Optimal Prediction of Synchronization-Preserving Races

Umang Mathur, Andreas Pavlogiannis and Mahesh Viswanathan
Concurrency: Software and Challenges

Ubiquitous computing paradigm

- Back-bone of big-data, AI revolutions

Challenging to write multi-threaded programs

- Large interleaving space
- Concurrency bugs
  - data races, deadlocks, etc.,
  - manifest in production despite rigorous testing
  - hard to reproduce
  - severe outcomes - loss of lives and money
Dynamic Analysis for Detecting Concurrency Bugs

Program Execution

Bug Found
Not Found
Witness \( \sigma^* \)

- Executes a prefix of each thread
- Every read sees the same write as in \( \sigma \)
Data Races

Observed trace $\sigma$

Witness trace $\sigma^*$

reordering

<table>
<thead>
<tr>
<th></th>
<th>$t_1$</th>
<th>$t_2$</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>acq($\ell$)</td>
<td></td>
</tr>
<tr>
<td>2</td>
<td>$w(x)$</td>
<td></td>
</tr>
<tr>
<td>3</td>
<td>rel($\ell$)</td>
<td></td>
</tr>
<tr>
<td>4</td>
<td></td>
<td>acq($\ell$)</td>
</tr>
<tr>
<td>5</td>
<td></td>
<td>$w(x)$</td>
</tr>
<tr>
<td>6</td>
<td></td>
<td>$r(x)$</td>
</tr>
<tr>
<td>7</td>
<td></td>
<td>rel($\ell$)</td>
</tr>
<tr>
<td>8</td>
<td></td>
<td>$r(x)$</td>
</tr>
</tbody>
</table>
Data Races

Witness \( \sigma^* \)

- Executes a prefix of each thread
- Every read sees the same write as in \( \sigma \)

Observed trace \( \sigma \)

Witness trace \( \sigma^* \)
Happens-Before Races

- HB not complete for this principle
- Here a race is missed
- Exposed without reordering critical sections
Happens-Before Races

Happens-before principle
Do not reorder conflicting critical sections
Happens-Before Races

Happens-before principle

Do not reorder conflicting critical sections

But!

- HB not complete for this principle
- Here a race is missed
- Exposed without reordering critical sections
Happens-Before Races

Happens-before principle
Do not reorder conflicting critical sections

But!

- HB not complete for this principle
- Here a race is missed
- Exposed without reordering critical sections
## Sync-Preserving Witness Reorderings

### Definition (Sync-preserving witnesses)

A **sync-preserving witness** of trace $\sigma$ is a witness $\sigma^*$ such that for every two acquires $\text{acq}_1(\ell), \text{acq}_2(\ell) \in \sigma^*$ we have

$$\text{acq}_1(\ell) <^{\sigma^*} \text{acq}_2(\ell) \implies \text{acq}_1(\ell) <^{\sigma} \text{acq}_2(\ell)$$
Sync-Preserving Witness Reorderings

**Definition (Sync-preserving witnesses)**

A sync-preserving witness of trace $\sigma$ is a witness $\sigma^*$ such that for every two acquires $\text{acq}_1(\ell), \text{acq}_2(\ell) \in \sigma^*$ we have

$$\text{acq}_1(\ell) <^{\sigma^*} \text{acq}_2(\ell) \implies \text{acq}_1(\ell) <^\sigma \text{acq}_2(\ell)$$

<table>
<thead>
<tr>
<th></th>
<th>$t_1$</th>
<th>$t_2$</th>
<th>$t_3$</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>$\text{acq}(\ell_1)$</td>
<td></td>
<td></td>
</tr>
<tr>
<td>2</td>
<td>$\text{rel}(\ell_1)$</td>
<td></td>
<td></td>
</tr>
<tr>
<td>3</td>
<td></td>
<td>$\text{acq}(\ell_2)$</td>
<td></td>
</tr>
<tr>
<td>4</td>
<td></td>
<td>$\text{rel}(\ell_2)$</td>
<td></td>
</tr>
<tr>
<td>5</td>
<td></td>
<td>$\text{acq}(\ell_1)$</td>
<td></td>
</tr>
<tr>
<td>6</td>
<td></td>
<td>$\text{rel}(\ell_1)$</td>
<td></td>
</tr>
<tr>
<td>7</td>
<td></td>
<td></td>
<td>$\text{acq}(\ell_2)$</td>
</tr>
<tr>
<td>8</td>
<td></td>
<td></td>
<td>$\text{rel}(\ell_2)$</td>
</tr>
</tbody>
</table>

**Observed trace**

<table>
<thead>
<tr>
<th></th>
<th>$t_1$</th>
<th>$t_2$</th>
<th>$t_3$</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>2</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>3</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>4</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>5</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>6</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>7</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>8</td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

**sync-preserving witness**

<table>
<thead>
<tr>
<th></th>
<th>$t_1$</th>
<th>$t_2$</th>
<th>$t_3$</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>$\text{acq}(\ell_2)$</td>
<td></td>
<td></td>
</tr>
<tr>
<td>2</td>
<td>$\text{rel}(\ell_2)$</td>
<td></td>
<td></td>
</tr>
<tr>
<td>3</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>4</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>5</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>6</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>7</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>8</td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
Sync-Preserving Data Races

**Definition (Sync-preserving data races)**

A predictable data race \((e_1, e_2)\) of \(\sigma\) is *sync-preserving* if it has a sync-preserving witness
Sync-Preserving Data Races

Definition (Sync-preserving data races)
A predictable data race \((e_1, e_2)\) of \(\sigma\) is sync-preserving if it has a sync-preserving witness.
**Sync-Preserving Data Races**

**Definition (Sync-preserving data races)**

A predictable data race \((e_1, e_2)\) of \(\sigma\) is **sync-preserving** if it has a sync-preserving witness.
How do sync-preserving races compare to HB races?
Sync-Preserving vs HB races

Theorem

Every HB race is also sync-preserving.
Sync-Preserving vs HB races

**Theorem**

*Every HB race is also sync-preserving.*

The opposite is not true

```
<table>
<thead>
<tr>
<th></th>
<th>t₁</th>
<th>t₂</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>w(x)</td>
<td></td>
</tr>
<tr>
<td>2</td>
<td>acq(l)</td>
<td></td>
</tr>
<tr>
<td>3</td>
<td>rel(l)</td>
<td></td>
</tr>
<tr>
<td>4</td>
<td>rel(l)</td>
<td></td>
</tr>
<tr>
<td>5</td>
<td>w(x)</td>
<td></td>
</tr>
<tr>
<td>6</td>
<td>acq(l)</td>
<td></td>
</tr>
</tbody>
</table>
```
Sync-Preserving vs HB races

**Theorem**

*Every HB race is also sync-preserving.*

The opposite is not true

Sync-preserving races need not be consecutive
Sync-Preserving vs HB races

**Theorem**

*Every HB race is also sync-preserving.*

The opposite is not true

Sync-preserving races need not be consecutive

1. \( w(x) \)
2. \( \text{acq}(\ell) \)
3. \( \text{rel}(\ell) \)
4. \( \text{acq}(\ell) \)
5. \( \text{rel}(\ell) \)
6. \( w(x) \)

1. \( w(x) \)
2. \( \text{acq}(\ell) \)
3. \( w(x) \)
4. \( \text{rel}(\ell) \)
5. \( \text{acq}(\ell) \)
6. \( w(x) \)
7. \( \text{rel}(\ell) \)

Sync-preservation is a notion broader than HB
How do sync-preserving races compare to other notions of *predictive* races?
Sync-preserving vs Predictive partial-orders

Sync-preserving vs WCP

Other orders — DC, SDP and WDP are unsound

refer to paper for counter-example to soundness of SDP
Sync-preserving vs Predictive partial-orders

Sync-preserving vs WCP

Sync-preserving ✓
WCP ✗

$\begin{array}{c|c|c}
\text{t}_1 & \text{t}_2 \\
1 & w(x) \\
2 & \text{acq}(\ell) \\
3 & w(y) \\
4 & \text{rel}(\ell) \\
5 & \text{acq}(\ell) \\
6 & w(y) \\
7 & \text{rel}(\ell) \\
8 & w(x) \\
\end{array}$

Other orders — DC, SDP and WDP are unsound

Refer to paper for counter-example to soundness of SDP
Sync-preserving vs WCP

Sync-preserving vs Predictive partial-orders

Other orders — DC, SDP and WDP are unsound.

Refer to paper for counter-example to soundness of SDP.
Sync-preserving vs Predictive partial-orders

Sync-preserving vs WCP

Sync-preserving
WCP

Sync-preserving
WCP

Other orders — DC, SDP\(^1\) and WDP are unsound

\(^1\)Refer to paper for counter-example to soundness of SDP
How fast can we predict sync-preserving races?
SyncP – An Algorithm For Sync-Preserving Races

Theorem

SyncP is a sound and complete algorithm for sync-preserving races. Given a trace $\sigma$ of length $N$, SyncP uses $\tilde{O}(N)$ time and space.
SyncP – An Algorithm For Sync-Preserving Races

Theorem
SyncP is a sound and complete algorithm for sync-preserving races. Given a trace $\sigma$ of length $N$, SyncP uses $\tilde{O}(N)$ time and space.

Soundness + Completeness + Efficiency
Sync-Preserving Ideals (SPIdeals)

How to decide if \((e_1, e_2)\) is a sync-preserving race?

\[
\text{Compute the sync-preserving ideal } SPIdeal(e_1, e_2)
\]

\[
t_1 \quad t_2 \quad t_3
\]

\[
1 \quad \text{acq}(\ell_1)
2 \quad \text{w}(x)
3 \quad \text{r}(x)
4 \quad e_1
5 \quad \text{w}(y)
6 \quad \text{r}(y)
7 \quad \text{rel}(\ell_1)
8 \quad \text{acq}(\ell_1)
9 \quad e_2
10 \quad \text{rel}(\ell_1)
11
12
13

Sync-preserving race!

No sync-preserving race!

Theorem \((e_1, e_2)\) is a sync-preserving race iff

\[
\text{SPIdeal}(e_1, e_2) \cap \{e_1, e_2\} = \emptyset
\]
Sync-Preserving Ideals (SPIdeals)

How to decide if \((e_1, e_2)\) is a sync-preserving race?
Compute the **sync-preserving ideal** \(\text{SPIdeal}(e_1, e_2)\)
Sync-Preserving Ideals (SPIdeals)

How to decide if \((e_1, e_2)\) is a sync-preserving race?

Compute the sync-preserving ideal \(\text{SPIdeal}(e_1, e_2)\)
Sync-Preserving Ideals (SPIdeals)

How to decide if \((e_1, e_2)\) is a sync-preserving race?

Compute the sync-preserving ideal \(\text{SPIdeal}(e_1, e_2)\)

<table>
<thead>
<tr>
<th></th>
<th>1</th>
<th>2</th>
<th>3</th>
<th>4</th>
<th>5</th>
<th>6</th>
<th>7</th>
<th>8</th>
<th>9</th>
<th>10</th>
<th>11</th>
<th>12</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td>acq((\ell_1))</td>
<td></td>
<td></td>
<td>w((x))</td>
<td>rel((\ell_1))</td>
<td>acq((\ell_2))</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>t_1</td>
<td>acq((\ell_1))</td>
<td>(e_1)</td>
<td>rel((\ell_1))</td>
<td></td>
<td>acq((\ell_2))</td>
<td>(e_2)</td>
<td>rel((\ell_2))</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>t_2</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>acq((\ell_1))</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>t_3</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>rel((\ell_1))</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
Sync-Preserving Ideals (SPIdeals)

How to decide if \((e_1, e_2)\) is a sync-preserving race?

Compute the sync-preserving ideal \(\text{SPIdeal}(e_1, e_2)\)
Sync-Preserving Ideals (SPIdeals)

How to decide if \((e_1, e_2)\) is a sync-preserving race?

Compute the **sync-preserving ideal** \(\text{SPIdeal}(e_1, e_2)\)
Sync-Preserving Ideals (SPIdeals)

How to decide if \((e_1, e_2)\) is a sync-preserving race?

Compute the sync-preserving ideal \(\text{SPIdeal}(e_1, e_2)\)

Sync-preserving race!
Sync-Preserving Ideals (SPIdeals)

How to decide if \((e_1, e_2)\) is a sync-preserving race?

**Compute the sync-preserving ideal** \(\text{SPIdeal}(e_1, e_2)\)

Sync-preserving race!
Sync-Preserving Ideals (SPIIdeals)

How to decide if \((e_1, e_2)\) is a sync-preserving race?

Compute the sync-preserving ideal \(SPIdeal(e_1, e_2)\)

Sync-preserving race!
Sync-Preserving Ideals (SPIdeals)

How to decide if \((e_1, e_2)\) is a sync-preserving race?

Compute the **sync-preserving ideal** \(SPIdeal(e_1, e_2)\)

Sync-preserving race!
How to decide if \((e_1, e_2)\) is a sync-preserving race?

Compute the **sync-preserving ideal** \(\text{SPIdeal}(e_1, e_2)\)

<table>
<thead>
<tr>
<th>(t_1)</th>
<th>(t_2)</th>
<th>(t_3)</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td></td>
<td></td>
</tr>
<tr>
<td>2</td>
<td></td>
<td></td>
</tr>
<tr>
<td>3</td>
<td></td>
<td></td>
</tr>
<tr>
<td>4</td>
<td>acq((l_1))</td>
<td></td>
</tr>
<tr>
<td>5</td>
<td></td>
<td></td>
</tr>
<tr>
<td>6</td>
<td>rel((l_1))</td>
<td></td>
</tr>
<tr>
<td>7</td>
<td></td>
<td></td>
</tr>
<tr>
<td>8</td>
<td>acq((l_2))</td>
<td></td>
</tr>
<tr>
<td>9</td>
<td></td>
<td></td>
</tr>
<tr>
<td>10</td>
<td>rel((l_2))</td>
<td></td>
</tr>
</tbody>
</table>

\(e_1\)

Sync-preserving race!

<table>
<thead>
<tr>
<th>(t_1)</th>
<th>(t_2)</th>
<th>(t_3)</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td></td>
<td></td>
</tr>
<tr>
<td>2</td>
<td></td>
<td></td>
</tr>
<tr>
<td>3</td>
<td></td>
<td></td>
</tr>
<tr>
<td>4</td>
<td></td>
<td></td>
</tr>
<tr>
<td>5</td>
<td>acq((l_1))</td>
<td></td>
</tr>
<tr>
<td>6</td>
<td></td>
<td></td>
</tr>
<tr>
<td>7</td>
<td></td>
<td></td>
</tr>
<tr>
<td>8</td>
<td>w((x))</td>
<td></td>
</tr>
<tr>
<td>9</td>
<td>r((x))</td>
<td></td>
</tr>
<tr>
<td>10</td>
<td></td>
<td></td>
</tr>
<tr>
<td>11</td>
<td></td>
<td></td>
</tr>
<tr>
<td>12</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

\(e_2\)

Sync-preserving race!
Sync-Preserving Ideals (SPIdeals)

How to decide if \((e_1, e_2)\) is a sync-preserving race?

Compute the sync-preserving ideal \(\text{SPIdeal}(e_1, e_2)\)
Sync-Preserving Ideals (SPIdeals)

How to decide if \((e_1, e_2)\) is a sync-preserving race?

Compute the sync-preserving ideal \(SPIdeal(e_1, e_2)\)

Sync-preserving race!

No sync-preserving race!
**Sync-Preserving Ideals (SPIdeals)**

How to decide if \((e_1, e_2)\) is a sync-preserving race?

Compute the **sync-preserving ideal** \(\text{SPIdeal}(e_1, e_2)\)

<table>
<thead>
<tr>
<th></th>
<th>(t_1)</th>
<th>(t_2)</th>
<th>(t_3)</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>acq((l_1))</td>
<td></td>
<td></td>
</tr>
<tr>
<td>2</td>
<td>w((x))</td>
<td></td>
<td></td>
</tr>
<tr>
<td>3</td>
<td>rel((l_1))</td>
<td></td>
<td></td>
</tr>
<tr>
<td>4</td>
<td>acq((l_1))</td>
<td>e_1</td>
<td></td>
</tr>
<tr>
<td>5</td>
<td>rel((l_1))</td>
<td></td>
<td></td>
</tr>
<tr>
<td>6</td>
<td>acq((l_2))</td>
<td></td>
<td></td>
</tr>
<tr>
<td>7</td>
<td>rel((l_2))</td>
<td></td>
<td></td>
</tr>
<tr>
<td>8</td>
<td>r((x))</td>
<td></td>
<td></td>
</tr>
<tr>
<td>9</td>
<td>acq((l_2))</td>
<td>e_2</td>
<td></td>
</tr>
<tr>
<td>10</td>
<td>rel((l_2))</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

**Sync-preserving race!**

<table>
<thead>
<tr>
<th></th>
<th>(t_1)</th>
<th>(t_2)</th>
<th>(t_3)</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>acq((l_1))</td>
<td></td>
<td></td>
</tr>
<tr>
<td>2</td>
<td>w((x))</td>
<td></td>
<td></td>
</tr>
<tr>
<td>3</td>
<td>r((x))</td>
<td>e_1</td>
<td></td>
</tr>
<tr>
<td>4</td>
<td>w((y))</td>
<td></td>
<td></td>
</tr>
<tr>
<td>5</td>
<td>r((y))</td>
<td></td>
<td></td>
</tr>
<tr>
<td>6</td>
<td>rel((l_1))</td>
<td>e_2</td>
<td></td>
</tr>
<tr>
<td>7</td>
<td>rel((l_1))</td>
<td></td>
<td></td>
</tr>
<tr>
<td>8</td>
<td>acq((l_1))</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

**No sync-preserving race!**

**Theorem**

\((e_1, e_2)\) is a sync-preserving race iff \(\text{SPIdeal}(e_1, e_2) \cap \{e_1, e_2\} = \emptyset\)
Monotonicity of SPIIdeals

How to test for all racy events?
Monotonicity of SPIdeals

How to test for all racy events?

SPIdeal(e₁, e₂) ⊆ SPIdeal(e₃, e₄)
Detecting All Racy Events

Pay the cost for SPIdeal($w_2(x)$, $w(x))$

Repeat for all thread pairs and variables

race?
Detecting All Racy Events

Race!

$w_1(x)$

$w(x)$

race?
Detecting All Racy Events

$w_1(x)$

$w(x)$

Not Race!

race?
Detecting All Racy Events

Pay the cost for SPIdeal(w₂(x), w(x)) \ SPIdeal(w₁(x), w(x))
Pay the cost for $\text{SPIdeal}(w_2(x), w(x)) \setminus \text{SPIdeal}(w_1(x), w(x))$

Repeat for all thread pairs and variables
What is the time/space complexity of sync-preserving races?
Optimality

**Theorem**

SyncP runs in $\tilde{O}(N)$ time and space

- Clearly, time is almost optimal (i.e., almost linear)
- Can space be improved?
- E.g., HB takes only $\tilde{O}(1)$ space
Optimality

**Theorem**

SyncP runs in $\tilde{O}(N)$ time and space

- Clearly, time is almost optimal (i.e., almost linear)
- Can space be improved?
- E.g., HB takes only $\tilde{O}(1)$ space

**Theorem**

Any streaming (i.e., one-pass) algorithm that decides whether there is a sync-preserving race must use $\Omega(N/\log^2 N)$ space.
## Optimality

<table>
<thead>
<tr>
<th>Theorem</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>SyncP runs in</strong> (\tilde{O}(N)) <strong>time and space</strong></td>
</tr>
</tbody>
</table>

- Clearly, time is almost optimal (i.e., almost linear)
- Can space be improved?
- E.g., HB takes only \(\tilde{O}(1)\) space

<table>
<thead>
<tr>
<th>Theorem</th>
</tr>
</thead>
<tbody>
<tr>
<td><em>Any streaming (i.e., one-pass) algorithm that decides whether there is a sync-preserving race must use</em> (\Omega(N / \log^2 N)) <strong>space.</strong></td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Optimality</th>
</tr>
</thead>
<tbody>
<tr>
<td>SyncP is nearly optimal for both time and space.</td>
</tr>
</tbody>
</table>
What happens beyond sync-preserving races?
Parametric On Sync-Reversals

- Sync-preserving: “Do not try to reorder any critical section on $\sigma$”
- What if we try to reorder only a few?

$\sim O(N)$ when $k = 0$

Hopefully small when $k = 1$ (?)
Sync-preserving: “Do not try to reorder any critical section on $\sigma$”

What if we try to reorder only a few?

**$k$-sync-reversal races**

Exposed by reordering at most $k$ critical sections of $\sigma$

- Sync-preserving = 0-sync-reversals
Parametric On Sync-Reversals

• Sync-preserving: “Do not try to reorder any critical section on $\sigma$”
• What if we try to reorder only a few?

$k$-sync-reversal races
Exposed by reordering at most $k$ critical sections of $\sigma$

• Sync-preserving = 0-sync-reversals

Question
What is the cost of $k$-sync-reversal races when $k$ is small (e.g., $k \leq 5$)?

• $\tilde{O}(N)$ when $k = 0$
• Hopefully small when $k = 1$ (?)
Hardness of 1 Sync-Reversals

Theorem

Dynamic race prediction on traces with a single lock and two critical sections is $W[1]$-hard parameterized by the number of threads.
Hardness of 1 Sync-Reversals

**Theorem**

*Dynamic race prediction on traces with a single lock and two critical sections is $W[1]$-hard parameterized by the number of threads.*

- NP complete
- No algorithm in time $2^T \cdot N^{O(1)}$
  - $N$ events, $T$ threads
Hardness of 1 Sync-Reversals

Theorem

*Dynamic race prediction on traces with a single lock and two critical sections is \(W[1]\)-hard parameterized by the number of threads.*

- NP complete
- No algorithm in time \(2^T \cdot \mathcal{N}^O(1)\)
  - \(\mathcal{N}\) events, \(T\) threads

Moral

The smallest possible synchronization (just 2 critical sections!) makes the problem as hard as in the general case.
A single sync-reversal can imply arbitrary many.

Problem as hard as general race prediction.

A single sync-reversal can imply arbitrary many.

Problem as hard as general race prediction.

A single sync-reversal can imply arbitrary many.
Hardness of 1 Sync-Reversals - Intuition

A single sync-reversal can imply arbitrary many problems as hard as general race prediction.

A single sync-reversal can imply arbitrary many

Problem as hard as general race prediction

Experiments
Setup

- Implementation of SyncP in Rapid\textsuperscript{2}
- Comparison with SHB, WCP, M2
- Counted sound racy events reported
- 1h timeout

\textsuperscript{2}https://github.com/umangm/rapid
<table>
<thead>
<tr>
<th>Benchmark</th>
<th>SHB</th>
<th>WCP</th>
<th>M2</th>
<th>SyncP</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>Races</td>
<td>Time</td>
<td>Races</td>
<td>Time</td>
</tr>
<tr>
<td>...</td>
<td>...</td>
<td>...</td>
<td>...</td>
<td>...</td>
</tr>
<tr>
<td>wronglock</td>
<td>2</td>
<td>0.02s</td>
<td>2</td>
<td>0.09s</td>
</tr>
<tr>
<td>clean</td>
<td>4</td>
<td>0.04s</td>
<td>4</td>
<td>0.14s</td>
</tr>
<tr>
<td>mergesort</td>
<td>1</td>
<td>11m10s</td>
<td>1</td>
<td>0.12s</td>
</tr>
<tr>
<td>readswrites</td>
<td>4</td>
<td>0.12s</td>
<td>4</td>
<td>0.41s</td>
</tr>
<tr>
<td>ftpserver</td>
<td>21</td>
<td>6.91s</td>
<td>21</td>
<td>1.34s</td>
</tr>
<tr>
<td>derby</td>
<td>10</td>
<td>0.01s</td>
<td>10</td>
<td>16.48s</td>
</tr>
<tr>
<td>jigsaw</td>
<td>4</td>
<td>0.41s</td>
<td>4</td>
<td>19.53s</td>
</tr>
<tr>
<td>sunflow</td>
<td>6</td>
<td>39.66s</td>
<td>6</td>
<td>47.14s</td>
</tr>
<tr>
<td>linkedlist</td>
<td>4</td>
<td>7.25s</td>
<td>3</td>
<td>27.07s</td>
</tr>
<tr>
<td>cryptorsa</td>
<td>5</td>
<td>3m4s</td>
<td>5</td>
<td>6m35s</td>
</tr>
<tr>
<td>xalan</td>
<td>10</td>
<td>0.15s</td>
<td>7</td>
<td>15m30s</td>
</tr>
<tr>
<td>luindex</td>
<td>1</td>
<td>24m40s</td>
<td>2</td>
<td>31m6s</td>
</tr>
<tr>
<td><strong>Totals</strong></td>
<td>157</td>
<td>1h51m</td>
<td>134</td>
<td>3h15m</td>
</tr>
</tbody>
</table>
Observation

Most races are sync-preserving and they can be detected fast.
Conclusion

Sync-preserving races:

- Subsume HB races
- Need not be consecutive
- Can be detected in $\tilde{O}(N)$ time and space
- Are abundant in practice, more than HB races

Thank you!
Sync-preserving races:

- Subsume HB races
- Need not be consecutive
- Can be detected in $\tilde{O}(N)$ time and space
- Are abundant in practice, more than HB races

- The $\tilde{O}(N)$ bound is optimal for both time and space
- Relaxing the definition even a bit makes race prediction intractable
  - Races with only 1 sync-reversal are NP-hard to detect
Sync-preserving races:

- Subsume HB races
- Need not be consecutive
- Can be detected in $\tilde{O}(N)$ time and space
- Are abundant in practice, more than HB races

- The $\tilde{O}(N)$ bound is optimal for both time and space
- Relaxing the definition even a bit makes race prediction intractable
  - Races with only 1 sync-reversal are NP-hard to detect

Thank you!