Verification of Distributed Data Structures

Causal Consistency

Ahmed Bouajjani, Constantin Enea, Rachid Guerraoui, Jad Hamza

INRIA EPFL
LIAFA

13 January 2016

Post-doc

Starting post-doc at INRIA EPFL with

- Alain Girault (INRIA Grenoble)
- Gregor Goessler (INRIA Grenoble)
- Rachid Guerraoui (EPFL)
- Barbara Jobstmann (EPFL)
- Viktor Kuncak (EPFL)

Fault Tolerance

- **Fault** tolerance
- Low **latency**

$\Rightarrow$ **Replicated objects**

Fault Tolerance

- **Fault** tolerance
- Low **latency**

$\Rightarrow$ **Replicated objects**

write(2)

write(1)

Fault Tolerance

- **Fault** tolerance
- Low **latency**

$\Rightarrow$ **Replicated objects**

Fault Tolerance

- **Fault** tolerance
- Low **latency**

$\Rightarrow$ **Replicated objects**

Which **consistency criteria**?

Weak consistency

Cannot ensure **linearizability** with partition tolerance and low latency[1]

[1] Brewer's Conjecture (CAP Theorem). Gilbert, Lynch.

Weak consistency

Cannot ensure **linearizability** with partition tolerance and low latency[1]

⇒ **Weak** criteria (Google Datastore, Amazon Dynamo, . . .)

- **Eventual/RYW/FIFO/Causal** consistency

[1] Brewer's Conjecture (CAP Theorem). Gilbert, Lynch.

Weak consistency

Cannot ensure **linearizability** with partition tolerance and low latency[1]

⇒ **Weak** criteria (Google Datastore, Amazon Dynamo, . . .)

- **Eventual/RYW/FIFO/Causal** consistency
- Depending on the needs of the **client**

[1] Brewer's Conjecture (CAP Theorem). Gilbert, Lynch.

Difficulties for implementation

- Implement **highly-available** objects
- **Synchronization** between **multiple components**
- **Faults** and **network failures**

Difficulties for implementation

- Implement **highly-available** objects
- **Synchronization** between **multiple components**
- **Faults** and **network failures**

$\Rightarrow$ Need for **verification** techniques

1. Explore the **decidability** limits of **causal consistency**
2. Reduce to more standard **model checking** problems

Problem

A **replicated object** implementation $\mathcal{I}$:

- several **sites** communicating with messages
- takes **requests** from clients (`write`, `read`)
- gives **return values**

Problem

A **replicated object** implementation $\mathcal{I}$:

- several **sites** communicating with messages
- takes **requests** from clients (`write`, `read`)
- gives **return values**

Problem (Causal Consistency)

Input: A **replicated object** $\mathcal{I}$ *and a* **specification** S

Problem

A **replicated object** implementation $\mathcal{I}$:

- several **sites** communicating with messages
- takes **requests** from clients (`write`, `read`)
- gives **return values**

Problem (Causal Consistency)

Input: A **replicated object** $\mathcal{I}$ *and a* **specification** S
Output: Are **all** *the* **executions** *of* $\mathcal{I}$ **causally consistent** *wrt* S*?*

Outline

- Definition of **causal consistency**

Outline

- Definition of **causal consistency**
- **Undecidability** of causal consistency
  - Known: Linearizability and Eventual Consistency decidable
  - the number of **sites** is 2
  - the data structure **specification** is finite state
  - the communication **channels** are bounded

Outline

- Definition of **causal consistency**
- **Undecidability** of causal consistency
  - Known: Linearizability and Eventual Consistency decidable
  - the number of **sites** is 2
  - the data structure **specification** is finite state
  - the communication **channels** are bounded
- **Decidability** for the **Key-Value Store**

Outline

- Definition of **causal consistency**
- **Undecidability** of causal consistency
  - Known: Linearizability and Eventual Consistency decidable
  - the number of **sites** is 2
  - the data structure **specification** is finite state
  - the communication **channels** are bounded
- **Decidability** for the **Key-Value Store**
  - **Bad patterns**: operations ordered in a particular way
  - Characterize **non causal consistency** using bad patterns

Outline

- Definition of **causal consistency**
- **Undecidability** of causal consistency
  - Known: Linearizability and Eventual Consistency decidable
  - the number of **sites** is 2
  - the data structure **specification** is finite state
  - the communication **channels** are bounded
- **Decidability** for the **Key-Value Store**
  - **Bad patterns**: operations ordered in a particular way
  - Characterize **non causal consistency** using bad patterns
  - Recognize bad patterns with **control-state reachability**

Outline

- Definition of **causal consistency**
- **Undecidability** of causal consistency
  - Known: Linearizability and Eventual Consistency decidable
  - the number of **sites** is 2
  - the data structure **specification** is finite state
  - the communication **channels** are bounded
- **Decidability** for the **Key-Value Store**
  - **Bad patterns**: operations ordered in a particular way
  - Characterize **non causal consistency** using bad patterns
  - Recognize bad patterns with **control-state reachability**

Specifications (Key-Value Store Example)

Specification: a set of sequences

Specifications (Key-Value Store Example)

Specification: a set of sequences

Example: Key-Value Store

- Two methods: `write` and `read`
- `write`(x, v) **writes** value v on variable x
- `read`(x) **reads** the last written value on x

Specifications (Key-Value Store Example)

Specification: a set of sequences

Example: Key-Value Store

- Two methods: `write` and `read`
- `write`(x, v) **writes** value v on variable x
- `read`(x) **reads** the last written value on x

Example of **valid behavior**:
$\texttt{write}(x, 1) \cdot \texttt{write}(y, 3) \cdot \texttt{write}(x, 5) \cdot \texttt{read}(x) \blacktriangleright 5 \cdot \texttt{read}(y) \blacktriangleright 3$

Specifications (Key-Value Store Example)

Specification: a set of sequences

Example: Key-Value Store

- Two methods: `write` and `read`
- `write`(x, v) **writes** value v on variable x
- `read`(x) **reads** the last written value on x

Example of **valid behavior**:
$\texttt{write}(x, 1) \cdot \texttt{write}(y, 3) \cdot \texttt{write}(x, 5) \cdot \texttt{read}(x) \blacktriangleright 5 \cdot \texttt{read}(y) \blacktriangleright 3$

Other specifications: may be **partial orders** instead of sequences
(Multi-Value Register, CRDTs[2], ...)

[2] Conflict-free Replicated Data Types. Shapiro, Preguiça, Baquero, Zawirski

Causal Consistency (with convergence)

- **Safety** component:
  order in which operations must be **visible** and **executed**

Causal Consistency (with convergence)

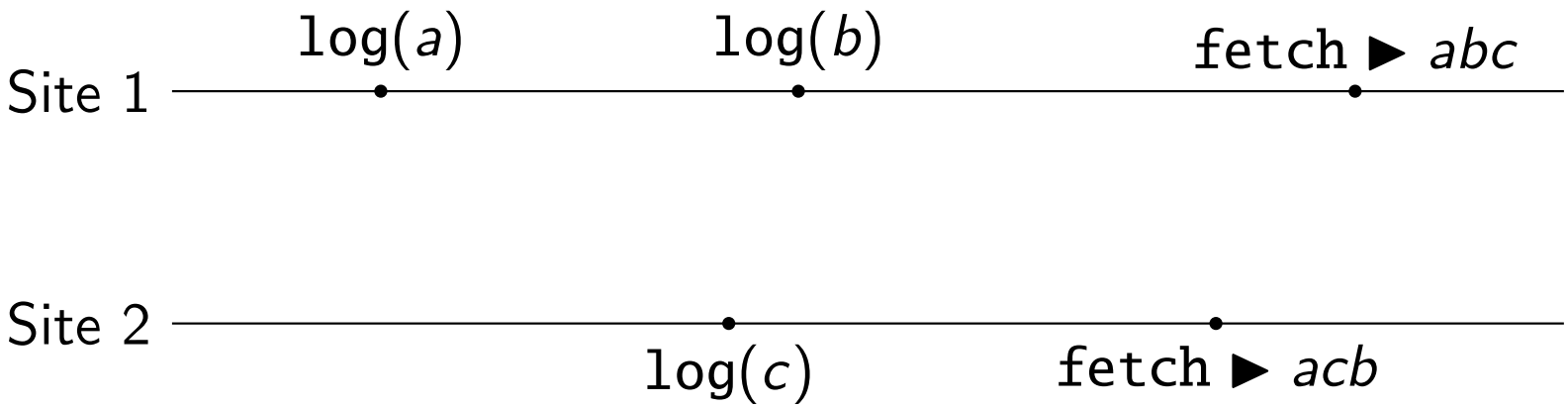
- **Safety** component:
  order in which operations must be **visible** and **executed**
- **Liveness** component:
  sites must converge towards the **same order**

Causal Consistency Safety

- `log`: **appends** an element to a list
- `fetch`: **returns** the whole list

Causal Consistency Safety

- `log`: **appends** an element to a list
- `fetch`: **returns** the whole list

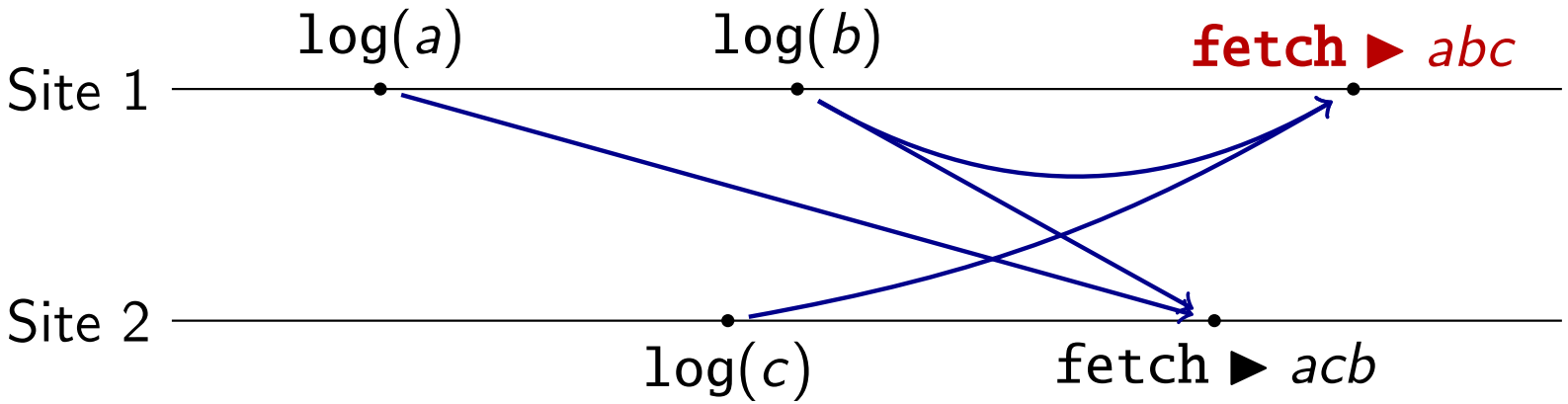


There exists a **causality order** such that, for all operations,

- there is an **interpretation order** on its **causal past**
- respecting causality and satisfying the **specification**

Causal Consistency Safety

- `log`: **appends** an element to a list
- `fetch`: **returns** the whole list

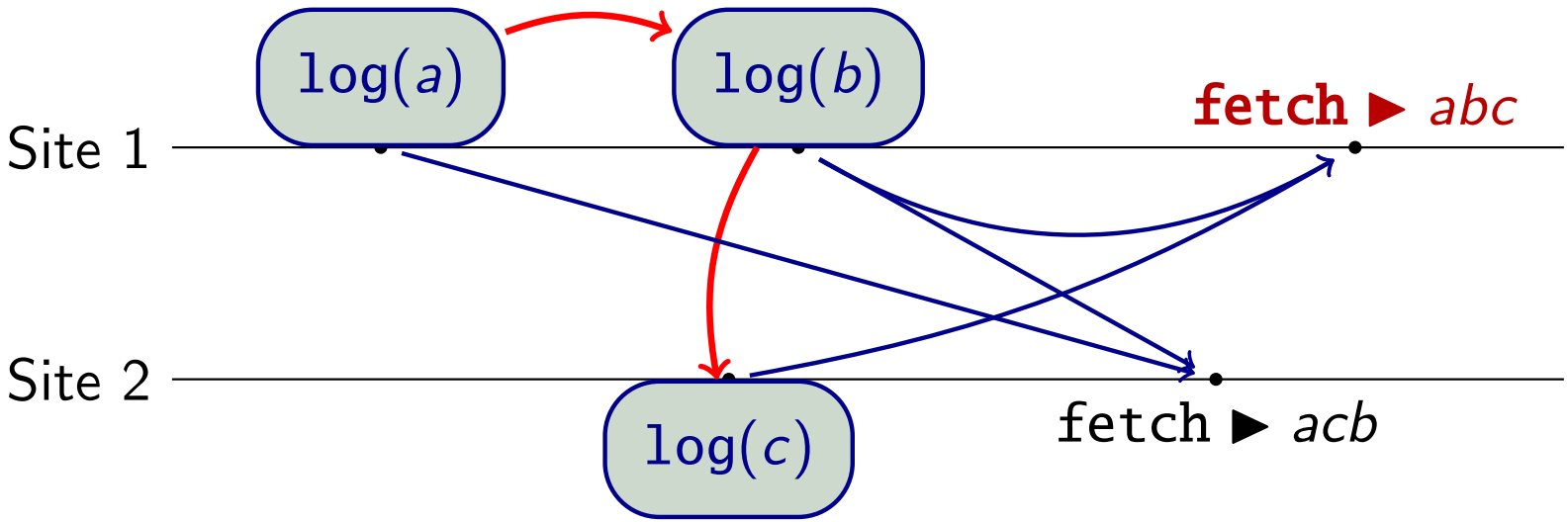


There exists a **causality order** such that, for all operations,

- there is an **interpretation order** on its **causal past**
- respecting causality and satisfying the **specification**

Causal Consistency Safety

- `log`: **appends** an element to a list
- `fetch`: **returns** the whole list

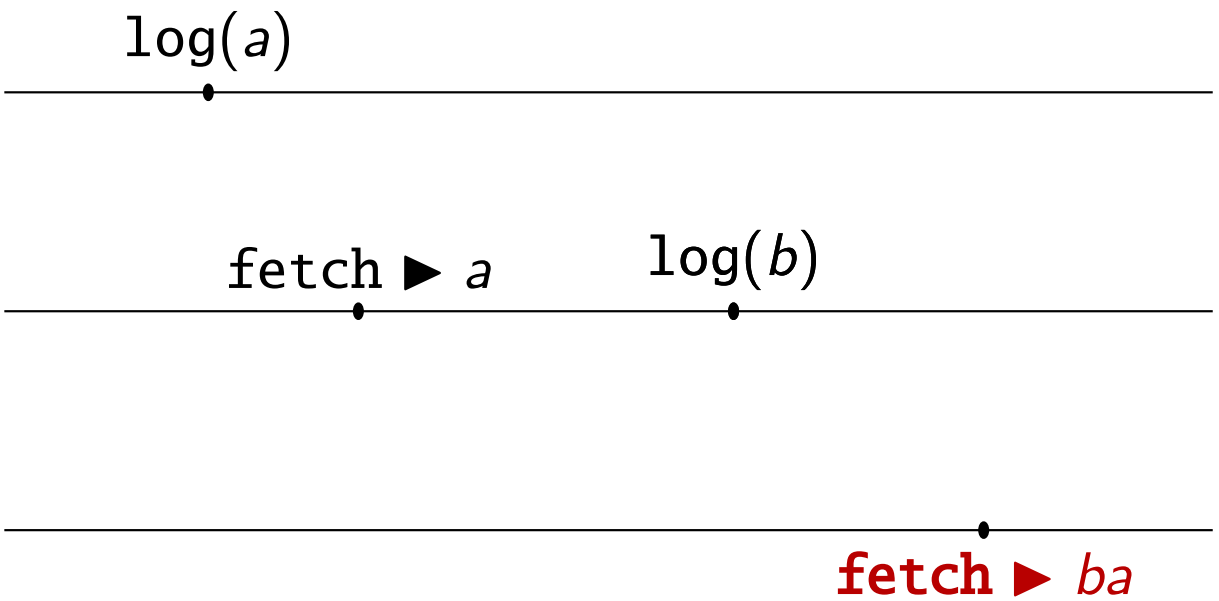


There exists a **causality order** such that, for all operations,

- there is an **interpretation order** on its **causal past**
- respecting causality and satisfying the **specification**

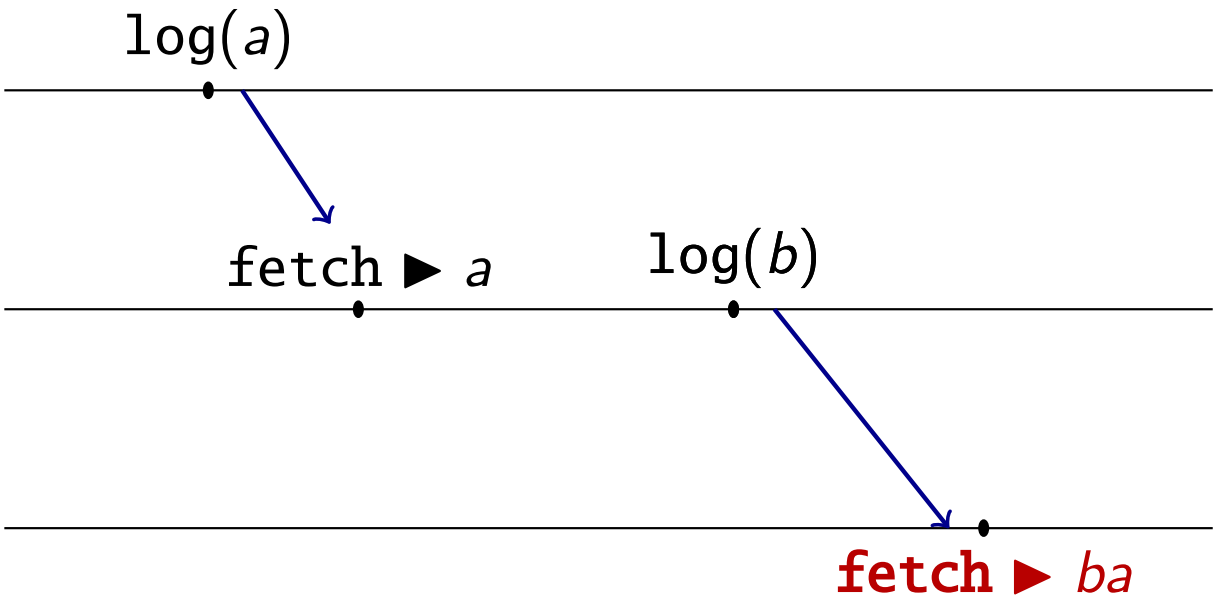
Example

Execution which is **not causally consistent**



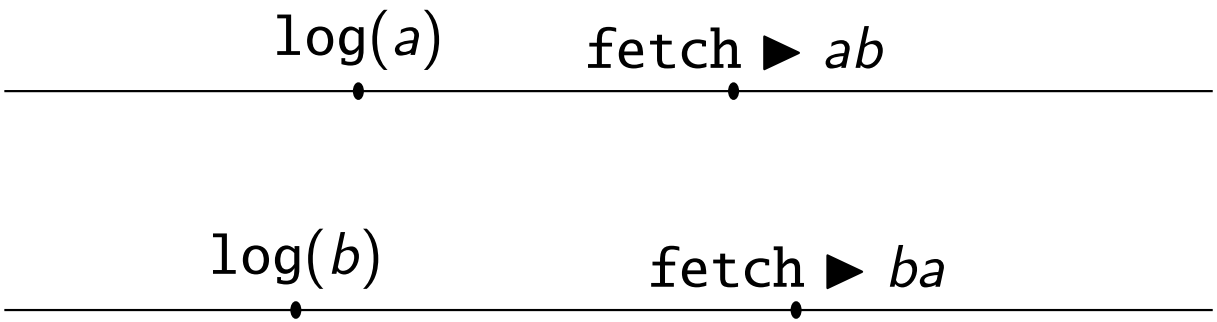
Example

Execution which is **not causally consistent**



Causal consistency and Linearizability

Execution which is **causally consistent**
but **not linearizable**

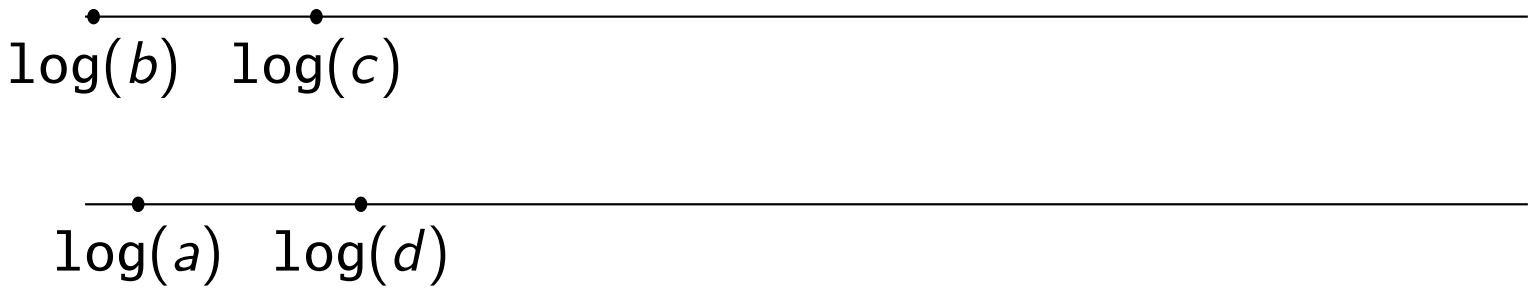


Liveness

Liveness: Ensures the **convergence** between all the sites

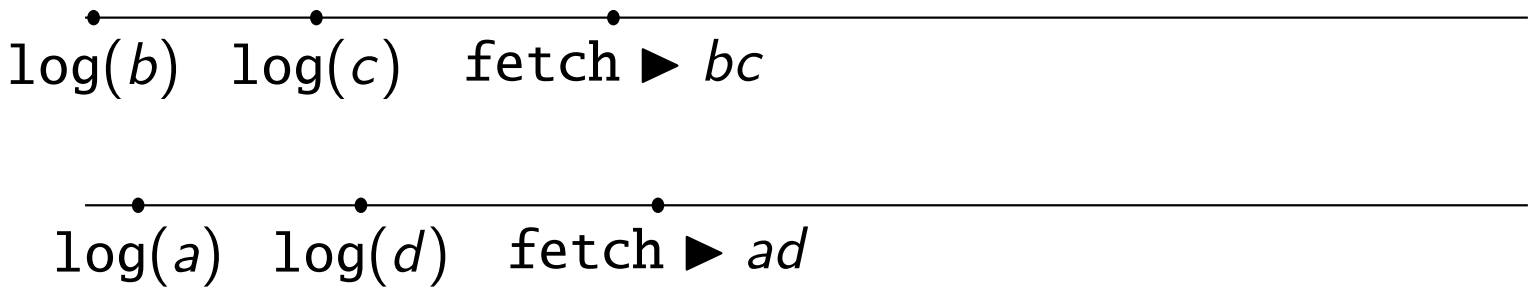
Liveness

Liveness: Ensures the **convergence** between all the sites



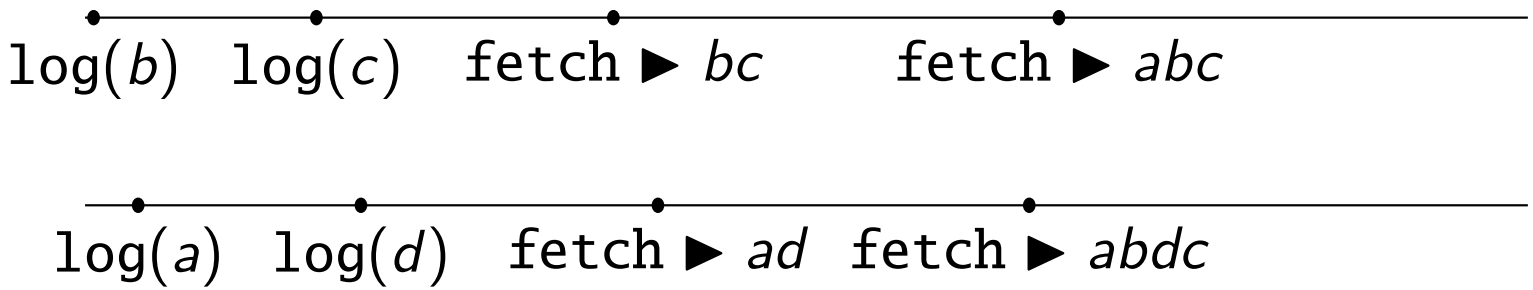
Liveness

Liveness: Ensures the **convergence** between all the sites



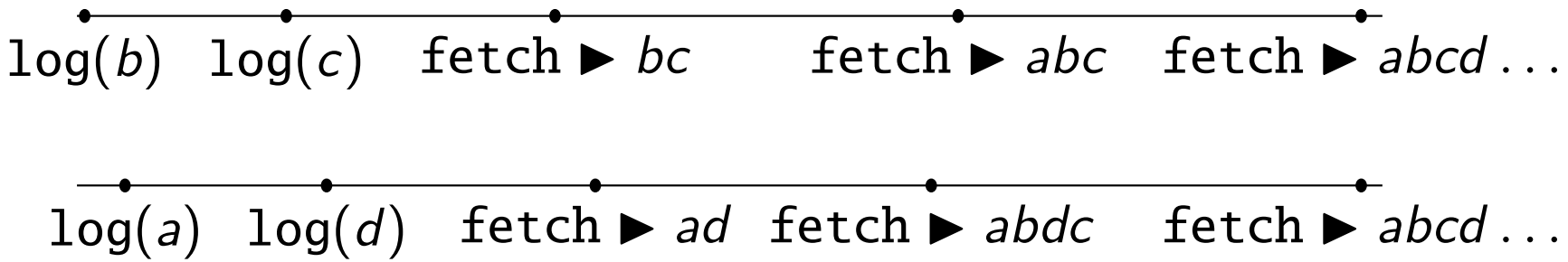
Liveness

Liveness: Ensures the **convergence** between all the sites



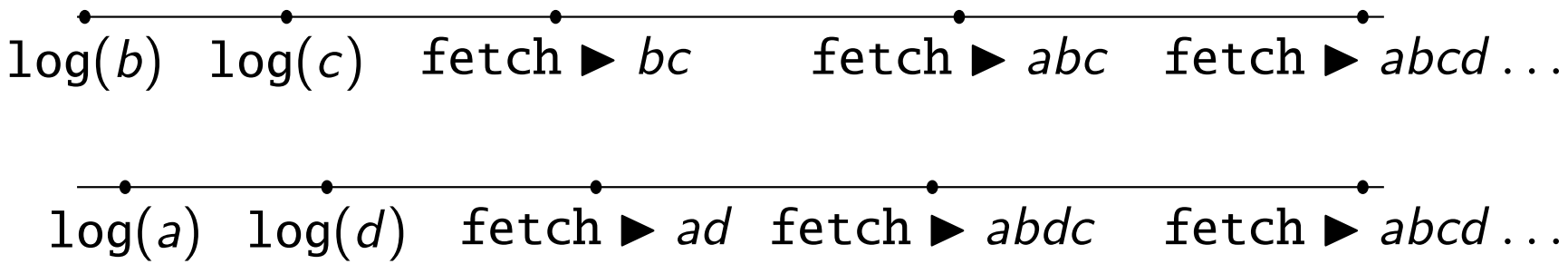
Liveness

Liveness: Ensures the **convergence** between all the sites



Liveness

Liveness: Ensures the **convergence** between all the sites



The prefix on which everyone agrees increases over time[3]

[3] Managing Update Conflicts in Bayou. Terry, Theimer, Petersen, Demers, Spreitzer, Hauser

Outline

- Definition of **causal consistency**
- **Undecidability** of causal consistency
  - Known: Linearizability and Eventual Consistency decidable
  - the number of **sites** is 2
  - the data structure **specification** is finite state
  - the communication **channels** are bounded
- **Decidability** for the **Key-Value Store**
  - **Bad patterns**: operations ordered in a particular way
  - Characterize **non causal consistency** using bad patterns
  - Recognize bad patterns with **control-state reachability**

Limits of decidability

For 2 sites, **finite-state** implementation and specification:

Theorem (Undecidability)

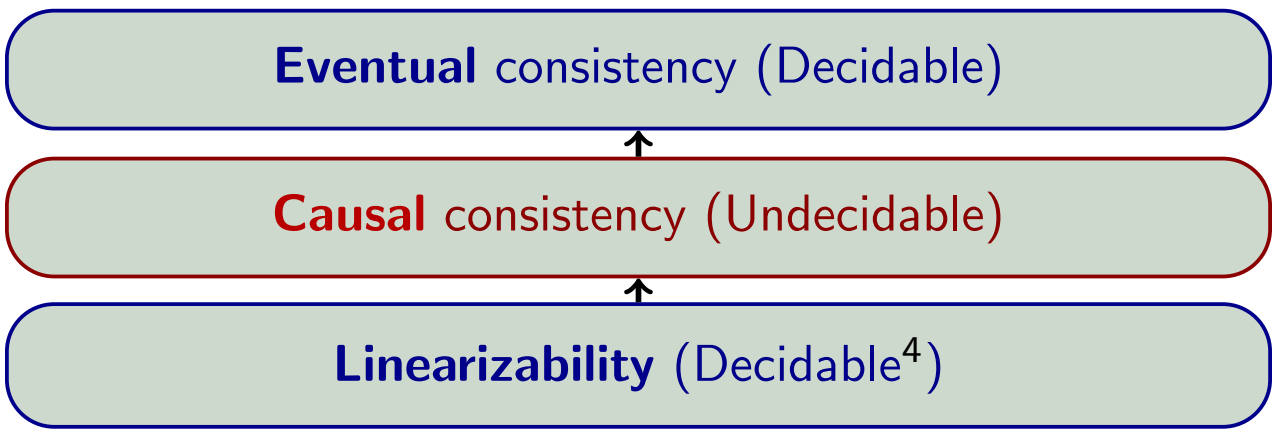
Checking whether **all executions** *of a* **replicated object** *implementation* $\mathcal{I}$ *are* **causally consistent** *is undecidable.*

Limits of decidability

For 2 sites, **finite-state** implementation and specification:

Theorem (Undecidability)

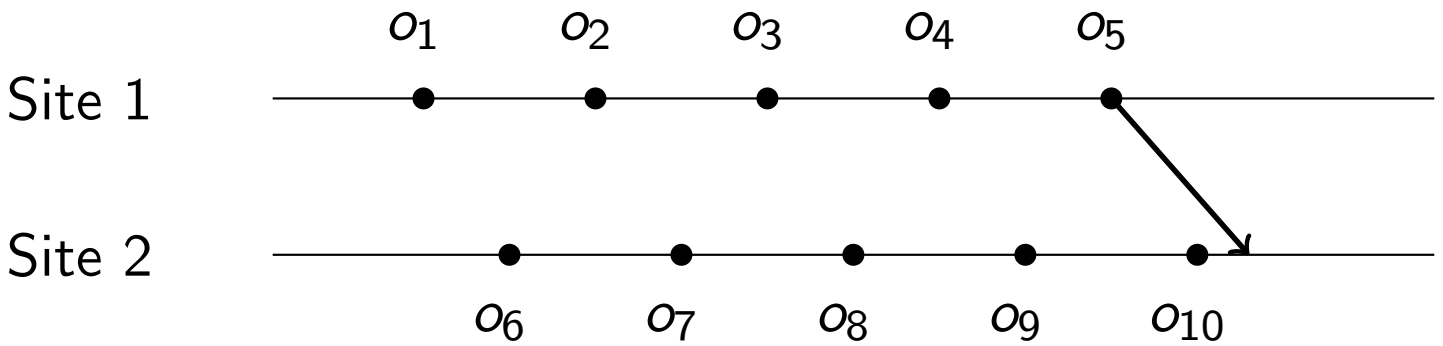
Checking whether **all executions** *of a* **replicated object** *implementation* $\mathcal{I}$ *are* **causally consistent** *is undecidable.*



[4] Model-Checking of Correctness Conditions. Alur et al. 1996

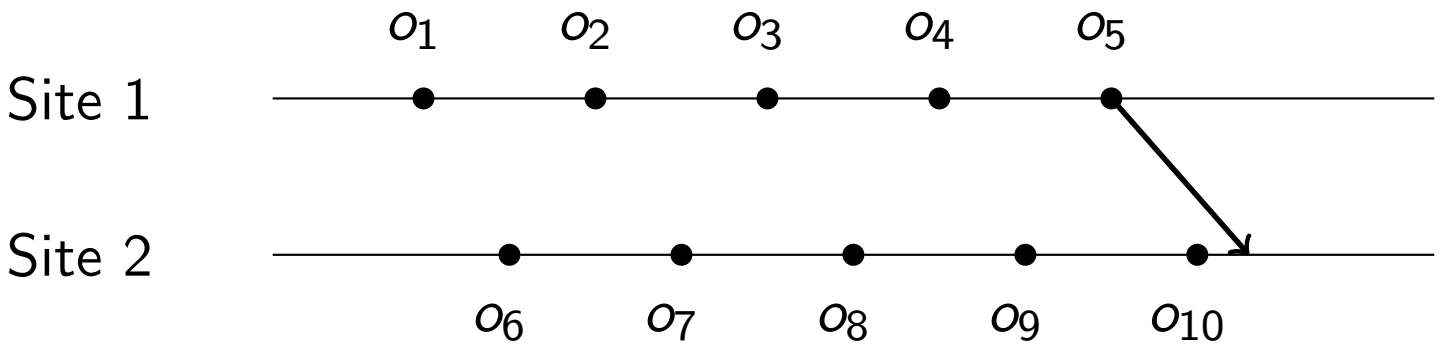
Undecidability: Intuition

Sites can be disconnected an **unbounded** amount of **time**



Undecidability: Intuition

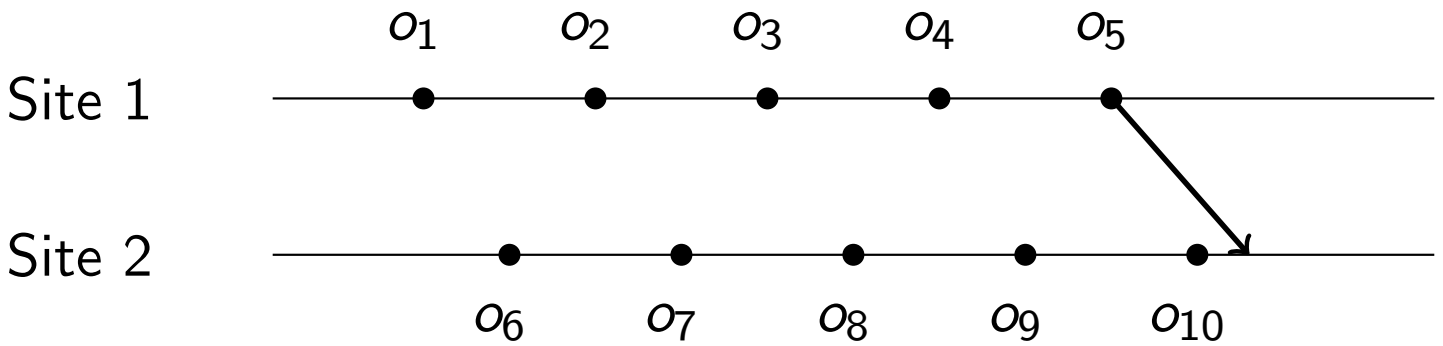
Sites can be disconnected an **unbounded** amount of **time**



- Operations can be **interleaved arbitrarily** far in the past

Undecidability: Intuition

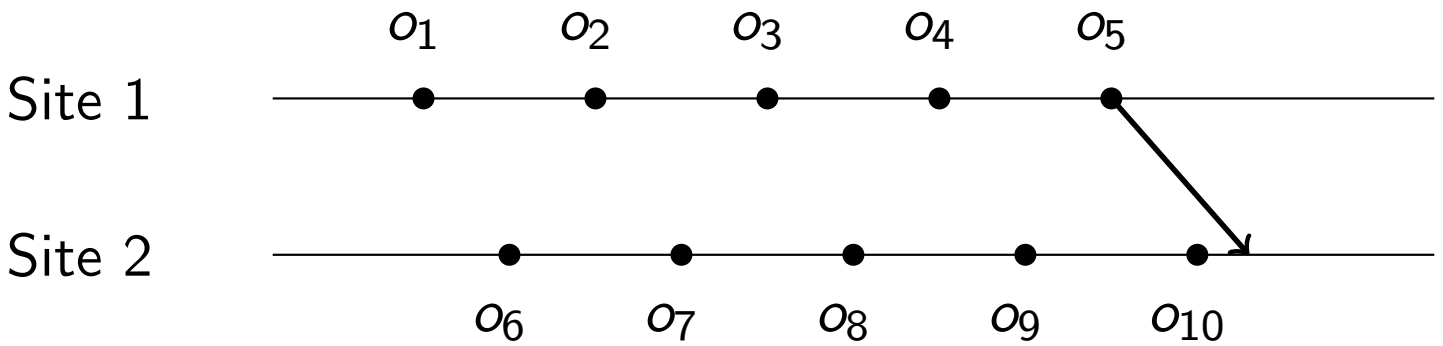
Sites can be disconnected an **unbounded** amount of **time**



- Operations can be **interleaved arbitrarily** far in the past
- Can encode the **Post Correspondence Problem**

Undecidability: Intuition

Sites can be disconnected an **unbounded** amount of **time**



- Operations can be **interleaved arbitrarily** far in the past
- Can encode the **Post Correspondence Problem**
- **Artificially** built specifications

Outline

- Definition of **causal consistency**
- **Undecidability** of causal consistency
  - Known: Linearizability and Eventual Consistency decidable
  - the number of **sites** is 2
  - the data structure **specification** is finite state
  - the communication **channels** are bounded
- **Decidability** for the **Key-Value Store**
  - **Bad patterns**: operations ordered in a particular way
  - Characterize **non causal consistency** using bad patterns
  - Recognize bad patterns with **control-state reachability**

Causal Consistency Characterization: Key-Value Store

Bad pattern: set of operations ordered is a particular way

Causal Consistency Characterization: Key-Value Store

Bad pattern: set of operations ordered is a particular way

Lemma (Characterization)

An execution is **not causally consistent** *if and only if it exhibits one out of four* **bad patterns**.

Causal Consistency Characterization: Key-Value Store

Bad pattern: set of operations ordered is a particular way

Lemma (Characterization)

An execution is **not causally consistent** *if and only if it exhibits one out of four* **bad patterns**.

- 1 bad pattern for causal consistency **safety**
- 3 bad patterns for causal consistency **liveness**

Causality

Assume unique values:

- *po* (program order): connects operations from the same site
- *rf* (reads-from relation): connects `write` to `read`
- *co* (causal order): defined as $(po \cup rf)^+$

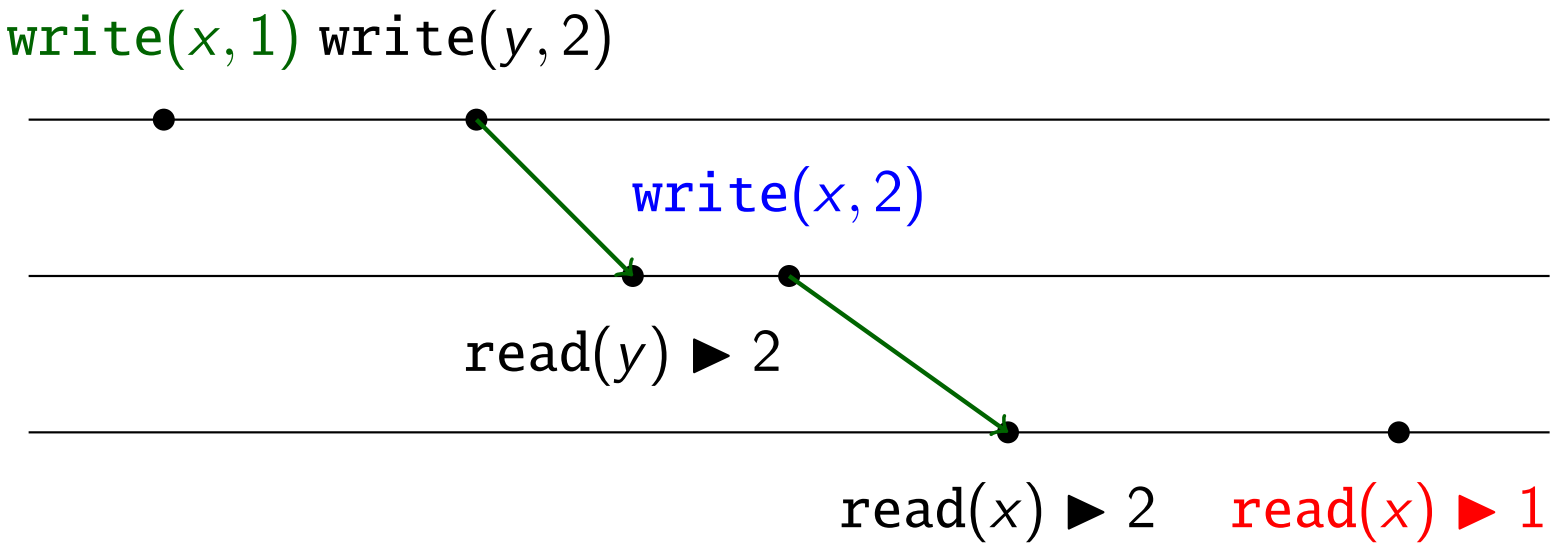
Bad Pattern for Causal Consistency Safety

- Two `write` operations w_1 and w_2 on the same variable
- One `read` operation r_1 that **reads-from** w_1
- **Causal** relations $w_1 <_{co} w_2 <_{co} r_1$

Bad Pattern for Causal Consistency Safety

- Two `write` operations w_1 and w_2 on the same variable
- One `read` operation r_1 that **reads-from** w_1
- **Causal** relations $w_1 <_{co} w_2 <_{co} r_1$

Example:



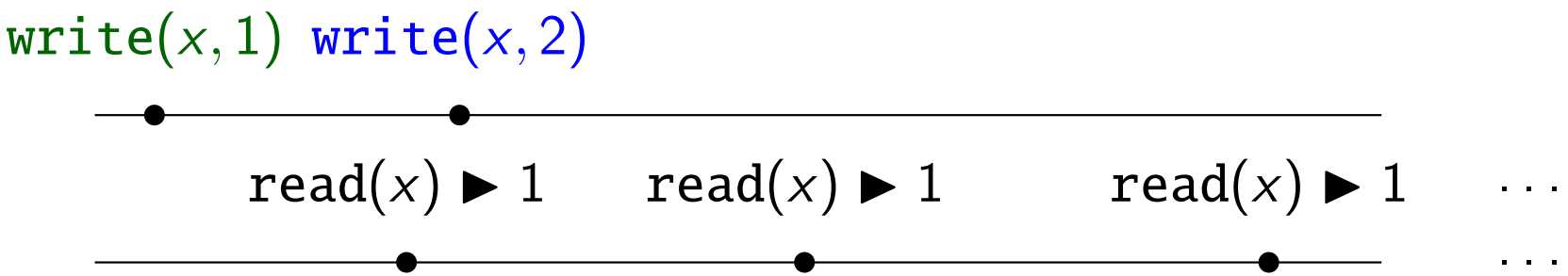
Bad Pattern 1 for Causal Consistency Liveness

- Two `write` operations w_1 and w_2 on the same variable
- An **infinite** number of `read` operations that **reads-from** w_1
- **Causal** relation $w_1 <_{co} w_2$

Bad Pattern 1 for Causal Consistency Liveness

- Two `write` operations w_1 and w_2 on the same variable
- An **infinite** number of `read` operations that **reads-from** w_1
- **Causal** relation $w_1 <_{co} w_2$

Example:



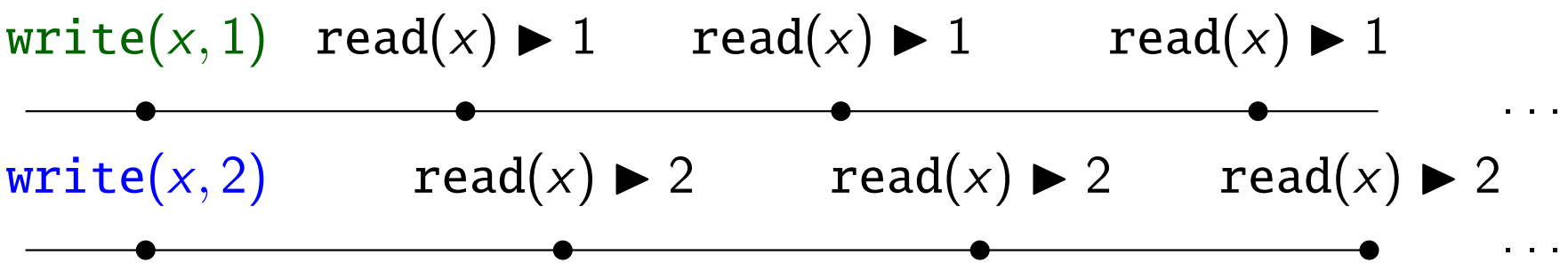
Bad Pattern 2 for Causal Consistency Liveness

- Two `write` operations w_1 and w_2 on the same variable
- An **infinite** number of `read` operations that **reads-from** w_1
- An **infinite** number of `read` operations that **reads-from** w_2

Bad Pattern 2 for Causal Consistency Liveness

- Two `write` operations w_1 and w_2 on the same variable
- An **infinite** number of `read` operations that **reads-from** w_1
- An **infinite** number of `read` operations that **reads-from** w_2

Example:



Decidability

Lemma (Characterization)

An execution is **not causally consistent** *if and only if it exhibits one out of four* **bad patterns**.

Decidability

Lemma (Characterization)

An execution is **not causally consistent** *if and only if it exhibits one out of four* **bad patterns**.

Theorem (Decidability)

Checking whether a **replicated object** *is* **causally consistent** *wrt* **Key-Value Store** *is* **decidable**.

Reduction to Model Checking

Reduction to (repeated) **control-state reachability** problems

Reduction to Model Checking

Reduction to (repeated) **control-state reachability** problems

$\Rightarrow$ Causal consistency (Key-Value Store) can be checked with:

- Tools for **reachability/safety/liveness**
- Manual or semi-automated proofs

Summary

- **Undecidability** of causal consistency in general
  - 2 sites
  - bounded communication channels
  - finite-state implementation
  - finite-state specification

Summary

- **Undecidability** of causal consistency in general
  - 2 sites
  - bounded communication channels
  - finite-state implementation
  - finite-state specification
- **Decidability** for the **Key-Value Store** based on
  - Characterization of non causal consistency using **bad patterns**
  - Generic **reduction** to (repeated) **reachability** problems

Future work

Basis for the INRIA EPFL post-doc

- Bad patterns for **other criteria** (FIFO consistency, . . .)
- for **other specifications** (Multi-Value Register, CRDTs, . . .)

Future work

Basis for the INRIA EPFL post-doc

- Bad patterns for **other criteria** (FIFO consistency, . . .)
- for **other specifications** (Multi-Value Register, CRDTs, . . .)
- Integrate this into **Leon** (Verification software LARA/EPFL)
- Application to existing **causally consistent systems**

Future work

Basis for the INRIA EPFL post-doc

- Bad patterns for **other criteria** (FIFO consistency, . . .)
- for **other specifications** (Multi-Value Register, CRDTs, . . .)
- Integrate this into **Leon** (Verification software LARA/EPFL)
- Application to existing **causally consistent systems**
- **Repair** and **Synthesis** of causally consistent implementations

Future work

Basis for the INRIA EPFL post-doc

- Bad patterns for **other criteria** (FIFO consistency, . . .)
- for **other specifications** (Multi-Value Register, CRDTs, . . .)
- Integrate this into **Leon** (Verification software LARA/EPFL)
- Application to existing **causally consistent systems**
- **Repair** and **Synthesis** of causally consistent implementations

Thank you

Undecidability: Post Correspondence Problem

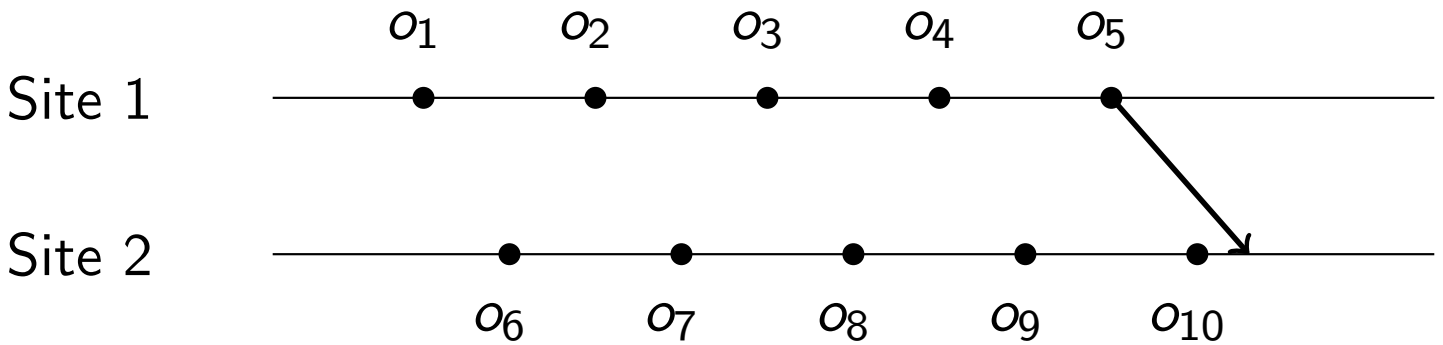
Input: a finite number of domino models $\frac{u_1}{v_1}, \frac{u_2}{v_2}, \cdots, \frac{u_n}{v_n}$
where the u_i's and v_i's are finite words.

Ouput: yes if there exists a (non-empty) sequence of dominoes where top $=$ bottom
(each domino model can be used an unbounded number of times)

Undecidability: Post Correspondence Problem

Input: a finite number of domino models $\frac{u_1}{v_1}, \frac{u_2}{v_2}, \cdots, \frac{u_n}{v_n}$
where the u_i's and v_i's are finite words.

Ouput: yes if there exists a (non-empty) sequence of dominoes where top = bottom
(each domino model can be used an unbounded number of times)



Bad Pattern 3 for Causal Consistency Liveness

- A `write` operation w_1
- An **infinite** number of `write` operations on the same variable
- An **infinite** number of `read` operations that **reads-from** w_1

Bad Pattern 3 for Causal Consistency Liveness

- A `write` operation w_1
- An **infinite** number of `write` operations on the same variable
- An **infinite** number of `read` operations that **reads-from** w_1

Example: