Outline

Lattice Agreement

Ordering issues

Generalized Lattice Agreement

Nullifying semantics

Sagar Chordia
With {Kapil, Rama, Kaushik}
PLATO - MSRI

Commutative Replicated Data-types

Applications

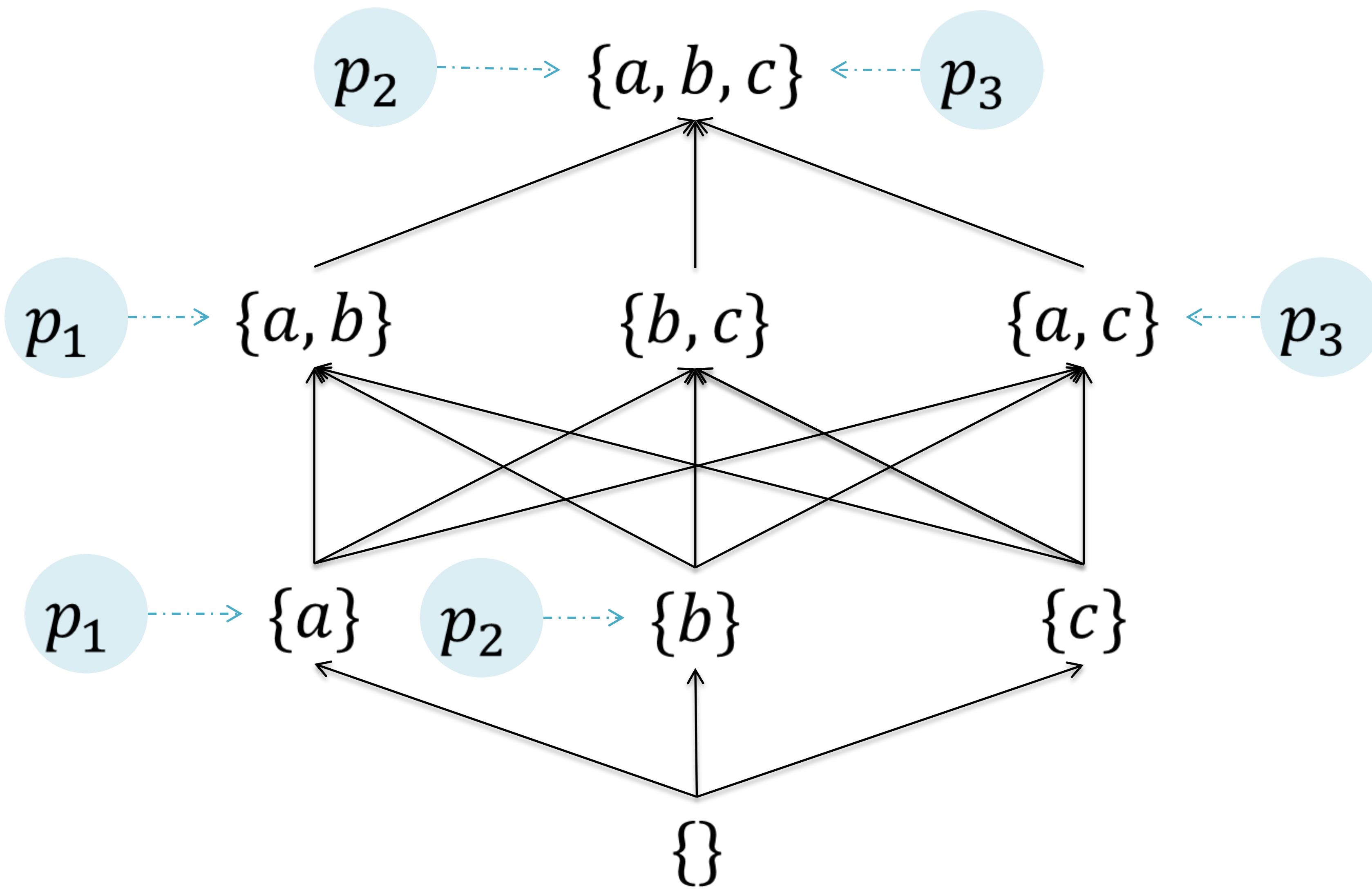
Lattice agreement

- N processes.
- Each process starts with an input value

  $u_i \in (L, \leq, V)$ a finite join semi lattice.

  – $\leq$ is partial order
  – For any two elements $a, b \in L$. $a \vee b$ exists.
- Every non-faulty process outputs a value v_i
  – v_i is join of input values including its own.
  – Any two output values v_i and v_j are comparable i.e. either $v_i \leq v_j$ or $v_j \leq v_i$
  – Every correct process eventually outputs a value.

Lattice agreement



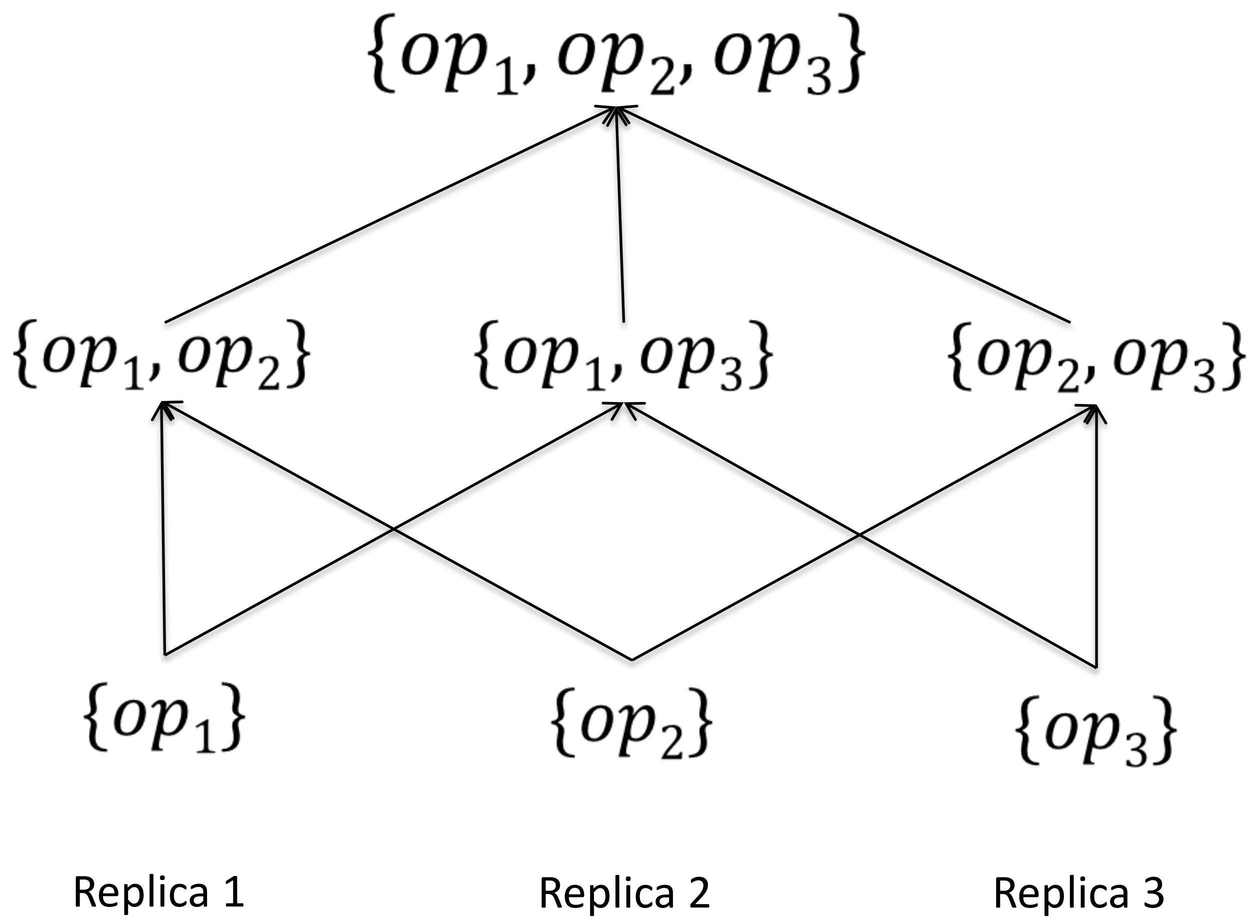
Generalized lattice agreement

- Generalization of lattice agreement
  - Process receives sequence of values u_i^j
  - Values belong to an infinite lattice.
- Processes output sequence of values v_i^j
  - Any two values are comparable.
  - Every value received by correct process is eventually included in an output value.
- Wait-free algorithm. $O(n)$ message delays.

State machine replication

- GLA can be used to implement a specific class of state machines
  - Two kinds of operations (a) updates, and (b) reads
  - All concurrent updates should *commute*
- Translation to GLA
  - Updates form a power-set lattice $(L, \leq, \vee)$
  - Replicas propose operations
  - Queries are serviced using states corresponding to output values.

SM with GLA

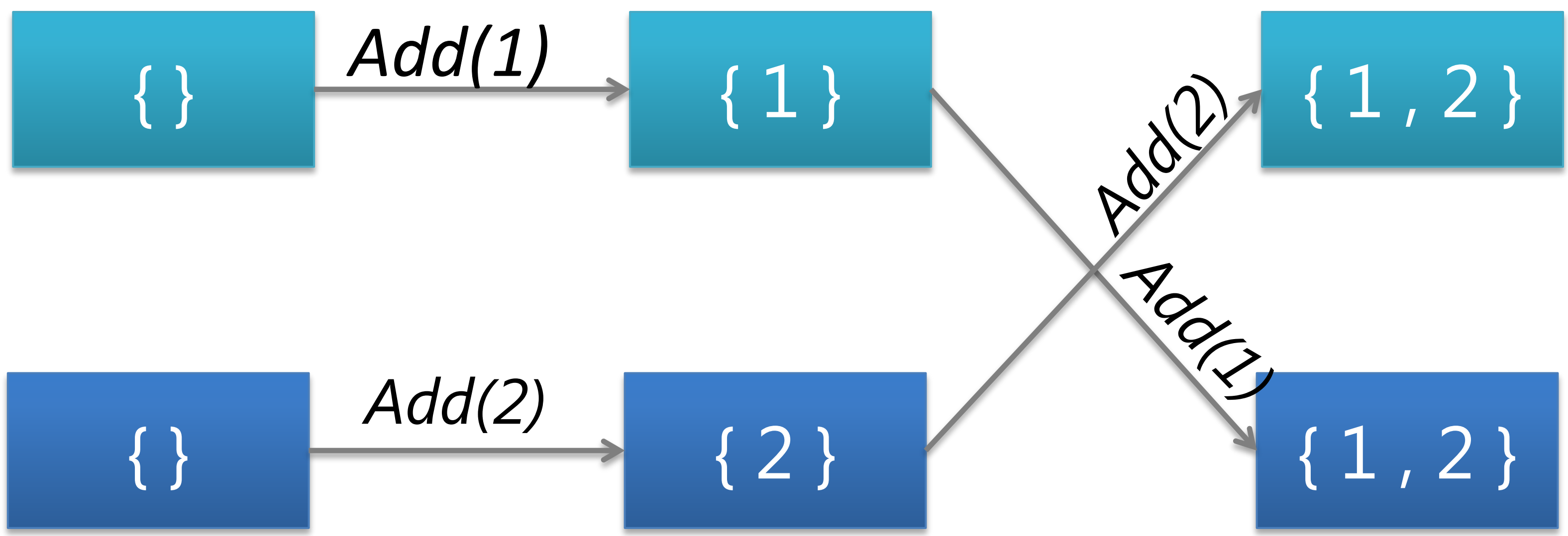

$\{op_1, op_2, op_3\}$
$\{op_1, op_2\}$
$\{op_1, op_3\}$
$\{op_2, op_3\}$
$\{op_1\}$
$\{op_2\}$
$\{op_3\}$
Replica 1
Replica 2
Replica 3

Guarantees

- Fault-tolerant model.
- Wait-free algorithm. $O(n)$ message delays.
- Strong consistencies
  - Sequential consistency
    - Serializability and program order of client operations is maintained.
  - Linearizability
    - Operations appear to execute instantaneously.

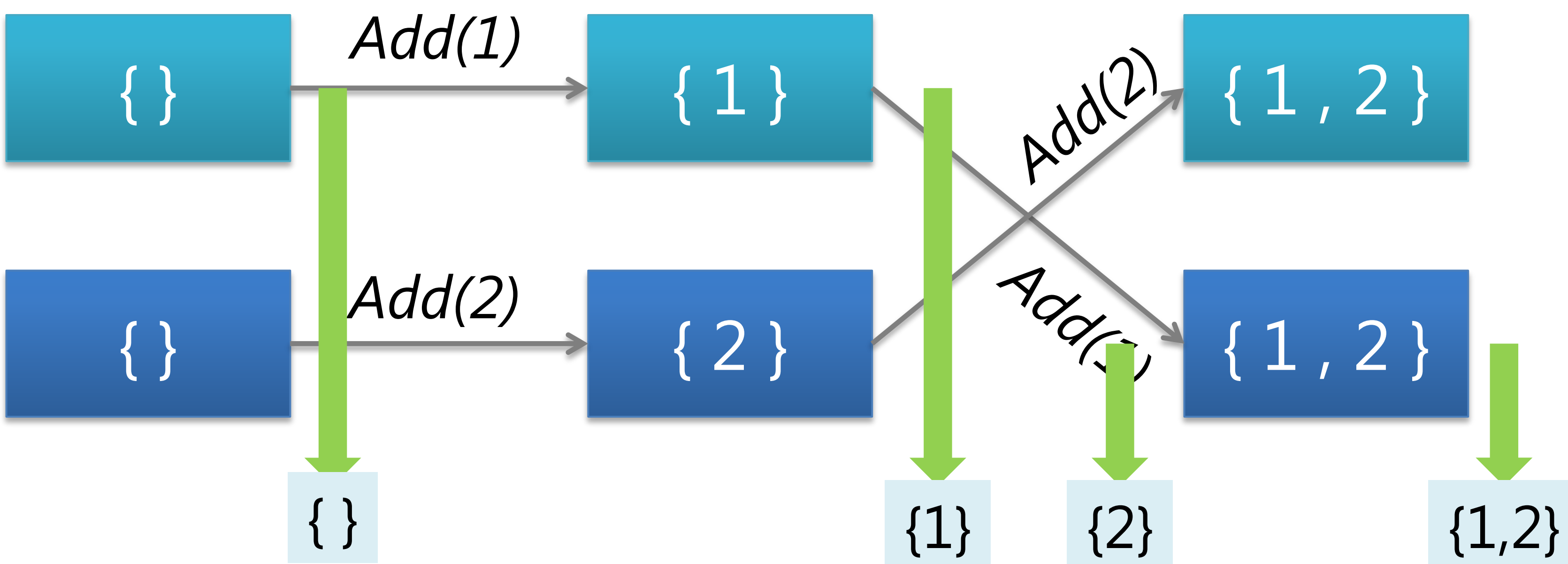
Data-types with GLA as SM

- Reads and updates are only operations.
- Updates are commutative
- Set with add() as only update operation.



Queries?

- Even if updates commute, queries may be inconsistent



- As GLA learns chain of values

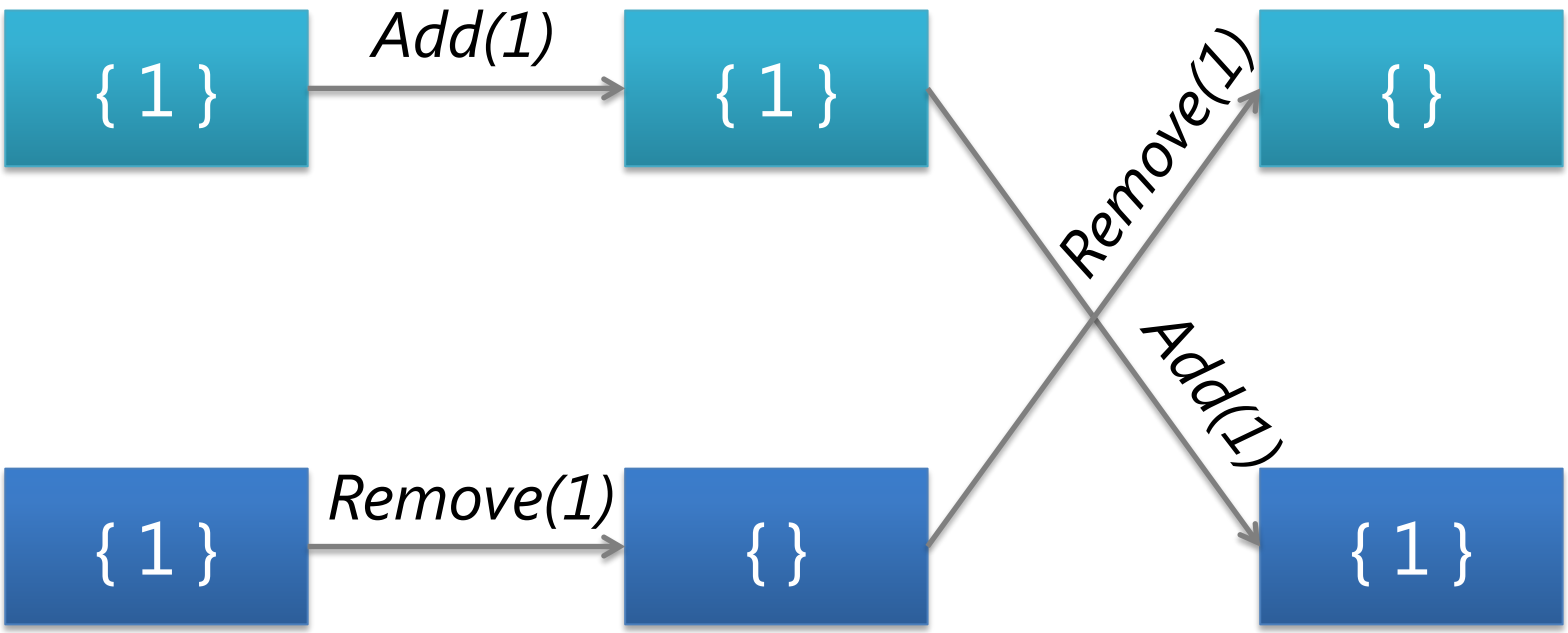
  {}, {2}, {1,2} ✓ {}, {1}, {1,2} ✓

Commutative and Convergent Replicated Data-Types

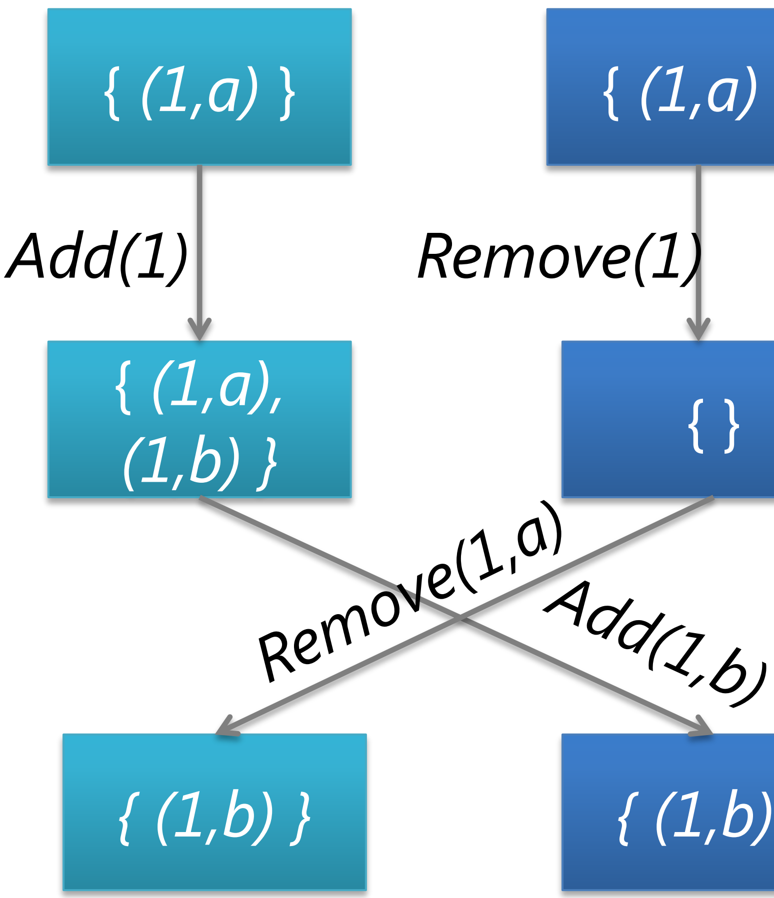
- Data types with inbuilt conflict resolution.
- Transforms non-commutative operations to commutative operations with certain tricks
- (Assumption)
  - All operations will reach all nodes eventually.
- (Guarantee)
  - All nodes will converge to same state.

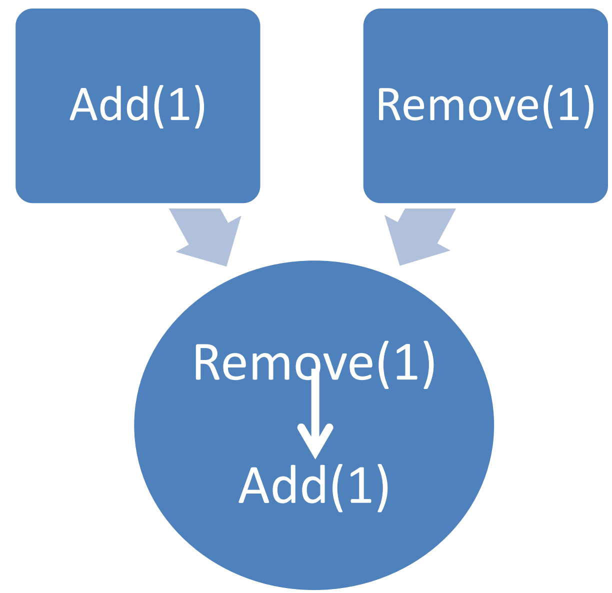
Replicated Set

- Remove() does not commute with add()



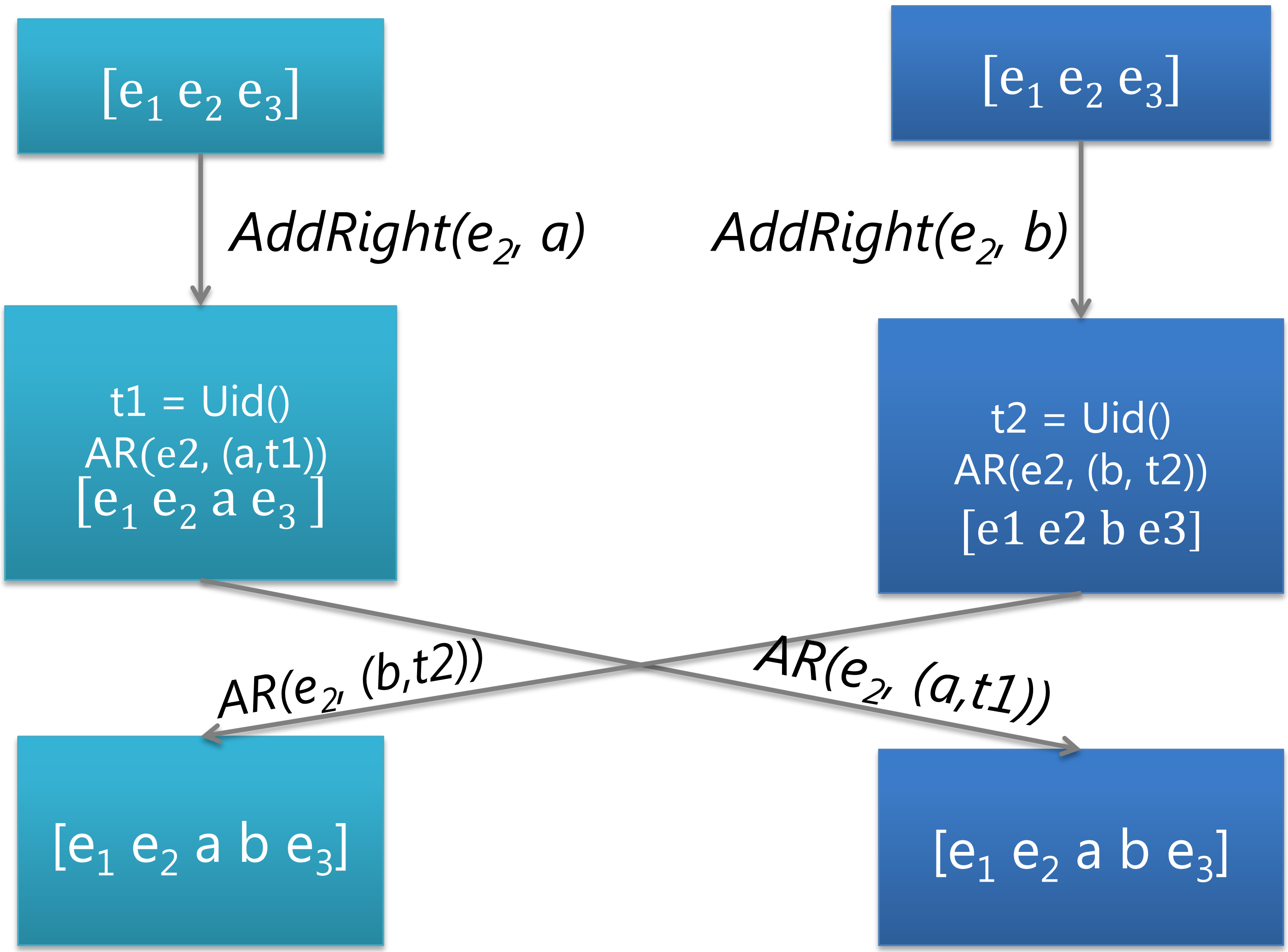
Observed Remove set

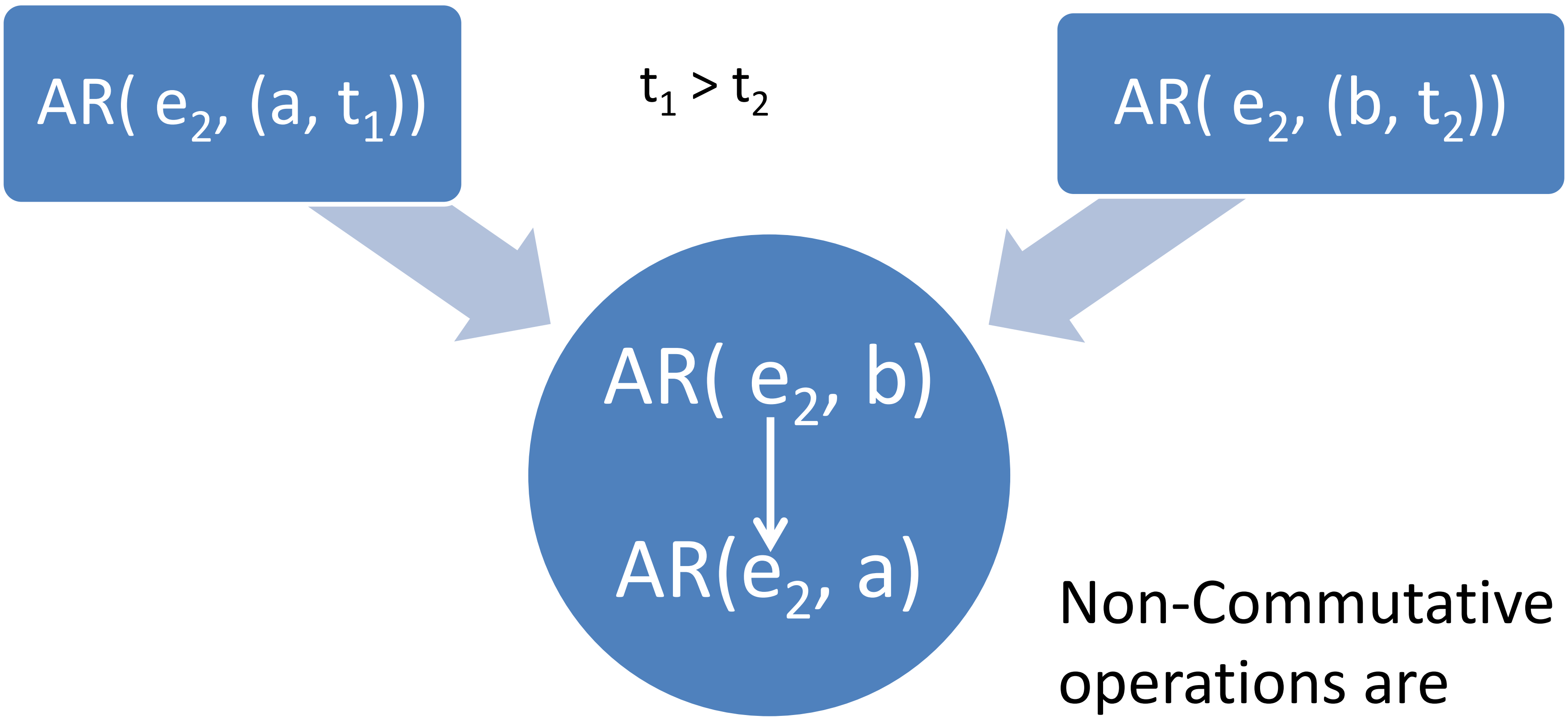




Orders non-commutative operations

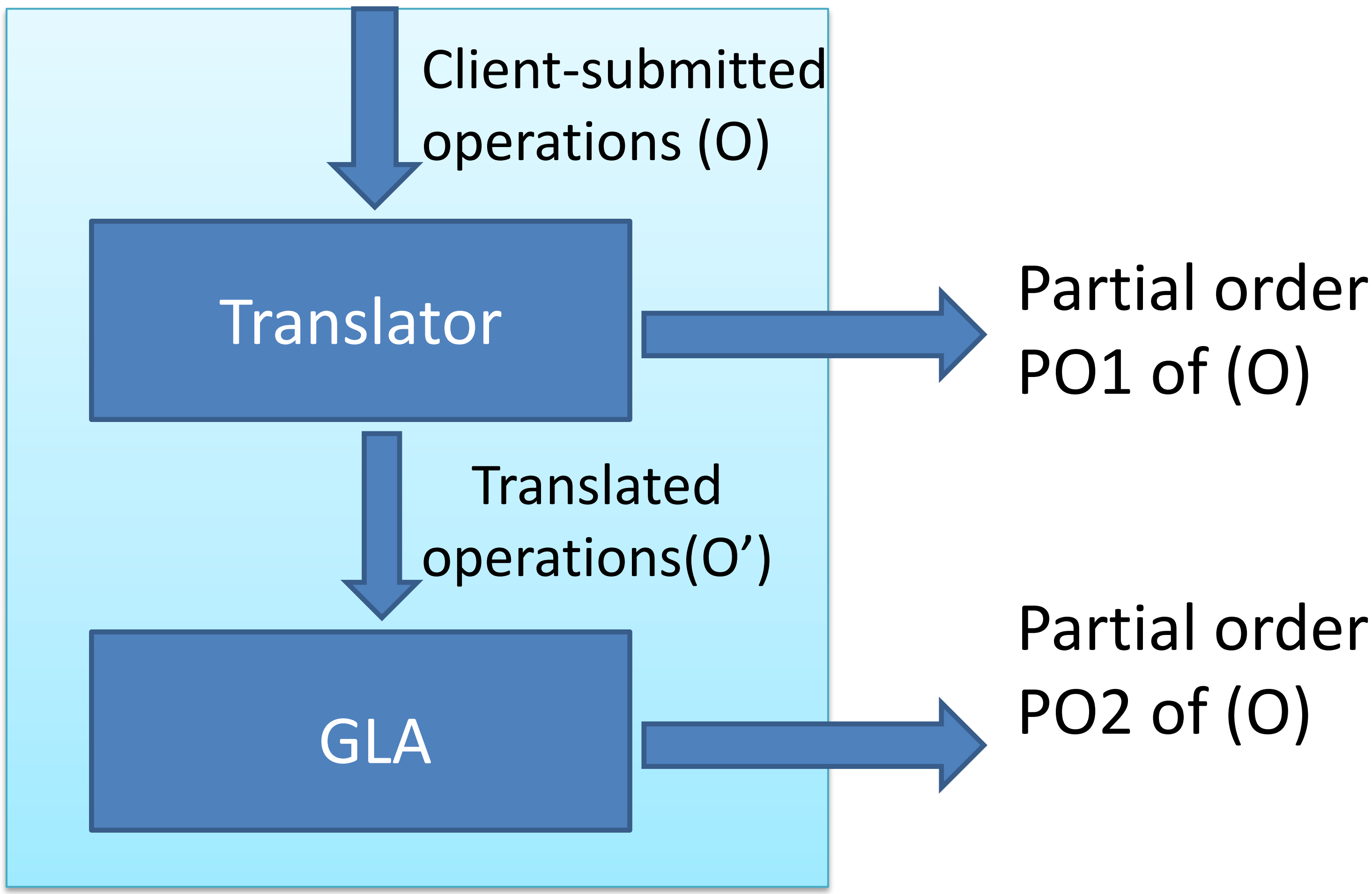
Replicated Growable array (RGA)





Non-Commutative operations are ordered

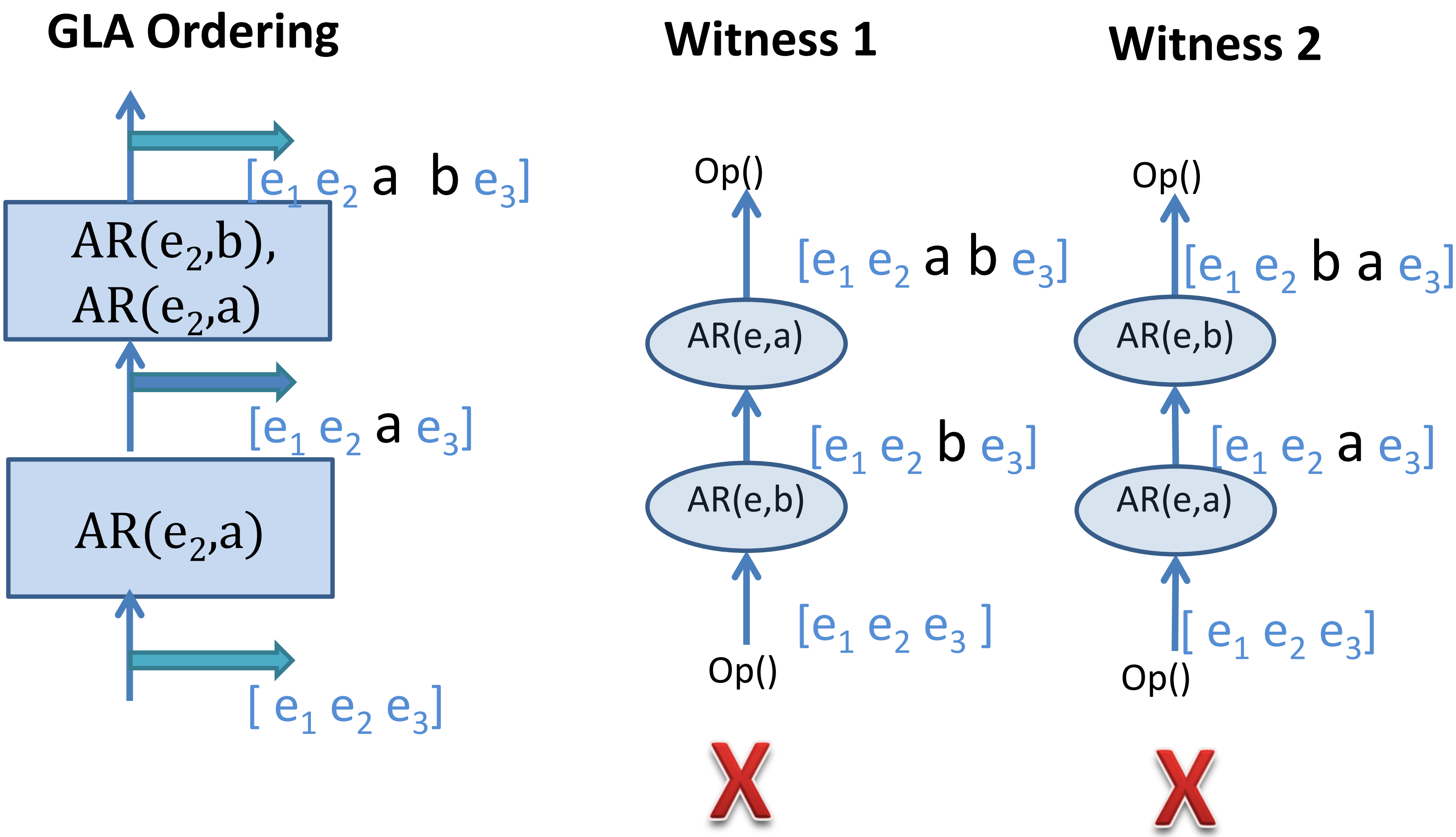
Big-picture



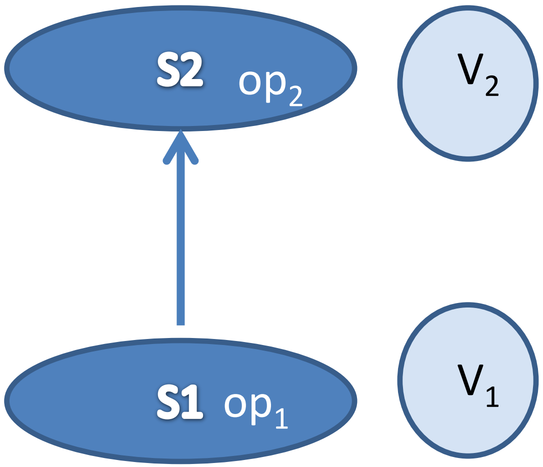
What if PO1 conflicts PO2?

Previous Example...

Sequence: [e_1 e_2 e_3]



Problem!



- op_1 and op_2 don't commute
- Deterministic ordering orders op_2 *_before_* op_1

- Effect:
  - v_1 exposes the effect of Op_1.
  - $Op_2 < Op_1$,
  - v_1 must also reflect the effect of op_2 (to guarantee consistency across values v_1 and v_2).
- In general, this could be difficult!

What extra conditions
we need for
strong consistency?

Solution

- Nullifying operations:
  - Op1(Op2(s)) = Op1(s) where s is the state.
  - Op1 is said to nullify Op2.

- In set Add(x), Remove(x) have nullifying property.
  - Add(x, Remove(x, s)) = Add(x,s)
  - Remove(x, Add(x, s)) = Remove(x,s)

- Sequence doesn't have this.
  - {AddRight(e,x), AddRight(e,y)} != {AddRight(e,y)}

- **Sufficient condition**: If all pairs of non-commutative operations have nullifying property then strong consistency is guaranteed.

Why nullifying works

Consider op1, op2 : two non-commutative operations.

{op1}

{op2}

Pre-deterministic ordering

{op1, op2} --> Read2

{op2} --> Read1

GLA ordering1

Witness: [op2, op1]

Read2

Read1

{op1, op2} --> Read2

{op1} --> Read1

GLA ordering2

Witness: [op2, op1]

Read1, Read2

Summary

- For data-types to have strong consistency
  - Commutative operations
    - Ordering decided by GLA.
  - Non-commutative operations
    - Some pre-deterministic ordering is decided.
    - All pairs of non-commutative operations should have nullifying semantics to obey this pre-defined order.

Characterization

- Data-types with nullifying semantics
  - Set (2P set, OR-set)
  - Graph(2P2P based, OR-set based).
  - Directed Acyclic graph
  - Key-value pair
  - Leaderboard
- Data-types which fail
  - Sequences (AddRight())
- **Can we do better**?

Thanks!

- References:
  - Generalized lattice agreement by *Jose, Sriram, Kaushik, Rama, Kapil*
  - CRDT by *Marc Shapiro*
  - Kapil's slides for MSR-summer school talk.
  - Windows 8 metro UI(Cover slide design)