The Complexity of Dynamic Data Race Prediction

Umang Mathur
University of Illinois at Urbana-Champaign

Andreas Pavlogiannis
Aarhus University

Mahesh Viswanathan
University of Illinois at Urbana-Champaign
Concurrency and Challenges
Concurrency and Challenges

- Concurrency - ubiquitous programming paradigm
Concurrenty and Challenges

• Concurrency - ubiquitous programming paradigm
Concurrency and Challenges

- Concurrency - ubiquitous programming paradigm
- Challenging to develop concurrent software
  - Large interleaving space
Concurrency and Challenges

- Concurrency - ubiquitous programming paradigm
- Challenging to develop concurrent software
  - Large interleaving space
- Concurrency bugs arise in production-level software
  - Despite rigorous testing
Concurrency and Challenges

- Concurrency - ubiquitous programming paradigm
- Challenging to develop concurrent software
  - Large interleaving space
- Concurrency bugs arise in production-level software
  - Despite rigorous testing
- **Data races**: most common source of concurrency issues
Data Race Detection

### Static analyses
- Analyze source code
- Undecidable problem
- Excessive false alarms

### Dynamic analyses
- Analyze executions at runtime
- Typically sound
- Widely adopted - TSan, Helgrind, etc.,
Concurrent Program Executions

• Sequences of events
• Event $e = <t, op>$
  • $t$ is the thread that performs $e$
  • $op$ is an operation
• Operations:
  • Read/Write to memory locations
  • Acquire and release of locks
• Well formed-ness
  • At most one thread holds a lock at any time

<table>
<thead>
<tr>
<th></th>
<th>$t1$</th>
<th>$t2$</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>acq(l)</td>
<td></td>
</tr>
<tr>
<td>2</td>
<td>w(x)</td>
<td></td>
</tr>
<tr>
<td>3</td>
<td>w(y)</td>
<td></td>
</tr>
<tr>
<td>4</td>
<td></td>
<td>r(x)</td>
</tr>
<tr>
<td>5</td>
<td>rel(l)</td>
<td></td>
</tr>
<tr>
<td>6</td>
<td></td>
<td>w(y)</td>
</tr>
</tbody>
</table>
Data Races and Dynamic Race Prediction
Data Races and Dynamic Race Prediction

An execution $\sigma$ has a data race if

• pair of conflicting events
• consecutive
Data Races and Dynamic Race Prediction

An execution $\sigma$ has a data race if
- pair of conflicting events
- consecutive

1. Same memory location
2. Different threads
3. At least one write
Data Races and Dynamic Race Prediction

An execution $\sigma$ has a data race if

- a pair of conflicting events
- consecutive

1. Same memory location
2. Different threads
3. At least one write

<table>
<thead>
<tr>
<th>t1</th>
<th>t2</th>
</tr>
</thead>
<tbody>
<tr>
<td>acq(l)</td>
<td></td>
</tr>
<tr>
<td>w(x)</td>
<td>r(x)</td>
</tr>
<tr>
<td>rel(l)</td>
<td></td>
</tr>
</tbody>
</table>
Data Races and Dynamic Race Prediction

An execution $\sigma$ has a data race if

- **pair of conflicting events**
- consecutive

1. Same memory location
2. Different threads
3. At least one write

<table>
<thead>
<tr>
<th>t1</th>
<th>t2</th>
</tr>
</thead>
<tbody>
<tr>
<td>acq(1)</td>
<td></td>
</tr>
<tr>
<td>w(x)</td>
<td>r(x)</td>
</tr>
<tr>
<td>rel(1)</td>
<td></td>
</tr>
</tbody>
</table>
Data Races and Dynamic Race Prediction

An execution $\sigma$ has a data race if

1. Same memory location
2. Different threads
3. At least one write

Given an execution $\sigma$, does $\sigma$ have a data race?

- Sensitive to thread scheduling
- Miss a lot of races
Data Races and Dynamic Race Prediction

An execution $\sigma$ has a data race if:
1. Same memory location
2. Different threads
3. At least one write

Given an execution $\sigma$, does $\sigma$ have a data race?
- Sensitive to thread scheduling
- Miss a lot of races
An execution $\sigma$ has a data race if
- pair of conflicting events
- consecutive

Given an execution $\sigma$, does $\sigma$ have a data race?
- Sensitive to thread scheduling
- Miss a lot of races

Data Race Detection

<table>
<thead>
<tr>
<th>t1</th>
<th>t2</th>
</tr>
</thead>
<tbody>
<tr>
<td>acq(l)</td>
<td></td>
</tr>
<tr>
<td>w(x)</td>
<td></td>
</tr>
<tr>
<td>rel(l)</td>
<td>r(x)</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>t1</th>
<th>t2</th>
</tr>
</thead>
<tbody>
<tr>
<td>acq(l)</td>
<td></td>
</tr>
<tr>
<td>w(x)</td>
<td></td>
</tr>
<tr>
<td>rel(l)</td>
<td>r(x)</td>
</tr>
</tbody>
</table>
Data Races and Dynamic Race Prediction

An execution $\sigma$ has a data race if:
- pair of conflicting events
- consecutive

1. Same memory location
2. Different threads
3. At least one write

Given an execution $\sigma$, does $\sigma$ have a data race?
- Sensitive to thread scheduling
- Miss a lot of races
An execution $\sigma$ has a data race if
- pair of conflicting events
- consecutive

Given an execution $\sigma$, does $\sigma$ have a data race?
- Sensitive to thread scheduling
- Miss a lot of races

Given an execution $\sigma$, is there a reordering* $\rho$ of $\sigma$, such that $\rho$ has a data race?

<table>
<thead>
<tr>
<th>t1</th>
<th>t2</th>
</tr>
</thead>
<tbody>
<tr>
<td>acq(l)</td>
<td>r(x)</td>
</tr>
<tr>
<td>w(x)</td>
<td>rel(l)</td>
</tr>
<tr>
<td></td>
<td>r(x)</td>
</tr>
</tbody>
</table>
Data Races and Dynamic Race Prediction

An execution $\sigma$ has a data race if:
1. Same memory location
2. Different threads
3. At least one write

Given an execution $\sigma$, does $\sigma$ have a data race?
- Sensitive to thread scheduling
- Miss a lot of races

Given an execution $\sigma$, is there a reordering* $\rho$ of $\sigma$, such that $\rho$ has a data race?
Which reorderings are allowed?
Which reorderings are allowed?

Any program that generates the observed execution must also generate the reordering.
Which reorderings are allowed?

Any program that generates the observed execution must also generate the reordering

Reorderings must satisfy some properties -
Which reorderings are allowed?

Any program that generates the observed execution must also generate the reordering

Reorderings must satisfy some properties -

1. Well formed -
Which reorderings are allowed?

Any program that generates the observed execution must also generate the reordering

Reorderings must satisfy some properties -

1. Well formed -
   • critical sections on same lock don’t overlap
Which reorderings are allowed?

Any program that generates the observed execution must also generate the reordering

Reorderings must satisfy some properties -

1. Well formed -
   - critical sections on same lock don’t overlap
Which reorderings are allowed?

Reorderings must satisfy some properties -

1. Well formed -
   • critical sections on same lock don’t overlap

Any program that generates the observed execution must also generate the reordering
Which reorderings are allowed?

Any program that generates the observed execution must also generate the reordering.

Reorderings must satisfy some properties -

1. Well formed -
   - critical sections on same lock don’t overlap

<table>
<thead>
<tr>
<th>t1</th>
<th>t2</th>
</tr>
</thead>
<tbody>
<tr>
<td>acq(l)</td>
<td>acq(l)</td>
</tr>
<tr>
<td>w(x)</td>
<td></td>
</tr>
<tr>
<td>rel(l)</td>
<td>rel(l)</td>
</tr>
</tbody>
</table>
Which reorderings are allowed?

Any program that generates the observed execution must also generate the reordering

Reorderings must satisfy some properties -

1. Well formed -
   - critical sections on same lock don’t overlap

<table>
<thead>
<tr>
<th>t1</th>
<th>t2</th>
</tr>
</thead>
<tbody>
<tr>
<td>acq(1)</td>
<td></td>
</tr>
<tr>
<td>w(x)</td>
<td></td>
</tr>
<tr>
<td>rel(l)</td>
<td>acq(l)</td>
</tr>
<tr>
<td></td>
<td>rel(l)</td>
</tr>
</tbody>
</table>
Which reorderings are allowed?

Reorderings must satisfy some properties -

1. Well formed -
   • critical sections on same lock don’t overlap
2. Preserve intra-thread ordering

Any program that generates the observed execution must also generate the reordering
Which reorderings are allowed?

Any program that generates the observed execution must also generate the reordering

Reorderings must satisfy some properties -

1. Well formed -
   - critical sections on same lock don’t overlap
2. Preserve intra-thread ordering
Which reorderings are allowed?

Any program that generates the observed execution must also generate the reordering

Reorderings must satisfy some properties -

1. Well formed -
   - critical sections on same lock don’t overlap

2. Preserve intra-thread ordering

<table>
<thead>
<tr>
<th>t1</th>
<th>t2</th>
</tr>
</thead>
<tbody>
<tr>
<td>acq(1)</td>
<td>acq(1)</td>
</tr>
<tr>
<td>w(x)</td>
<td>rel(1)</td>
</tr>
<tr>
<td>rel(1)</td>
<td></td>
</tr>
</tbody>
</table>

Any program that generates the observed execution must also generate the reordering

<table>
<thead>
<tr>
<th>t1</th>
<th>t2</th>
</tr>
</thead>
<tbody>
<tr>
<td>acq(1)</td>
<td>acq(1)</td>
</tr>
<tr>
<td>w(x)</td>
<td>rel(1)</td>
</tr>
<tr>
<td>rel(1)</td>
<td></td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>t1</th>
<th>t2</th>
</tr>
</thead>
<tbody>
<tr>
<td>acq(1)</td>
<td>acq(1)</td>
</tr>
<tr>
<td>w(x)</td>
<td>rel(1)</td>
</tr>
<tr>
<td>rel(1)</td>
<td></td>
</tr>
</tbody>
</table>

Any program that generates the observed execution must also generate the reordering
Which reorderings are allowed?

Reorderings must satisfy some properties -

1. Well formed -
   - critical sections on same lock don’t overlap
2. Preserve intra-thread ordering
3. Preserve control flow

Any program that generates the observed execution must also generate the reordering
Reorderings must satisfy some properties -

1. Well formed -
   - critical sections on same lock don’t overlap
2. Preserve intra-thread ordering
3. Preserve control flow
   - Every read sees its original write

Any program that generates the observed execution must also generate the reordering
Which reorderings are allowed?

Reorderings must satisfy some properties -

1. **Well formed** -
   - critical sections on same lock don’t overlap

2. **Preserve intra-thread ordering**

3. **Preserve control flow**
   - Every read sees its original write

Any program that generates the observed execution must also generate the reordering.
Which reorderings are allowed?

Reorderings must satisfy some properties -

1. Well formed -
   - critical sections on same lock don’t overlap
2. Preserve intra-thread ordering
3. Preserve control flow
   - Every read sees its original write

Any program that generates the observed execution must also generate the reordering
Which reorderings are allowed?

Any program that generates the observed execution must also generate the reordering

Reorderings must satisfy some properties -

1. Well formed -
   - critical sections on same lock don’t overlap
2. Preserve intra-thread ordering
3. Preserve control flow
   - Every read sees its original write

* Herlihy and Wing, Linearizability: A Correctness Condition for Concurrent Objects, TOPLAS 1990
  Smaragdakis et al, Sound predictive race detection in polynomial time, POPL 2012
Some History

Given an execution $\sigma$, is there a **correct reordering** of $\sigma$ with a data race?
Some History

Data Race Prediction

Given an execution $\sigma$, is there a correct reordering of $\sigma$ with a data race?
Some History

**Data Race Prediction**

Given an execution $\sigma$, is there a **correct reordering** of $\sigma$ with a data race?

1970  \hspace{2cm} 2003

Initial developments based on HB
Some History

Data Race Prediction

Given an execution $\sigma$, is there a \textit{correct reordering} of $\sigma$ with a data race?

Happens-Before

1970 → 2003

Initial developments based on HB
Some History

Given an execution $\sigma$, is there a **correct reordering** of $\sigma$ with a data race?
Some History

Data Race Prediction

Given an execution $\sigma$, is there a correct reordering of $\sigma$ with a data race?

Initial developments based on HB

1970

2003

Happens-Before

DJIT

Eraser

1970

2003

Happens-Before
Some History

Given an execution $\sigma$, is there a **correct reordering** of $\sigma$ with a data race?

**Data Race Prediction**

- **1970**
  - Happens-Before
  - Initial developments based on HB

- **2003**
  - DJIT
  - Eraser
  - DJIT+ (Vector clocks)
Some History

Given an execution $\sigma$, is there a **correct reordering** of $\sigma$ with a data race?

**Data Race Prediction**

1970

- DJIT
- Eraser
- Happens-Before
- DJIT+ (Vector clocks)

2003

Initial developments based on HB

2005

Explicit or Symbolic Search

2011

- 1970
- 2003
- 2005
- 2011
Some History

Given an execution $\sigma$, is there a correct reordering of $\sigma$ with a data race?

**Data Race Prediction**

- **DJIT**
- **Eraser**
- **Happens-Before**
- **Djit+ (Vector clocks)**
- **Exhaustive enumeration**
- **Explicit or Symbolic Search**

- 1970
- 2003
- 2005
- 2011

Initial developments based on HB — Exhaustive enumeration — Explicit or Symbolic Search

- 1970
- 2003
- 2005
- 2011
Data Race Prediction

Given an execution $\sigma$, is there a correct reordering of $\sigma$ with a data race?

1970
- Happens-Before
- Initial developments based on HB

2003
- DJIT
- DJIT+ (Vector clocks)

2005
- Exhaustive enumeration

2011
- Explicit or Symbolic Search
- SAT solving
Given an execution $\sigma$, is there a correct reordering of $\sigma$ with a data race?

**Data Race Prediction**

- 1970: Initial developments based on HB
- 2003: DJIT
- 2005: DJIT+
- 2007: Exhaustive enumeration
- 2009: SAT solving
- 2011: Explicit or Symbolic Search
Given an execution $\sigma$, is there a **correct reordering** of $\sigma$ with a data race?
Some History

Data Race Prediction

Given an execution $\sigma$, is there a correct reordering of $\sigma$ with a data race?

Initial developments based on HB

Explicit or Symbolic Search

Polynomial time algorithms, better than HB

DJIT

Eraser

Happens-Before

DJIT+ (Vector clocks)

Exhaustive enumeration

SAT solving

WCP partial order

CP partial order

DC

M2

SDP

Goldilocks

FastTrack

2007

2009

2011

2012

>12 papers per year

Polynomial time algorithms, better than HB

DC

M2

SDP
Some History

Given an execution $\sigma$, is there a correct reordering of $\sigma$ with a data race?

Data Race Prediction

- **Happens-Before** (1970)
- **DJIT**
- **Eraser**
- **Happens-Before** (Vector clocks)**DJIT+**
- **Exhaustive enumeration** (2005)
- **FastTrack** (2007)
- **Goldilocks**
- **CP partial order** (2011)
- **WCP partial order** (2017)
- **SAT solving** (2009)
- **DC**
- **M2**
- **M2**
- **SDP** (2018)
- **Polynomial time algorithms, better than HB** (2019)

Initial developments based on HB

Explicit or Symbolic Search

Polynomial time algorithms, better than HB

Complexity: PTIME

Completeness
Some History

Given an execution $\sigma$, is there a correct reordering of $\sigma$ with a data race?

- **Data Race Prediction**

1970

- Happens-Before
- DJIT
- Eraser

2003

- DJIT+
  - (Vector clocks)

2005

- Goldilocks
- FastTrack

2007

- Exhaustive enumeration

2009

- SAT solving

2011

- CP partial order

2012

- WCP partial order

2017

- DC
- SHB

2018

- M2

2019

- SDP

>12 papers per year

Initial developments based on HB

Explicit or Symbolic Search

Polynomial time algorithms, better than HB

- Complexity
- PTIME
- Completeness

- Complexity
- Exponential
- Completeness

- Complexity
- Exponential
- Completeness

1970: Initial developments based on HB

2003: DJIT+

2005: Goldilocks, FastTrack

2007: Exhaustive enumeration

2009: SAT solving

2011: CP partial order

2012: WCP partial order

2017-2019: >12 papers per year
Given an execution $\sigma$, is there a correct reordering of $\sigma$ with a data race?

**Data Race Prediction**

- **1970**: Initial developments based on HB
- **2003**: DJIT, Eraser
- **2005**: Happens-Before, DJIT+ (Vector clocks)
- **2007**: Exhaustive enumeration
- **2009**: FastTrack
- **2011**: CP partial order
- **2012**: Goldilocks
- **2017**: WCP partial order
- **2018**: CP partial order
- **2019**: WCP partial order

**Timeline**
- 1970: Initial developments based on HB
- 2003: DJIT, Eraser
- 2005: Happens-Before, DJIT+ (Vector clocks)
- 2007: Exhaustive enumeration
- 2009: FastTrack
- 2011: CP partial order
- 2012: Goldilocks
- 2017: WCP partial order
- 2018: CP partial order
- 2019: WCP partial order

**Algorithms**
- **1970**: Initial developments based on HB
- **2003**: DJIT, Eraser
- **2005**: Happens-Before, DJIT+ (Vector clocks)
- **2007**: Exhaustive enumeration
- **2009**: FastTrack
- **2011**: CP partial order
- **2012**: Goldilocks
- **2017**: WCP partial order
- **2018**: CP partial order
- **2019**: WCP partial order

**Complexity**
- **PTIME**: Polynomial time algorithms, better than HB
- **Exponential**: Complexity
- **Completeness**: PTIME
Some History

Given an execution \(\sigma\), is there a correct reordering of \(\sigma\) with a data race?

Is there a complete algorithm that runs in polynomial time?
Some History

Given an execution $\sigma$, is there a correct reordering of $\sigma$ with a data race?

Is there a complete algorithm that runs in polynomial time?

What is the exact complexity of Data Race Prediction?
How hard is Data Race Prediction?
How hard is Data Race Prediction?
How hard is Data Race Prediction?

• **Input:** Trace $\sigma$ and events $e_1$ and $e_2$
  
  [$n$ events, $k$ threads, $d$ memory locations and locks]
How hard is Data Race Prediction?

• **Input:** Trace $\sigma$ and events $e_1$ and $e_2$
  
  [n events, k threads, d memory locations and locks]

• **Output:** YES iff there is a correct reordering of $\sigma$ that exhibits data race ($e_1, e_2$).
How hard is Data Race Prediction?

**Input:** Trace \( \sigma \) and events \( e_1 \) and \( e_2 \) 
[\( n \) events, \( k \) threads, \( d \) memory locations and locks]

**Output:** YES iff there is a correct reordering of \( \sigma \) that exhibits data race \((e_1, e_2)\).
How hard is Data Race Prediction?

**Data Race Prediction**

- **Input:** Trace $\sigma$ and events $e_1$ and $e_2$
  
  [$n$ events, $k$ threads, $d$ memory locations and locks]

- **Output:** YES iff there is a correct reordering of $\sigma$ that exhibits data race $(e_1, e_2)$.

**(Easy) Upper Bounds**

1. **NP**
   - Guess an alternate reordering and check
     if it is a correct reordering
How hard is Data Race Prediction?

**Data Race Prediction**

- **Input:** Trace $\sigma$ and events $e_1$ and $e_2$  
  [n events, k threads, d memory locations and locks]
- **Output:** YES iff there is a correct reordering of $\sigma$ that exhibits data race $(e_1, e_2)$.

**Easy Upper Bounds**

1. **NP**
   - Guess an alternate reordering and check if it is a correct reordering
2. **O($k^n$)** - Enumeration based techniques:
   - At every step, choose which thread to execute
How hard is Data Race Prediction?

**Input:** Trace $\sigma$ and events $e_1$ and $e_2$

$[n \text{ events, } k \text{ threads, } d \text{ memory locations and locks}]$

**Output:** YES iff there is a correct reordering of $\sigma$ that exhibits data race ($e_1, e_2$).

**(Easy) Upper Bounds**

1. **NP**
   - Guess an alternate reordering and check if it is a correct reordering

2. $O(k^n)$ - Enumeration based techniques:
   - At every step, choose which thread to execute

3. $O(SAT(poly(n)))$ - SAT solving based techniques
How hard is Data Race Prediction?

**Data Race Prediction**

- **Input:** Trace $\sigma$ and events $e_1$ and $e_2$
  
  $[n \text{ events, } k \text{ threads, } d \text{ memory locations and locks}]$

- **Output:** YES iff there is a correct reordering of $\sigma$ that exhibits data race ($e_1$, $e_2$).

### (Easy) Upper Bounds

1. **NP**
   - Guess an alternate reordering and check if it is a correct reordering

2. $O(k^n)$ - Enumeration based techniques:
   - At every step, choose which thread to execute

3. $O(SAT(poly(n)))$ - SAT solving based techniques

### Lower Bound
How hard is Data Race Prediction?

**Input:** Trace $\sigma$ and events $e_1$ and $e_2$

$[n$ events, $k$ threads, $d$ memory locations and locks$]$

**Output:** YES iff there is a correct reordering of $\sigma$ that exhibits data race $(e_1, e_2)$.

### (Easy) Upper Bounds

1. **NP**
   • Guess an alternate reordering and check if it is a correct reordering
2. $O(k^n)$ - Enumeration based techniques:
   • At every step, choose which thread to execute
3. $O(SAT(poly(n)))$ - SAT solving based techniques

### Lower Bound

• Unknown!
• Is it **NP**-hard? Is enumeration unavoidable?
• Is it polynomial time?
How hard is Data Race Prediction?

- **Input:** Trace $\sigma$ and events $e_1$ and $e_2$
  
  [n events, k threads, d memory locations and locks]

- **Output:** YES iff there is a correct reordering of $\sigma$ that exhibits data race ($e_1, e_2$).
How hard is Data Race Prediction?

**Input:** Trace $\sigma$ and events $e_1$ and $e_2$
- [n events, k threads, d memory locations and locks]

**Output:** YES iff there is a correct reordering of $\sigma$ that exhibits data race ($e_1$, $e_2$).

---

Closely Related Work

1. [Netzer and Miller, '89-'92]: **NP**-hardness of data race prediction.
   - Too strong notion of correct reordering
   - Hardness comes from more powerful synchronization primitives
How hard is Data Race Prediction?

**Data Race Prediction**

- **Input:** Trace $\sigma$ and events $e_1$ and $e_2$
  
  [n events, k threads, d memory locations and locks]

- **Output:** YES iff there is a correct reordering of $\sigma$ that exhibits data race $(e_1, e_2)$.

**Closely Related Work**

1. [Netzer and Miller, ’89-’92]: $\textbf{NP}$-hardness of data race prediction.
   - Too strong notion of correct reordering
   - Hardness comes from more powerful synchronization primitives

2. [Gibbons and Korach, ’97]: $\textbf{NP}$-hardness of Verifying Sequential Consistency
   - Different problem than data race detection
Contributions

- **Input:** Trace $\sigma$ and events $e_1$ and $e_2$
  
  [n events, k threads, d memory locations and locks]

- **Output:** YES iff there is a correct reordering of $\sigma$ that exhibits data race ($e_1$, $e_2$).
Contributions

- **Input:** Trace $\sigma$ and events $e_1$ and $e_2$
  
  [n events, k threads, d memory locations and locks]

- **Output:** YES iff there is a correct reordering of $\sigma$ that exhibits data race ($e_1, e_2$).

Extensive study of complexity theoretic questions in data race prediction†
Contributions

Data Race Prediction

- **Input:** Trace $\sigma$ and events $e_1$ and $e_2$
  
  [n events, k threads, d memory locations and locks]

- **Output:** YES iff there is a correct reordering of $\sigma$ that exhibits data race $(e_1, e_2)$.

Extensive study of complexity theoretic questions in data race prediction†
Contributions

Data Race Prediction

- **Input:** Trace $\sigma$ and events $e_1$ and $e_2$
  - $n$ events, $k$ threads, $d$ memory locations and locks
- **Output:** YES iff there is a correct reordering of $\sigma$ that exhibits data race ($e_1, e_2$).

Extensive study of complexity theoretic questions in data race prediction†

General Case

1. **Poly-time Upper bound (when $k$ is constant)**
   Algorithm for race prediction $O(kn^{2(k-1)})$
Contributions

- **Input:** Trace $\sigma$ and events $e_1$ and $e_2$
  - $[n \text{ events, } k \text{ threads, } d \text{ memory locations and locks}]$
- **Output:** YES iff there is a correct reordering of $\sigma$ that exhibits data race $(e_1, e_2)$.

Extensive study of complexity theoretic questions in data race prediction

General Case

1. **Poly-time Upper bound (when $k$ is constant)**
   - Algorithm for race prediction $O(nk^{2(k-1)})$

2. **Lower bound:** $\mathcal{W}[1]$ hard in parameter $k$
   - $\mathsf{NP}$-hard when $k$ is not constant
   - Not even FPT in $k$
Contributions

Data Race Prediction

- **Input**: Trace $\sigma$ and events $e_1$ and $e_2$
  - [n events, k threads, d memory locations and locks]
- **Output**: YES iff there is a correct reordering of $\sigma$ that exhibits data race ($e_1$, $e_2$).

Extensive study of complexity theoretic questions in data race prediction†

1. **Poly-time Upper bound (when k is constant)**
   - Algorithm for race prediction $O(kn^{2(k-1)})$

2. **Lower bound**: W[1] hard in parameter k
   - $\textbf{NP}$-hard when k is not constant
   - Not even FPT in k
Contributions

Data Race Prediction

- **Input:** Trace $\sigma$ and events $e_1$ and $e_2$
  - $[n$ events, $k$ threads, $d$ memory locations and locks$]$  
- **Output:** YES iff there is a correct reordering of $\sigma$ that exhibits data race $(e_1, e_2)$.

Extensive study of complexity theoretic questions in data race prediction†

General Case

1. **Poly-time Upper bound (when $k$ is constant)**
   - Algorithm for race prediction $O(kn^{2(k-1)})$

2. **Lower bound:** $W[1]$ hard in parameter $k$
   - $\textbf{NP}$-hard when $k$ is not constant
   - Not even FPT in $k$

Special Cases

3. **Restricting the space of input traces**
   - $O(n^2)$ time algorithm
   - Matching (conditional) lower bound
Contributions

Data Race Prediction

• **Input:** Trace $\sigma$ and events $e_1$ and $e_2$
  
  $[n \text{ events, } k \text{ threads, } d \text{ memory locations and locks}]$

• **Output:** YES iff there is a correct reordering of $\sigma$ that exhibits data race $(e_1, e_2)$.

Extensive study of complexity theoretic questions in data race prediction†

General Case

1. **Poly-time Upper bound (when $k$ is constant)**
   
   Algorithm for race prediction $O(kn^{2(k-1)})$

2. **Lower bound:** $\mathcal{W}[1]$ hard in parameter $k$
   
   • $\mathcal{NP}$-hard when $k$ is not constant
   
   • Not even FPT in $k$

Special Cases

3. **Restricting the space of input traces**
   
   • $O(n^2)$ time algorithm
   
   • Matching (conditional) lower bound

4. **Restricting the space of data races to be reported**
   
   • **Linear** time algorithm
1. Trace Ideals and Realizability
2. Algorithm for Data Race Prediction (General case)
3. Data Race Prediction for Acyclic Communication Topology
4. Distance Bounded Data Race Prediction
5. Lower Bound (General Case)
6. Lower Bound for 2 threads
7. Conclusions and Future Work
1. **Trace Ideals and Realizability**

2. Algorithm for Data Race Prediction (General case)

3. Data Race Prediction for Acyclic Communication Topology

4. Distance Bounded Data Race Prediction

5. Lower Bound (General Case)

6. Lower Bound for 2 threads

7. Conclusions and Future Work
## Some Notations

<table>
<thead>
<tr>
<th></th>
<th>t₁</th>
<th>t₂</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>acq(l)</td>
<td></td>
</tr>
<tr>
<td>2</td>
<td>rel(l)</td>
<td></td>
</tr>
<tr>
<td>3</td>
<td>w(x)</td>
<td></td>
</tr>
<tr>
<td>4</td>
<td></td>
<td>r(x)</td>
</tr>
<tr>
<td>5</td>
<td></td>
<td>acq(l)</td>
</tr>
<tr>
<td>6</td>
<td></td>
<td>rel(l)</td>
</tr>
</tbody>
</table>
Some Notations

Let $\sigma$ be an execution trace.

<table>
<thead>
<tr>
<th></th>
<th>t1</th>
<th>t2</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>acq(l)</td>
<td></td>
</tr>
<tr>
<td>2</td>
<td>rel(l)</td>
<td></td>
</tr>
<tr>
<td>3</td>
<td>w(x)</td>
<td></td>
</tr>
<tr>
<td>4</td>
<td>r(x)</td>
<td></td>
</tr>
<tr>
<td>5</td>
<td>acq(l)</td>
<td></td>
</tr>
<tr>
<td>6</td>
<td>rel(l)</td>
<td></td>
</tr>
</tbody>
</table>
Some Notations

Let \( \sigma \) be an execution trace.

- \( \text{TO}_\sigma = \) intra-thread ordering on events of \( \sigma \)

\[
\begin{array}{c|cc}
 & t_1 & t_2 \\
1 & \text{acq}(l) & \\
2 & \text{rel}(l) & \\
3 & w(x) & \\
4 & \text{r}(x) & \\
5 & \text{acq}(l) & \\
6 & \text{rel}(l) & \\
\end{array}
\]
Some Notations

Let $\sigma$ be an execution trace.

- TO$_\sigma$ = intra-thread ordering on events of $\sigma$
- RF$_\sigma$ : reads-from mapping of $\sigma$:
  - For a read event $r$, RF$_\sigma$(r) = w
  - For a release event rel, RF$_\sigma$(rel) = acq

<table>
<thead>
<tr>
<th></th>
<th>t1</th>
<th>t2</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>acq(1)</td>
<td></td>
</tr>
<tr>
<td>2</td>
<td>rel(1)</td>
<td></td>
</tr>
<tr>
<td>3</td>
<td>w(x)</td>
<td></td>
</tr>
<tr>
<td>4</td>
<td>r(x)</td>
<td></td>
</tr>
<tr>
<td>5</td>
<td>acq(1)</td>
<td></td>
</tr>
<tr>
<td>6</td>
<td>rel(1)</td>
<td></td>
</tr>
</tbody>
</table>
Some Notations

Let $\sigma$ be an execution trace.

- $\text{TO}_\sigma$ = intra-thread ordering on events of $\sigma$
- $\text{RF}_\sigma$ : reads-from mapping of $\sigma$:
  - For a read event $r$, $\text{RF}_\sigma(r) = w$
  - For a release event $\text{rel}$, $\text{RF}_\sigma(\text{rel}) = \text{acq}$
- $\text{TRF}_\sigma$ = smallest partial order that includes $\text{TO}_\sigma$ and respects $\text{RF}_\sigma$
**Some Notations**

Let $\sigma$ be an execution trace.

- $\text{TO}_\sigma$ = intra-thread ordering on events of $\sigma$
- $\text{RF}_\sigma$ : reads-from mapping of $\sigma$:
  - For a read event $r$, $\text{RF}_\sigma(r) = w$
  - For a release event $\text{rel}$, $\text{RF}_\sigma(\text{rel}) = \text{acq}$
- $\text{TRF}_\sigma$ = smallest partial order that includes $\text{TO}_\sigma$ and respects $\text{RF}_\sigma$

Let $\rho$ be a sequence of events with $\text{Events}(\rho) \subseteq \text{Events}(\sigma)$.

$\rho$ is a correct reordering of an execution trace $\sigma$ if

(i) For every lock $\ell$, there is at most one unmatched acquire of $\ell$ in $\rho$
(ii) $\rho$ respects $\text{TO}_\sigma$
(iii) $\text{RF}_\rho \subseteq \text{RF}_\sigma$
Trace Ideals and Realizability
Trace Ideals and Realizability

A feasible trace ideal $I$ of $\sigma$ is a set of events such that
A feasible trace ideal $I$ of $\sigma$ is a set of events such that
- $I$ is downward-closed under $\text{TRF}_\sigma$, and
Trace Ideals and Realizability

A feasible trace ideal $I$ of $\sigma$ is a set of events such that

- $I$ is downward-closed under $\text{TRF}_{\sigma}$, and
- for every lock $\ell$, there is at most one unmatched acquire of $\ell$ in $I$
A feasible trace ideal $I$ of $\sigma$ is a set of events such that

- $I$ is downward-closed under $\text{TRF}_\sigma$, and
- for every lock $\ell$, there is at most one unmatched acquire of $\ell$ in $I$
A feasible trace ideal $I$ of $\sigma$ is a set of events such that

- $I$ is downward-closed under $\text{TRF}_\sigma$, and
- for every lock $\ell$, there is at most one unmatched acquire of $\ell$ in $I$
A **feasible trace ideal** $I$ of $\sigma$ is a set of events such that

- $I$ is downward-closed under $\text{TRF}_\sigma$, and
- for every lock $\ell$, there is at most one unmatched acquire of $\ell$ in $I$. 

<table>
<thead>
<tr>
<th></th>
<th>t1</th>
<th>t2</th>
<th>t3</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td></td>
<td>acq(l)</td>
<td></td>
</tr>
<tr>
<td>2</td>
<td></td>
<td>w(x)</td>
<td></td>
</tr>
<tr>
<td>3</td>
<td></td>
<td>rel(l)</td>
<td></td>
</tr>
<tr>
<td>4</td>
<td></td>
<td></td>
<td>r(x)</td>
</tr>
<tr>
<td>5</td>
<td>acq(l)</td>
<td></td>
<td></td>
</tr>
<tr>
<td>6</td>
<td>rel(l)</td>
<td></td>
<td></td>
</tr>
<tr>
<td>7</td>
<td>w(y)</td>
<td></td>
<td></td>
</tr>
<tr>
<td>8</td>
<td></td>
<td></td>
<td>w(y)</td>
</tr>
</tbody>
</table>
A **feasible trace ideal** $I$ of $\sigma$ is a set of events such that

- $I$ is downward-closed under $\text{TRF}_\sigma$, and
- for every lock $\ell$, there is at most one unmatched acquire of $\ell$ in $I$

The **canonical rf-poset** of ideal $I$ is the smallest partial order $\mathcal{P}(I)$ such that
A feasible trace ideal $I$ of $\sigma$ is a set of events such that

1. $I$ is downward-closed under $\text{TRF}_\sigma$, and
2. for every lock $\ell$, there is at most one unmatched acquire of $\ell$ in $I$.

The canonical rf-poset of ideal $I$ is the smallest partial order $\mathcal{P}(I)$ such that

$\text{TRF}_\sigma \downarrow I \subseteq \mathcal{P}(I)$.
Trace Ideals and Realizability

A **feasible trace ideal** $I$ of $\sigma$ is a set of events such that

- $I$ is downward-closed under $\text{TRF}_\sigma$, and
- for every lock $\ell$, there is at most one unmatched acquire of $\ell$ in $I$

The **canonical rf-poset** of ideal $I$ is the smallest partial order $\mathcal{P}(I)$ such that

- $\text{TRF}_\sigma \downarrow I \subseteq \mathcal{P}(I)$
- For every lock $\ell$ with an unmatched acquire $\text{acq}_\text{unmatched} \in I$, and for every $\ell$-release event $\text{rel} \in I$, we have $\text{rel} \leq_{\mathcal{P}(I)} \text{acq}_\text{unmatched}$
Trace Ideals and Realizability

A **feasible trace ideal** $I$ of $\sigma$ is a set of events such that

- $I$ is downward-closed under $\text{TRF}_\sigma$, and
- for every lock $\ell$, there is at most one unmatched acquire of $\ell$ in $I$

The **canonical rf-poset** of ideal $I$ is the smallest partial order $\mathcal{P}(I)$ such that

- $\text{TRF}_\sigma \downarrow I \subseteq \mathcal{P}(I)$
- For every lock $\ell$ with an unmatched acquire $\text{acq}_{\text{unmatched}} \in I$, and for every $\ell$-release event $\text{rel} \in I$, we have $\text{rel} \leq \mathcal{P}(I) \text{acq}_{\text{unmatched}}$

### Realizability of Trace Ideals

Given a feasible trace ideal $I$ of $\sigma$, check if there is a linearization $\sigma^*$ of the canonical rf-poset $\mathcal{P}(I)$ such that $\text{RF}_{\sigma^*} \subseteq \text{RF}_{\sigma}$
Outline

1. Trace Ideals and Realizability

2. Algorithm for Data Race Prediction (General case)

3. Data Race Prediction for Acyclic Communication Topology

4. Distance Bounded Data Race Prediction

5. Lower Bound (General Case)

6. Lower Bound for 2 threads

7. Conclusions and Future Work
Data Race Prediction as Ideal Realizability
A pair of conflicting events \((e_1, e_2)\) is a predictable data race of \(\sigma\) iff there exists a feasible trace ideal \(I\) of \(\sigma\) such that \(I\) is realizable and both \(e_1\) and \(e_2\) are enabled in \(I\).
A pair of conflicting events \((e_1, e_2)\) is a predictable data race of \(\sigma\) iff
there exists a feasible trace ideal \(I\) of \(\sigma\) such that \(I\) is realizable
and both \(e_1\) and \(e_2\) are enabled in \(I\).

Proposition

If \(\sigma^*\) is a witness to the realizability of a feasible trace ideal \(I\) of \(\sigma\), then \(\sigma^*\) is a correct reordering of \(\sigma\).
A pair of conflicting events \((e_1, e_2)\) is a predictable data race of \(\sigma\) iff there exists a feasible trace ideal \(I\) of \(\sigma\) such that \(I\) is realizable and both \(e_1\) and \(e_2\) are enabled in \(I\).

**Proposition**

If \(\sigma^*\) is a witness to the realizability of a feasible trace ideal \(I\) of \(\sigma\), then \(\sigma^*\) is a correct reordering of \(\sigma\).

Event \(e\) of \(\sigma\) is enabled in a feasible trace ideal \(I\) of \(\sigma\) if \(I' = I \cup \{e\}\) is closed under \(\text{TO}_\sigma\).
A pair of conflicting events \((e_1, e_2)\) is a predictable data race of \(\sigma\) iff there exists a feasible trace ideal \(I\) of \(\sigma\) such that \(I\) is realizable and both \(e_1\) and \(e_2\) are enabled in \(I\).

**Proposition**

If \(\sigma^*\) is a witness to the realizability of a feasible trace ideal \(I\) of \(\sigma\), then \(\sigma^*\) is a correct reordering of \(\sigma\).

Event \(e\) of \(\sigma\) is enabled in a feasible trace ideal \(I\) of \(\sigma\) if \(I' = I \cup \{e\}\) is closed under \(\text{TO}_\sigma\).
Data Race Prediction as Ideal Realizability

A pair of conflicting events \( (e_1, e_2) \) is a predictable data race of \( \sigma \) iff there exists a feasible trace ideal \( I \) of \( \sigma \) such that \( I \) is \textit{realizable} and both \( e_1 \) and \( e_2 \) are \textit{enabled} in \( I \).

**Proposition**

If \( \sigma^* \) is a witness to the realizability of a feasible trace ideal \( I \) of \( \sigma \), then \( \sigma^* \) is a correct reordering of \( \sigma \).

Event \( e \) of \( \sigma \) is \textit{enabled} in a feasible trace ideal \( I \) of \( \sigma \) if \( I' = I \cup \{e\} \) is closed under \( \text{TO}_\sigma \).
Algorithm for Data Race Prediction
Algorithm for Data Race Prediction

**Execution** $\sigma$

Event pair $(e_1, e_2)$

Enumerate feasible ideals of $\sigma$ with $(e_1, e_2)$ enabled

Not realizable

Realizable

Ideal $I$

No more ideals

Check realizability of ideal $I$
Algorithm for Data Race Prediction

Execution $\sigma$

Event pair $(e_1, e_2)$

$O(\alpha)$ feasible ideals

Enumerate feasible ideals of $\sigma$ with $(e_1, e_2)$ enabled

Check realizability of ideal $I$

Ideal $I$

No more ideals

Not realizable

Realizable

NO

YES
Algorithm for Data Race Prediction

Execution $\sigma$

Event pair $(e_1, e_2)$

O($\alpha$) feasible ideals

Enumerate feasible ideals of $\sigma$ with $(e_1, e_2)$ enabled

Ideal $I$

Check realizability of ideal $I$

Realizable

Not realizable

No more ideals

NO

YES

Checking ideal realizability = $O(\beta)$
Algorithm for Data Race Prediction

Execution $\sigma$

Event pair $(e_1, e_2)$

\[ \text{O}(\alpha) \text{ feasible ideals} \]

Enumerate feasible ideals of $\sigma$ with $(e_1, e_2)$ enabled

Ideal $I$

Check realizability of ideal $I$

Realizable

Not realizable

No more ideals

\[ \text{Time complexity} = \text{O}(\alpha \cdot \beta) \]

Checking ideal realizability = $\text{O}(\beta)$
Algorithm for Data Race Prediction
Lemma

Let $\sigma$ be an execution with $n$ events, $k$ threads, lock nesting depth $\gamma$ and lock-dependence factor $\zeta$. Let $e_1$ and $e_2$ be events of $\sigma$. There are $O(\min(n, k \cdot \gamma \cdot \zeta)^{k-2})$ feasible ideals of $\sigma$ in which both $e_1$ and $e_2$ are enabled.
Algorithm for Data Race Prediction

Let $\sigma$ be an execution with $n$ events, $k$ threads, lock nesting depth $\gamma$ and lock-dependence factor $\zeta$. Let $e_1$ and $e_2$ be events of $\sigma$.

There are $O(\min(n, k \cdot \gamma \cdot \zeta)^{k-2})$ feasible ideals of $\sigma$ in which both $e_1$ and $e_2$ are enabled.
Algorithm for Data Race Prediction

Lemma

Let $\sigma$ be an execution with $n$ events, $k$ threads, lock nesting depth $\gamma$ and lock-dependence factor $\zeta$.

Let $e_1$ and $e_2$ be events of $\sigma$.

There are $O(\min(n, k \cdot \gamma \cdot \zeta)^{k-2})$ feasible ideals of $\sigma$ in which both $e_1$ and $e_2$ are enabled.

Lemma

Let $\sigma$ be an execution with $n$ events and $k$ threads and let $I$ be a feasible ideal of $\sigma$.

The realizability of $I$ can be checked in time $O(k \cdot n^k)$.
Algorithm for Data Race Prediction

**Lemma**

Let $\sigma$ be an execution with $n$ events, $k$ threads, lock nesting depth $\gamma$ and lock-dependence factor $\zeta$. Let $e_1$ and $e_2$ be events of $\sigma$. There are $O(\min(n, k\cdot\gamma\cdot\zeta)^{k-2})$ feasible ideals of $\sigma$ in which both $e_1$ and $e_2$ are enabled.

**Lemma**

Let $\sigma$ be an execution with $n$ events and $k$ threads and let $I$ be a feasible ideal of $\sigma$. The realizability of $I$ can be checked in time $O(k\cdot n^k)$.

**Theorem**

Let $\sigma$ be an execution with $n$ events, $k$ threads, lock nesting depth $\gamma$ and lock-dependence factor $\zeta$. Let $(e_1, e_2)$ be a conflicting pair of events of $\sigma$. The dynamic race prediction problem on can be solved in time $O(\min(n, k\cdot\gamma\cdot\zeta)^{k-2}\cdot k\cdot n^k)$.
Outline

1. Trace Ideals and Realizability
2. Algorithm for Data Race Prediction (General case)
3. **Data Race Prediction for Acyclic Communication Topology**
4. Distance Bounded Data Race Prediction
5. Lower Bound (General Case)
6. Lower Bound for 2 threads
7. Conclusions and Future Work
Tree Communication Topology
The communication topology of \( \sigma \) is an undirected graph \( G_\sigma = (V_\sigma, E_\sigma) \):

- \( V_\sigma \) = set of threads in \( \sigma \)
- \( E_\sigma = \{(t, t') \mid t \neq t' \text{ are threads that perform conflicting accesses or acquire a common lock}\} \)
The communication topology of $\sigma$ is an undirected graph $G_\sigma = (V_\sigma, E_\sigma)$:

- $V_\sigma = \text{set of threads in } \sigma$
- $E_\sigma = \{(t, t') \mid t \neq t' \text{ are threads that perform conflicting accesses or acquire a common lock}\}$
Acyclic Communication Topology
Acyclic Communication Topology

Server-client
Pipeline
Divide-and-conquer
Two threads
The realizability of a feasible ideal $I$ of an execution $\sigma$ (with $n$ events, $k$ threads and $d$ variables) having an acyclic communication topology can be determined in $O(k^2d^2n^2\log n)$ time.
The realizability of a feasible ideal $I$ of an execution $\sigma$ (with $n$ events, $k$ threads and $d$ variables) having an acyclic communication topology can be determined in $O(k^2d^2n^2\log n)$ time.

Lemma

Let $\sigma$ be an execution with an acyclic communication topology and let $e_1$ and $e_2$ be events of $\sigma$. There is a feasible ideal $I_{(e_1,e_2)}$ such that $I_{(e_1,e_2)}$ is realizable iff $(e_1, e_2)$ is a predictable data race of $\sigma$. Lemma
Acyclic Communication Topology

Server-client
Pipeline
Divide-and-conquer
Two threads

\[ \sigma(e_1, e_2) \]

Enumerate feasible ideals of \( \sigma \) with \((e_1, e_2)\) enabled

Check realizability of ideal \( I \)

Ideal \( I \)

No more ideals

Not realizable

Realizable

NO

YES
Acyclic Communication Topology

- Server-client
- Pipeline
- Divide-and-conquer
- Two threads

Ideal $I$ only admits one feasible ideal $(e_1, e_2)$.

- Check realizability of ideal $I$.
- If not realizable, no more ideals.
- If realizable, $O(k^2 \cdot d^2 \cdot n^2 \cdot \log n)$ time for checking realizability.

- Only 1 feasible ideal is sufficient.
Acyclic Communication Topology

Server-client
Pipeline
Divide-and-conquer
Two threads

Time complexity when execution has acyclic topology = $O(k^2 d^2 n^2 \log n)$

Only 1 feasible ideal is sufficient

Enumerate feasible ideals of $\sigma$ with $(e_1, e_2)$ enabled

Ideal I

Check realizability of ideal I

Realizable

Not realizable

No more ideals

$\sigma$ 

$(e_1, e_2)$
Outline

1. Trace Ideals and Realizability
2. Algorithm for Data Race Prediction (General case)
3. Data Race Prediction for Acyclic Communication Topology
4. **Distance Bounded Data Race Prediction**
5. Lower Bound (General Case)
6. Lower Bound for 2 threads
7. Conclusions and Future Work
Distance-Bounded Data Races
Distance-Bounded Data Races

Set of reversals between $\sigma$ and its correct reordering $\rho$: 
Distance-Bounded Data Races

Set of reversals between $\sigma$ and its correct reordering $\rho$:

$$\text{Rev}(\sigma, \rho) = \{(e_1, e_2) \mid e_1 \text{ and } e_2 \text{ are conflicting writes or acquire of same lock such that}$$

$$e_1 \leq_\sigma e_2 \text{ and } e_2 \leq_\rho e_1\}$$
Distance-Bounded Data Races

**Set of reversals** between $\sigma$ and its correct reordering $\rho$:

$$\text{Rev}(\sigma, \rho) = \{(e_1, e_2) \mid e_1 \text{ and } e_2 \text{ are conflicting writes or acquire of same lock such that}$$

$$e_1 \leq_\sigma e_2 \text{ and } e_2 \leq_\rho e_1\}$$

**Distance** between $\sigma$ and $\rho$ is $\delta(\sigma, \rho) = |\text{Rev}(\sigma, \rho)|$
Distance-Bounded Data Races

**Set of reversals** between $\sigma$ and its correct reordering $\rho$:

$$\text{Rev}(\sigma, \rho) = \{(e_1, e_2) \mid e_1 \text{ and } e_2 \text{ are conflicting writes or acquires of same lock such that } e_1 \leq_\sigma e_2 \text{ and } e_2 \leq_\rho e_1\}$$

**Distance** between $\sigma$ and $\rho$ is $\delta(\sigma, \rho) = |\text{Rev}(\sigma, \rho)|$

---

<table>
<thead>
<tr>
<th></th>
<th>t1</th>
<th>t2</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>acq(1)</td>
<td></td>
</tr>
<tr>
<td>2</td>
<td>rel(1)</td>
<td></td>
</tr>
<tr>
<td>3</td>
<td>w(x)</td>
<td></td>
</tr>
<tr>
<td>4</td>
<td></td>
<td>w(x)</td>
</tr>
<tr>
<td>5</td>
<td></td>
<td>acq(1)</td>
</tr>
<tr>
<td>6</td>
<td>rel(1)</td>
<td></td>
</tr>
</tbody>
</table>

---

<table>
<thead>
<tr>
<th></th>
<th>t1</th>
<th>t2</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td></td>
<td>w(x)</td>
</tr>
<tr>
<td>2</td>
<td></td>
<td>acq(1)</td>
</tr>
<tr>
<td>3</td>
<td>w(x)</td>
<td></td>
</tr>
<tr>
<td>4</td>
<td>acq(1)</td>
<td></td>
</tr>
<tr>
<td>5</td>
<td>rel(1)</td>
<td></td>
</tr>
<tr>
<td>6</td>
<td>w(x)</td>
<td></td>
</tr>
</tbody>
</table>
Distance-Bounded Data Races

**Set of reversals** between $\sigma$ and its correct reordering $\rho$:

$$\text{Rev}(\sigma, \rho) = \{(e_1, e_2) \mid e_1 \text{ and } e_2 \text{ are conflicting writes or acquires of same lock such that } e_1 \leq_\sigma e_2 \text{ and } e_2 \leq_\rho e_1\}$$

**Distance** between $\sigma$ and $\rho$ is $\delta(\sigma, \rho) = |\text{Rev}(\sigma, \rho)|$
Distance-Bounded Data Races

Set of reversals between $\sigma$ and its correct reordering $\rho$:

$$\text{Rev}(\sigma, \rho) = \{(e_1, e_2) \mid e_1 \text{ and } e_2 \text{ are conflicting writes or acquires of same lock such that } e_1 \leq_\sigma e_2 \text{ and } e_2 \leq_\rho e_1\}$$

Distance between $\sigma$ and $\rho$ is $\delta(\sigma, \rho) = |\text{Rev}(\sigma, \rho)|$
Distance-Bounded Data Races

**Set of reversals** between $\sigma$ and its correct reordering $\rho$: 
\[
\text{Rev}(\sigma, \rho) = \{(e_1, e_2) \mid e_1 \text{ and } e_2 \text{ are conflicting writes or acquires of same lock such that } e_1 \leq_{\sigma} e_2 \text{ and } e_2 \leq_{\rho} e_1\}
\]

**Distance** between $\sigma$ and $\rho$ is $\delta(\sigma, \rho) = |\text{Rev}(\sigma, \rho)|$

<table>
<thead>
<tr>
<th>t1</th>
<th>t2</th>
</tr>
</thead>
<tbody>
<tr>
<td>1 acq(l)</td>
<td></td>
</tr>
<tr>
<td>2 rel(l)</td>
<td></td>
</tr>
<tr>
<td>3 w(x)</td>
<td></td>
</tr>
<tr>
<td>4 w(x)</td>
<td></td>
</tr>
<tr>
<td>5 acq(l)</td>
<td></td>
</tr>
<tr>
<td>6 rel(l)</td>
<td></td>
</tr>
</tbody>
</table>

Let $\ell \in \mathbb{N}$ be a constant. Given an execution $\sigma$ and pair of conflicting events $(e_1, e_2)$ of $\sigma$, the answer to the $\ell$-bounded data race prediction problem is...
Distance-Bounded Data Races

Set of reversals between $\sigma$ and its correct reordering $\rho$:

$$\text{Rev}(\sigma, \rho) = \{(e_1, e_2) \mid e_1 \text{ and } e_2 \text{ are conflicting writes or acquires of same lock such that } e_1 \leq_\sigma e_2 \text{ and } e_2 \leq_\rho e_1\}$$

Distance between $\sigma$ and $\rho$ is $\delta(\sigma, \rho) = |\text{Rev}(\sigma, \rho)|$

Let $\ell \in \mathbb{N}$ be a constant. Given an execution $\sigma$ and pair of conflicting events $(e_1, e_2)$ of $\sigma$, the answer to the $\ell$-bounded data race prediction problem is

- NO if $(e_1, e_2)$ is not a predictable race of $\sigma$
Distance-Bounded Data Races

Set of reversals between \( \sigma \) and its correct reordering \( \rho \):

\[
\text{Rev}(\sigma, \rho) = \{(e_1, e_2) \mid e_1 \text{ and } e_2 \text{ are conflicting writes or acquire of same lock such that } e_1 \leq_{\sigma} e_2 \text{ and } e_2 \leq_{\rho} e_1\}
\]

Distance between \( \sigma \) and \( \rho \) is \( \delta(\sigma, \rho) = |\text{Rev}(\sigma, \rho)| \)

Let \( \ell \in \mathbb{N} \) be a constant. Given an execution \( \sigma \) and pair of conflicting events \((e_1, e_2)\) of \( \sigma \), the answer to the \( \ell \)-bounded data race prediction problem is

- NO if \((e_1, e_2)\) is not a predictable race of \( \sigma \)
- YES, if there is a correct reordering \( \sigma^* \) of \( \sigma \) witnessing the race \((e_1, e_2)\) such that \( \delta(\sigma, \sigma^*) \leq \ell \)
**Distance-Bounded Data Races**

**Set of reversals** between $\sigma$ and its correct reordering $\rho$:

$$\text{Rev}(\sigma, \rho) = \{(e_1, e_2) \mid e_1 \text{ and } e_2 \text{ are conflicting writes or acquires of same lock such that } e_1 \leq_\sigma e_2 \text{ and } e_2 \leq_\rho e_1\}$$

**Distance** between $\sigma$ and $\rho$ is $\delta(\sigma, \rho) = |\text{Rev}(\sigma, \rho)|$

Let $\ell \in \mathbb{N}$ be a constant. Given an execution $\sigma$ and pair of conflicting events $(e_1, e_2)$ of $\sigma$, the answer to the $\ell$-bounded data race prediction problem is:

- **NO** if $(e_1, e_2)$ is not a predictable race of $\sigma$
- **YES**, if there is a correct reordering $\sigma^*$ of $\sigma$ witnessing the race $(e_1, e_2)$ such that $\delta(\sigma, \sigma^*) \leq \ell$
- **YES or NO**, otherwise
Distance-Bounded Data Races
Distance-Bounded Data Races

Let $\ell \in \mathbb{N}$ be a constant. Given an execution $\sigma$ and a feasible ideal $I$ of $\sigma$, the answer to the $\ell$-bounded realizability problem for $I$ is

- NO if $I$ is not realizable
- YES, if there is a witness $\sigma^*$ of realizability of $I$ such that $\delta(\sigma, \sigma^*) \leq \ell$
- YES or NO, otherwise
Distance-Bounded Data Races

Let $\ell \in \mathbb{N}$ be a constant. Given an execution $\sigma$ and a feasible ideal $I$ of $\sigma$, the answer to the $\ell$-bounded realizability problem for $I$ is

- NO if $I$ is not realizable
- YES, if there is a witness $\sigma^*$ of realizability of $I$ such that $\delta(\sigma, \sigma^*) \leq \ell$
- YES or NO, otherwise

Iterate over all possible subsets of events of $I$ of size $\leq \ell$ of pairs of conflicting writes or lock acquires and invert them.

$O(|I|^{2\ell}) = O(n^{2\ell})$
Distance-Bounded Data Races
Distance-Bounded Data Races

Lemma

Let $\ell \in \mathbb{N}$ be a constant. Let $\sigma$ be an execution with $n$ events and $k$ threads and let $I$ be a feasible ideal of $\sigma$. The $\ell$-bounded realizability of $I$ can be checked in time $O(k^{\ell+O(1)} \cdot n)$. 


Distance-Bounded Data Races

Lemma

Let $\ell \in \mathbb{N}$ be a constant. Let $\sigma$ be an execution with $n$ events and $k$ threads and let $I$ be a feasible ideal of $\sigma$. The $\ell$-bounded realizability of $I$ can be checked in time $O(k\ell + O(1) \cdot n)$.
Distance-Bounded Data Races

Let $\ell \in \mathbb{N}$ be a constant. Let $\sigma$ be an execution with $n$ events and $k$ threads and let $I$ be a feasible ideal of $\sigma$.

The $\ell$-bounded realizability of $I$ can be checked in time $O(k\ell + O(1) \cdot n)$. 

<table>
<thead>
<tr>
<th>Ideal $I$</th>
</tr>
</thead>
<tbody>
<tr>
<td>Enumerate feasible ideals of $\sigma$ with $(e_1, e_2)$ enabled</td>
</tr>
<tr>
<td>Check realizability of ideal $I$</td>
</tr>
<tr>
<td>Not realizable</td>
</tr>
<tr>
<td>Realizable</td>
</tr>
</tbody>
</table>

$O(1)$ ideals when $\gamma$ and $\zeta$ are constant

No more ideals

$\text{NO}$

$\text{YES}$
Distance-Bounded Data Races

Let $\ell \in \mathbb{N}$ be a constant. Let $\sigma$ be an execution with $n$ events and $k$ threads and let $I$ be a feasible ideal of $\sigma$.

The $\ell$-bounded realizability of $I$ can be checked in time $O(k\ell + O(1)\cdot n)$.

Lemma

Time complexity of $\ell$-bounded data race prediction = $O(k\ell + O(1)\cdot n)$
Outline

1. Trace Ideals and Realizability
2. Algorithm for Data Race Prediction (General case)
3. Data Race Prediction for Acyclic Communication Topology
4. Distance Bounded Data Race Prediction
5. **Lower Bound (General Case)**
6. Lower Bound for 2 threads
7. Conclusions and Future Work
Lower Bound for Data Race Prediction
Lower Bound for Data Race Prediction

- When \( k \) (number of threads) is constant, data race prediction is in \( \mathcal{P} \).
• When $k$ (number of threads) is constant, data race prediction is in $P$.
• Is the problem in $P$ even otherwise?
Lower Bound for Data Race Prediction

- When $k$ (number of threads) is constant, data race prediction is in $\mathbf{P}$.
- Is the problem in $\mathbf{P}$ even otherwise?
- Is the problem in $\mathbf{FPT}$ with $k$ as a parameter?
Lower Bound for Data Race Prediction

- When $k$ (number of threads) is constant, data race prediction is in $\mathbb{P}$.
- Is the problem in $\mathbb{P}$ even otherwise?
- Is the problem in $\mathbb{FPT}$ with $k$ as a parameter?

Can be solved in time $O(f(k) \cdot n^c)$?
Lower Bound for Data Race Prediction

• When $k$ (number of threads) is constant, data race prediction is in $\text{P}$. 
• Is the problem in $\text{P}$ even otherwise?
• Is the problem in $\text{FPT}$ with $k$ as a parameter?

Data Race Prediction and ideal realizability are $\textbf{W}[1]$-hard in the parameter $k$ (number of threads).
Lower Bound for Data Race Prediction

- When $k$ (number of threads) is constant, data race prediction is in $\mathbf{P}$.
- Is the problem in $\mathbf{P}$ even otherwise?
- Is the problem in $\mathbf{FPT}$ with $k$ as a parameter?

Data Race Prediction and ideal realizability are $\mathbf{W[1]}$-hard in the parameter $k$ (number of threads).

- $\mathbf{FPT}$-reduction from INDEPENDENT-SET($c$)
- Reduction takes time $O(\text{poly}(n+k))$ time
- $\mathbf{NP}$-hardness follows
- Improves previous result [Gibbons and Korach, ’97]
Lower Bound for Data Race Prediction

- When $k$ (number of threads) is constant, data race prediction is in $\mathbf{P}$.
- Is the problem in $\mathbf{P}$ even otherwise?
- Is the problem in $\mathbf{FPT}$ with $k$ as a parameter?

Can be solved in time $O(f(k) \cdot n^c)$?

**Theorem**

Data Race Prediction and ideal realizability are $\mathbf{W[1]}$-hard in the parameter $k$ (number of threads).

- $\mathbf{FPT}$-reduction from INDEPENDENT-SET($c$)
- Reduction takes time $O(\text{poly}(n+k))$ time
- $\mathbf{NP}$-hardness follows
- Improves previous result [Gibbons and Korach, ’97]

Unlikely, unless ETH fails

ETH implies $\mathbf{FPT} \subsetneq \mathbf{W[1]}$
Outline

1. Trace Ideals and Realizability
2. Algorithm for Data Race Prediction (General case)
3. Data Race Prediction for Acyclic Communication Topology
4. Distance Bounded Data Race Prediction
5. Lower Bound (General Case)
6. Lower Bound for 2 threads
7. Conclusions and Future Work
Lower Bound for 2 Threads
Lower Bound for 2 Threads

Let $\sigma$ be an execution with $n$ events, $k \geq 2$ threads and $d \geq 9$ variables (and $\geq 1$ lock).

There is no algorithm that solves the data race prediction problem for in time $O(n^{2-\epsilon})$, for any $\epsilon > 0$, unless the Orthogonal Vectors conjecture fails.
Lower Bound for 2 Threads

Let $\sigma$ be an execution with $n$ events, $k \geq 2$ threads and $d \geq 9$ variables (and $\geq 1$ lock).

There is no algorithm that solves the data race prediction problem for in time $O(n^{2-\epsilon})$, for any $\epsilon > 0$, unless the Orthogonal Vectors conjecture fails.

Same lower bound for ideal realizability
Lower Bound for 2 Threads

Let $\sigma$ be an execution with $n$ events, $k \geq 2$ threads and $d \geq 9$ variables (and $\geq 1$ lock). There is no algorithm that solves the data race prediction problem for in time $O(n^{2-\epsilon})$, for any $\epsilon > 0$, unless the Orthogonal Vectors conjecture fails.

Theorem

Same lower bound for ideal realizability

SETH implies Orthogonal Vectors Conjecture
Let $\sigma$ be an execution with $n$ events, $k \geq 2$ threads and $d \geq 9$ variables (and $\geq 1$ lock). There is no algorithm that solves the data race prediction problem for in time $O(n^{2-\epsilon})$, for any $\epsilon > 0$, unless the Orthogonal Vectors conjecture fails.

**Theorem**

Same lower bound for ideal realizability

SETH implies Orthogonal Vectors Conjecture

Applies to acyclic topology
Outline

1. Trace Ideals and Realizability
2. Algorithm for Data Race Prediction (General case)
3. Data Race Prediction for Acyclic Communication Topology
4. Distance Bounded Data Race Prediction
5. Lower Bound (General Case)
6. Lower Bound for 2 threads

7. Conclusions and Future Work
Conclusions
Conclusions

Summary
Conclusions

Summary

• Dynamic Race Prediction
  • Complexity-theoretic investigation
Conclusions

Summary

• Dynamic Race Prediction
  • Complexity-theoretic investigation

• Trace Ideals and Realizability
Conclusions

Summary

• Dynamic Race Prediction
  • Complexity-theoretic investigation
• Trace Ideals and Realizability
• General case:
  • $O(kn^{2k-2})$ time algorithm
  • $O(k \cdot n^k)$ when $\gamma$ and $\zeta$ are constant
  • $W[1]$-hardness
  • Quadratic lower bound for $k=2$ threads
Conclusions

Summary

• Dynamic Race Prediction
  • Complexity-theoretic investigation

• Trace Ideals and Realizability

• General case:
  • $O(kn^{2k-2})$ time algorithm
  • $O(k \cdot n^k)$ when $\gamma$ and $\zeta$ are constant
  • $\mathbf{W}[1]$-hardness
  • Quadratic lower bound for $k=2$ threads

• Acyclic topology
  • Quadratic time upper and lower bounds
Conclusions

Summary

- Dynamic Race Prediction
  - Complexity-theoretic investigation
- Trace Ideals and Realizability
- General case:
  - $O(kn^{2k-2})$ time algorithm
  - $O(k \cdot n^k)$ when $\gamma$ and $\zeta$ are constant
  - $W[1]$-hardness
  - Quadratic lower bound for $k=2$ threads
- Acyclic topology
  - Quadratic time upper and lower bounds
- Distance Bounded data race prediction
  - Linear time realizability
Conclusions

Summary

• Dynamic Race Prediction
  • Complexity-theoretic investigation

• Trace Ideals and Realizability

• General case:
  • $O(kn^{2k-2})$ time algorithm
  • $O(k \cdot n^k)$ when $\gamma$ and $\zeta$ are constant
  • $\text{W}[1]$-hardness
  • Quadratic lower bound for $k=2$ threads

• Acyclic topology
  • Quadratic time upper and lower bounds

Future Work

• Distance Bounded data race prediction
  • Linear time realizability
Conclusions

Summary

- Dynamic Race Prediction
  - Complexity-theoretic investigation
- Trace Ideals and Realizability
- General case:
  - $O(kn^{2k-2})$ time algorithm
  - $O(k \cdot n^k)$ when $\gamma$ and $\zeta$ are constant
  - $W[1]$-hardness
  - Quadratic lower bound for $k=2$ threads
- Acyclic topology
  - Quadratic time upper and lower bounds
- Distance Bounded data race prediction
  - Linear time realizability

Future Work

- Tighter bounds
  - Is data race prediction $W[k]$-complete for some $k$?
- Space-time tradeoffs
Conclusions

Summary

- Dynamic Race Prediction
  - Complexity-theoretic investigation
- Trace Ideals and Realizability
- General case:
  - $O(kn^{2k-2})$ time algorithm
  - $O(k \cdot n^k)$ when $\gamma$ and $\zeta$ are constant
  - $\mathcal{W}[1]$-hardness
  - Quadratic lower bound for $k=2$ threads
- Acyclic topology
  - Quadratic time upper and lower bounds
- Distance Bounded data race prediction
  - Linear time realizability

Future Work

- Tighter bounds
  - Is data race prediction $\mathcal{W}[k]$-complete for some $k$?
  - Space-time tradeoffs
- Other algorithmic paradigms
  - Randomized algorithms
  - Parallel algorithms
Conclusions

Summary

- Dynamic Race Prediction
  - Complexity-theoretic investigation
- Trace Ideals and Realizability
- General case:
  - $O(kn^{2k-2})$ time algorithm
  - $O(k \cdot n^k)$ when $\gamma$ and $\zeta$ are constant
  - $W[1]$-hardness
  - Quadratic lower bound for $k=2$ threads
- Acyclic topology
  - Quadratic time upper and lower bounds
- Distance Bounded data race prediction
  - Linear time realizability

Future Work

- Tighter bounds
  - Is data race prediction $W[k]$-complete for some $k$?
  - Space-time tradeoffs
- Other algorithmic paradigms
  - Randomized algorithms
  - Parallel algorithms
- Practicability constraints -
  - Linear time algorithms
  - (Sub-)linear dependence on $d$
  - (Sub-)linear space overhead
Thank You!