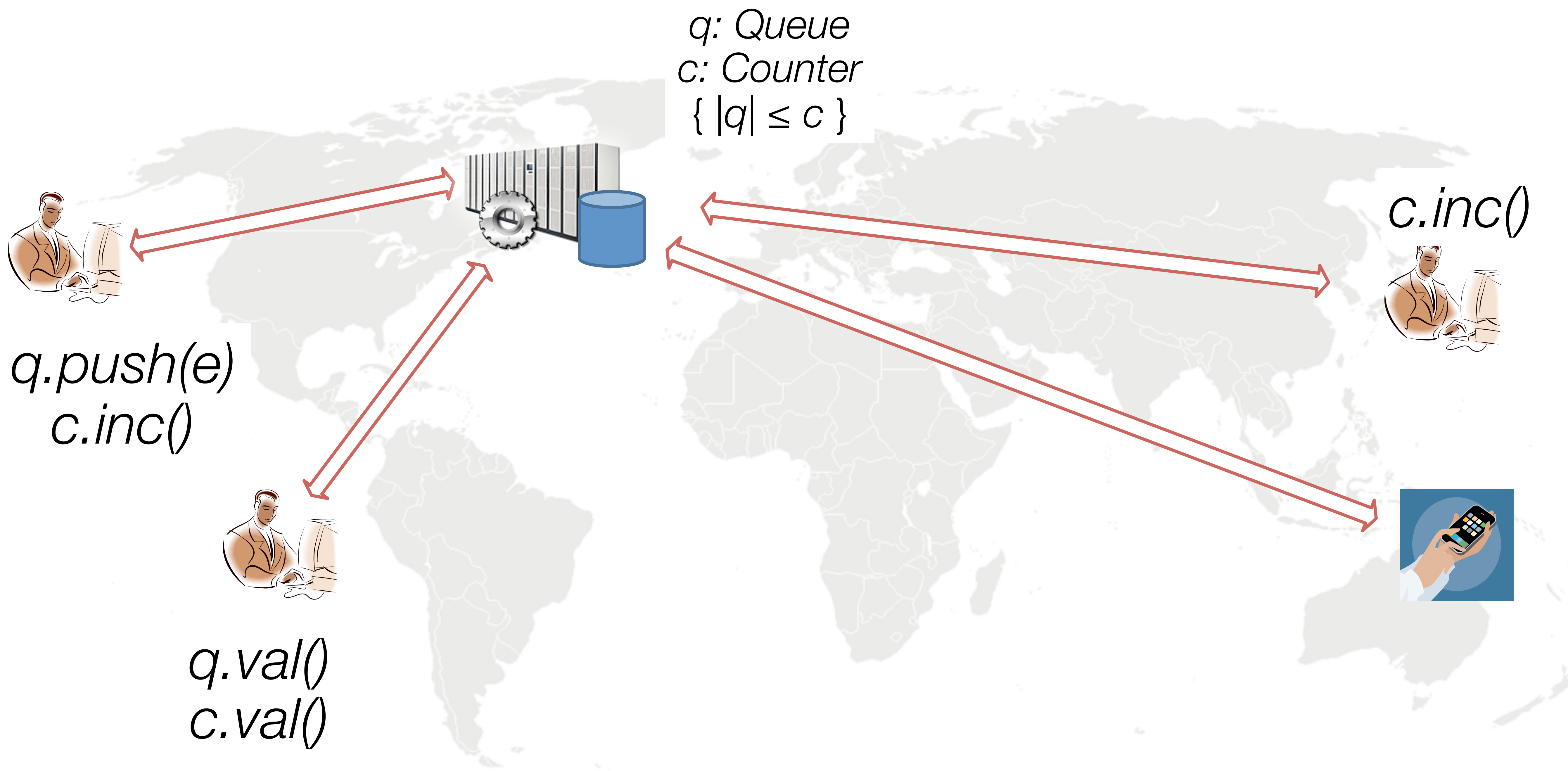
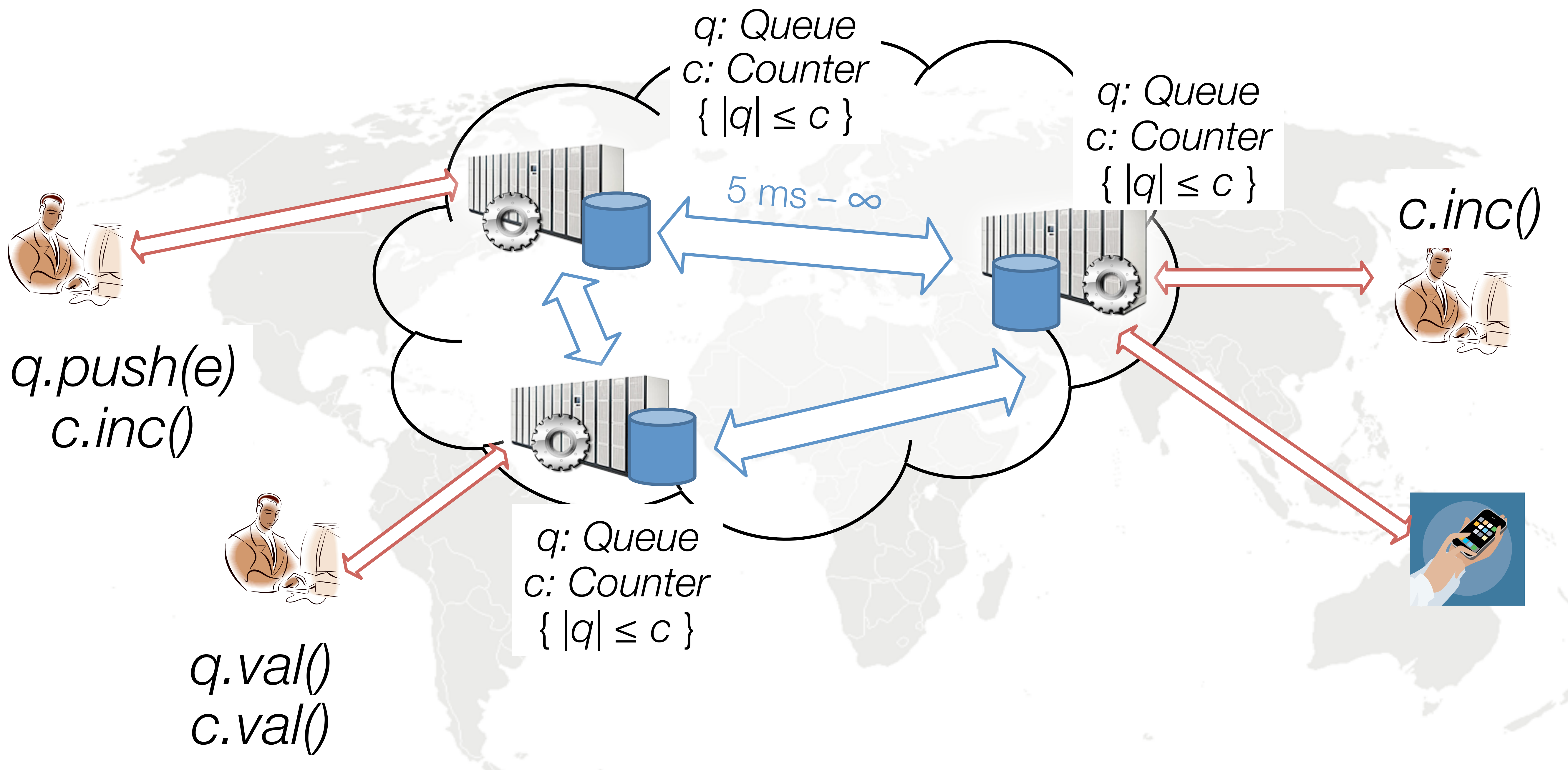
Distributed Consistency

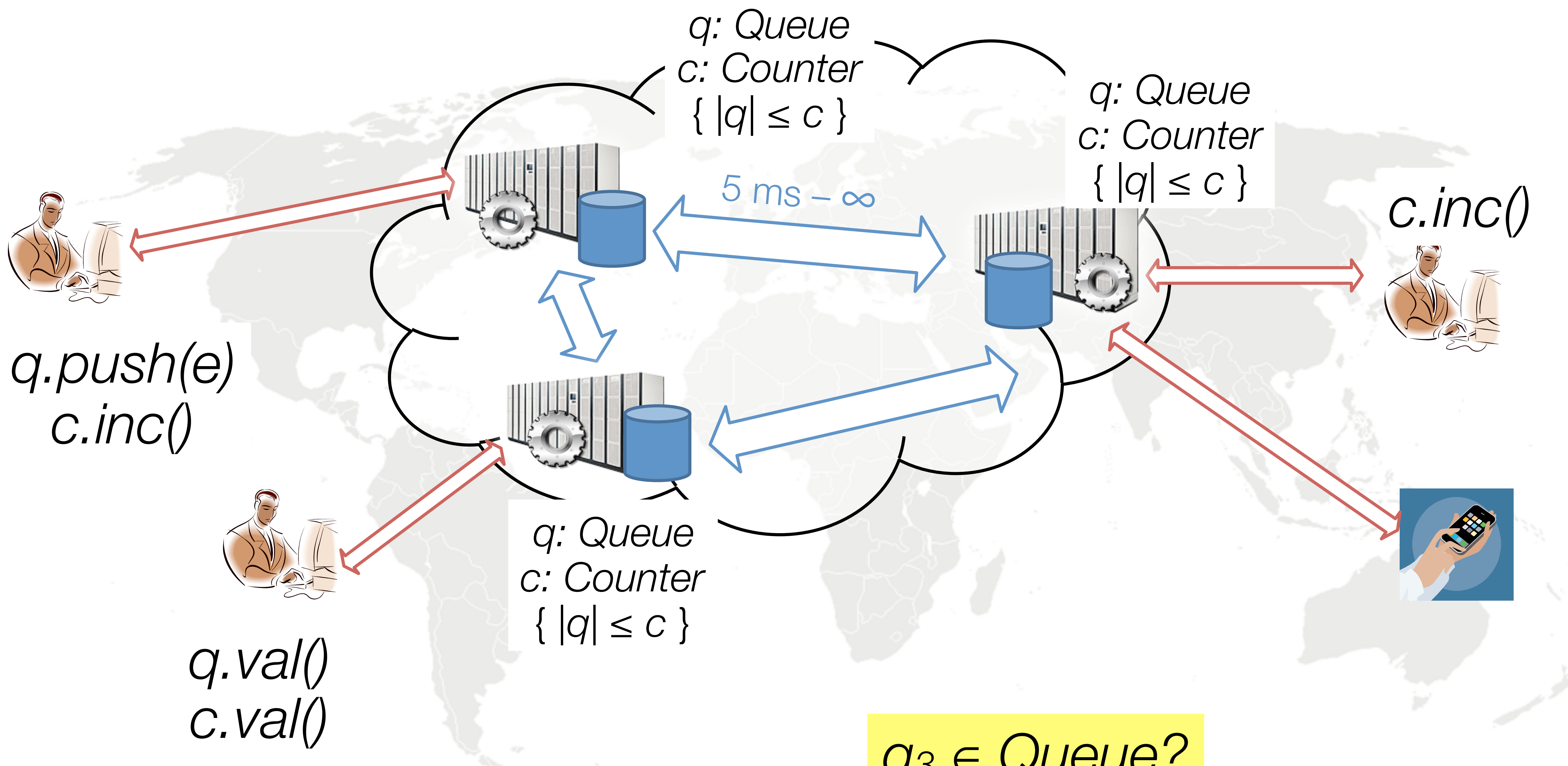
Shared database



Geo-replicated



Geo-replicated

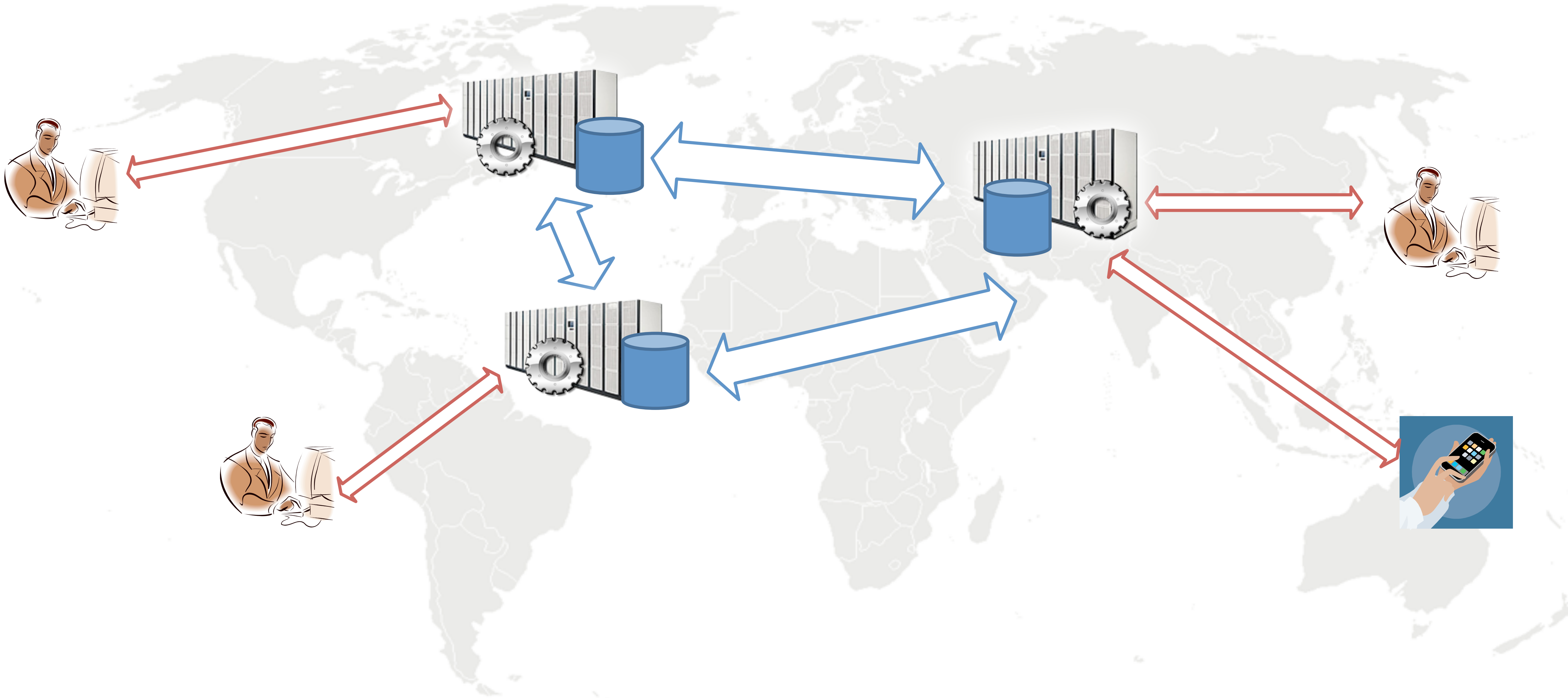


$q_3 \in$ Queue?
$q_1 = q_2$?
$|q_1| \leq c_4$?

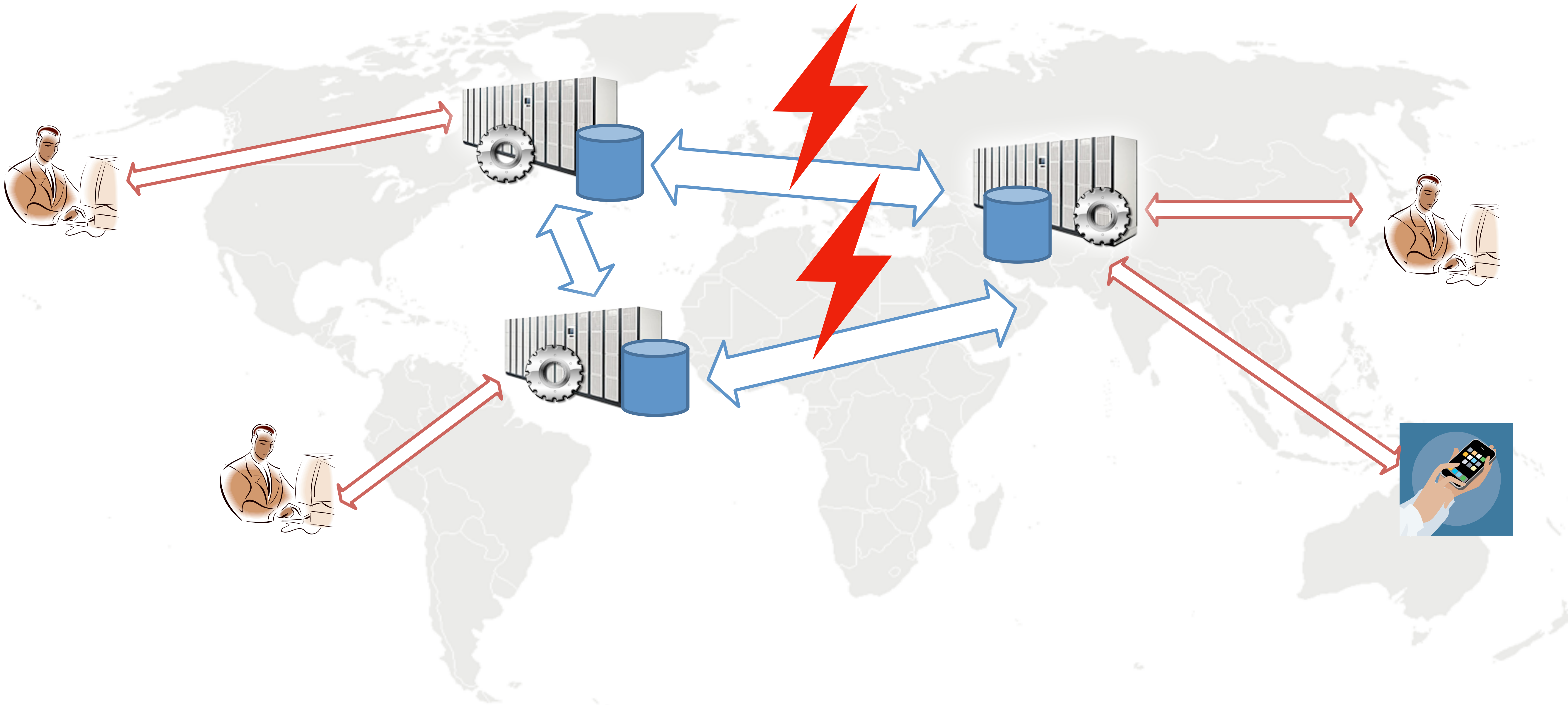
Consistency

- More replicas:
  - Better read availability, responsiveness, performance
  - More work to keep replicas in sync
- Consistent = behavior similar to sequential:
  - Satisfies specs: does q behave like a queue?
  - Replicas agree: is q identical everywhere?
  - Objects agree: is $|q| \leq c$?
  - Same flow of time? *q1.push()* before *q2.push()*

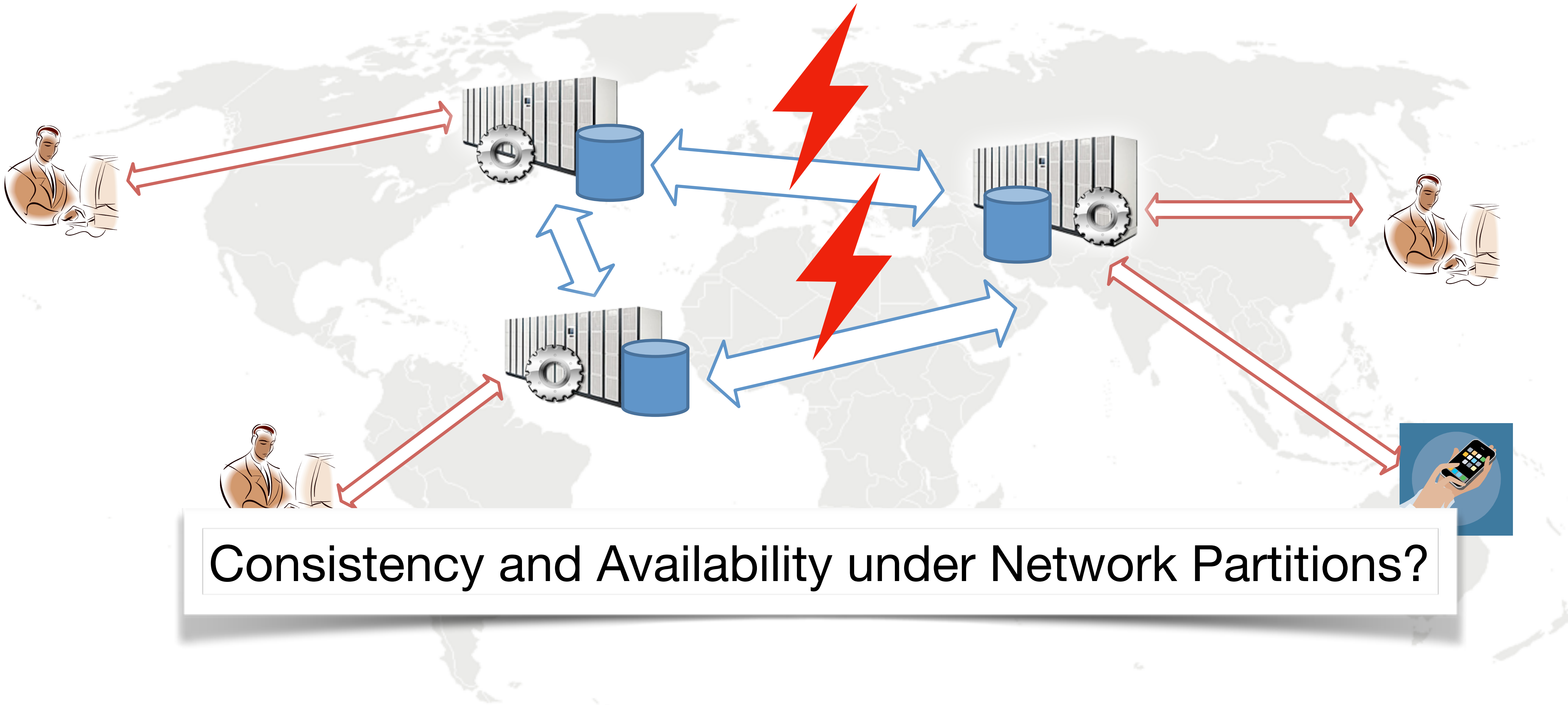
CAP Theorem



CAP Theorem

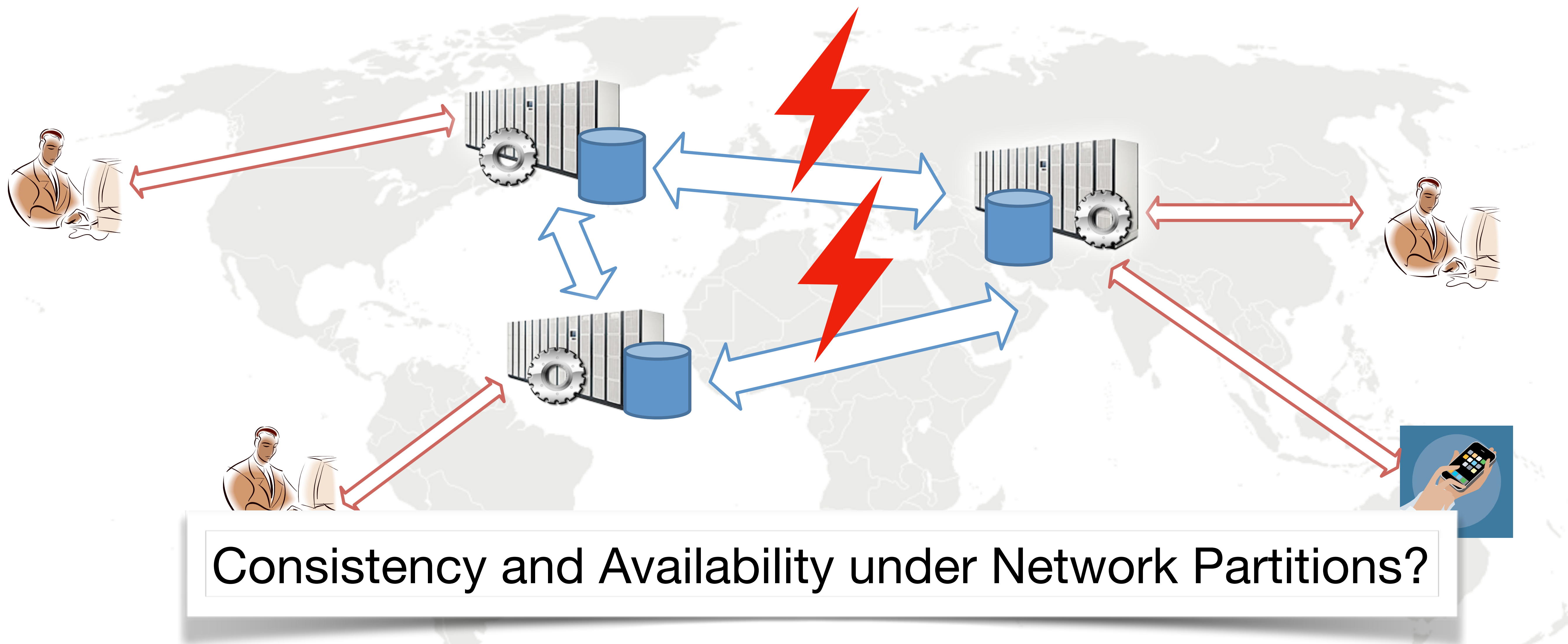


CAP Theorem



Consistency and Availability under Network Partitions?

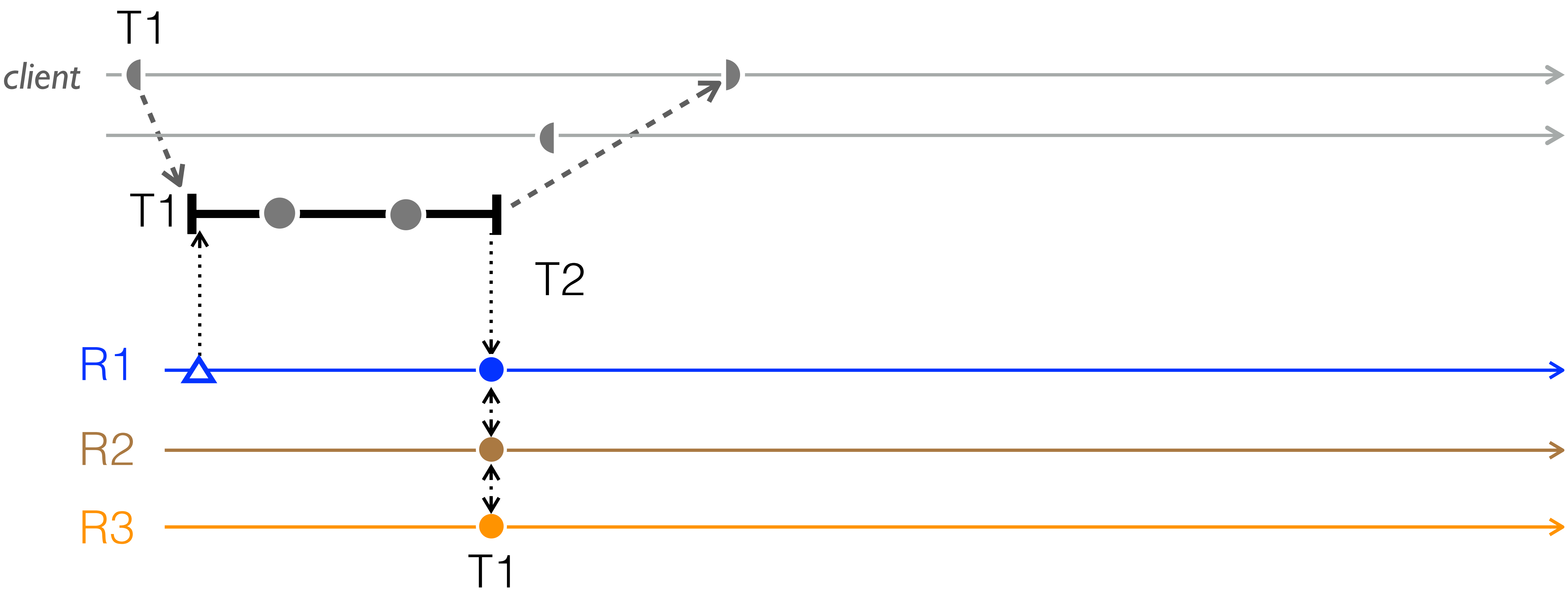
CAP Theorem



Impossible:

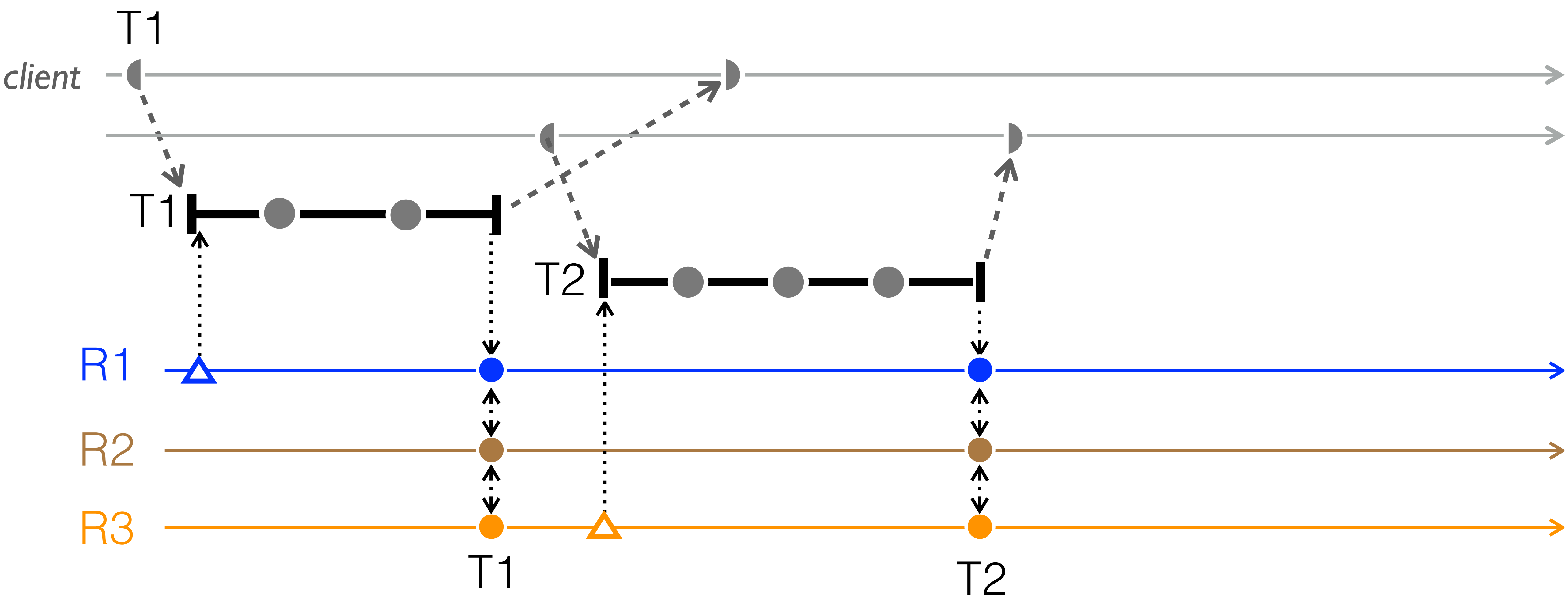
- Consistency: the system has to stop until the network is restored
- Availability: we have to let different replicas diverge (for a while)

Strict Serialisability

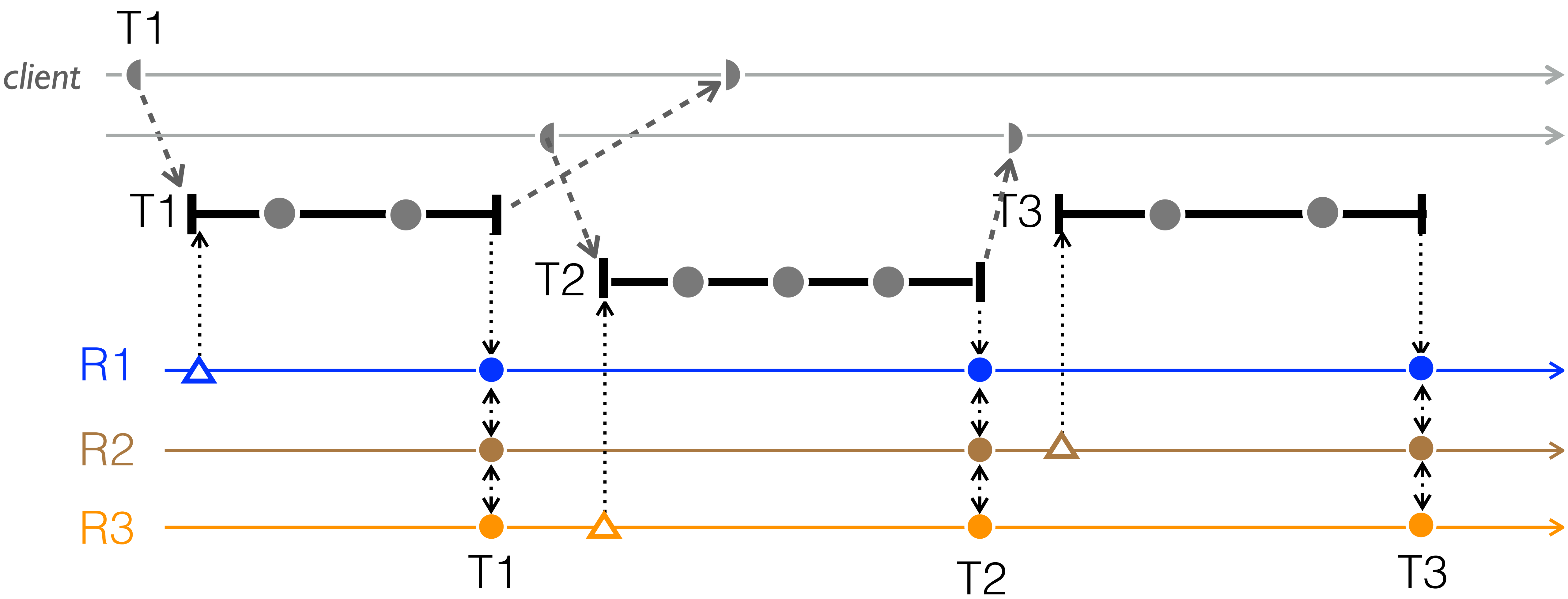


[Consistency in 3D]

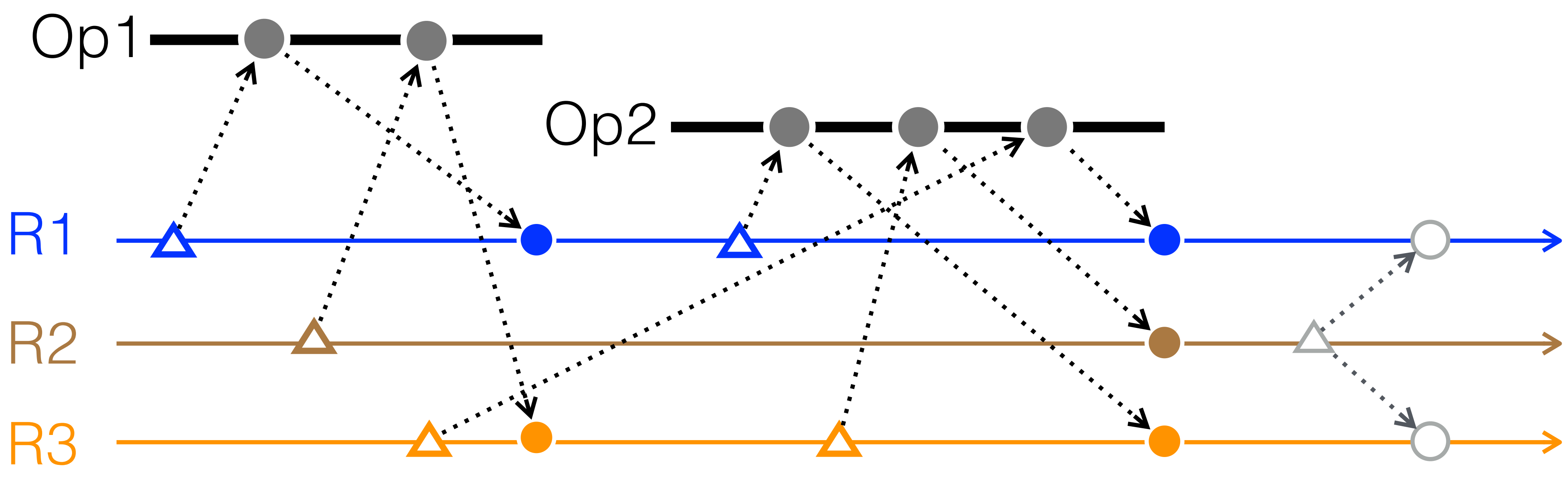
Strict Serialisability



Strict Serialisability



Eventual consistency



[Consistency in 3D]

Replicated operation

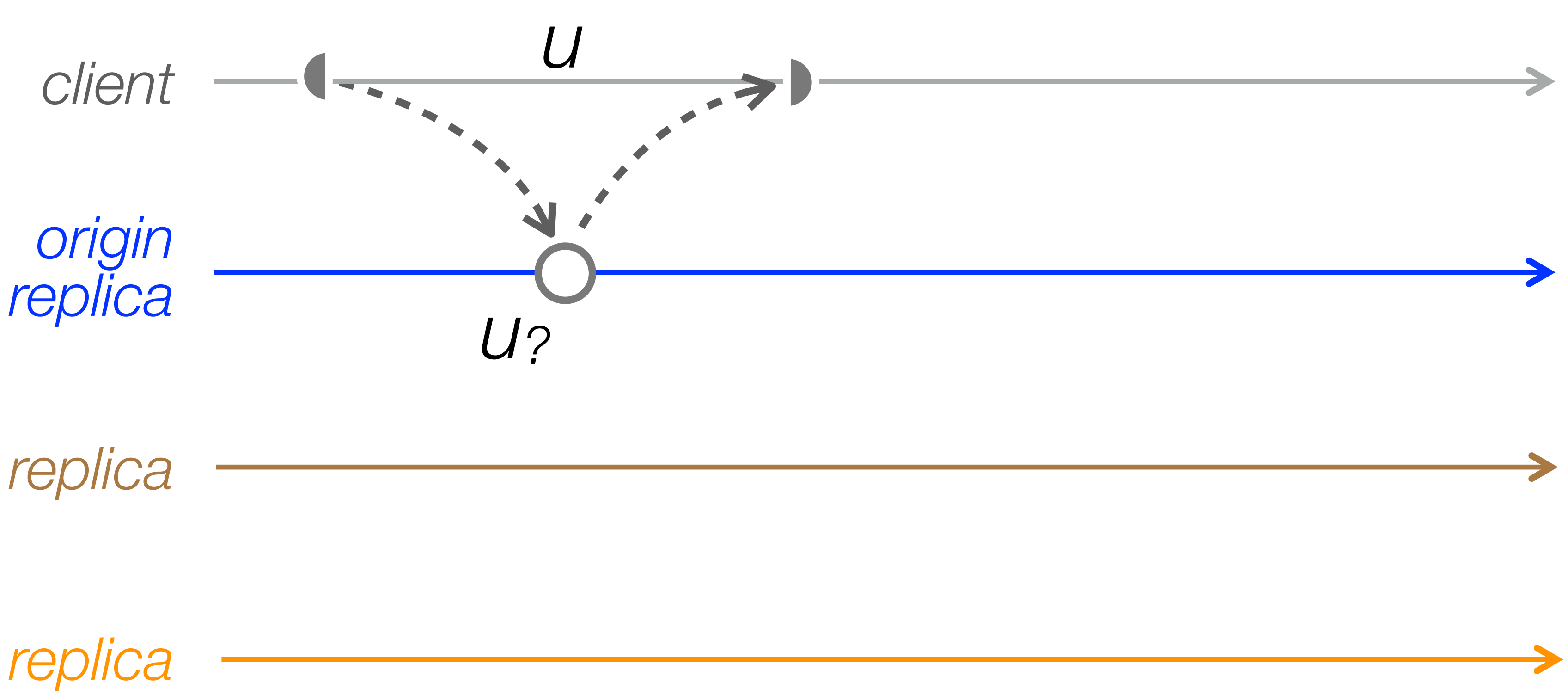
u: $state \curvearrowright (retval, (state \curvearrowright state))$

Prepare (@origin) $u_?$; deliver $u_!$

Read one, write all (ROWA)

Deferred-update replication (DUR)

Replicated operation



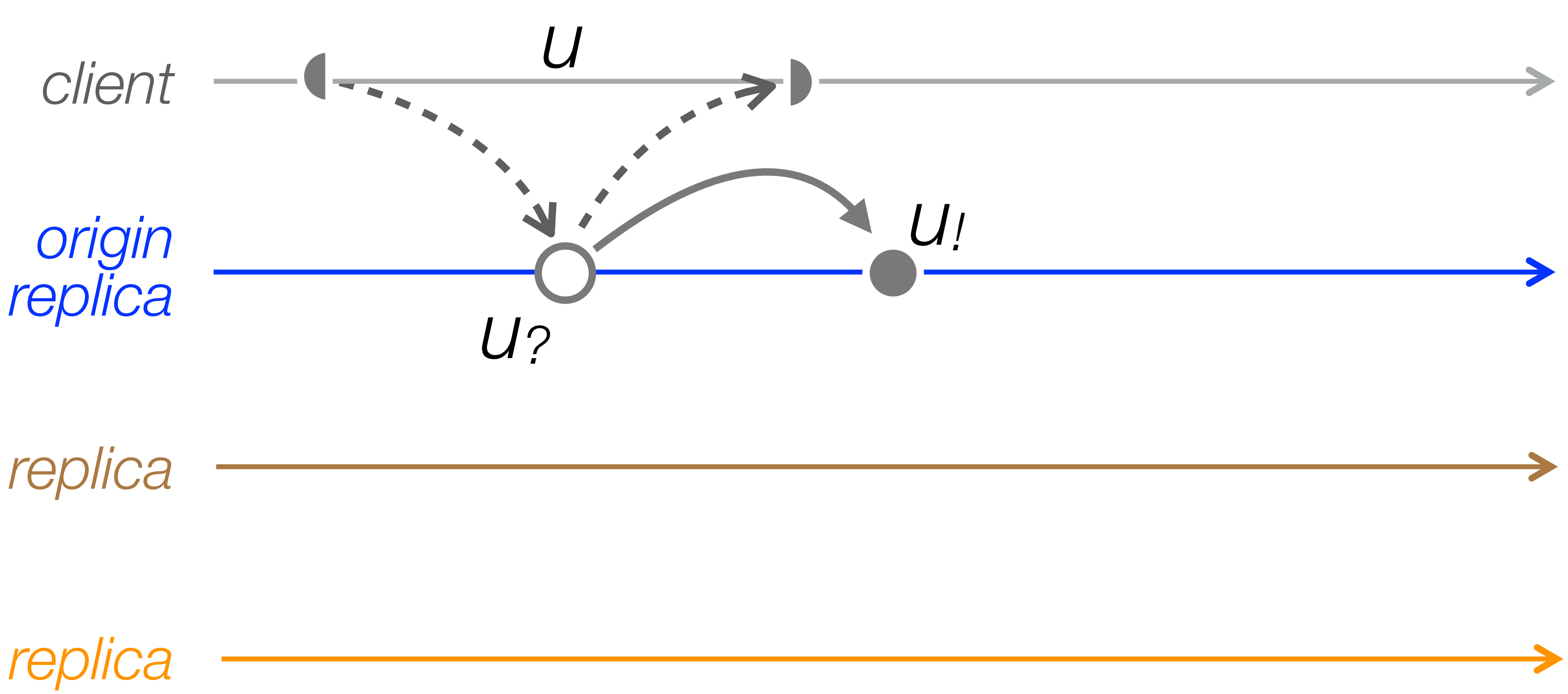
$u\colon \mathit{state} \curvearrowright (\mathit{retval}, (\mathit{state} \curvearrowright \mathit{state}))$

Prepare (@origin) $u_?$; deliver $u_!$

Read one, write all (ROWA)

Deferred-update replication (DUR)

Replicated operation



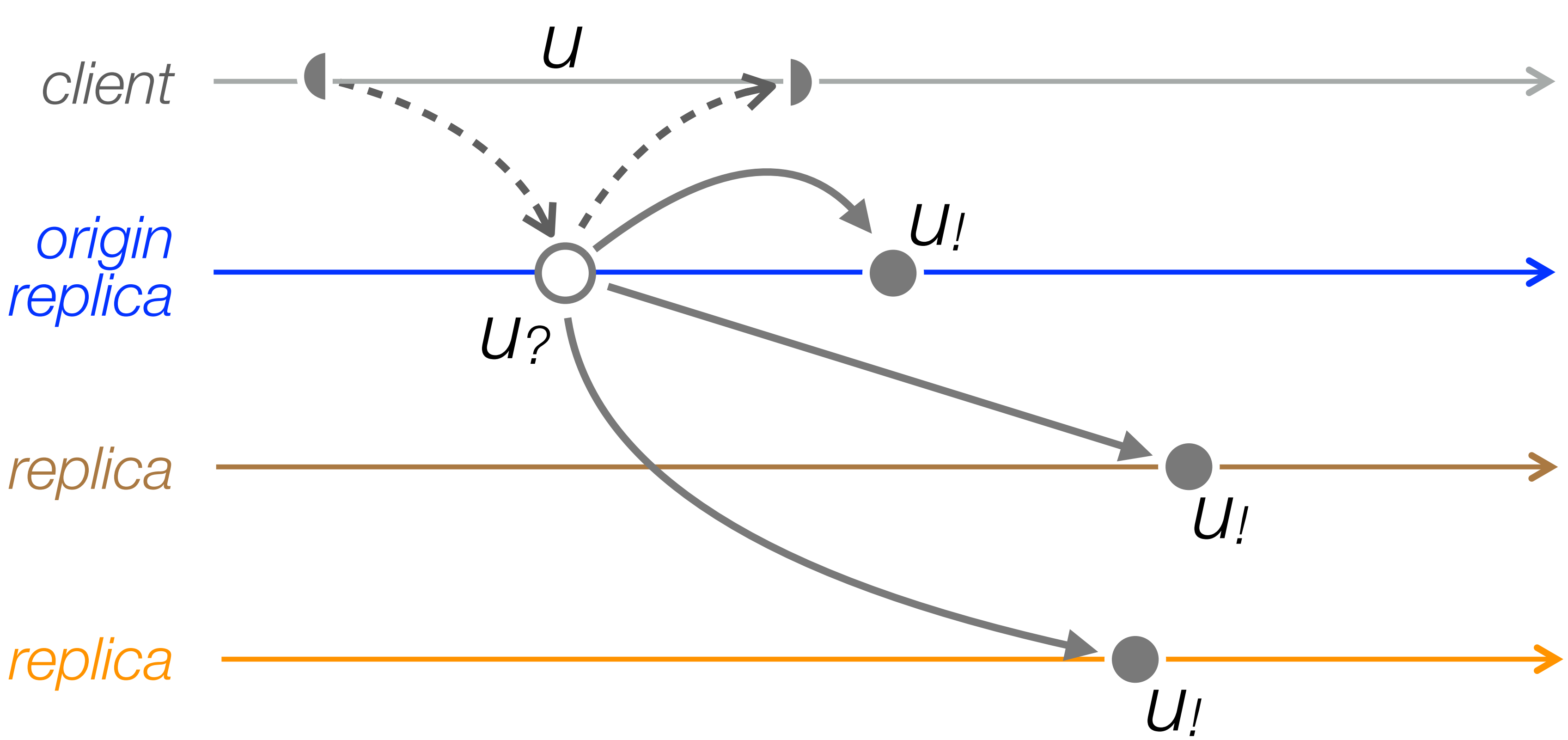
u: state ⤻ (retval, (state ⤻ state))

Prepare (@origin) $u_?$; deliver $u_!$

Read one, write all (ROWA)

Deferred-update replication (DUR)

Replicated operation



u: *state* $\curvearrowright$ (*retval*, (*state* $\curvearrowright$ *state*))

Prepare (@origin) $u_?$; deliver $u_!$

Read one, write all (ROWA)

Deferred-update replication (DUR)

In this Class

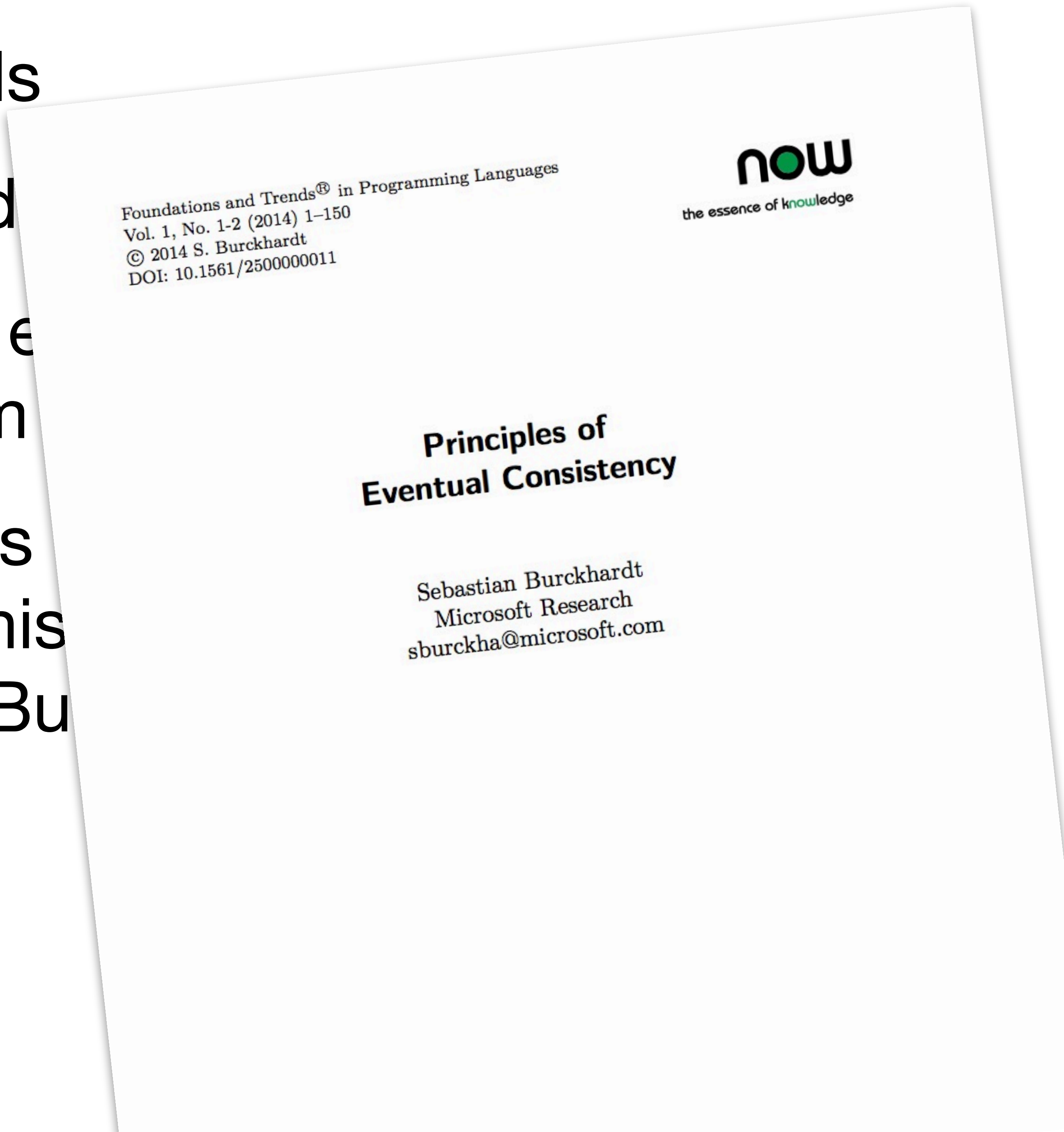
- We will adopt a client view of consistency
  - Just like with Memory Models
- Different protocols implement different consistency criteria
  - Stronger protocols are more expensive in performance but limit the non-determinism for clients
- We will use a uniform semantics for clients that over-approximates the non-determinism

[Burckhardt,Gotsman,Yang'15]

In this Class

- We will adopt a client view of consistency
  - Just like with Memory Models
- Different protocols implement d
  - Stronger protocols are more e
    but limit the non-determinism
- We will use a uniform semantics
  approximates the non-determinis

[Bu


Foundations and Trends® in Programming Languages
Vol. 1, No. 1-2 (2014) 1–150
© 2014 S. Burckhardt
DOI: 10.1561/2500000011

now
the essence of knowledge

Principles of Eventual Consistency

Sebastian Burckhardt
Microsoft Research
sburckha@microsoft.com

Client View of the system

- We will explain in an axiomatic way (à la CAT) the possible results of each operation
- As in CAT, we will posit the existence of certain orders that explain why a behavior is possible
- We will describe Distributed Data Structure specifications by exploiting these orders
- Implementations of distributed data structures can be verified agains these specifications
  - We will not talk about verification

Client Operations

- Client submit operations which can in turn be transactions
- A client is represented as a Session
- A single session could issue multiple operations and transactions
- We will consider a session to be the equivalent of the program order in relaxed memory models

Axiomatic Consistency Histories

$$H = (E, \mathsf{op}, \mathsf{rval}, \mathsf{rb}, \mathsf{ss})$$

Axiomatic Consistency Histories

A set of events

$$H = (E, \mathsf{op}, \mathsf{rval}, \mathsf{rb}, \mathsf{ss})$$

Axiomatic Consistency Histories

A set of events

$$H = (E, \mathsf{op}, \mathsf{rval}, \mathsf{rb}, \mathsf{ss})$$

Operation function:
op(e) is the operation of event e

Axiomatic Consistency Histories

A set of events

$$H = (E, \mathsf{op}, \mathsf{rval}, \mathsf{rb}, \mathsf{ss})$$

Operation function:
op(e) is the operation of event e

Return value function:
rval(e) is the value returned by the operation in e

Axiomatic Consistency Histories

A set of events

$$H = (E, \mathsf{op}, \mathsf{rval}, \mathsf{rb}, \mathsf{ss})$$

<u>Operation function</u>:
op(e) is the operation of event e

Returns Before partial order

<u>Return value function</u>:
rval(e) is the value returned by the operation in e

Axiomatic Consistency Histories

A set of events

Same Session Equivalence

$$H = (E, \mathsf{op}, \mathsf{rval}, \mathsf{rb}, \mathsf{ss})$$

<u>Operation function</u>:

op(e) is the operation of event e

Returns Before partial order

<u>Return value function</u>:

rval(e) is the value returned by the operation in e

Axiomatic Consistency Histories

A set of events

Same Session Equivalence

$$H = (E, \mathsf{op}, \mathsf{rval}, \mathsf{rb}, \mathsf{ss})$$

Operation function:
op(e) is the operation of event e

Returns Before partial order

Return value function:
rval(e) is the value returned by the operation in e

Session order $\mathsf{so} = \mathsf{ss} \cap \mathsf{rb}$

Axiomatic Consistency
Abstract Executions

$$A = (\underbrace{E, \mathsf{op}, \mathsf{rval}, \mathsf{rb}, \mathsf{ss}}_{\text{History}}, \mathsf{vis}, \mathsf{ar})$$

Axiomatic Consistency Abstract Executions

<u>Visibility partial order</u>:

$e \xrightarrow{\text{vis}} e'$ represents that the effects of e are available when executing e'

$$A = (\underbrace{E, \mathsf{op}, \mathsf{rval}, \mathsf{rb}, \mathsf{ss}}_{\text{History}}, \mathsf{vis}, \mathsf{ar})$$

Axiomatic Consistency Abstract Executions

$$A = (\underbrace{E, \mathsf{op}, \mathsf{rval}, \mathsf{rb}, \mathsf{ss}}_{\text{History}}, \mathsf{vis}, \mathsf{ar})$$

Visibility partial order:

$e \xrightarrow{\mathsf{vis}} e'$ represents that the effects of e are available when executing e'

Arbitration TOTAL order:

$e \xrightarrow{\mathsf{ar}} e'$ represents that the globally the effects of e are assume to "happen prior" to the effects of e'

Session Guarantees (Anomalies)

Session Guarantees for Weakly Consistent Replicated Data

Douglas B. Terry, Alan J. Demers, Karin Petersen, Mike J. Spreitzer, Marvin M. Theimer, and Brent B. Welch

Computer Science Laboratory
Xerox Palo Alto Research Center
Palo Alto, California 94304

Abstract

Four per-session guarantees are proposed to aid users and applications of weakly consistent replicated data: Read Your Writes, Monotonic Reads, Writes Follow Reads, and Monotonic Writes. The intent is to present individual applications with a view of the database that is consistent with their own actions, even if they read and write from various, potentially inconsistent servers. The guarantees can be layered on existing systems that employ a read-any/write-any replication scheme while retaining the principal benefits of such a scheme, namely high-availability, simplicity, scalability, and support for disconnected operation. These session guarantees were developed in the context of the Bayou project at Xerox PARC in which we are designing and building a replicated storage system to support the needs of mobile computing users who may be only intermittently connected.

1. Introduction

Techniques for managing weakly consistent replicated

may want to read and update data copied onto their portable computers even if they did not have the foresight to lock it before either a voluntary or an involuntary disconnection occurred. Also, the presence of slow or expensive communications links in the system can make maintaining closely synchronized copies of data difficult or uneconomical.

Unfortunately, the lack of guarantees concerning the ordering of read and write operations in weakly consistent systems can confuse users and applications, as reported in experiences with Grapevine [21]. A user may read some value for a data item and then later read an older value. Similarly, a user may update some data item based on reading some other data, while others read the updated item without seeing the data on which it is based. A serious problem with weakly consistent systems is that inconsistencies can appear even when only a single user or application is making data modifications. For example, a mobile client of a distributed database system could issue a write at one server, and later issue a read at a different server. The client would see inconsistent results unless the two servers had synchronized with one another sometime between the two operations.

In this paper, we introduce *session guarantees* that alleviate this problem of weakly consistent systems while

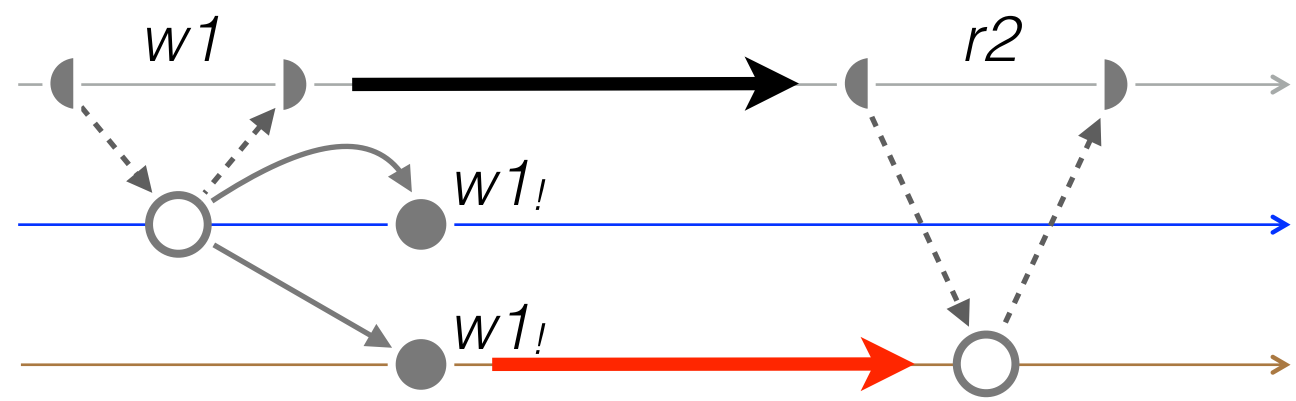
Session Guarantees (Anomalies)

$$
\boxed{
\begin{tikzcd}
\mathrm{wr(x,1)} \arrow[d, "\mathsf{rb}"] \\
\mathrm{rd(x,0)}
\end{tikzcd}
}
$$

Session Guarantees (Anomalies)

$$\boxed{\begin{array}{c} \mathrm{wr}(\mathrm{x},1) \\ \big\downarrow \mathrm{rb} \\ \mathrm{rd}(\mathrm{x},0) \end{array}}$$

Read My Writes

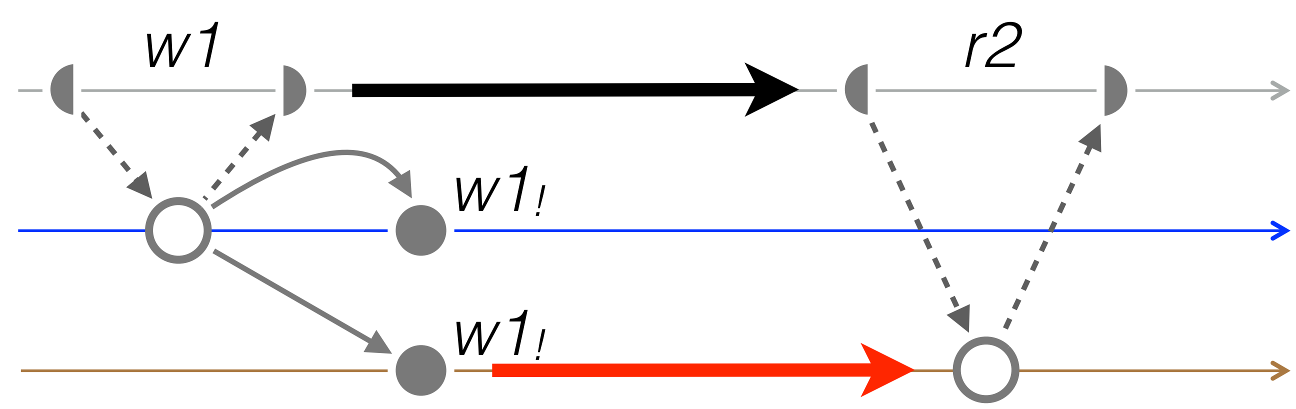


Client / RMW: *r2* must include *w1*

Session Guarantees (Anomalies)

$$\mathrm{wr}(x,1) \xrightarrow{\text{rb}} \mathrm{rd}(x,0)$$

Read My Writes



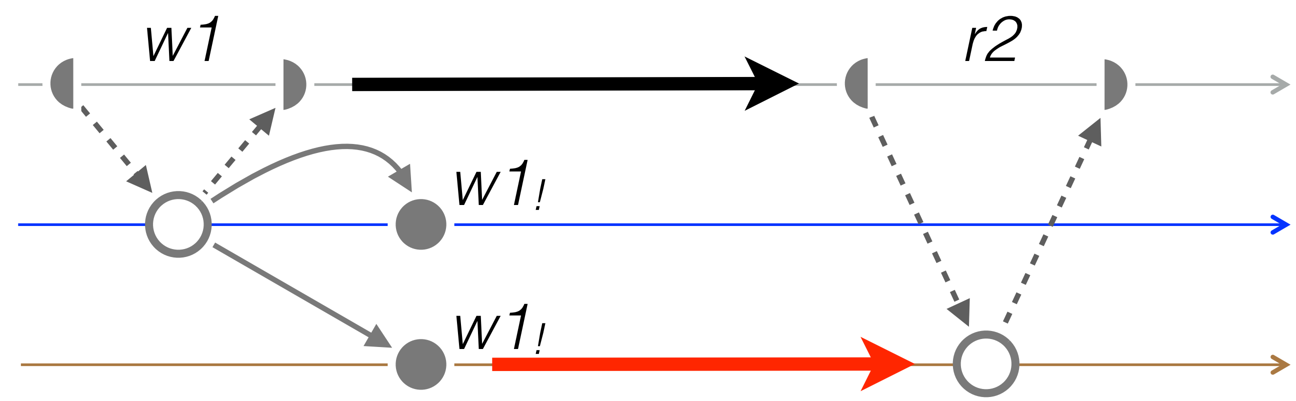
Client / RMW: *r2* must include *w1*

$$\mathrm{wr}(x,1) \xrightarrow{\text{vis}} \mathrm{rd}(x,1) \xrightarrow{\text{rb}} \mathrm{rd}(x,0)$$

Session Guarantees (Anomalies)

$$\begin{array}{c} wr(x,1) \\ \downarrow \mathsf{rb} \\ rd(x,0) \end{array}$$

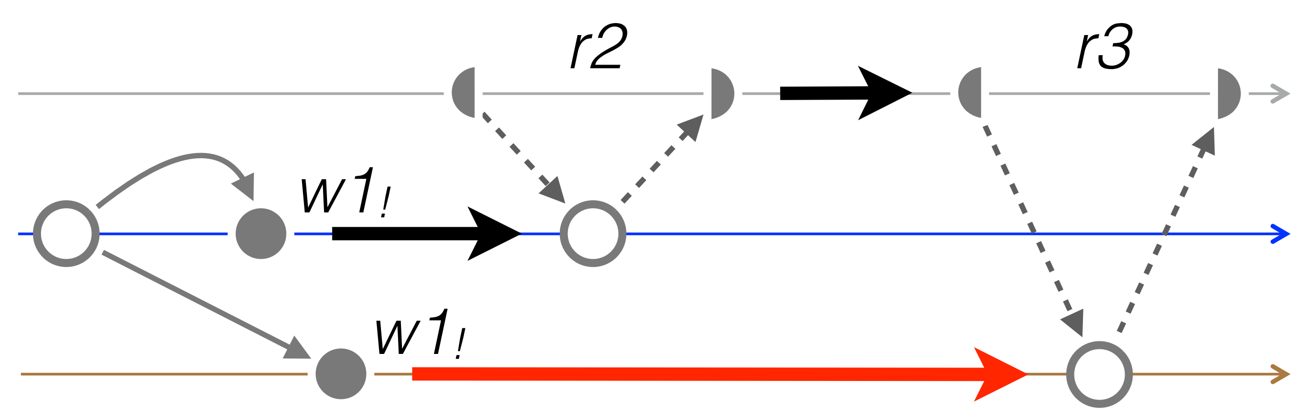
Read My Writes



Client / RMW: *r2* must include *w1*

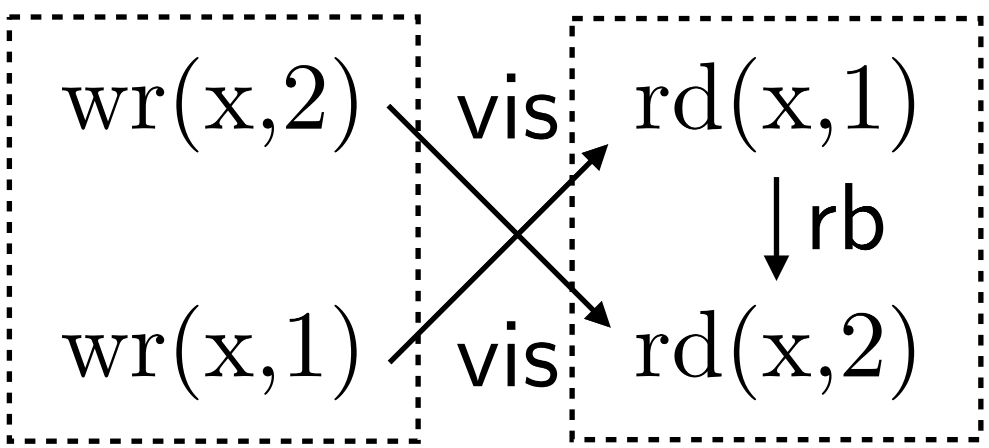
$$wr(x,1) \xrightarrow{\mathsf{vis}} \begin{array}{c} rd(x,1) \\ \downarrow \mathsf{rb} \\ rd(x,0) \end{array}$$

Monotonic reads

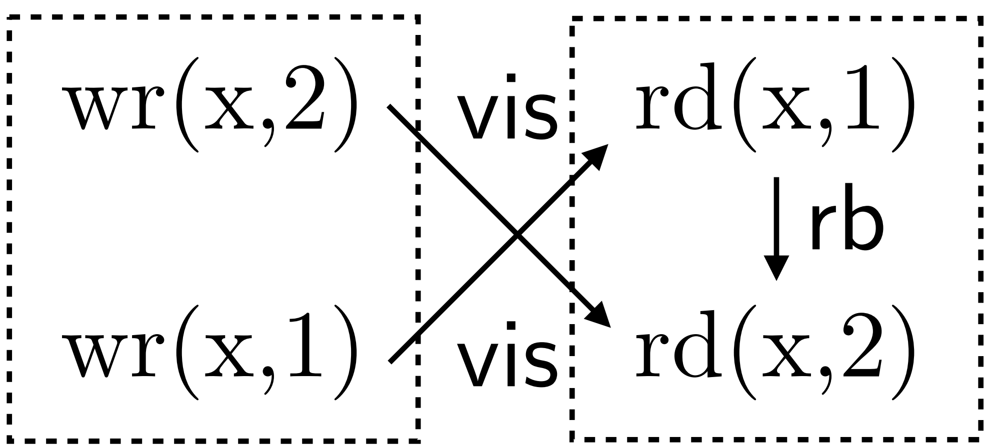


Client / No rollback: *r3* must include *w1*

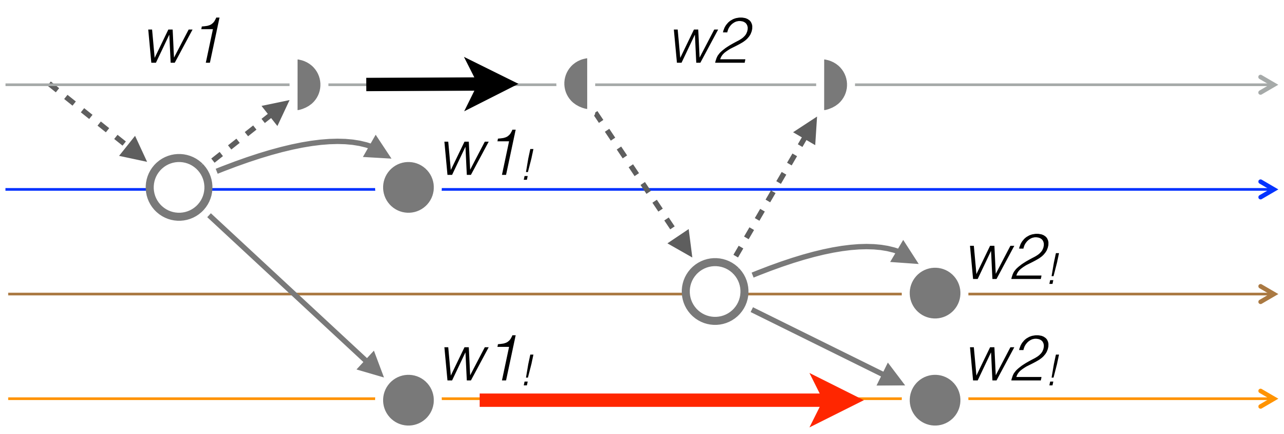
Session Guarantees (Anomalies)



Session Guarantees (Anomalies)

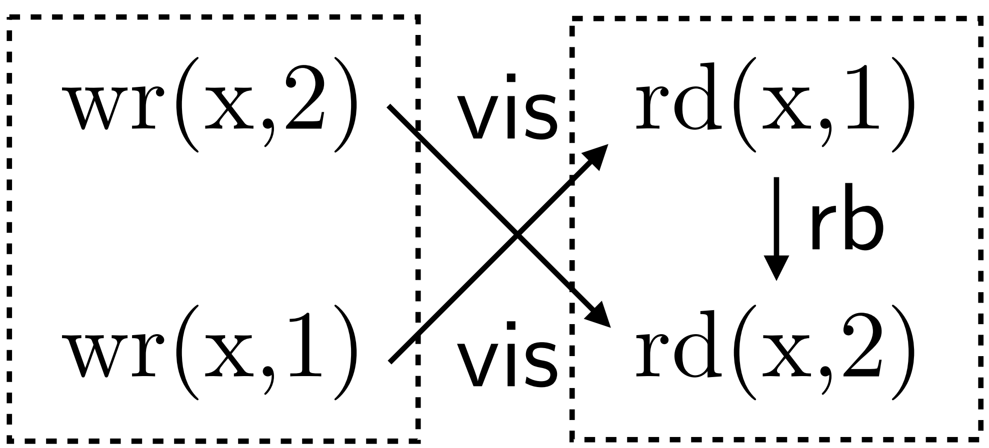


Monotonic writes

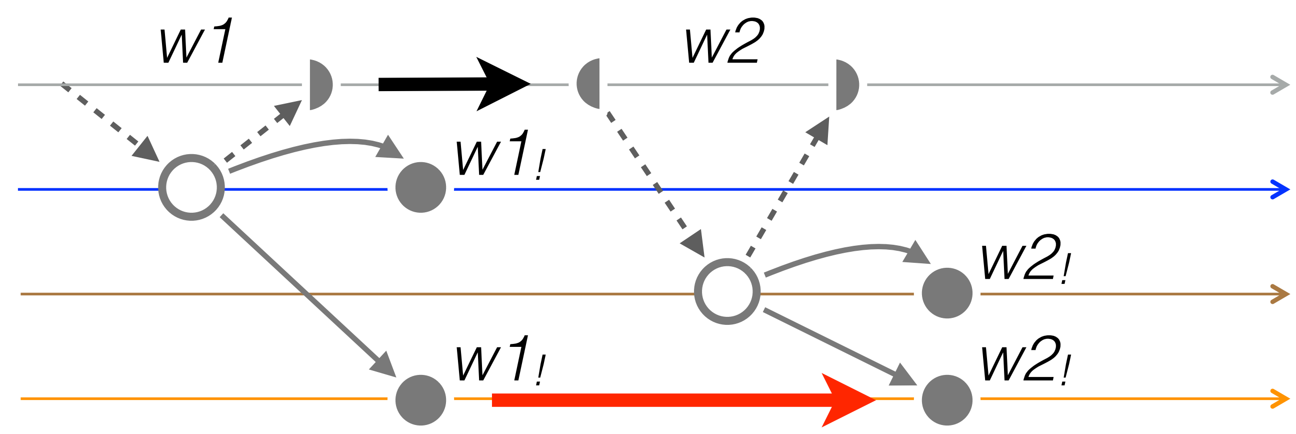


Global / No rollback: *r3* must include *w1*

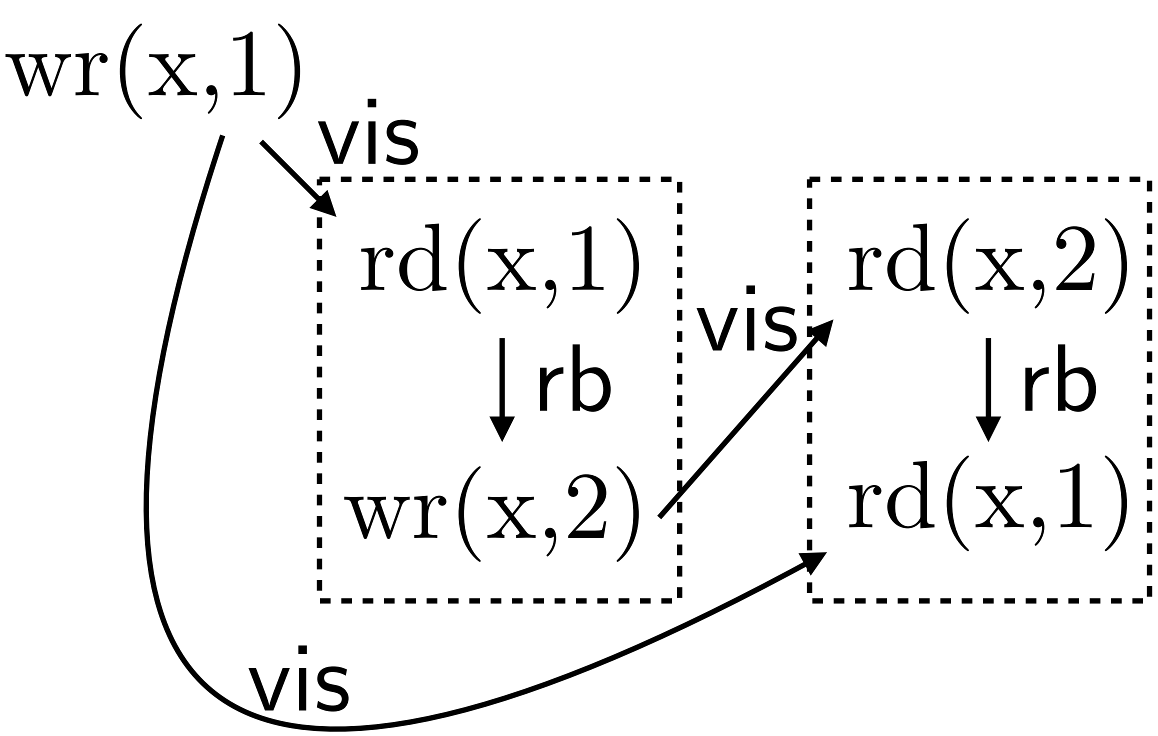
Session Guarantees (Anomalies)



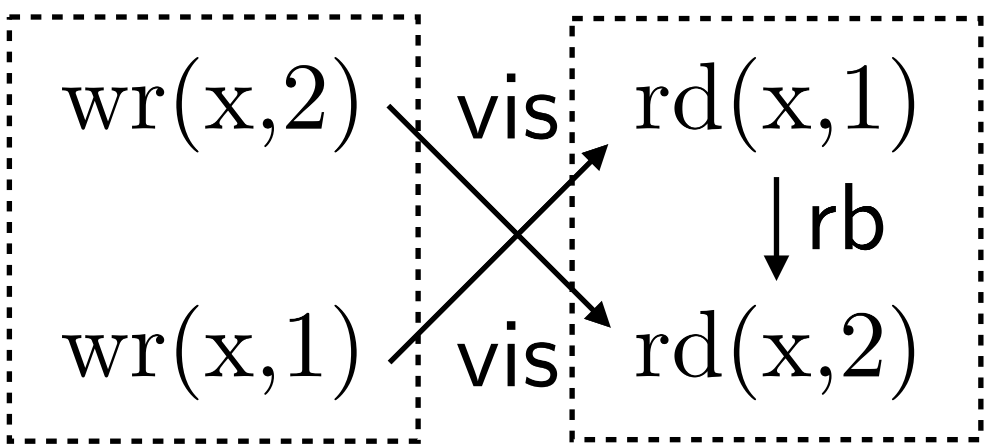
Monotonic writes

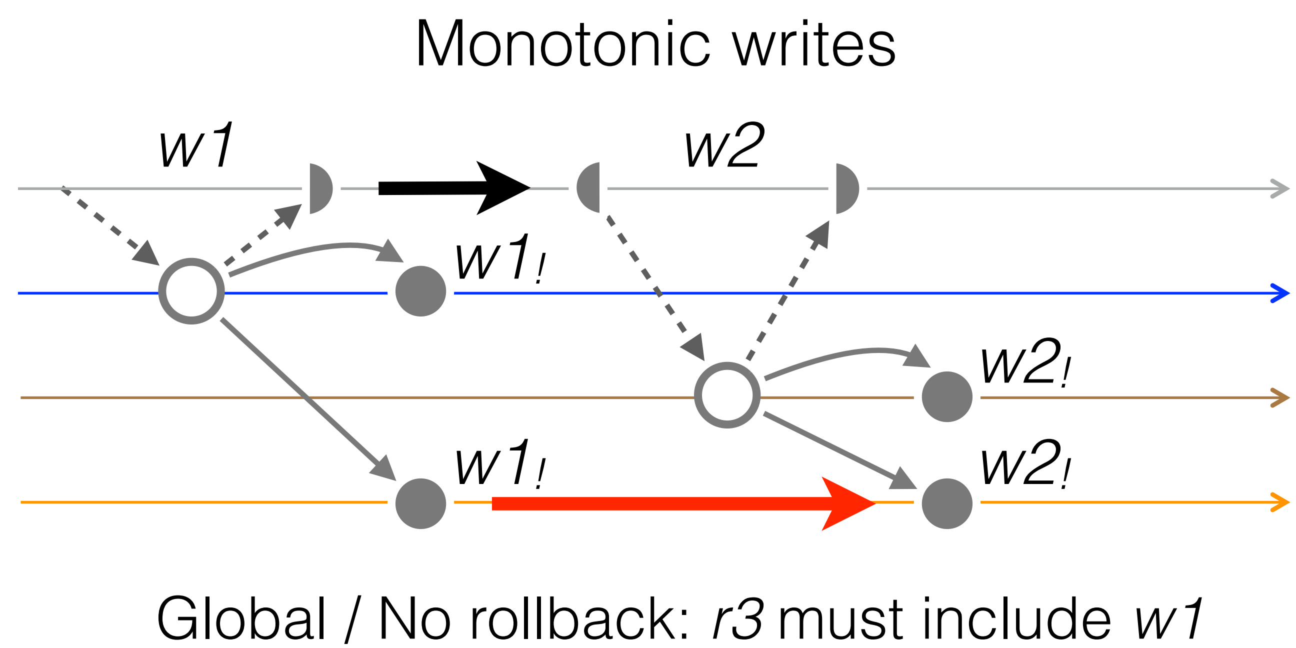


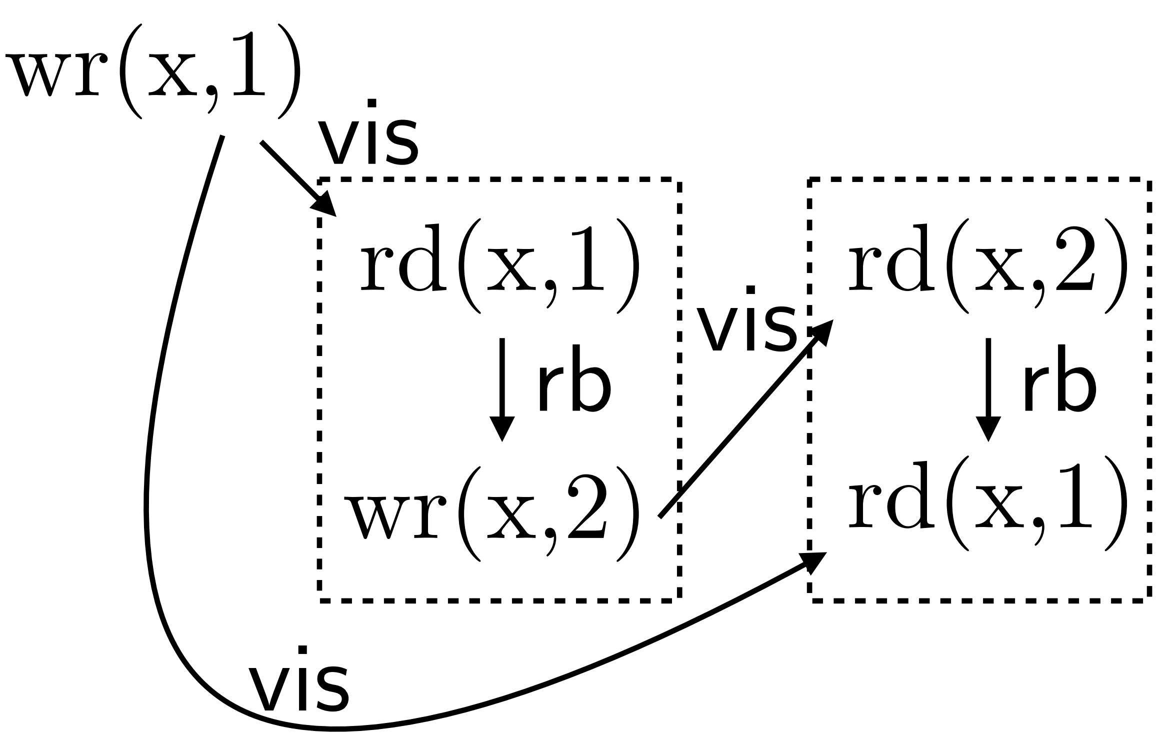
Global / No rollback: *r3* must include *w1*

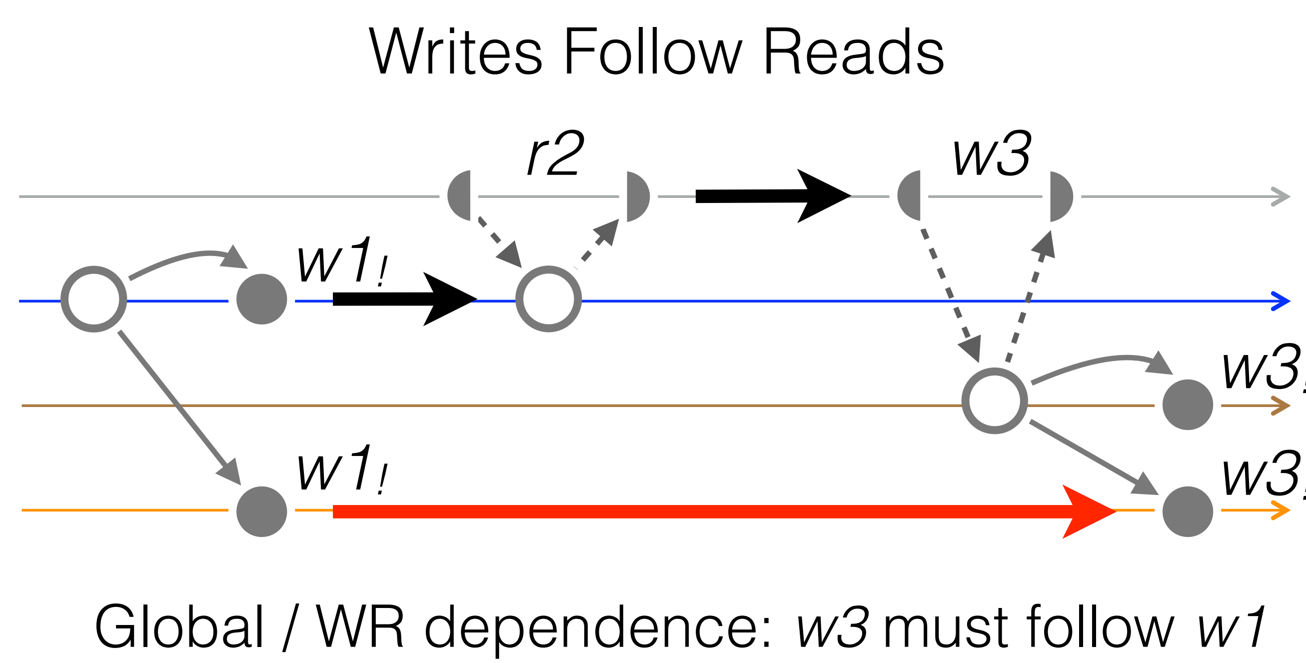


Session Guarantees (Anomalies)

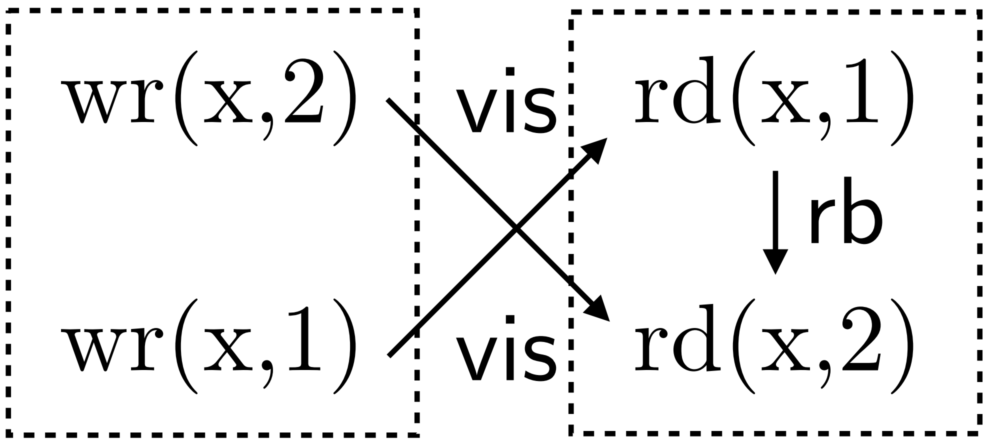




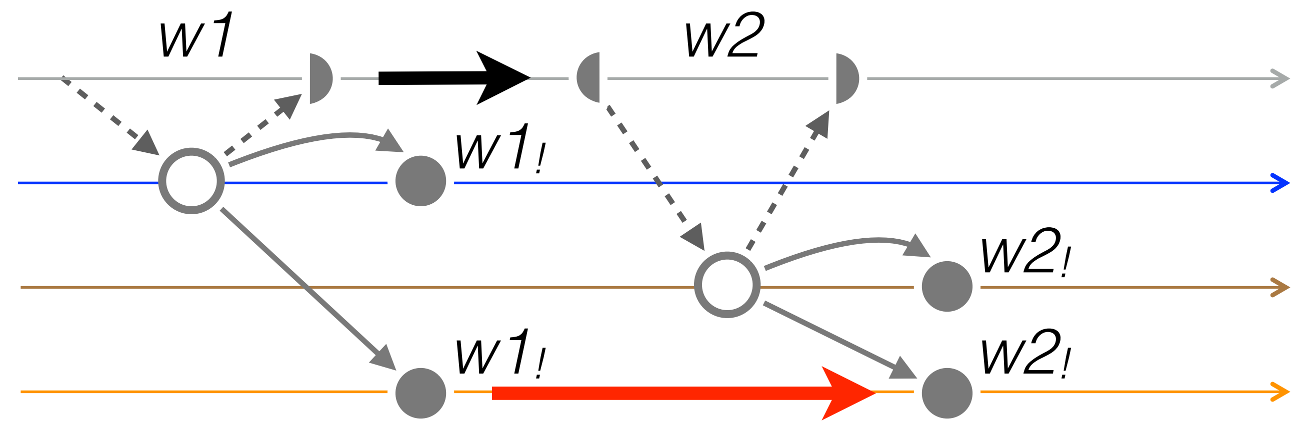




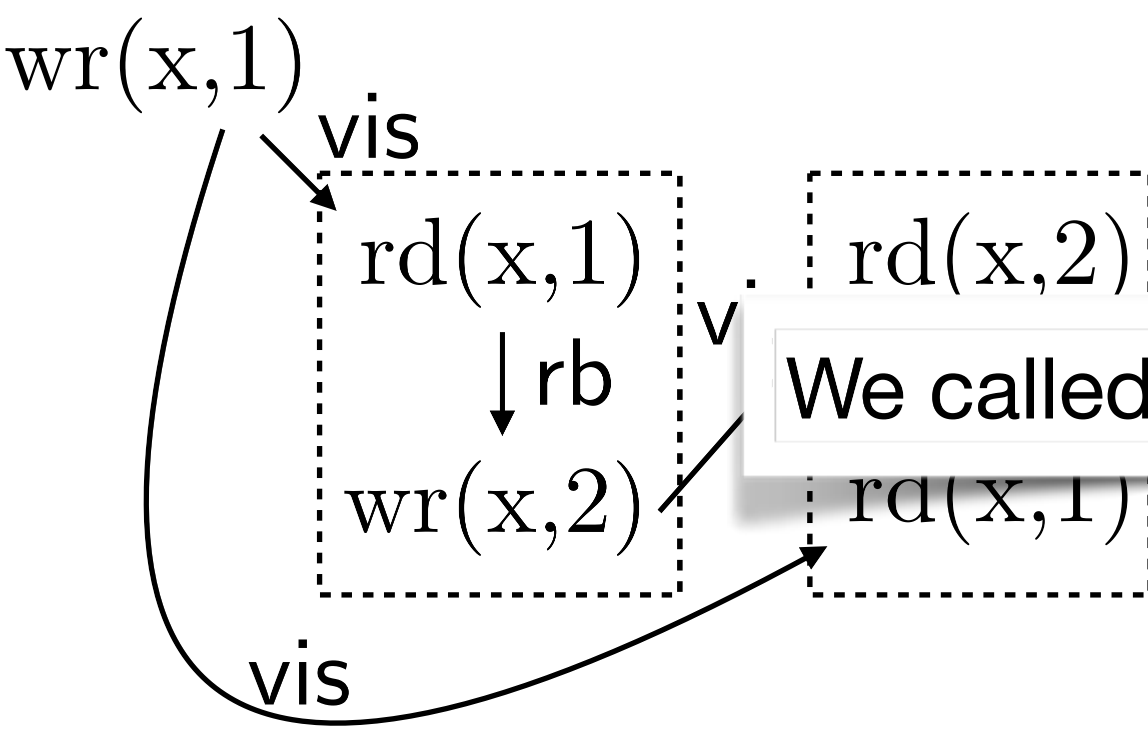
Session Guarantees (Anomalies)



Monotonic writes

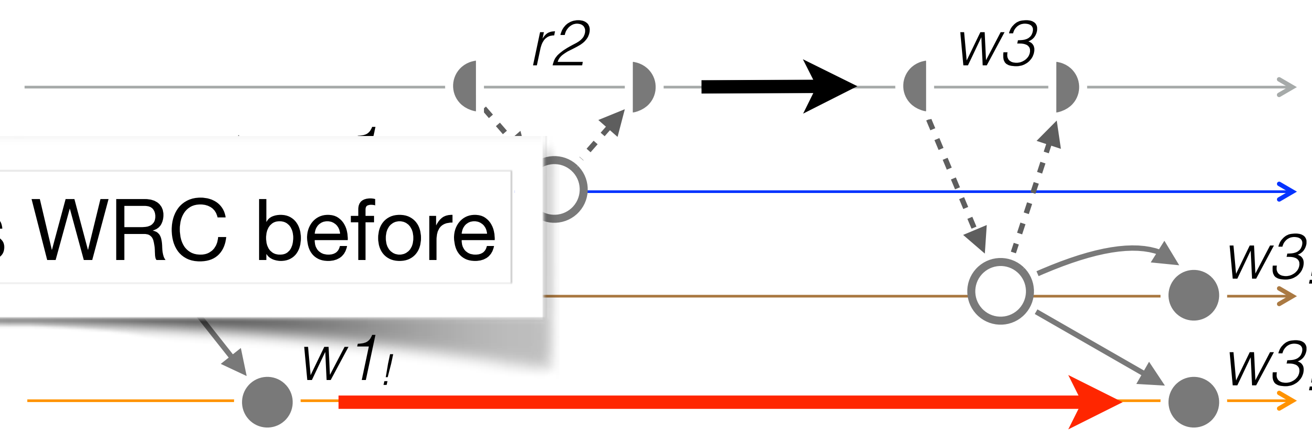


Global / No rollback: *r3* must include *w1*



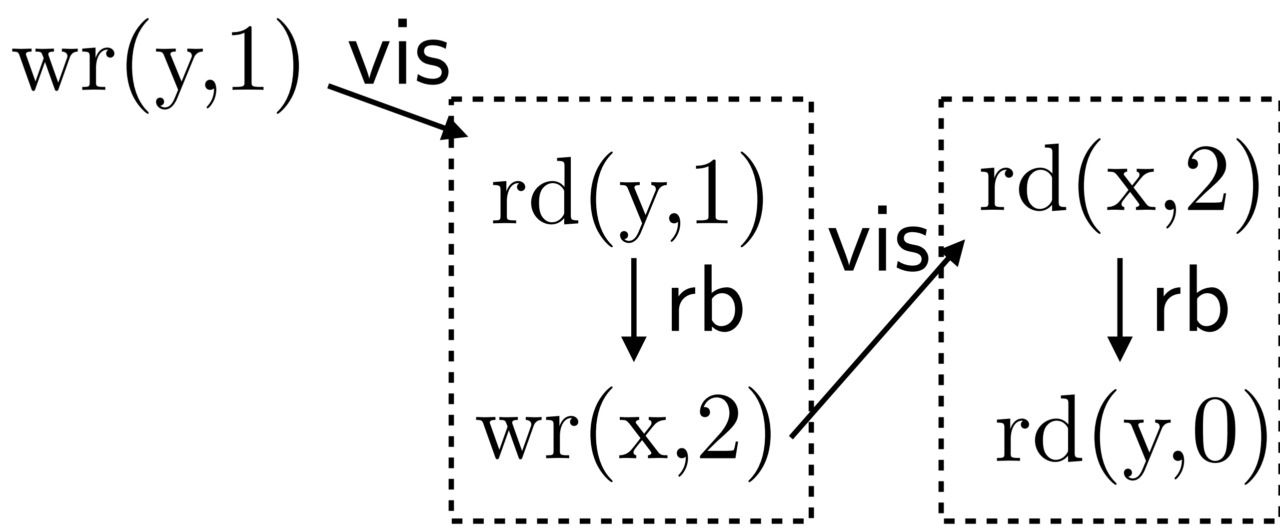
We called this WRC before

Writes Follow Reads



Global / WR dependence: *w3* must follow *w1*

Causality



Specifying Objects

Set of object states

Sequential type $\mathcal{S} = (\Sigma, \sigma_0, \delta)$

Initial Object state

$\sigma_0 \in \Sigma$

Transition Relation

$\delta : \mathsf{Operations} \times \Sigma \to (\mathsf{Values} \times \Sigma)$

A register $\mathcal{R}eg = (\mathbb{N}, 0, \delta_r)$

$$\delta_r(n, \mathsf{rd}) = (n, n)$$
$$\delta_r(n, \mathsf{wr(m)}) = (\perp, m)$$

From Burckhardt’s

State (and initial state)	Oper.	Returned value	Updated state	Condition

Counter $\mathcal{S}_{ctr}$

$n \in \mathbb{N}_0$ (initially 0)	rd	n	same	
	inc	ok	$n+1$	

Register $\mathcal{S}_{reg}$

$v \in$ Values (initially $undef$)	rd	v	same	
	wr(v')	ok	v'	

Key-Value Store $\mathcal{S}_{kvs}$

f : Values $\rightharpoonup_{\text{fin}}$ Values (initially $\emptyset$)	rd(k)	$undef$	same	if $f(k) = \bot$
		f(k)	same	if $f(k) \neq \bot$
	wr(k, v)	ok	$f[k \mapsto v]$	

Set $\mathcal{S}_{set}$

$A \in \mathcal{P}_{\text{fin}}$(Values) (initially $\emptyset$)	rd	A	same	
	add(v)	ok	$A \cup \{v\}$	
	rem(v)	ok	$A \setminus \{v\}$	

Queue $\mathcal{S}_{queue}$

$w \in$ Values* (initially ϵ)	enq(v)	ok	$w \cdot v$	
	deq	v	w'	if $w = v \cdot w'$
		∇		if $w = \epsilon$

What is the state in a DS?

- Much like in memory models, there is no unique state σ at every point
- Instead, we have to define the data type based on what is visible at the replica where the operation happens
- We need to change our sequential specifications

Replicated DT Specifications

Instead of a state we use a context for operations

$$C = (E, \mathsf{op}, \mathsf{vis}, \mathsf{ar}) \qquad \mathcal{C} \text{ Type of all contexts}$$

We specify an object based on contexts

$$\mathcal{F} : \mathsf{Operations} \times \mathcal{C} \rightarrow \mathsf{Values}$$

Counter

$$\mathcal{F}_{ctr}(\mathsf{rd}, (E, \mathsf{op}, \mathsf{vis}, \mathsf{ar})) = \left|\{e' \in E \mid \mathsf{op}(e') = \mathsf{inc}\}\right|$$

What about conflicts

$$\begin{tikzcd}[column sep=small] \mathrm{wr}(\mathrm{x},2) \arrow[dr, "\text{vis}"] & & \mathrm{wr}(\mathrm{x},1) \arrow[dl, "\text{vis}"] \\ & \mathsf{rd}(x, ?) & \end{tikzcd}$$

- This non-determinism is problematic

What about conflicts

$$
\begin{tikzcd}
\mathrm{wr}(\mathrm{x},2) \arrow[dr, "\mathsf{vis}"] & & \mathrm{wr}(\mathrm{x},1) \arrow[dl, "\mathsf{vis}"] \\
& \mathsf{rd}(x, ?) &
\end{tikzcd}
$$

- This non-determinism is problematic
- Could lead to divergent replicas

What about conflicts

$$\begin{tikzcd} \mathrm{wr}(\mathrm{x},2) \arrow[dr, "\text{vis}"] & & \mathrm{wr}(\mathrm{x},1) \arrow[dl, "\text{vis}"] \\ & \mathrm{rd}(x, ?) & \end{tikzcd}$$

- This non-determinism is problematic
- Could lead to divergent replicas
- Yet, synchronization at this scale is too expensive

What about conflicts

$$
\begin{tikzcd}
\mathrm{wr}(\mathrm{x},2) \arrow[dr, "\text{vis}"] & & \mathrm{wr}(\mathrm{x},1) \arrow[dl, "\text{vis}"] \\
& \mathrm{rd}(x, ?) &
\end{tikzcd}
$$

- This non-determinism is problematic
- Could lead to divergent replicas
- Yet, synchronization at this scale is too expensive
- Conflict-Free Replicated Data Types to the rescue

What about conflicts

- This non-deter
- Could lead to c
- Yet, synchroniza
- Conflict-Free Re

INRIA
ROCQUENCOURT

Conflict-free 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

Thème COM — Systèmes communicants
Projet Regal

Rapport de recherche n° 7687 — version 2[†] — version initiale 19 juillet 2011 — version révisée 25 août 2011 — 18 pages

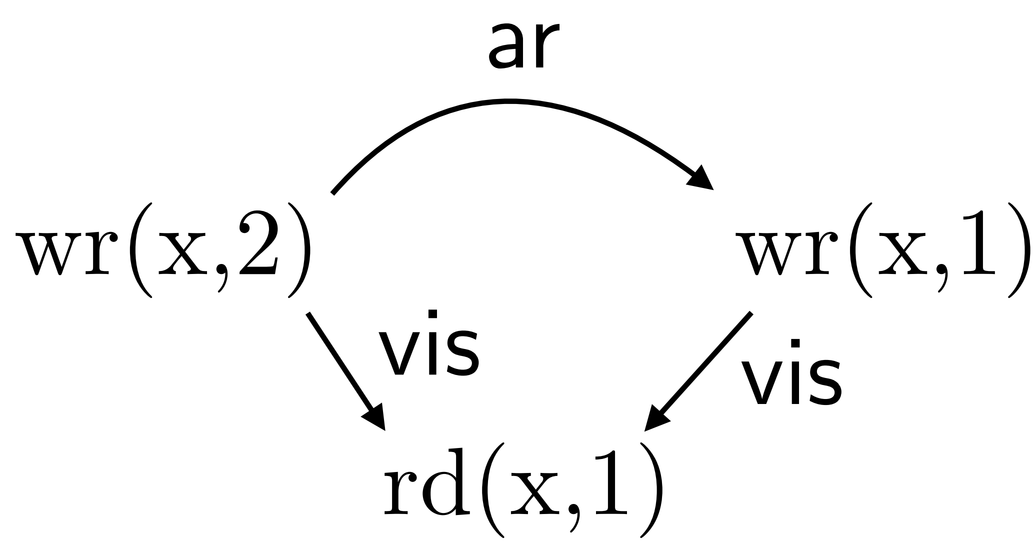
Abstract: Replicating data under Eventual Consistency (EC) allows any replica to accept updates without remote synchronisation. This ensures performance and scalability in large-scale distributed systems (e.g., clouds). However, published EC approaches are ad-hoc and error-prone. Under a formal Strong Eventual Consistency (SEC) model, we study sufficient conditions for convergence. A data type that satisfies these conditions is called a Conflict-free Replicated Data Type (CRDT). Replicas of any CRDT are guaranteed to converge in a self-stabilising manner, despite any number of failures. This paper formalises two popular approaches (state- and operation-based) and their relevant sufficient conditions. We study a ... such as sets with clean semantics, supporting both *add* and *remove* ... Graph data type. CRDT types can be ... interesting theoretical

What about conflicts

$$
\begin{tikzcd}
\mathrm{wr}(\mathrm{x},2) \arrow[dr, "\text{vis}"] & & \mathrm{wr}(\mathrm{x},1) \arrow[dl, "\text{vis}"] \\
& \text{rd}(x, ?) &
\end{tikzcd}
$$

- This non-determinism is problematic
- Could lead to divergent replicas
- Yet, synchronization at this scale is too expensive
- Conflict-Free Replicated Data Types to the rescue
  - aka. Convergent RDTs, Commutative RDTs
  - They enforce a winning strategy between conglicts

What about conflicts



- This non-determinism is problematic
- Could lead to divergent replicas
- Yet, synchronization at this scale is too expensive
- Conflict-Free Replicated Data Types to the rescue
    - aka. Convergent RDTs, Commutative RDTs
    - They enforce a winning strategy between conglicts

What about conflicts

$$\begin{tikzcd} \mathrm{wr}(x,2) \arrow[rr, bend left, "\mathsf{ar}"] \arrow[dr, "\mathsf{vis}"'] & & \mathrm{wr}(x,1) \arrow[dl, "\mathsf{vis}"] \\ & \mathrm{rd}(x,1) & \end{tikzcd}$$

Last Writer Wins Register

$$\mathcal{F}_{reg}(\mathsf{rd}, (E, \mathsf{op}, \mathsf{vis}, \mathsf{ar})) = \begin{cases} \mathit{undef} & \text{if } \mathsf{writes}(E) = \emptyset \\ v & \text{if } \mathsf{op}(\max_{\mathsf{ar}} \mathsf{writes}(E)) = \mathsf{wr}(v) \end{cases}$$

RDTs

Multi-Value Register

$$\mathcal{F}_{mvr}(\mathsf{rd}, (E, \mathsf{op}, \mathsf{vis}, \mathsf{ar})) =$$

$$\{v \mid \exists e \in E : \mathsf{op}(e) = \mathsf{wr}(v) \text{ and } \forall e' \in \mathsf{writes}(E) : e \not\xrightarrow{\mathsf{vis}} e'\}$$

RDTs

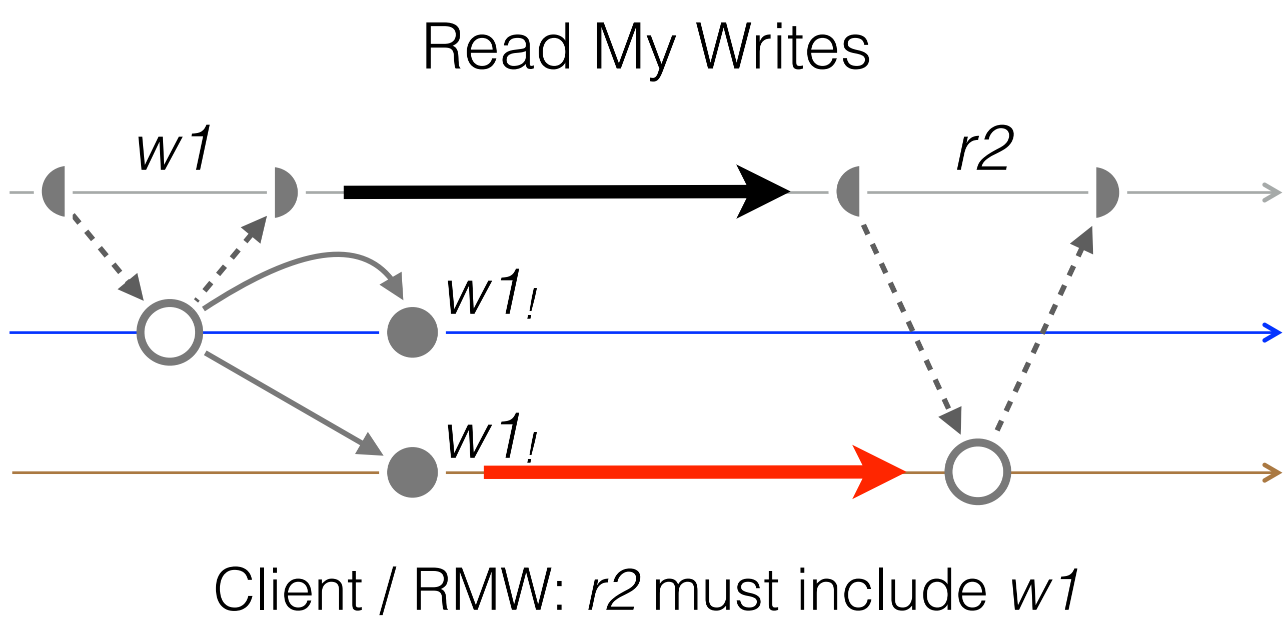
Multi-Value Register

$$\mathcal{F}_{mvr}(\mathsf{rd}, (E, \mathsf{op}, \mathsf{vis}, \mathsf{ar})) =$$
$$\{v \mid \exists e \in E : \mathsf{op}(e) = \mathsf{wr}(v) \text{ and } \forall e' \in \mathsf{writes}(E) : e \overset{\mathsf{vis}}{\not\longrightarrow} e'\}$$

Add-wins set

$$\mathcal{F}_{\mathrm{awset}}(\mathsf{contains}(v), (E, \mathsf{op}, \mathsf{vis}, \mathsf{ar})) = true \quad \overset{\mathrm{def}}{\Longleftrightarrow}$$
$$\exists e \in E : \ \mathsf{op}(e) = \mathsf{add}(v) \ \wedge \ \neg(\exists e' \in E : \mathsf{op}(e') = \mathsf{rem}(v) \wedge e \xrightarrow{\mathsf{vis}} e')$$

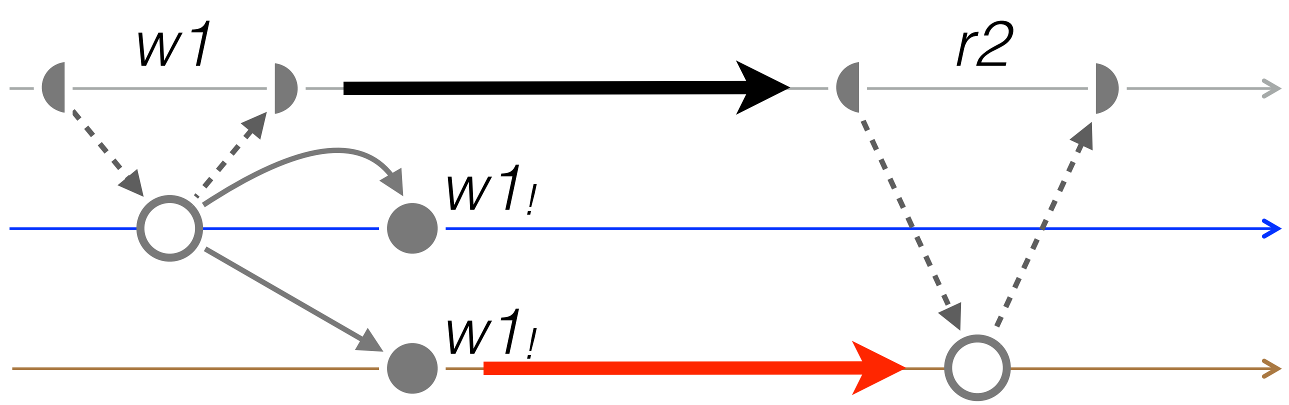
Consistency Axioms



so $\subseteq$ vis

Consistency Axioms

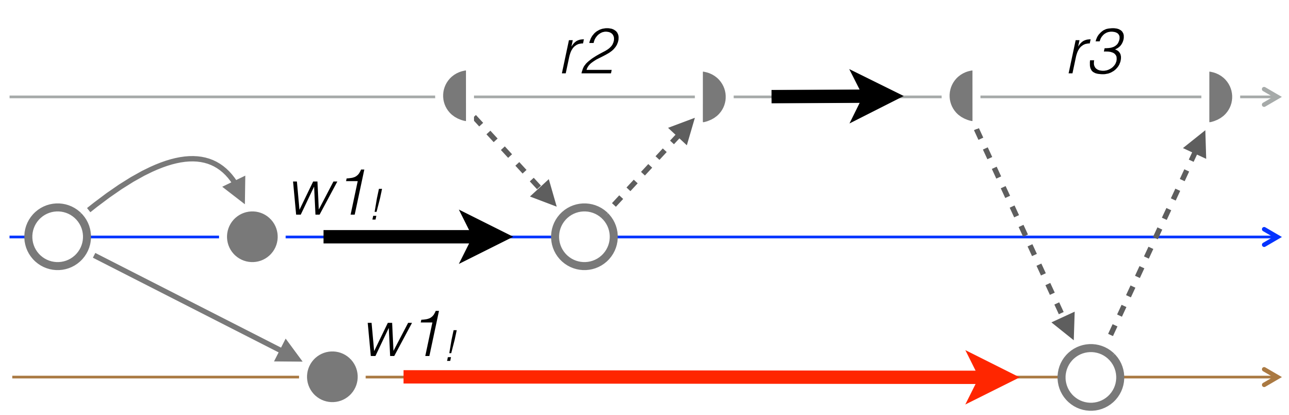
Read My Writes



Client / RMW: *r2* must include *w1*

so ⊆ vis

Monotonic reads

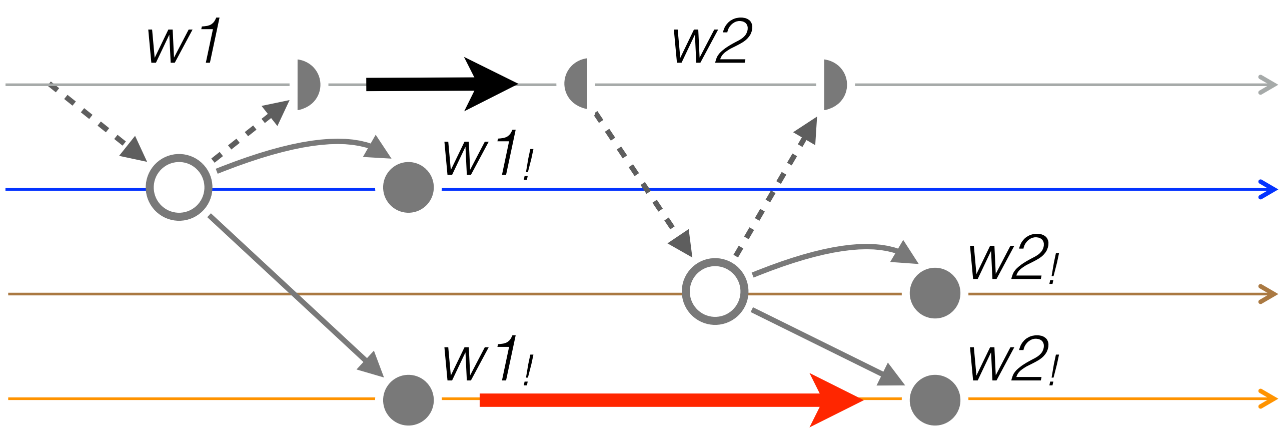


Client / No rollback: *r3* must include *w1*

vis; so ⊆ vis

Consistency Axioms

Monotonic writes

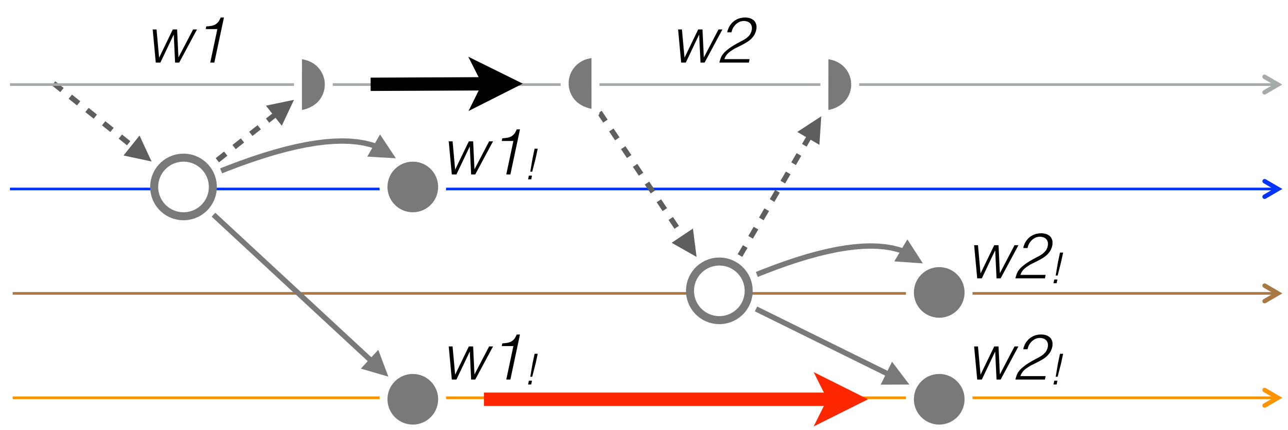


Global / No rollback: *r3* must include *w1*

$$\mathsf{ss} \cap (\mathsf{wr} \times \mathsf{wr}) \subseteq \mathsf{ar}$$

Consistency Axioms

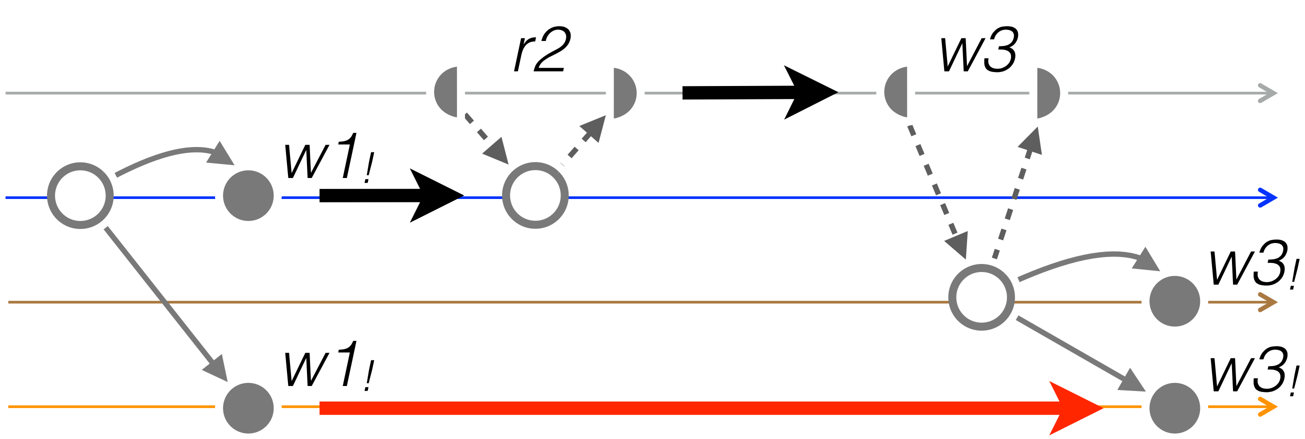
Monotonic writes



Global / No rollback: *r3* must include *w1*

$$ss \cap (wr \times wr) \subseteq ar$$

Writes Follow Reads



Global / WR dependence: *w3* must follow *w1*

Causal Visibility

$$hb = (vis \cup so)^+$$

$$hb \subseteq vis$$

Consistency Models

$$\textsc{ReadMyWrites} \stackrel{\text{def}}{=} (\mathsf{so} \subseteq \mathsf{vis})$$
$$\textsc{MonotonicReads} \stackrel{\text{def}}{=} (\mathsf{vis}\,;\mathsf{so}) \subseteq \mathsf{vis}$$
$$\textsc{ConsistentPrefix} \stackrel{\text{def}}{=} (\mathsf{ar}\,;(\mathsf{vis} \cap \neg\mathsf{ss})) \subseteq \mathsf{vis}$$

$$\textsc{NoCircularCausality} \stackrel{\text{def}}{=} \mathsf{acyclic}(\mathsf{hb})$$
$$\textsc{CausalVisibility} \stackrel{\text{def}}{=} (\mathsf{hb} \subseteq \mathsf{vis})$$
$$\textsc{CausalArbitration} \stackrel{\text{def}}{=} (\mathsf{hb} \subseteq \mathsf{ar})$$
$$\textsc{Causality} \stackrel{\text{def}}{=} \textsc{CausalVisibility} \wedge \textsc{CausalArbitration}$$

$$\textsc{SingleOrder} \stackrel{\text{def}}{=} \mathsf{vis} = \mathsf{ar}$$
$$\textsc{Realtime} \stackrel{\text{def}}{=} \mathsf{rb} \subseteq \mathsf{ar}$$

Principles of Eventual Consistency
Sebastian Burckhardt'14

Consistency Models

$$\textsc{Linearizability}(\mathcal{F}) = \textsc{SingleOrder} \wedge \textsc{RealTime} \wedge \textsc{RVal}(\mathcal{F})$$

$$\textsc{ReadMyWrites} \stackrel{\text{def}}{=} (\mathsf{so} \subseteq \mathsf{vis})$$
$$\textsc{MonotonicReads} \stackrel{\text{def}}{=} (\mathsf{vis}\,;\mathsf{so}) \subseteq \mathsf{vis}$$
$$\textsc{ConsistentPrefix} \stackrel{\text{def}}{=} (\mathsf{ar}\,;(\mathsf{vis} \cap \neg\mathsf{ss})) \subseteq \mathsf{vis}$$

$$\textsc{NoCircularCausality} \stackrel{\text{def}}{=} \mathsf{acyclic}(\mathsf{hb})$$
$$\textsc{CausalVisibility} \stackrel{\text{def}}{=} (\mathsf{hb} \subseteq \mathsf{vis})$$
$$\textsc{CausalArbitration} \stackrel{\text{def}}{=} (\mathsf{hb} \subseteq \mathsf{ar})$$
$$\textsc{Causality} \stackrel{\text{def}}{=} \textsc{CausalVisibility} \wedge \textsc{CausalArbitration}$$

$$\textsc{SingleOrder} \stackrel{\text{def}}{=} \mathsf{vis} = \mathsf{ar}$$
$$\textsc{RealTime} \stackrel{\text{def}}{=} \mathsf{rb} \subseteq \mathsf{ar}$$

Principles of Eventual Consistency
Sebastian Burckhardt'14

Consistency Models

$$\textsc{Linearizability}(\mathcal{F}) = \textsc{SingleOrder} \wedge \textsc{RealTime} \wedge \textsc{RVal}(\mathcal{F})$$

$$\textsc{Sequential Consistency}(\mathcal{F}) = \textsc{SingleOrder} \wedge \textsc{ReadMyWrites} \wedge \textsc{RVal}(\mathcal{F})$$

$$\textsc{ReadMyWrites} \stackrel{\text{def}}{=} (\mathsf{so} \subseteq \mathsf{vis})$$

$$\textsc{MonotonicReads} \stackrel{\text{def}}{=} (\mathsf{vis}\,;\mathsf{so}) \subseteq \mathsf{vis}$$

$$\textsc{ConsistentPrefix} \stackrel{\text{def}}{=} (\mathsf{ar}\,;(\mathsf{vis} \cap \neg\mathsf{ss})) \subseteq \mathsf{vis}$$

$$\textsc{NoCircularCausality} \stackrel{\text{def}}{=} \mathsf{acyclic}(\mathsf{hb})$$

$$\textsc{CausalVisibility} \stackrel{\text{def}}{=} (\mathsf{hb} \subseteq \mathsf{vis})$$

$$\textsc{CausalArbitration} \stackrel{\text{def}}{=} (\mathsf{hb} \subseteq \mathsf{ar})$$

$$\textsc{Causality} \stackrel{\text{def}}{=} \textsc{CausalVisibility} \wedge \textsc{CausalArbitration}$$

$$\textsc{SingleOrder} \stackrel{\text{def}}{=} \mathsf{vis} = \mathsf{ar}$$

$$\textsc{RealTime} \stackrel{\text{def}}{=} \mathsf{rb} \subseteq \mathsf{ar}$$

Principles of Eventual Consistency
Sebastian Burckhardt'14

Consistency Models

$$\textsc{Linearizability}(\mathcal{F}) = \textsc{SingleOrder} \wedge \textsc{RealTime} \wedge \textsc{RVal}(\mathcal{F})$$

$$\textsc{Sequential Consistency}(\mathcal{F}) = \textsc{SingleOrder} \wedge \textsc{ReadMyWrites} \wedge \textsc{RVal}(\mathcal{F})$$

$$\textsc{Causal Consistency}(\mathcal{F}) = \textsc{Causality} \wedge \textsc{RVal}(\mathcal{F})$$

$$\textsc{ReadMyWrites} \stackrel{\text{def}}{=} (\mathsf{so} \subseteq \mathsf{vis})$$

$$\textsc{MonotonicReads} \stackrel{\text{def}}{=} (\mathsf{vis}\,;\mathsf{so}) \subseteq \mathsf{vis}$$

$$\textsc{ConsistentPrefix} \stackrel{\text{def}}{=} (\mathsf{ar}\,;(\mathsf{vis} \cap \neg\mathsf{ss})) \subseteq \mathsf{vis}$$

$$\textsc{NoCircularCausality} \stackrel{\text{def}}{=} \mathsf{acyclic}(\mathsf{hb})$$

$$\textsc{CausalVisibility} \stackrel{\text{def}}{=} (\mathsf{hb} \subseteq \mathsf{vis})$$

$$\textsc{CausalArbitration} \stackrel{\text{def}}{=} (\mathsf{hb} \subseteq \mathsf{ar})$$

$$\textsc{Causality} \stackrel{\text{def}}{=} \textsc{CausalVisibility} \wedge \textsc{CausalArbitration}$$

$$\textsc{SingleOrder} \stackrel{\text{def}}{=} \mathsf{vis} = \mathsf{ar}$$

$$\textsc{RealTime} \stackrel{\text{def}}{=} \mathsf{rb} \subseteq \mathsf{ar}$$

Principles of Eventual Consistency
Sebastian Burckhardt'14

Consistency Models

$$\textsc{Linearizability}(\mathcal{F}) = \textsc{SingleOrder} \wedge \textsc{RealTime} \wedge \textsc{RVal}(\mathcal{F})$$

$$\textsc{Sequential Consistency}(\mathcal{F}) = \textsc{SingleOrder} \wedge \textsc{ReadMyWrites} \wedge \textsc{RVal}(\mathcal{F})$$

$$\textsc{Causal Consistency}(\mathcal{F}) = \textsc{Causality} \wedge \textsc{RVal}(\mathcal{F})$$

$$\textsc{Eventual Consistency}(\mathcal{F}) = \textsc{NoCircularCausality} \wedge \textsc{RVal}(\mathcal{F})$$

$$\textsc{ReadMyWrites} \stackrel{\text{def}}{=} (\mathsf{so} \subseteq \mathsf{vis})$$
$$\textsc{MonotonicReads} \stackrel{\text{def}}{=} (\mathsf{vis}\,;\mathsf{so}) \subseteq \mathsf{vis}$$
$$\textsc{ConsistentPrefix} \stackrel{\text{def}}{=} (\mathsf{ar}\,;(\mathsf{vis} \cap \neg\mathsf{ss})) \subseteq \mathsf{vis}$$

$$\textsc{NoCircularCausality} \stackrel{\text{def}}{=} \mathsf{acyclic}(\mathsf{hb})$$
$$\textsc{CausalVisibility} \stackrel{\text{def}}{=} (\mathsf{hb} \subseteq \mathsf{vis})$$
$$\textsc{CausalArbitration} \stackrel{\text{def}}{=} (\mathsf{hb} \subseteq \mathsf{ar})$$
$$\textsc{Causality} \stackrel{\text{def}}{=} \textsc{CausalVisibility} \wedge \textsc{CausalArbitration}$$

$$\textsc{SingleOrder} \stackrel{\text{def}}{=} \mathsf{vis} = \mathsf{ar}$$
$$\textsc{RealTime} \stackrel{\text{def}}{=} \mathsf{rb} \subseteq \mathsf{ar}$$

Principles of Eventual Consistency
Sebastian Burckhardt'14

Strong vs. weak?

Predictable

Low performance

LINEARIZABILITY

↑

SEQUENTIAL CONSISTENCY

↑

CAUSAL CONSISTENCY

↑

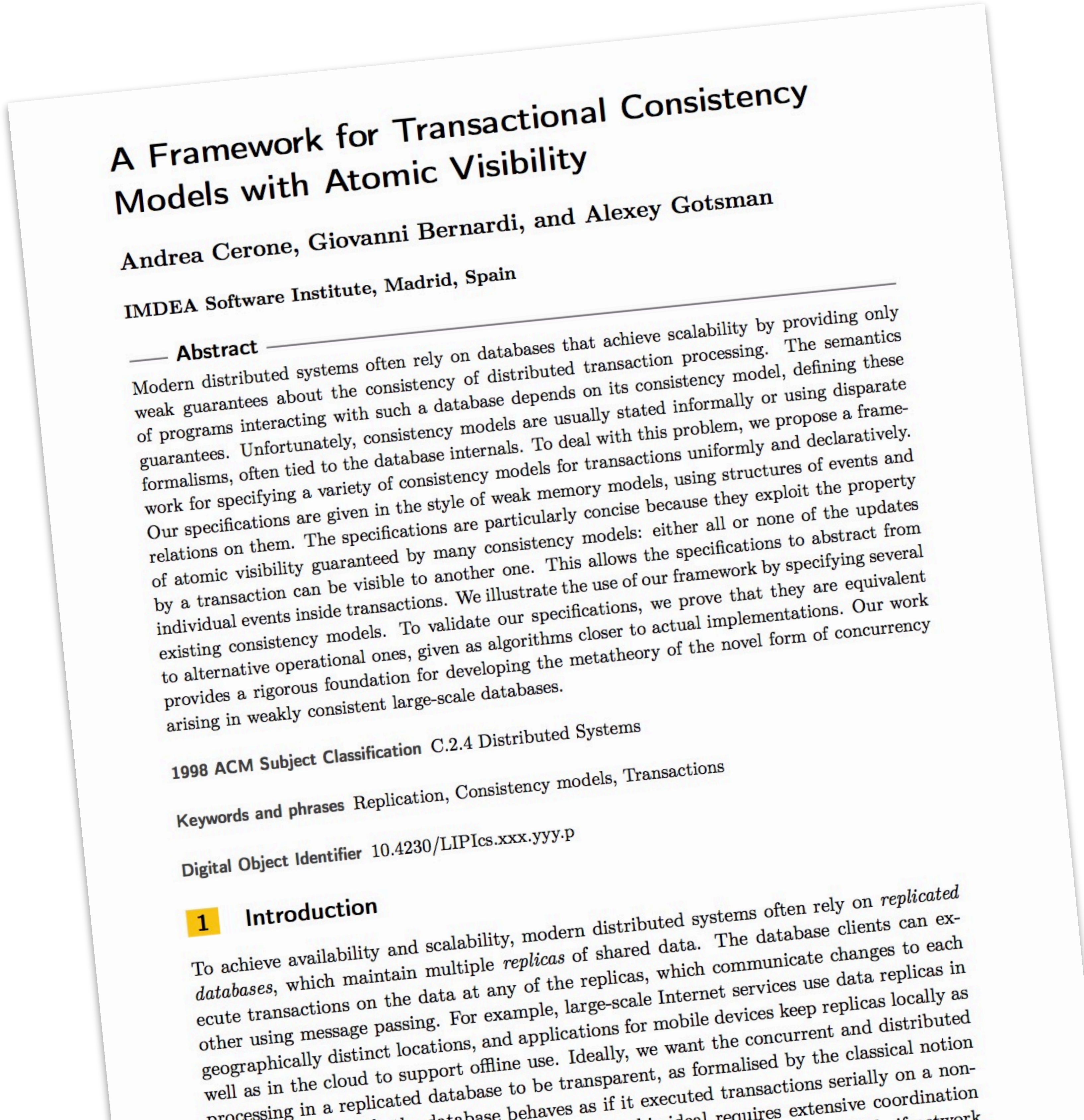
EVENTUAL CONSISTENCY

Hard to program

High performance

Replicated Data Bases

Replicated Data Bases



A Framework for Transactional Consistency Models with Atomic Visibility

Andrea Cerone, Giovanni Bernardi, and Alexey Gotsman

IMDEA Software Institute, Madrid, Spain

Abstract

Modern distributed systems often rely on databases that achieve scalability by providing only weak guarantees about the consistency of distributed transaction processing. The semantics of programs interacting with such a database depends on its consistency model, defining these guarantees. Unfortunately, consistency models are usually stated informally or using disparate formalisms, often tied to the database internals. To deal with this problem, we propose a framework for specifying a variety of consistency models for transactions uniformly and declaratively. Our specifications are given in the style of weak memory models, using structures of events and relations on them. The specifications are particularly concise because they exploit the property of atomic visibility guaranteed by many consistency models: either all or none of the updates by a transaction can be visible to another one. This allows the specifications to abstract from individual events inside transactions. We illustrate the use of our framework by specifying several existing consistency models. To validate our specifications, we prove that they are equivalent to alternative operational ones, given as algorithms closer to actual implementations. Our work provides a rigorous foundation for developing the metatheory of the novel form of concurrency arising in weakly consistent large-scale databases.

1998 ACM Subject Classification C.2.4 Distributed Systems

Keywords and phrases Replication, Consistency models, Transactions

Digital Object Identifier 10.4230/LIPIcs.xxx.yyy.p

1 Introduction

To achieve availability and scalability, modern distributed systems often rely on *replicated databases*, which maintain multiple *replicas* of shared data. The database clients can execute transactions on the data at any of the replicas, which communicate changes to each other using message passing. For example, large-scale Internet services use data replicas in geographically distinct locations, and applications for mobile devices keep replicas locally as well as in the cloud to support offline use. Ideally, we want the concurrent and distributed processing in a replicated database to be transparent, as formalised by the classical notion

Replicated Data Bases

- Operations are transactions
- Each transaction issues a number of reads and writes
- Writes are ordered by program order (po)
- Visibility and Arbitration relate “transactions” instead of individual reads and writes
- We consider only the case of a key-value store data base

Replicated Data Bases

Replicated Data Bases

Transaction $T = (E, \mathsf{po})$

Replicated Data Bases

Transaction $T = (E, \mathsf{po})$

History $H = \{T_0, T_1, ..., T_n\}$

Replicated Data Bases

Transaction $T = (E, \mathsf{po})$

History $H = \{T_0, T_1, ..., T_n\}$

Abstract Execution $A = (H, \mathsf{vis}, \mathsf{ar})$

Replicated Data Bases

Transaction $T = (E, \mathsf{po})$

History $H = \{T_0, T_1, ..., T_n\}$

Abstract Execution $A = (H, \mathsf{vis}, \mathsf{ar})$

These are relation on transactions now

$$T_1 \xrightarrow{\mathsf{vis}} T_2$$

Replicated Data Bases

Consistency Axioms

Replicated Data Bases

Consistency Axioms

Read Consistency

- Every read in transaction T sees a visible write to T that is the maximum visible write according to arbitration

$$
\begin{aligned}
rd(x,v) \in E(T) \Rightarrow \exists T', \quad & T' \in \mathsf{vis}_H^{-1}(T) \wedge \\
& wr(x,1) \in E(T') \wedge \\
& max_{\mathsf{ar}}(\mathsf{vis}_H^{-1}(T) \cap Writes_H(x)) = T'
\end{aligned}
$$

Replicated Data Bases

Consistency Axioms

Read Consistency

- Every read in transaction T sees a visible write to T that is the maximum visible write according to arbitration

$$rd(x,v) \in E(T) \Rightarrow \exists T', \quad \begin{aligned} & T' \in \mathsf{vis}_H^{-1}(T) \wedge \\ & wr(x,1) \in E(T') \wedge \\ & max_{\mathsf{ar}}(\mathsf{vis}_H^{-1}(T) \cap Writes_H(x)) = T' \end{aligned}$$

Transitive Visibility $\quad \mathsf{vis}^{+} \subseteq \mathsf{vis}$

Replicated Data Bases

Consistency Axioms

Read Consistency

- Every read in transaction T sees a visible write to T that is the maximum visible write according to arbitration

$$rd(x,v) \in E(T) \Rightarrow \exists T', \quad \begin{array}{l} T' \in \mathsf{vis}_H^{-1}(T) \wedge \\ wr(x,1) \in E(T') \wedge \\ max_{\mathsf{ar}}(\mathsf{vis}_H^{-1}(T) \cap Writes_H(x)) = T' \end{array}$$

Transitive Visibility $\quad \mathsf{vis}^+ \subseteq \mathsf{vis}$

Prefix Consistent $\quad \mathsf{ar}; \mathsf{vis} \subseteq \mathsf{vis}$

Replicated Data Bases

Consistency Axioms

Read Consistency

- Every read in transaction T sees a visible write to T that is the maximum visible write according to arbitration

$$rd(x,v) \in E(T) \Rightarrow \exists T', \quad \begin{array}{l} T' \in \mathsf{vis}_H^{-1}(T) \wedge \\ wr(x,1) \in E(T') \wedge \\ max_{\mathsf{ar}}(\mathsf{vis}_H^{-1}(T) \cap Writes_H(x)) = T' \end{array}$$

Transitive Visibility $\quad \mathsf{vis}^+ \subseteq \mathsf{vis}$

Prefix Consistent $\quad \mathsf{ar};\mathsf{vis} \subseteq \mathsf{vis}$

Total Visibility $\quad \forall T_1, T2.\ T_1 \xrightarrow{\mathsf{vis}} T_2 \ \vee\ T_2 \xrightarrow{\mathsf{vis}} T_1$

Replicated Data Bases

Consistency Axioms

Read Consistency

- Every read in transaction T sees a visible write to T that is the maximum visible write according to arbitration

$$rd(x,v) \in E(T) \Rightarrow \exists T', \quad \begin{array}{l} T' \in \mathsf{vis}_H^{-1}(T) \wedge \\ wr(x,1) \in E(T') \wedge \\ max_{\mathsf{ar}}(\mathsf{vis}_H^{-1}(T) \cap Writes_H(x)) = T' \end{array}$$

Transitive Visibility $\quad \mathsf{vis}^+ \subseteq \mathsf{vis}$

Prefix Consistent $\quad \mathsf{ar};\mathsf{vis} \subseteq \mathsf{vis}$

Total Visibility $\quad \forall T_1, T2.\ T_1 \xrightarrow{\mathsf{vis}} T_2 \ \vee \ T_2 \xrightarrow{\mathsf{vis}} T_1$

No Conflict $\quad \{T_1, T_2\} \subseteq Writes_H(x) \Rightarrow T_1 \xrightarrow{\mathsf{vis}} T_2 \ \vee \ T_2 \xrightarrow{\mathsf{vis}} T_1$

Replicated Data Bases

Consistency Models

Φ	Consistency model	Axioms (Figure 2)	Fractured reads	Causality violation	Lost update	Long fork	Write skew
RA	Read Atomic [6]	Int, Ext	✘	✓	✓	✓	✓
CC	Causal consistency [12, 19]	Int, Ext, TransVis	✘	✘	✓	✓	✓
PSI	Parallel snapshot isolation [21, 24]	Int, Ext, TransVis, NoConflict	✘	✘	✘	✓	✓
PC	Prefix consistency [13]	Int, Ext, Prefix	✘	✘	✓	✘	✓
SI	Snapshot isolation [8]	Int, Ext, Prefix, NoConflict	✘	✘	✘	✘	✓
SER	Serialisability [20]	Int, Ext, TotalVis	✘	✘	✘	✘	✘

RA
∩
CC
PC **PSI**
SI
∩
SER

Replicated Data Bases

Consistency Models

Φ	Consistency model	Axioms (Figure 2)	Fractured reads	Causality violation	Lost update	Long fork	Write skew
RA	Read Atomic [6]	Int, Ext	✘	✓	✓	✓	✓
CC	Causal consistency [12, 19]	Int, Ext, TransVis	✘	✘	✓	✓	✓
PSI	Parallel snapshot isolation [21, 24]	Int, Ext, TransVis, NoConflict	✘	✘	✘	✓	✓
PC	Prefix consistency [13]	Int, Ext, Prefix	✘	✘	✓	✘	✓
SI	Snapshot isolation [8]	Int, Ext, Prefix, NoConflict	✘	✘	✘	✘	✓
SER	Serialisability [20]	Int, Ext, TotalVis	✘	✘	✘	✘	✘

RA

CC

PC **PSI**

SI

SER

$$\mathsf{CC} \equiv \textsc{Int} \wedge \textsc{Ext} \wedge \textsc{SerTotal}$$

$$\mathsf{PSI} \equiv \mathsf{CC} \wedge \textsc{Conflict}$$

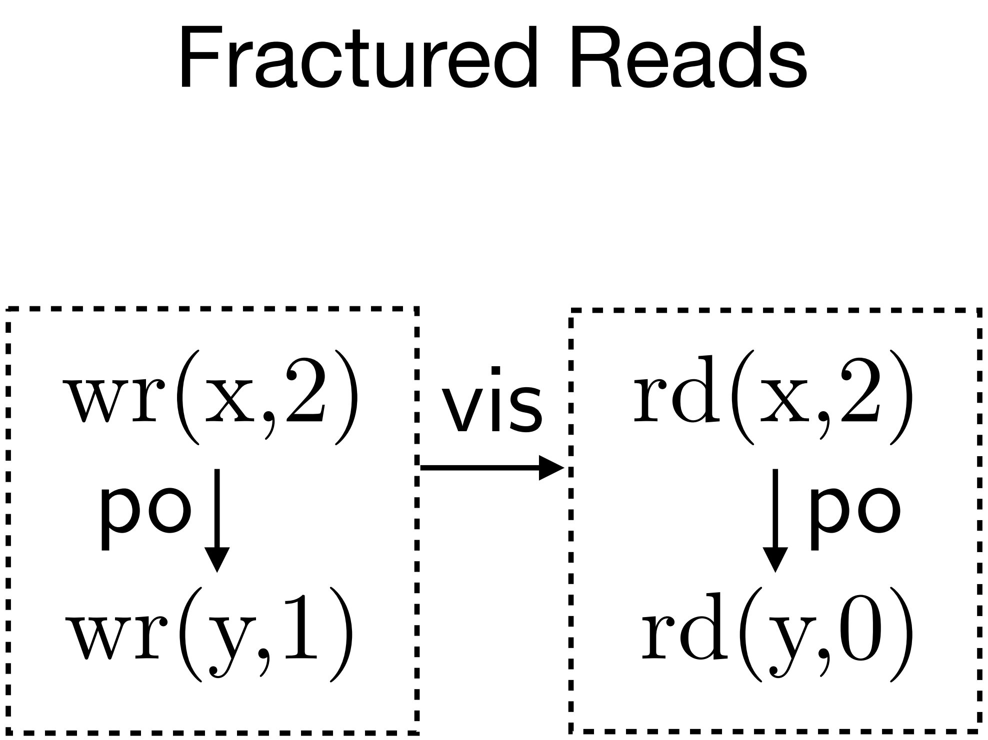
$$\mathsf{PC} \equiv \mathsf{CC} \wedge \textsc{Prefix}$$

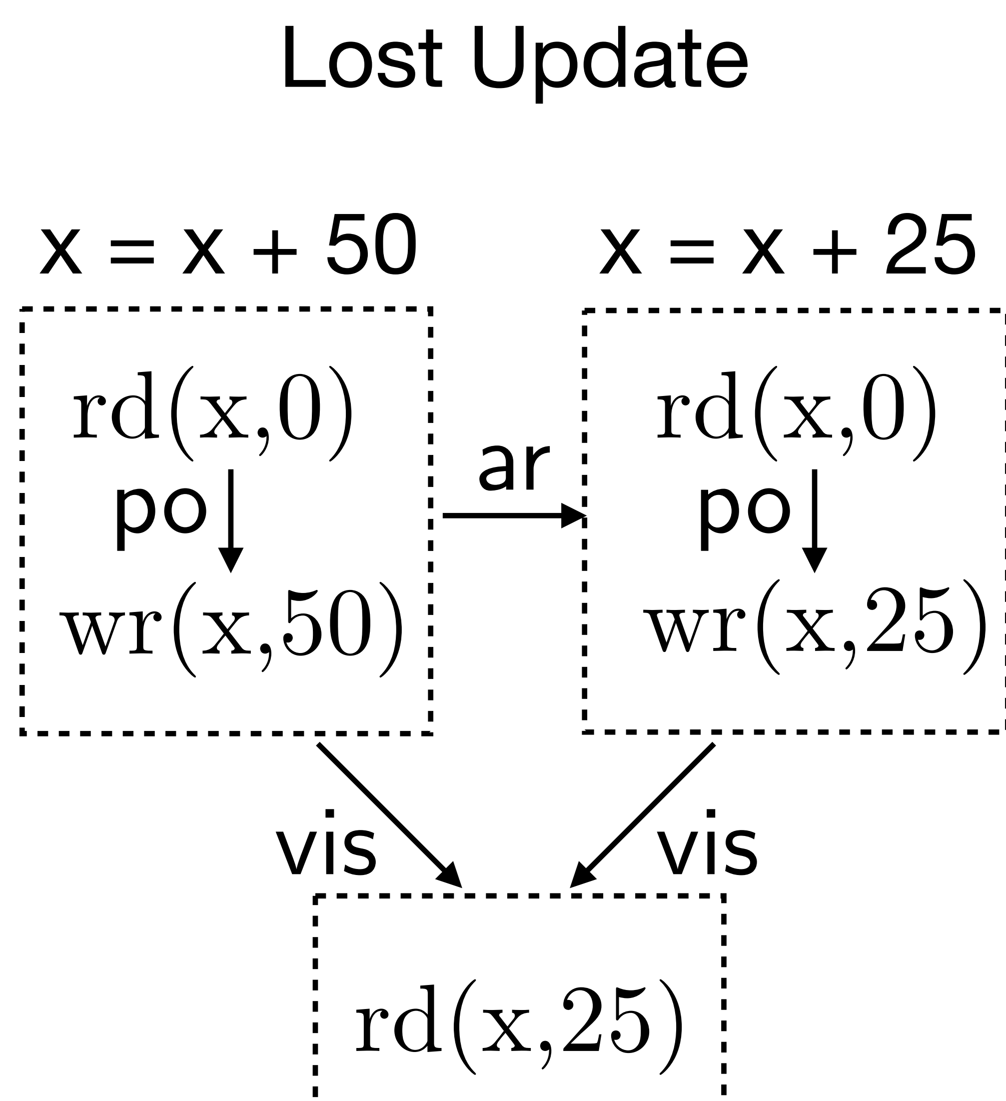
$$\mathsf{SI} \equiv \mathsf{PSI} \wedge \textsc{Prefix}$$

$$\mathsf{SER} \equiv \textsc{Int} \wedge \textsc{Ext} \wedge \textsc{TotalHB}$$

Replicated Data Bases

Anomalies

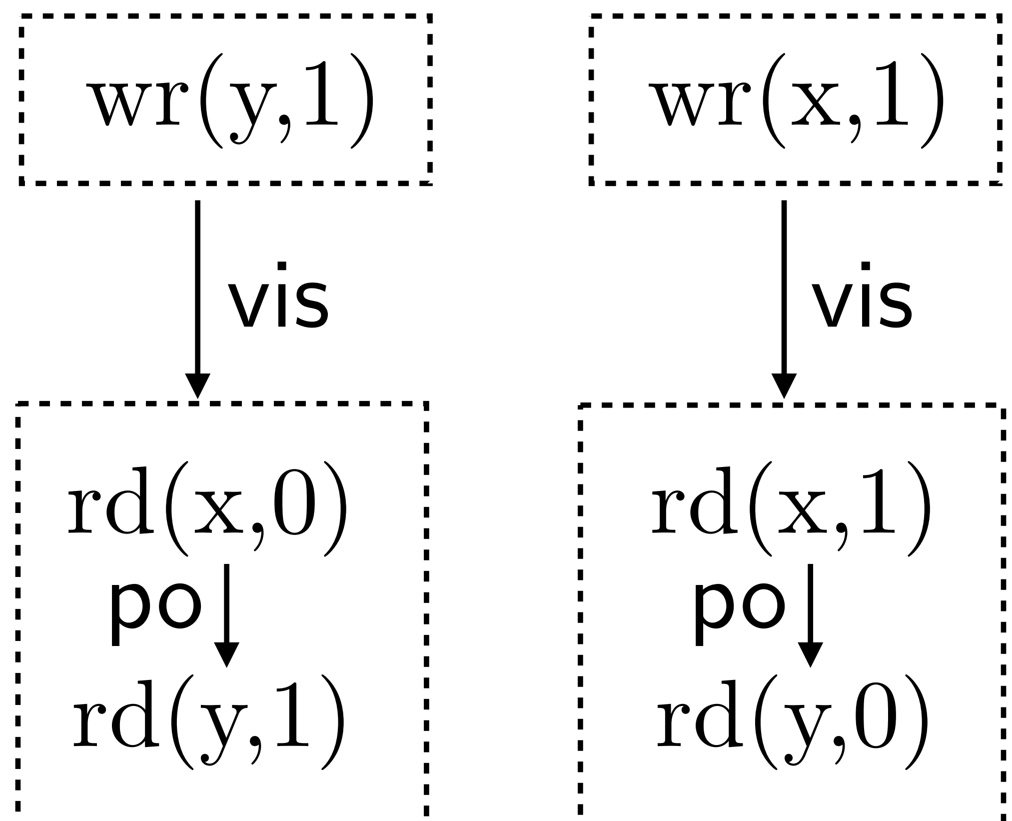




Replicated Data Bases

Anomalies

Long Fork (IRIW)



Tiny Demo

A Forest of Models

Strong vs. weak?

Predictable

Strict Serializability

Low performance

Hard to program

Eventual Consistency

High performance

Strong vs. weak?

Predictable

Strict Serializability

Low performance

Snapshot Isolation

Hard to program

Eventual Consistency

High performance

Strong vs. weak?

Predictable

Low performance

Strict Serializability

Snapshot Isolation

PRAM

Eventual Consistency

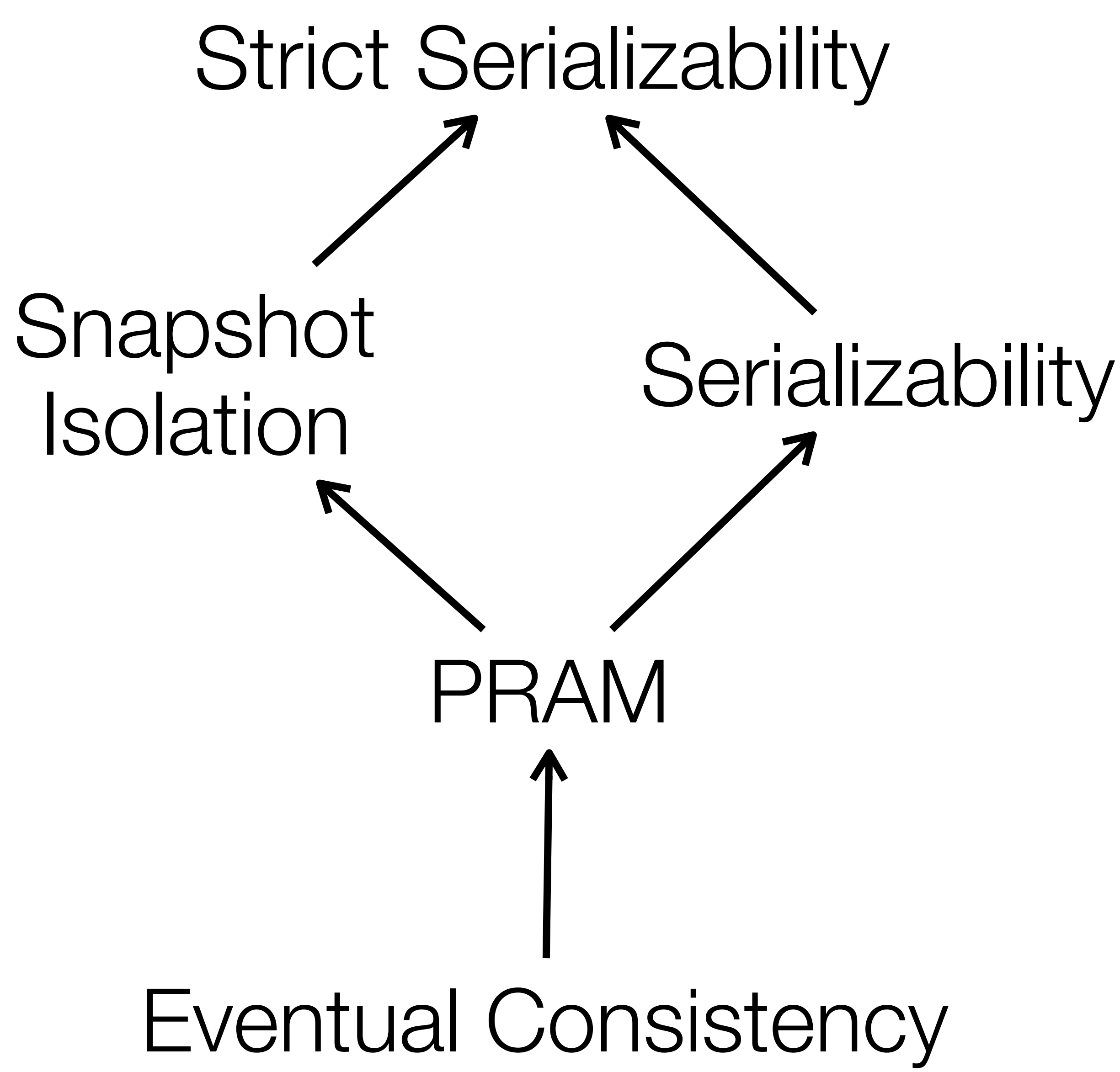
Hard to program

High performance

Strong vs. weak?

Predictable

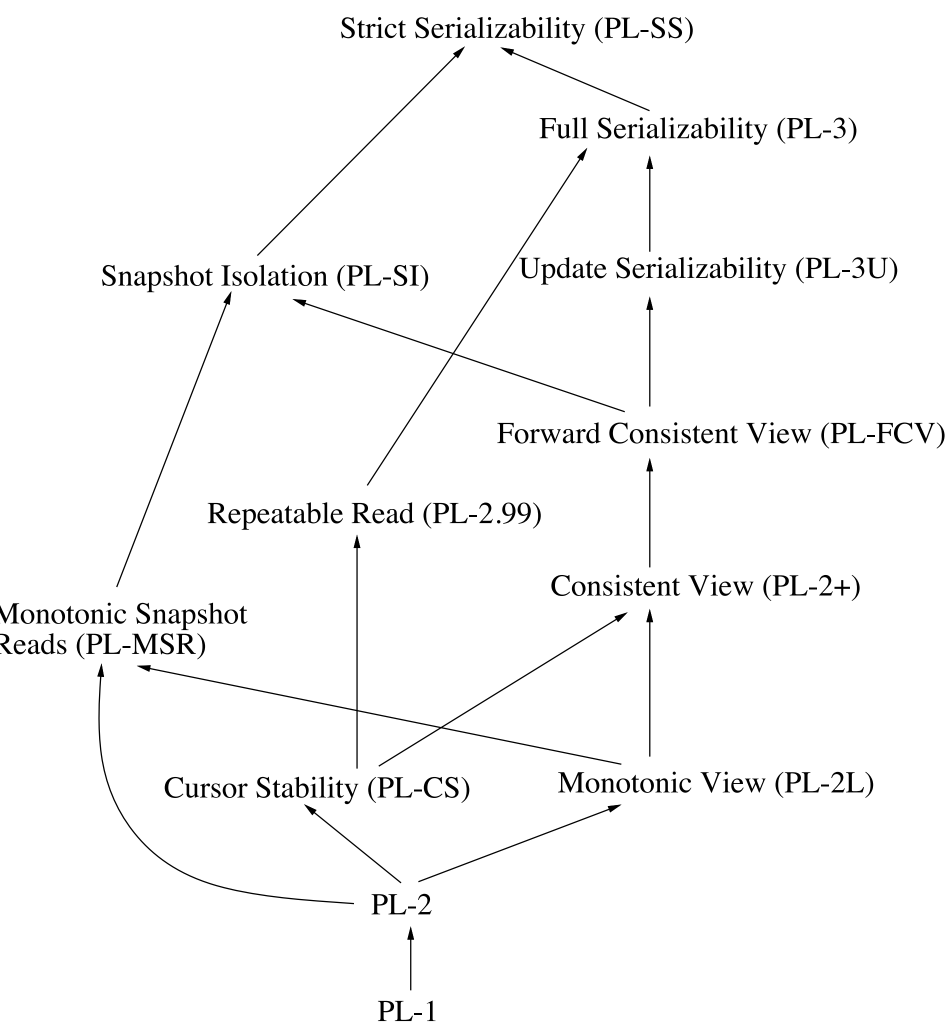
Low performance



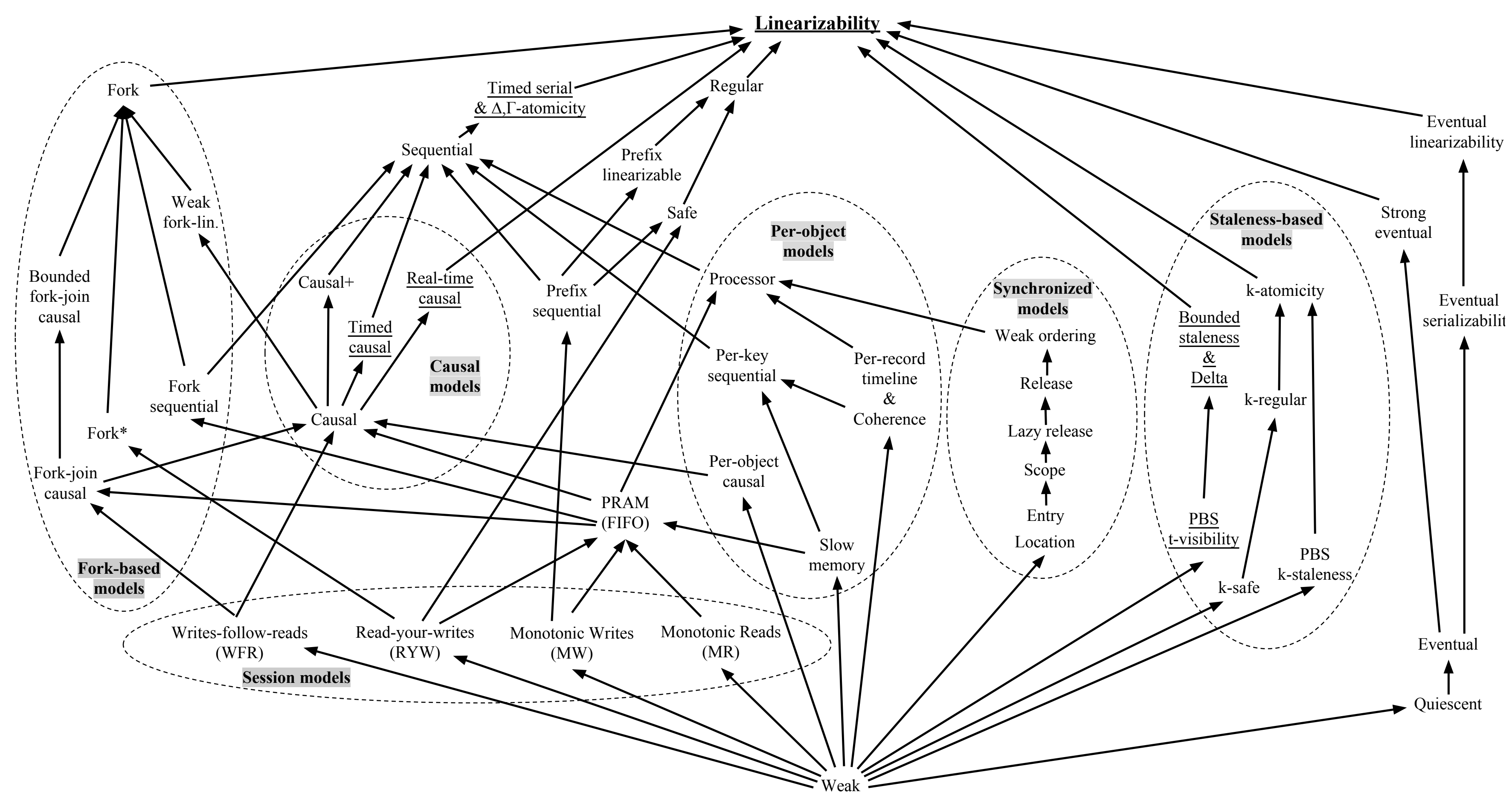
Hard to program

High performance

Strong vs. weak?



Transactional
Adya 1999



Non-transactional
Viotti & Vukolić 2016

Consistency & Invariants

Consistency & Invariants

Consistency in 3D*

Marc Shapiro[1], Masoud Saeida Ardekani[2], and Gustavo Petri[3]

1 Sorbonne-Universités-UPMC-LIP6 & Inria Paris

2 Purdue University †

3 IRIF, Université Paris Diderot

Abstract

Comparisons of different consistency models often try to place them in a linear strong-to-weak order. However this view is clearly inadequate, since it is well known, for instance, that Snapshot Isolation and Serialisability are incomparable. In the interest of a better understanding, we propose a new classification, along three dimensions, related to: a total order of writes, a causal order of reads, and transactional composition of multiple operations. A model may be stronger than another on one dimension and weaker on another. We believe that this new classification scheme is both scientifically sound and has good explicative value. The current paper presents the three-dimensional design space intuitively.

1998 ACM Subject Classification C.2.4 Distributed databases; D.1.3 Concurrent programming; D.2.4 Software/Program Verification; E.1 Distributed data structures

Keywords and phrases Consistency models; Replicated data; Structural invariants; Correctness of distributed systems;

Digital Object Identifier 10.4230/LIPIcs.CONCUR.2016.<article-no>

1 Introduction

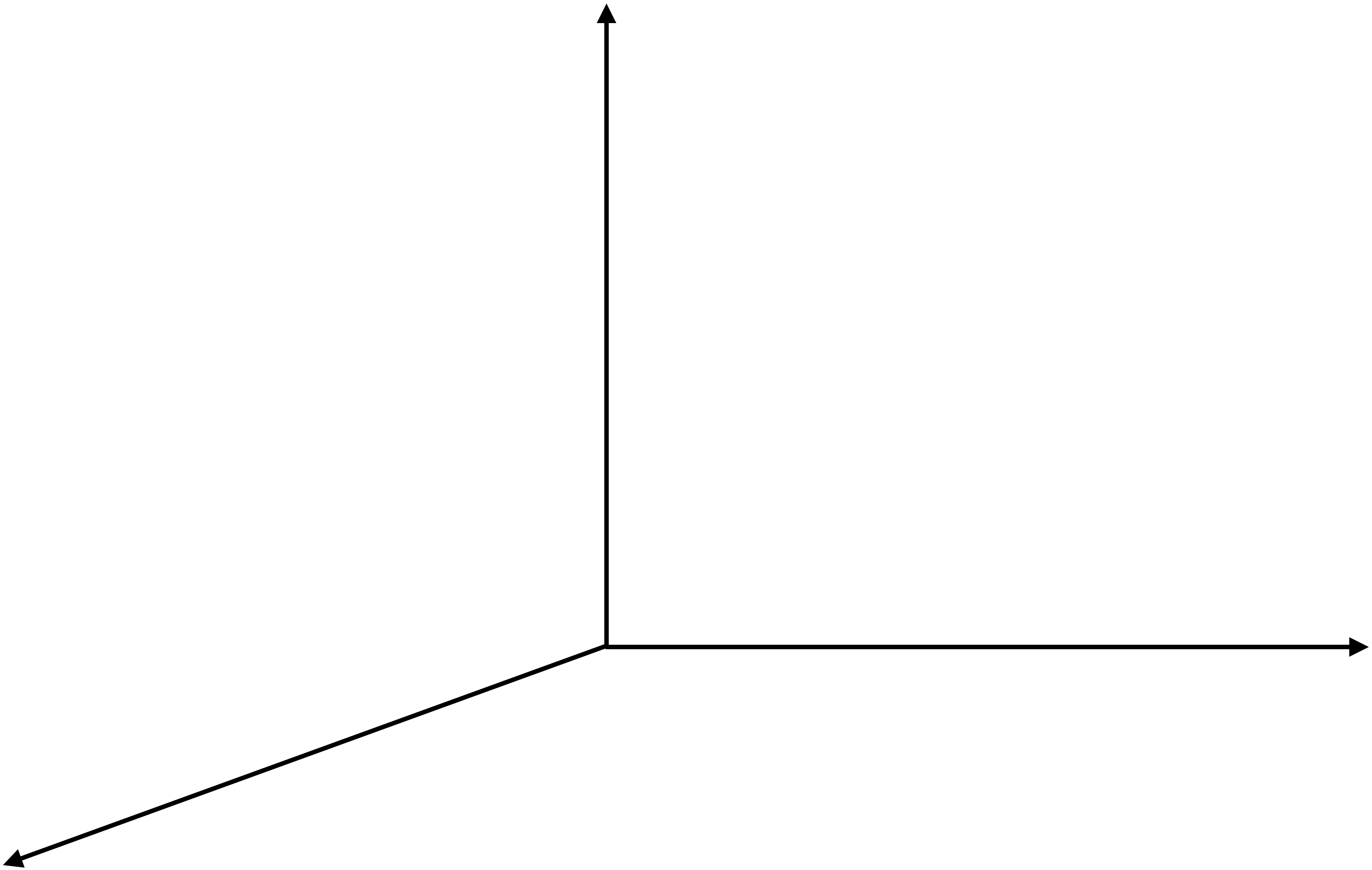
Consistency & Invariants

- Consistency in 3D
  - Characterization of consistency models according to the guarantees they provide
- Dimensions of Guarantees
  - Single object
  - Propagation of effects on different objects
  - Composition of objects

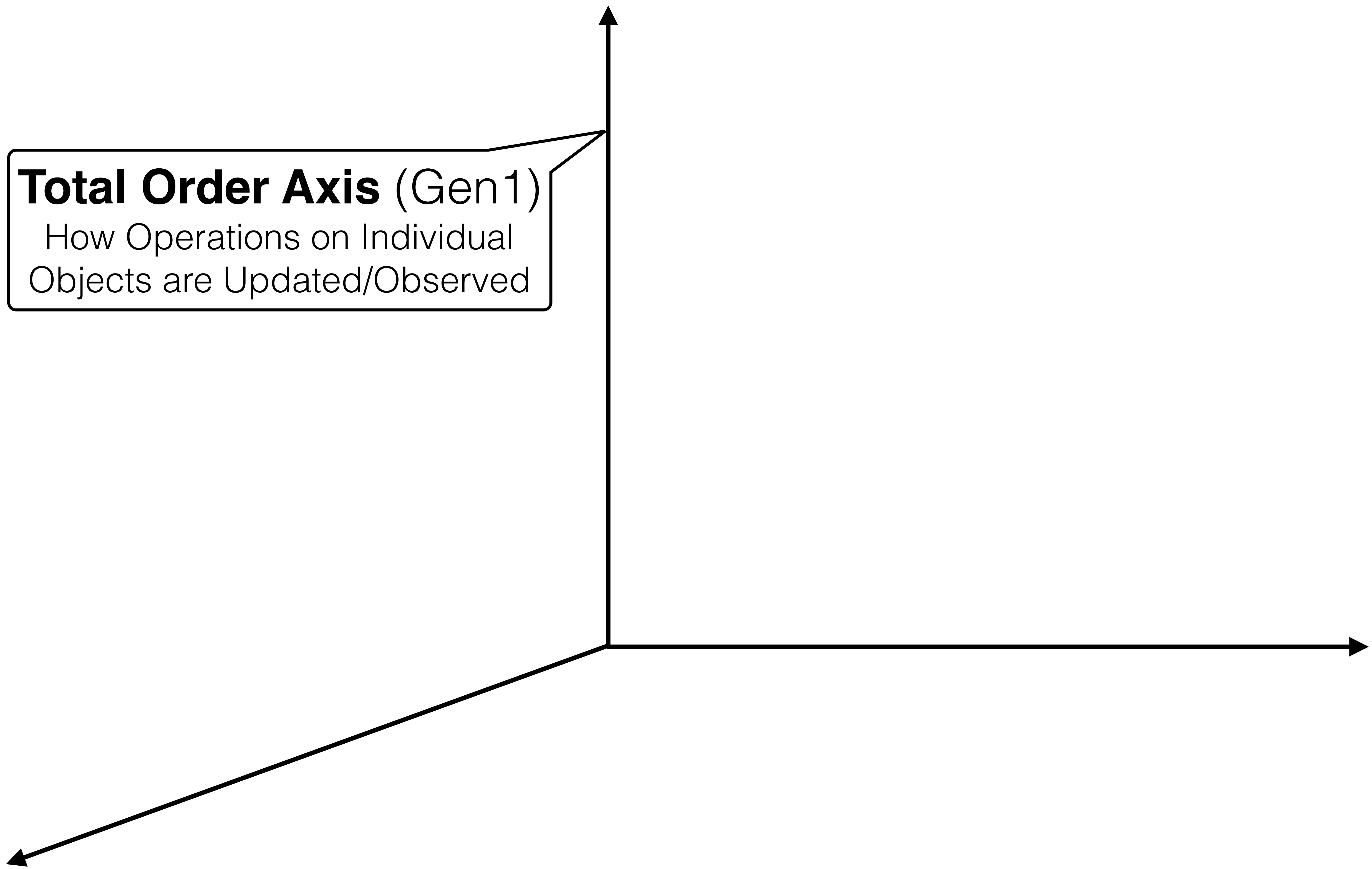
Three classes…

	…of invariant	… of protocol
Gen1	Constrain value of an object	Total order of operations
PO	Ordering between operations	Visibility
EQ	State equivalence between objects	Composition

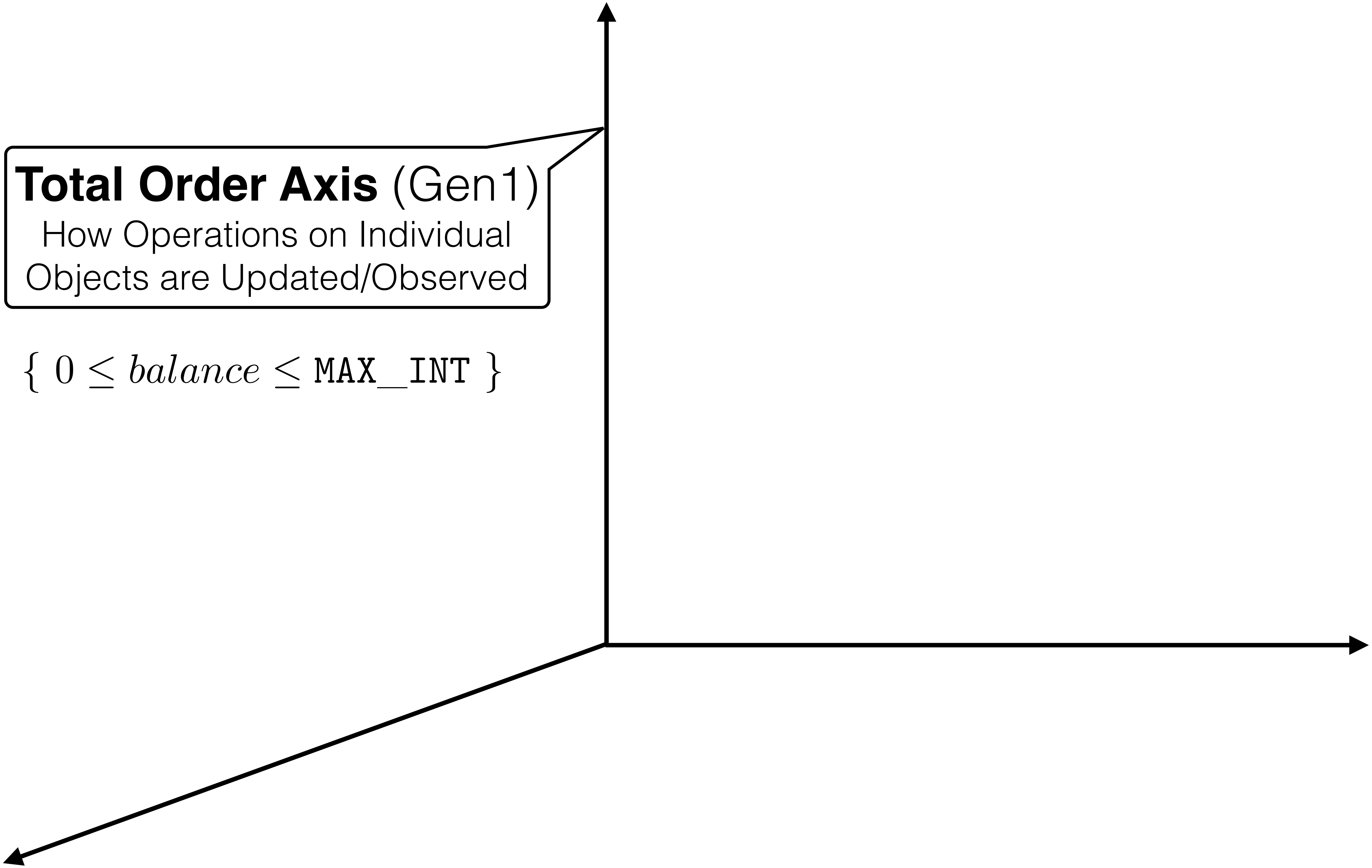
Consistency in 3D



Consistency in 3D

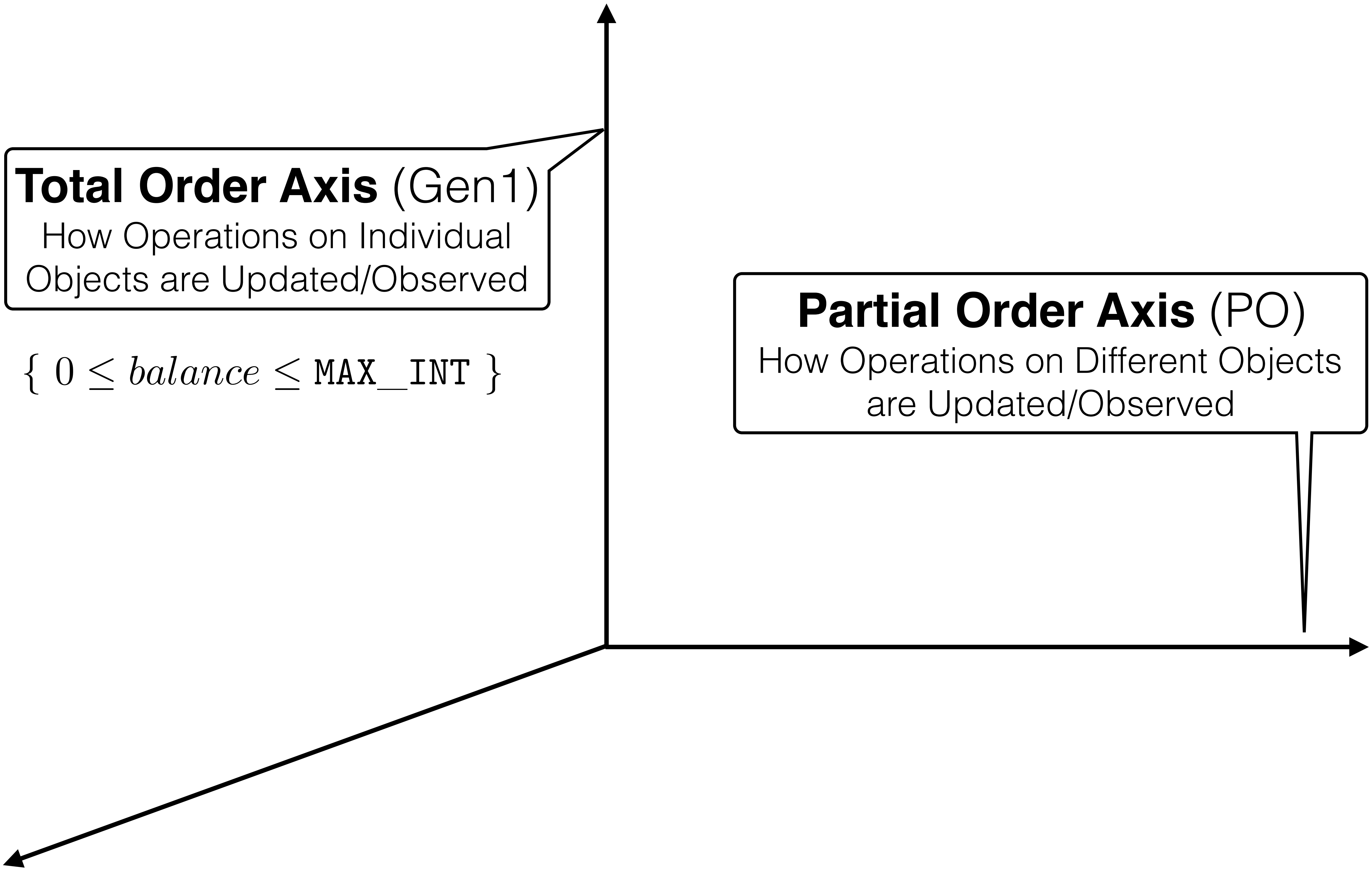


Consistency in 3D

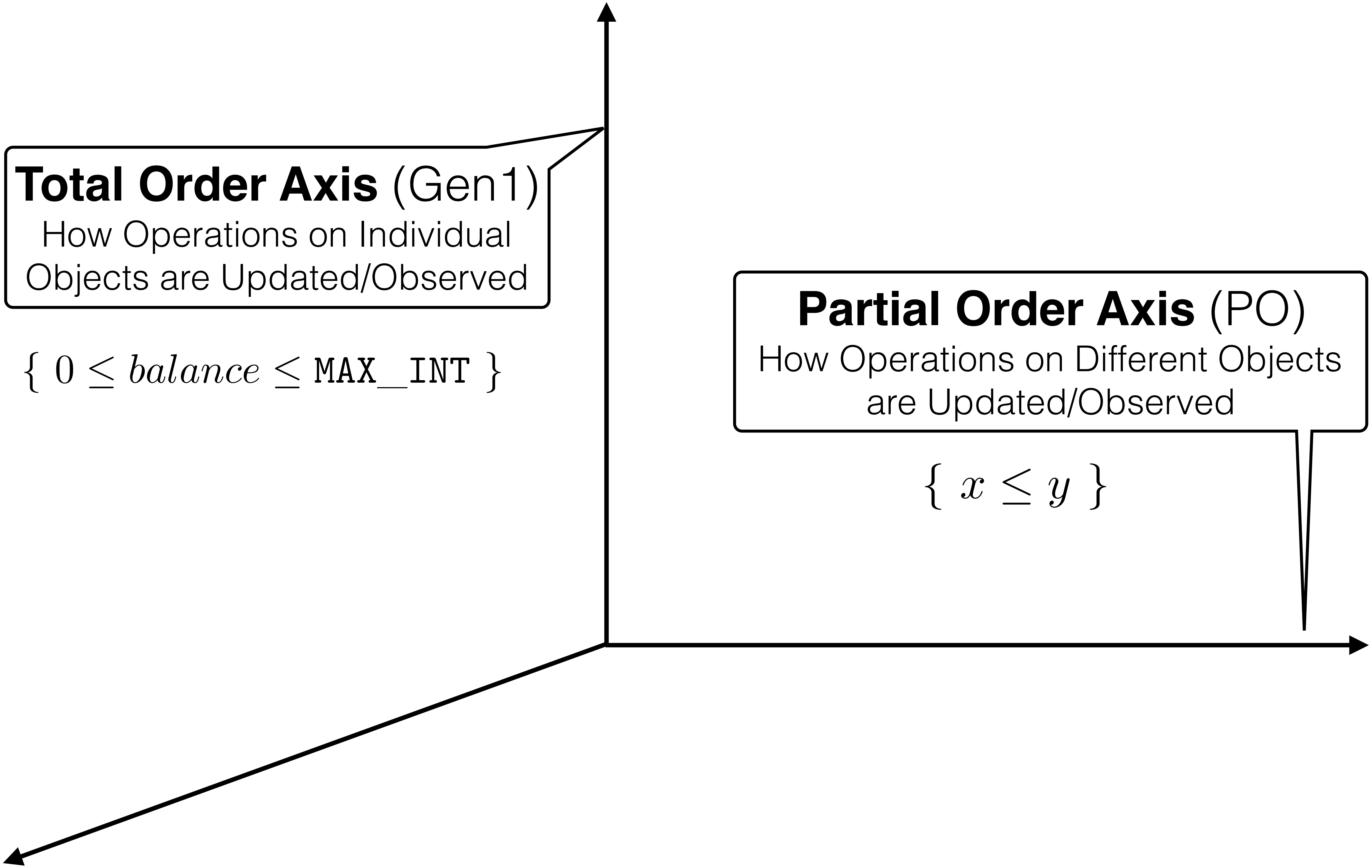


$$\{\ 0 \leq balance \leq \texttt{MAX\_INT}\ \}$$

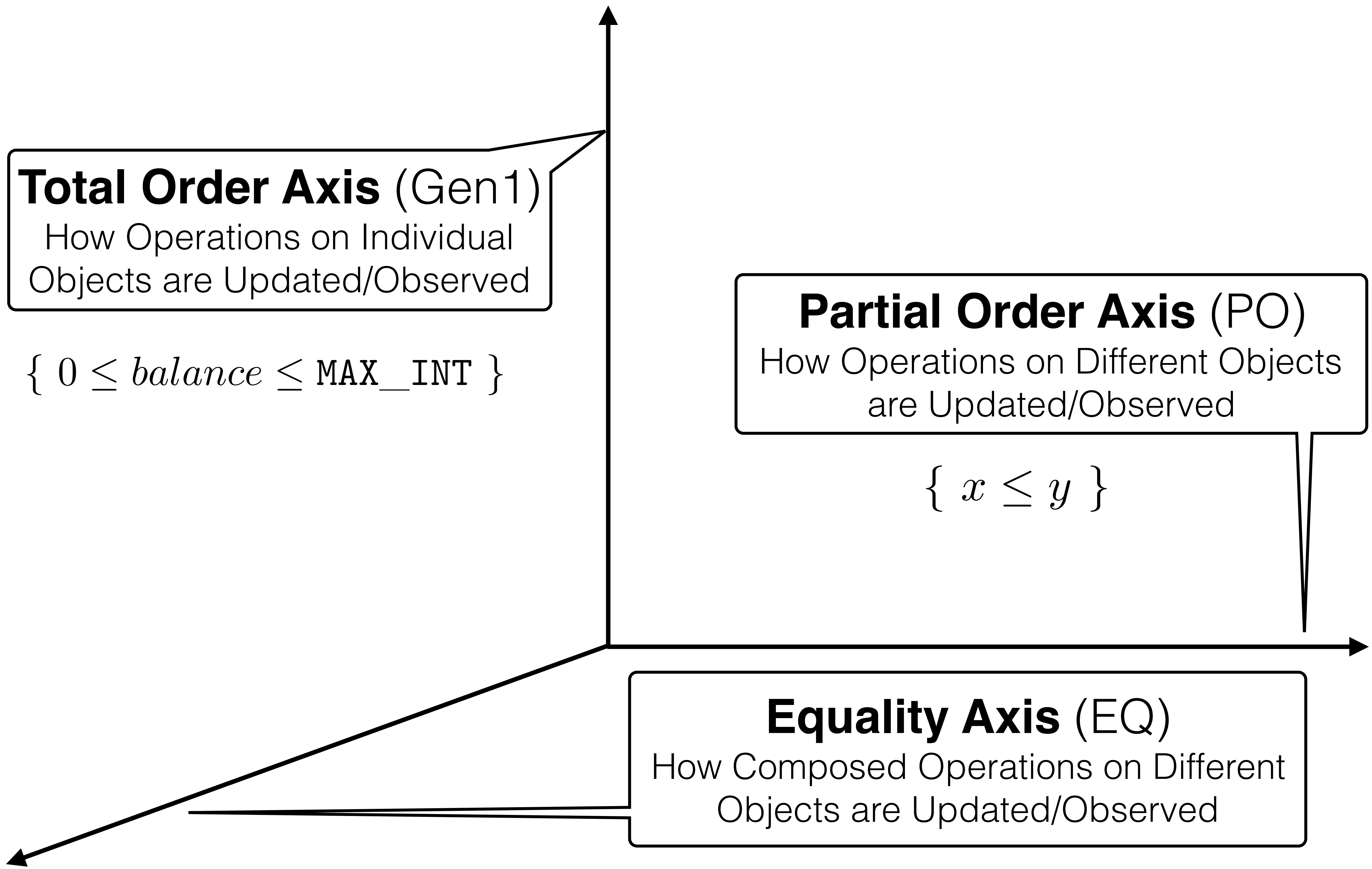
Consistency in 3D



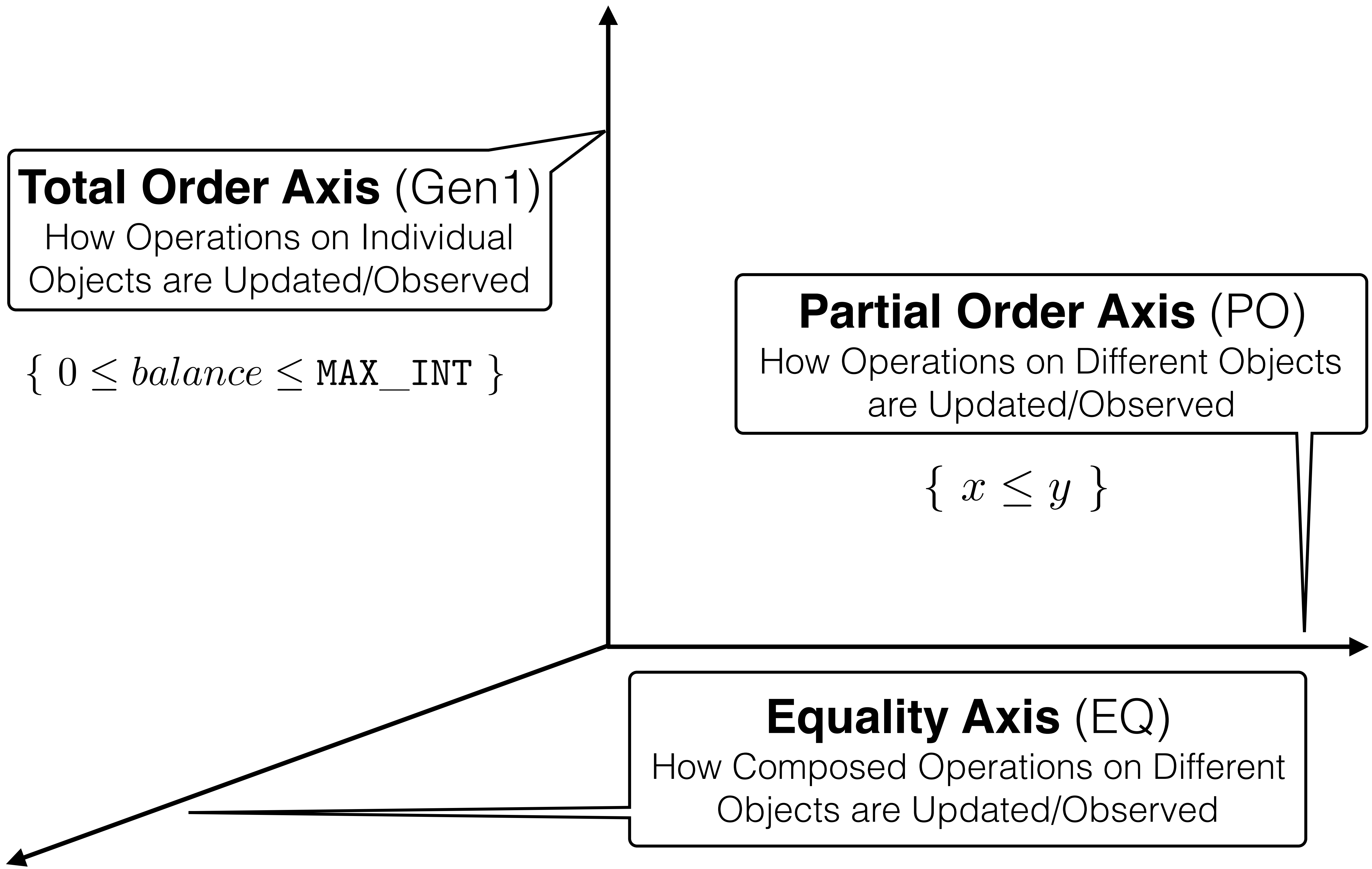
Consistency in 3D



Consistency in 3D

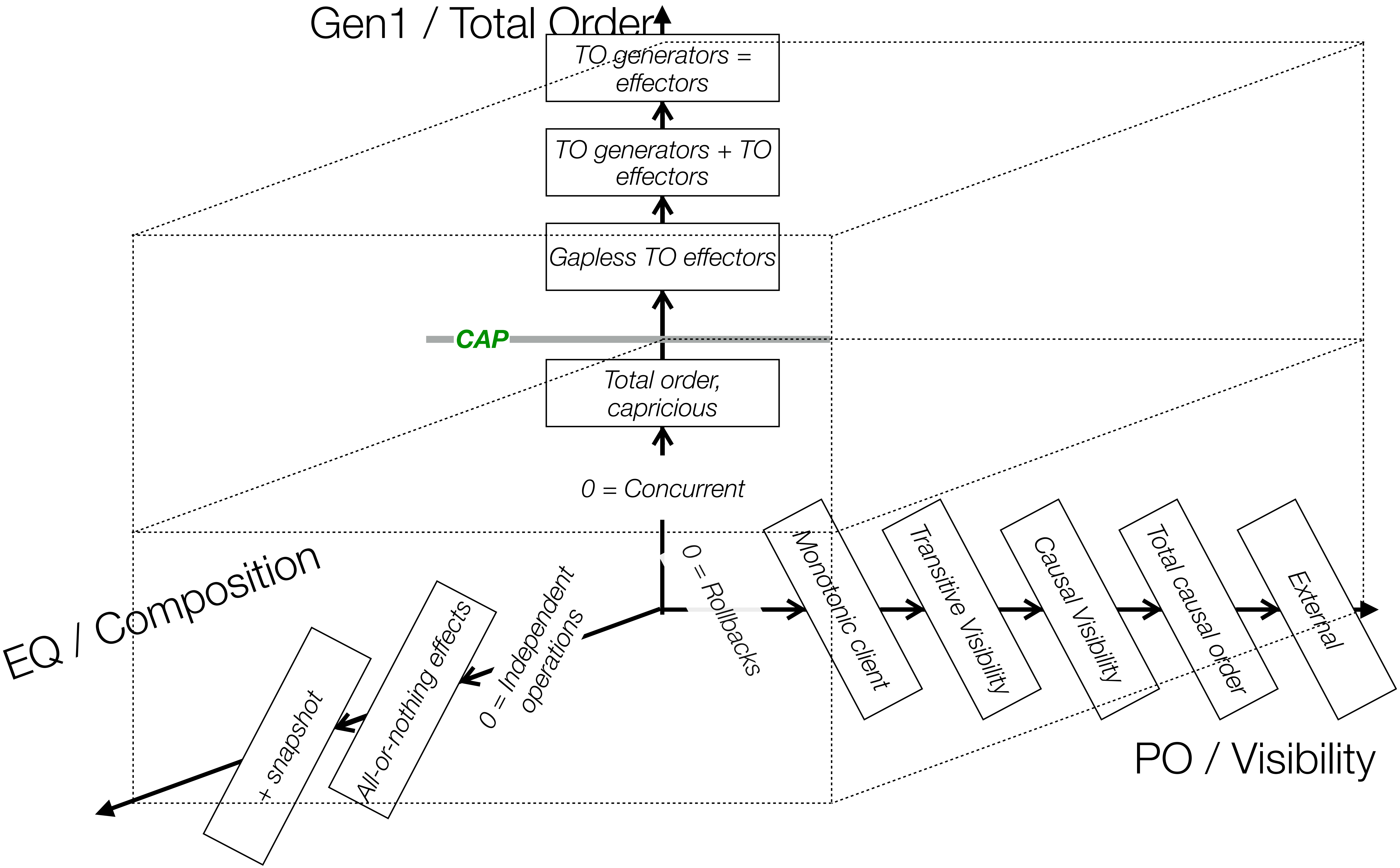


Consistency in 3D



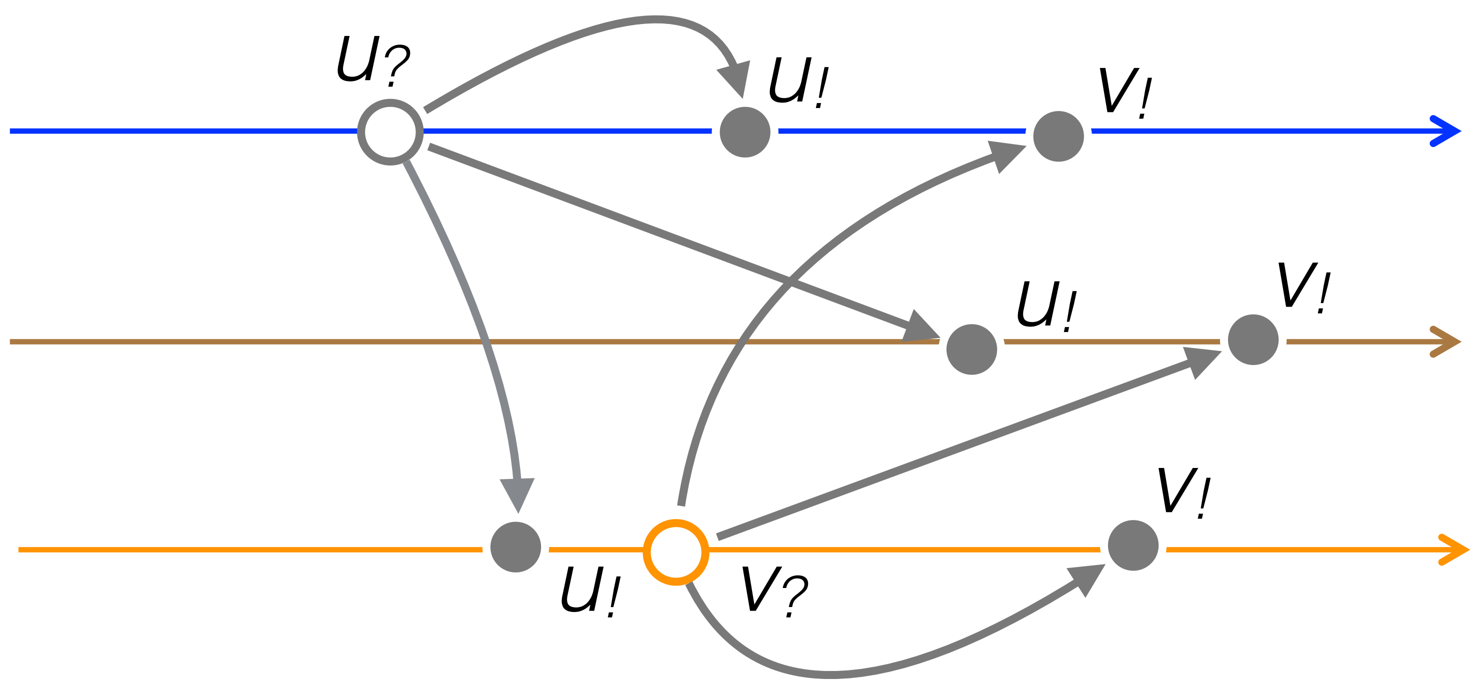
$$\{ \ x \in \mathsf{friendsOf}(\mathsf{y}) \iff y \in \mathsf{friendsOf}(x) \ \}$$

Three dimensions



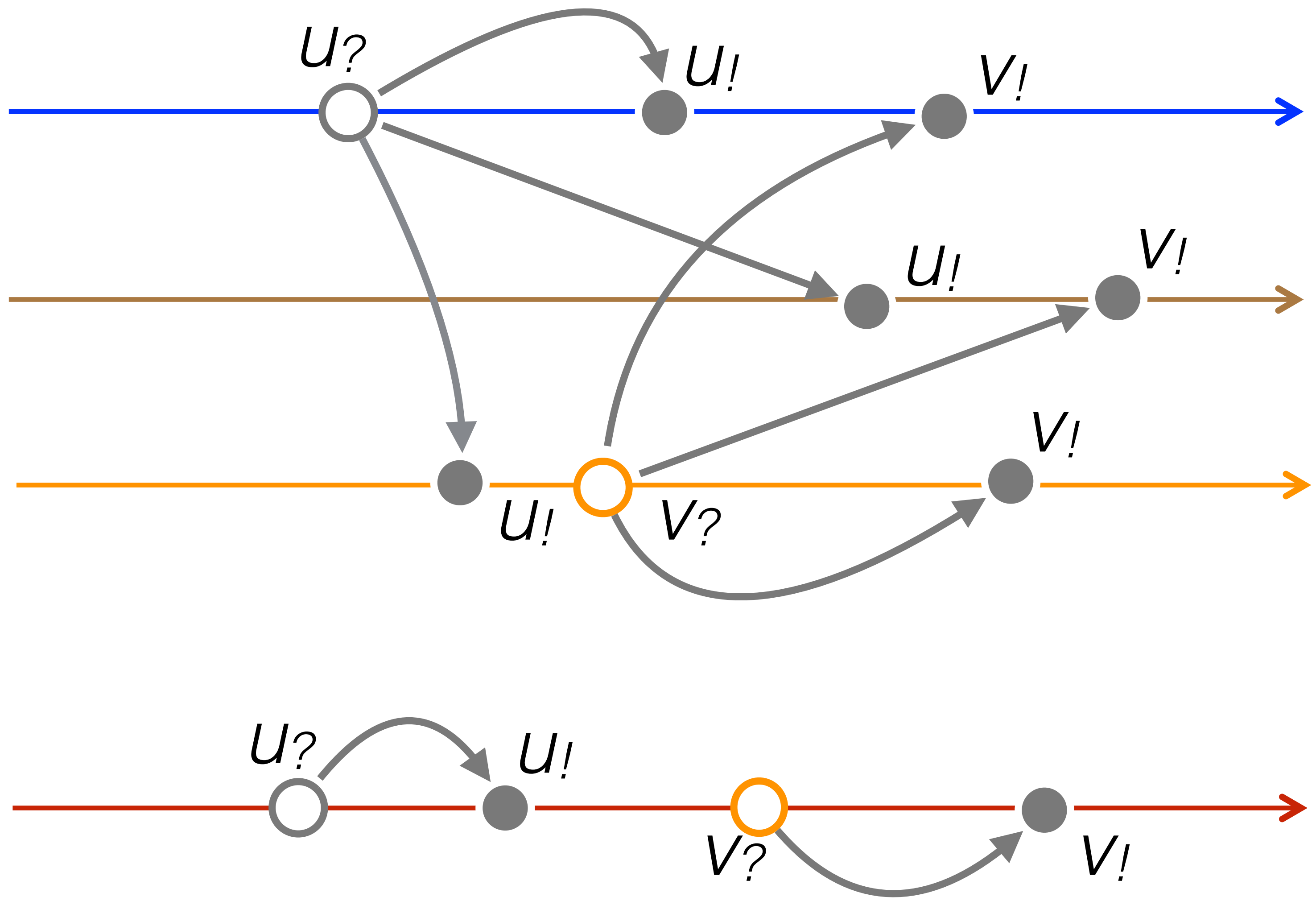
Total Order Axis

- Assumption: Single Object
- Total Order of Effectors and Generators (TOE=TOG)



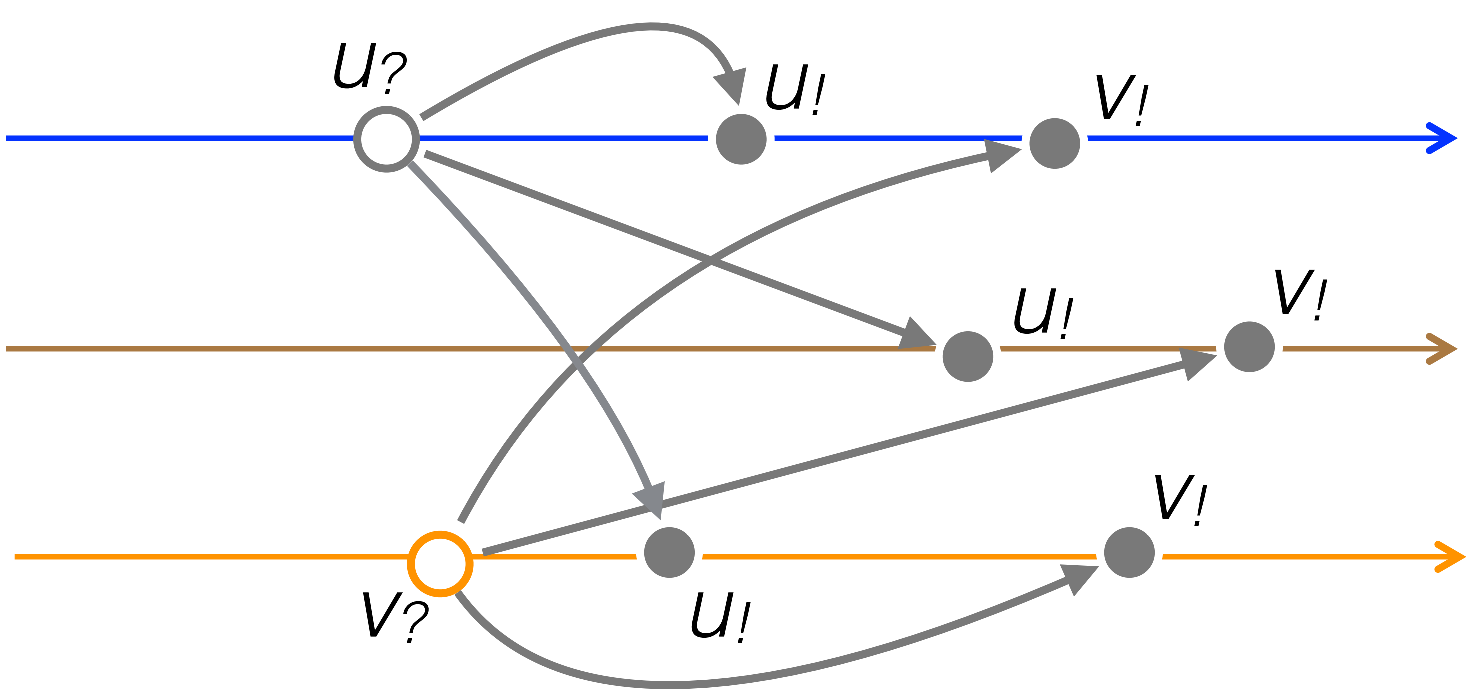
Total Order Axis

- Assumption: Single Object
- Total Order of Effectors and Generators (TOE=TOG)



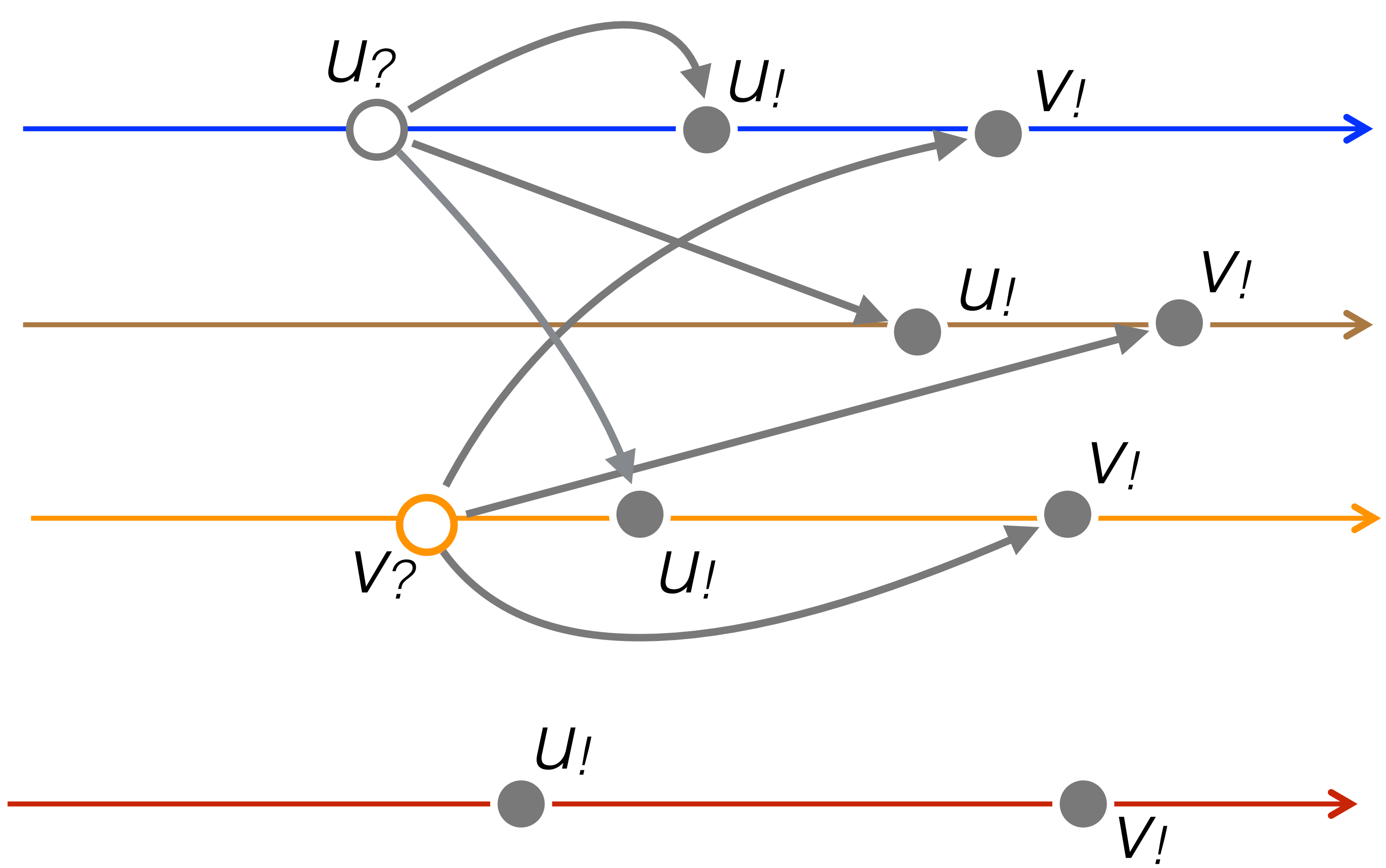
Total Order Axis

- Assumption: Single Object
- Total Order of Effectors ~~and Generators~~ (TOE_1)



Total Order Axis

- Assumption: Single Object
- Total Order of Effectors ~~and Generators~~ (TOE_1)



Total Order Axis

- Assumption: Single Object
- Total Order of Effectors ~~and Generators~~ (TOE_1)

Total Order Axis

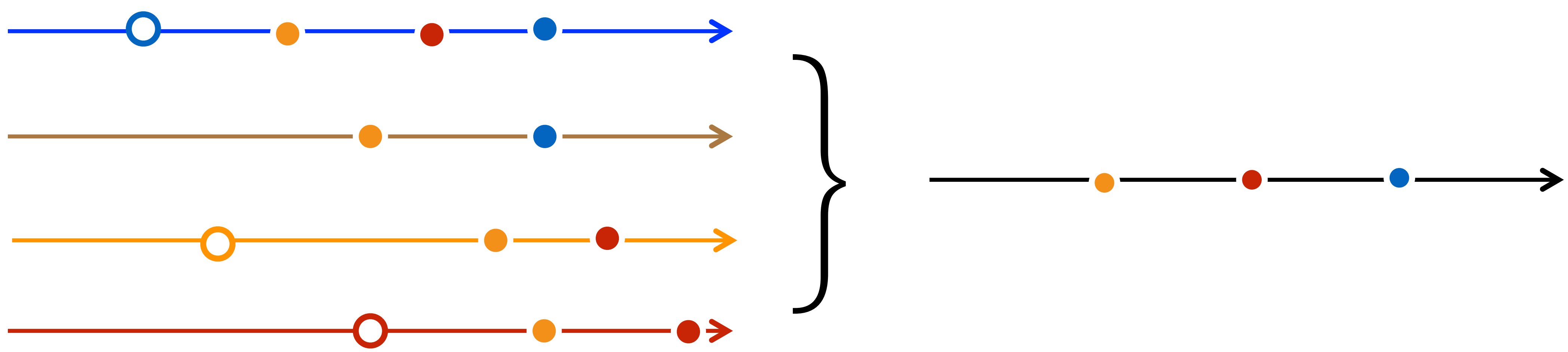
- Assumption: Single Object
- Total Order of Effectors ~~and Generators~~ (TOE_1)
  - Gapless TOE_1: all replicas apply all effectors in the same order

Total Order Axis

- Assumption: Single Object
- Total Order of Effectors ~~and Generators~~ (TOE_1)
  - Gapless TOE_1: all replicas apply all effectors in the same order
  - Capricious TOE_1: replicas apply a subset of the effectors in an order consistent with a global total order

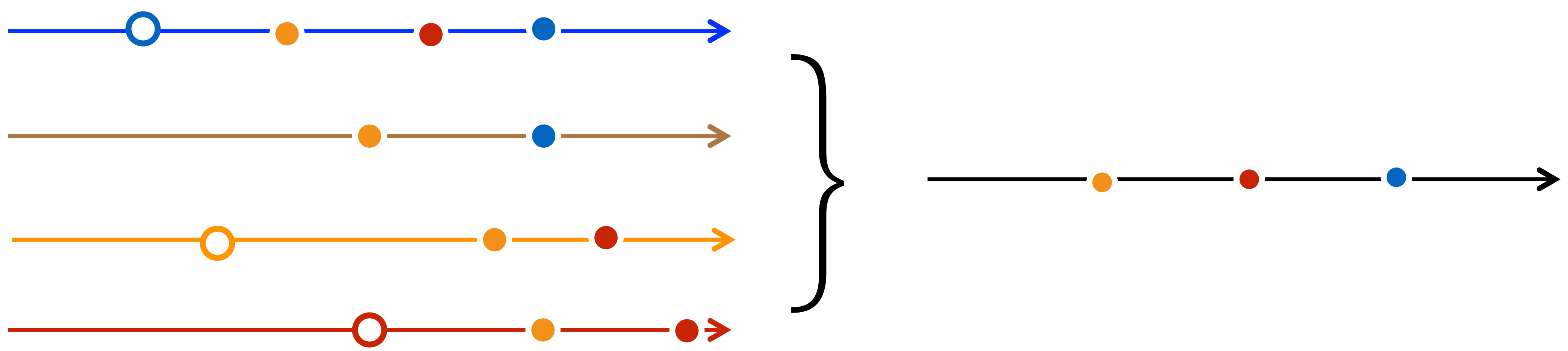
Total Order Axis

- Assumption: Single Object
- Total Order of Effectors ~~and Generators~~ (TOE_1)
  - Gapless TOE_1: all replicas apply all effectors in the same order
  - Capricious TOE_1: replicas apply a subset of the effectors in an order consistent with a global total order

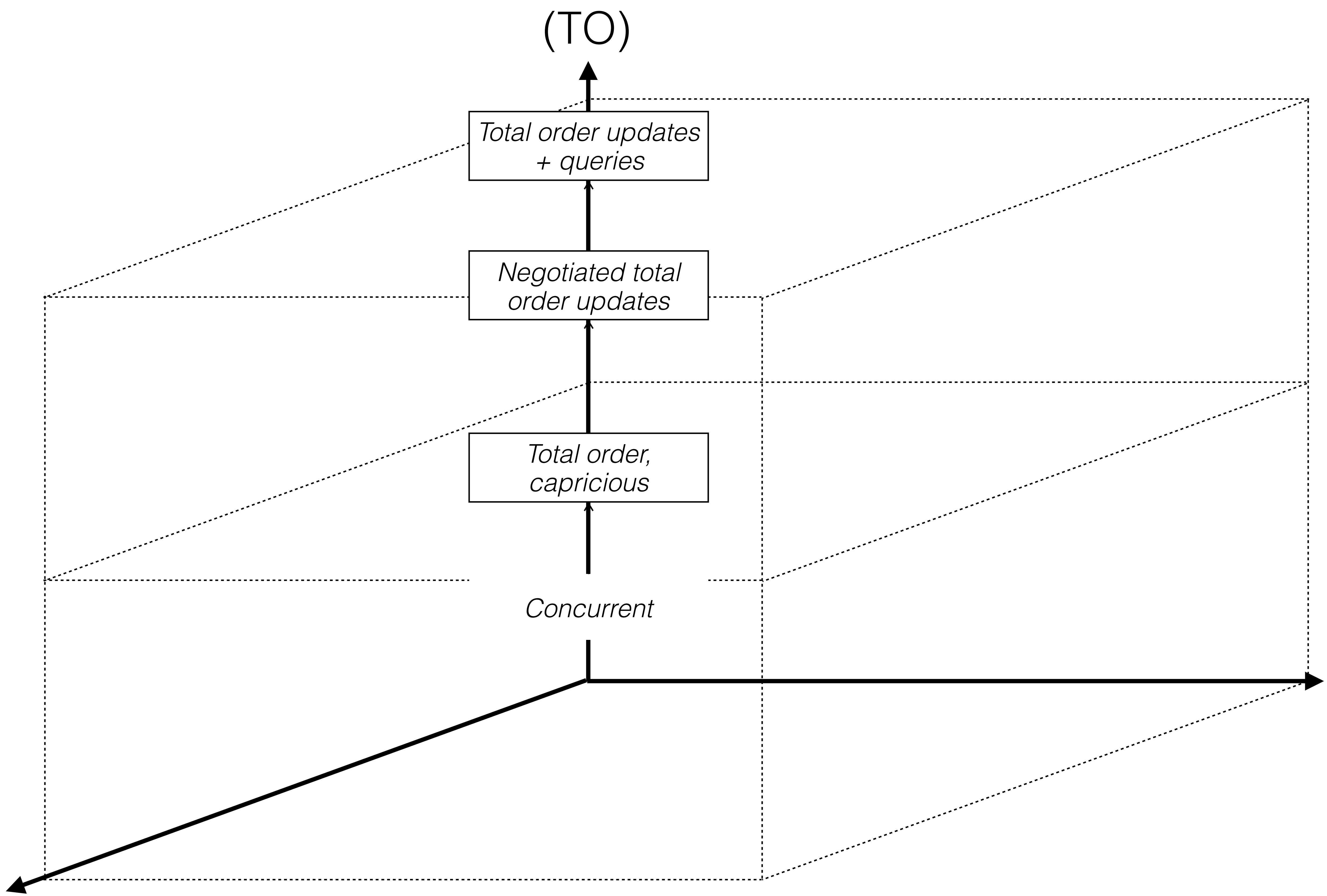


Total Order Axis

- Assumption: Single Object
- Total Order of Effectors ~~and Generators~~ (TOE_1)
    - Gapless TOE_1: all replicas apply all effectors in the same order
    - Capricious TOE_1: replicas apply a subset of the effectors in an order consistent with a global total order



- Concurrent Updates (No Global Ordering)



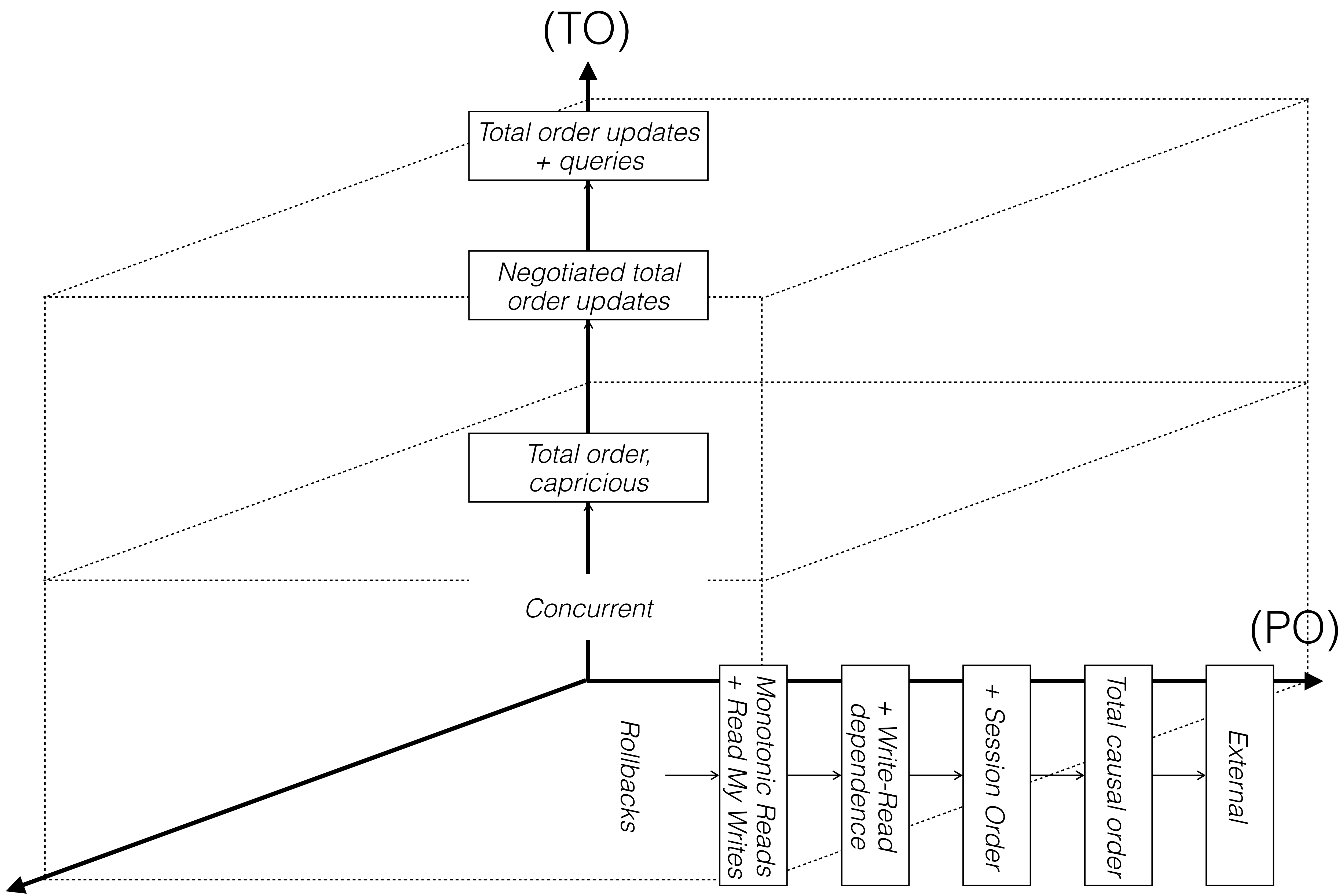
(TO)
Total order updates + queries
Negotiated total order updates
Total order, capricious
Concurrent

Partial Order Axis

- Assumption: Multiple (2) Objects

- Client Guarantees:
  - Read Own Writes
  - Monotonicity (Reads/Writes)
  - Preservation of (anti)Dependencies

- Visibility Properties:
  - Transitive Visibility
  - Causal Visibility



(TO)
Total order updates + queries
Negotiated total order updates
Total order, capricious
Concurrent
(PO)
Rollbacks
Monotonic Reads + Read My Writes
+ Write-Read dependence
+ Session Order
Total causal order
External

Partial Order Axis (Invariants)

- Assumptions:
  - (i) Multiple Object,
  - (ii) State Based,
  - (iii) O is a valid object for I

- Invariants Relating Objects
  - $x \leq y$
  - $P(x) \implies Q(y)$
- Programming:
  - Demarcation Protocol
  - Escrow

Equality Order Axis

- Assumption: Multiple (n) Objects
- Transactions

Equality Order Axis

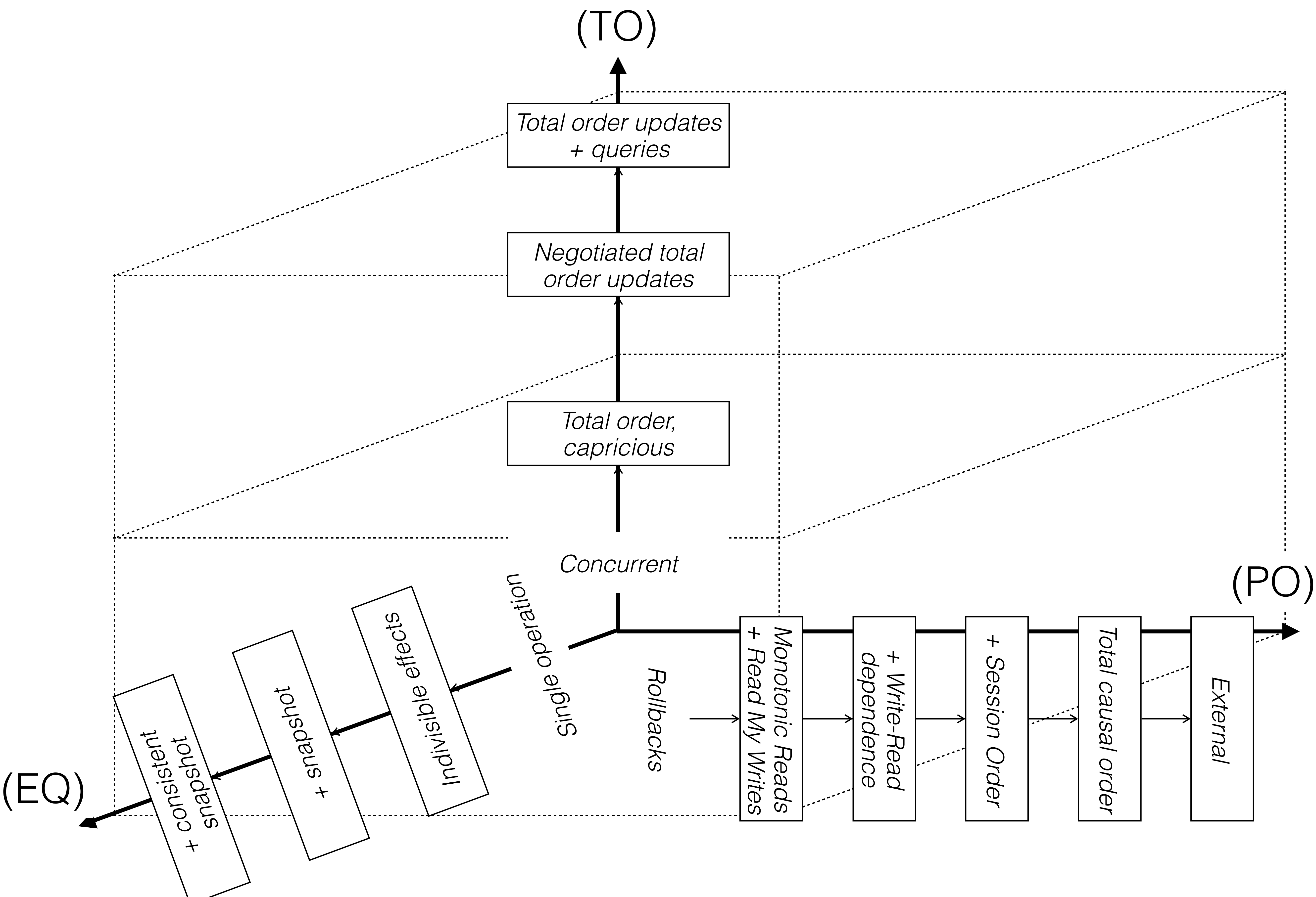
- Assumption: Multiple (n) Objects
- Transactions
  - Write-atomicity: All-or-nothing

Equality Order Axis

- Assumption: Multiple (n) Objects
- Transactions
    - Write-atomicity: All-or-nothing
    - Read-atomicity: Snapshot

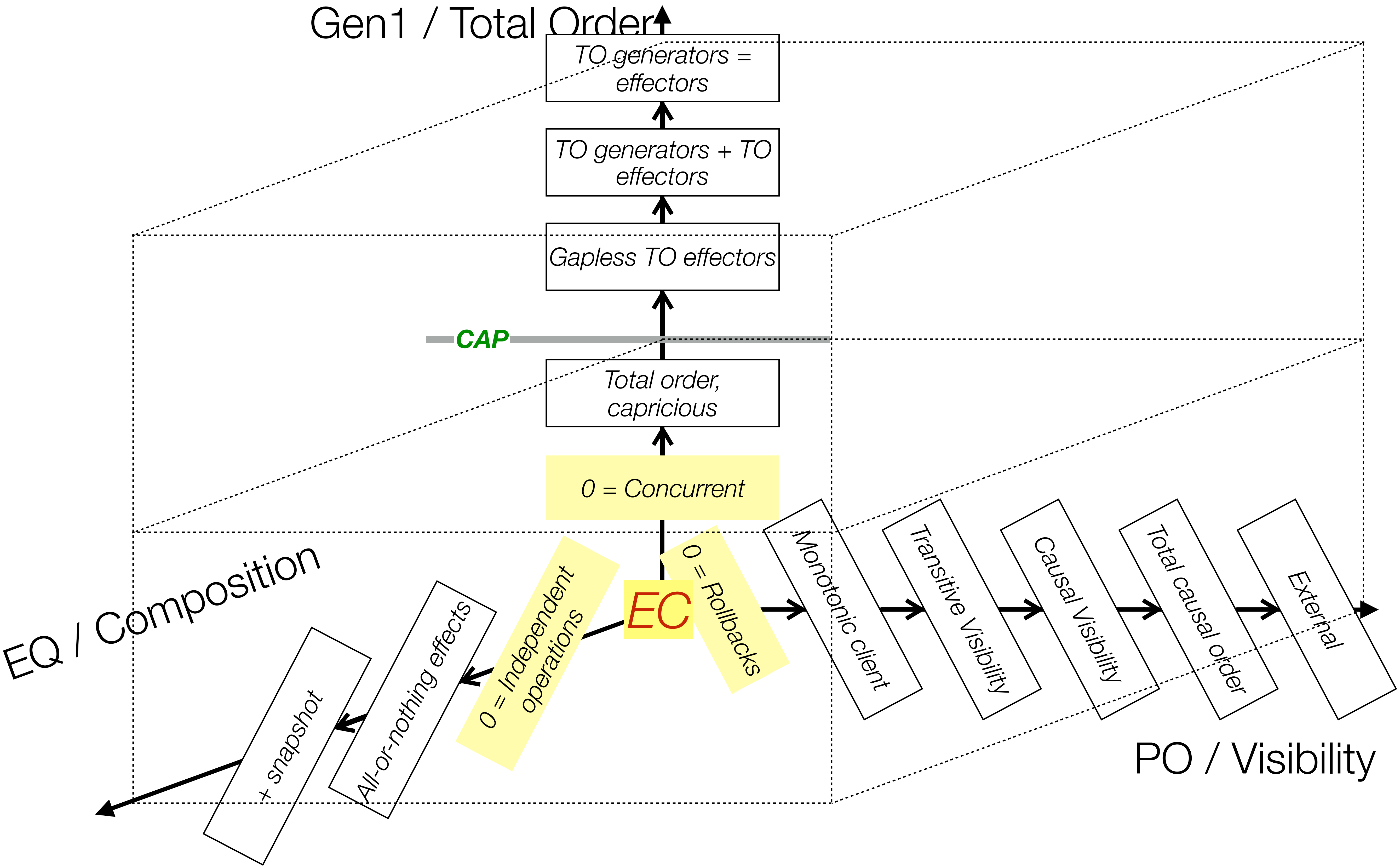
Equality Order Axis

- Assumption: Multiple (n) Objects
- Transactions
  - Write-atomicity: All-or-nothing
  - Read-atomicity: Snapshot
  - Consistent Snapshot

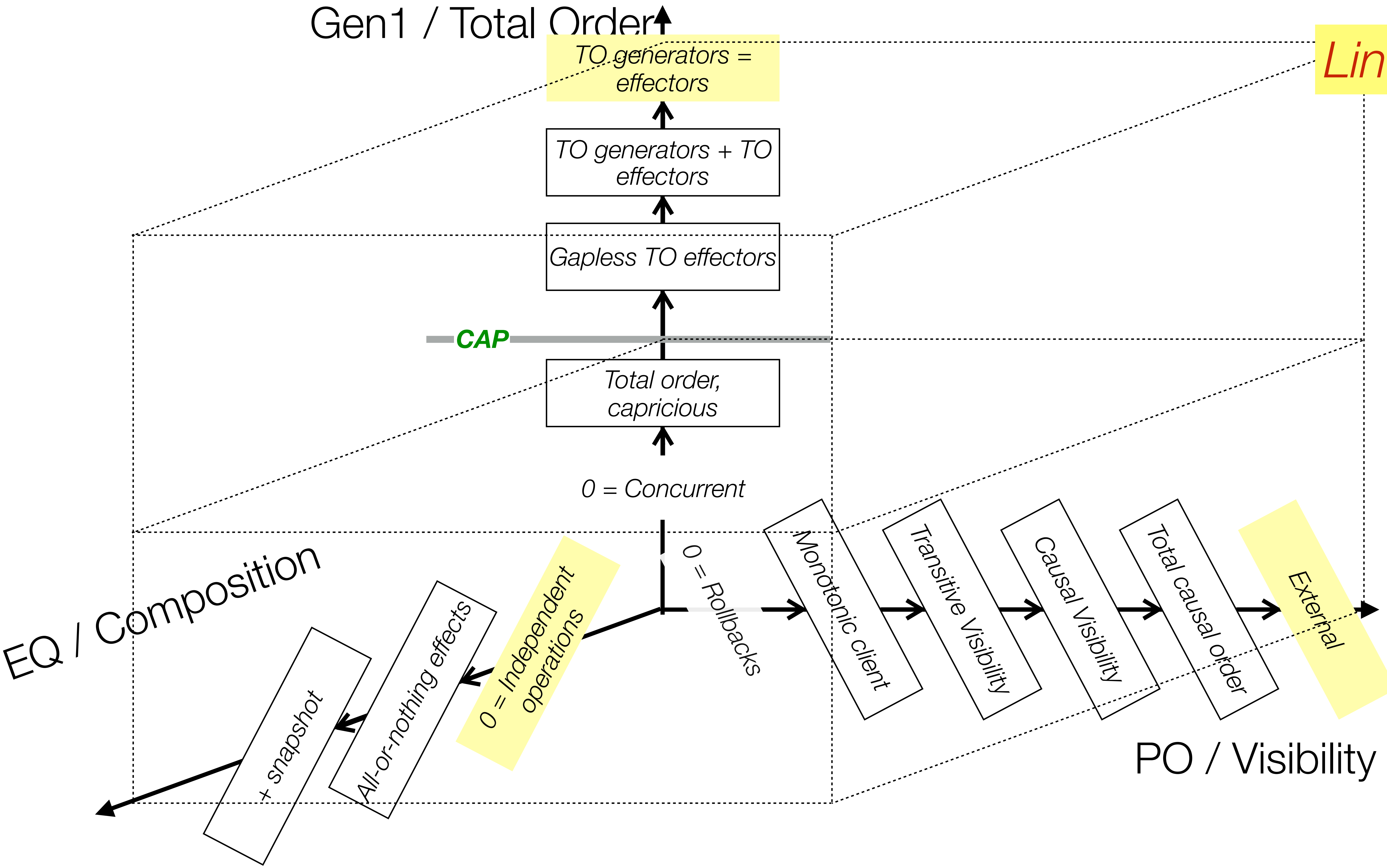


(TO)
Total order updates + queries
Negotiated total order updates
Total order, capricious
Concurrent
(PO)
Single operation
Indivisible effects
+ snapshot
+ consistent snapshot
(EQ)
Rollbacks
Monotonic Reads + Read My Writes
+ Write-Read dependence
+ Session Order
Total causal order
External

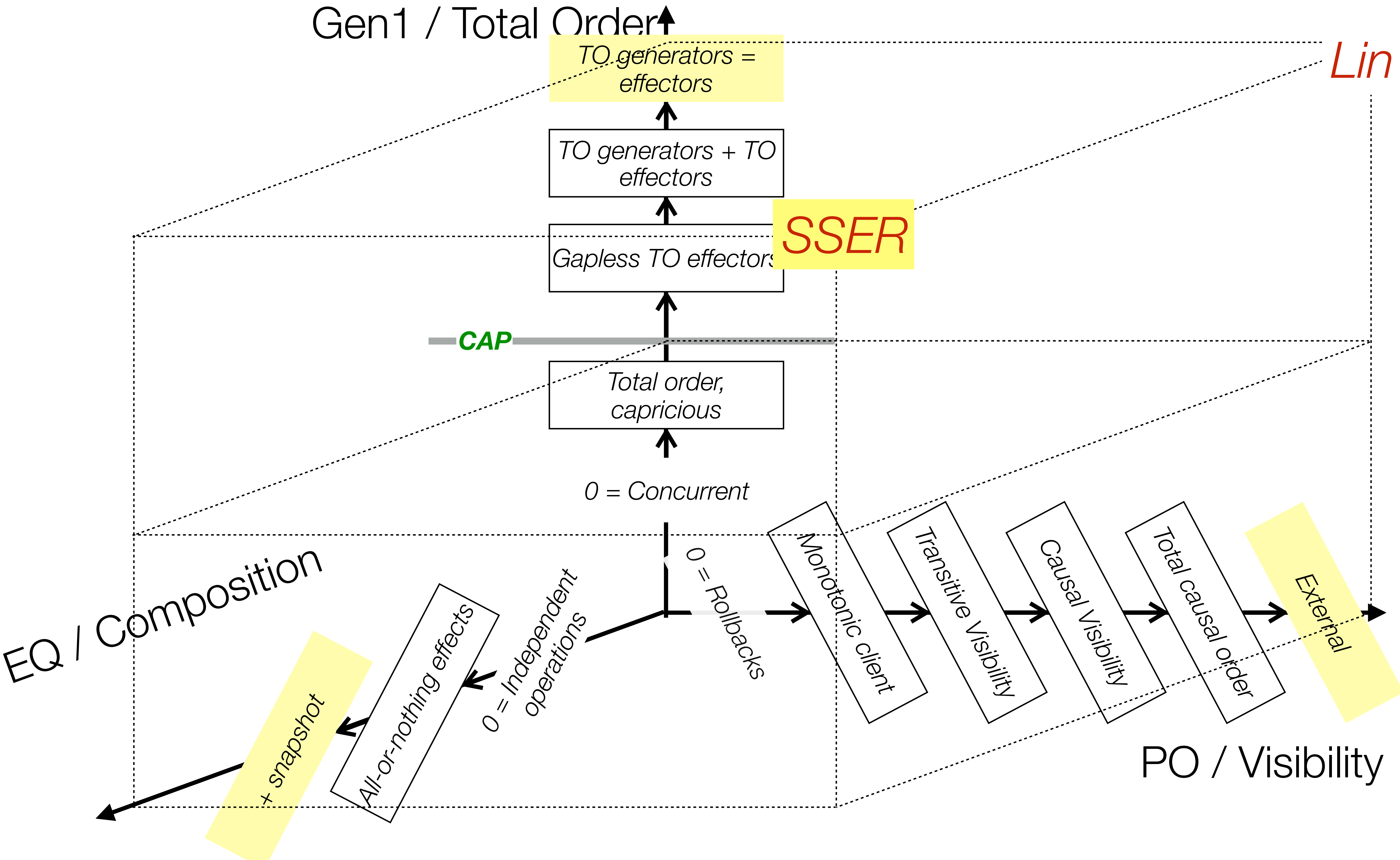
Three dimensions



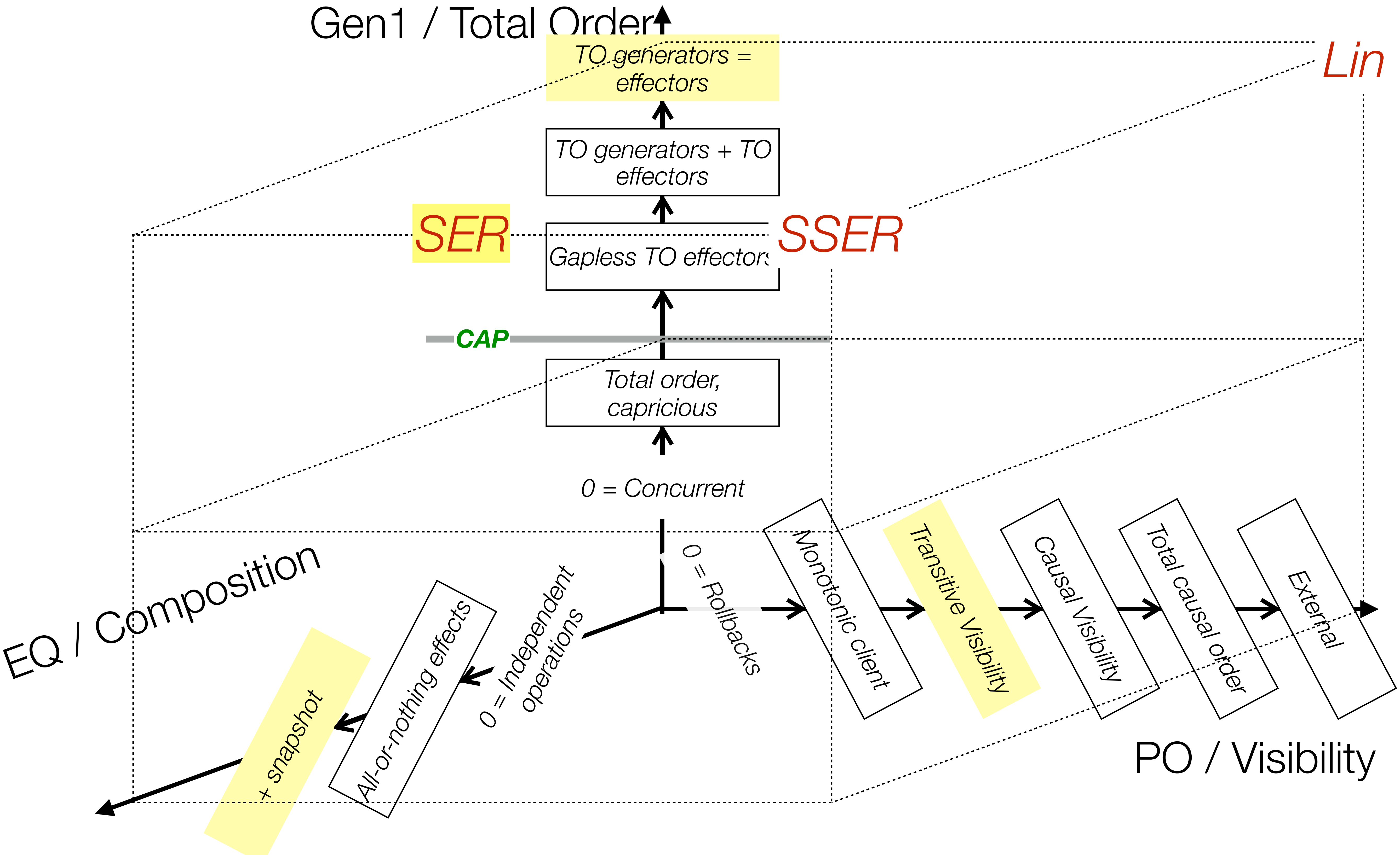
Three dimensions



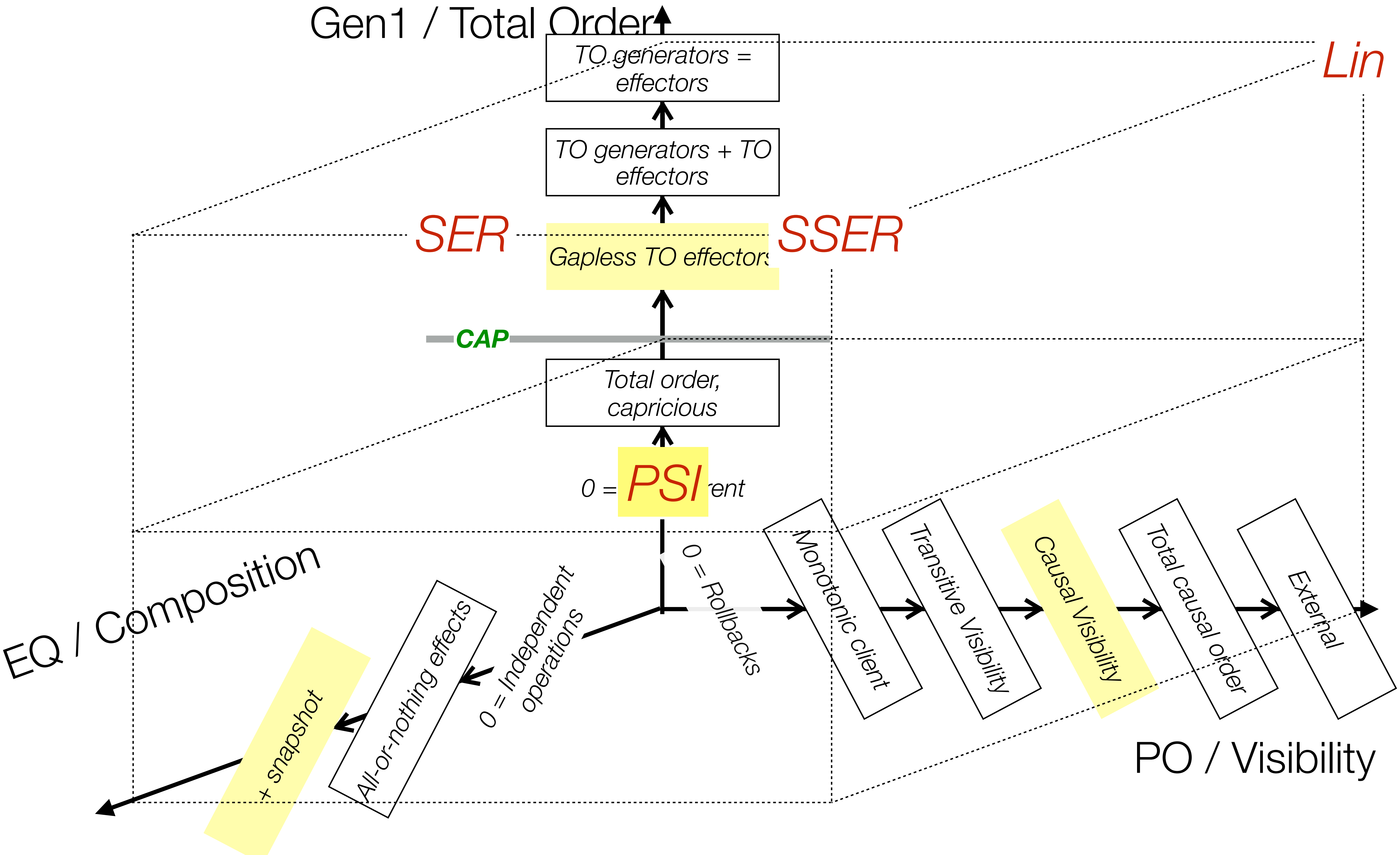
Three dimensions



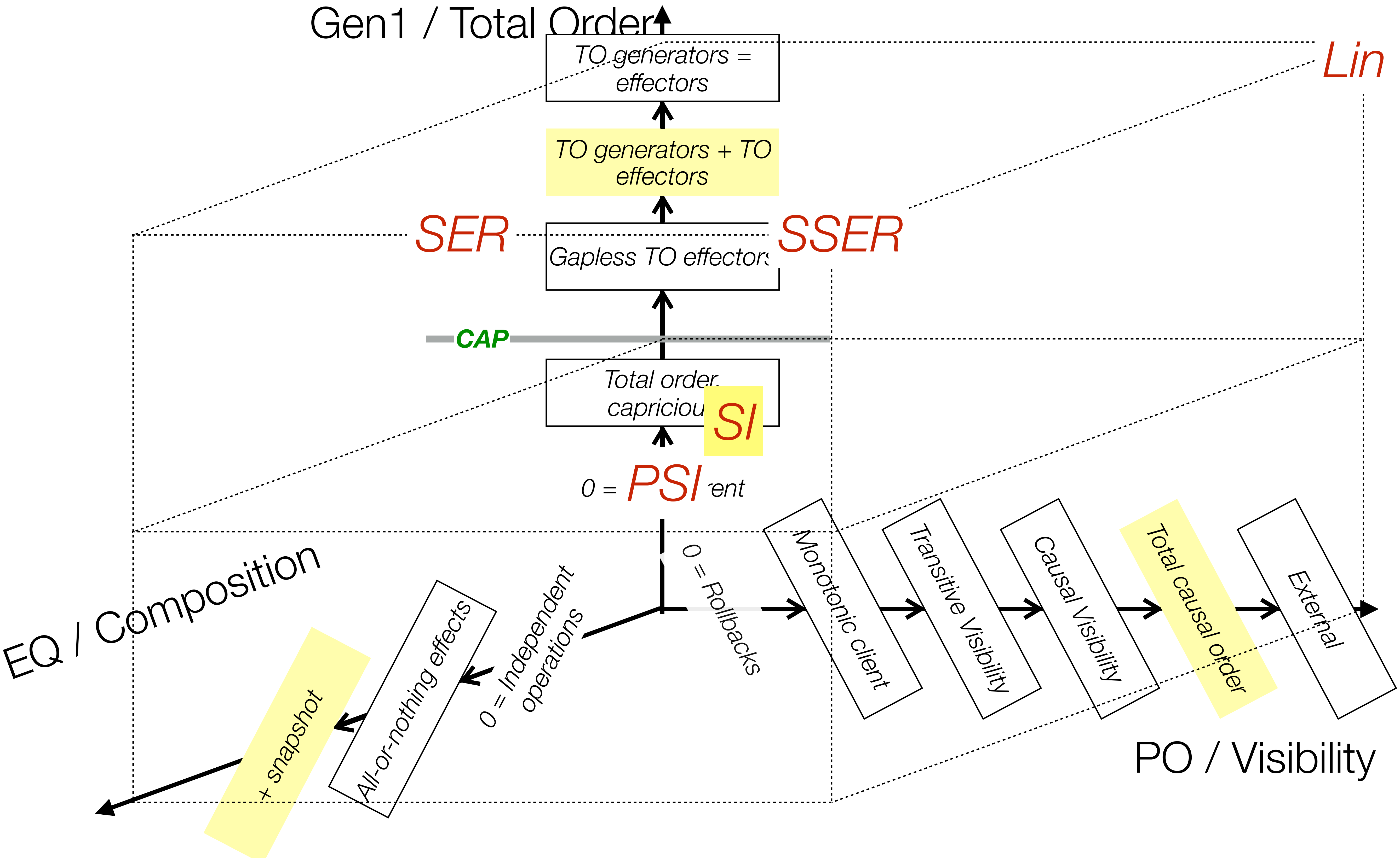
Three dimensions



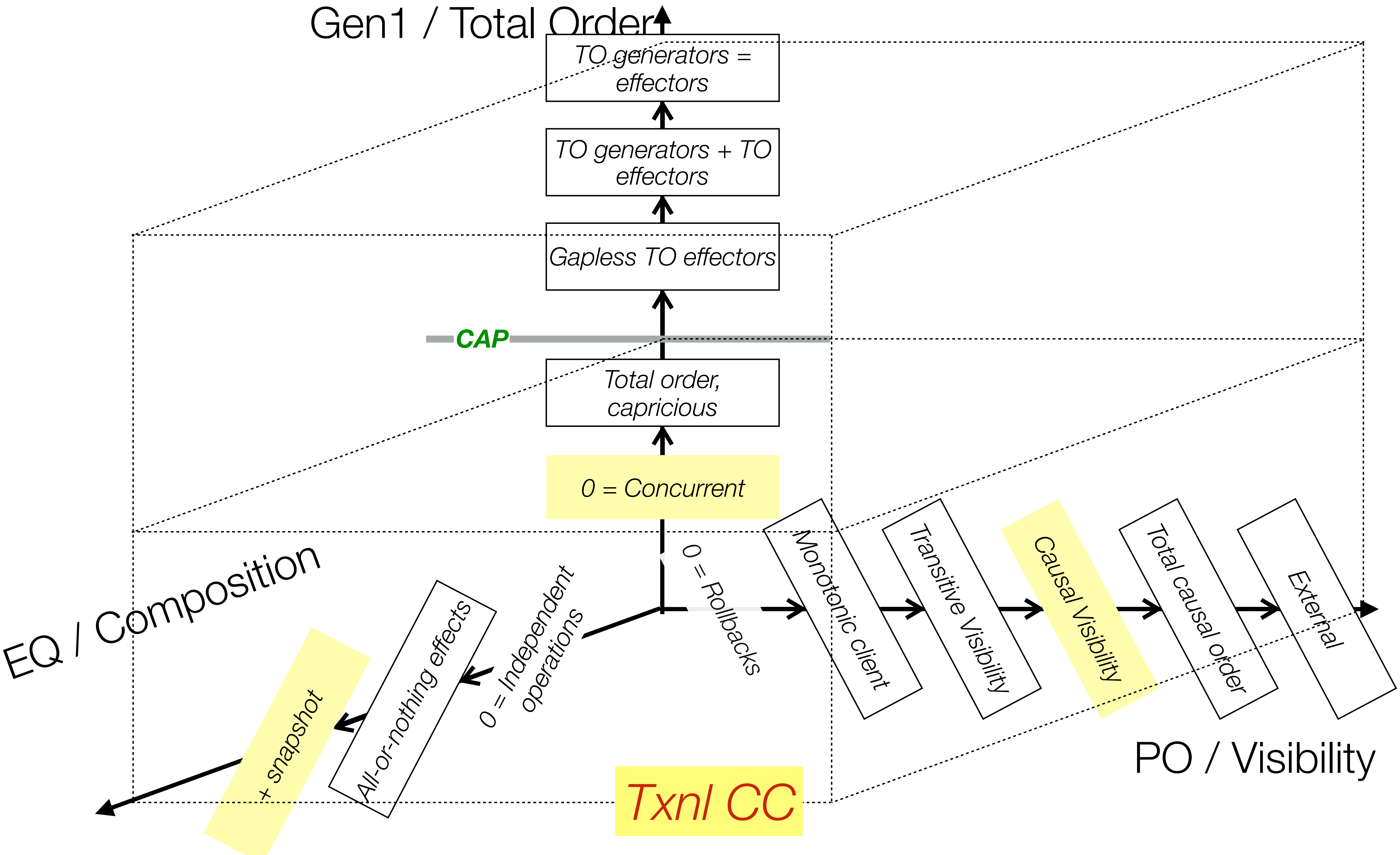
Three dimensions



Three dimensions



Three dimensions



End of day 4