Eventually-Consistent Data Structures

Sean Cribbs
@seancribbs #CRDT
Berlin Buzzwords 2012

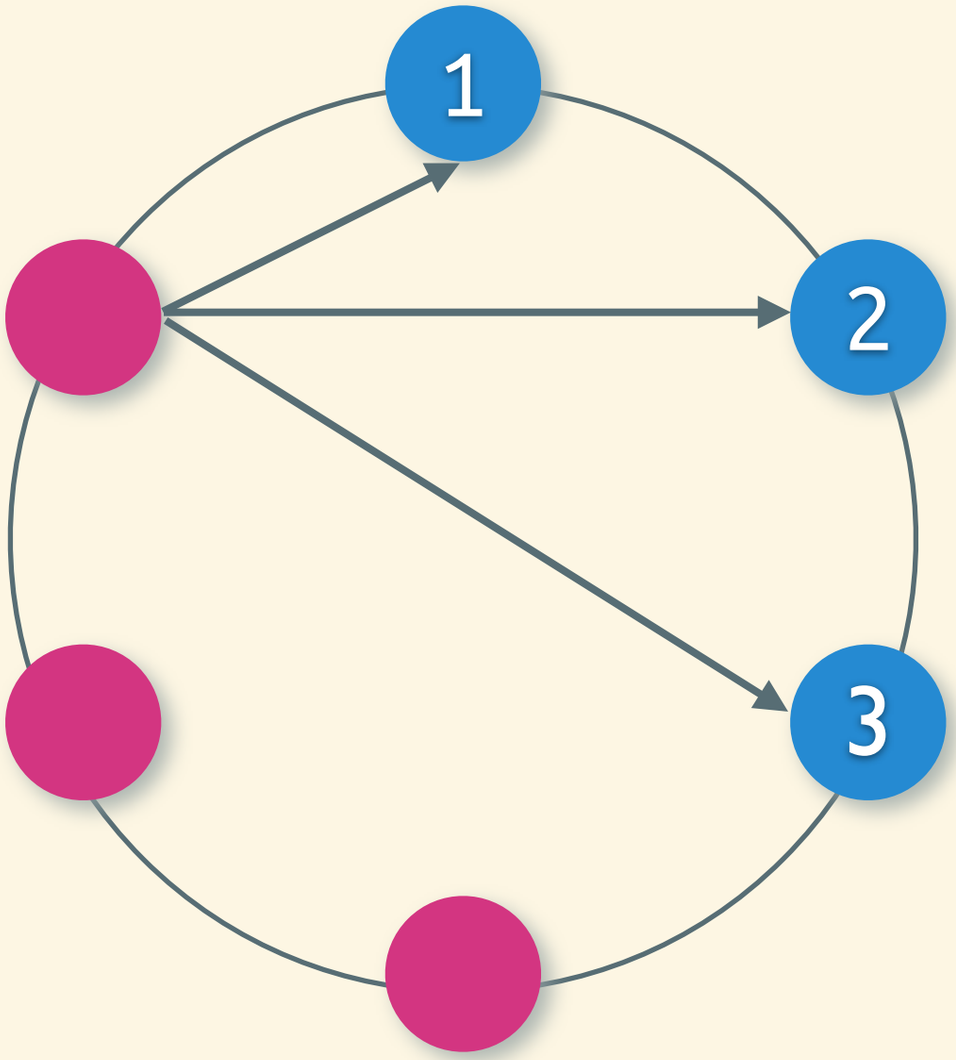
I work for Basho

We make riak

Riak is Eventually Consistent

So are Voldemort and Cassandra

Eventual Consistency



Replicated

Loose coordination

Forward progression

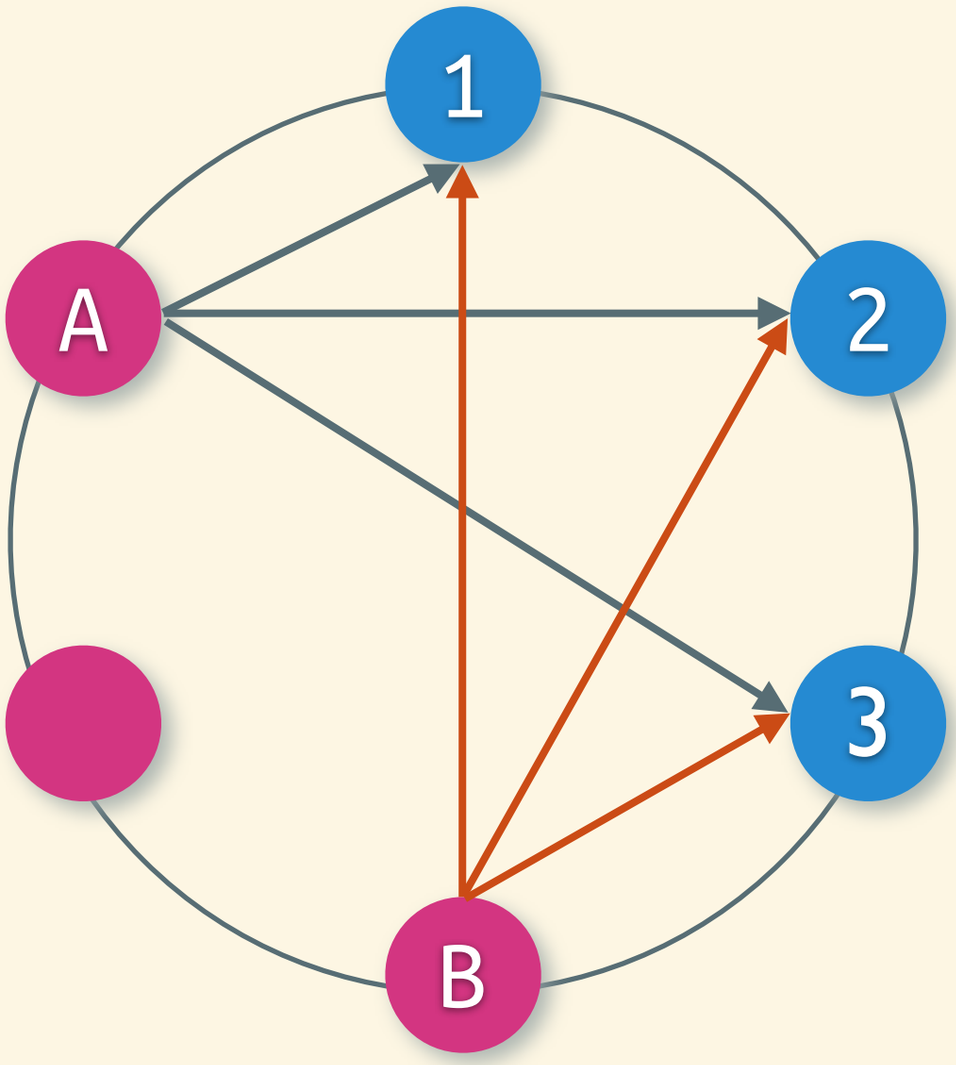
Eventual is Good

✔ Fault-tolerant

✔ Highly available

✔ Low-latency

Consistency?

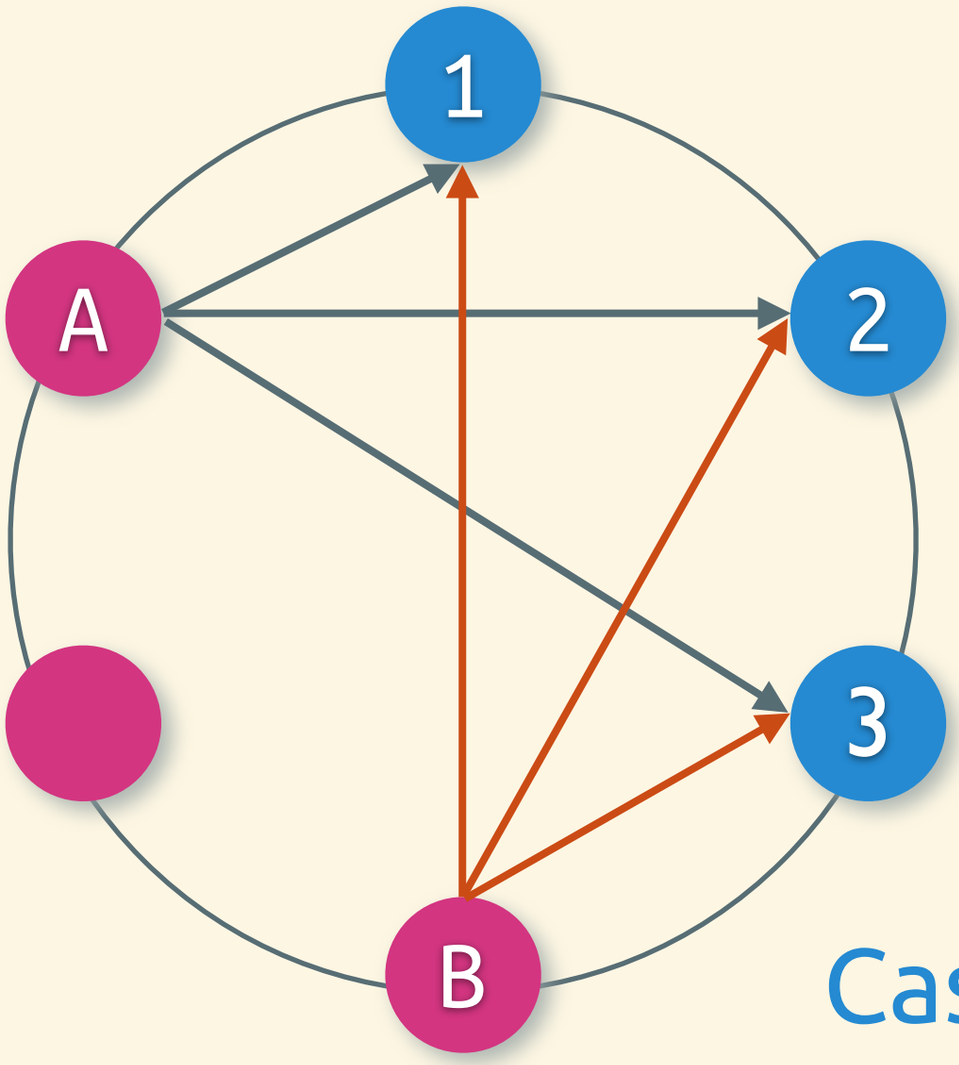


No clear winner!

Throw one out?

Keep both?

Consistency?



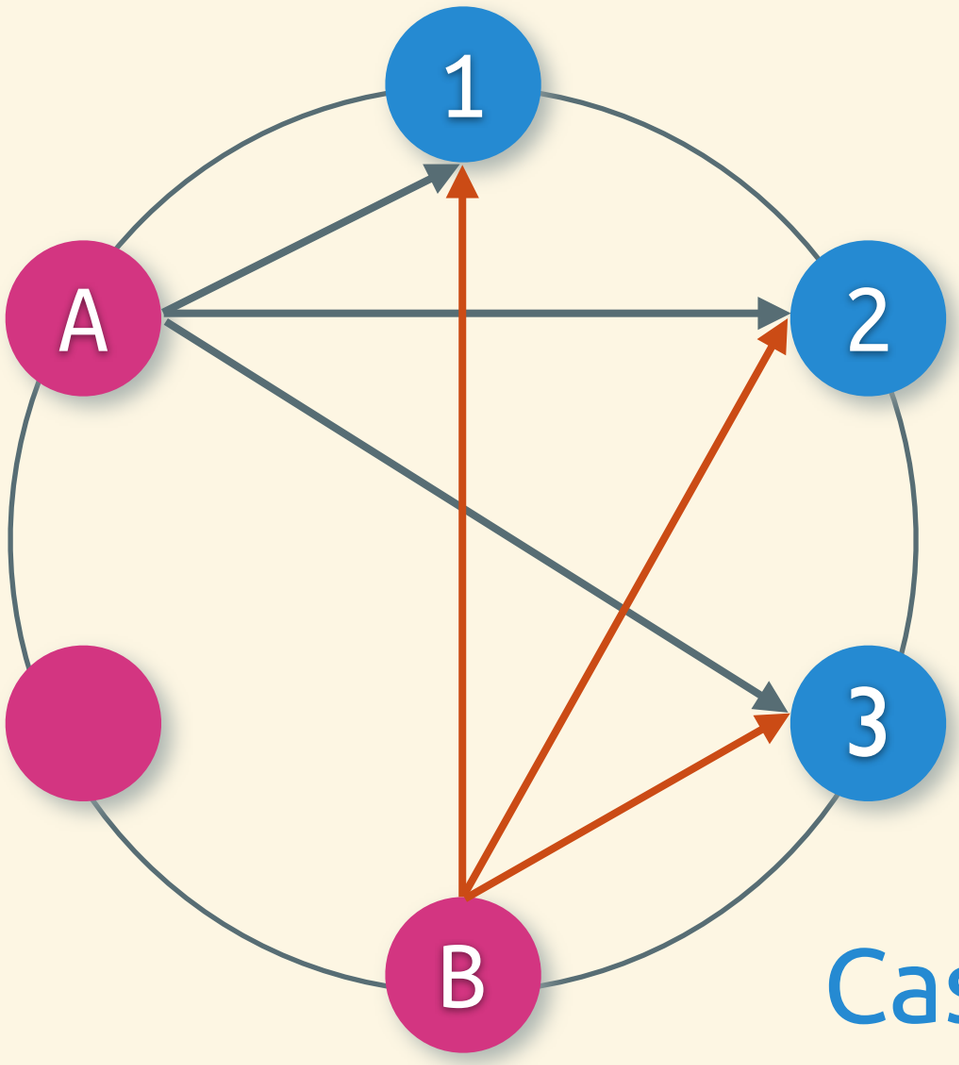
No clear winner!

Throw one out?

Keep both?

Cassandra

Consistency?



No clear winner!

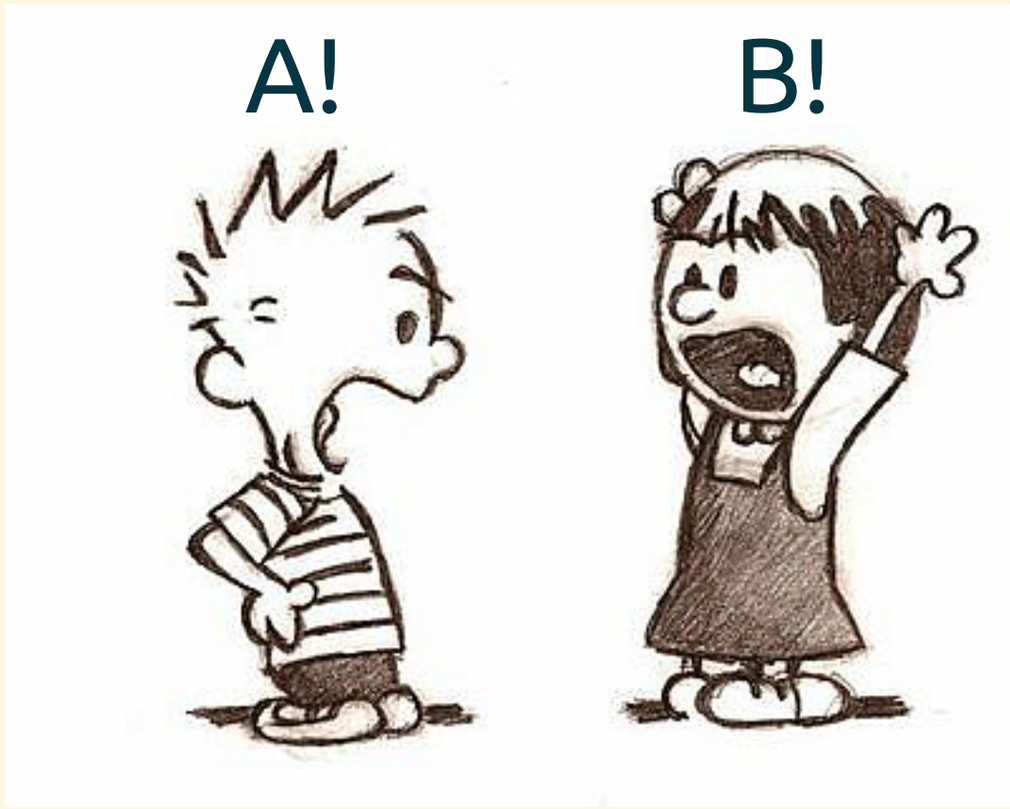
Throw one out?

Keep both?

Cassandra

Riak & Voldemort

Conflicts!



Now what?

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

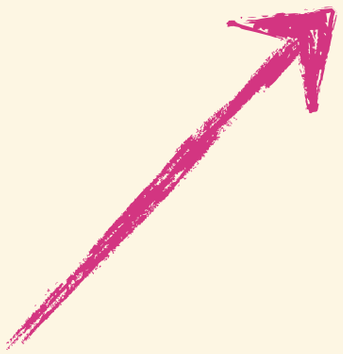
"Ad hoc approaches have proven brittle and error-prone"

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

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

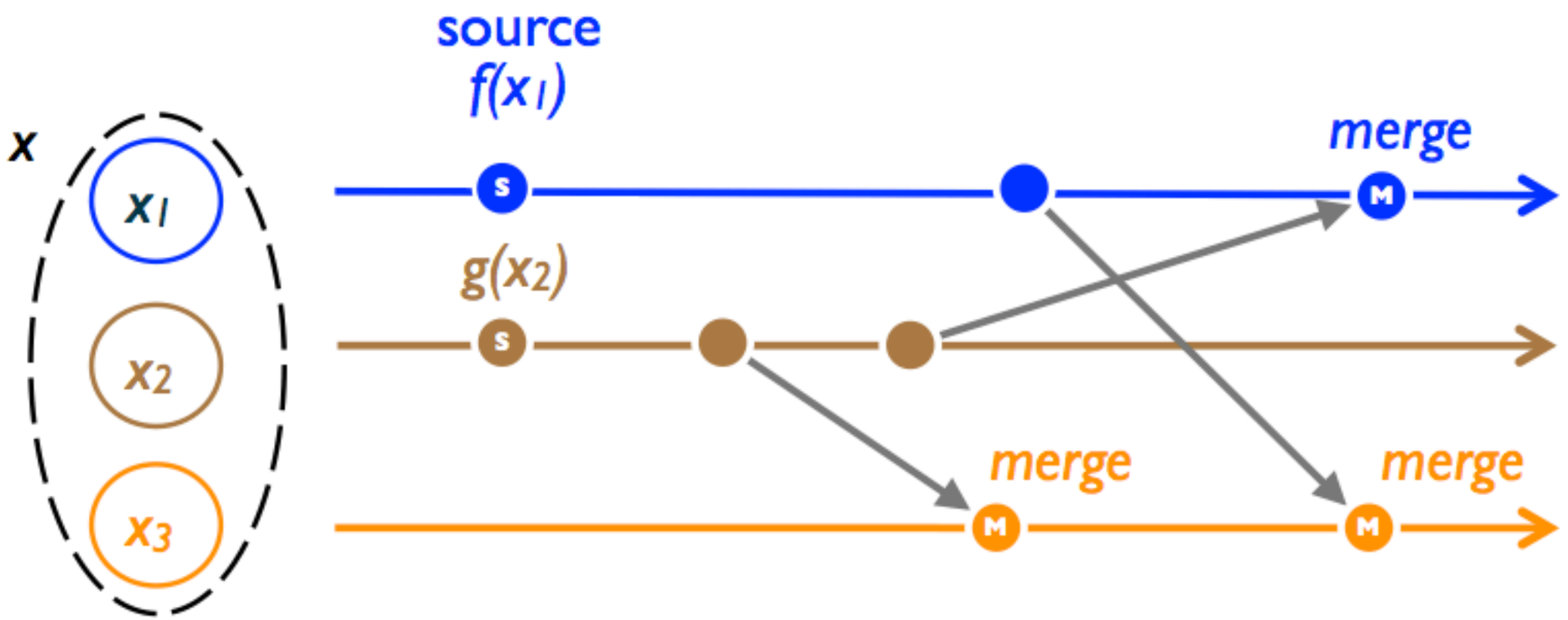


Figure 4: State-based replication

Commutative CRDTs

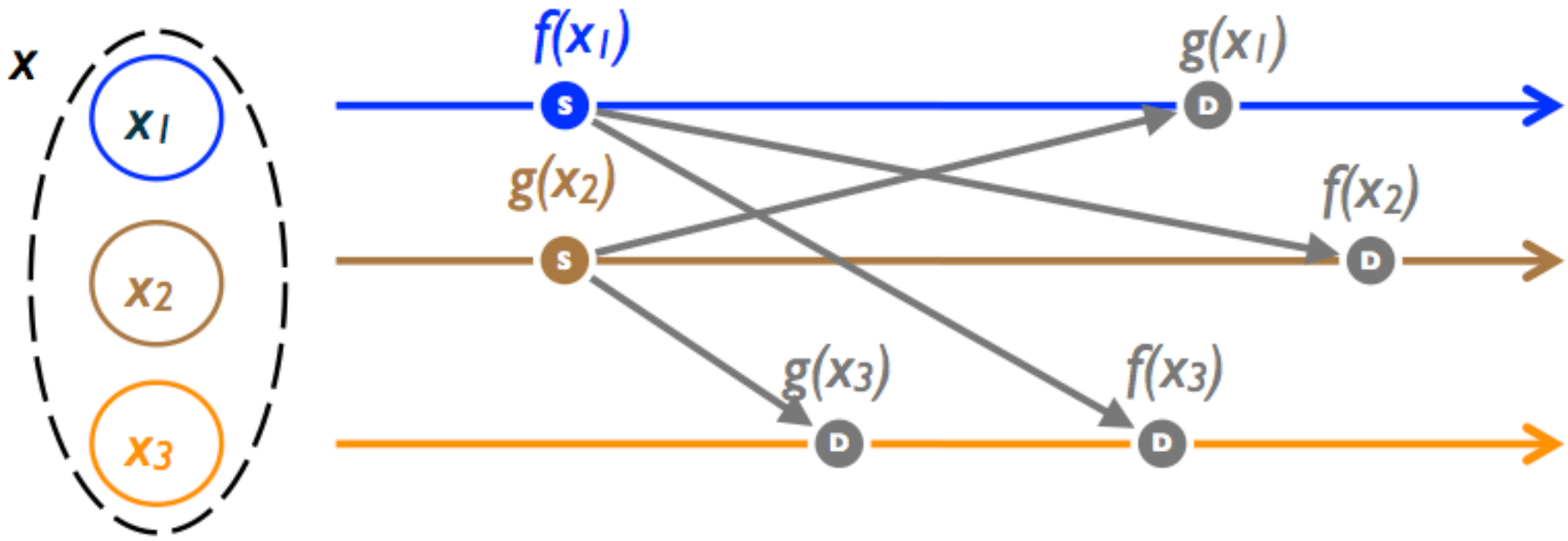


Figure 6: Operation-Based Replication

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}
```

```
{a,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}
```

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
```

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
```

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={}}
```

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}}
```

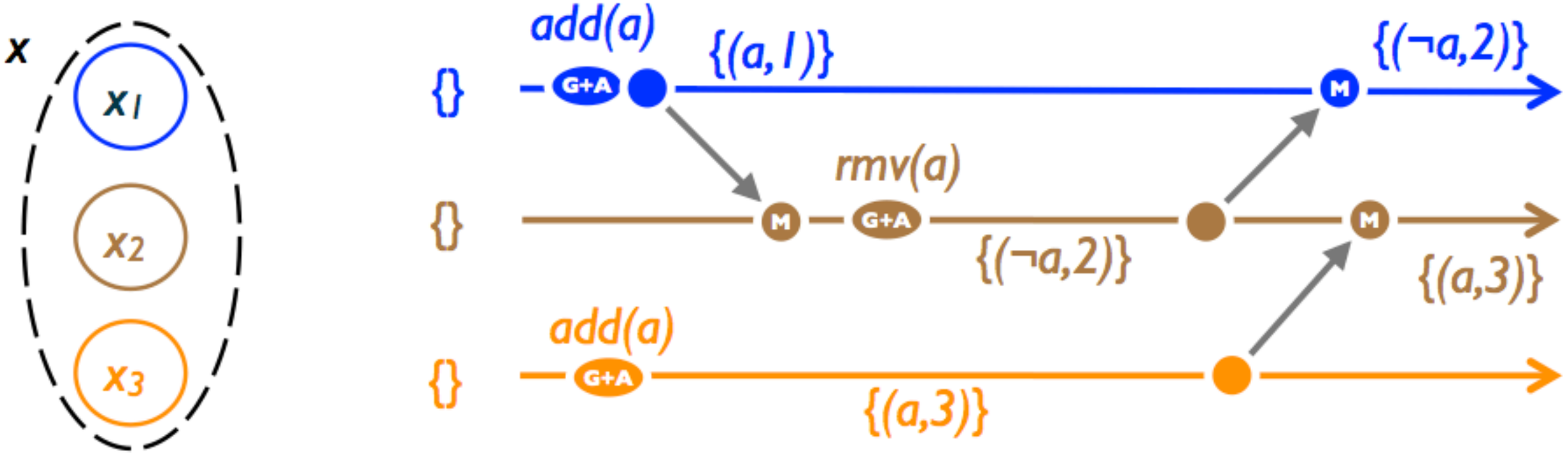
2P-Set

```
// 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

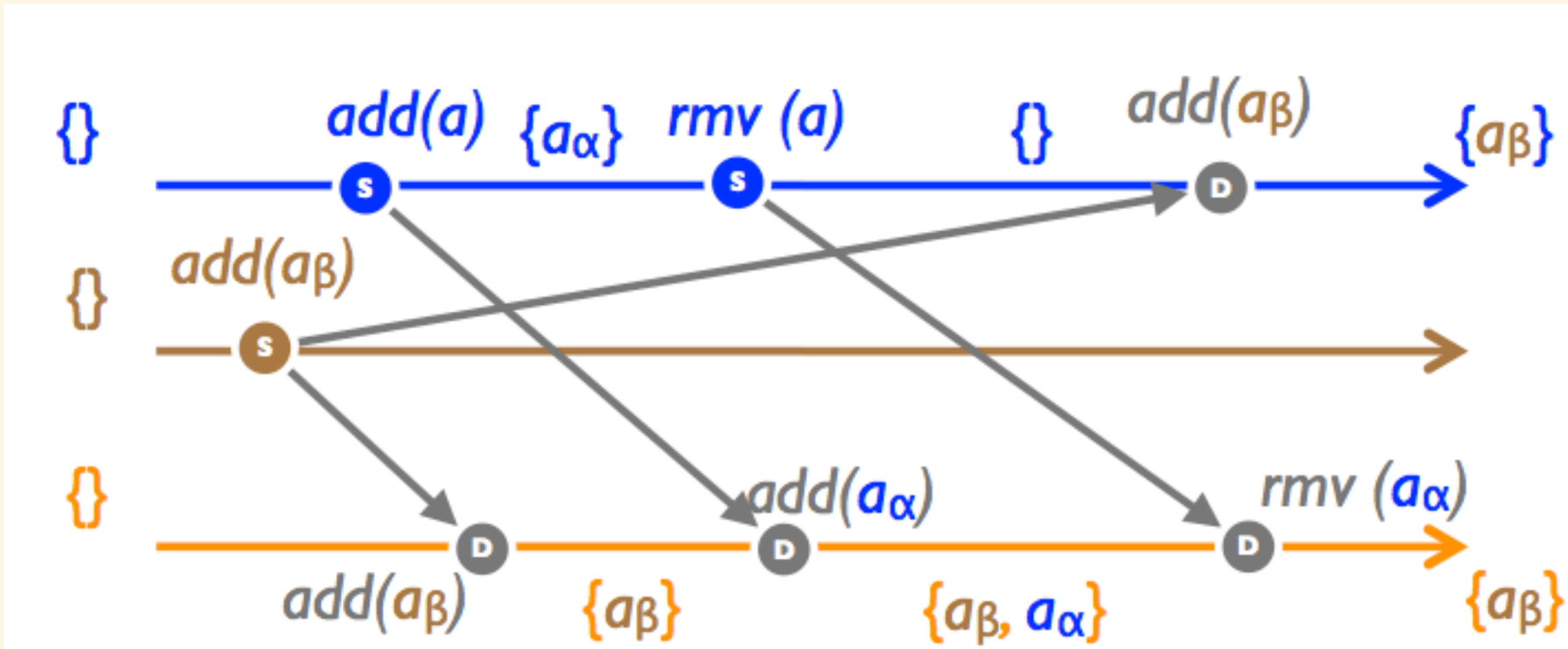
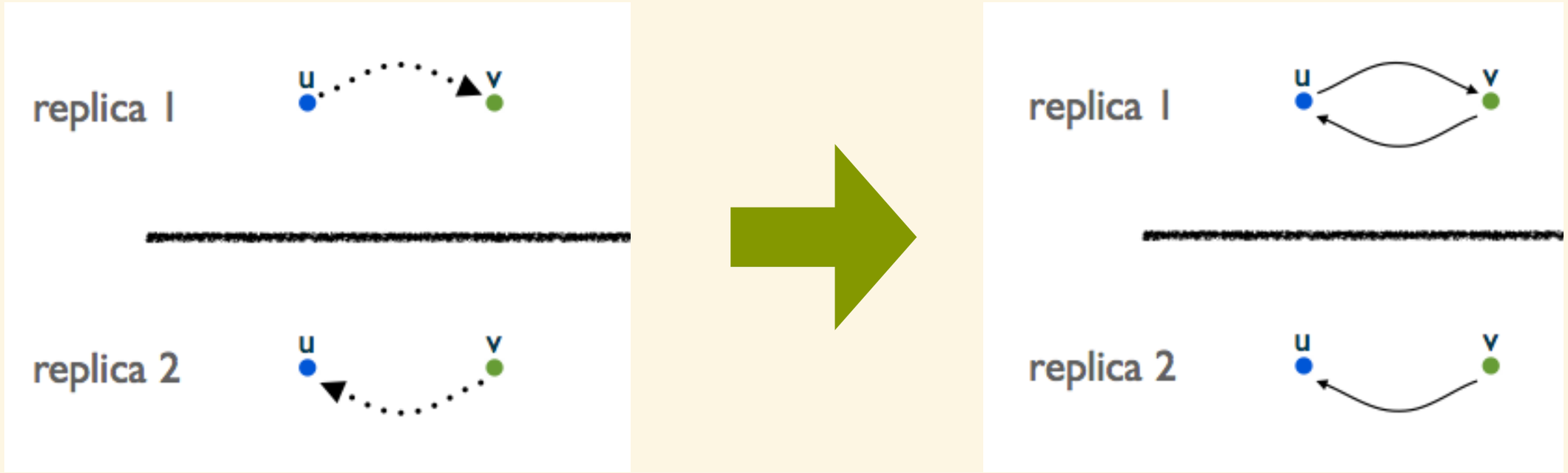


Figure 14: Observed-Remove Set (op-based)

Graphs

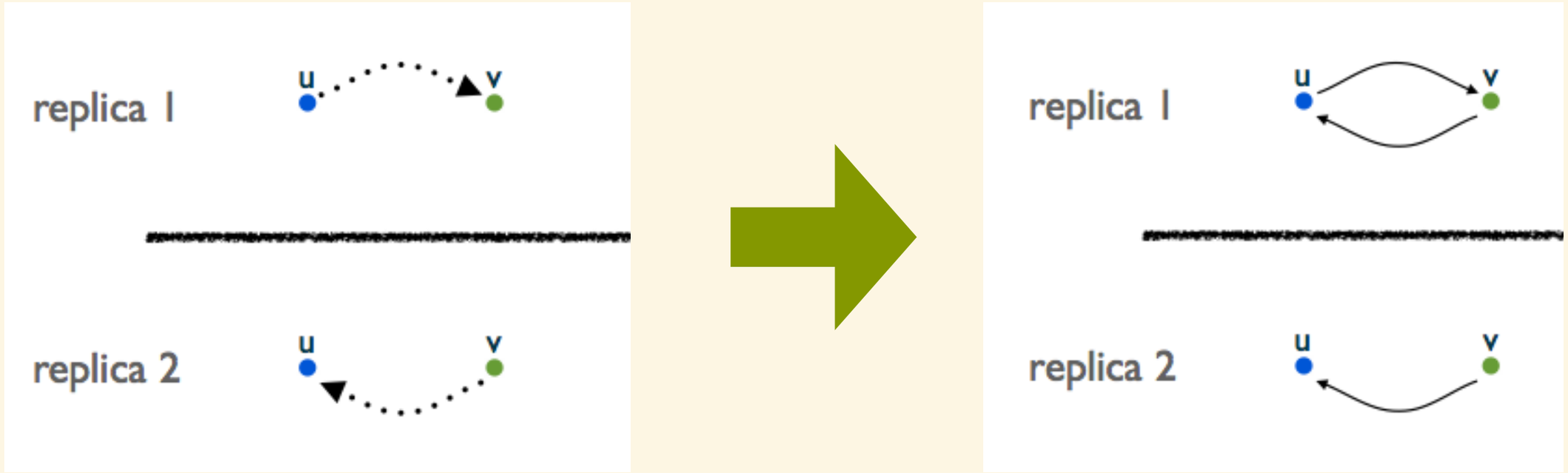
$$G = (V,E)$$
$$E \subseteq V \times V$$



Graphs

$G = (V,E)$

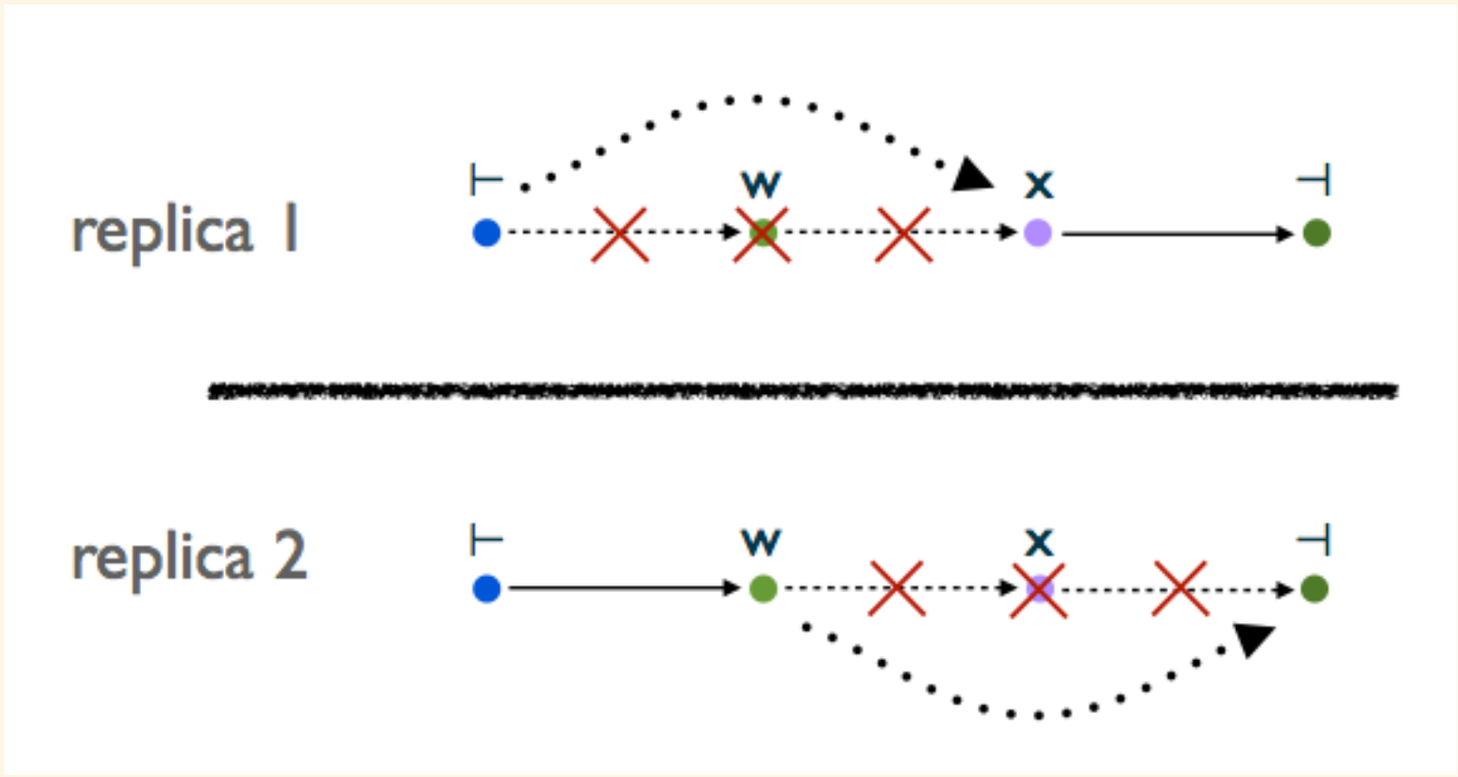
$E \subseteq V \times V$



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`
- Server
  - Very few options

Thanks