The Complexity of Dynamic Data Race Prediction

Umang Mathur

University of Illinois at Urbana-Champaign

Andreas Pavlogiannis

Aarhus University

Mahesh Viswanathan

University of Illinois at Urbana-Champaign

Concurrency and Challenges

Concurrency and Challenges

- Concurrency - ubiquitous programming paradigm

Concurrency and Challenges

- Concurrency - ubiquitous programming paradigm

Concurrency and Challenges

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

Concurrency and Challenges

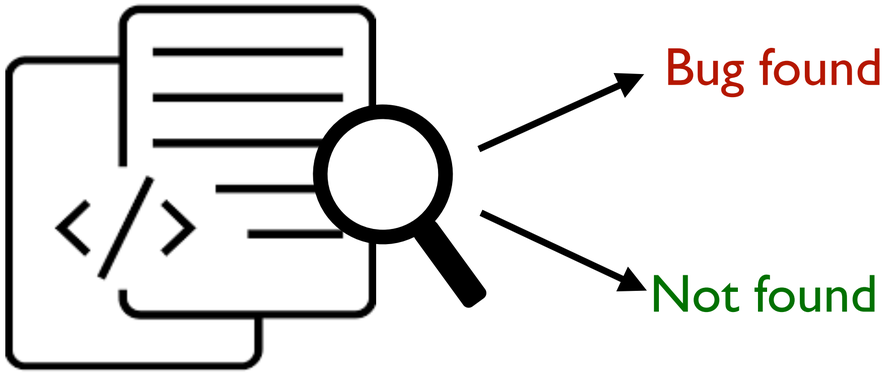
- Concurrency - ubiquitous programming paradigm
- Challenging to develop concurrent software
  - Large interleaving space
- Concurrency bugs arise in production-level software
  - Despite rigorous testing

Concurrency and Challenges

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

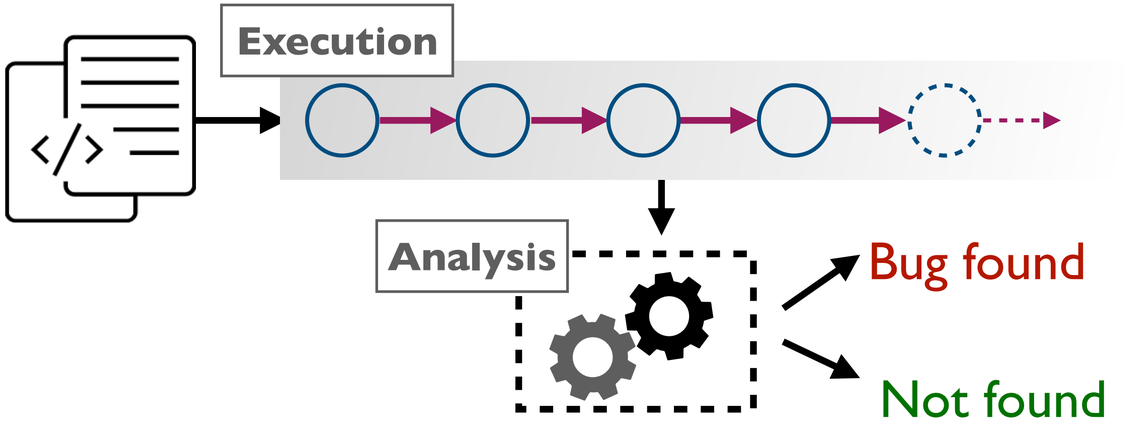
Data Race Detection

Static analyses



- Analyze source code
- Undecidable problem
- Excessive false alarms

Dynamic analyses



- Analyze executions at runtime
- Typically sound
- Widely adopted - TSan, Helgrind, etc.,

Concurrent Program Executions

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

	t1	t2
1	acq(l)	
2	w(x)	
3	w(y)	
4		r(x)
5	rel(l)	
6		w(y)

Data Races and Dynamic Race Prediction

Data Races and Dynamic Race Prediction

Data Race

An execution σ has a data race if

- pair of conflicting events
- consecutive

Data Races and Dynamic Race Prediction

Data Race

An execution σ has a data race if

- pair of conflicting events
- consecutive

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

Data Races and Dynamic Race Prediction

Data Race

An execution σ has a data race if

- pair of conflicting events
- consecutive

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

t1	t2
acq(l)	
w(x)	
	r(x)
rel(l)	

Data Races and Dynamic Race Prediction

Data Race

An execution σ has a data race if

- pair of conflicting events
- consecutive

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

t1	t2
acq(l)	
w(x)	
	r(x)
rel(l)	

Data Race
Detection

Data Races and Dynamic Race Prediction

Data Race

An execution σ has a data race if

- pair of conflicting events
- consecutive

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

t1	t2
acq(l)	
w(x)	
	r(x)
rel(l)	

Data Race
Detection

Given an execution σ, does σ have a data race?

- Sensitive to thread scheduling
- Miss a lot of races

Data Races and Dynamic Race Prediction

Data Race

An execution σ has a data race if

- pair of conflicting events
- consecutive

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

t1	t2
acq(l)	
w(x)	
	r(x)
rel(l)	

Data Race
Detection

Given an execution σ, does σ have a data race?

- Sensitive to thread scheduling
- Miss a lot of races

t1	t2
acq(l)	
w(x)	
rel(l)	
	r(x)

Data Races and Dynamic Race Prediction

Data Race

An execution σ has a data race if

- pair of conflicting events
- consecutive

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

t1	t2
acq(l)	
w(x)	
	r(x)
rel(l)	

Data Race
Detection

Given an execution σ, does σ have a data race?

- Sensitive to thread scheduling
- Miss a lot of races

t1	t2
acq(l)	
w(x)	
rel(l)	
	r(x)

Data Races and Dynamic Race Prediction

Data Race

An execution σ has a data race if

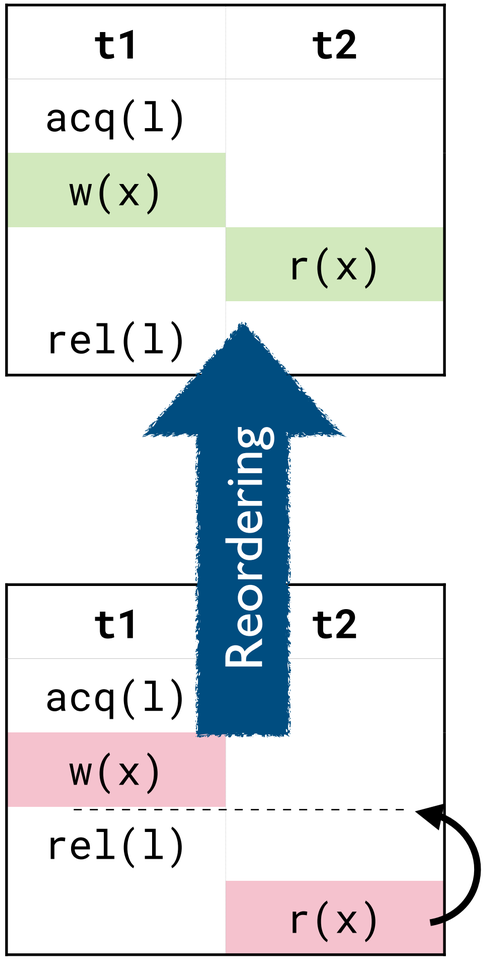
- pair of conflicting events
- consecutive

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

Data Race
Detection

Given an execution σ, does σ have a data race?

- Sensitive to thread scheduling
- Miss a lot of races



Data Races and Dynamic Race Prediction

Data Race

An execution σ has a data race if

- pair of conflicting events
- consecutive

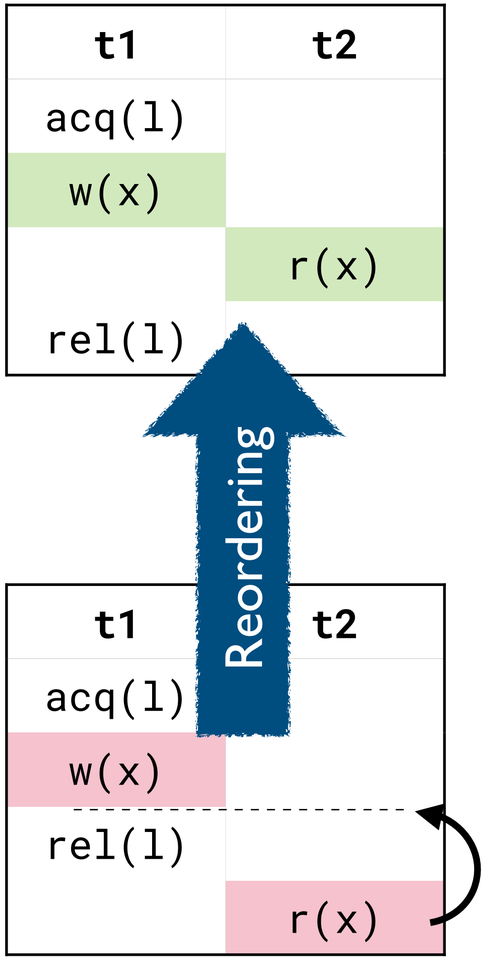
1. Same memory location
2. Different threads
3. At least one write

Data Race Detection

Given an execution σ, does σ have a data race?

- Sensitive to thread scheduling
- Miss a lot of races

Given an execution σ, is there a **reordering*** ρ of σ, such that ρ has a data race?



Data Races and Dynamic Race Prediction

Data Race

An execution σ has a data race if

- pair of conflicting events
- consecutive

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

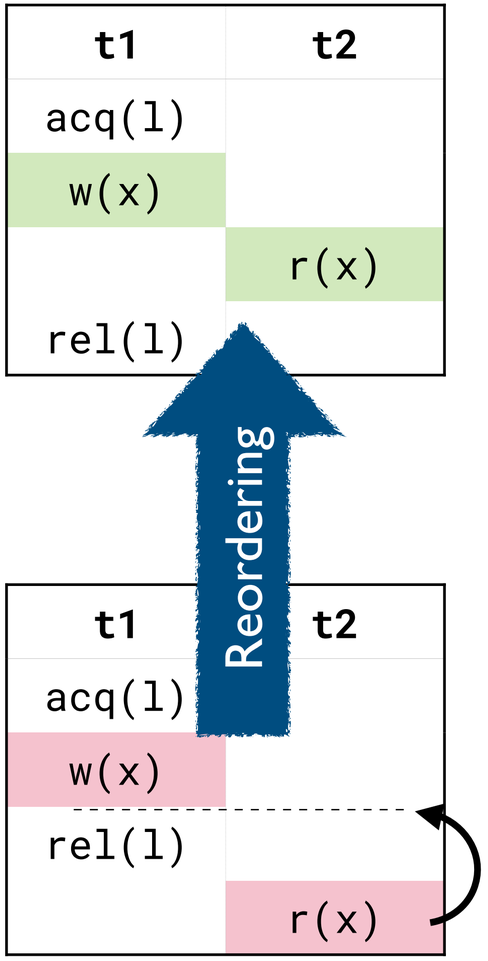
Data Race Detection

Given an execution σ, does σ have a data race?

- Sensitive to thread scheduling
- Miss a lot of races

Data Race Prediction

Given an execution σ, is there a **reordering*** ρ of σ, such that ρ has a data race?



Which reorderings are allowed?

Which reorderings are allowed?

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

Which reorderings are allowed?

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

Reorderings must satisfy some properties -

Which reorderings are allowed?

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

Reorderings must satisfy some properties -

1. Well formed -

Which reorderings are allowed?

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

Reorderings must satisfy some properties -

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

Which reorderings are allowed?

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

Reorderings must satisfy some properties -

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

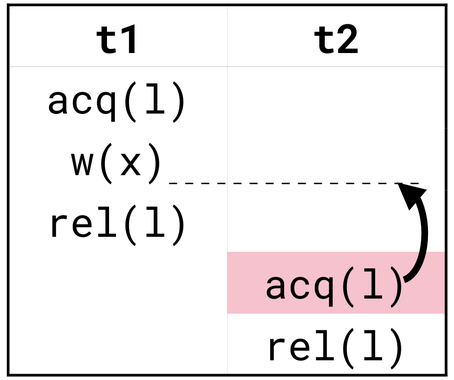
t1	t2
acq(l)	
w(x)	
rel(l)	
	acq(l)
	rel(l)

Which reorderings are allowed?

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

Reorderings must satisfy some properties -

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

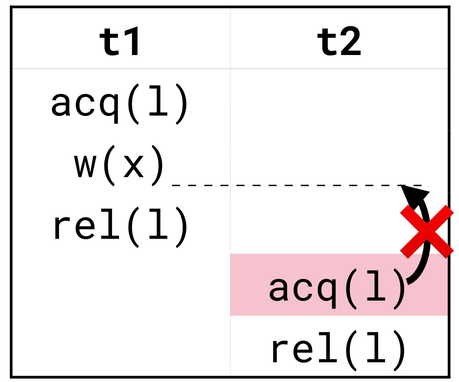


Which reorderings are allowed?

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

Reorderings must satisfy some properties -

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

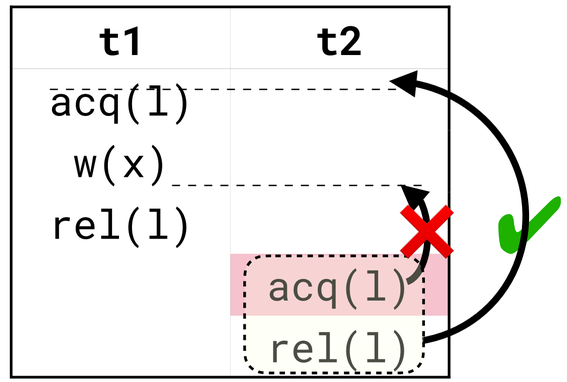


Which reorderings are allowed?

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

Reorderings must satisfy some properties -

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

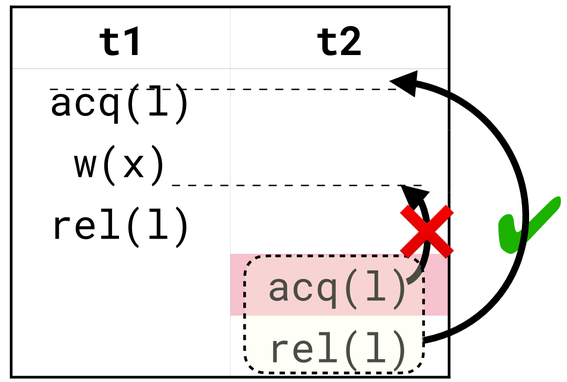


Which reorderings are allowed?

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

Reorderings must satisfy some properties -

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

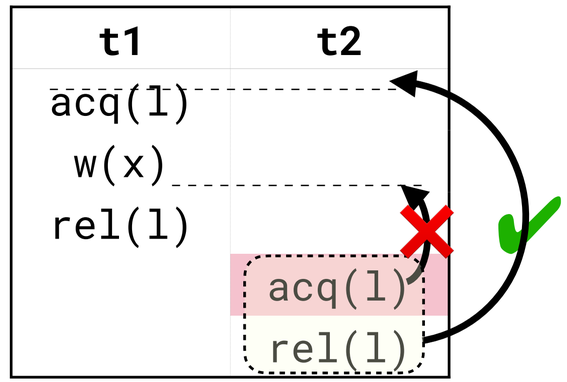


Which reorderings are allowed?

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

Reorderings must satisfy some properties -

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



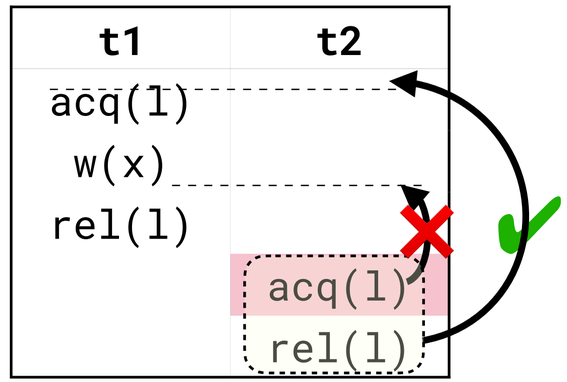
t1	t2
acq(l)	
w(x)	
rel(l)	
	r(x)

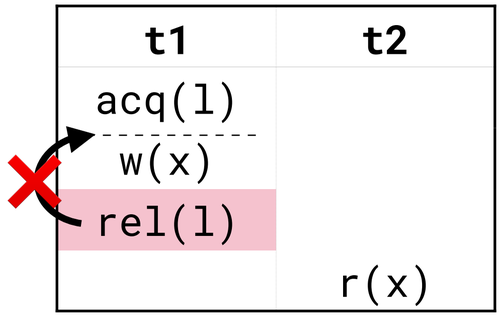
Which reorderings are allowed?

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

Reorderings must satisfy some properties -

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



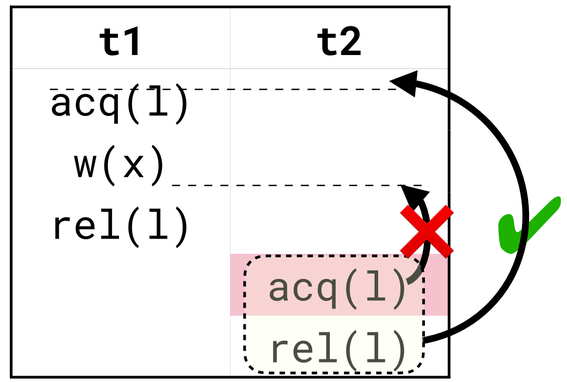


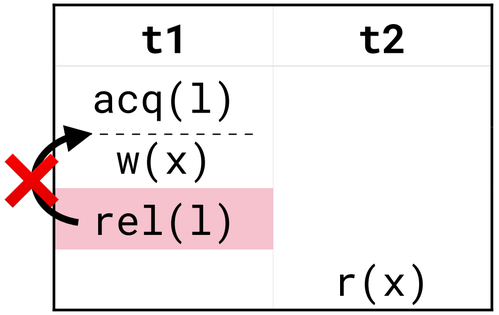
Which reorderings are allowed?

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

Reorderings must satisfy some properties -

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



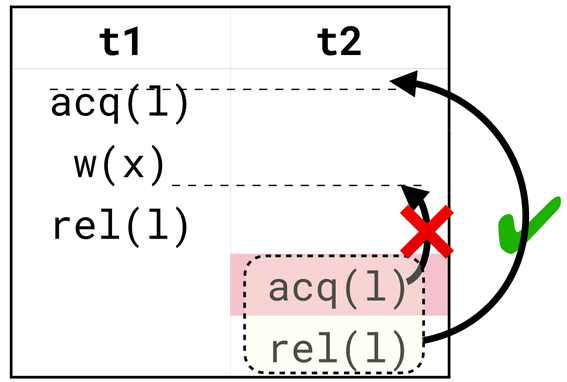


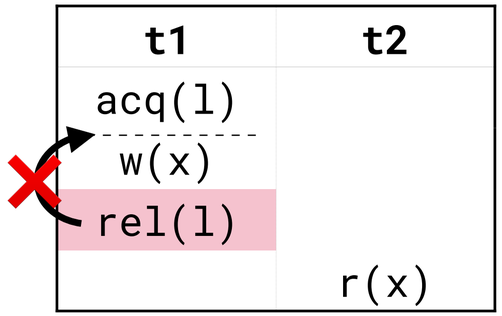
Which reorderings are allowed?

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

Reorderings must satisfy some properties -

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



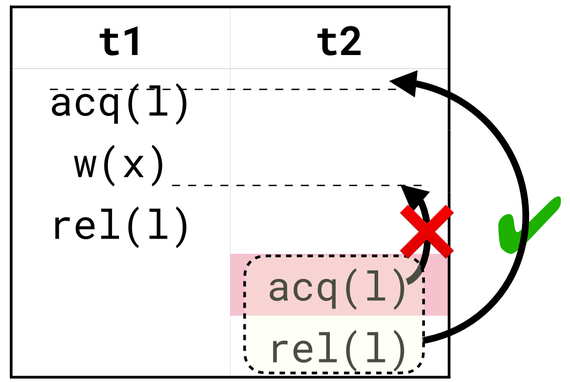


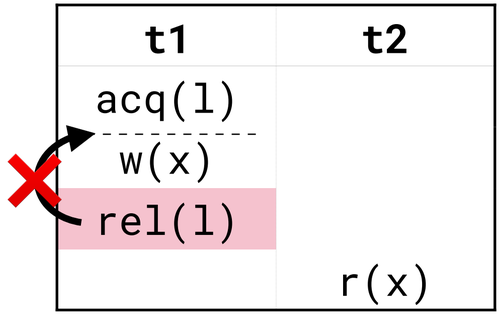
Which reorderings are allowed?

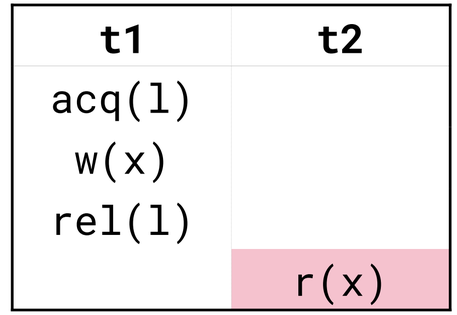
Any program that generates the observed execution must also generate the reordering

Reorderings must satisfy some properties -

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





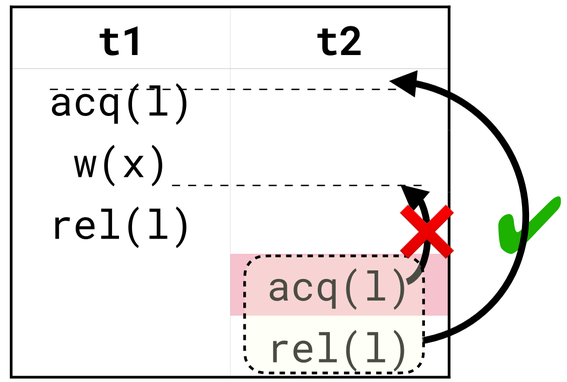


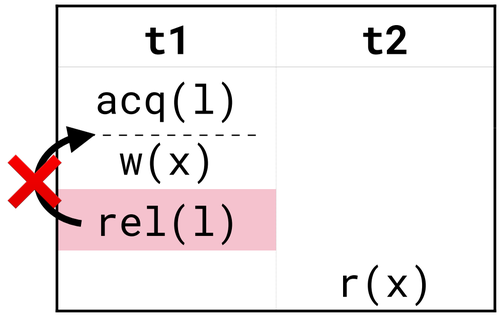
Which reorderings are allowed?

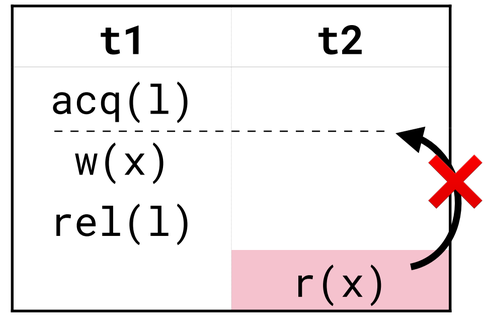
Any program that generates the observed execution must also generate the reordering

Reorderings must satisfy some properties -

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







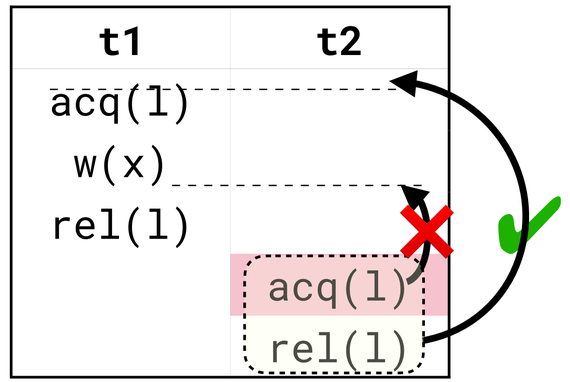
Which reorderings are allowed?

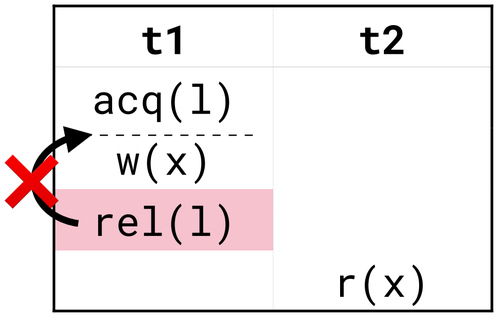
Any program that generates the observed execution must also generate the reordering

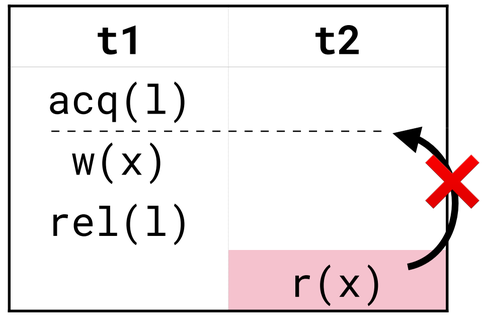
Correct reordering*

Reorderings must satisfy some properties -

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







* Herlihy and Wing, Linearizability: A Correctness Condition for Concurrent Objects, TOPLAS 1990

Smaragdakis et al, Sound predictive race detection in polynomial time, POPL 2012

Some History

Data Race Prediction

Given an execution σ, is there a **correct reordering** of σ with a data race?

Some History

Data Race Prediction

Given an execution σ, is there a **correct reordering** of σ with a data race?

Some History

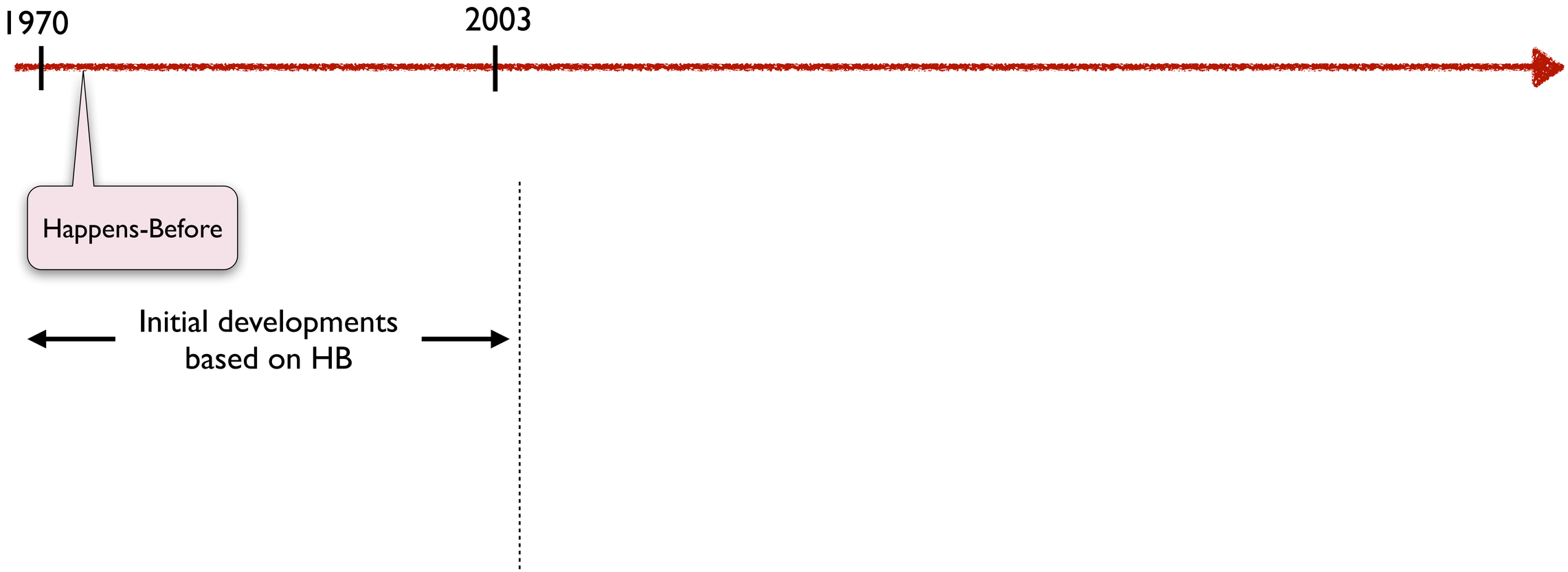
Data Race Prediction

Given an execution σ, is there a **correct reordering** of σ with a data race?

Some History

Data Race Prediction

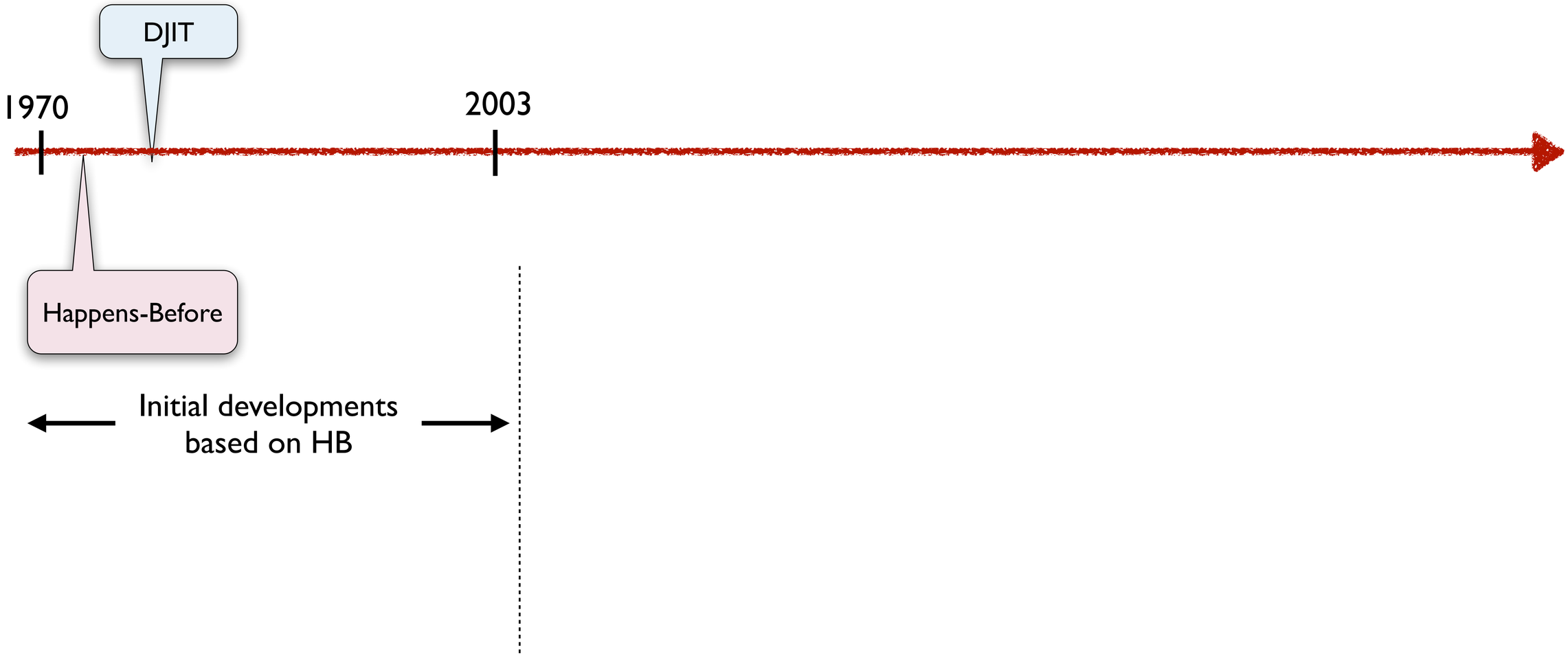
Given an execution σ, is there a **correct reordering** of σ with a data race?



Some History

Data Race Prediction

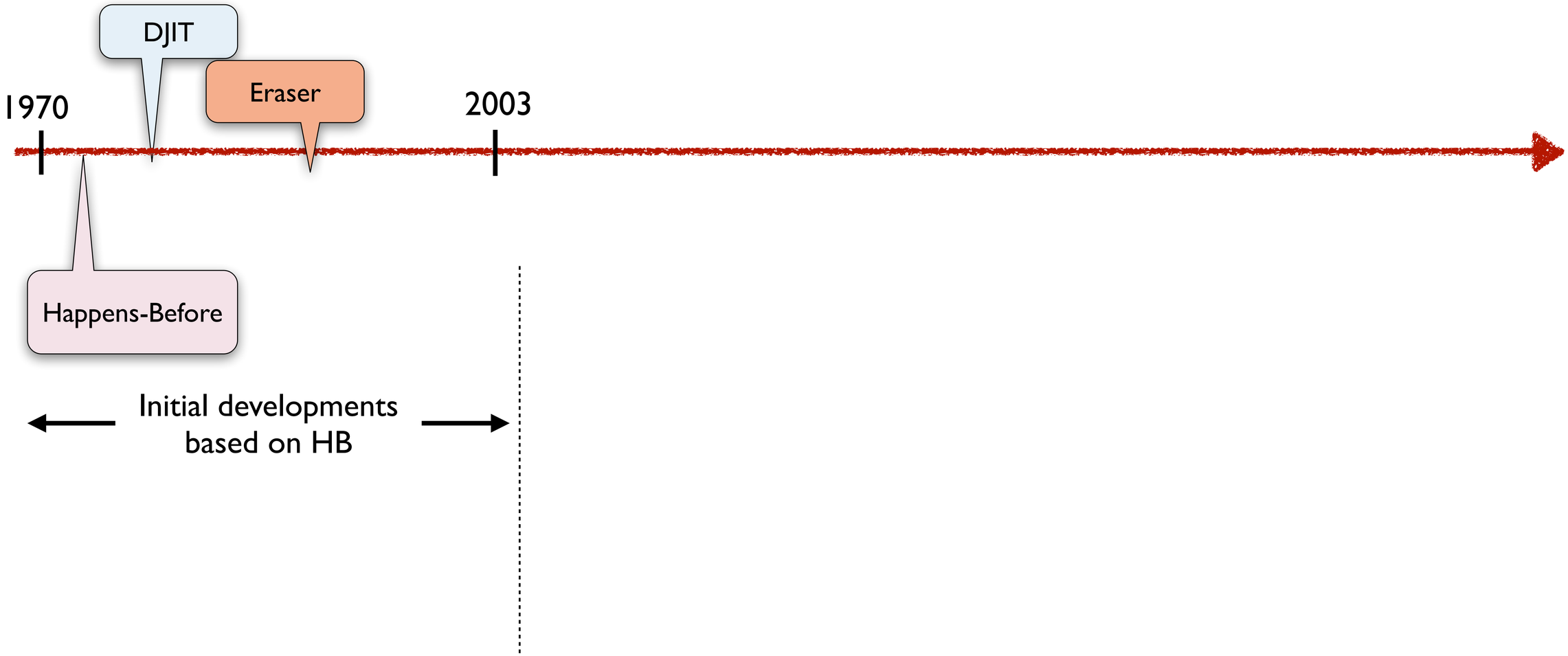
Given an execution σ, is there a **correct reordering** of σ with a data race?



Some History

Data Race Prediction

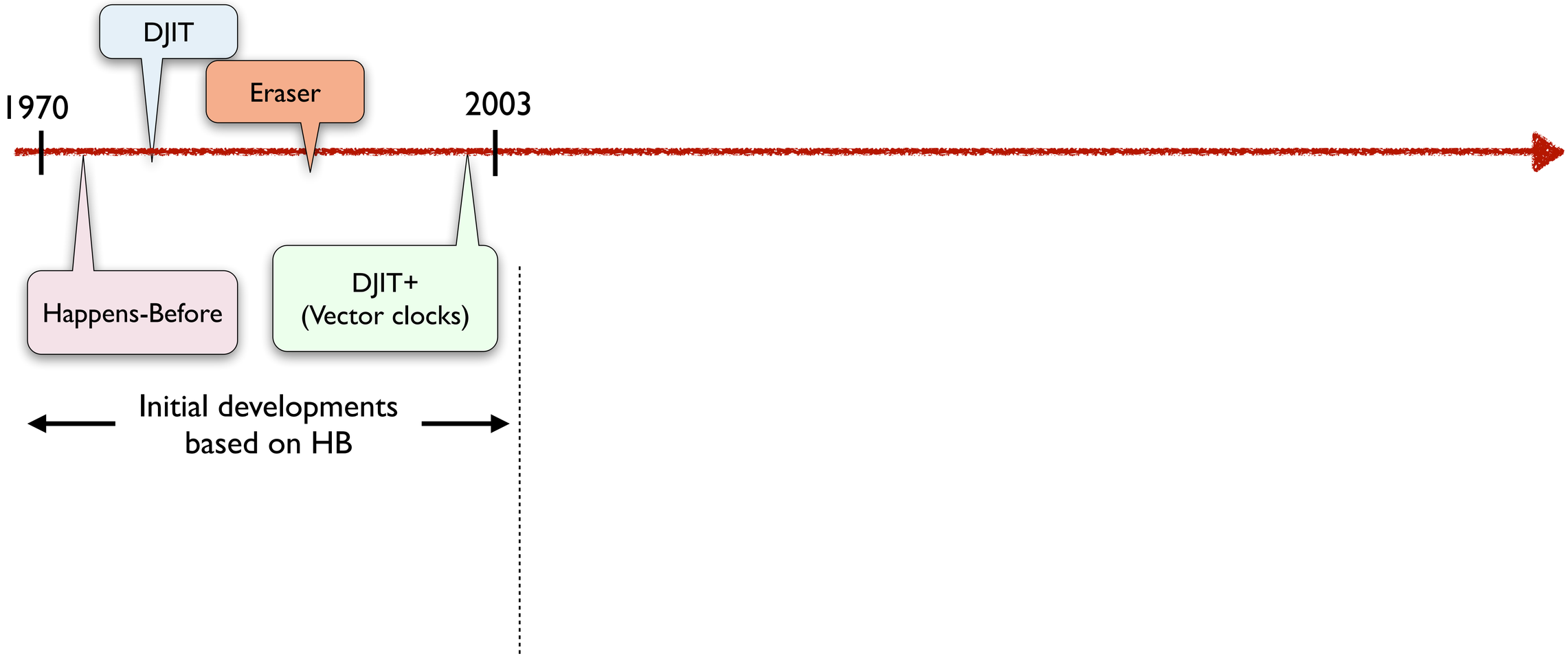
Given an execution σ, is there a **correct reordering** of σ with a data race?



Some History

Data Race Prediction

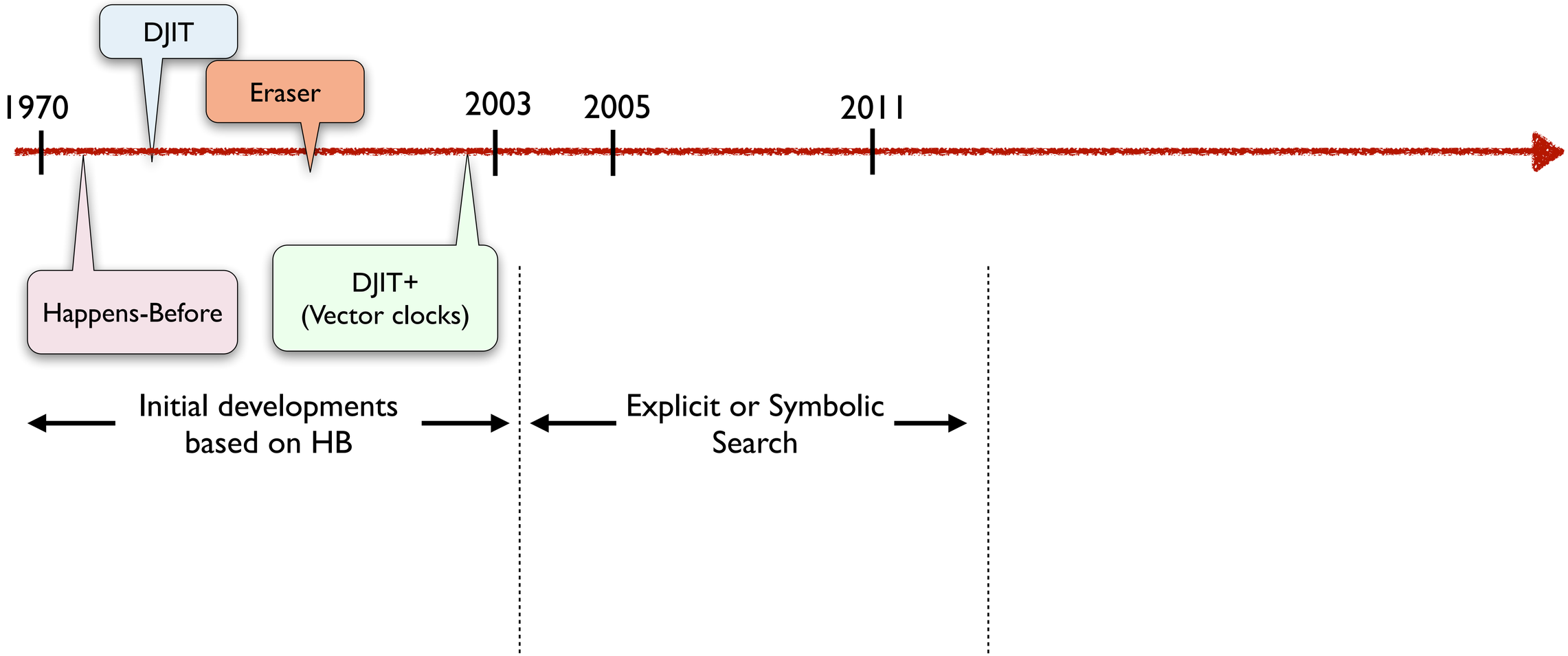
Given an execution σ, is there a **correct reordering** of σ with a data race?



Some History

Data Race Prediction

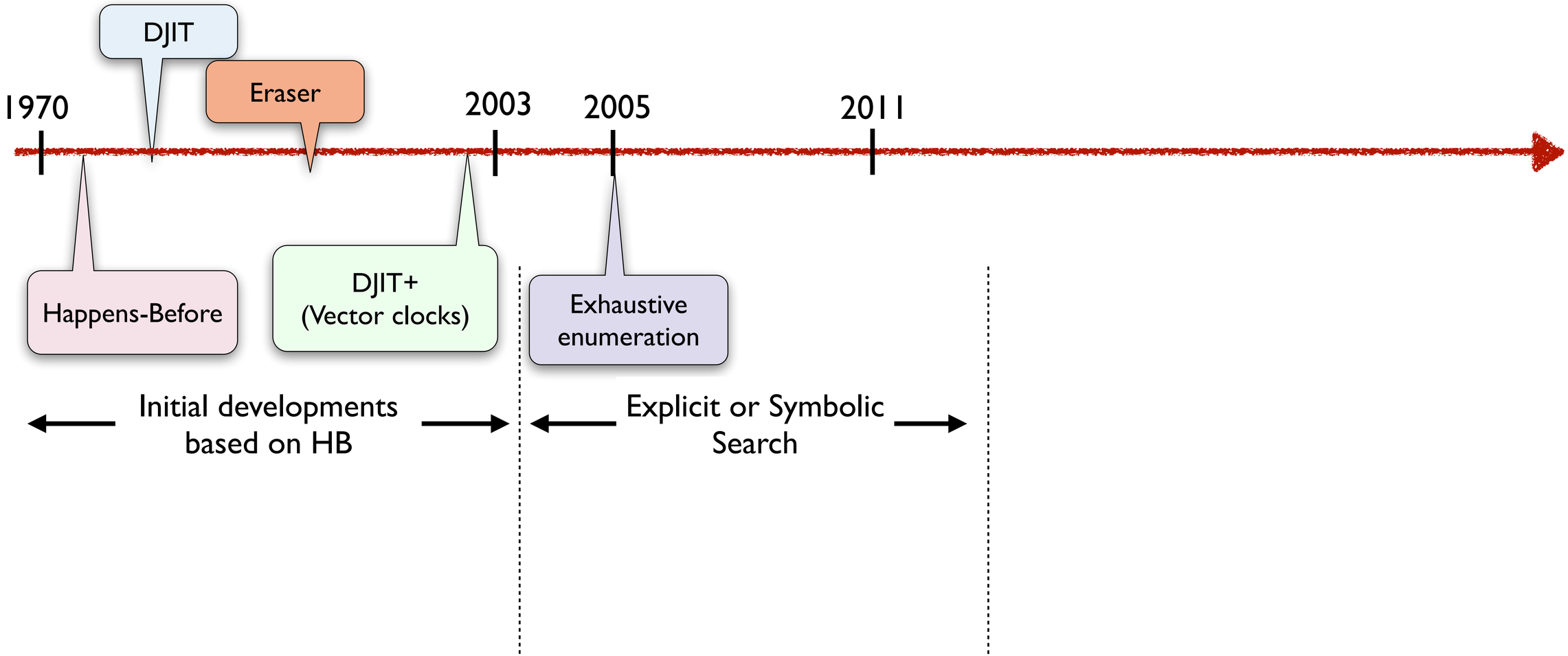
Given an execution σ, is there a **correct reordering** of σ with a data race?



Some History

Data Race Prediction

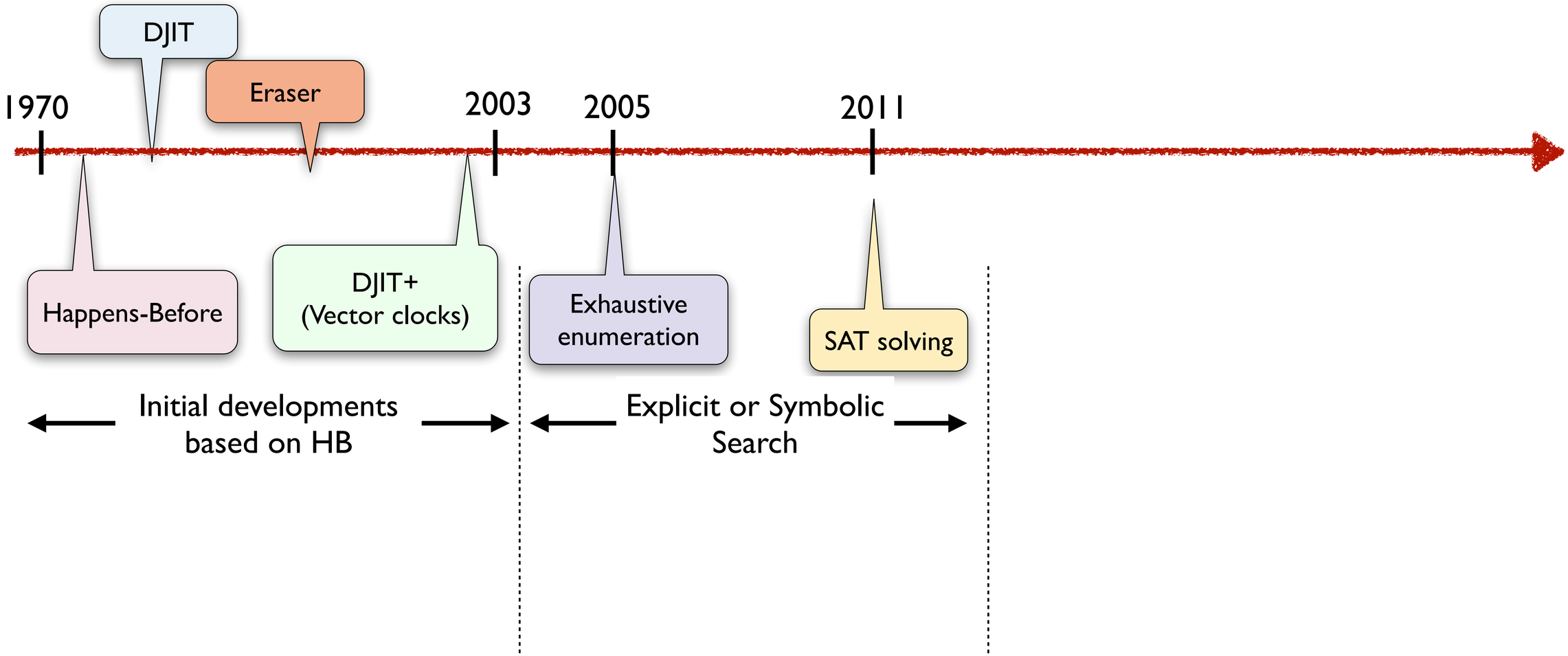
Given an execution σ, is there a **correct reordering** of σ with a data race?



Some History

Data Race Prediction

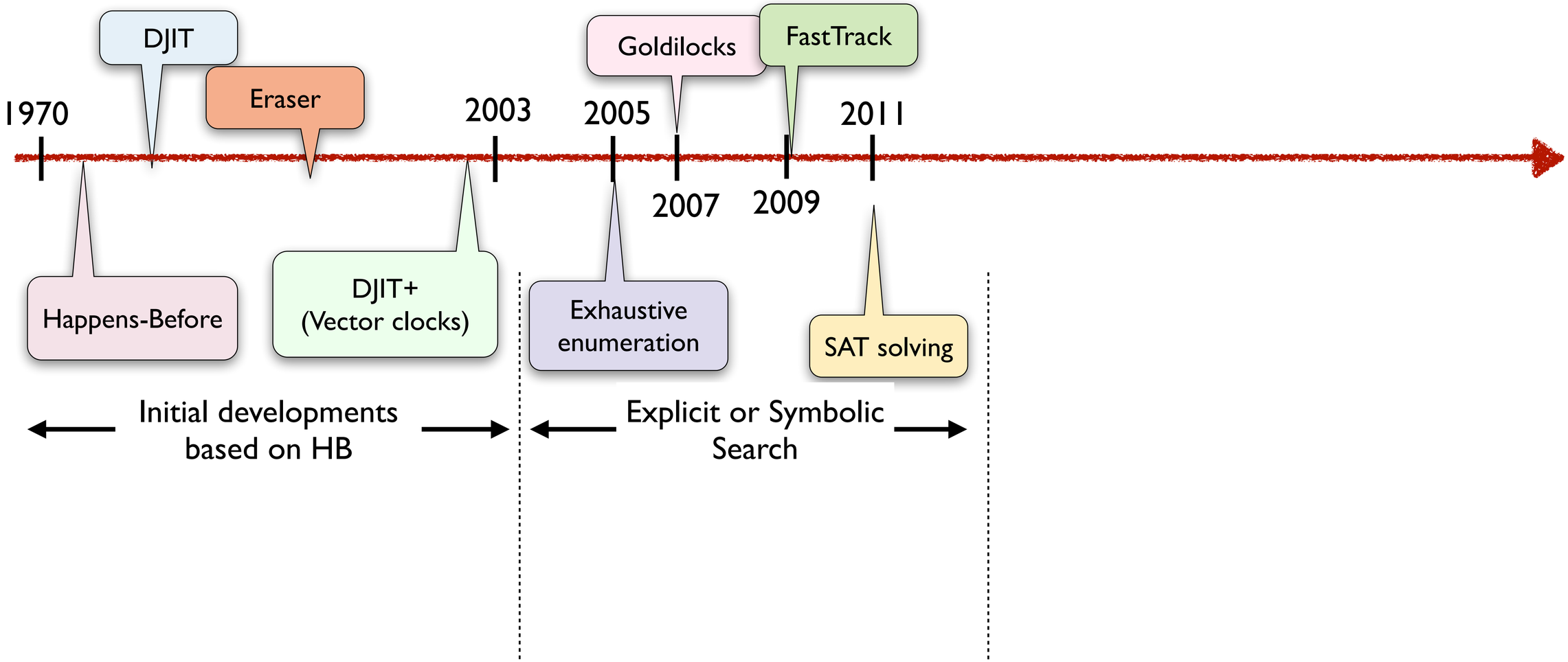
Given an execution σ, is there a **correct reordering** of σ with a data race?



Some History

Data Race Prediction

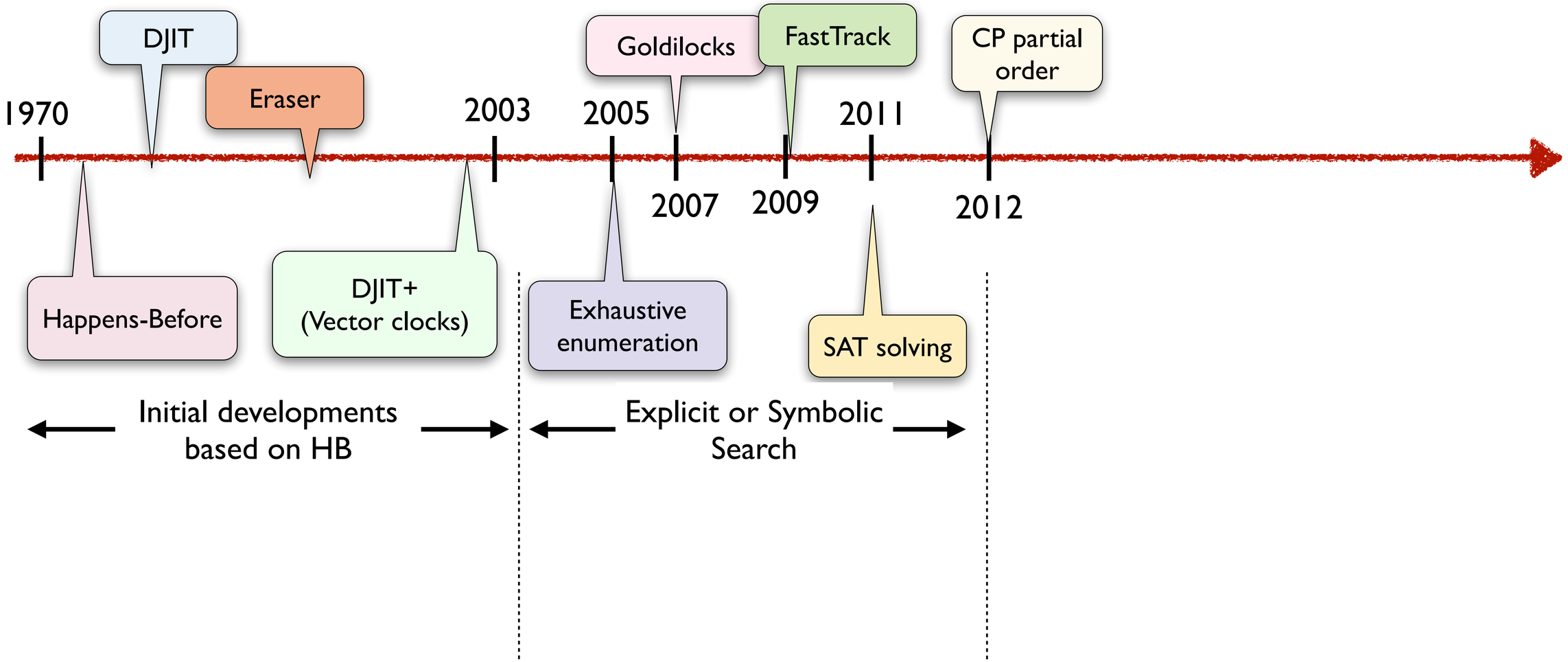
Given an execution σ, is there a **correct reordering** of σ with a data race?



Some History

Data Race Prediction

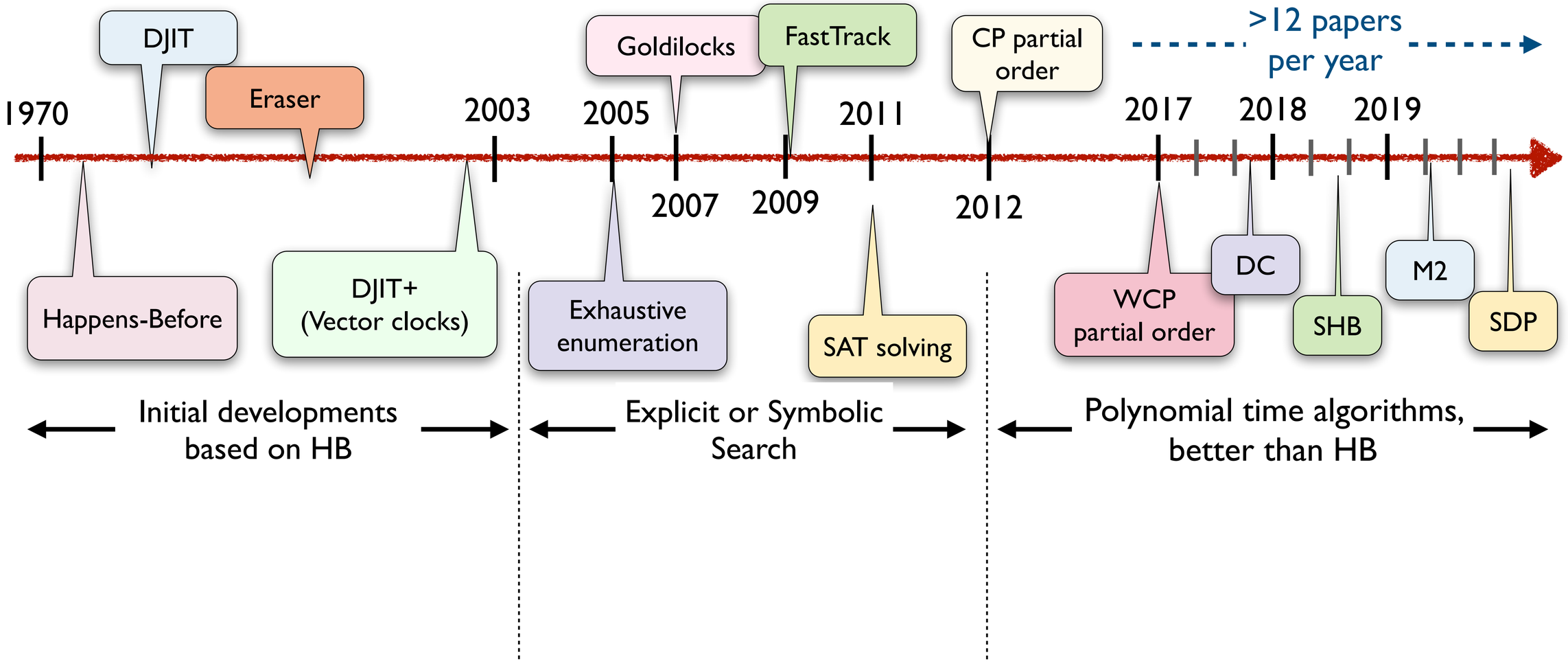
Given an execution σ, is there a **correct reordering** of σ with a data race?



Some History

Data Race Prediction

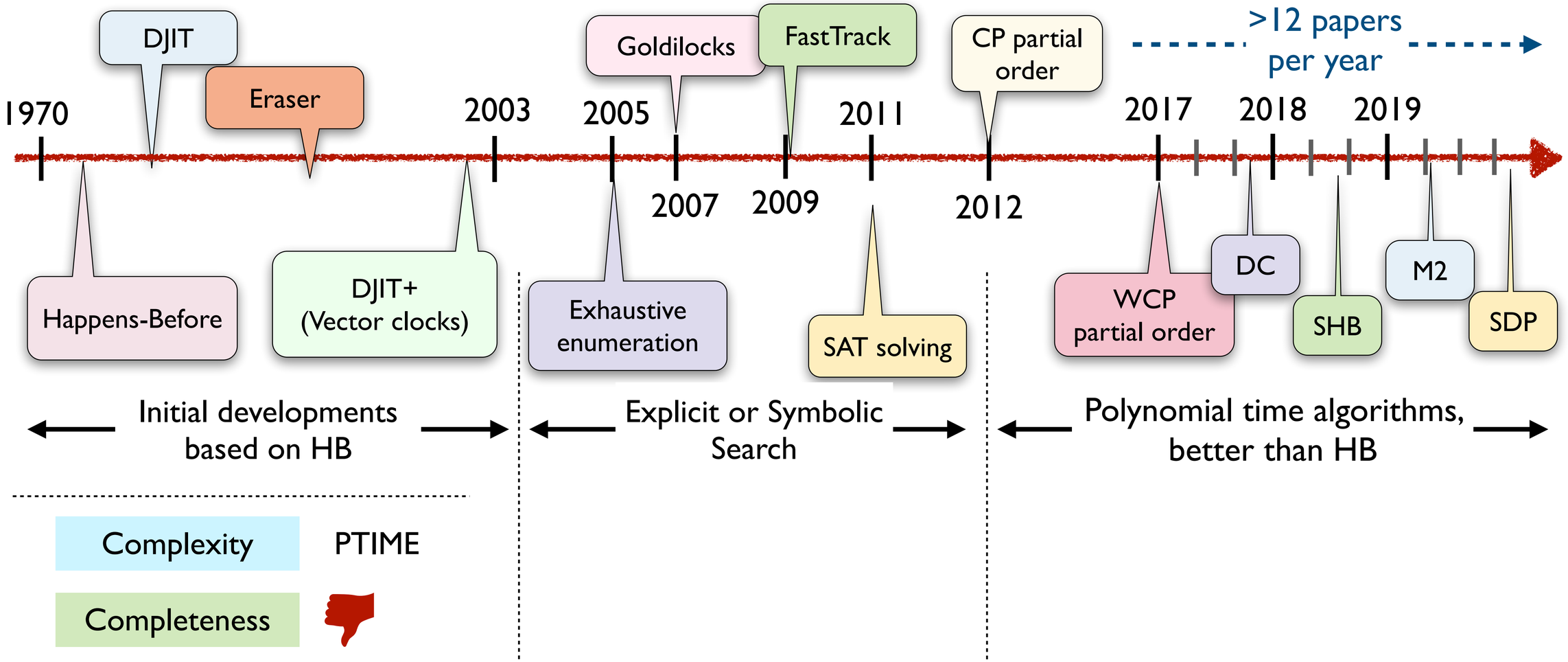
Given an execution σ, is there a **correct reordering** of σ with a data race?



Some History

Data Race Prediction

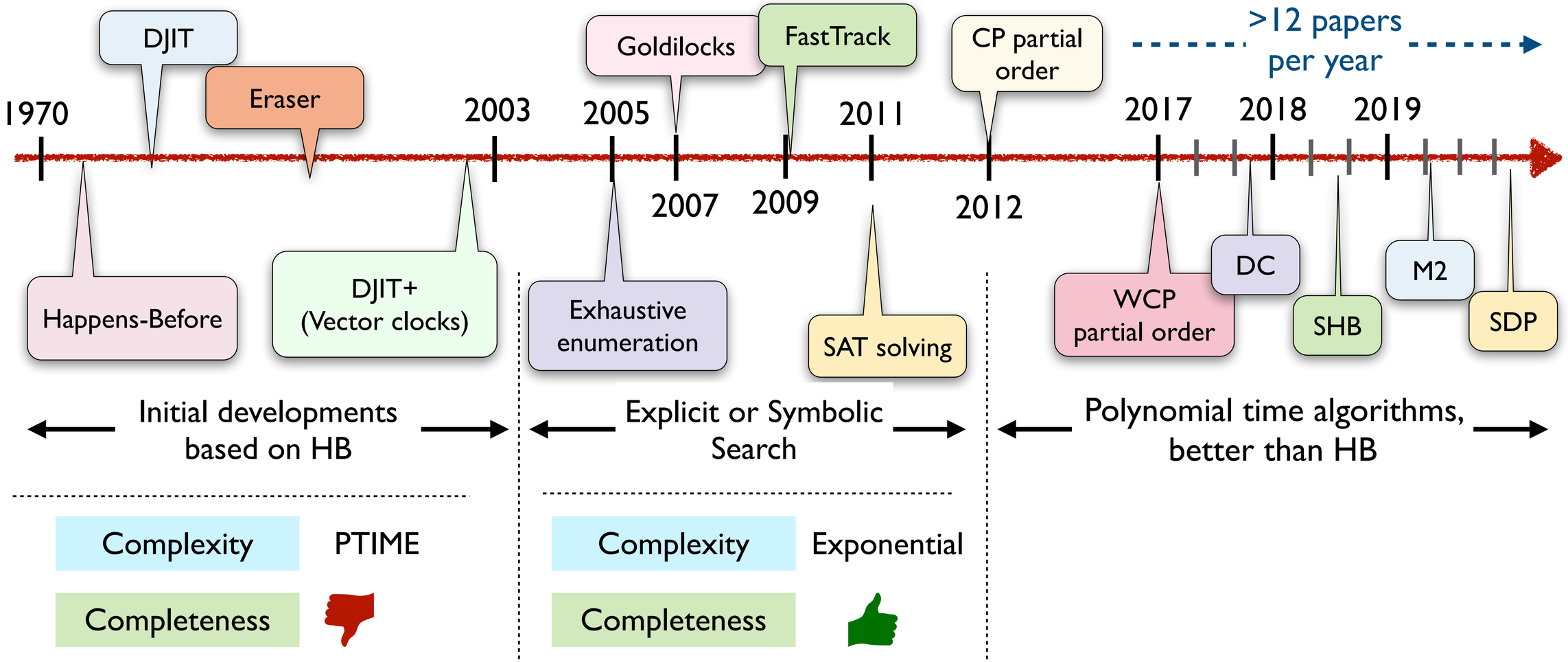
Given an execution σ, is there a **correct reordering** of σ with a data race?



Some History

Data Race Prediction

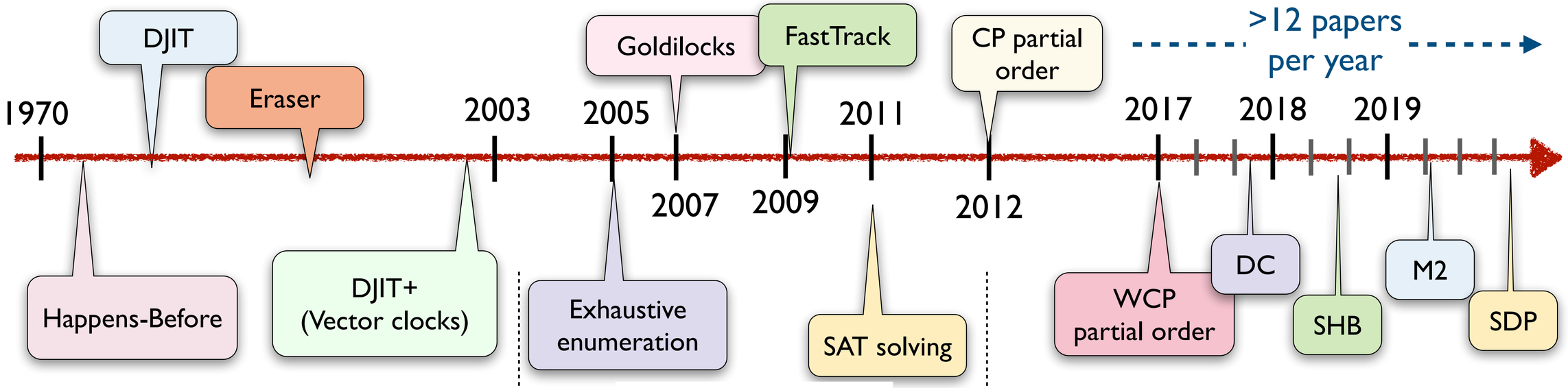
Given an execution σ, is there a **correct reordering** of σ with a data race?



Some History

Data Race Prediction

Given an execution σ, is there a **correct reordering** of σ with a data race?



Initial developments based on HB

Explicit or Symbolic Search

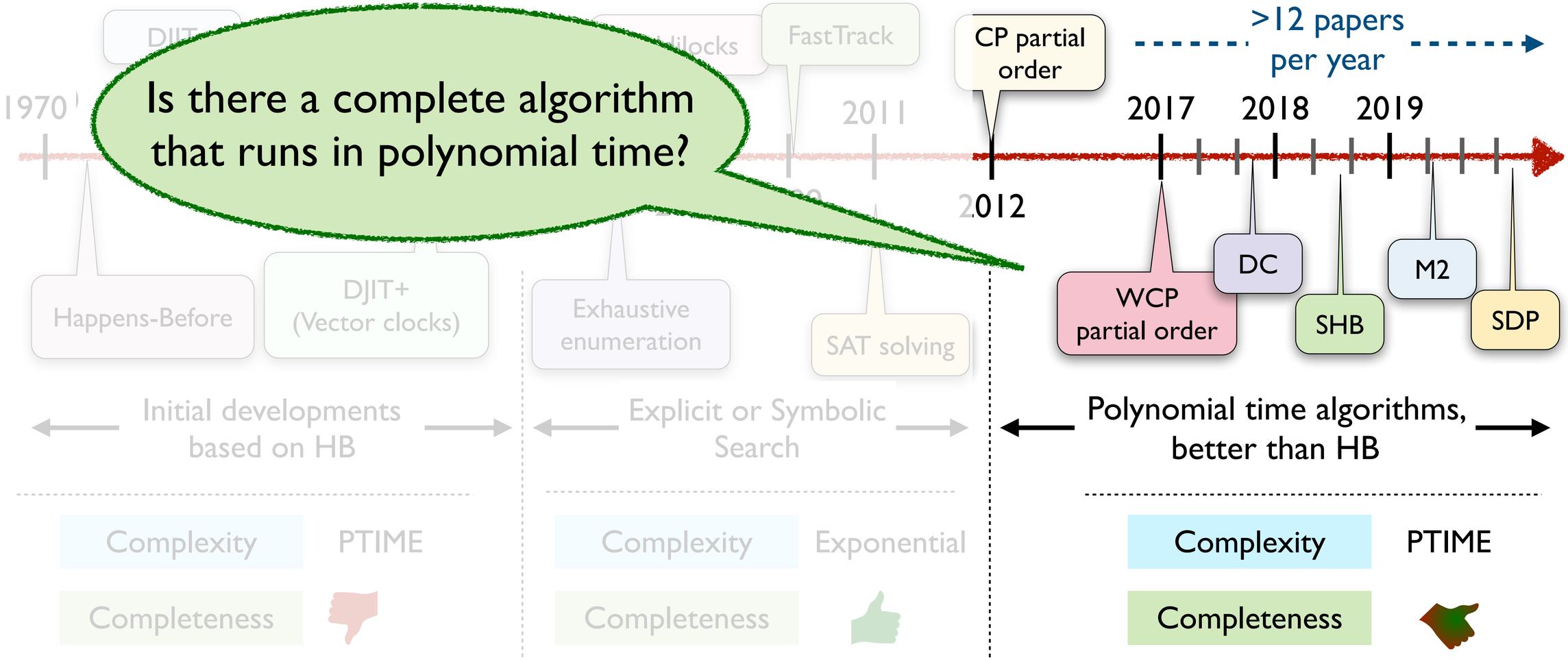
Polynomial time algorithms, better than HB

Complexity	PTIME	Complexity	Exponential	Complexity	PTIME
Completeness	👎	Completeness	👍	Completeness	👉

Some History

Data Race Prediction

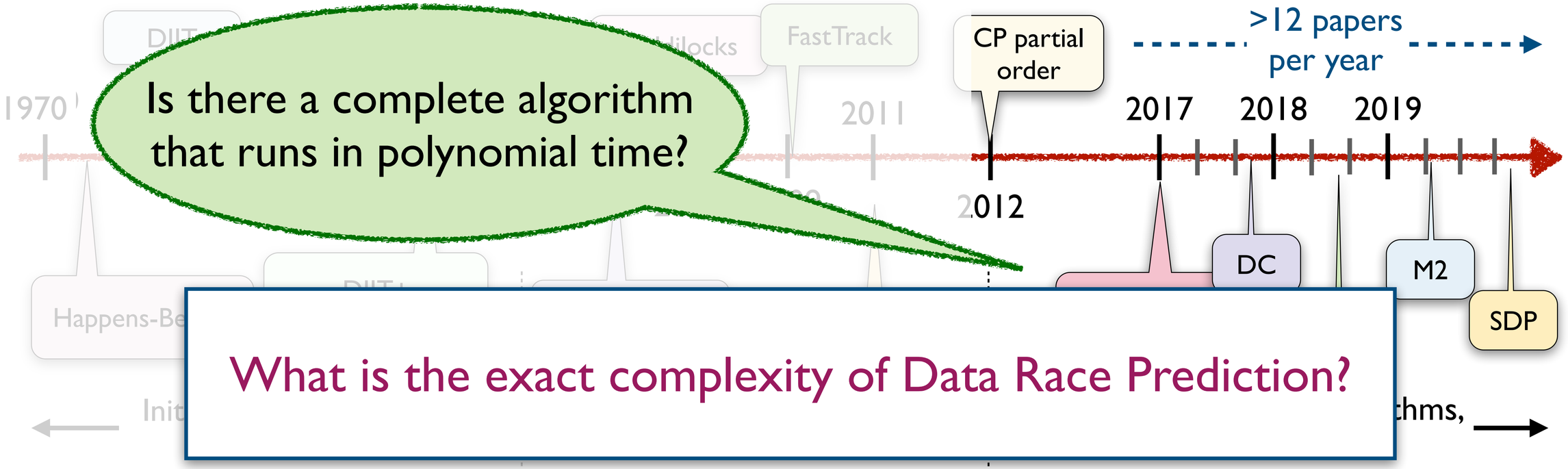
Given an execution σ, is there a **correct reordering** of σ with a data race?



Some History

Data Race Prediction

Given an execution σ, is there a **correct reordering** of σ with a data race?



How hard is Data Race Prediction?

How hard is Data Race Prediction?

Data Race Prediction

How hard is Data Race Prediction?

Data Race Prediction

- **Input:** Trace σ and events e_1 and e_2

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

How hard is Data Race Prediction?

Data Race Prediction

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

How hard is Data Race Prediction?

Data Race Prediction

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

(Easy) Upper Bounds

How hard is Data Race Prediction?

Data Race Prediction

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

(Easy) Upper Bounds

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

How hard is Data Race Prediction?

Data Race Prediction

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

(Easy) Upper Bounds

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

How hard is Data Race Prediction?

Data Race Prediction

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

(Easy) Upper Bounds

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

How hard is Data Race Prediction?

Data Race Prediction

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

(Easy) Upper Bounds

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

Lower Bound

How hard is Data Race Prediction?

Data Race Prediction

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

(Easy) Upper Bounds

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

Lower Bound

- Unknown!
- Is it **NP**-hard? Is enumeration unavoidable?
- Is it polynomial time?

How hard is Data Race Prediction?

Data Race Prediction

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

Closely Related Work

How hard is Data Race Prediction?

Data Race Prediction

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

Closely Related Work

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

How hard is Data Race Prediction?

Data Race Prediction

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

Closely Related Work

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

2. [Gibbons and Korach, '97]: **NP**-hardness of Verifying Sequential Consistency
   - Different problem than data race detection

Contributions

Data Race Prediction

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

Contributions

Data Race Prediction

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

Extensive study of complexity theoretic questions in data race prediction†

Contributions

Data Race Prediction

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

Extensive study of complexity theoretic questions in data race prediction†

General Case

Contributions

Data Race Prediction

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

Extensive study of complexity theoretic questions in data race prediction†

General Case

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

Contributions

Data Race Prediction

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

Extensive study of complexity theoretic questions in data race prediction†

General Case

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

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

Contributions

Data Race Prediction

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

Extensive study of complexity theoretic questions in data race prediction†

General Case

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

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

Special Cases

Contributions

Data Race Prediction

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

Extensive study of complexity theoretic questions in data race prediction†

General Case

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

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

Special Cases

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

Contributions

Data Race Prediction

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

Extensive study of complexity theoretic questions in data race prediction†

General Case

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

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

Special Cases

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

4. Restricting the space of data races to be reported
   - **Linear** time algorithm

Outline

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

Outline

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

Some Notations

	t1	t2
1	acq(l)	
2	rel(l)	
3	w(x)	
4		r(x)
5		acq(l)
6		rel(l)

Some Notations

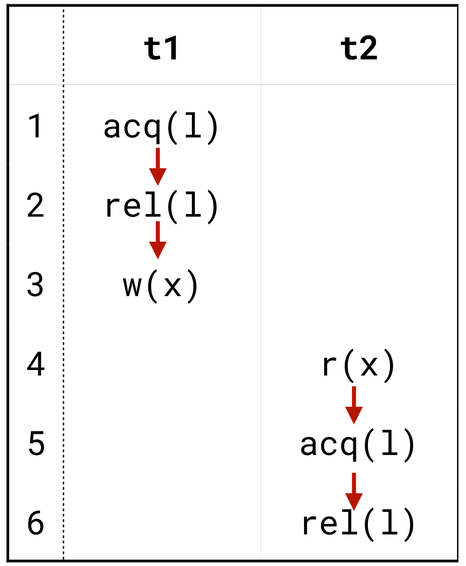
Let σ be an execution trace.

	t1	t2
1	acq(l)	
2	rel(l)	
3	w(x)	
4		r(x)
5		acq(l)
6		rel(l)

Some Notations

Let σ be an execution trace.

- TO_σ = intra-thread ordering on events of σ



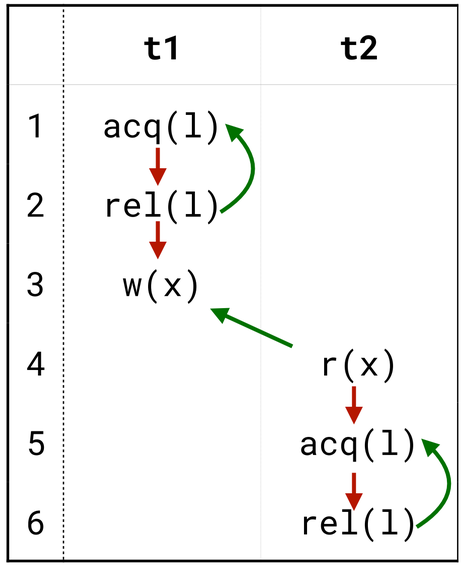
Some Notations

Let σ be an execution trace.

- TO_σ = intra-thread ordering on events of σ
- RF_σ : reads-from mapping of σ:
  - For a read event **r**, RF_σ(**r**) = **w**
  - For a release event **rel**, RF_σ(**rel**) = **acq**

last write on the same memory location

matching acquire



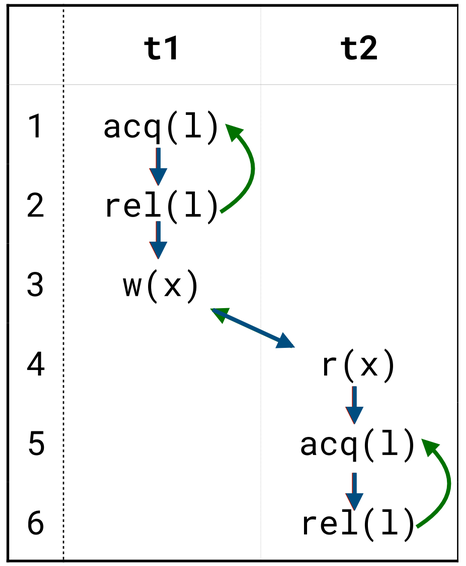
Some Notations

Let σ be an execution trace.

- TO_σ = intra-thread ordering on events of σ
- RF_σ : reads-from mapping of σ:
  - For a read event `r`, RF_σ(`r`) = `w`
  - For a release event `rel`, RF_σ(`rel`) = `acq`
- TRF_σ = smallest partial order that includes TO_σ and respects RF_σ

last write on the same memory location

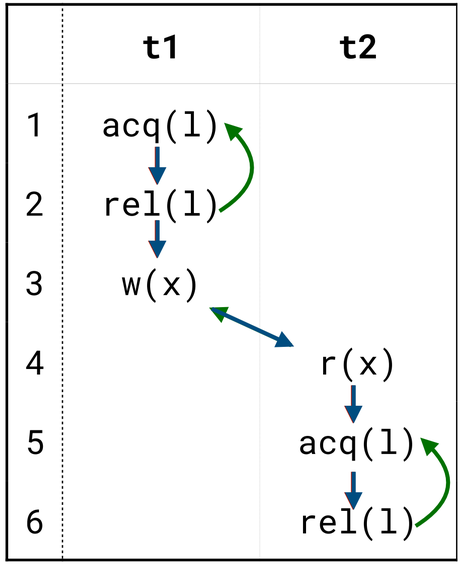
matching acquire



Some Notations

Let σ be an execution trace.

- TO_σ = intra-thread ordering on events of σ
- RF_σ : reads-from mapping of σ:
  - For a read event r, $RF_\sigma(r) = w$ (last write on the same memory location)
  - For a release event rel, $RF_\sigma(rel) = acq$ (matching acquire)
- TRF_σ = smallest partial order that includes TO_σ and respects RF_σ



Observation

Let ρ be a sequence of events with Events(ρ) $\subseteq$ Events(σ).

ρ is a correct reordering of an execution trace σ if

(i) For every lock ℓ, there is at most one unmatched acquire of ℓ in ρ

(ii) ρ respects TO_σ

(iii) $RF_\rho \subseteq RF_\sigma$

Trace Ideals and Realizability

Trace Ideals and Realizability

A feasible trace ideal I of σ is a set of events such that

Trace Ideals and Realizability

A feasible trace ideal I of σ is a set of events such that

- I is downward-closed under TRF_σ, and

Trace Ideals and Realizability

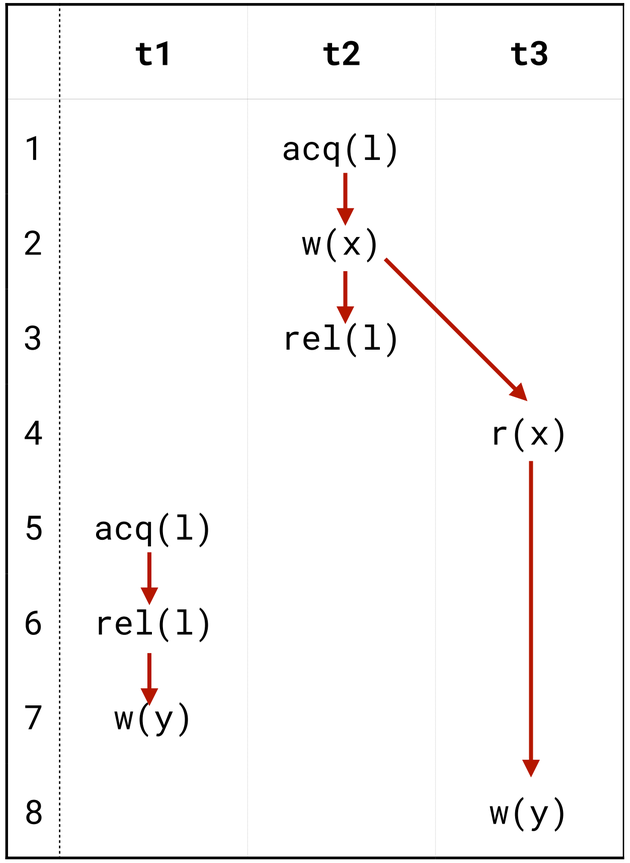
A feasible trace ideal I of σ is a set of events such that

- I is downward-closed under TRF_σ, and
- for every lock ℓ, there is at most one unmatched acquire of ℓ in I

Trace Ideals and Realizability

A feasible trace ideal I of σ is a set of events such that

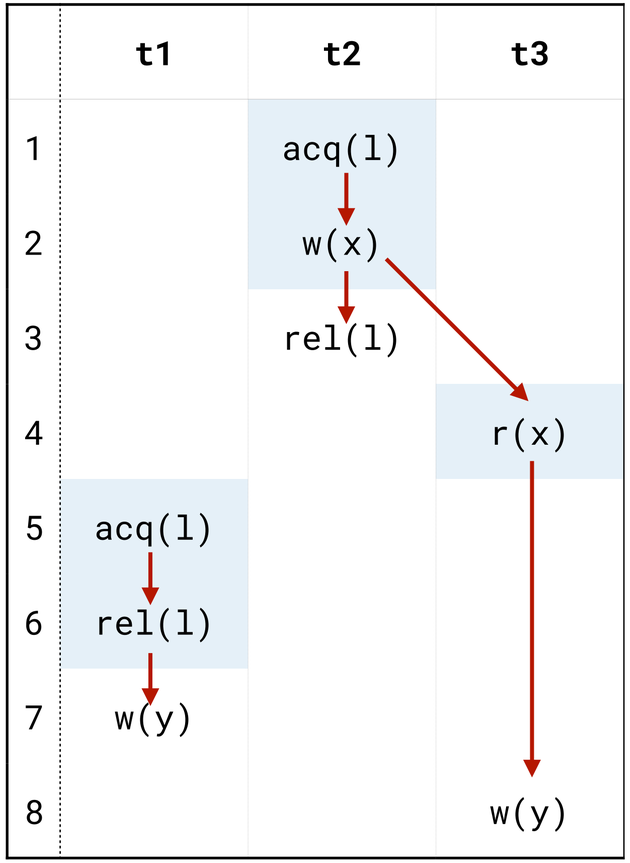
- I is downward-closed under TRF_σ, and
- for every lock ℓ, there is at most one unmatched acquire of ℓ in I



Trace Ideals and Realizability

A feasible trace ideal I of σ is a set of events such that

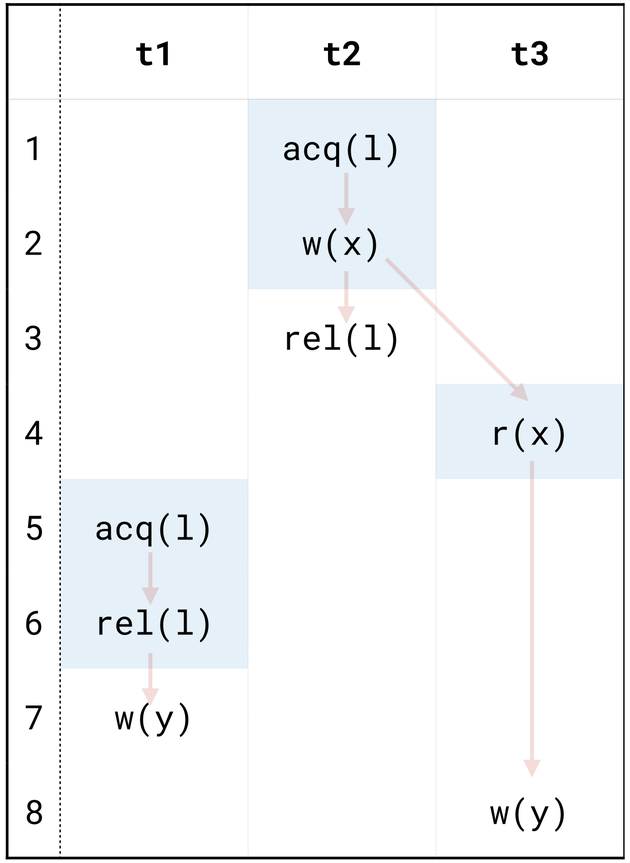
- I is downward-closed under TRF_σ, and
- for every lock ℓ, there is at most one unmatched acquire of ℓ in I



Trace Ideals and Realizability

A feasible trace ideal I of σ is a set of events such that

- I is downward-closed under TRF_σ, and
- for every lock ℓ, there is at most one unmatched acquire of ℓ in I

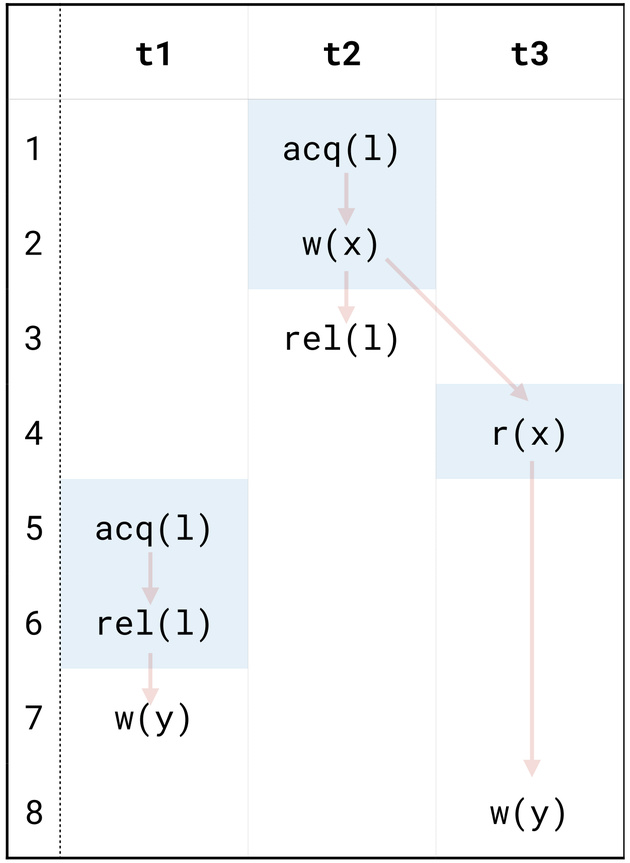


Trace Ideals and Realizability

A feasible trace ideal I of σ is a set of events such that

- I is downward-closed under TRF_σ, and
- for every lock ℓ, there is at most one unmatched acquire of ℓ in I

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



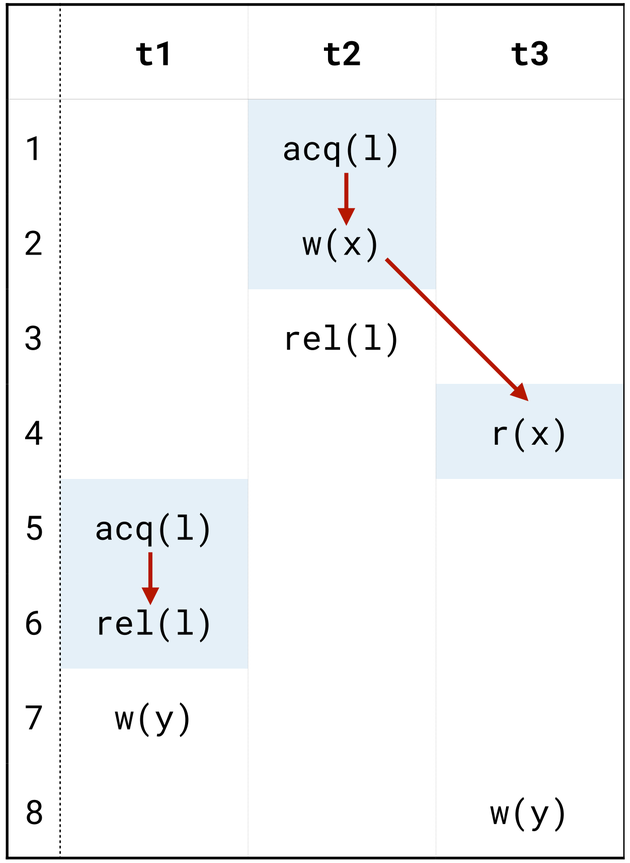
Trace Ideals and Realizability

A feasible trace ideal I of σ is a set of events such that

- I is downward-closed under TRF_σ, and
- for every lock ℓ, there is at most one unmatched acquire of ℓ in I

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

- $\mathrm{TRF}_\sigma \downarrow_I \subseteq \mathscr{P}(I)$



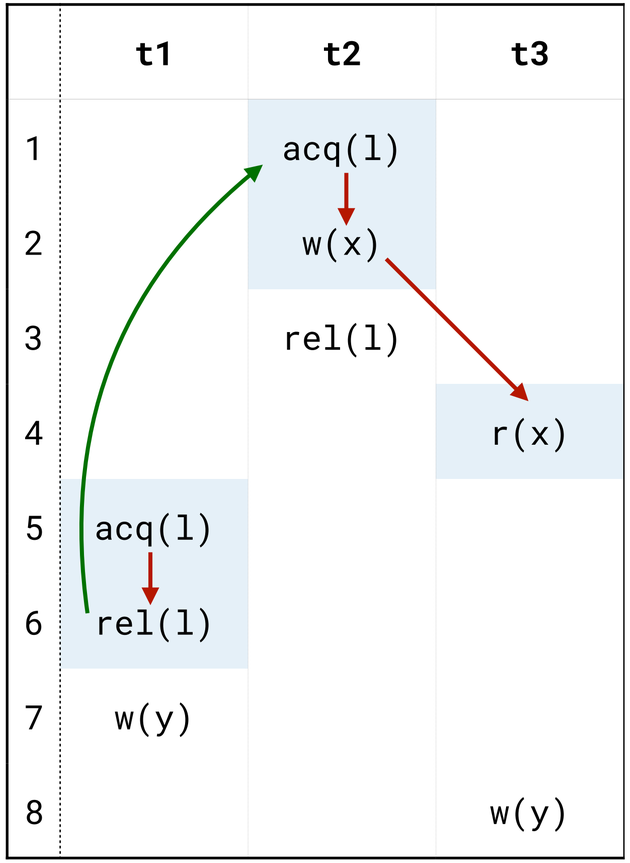
Trace Ideals and Realizability

A feasible trace ideal I of σ is a set of events such that

- I is downward-closed under TRF_σ, and
- for every lock ℓ, there is at most one unmatched acquire of ℓ in I

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

- $\mathrm{TRF}_\sigma \downarrow_I \subseteq \mathscr{P}(I)$
- For every lock ℓ with an unmatched acquire $\mathtt{acq}_{\mathtt{unmatched}} \in I$, and for every ℓ-release event $\mathtt{rel} \in I$, we have $\mathtt{rel} \leq_{\mathscr{P}(I)} \mathtt{acq}_{\mathtt{unmatched}}$



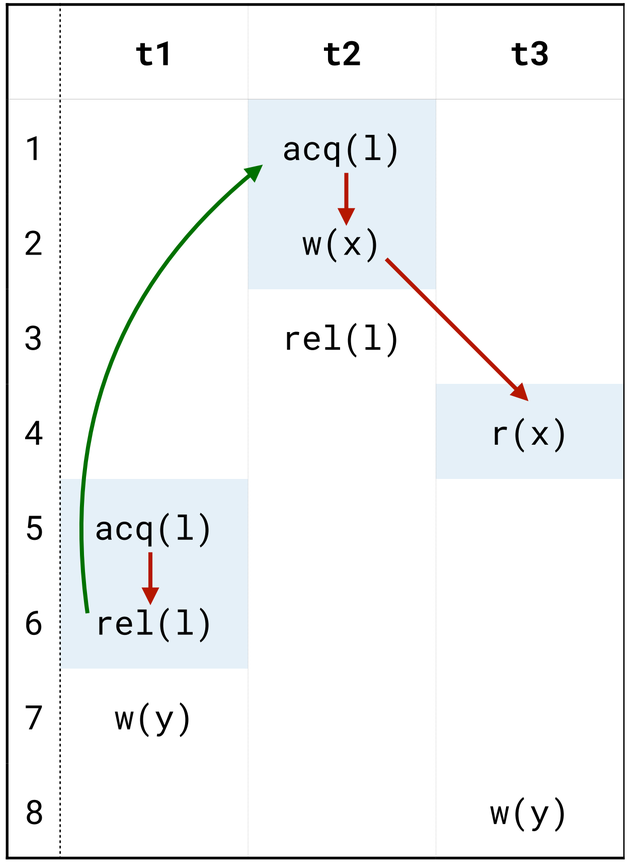
Trace Ideals and Realizability

A feasible trace ideal I of σ is a set of events such that

- I is downward-closed under TRF_σ, and
- for every lock ℓ, there is at most one unmatched acquire of ℓ in I

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

- $\mathrm{TRF}_\sigma \downarrow_I \subseteq \mathscr{P}(I)$
- For every lock ℓ with an unmatched acquire $\mathbf{acq}_{\mathtt{unmatched}} \in I$, and for every ℓ-release event $\mathbf{rel} \in I$, we have $\mathbf{rel} \leq_{\mathscr{P}(I)} \mathbf{acq}_{\mathtt{unmatched}}$



Realizability of Trace Ideals

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

Outline

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

Data Race Prediction as Ideal Realizability

Data Race Prediction as Ideal Realizability

Proposition

A pair of conflicting events (e_1, e_2) is a predictable data race of σ

iff

there exists a feasible trace ideal I of σ such that I is ***realizable***

and both e_1 and e_2 are **enabled** in I.

Data Race Prediction as Ideal Realizability

Proposition

A pair of conflicting events (e_1, e_2) is a predictable data race of σ iff

there exists a feasible trace ideal I of σ such that I is ***realizable*** and both e_1 and e_2 are **enabled** in I.

If σ^* is a witness to the realizability of a feasible trace ideal I of σ, then σ^* is a correct reordering of σ

Data Race Prediction as Ideal Realizability

Proposition

A pair of conflicting events (e_1, e_2) is a predictable data race of σ iff

there exists a feasible trace ideal I of σ such that I is ***realizable*** and both e_1 and e_2 are **enabled** in I.

If σ^* is a witness to the realizability of a feasible trace ideal I of σ, then σ^* is a correct reordering of σ

Event e of σ is enabled in a feasible trace ideal I of σ if $I' = I \cup \{e\}$ is closed under TO_σ

Data Race Prediction as Ideal Realizability

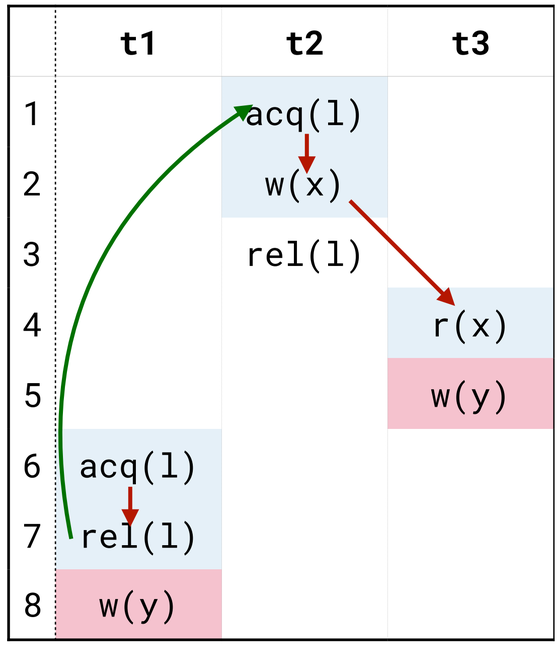
Proposition

A pair of conflicting events (e_1, e_2) is a predictable data race of **σ** iff

there exists a feasible trace ideal I of **σ** such that I is ***realizable*** and both e_1 and e_2 are **enabled** in I.

If **σ*** is a witness to the realizability of a feasible trace ideal I of **σ**, then **σ*** is a correct reordering of **σ**

Event e of **σ** is enabled in a feasible trace ideal I of **σ** if $I' = I \cup \{e\}$ is closed under TO_{σ}



Data Race Prediction as Ideal Realizability

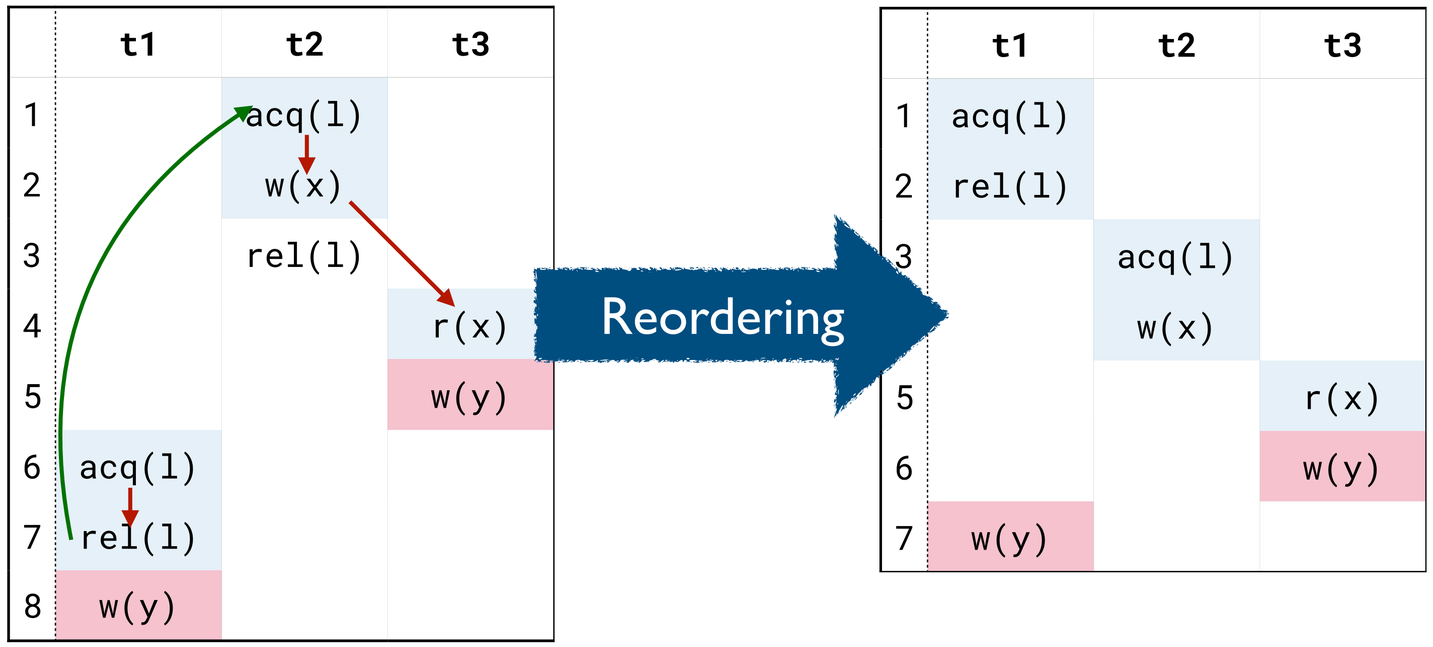
Proposition

A pair of conflicting events (e_1, e_2) is a predictable data race of **σ** iff

there exists a feasible trace ideal I of **σ** such that I is ***realizable*** and both e_1 and e_2 are **enabled** in I.

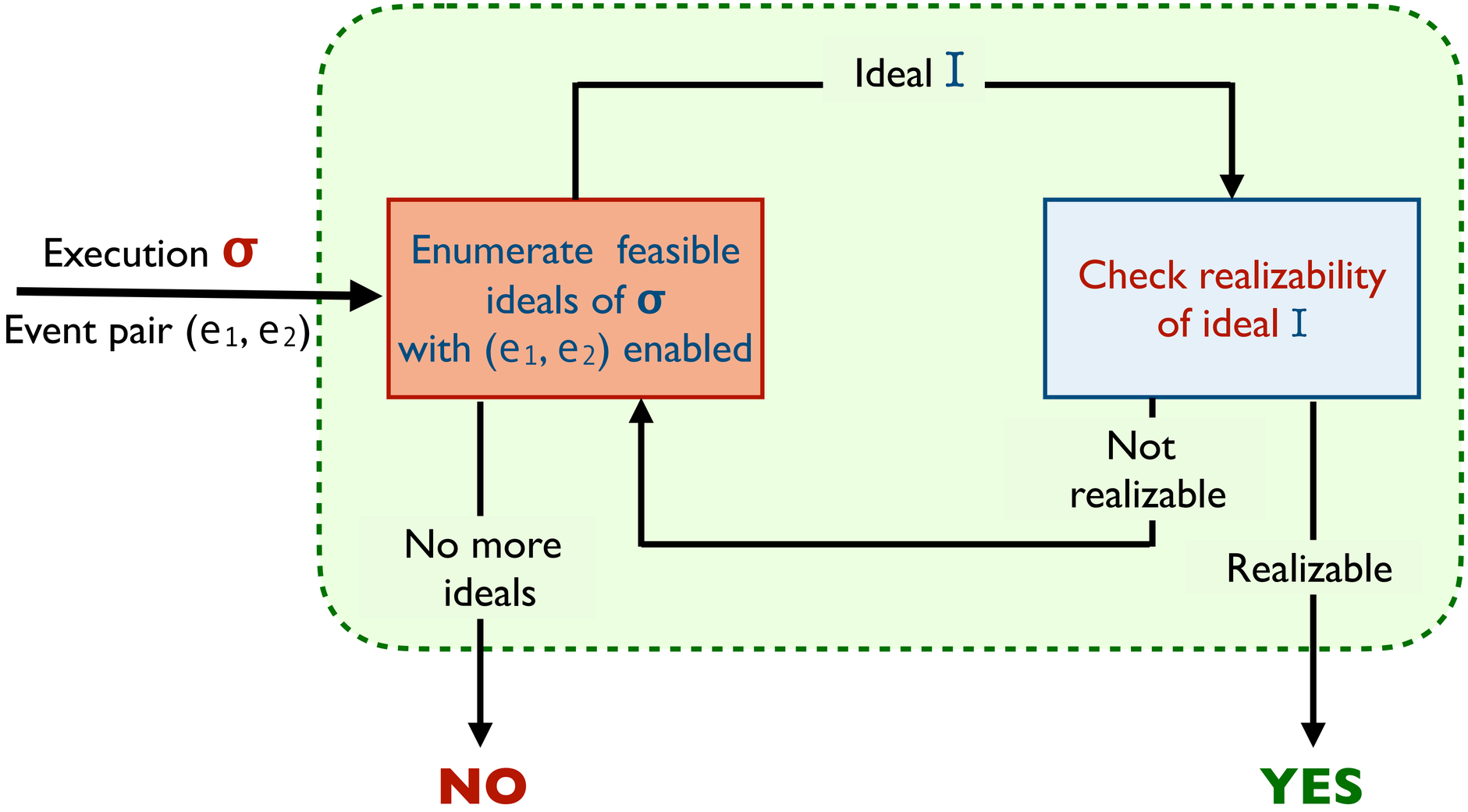
If **σ*** is a witness to the realizability of a feasible trace ideal I of **σ**, then **σ*** is a correct reordering of **σ**

Event e of **σ** is enabled in a feasible trace ideal I of **σ** if $I' = I \cup \{e\}$ is closed under TO_{σ}

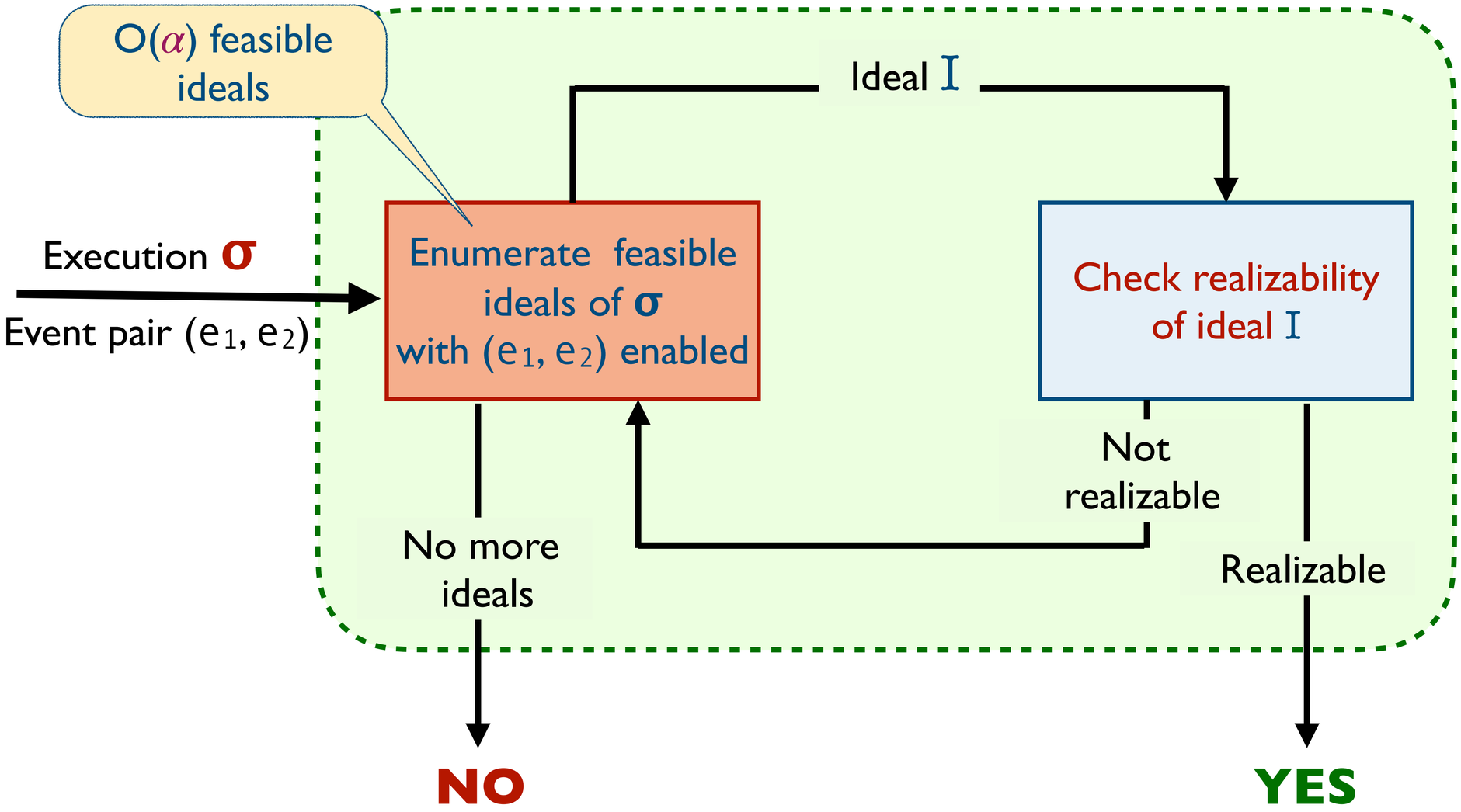


Algorithm for Data Race Prediction

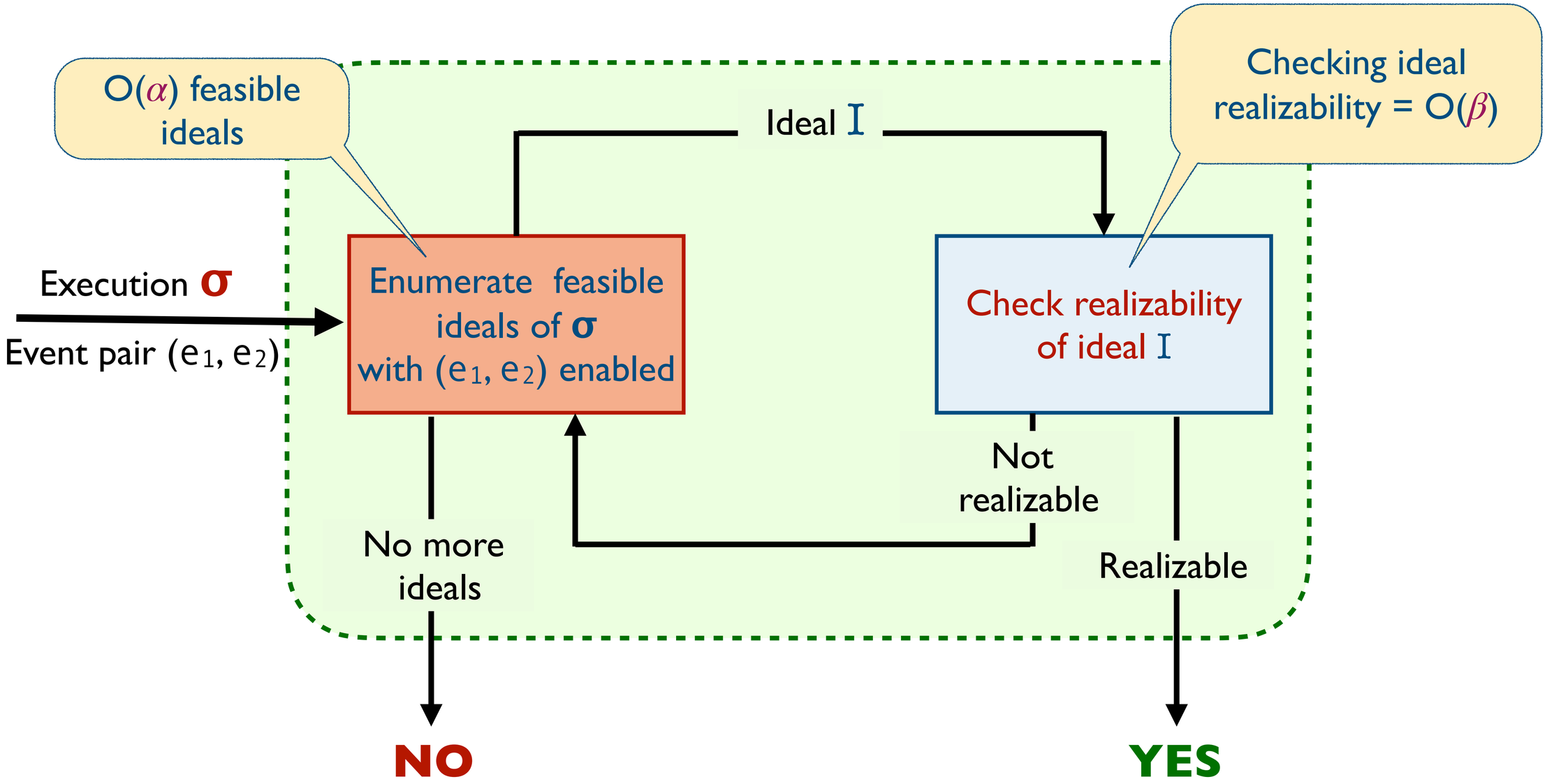
Algorithm for Data Race Prediction



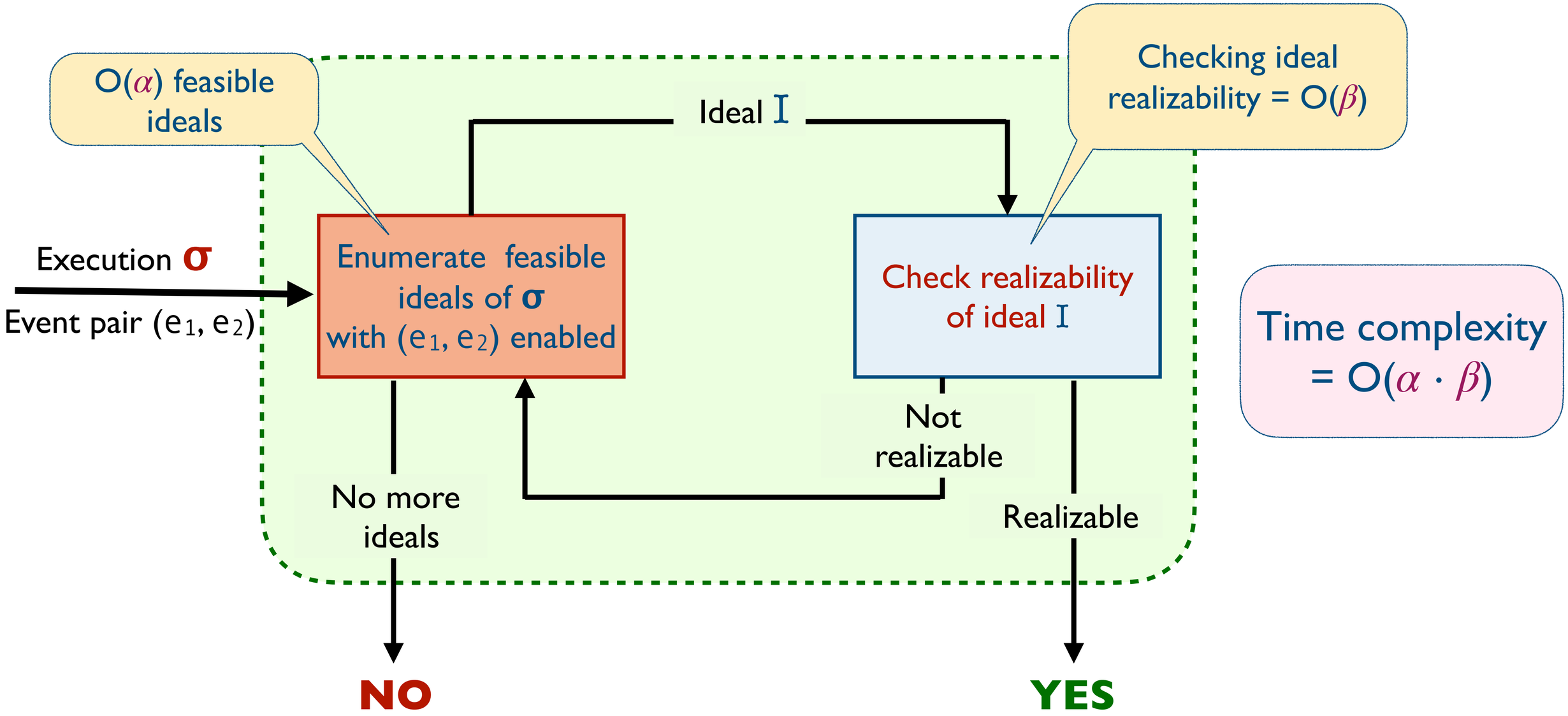
Algorithm for Data Race Prediction



Algorithm for Data Race Prediction



Algorithm for Data Race Prediction



Algorithm for Data Race Prediction

Algorithm for Data Race Prediction

Lemma

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

Algorithm for Data Race Prediction

Lemma

Let σ be an execution with n events, k threads, lock nesting depth γ and lock-dependence factor ζ.

Let e_1 and e_2 be events of σ.

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

Measures data flow between critical sections

Algorithm for Data Race Prediction

Lemma

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

Measures data flow between critical sections

Lemma

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

Algorithm for Data Race Prediction

Lemma

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

Measures data flow between critical sections

Lemma

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

Theorem

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

Outline

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

Tree Communication Topology

Tree Communication Topology

The communication topology of σ is an undirected graph $G_\sigma = (V_\sigma, E_\sigma)$:

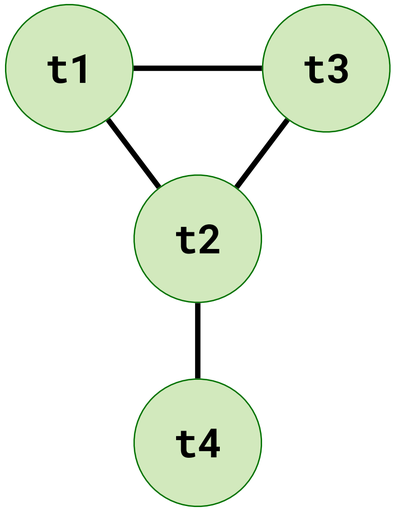
- V_σ = set of threads in σ
- E_σ = {(t, t') | t ≠ t' are threads that perform conflicting accesses or acquire a common lock}

Tree Communication Topology

The communication topology of σ is an undirected graph $G_\sigma = (V_\sigma, E_\sigma)$:

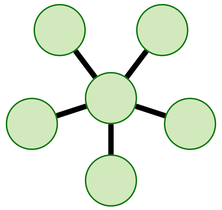
- V_σ = set of threads in σ
- E_σ = {(t, t') | t ≠ t' are threads that perform conflicting accesses or acquire a common lock}

	t1	t2	t3	t4
1		acq(l)		
2		w(x)		
3		rel(l)		
4			r(x)	
5	acq(l)			
6	rel(l)			
7	w(y)			
8				r(x)
9			w(y)	

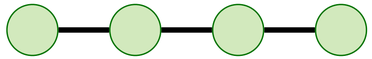


Acyclic Communication Topology

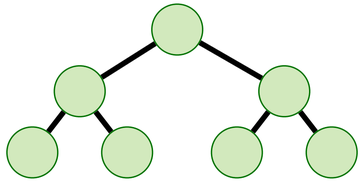
Acyclic Communication Topology



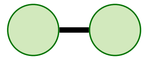
Server-client



Pipeline

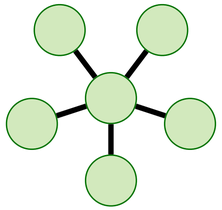


Divide-and-conquer

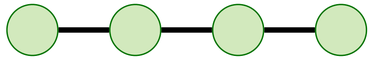


Two threads

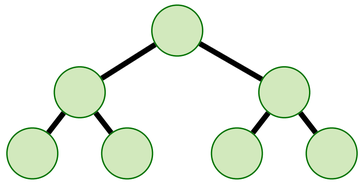
Acyclic Communication Topology



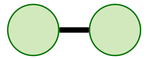
Server-client



Pipeline



Divide-and-conquer

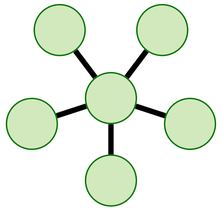


Two threads

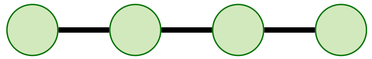
Lemma

The realizability of a feasible ideal I of an execution σ (with n events, k threads and d variables) having an acyclic communication topology can be determined in $O(k^2 \cdot d^2 \cdot n^2 \cdot \log n)$ time.

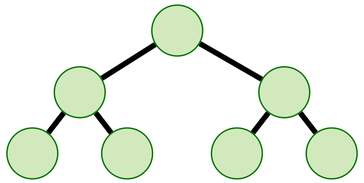
Acyclic Communication Topology



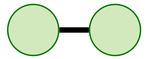
Server-client



Pipeline



Divide-and-conquer



Two threads

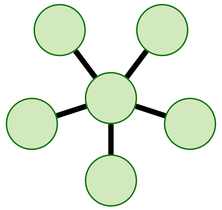
Lemma

The realizability of a feasible ideal I of an execution σ (with n events, k threads and d variables) having an acyclic communication topology can be determined in $O(k^2 \cdot d^2 \cdot n^2 \cdot \log n)$ time.

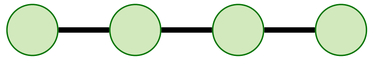
Lemma

Let σ be an execution with an acyclic communication topology and let e_1 and e_2 be events of σ. There is a feasible ideal $I_{(e1, e2)}$ such that $I_{(e1, e2)}$ is realizable iff (e_1, e_2) is a predictable data race of σ.

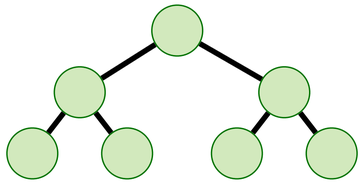
Acyclic Communication Topology



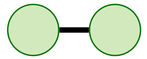
Server-client



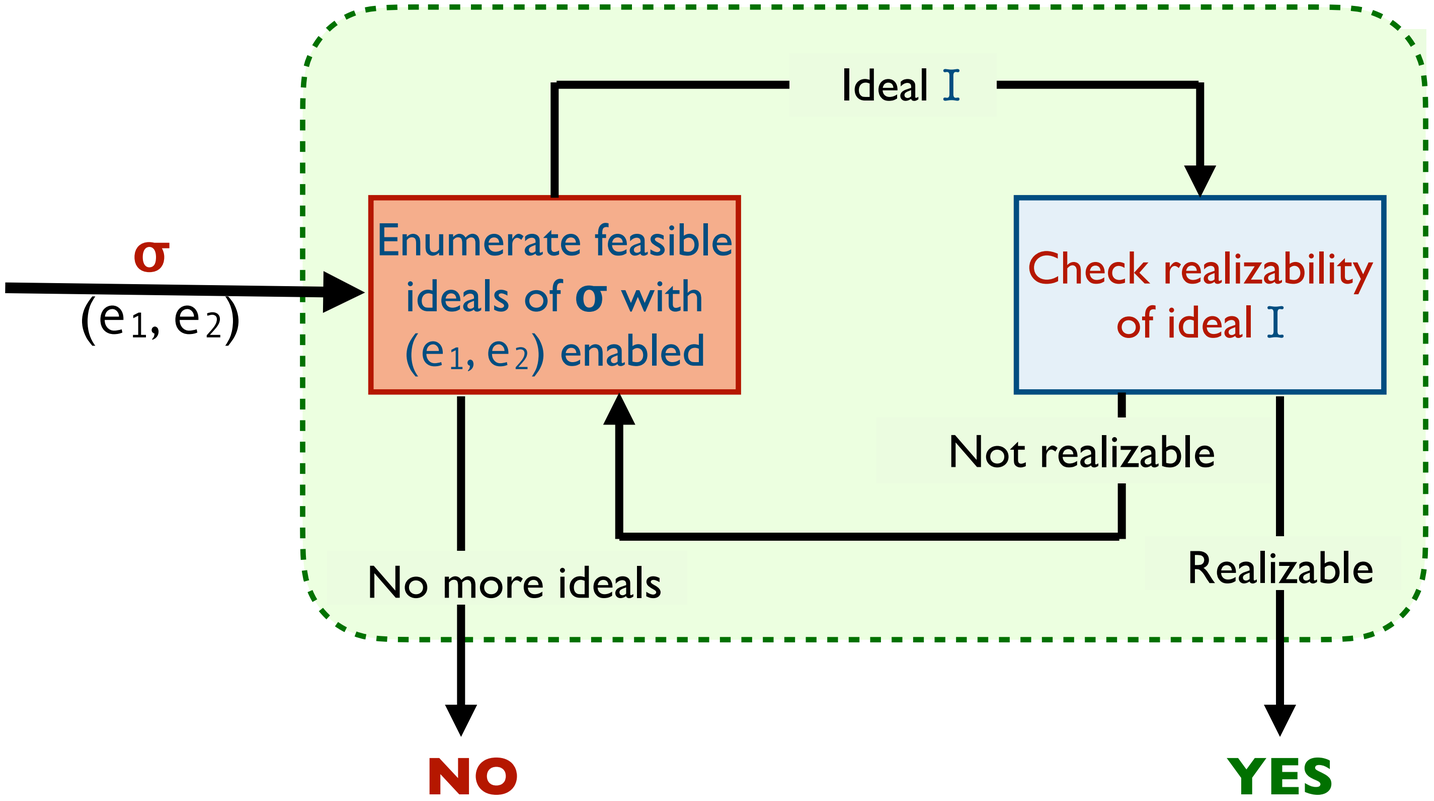
Pipeline



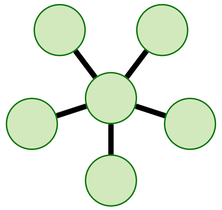
Divide-and-conquer



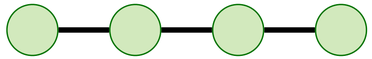
Two threads



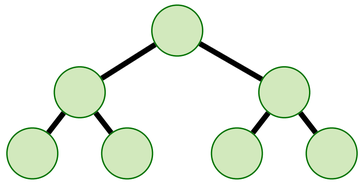
Acyclic Communication Topology



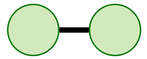
Server-client



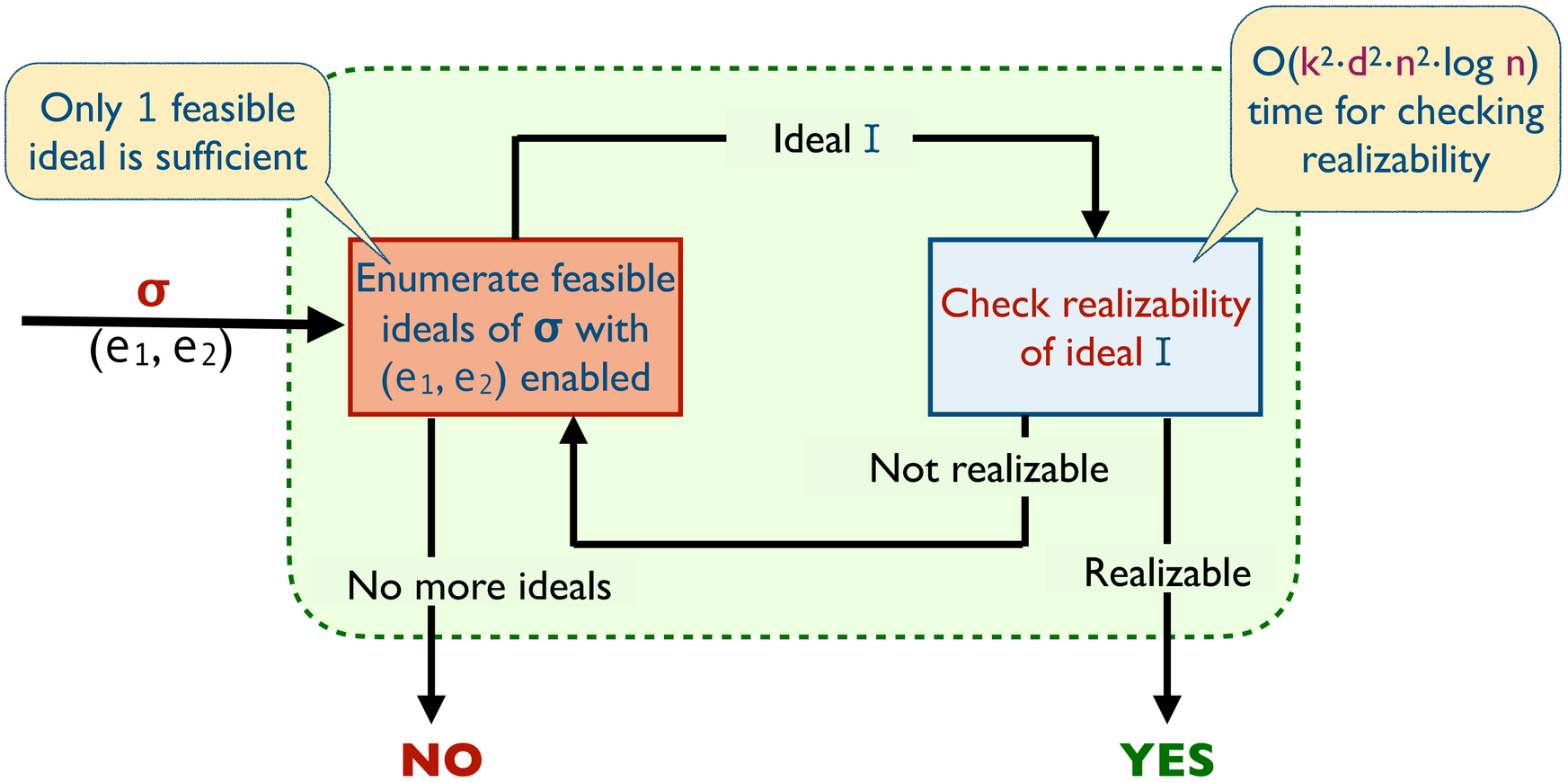
Pipeline



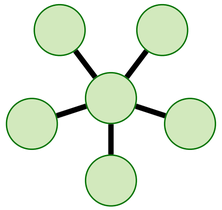
Divide-and-conquer



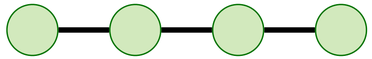
Two threads



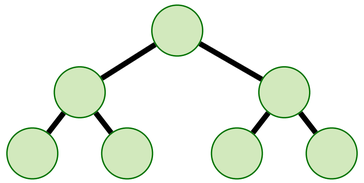
Acyclic Communication Topology



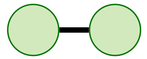
Server-client



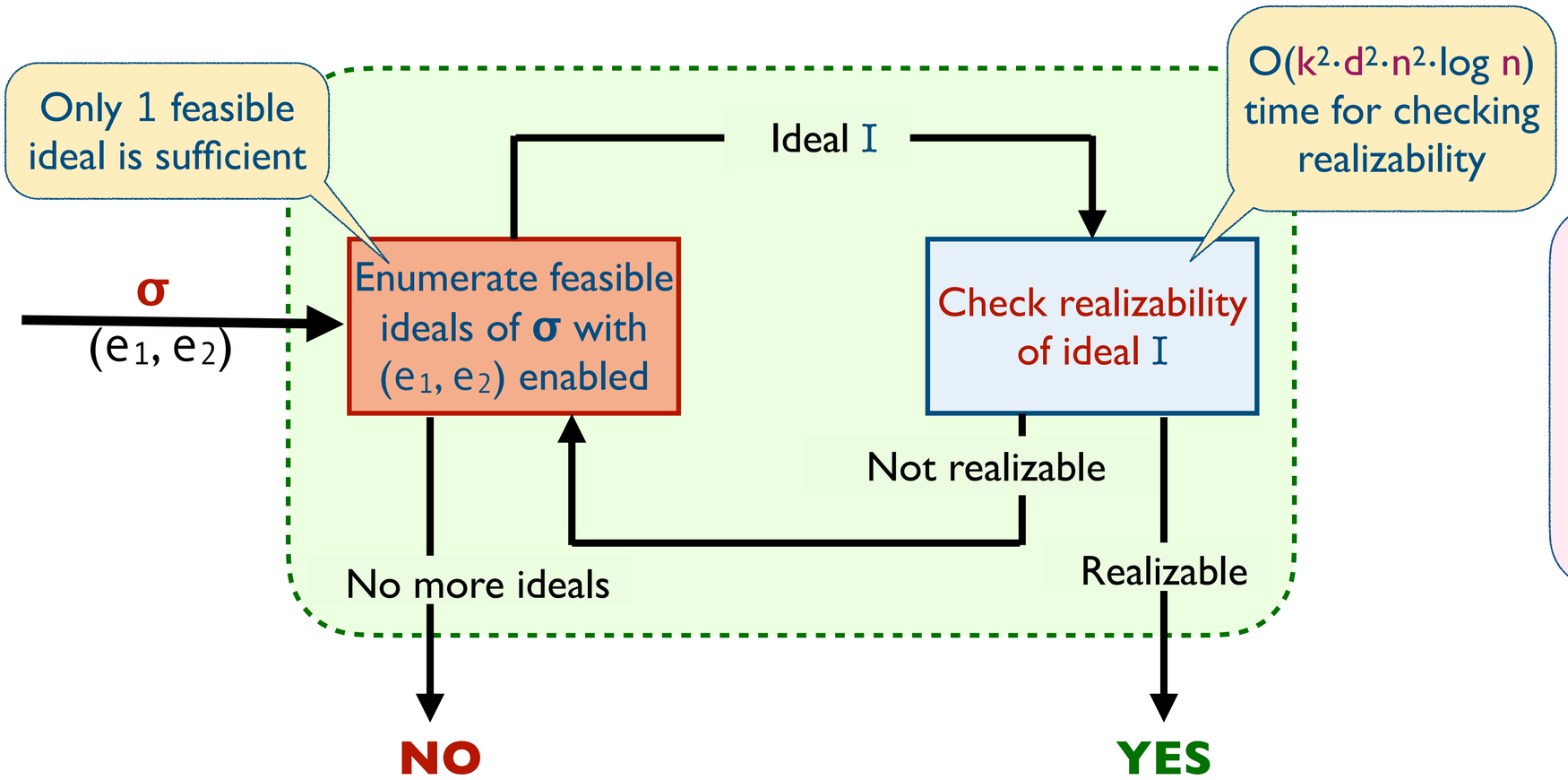
Pipeline



Divide-and-conquer



Two threads



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

Outline

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

Distance-Bounded Data Races

Distance-Bounded Data Races

Set of reversals between σ and its correct reordering ρ:

Distance-Bounded Data Races

Set of reversals between σ and its correct reordering ρ:

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

Distance-Bounded Data Races

Set of reversals between σ and its correct reordering ρ:

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

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

Distance-Bounded Data Races

Set of reversals between **σ** and its correct reordering **ρ**:

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

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

	t1	t2
1	acq(l)	
2	rel(l)	
3	w(x)	
4		w(x)
5		acq(l)
6		rel(l)

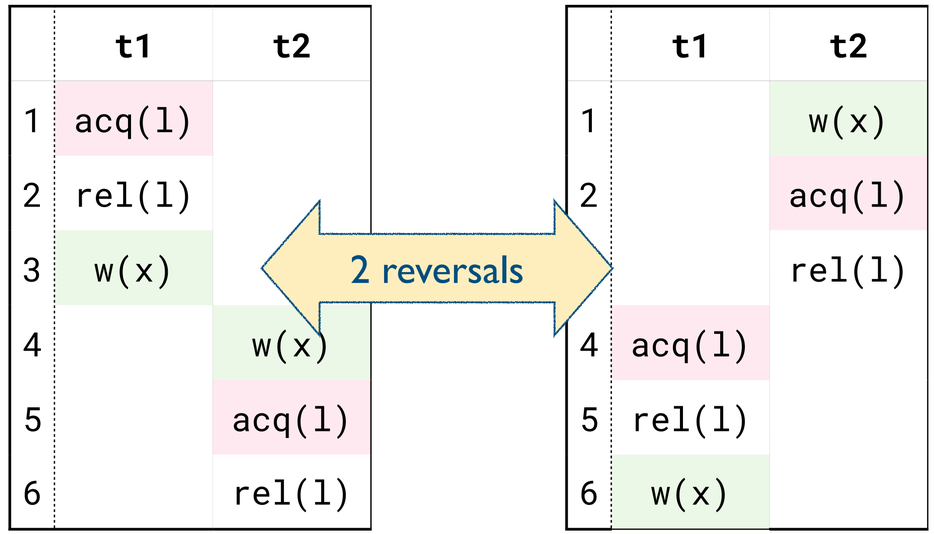
	t1	t2
1		w(x)
2		acq(l)
3		rel(l)
4	acq(l)	
5	rel(l)	
6	w(x)	

Distance-Bounded Data Races

Set of reversals between **σ** and its correct reordering **ρ**:

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

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

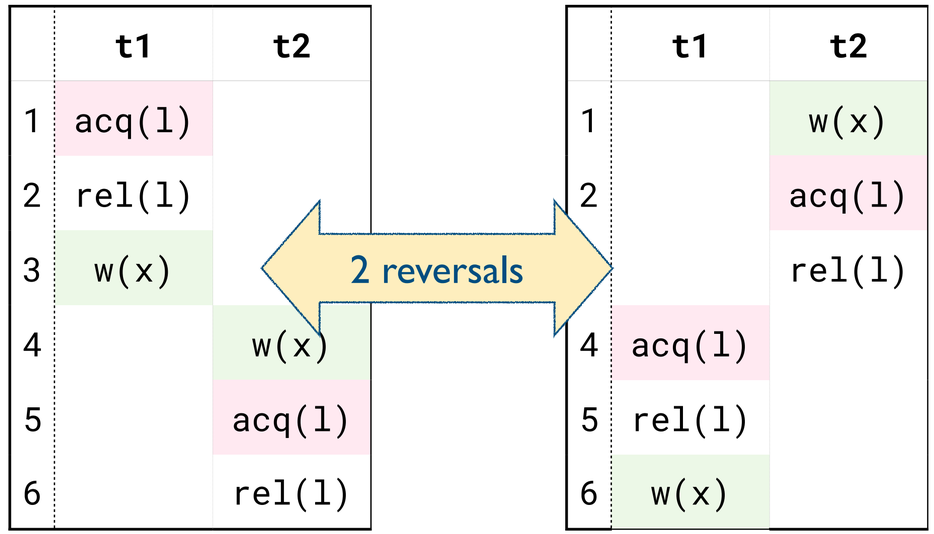


Distance-Bounded Data Races

Set of reversals between σ and its correct reordering ρ:

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

Distance between σ and ρ is $\delta(\sigma, \rho) = |Rev(\sigma, \rho)|$



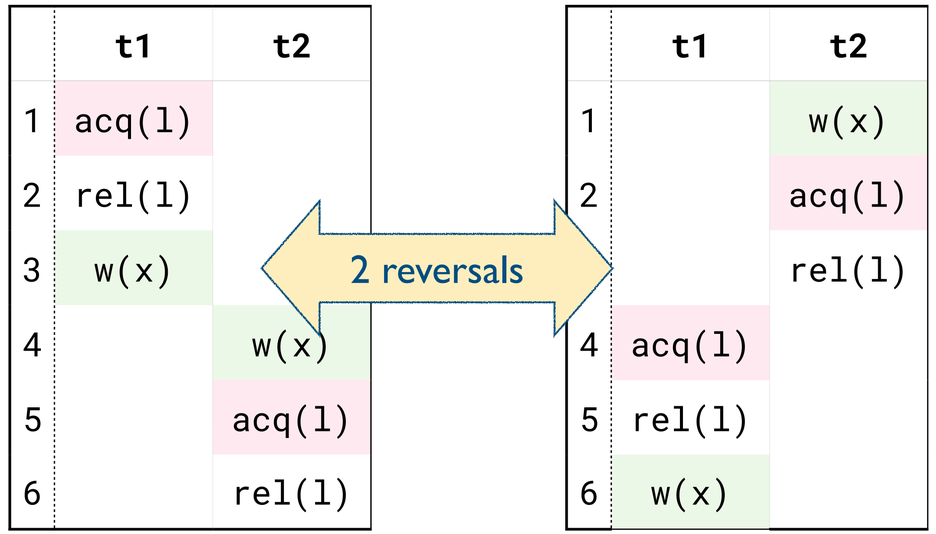
Distance-bounded Race Prediction

Distance-Bounded Data Races

Set of reversals between σ and its correct reordering ρ:

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

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



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

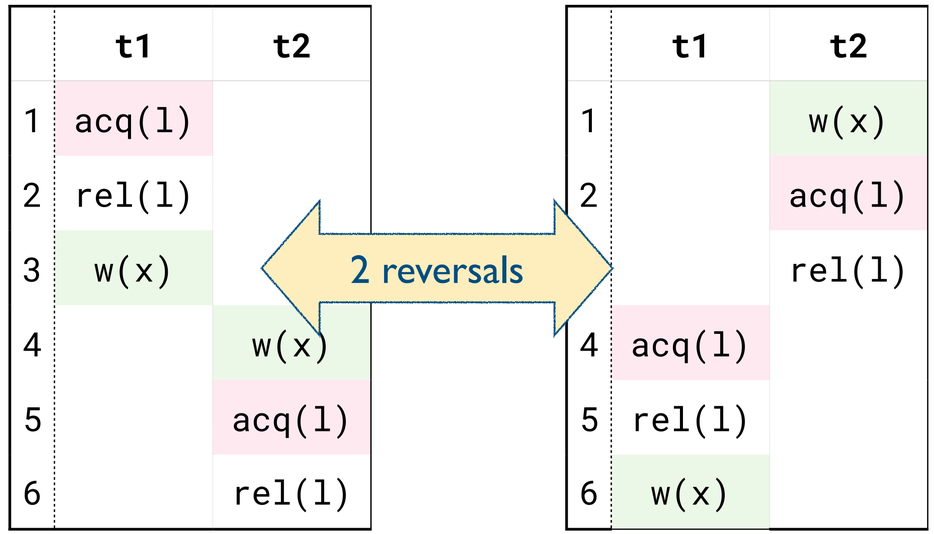
Distance-bounded Race Prediction

Distance-Bounded Data Races

Set of reversals between σ and its correct reordering ρ:

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

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



Distance-bounded Race Prediction

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

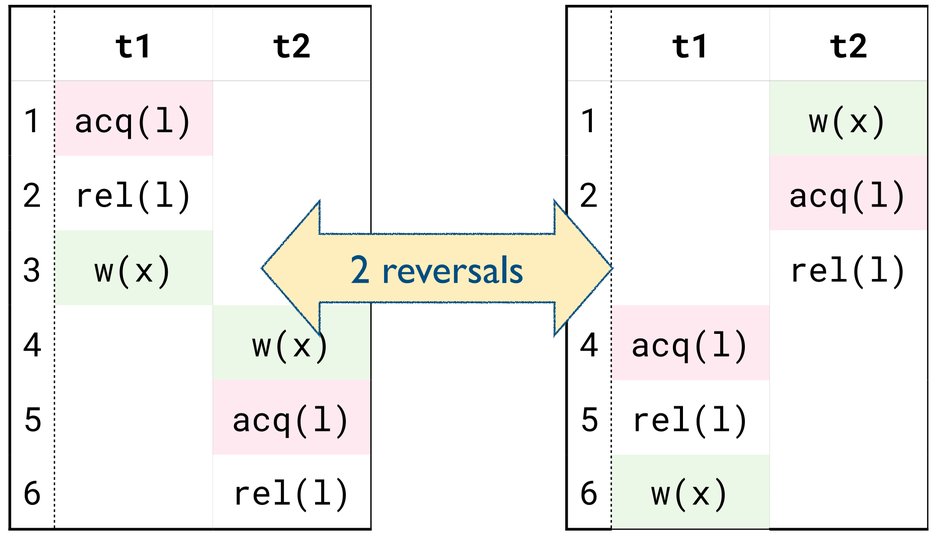
- NO if (e_1, e_2) is not a predictable race of σ

Distance-Bounded Data Races

Set of reversals between **σ** and its correct reordering **ρ**:

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

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



Distance-bounded Race Prediction

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

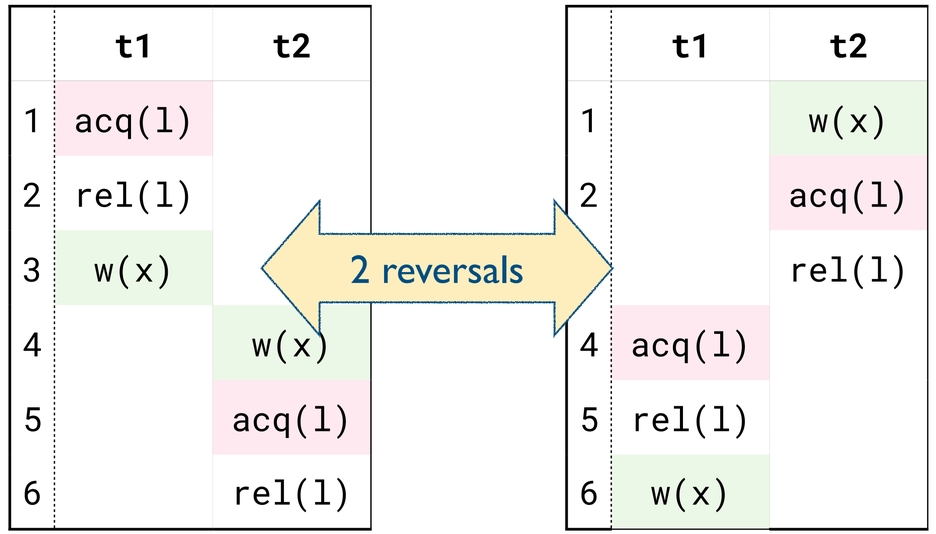
- NO if (e_1, e_2) is not a predictable race of **σ**
- YES, if there is a correct reordering **σ*** of **σ** witnessing the race (e_1, e_2) such that $\delta(\sigma, \sigma^*) \leq \ell$

Distance-Bounded Data Races

Set of reversals between σ and its correct reordering ρ:

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

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



Distance-bounded Race Prediction

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

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

Distance-Bounded Data Races

Distance-Bounded Data Races

Distance-bounded Ideal Realizability

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

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

Distance-Bounded Data Races

Distance-bounded Ideal Realizability

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

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

Naive Bound

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

$O(|I|^{2\ell}) = O(n^{2\ell})$

Distance-Bounded Data Races

Distance-Bounded Data Races

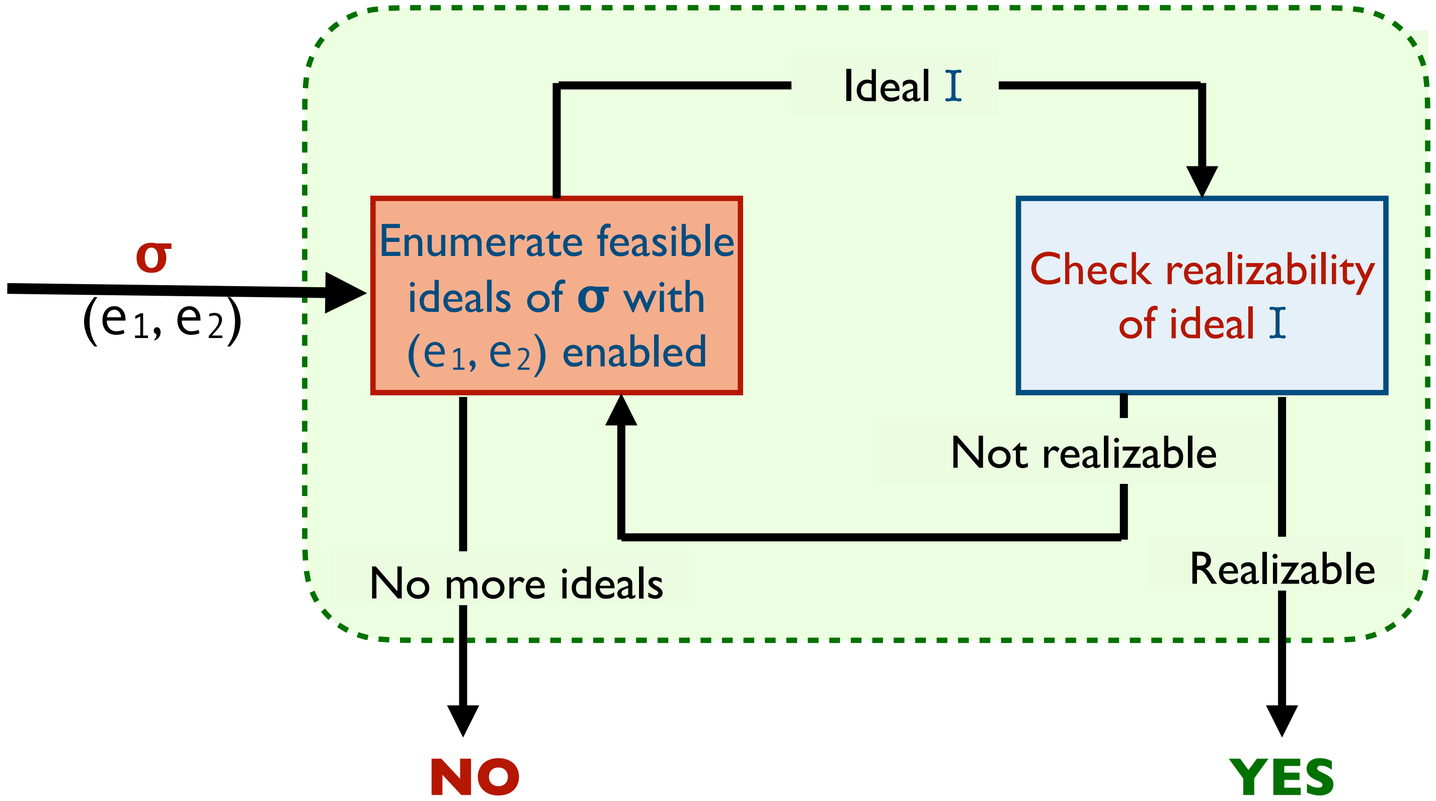
Lemma

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

Distance-Bounded Data Races

Lemma

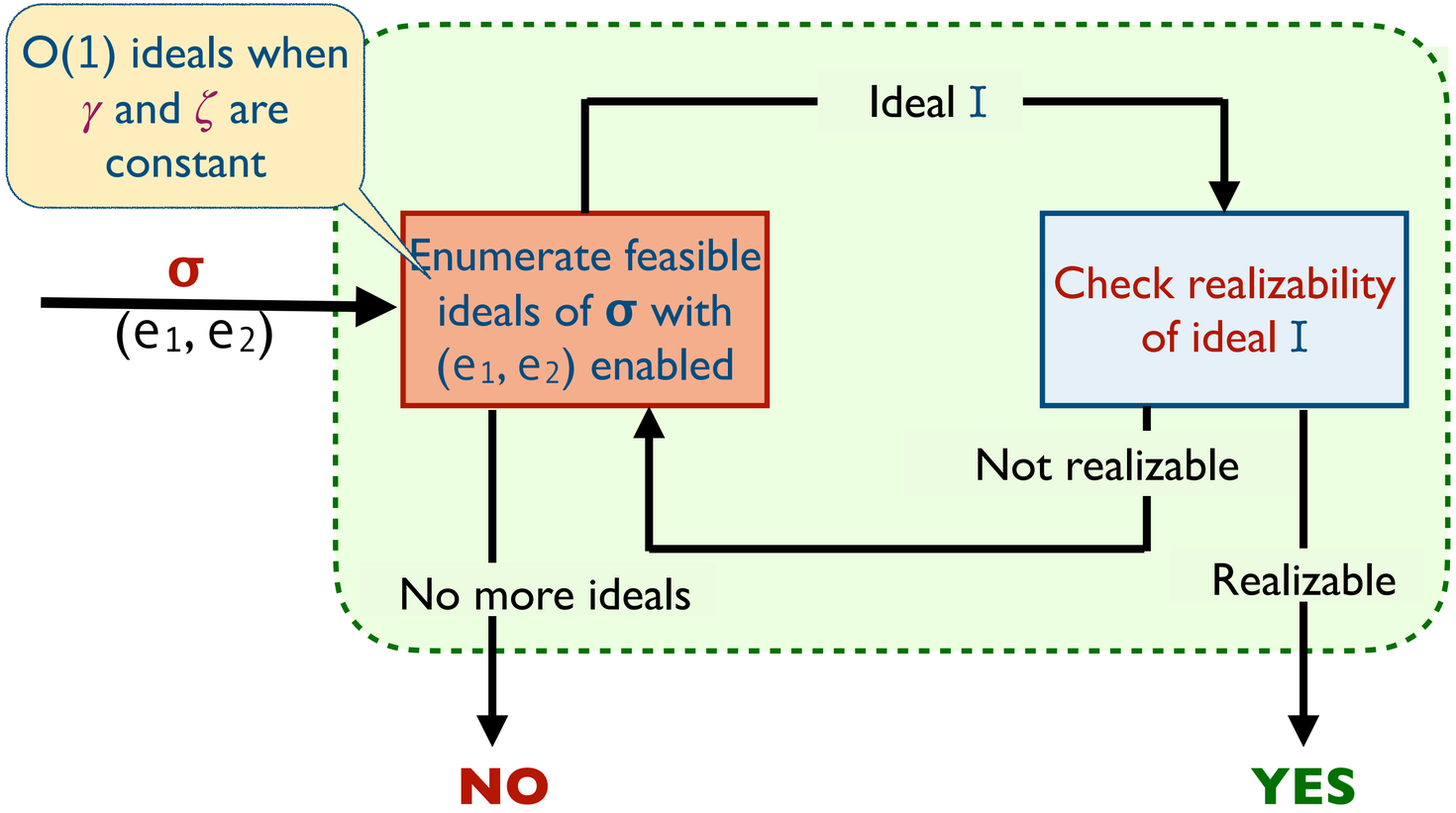
Let $\ell \in \mathbb{N}$ be a constant. Let σ be an execution with n events and k threads and let I be a feasible ideal of σ.
The ℓ-bounded realizability of I can be checked in time $O(k^{\ell+O(1)} \cdot n)$.



Distance-Bounded Data Races

Lemma

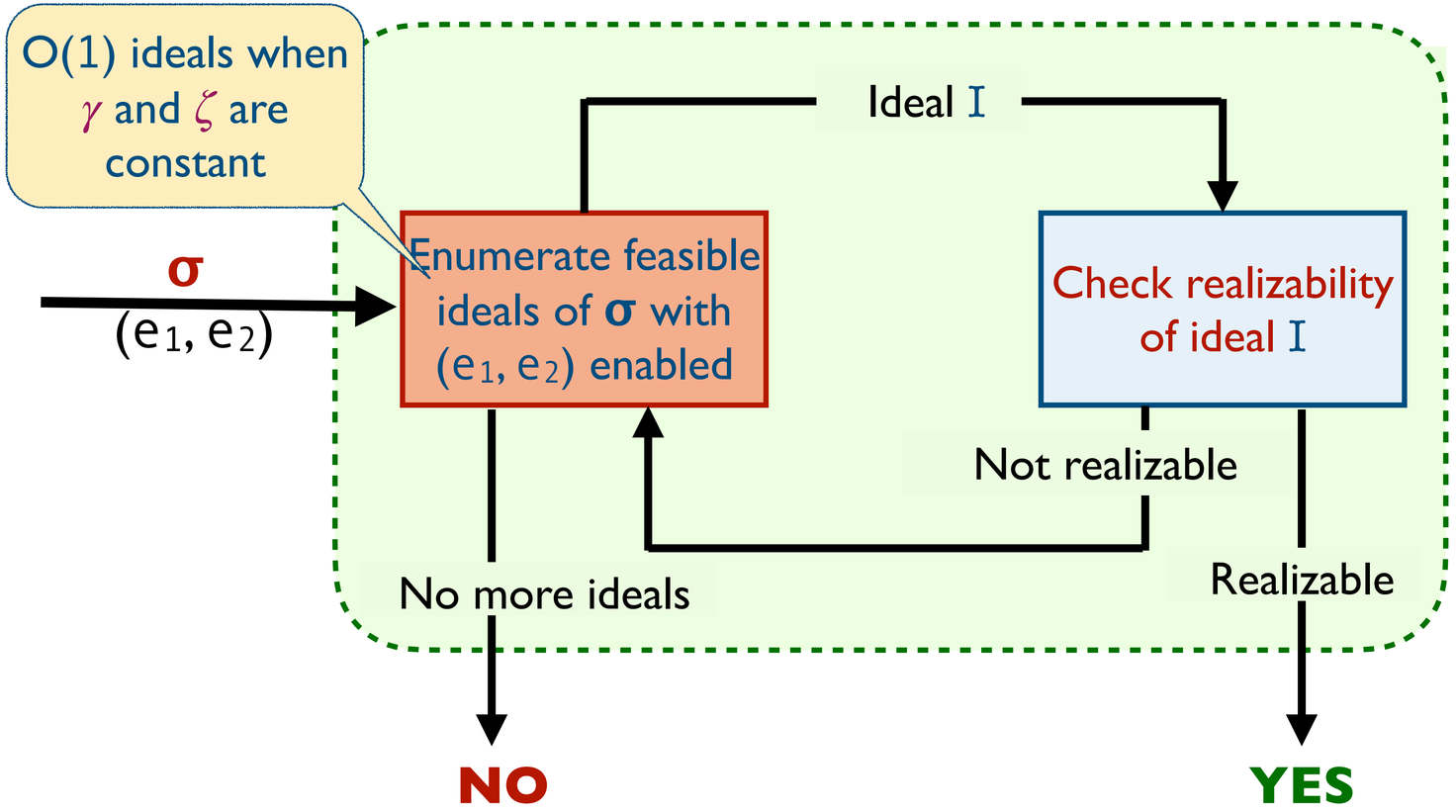
Let $\ell \in \mathbb{N}$ be a constant. Let σ be an execution with n events and k threads and let I be a feasible ideal of σ.
The ℓ-bounded realizability of I can be checked in time $O(k^{\ell+O(1)} \cdot n)$.



Distance-Bounded Data Races

Lemma

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



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

Outline

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

Lower Bound for Data Race Prediction

Lower Bound for Data Race Prediction

- When k (number of threads) is constant, data race prediction is in $\mathbf{P}$.

Lower Bound for Data Race Prediction

- When k (number of threads) is constant, data race prediction is in **P**.
- Is the problem in **P** even otherwise?

Lower Bound for Data Race Prediction

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

Lower Bound for Data Race Prediction

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

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

Lower Bound for Data Race Prediction

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

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

Theorem

Data Race Prediction and ideal realizability are **W[1]**-hard in the parameter k (number of threads).

Lower Bound for Data Race Prediction

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

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

Theorem

Data Race Prediction and ideal realizability are **W[1]**-hard in the parameter k (number of threads).

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

Lower Bound for Data Race Prediction

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

Unlikely, unless ETH fails

ETH implies **FPT** ⊊ **W[1]**

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

Theorem

Data Race Prediction and ideal realizability are **W[1]**-hard in the parameter k (number of threads).

- **FPT**-reduction from INDEPENDENT-SET(c)
- Reduction takes time O(poly(n+k)) time
- **NP**-hardness follows
- Improves previous result [Gibbons and Korach, '97]

Outline

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

Lower Bound for 2 Threads

Lower Bound for 2 Threads

Theorem

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

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

Lower Bound for 2 Threads

Theorem

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

There is no algorithm that solves the data race prediction problem for in time

$O(n^{2-\epsilon})$, for any $\epsilon>0$, unless the Orthogonal Vectors conjecture fails.

Same lower bound for ideal realizability

Lower Bound for 2 Threads

Theorem

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

There is no algorithm that solves the data race prediction problem for in time

$O(n^{2-\epsilon})$, for any $\epsilon>0$, unless the Orthogonal Vectors conjecture fails.

Same lower bound for ideal realizability

SETH implies Orthogonal Vectors Conjecture

Lower Bound for 2 Threads

Theorem

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

There is no algorithm that solves the data race prediction problem for in time

$O(n^{2-\epsilon})$, for any $\epsilon>0$, unless the Orthogonal Vectors conjecture fails.

Same lower bound for ideal realizability

SETH implies Orthogonal Vectors Conjecture

Applies to acyclic topology

Outline

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

Conclusions

Conclusions

Summary

Conclusions

Summary

- Dynamic Race Prediction
  - Complexity-theoretic investigation

Conclusions

Summary

- Dynamic Race Prediction
  - Complexity-theoretic investigation
- Trace Ideals and Realizability

Conclusions

Summary

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

Conclusions

Summary

- Dynamic Race Prediction
  - Complexity-theoretic investigation
- Trace Ideals and Realizability
- General case:
  - $O(kn^{2k-2})$ time algorithm
  - $O(k \cdot n^k)$ when γ and ζ are constant
  - **W[1]**-hardness
  - Quadratic lower bound for $k=2$ threads
- Acyclic topology
  - Quadratic time upper and lower bounds

Conclusions

Summary

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

Conclusions

Summary

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

Future Work

Conclusions

Summary

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

Future Work

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

Conclusions

Summary

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

Future Work

- Tighter bounds
  - Is data race prediction **W[k]**-complete for some k?
  - Space-time tradeoffs
- Other algorithmic paradigms
  - Randomized algorithms
  - Parallel algorithms

Conclusions

Summary

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

Future Work

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

Thank You!