Concurrency Bugs

Thread 1: Withdraw(x)

1. if balance \( \geq x \) then
2. balance \( \leftarrow \) balance \( - x \)
### Concurrency Bugs

**Thread 1: Withdraw(x)**

1. if balance $\geq x$ then
2. balance $\leftarrow$ balance $-$ x

**Thread 2: Withdraw(x)**

1. if balance $\geq x$ then
2. balance $\leftarrow$ balance $-$ x

No control over scheduling
"Heisenbugs" lie in scheduling subtleties

Formal verification to the rescue

Value-Centric Dynamic Partial Order Reduction
Concurrent Bugs

**Thread 1:** Withdraw($x$)

1. if balance $\geq x$ then
2. balance $\leftarrow$ balance $-$ $x$

**Thread 2:** Withdraw($x$)

1. if balance $\geq x$ then
2. balance $\leftarrow$ balance $-$ $x$

Withdraw(5)

balance = 8

Withdraw(5)
**Concurrency Bugs**

**Thread 1:** Withdraw(x)

1. if balance \( \geq x \) then
2. balance ← balance − x

**Thread 2:** Withdraw(x)

1. if balance \( \geq x \) then
2. balance ← balance − x

Withdraw(5)  
Withdraw(5)

balance = 8
Concurrency Bugs

Thread 1: Withdraw(x)

1 if balance ⩾ x then
2 balance ← balance − x

Thread 2: Withdraw(x)

1 if balance ⩾ x then
2 balance ← balance − x

Withdraw(5) Withdraw(5)

balance = 8
Concurrency Bugs

Thread 1: Withdraw(x)

1 if balance ≥ x then
2 balance ← balance − x

Thread 2: Withdraw(x)

1 if balance ≥ x then
2 balance ← balance − x

Withdraw(5)  Withdraw(5)

balance = 8 → 3
Concurrency Bugs

**Thread 1: Withdraw(x)**
1. if balance $\geq x$ then
2. balance ← balance − x

**Thread 2: Withdraw(x)**
1. if balance $\geq x$ then
2. balance ← balance − x

Withdraw(5) Withdraw(5)

balance = 8 → 3 → −2

Formal verification to the rescue
### Concurrency Bugs

**Thread 1: Withdraw(x)**

1. if balance \( \geq x \) then
2. balance \( \leftarrow \) balance \( - x \)

**Thread 2: Withdraw(x)**

1. if balance \( \geq x \) then
2. balance \( \leftarrow \) balance \( - x \)

Withdraw(5)    Withdraw(5)

balance = 8 \( \rightarrow \) 3 \( \rightarrow \) -2

- No control over scheduling
- “Heisenbugs” lie in scheduling subtleties
- Formal verification to the rescue
Concurrency Setting

- $k$ deterministic threads
  - No randomization
  - Fixed inputs
- All nondeterministic behavior comes from the scheduler
- Goal local-state reachability: catch bugs, e.g. assertion violations
Concurrency Setting

- \( k \) deterministic threads
  - No randomization
  - Fixed inputs

- All nondeterministic behavior comes from the scheduler

- Goal local-state reachability: catch bugs, e.g. assertion violations

- Algorithmic problem: visit all local states of each process

- **Stateless:** cannot remember all system states
Concurrency Setting

- $k$ deterministic threads
  - No randomization
  - Fixed inputs
- All nondeterministic behavior comes from the scheduler
- Goal local-state reachability: catch bugs, e.g. assertion violations

- Algorithmic problem: visit all local states of each process
- **Stateless**: cannot remember all system states

- Examine all traces
- $n!$ many
- We can do better: DPOR
A pair of events \((e_1, e_2)\) is non-commutative if \(e_1\) and \(e_2\) use the same variable, and at least one is a write operation.

\[
\begin{align*}
&\overline{t_1} \\
&w(x) & w(x) \\
&w(y) & w(y) \\
&w(z) & w(y) \\
&w(y) & w(z) \\
&r(z) & r(z) \\
&r(x) & r(x)
\end{align*}
\]
A pair of events \((e_1, e_2)\) is **non-commutative** if

- \(e_1, e_2\) use the same variable, and at last one is a write

\[
\begin{align*}
& t_1 & t_2 \\
& w(x) & w(x) \\
& w(y) & w(y) \\
& w(z) & w(y) \\
& w(y) & w(z) \\
& r(z) & r(z) \\
& r(x) & r(x)
\end{align*}
\]
A pair of events \((e_1, e_2)\) is \textbf{non-commutative} if

- \(e_1, e_2\) use the same variable, and at last one is a write.
Dynamic Partial Order Reduction

Time: $O(\alpha \cdot \beta)$

$\alpha = |T/\sim|$

$\beta =$ amortized time per class

Question: How coarse can I make $\sim$ while keeping $\beta = \text{poly}(n)$?
Dynamic Partial Order Reduction

$T$
Dynamic Partial Order Reduction

\[ T \]

Time: \( O(\alpha \cdot \beta) \)

\[ \alpha = \frac{|T/\sim|}{\beta} \]

Question: How coarse can I make \( \sim \) while keeping \( \beta = \text{poly}(n) \)?
Dynamic Partial Order Reduction

\[ T \]

\[ \alpha = |T/\sim| \]

\[ \beta = \text{amortized time per class} \]

Question: How coarse can I make \( \sim \) while keeping \( \beta = \text{poly}(n) \)?
Dynamic Partial Order Reduction

Time: $O(\alpha \cdot \beta)$

- $\alpha = |T/\sim|$
- $\beta = \text{amortized time per class}$
Dynamic Partial Order Reduction

Time: $O(\alpha \cdot \beta)$
- $\alpha = |\mathcal{T}/\sim|$ 
- $\beta =$ amortized time per class

Question:
- How coarse can I make $\sim$
- while keeping $\beta = \text{poly}(n)$?
Motivation

Thread $p_1$:
1. $w(x, 1)$

Thread $p_2$:
1. $w(x, 1)$
2. $r(x)$
Motivation

<table>
<thead>
<tr>
<th>Happens-Before</th>
<th>( t_1 )</th>
<th>( t_2 )</th>
<th>( t_3 )</th>
</tr>
</thead>
<tbody>
<tr>
<td>( w )</td>
<td>( w )</td>
<td>( w )</td>
<td></td>
</tr>
<tr>
<td>( w )</td>
<td>( w )</td>
<td>( r )</td>
<td></td>
</tr>
<tr>
<td>( r )</td>
<td>( r )</td>
<td>( w )</td>
<td></td>
</tr>
</tbody>
</table>
Motivation

Thread $p_1$:
1. $w(x, 1)$

Thread $p_2$:
1. $w(x, 1)$
2. $r(x)$

<table>
<thead>
<tr>
<th>Happens-Before</th>
<th>Data-centric [Chalupa et al ’18]</th>
</tr>
</thead>
<tbody>
<tr>
<td>$t_1$</td>
<td>$t_2$</td>
</tr>
<tr>
<td>$w$</td>
<td>$w$</td>
</tr>
<tr>
<td>$w$</td>
<td>$w$</td>
</tr>
<tr>
<td>$r$</td>
<td>$r$</td>
</tr>
</tbody>
</table>
## Motivation

Thread $p_1$:
1. $w(x, 1)$

Thread $p_2$:
1. $w(x, 1)$
2. $r(x)$

### Happens-Before

<table>
<thead>
<tr>
<th></th>
<th>$t_1$</th>
<th>$t_2$</th>
<th>$t_3$</th>
</tr>
</thead>
<tbody>
<tr>
<td>Initial</td>
<td>$w$</td>
<td>$w$</td>
<td>$w$</td>
</tr>
<tr>
<td>$w$</td>
<td>$w$</td>
<td>$w$</td>
<td>$r$</td>
</tr>
<tr>
<td>$r$</td>
<td>$r$</td>
<td>$w$</td>
<td>$w$</td>
</tr>
</tbody>
</table>

### Data-centric [Chalupa et al '18]

<table>
<thead>
<tr>
<th></th>
<th>$t_1$</th>
<th>$t_2$</th>
</tr>
</thead>
<tbody>
<tr>
<td>Initial</td>
<td>$w$</td>
<td>$w$</td>
</tr>
<tr>
<td>$w$</td>
<td>$w$</td>
<td>$w$</td>
</tr>
<tr>
<td>$r$</td>
<td>$r$</td>
<td>$r$</td>
</tr>
</tbody>
</table>

### This work

<table>
<thead>
<tr>
<th></th>
<th>$t_1$</th>
</tr>
</thead>
<tbody>
<tr>
<td>Initial</td>
<td>$w$</td>
</tr>
<tr>
<td>$w$</td>
<td>$w$</td>
</tr>
<tr>
<td>$r$</td>
<td>$r$</td>
</tr>
</tbody>
</table>
Realizability of Abstraction

Given an abstract object $A$, decide whether $[A] \neq \emptyset$. 

\[
[A] = \begin{bmatrix}
t_1 & t_2 & t_3 \\
w & w & w \\
w & w & r \\
r & r & w \\
\end{bmatrix}
\]
\( A = (X_1, X_2, P, \text{GoodW}) \)

- \( X_1 \) has all events of \( p_1 \)
- \( P|X_2 \) is a Happens-Before
- GoodW is the *good writes* function
  - \( r \mapsto \{w_1, \ldots w_j\} \)
\( \mathcal{A} = (X_1, X_2, P, \text{GoodW}) \)

- \( X_1 \) has all events of \( p_1 \)
- \( P|X_2 \) is a Happens-Before
- GoodW is the \textit{good writes} function
  
  \( r \mapsto \{w_1, \ldots w_j\} \)

**Realizability of \( \mathcal{A} \)**

Find if \( P \) can be linearized to a trace \( t \) such that every read sees a good write.
Definition (Closed Annotated Partial Orders)

Call $\mathcal{A} = (X_1, X_2, P, \text{GoodW})$ **closed** if for every read $r$

1. There is a good write $w < P_r$.
2. There is a good maximal write for $r$.
3. For every minimal bad write $w' < P_r$ there exists a good write $w$ with $w' < P_w$. 

$\mathcal{A} = (X_1, X_2, P, \text{GoodW})$ 

Value-Centric Dynamic Partial Order Reduction
Definition (Closed Annotated Partial Orders)

Call $\mathcal{A} = (X_1, X_2, P, \text{GoodW})$ closed if for every read $r$

1. There is a good write $w <_P r$. 

![Diagram of closed order]

$w \quad w' \quad r$

$w \quad w' \quad r$
Definition (Closed Annotated Partial Orders)

Call \( \mathcal{A} = (X_1, X_2, P, \text{GoodW}) \) closed if for every read \( r \)

1. There is a good write \( w < P r \).
2. There is a good maximal write for \( r \).
Definition (Closed Annotated Partial Orders)

Call $A = (X_1, X_2, P, \text{GoodW})$ closed if for every read $r$

1. There is a good write $w <_P r$.
2. There is a good maximal write for $r$.
3. For every minimal bad write $w' <_P r$ there exists a good write $w$ with $w' <_P w$. 

\[ \begin{array}{cccccc} 
 w & \downarrow & w' & \downarrow & w' & \downarrow \\
 w & \downarrow & r & \downarrow & w & \downarrow \\
 w & \downarrow & r & \downarrow & w & \downarrow \\
 w & \downarrow & r & \downarrow & w & \downarrow \\
 \end{array} \]
Take annotated partial order $\mathcal{A} = (X_1, X_2, P, \text{GoodW})$

Lemma

If $\mathcal{A}$ is closed then it is realizable.

• What if $\mathcal{A}$ is not closed?
Take annotated partial order $\mathcal{A} = (X_1, X_2, P, \text{GoodW})$

**Lemma**

*If $\mathcal{A}$ is closed then it is realizable.*

- What if $\mathcal{A}$ is not closed?

**Lemma**

*Either $\mathcal{A}$ is not realizable, or there is a unique minimal strengthening $Q$ of $P$ such that*

- $\mathcal{B} = (X_1, X_2, Q, \text{GoodW})$ *is closed*
- *Any witness for $\mathcal{B}$ is a witness for $\mathcal{A}$*

**Definition (Closure)**

Call $\mathcal{B}$ the closure of $\mathcal{A}$ (if it exists).
Lemma (1)

An annotated partial order is realizable iff it has a closure.
Lemma (1)

An annotated partial order is realizable iff it has a closure.

Lemma (2)

We can compute the closure of annotated partial orders in $O(\text{poly}(n))$ time.

Lemma 1 + Lemma 2 $\implies$ Realizability!
Relax Happens-Before $\mathcal{HB}$ $\iff$ Value-Happens Before $\mathcal{VHB}$
Relax Happens-Before $\mathcal{HB} \quad \mapsto \quad$ Value-Happens Before $\mathcal{VHB}$

\textbf{Theorem}

$\sim_{\mathcal{VHB}}$ induces a partitioning that is at least as coarse as

- the Happens-Before Partitioning [Abdulla et al ’14]
- the Data-centric Partitioning [Chalupa et al ’18]

and can be exponentially coarser (value-based).
Relax Happens-Before $\mathcal{HB}$ $\mapsto$ Value-Happens Before $\mathcal{VHB}$

**Theorem**

$\sim_{\mathcal{VHB}}$ induces a partitioning that is at least as coarse as
- the Happens-Before Partitioning [Abdulla et al ’14]
- the Data-centric Partitioning [Chalupa et al ’18]

and can be exponentially coarser (value-based).

**Theorem**

VC-DPOR explores all local states and runs in time $O(\alpha \cdot \beta)$, where
- $\alpha = |\mathcal{T} / \sim_{\mathcal{VHB}}|$
- $\beta = \text{poly}(n)$, where $n =$ length of the longest trace in $\mathcal{T}$. 
Implemented VC-DPOR
Based on Nidhugg for LLVM IR

How to evaluate coarseness?

- **Source, Optimal** P. Abdulla et al. “Optimal Dynamic Partial Order Reduction”. In: POPL. 2014
- **Optimal** S. Aronis et al. “Optimal Dynamic Partial Order Reduction with Observers”. In: TACAS. 2018
Controlled Value Reduction

**Figure:** # Partitioning
### Experiments 1: Mutual Exclusion

**Benchmark**

<table>
<thead>
<tr>
<th>Benchmark</th>
<th># Partitioning</th>
<th>VC-DPOR</th>
<th>Source</th>
<th>Optimal</th>
<th>Optimal*</th>
<th>DC-DPOR</th>
</tr>
</thead>
<tbody>
<tr>
<td>tsay(2)</td>
<td>2488</td>
<td>7469</td>
<td>7469</td>
<td>7469</td>
<td>7469</td>
<td></td>
</tr>
<tr>
<td>tsay(3)</td>
<td>241822</td>
<td>1414576</td>
<td>1414576</td>
<td>1414576</td>
<td>1414576</td>
<td></td>
</tr>
<tr>
<td>tsay(4)</td>
<td>24609389</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
</tr>
<tr>
<td>pet_fis(2)</td>
<td>1371</td>
<td>4386</td>
<td>4386</td>
<td>4386</td>
<td>4386</td>
<td></td>
</tr>
<tr>
<td>pet_fis(3)</td>
<td>70448</td>
<td>430004</td>
<td>430004</td>
<td>430004</td>
<td>430004</td>
<td></td>
</tr>
<tr>
<td>pet_fis(4)</td>
<td>3747718</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
</tr>
<tr>
<td>burns(1)</td>
<td>67</td>
<td>849</td>
<td>849</td>
<td>849</td>
<td>849</td>
<td></td>
</tr>
<tr>
<td>burns(2)</td>
<td>11297</td>
<td>1490331</td>
<td>1490331</td>
<td>1490331</td>
<td>1490331</td>
<td></td>
</tr>
<tr>
<td>burns(3)</td>
<td>1638338</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
</tr>
</tbody>
</table>

**Time**

<table>
<thead>
<tr>
<th>Benchmark</th>
<th>Time</th>
<th>VC-DPOR</th>
<th>Source</th>
<th>Optimal</th>
<th>Optimal*</th>
<th>DC-DPOR</th>
</tr>
</thead>
<tbody>
<tr>
<td>tsay(2)</td>
<td>0.81s</td>
<td>2.46s</td>
<td>2.76s</td>
<td>2.99s</td>
<td>1.82s</td>
<td></td>
</tr>
<tr>
<td>tsay(3)</td>
<td>1m38s</td>
<td>10m2s</td>
<td>10m54s</td>
<td>12m1s</td>
<td>7m42s</td>
<td></td>
</tr>
<tr>
<td>tsay(4)</td>
<td>3h51m</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td></td>
</tr>
<tr>
<td>pet_fis(2)</td>
<td>0.69s</td>
<td>1.56s</td>
<td>1.61s</td>
<td>1.73s</td>
<td>1.16s</td>
<td></td>
</tr>
<tr>
<td>pet_fis(3)</td>
<td>34.03s</td>
<td>2m54s</td>
<td>3m10s</td>
<td>3m31s</td>
<td>2m20s</td>
<td></td>
</tr>
<tr>
<td>pet_fis(4)</td>
<td>41m31s</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td></td>
</tr>
<tr>
<td>burns(1)</td>
<td>0.09s</td>
<td>0.45s</td>
<td>0.40s</td>
<td>0.44s</td>
<td>0.49s</td>
<td></td>
</tr>
<tr>
<td>burns(2)</td>
<td>16.27s</td>
<td>16m49s</td>
<td>17m32s</td>
<td>20m4s</td>
<td>26m4s</td>
<td></td>
</tr>
<tr>
<td>burns(3)</td>
<td>1h0m</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td></td>
</tr>
</tbody>
</table>
### Experiments 2: SV-COMP

#### Benchmark # Partitioning

<table>
<thead>
<tr>
<th>Benchmark</th>
<th>VC-DPOR</th>
<th>Source</th>
<th>Optimal</th>
<th>Optimal*</th>
<th>DC-DPOR</th>
</tr>
</thead>
<tbody>
<tr>
<td>parker(6)</td>
<td>38670</td>
<td>1100917</td>
<td>1100917</td>
<td>1023567</td>
<td>985807</td>
</tr>
<tr>
<td>parker(7)</td>
<td>52465</td>
<td>1735432</td>
<td>1735432</td>
<td>1613807</td>
<td>1554237</td>
</tr>
<tr>
<td>parker(8)</td>
<td>68360</td>
<td>2576147</td>
<td>2576147</td>
<td>2395947</td>
<td>2307467</td>
</tr>
<tr>
<td>Boop(6)</td>
<td>248212</td>
<td>35079696</td>
<td>35079696</td>
<td>4750426</td>
<td>1468774</td>
</tr>
<tr>
<td>Boop(7)</td>
<td>420033</td>
<td>-</td>
<td>-</td>
<td>10134616</td>
<td>2874202</td>
</tr>
<tr>
<td>Boop(8)</td>
<td>677870</td>
<td>-</td>
<td>-</td>
<td>20003512</td>
<td>5268064</td>
</tr>
<tr>
<td>scull_true(3)</td>
<td>3426</td>
<td>617706</td>
<td>617706</td>
<td>436413</td>
<td>172931</td>
</tr>
<tr>
<td>scull_true(4)</td>
<td>8990</td>
<td>2732933</td>
<td>2732933</td>
<td>1840022</td>
<td>656100</td>
</tr>
<tr>
<td>scull_true(5)</td>
<td>19881</td>
<td>9488043</td>
<td>9488043</td>
<td>6070688</td>
<td>1988798</td>
</tr>
</tbody>
</table>

#### Benchmark Time

<table>
<thead>
<tr>
<th>Benchmark</th>
<th>VC-DPOR</th>
<th>Source</th>
<th>Optimal</th>
<th>Optimal*</th>
<th>DC-DPOR</th>
</tr>
</thead>
<tbody>
<tr>
<td>parker(6)</td>
<td>1m29s</td>
<td>23m5s</td>
<td>24m29s</td>
<td>24m54s</td>
<td>46m41s</td>
</tr>
<tr>
<td>parker(7)</td>
<td>2m23s</td>
<td>41m28s</td>
<td>44m41s</td>
<td>45m13s</td>
<td>1h27m</td>
</tr>
<tr>
<td>parker(8)</td>
<td>3m35s</td>
<td>1h9m</td>
<td>1h15m</td>
<td>1h17m</td>
<td>2h29m</td>
</tr>
<tr>
<td>Boop(6)</td>
<td>3m26s</td>
<td>2h54m</td>
<td>2h49m</td>
<td>26m22s</td>
<td>12m33s</td>
</tr>
<tr>
<td>Boop(7)</td>
<td>6m33s</td>
<td>-</td>
<td>-</td>
<td>1h0m</td>
<td>27m21s</td>
</tr>
<tr>
<td>Boop(8)</td>
<td>11m54s</td>
<td>-</td>
<td>-</td>
<td>2h7m</td>
<td>56m13s</td>
</tr>
<tr>
<td>scull_true(3)</td>
<td>19.77s</td>
<td>9m46s</td>
<td>10m22s</td>
<td>9m7s</td>
<td>4m46s</td>
</tr>
<tr>
<td>scull_true(4)</td>
<td>1m7s</td>
<td>51m37s</td>
<td>54m33s</td>
<td>46m12s</td>
<td>25m56s</td>
</tr>
<tr>
<td>scull_true(5)</td>
<td>3m8s</td>
<td>3h29m</td>
<td>3h42m</td>
<td>2h54m</td>
<td>1h47m</td>
</tr>
</tbody>
</table>
## Experiments 3: Dynamic Programming

### Benchmark: # Partitioning

<table>
<thead>
<tr>
<th>Benchmark</th>
<th>VC-DPOR</th>
<th>Source</th>
<th>Optimal</th>
<th>Optimal*</th>
<th>DC-DPOR</th>
</tr>
</thead>
<tbody>
<tr>
<td>rod_cut(7)</td>
<td>4324</td>
<td>102128</td>
<td>102128</td>
<td>51974</td>
<td>23143</td>
</tr>
<tr>
<td>rod_cut(8)</td>
<td>14744</td>
<td>508646</td>
<td>508646</td>
<td>257707</td>
<td>114624</td>
</tr>
<tr>
<td>rod_cut(9)</td>
<td>50320</td>
<td>2574752</td>
<td>-</td>
<td>1300067</td>
<td>577682</td>
</tr>
<tr>
<td>coin_min(8)</td>
<td>46535</td>
<td>1902262</td>
<td>1902262</td>
<td>981936</td>
<td>382275</td>
</tr>
<tr>
<td>coin_min(9)</td>
<td>154663</td>
<td>-</td>
<td>-</td>
<td>1634899</td>
<td></td>
</tr>
<tr>
<td>coin_min(11)</td>
<td>1312252</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td></td>
</tr>
<tr>
<td>bin_nocon(7)</td>
<td>13202</td>
<td>1664672</td>
<td>1664672</td>
<td>471151</td>
<td>121350</td>
</tr>
<tr>
<td>bin_nocon(8)</td>
<td>44802</td>
<td>-</td>
<td>-</td>
<td>2825725</td>
<td>603668</td>
</tr>
<tr>
<td>bin_nocon(11)</td>
<td>922114</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td></td>
</tr>
</tbody>
</table>

### Benchmark: Time

<table>
<thead>
<tr>
<th>Benchmark</th>
<th>VC-DPOR</th>
<th>Source</th>
<th>Optimal</th>
<th>Optimal*</th>
<th>DC-DPOR</th>
</tr>
</thead>
<tbody>
<tr>
<td>rod_cut(7)</td>
<td>33.23s</td>
<td>4m14s</td>
<td>7m43s</td>
<td>3m47s</td>
<td>1m28s</td>
</tr>
<tr>
<td>rod_cut(8)</td>
<td>3m4s</td>
<td>27m32s</td>
<td>57m42s</td>
<td>28m2s</td>
<td>12m9s</td>
</tr>
<tr>
<td>rod_cut(9)</td>
<td>17m24s</td>
<td>3h0m</td>
<td>-</td>
<td>3h27m</td>
<td>1h39m</td>
</tr>
<tr>
<td>coin_min(8)</td>
<td>3m0s</td>
<td>1h13m</td>
<td>2h12m</td>
<td>1h12m</td>
<td>14m0s</td>
</tr>
<tr>
<td>coin_min(9)</td>
<td>11m36s</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>1h8m</td>
</tr>
<tr>
<td>coin_min(11)</td>
<td>2h4m</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
</tr>
<tr>
<td>bin_nocon(7)</td>
<td>29.57s</td>
<td>48m34s</td>
<td>1h26m</td>
<td>26m11s</td>
<td>2m4s</td>
</tr>
<tr>
<td>bin_nocon(8)</td>
<td>1m54s</td>
<td>-</td>
<td>3h17m</td>
<td>12m32s</td>
<td>-</td>
</tr>
<tr>
<td>bin_nocon(11)</td>
<td>1h0m</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
</tr>
</tbody>
</table>
### Partitioning

<table>
<thead>
<tr>
<th>Benchmark</th>
<th>VC-DPOR</th>
<th>Source</th>
<th>Optimal</th>
<th>Optimal*</th>
<th>DC-DPOR</th>
</tr>
</thead>
<tbody>
<tr>
<td>X2Tv9(3)</td>
<td>7234</td>
<td>7304</td>
<td>7304</td>
<td>7304</td>
<td>7304</td>
</tr>
<tr>
<td>X2Tv9(4)</td>
<td>150535</td>
<td>153725</td>
<td>153725</td>
<td>153725</td>
<td>153725</td>
</tr>
<tr>
<td>X2Tv9(5)</td>
<td>3261067</td>
<td>332491</td>
<td>332491</td>
<td>332491</td>
<td>332491</td>
</tr>
<tr>
<td>pthread5(1)</td>
<td>20</td>
<td>24</td>
<td>24</td>
<td>24</td>
<td>20</td>
</tr>
<tr>
<td>pthread5(2)</td>
<td>1470</td>
<td>1890</td>
<td>1890</td>
<td>1806</td>
<td>1470</td>
</tr>
<tr>
<td>pthread5(3)</td>
<td>226800</td>
<td>302400</td>
<td>302400</td>
<td>280800</td>
<td>226800</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Benchmark</th>
<th>VC-DPOR</th>
<th>Source</th>
<th>Optimal</th>
<th>Optimal*</th>
<th>DC-DPOR</th>
</tr>
</thead>
<tbody>
<tr>
<td>X2Tv9(3)</td>
<td>2.53s</td>
<td>2.11s</td>
<td>2.23s</td>
<td>2.49s</td>
<td>2.41s</td>
</tr>
<tr>
<td>X2Tv9(4)</td>
<td>1m3s</td>
<td>52.80s</td>
<td>56.85s</td>
<td>1m3s</td>
<td>56.86s</td>
</tr>
<tr>
<td>X2Tv9(5)</td>
<td>29m53s</td>
<td>22m17s</td>
<td>24m10s</td>
<td>27m11s</td>
<td>27m10s</td>
</tr>
<tr>
<td>pthread5(1)</td>
<td>0.05s</td>
<td>0.04s</td>
<td>0.04s</td>
<td>0.06s</td>
<td>0.06s</td>
</tr>
<tr>
<td>pthread5(2)</td>
<td>0.67s</td>
<td>0.38s</td>
<td>0.45s</td>
<td>0.54s</td>
<td>0.67s</td>
</tr>
<tr>
<td>pthread5(3)</td>
<td>2m30s</td>
<td>1m14s</td>
<td>1m17s</td>
<td>1m17s</td>
<td>2m21s</td>
</tr>
</tbody>
</table>
Conclusion

- Stateless bounded model checking of concurrent programs
- New algorithm VC-DPOR
- A combination of Value-based + Happens-Before
- Efficient (poly-time) amortized exploration time
- Practical speedups
Thank you!
Questions?
Definition (CHB)
The Causally-Happens-Before partial order of a trace $t$ is the weakest partial order $\rightarrow_t$ s.t.
\begin{itemize}
  \item $\rightarrow_t \subseteq \text{TO}$
  \item $\text{RF}_t(r) \xrightarrow{t} r$
\end{itemize}

Definition (Side Function)
The side function $S_t$ of a trace $t$ is defined over the reads of the root thread, s.t.
\[
S_t(r) = \begin{cases} 
1, & \text{if } \text{RF}_t(r) \text{ is local to } r \\ 
2, & \text{otherwise}
\end{cases}
\]
Definition (VHB)

We have $t_1 \sim_{VHB} t_2$ if

1. $\text{Events}(t_1) = \text{Events}(t_2)$, $\text{value}_{t_1} = \text{value}_{t_2}$ and $S_{t_1} = S_{t_2}$
2. $\rightarrow_{t_1} | \text{Reads} = \rightarrow_{t_2} | \text{Reads}$
3. $\rightarrow_{t_1} | \mathcal{E} \neq p_1 = \rightarrow_{t_2} | \mathcal{E} \neq p_1$ threads.