Model Checking with Maximal Causality Reduction

Jeff Huang
Assistant Professor

Parasol

COMPUTER SCIENCE & ENGINEERING
TEXAS A&M UNIVERSITY
A Real Bug – $12 million loss of equipment


curPos = new Point(1,2);
class Point { int x, y; }

Thread 1:
newPos = new
    Point(curPos.x+1, curPos.y+1);

Thread 2:
while (newPos != null)
if (newPos.x+1 != newPos.y)
    ERROR

x=0
y=0
x=curPos.x+1
y=curPos.y+1
curPos = object
A Real Bug – $12 million loss of equipment


curPos = new Point(1,2);
class Point { int x, y; }

Thread 1:
newPos = new
    Point(curPos.x+1, curPos.y+1);

Thread 2:
while (newPos != null)
if (newPos.x+1 != newPos.y)
ERROR

statements are out of program order
Maximal Causality Reduction

• Open source:  https://github.com/parasol-aser/JMCR

• Implementation
  - Java 8, multi-threading
  - The Z3 SMT solver

• Evaluation
  - Takes only two runs to find the error in <1s
  - Orders of magnitude more effective than partial order reduction and bounded model checking
  - Finding new errors (data races and NPEs) in extensively studied popular benchmarks
2007 Turing Award

Edmund Clarke  Allen Emerson  Joseph Sifakis

For their role in “developing Model-Checking into a highly effective verification technology …”
The Key Challenge: State Explosion

ACM 2007 Turing Award
Edmund Clarke, Allen Emerson, and Joseph Sifakis

Model Checking: Algorithmic Verification and Debugging

ACM Turing Award Citation
In 1981, Edmund M. Clarke and E. Allen Emerson, working in the USA, and Joseph Sifakis working independently in France, authored seminal papers that founded what has become the highly successful field of Model Checking. This verification technology provides an algorithmic means of precisely describe what constitutes correct behavior. This makes it possible to contemplate mathematically establishing that the program behavior conforms to the correctness specification. In most early work, this entailed constructing a formal proof of correctness. In contradistinction, Model Checking avoids proofs.
The Key Challenge: State Explosion

ACM 2007 Turing Award
Edmund Clarke, Allen Emerson, and Joseph Sifakis
Model Checking: Algorithmic Verification and Debugging

4. **State Explosion** Challenges for the Future

The state explosion problem is likely to remain the major challenge in Model Checking. There are many directions for future research on this problem, some of which are listed below.

- Software Model Checking, in particular, combining Model Checking and Static Analysis
Two Classical Approaches

- **Partial Order Reduction** [2014 CAV Award] to Godefroid, Peled, Valmari, and Wolper
  - Reduce the size of the state space that needs to be searched
  - Exploit the *independence* between concurrently executed transitions, which result in the same state

- **Bounded Model Checking** Clarke, Biere, Raimi, Zhu (2001)
  - Limit the searched space to a certain bound
Partial Order Reduction

The two sequences

- \( s \xrightarrow{\alpha} s_1 \xrightarrow{\beta} r \)
- \( s \xrightarrow{\beta} s_2 \xrightarrow{\alpha} r \)

belong to the same equivalent class.

If the specification does not distinguish between these sequences, it is beneficial to consider only one with \( 2 + 1 \) states.
Partial Order Reduction

The two sequences

- $s \rightarrow \alpha \rightarrow s_1 \rightarrow \beta \rightarrow r$
- $s \rightarrow \beta \rightarrow s_2 \rightarrow \alpha \rightarrow r$

belong to the same equivalent class.

If the specification does not distinguish between these sequences, it is beneficial to consider only one with $2 + 1$ states.
Bounded Model Checking

- Restrict search to states that are reachable from initial state within fixed number $k$ of transitions.

Can the given property fail in $k$-steps?

$I(V_0) \land T(V_0, V_1) \land \cdots \land T(V_{k-1}, V_k) \land (\vdash P(V_0) \land \cdots \land \vdash P(V_k))$

Initial state $\xrightarrow[k]{\text{k-steps}}$ Property fails in some step
Bounded Model Checking

- Restrict search to states that are reachable from initial state within fixed number $k$ of transitions.

Can the given property fail in $k$-steps?

\[ I(V_0) \land T(V_0, V_1) \land \ldots \land T(V_{k-1}, V_k) \land (\exists P(V_0) \land \ldots \land \exists P(V_k)) \]

Initial state $q_0$, $k=3$ steps, $q_1$, $k=7$ steps, $q_7$, $k=10$ steps.

Property fails in some step.

Incomplete i.e., limited to $k$
Example

initially $x=y=0$

<table>
<thead>
<tr>
<th>T1</th>
<th>T2</th>
<th>T3</th>
</tr>
</thead>
<tbody>
<tr>
<td>loop twice:</td>
<td>loop twice:</td>
<td>loop twice:</td>
</tr>
<tr>
<td>1: lock(l)</td>
<td>5: lock(l)</td>
<td>11: if($x&gt;1$)</td>
</tr>
<tr>
<td>2: $x=1$</td>
<td>6: $x=0$</td>
<td>12: if($y==3$)</td>
</tr>
<tr>
<td>3: $y=1$</td>
<td>7: unlock(l)</td>
<td>13:</td>
</tr>
<tr>
<td>4: unlock(l)</td>
<td>8: if($x&gt;0$)</td>
<td>14: else</td>
</tr>
<tr>
<td></td>
<td>9: $y++$</td>
<td>15: $y=2$</td>
</tr>
<tr>
<td></td>
<td>10: $x=2$</td>
<td></td>
</tr>
</tbody>
</table>
Example

Initially $x = y = 0$

**T1**
- Loop twice:
  1: lock(l)
  2: $x = 1$
  3: $y = 1$
  4: unlock(l)

**T2**
- Loop twice:
  5: lock(l)
  6: $x = 0$
  7: unlock(l)
  8: if($x > 0$)
  9: $y++$
  10: $x = 2$

**T3**
- Loop twice:
  11: if($x > 1$)
  12: if($y == 3$)
  13: Error
  14: else
  15: $y = 2$

T2T2T2 - T1T1T1T1 - T2T2T2T2 - T3T3T3 - T2T2T2T2 - T1T1 - T2T2T2T2 - T3T3
Example

Initially $x=y=0$

<table>
<thead>
<tr>
<th>T1</th>
<th>T2</th>
<th>T3</th>
</tr>
</thead>
</table>

**Loop twice:**

1: lock(l)
2: $x=1$
3: $y=1$
4: unlock(l)

5: lock(l)
6: $x=0$
7: unlock(l)
8: if($x>0$)
9: $y++$
10: $x=2$
11: if($x>1$)
12: if($y==3$)
13: $y=2$

**Error**

7 thread context switches
Example

Initially $x = y = 0$

<table>
<thead>
<tr>
<th>T1</th>
<th>T2</th>
<th>T3</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>loop twice:</strong></td>
<td><strong>loop twice:</strong></td>
<td><strong>loop twice:</strong></td>
</tr>
</tbody>
</table>

DFS explores **3,293,931** runs in an hour **without** finding the error

3: $y = 1$
4: `unlock(l)`
8: `if(x > 0)`
9: `$y++$`
10: `$x = 2$`
15: `$y = 2$`
14: `else`

**7 thread context switches**
**Example**  

Initially $x = y = 0$

<table>
<thead>
<tr>
<th>T1</th>
<th>T2</th>
<th>T3</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>loop twice:</strong></td>
<td><strong>loop twice:</strong></td>
<td><strong>loop twice:</strong></td>
</tr>
<tr>
<td>1: lock(l)</td>
<td>5: lock(l)</td>
<td>11: if($x &gt; 1$)</td>
</tr>
<tr>
<td>2: $x = 1$</td>
<td>6: $x = 0$</td>
<td>12: if($y == 3$)</td>
</tr>
<tr>
<td>3: $y = 1$</td>
<td>7: unlock(l)</td>
<td>13: <strong>Error</strong></td>
</tr>
<tr>
<td>4: unlock(l)</td>
<td>8: if($x &gt; 0$)</td>
<td>14: else</td>
</tr>
<tr>
<td></td>
<td>9: $y++$</td>
<td>15: $y = 2$</td>
</tr>
<tr>
<td></td>
<td>10: $x = 2$</td>
<td></td>
</tr>
</tbody>
</table>

T2T2T2 - T1T1T1T1 - T2T2T2T2 - T3T3T3 - T2T2T2T2 - T1T1 - T2T2T2T2 - T3T3  

7 thread context switches
**Example**

Initially $x = y = 0$

<table>
<thead>
<tr>
<th>Line</th>
<th>Code</th>
<th>Line</th>
<th>Code</th>
</tr>
</thead>
<tbody>
<tr>
<td>2</td>
<td>$x = 1$</td>
<td>7</td>
<td>unlock(l)</td>
</tr>
<tr>
<td>3</td>
<td>$y = 1$</td>
<td>8</td>
<td>if($x &gt; 0$)</td>
</tr>
<tr>
<td>4</td>
<td>unlock(l)</td>
<td>9</td>
<td>$y++$</td>
</tr>
<tr>
<td>10</td>
<td>$x = 2$</td>
<td>12</td>
<td>if($y == 3$)</td>
</tr>
<tr>
<td>13</td>
<td></td>
<td>14</td>
<td>else</td>
</tr>
<tr>
<td>15</td>
<td></td>
<td>15</td>
<td>$y = 2$</td>
</tr>
</tbody>
</table>

**Bounded Model Checking**

Bounding #thread preemptions

- 77,322 executions
- 20 seconds

7 thread context switches
Example

Initially $x = y = 0$

Bounded Model Checking
Bounding \#thread preemptions

Partial Order Reduction
Based on happens-before

77,322 executions
20 seconds

3,782 executions
3 seconds

12: if(y == 3)
13: Error
14: else
Example

initially x=y=0

Bounded Model Checking
Bounding #thread preemptions

Partial Order Reduction
Based on happens-before

Maximal Causality Reduction

77,322 executions
20 seconds

3,782 executions
3 seconds

46 executions
2 seconds

12: if(y==3)
13: Error
14: else
Example

Initially: $x = y = 0$

**Bounded Model Checking**
Bounding #thread preemptions

+ $egin{array}{ll}
2: & x = 1 \\
3: & y = 1 \\
7: & \text{unlock}(l) \\
8: & \text{if}(x > 0) \\
12: & \text{if}(y == 3) \\
13: & \text{Error}
\end{array}$

77,322 executions
20 seconds

**Partial Order Reduction**
Based on happens-before

**Maximal Causality Reduction**

46 executions
2 seconds

**Happens-Before Limitation**
Happens-Before Limitation

Enforces dependence between conflicting reads and writes

Happens-before: **six** non-redundant transitions

p: write x  q: write x  r: read x

p.q.r  p.r.q  q.p.r  q.r.p  r.p.q  r.q.p
Happens-Before Limitation

Enforces dependence between conflicting reads and writes

\[ p: \text{write } x \]
\[ q: \text{write } x \]
\[ r: \text{read } x \]

Happens-before: \textbf{six} non-redundant transitions
\[ p.q.r \quad p.r.q \quad q.p.r \quad q.r.p \quad r.p.q \quad r.q.p \]

In fact: only \textbf{four} are non-redundant
\[ p.q.r \quad == \quad q.r.p \quad \quad r.q.p \quad == \quad r.p.q \]
Happens-Before Limitation

Enforces dependence between conflicting reads and writes

\[ \text{p: write } x \quad \text{q: write } x \quad \text{r: read } x \]

Happens-before: **six** non-redundant transitions

\[ \text{p.q.r} \quad \text{p.r.q} \quad \text{q.p.r} \quad \text{q.r.p} \quad \text{r.p.q} \quad \text{r.q.p} \]

In fact: only **four** are non-redundant

\[ \text{p.q.r} \quad \text{q.r.p} \quad \text{r.q.p} \quad \text{r.p.q} \]

\text{r} \text{ is the only read}
Happens-Before Limitation

Enforces dependence between conflicting reads and writes

\[
\begin{align*}
\text{p:} & \quad \text{write } x \\
\text{q:} & \quad \text{write } x \\
\text{r:} & \quad \text{read } x
\end{align*}
\]

Happens-before: six non-redundant transitions

If \( p \) and \( q \) write the same value, then only two non-redundant transitions:

\[
\begin{align*}
p.q.r & = q.p.r = q.r.p = p.r.q \\
r.q.p & = r.p.q
\end{align*}
\]
Example

Initially \( x = y = 0 \)

<table>
<thead>
<tr>
<th>T1</th>
<th>T2</th>
<th>T3</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>loop twice:</strong></td>
<td><strong>loop twice:</strong></td>
<td><strong>loop twice:</strong></td>
</tr>
<tr>
<td>1: lock(l)</td>
<td>5: lock(l)</td>
<td>11: if(( x &gt; 1 ))</td>
</tr>
<tr>
<td>2: ( x = 1 )</td>
<td>6: ( x = 0 )</td>
<td>12: if(( y == 3 ))</td>
</tr>
<tr>
<td>3: ( y = 1 )</td>
<td>8: if(( x &gt; 0 ))</td>
<td>13:</td>
</tr>
<tr>
<td>4: unlock(l)</td>
<td>9: ( y++ )</td>
<td>14: else</td>
</tr>
<tr>
<td></td>
<td>10: ( x = 2 )</td>
<td>15: ( y = 2 )</td>
</tr>
</tbody>
</table>
Example

Initially $x=y=0$

<table>
<thead>
<tr>
<th></th>
<th>T1</th>
<th>T2</th>
<th>T3</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>lock(l)</td>
<td>lock(l)</td>
<td>11: if($x&gt;1$)</td>
</tr>
<tr>
<td>2</td>
<td>$x=1$</td>
<td>6: $x=0$</td>
<td>12: if($y==3$)</td>
</tr>
<tr>
<td>3</td>
<td>$y=1$</td>
<td>8: if($x&gt;0$)</td>
<td>13:</td>
</tr>
<tr>
<td>4</td>
<td>unlock(l)</td>
<td>9: $y++$</td>
<td>14: else</td>
</tr>
<tr>
<td></td>
<td>loop twice:</td>
<td>10: $x=2$</td>
<td>15: $y=2$</td>
</tr>
</tbody>
</table>
Example

Initially, \( x = y = 0 \)

Bounded Model Checking

Bounding \#thread preemptions

```
2: \textcolor{red}{x=1} \quad \text{loop \textbf{10 times}}
3: \textcolor{red}{y=1}
4: \text{unlock}(l)
5: \text{lock}(l)
8: \textcolor{red}{if(x>0)}
9: \textcolor{red}{y++}
10: \textcolor{red}{x=2}
11: \text{if(x>1)}
12: \textcolor{red}{if(y==3)}\quad \textbf{Error}
13: 
14: \text{else}
15: \textcolor{red}{y=2}
```
Example
initially x=y=0

Bounded Model Checking
Bounding #thread preemptions
520,959 executions
183 seconds

Partial Order Reduction
Based on happens-before
221,852 executions
93 seconds
Example

initially $x=y=0$

Bounded Model Checking
Bounding #thread preemptions

Partial Order Reduction
Based on happens-before

Maximal Causality Reduction

520,959 executions
183 seconds

221,852 executions
93 seconds

50 executions
4 seconds
Maximal Causality Reduction

Key idea: characterizing redundant transitions with maximal causality

\[ \Omega(t) \]: a maximal set of equivalent transitions

\[ t \]: takes the value of reads and writes into consideration

\[ \Omega(t) \]: contains all transitions which all programs that can generate \( t \) can also generate

Serbanuta, Chen and Rosu, Maximal Causal Models for Sequentially Consistent Systems, RV'12
Maximal Causality Reduction

1. Online tracing $t$
2. Construct $\Omega(t)$
3. Offline property checking with $\Omega(t)$
4. Generate new seed interleavings with $\Omega(t)$
Maximal Causality Reduction

Seed interleaving: an interleaving in $\Omega(t)$ with at least one read forced to see a different value

Following a seed interleaving will produce a new state

1. Online tracing $t$
2. Construct $\Omega(t)$
3. Offline property checking with $\Omega(t)$
4. Generate new...
Maximal Causality Reduction

\[ N = 1, 2, \ldots, 10 \]

Initially \( x = y = 0 \)

<table>
<thead>
<tr>
<th>T1</th>
<th>T2</th>
<th>T3</th>
</tr>
</thead>
<tbody>
<tr>
<td>loop twice:</td>
<td>loop twice:</td>
<td>loop twice:</td>
</tr>
<tr>
<td>1: lock(l)</td>
<td>5: lock(l)</td>
<td>11: if(x&gt;1)</td>
</tr>
<tr>
<td>2: ( x = 1 )</td>
<td>6: ( x = 0 )</td>
<td>12: if(y==3)</td>
</tr>
<tr>
<td>3: ( y = 1 )</td>
<td>8: if(x&gt;0)</td>
<td>13:</td>
</tr>
<tr>
<td>4: unlock(l)</td>
<td>9: ( y++ )</td>
<td>else</td>
</tr>
<tr>
<td></td>
<td>10: ( x = 2 )</td>
<td>14:</td>
</tr>
<tr>
<td></td>
<td></td>
<td>15: ( y = 2 )</td>
</tr>
</tbody>
</table>

Loop \( N \) times

Error
Maximal Causality Reduction

\[ N = 1, 2, \ldots, 10 \]
Maximal Causality Reduction

\[ N = 1, 2, \ldots, 10 \]

MCR is almost insensitive to N when N>3

Reduced \#explorations by BMC+POR

by *two orders of magnitude*
Maximal Causality Reduction

1. Online tracing $t$
2. Construct $\Omega(t)$
3. Offline property checking with $\Omega(t)$
4. Generate new seed interleavings with $\Omega(t)$
Maximal Causality Reduction

A constraint-based approach

1. Online tracing
2. Construct $\Omega(t)$
3. Offline property checking with $\Omega(t)$
4. Generate new seed interleavings with $\Omega(t)$
Constructing $\Omega(t)$

A constraint-based approach

Introduce an ORDER variable for each event in the trace $t$

**Must-happen-before constraints** ($\Phi_{mhb}$)

E.g., $O_1 < O_2$ if events $e_1$ and $e_2$ are by the same thread, and $e_1$ occurs before $e_2$

**Lock-mutual-exclusion constraints** ($\Phi_{lock}$)

$$\bigwedge_{(e_a,e_b),(e_c,e_d) \in S_t} (O_{e_b} < O_{e_c} \lor O_{e_d} < O_{e_a})$$

**Data-validity constraints** ($\Phi_{rw}$)

$$\Phi_{rw}(e) \equiv \bigwedge_{r \in e} \Phi_{value}(r, value(r))$$

$$\Phi_{value}(r, v') \equiv \bigvee_{w \in W_{v'}} (\Phi_{rw}(w) \land O_w < O_r \land (O_{w'} < O_w \lor O_r < O_{w'}))$$

$$\Phi_{rw} \equiv \bigvee_{e \in \tau} \Phi_{rw}(e)$$
Constructing $\Omega(t)$

A constraint-based approach

Introduce an ORDER variable for each event in the trace $t$

**Must-happen-before constraints ($\Phi_{mhb}$)**

E.g., $O_1 < O_2$ if events $e_1$ and $e_2$ are by the same thread, and $e_1$ occurs before $e_2$

**Lock-mutual-exclusion constraints ($\Phi_{lock}$)**

$\bigwedge_{(e_a,e_b),(e_c,e_d) \in S_t} (O_{e_b} < O_{e_c} \lor O_{e_d} < O_{e_a})$

**Data-validity constraints ($\Phi_{rw}$)**

An event is feasible if every read that must-happen-before it in the trace $t$ returns the same value as that in $t$

$\Phi_{rw} \equiv \bigvee_{e \in \tau} \Phi_{rw}(e)$
Generating Seed Interleavings

Main idea: enforce a read to see a new value

\[
\begin{align*}
&\text{for } r = \text{read}(t, x, v) \in \tau \text{ do} \\
&\text{for } w = \text{write}(_, x, v') \in \tau \land v' \neq v \text{ do} \\
&\quad \Phi_{seed}(r, w) \equiv \Phi_{sync} \land \Phi_{rw}(r) \land \Phi_{rw}(w) \land \Phi_{value}(r, v')
\end{align*}
\]
Generating Seed Interleavings

Main idea: enforce a read to see a new value

```latex
\begin{align*}
\text{for } r &= \text{read}(t, x, v) \in \tau \text{ do} \\
\text{for } w &= \text{write}(\_, x, v') \in \tau \land v' \neq v \text{ do} \\
\Phi_{seed}(r, w) &= \Phi_{sync} \land \Phi_{rw}(r) \land \Phi_{rw}(w) \land \Phi_{value}(r, v')
\end{align*}
```

- Every seed interleaving is feasible and has at least one new event: a read event that returns a new value
- Termination: when no new seed interleaving can be generated
Generating Seed Interleavings

Main idea: enforce a read to see a new value

\[
\text{for } r = \text{read}(t, x, v) \in \tau \text{ do }
\]
\[
\text{for } w = \text{write}(\_, x, v') \in \tau \land v' \neq v \text{ do }
\]
\[
\Phi_{seed}(r, w) \equiv \Phi_{sync} \land \Phi_{rw}(r) \land \Phi_{rw}(w) \land \Phi_{value}(r, v')
\]

- Every event of a new value is non-redundant

- Termination: when no new seed interleaving can be generated
Generating Seed Interleavings

Main idea: enforce a read to see a new value

\[
\text{for } r = \text{read}(t, x, v) \in \tau \text{ do }
\]
\[
\text{for } w = \text{write}(_, x, v') \in \tau \land v' \neq v \text{ do }
\]
\[
\Phi_{seed}(r, w) \equiv \Phi_{sync} \land \Phi_{rw}(r) \land \Phi_{rw}(w) \land \Phi_{value}(r, v')
\]

- Every read sees a new event
- No seed interleaving is redundant
- No seed interleaving is missed
Seed Interleaving Exploration

Initially $x=y=0$

<table>
<thead>
<tr>
<th>$T1$</th>
<th>$T2$</th>
<th>$T3$</th>
</tr>
</thead>
<tbody>
<tr>
<td>loop twice:</td>
<td>loop twice:</td>
<td>loop twice:</td>
</tr>
<tr>
<td>1: lock(l)</td>
<td>5: lock(l)</td>
<td>11: if($x&gt;1$)</td>
</tr>
<tr>
<td>2: $x=1$</td>
<td>6: $x=0$</td>
<td>12: if($y==3$)</td>
</tr>
<tr>
<td>3: $y=1$</td>
<td>7: unlock(l)</td>
<td>13:</td>
</tr>
<tr>
<td>4: unlock(l)</td>
<td>8: if($x&gt;0$)</td>
<td>14: else</td>
</tr>
<tr>
<td></td>
<td>9: $y++$</td>
<td>15: $y=2$</td>
</tr>
<tr>
<td></td>
<td>10: $x=2$</td>
<td></td>
</tr>
</tbody>
</table>

Diagram:

- $s0$ is the initial state.
- $s1$, $s2$, $s3$, and $s4$ are the states after the loops in $T1$, $T2$, and $T3$, respectively.
- $s1.1$, $s1.2$, $s1.3$, $s4.1$, $s4.2$, and $s4.3$ are the states after the first, second, and third mutations, respectively.
- $s1.1.2.1$, $s1.1.2.2$, $s4.3.1.1$, and $s4.3.1.2$ are further states after additional mutations.

Note: The diagram shows a tree-like structure with states branching out, illustrating the exploration process.
Seed Interleaving Exploration

\[ \begin{align*}
&\text{S0} \\
&\quad \downarrow \\
&\quad \text{S1} \\
&\quad \downarrow \\
&\quad \text{S1.1.2.2} \\
&\quad \downarrow \\
&\quad \text{s1.1} \quad \text{s1.2} \quad \text{s1.3} \quad \cdots \quad \text{s4.1} \quad \text{s4.2} \quad \text{s4.3} \\
&\quad \downarrow \\
&\quad \text{s1.1.2.1} \quad \text{s1.1.2.2} \\
&\quad \downarrow \\
&\quad \text{s4.3.1.1} \quad \text{s4.3.1.2} \\
\end{align*} \]
Checking Property Constraints

Checking assertions:

\[ \Phi_{sync} \land (\bigwedge_{e \in R} \Phi_{rw}(e) \land \nu(e)) \land \phi_{assert}(R) \]

- synchronization constraints
- data-validity constraints
- assertion formula over a set of reads
Checking assertions:

\[ \Phi_{\text{sync}} \land ( \bigwedge_{e \in R} \Phi_{\text{rw}}(e) \land \nu(e)) \land \phi_{\text{assert}}(R) \]

- \( \Phi_{\text{sync}} \): synchronization constraints
- \( \bigwedge_{e \in R} \Phi_{\text{rw}}(e) \land \nu(e) \): data-validity constraints over a set of reads
- \( \phi_{\text{assert}}(R) \): assertion formula over a set of reads

E.g., Null Pointer Dereferences:

\[ \Phi_{\text{sync}} \land \Phi_{\text{rw}}(e) \land (\text{value}(e) = \text{NULL}) \]
Checking Property
Constraints

Checking assertions:

\[ \Phi_{\text{sync}} \land (\bigwedge_{e \in R} \Phi_{\text{rw}}(e) \land \nu(e)) \land \phi_{\text{assert}}(R) \]

- synchronization constraints
- data-validity constraints
- assertion formula over a set of reads

E.g., Null Pointer Deferences:

\[ \Phi_{\text{sync}} \land \Phi_{\text{rw}}(e) \land (\text{value}(e) = \text{NULL}) \]

Checking data races:

\[ \Phi_{\text{sync}} \land (O_{e_a} = O_{e_b}) \land \Phi_{\text{rw}}(e_a) \land \Phi_{\text{rw}}(e_b) \]
Relaxed Memory Models

**Must-happen-before constraints** \((\Phi_{mhb})\)

**Init:** \(x=y=0\)

**thread 1:**
- \(x = 1\) \(//a1\)
- \(a = y\) \(//a2\)

**thread 2:**
- \(y = 1\) \(//b1\)
- \(b = x\) \(//b2\)

**Under SC:**
- \(O_{a1} < O_{a2}\)
- \(O_{b1} < O_{b2}\)

**Under TSO/PSO**
- \(O_{a1}, O_{a2}, O_{b1}, O_{b2}\)
A Real Bug – $12 million loss of equipment


Init: x=1, y=2

T1
1: T2.start()
2: z=0
3: x++
4: y++
5: z=1
6: T2.join()

T2
7: if (z==1)
8:   x assert(x+1==y)
A Real Bug – $12 million loss of equipment


Init: x=1, y=2

T1
1: T2.start()
2: z=0
3: x++
4: y++
5: z=1
6: T2.join()

T2
7: if (z==1)
8: \(\times\) assert(x+1==y)

Read-Write Constraints
\((R_z^7 = 0 \land O_7 < O_2) \lor (R_z^7 = W_z^5 \land O_5 < O_7 \land (O_2 < O_5 \lor O_7 < O_2))\)

Memory Order Constraints
SC
\[0_1 < 0_2 < 0_3^{Rx} < 0_3^{Wx} < 0_4^{Rx}\]
PSO
\[0_1 < 0_2 < 0_5 < 0_6\]
\[0_3^{Rx} < 0_3^{Wx} < 0_4^{Rx} < 0_4^{Wx}\]

Path Constraints
\(R_z^7 = 1\)

Failure Constraints
\(R_x^8 + 1! = R_y^8\)
A Real Bug – $12 million loss of equipment


**Init:** $x=1$, $y=2$

**T1**

1: $\text{T2.start()}$
2: $z=0$
3: $x++$
4: $y++$
5: $z=1$
6: $\text{T2.join()}$

**T2**

7: if ($z==1$)
8: $\times\text{ assert}(x+1==y)$

---

**Read-Write Constraints**

$R^7_Z = 0 \land O_7 < O_2 \lor (R^7_Z = W^5_Z \land O_5 < O_7 \land (O_2 < O_5 \lor O_7 < O_2))$

**Memory Order Constraints**

**SC**

- $O_1 < O_2 < O^{R_x}_3 < O^{W_x}_3 < O^{R_x}_4$
- $< O^{W_x}_4 < O_5 < O_6$
- $O_7 < O^x_8 < O^y_8$

**PSO**

- $O_1 < O_2 < O_5 < O_6$
- $O^{R_x}_3 < O^{W_x}_3 < O^{R_x}_4 < O^{W_x}_4$
- $O_7 < O^x_8 < O^y_8$

**Path Constraints**

$R^7_Z = 1$

**Failure Constraints**

$R^8_x + 1! = R^8_y$
A Real Bug – $12 million loss of equipment


**Init:** $x=1, y=2$

**T1**
1: `T2.start()`
2: `z=0`
3: `x++`
4: `y++`
5: `z=1`
6: `T2.join()`

**T2**
7: `if (z==1)`
8: `assert(x+1==y)`

---

**Read-Write Constraints**

$(R_7^z = 0 \land O_7 < O_2) \lor (R_7^z = W_5^z \land O_5 < O_7 \land (O_2 < O_5 \lor O_7 < O_2))$

**Memory Order Constraints**

**SC**

$0_1 < 0_2 < 0_3^{Rx} < 0_3^{Wx} < 0_4^{Rx}$

$< 0_4^{Wx} < 0_5 < 0_6$

$0_7 < 0_8^x < 0_8^y$

**PSO**

execution should be allowed by the memory model

$0_1 < 0_2 < 0_5 < 0_6$

$0_3^{Rx} < 0_3^{Wx} < 0_4^{Rx} < 0_4^{Wx}$

$0_7 < 0_8^x < 0_8^y$

**Path Constraints**

$R_7^z = 1$

**Failure Constraints**

$R_8^{Rx} + 1! = R_8^{Ry}$

match a read to a write

hb

PSO reordering

rf
A Real Bug – $12 million loss of equipment


```
Init: x=1, y=2
T1
1: T2.start()
2: z=0
3: x++
4: y++
5: z=1
6: T2.join()
T2
7: if (z==1)
   8:  x assert(x+1==y)
```

Read-Write Constraints

\[
(R_z^7 = 0 \land O_7 < O_2) \lor (R_z^7 = W_z^5 \land O_5 < O_7 \land (O_2 < O_5 \lor O_7 < O_2))
\]

Memory Order Constraints

**SC**

\[
O_1 < O_2 < O_3^{R_x} < O_3^{W_x} < O_4^{R_x} < O_4^{W_x} < O_5 < O_6
\]

**PSO**

\[
O_3^{R_x} < O_3^{W_x} < O_4^{R_x} < O_4^{W_x} < O_7 < O_8^x < O_8^y
\]

Path Constraints

\[
R_z^7 = 1
\]

Failure Constraints

\[
R_x^7 + 1! = R_y^8
\]

match a read to a write
execution should be allowed by the memory model
make the error happen
A Real Bug – $12 million loss of equipment


Init: x=1, y=2
T1
1: T2.start()
2: z=0
3: x++
4: y++
5: z=1
6: T2.join()
T2
7: if (z==1)
   match a read to a write
   \( R_7^7 = 0 \land O_7 < O_2 \lor \)
   \( R_7^7 = W_7^5 \land O_5 < O_7 \land (O_2 < O_5 \lor O_7 < O_2) \)

Read-Write Constraints

Memory Order Constraints

SC
PSO
execution should be allowed by the memory model

0_1 < O_2 < O_3^{R_x} < O_3^{W_x} < O_4^{A_x}
< O_4^{W_x} < O_5 < O_6
0_7 < O_8^x < O_8^y
0_1 < O_2 < O_5 < O_6
0_3^{R_x} < O_3^{W_x} < O_4^{R_x} < O_4^{W_x}
0_7 < O_8^x < O_8^y

Path Constraints

Solution from the SMT solver:
O_1=1, O_2=2, O_3=3, O_5=4, O_7=5, O_8=6, O_4=7

Schedule: 1-2-3-5-7-8-4

Violation
A Real Bug – $12 million loss of equipment

Init: x=1, y=2

T1
1: T2.start()
2: z=0
3: x++
4: y++
5: z=1
6: T2.join()

T2
7: if (z==1)
8: × assert(x+1==y)

Read-Write Constraints
\[ (R_7^Z = 0 \land O_7 < O_2) \lor (R_7^Z = W_5^Z \land O_5 < O_7 \land (O_2 < O_5 \lor O_7 < O_2)) \]

Memory Order Constraints
SC
\[ O_1 < O_2 < O_3^{R_x} < O_3^{W_x} < O_4^{R_x} < O_4^{W_x} \]
\[ O_7 < O_8^x < O_8^y \]
PSO
\[ O_1 < O_2 < O_5 < O_6 \]
\[ O_3^{R_x} < O_3^{W_x} < O_4^{R_x} < O_4^{W_x} \]
\[ O_7 < O_8^x < O_8^y \]

Path Constraints
\[ R_7^Z = 1 \]

Failure Constraints
\[ R_8^8 + 1! = R_8^y \]

Solution from the SMT solver:
O_1=1, O_2=2, O_3=3, O_5=4, O_7=5, O_8=6, O_4=7

Schedule: 1-2-3-5-7-8-4
A Real Bug – $12 million loss of equipment

Init: x=1, y=2

T1
1: T2.start()
2: z=0
3: x++
4: y++
5: z=1
6: T2.join()

T2
7: if (z==1)
8: x assert(x+1==y)

Read-Write Constraints
\[
(R_z^7 = 0 \land O_7 < O_2) \lor \\
(R_z^7 = W_z^5 \land O_5 < O_7 \land (O_2 < O_5 \lor O_7 < O_2))
\]

Memory Order Constraints

<table>
<thead>
<tr>
<th>SC</th>
<th>PSO</th>
</tr>
</thead>
<tbody>
<tr>
<td>(0_1 &lt; O_2 &lt; O_3^{Rx} &lt; O_3^{Wx} &lt; O_4^{Rx})</td>
<td>(0_1 &lt; O_2 &lt; O_5 &lt; O_6)</td>
</tr>
<tr>
<td>(&lt; O_4^{Wx} &lt; O_5 &lt; O_6)</td>
<td>(O_3^{Rx} &lt; O_3^{Wx} &lt; O_4^{Rx} &lt; O_4^{Wx})</td>
</tr>
<tr>
<td>(O_7 &lt; O_8^x &lt; O_8^y)</td>
<td>(O_7 &lt; O_8^x &lt; O_8^y)</td>
</tr>
</tbody>
</table>

Path Constraints
\(R_z^7 = 1\)

Failure Constraints
\(R_x^8 + 1! = R_y^8\)

Solution from the SMT solver:
\(O_1=1, O_2=2, O_3=3, O_5=4, O_7=5, O_8=6, O_4=7\)

Schedule: 1-2-3-5-7-8-4
## Finding Known Errors

<table>
<thead>
<tr>
<th>Program</th>
<th>LoC</th>
<th>#Threads</th>
<th>#Events</th>
<th>#Executions (Total Time)</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td>ICB</td>
</tr>
<tr>
<td>Example</td>
<td>79</td>
<td>3</td>
<td>32</td>
<td>77322(20s)</td>
</tr>
<tr>
<td>Account</td>
<td>373</td>
<td>5</td>
<td>51</td>
<td>111(0.2s)</td>
</tr>
<tr>
<td>Airline</td>
<td>136</td>
<td>6</td>
<td>67</td>
<td>669(1.8s)</td>
</tr>
<tr>
<td>Allocation</td>
<td>348</td>
<td>3</td>
<td>125</td>
<td>15(0.1s)</td>
</tr>
<tr>
<td>BubbleSort</td>
<td>175</td>
<td>5</td>
<td>133</td>
<td>592(1.2s)</td>
</tr>
<tr>
<td>MTList</td>
<td>5759</td>
<td>27</td>
<td>685</td>
<td>OOM</td>
</tr>
<tr>
<td>MTSet</td>
<td>7086</td>
<td>22</td>
<td>724</td>
<td>OOM</td>
</tr>
<tr>
<td>PingPong</td>
<td>388</td>
<td>6</td>
<td>44</td>
<td>648(3s)</td>
</tr>
<tr>
<td>Pool</td>
<td>10K</td>
<td>3</td>
<td>170</td>
<td>24(0.3s)</td>
</tr>
<tr>
<td>StringBuffer</td>
<td>1339</td>
<td>3</td>
<td>70</td>
<td>12(0.1s)</td>
</tr>
</tbody>
</table>
Finding Known Errors

<table>
<thead>
<tr>
<th>Program</th>
<th>LoC</th>
<th>#Threads</th>
<th>#Events</th>
<th>#Executions (Total Time)</th>
<th>ICB</th>
<th>ICB+DPOR</th>
<th>MCR</th>
</tr>
</thead>
<tbody>
<tr>
<td>Example</td>
<td>79</td>
<td>3</td>
<td>32</td>
<td>77322(20s)</td>
<td>3782(3s)</td>
<td>46(2s)</td>
<td></td>
</tr>
<tr>
<td>Account</td>
<td>373</td>
<td>5</td>
<td>51</td>
<td>111(0.2s)</td>
<td>20(0.2s)</td>
<td>2(0.3s)</td>
<td></td>
</tr>
<tr>
<td>Airline</td>
<td>126</td>
<td>6</td>
<td>67</td>
<td>669(1.8s)</td>
<td>19(0.8s)</td>
<td>2(0.3s)</td>
<td></td>
</tr>
<tr>
<td>Allocation</td>
<td>348</td>
<td>3</td>
<td>125</td>
<td>15(0.1s)</td>
<td>8(0.3s)</td>
<td>2(0.3s)</td>
<td></td>
</tr>
<tr>
<td>BubbleSort</td>
<td>175</td>
<td>5</td>
<td>133</td>
<td>592(1.2s)</td>
<td>400(2.7s)</td>
<td>4(4.8s)</td>
<td></td>
</tr>
<tr>
<td>MTList</td>
<td>5759</td>
<td>27</td>
<td>685</td>
<td>5173(290s)</td>
<td>8(97s)</td>
<td></td>
<td></td>
</tr>
<tr>
<td>MTSet</td>
<td>7086</td>
<td>22</td>
<td>724</td>
<td>5480(267s)</td>
<td>21(159s)</td>
<td></td>
<td></td>
</tr>
<tr>
<td>PingPong</td>
<td>388</td>
<td>6</td>
<td>44</td>
<td>648(3s)</td>
<td>37(0.5s)</td>
<td>2(0.7s)</td>
<td></td>
</tr>
<tr>
<td>Pool</td>
<td>10K</td>
<td>3</td>
<td>170</td>
<td>24(0.3s)</td>
<td>6(0.3s)</td>
<td>3(0.4s)</td>
<td></td>
</tr>
<tr>
<td>StringBuffer</td>
<td>1339</td>
<td>3</td>
<td>70</td>
<td>12(0.1s)</td>
<td>10(0.5s)</td>
<td>2(0.4s)</td>
<td></td>
</tr>
</tbody>
</table>

MCR reduces #runs taken by BMC+POR by orders of magnitude!

MCR takes less time in half of the benchmarks
## State-space Exploration

<table>
<thead>
<tr>
<th>program</th>
<th>Finished</th>
<th>Timeout</th>
<th>OOM</th>
<th>#Executions (Total Time)</th>
<th>#Race</th>
<th>#NPE</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>ICB</td>
<td>ICB+DP</td>
<td>MCR</td>
<td>ICB</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Example</td>
<td>X</td>
<td>✓</td>
<td>✓</td>
<td>3.3M (1h)</td>
<td>7</td>
<td>10</td>
</tr>
<tr>
<td></td>
<td>ICB</td>
<td>ICB+DPO R</td>
<td>MCR</td>
<td>ICB+DPO R</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Account</td>
<td>X</td>
<td>✓</td>
<td>✓</td>
<td>1.5M (1h)</td>
<td>3(0)</td>
<td>3(0)</td>
</tr>
<tr>
<td>Airline</td>
<td>X</td>
<td>✓</td>
<td>✓</td>
<td>326K (1h)</td>
<td>0(0)</td>
<td>0(0)</td>
</tr>
<tr>
<td>Allocation</td>
<td>X</td>
<td>X</td>
<td>✓</td>
<td>1.4M (1h)</td>
<td>0(0)</td>
<td>0(0)</td>
</tr>
<tr>
<td>BubbleSort</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>327K (1h)</td>
<td>4(0)</td>
<td>6(0)</td>
</tr>
<tr>
<td>MTList</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>-</td>
<td>1(0)</td>
<td>1(0)</td>
</tr>
<tr>
<td>MTSet</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>-</td>
<td>5(0)</td>
<td>5(0)</td>
</tr>
<tr>
<td>PingPong</td>
<td>X</td>
<td>X</td>
<td>✓</td>
<td>343K (1h)</td>
<td>6(1)</td>
<td>7(1)</td>
</tr>
<tr>
<td>Pool</td>
<td>X</td>
<td>✓</td>
<td>✓</td>
<td>510K (1h)</td>
<td>0(0)</td>
<td>0(0)</td>
</tr>
<tr>
<td>StringBuffer</td>
<td>X</td>
<td>✓</td>
<td>✓</td>
<td>1.3M (1h)</td>
<td>0(0)</td>
<td>0(0)</td>
</tr>
</tbody>
</table>
## State-space Exploration

For most benchmarks, MCR finished in an hour. For half of the benchmarks, BMC+POR either out of memory or did not finish in an hour.

<table>
<thead>
<tr>
<th>program</th>
<th>Finished</th>
<th>Timeout</th>
<th>OOM</th>
<th>#Executions (Total Time)</th>
<th>#Race</th>
<th>#NPE</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>ICB</td>
<td>ICB+DP</td>
<td>MCR</td>
<td>ICB</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Allocation</td>
<td>X</td>
<td>■</td>
<td>✔</td>
<td>-</td>
<td>1.4M(1h)</td>
<td>30(5.6s)</td>
</tr>
<tr>
<td>BubbleSort</td>
<td>X</td>
<td>■</td>
<td>■</td>
<td>-</td>
<td>327K(1h)</td>
<td>14K(1h)</td>
</tr>
<tr>
<td>MTList</td>
<td>X</td>
<td>X</td>
<td>■</td>
<td>-</td>
<td>382(1h)</td>
<td>1</td>
</tr>
<tr>
<td>MTSet</td>
<td>X</td>
<td>X</td>
<td>■</td>
<td>-</td>
<td>457(1h)</td>
<td>5</td>
</tr>
<tr>
<td>PingPong</td>
<td>■</td>
<td>■</td>
<td>✔</td>
<td>343K(1h)</td>
<td>973K(1h)</td>
<td>413(13s)</td>
</tr>
<tr>
<td>Pool</td>
<td>■</td>
<td>✔</td>
<td>✔</td>
<td>510K(1h)</td>
<td>1.5K(1.9s)</td>
<td>3(0.9s)</td>
</tr>
<tr>
<td>StringBuffer</td>
<td>■</td>
<td>✔</td>
<td>✔</td>
<td>1.3M(1h)</td>
<td>427(0.8s)</td>
<td>3(0.4s)</td>
</tr>
</tbody>
</table>
## State-space Exploration

<table>
<thead>
<tr>
<th>Program</th>
<th>Finished</th>
<th>Timeout</th>
<th>OOM</th>
<th>#Executions (Total Time)</th>
<th>#Race</th>
<th>#NPE</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>ICB</td>
<td>ICB+DP</td>
<td>MAR</td>
<td>ICB</td>
<td>ICB+DPO</td>
<td>MAR</td>
</tr>
<tr>
<td>ICB</td>
<td>✔</td>
<td>✔</td>
<td>✔</td>
<td>3.3M (1h)</td>
<td>26K (10s)</td>
<td>50 (2s)</td>
</tr>
<tr>
<td>ICB+DPO</td>
<td>✔</td>
<td>✔</td>
<td>✔</td>
<td>3.3M (1h)</td>
<td>26K (10s)</td>
<td>50 (2s)</td>
</tr>
<tr>
<td>MCR</td>
<td>✔</td>
<td>✔</td>
<td>✔</td>
<td>3.3M (1h)</td>
<td>26K (10s)</td>
<td>50 (2s)</td>
</tr>
</tbody>
</table>

**For most benchmarks, MCR finished in an hour**

**For half of the benchmarks, BMC+POR either out of memory or did not finish in an hour**

**MCR found 9 more data races and 7 more NPE than BMC+POR**

<table>
<thead>
<tr>
<th>Program</th>
<th>Finished</th>
<th>Timeout</th>
<th>OOM</th>
<th>#Executions (Total Time)</th>
<th>#Race</th>
<th>#NPE</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>ICB</td>
<td>ICB+DP</td>
<td>MAR</td>
<td>ICB</td>
<td>ICB+DPO</td>
<td>MAR</td>
</tr>
<tr>
<td>P1Set</td>
<td>✔</td>
<td>✔</td>
<td>✔</td>
<td>1.4M (1h)</td>
<td>30 (5.6s)</td>
<td>0 (0)</td>
</tr>
<tr>
<td>PingPong</td>
<td>✔</td>
<td>✔</td>
<td>✔</td>
<td>343K (1h)</td>
<td>973K (1h)</td>
<td>413 (13s)</td>
</tr>
<tr>
<td>Pool</td>
<td>✔</td>
<td>✔</td>
<td>✔</td>
<td>343K (1h)</td>
<td>973K (1h)</td>
<td>413 (13s)</td>
</tr>
<tr>
<td>StringBuffer</td>
<td>✔</td>
<td>✔</td>
<td>✔</td>
<td>1.3M (1h)</td>
<td>427 (0.8s)</td>
<td>3 (0.4s)</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Program</th>
<th>Finished</th>
<th>Timeout</th>
<th>OOM</th>
<th>#Executions (Total Time)</th>
<th>#Race</th>
<th>#NPE</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>ICB</td>
<td>ICB+DP</td>
<td>MAR</td>
<td>ICB</td>
<td>ICB+DPO</td>
<td>MAR</td>
</tr>
<tr>
<td>MTSet</td>
<td>✔</td>
<td>✔</td>
<td>✔</td>
<td>382 (1h)</td>
<td>1 (0)</td>
<td>1 (0)</td>
</tr>
<tr>
<td>MTSet</td>
<td>✔</td>
<td>✔</td>
<td>✔</td>
<td>382 (1h)</td>
<td>1 (0)</td>
<td>1 (0)</td>
</tr>
<tr>
<td>PingPong</td>
<td>✔</td>
<td>✔</td>
<td>✔</td>
<td>343K (1h)</td>
<td>973K (1h)</td>
<td>413 (13s)</td>
</tr>
<tr>
<td>Pool</td>
<td>✔</td>
<td>✔</td>
<td>✔</td>
<td>343K (1h)</td>
<td>973K (1h)</td>
<td>413 (13s)</td>
</tr>
<tr>
<td>StringBuffer</td>
<td>✔</td>
<td>✔</td>
<td>✔</td>
<td>1.3M (1h)</td>
<td>427 (0.8s)</td>
<td>3 (0.4s)</td>
</tr>
</tbody>
</table>
# TSO and PSO Results

<table>
<thead>
<tr>
<th>Program</th>
<th>DPOR (rinspect)</th>
<th>MCR</th>
<th>#Executions Reduction</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>SC</td>
<td>TSO</td>
<td>PSO</td>
</tr>
<tr>
<td>Dekker</td>
<td>248</td>
<td>252</td>
<td>508</td>
</tr>
<tr>
<td>Lamport</td>
<td>128</td>
<td>208</td>
<td>2672</td>
</tr>
<tr>
<td>Bakery</td>
<td>350</td>
<td>1164</td>
<td>2040</td>
</tr>
<tr>
<td>Peterson</td>
<td>36</td>
<td>95</td>
<td>120</td>
</tr>
<tr>
<td>StackUnsafe</td>
<td>252</td>
<td>252</td>
<td>252</td>
</tr>
<tr>
<td>RVExample</td>
<td>1959</td>
<td>-</td>
<td>-</td>
</tr>
<tr>
<td>Example (N=1 to 4)</td>
<td>4</td>
<td>4</td>
<td>-</td>
</tr>
<tr>
<td></td>
<td>105</td>
<td>105</td>
<td>-</td>
</tr>
<tr>
<td></td>
<td>4282</td>
<td>4282</td>
<td>-</td>
</tr>
<tr>
<td></td>
<td>14840</td>
<td>14840</td>
<td>-</td>
</tr>
<tr>
<td><strong>Avg.</strong></td>
<td><strong>435</strong></td>
<td><strong>394</strong></td>
<td><strong>1118</strong></td>
</tr>
</tbody>
</table>
# TSO and PSO Results

<table>
<thead>
<tr>
<th>Program</th>
<th>DPOR (rinspect)</th>
<th>MCR</th>
<th>#Executions Reduction</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>SC</td>
<td>TSO</td>
<td>PSO</td>
</tr>
<tr>
<td>Dekker</td>
<td>248</td>
<td>252</td>
<td>508</td>
</tr>
<tr>
<td>Lamport</td>
<td>128</td>
<td>208</td>
<td>2672</td>
</tr>
<tr>
<td>Bakery</td>
<td>350</td>
<td>1164</td>
<td>2040</td>
</tr>
</tbody>
</table>

**MCR explores 5X-10X fewer executions than POR for TSO and PSO memory models**
Maximal Causality Reduction
Parallelization

- MCR is for massive parallelization
- Online exploration with different seed interleavings is parallel
- In each iteration, multiple seed interleavings can be generated in parallel

\[
\text{parfor } r = \text{read}(t, x, v) \in \tau \text{ do}
\]

\[
\text{parfor } w = \text{write}(\_ , x, v') \in \tau \land v' \neq v \text{ do}
\]

\[
\Phi_{\text{seed}}(r, w) \equiv \Phi_{\text{sync}} \land \Phi_{\text{rw}}(r) \land \Phi_{\text{rw}}(w) \land \Phi_{\text{value}}(r, v')
\]
## Results on Real Systems

<table>
<thead>
<tr>
<th>program</th>
<th>#Races</th>
<th>#NPEs</th>
<th>#Runs</th>
<th>MCR</th>
<th>MCR-Parallel</th>
</tr>
</thead>
<tbody>
<tr>
<td>Jigsaw</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>2</td>
<td>1</td>
<td>307 (OOM)</td>
<td>20</td>
<td>38</td>
</tr>
<tr>
<td></td>
<td>1</td>
<td>2</td>
<td>425 (OOM)</td>
<td>6</td>
<td>10</td>
</tr>
<tr>
<td></td>
<td>4</td>
<td>0</td>
<td>1229 (OOM)</td>
<td>6</td>
<td>7</td>
</tr>
<tr>
<td></td>
<td>4</td>
<td>0</td>
<td>1072 (OOM)</td>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>Weblech</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>4</td>
<td>0</td>
<td>1229 (OOM)</td>
<td>185</td>
<td>3311</td>
</tr>
<tr>
<td></td>
<td>4</td>
<td>0</td>
<td>1072 (OOM)</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
### Results on Real Systems

<table>
<thead>
<tr>
<th>Program</th>
<th>ICB</th>
<th>ICB+DP OR</th>
<th>MCR</th>
<th>MCR-Parallel</th>
</tr>
</thead>
<tbody>
<tr>
<td>Jigsaw</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>#Races</td>
<td>2</td>
<td>7</td>
<td>20</td>
<td>38</td>
</tr>
<tr>
<td>#NPEs</td>
<td>1</td>
<td>2</td>
<td>6</td>
<td>1</td>
</tr>
<tr>
<td>#Runs</td>
<td>307 (OOM)</td>
<td>425 (OOM)</td>
<td>32</td>
<td>789</td>
</tr>
<tr>
<td>Weblech</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>#Races</td>
<td>4</td>
<td>4</td>
<td>6</td>
<td>7</td>
</tr>
<tr>
<td>#NPEs</td>
<td>0</td>
<td>0</td>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>#Runs</td>
<td>1229 (OOM)</td>
<td>1072 (OOM)</td>
<td>185</td>
<td>3311</td>
</tr>
</tbody>
</table>

Parallel-MCR explored many more states and detected many more data races and NPEs than MCR.
## Results on Real Systems

<table>
<thead>
<tr>
<th>program</th>
<th>ICB</th>
<th>ICB+DP OR</th>
<th>MCR</th>
<th>MCR-Parallel</th>
</tr>
</thead>
<tbody>
<tr>
<td>Jigsaw</td>
<td>2</td>
<td>7</td>
<td>20</td>
<td>38</td>
</tr>
<tr>
<td>#Races</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>#NPEs</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>#Runs</td>
<td>307</td>
<td>(OOM)</td>
<td>425</td>
<td>(OOM)</td>
</tr>
</tbody>
</table>

**Parallel-MCR** explored many more states and detected many more data races and NPEs than **MCR**!

**Found five new bugs** (i.e., data races and NPEs)!

<table>
<thead>
<tr>
<th>Weblech</th>
<th>ICB</th>
<th>ICB+DP OR</th>
<th>MCR</th>
<th>MCR-Parallel</th>
</tr>
</thead>
<tbody>
<tr>
<td>#Races</td>
<td>4</td>
<td>4</td>
<td>6</td>
<td>7</td>
</tr>
<tr>
<td>#NPEs</td>
<td>0</td>
<td>0</td>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>#Runs</td>
<td>1229 (OOM)</td>
<td>1072 (OOM)</td>
<td>185</td>
<td>3311</td>
</tr>
</tbody>
</table>
References

- **ECOOP’17**: Shiyou Huang and Jeff Huang, "Speeding Up Maximal Causality Reduction with Static Dependency Analysis"
- **OOPSLA’16**: Shiyou Huang and Jeff Huang, "Maximal Causality Reduction for TSO and PSO"
- **PLDI’15**: Jeff Huang, "Stateless Model Checking Concurrent Programs with Maximal Causality Reduction"
- **PLDI’14**: Jeff Huang, Patrick Meredith and Grigore Rosu "Maximal Sound Predictive Race Detection with Control Flow Abstraction"
Takeaway

- A new advance in Model-Checking
  - **Maximal Causality Reduction (MCR)**
  - MCR dramatically improves scalability of BMC and POR
  - **Minimal** state-space exploration and *embarrassingly parallel*

- **MCR open source**
  - [https://github.com/parasol-aser/JMCR](https://github.com/parasol-aser/JMCR)