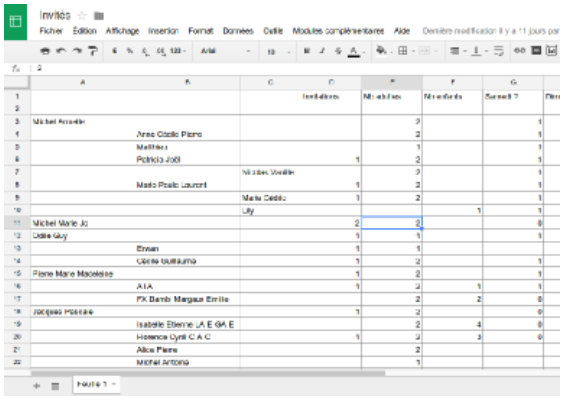
Consistent Shared Data Types: Beyond Memory

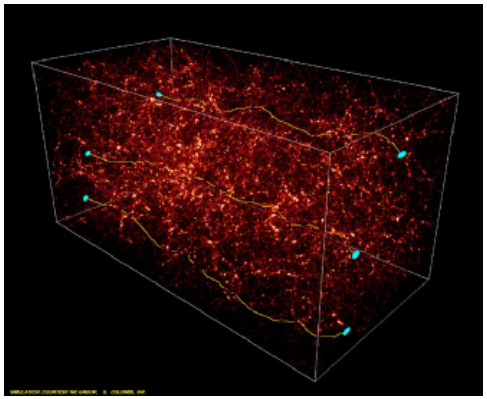
Matthieu Perrin Achour Mostefaoui Claude Jard

AeLoS / GDD
LINA, University of Nantes

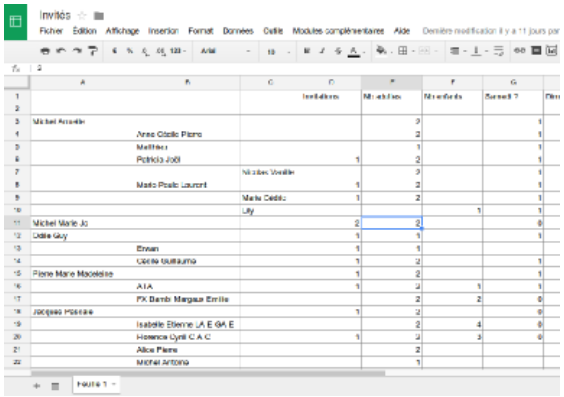
October, 2nd 2014

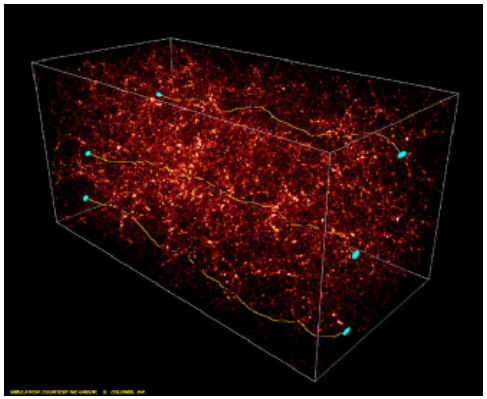
Shared objects

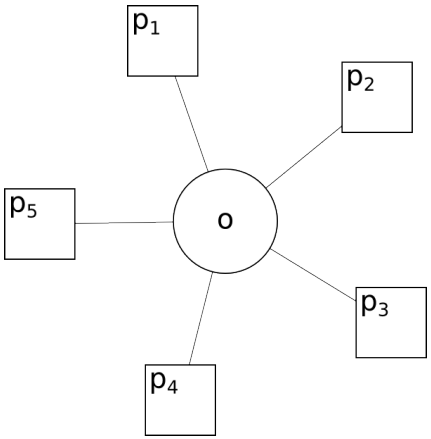




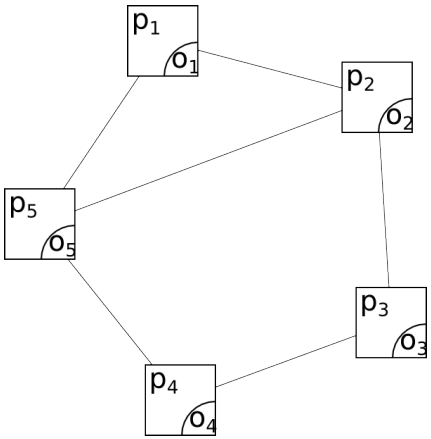
Shared objects







Centralized system



Decentralized system

The CAP theorem

Strong Consistency

Sequential consistency:

- total order on the events
- respect of the process order

Availability

- all the calls eventually return

Partition tolerance

- processes from different partitions cannot communicate

The CAP theorem

Strong Consistency

Sequential consistency:

- total order on the events
- respect of the process order

Availability

- all the calls eventually return

Partition tolerance

- processes from different partitions cannot communicate

Impossible [Gilbert, Lynch, 2002]

State of the art

- Linearizability
- Many kinds of memories
  - safe/regular/atomic register
  - sequentially consistent memory
  - causal memory
  - PRAM
  - ...
- Eventual consistency
- CRDTs
- ...

Outline

Update-Query Abstract Data Type

$T = (U, Q_i, Q_o, S, s_0, \mu, \varphi)$:

U: updates

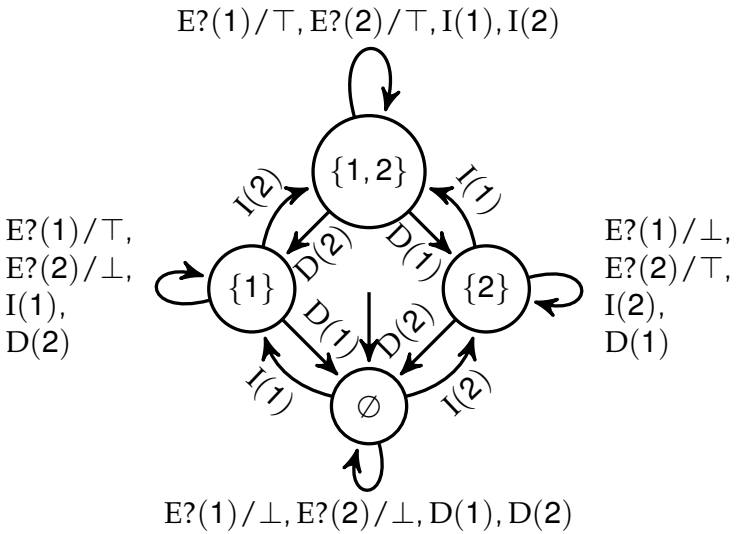
$Q = Q_i \times Q_o$: queries

S: abstract states

$s_0 \in S$: initial state

$\mu : S \times U \to S$: transition function

$\varphi : S \times Q_i \to Q_o$: output function



Language $L(T)$

The words labelling a path starting from s_0

Distributed histories

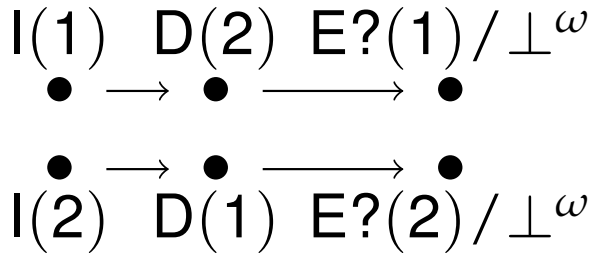
$H = (U, Q, E, \Lambda, \mapsto)$:

U, Q: update and query operations

E: events

$\Lambda : E \to U \cup Q$: labelling function

$\mapsto \in E \times E$: program order
$\forall e \in E, \{e' \in E : e' \mapsto e\}$ is finite



Consistency criteria

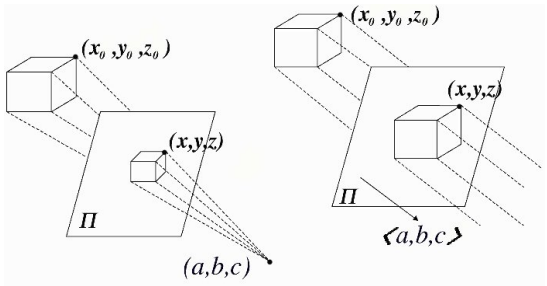
Sequential specification

$$L : \left\{ \begin{array}{ccc} \text{UQ-ADT} & \rightarrow & \mathcal{P}((U \cup Q)^\star) \\ T & \mapsto & L(T) \end{array} \right.$$

Consistency criteria

Sequential specification

$$L : \begin{cases} \text{UQ-ADT} & \to & \mathcal{P}((U \cup Q)^\star) \\ T & \mapsto & L(T) \end{cases}$$



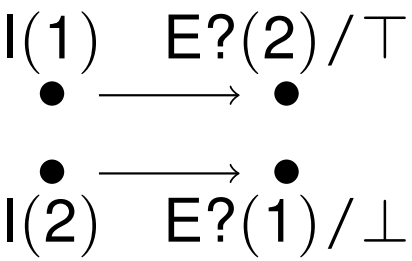
Consistency criterion

$$C : \begin{cases} \text{UQ-ADT} & \to & \mathcal{P}(\mathbb{H}(U, Q)) \\ T & \mapsto & C(T) \end{cases}$$

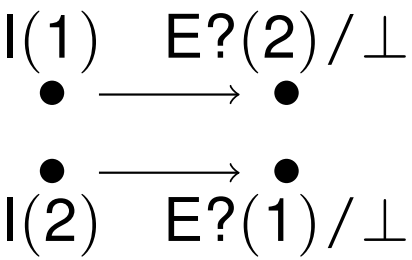
Sequential Consistency

Sequential consistency

$$SC : T \mapsto \{H \in \mathbb{H} : \mathrm{lin}(H) \cap L(T) \neq \emptyset\}$$



$$\mathrm{I}(2) \cdot \mathrm{E?}(1)/\bot \cdot \mathrm{I}(1) \cdot \mathrm{E?}(2)/\top \in \mathrm{lin}(H) \cap L(T)$$

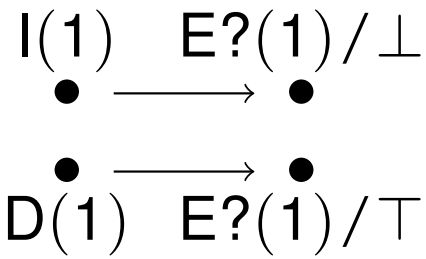


$$\mathrm{lin}(H) \cap L(T) = \emptyset$$

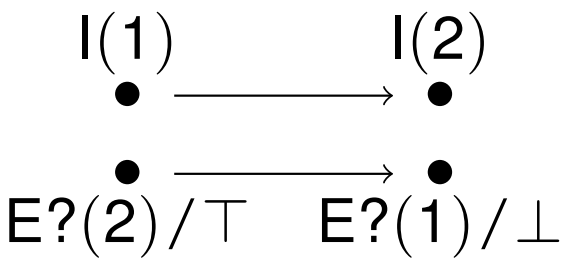
Pipelined Consistency

Pipelined consistency

$$PC : T \mapsto \{H \in \mathbb{H} : \forall p, \mathrm{lin}(H_{U_H \cup p}) \cap L(T) \neq \emptyset\}$$



$$\mathrm{I}(1) \cdot \mathrm{D}(1) \cdot \mathrm{E?}(1)/\bot \in \mathrm{lin}(H_{U_H \cup p_1}) \cap L(T)$$
$$\mathrm{D}(1) \cdot \mathrm{I}(1) \cdot \mathrm{E?}(1)/\top \in \mathrm{lin}(H_{U_H \cup p_2}) \cap L(T)$$



$$\mathrm{I}(1) \cdot \mathrm{I}(2) \in \mathrm{lin}(H_{U_H \cup p_1}) \cap L(T)$$
$$\mathrm{lin}(H_{U_H \cup p_2}) \cap L(T) = \emptyset$$

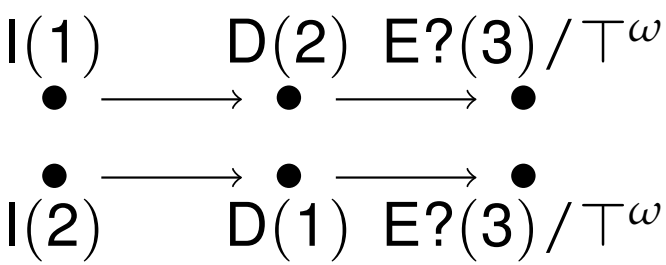
Eventual Consistency

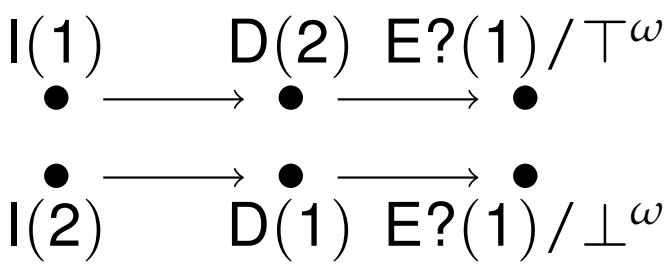
Eventual consistency

$$EC : T \mapsto \{H \in \mathbb{H} : Write4Ever \vee Converge\}$$

$$\text{Write4Ever} \equiv |H_U| = \infty$$

$$\text{Converge} \equiv \exists s \in S, |\{q_i/q_o \in H_Q : \varphi(s, q_i) \neq q_o\}| < \infty$$





- Independant of $L(T)$

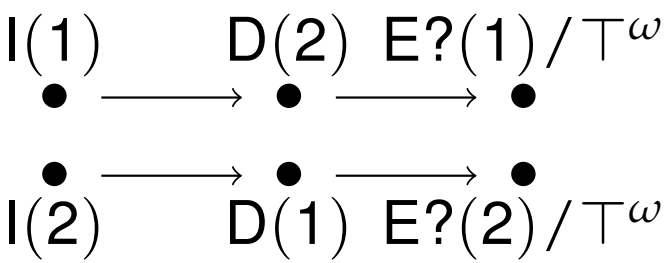
Update Consistency

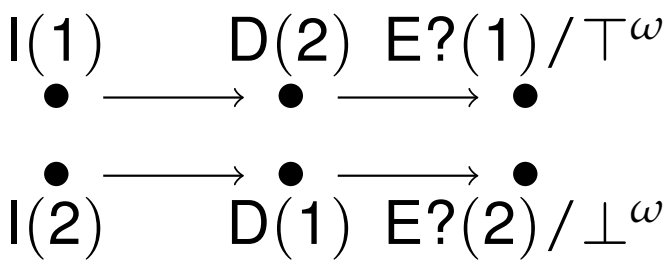
Update consistency

$$UC : T \mapsto \{H \in \mathbb{H} : \mathit{Write4Ever} \vee \mathit{Converge}\}$$

$$\text{Write4Ever} \equiv |H_U| = \infty$$

$$\text{Converge} \equiv \exists Q' \subset H_Q \text{ finite: } \mathrm{lin}(H_{E_H \setminus Q'}) \cap L(T) \neq \emptyset$$





- All UQ-ATDs can be implemented in a partitionable system

Lattice of consistency criteria

$C_1 \leq C_2$

$$C_1 \leq C_2 \Leftrightarrow \forall T, C_2(T) \subset C_1(T)$$

- $PC \leq SC$, $EC \leq SC$
- correct with $C_1 \Rightarrow$ correct with C_2
- $\forall C, (C_\perp : T \mapsto \mathbb{H}) \leq C \leq (C_\top : T \mapsto \emptyset)$

Lattice of consistency criteria

$C_1 \leq C_2$

$$C_1 \leq C_2 \Leftrightarrow \forall T, C_2(T) \subset C_1(T)$$

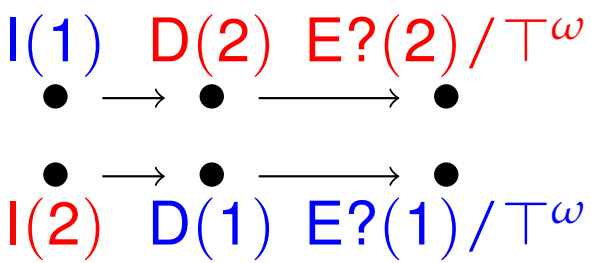
- $PC \leq SC$, $EC \leq SC$
- correct with $C_1 \Rightarrow$ correct with C_2
- $\forall C, (C_\perp : T \mapsto \mathbb{H}) \leq C \leq (C_\top : T \mapsto \emptyset)$

$C_1 + C_2$

$$C_1 + C_2 = T \mapsto C_1(T) \cap C_2(T)$$

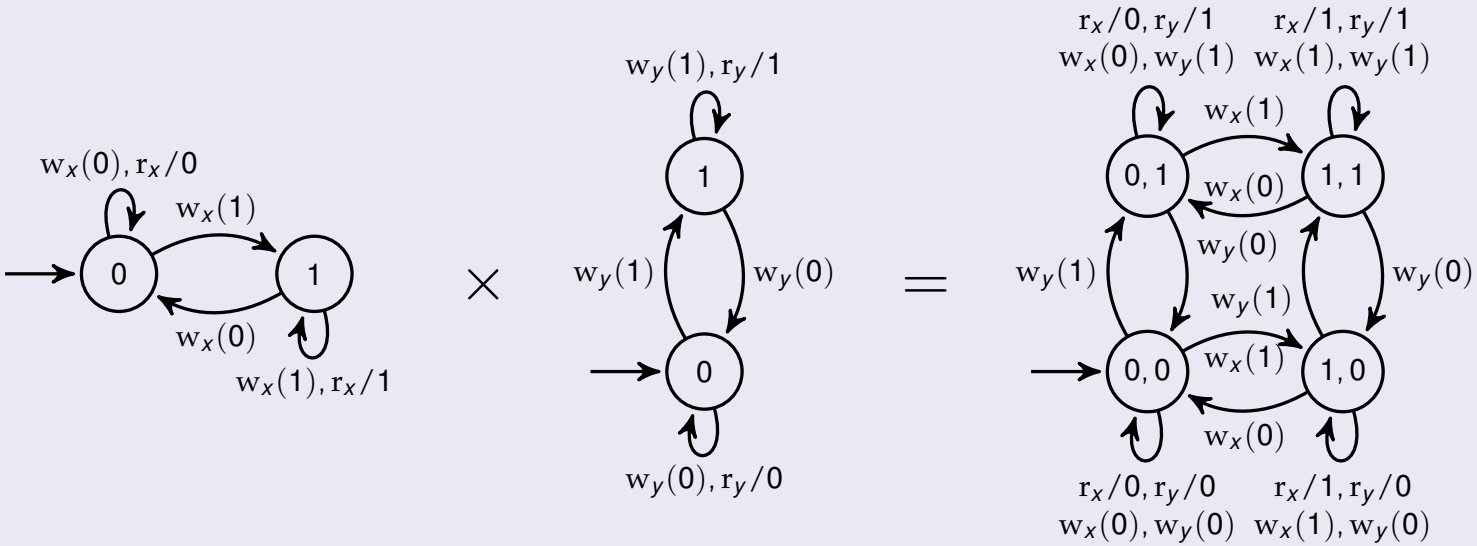
- $(\mathscr{C}, \leq, +)$ is a join-semilattice
- $\forall C, C_\perp + C = C \wedge C_\top + C = C_\top$
- $EC + PC \leq SC$

Composition



Composition

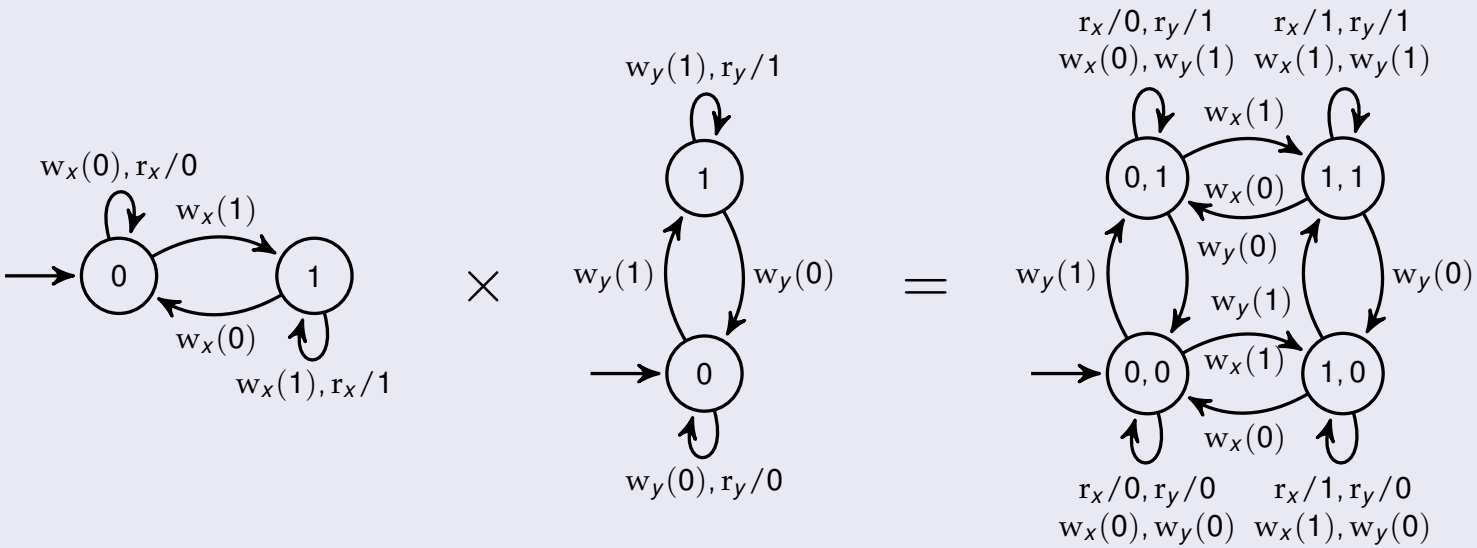
Composition of types



- $L(T_1 \times T_2) = \mathit{interleavings}(L(T_1), L(T_2))$

Composition

Composition of types



- $L(T_1 \times T_2) = \mathit{interleavings}(L(T_1), L(T_2))$

Composition of histories

- $H \in H_1 \times H_2$ if $E = E_1 \cup E_2$, $H_{E_1} = H_1$ and $H_{E_2} = H_2$

Composition

Composition of consistency criteria

$$\frac{H \in H_1 \times H_2 \qquad H_1 \in C_1(O_1) \qquad H_2 \in C_2(O_2)}{H \in C_1 \times C_2(O_1 \times O_2)}$$

- $C_1 \times C_2 \leq C_1$
- $C_\perp \times C = C_\perp$, $C_\top \times C = C$

Composition

Composition of consistency criteria

$$\frac{H \in H_1 \times H_2 \qquad H_1 \in C_1(O_1) \qquad H_2 \in C_2(O_2)}{H \in C_1 \times C_2(O_1 \times O_2)}$$

- $C_1 \times C_2 \leq C_1$
- $C_\perp \times C = C_\perp$, $C_\top \times C = C$

Composability

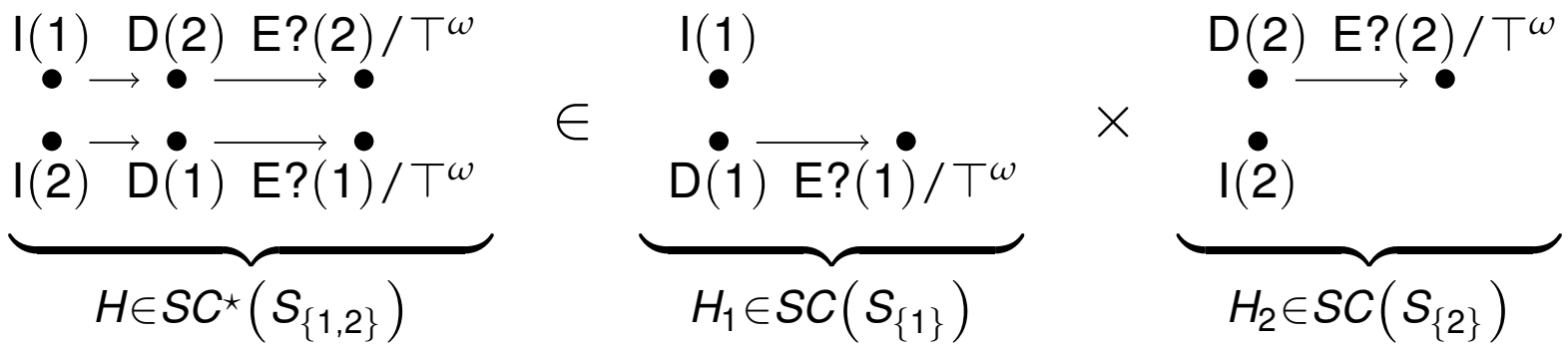
C composable if $C = C^2$

- $EC = EC^2$
- $PC \leq C \leq SC \Rightarrow C \neq C^2$
- $C^\star = \lim_{n\to\infty} C^n = T \mapsto \bigcup_{n\in\mathbb{N}} C^n(T)$

Cache consistency

Cache consistency

$$SC^\star$$



- $H(\mathrm{OR-set}) \subset SC^\star(S_{\mathbb{N}})$
- $(S)EC \leq SC^\star$
- $UC(\mathrm{Set}) \subset SC^\star(\mathrm{Set})$ but $SC^\star(\mathrm{Stack}) \subset UC(\mathrm{Stack})$

Causal memory

$\rightsquigarrow$ is a writes-into order if:

- $\forall e \rightsquigarrow e'$, $\Lambda(e) = \mathrm{w}_x(n)$ and $\Lambda(e') = \mathrm{r}_x/n$,
- $\forall e$, $|\{e' \in E : e' \rightsquigarrow e\}| \leq 1$,
- for all $e \in E$ such that $\Lambda(e) = \mathrm{r}_x/n$ and there is no $e' \in E$ such that $e' \rightsquigarrow e$, then $n = 0$.

H is M-causal if $\exists \rightsquigarrow$ such that:

- $\rightarrow = (\rightsquigarrow \cup \mapsto)^\star$ is a partial order
- $\forall p \in \mathscr{P}_H$, $\mathrm{lin}\left(H^{\rightarrow}_{U_H \cup p}\right) \cap L(M) \neq \emptyset$

Causal consistency - definition

Causal consistency

- $CC : T \mapsto \{H \in \mathbb{H} : \exists (\rightarrow_p)_{p \in \mathcal{P}_H}, TO \wedge PO \wedge PC \wedge CH\}$

$$TO \equiv \forall p \in \mathcal{P}_H, \rightarrow_p \text{ is a total order}$$
$$PO \equiv \forall p \in \mathcal{P}_H, \mapsto \subset \rightarrow_p$$
$$PC \equiv \forall p \in \mathcal{P}_H, \text{lin}\left(H^{\rightarrow p}_{U_H \cup p}\right) \cap L(T) \neq \emptyset$$
$$CH \equiv \forall p, p' \in \mathcal{P}_H, \forall e \in E, \forall e' \in p, (e \rightarrow_p e') \Rightarrow (e \rightarrow_{p'} e')$$

- Causal order: $\rightarrow = \left(\bigcap_{p \in \mathcal{P}_H} \rightarrow_p\right)$

Causal consistency - properties

Property 1

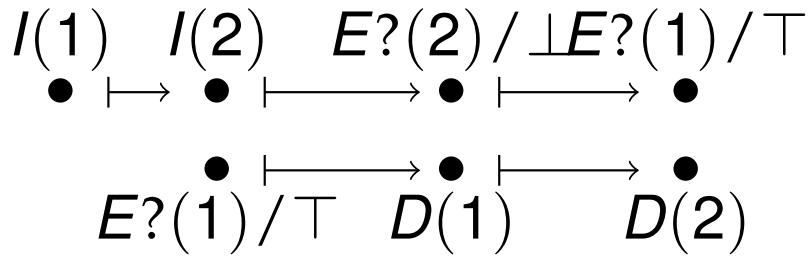
- $PC \leq CC \leq SC$
- $CC^2 \neq CC$

Property 2

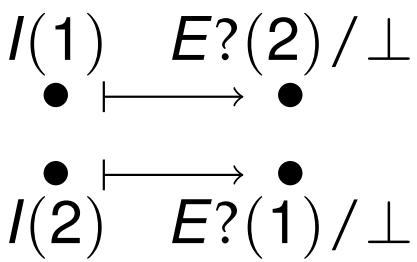
If $H \in CC(T)$, there exists $\rightarrow$ such that:

- program order: $\forall p \in \mathcal{P}_H, \text{lin}\left(H^{\rightarrow}_{U_H \cup p}\right) \cap L(T) \neq \emptyset$
- data order: $\forall q \in Q_H, \text{lin}\left(H^{\rightarrow}_{\{u \in U_H : u \rightarrow q\} \cup \{q\}}\right) \cap L(T) \neq \emptyset$

Causal consistency - examples

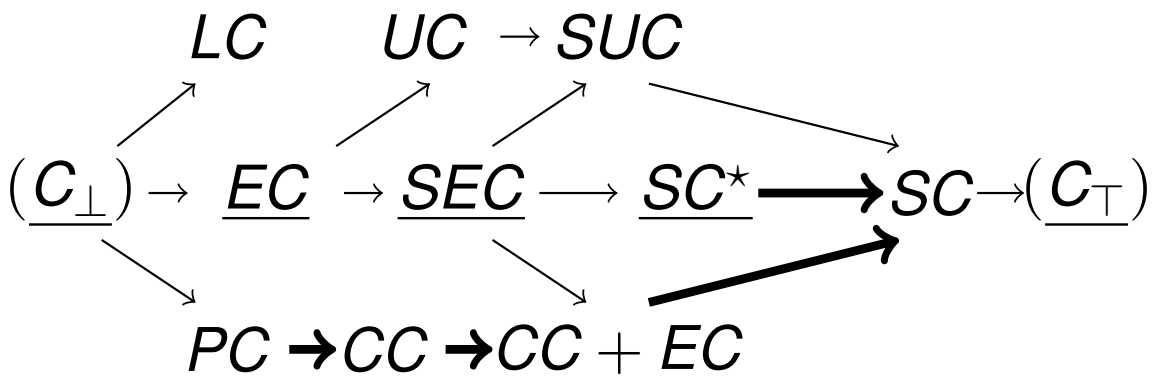


$$D(1) \cdot I(1) \cdot I(2) \cdot D(2) \cdot E?(2)/\bot \cdot E?(1)/\top \in \mathrm{lin}(H_{U_H \cup p_1}) \cap L(T)$$
$$I(1) \cdot E?(1)/\top \cdot D(1) \cdot I(2) \cdot D(2) \in \mathrm{lin}(H_{U_H \cup p_2}) \cap L(T)$$



$$I(1) \cdot E?(2)/\bot \cdot I(2) \in \mathrm{lin}(H_{U_H \cup p_1}) \cap L(T)$$
$$I(2) \cdot E?(1)/\bot \cdot I(1) \in \mathrm{lin}(H_{U_H \cup p_2}) \cap L(T)$$

Sum up



Data integrity

Type graph

- a set V of vertices
- a set E of edges
- insert a vertex or an edge
- delete a vertex or an edge

Integrity constraint (P)

At any time, if $(v, v') \in E$, then $v \in V$ and $v' \in V$.

Data integrity

```
1  class Graph
2  |  var vertices ⊂ ℕ ← ∅;
3  |  var edges ⊂ ℕ × ℕ ← ∅;
4  |  ...
5  |  update Dv (v ∈ ℕ)                    /* delete vertex */
6  |  |  edges ← edges \ (V × {v} ∪ {v} × V);
7  |  |  vertices ← vertices \ {v};
8  |  end
9  |  << P; Dv(v); P >>
10 |  ...
11 end
12 void main ()
13 |  C < Graph > g;
14 |  ...
15 end
```