On Checking Correctness of Concurrent Data Structures

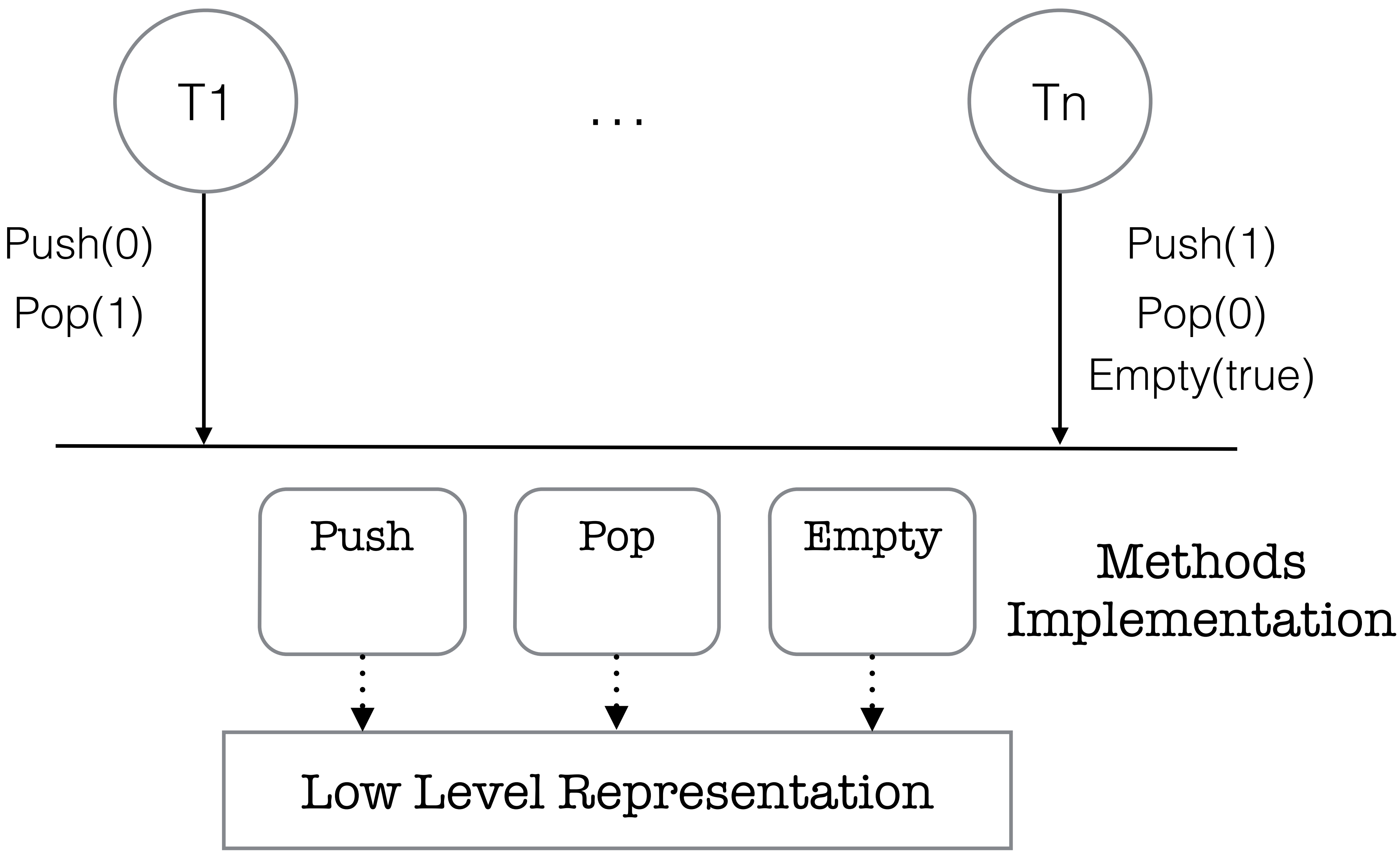
Ahmed Bouajjani
LIAFA, Univ Paris Diderot - Paris 7

Joint work with

Michael Emmi   Constantin Enea   Jad Hamza
IMDEA   LIAFA, U Paris 7

FRIDA worskhop - 23 July 2014, Vienna

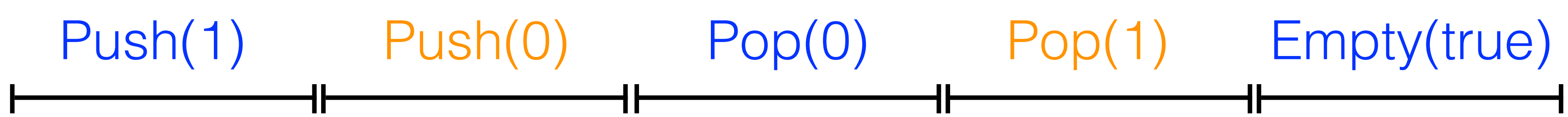
Concurrent Data Structures



Different atomicity levels

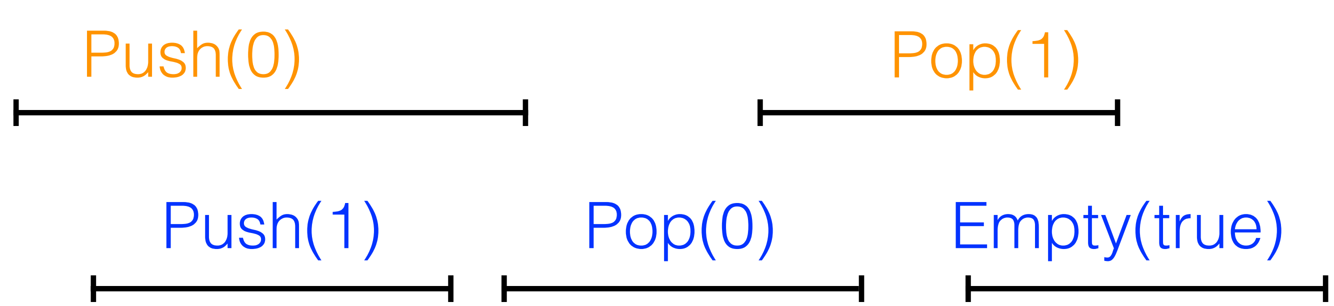
Client view:

- Operations are atomic
- Thread executions are interleaved

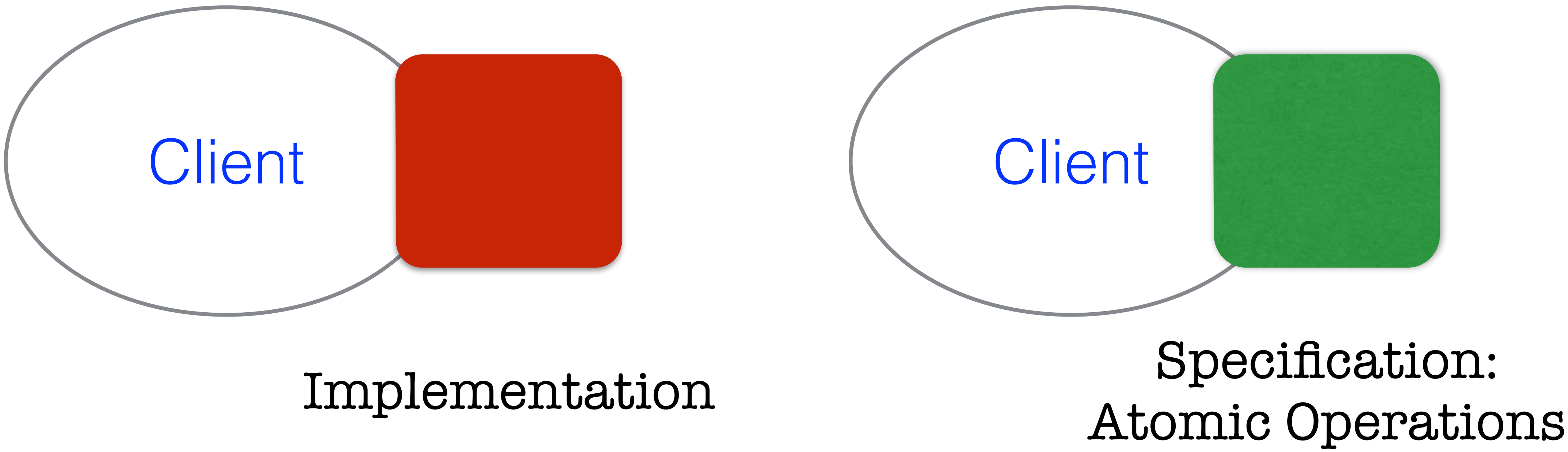


Implementation:

- Atomic operations: Coarse-grain locking
- Performances: Avoid coarse-grain locking
- => Execution intervals may overlap

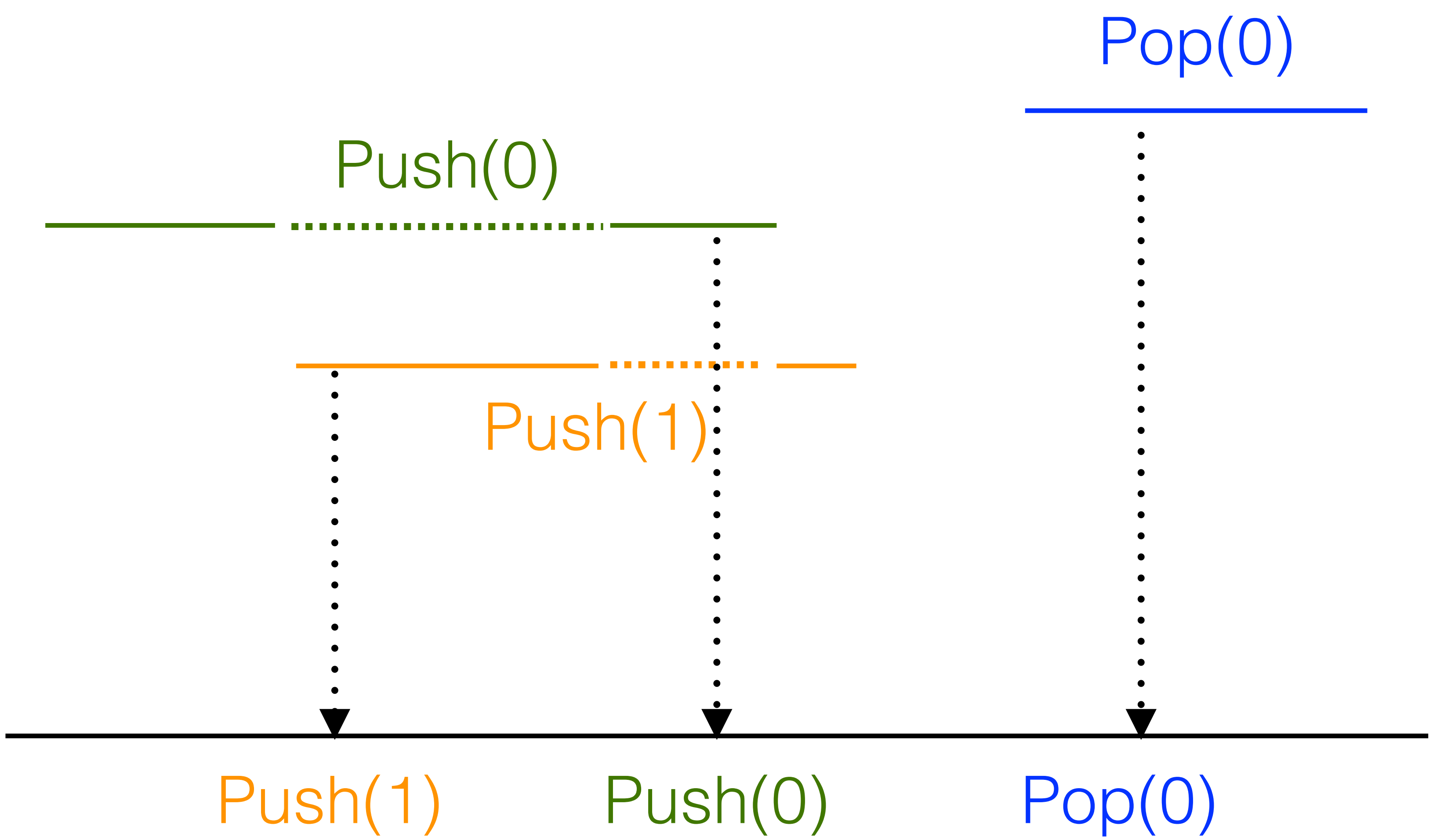


Observational Refinement



For every Client,
Client x Impl included in Client x Spec

Linearizability



Valid sequence of the specification

- history = sequence of call and return events
- reorder call/return events, while preserving returns —> calls
- find “linearization points” within execution time intervals

Linearizability implies Observational Refinement

Complexity

Finite number of threads, regular specification

- Linearizability is in EXPSPACE [Alur et al. 1996]

  - Enumeration of all possible linearizations
  - Tools for finding lineerizabiity violations (Line-up)

- Lower bound: EXSPACE-hardness [Jad Hamza '14]

Unbounded Number of Threads ?

[B, Emmi, Enea, Hamza, ESOP'13]

Finite-state threads, regular specifications

- Linearizability is undecidable

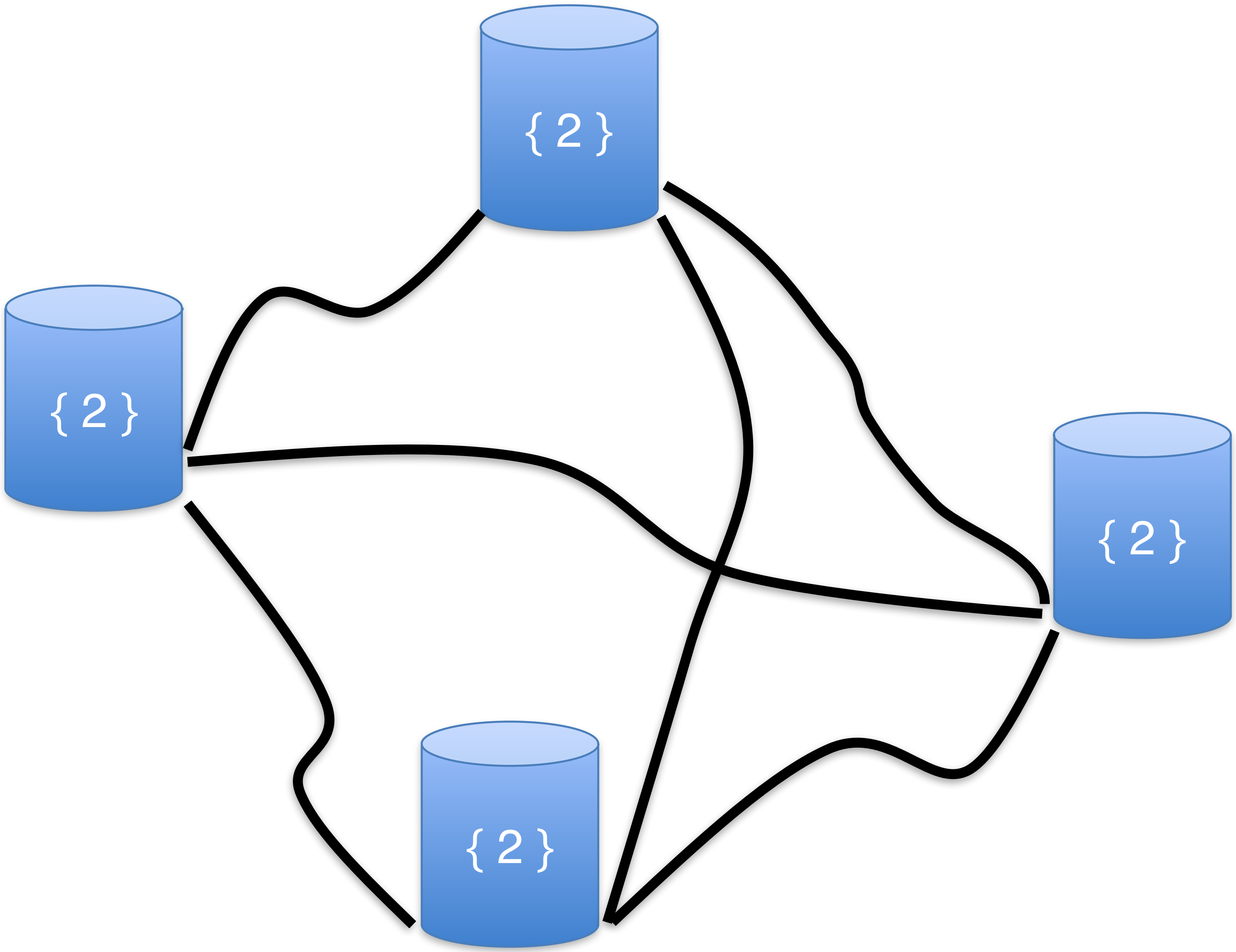
Reachability in 2-counter machines reducible to non-linearizability

- Static Linearizability:
  - Fixed linearizations points, except for read-only methods
  - Relevant for a wide class of implementations

- Static Linearizability is decidable
  - Reduction to state reachability (works for infinite-state threads)
  - Reachability in FSM/VASS for fixed/unbounded number of threads
  - P/EXP-SPACE-complete for a fixed/unbounded number of threads

Distributed, Replicated Data Structures

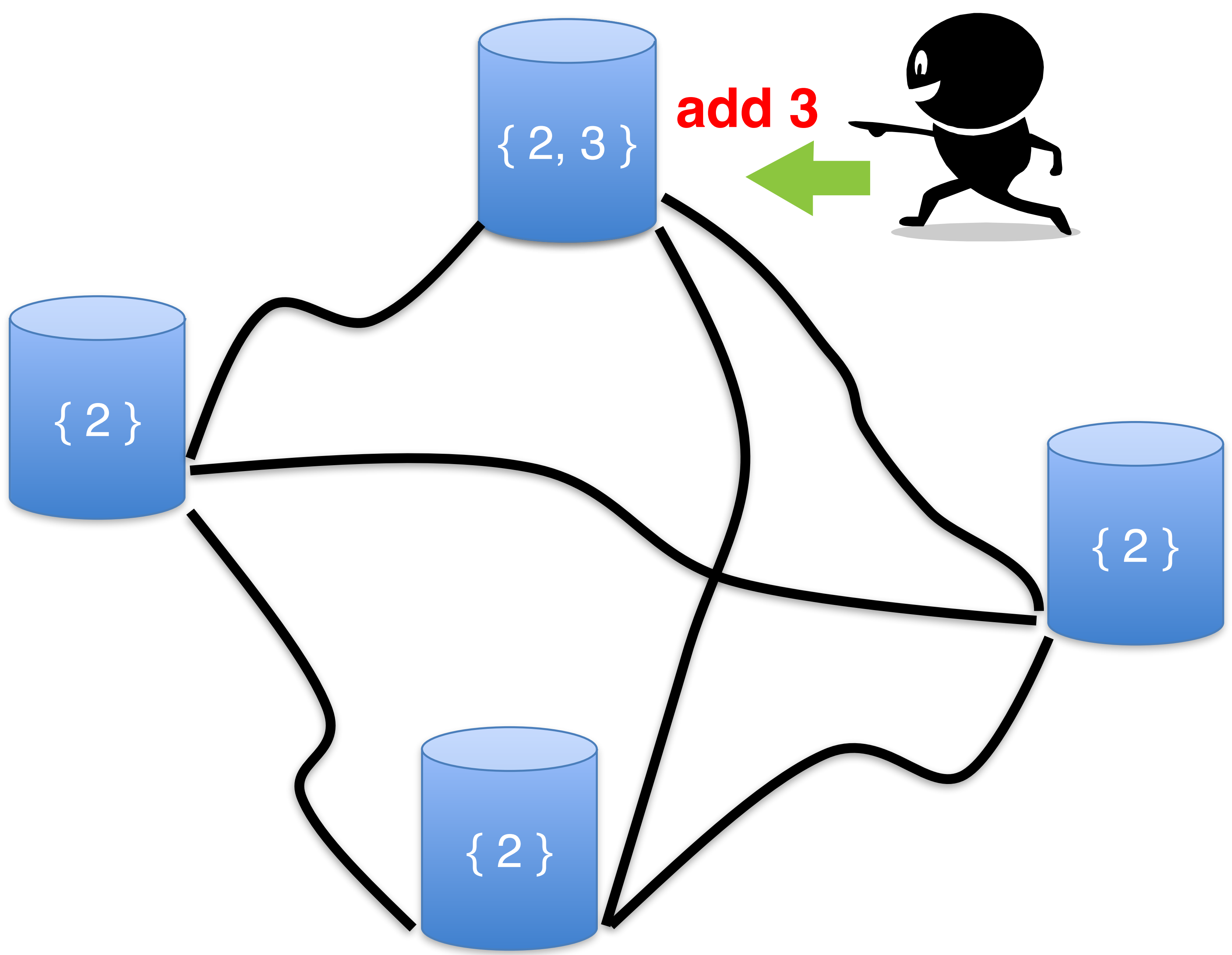
- Interactive collaborative applications (e.g., online stores)
- Replicated objects accessible at multiple sites in a network



Optimistic data replication

Optimistic replication: replicas are allowed to diverge

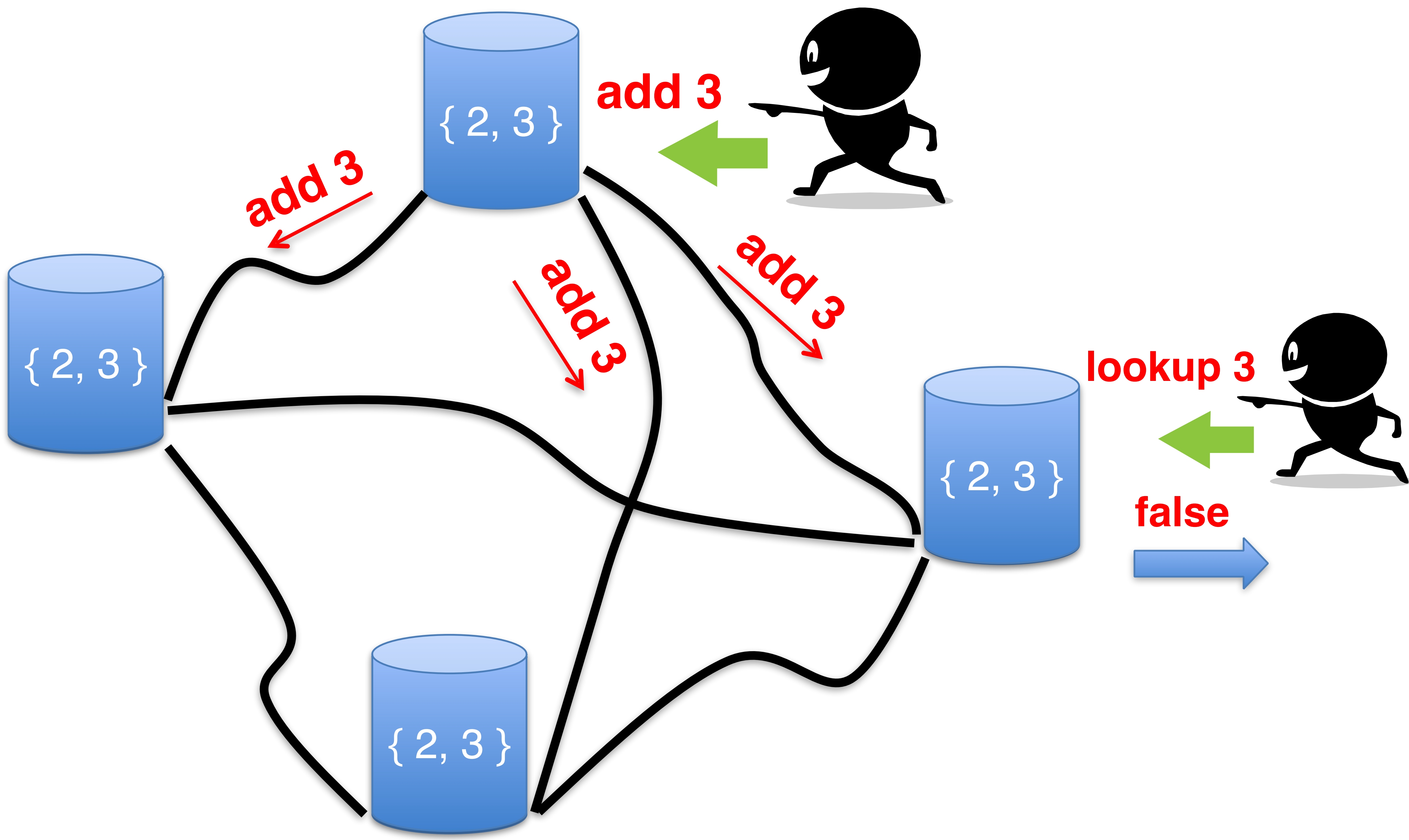
– operations are applied immediately at the submission site



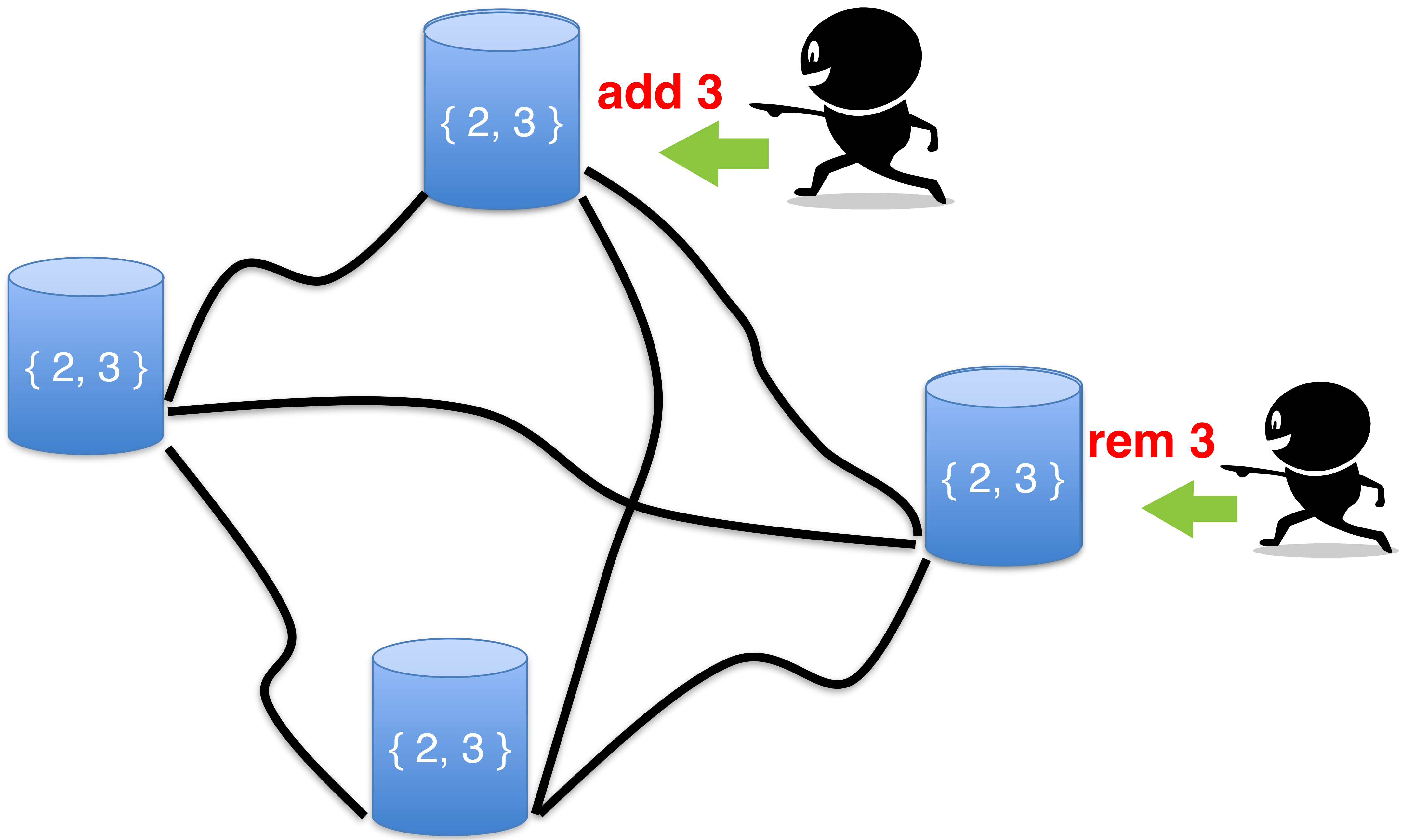
Optimistic data replication

Optimistic replication: replicas are allowed to diverge

- operations are applied immediately at the submission site
- in the background, sites exchange and apply remote operations



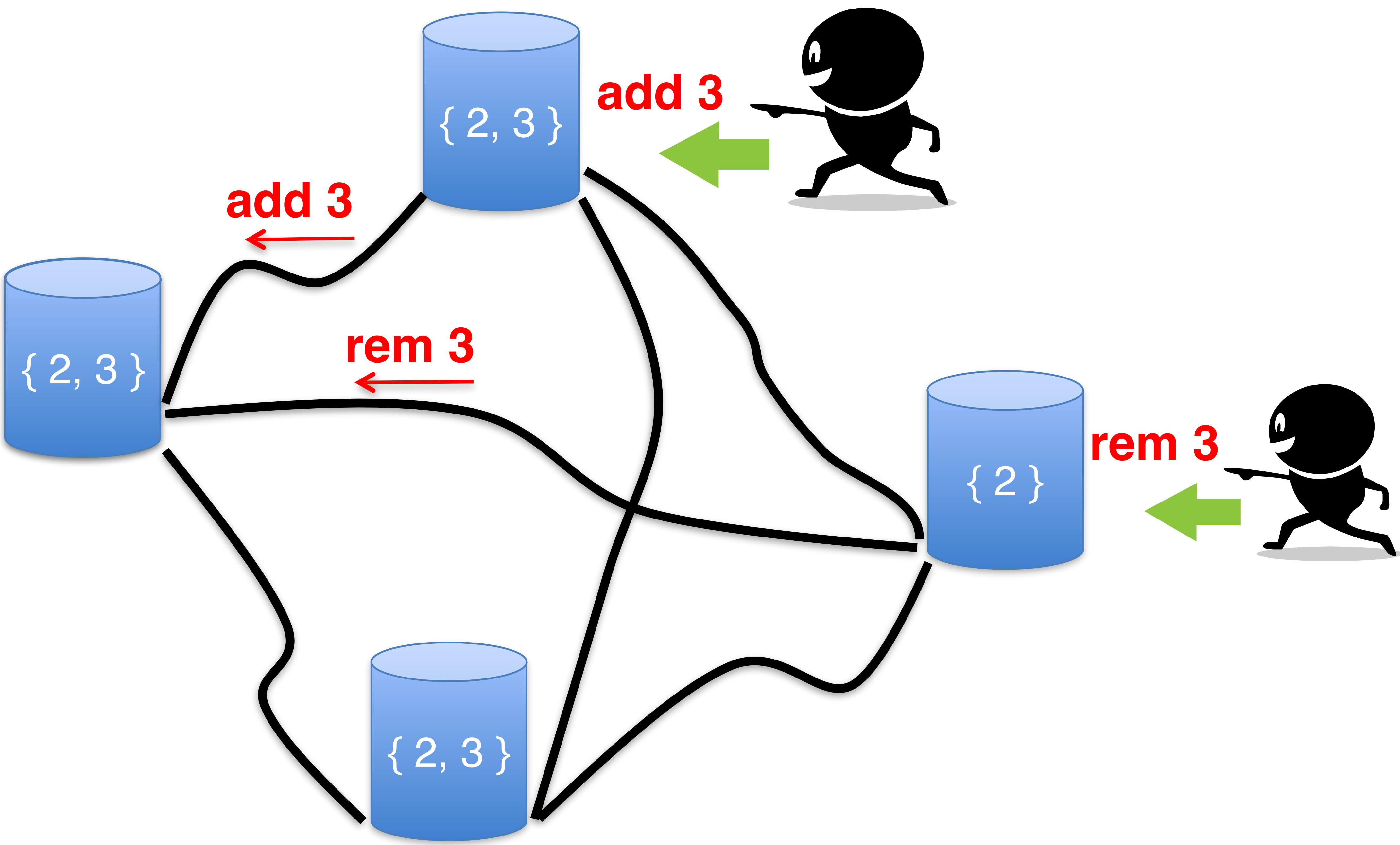
Concurrent operations



Concurrent operations

Solving conflicts between concurrent operations

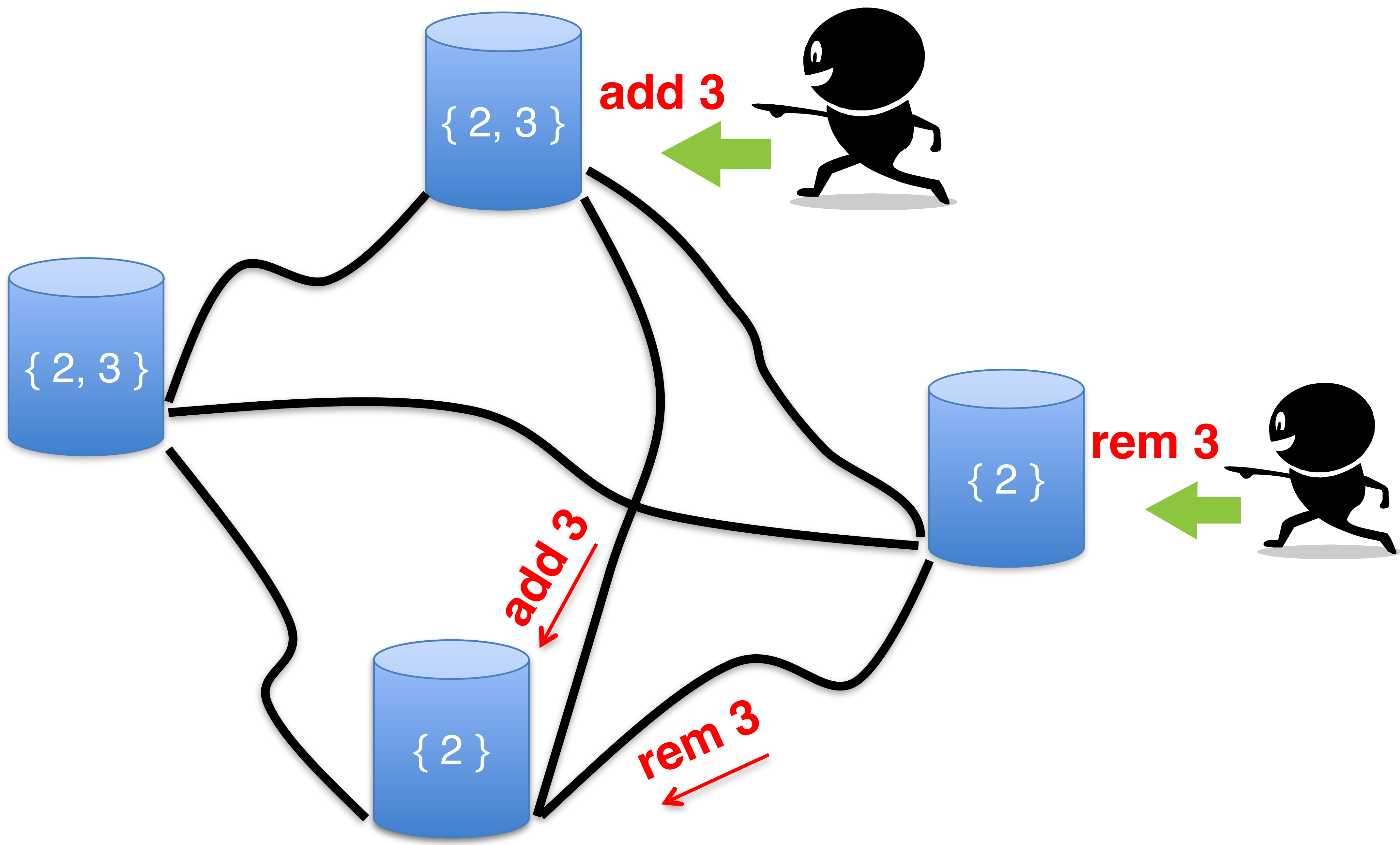
- speculate and roll-back, e.g., Google App Engine Datastore



Concurrent operations

Solving conflicts between concurrent operations

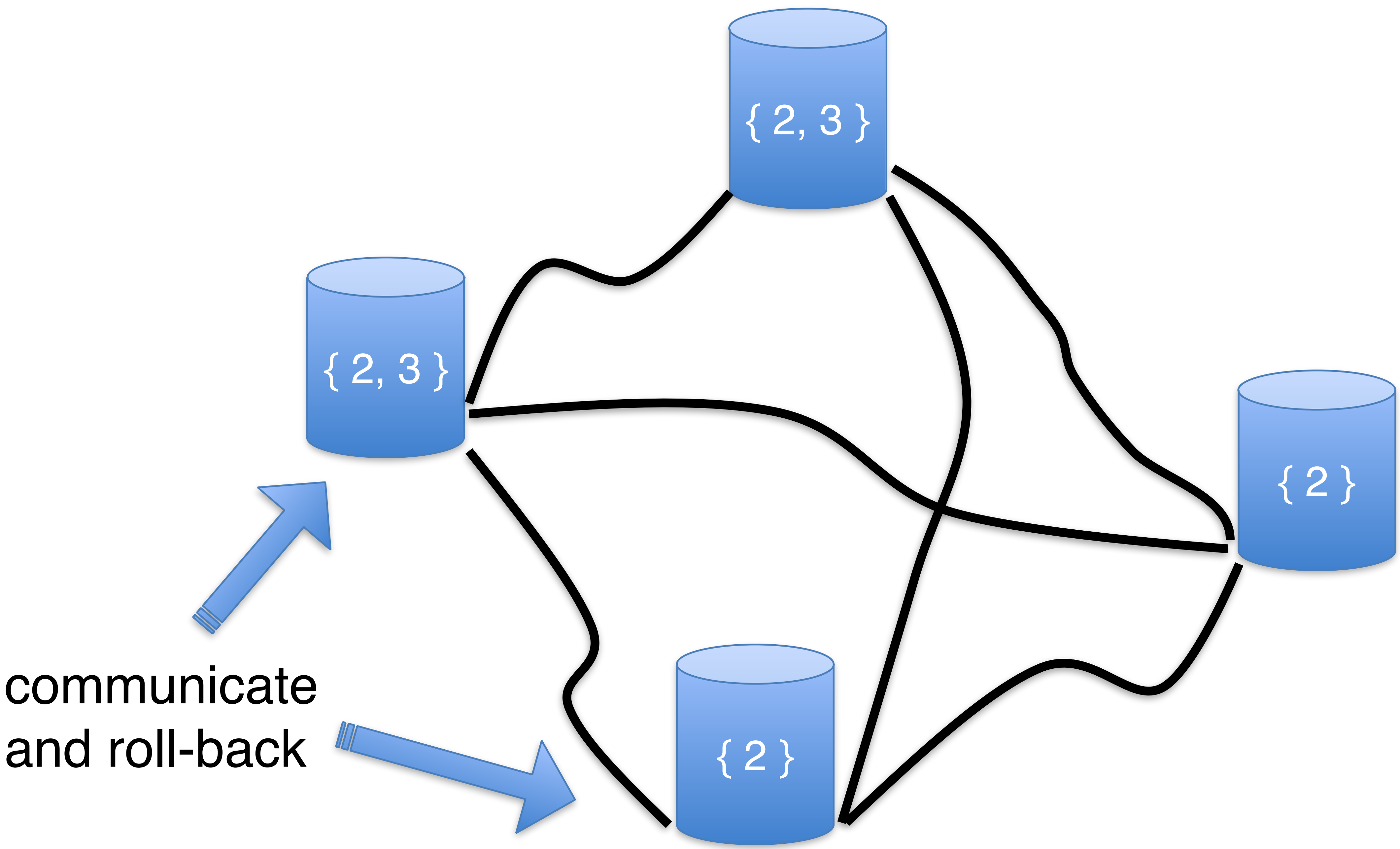
– speculate and roll-back, e.g., Google App Engine Datastore



Concurrent operations

Solving conflicts between concurrent operations

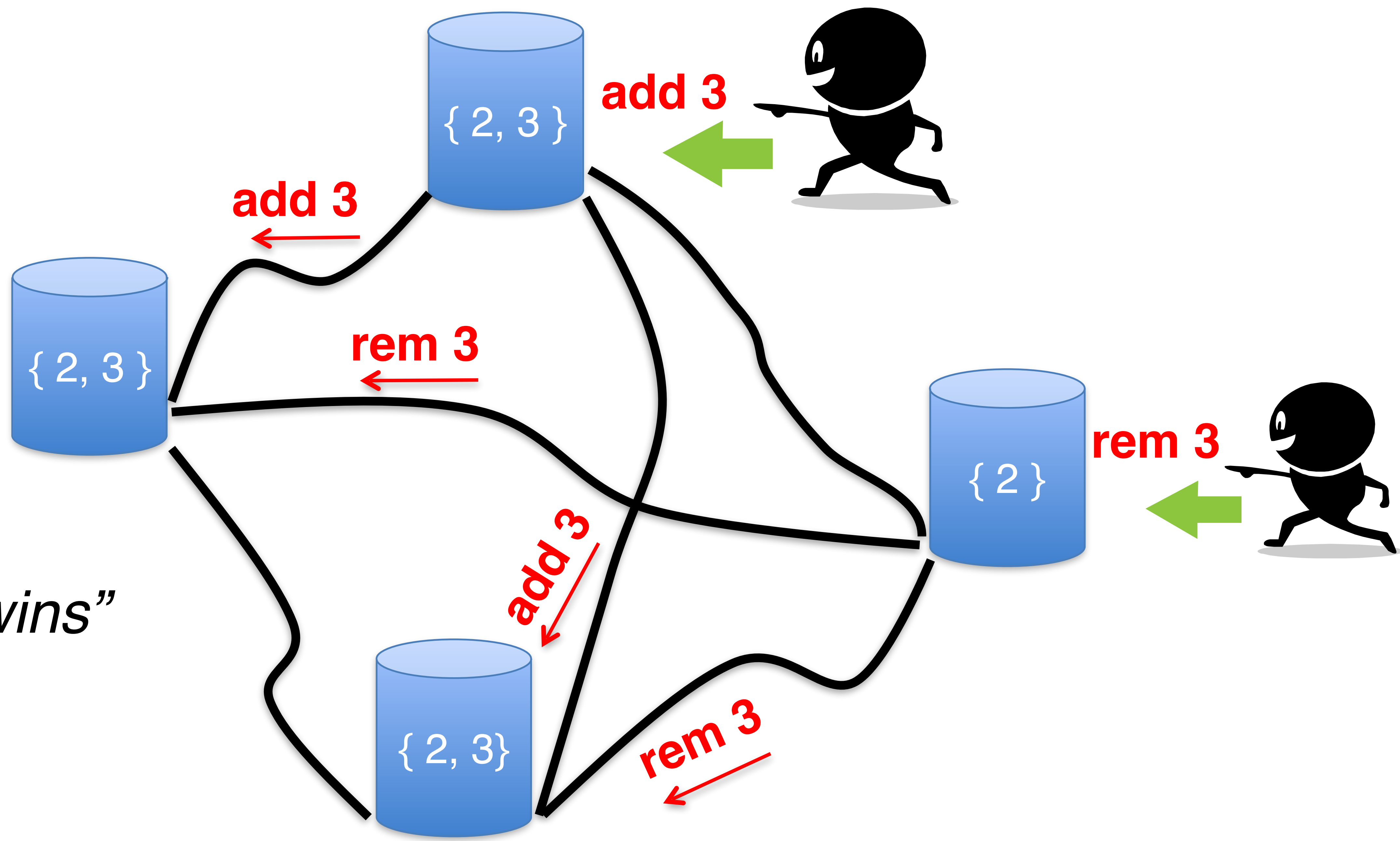
– speculate and roll-back, e.g., Google App Engine Datastore



Concurrent operations

Solving conflicts between concurrent operations

- speculate and eventually, roll-back, e.g., Google App Engine Datastore
- convergent conflict resolution, e.g., CRDTs [Shapiro et al.'11]



Consistency ?

CAP (Consistency-Availability-Partition tolerance) theorem
[GL '02]
implies
Strong consistency (linearizability) is impossible

==> Other (Weaker) Correctness Criteria:
Eventual Consistency, Causal Consistency, etc.

- Formal definition ?
- Verification ?

Eventual Consistency

[B, Enea, Hamza, POPL'14]
(other formalization: Burckhardt et al. POPL'14)

- At each moment, a site has a local view: a (sub)set of the operations that have been executed in the system
- Each operation is executed in the context of a local view
- Specification: a set of (happen-before) partial order relations explaining executions of operations

- Safety Part:

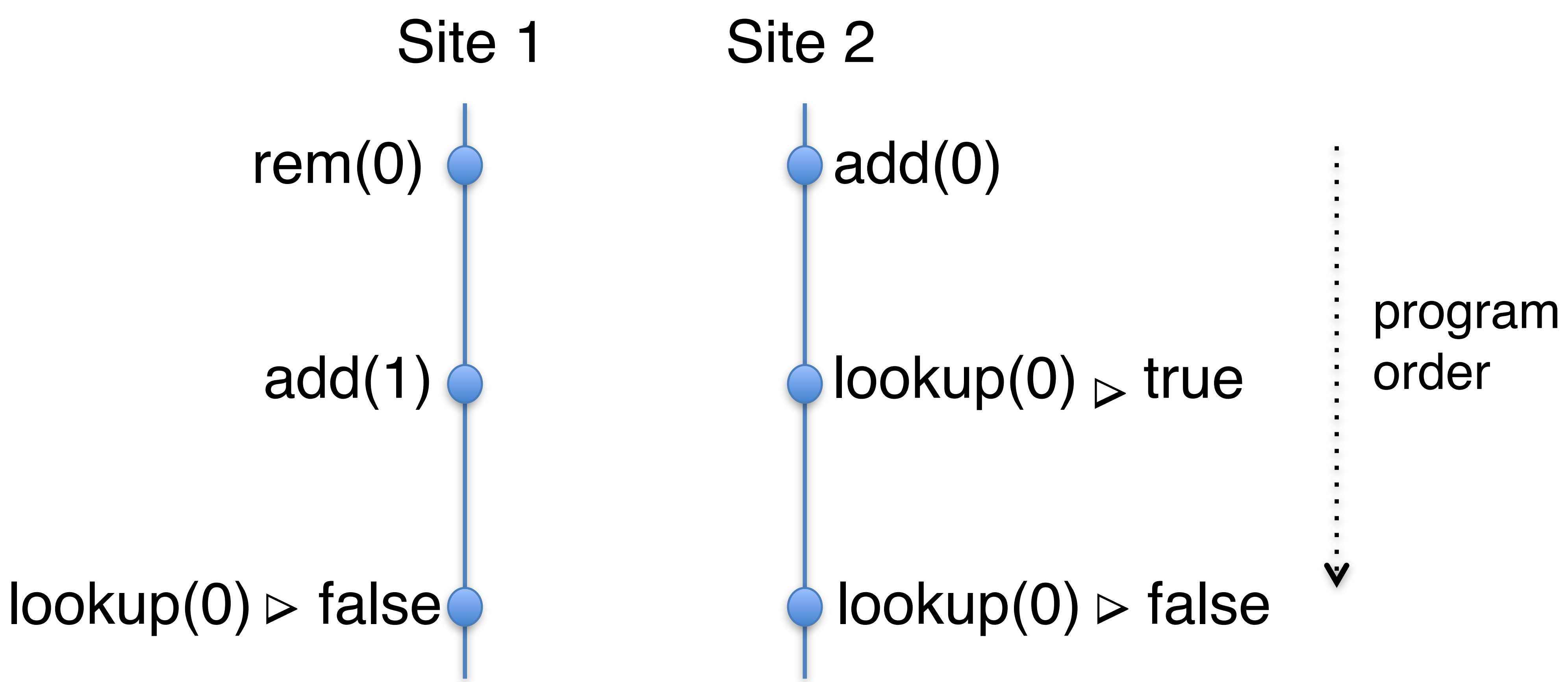
  Executions must satisfy some specification

- Liveness Part:

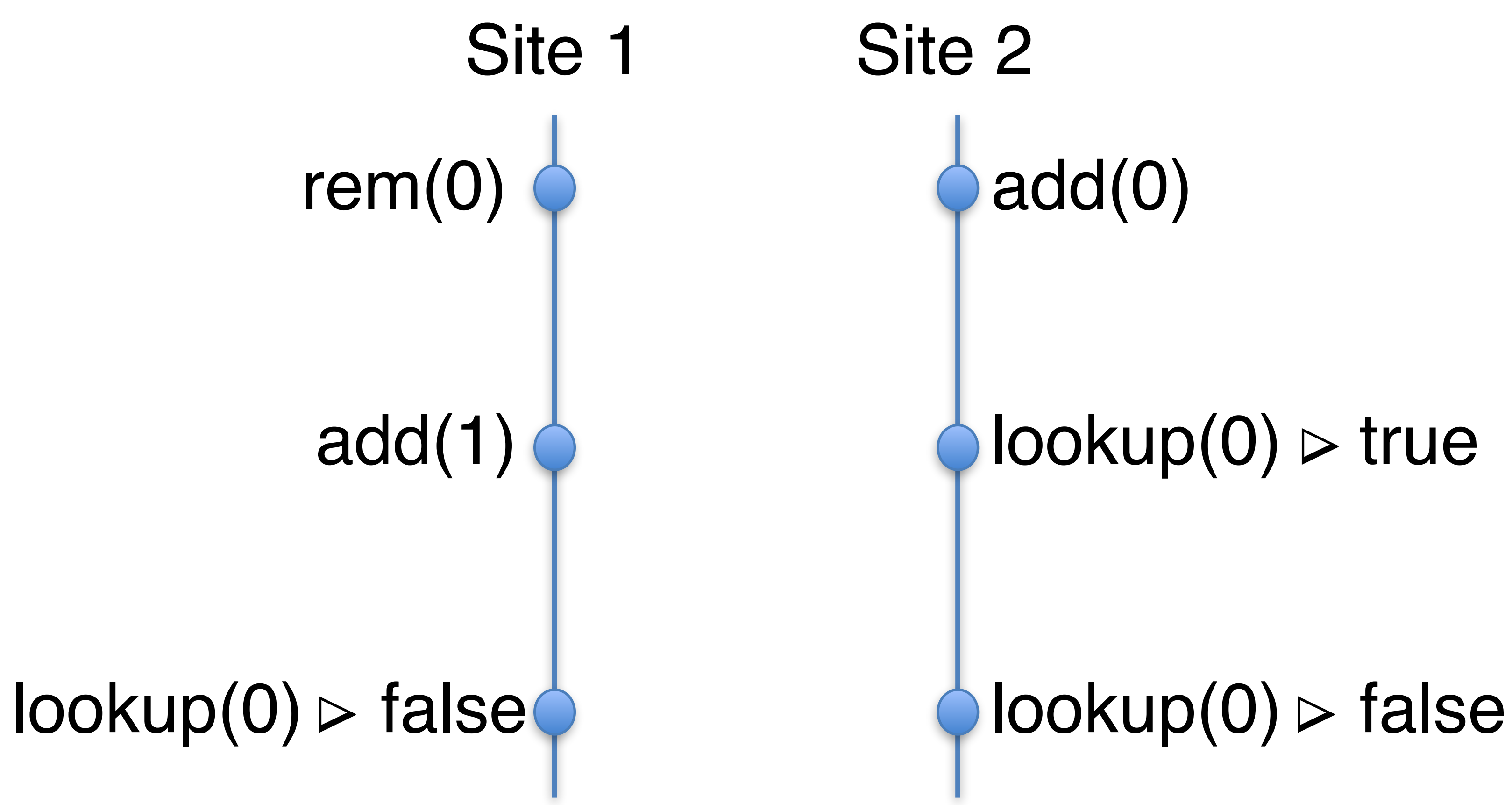
  Local views must converge toward global view

Modeling behaviors as traces

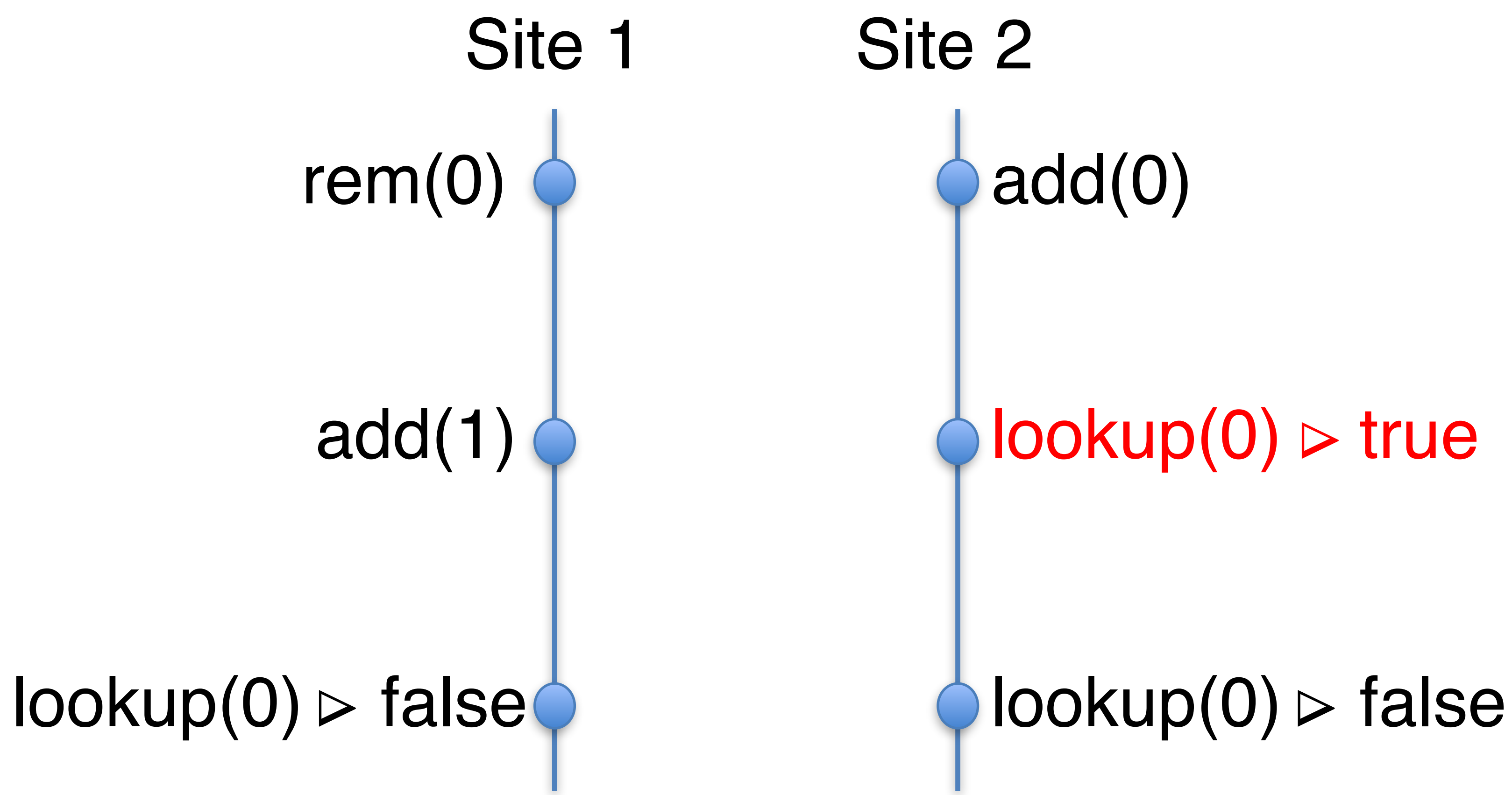
- operations are instances of a set of methods (add, rem, lookup)
- traces record the submitted operations and their return values
- trace = a partially-ordered set (poset) of operations
  - operations submitted to the same site are ordered



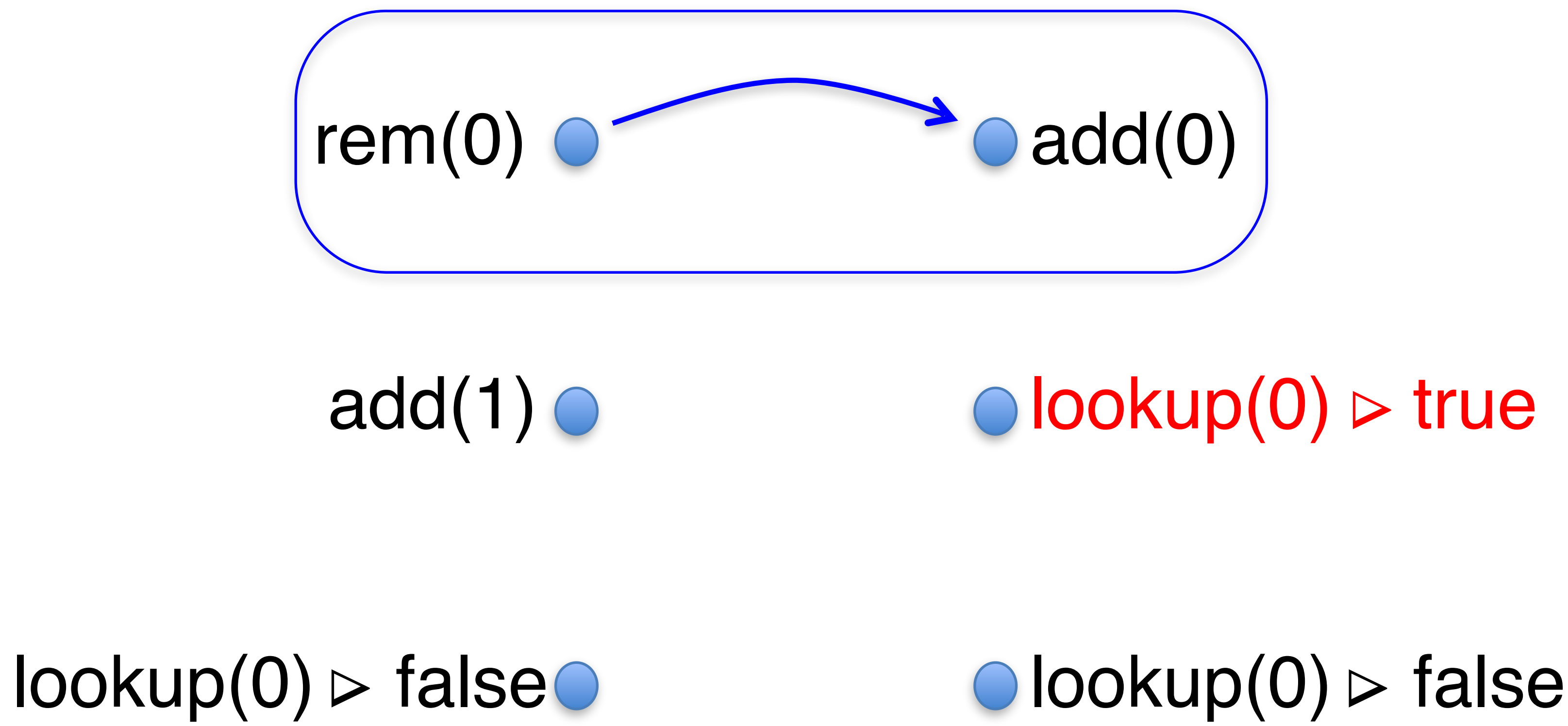
A trace is safe w.r.t a specification



A trace is safe w.r.t a specification

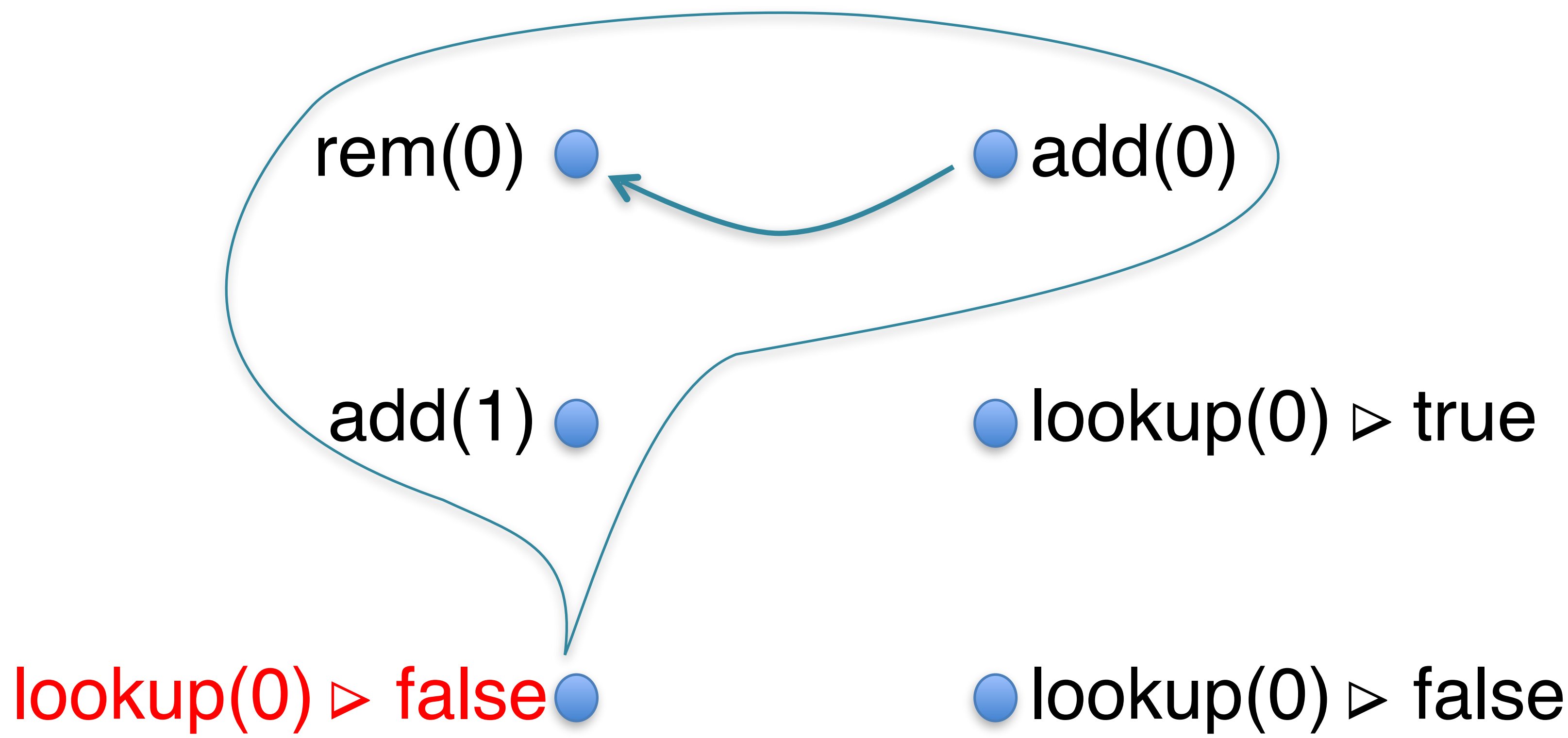


A trace is safe w.r.t a specification



- there exists a poset in the Spec.(lookup(0)▹true)
  (posets s.t. the projection on operations with input 0 has a maximal add(0))
- this poset is called local interpretation

A trace is safe w.r.t a specification

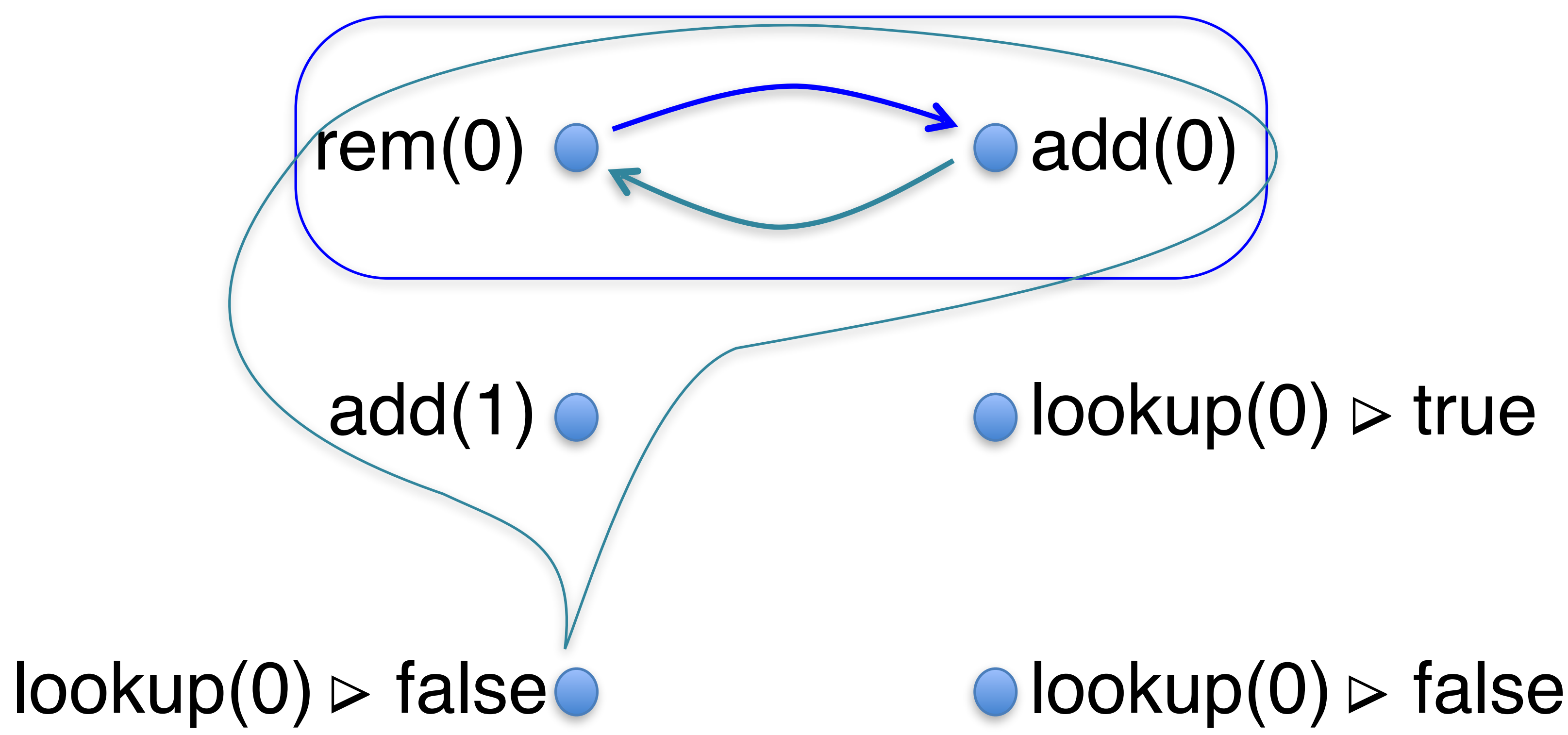


- there exists a poset in the Spec.(lookup(0) ▷ false)

  (posets s.t. the projection on operations with input 0 has **no** maximal add(0))

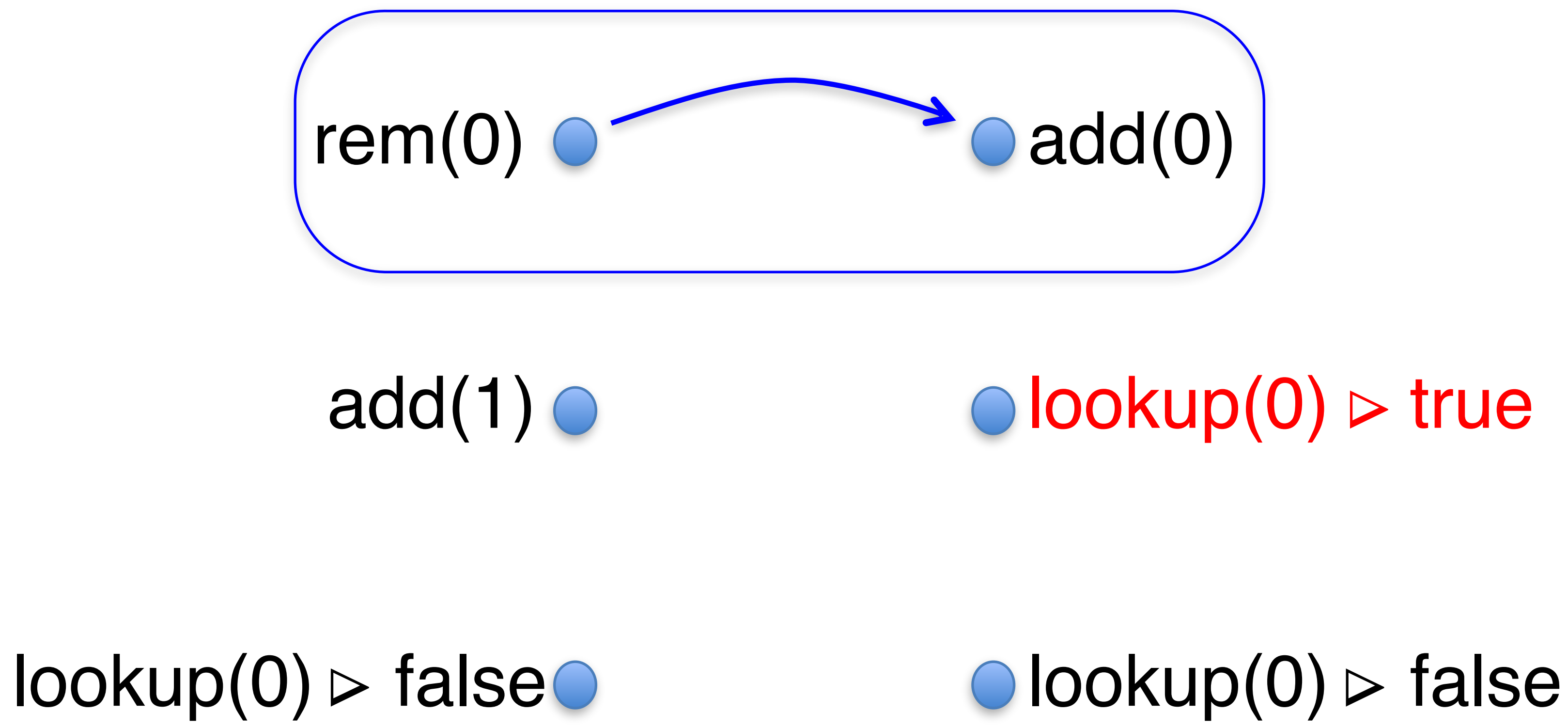
- this poset is called local interpretation

A trace is safe w.r.t a specification

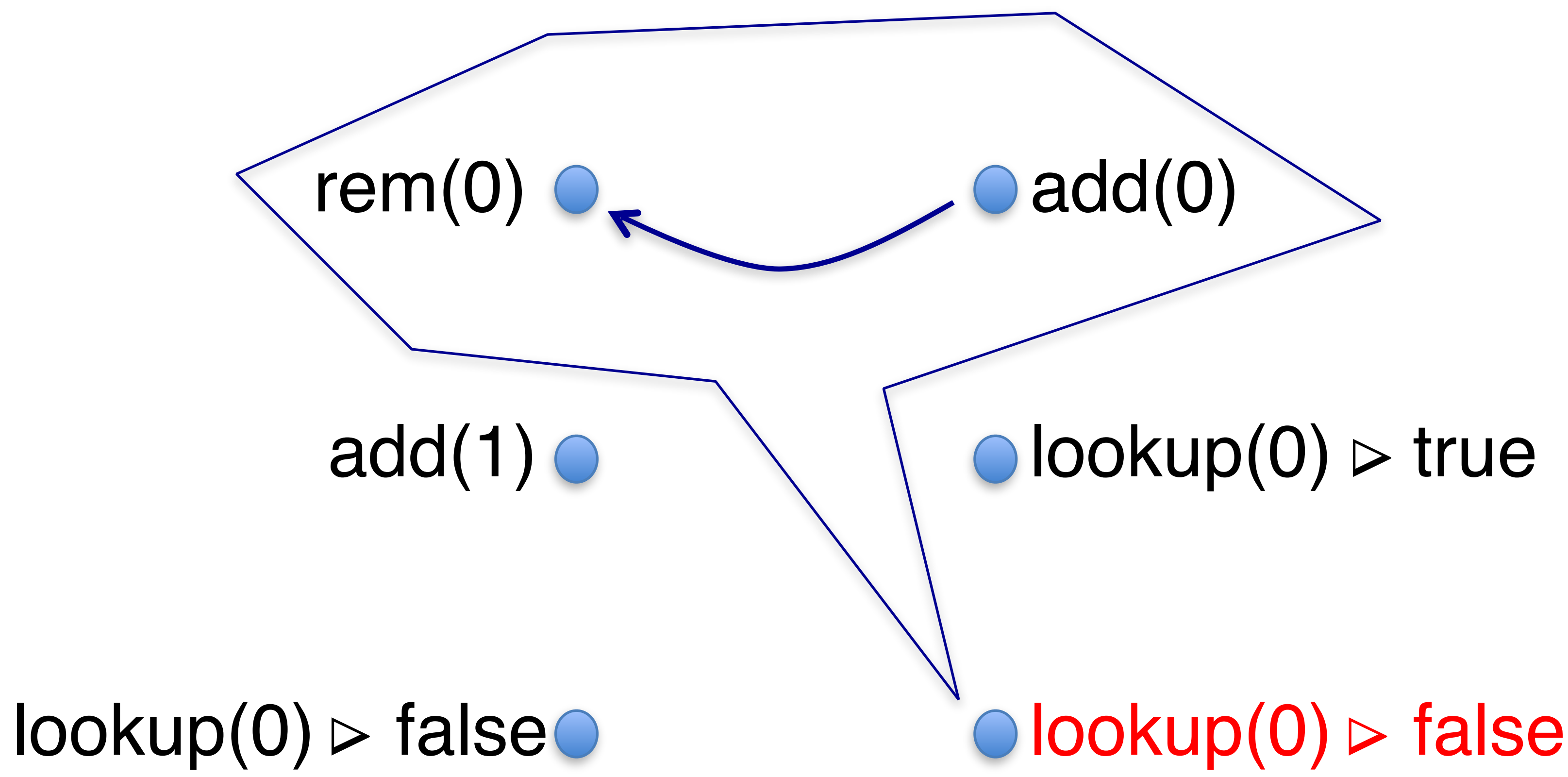


- local interpretations on different sites **do not** have to agree on the order between common operations
  - messages are received in different orders

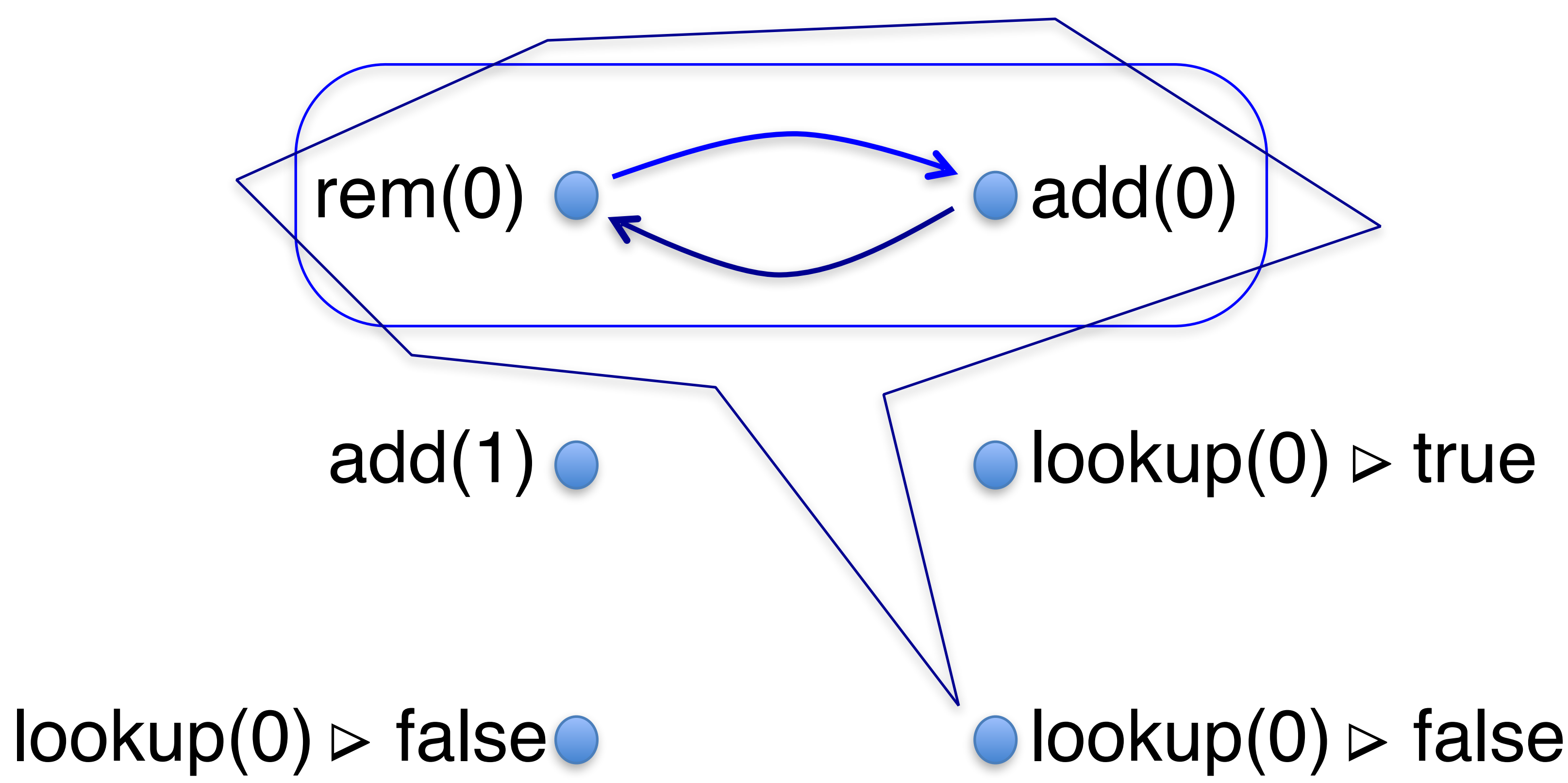
A trace is safe w.r.t a specification



A trace is safe w.r.t a specification

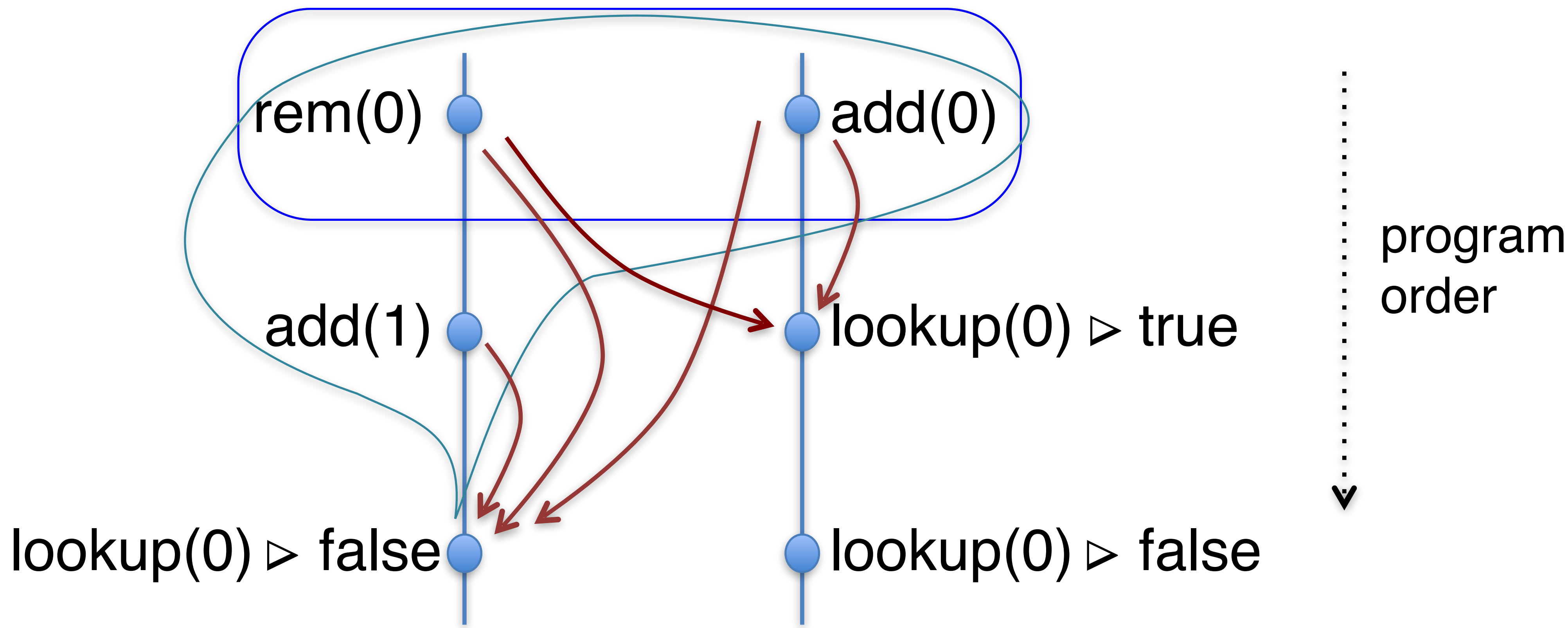


A trace is safe w.r.t a specification



- local interpretations on the same site **do not** have to agree on the order between common operations
  – conflict resolution may be non-deterministic, e.g., speculative executions

A trace is safe w.r.t a specification

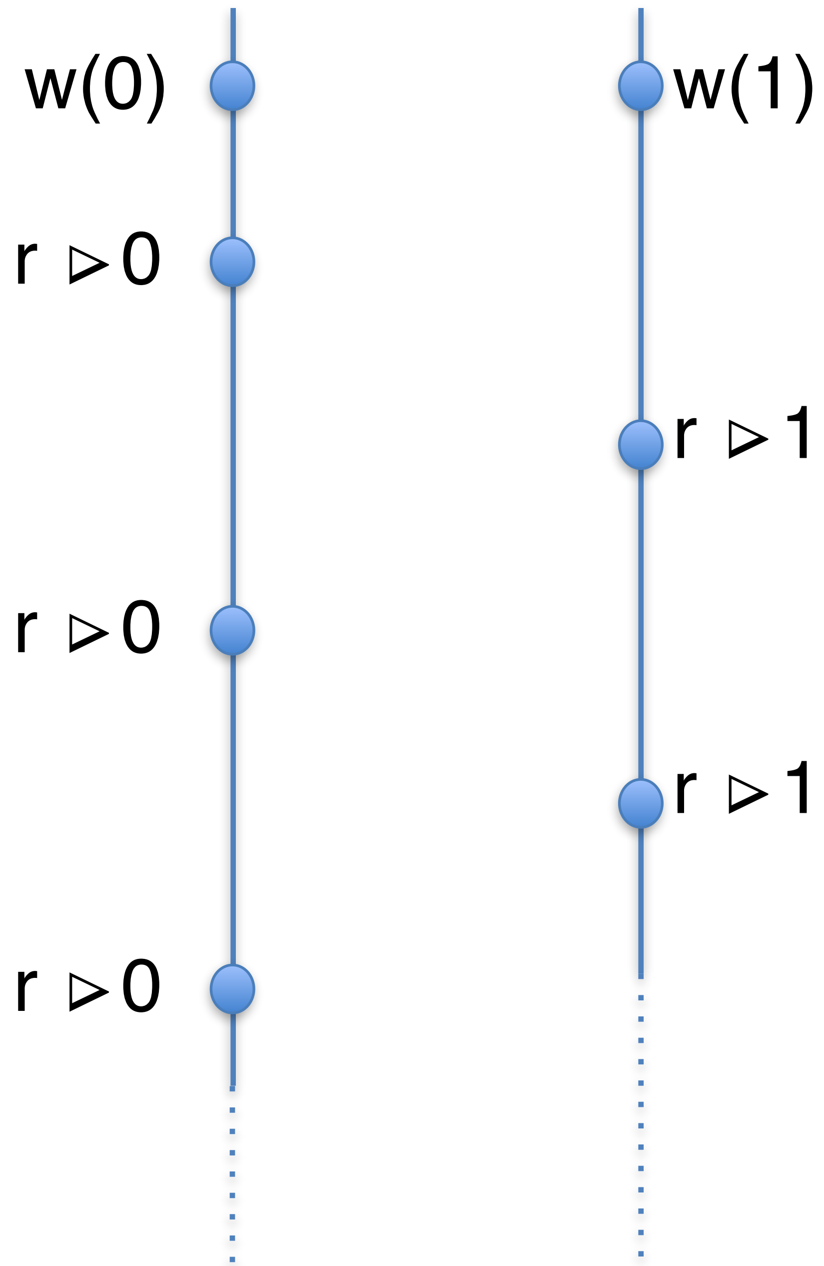


- executed-before ∪ program order is acyclic
  (aka "visibility")
  - α executed-before β iff α belongs to the local interpretation of β
  - messages that announce some operation are sent after the operation itself is submitted to a site

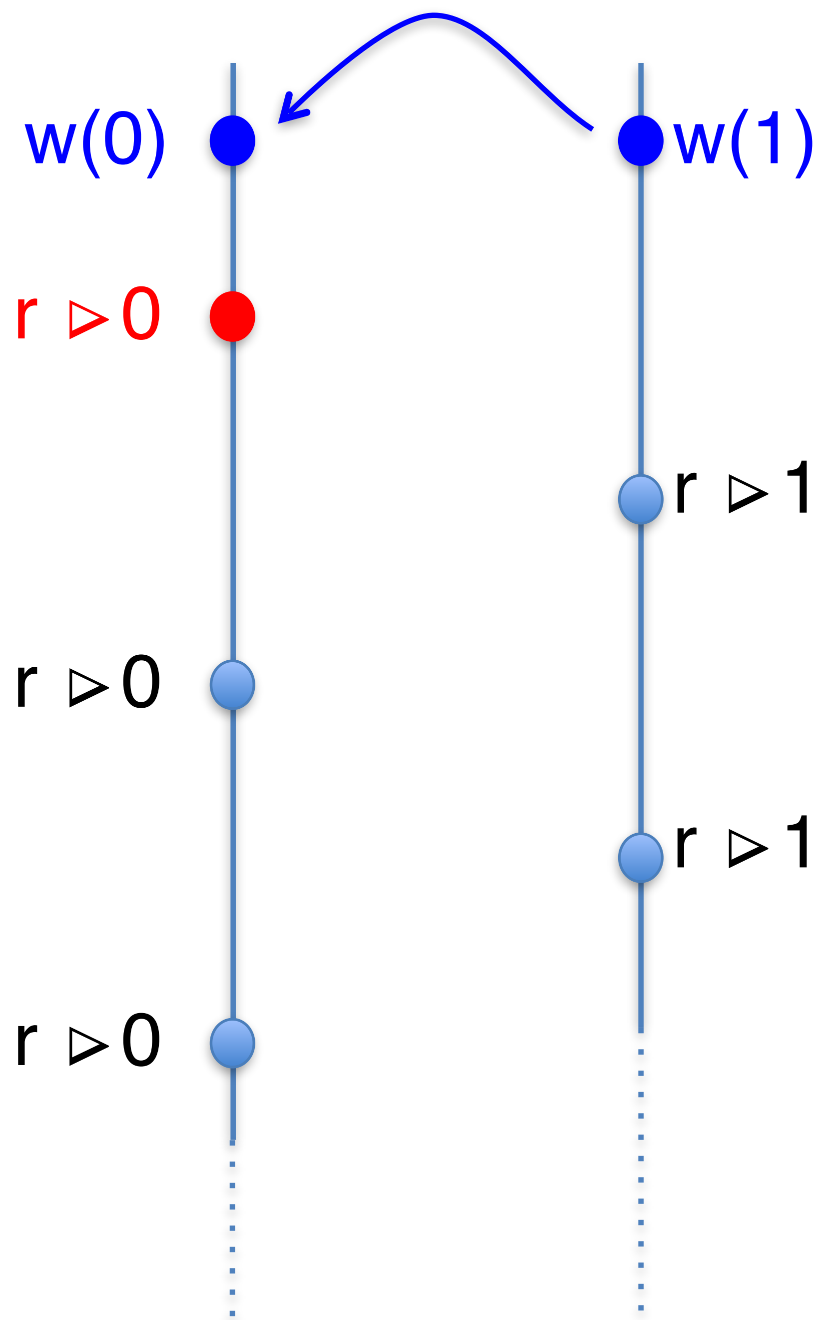
A trace is safe w.r.t a specification

1. For each operation O there exists a local interpretation (poset of ops.) in the specification of O
2. executed-before ∪ program order is acyclic

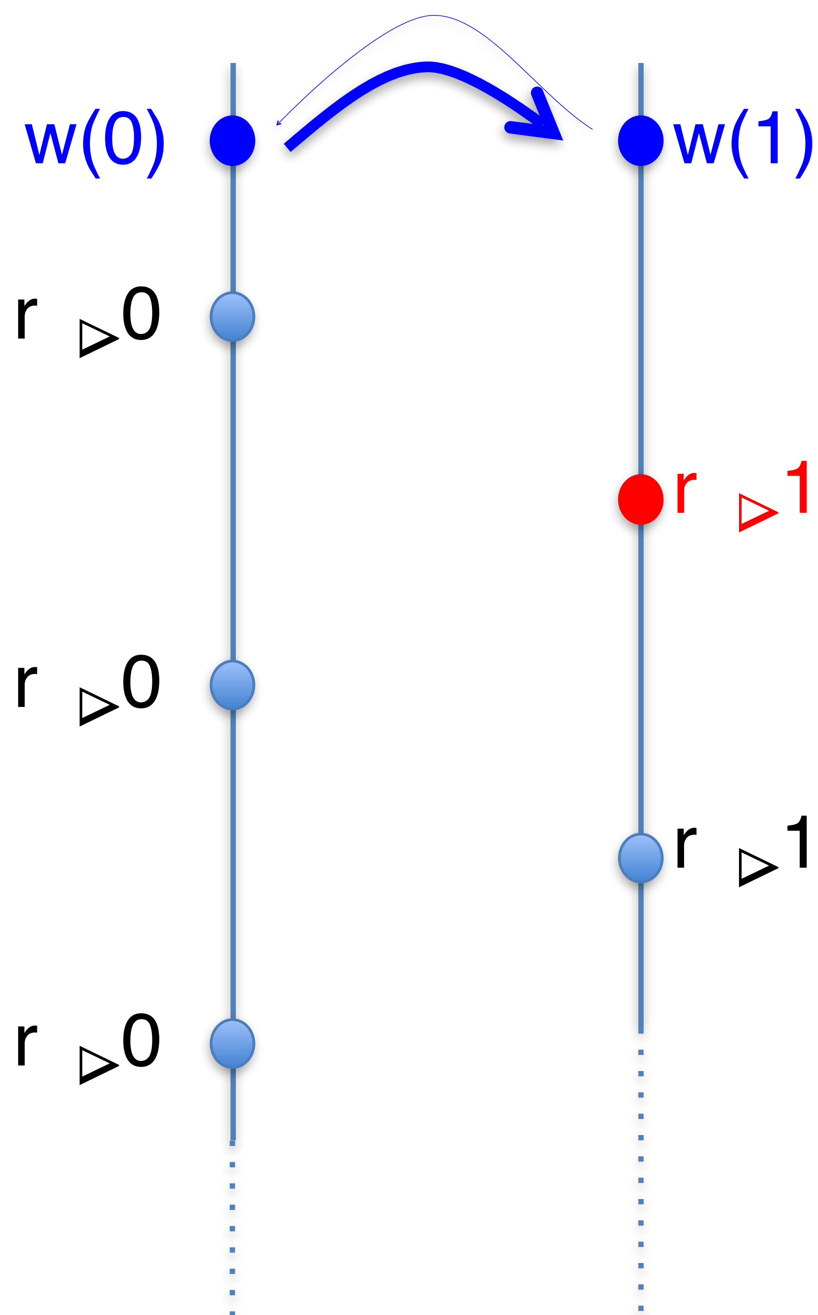
Violation of convergence



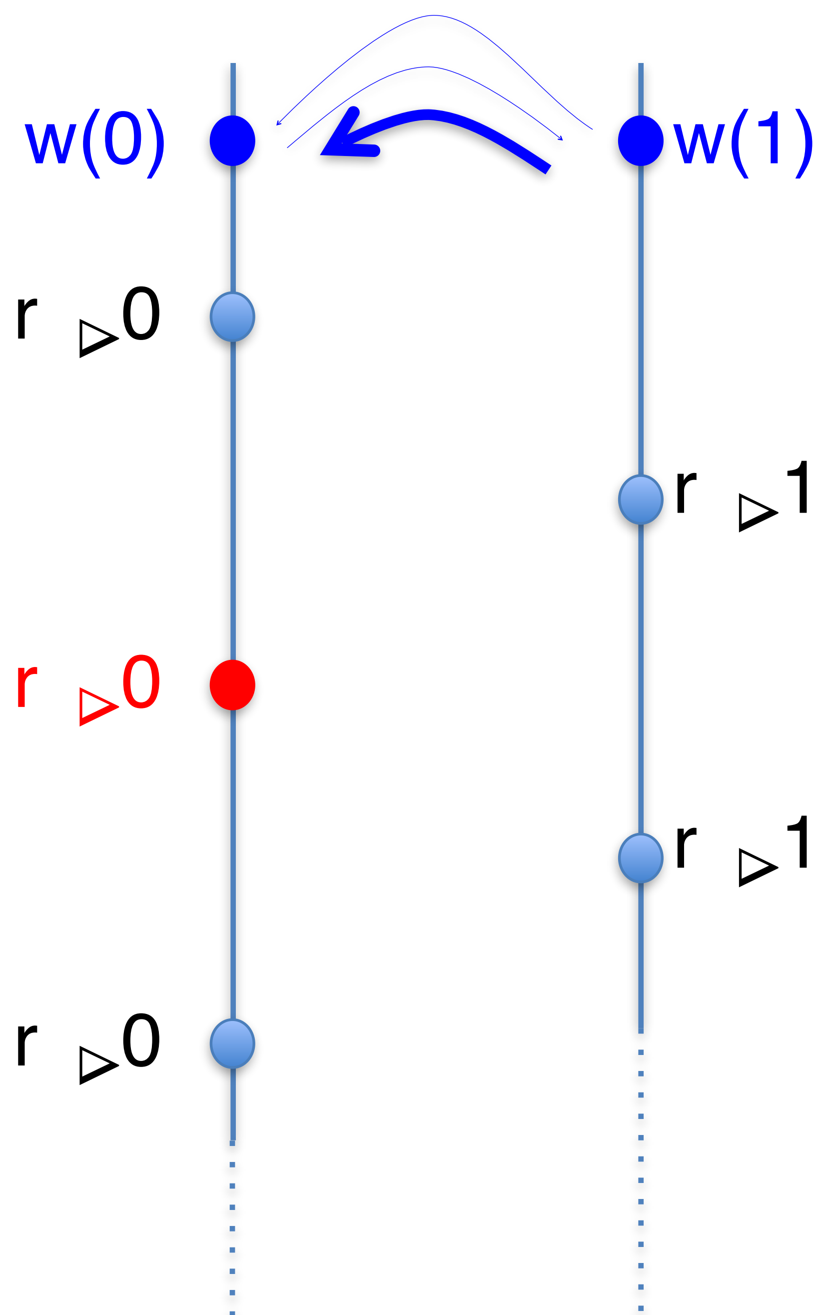
Violation of convergence



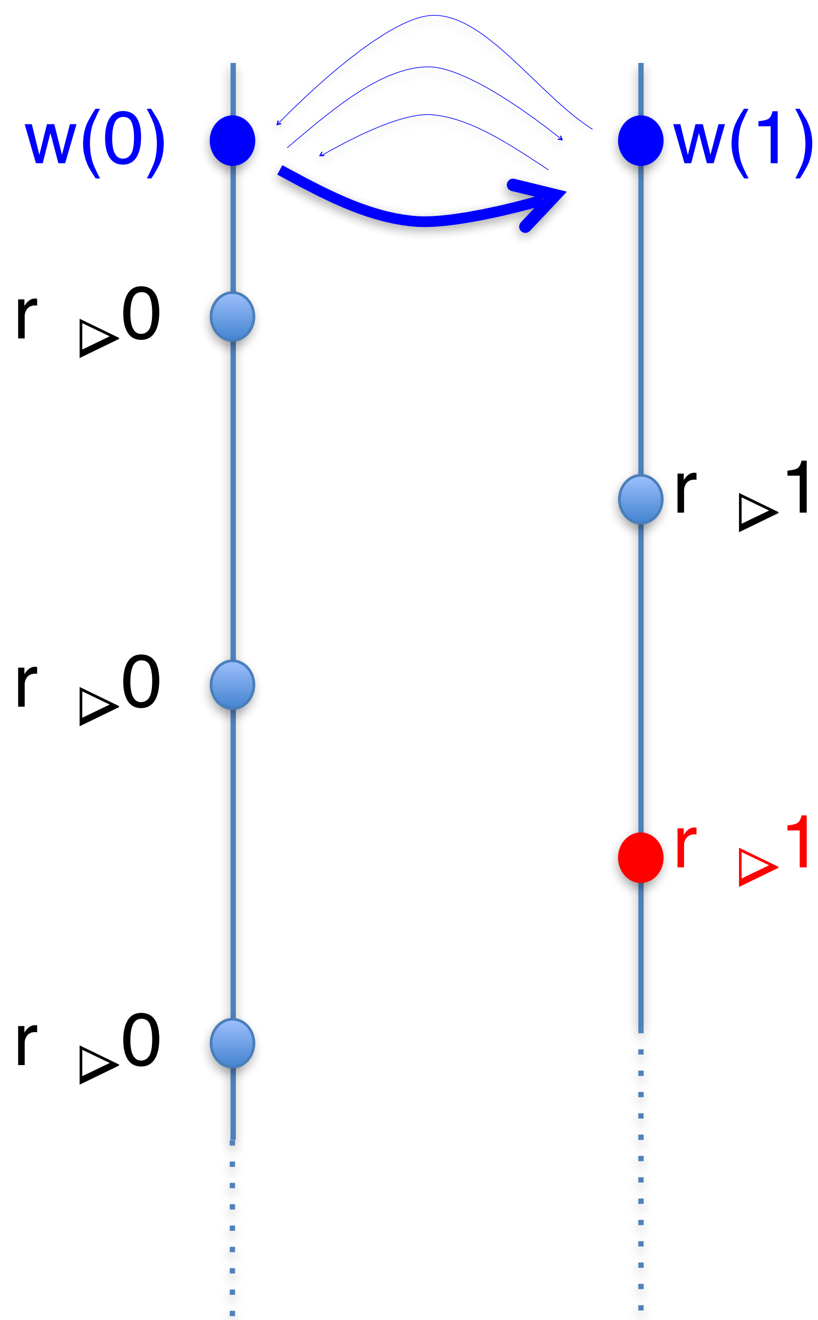
Violation of convergence



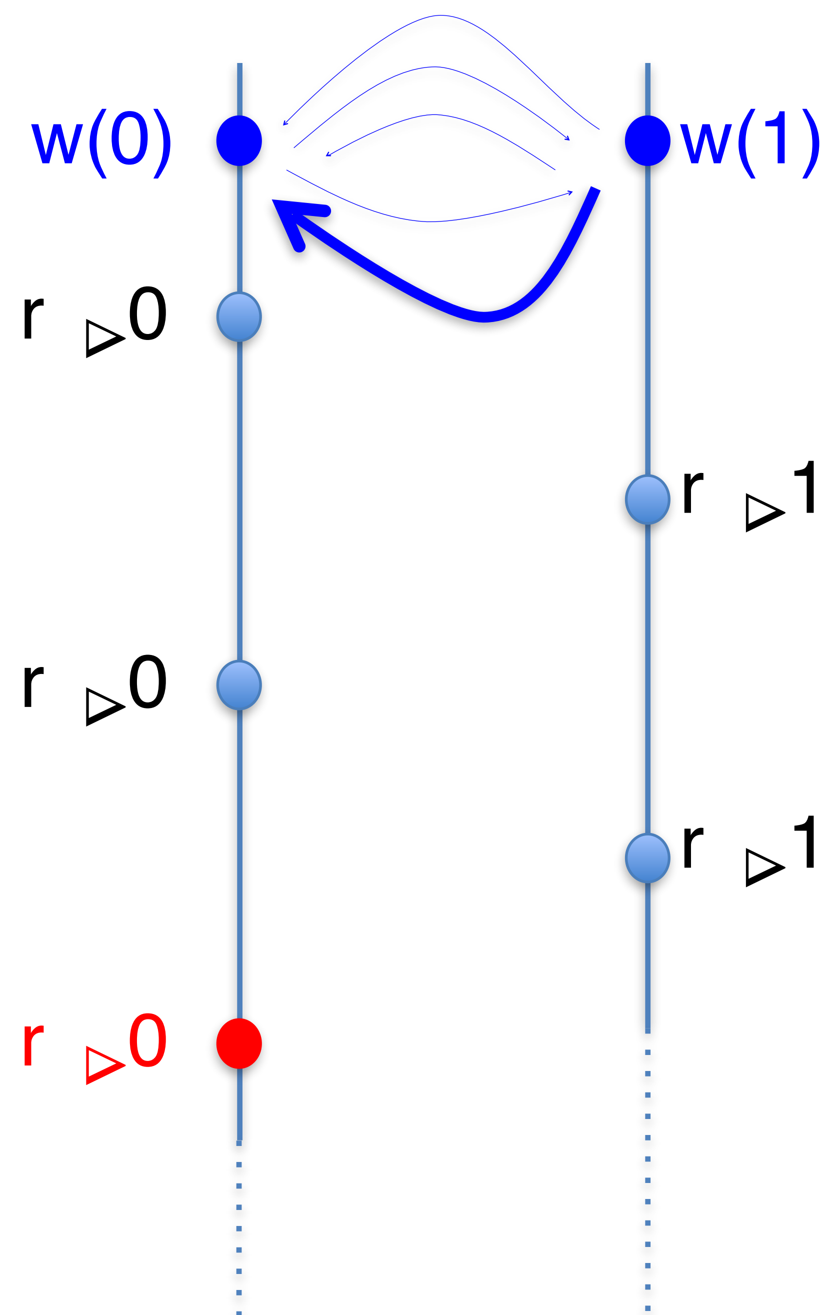
Violation of convergence



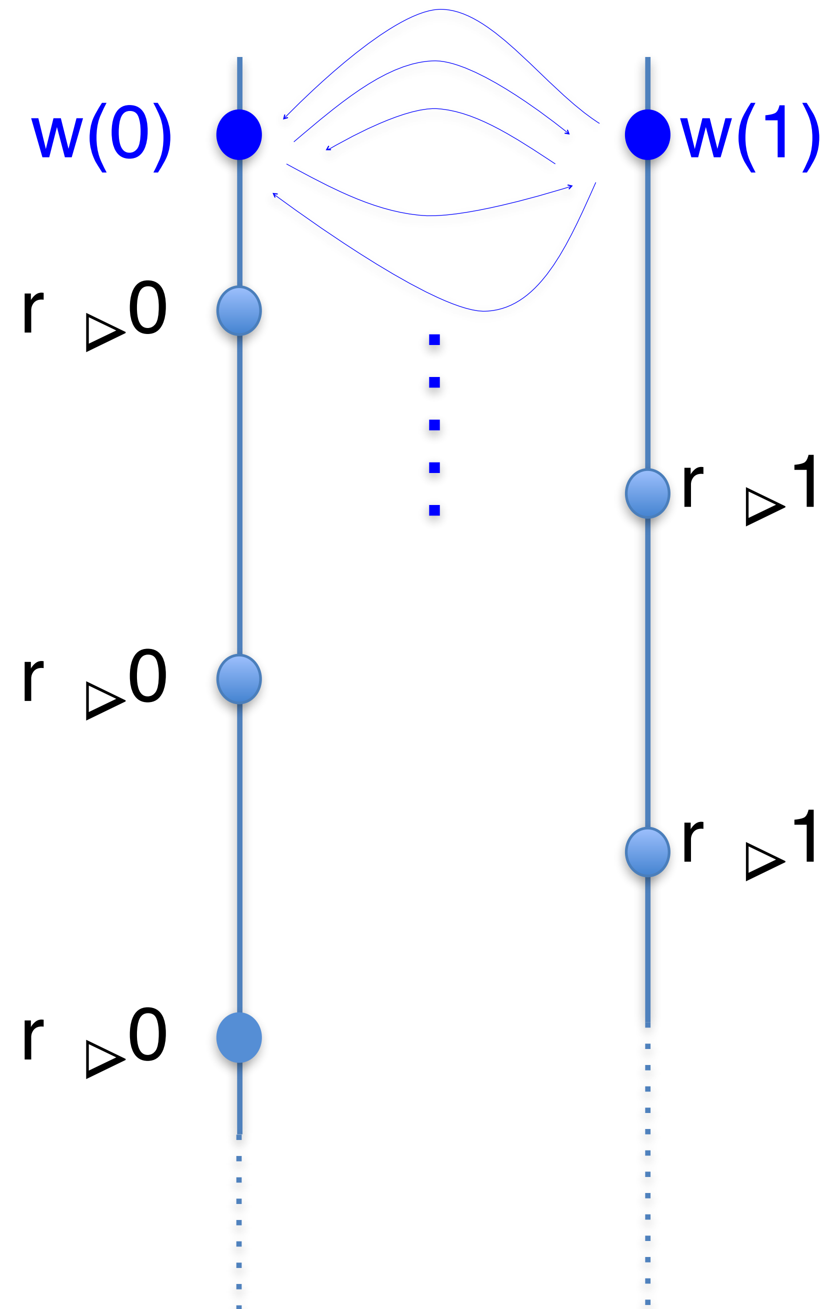
Violation of convergence



Violation of convergence



Violation of convergence



Convergence

For every infinite trace

There is a *global (infinite) interpretation* GI s.t.
For every prefix P of GI,
there is a *finite* number of local interpretations LI s.t.
P is not a prefix of LI

i.e.,

After some point,
all local interpretations have P as a prefix

Checking Eventual Consistency

- Reduction to state-reachability / model-checking
- Given a Spec, construct a monitor M[Spec] s.t.

For every Impl,

Impl is not EC wrt Spec

iff

Impl x M[Spec] can reach a specific control state

(violates some LTL formula)

Checking Eventual Consistency

Basic ingredients for the reduction

- Monitor tracks sequentializations of traces

A trace is unsafe iff all its sequentializations contain an operation that can not be explained using its past

- We only need to reason about minimal local interpretations
- We can use counting:
  - Count the number of occurrences of each operation
  - Compare with minimal posets in the specification
- Safety: Monitor checks an invariant on counters
- Convergence: Monitor checks a condition on repeating operations

Decidability, Complexity

- Class of implementations:
  - fixed number of boolean programs communicating through unbounded unordered channels
- Class of specifications:
  - new class of finite automata for representing posets
- Verifying Eventual Consistency is decidable
  - Verifying Safety is reducible to Coverability in Petri Nets (in 2EXPSPACE and EXPSPACE-hard)
  - Verifying Convergence is reducible to Reachability in Petri Nets (EXPSPACE-hard)

Conclusion/Future work

- Reductions to Invariant Checking/MC
- Decidability for Static Linearizability and Eventual Consistency
- High complexity, undecidability (linearizability in general)
- Semantical issues:
  - Formal definition of correctness criteria
  - (Weak) memory models/guarantees of the underlying infrastructure
- Decidability/Complexity issues:
  - Other criteria such as causal consistency
  - Classes of implementations/specifications
- Efficient approximate analysis:
  - Abstractions for proving correctness
  - Under-approximations for detecting violations