Fast, Sound, and Effectively Complete Dynamic Race Prediction

Andreas Pavlogiannis

AARHUS UNIVERSITY

January 23, 2020

Concurrency is Hard

Thread 1: Withdraw(x)

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

Concurrency is Hard

Thread 1: Withdraw(x)

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

Thread 2: Withdraw(x)

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

Concurrency is Hard

Thread 1: Withdraw(x)

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

Thread 2: Withdraw(x)

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

Withdraw(5)

Withdraw(5)

balance $= 8$

Concurrency is Hard

Thread 1: Withdraw(x)

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

Thread 2: Withdraw(x)

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

Withdraw(5)

Withdraw(5)

balance = 8

Concurrency is Hard

Thread 1: Withdraw(x)

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

Thread 2: Withdraw(x)

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

Withdraw(5)

Withdraw(5)

balance = 8

Concurrency is Hard

Thread 1: Withdraw(x)

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

Thread 2: Withdraw(x)

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

Withdraw(5)

Withdraw(5)

$$\text{balance} = 8 \rightarrow 3$$

Concurrency is Hard

Thread 1: Withdraw(x)

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

Thread 2: Withdraw(x)

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

Withdraw(5)

Withdraw(5)

$$\text{balance} = 8 \rightarrow 3 \rightarrow -2$$

Concurrency is Hard

Thread 1: Withdraw(x)

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

Thread 2: Withdraw(x)

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

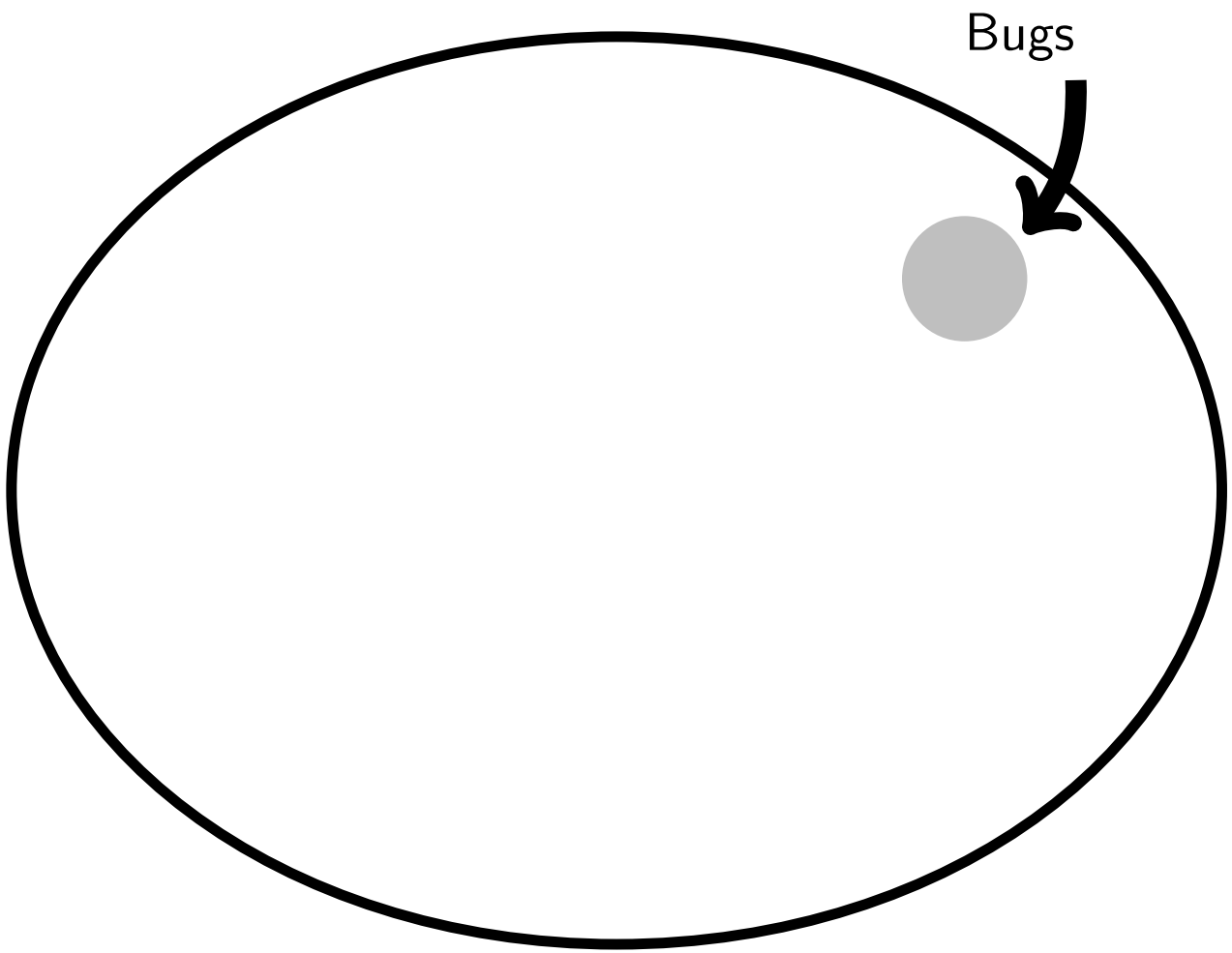
Withdraw(5)

Withdraw(5)

$$\text{balance} = 8 \rightarrow 3 \rightarrow -2$$

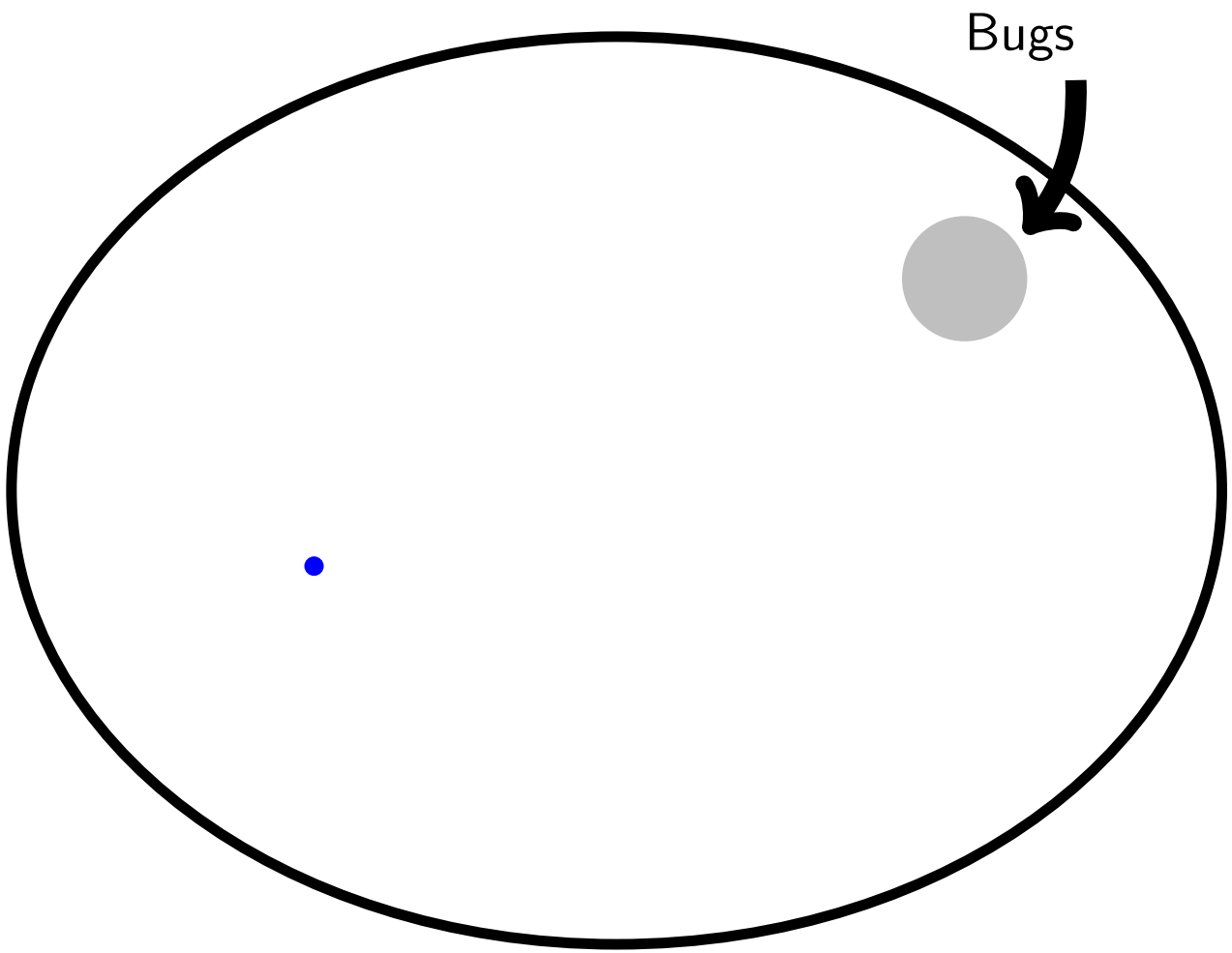
- No control on scheduler

Testing: Shooting Darts in the Dark



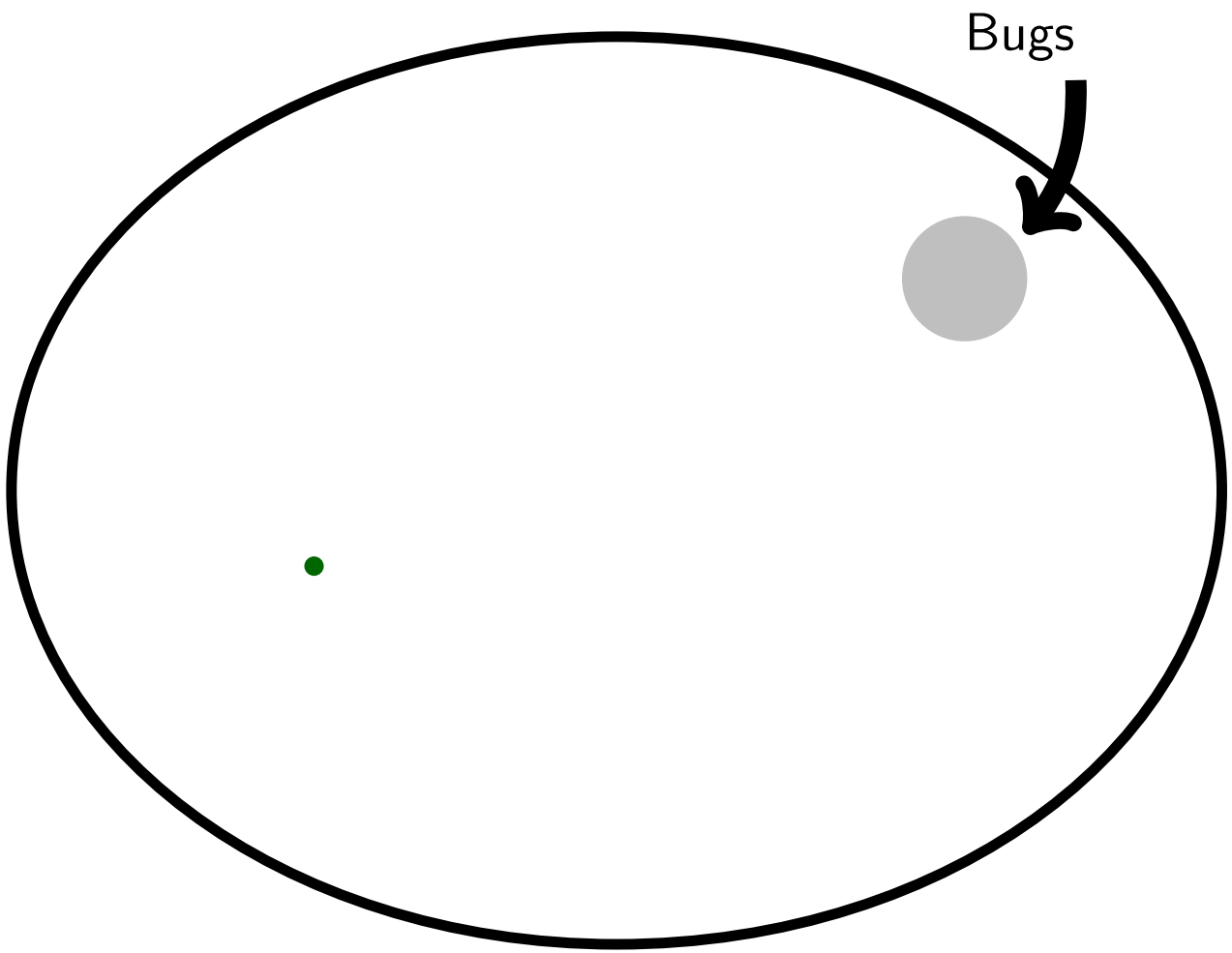
- Let me run it
- Let me try again
- Must be correct

Testing: Shooting Darts in the Dark



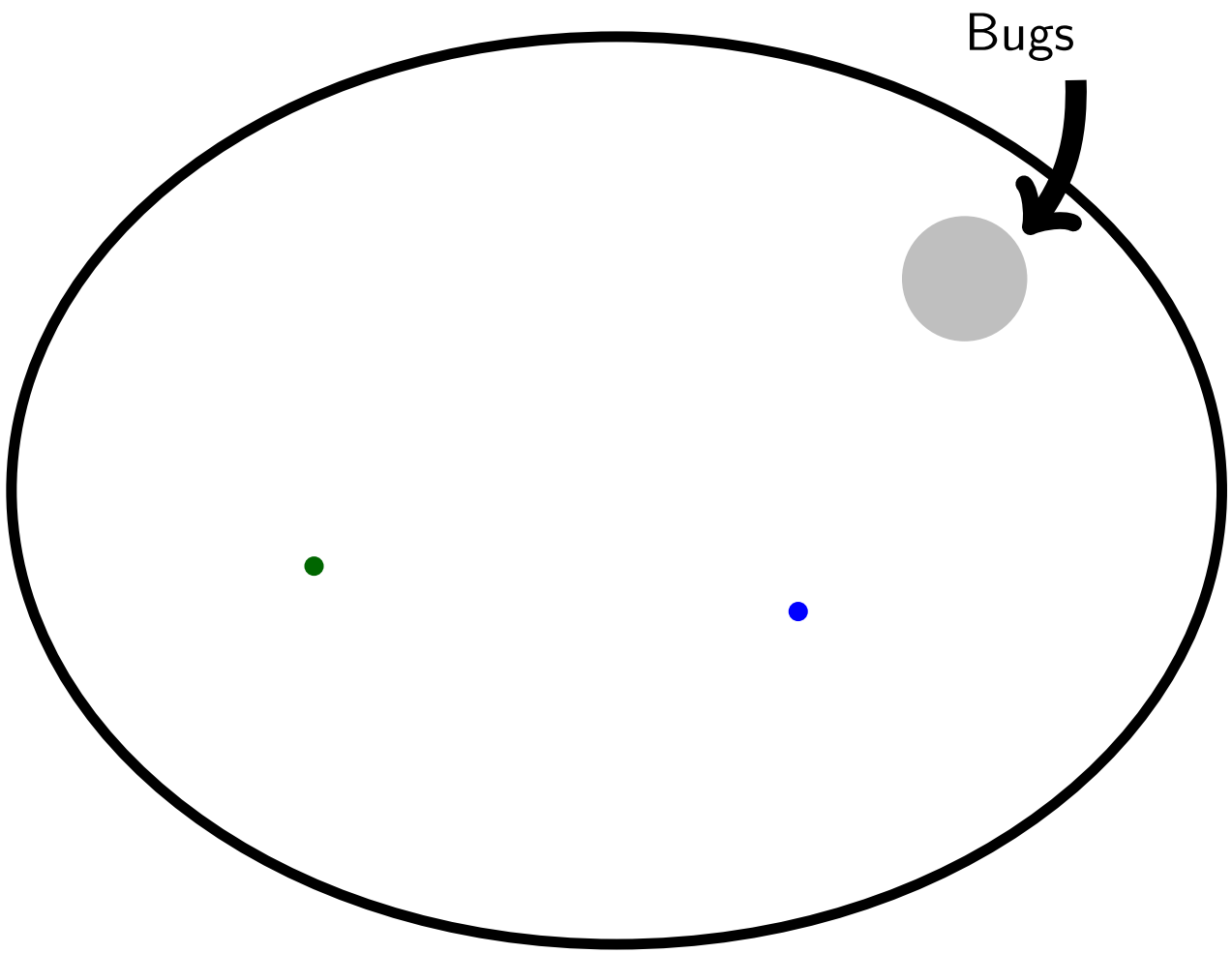
- Let me run it
- Let me try again
- Must be correct

Testing: Shooting Darts in the Dark



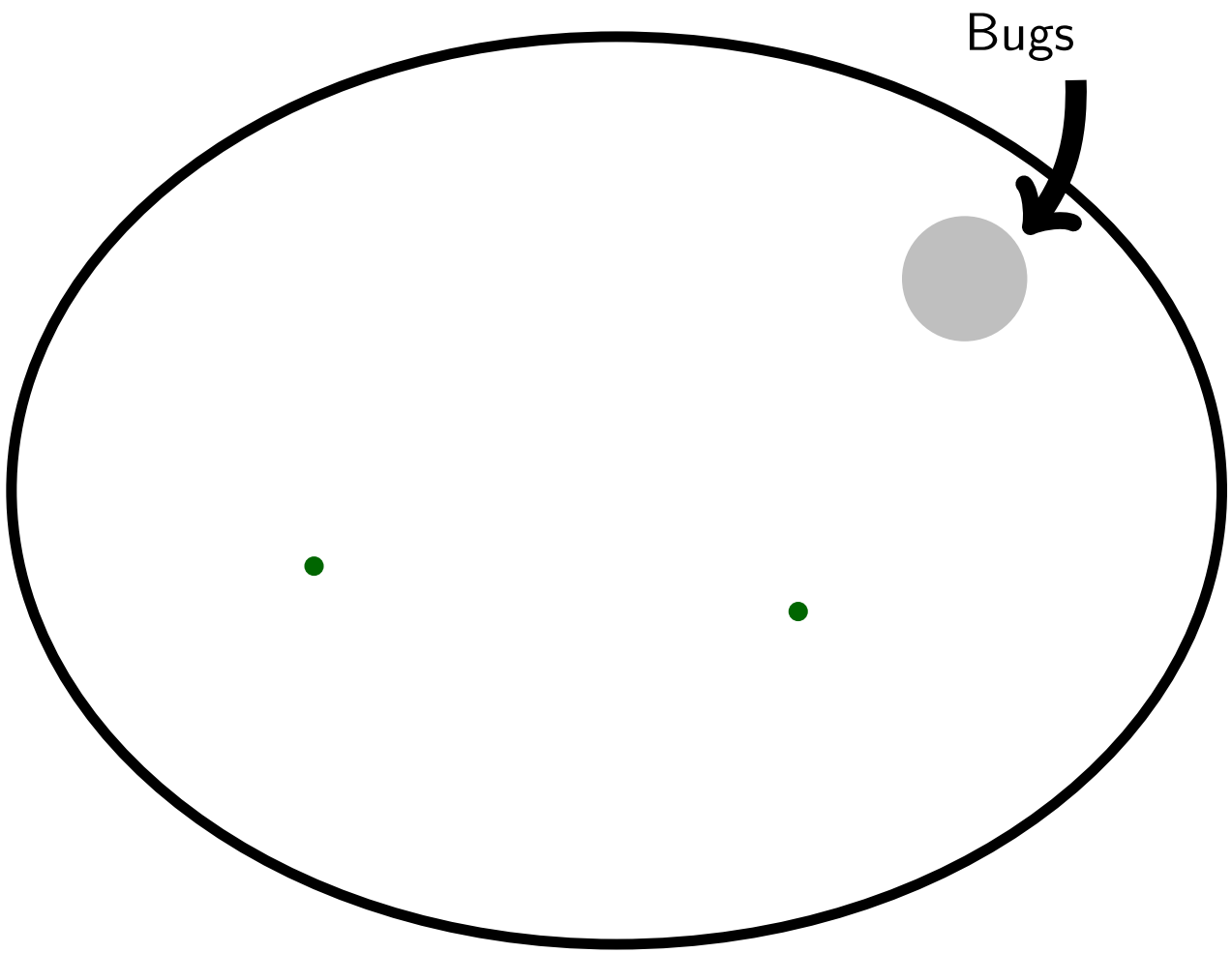
- Let me run it
- Let me try again
- Must be correct

Testing: Shooting Darts in the Dark



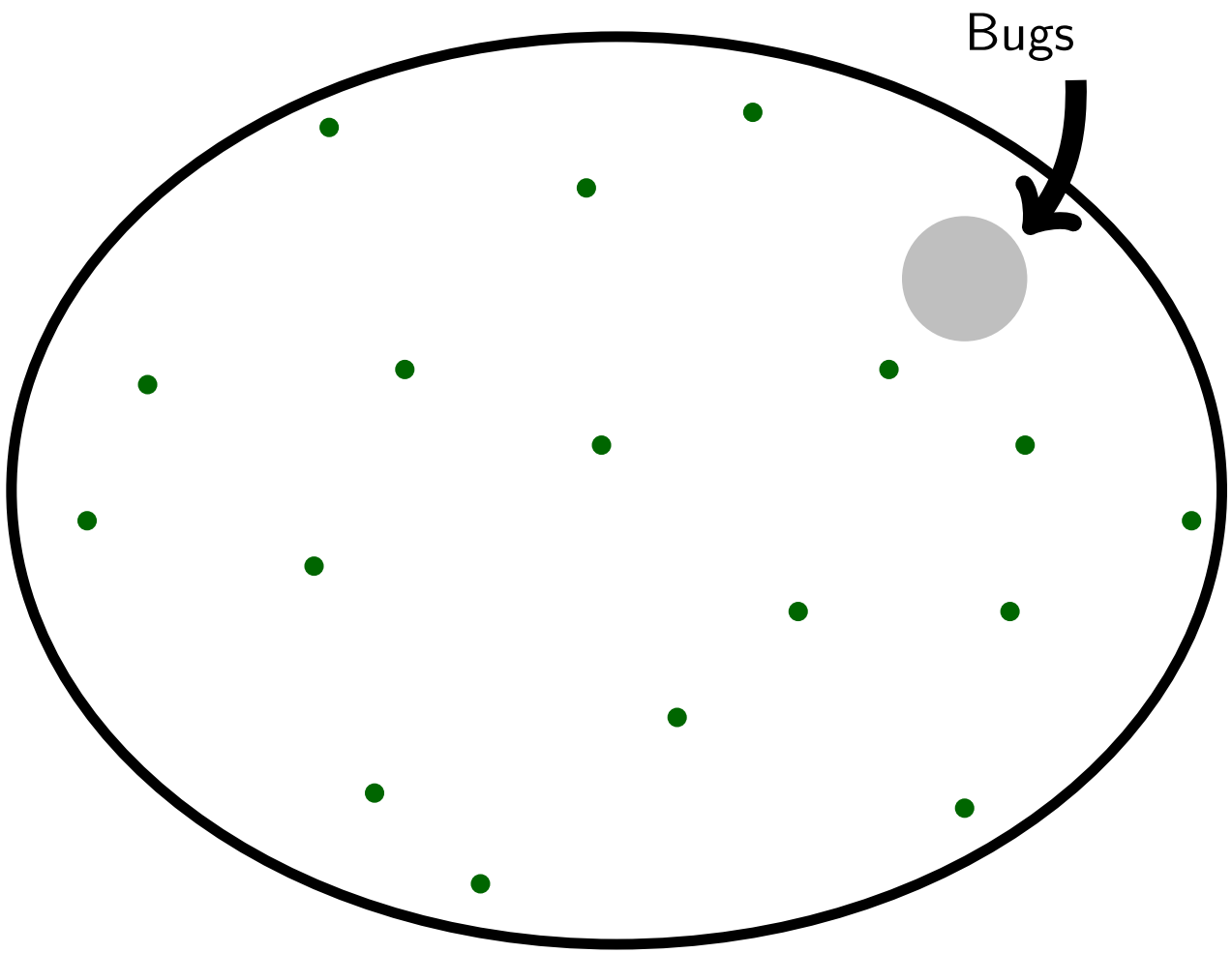
- Let me run it
- Let me try again
- Must be correct

Testing: Shooting Darts in the Dark



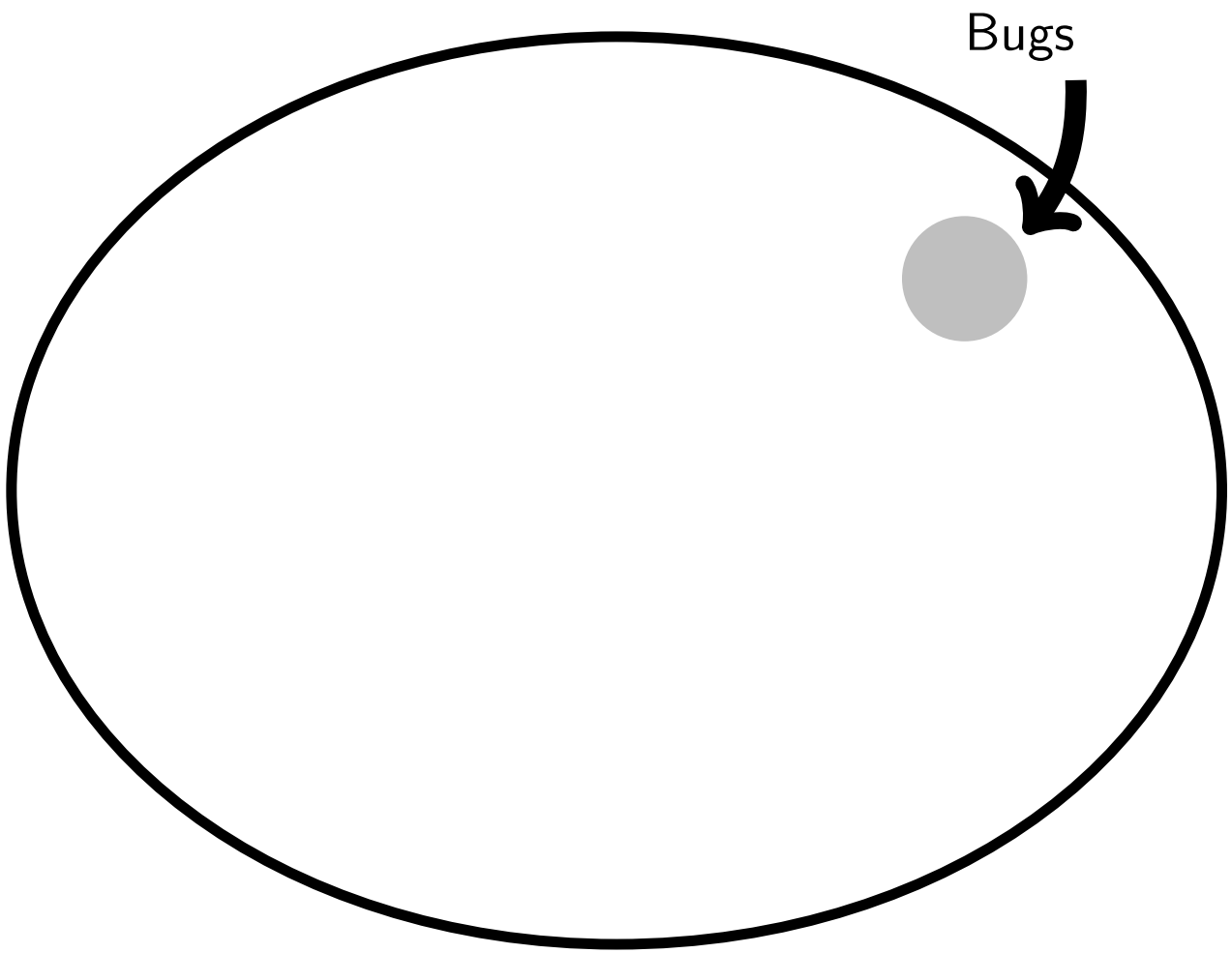
- Let me run it
- Let me try again
- Must be correct

Testing: Shooting Darts in the Dark



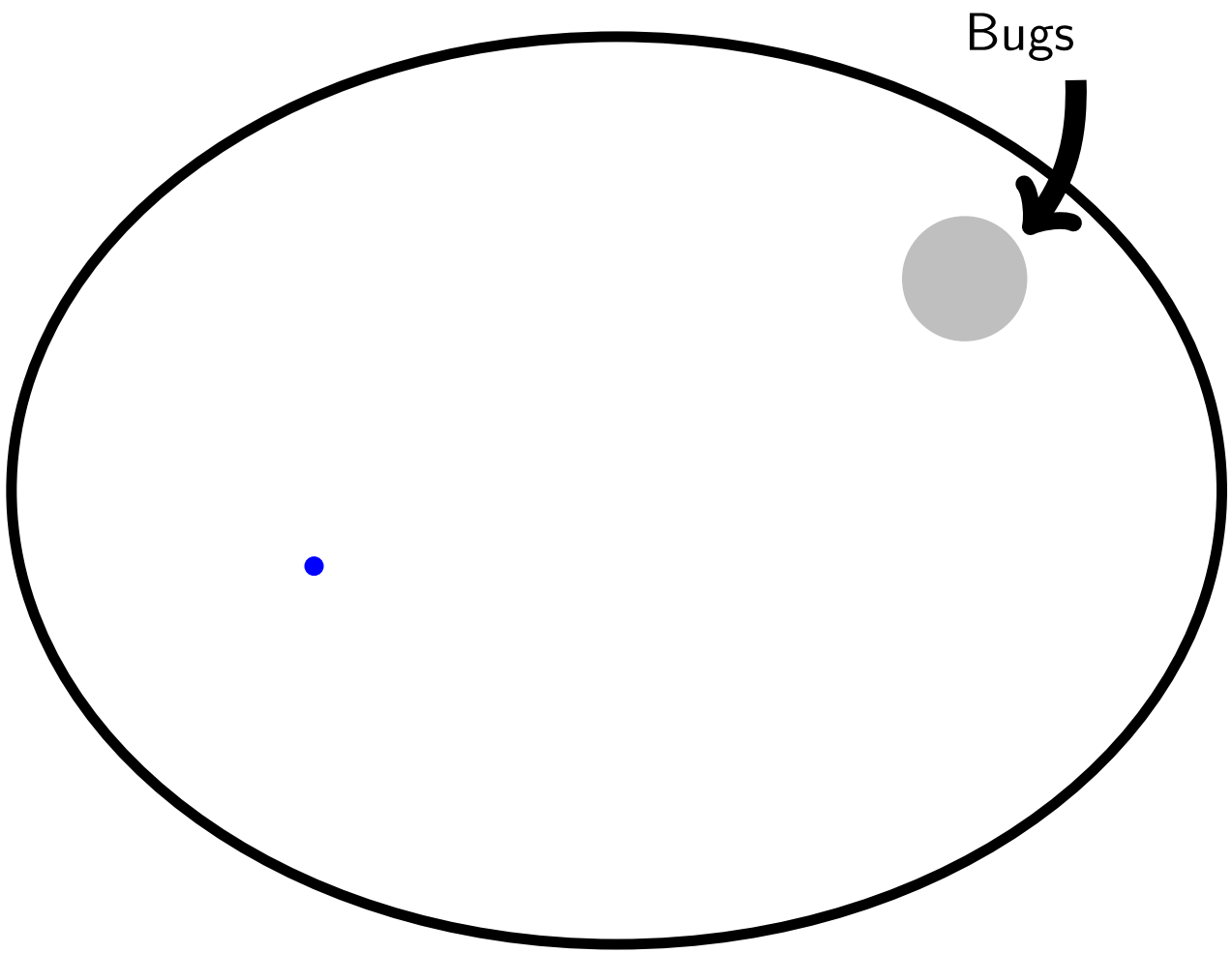
- Let me run it
- Let me try again
- Must be correct

Predictive Techniques: The Darts get Fatter



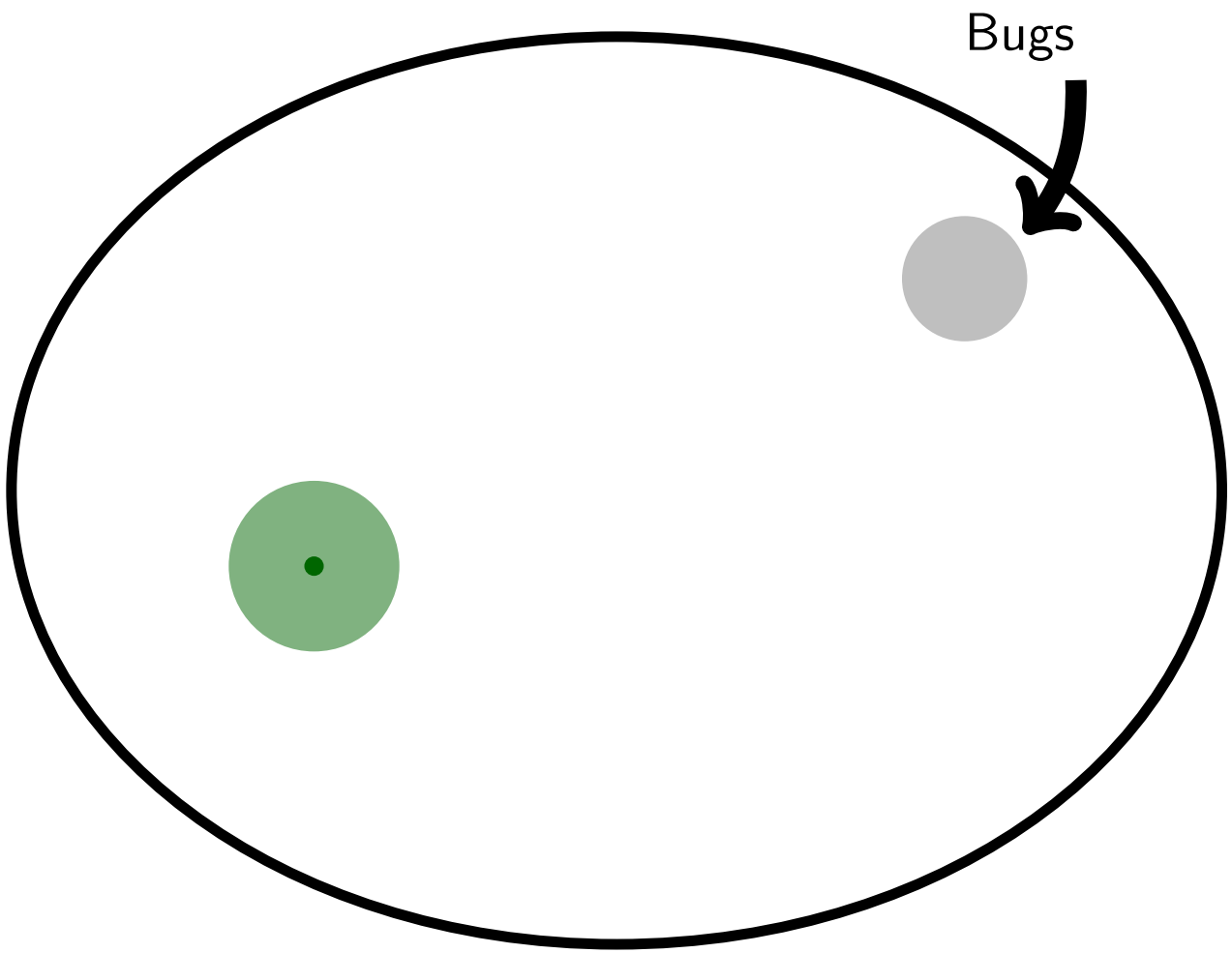
- Let me run it
- Reason about neighborhood
- **Bug!**

Predictive Techniques: The Darts get Fatter



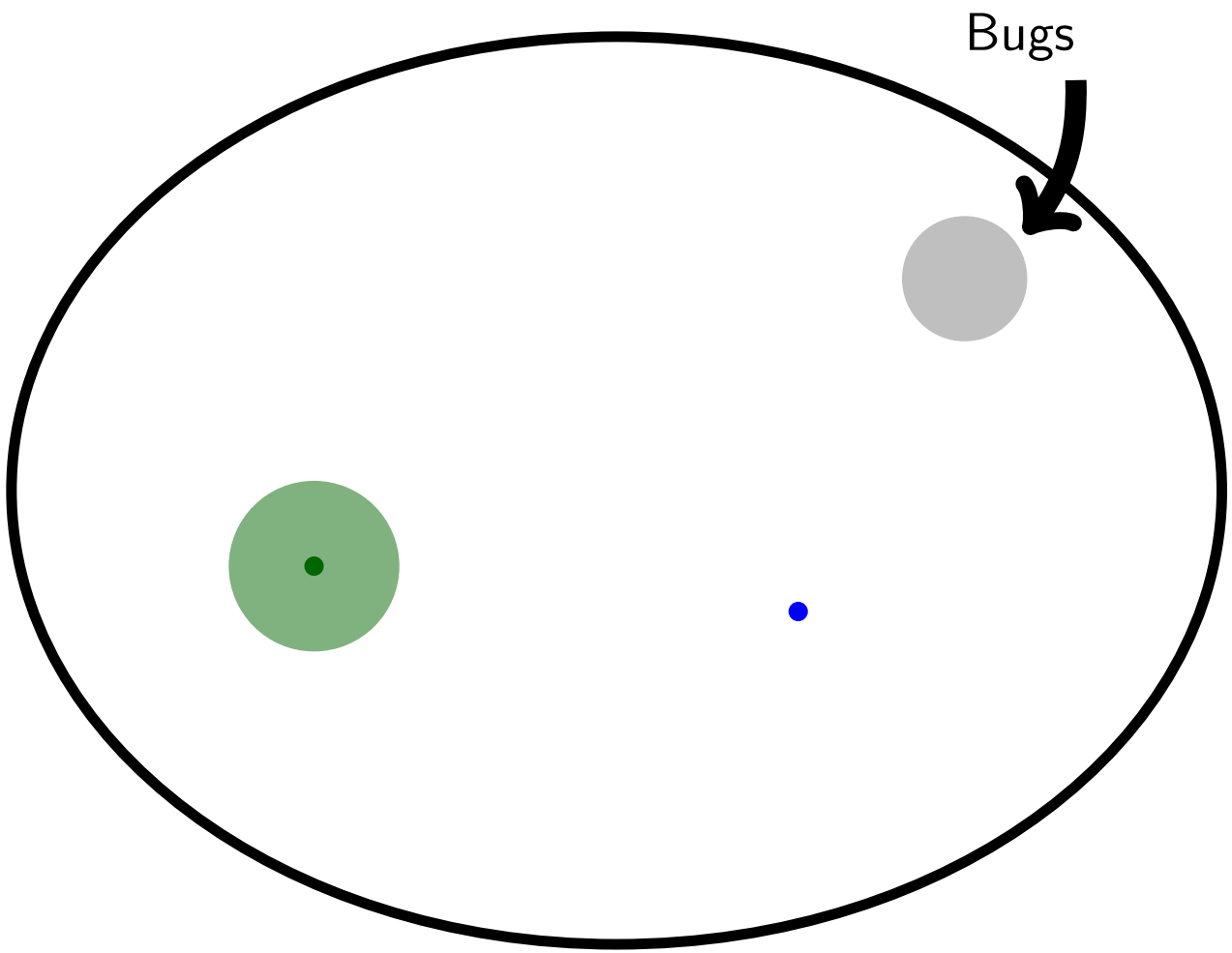
- Let me run it
- Reason about neighborhood
- **Bug!**

Predictive Techniques: The Darts get Fatter



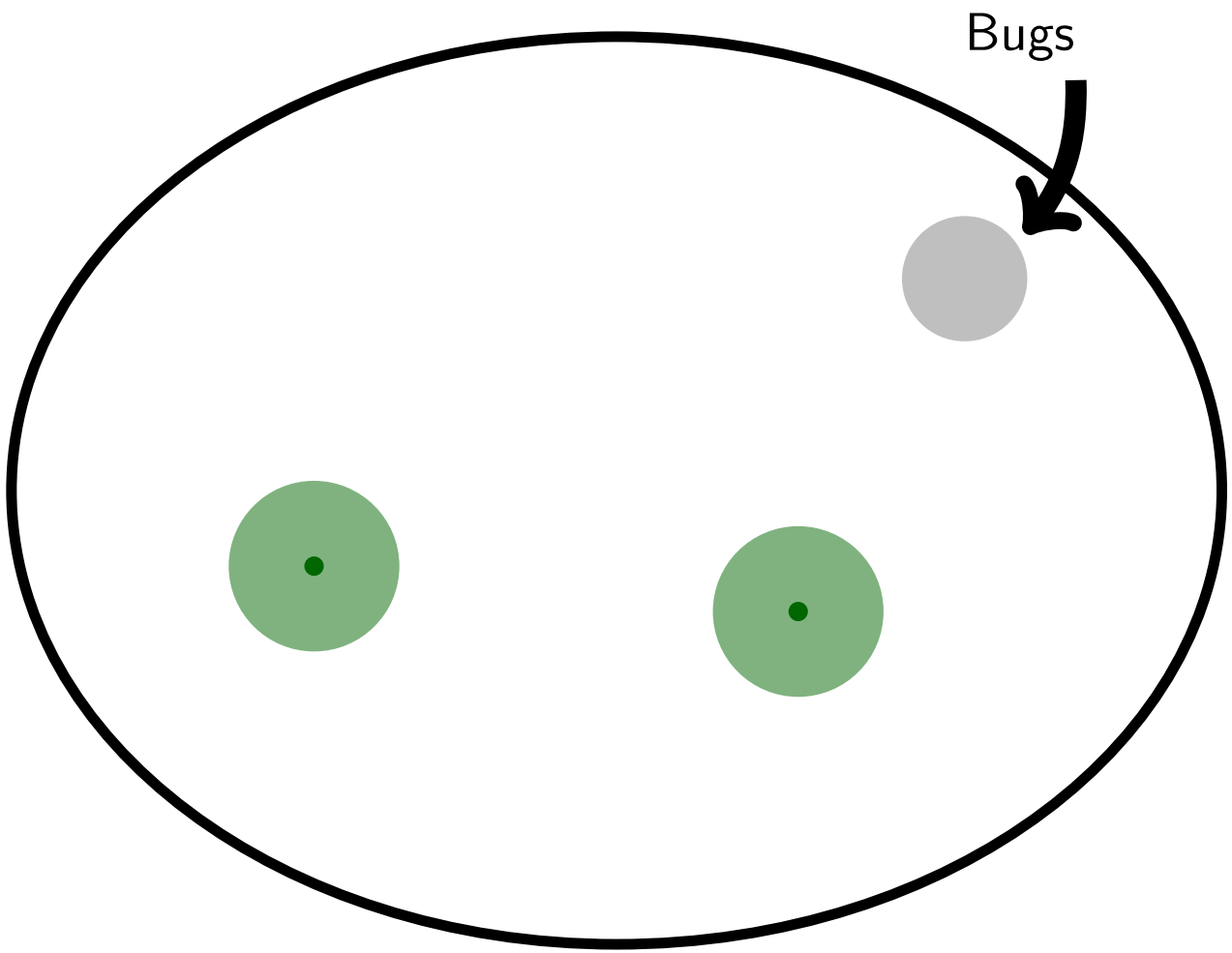
- Let me run it
- Reason about neighborhood
- Bug!

Predictive Techniques: The Darts get Fatter



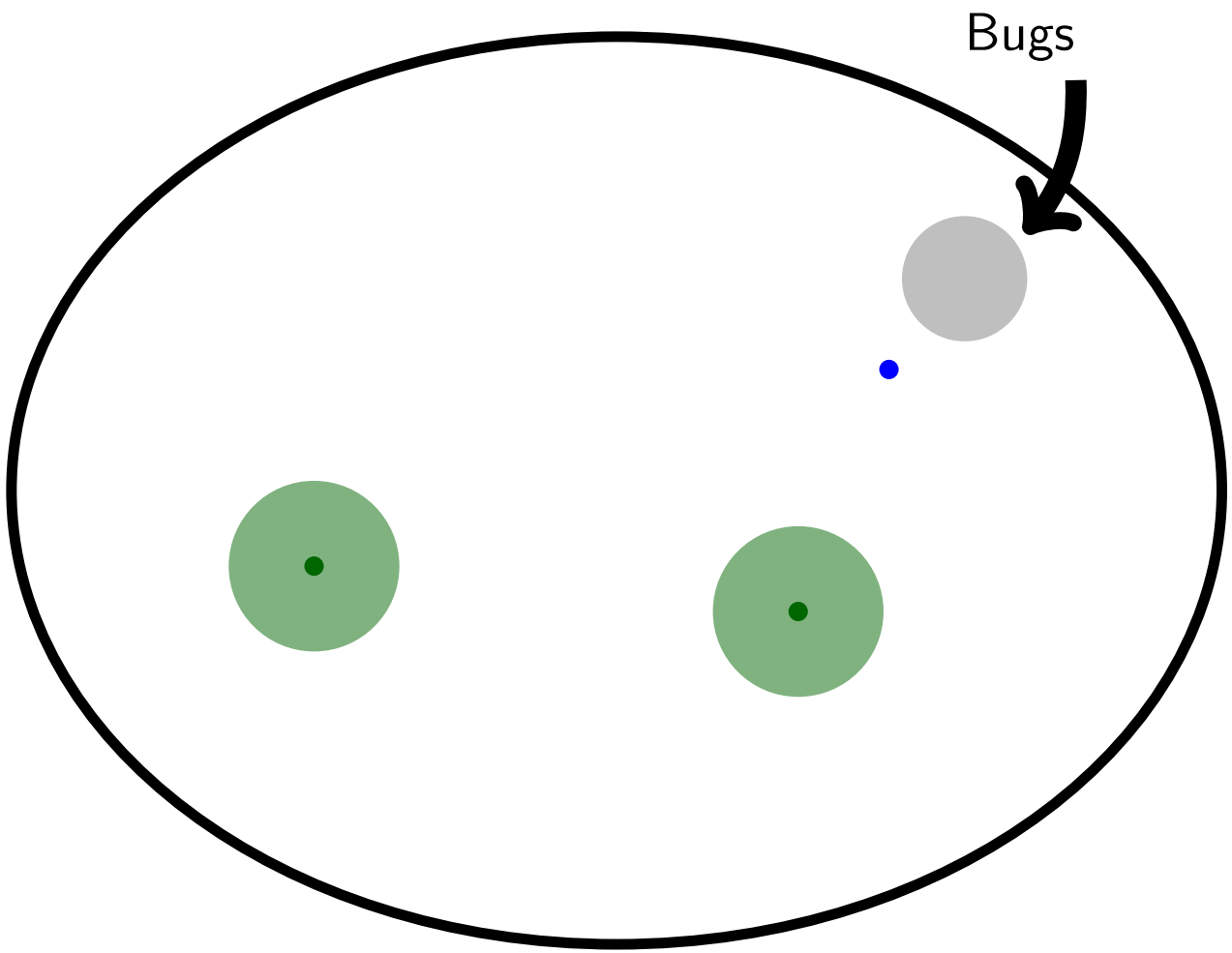
- Let me run it
- Reason about neighborhood
- Bug!

Predictive Techniques: The Darts get Fatter



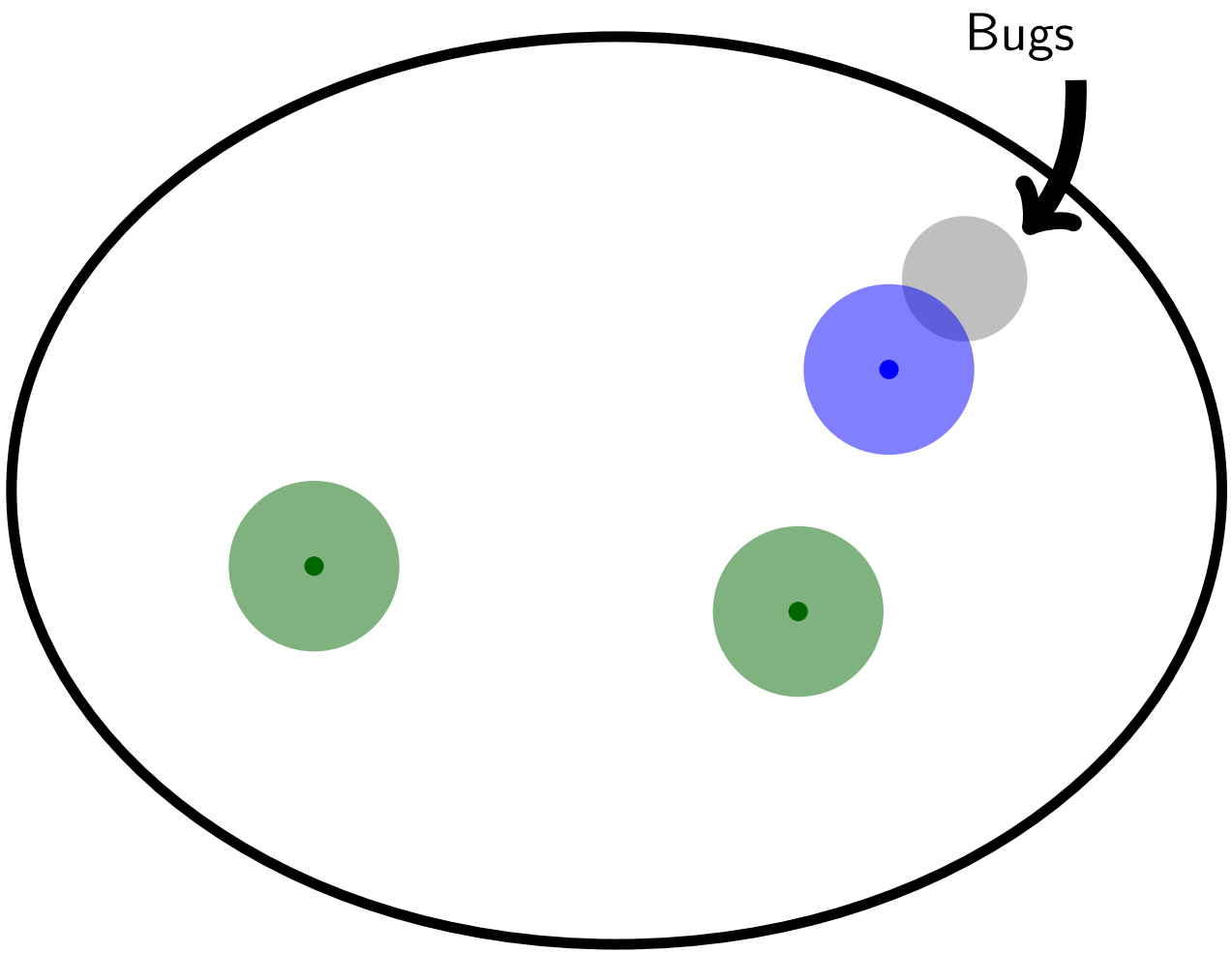
- Let me run it
- Reason about neighborhood
- Bug!

Predictive Techniques: The Darts get Fatter



- Let me run it
- Reason about neighborhood
- Bug!

Predictive Techniques: The Darts get Fatter



- Let me run it
- Reason about neighborhood
- **Bug!**

Predicting Data Races

	τ_1	τ_2
		$\mathsf{acq}(\ell)$
		$w(x)$
		$r(x)$
		$\mathsf{rel}(\ell)$
	$\mathsf{acq}(\ell)$	
	$\mathbf{w(x)}$	$\mathbf{r(x)}$

Predicting Data Races

	τ_1	τ_2
	acq(ℓ)	
	w(x)	
	rel(ℓ)	
		acq(ℓ)
		$w(x)$
		$r(x)$
		rel(ℓ)
		r(x)

We saw this . . .

	τ_1	τ_2
		acq(ℓ)
		$w(x)$
		$r(x)$
		rel(ℓ)
	acq(ℓ)	
	w(x)	**r(x)**

. . . we predicted this

Plenty of Literature

1. M. Christiaens and K. D. Bosschere. "TRaDe: Data Race Detection for Java". In: ICCS. 2001
2. E. Pozniansky and A. Schuster. "Efficient On-the-fly Data Race Detection in Multithreaded C++ Programs". In: *SIGPLAN Not.* (2003)
3. T. Elmas, S. Qadeer, and S. Tasiran. "Goldilocks: A Race and Transaction-aware Java Runtime". In: PLDI. 2007
4. F. Chen, T. F. Serbanuta, and G. Rosu. "jPredictor: A Predictive Runtime Analysis Tool for Java". In: ICSE. 2008
5. C. Flanagan and S. N. Freund. "FastTrack: Efficient and Precise Dynamic Race Detection". In: PLDI. 2009
6. Y. Smaragdakis et al. "Sound Predictive Race Detection in Polynomial Time". In: POPL. 2012
7. J. Huang, P. O. Meredith, and G. Rosu. "Maximal Sound Predictive Race Detection with Control Flow Abstraction". In: PLDI. 2014
8. D. Kini, U. Mathur, and M. Viswanathan. "Dynamic Race Prediction in Linear Time". In: PLDI. 2017
9. J. Roemer, K. Genç, and M. D. Bond. "High-coverage, Unbounded Sound Predictive Race Detection". In: PLDI. 2018
10. 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)
11. K. Genç et al. "Dependence-aware, Unbounded Sound Predictive Race Detection". In: *Proc. ACM Program. Lang.* OOPSLA (2019)

Approaches

Fast heuristics

- Polynomial-time
- Unsound (false positives)
- Incomplete (false negatives)

Incompleteness is ok

This Paper

M2: a new algorithm for dynamic data-race prediction

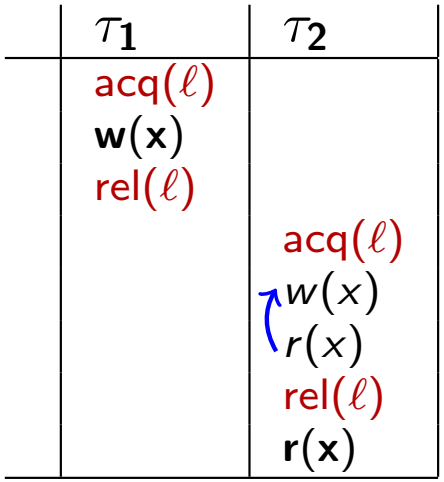
Theory

1. Polynomial-time
2. Sound
3. Complete for 2 threads

Practice

4. Criteria for over-approximating false negatives
5. Practically complete on a challenging data set

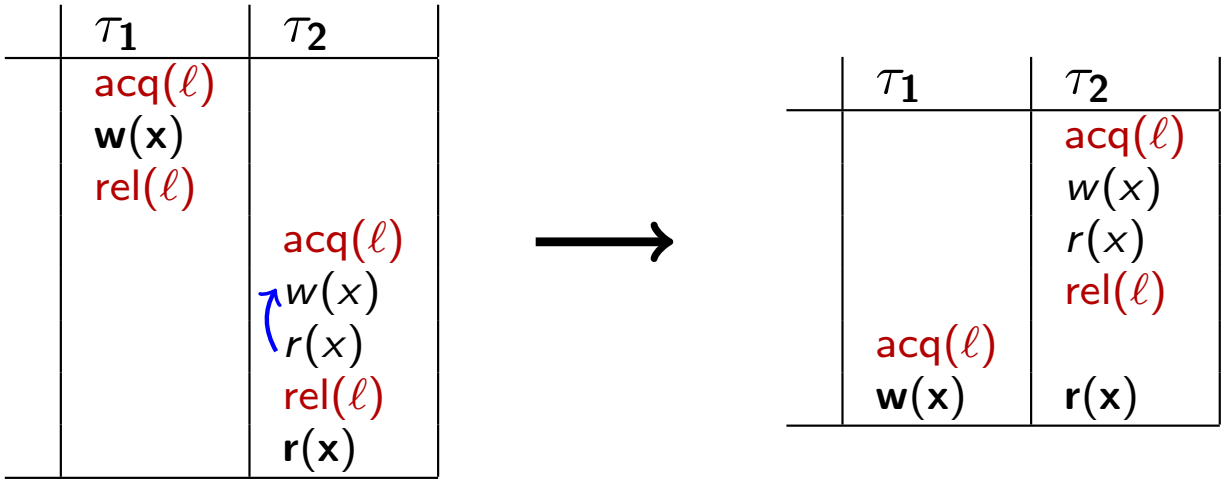
Data Race Prediction



Input

- A trace t
- A local execution τ_i in thread i
- An observation function $\mathcal{O}_t : \text{Reads} \to \text{Writes}$
- A conflicting pair (e_1, e_2)

Data Race Prediction



Input

- A trace t
- A local execution τ_i in thread i
- An observation function $\mathcal{O}_t : \text{Reads} \to \text{Writes}$
- A conflicting pair (e_1, e_2)

Output

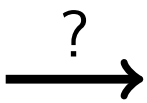
- A witness trace t^*
- Executes a prefix of each thread
- $\mathcal{O}_{t^*}$ agrees with $\mathcal{O}_t$
- e_1, e_2 appear at the end

Partial-order Approaches

1. Build a partial order on the events of t
2. Report race iff $e_1 \not\to e_2$.

Miss simple races

	τ_1	τ_2
	acq(ℓ)	
	$w(y)$	
	w(x)	
	rel(ℓ)	
		acq(ℓ)
		$w(y)$
		rel(ℓ)
		$r(y)$
		r(x)

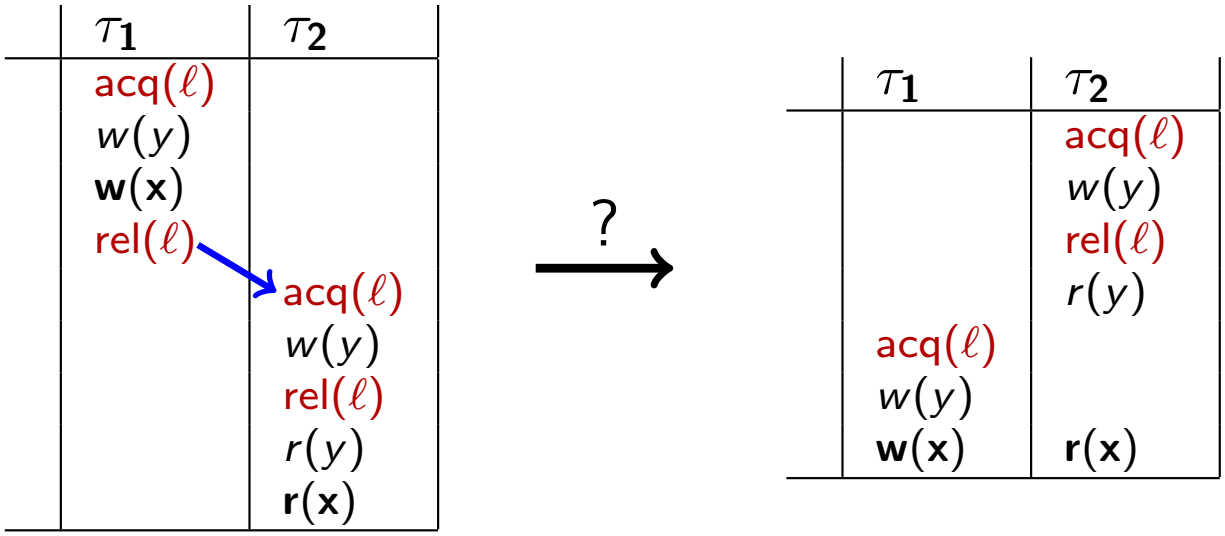


	τ_1	τ_2
		acq(ℓ)
		$w(y)$
		rel(ℓ)
		$r(y)$
	acq(ℓ)	
	$w(y)$	
	w(x)	**r(x)**

Partial-order Approaches

1. Build a partial order on the events of t
2. Report race iff $e_1 \not\to e_2$.

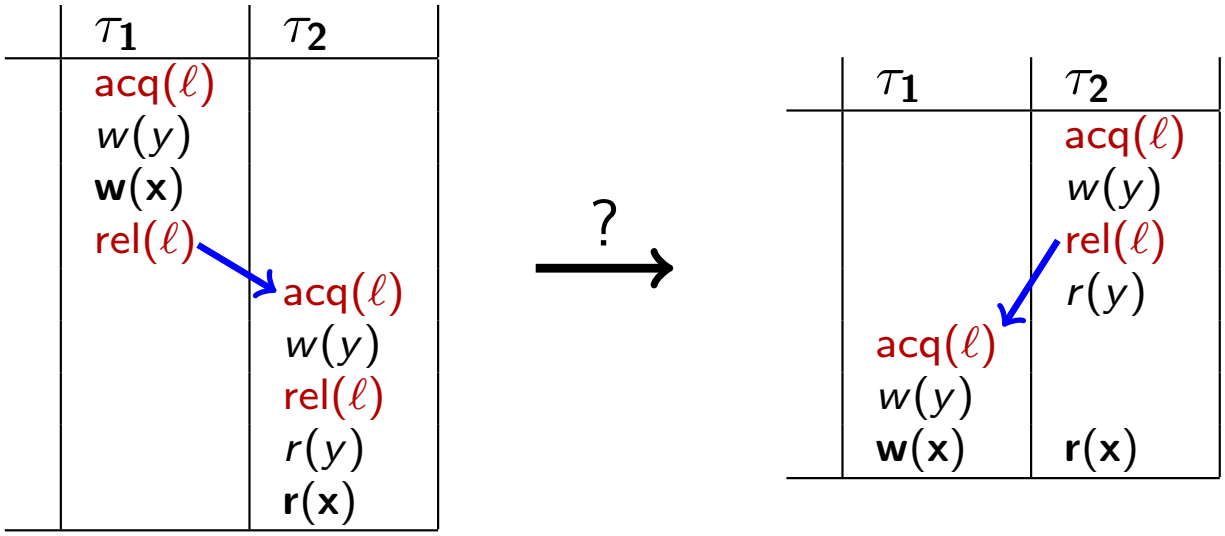
Miss simple races



Partial-order Approaches

1. Build a partial order on the events of t
2. Report race iff $e_1 \not\to e_2$.

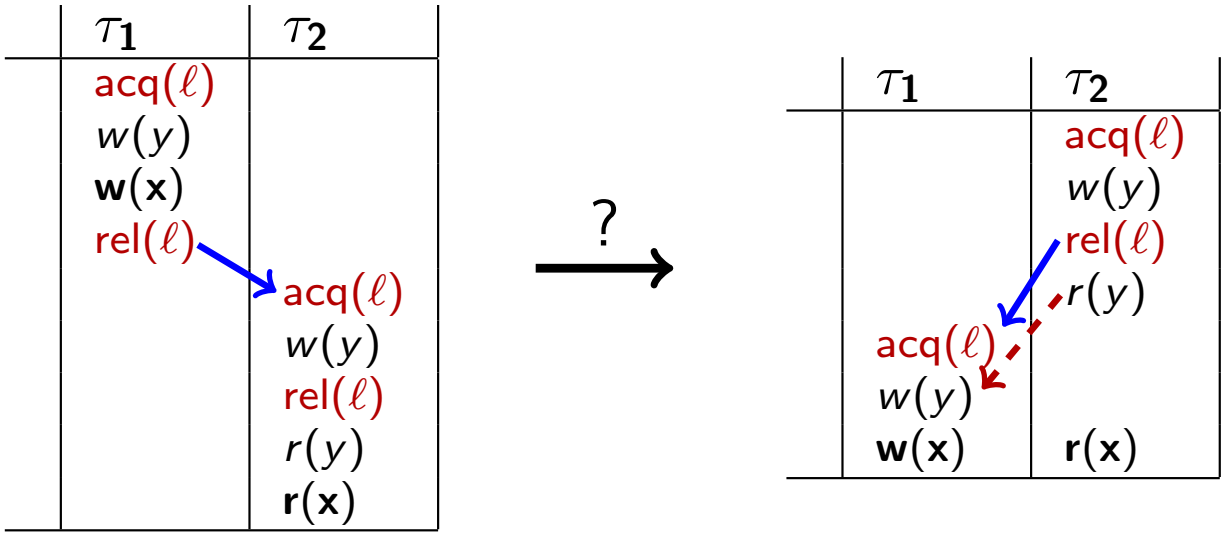
Miss simple races



Partial-order Approaches

1. Build a partial order on the events of t
2. Report race iff $e_1 \not\to e_2$.

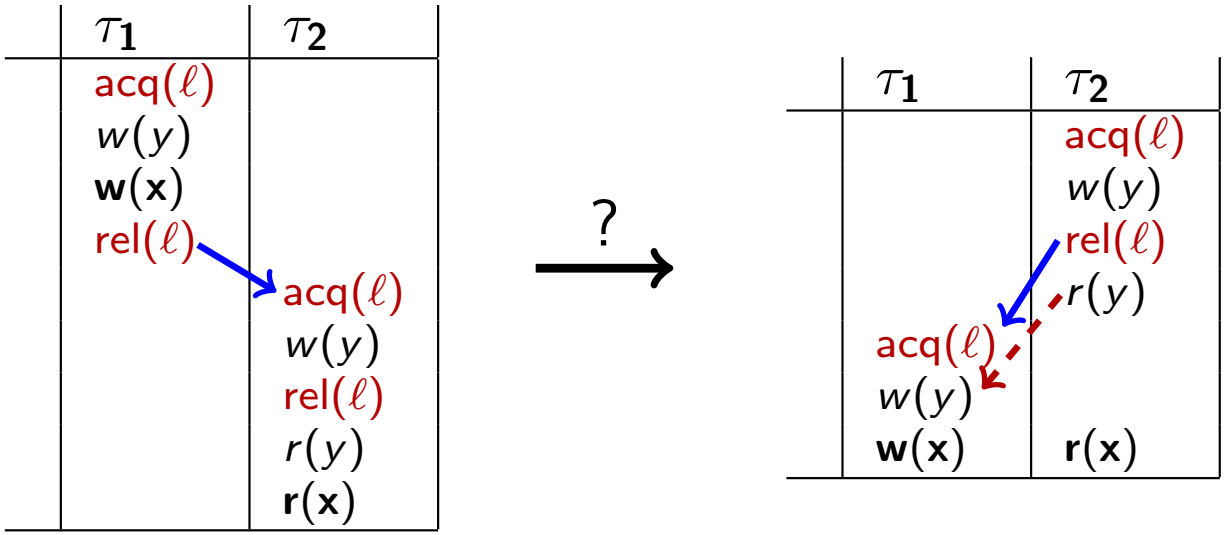
Miss simple races



Partial-order Approaches

1. Build a partial order on the events of t
2. Report race iff $e_1 \not\to e_2$.

Miss simple races



Definition

A partial order $\to$ is realizable if it has a linearization t^* such that $\mathcal{O}_{t^*} = \mathcal{O}_t$.

Partial-order Closure

Definition

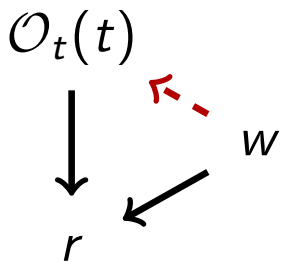
Partial order $\rightarrow$ is closed if

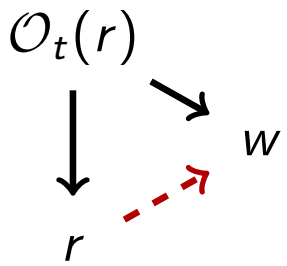
Partial-order Closure

Definition

Partial order $\rightarrow$ is closed if

1. For every read r with observation $\mathcal{O}_t(r)$ and conflicting w
   - if $w \rightarrow r$ then $w \rightarrow \mathcal{O}_t(r)$
   - if $\mathcal{O}_t(t) \rightarrow w'$ then $r \rightarrow w'$



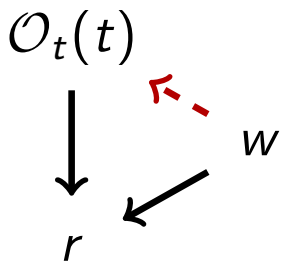


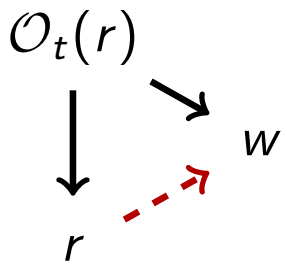
Partial-order Closure

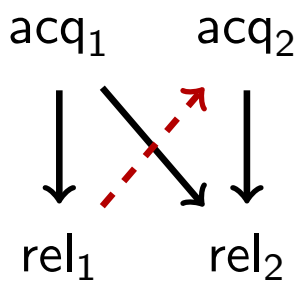
Definition

Partial order $\rightarrow$ is closed if

1. For every read r with observation $\mathcal{O}_t(r)$ and conflicting w
   - if $w \rightarrow r$ then $w \rightarrow \mathcal{O}_t(r)$
   - if $\mathcal{O}_t(t) \rightarrow w'$ then $r \rightarrow w'$
2. For every two critical sections $\text{acq}_1 \ldots \text{rel}_1$ and $\text{acq}_2 \ldots \text{rel}_2$ on the same lock
   - if $\text{acq}_1 \rightarrow \text{rel}_2$ then $\text{rel}_2 \rightarrow \text{acq}_1$





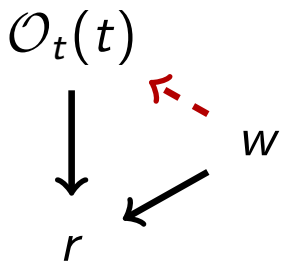


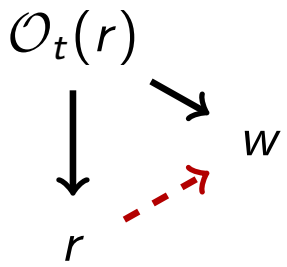
Partial-order Closure

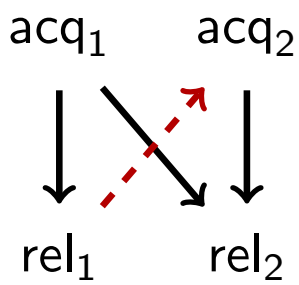
Definition

Partial order $\rightarrow$ is closed if

1. For every read r with observation $\mathcal{O}_t(r)$ and conflicting w
   - if $w \rightarrow r$ then $w \rightarrow \mathcal{O}_t(r)$
   - if $\mathcal{O}_t(t) \rightarrow w'$ then $r \rightarrow w'$
2. For every two critical sections $\text{acq}_1 \ldots \text{rel}_1$ and $\text{acq}_2 \ldots \text{rel}_2$ on the same lock
   - if $\text{acq}_1 \rightarrow \text{rel}_2$ then $\text{rel}_2 \rightarrow \text{acq}_1$







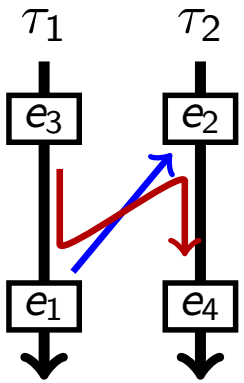
How fast can we compute the closure?

Incremental Maintenance of a Partial Order $\rightarrow$

$\text{Update}(e_1 \rightarrow e_2)$ $\quad$ $\text{Query}(e_3 \rightarrow e_4)$

Incremental Maintenance of a Partial Order $\rightarrow$

$\text{Update}(e_1 \rightarrow e_2)$ $\quad$ $\text{Query}(e_3 \rightarrow e_4)$

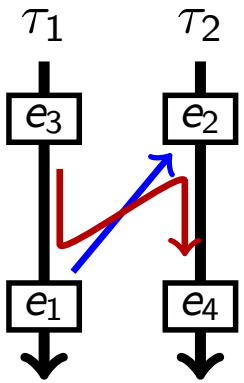


Update in $O(1)$

Query in $O(n)$

Incremental Maintenance of a Partial Order $\rightarrow$

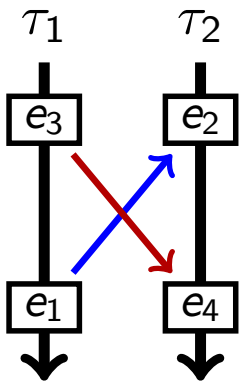
Update($e_1 \rightarrow e_2$)



Update in $O(1)$

Query in $O(n)$

Query($e_3 \rightarrow e_4$)

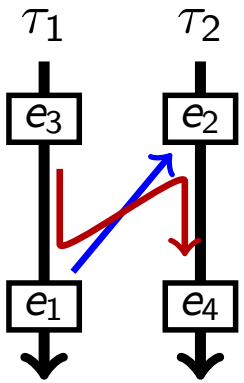


Update in $O(n)$

Query in $O(1)$

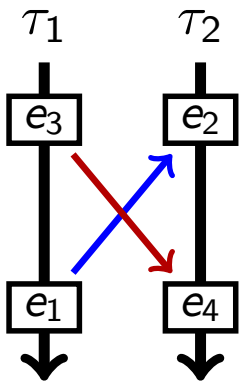
Incremental Maintenance of a Partial Order $\rightarrow$

Update($e_1 \rightarrow e_2$) Query($e_3 \rightarrow e_4$)



Update in $O(1)$

Query in $O(n)$



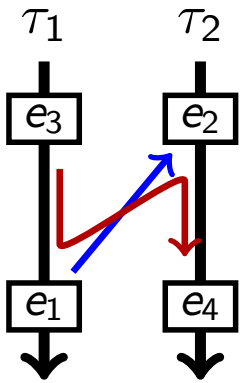
Update in $O(n)$

Query in $O(1)$

Each case costs $O(n)$ $\rightarrow$ too large, $n \sim 10^8$

Incremental Maintenance of a Partial Order $\rightarrow$

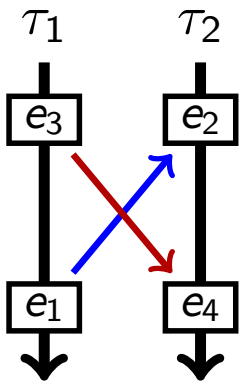
Update($e_1 \rightarrow e_2$)



Update in $O(1)$

Query in $O(n)$

Query($e_3 \rightarrow e_4$)



Update in $O(n)$

Query in $O(1)$

Theorem (Incremental Partial Orders)

A partial order $\rightarrow$ can be maintained incrementally with time bounds

1. $O(\log n)$ *per* Update,
2. $O(\log n)$ *per* Query.

Complexity of Closure

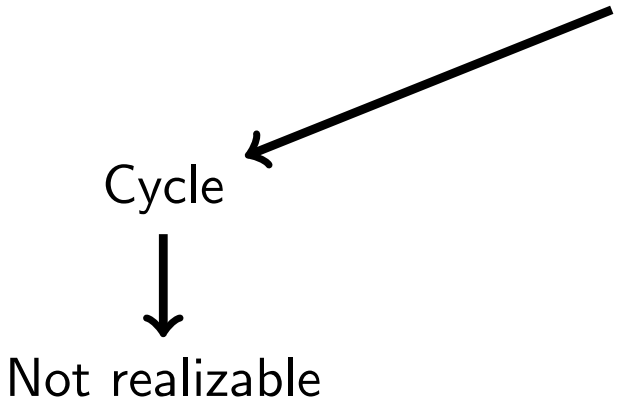
Theorem

The closure of a partial order can be computed in amortized $O(\log n)$ time (after $O(n)$ preprocessing).

Complexity of Closure

Theorem

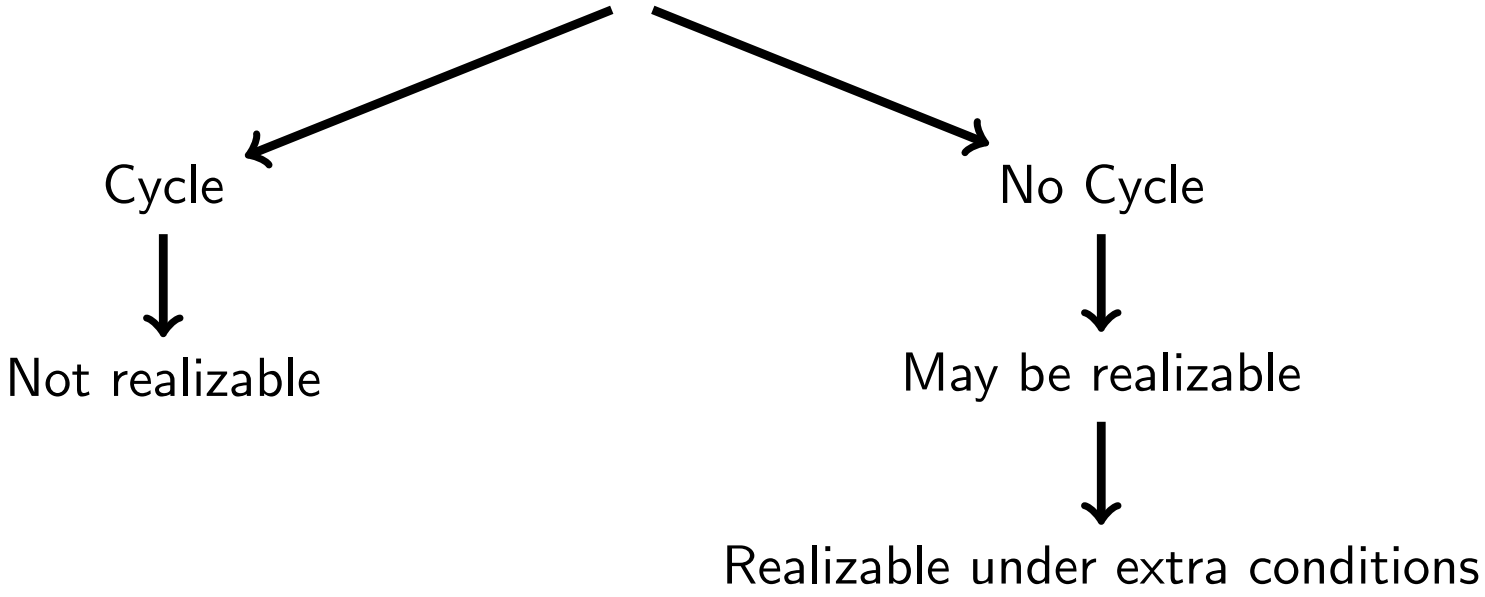
The closure of a partial order can be computed in amortized $O(\log n)$ time (after $O(n)$ preprocessing).



Complexity of Closure

Theorem

The closure of a partial order can be computed in amortized $O(\log n)$ time (after $O(n)$ preprocessing).



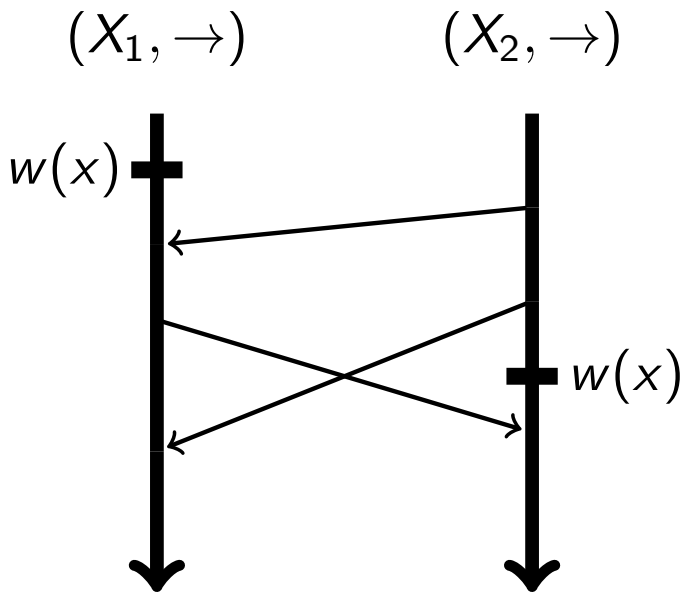
MaxMin Linearizations

Theorem (MaxMin Linearizations)

Take a closed poset $(X, \rightarrow)$ where X is partitioned into X_1, X_2, where

1. $(X_1, \rightarrow)$ *is a total order*
2. $(X_2, \rightarrow)$ *orders all conflicting pairs.*

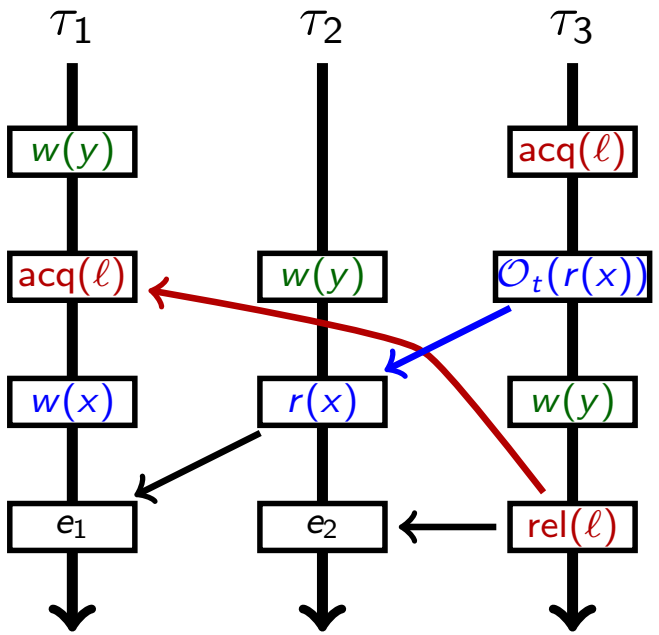
Then $(X, \rightarrow)$ is realizable.



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

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

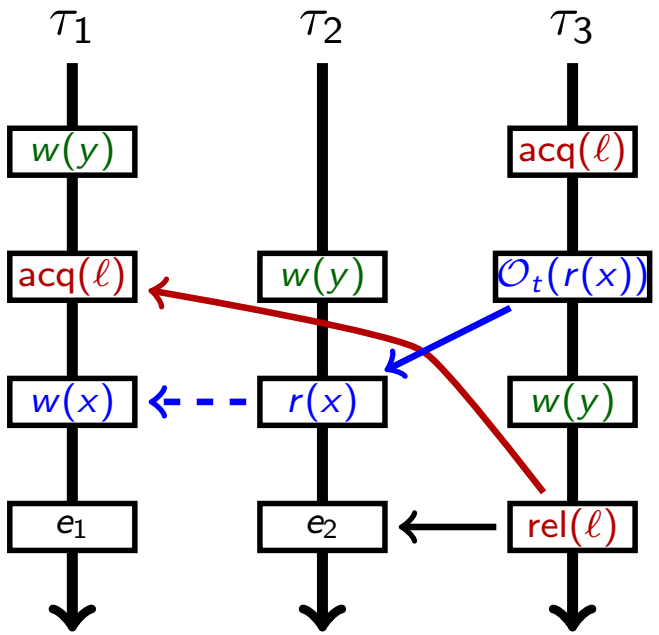
1 Start with a partial order $\rightarrow$
   (a) e_1, e_2 are maximal
   (b) $\mathcal{O}_t(r) \rightarrow r$ for every read r
   (c) $\text{rel}_1 \rightarrow \text{acq}_2$ if acq_2 does not close



M2

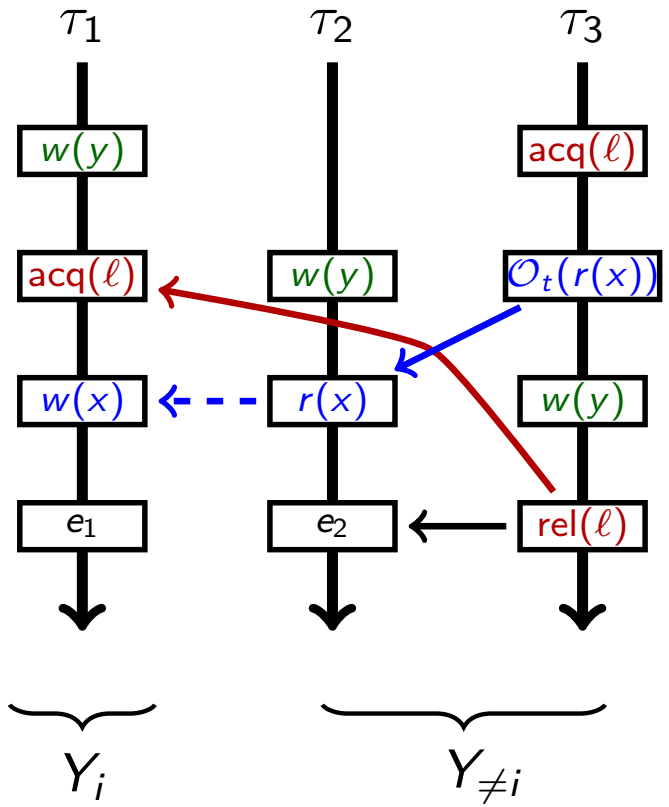
Input: A trace t, a conflicting pair (e_1, e_2)

1 Start with a partial order $\rightarrow$
   (a) e_1, e_2 are maximal
   (b) $\mathcal{O}_t(r) \rightarrow r$ for every read r
   (c) $\text{rel}_1 \rightarrow \text{acq}_2$ if acq_2 does not close
2 Close $\rightarrow$



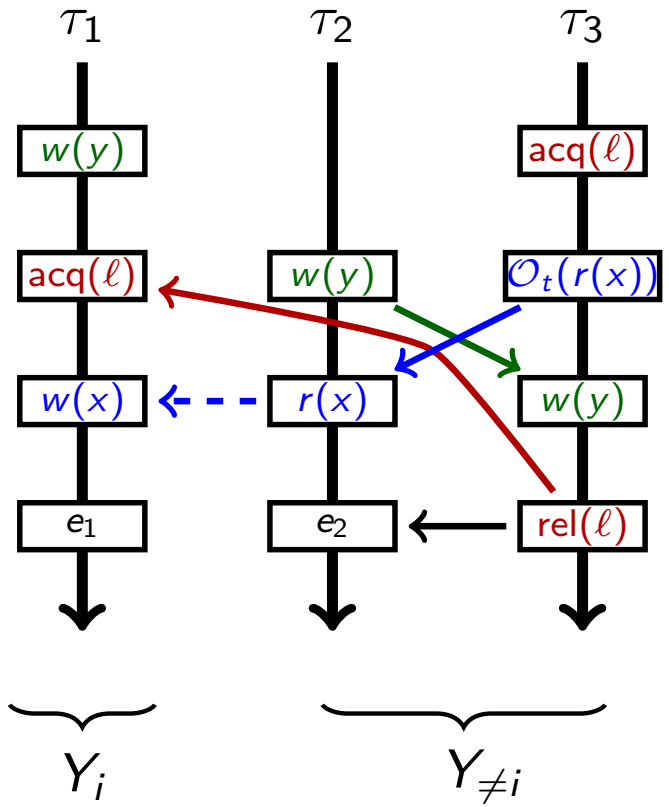
Input: A trace t, a conflicting pair (e_1, e_2)

1 Start with a partial order $\rightarrow$
   (a) e_1, e_2 are maximal
   (b) $\mathcal{O}_t(r) \rightarrow r$ for every read r
   (c) $\text{rel}_1 \rightarrow \text{acq}_2$ if acq_2 does not close
2 Close $\rightarrow$
3 Let Y_i be all events of thread i
   and $Y_{\neq i}$ all other events



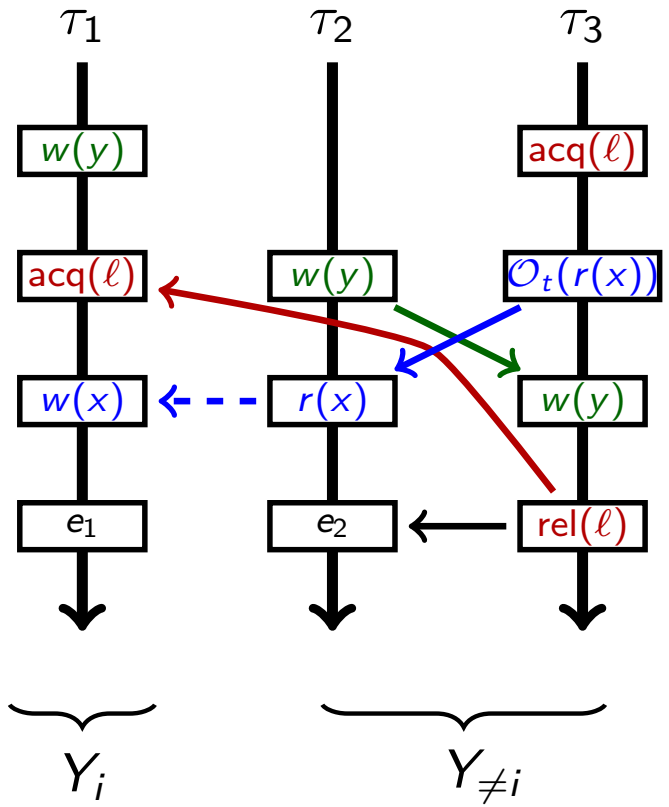
Input: A trace t, a conflicting pair (e_1, e_2)

1 Start with a partial order $\rightarrow$
   (a) e_1, e_2 are maximal
   (b) $\mathcal{O}_t(r) \rightarrow r$ for every read r
   (c) $\text{rel}_1 \rightarrow \text{acq}_2$ if acq_2 does not close
2 Close $\rightarrow$
3 Let Y_i be all events of thread i
   and $Y_{\neq i}$ all other events
4 **while** $\exists \overline{e}_1, \overline{e}_2 \in Y_{\neq i}$ *conflicting and unordered* **do**
5 | Order $\overline{e}_1, \overline{e}_2$ in t as in t, then close $\rightarrow$
6 **end**



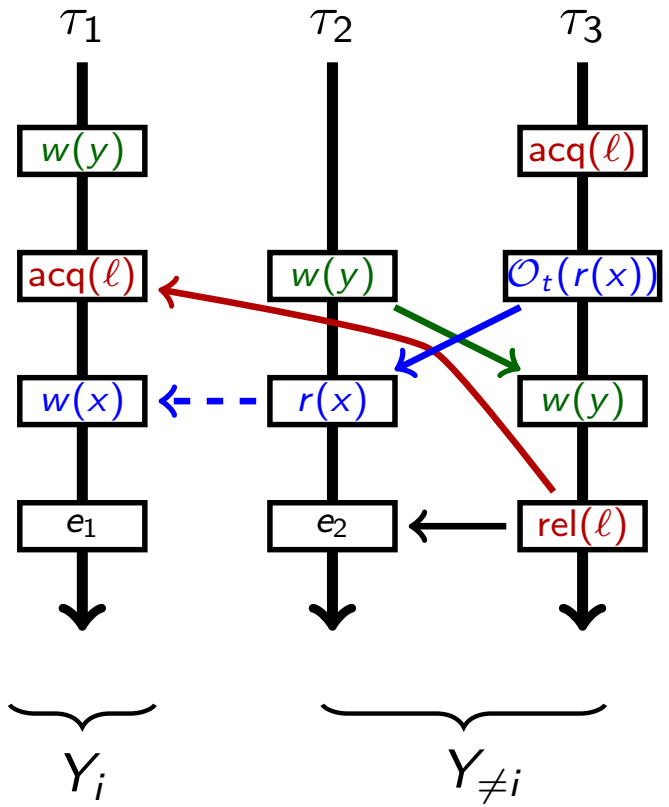
Input: A trace t, a conflicting pair (e_1, e_2)

1 Start with a partial order $\rightarrow$
   (a) e_1, e_2 are maximal
   (b) $\mathcal{O}_t(r) \rightarrow r$ for every read r
   (c) $\text{rel}_1 \rightarrow \text{acq}_2$ if acq_2 does not close
2 Close $\rightarrow$
3 Let Y_i be all events of thread i
   and $Y_{\neq i}$ all other events
4 **while** $\exists \overline{e}_1, \overline{e}_2 \in Y_{\neq i}$ *conflicting and unordered* **do**
5 | Order $\overline{e}_1, \overline{e}_2$ in t as in t, then close $\rightarrow$
6 **end**
7 Use MaxMin to linearize $\rightarrow$ to a witness t^*



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

1 Start with a partial order $\rightarrow$
   (a) e_1, e_2 are maximal
   (b) $\mathcal{O}_t(r) \rightarrow r$ for every read r
   (c) $\text{rel}_1 \rightarrow \text{acq}_2$ if acq_2 does not close
2 Close $\rightarrow$
3 Let Y_i be all events of thread i
   and $Y_{\neq i}$ all other events
4 **while** $\exists \overline{e}_1, \overline{e}_2 \in Y_{\neq i}$ *conflicting and unordered* **do**
5 | Order $\overline{e}_1, \overline{e}_2$ in t as in t, then close $\rightarrow$
6 **end**
7 Use MaxMin to linearize $\rightarrow$ to a witness t^*



Experiments

Comparison with

- HB C. Flanagan and S. N. Freund. "FastTrack: Efficient and Precise Dynamic Race Detection". In: PLDI. 2009
- WCP D. Kini, U. Mathur, and M. Viswanathan. "Dynamic Race Prediction in Linear Time". In: PLDI. 2017
- DC J. Roemer, K. Genç, and M. D. Bond. "High-coverage, Unbounded Sound Predictive Race Detection". In: PLDI. 2018
- SHB 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)

Benchmark	Size	HB	WCP	DC	SHB	M2	
		Races	Races	Races	Races	Races	FN
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**		

Benchmark	Size	HB	WCP	DC	SHB	M2	
		Races	Races	Races	Races	Races	FN
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**	

Benchmark	Size	HB	WCP	DC	SHB	M2	
		Races	Races	Races	Races	Races	FN
critical	49	3	3	3	8	8	0
airtickets	116	3	3	3	4	4	0
account	125	1	1	1	1	1	0
pingpong	126	2	2	2	2	2	0
bbuffer	322	2	2	2	2	2	0
mergesort	3K	1	1	1	1	**2**	0
bubblesort	4K	4	4	5	6	6	0
raytracer	16K	3	3	3	3	3	0
ftpserver	48K	23	23	24	23	**26**	0
moldyn	164K	2	2	2	2	2	0
derby	1M	12	12	12	12	12	0
jigsaw	3M	8	10	10	9	**11**	0
bufwriter	11M	2	2	2	2	2	0
hsqldb	18M	4	4	5	9	9	0
cryptorsa	57M	5	5	7	5	7	0
eclipse	86M	33	34	39	54	**67**	0
xalan	122M	7	7	9	11	**15**	0
lusearch	216M	30	30	30	52	52	0
Total	**514M**	**145**	**148**	**160**	**206**	**231**	**0**

Conclusion

M2: a new algorithm for dynamic data-race prediction

1. Polynomial-time
2. Sound
3. Complete for 2 threads
4. Criteria for detecting completeness dynamically
5. Practically complete on a challenging data set

Conclusion

M2: a new algorithm for dynamic data-race prediction

1. Polynomial-time
2. Sound
3. Complete for 2 threads
4. Criteria for detecting completeness dynamically
5. Practically complete on a challenging data set

Thank you!

Questions?

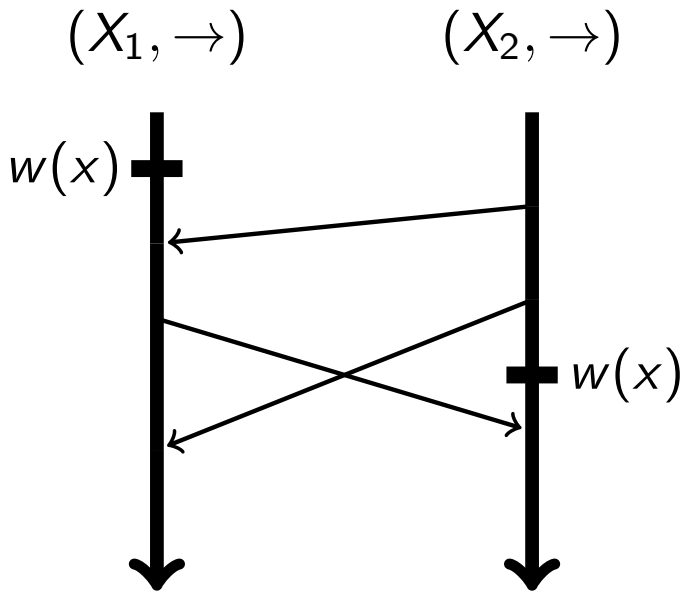
MaxMin Linearizations

Theorem (MaxMin Linearizations)

Take a closed poset $(X, \rightarrow)$ where X is partitioned into X_1, X_2, where

1. $(X_1, \rightarrow)$ *is a total order*
2. $(X_2, \rightarrow)$ *orders all conflicting pairs.*

Then $(X, \rightarrow)$ is realizable.



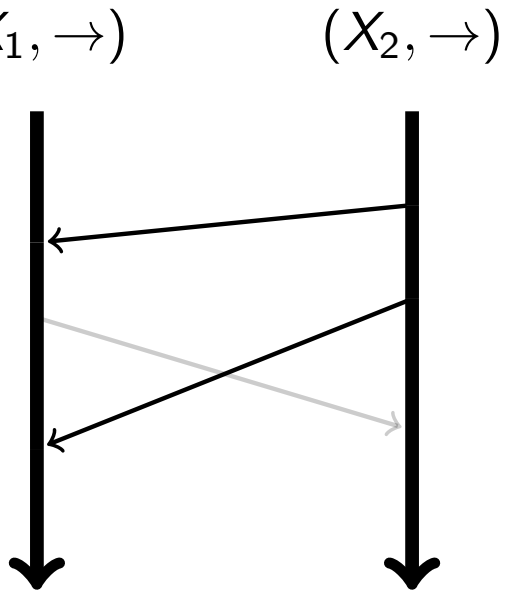
MaxMin Linearizations

Theorem (MaxMin Linearizations)

Take a closed poset $(X, \to)$ where X is partitioned into X_1, X_2, where

1. $(X_1, \to)$ *is a total order*
2. $(X_2, \to)$ *orders all conflicting pairs.*

Then $(X, \to)$ is realizable.



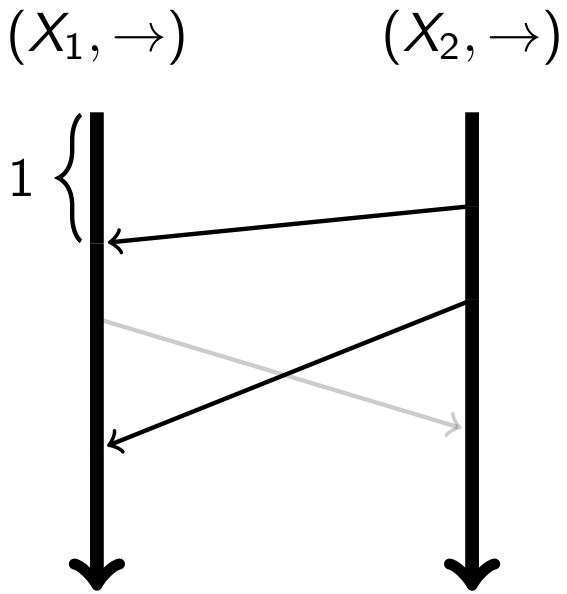
MaxMin Linearizations

Theorem (MaxMin Linearizations)

Take a closed poset $(X, \to)$ where X is partitioned into X_1, X_2, where

1. $(X_1, \to)$ *is a total order*
2. $(X_2, \to)$ *orders all conflicting pairs.*

Then $(X, \to)$ is realizable.



$$t^* = \langle 1 \rangle$$

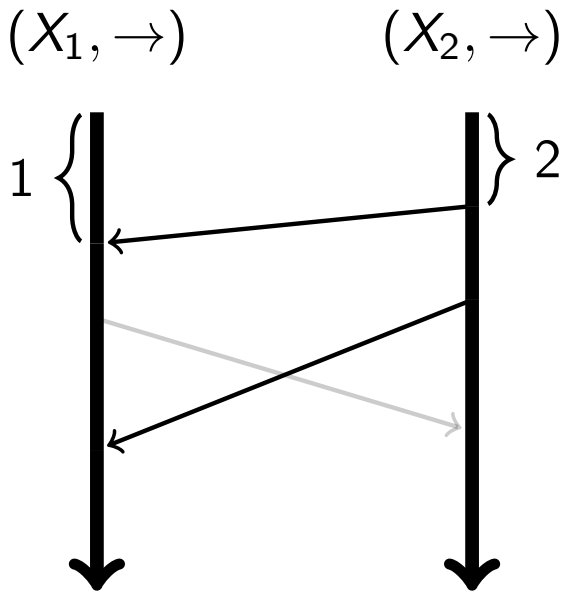
MaxMin Linearizations

Theorem (MaxMin Linearizations)

Take a closed poset $(X, \to)$ where X is partitioned into X_1, X_2, where

1. $(X_1, \to)$ *is a total order*
2. $(X_2, \to)$ *orders all conflicting pairs.*

Then $(X, \to)$ is realizable.



$$t^* = \langle 1 \rangle \circ \langle 2 \rangle$$

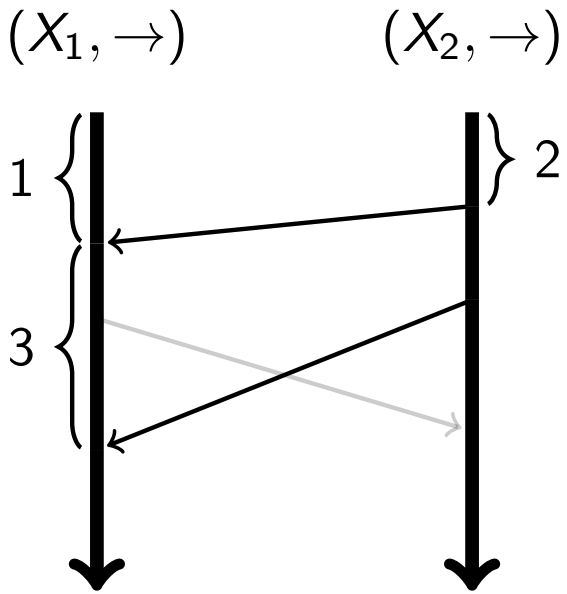
MaxMin Linearizations

Theorem (MaxMin Linearizations)

Take a closed poset $(X, \to)$ where X is partitioned into X_1, X_2, where

1. $(X_1, \to)$ *is a total order*
2. $(X_2, \to)$ *orders all conflicting pairs.*

Then $(X, \to)$ is realizable.



$$t^* = \langle 1 \rangle \circ \langle 2 \rangle \circ \langle 3 \rangle$$

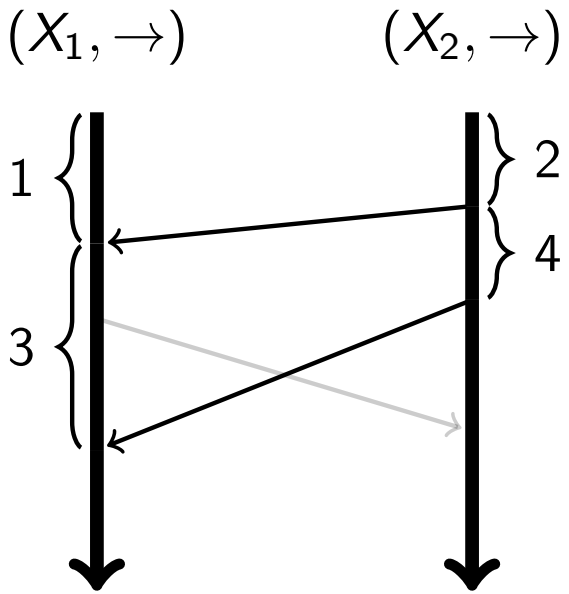
MaxMin Linearizations

Theorem (MaxMin Linearizations)

Take a closed poset $(X, \to)$ where X is partitioned into X_1, X_2, where

1. $(X_1, \to)$ *is a total order*
2. $(X_2, \to)$ *orders all conflicting pairs.*

Then $(X, \to)$ is realizable.



$$t^* = \langle 1 \rangle \circ \langle 2 \rangle \circ \langle 3 \rangle \circ \langle 4 \rangle$$

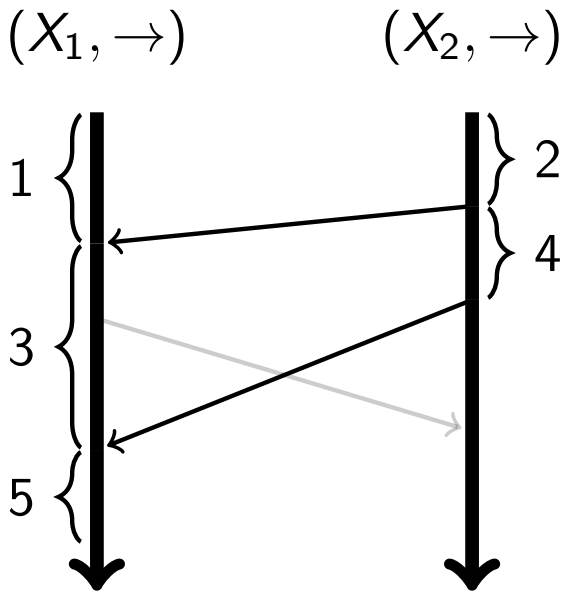
MaxMin Linearizations

Theorem (MaxMin Linearizations)

Take a closed poset $(X, \to)$ where X is partitioned into X_1, X_2, where

1. $(X_1, \to)$ *is a total order*
2. $(X_2, \to)$ *orders all conflicting pairs.*

Then $(X, \to)$ is realizable.



$$t^* = \langle 1 \rangle \circ \langle 2 \rangle \circ \langle 3 \rangle \circ \langle 4 \rangle \circ \langle 5 \rangle$$

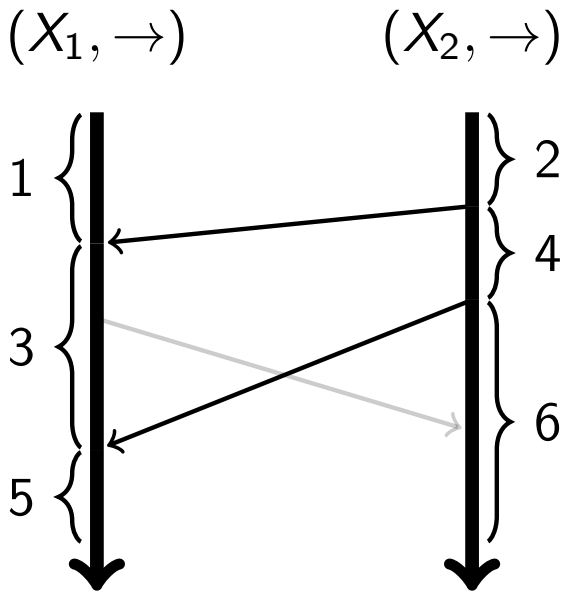
MaxMin Linearizations

Theorem (MaxMin Linearizations)

Take a closed poset $(X, \to)$ where X is partitioned into X_1, X_2, where

1. $(X_1, \to)$ *is a total order*
2. $(X_2, \to)$ *orders all conflicting pairs.*

Then $(X, \to)$ is realizable.



$$t^* = \langle 1 \rangle \circ \langle 2 \rangle \circ \langle 3 \rangle \circ \langle 4 \rangle \circ \langle 5 \rangle \circ \langle 6 \rangle$$

Race Distances

Benchmark	Mean Distance	Max Distance
ftpserver	939	12K
jigsaw	1K	4K
hsqldb	92K	1M
cryptorsa	1M	8M
eclipse	11M	53M
xalan	2.0K	43K
lusearch	44M	125M

Times

Benchmark	HB	WCP	DC	SHB	M2
array	0.30s	0.28s	2.12s	0.29s	0.12s
critical	0.29s	0.30s	2.11s	0.28s	0.10s
airtickets	0.32s	0.31s	2.10s	0.31s	0.12s
account	0.31s	0.32s	2.12s	0.30s	0.12s
pingpong	0.30s	0.31s	2.04s	0.31s	0.09s
bbuffer	0.30s	0.31s	2.12s	0.31s	0.09s
mergesort	0.36s	0.41s	2.16s	0.37s	0.18s
bubblesort	0.46s	0.54s	2.28s	0.62s	0.71s
raytracer	0.51s	0.56s	2.57s	0.51s	0.23s
ftpserver	0.79s	1.28s	2.75s	0.73s	0.88s
moldyn	1.50s	1.81s	3.88s	1.52s	1.08s
derby	8.53s	14.54s	15.29s	8.32s	7.84s
jigsaw	17.51s	21.80s	40.89s	17.93s	14.65s
bufwriter	48.64s	2m0s	2m59s	47.71s	57.37s
hsqldb	3m53s	3m5s	4m23s	3m53s	7m1s
cryptorsa	3m42s	3m0s	6m58s	3m29s	6m6s
eclipse	8m1s	7m0s	14m44s	7m11s	45m23s
xalan	8m58s	8m25s	20m12s	9m8s	7m15s
lusearch	16m4s	9m59s	2h49m6s	15m28s	8m9s
Total	**42m0s**	**34m14s**	**3h39m**	**40m31s**	**1h15m**

Missed Racy Memory Locations

Benchmark	HB	WCP	DC	SHB	M2
ftpserver	3	3	3	3	0
jigsaw	2	0	0	2	0
cryptorsa	1	1	0	1	0
eclipse	16	11	8	10	0
xalan	3	2	2	2	0
lusearch	11	11	11	0	0
Total	**36**	**28**	**24**	**18**	**0**

Benchmark Statistics

Benchmark	size	# threads	# variables	# locks
array	44	2	30	2
critical	49	3	30	0
airtickets	116	2	46	0
account	125	3	41	3
pingpong	126	4	54	0
bbuffer	322	2	73	2
mergesort	3.0K	4	621	3
bubblesort	4.0K	10	196	3
raytracer	16K	2	3.0K	8
ftpserver	48K	10	5.0K	304
moldyn	164K	2	1.0K	2
derby	1.0M	3	185K	1.0K
jigsaw	3.0M	13	103K	280
bufwriter	11M	5	56	1
hsqldb	18M	43	946K	412
cryptorsa	57M	7	1.0M	8.0K
eclipse	86M	14	10M	8.0K
xalan	122M	6	4.0M	2.0K
lusearch	216M	7	5.0M	118