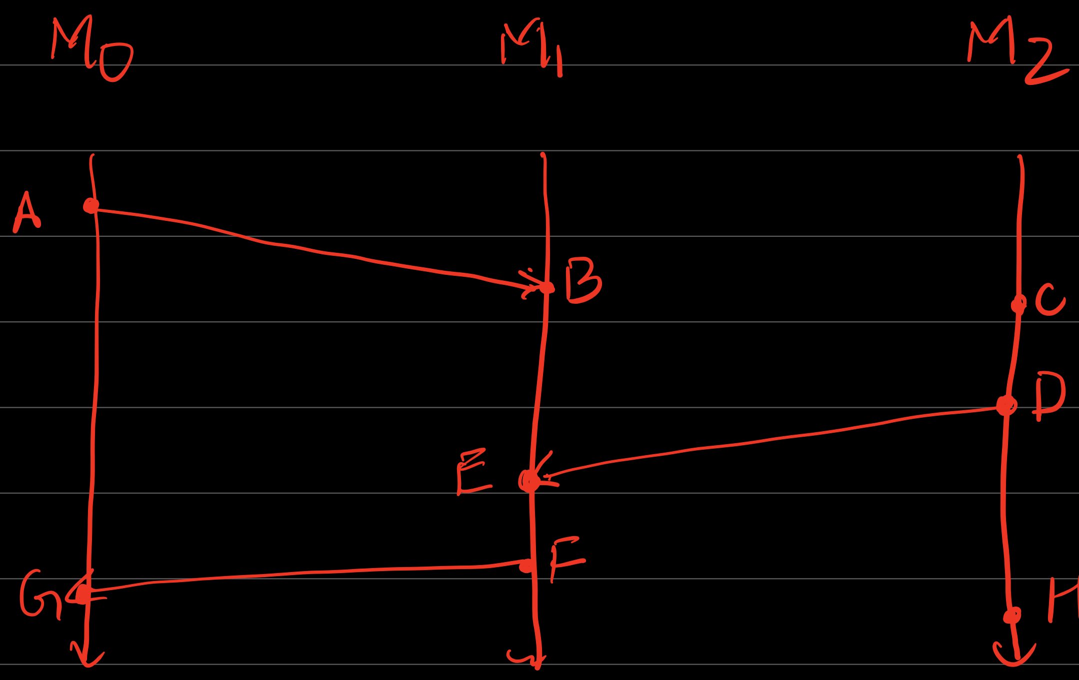
Recap: happens before

- → A happens before B in same process
- → A is send & B is receive message
- → Transitivity: $A \rightarrow C$ & $C \rightarrow B \Rightarrow A \rightarrow B$



$A \rightarrow E$ (Transitivity)

$\left.\begin{array}{l} A \not\rightarrow D \\ D \not\rightarrow A \end{array}\right\}$ $A \parallel D$ (concurrent)

Partial Order

A set S together with a binary relation often written $\leq$, that lets you compare elements of S, & has the following properties.

- Reflexivity: for all $a \in S$, $a \leq a$

- **Antisymmetry**: For all $a, b \in S$
  if $a \le b$ & $b \le a$
  then $a = b$

- **Transitivity**: For all $a, b, c \in S$
  $a \le b$ & $b \le c$
  then $a \le c$

For happens-before relation:

S - set of events in execution

$\rightarrow$ $\Rightarrow$ Transitive : ✓

Antisymmetry: $A \rightarrow B$ & $B \rightarrow A$
Not possible.
This is **vacuously true** ✓

Reflexivity: $A \rightarrow A$
Not true ✗

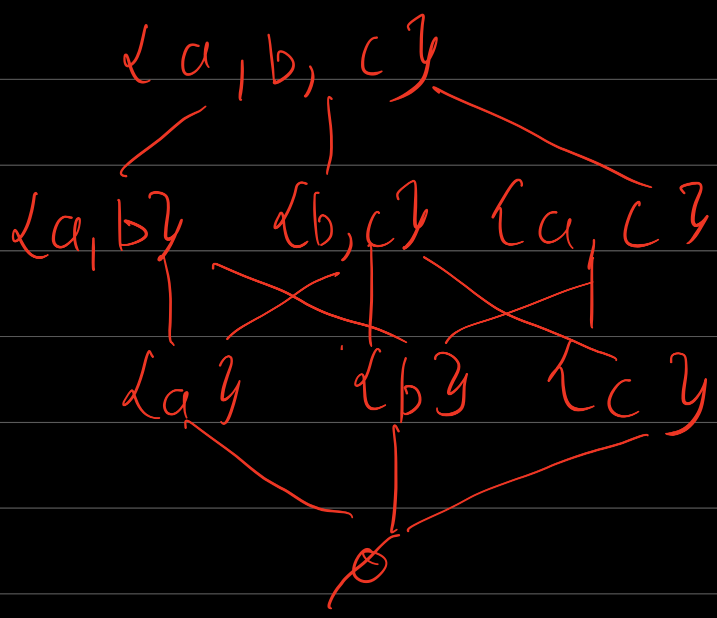
$\therefore$ Happens before $(\rightarrow)$ is an irreflexive partial order

A (real) example of partial order

Set containment $\{a, b, c\}$

$S = \{\{\emptyset\}, \{a\}, \{b\}, \{c\}, \{a, b\}, \{b, c\}, \{a, c\}, \{a, b, c\}\}$

S can be ordered by set inclusion



Partial order : Reflexivity : $\{a\} \leq \{a\}$ ✓

Antisymmetry : $\{a, b\} \leq \{b, a\}$ ✓

Transitivity : $\{a\} \subseteq \{a, b\}$

$\{a, b\} \subseteq \{a, b, c\}$

$\Rightarrow \{a\} \subseteq \{a, b, c\}$ ✓

$\therefore$ The tuple $(S, \subseteq)$ is a partial order

Clocks

Physical clocks
- time of day clocks
- monotonic clocks

Logical clocks

Ordering of events only!

Lamport clock

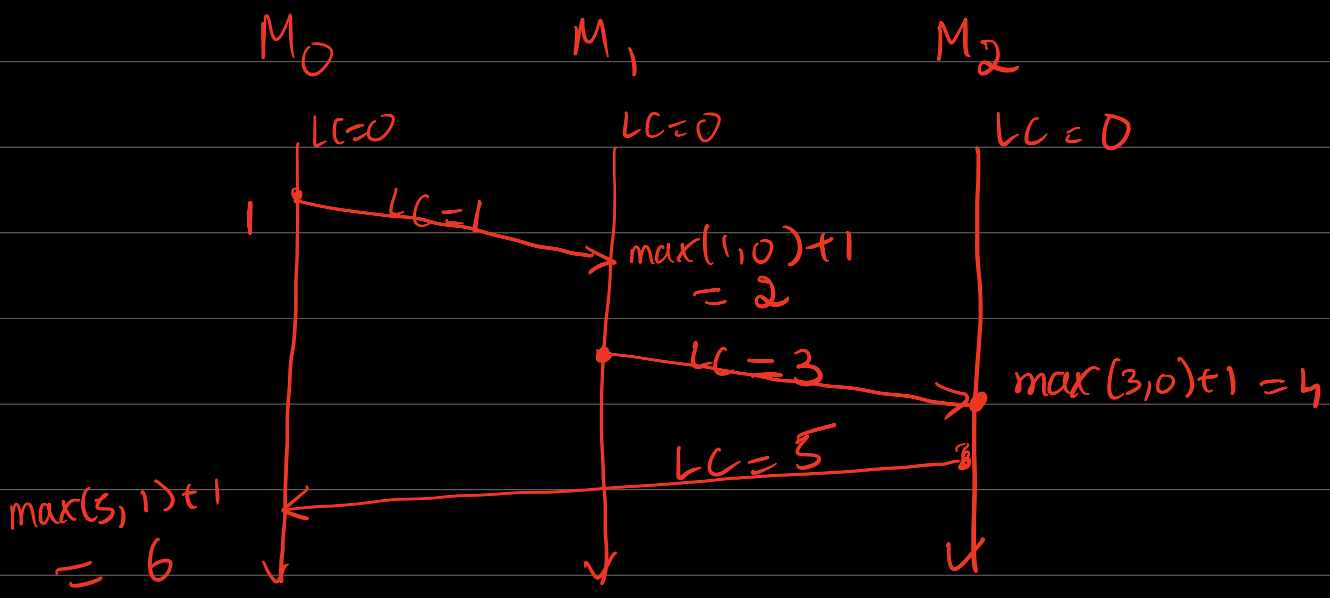
LC(A) - the lamport clock of event A

LC(A) = 3

Clock condition

→ if A → B then LC(A) < LC(B)

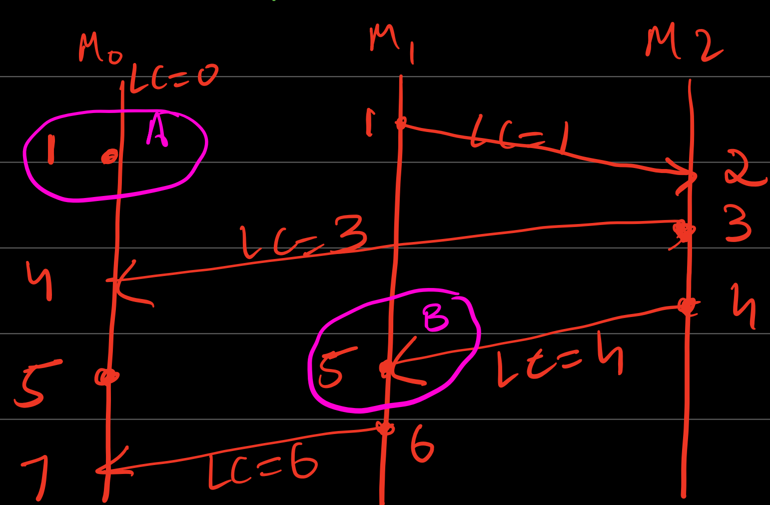
Lamport clocks are consistent with causality (i.e happen before relation)



Lamport clock algorithm

- Every process has to keep a counter, init to 0
- On every event on a process, increment by 1
- when sending a message, include its current counter
- When receiving a message, a process its counter to max(local, received) + 1

$LC(A) < LC(B) \not\Rightarrow A \rightarrow B$ Not always



On a lamport diagram, if you can reach B from A, then $A \to B$

For the above example, we can't reach B from A. $\therefore A \not\to B$ even though $LC(A) < LC(B)$

$\therefore$ logical clock is consistent with causality:

$A \to B \Rightarrow$ logical clock of A < logical clock of B

lamport clocks have this property!!

Logical clock characterizes causality:

logical clock of A < logical clock of B $\Rightarrow A \to B$

lamport clocks DONT this property!!

What can you do with $P \Rightarrow Q$?

Take contrapositive

$\sim Q \Rightarrow \sim P$

not

$A \to B \Rightarrow LC(A) < LC(B)$

$$\boxed{\neg(LC(A) < LC(B)) \Rightarrow \neg(A \to B)}$$

Rule out things NOT HAVING caused other things!

Above eq shows either $B \to A$, or $A \| B$ (Not sure which one is true, but definitely $A \to B$ is not true)

Leslie Lamport implemented Lamport Clocks!

Vector clocks

$$A \to B \Leftrightarrow VC(A) < VC(B)$$

if and only if

1) every process keeps a vector of integers initialized to zeros

vector of length N for N processes

[0, 0, 0]

2) On every event, a process increments its own position in its vector clock. This is for ALL events

3). When sending a message, process includes it's current vector clock. Occurs after step 2, since 'send' in an event

4). When receiving a message,

a process will update its vector clock to max(local, received)

This occurs AFTER incrementing its position, since receives are events

Max of vectors
Pointwise max!

$$[1, 2, 4] \; \& \; [7, 0, 2]$$
$$\max = [7, 2, 4]$$

[5,0,0] ⇒ vector clock gives more info
5 says 5 events in A &
0 in both B & C