Eventually-Consistent Data Structures

Sean Cribbs
@seancribbs #CRDT
StrangeLoop 2012

I work for Basho

We make riak

Riak is Eventually Consistent

So are Voldemort and Cassandra

No ACID!

Duals or Duels?

Duals or Duels?

object-oriented / functional

Duals or Duels?

object-oriented / functional

static / dynamic

Duals or Duels?

object-oriented / functional

static / dynamic

consistency / availability

Duals or Duels?

object-oriented / functional

static / dynamic

consistency / availability

throughput / latency

Duals or Duels?

object-oriented / functional

static / dynamic

consistency / availability

throughput / latency

threaded / evented

Duals or Duels?

object-oriented / functional

static / dynamic

consistency / availability

throughput / latency

threaded / evented

safety / liveness

Safety / Liveness

Proving the Correctness of Multiprocess Programs - Leslie Lamport
(March 1977)

Safety / Liveness

Proving the Correctness of Multiprocess Programs - Leslie Lamport (March 1977)

- **Safety:** “nothing bad happens” (partial correctness)

Safety / Liveness

Proving the Correctness of Multiprocess Programs - Leslie Lamport (March 1977)

- **Safety:** “nothing bad happens” (partial correctness)
- **Liveness:** “something good eventually happens” (termination)

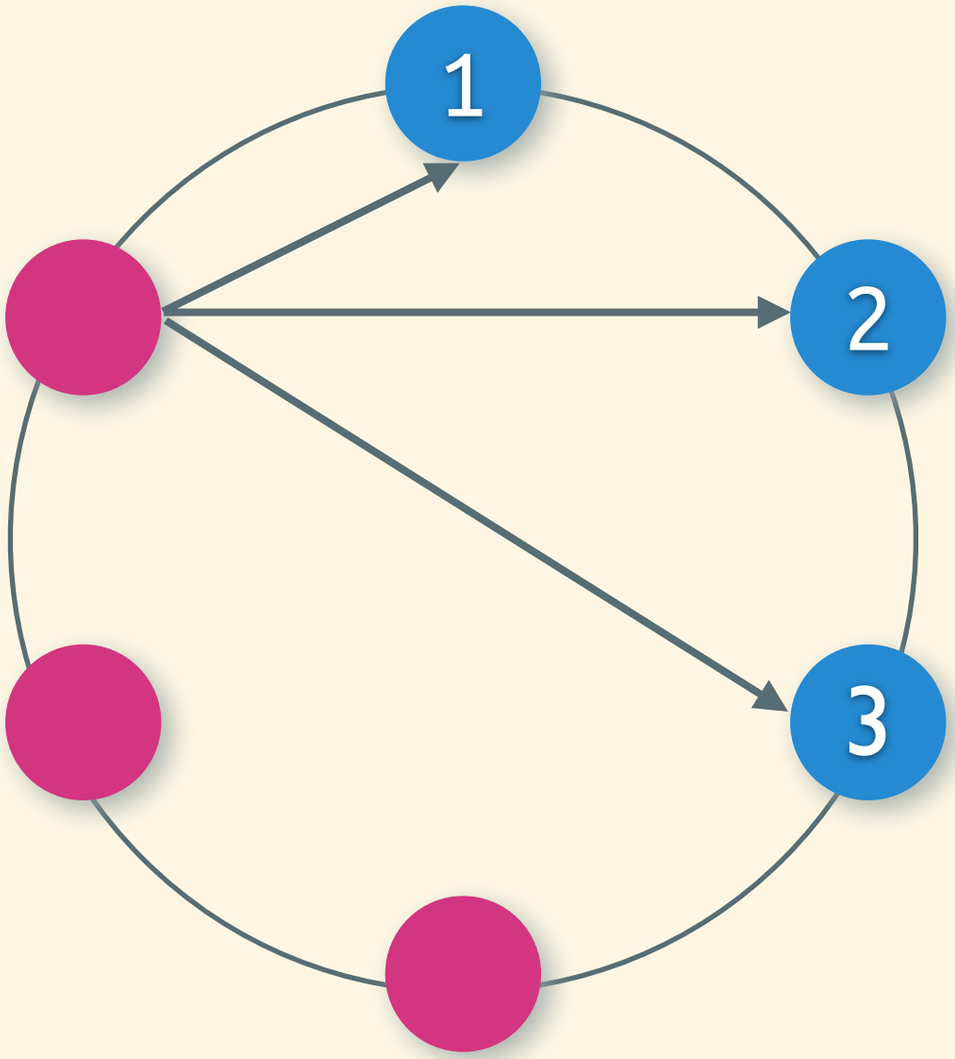
Safety / Liveness

Proving the Correctness of Multiprocess Programs - Leslie Lamport (March 1977)

- **Safety:** “nothing bad happens” (partial correctness)
- **Liveness:** “something good eventually happens” (termination)

“Safety and liveness: Eventual consistency is not safe” - Peter Bailis
http://www.bailis.org/blog/safety-and-liveness-eventual-consistency-is-not-safe/

Eventual Consistency



Replicated

Loose coordination

Convergence

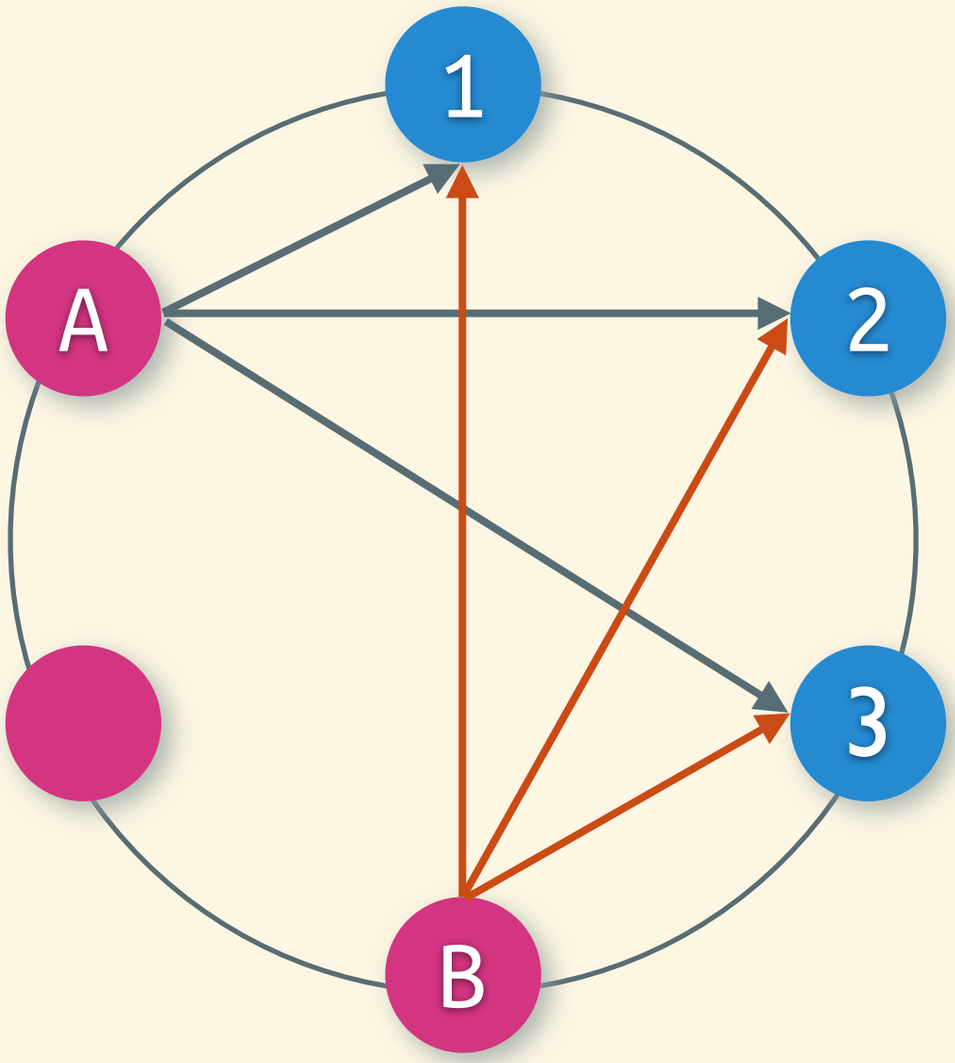
Eventual is Good

✔ Fault-tolerant

✔ Highly available

✔ Low-latency

Consistency?

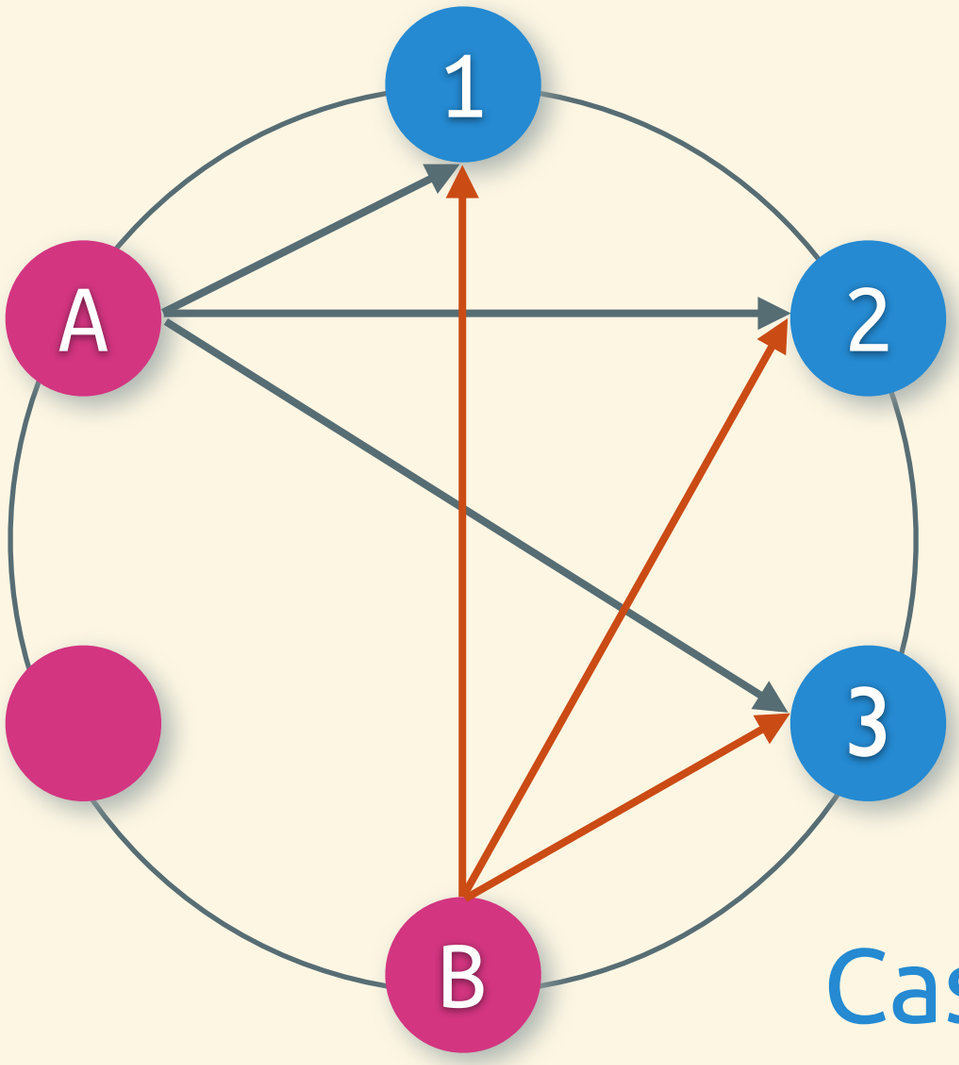


No clear winner!

Throw one out?

Keep both?

Consistency?



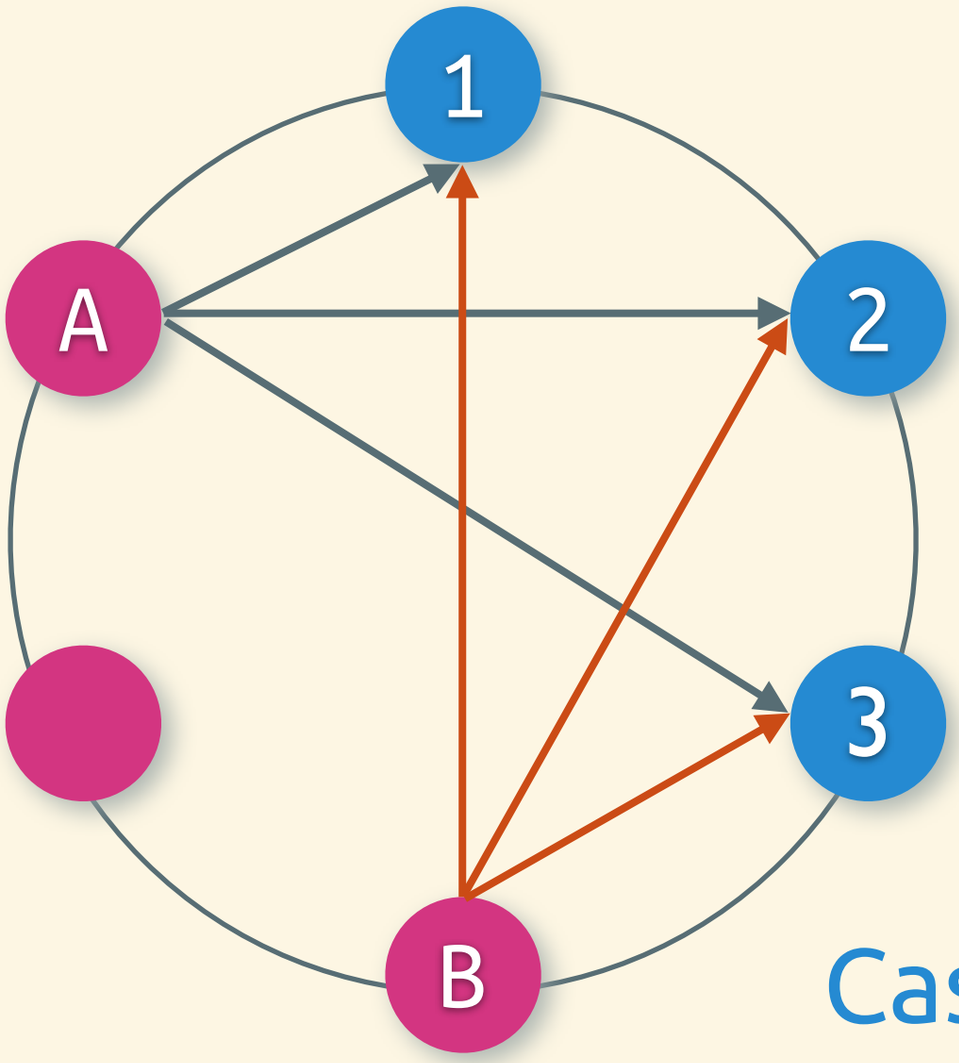
No clear winner!

Throw one out?

Keep both?

Cassandra

Consistency?



No clear winner!

Throw one out?

Keep both?

Cassandra

Riak & Voldemort

Conflicts!

A!

B!

Semantic Resolution

- Your app knows the domain - use business rules to resolve
- Amazon Dynamo’s shopping cart

Semantic Resolution

- Your app knows the domain - use business rules to resolve
- Amazon Dynamo's shopping cart

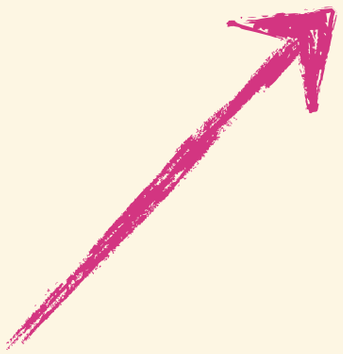
"Ad hoc approaches have proven brittle and error-prone"

Conflict-Free Replicated Data Types

Conflict-Free Replicated Data Types

useful abstractions

Conflict-Free Replicated Data Types



multiple independent copies

useful abstractions

resolves automatically
toward a single value

Conflict-Free Replicated Data Types

multiple independent
copies

useful abstractions

Logic and Lattices for Distributed Programming

Neil Conway
William Marczak
Peter Alvaro
Joseph M. Hellerstein
David Maier

http://db.cs.berkeley.edu/papers/UCB-lattice-tr.pdf

Bounded Join Semi-Lattices

Bounded Join Semi-Lattices

$\langle S, \sqcup, \bot \rangle$

Bounded Join Semi-Lattices

$\langle S, \sqcup, \bot \rangle$

- S is a set

Bounded Join Semi-Lattices

$\langle S, \sqcup, \bot \rangle$

- S is a set
- $\sqcup$ is a least-upper bound (join/merge) on S

Bounded Join Semi-Lattices

$\langle S, \sqcup, \bot \rangle$

- S is a set
- $\sqcup$ is a least-upper bound (join/merge) on S
- $\bot \in S$

Bounded Join Semi-Lattices

$\langle S, \sqcup, \bot \rangle$

- S is a set
- $\sqcup$ is a least-upper bound (join/merge) on S
- $\bot \in S$
- $\forall x, y \in S: x \leq_S y \Leftrightarrow x \sqcup y = y$

Bounded Join Semi-Lattices

$\langle S, \sqcup, \bot \rangle$

- S is a set
- $\sqcup$ is a least-upper bound (join/merge) on S
- $\bot \in S$
- $\forall x, y \in S: x \leq_S y \Leftrightarrow x \sqcup y = y$
- $\forall x \in S: \; x \sqcup \bot = x$

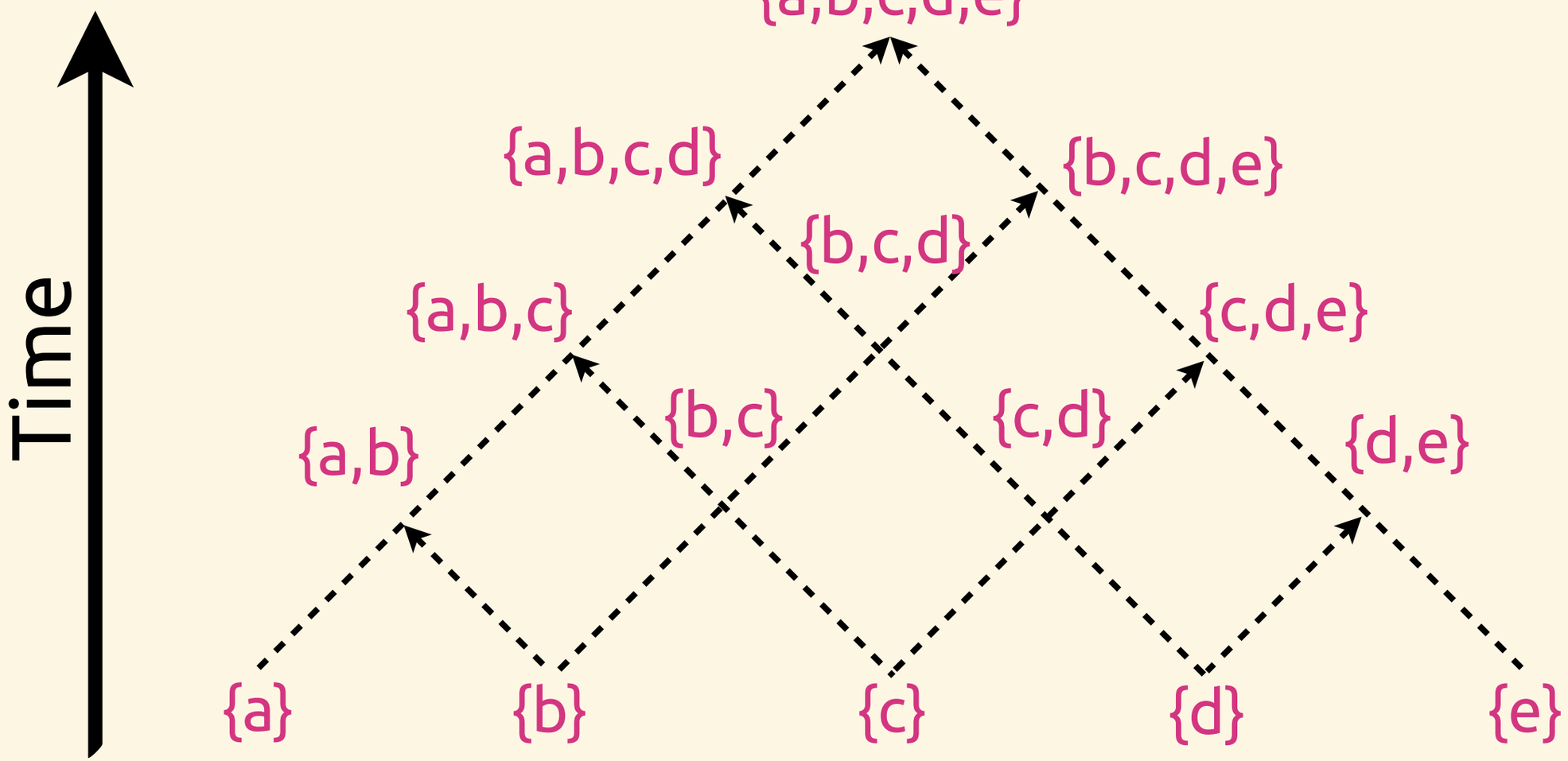
lmax Lattice

$$S := \mathscr{R}$$

$$a \sqcup b := \max(a,b)$$

$$\bot := -\infty$$

lset Lattice



INSTITUT NATIONAL DE RECHERCHE EN INFORMATIQUE ET EN AUTOMATIQUE

A comprehensive study of Convergent and Commutative Replicated Data Types

Marc Shapiro, INRIA & LIP6, Paris, France
Nuno Preguiça, CITI, Universidade Nova de Lisboa, Portugal
Carlos Baquero, Universidade do Minho, Portugal

Marek Zawirski, INRIA & UPMC, Paris, France

CRDT Flavors

- **Convergent:** State
  - Weak messaging requirements
- **Commutative:** Operations
  - Reliable broadcast required
  - Causal ordering sufficient

Convergent CRDTs

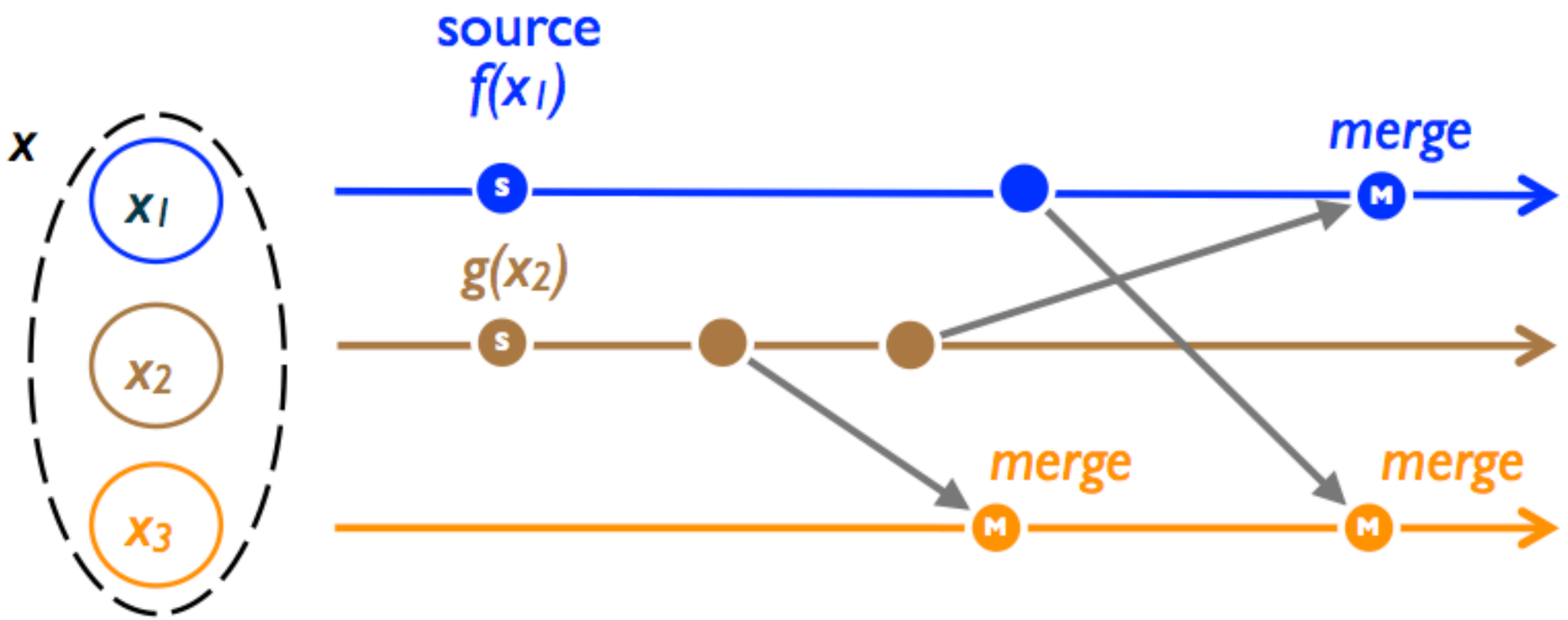


Figure 4: State-based replication

Commutative CRDTs

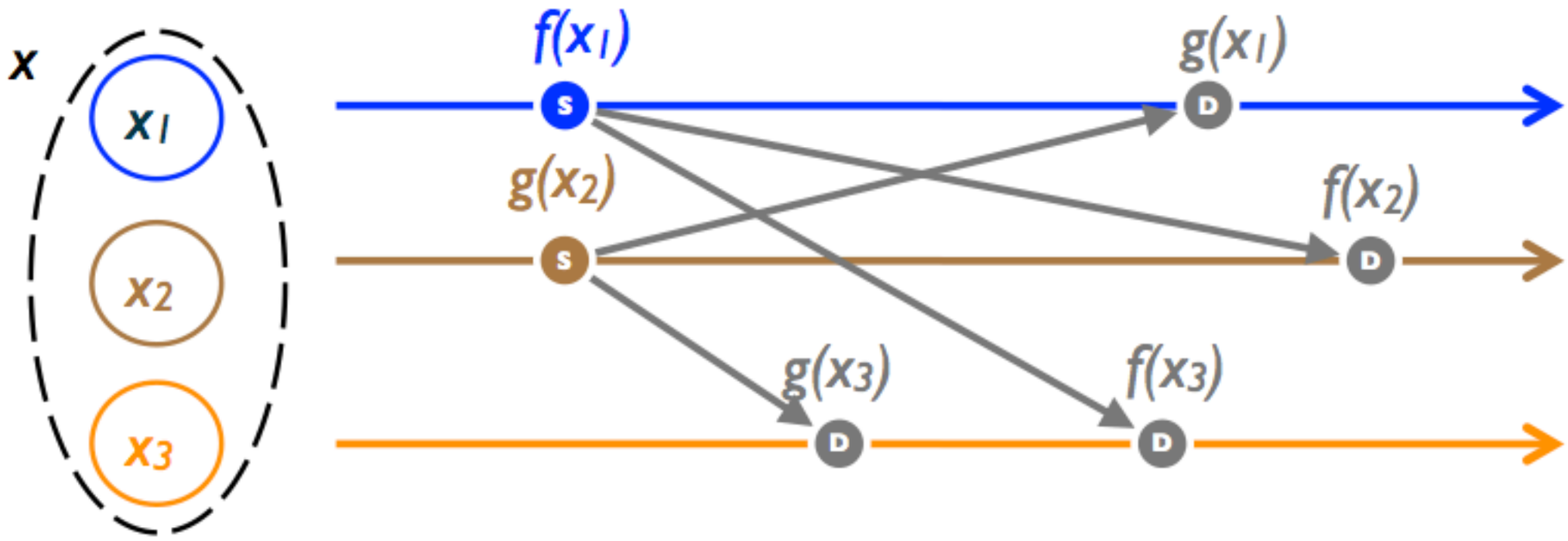


Figure 6: Operation-Based Replication

Registers

A place to put your stuff

Registers

- Last-Write Wins (LWW-Register)
  - e.g. Columns in Cassandra
- Multi-Valued (MV-Register)
  - e.g. Objects (values) in Riak

Counters

Keeping tabs

G-Counter

G-Counter

```
// Starts empty
[]
```

G-Counter

```
// Starts empty
[]

// A increments twice, forwarding state
[{a,1}]    // == 1
[{a,2}]    // == 2
```

G-Counter

```
// Starts empty
[]

// A increments twice, forwarding state
[{a,1}]    // == 1
[{a,2}]    // == 2

                    // B increments
                    [{b,1}]    // == 1
```

G-Counter

```
// Starts empty
[]

// A increments twice, forwarding state
[{a,1}]    // == 1
[{a,2}]    // == 2

                                  // B increments
                                  [{b,1}]    // == 1

// Merging
[{a,2}, {b,1}]                    [{a,1}, {b,1}]
```

G-Counter

```
// Starts empty
[]

// A increments twice, forwarding state
[{a,1}]    // == 1
[{a,2}]    // == 2

                                  // B increments
                                  [{b,1}]    // == 1

// Merging
[{a,2}, {b,1}]                    [{a,1}, {b,1}]
                                  [{a,2}, {b,1}]
// == 3, converged
```

PN-Counter

```
// A PN-Counter
{
   P = [{a,10},{b,2}],
   N = [{a,1},{c,5}]
}
// == (10+2)-(1+5) == 12-6 == 6
```

Sets

Members Only

G-Set

G-Set

```
// Starts empty
{}
```

G-Set

```
// Starts empty
{}

// A adds a and b, forwarding state
{a}
{a,b}
```

G-Set

```
// Starts empty
{}

// A adds a and b, forwarding state
{a}
{a,b}
```

```
// B adds c
{c}
```

G-Set

```
// Starts empty
{}

// A adds a and b, forwarding state
{a}
{a,b}
```

```
// B adds c
{c}
```

```
// Merging
{a,b,c}

// converged
```

```
{a,c}
{a,b,c}
```

2P-Set

2P-Set

```
// Starts empty
{A={},R={}}
```

2P-Set

```
// Starts empty
{A={},R={}}

// A adds a and b, forwarding state,
// removes a
{A={a},  R={}}     // == {a}
{A={a,b},R={}}   // == {a,b}
{A={a,b},R={a}}  // == {b}
```

2P-Set

```
// Starts empty
{A={},R={}}

// A adds a and b, forwarding state,
// removes a
{A={a},  R={}}     // == {a}
{A={a,b},R={}}   // == {a,b}
{A={a,b},R={a}}  // == {b}

                   // B adds c
                   {A={c},R={}} // == {c}
```

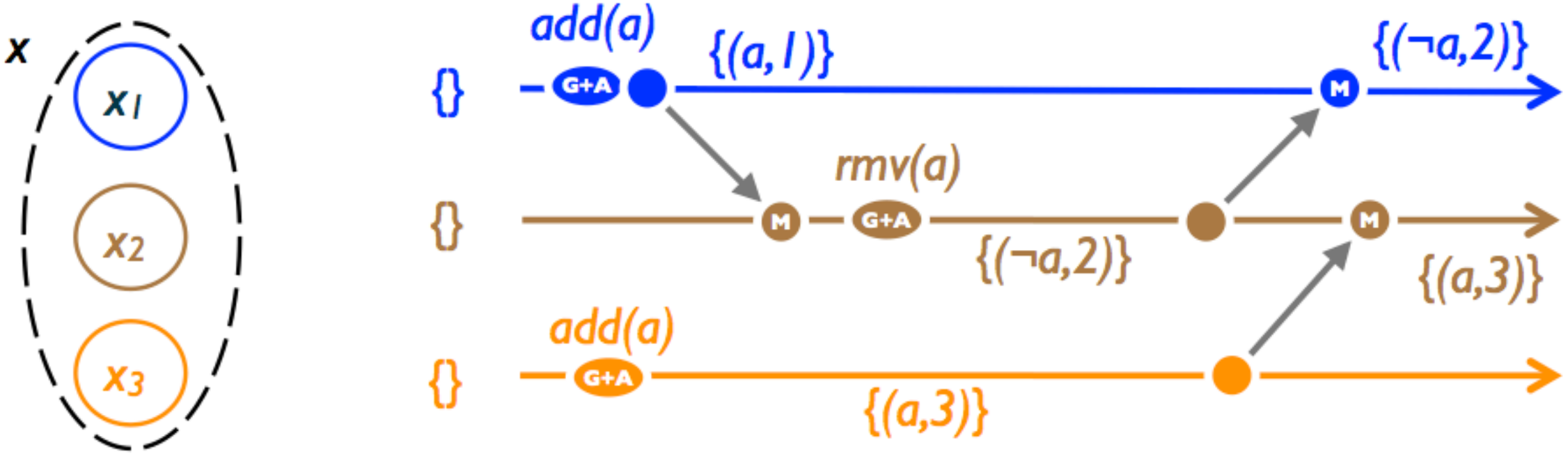
2P-Set

```
// Starts empty
{A={},R={}}

// A adds a and b, forwarding state,
// removes a
{A={a},  R={}}      // == {a}
{A={a,b},R={}}    // == {a,b}
{A={a,b},R={a}}  // == {b}

                                // B adds c
                                {A={c},R={}} // == {c}
// Merging
{A={a,b,c},R={a}}     {A={a,c},  R={}}
                                {A={a,b,c},R={}}
                                {A={a,b,c},R={a}}
// converged == {b,c}
```

LWW-Element-Set



OR-Set

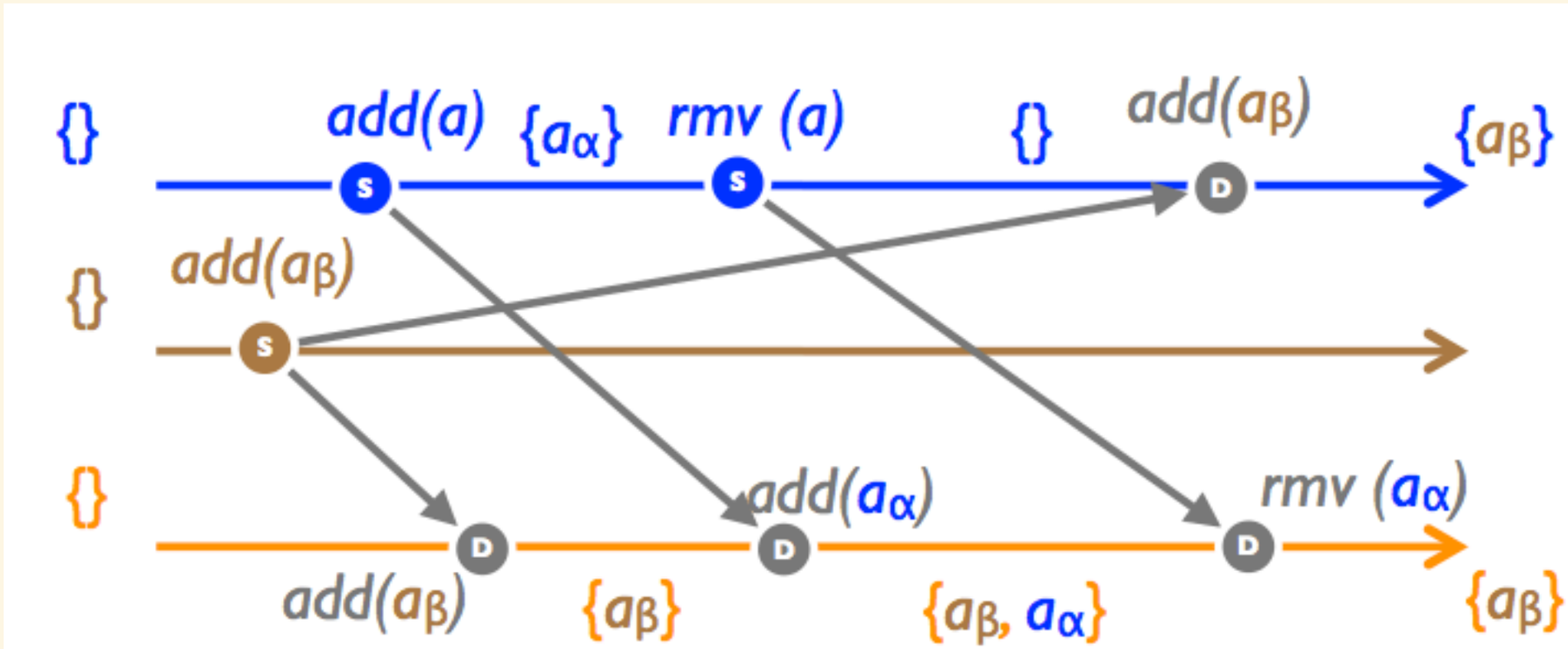
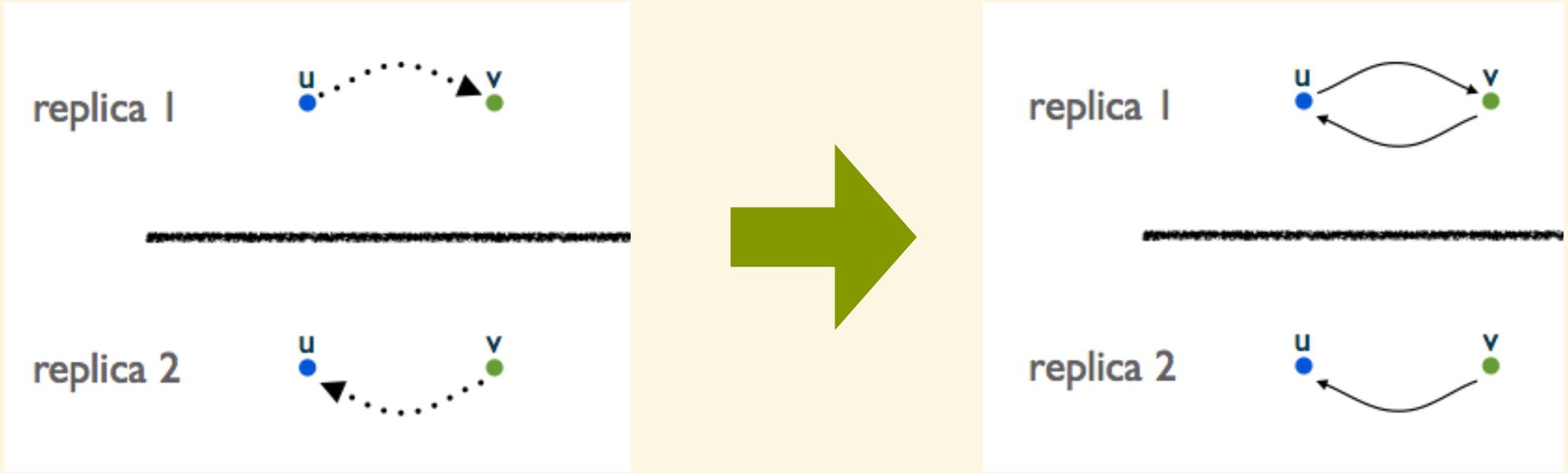


Figure 14: Observed-Remove Set (op-based)

Graphs

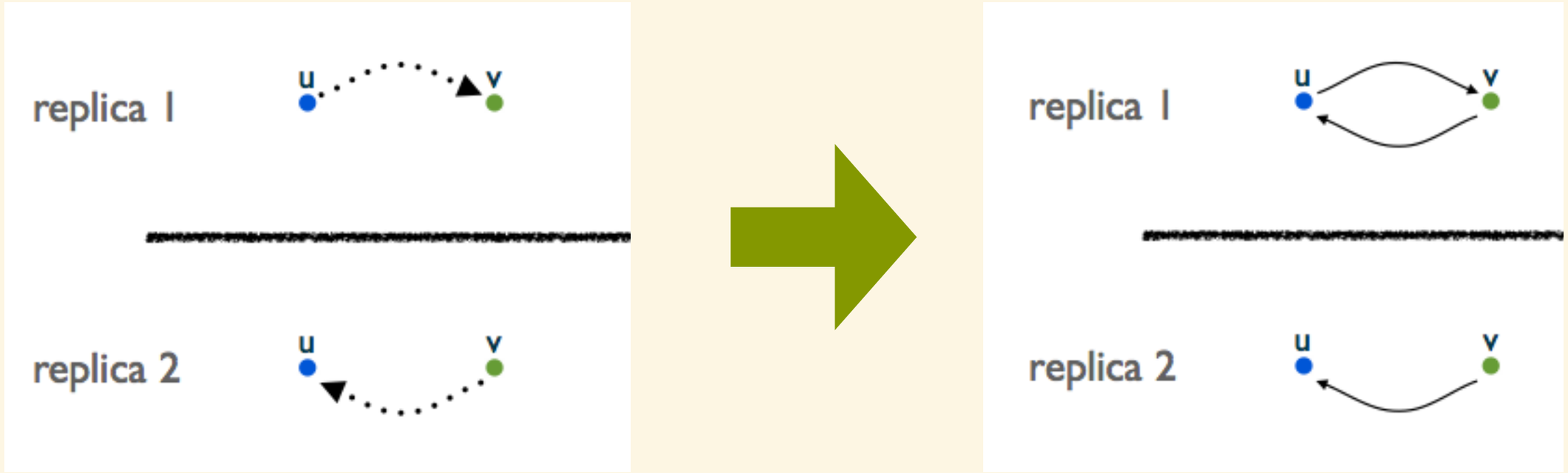
$$G = (V,E)$$
$$E \subseteq V \times V$$



Graphs

$G = (V,E)$

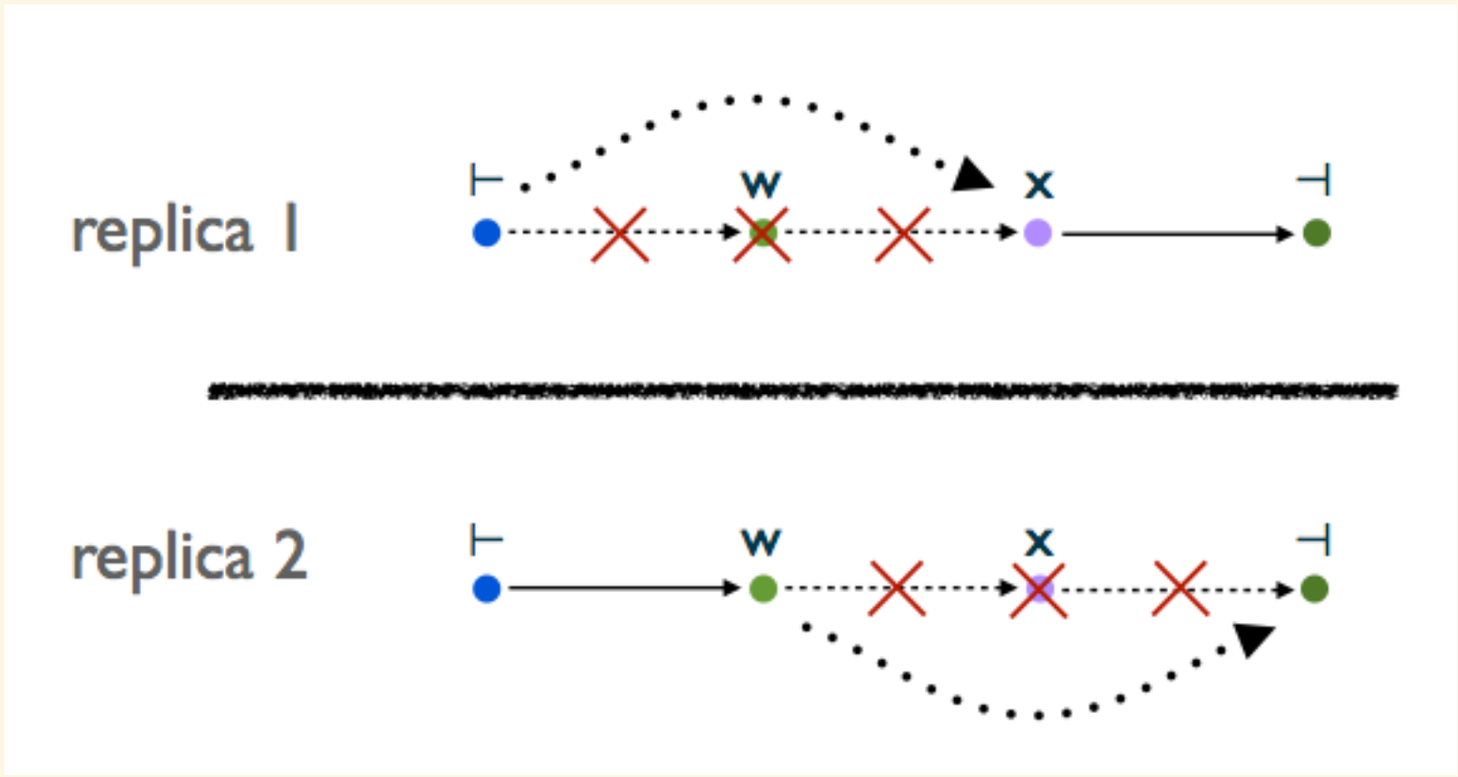
$E \subseteq V \times V$



Graphs

$G = (V,E)$

$E \subseteq V \times V$



Use-Cases

- Social graph (OR-Set or a Graph)
- Web page visits (G-Counter)
- Shopping Cart (Modified OR-Set)
- “Like” button (U-Set)

Challenges: GC

- CRDTs are inefficient
- Synchronization may be required

Challenges: Responsibility

- Client
  - Erlang: `mochi/statebox`
  - Clojure: `reiddraper/knockbox`
  - Ruby: `aphyr/meangirls`, `bkerley/hanover`
- Server
  - Very few options, Riak soon

Thanks