Dynamic Data-Race Prediction

Fundamentals, Theory and Practice

Umang Mathur and **Andreas Pavlogiannis**

UNIVERSITY OF ILLINOIS URBANA-CHAMPAIGN

AARHUS UNIVERSITY

Welcome!

- The tutorial is recorded using Zoom
- Streamed on '**ACM SIGPLAN**' YouTube channel
- Multiple sections with Q&A breaks after each section
- Please be muted outside Q&A
- If your question can't wait, raise hand in zoom
    - Click "Reactions", then "raise hand" (✋)
- You can also ask questions in chat
    - Zoom chat is disabled – use the clowdr chat in our tutorial's room
- Tutorial slides & related material accessible on the tutorial's web-page (on `popl21.sigplan.org`)

Outline

Introduction

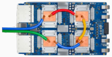
Concurrency : Software and Challenges

- Ubiquitous in modern software

Concurrency : Software and Challenges

- Ubiquitous in modern software

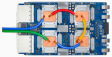
- Back-bone of big-data, AI revolutions



Concurrency : Software and Challenges

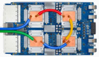
- Ubiquitous in modern software

- Back-bone of big-data, AI revolutions



Challenging to write concurrent software

- Ubiquitous in modern software
- Back-bone of big-data, AI revolutions

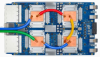


Challenging to write concurrent software

- Large interleaving space

Concurrency : Software and Challenges

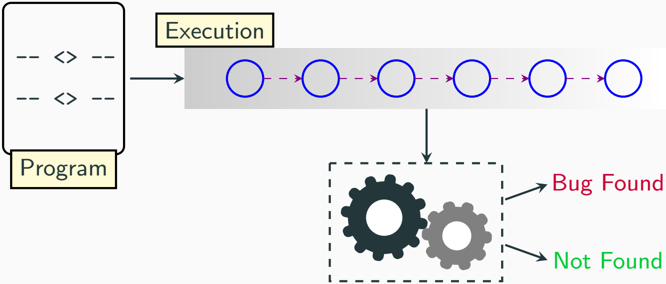
- Ubiquitous in modern software
- Back-bone of big-data, AI revolutions



Challenging to write concurrent software

- Large interleaving space
- Concurrency bugs
  - data races, deadlocks, etc.,
  - manifest in production
  - despite rigorous testing
  - severe outcomes

Dynamic Analysis for Detecting Concurrency Bugs



What constitutes a good dynamic analysis?

Three desirable properties of an effective dynamic analysis:

What constitutes a good dynamic analysis?

Three desirable properties of an effective dynamic analysis:

(1) Scalability to real world applications

- executions with billions of events.

What constitutes a good dynamic analysis?

Three desirable properties of an effective dynamic analysis:

(1) Scalability to real world applications

- executions with billions of events.

(2) Soundness, or, no false positives

- otherwise users need to manually confirm each reported bug.

What constitutes a good dynamic analysis?

Three desirable properties of an effective dynamic analysis:

(1) Scalability to real world applications

- executions with billions of events.

(2) Soundness, or, no false positives

- otherwise users need to manually confirm each reported bug.

(3) High predictive power (not miss many bugs)

- high coverage.

This Tutorial

In this tutorial, we will discuss

- foundations of dynamic data race detection,
- recent algorithmic advances in effective race detection, and
- the fundamental limits in dynamic detection of data races.

This Tutorial

In this tutorial, we will discuss

- foundations of dynamic data race detection,
- recent algorithmic advances in effective race detection, and
- the fundamental limits in dynamic detection of data races.

This Tutorial

In this tutorial, we will discuss

- foundations of dynamic data race detection,
- recent algorithmic advances in effective race detection, and
- the fundamental limits in dynamic detection of data races.

This Tutorial

In this tutorial, we will discuss

- foundations of dynamic data race detection,
- recent algorithmic advances in effective race detection, and
- the fundamental limits in dynamic detection of data races.

Preliminaries

Setting - Concurrent programs

```
public class Test extends Thread{

    static int x,y;
    static Object lock;
    public int id;

    @Override
    public void run() {
        if(id == 1){
            y = x + 5;
        }
        else{
            synchronized(lock){
                y = x + 10;
            }
        }
    }

    public static void main(String args[])
      throws Exception {
        final Test t1 = new Test();
        final Test t2 = new Test();
        t1.id = 1;
        t2.id = 2;
        t1.start();
        t2.start();
        t1.join();
        t2.join();
    }
}
```

Concurrent Program

- Threads
- Local variables
- Shared memory
- Synchronization constructs (locks)
- Sequential consistency

Setting: Execution Traces

- Sequences of events

	t_1	t_2
1	acq(ℓ)	
2	$r(x)$	
3	$w(y)$	
4		$r(x)$
5	rel(ℓ)	
6		acq(ℓ)
7		$w(y)$
8		acq(ℓ)

Setting: Execution Traces

- Sequences of events
- Event $e = \langle t, op \rangle$.

	t_1	t_2
1	$\mathsf{acq}(\ell)$	
2	$r(x)$	
3	$w(y)$	
4		$r(x)$
5	$\mathsf{rel}(\ell)$	
6		$\mathsf{acq}(\ell)$
7		$w(y)$
8		$\mathsf{acq}(\ell)$

Setting: Execution Traces

- Sequences of events
- Event $e = \langle t, op \rangle$.
- op can be

	t_1	t_2
1	acq(ℓ)	
2	$r(x)$	
3	$w(y)$	
4		$r(x)$
5	rel(ℓ)	
6		acq(ℓ)
7		$w(y)$
8		acq(ℓ)

Setting: Execution Traces

- Sequences of events
- Event $e = \langle t, op \rangle$.
- op can be
  - $r(\cdot)$, $w(\cdot)$

	t_1	t_2
1	acq(ℓ)	
2	$r(x)$	
3	$w(y)$	
4		$r(x)$
5	rel(ℓ)	
6		acq(ℓ)
7		$w(y)$
8		acq(ℓ)

Setting: Execution Traces

- Sequences of events
- Event $e = \langle t, op \rangle$.
- *op* can be
  - $r(\cdot)$, $w(\cdot)$
  - acq$(\cdot)$, rel$(\cdot)$

	t_1	t_2
1	acq(ℓ)	
2	$r(x)$	
3	$w(y)$	
4		$r(x)$
5	rel(ℓ)	
6		acq(ℓ)
7		$w(y)$
8		acq(ℓ)

Setting: Execution Traces

- Sequences of events
- Event $e = \langle t, op \rangle$.
- *op* can be
  - $r(\cdot)$, $w(\cdot)$
  - acq$(\cdot)$, rel$(\cdot)$
- Well formedness - critical sections on the same lock do not overlap

	t_1	t_2
1	acq(ℓ)	
2	$r(x)$	
3	$w(y)$	
4		$r(x)$
5	rel(ℓ)	
6		acq(ℓ)
7		$w(y)$
8		acq(ℓ)

Setting: Execution Traces

- Sequences of events
- Event $e = \langle t, op \rangle$.
- op can be
  - $r(\cdot)$, $w(\cdot)$
  - acq$(\cdot)$, rel$(\cdot)$
- Well formedness - critical sections on the same lock do not overlap

	t_1	t_2
1	acq(ℓ)	
2	$r(x)$	
3	$w(y)$	
4		$r(x)$
5	rel(ℓ)	
6		acq(ℓ)
7		$w(y)$
8		acq(ℓ)

Data Races

A data race in σ is a pair (e_1, e_2) of events such that

- e_1 and e_2 are conflicting
  - access same memory location x
  - at least one writes to x
- e_1 and e_2 are concurrent
  - e_1 and e_2 are simultaneously enabled in some prefix π of σ

	t_1	t_2
1	acq(ℓ)	
2	$w(y)$	
3	$w(x)$	
4		$r(x)$
5	rel(ℓ)	
6		acq(ℓ)
7		$w(y)$
8		acq(ℓ)

Data Races

A data race in σ is a pair (e_1, e_2) of events such that

- e_1 and e_2 are conflicting
  - access same memory location x
  - at least one writes to x
- e_1 and e_2 are concurrent
  - e_1 and e_2 are simultaneously enabled in some prefix π of σ

	t_1	t_2
1	$\mathrm{acq}(\ell)$	
2	$w(y)$	
3	$w(x)$	
4		$r(x)$
5	$\mathrm{rel}(\ell)$	
6		$\mathrm{acq}(\ell)$
7		$w(y)$
8		$\mathrm{acq}(\ell)$

Data Races

A data race in σ is a pair (e_1, e_2) of events such that

- e_1 and e_2 are conflicting
  - access same memory location x
  - at least one writes to x
- e_1 and e_2 are concurrent
  - e_1 and e_2 are simultaneously enabled in some prefix π of σ

	t_1	t_2
1	acq(ℓ)	
2	$w(y)$	
3	$w(x)$	
4		$r(x)$
5	rel(ℓ)	
6		acq(ℓ)
7		$w(y)$
8		acq(ℓ)

How to detect data races dynamically?

Detecting Data Races

How to detect data races dynamically?

- Execute a program, check if it witnesses a data race

Detecting Data Races

How to detect data races dynamically?

- Execute a program, check if it witnesses a data race

	t_1	t_2
1	$\mathsf{acq}(\ell)$	
2	$r(x)$	
3		$w(z)$
4	$\mathsf{rel}(\ell)$	
5		$w(x)$

Execution σ

Detecting Data Races

How to detect data races dynamically?

- Execute a program, check if it witnesses a data race

	t_1	t_2
1	acq(ℓ)	
2	$r(x)$	
3		$w(z)$
4	rel(ℓ)	
5		$w(x)$

Execution σ

No data race

Detecting Data Races

How to detect data races dynamically?

- Execute a program, check if it witnesses a data race

	t_1	t_2
1	acq(ℓ)	
2	$r(x)$	
3		$w(z)$
4	rel(ℓ)	
5		$w(x)$

Execution σ

No data race

	t_1	t_2
1	acq(ℓ)	
2		$w(z)$
3	$r(x)$	
4		$w(x)$
5	rel(ℓ)	

Execution σ'

Detecting Data Races

How to detect data races dynamically?

- Execute a program, check if it witnesses a data race

	t_1	t_2
1	$\mathsf{acq}(\ell)$	
2	$r(x)$	
3		$w(z)$
4	$\mathsf{rel}(\ell)$	
5		$w(x)$

Execution σ

No data race

	t_1	t_2
1	$\mathsf{acq}(\ell)$	
2		$w(z)$
3	$r(x)$	
4		$w(x)$
5	$\mathsf{rel}(\ell)$	

Execution σ'

Data race exists

Detecting Data Races

How to detect data races dynamically?

- Execute a program, check if it witnesses a data race

	t_1	t_2
1	$\mathsf{acq}(\ell)$	
2	$r(x)$	
3		$w(z)$
4	$\mathsf{rel}(\ell)$	
5		$w(x)$

Execution σ

No data race

	t_1	t_2
1	$\mathsf{acq}(\ell)$	
2		$w(z)$
3	$r(x)$	
4		$w(x)$
5	$\mathsf{rel}(\ell)$	

Execution σ'

Data race exists

Executions can be reordered to expose data races!

Which reorderings are allowed?

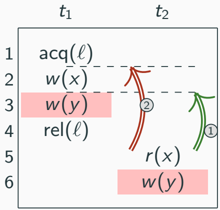
	t_1	t_2
1	$\mathsf{acq}(\ell)$	
2	$w(x)$	
3	$w(y)$	
4	$\mathsf{rel}(\ell)$	
5		$r(x)$
6		$w(y)$

Which reorderings are allowed?

	t_1	t_2
1	acq(ℓ)	
2	$w(x)$	
3	$w(y)$	
4	rel(ℓ)	
5		$r(x)$
6		$w(y)$

Reordering 1 ✓

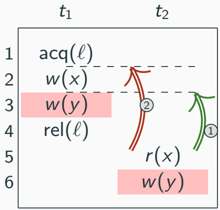
Which reorderings are allowed?



Reordering ① ✓

Reordering ② ✗

Which reorderings are allowed?



Reordering 1 ✓

Reordering 2 ✗

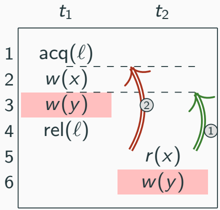
t_1	t_2

`synchronized(ℓ){`
`  x = 42;`
`  y = 1;`
`}`
	`if(x == 42){`
	`  y = 2;`
	`}`

Possible source program

(Initially, `x = 0`, `y = 0`)

Which reorderings are allowed?



Reordering 1 ✓

Reordering 2 ✗

t_1

```
synchronized(ℓ){
  x = 42;
  y = 1;
}
```

t_2

```
if(x == 42){
  y = 2;
}
```

Possible source program

(Initially, `x = 0`, `y = 0`)

Source agnostic analysis - some reorderings are not allowed

Formalizing Dynamic Predictive Analysis

Principle behind predictive analysis

A reordering σ^* of an observed execution σ is allowed if any program P that generates σ can also generate σ^*

Formalizing Dynamic Predictive Analysis

Principle behind predictive analysis

A reordering σ^* of an observed execution σ is allowed if any program P that generates σ can also generate σ^*

How do we formalize predictive analysis?

Formalizing Dynamic Predictive Analysis

Principle behind predictive analysis

A reordering σ^* of an observed execution σ is allowed if any program P that generates σ can also generate σ^*

How do we formalize predictive analysis?

- Semantics of underlying programming language

Formalizing Dynamic Predictive Analysis

Principle behind predictive analysis

A reordering σ^* of an observed execution σ is allowed if any program P that generates σ can also generate σ^*

How do we formalize predictive analysis?

- Semantics of underlying programming language
- Properties of concurrent objects and operations on them [HW90]
  - locks, shared variables, threads

Formalizing Dynamic Predictive Analysis

Principle behind predictive analysis

A reordering σ^* of an observed execution σ is allowed if any program P that generates σ can also generate σ^*

How do we formalize predictive analysis?

- Semantics of underlying programming language
- Properties of concurrent objects and operations on them [HW90]
  - locks, shared variables, threads
- Local determinism and prefix closedness [ŞCR13]

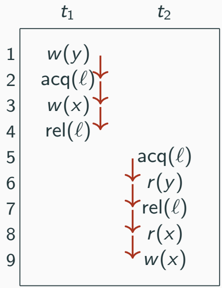
Some Notation

- Events_σ - set of events of σ.

	t_1	t_2
1	$w(y)$	
2	$\text{acq}(\ell)$	
3	$w(x)$	
4	$\text{rel}(\ell)$	
5		$\text{acq}(\ell)$
6		$r(y)$
7		$\text{rel}(\ell)$
8		$r(x)$
9		$w(x)$

Some Notation

- Events$_\sigma$ - set of events of σ.
- Thread order (a.k.a program order) $\leq^\sigma_{\mathsf{TO}}$



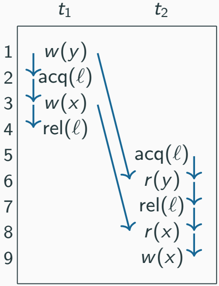
Some Notation

- Events_σ - set of events of σ.
- Thread order (a.k.a program order) $\leq^{\sigma}_{\mathsf{TO}}$
- Reads-from mapping $\mathcal{RF}_\sigma : \text{Reads}_\sigma \to \text{Writes}_\sigma$
  - $\mathcal{RF}_\sigma(e)$ is the last write event before e in σ that writes to the same memory location as x.
  - e reads the value written by $\mathcal{RF}_\sigma(e)$

	t_1	t_2
1	$w(y)$	
2	$\text{acq}(\ell)$	
3	$w(x)$	
4	$\text{rel}(\ell)$	
5		$\text{acq}(\ell)$
6		$r(y)$
7		$\text{rel}(\ell)$
8		$r(x)$
9		$w(x)$

Some Notation

- Events_σ - set of events of σ.
- Thread order (a.k.a program order) $\leq^{\sigma}_{\text{TO}}$
- Reads-from mapping $\mathcal{RF}_\sigma : \text{Reads}_\sigma \to \text{Writes}_\sigma$
  - $\mathcal{RF}_\sigma(e)$ is the last write event before e in σ that writes to the same memory location as x.
  - e reads the value written by $\mathcal{RF}_\sigma(e)$
- $\leq^{\sigma}_{\text{TRF}} = (\leq^{\sigma}_{\text{TO}} \cup \mathcal{RF}^{-1}_\sigma)^+$



Correct Reordering

Correct Reordering [Sma+12]

A sequence σ^* is a correct reordering of trace σ if

1. σ^* is a well-formed trace
2. $\mathsf{Events}_{\sigma^*} \subseteq \mathsf{Events}_\sigma$
3. $\mathsf{Events}_{\sigma^*}$ is downward closed with respect $\leq^{\sigma}_{\mathsf{TRF}}$
4. $\leq^{\sigma^*}_{\mathsf{TO}} = \leq^{\sigma}_{\mathsf{TO}}|_{\mathsf{Events}_{\sigma^*}}$
5. $\mathcal{RF}_{\sigma^*} = \mathcal{RF}_\sigma|_{\mathsf{Events}_{\sigma^*}}$

Any program that can generate σ can generate all correct reorderings of σ

Data Race Prediction

A pair (e_1, e_2) in σ is a predictable data race if

- e_1 and e_2 are conflicting
- there is a correct reordering σ^* of σ, such that e_1 and e_2 are both σ-enabled in σ^*

	t_1	t_2
1	$r(x)$	
2	$\mathsf{acq}(\ell)$	
3	$w(x)$	
4	$\mathsf{rel}(\ell)$	
5		$\mathsf{acq}(\ell)$
6		$w(x)$
7		$\mathsf{rel}(\ell)$

Data Race Prediction

A pair (e_1, e_2) in σ is a predictable data race if

- e_1 and e_2 are conflicting
- there is a correct reordering σ^* of σ, such that e_1 and e_2 are both σ-enabled in σ^*

	t_1	t_2
1	$r(x)$	
2	acq(ℓ)	
3	$w(x)$	
4	rel(ℓ)	
5		acq(ℓ)
6		$w(x)$
7		rel(ℓ)

Data Race Prediction

A pair (e_1, e_2) in σ is a predictable data race if

- e_1 and e_2 are conflicting
- there is a correct reordering σ^* of σ, such that e_1 and e_2 are both σ-enabled in σ^*

	t_1	t_2
1	$r(x)$	
2	acq(ℓ)	
3	$w(x)$	
4	rel(ℓ)	
5		acq(ℓ)
6		$w(x)$
7		rel(ℓ)

Correct Reordering

	t_1	t_2
1		acq(ℓ)
2	$r(x)$	
3		$w(x)$

Data Race Prediction

A pair (e_1, e_2) in σ is a predictable data race if

- e_1 and e_2 are conflicting
- there is a correct reordering σ^* of σ, such that e_1 and e_2 are both σ-enabled in σ^*

If a program P has an execution with a predictable data race, then P has an execution with a data race

	t_1	t_2
1	$r(x)$	
2	acq(ℓ)	
3	$w(x)$	
4	rel(ℓ)	
5		acq(ℓ)
6		$w(x)$
7		rel(ℓ)

Correct Reordering

	t_1	t_2
1		acq(ℓ)
2	$r(x)$	
3		$w(x)$

Soundness and Completeness

Soundness

Completeness

Decision problem

$\mathcal{A}(\sigma) \in \{\text{YES}, \text{NO}\}$

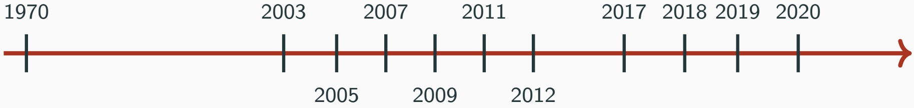
$\mathcal{A}$ is sound if

- whenever $\mathcal{A}(\sigma) = \text{YES}$, then σ has a predictable data race.

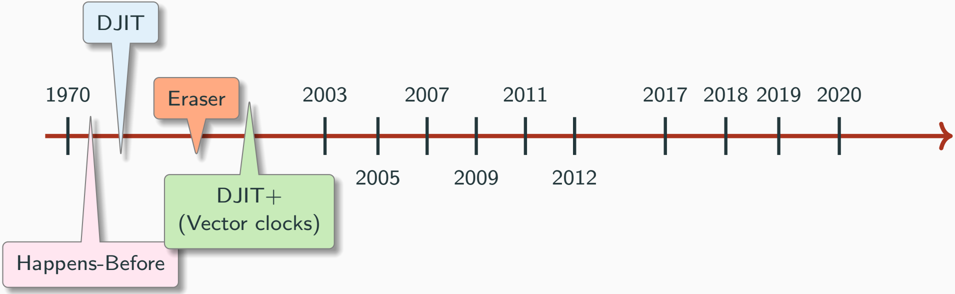
$\mathcal{A}$ is complete if

- $\mathcal{A}(\sigma) = \text{YES}$ whenever σ has a predictable data race.

History of Data Race Prediction

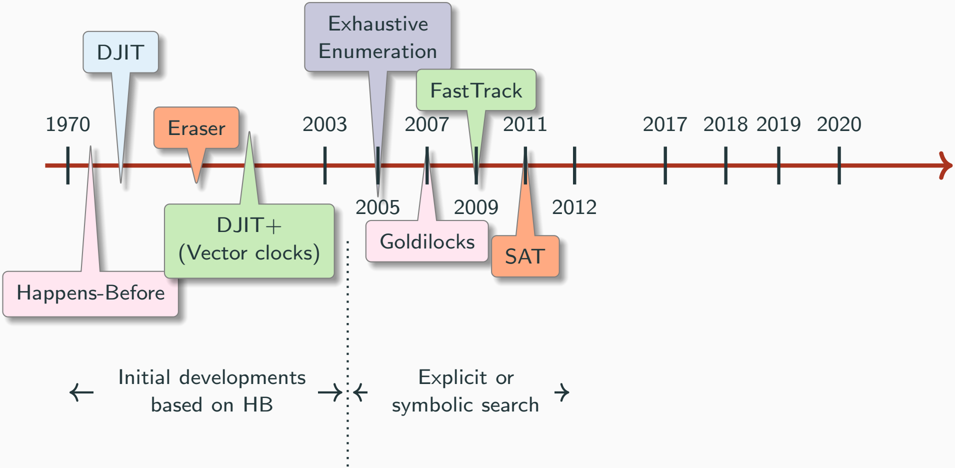


History of Data Race Prediction

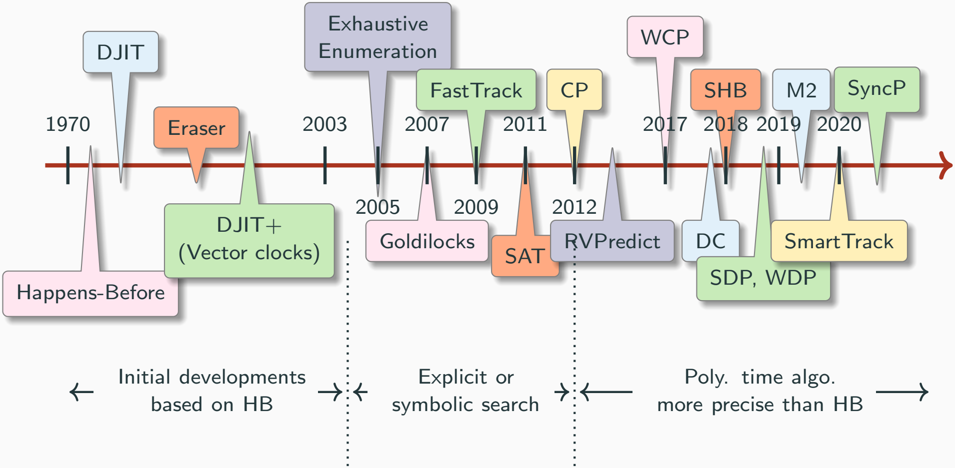


← Initial developments based on HB →

History of Data Race Prediction



History of Data Race Prediction



Eraser's lockset algorithm

Lockset principle (Eraser [Sav+97] style race detection):

Two events cannot be simultaneously enabled in any correct reordering if they are protected by a common lock.

	t_1	t_2
1	acq(ℓ)	
2	$w(x)$	
3	rel(ℓ)	
4		acq(ℓ)
5		$w(x)$
6		rel(ℓ)

Eraser's lockset algorithm

Lockset principle (Eraser [Sav+97] style race detection):

> *Two events cannot be simultaneously enabled in any correct reordering if they are protected by a common lock.*

	t_1	t_2
1	acq(ℓ)	
2	$w(x)$	
3	rel(ℓ)	
4		acq(ℓ)
5		$w(x)$
6		rel(ℓ)

	t_1	t_2
1	$w(y)$	
2	acq(ℓ)	
3	$w(x)$	
4	rel(ℓ)	
5		acq(ℓ)
6		$r(x)$
7		rel(ℓ)
8		$w(y)$

Unsound

Enumeration based techniques

- Data race prediction is in NP
    - Guess a sequence of events and check if it is a correct reordering having a race

Enumeration based techniques

- Data race prediction is in NP
  - Guess a sequence of events and check if it is a correct reordering having a race
- Explicit enumerative techniques [SRA05; CR07]
  - Generate all correct reorderings and check
  - $O(\text{poly}(2^n))$

Enumeration based techniques

- Data race prediction is in NP
  - Guess a sequence of events and check if it is a correct reordering having a race
- Explicit enumerative techniques [SRA05; CR07]
  - Generate all correct reorderings and check
  - $O(\text{poly}(2^n))$
- Symbolic techniques [Wan+09; Sai+11; HMR14]
  - Encode the existence of a correct reordering with race as a SAT/SMT constraint
  - $O(\text{SAT}(\text{poly}(n)))$

Enumeration based techniques

- Data race prediction is in NP
  - Guess a sequence of events and check if it is a correct reordering having a race
- Explicit enumerative techniques [SRA05; CR07]
  - Generate all correct reorderings and check
  - $O(\text{poly}(2^n))$
- Symbolic techniques [Wan+09; Sai+11; HMR14]
  - Encode the existence of a correct reordering with race as a SAT/SMT constraint
  - $O(\text{SAT}(\text{poly}(n)))$
- Sound and complete, but don't scale

Happens-Before

For a trace σ, the happens-before relation on σ is the smallest partial order $\leq^{\sigma}_{\mathsf{HB}}$ such that

1. $\leq^{\sigma}_{\mathsf{TO}} \subseteq \leq^{\sigma}_{\mathsf{HB}}$,

	t_1	t_2	t_3
1	$\mathsf{acq}(\ell_1)$		
2	$w(x)$		
3	$\mathsf{rel}(\ell_1)$		
4		$\mathsf{acq}(\ell_1)$	
5		$\mathsf{acq}(\ell_2)$	
6		$\mathsf{rel}(\ell_2)$	
7			$\mathsf{acq}(\ell_2)$
8			$w(x)$
9		$\mathsf{rel}(\ell_1)$	
10			$\mathsf{rel}(\ell_2)$

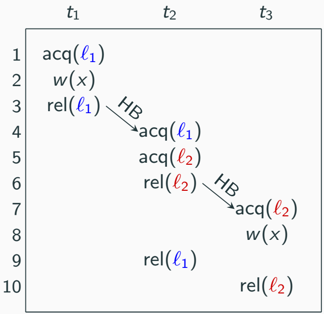
For a trace σ, the happens-before relation on σ is the smallest partial order $\leq^{\sigma}_{\mathrm{HB}}$ such that

1. $\leq^{\sigma}_{\mathrm{TO}} \subseteq \leq^{\sigma}_{\mathrm{HB}}$,
2. for any lock ℓ and events $e_1 = \langle \cdot, \mathrm{rel}(\ell) \rangle$ and $e_2 = \langle \cdot, \mathrm{acq}(\ell) \rangle$ with $e_1 \leq^{\sigma}_{\mathrm{tr}} e_2$, $e_1 \leq^{\sigma}_{\mathrm{HB}} e_2$

	t_1	t_2	t_3
1	$\mathrm{acq}(\ell_1)$		
2	$w(x)$		
3	$\mathrm{rel}(\ell_1)$	HB	
4		$\mathrm{acq}(\ell_1)$	
5		$\mathrm{acq}(\ell_2)$	
6		$\mathrm{rel}(\ell_2)$	
7			$\mathrm{acq}(\ell_2)$
8			$w(x)$
9		$\mathrm{rel}(\ell_1)$	
10			$\mathrm{rel}(\ell_2)$

For a trace σ, the happens-before relation on σ is the smallest partial order $\leq_{\text{HB}}^{\sigma}$ such that

1. $\leq_{\text{TO}}^{\sigma} \subseteq \leq_{\text{HB}}^{\sigma}$,
2. for any lock ℓ and events $e_1 = \langle \cdot, \text{rel}(\ell) \rangle$ and $e_2 = \langle \cdot, \text{acq}(\ell) \rangle$ with $e_1 \leq_{\text{tr}}^{\sigma} e_2$, $e_1 \leq_{\text{HB}}^{\sigma} e_2$



Happens-Before - Race Detection

HB-race : Pair of conflicting events (e_1, e_2) such that $e_1 \not\leq_{\mathsf{HB}} e_2$ and $e_2 \not\leq_{\mathsf{HB}} e_1$.

Soundness of HB

If a trace σ has an HB-race, then σ has a predictable data race.

Happens-Before - Race Detection

HB-race : Pair of conflicting events (e_1, e_2) such that $e_1 \not\leq_{\mathrm{HB}} e_2$ and $e_2 \not\leq_{\mathrm{HB}} e_1$.

Soundness of HB

If a trace σ has an HB-race, then σ has a predictable data race.

Algorithm:

- Vector clock algorithm, single pass, streaming
- Linear time and constant space
- Implemented by most commercial race detectors - TSan [SI09], FastTrack [FF09], Helgrind, Intel Inspector, etc.,

Happens-Before - Predictive Power

	t_1	t_2
1	$w(x)$	
2	acq(ℓ)	
3	rel(ℓ)	
4		acq(ℓ)
5		rel(ℓ)
6		$w(x)$

HB

Happens-Before - Predictive Power

	t_1	t_2
1	$w(x)$	
2	acq(ℓ)	
3	rel(ℓ)	
4		acq(ℓ)
5		rel(ℓ)
6		$w(x)$

	t_1	t_2
1		acq(ℓ)
2		rel(ℓ)
3		$w(x)$
4	$w(x)$	
5	acq(ℓ)	
6	rel(ℓ)	

Witness correct reordering

Happens-Before - Predictive Power

	t_1	t_2
1	$w(x)$	
2	acq(ℓ)	
3	rel(ℓ)	
4		acq(ℓ)
5		rel(ℓ)
6		$w(x)$

Predictable race missed by HB

	t_1	t_2
1		acq(ℓ)
2		rel(ℓ)
3		$w(x)$
4	$w(x)$	
5	acq(ℓ)	
6	rel(ℓ)	

Witness correct reordering

Happens-Before - Predictive Power

	t_1	t_2
1	$w(x)$	
2	acq(ℓ)	
3	rel(ℓ)	
4		acq(ℓ)
5		rel(ℓ)
6		$w(x)$

Predictable race missed by HB

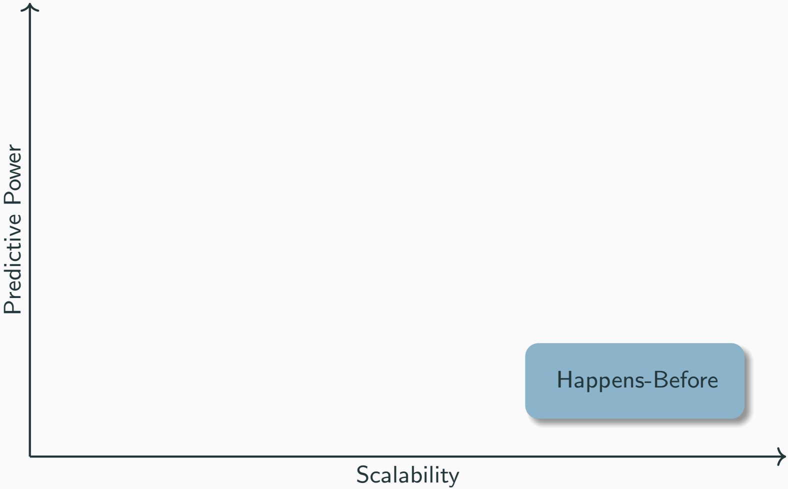
HB is conservative!

HB misses simple races
(Poor predictive power)

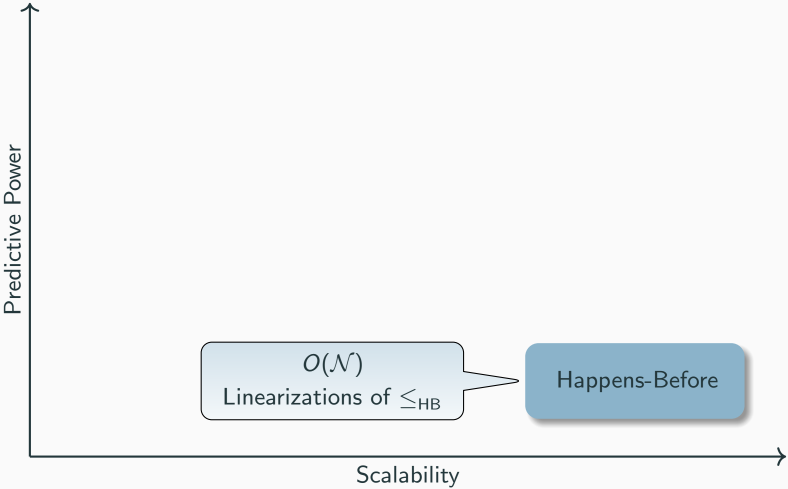
	t_1	t_2
1		acq(ℓ)
2		rel(ℓ)
3		$w(x)$
4	$w(x)$	
5	acq(ℓ)	
6	rel(ℓ)	

Witness correct reordering

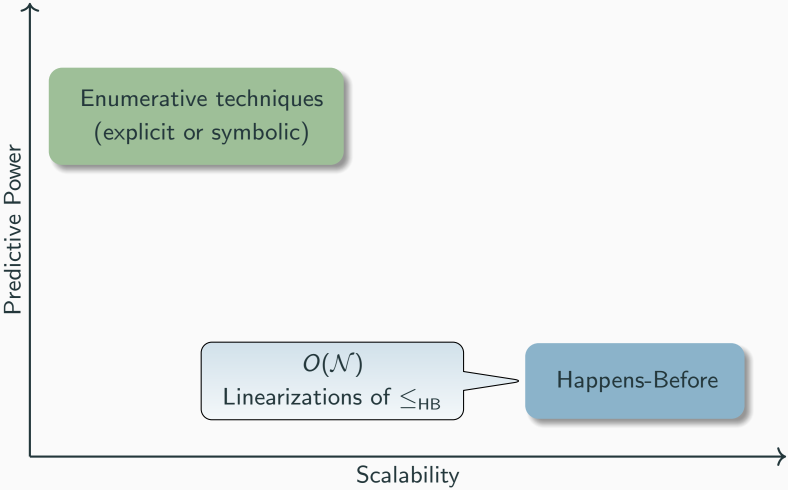
Historical (Sound) Predictive Analysis



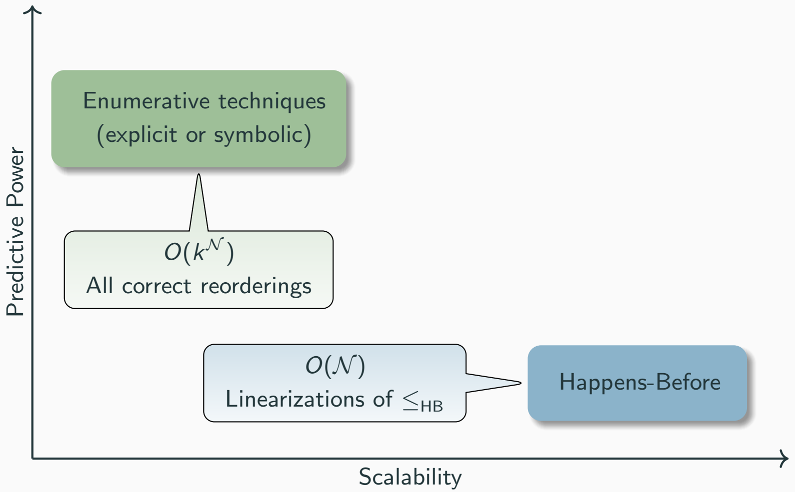
Historical (Sound) Predictive Analysis



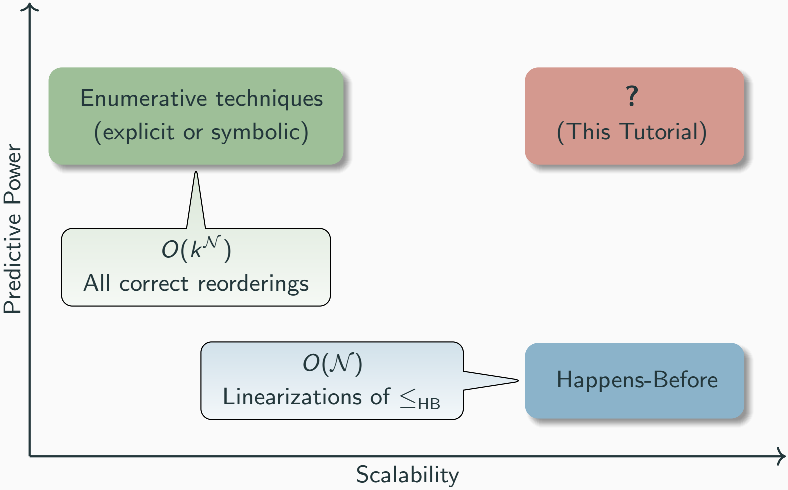
Historical (Sound) Predictive Analysis



Historical (Sound) Predictive Analysis



Historical (Sound) Predictive Analysis



Schedulable Happens-Before

What happens after the first race?

Soundness Guarantee of HB

Soundness of HB

If a trace σ has an HB-race, then σ has a predictable data race.

Soundness Guarantee of HB

Soundness of HB

If a trace σ has an HB-race, then σ has a predictable data race.

	t_1	t_2
1	$r(x)$	
2	$w(y)$	
3		$r(y)$
4		$w(x)$

Execution

Soundness Guarantee of HB

Soundness of HB

If a trace σ has an HB-race, then σ has a predictable data race.

	t_1	t_2
1	$r(x)$	
2	$w(y)$	
3		$r(y)$
4		$w(x)$

Execution

Soundness Guarantee of HB

Soundness of HB

If a trace σ has an HB-race, then σ has a predictable data race.

	t_1	t_2
1	$r(x)$	
2	$w(y)$	
3		$r(y)$
4		$w(x)$

Execution

Soundness of HB

If a trace σ has an HB-race, then σ has a predictable data race.

	t_1	t_2
1	$r(x)$	
2	$w(y)$	
3		$r(y)$
4		$w(x)$

Execution

t_1

```
y = x + 5;
```

t_2

```
if(y == 5){
  x = 10;
} else {
  while (true);
}
```

Possible Program
(Initially, `x = 0`, `y = 0`)

Soundness Guarantee of HB

Soundness of HB

If a trace σ has an HB-race, then σ has a predictable data race.

What it is

If σ has
an HB-race,
then σ has some
predictable race

Soundness Guarantee of HB

Soundness of HB

If a trace σ has an HB-race, then σ has a predictable data race.

What it is

If σ has an HB-race, then σ has some predictable race

What it is NOT

If (e_1, e_2) is an HB-race in σ, then (e_1, e_2) is a predictable race

(More precise) Soundness Guarantee of HB

First HB-race : HB-race (e_1, e_2) such that there is no earlier HB-race in σ:

- for every HB-race (e_1', e_2'), either $e_2 \leq_{\text{tr}}^{\sigma} e_2'$, or $(e_2 = e_2'$ and $e_1' \leq_{\text{tr}}^{\sigma} e_1)$.

	t_1	t_2
1	$r(x)$	
2	$w(y)$	
3		$r(y)$
4		$w(x)$

More Precise Soundness of HB

If a trace σ has an HB-race, then the first HB-race in σ is a predictable data race.

What about other HB races?

- Not every HB-race is predictable.
- Deciding whether an arbitrary HB-race (e_1, e_2) is predictable is intractable
- What if we restrict to HB-respecting orderings?

Schedulable Happens-Before

For a trace σ, the **schedulable-happens-before** relation on σ is the smallest partial order $\leq^{\sigma}_{\mathsf{SHB}}$ such that

1. $\leq^{\sigma}_{\mathsf{HB}} \subseteq \leq^{\sigma}_{\mathsf{SHB}}$,
2. for events e_1, e_2 such that $\mathcal{RF}_{\sigma}(e_2) = e_1$, then $e_1 \leq^{\sigma}_{\mathsf{SHB}} e_2$

	t_1	t_2	t_3
1	acq(ℓ)		
2	rel(ℓ)		
3	$w(z)$		
4		acq(ℓ)	
5		rel(ℓ)	
6		$w(x)$	
7			$r(x)$
8			$w(z)$

Schedulable Happens-Before

For a trace σ, the **schedulable-happens-before** relation on σ is the smallest partial order $\leq_{\mathsf{SHB}}^{\sigma}$ such that

1. $\leq_{\mathsf{HB}}^{\sigma} \subseteq \leq_{\mathsf{SHB}}^{\sigma}$,
2. for events e_1, e_2 such that $\mathcal{RF}_{\sigma}(e_2) = e_1$, then $e_1 \leq_{\mathsf{SHB}}^{\sigma} e_2$

	t_1	t_2	t_3
1	acq(ℓ)		
2	rel(ℓ)		
3	$w(z)$		
4		acq(ℓ)	
5		rel(ℓ)	
6		$w(x)$	
7			$r(x)$
8			$w(z)$

Schedulable Happens-Before

For a trace σ, the **schedulable-happens-before** relation on σ is the smallest partial order $\leq^{\sigma}_{\mathsf{SHB}}$ such that

1. $\leq^{\sigma}_{\mathsf{HB}} \subseteq \leq^{\sigma}_{\mathsf{SHB}}$,
2. for events e_1, e_2 such that $\mathcal{RF}_{\sigma}(e_2) = e_1$, then $e_1 \leq^{\sigma}_{\mathsf{SHB}} e_2$

	t_1	t_2	t_3
1	acq(ℓ)		
2	rel(ℓ)		
3	$w(z)$		
4		acq(ℓ)	
5		rel(ℓ)	
6		$w(x)$	
7			$r(x)$
8			$w(z)$

Beyond the First Race - Soundness of SHB

Soundness of SHB

Let (e_1, e_2) be an HB-race of σ. Then,

(e_1, e_2) is a predictable race witnessed by a HB-respecting correct reordering of σ

iff

there is no e_3 for which $e_1 \leq^{\sigma}_{\mathsf{SHB}} e_3 \leq^{\sigma}_{\mathsf{SHB}} e_2$ or $e_2 \leq^{\sigma}_{\mathsf{SHB}} e_3 \leq^{\sigma}_{\mathsf{SHB}} e_1$.

SHB Algorithm and Evaluation

Algorithm

- One pass, streaming algorithm
- Vector clock algorithm, similar to HB
- Can be easily implemented in existing HB-based race detectors.
- Linear time and constant space (like HB)

Evaluation

SHB Algorithm and Evaluation

Algorithm

- One pass, streaming algorithm
- Vector clock algorithm, similar to HB
- Can be easily implemented in existing HB-based race detectors.
- Linear time and constant space (like HB)

Evaluation

- Implemented in RAPID (`github.com/umangm/rapid`)
- Evaluated on standard benchmarks
- 50% fewer race reports than HB
- Most others were confirmed to be false positives
- Scales linearly (like HB)

Weak Causal Precedence

Predicting races beyond HB in linear time

Conservativeness of HB

	t_1	t_2
1	$w(x)$	
2	acq(ℓ)	
3	rel(ℓ) $\xrightarrow{\text{HB}}$	
4		acq(ℓ)
5		rel(ℓ)
6		$w(x)$

HB is conservative!

- HB orders all critical sections on the same lock

Conservativeness of HB

	t_1	t_2
1	$w(x)$	
2	$\mathrm{acq}(\ell)$	
3	$\mathrm{rel}(\ell) \xrightarrow{\text{HB}}$	
4		$\mathrm{acq}(\ell)$
5		$\mathrm{rel}(\ell)$
6		$w(x)$

HB is conservative!

- HB orders all critical sections on the same lock
- Space of correct reorderings = linearizations of $\leq_{\mathrm{HB}}$
  - Linear time (scalability)
  - No false positives (soundness)

Conservativeness of HB

	t_1	t_2
1	$w(x)$	
2	acq(ℓ)	
3	rel(ℓ)	
4		acq(ℓ)
5		rel(ℓ)
6		$w(x)$

HB

HB is conservative!

- HB orders all critical sections on the same lock
- Space of correct reorderings = linearizations of $\leq_{\text{HB}}$
  - Linear time (scalability)
  - No false positives (soundness)
- Does not reflect any 'hard' orderings

Conservativeness of HB

	t_1	t_2
1	$w(x)$	
2	acq(ℓ)	
3	rel(ℓ)	
4		acq(ℓ)
5		rel(ℓ)
6		$w(x)$

HB is conservative!

- HB orders all critical sections on the same lock
- Space of correct reorderings = linearizations of $\leq_{\text{HB}}$
  - Linear time (scalability)
  - No false positives (soundness)
- Does not reflect any 'hard' orderings
- Misses simple races

Tackling HB's Conservativeness

Can we relax some HB-orderings?

Tackling HB's Conservativeness

Can we relax some HB-orderings?

- Naively $\implies$ infeasible reorderings

	t_1	t_2
1	$w(x)$	
2	acq(ℓ)	
3	rel(ℓ)	
4		acq(ℓ)
5		rel(ℓ)
6		$w(x)$

Can reorder critical sections

Tackling HB's Conservativeness

Can we relax some HB-orderings?

- Naively $\implies$ infeasible reorderings

	t_1	t_2
1	$w(x)$	
2	acq(ℓ)	
3	rel(ℓ)	
4		acq(ℓ)
5		rel(ℓ)
6		$w(x)$

Can reorder critical sections

	t_1	t_2
1	acq(ℓ)	
2	$w(x)$	
3	rel(ℓ)	
4		acq(ℓ)
5		$r(x)$
6		rel(ℓ)

Cannot reorder critical sections

Tackling HB's Conservativeness

Can we relax some HB-orderings?

- Naively $\implies$ infeasible reorderings
- Carefully $\implies$ expensive (?)

	t_1	t_2
1	$w(x)$	
2	acq(ℓ)	
3	rel(ℓ)	
4		acq(ℓ)
5		rel(ℓ)
6		$w(x)$

Can reorder critical sections

	t_1	t_2
1	acq(ℓ)	
2	$w(x)$	
3	rel(ℓ)	
4		acq(ℓ)
5		$r(x)$
6		rel(ℓ)

Cannot reorder critical sections

Tackling HB's Conservativeness

Can we relax some HB-orderings?

- Naively $\implies$ infeasible reorderings
- Carefully $\implies$ expensive (?)

	t_1	t_2
1	$w(x)$	
2	$\mathsf{acq}(\ell)$	
3	$\mathsf{rel}(\ell)$	
4		$\mathsf{acq}(\ell)$
5		$\mathsf{rel}(\ell)$
6		$w(x)$

Can reorder critical sections

	t_1	t_2
1	$\mathsf{acq}(\ell)$	
2	$w(x)$	
3	$\mathsf{rel}(\ell)$	
4		$\mathsf{acq}(\ell)$
5		$r(x)$
6		$\mathsf{rel}(\ell)$

Cannot reorder critical sections

WCP effectively identifies when to (not) order critical sections

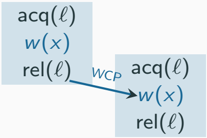
Weak Causal Precedence (WCP)

For a trace σ, $\prec^{\sigma}_{\mathsf{WCP}}$ is the smallest transitive order with the following properties:

Weak Causal Precedence (WCP)

For a trace σ, $\prec^{\sigma}_{\text{WCP}}$ is the smallest transitive order with the following properties:

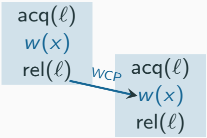
1. Let C_1 and C_2 be critical sections on the same lock ℓ. If there are **conflicting** events $e_1 \in C_1$ and $e_2 \in C_2$, then $\mathrm{rel}(C_1) \prec^{\sigma}_{\text{WCP}} e_2$



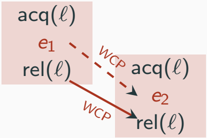
Weak Causal Precedence (WCP)

For a trace σ, $\prec^{\sigma}_{\mathrm{WCP}}$ is the smallest transitive order with the following properties:

1. Let C_1 and C_2 be critical sections on the same lock ℓ. If there are **conflicting** events $e_1 \in C_1$ and $e_2 \in C_2$, then $\mathrm{rel}(C_1) \prec^{\sigma}_{\mathrm{WCP}} e_2$



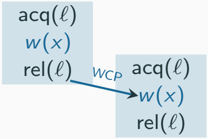
2. Let C_1 and C_2 be critical sections on the same lock ℓ. If there are events $e_1 \in C_1$ and $e_2 \in C_2$ such that $e_1 \prec^{\sigma}_{\mathrm{WCP}} e_2$, then $\mathrm{rel}(C_1) \prec^{\sigma}_{\mathrm{WCP}} \mathrm{rel}(C_2)$



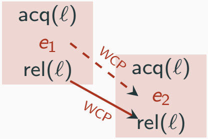
Weak Causal Precedence (WCP)

For a trace σ, $\prec^{\sigma}_{\mathrm{WCP}}$ is the smallest transitive order with the following properties:

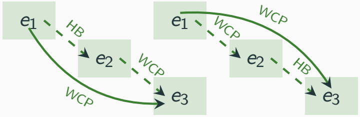
1. Let C_1 and C_2 be critical sections on the same lock ℓ. If there are **conflicting** events $e_1 \in C_1$ and $e_2 \in C_2$, then $\mathrm{rel}(C_1) \prec^{\sigma}_{\mathrm{WCP}} e_2$



2. Let C_1 and C_2 be critical sections on the same lock ℓ. If there are events $e_1 \in C_1$ and $e_2 \in C_2$ such that $e_1 \prec^{\sigma}_{\mathrm{WCP}} e_2$, then $\mathrm{rel}(C_1) \prec^{\sigma}_{\mathrm{WCP}} \mathrm{rel}(C_2)$



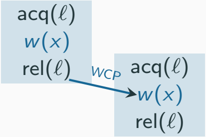
3. $\prec^{\sigma}_{\mathrm{WCP}}$ is closed under composition with $\leq^{\sigma}_{\mathrm{HB}}$:
   - $\leq^{\sigma}_{\mathrm{HB}} \circ \prec^{\sigma}_{\mathrm{WCP}} \subseteq \prec^{\sigma}_{\mathrm{WCP}}$
   - $\prec^{\sigma}_{\mathrm{WCP}} \circ \leq^{\sigma}_{\mathrm{HB}} \subseteq \prec^{\sigma}_{\mathrm{WCP}}$



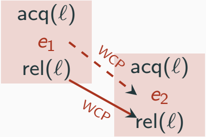
Weak Causal Precedence (WCP)

For a trace σ, $\prec^{\sigma}_{\mathrm{WCP}}$ is the smallest transitive order with the following properties:

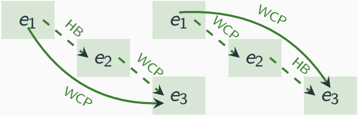
1. Let C_1 and C_2 be critical sections on the same lock ℓ. If there are **conflicting** events $e_1 \in C_1$ and $e_2 \in C_2$, then $\mathrm{rel}(C_1) \prec^{\sigma}_{\mathrm{WCP}} e_2$



2. Let C_1 and C_2 be critical sections on the same lock ℓ. If there are events $e_1 \in C_1$ and $e_2 \in C_2$ such that $e_1 \prec^{\sigma}_{\mathrm{WCP}} e_2$, then $\mathrm{rel}(C_1) \prec^{\sigma}_{\mathrm{WCP}} \mathrm{rel}(C_2)$



3. $\prec^{\sigma}_{\mathrm{WCP}}$ is closed under composition with $\leq^{\sigma}_{\mathrm{HB}}$:
   - $\leq^{\sigma}_{\mathrm{HB}} \circ \prec^{\sigma}_{\mathrm{WCP}} \subseteq \prec^{\sigma}_{\mathrm{WCP}}$
   - $\prec^{\sigma}_{\mathrm{WCP}} \circ \leq^{\sigma}_{\mathrm{HB}} \subseteq \prec^{\sigma}_{\mathrm{WCP}}$



Finally, $\leq^{\sigma}_{\mathrm{WCP}} = \prec^{\sigma}_{\mathrm{WCP}} \cup \leq^{\sigma}_{\mathrm{TO}}$

Soundness of WCP

WCP-race : Pair of conflicting events (e_1, e_2) such that $e_1 \not\leq_{\mathsf{WCP}} e_2$ and $e_2 \not\leq_{\mathsf{WCP}} e_1$.

Soundness of WCP

If a trace σ has an WCP-race, then σ either has a **predictable data race** or a **predictable deadlock**.

WCP versus HB

Comparison with HB

WCP is strictly more predictive than HB:

- For every trace σ, $\leq^{\sigma}_{\mathsf{WCP}} \subseteq \leq^{\sigma}_{\mathsf{HB}}$. Thus, every HB-race is also a WCP-race.
- There is a trace σ for which $\leq^{\sigma}_{\mathsf{WCP}} \subsetneq \leq^{\sigma}_{\mathsf{HB}}$. Thus, WCP can catch races that HB can miss.

WCP versus HB

Comparison with HB

WCP is strictly more predictive than HB:

- For every trace σ, $\leq^{\sigma}_{\mathsf{WCP}} \subseteq \leq^{\sigma}_{\mathsf{HB}}$. Thus, every HB-race is also a WCP-race.
- There is a trace σ for which $\leq^{\sigma}_{\mathsf{WCP}} \subsetneq \leq^{\sigma}_{\mathsf{HB}}$. Thus, WCP can catch races that HB can miss.

	t_1	t_2
1	$w(x)$	
2	acq(ℓ)	
3	rel(ℓ)	
4		acq(ℓ)
5		rel(ℓ)
6		$w(x)$

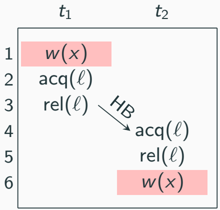
Critical sections can be swapped
Race detected by WCP, missed by HB

WCP versus HB

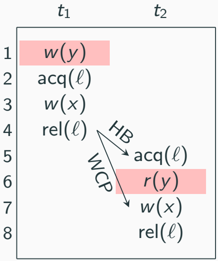
Comparison with HB

WCP is strictly more predictive than HB:

- For every trace σ, $\leq^{\sigma}_{\text{WCP}} \subseteq \leq^{\sigma}_{\text{HB}}$. Thus, every HB-race is also a WCP-race.
- There is a trace σ for which $\leq^{\sigma}_{\text{WCP}} \subsetneq \leq^{\sigma}_{\text{HB}}$. Thus, WCP can catch races that HB can miss.



Critical sections can be swapped
Race detected by WCP, missed by HB



Critical sections cannot be swapped
Race detected by WCP, missed by HB

WCP versus HB

Comparison with HB

WCP is strictly more predictive than HB:

- For every trace σ, $\leq^{\sigma}_{\mathrm{WCP}} \subseteq \leq^{\sigma}_{\mathrm{HB}}$. Thus, every HB-race is also a WCP-race.
- There is a trace σ for which $\leq^{\sigma}_{\mathrm{WCP}} \subsetneq \leq^{\sigma}_{\mathrm{HB}}$. Thus, WCP can catch races that HB can miss.

	t_1	t_2
1	$w(x)$	
2	acq(ℓ)	
3	rel(ℓ)	
4		acq(ℓ)
5		rel(ℓ)
6		$w(x)$

Critical sections can be swapped
Race detected by WCP, missed by HB

	t_1	t_2
1	$w(y)$	
2	acq(ℓ)	
3	$w(x)$	
4	rel(ℓ)	
5		acq(ℓ)
6		$r(y)$
7		$w(x)$
8		rel(ℓ)

Critical sections cannot be swapped
Race detected by WCP, missed by HB

	t_1	t_2
1	acq(ℓ)	
2	$r(x)$	
3	$w(x)$	
4	rel(ℓ)	
5		acq(ℓ)
6		$r(x)$
7		$w(x)$
8		rel(ℓ)

No predictable race

Race prediction using WCP

- Vector Clock algorithm
- Uses constant number of vector clocks + FIFO queues

Race prediction using WCP

- Vector Clock algorithm
- Uses constant number of vector clocks + FIFO queues
- One pass, streaming
- **Linear time** - $O(\mathcal{N} \cdot \mathcal{T}(L + \mathcal{T}))$
- Linear space in worst case - $O(\mathcal{N} \cdot \mathcal{T})$
    - Low space overhead in practice

Race prediction using WCP

- Vector Clock algorithm
- Uses constant number of vector clocks + FIFO queues
- One pass, streaming
- **Linear time** - $O(\mathcal{N} \cdot \mathcal{T}(L + \mathcal{T}))$
- Linear space in worst case - $O(\mathcal{N} \cdot \mathcal{T})$
    - Low space overhead in practice

> **WCP** detects more races than **HB** and runs in **linear time** (like HB)

Evaluating WCP

RAPID dynamic analysis framework

github.com/umangm/rapid

Evaluating WCP

RAPID dynamic analysis framework

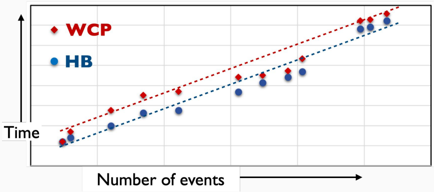
github.com/umangm/rapid

Benchmarks [KMV17]

- 18 benchmark programs (Dacapo, Apache projects, IBM Contest suite, Java Grande Forum)
- Trace sizes- 50 to 216M

RAPID dynamic analysis framework

`github.com/umangm/rapid`



WCP scales linearly, like HB

Benchmarks [KMV17]

- 18 benchmark programs (Dacapo, Apache projects, IBM Contest suite, Java Grande Forum)
- Trace sizes- 50 to 216M

Technique	Races	Avg. Time
HB	182	144 s
WCP	**190**	**198 s**
SMT*	51	2258 s

* RVPREDICT (commercial race detector)

Fundamentals

What is the cost of a sound and complete algorithm?

Approaches So Far

- Existing sound and complete techniques take exponential time
- Search over the exponential space of correct reorderings
  - Explicit (enumerate all correct reorderings)
  - Symbolic (SAT/SMT)

Approaches So Far

- Existing sound and complete techniques take exponential time
- Search over the exponential space of correct reorderings
  - Explicit (enumerate all correct reorderings)
  - Symbolic (SAT/SMT)

Is exponential time necessary?

What Is The Search Space?

Recall our definition of a correct reordering

	t_1	t_2
1	acq(ℓ)	
2	$\mathbf{e_1}$	
3	rel(ℓ)	
4		acq(ℓ)
5		$w(x)$
6		$r(x)$
7		rel(ℓ)
8		$\mathbf{e_2}$

Input trace

	t_1	t_2
1		acq(ℓ)
2		$w(x)$
3		$r(x)$
4		rel(ℓ)
5	acq(ℓ)	

Correct reordering

What Is The Search Space?

Recall our definition of a correct reordering

	t_1	t_2
1	acq(ℓ)	
2	$\mathbf{e_1}$	
3	rel(ℓ)	
4		acq(ℓ)
5		$w(x)$
6		$r(x)$
7		rel(ℓ)
8		$\mathbf{e_2}$

Input trace

$\rightarrow$

	t_1	t_2
1	acq(ℓ)	
2		acq(ℓ)
3		$w(x)$
4		$r(x)$
5		rel(ℓ)

Subset trace

$\rightarrow$

	t_1	t_2
1		acq(ℓ)
2		$w(x)$
3		$r(x)$
4		rel(ℓ)
5	acq(ℓ)	

Correct reordering

What Is The Search Space?

Recall our definition of a correct reordering

Input trace

	t_1	t_2
1	acq(ℓ)	
2	$\mathbf{e_1}$	
3	rel(ℓ)	
4		acq(ℓ)
5		$w(x)$
6		$r(x)$
7		rel(ℓ)
8		$\mathbf{e_2}$

$\rightarrow$

Subset trace

	t_1	t_2
1	acq(ℓ)	
2		acq(ℓ)
3		$w(x)$
4		$r(x)$
5		rel(ℓ)

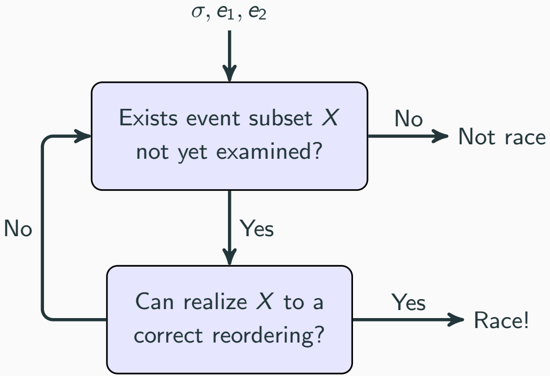
$\rightarrow$

Correct reordering

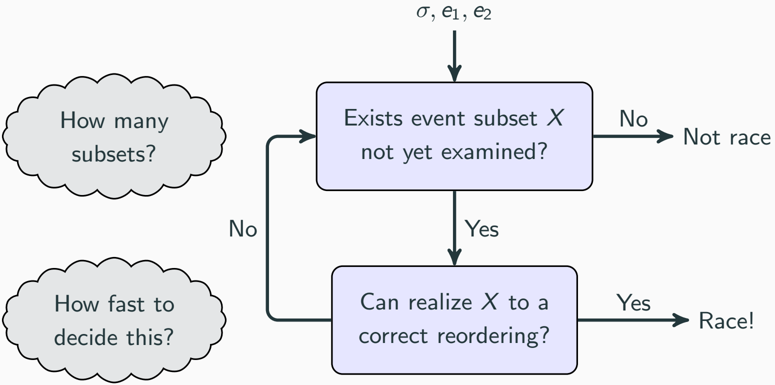
	t_1	t_2
1		acq(ℓ)
2		$w(x)$
3		$r(x)$
4		rel(ℓ)
5	acq(ℓ)	

Witness = Choose event set + choose ordering

General Approach In Data Race Prediction



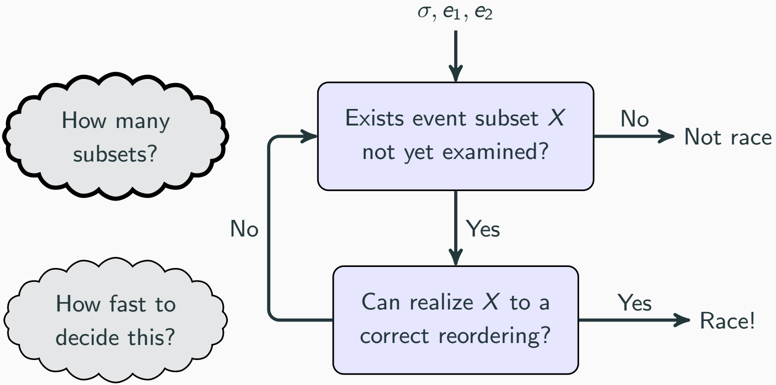
General Approach In Data Race Prediction



Complexity $O(\alpha \cdot \beta)$

- α is the number of guesses for X
- β is the complexity to decide a correct ordering of X

General Approach In Data Race Prediction



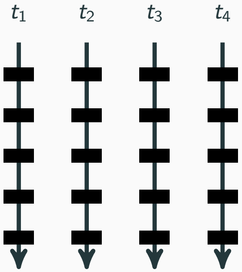
Complexity $O(\alpha \cdot \beta)$

- α is the number of guesses for X
- β is the complexity to decide a correct ordering of X

How Many Event Subsets?

$\mathcal{N}$ events, $\mathcal{T}$ threads

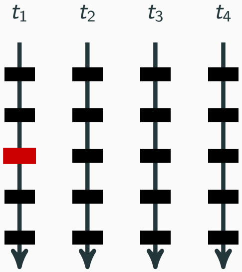
- $2^{\mathcal{N}}$ subsets in general



How Many Event Subsets?

$\mathcal{N}$ events, $\mathcal{T}$ threads

- $2^{\mathcal{N}}$ subsets in general

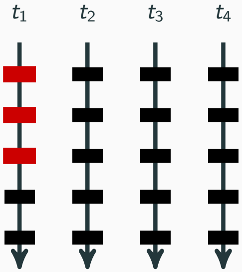


Event subsets downwards closed wrt $<_{\mathsf{TO}}$

How Many Event Subsets?

$\mathcal{N}$ events, $\mathcal{T}$ threads

- $2^{\mathcal{N}}$ subsets in general

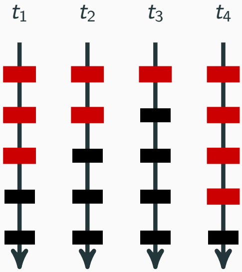


Event subsets downwards closed wrt $<_{\mathsf{TO}}$

How Many Event Subsets?

$\mathcal{N}$ events, $\mathcal{T}$ threads

- $2^{\mathcal{N}}$ subsets in general

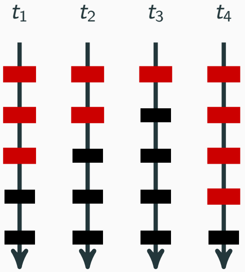


Event subsets downwards closed wrt $<_{\mathrm{TO}}$

How Many Event Subsets?

$\mathcal{N}$ events, $\mathcal{T}$ threads

- $2^{\mathcal{N}}$ subsets in general
- $\mathcal{N}^{\mathcal{T}}$ event subsets suffice
- Polynomial for small $\mathcal{T}$



Event subsets downwards closed wrt $<_{\mathsf{TO}}$

How Much Time To Decide Realizability?

Realizability of RF-Posets

Input: $(X, P, \mathcal{RF})$

- X is an event set
- $<_P$ is a partial order over X with $<_{\mathsf{TRF}} \subseteq <_P$
- $\mathcal{RF}$ is a reads-from function over X

Decide: Can we linearize $(X, <_P)$ to a trace σ^* such that $\mathcal{RF}_{\sigma^*} = \mathcal{RF}$?

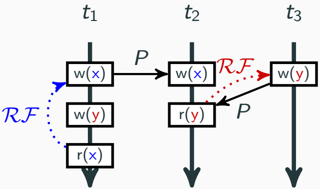
How Much Time To Decide Realizability?

Realizability of RF-Posets

Input: $(X, P, \mathcal{RF})$

- X is an event set
- $<_P$ is a partial order over X with $<_{\text{TRF}} \subseteq <_P$
- $\mathcal{RF}$ is a reads-from function over X

Decide: Can we linearize $(X, <_P)$ to a trace σ^* such that $\mathcal{RF}_{\sigma^*} = \mathcal{RF}$?



$\Longrightarrow$

	t_1	t_2	t_3
1	$w(x)$		
2	$w(y)$		
3	$r(x)$		
4		$w(x)$	
5			$w(y)$
6		$r(y)$	

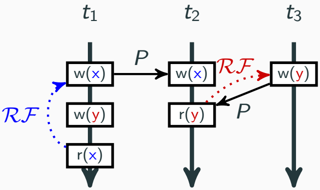
How Much Time To Decide Realizability?

Realizability of RF-Posets

Input: $(X, P, \mathcal{RF})$

- X is an event set
- $<_P$ is a partial order over X with $<_{\mathsf{TRF}} \subseteq <_P$
- $\mathcal{RF}$ is a reads-from function over X

Decide: Can we linearize $(X, <_P)$ to a trace σ^* such that $\mathcal{RF}_{\sigma^*} = \mathcal{RF}$?



$\Longrightarrow$

	t_1	t_2	t_3
1	$w(x)$		
2	$w(y)$		
3	$r(x)$		
4		$w(x)$	
5			$w(y)$
6		$r(y)$	

Lemma

RF-Poset realizability can be decided in $O(\mathcal{T} \cdot \mathcal{N}^{\mathcal{T}})$.

Upper-bound

Theorem

Dynamic data-race prediction can be solved in $O(\mathcal{T} \cdot \mathcal{N}^{2 \cdot \mathcal{T}})$ time.

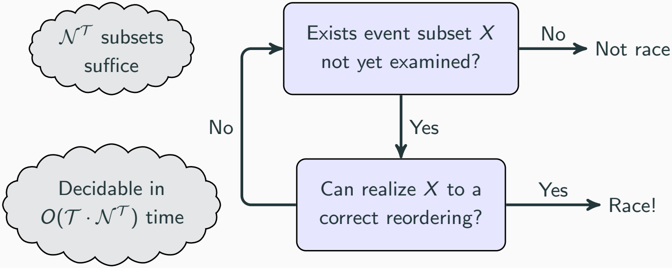
- $\mathcal{N}$ *events,* $\mathcal{T}$ *threads*

Upper-bound

Theorem

Dynamic data-race prediction can be solved in $O(\mathcal{T} \cdot \mathcal{N}^{2 \cdot \mathcal{T}})$ time.

- *$\mathcal{N}$ events, $\mathcal{T}$ threads*

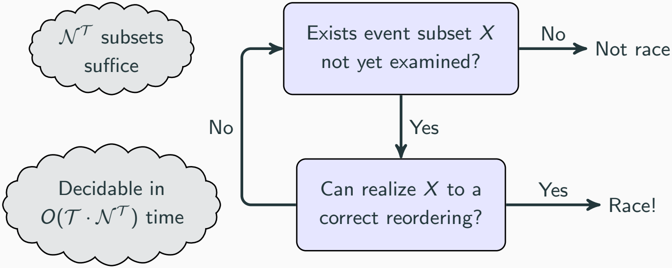


Upper-bound

Theorem

Dynamic data-race prediction can be solved in $O(\mathcal{T} \cdot \mathcal{N}^{2 \cdot \mathcal{T}})$ time.

- $\mathcal{N}$ *events,* $\mathcal{T}$ *threads*



Can we do better? $O(\mathcal{N}^{O(1)})$? $O(2^{\mathcal{T}} \cdot \mathcal{N}^{O(1)})$?

Lower-bound

Theorem

Data-race prediction is W[1]*-hard parameterized by the number of threads* $\mathcal{T}$.

Lower-bound

Theorem

Data-race prediction is W[1]*-hard parameterized by the number of threads* $\mathcal{T}$*.*

- NP-hard when $\mathcal{T} = \Omega(\mathcal{N})$
- Unlikely to solve in $f(\mathcal{T}) \cdot \mathcal{N}^{O(1)}$, for any function f
- Our upper bound $O(\mathcal{T} \cdot \mathcal{N}^{2 \cdot \mathcal{T}})$ is "tight"

Lower-bound

Theorem

Data-race prediction is W[1]*-hard parameterized by the number of threads* $\mathcal{T}$*.*

- NP-hard when $\mathcal{T} = \Omega(\mathcal{N})$
- Unlikely to solve in $f(\mathcal{T}) \cdot \mathcal{N}^{O(1)}$, for any function f
- Our upper bound $O(\mathcal{T} \cdot \mathcal{N}^{2 \cdot \mathcal{T}})$ is "tight"

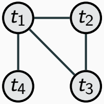
Other meaningful parameterizations that make race prediction tractable?

Acyclic Communication Topologies

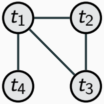
Communication Topology

An undirected graph $G = (V, E)$

- $V = \{t_1, \ldots, t_T\}$ is the set of threads
- $(t_i, t_j) \in E$ if t_i, t_j communicate over shared variables/locks



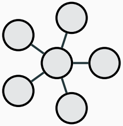
Communication Topology



An undirected graph $G = (V, E)$

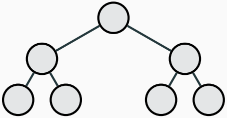
- $V = \{t_1, \ldots, t_T\}$ is the set of threads
- $(t_i, t_j) \in E$ if t_i, t_j communicate over shared variables/locks

Focus on acyclic topologies



Sever-clients

Pipeline



Divide and concquer

RF-Poset Closure

RF-Poset Closure

RF-Poset $(X, <_P, \mathcal{RF})$ is closed if

RF-Poset Closure

RF-Poset Closure

RF-Poset $(X, <_P, \mathcal{RF})$ is closed if

1. For every read r and conflicting write w
   - if $w <_P r$ then $w <_P \mathcal{RF}(r)$
   - if $\mathcal{RF}(r) <_P w'$ then $r <_P w'$

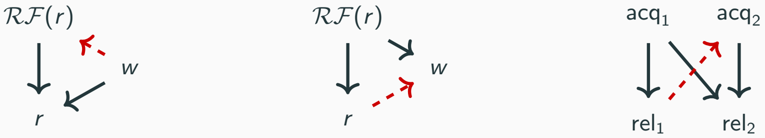
$$\begin{tikzcd}
\mathcal{RF}(r) \arrow[dd] & \\
& w \arrow[dl] \arrow[ul, dashed] \\
r &
\end{tikzcd}
\qquad\qquad
\begin{tikzcd}
\mathcal{RF}(r) \arrow[dd] \arrow[dr] & \\
& w \\
r \arrow[ur, dashed] &
\end{tikzcd}$$

RF-Poset Closure

RF-Poset Closure

RF-Poset $(X, <_P, \mathcal{RF})$ is closed if

1. For every read r and conflicting write w
   - if $w <_P r$ then $w <_P \mathcal{RF}(r)$
   - if $\mathcal{RF}(r) <_P w'$ then $r <_P w'$
2. For every two critical sections $\mathrm{acq}_1 \ldots \mathrm{rel}_1$ and $\mathrm{acq}_2 \ldots \mathrm{rel}_2$ on the same lock
   - if $\mathrm{acq}_1 <_P \mathrm{rel}_2$ then $\mathrm{rel}_1 <_P \mathrm{acq}_2$

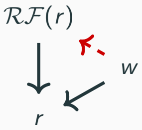


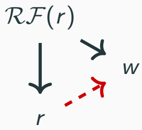
RF-Poset Closure

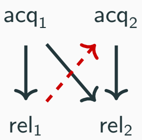
RF-Poset Closure

RF-Poset $(X, <_P, \mathcal{RF})$ is closed if

1. For every read r and conflicting write w
   - if $w <_P r$ then $w <_P \mathcal{RF}(r)$
   - if $\mathcal{RF}(r) <_P w'$ then $r <_P w'$
2. For every two critical sections $\text{acq}_1 \dots \text{rel}_1$ and $\text{acq}_2 \dots \text{rel}_2$ on the same lock
   - if $\text{acq}_1 <_P \text{rel}_2$ then $\text{rel}_1 <_P \text{acq}_2$







How fast can we compute the closure?

Complexity of Closure

Lemma

The closure of an RF-Poset can be computed in $O(\mathcal{T}^2 \cdot V \cdot \mathcal{N}^2 \cdot \log \mathcal{N})$ *time.*

Complexity of Closure

Lemma

The closure of an RF-Poset can be computed in $O(\mathcal{T}^2 \cdot V \cdot \mathcal{N}^2 \cdot \log \mathcal{N})$ time.

Cycle

$\downarrow$

Not realizable

Complexity of Closure

Lemma

The closure of an RF-Poset can be computed in $O(\mathcal{T}^2 \cdot V \cdot \mathcal{N}^2 \cdot \log \mathcal{N})$ time.

Cycle

Not realizable

No Cycle

May be realizable

Realizable under extra conditions

Complexity of Closure

Lemma

The closure of an RF-Poset can be computed in $O(\mathcal{T}^2 \cdot V \cdot \mathcal{N}^2 \cdot \log \mathcal{N})$ time.

Cycle

Not realizable

No Cycle

May be realizable

Realizable under extra conditions

Acyclic Topology

Upper-Bound On Acyclic Topologies

Theorem

Dynamic data-race prediction over acyclic communication topologies can be solved in $O(\mathcal{T}^2 \cdot V \cdot \mathcal{N}^2 \cdot \log \mathcal{N})$ time.

- $\mathcal{N}$ *events,* $\mathcal{T}$ *threads,* V *variables/locks*

No exponential dependence on any parameter!

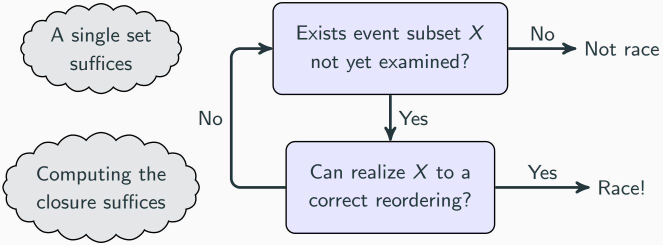
Upper-Bound On Acyclic Topologies

Theorem

Dynamic data-race prediction over acyclic communication topologies can be solved in $O(\mathcal{T}^2 \cdot V \cdot \mathcal{N}^2 \cdot \log \mathcal{N})$ time.

- *$\mathcal{N}$ events, $\mathcal{T}$ threads, V variables/locks*

No exponential dependence on any parameter!



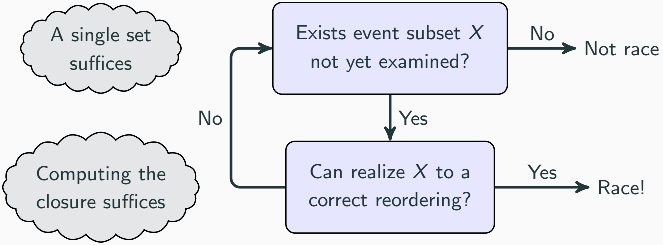
Upper-Bound On Acyclic Topologies

Theorem

Dynamic data-race prediction over acyclic communication topologies can be solved in $O(\mathcal{T}^2 \cdot V \cdot \mathcal{N}^2 \cdot \log \mathcal{N})$ *time.*

- $\mathcal{N}$ *events,* $\mathcal{T}$ *threads,* V *variables/locks*

No exponential dependence on any parameter!



Can we do better? $O(\mathcal{N}^{2-\epsilon})$?

Lower-Bound On Acyclic Topologies

Theorem (Quadratic hardness of race prediction)

Consider traces σ over

- $\mathcal{T} = 2$ *threads*
- $V = 8$ *variables*
- *a single lock*

For any fixed $\epsilon > 0$, data race prediction over σ has no $O(\mathcal{N}^{2-\epsilon})$ algorithm.

Lower-Bound On Acyclic Topologies

Theorem (Quadratic hardness of race prediction)

Consider traces σ over

- $\mathcal{T} = 2$ *threads*
- $V = 8$ *variables*
- *a single lock*

For any fixed $\epsilon > 0$, data race prediction over σ has no $O(\mathcal{N}^{2-\epsilon})$ algorithm.

Optimality for acyclic topologies

- Recall our bound $O(\mathcal{T}^2 \cdot V \cdot \mathcal{N}^2 \cdot \log \mathcal{N})$
- Nearly optimal for moderate number of threads and variables ($\mathcal{T}, V = \text{polylog}(\mathcal{N})$)

In Summary

	General	Acyclic Topologies
Upper-bound	$O(\mathcal{T} \cdot \mathcal{N}^{2 \cdot \mathcal{T}})$	$O(\mathcal{T}^2 \cdot V \cdot \mathcal{N}^2 \cdot \log \mathcal{N})$
Lower-bound	W[1]-hard	No $O(\mathcal{N}^{2-\epsilon})$

- $\mathcal{N}$ events
- $\mathcal{T}$ threads
- V variables/locks

M2

Turning complete theory into (almost) complete practice

Choosing The Event Set

Witness = Choose event set + choose ordering

Choosing The Event Set

Witness = **Choose event set** + choose ordering

Heuristic: close all critical sections not containing e_1 and e_2

Choosing The Event Set

Witness = **Choose event set** + choose ordering

Heuristic: close all critical sections not containing e_1 and e_2

Choose X as the smallest set so that

1. X contains all $<_{\mathsf{TO}}$ predecessors of e_1, e_2
2. X is downwards closed wrt $<_{\mathsf{TRF}}$
3. No open critical sections except those containing e_1, e_2

Choosing The Event Set

Witness = **Choose event set** + choose ordering

Heuristic: close all critical sections not containing e_1 and e_2

Choose X as the smallest set so that

1. X contains all $<_{\mathrm{TO}}$ predecessors of e_1, e_2
2. X is downwards closed wrt $<_{\mathrm{TRF}}$
3. No open critical sections except those containing e_1, e_2

	t_1	t_2	t_3
1	$w(y)$		
2	$\mathrm{acq}(\ell)$		
3	$w(x)$		
4	$\mathbf{e_1}$		
5	$\mathrm{rel}(\ell)$		
6		$w(y)$	
7		$r(y)$	
8			$\mathrm{acq}(\ell)$
9			$w(x)$
10			$w(y)$
11			$\mathrm{rel}(\ell)$
12		$r(x)$	
13		$\mathbf{e_2}$	

Choosing The Event Set

Witness = **Choose event set** + choose ordering

Heuristic: close all critical sections not containing e_1 and e_2

Choose X as the smallest set so that

1. X contains all $<_{\text{TO}}$ predecessors of e_1, e_2
2. X is downwards closed wrt $<_{\text{TRF}}$
3. No open critical sections except those containing e_1, e_2

	t_1	t_2	t_3
1	$w(y)$		
2	$\text{acq}(\ell)$		
3	$w(x)$		
4	$\mathbf{e_1}$		
5	$\text{rel}(\ell)$		
6		$w(y)$	
7		$r(y)$	
8			$\text{acq}(\ell)$
9			$w(x)$
10			$w(y)$
11			$\text{rel}(\ell)$
12		$r(x)$	
13		$\mathbf{e_2}$	

Choosing The Event Set

Witness = **Choose event set** + choose ordering

Heuristic: close all critical sections not containing e_1 and e_2

Choose X as the smallest set so that

1. X contains all $<_{\mathrm{TO}}$ predecessors of e_1, e_2
2. X is downwards closed wrt $<_{\mathrm{TRF}}$
3. No open critical sections except those containing e_1, e_2

	t_1	t_2	t_3
1	$w(y)$		
2	$\mathsf{acq}(\ell)$		
3	$w(x)$		
4	$\mathbf{e_1}$		
5	$\mathsf{rel}(\ell)$		
6		$w(y)$	
7		$r(y)$	
8			$\mathsf{acq}(\ell)$
9			$w(x)$
10			$w(y)$
11			$\mathsf{rel}(\ell)$
12		$r(x)$	
13		$\mathbf{e_2}$	

Choosing The Event Set

Witness = **Choose event set** + choose ordering

Heuristic: close all critical sections not containing e_1 and e_2

Choose X as the smallest set so that

1. X contains all $<_{\mathsf{TO}}$ predecessors of e_1, e_2
2. X is downwards closed wrt $<_{\mathsf{TRF}}$
3. No open critical sections except those containing e_1, e_2

	t_1	t_2	t_3
1	$w(y)$		
2	$\mathsf{acq}(\ell)$		
3	$w(x)$		
4	$\mathbf{e_1}$		
5	$\mathsf{rel}(\ell)$		
6		$w(y)$	
7		$r(y)$	
8			$\mathsf{acq}(\ell)$
9			$w(x)$
10			$w(y)$
11			$\mathsf{rel}(\ell)$
12		$r(x)$	
13		$\mathbf{e_2}$	

Choosing The Event Set

Witness = **Choose event set** + choose ordering

Heuristic: close all critical sections not containing e_1 and e_2

Choose X as the smallest set so that

1. X contains all $<_{\mathsf{TO}}$ predecessors of e_1, e_2
2. X is downwards closed wrt $<_{\mathsf{TRF}}$
3. No open critical sections except those containing e_1, e_2

Only 1 event subset

	t_1	t_2	t_3
1	$w(y)$		
2	$\mathsf{acq}(\ell)$		
3	$w(x)$		
4	$\mathbf{e_1}$		
5	$\mathsf{rel}(\ell)$		
6		$w(y)$	
7		$r(y)$	
8			$\mathsf{acq}(\ell)$
9			$w(x)$
10			$w(y)$
11			$\mathsf{rel}(\ell)$
12		$r(x)$	
13		$\mathbf{e_2}$	

Choosing The Event Set

Witness = **Choose event set** + choose ordering

Heuristic: close all critical sections not containing e_1 and e_2

Choose X as the smallest set so that

1. X contains all $<_{\mathrm{TO}}$ predecessors of e_1, e_2
2. X is downwards closed wrt $<_{\mathrm{TRF}}$
3. No open critical sections except those containing e_1, e_2

Only 1 event subset

If $\{e_1, e_2\} \cap X \neq \varnothing$ report "No race"

	t_1	t_2	t_3
1	$w(y)$		
2	$\mathrm{acq}(\ell)$		
3	$w(x)$		
4	$\mathbf{e_1}$		
5	$\mathrm{rel}(\ell)$		
6		$w(y)$	
7		$r(y)$	
8			$\mathrm{acq}(\ell)$
9			$w(x)$
10			$w(y)$
11			$\mathrm{rel}(\ell)$
12		$r(x)$	
13		$\mathbf{e_2}$	

Constructing The Witness

Witness = Choose event set + **choose ordering**

Constructing The Witness

Witness = Choose event set + **choose ordering**

Recall our solution for rf-poset realizability on acyclic topologies

Lemma

The closure of an RF-Poset can be computed in $O(\mathcal{T}^2 \cdot V \cdot \mathcal{N}^2 \cdot \log \mathcal{N})$ *time.*

Cycle

Not realizable

No Cycle

May be realizable

Realizable under extra conditions

Constructing The Witness

Witness = Choose event set + **choose ordering**

Recall our solution for rf-poset realizability on acyclic topologies

Lemma

The closure of an RF-Poset can be computed in $O(\mathcal{T}^2 \cdot V \cdot \mathcal{N}^2 \cdot \log \mathcal{N})$ *time.*

Cycle → Not realizable

No Cycle → May be realizable → Realizable under extra conditions

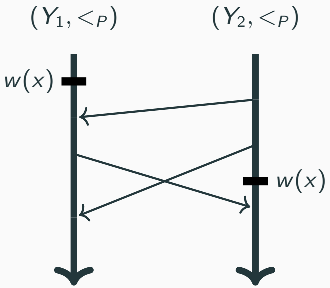
MaxMin **Linearizations**

Theorem (MaxMin **Linearizations)**

Take a closed rf-poset $(X, P, \mathcal{RF})$ where X is partitioned into Y_1, Y_2, where

1. $(Y_1, <_P)$ *is a total order*
2. $(Y_2, <_P)$ *orders all conflicting writes.*

Then $(X, P, \mathcal{RF})$ is realizable.



M2

Algorithm: M2

Input: A trace σ, a conflicting pair (e_1, e_2)

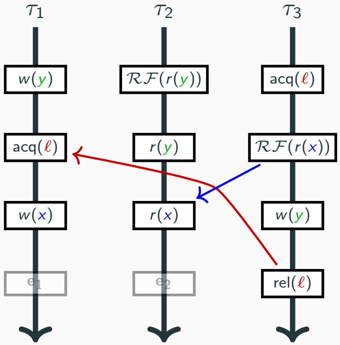
Algorithm: M2

Input: A trace σ, a conflicting pair (e_1, e_2)

Start with a partial order $\rightarrow$

(a) $\mathcal{RF}_\sigma(r) \rightarrow r$ for every read r

(b) $\mathsf{rel}_1 \rightarrow \mathsf{acq}_2$ if acq_2 does not close



Algorithm: M2

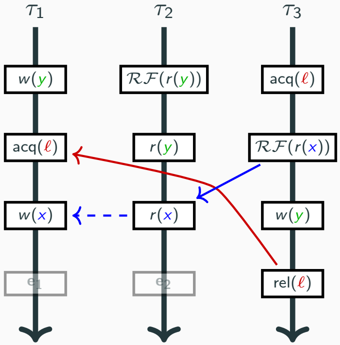
Input: A trace σ, a conflicting pair (e_1, e_2)

Start with a partial order $\rightarrow$

(a) $\mathcal{RF}_\sigma(r) \rightarrow r$ for every read r

(b) $\mathrm{rel}_1 \rightarrow \mathrm{acq}_2$ if acq_2 does not close

Close $\rightarrow$



Algorithm: M2

Input: A trace σ, a conflicting pair (e_1, e_2)

Start with a partial order $\rightarrow$

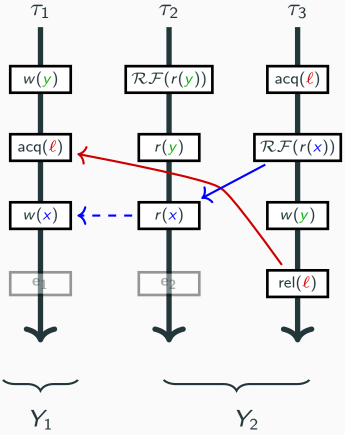
(a) $\mathcal{RF}_\sigma(r) \rightarrow r$ for every read r

(b) $\mathsf{rel}_1 \rightarrow \mathsf{acq}_2$ if acq_2 does not close

Close $\rightarrow$

Let Y_1 be all events in the thread of e_1

and Y_2 all other events



M2

Algorithm: M2

Input: A trace σ, a conflicting pair (e_1, e_2)

Start with a partial order $\rightarrow$

(a) $\mathcal{RF}_\sigma(r) \rightarrow r$ for every read r

(b) $\mathsf{rel}_1 \rightarrow \mathsf{acq}_2$ if acq_2 does not close

Close $\rightarrow$

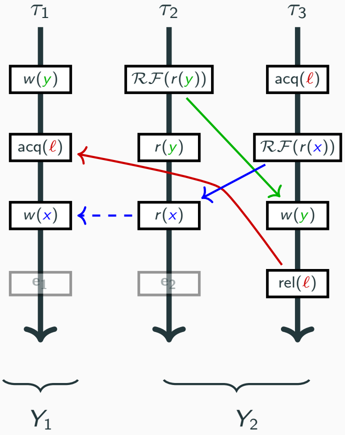
Let Y_1 be all events in the thread of e_1

and Y_2 all other events

while $\exists \overline{w}_1, \overline{w}_2 \in Y_{\neq i}$ *conflicting and unordered* **do**

| Order $\overline{e}_1, \overline{e}_2$ in σ as in σ, then close $\rightarrow$

end



M2

Algorithm: M2

Input: A trace σ, a conflicting pair (e_1, e_2)

Start with a partial order $\rightarrow$

(a) $\mathcal{RF}_\sigma(r) \rightarrow r$ for every read r

(b) $\mathsf{rel}_1 \rightarrow \mathsf{acq}_2$ if acq_2 does not close

Close $\rightarrow$

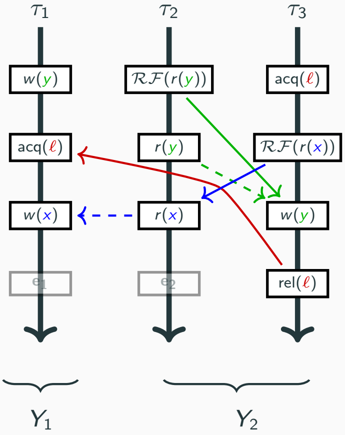
Let Y_1 be all events in the thread of e_1

and Y_2 all other events

while $\exists \overline{w}_1, \overline{w}_2 \in Y_{\neq i}$ *conflicting and unordered* **do**

| Order $\overline{e}_1, \overline{e}_2$ in σ as in σ, then close $\rightarrow$

end



Algorithm: M2

Input: A trace σ, a conflicting pair (e_1, e_2)

Start with a partial order $\to$

(a) $\mathcal{RF}_\sigma(r) \to r$ for every read r

(b) $\mathsf{rel}_1 \to \mathsf{acq}_2$ if acq_2 does not close

Close $\to$

Let Y_1 be all events in the thread of e_1

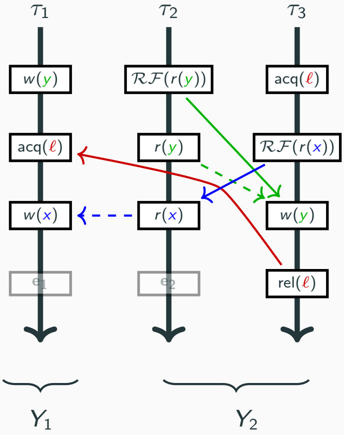
and Y_2 all other events

while $\exists \overline{w}_1, \overline{w}_2 \in Y_{\neq i}$ *conflicting and unordered* **do**

Order $\overline{e}_1, \overline{e}_2$ in σ as in σ, then close $\to$

end

Use MaxMin to linearize $\to$ to a witness σ^*



M2 **Illustration**

	t_1	t_2	t_3
1	$w(y)$		
2	$\mathrm{acq}(\ell)$		
3	$w(x)$		
4	$\mathbf{e_1}$		
5	$\mathrm{rel}(\ell)$		
6		$w(y)$	
7		$r(y)$	
8			$\mathrm{acq}(\ell)$
9			$w(x)$
10			$w(y)$
11			$\mathrm{rel}(\ell)$
12		$r(x)$	
13		$\mathbf{e_2}$	

Input trace

$\longrightarrow$ M2 $\longrightarrow$

	t_1	t_2	t_3
1	$w(y)$		
2		$w(y)$	
3		$r(y)$	
4			$\mathrm{acq}(\ell)$
5			$w(x)$
6			$w(y)$
7			$\mathrm{rel}(\ell)$
8	$\mathrm{acq}(\ell)$		
9		$r(x)$	
10	$w(x)$		
11	$\mathbf{e_1}$	$\mathbf{e_2}$	

Correct reordering

	t_1	t_2	t_3
1	$w(y)$		
2	$\mathsf{acq}(\ell)$		
3	$w(x)$		
4	$\mathbf{e}_1$		
5	$\mathsf{rel}(\ell)$		
6		$w(y)$	
7		$r(y)$	
8			$\mathsf{acq}(\ell)$
9			$w(x)$
10			$w(y)$
11			$\mathsf{rel}(\ell)$
12		$r(x)$	
13		$\mathbf{e}_2$	

Input trace

$\longrightarrow$ M2 $\longrightarrow$

	t_1	t_2	t_3
1	$w(y)$		
2		$w(y)$	
3		$r(y)$	
4			$\mathsf{acq}(\ell)$
5			$w(x)$
6			$w(y)$
7			$\mathsf{rel}(\ell)$
8	$\mathsf{acq}(\ell)$		
9		$r(x)$	
10	$w(x)$		
11	$\mathbf{e}_1$	$\mathbf{e}_2$	

Correct reordering

Theorem

M2 *makes sound predictions of data races with complexity* $O(\mathcal{T}^2 \cdot V \cdot \mathcal{N}^4 \cdot \log \mathcal{N})$.

- *For* $\mathcal{T} = 2$ *threads,* M2 *is also complete.*

Experiments

Benchmark	Size	HB	WCP	DC	SHB	**M2**
		Races	**Races**	**Races**	**Races**	**Races**
`critical`	49	3	3	3	8	
`airtickets`	116	3	3	3	4	
`account`	125	1	1	1	1	
`pingpong`	126	2	2	2	2	
`bbuffer`	322	2	2	2	2	
`mergesort`	3K	1	1	1	1	
`bubblesort`	4K	4	4	5	6	
`raytracer`	16K	3	3	3	3	
`ftpserver`	48K	23	23	24	23	
`moldyn`	164K	2	2	2	2	
`derby`	1M	12	12	12	12	
`jigsaw`	3M	8	10	10	9	
`bufwriter`	11M	2	2	2	2	
`hsqldb`	18M	4	4	5	9	
`cryptorsa`	57M	5	5	7	5	
`eclipse`	86M	33	34	39	54	
`xalan`	122M	7	7	9	11	
`lusearch`	216M	30	30	30	52	
Total	**514M**	**145**	**148**	**160**	**206**	

Experiments

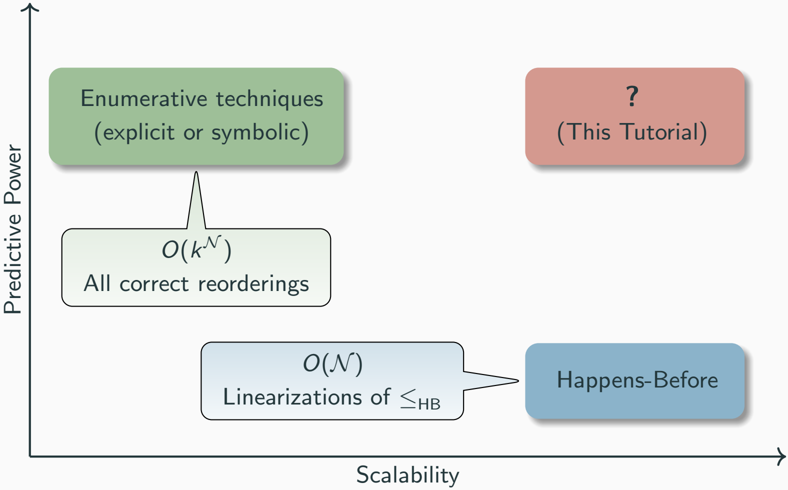
Benchmark	Size	HB	WCP	DC	SHB	**M2**
		Races	**Races**	**Races**	**Races**	**Races**
critical	49	3	3	3	8	8
airtickets	116	3	3	3	4	4
account	125	1	1	1	1	1
pingpong	126	2	2	2	2	2
bbuffer	322	2	2	2	2	2
mergesort	3K	1	1	1	1	2
bubblesort	4K	4	4	5	6	6
raytracer	16K	3	3	3	3	3
ftpserver	48K	23	23	24	23	26
moldyn	164K	2	2	2	2	2
derby	1M	12	12	12	12	12
jigsaw	3M	8	10	10	9	11
bufwriter	11M	2	2	2	2	2
hsqldb	18M	4	4	5	9	9
cryptorsa	57M	5	5	7	5	7
eclipse	86M	33	34	39	54	67
xalan	122M	7	7	9	11	15
lusearch	216M	30	30	30	52	52
Total	**514M**	**145**	**148**	**160**	**206**	**231**

Times	HB	WCP	DC	SHB	**M2**
Total	42m0s	34m14s	3h39m	40m31s	1h15m

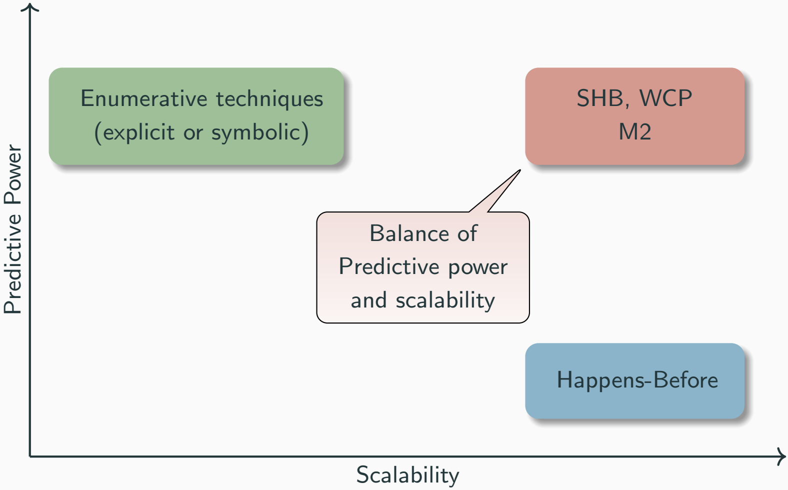
Conclusion

Wrap-up and a small teaser

Advances in Predictive Analysis (This Tutorial)



Advances in Predictive Analysis (This Tutorial)



Other Recent Works

- Doesn't commute (DC) [RGB18]
- SDP, WDP [Gen+19]
- WDC [RGB20]
- DigHR [Luo+18]
- All unsound - soundness is a challenge

Open Challenges

Open Challenges

1. Most efficient race detection

- The most efficient detectors must operate in linear time
- Challenge: Sound, linear-time + as precise as possible

Open Challenges

1. Most efficient race detection

- The most efficient detectors must operate in linear time
- Challenge: Sound, linear-time + as precise as possible

2. Relaxing the notion of witnesses

- Utilize static information for increased precision (+ efficiency)
- Relax the $\mathcal{RF}$ function to be based on values

Open Challenges

1. Most efficient race detection

- The most efficient detectors must operate in linear time
- Challenge: Sound, linear-time + as precise as possible

2. Relaxing the notion of witnesses

- Utilize static information for increased precision (+ efficiency)
- Relax the $\mathcal{RF}$ function to be based on values

3. Other Violations

- Data races are the most well studied
- Deadlocks?
- Atomicity?
- ...

Open Challenges (Cont.)

4. Relaxed Memory Models

More permissive reorderings $\rightarrow$ more data races

Open Challenges (Cont.)

4. Relaxed Memory Models

More permissive reorderings $\rightarrow$ more data races

5. Mixing Paradigms

- Mixing shared memory and message passing becomes the norm
  - E.g., in Go lang
- Concurrent data structures

Open Challenges (Cont.)

4. Relaxed Memory Models

More permissive reorderings $\rightarrow$ more data races

5. Mixing Paradigms

- Mixing shared memory and message passing becomes the norm
  - E.g., in Go lang
- Concurrent data structures

6. More benchmarks

- DaCapo
- JavaGrande
- SIR
- ...
- new?

Synchronization-Preserving Races

- HB principle: "do not reorder critical sections"
- Not complete for this principle
- Here a race is missed
- Exposed without reordering critical sections

	t_1	t_2
1	$\mathbf{w}(\mathbf{x})$	
2	acq(ℓ)	
3	rel(ℓ)	
4		acq(ℓ)
5		rel(ℓ)
6		$\mathbf{w}(\mathbf{x})$

Synchronization-Preserving Races

- HB principle: "do not reorder critical sections"
- Not complete for this principle
- Here a race is missed
- Exposed without reordering critical sections

	t_1	t_2
1	$\mathbf{w}(\mathbf{x})$	
2	acq(ℓ)	
3	rel(ℓ)	
4		acq(ℓ)
5		rel(ℓ)
6		$\mathbf{w}(\mathbf{x})$

HB

What is the cost of a complete algorithm for this principle?

U. Mathur, A. Pavlogiannis, and M. Viswanathan. "Optimal Prediction of Synchronization-Preserving Races". In: POPL '21. To Appear. 2021

Tune in for the POPL talk!

- Watch the talk https://app.clowdr.org/conference/popl2021/item/6709324c-cd56-4d26-b2b3-266fa5e668f5
- Q&A on Thu 21 Jan 16:00 - 17:00 (CET): Concurrency (Shared Memory) at POPL-B

Literature Basis For This Tutorial I

- J. FIDGE. “Timestamps in message-passing systems that preserve the partial ordering”. In: *Proc. 11th Australian Comput. Science Conf.* 1988, pp. 56–66.
- D. Kini, U. Mathur, and M. Viswanathan. “Dynamic Race Prediction in Linear Time”. In: *Proceedings of the 38th ACM SIGPLAN Conference on Programming Language Design and Implementation*. PLDI 2017. ACM, 2017, pp. 157–170.
- L. Lamport. “Time, Clocks, and the Ordering of Events in a Distributed System”. In: *Commun. ACM* 7 (1978), pp. 558–565.
- F. Mattern. “Virtual Time and Global States of Distributed Systems”. In: *Parallel and Distributed Algorithms: proceedings of the International Workshop on Parallel & Distributed Algorithms*. Elsevier Science Publishers B. V., 1989, pp. 215–226.
- U. Mathur, D. Kini, and M. Viswanathan. “What Happens-after the First Race? Enhancing the Predictive Power of Happens-before Based Dynamic Race Detection”. In: *Proc. ACM Program. Lang.* OOPSLA (2018), 145:1–145:29.
- U. Mathur, A. Pavlogiannis, and M. Viswanathan. “The Complexity of Dynamic Data Race Prediction”. In: *Proceedings of the 35th Annual ACM/IEEE Symposium on Logic in Computer Science*. LICS ’20. Association for Computing Machinery, 2020, pp. 713–727.
- A. Pavlogiannis. “Fast, Sound, and Effectively Complete Dynamic Race Prediction”. In: *Proc. ACM Program. Lang.* POPL (2019).

Other Related Literature I

M. Christiaens and K. D. Bosschere. "TRaDe: Data Race Detection for Java". In: *Proceedings of the International Conference on Computational Science-Part II*. ICCS '01. Springer-Verlag, 2001, pp. 761–770.

T. Elmas, S. Qadeer, and S. Tasiran. "Goldilocks: A Race and Transaction-aware Java Runtime". In: *Proceedings of the 28th ACM SIGPLAN Conference on Programming Language Design and Implementation*. PLDI '07. ACM, 2007, pp. 245–255.

C. Flanagan and S. N. Freund. "FastTrack: Efficient and Precise Dynamic Race Detection". In: *Proceedings of the 30th ACM SIGPLAN Conference on Programming Language Design and Implementation*. PLDI '09. ACM, 2009, pp. 121–133.

K. Genç et al. "Dependence-Aware, Unbounded Sound Predictive Race Detection". In: *Proc. ACM Program. Lang.* OOPSLA (2019).

J. Huang, P. O. Meredith, and G. Rosu. "Maximal Sound Predictive Race Detection with Control Flow Abstraction". In: *Proceedings of the 35th ACM SIGPLAN Conference on Programming Language Design and Implementation*. PLDI '14. ACM, 2014, pp. 337–348.

P. Luo et al. "DigHR: precise dynamic detection of hidden races with weak causal relation analysis." In: *J. Supercomput.* 6 (2018), pp. 2684–2704.

U. Mathur, A. Pavlogiannis, and M. Viswanathan. "Optimal Prediction of Synchronization-Preserving Races". In: POPL '21. To Appear. 2021.

Other Related Literature II

- J. Roemer, K. Genç, and M. D. Bond. "High-coverage, Unbounded Sound Predictive Race Detection". In: *Proceedings of the 39th ACM SIGPLAN Conference on Programming Language Design and Implementation*. PLDI 2018. ACM, 2018, pp. 374–389.
- J. Roemer, K. Genç, and M. D. Bond. "SmartTrack: Efficient Predictive Race Detection". In: *Proceedings of the 41st ACM SIGPLAN Conference on Programming Language Design and Implementation*. PLDI 2020. Association for Computing Machinery, 2020, pp. 747–762.
- Y. Smaragdakis et al. "Sound Predictive Race Detection in Polynomial Time". In: *Proceedings of the 39th Annual ACM SIGPLAN-SIGACT Symposium on Principles of Programming Languages*. POPL '12. ACM, 2012, pp. 387–400.
- M. Sulzmann and K. Stadtmüller. "Predicting All Data Race Pairs for a Specific Schedule". In: *Proceedings of the 16th ACM SIGPLAN International Conference on Managed Programming Languages and Runtimes*. MPLR 2019. Association for Computing Machinery, 2019, pp. 72–84.
- M. Sulzmann and K. Stadtmüller. "Efficient, near Complete, and Often Sound Hybrid Dynamic Data Race Prediction". In: *Proceedings of the 17th International Conference on Managed Programming Languages and Runtimes*. Association for Computing Machinery, 2020, pp. 30–51.

Thank You!