | 56.88         | >1800.00   | 12.01 |
| s3_srvr.11    | 49.05       | 49.05         | >1800.00   | 4.80 |
| s3_srvr.12    | 38.66       | 38.66         | >1800.00   | 6.11 |
| s3_srvr.13    | 251.56      | 251.56        | >1800.00   | 15.20 |
| s3_srvr.14    | 39.34       | 53.93         | 1656.54    | 4.63 |
| s3_srvr.15    | 40.19       | 77.51         | >1800.00   | 10.19 |
| s3_srvr.16    | 39.54       | 55.97         | >1800.00   | 5.21 |
| **TOTAL**     | **24 / 2296.25** | **22 / 1637.84** | **10 / 5747.10** | **24 / 188.67** |

2 more instances, ~9.2x faster
Conclusions

♦ Large-Block Encoding: new approach to ART-based SW MC, for better exploiting modern SMT solvers
  ♦ Move cost from explicit path enumeration to symbolic computations within the SMT solver
  ♦ Nice improvements on standard benchmark programs

♦ Future work
  ♦ Dynamic computation of large blocks
    ♦ Allows for adjusting the symbolic/explicit tradeoff “on the fly”
  ♦ Experiment with approximated abstractions, cheaper than Boolean
  ♦ Extend to McMillan's CAV'06 approach (no predicate abstraction, use interpolants directly)