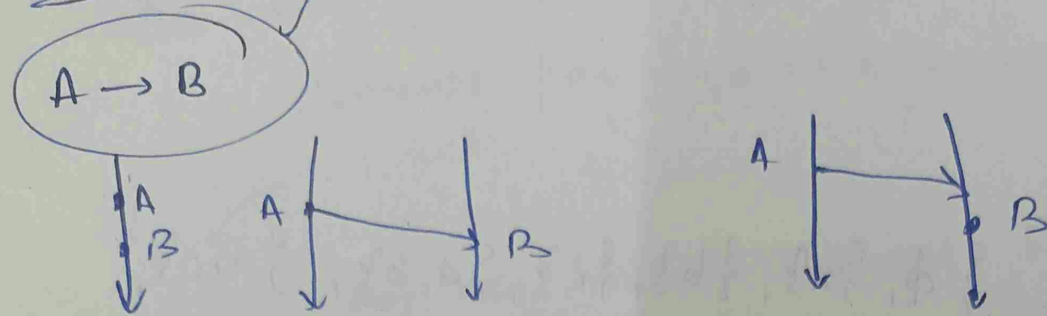
Lec-03

→ Irreflexive partial order



The set of all of the events in the system are ordered by this partial order,

but the "happen before" is a slightly weird partial order, as it doesn't has reflexivity,

A Standard Partial order has: (set S, binary relation ≤)

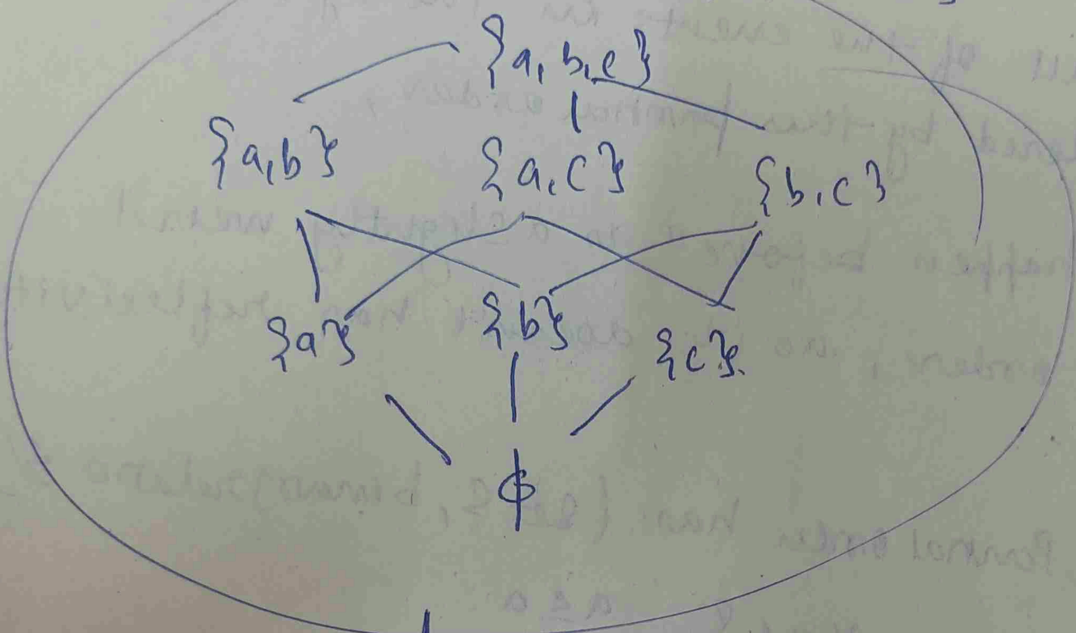
- Reflexive → $\forall a \in S$, $a \leq a$.
- Antisymmetry → $\forall a, b \in S$, $a \leq b$ & $b \leq a$ then $a = b$.
- Transitive : $\forall a, b, c \in S$

  $a \leq b$ & $b \leq c$ then $a \leq c$.

Set Inclusion

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

Power set $= S = \{\phi, \{a\}, \{b\}, \{c\}, \{a, b\}, \{a, c\}, \{b, c\}, \{a, b, c\}\}$



So, this is a example of a partial order that is truly a partial order., as all three holds.

Reflexive -> $\{a\} \subseteq \{a\}$
subset

Antisymmetry ✓

Transitivity ✓

~~Iska~~ lekin partial order mai kisi bhi do elements kar paya aisa jaroori ni h.

Jisme aap kisi bhi do element ko relate kar pao usse "total order" kehte hai

eg:- Integer.

$S = \mathbb{N}$ relation $\leq$ → less than or equal

N

⋮

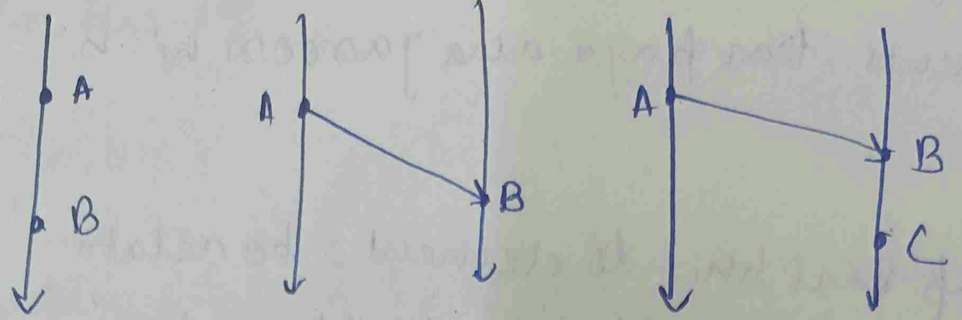
2

|

|

0

Ye h total order.



So, if you have got a bunch of events, you can use this defn to figure out to know what happened before what ~~or~~

Previously, we looked at a set of events and we sort of computed by hand what was in the happens before relation ~~for~~

but we wanna come up with an algorithm by which a computer can straightforwardly compute what's in that happens before relation.

Alternative to using physical clock

↳ logical clock

⇓

Only tell about ordering of events and don't tell anything about elapsed time or time of day

~~So~~ The simplest type of logical clocks:

<u>Lamport clock</u>

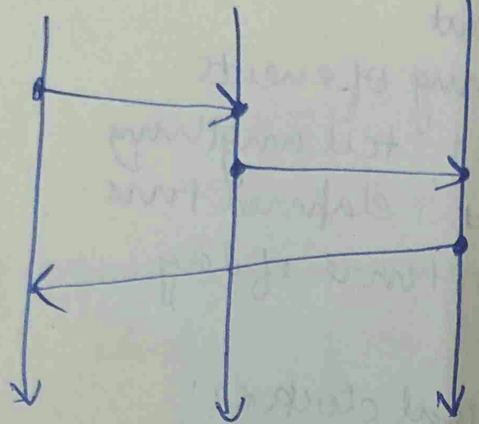
→ Just a way of assigning numbers to events

eg LC(A) = 3 → Ye 3 kya h?.

$\boxed{\text{If } A \to B \text{ then } LC(A) < LC(B)}$

↳ Lamport clocks are consistent with causality.

Assigning LCs to Events



(i) Every process has a counter initially 0.

(ii) On every event, a process increments its counter.

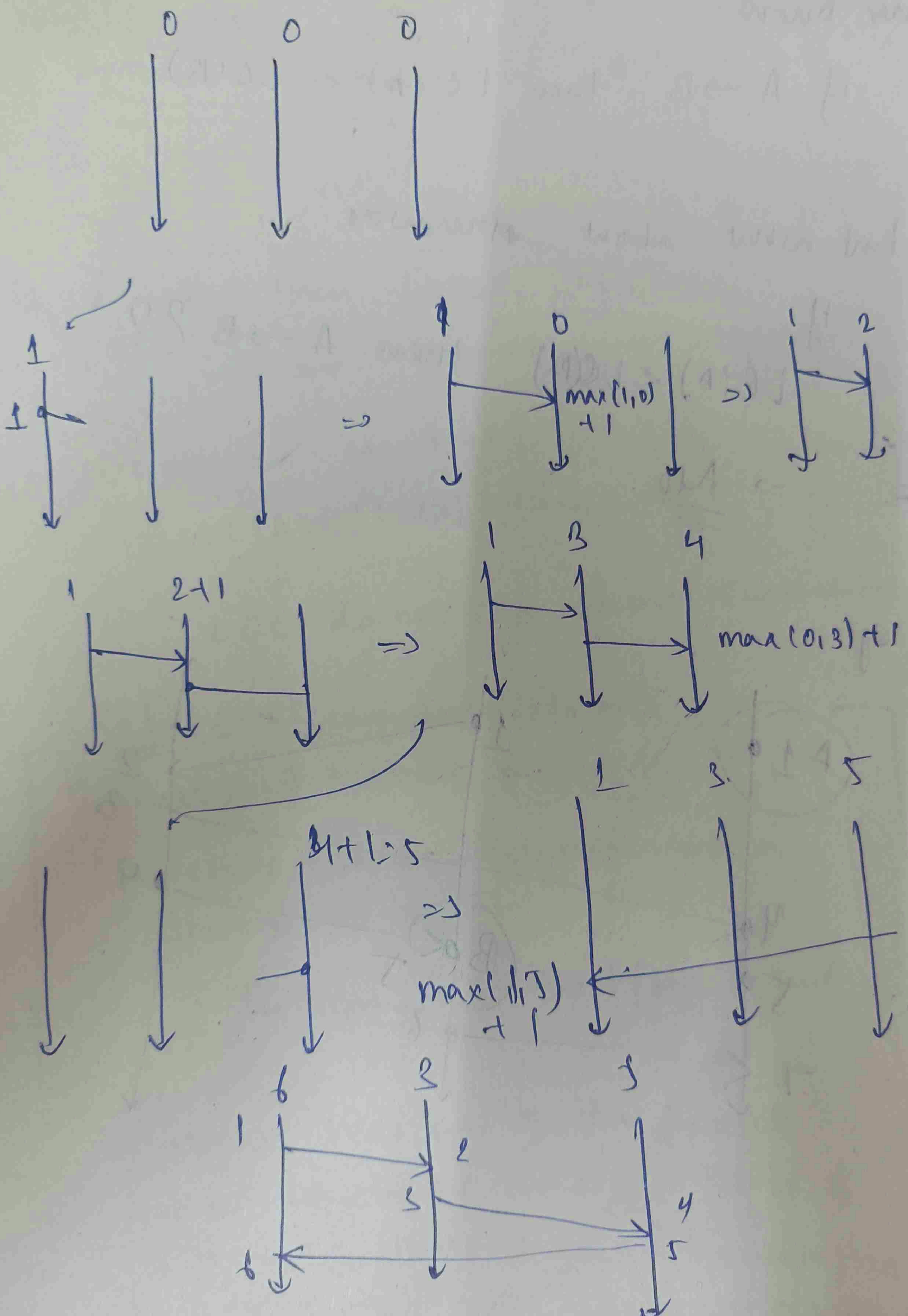
(iii) When sending a message, a process includes its current counter along with the message.

(iv) When receiving a message, set your counter to

$$\boxed{\max(\text{local counter}, \text{message counter}) + 1}$$

So, we have a way of assigning numbers to events now.

Let's try.



0 0 0
1
1
=>
1 0
max(1,0)
+1
=>
1 2
1 2+1
=>
1 3 4
max(0,3)+1
4+1=5
=>
1 3. 5
max(1,3)
+1
6 3 5
1 2 3 4 5 6

we know

if $A \rightarrow B$ then $LC(A) < LC(B)$

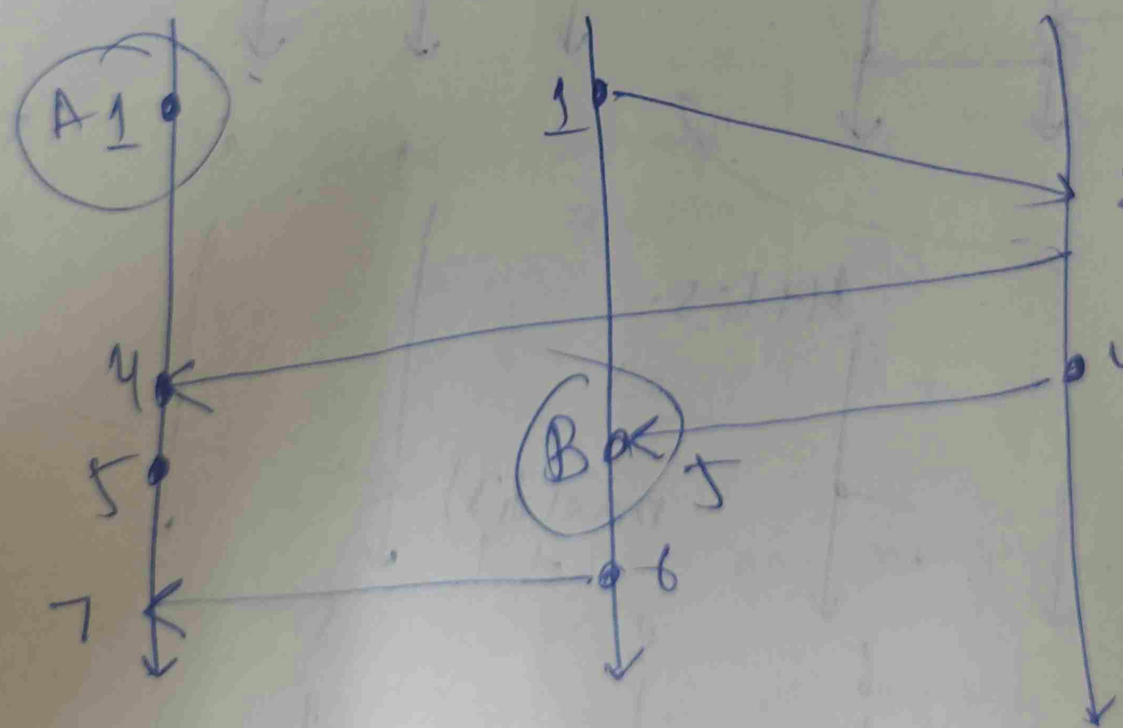
but what about other way

if

$LC(A) < LC(B)$ then $A \rightarrow B$??

$\dashrightarrow$ No

eg:-



Although ~~A~~ $LC(B) > LC(A)$ but I can't say by ~~theorem~~ rules that ~~A~~ $A \to B$,

if $A \to B$ then $LC(A) < LC(B)$

LCs are consistent with causality

it's not the case that

if $LC(A) < LC(B)$ then $A \to B$

LCs do not characterise causality.

Rs paper (Recommend to read)

Schwarz & Mattern (1994)

"Detecting Causal Relationships in Distributed System:

In Search of the Holy Grail".

Mattern was one of the people who developed another kind of logical clock that does characterize causality unlike Lamport clock

Even after the limitations of Lamport clock, we still use it cause.

if $A \to B$, then $\neg Q \Rightarrow \neg P$

if
$\neg(LC(A) < LC(B))$, then $\neg(A \to B)$

(

if $LC(A)$ is not less than $LC(B)$
then A did not happen before B

we can't say that A happened before B