Time, liveness & Safety

Last class

- What, how & why.
- The asynchronous model
  - No bounds on MESSAGE DELAYS (time b/w send & receive)
  - No bounds on PROCESSING DELAY
  - No global clocks
- Network & FAIRNESS
- Message passing

TODAY: How to REASON about PROTOCOLS (ALGORITHMS) designed for the ASYNCHRONOUS MODEL.

REASONING ABOUT PROGRAMS

- Is it correct?
- What outputs can it produce?

- How long does it take (to execute some op)?
  HOW MANY MESSAGES DOES IT SEND (...)?
- IS IT FAULT TOLERANT?
- WHAT CONSISTENCY GUARANTEES DOES IT PROVIDE?

...

```
int i = 0;                      (1)
while (true) {                  (2)
    if (i % 7 == 0 && i % 2 != 0) {   (3)
        print(i)                (4)
    }
    i++;                        (5)
}
```

(a) PRINT ONLY ODD NUMBERS

(b) PRINTS IN ASCENDING ORDER

```
swap(x, y) {
    x = x ^ y
    y = x ^ y
    x = x ^ y
}
```

(a) $x_{out} = x_{in}$; $y_{out} = y_{in}$

OBSERVATION: CORRECTNESS OFTEN DEPENDS ON ORDER OF OPERATIONS

Today

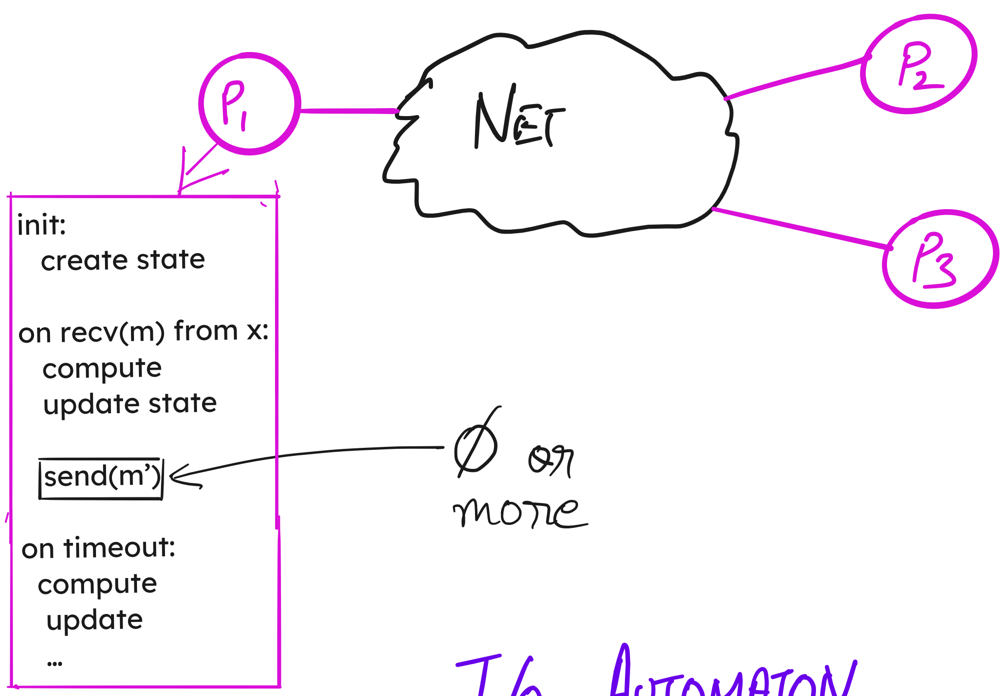
Q1 Modeling Order Of Events In Distributed Systems :— Traces

Q2 Correctness In Terms Of Trace Properties

- Safety
- Liveness

Q3 Recovering Traces From Actual Executions

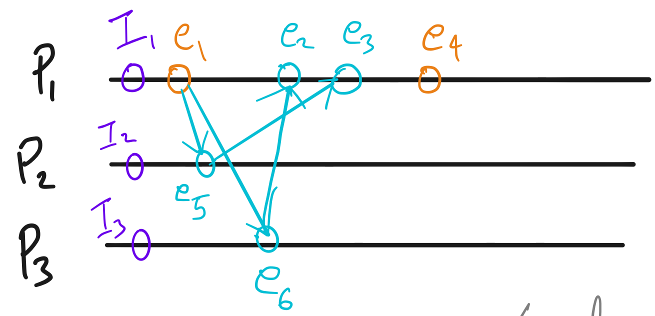
But First: Need To Add Something To Model



I/O Automaton

(Nancy Lynch & Others)

$<$ $e_6 \leq e_5$ $e_1 < e_2$



ORDER OF EVENTS ON ONE PROCESS

Local to P_1 → $\{(I_1, e_1)$; (e_1, e_2) . . .

(I_2, e_5); (e_1, e_5);

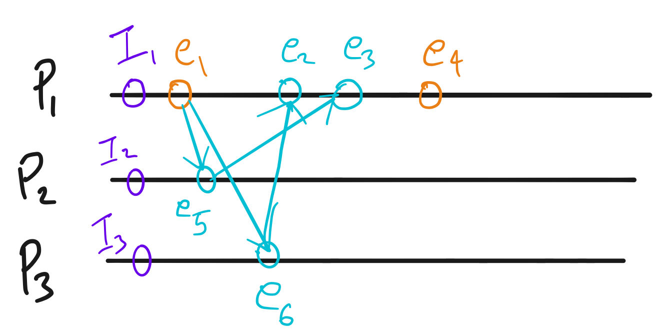
(e_5, e_3) (e_1, e_6)

(e_6, e_2) (e_2, e_3)

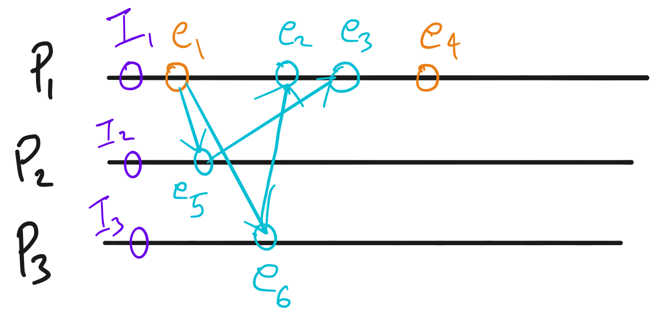
→ Partial Order

CAUSALITY

PARTIAL ORDER



Total Order



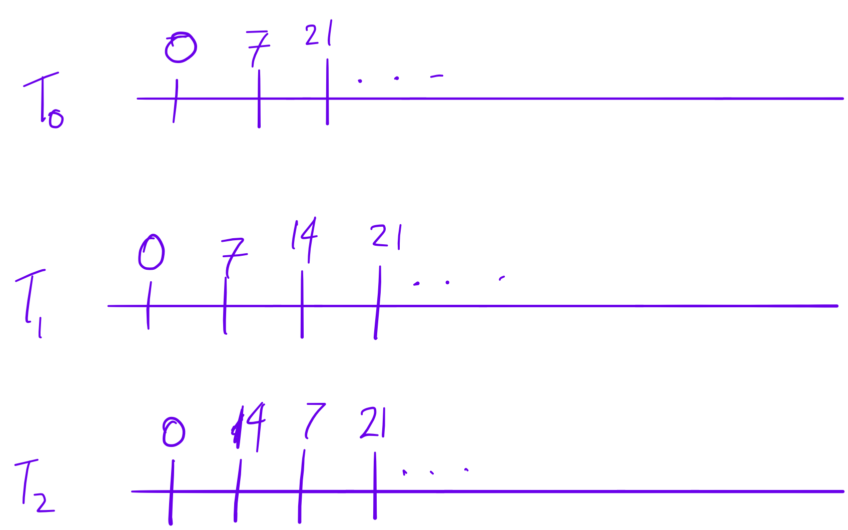
Trace

$\mathbb{Z}^+ \to$ event

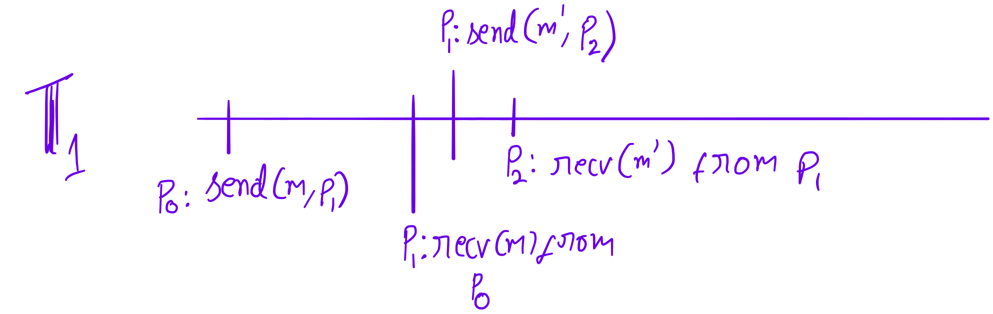
Correctness As A Trace Property

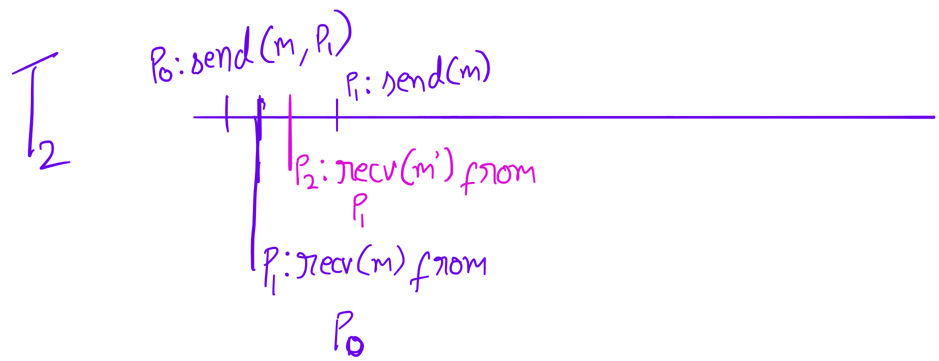
Correctness conditions restrict what traces can be produced by the program

(b) Prints In Ascending Order

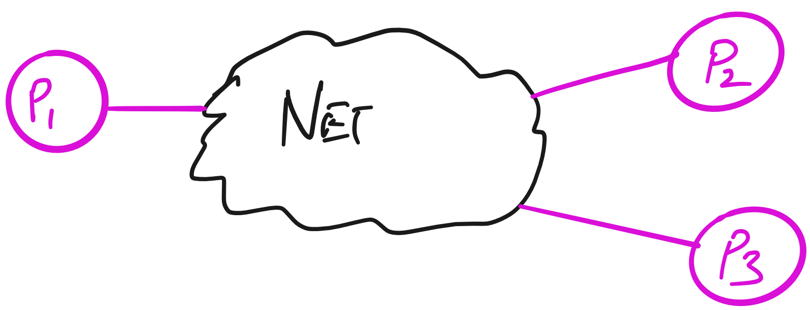


No messages are received without being sent first



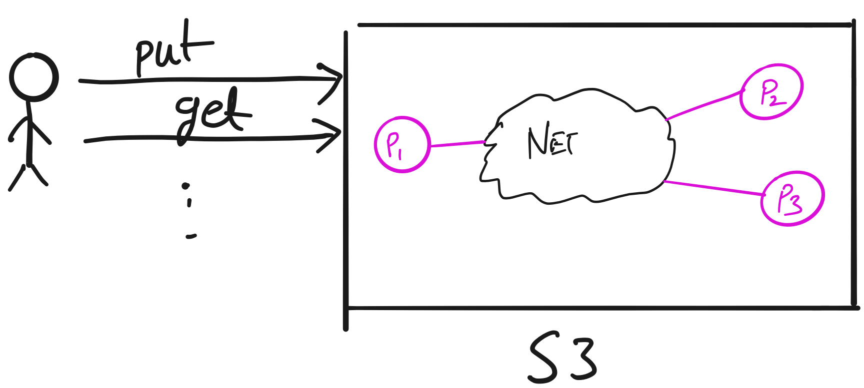


Why look at correctness this way?



- Concurrent processes

- Often visible to the rest of th world as one system



- Correctness is all about what

States anyone outside can observe

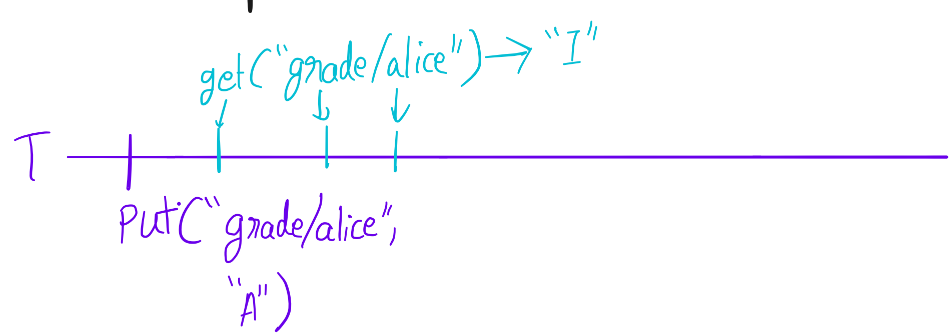
Correctness Invariants

Safety: Condition Should Always Hold.

- No recv before send.
- No even number is printed

Liveness: Condition Will Eventually Hold

- Puts to S3 will eventually update val



- A message sent infinitely often will be received infinitely often

Test: Give a trace T can you find an extension (SUFFIX) S, s.t.

TS is admissible + meets liveness condition

Reading: Liveness, uniform liveness, absolute liveness

- Liveness. Given T, $\exists S$ s.t. TS is admissible & the liveness condition holds

- Uniform $\exists S$, s.t. $\forall T$, if T is admissible TS is admissible & liveness condition holds

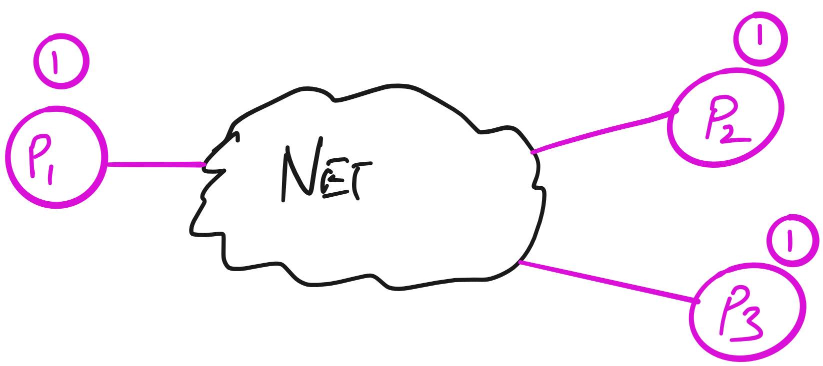
T that is admissible

- Absolute: Given a trace T that is admissib… for which the LIVENESS CONDITION holds
  ▷ T' an admissible trace

T'T is admissible + liveness holds.

Clocks:

RECOVERING TRACES IN REALITY



Assumptions:

- No global clock
- Processes have local clocks but

ASYNCHRONY:

ⓐ Run at unknown freq.

ⓑ Freq. differs b/w processes

What is desired: CAPTURE CAUSALITY

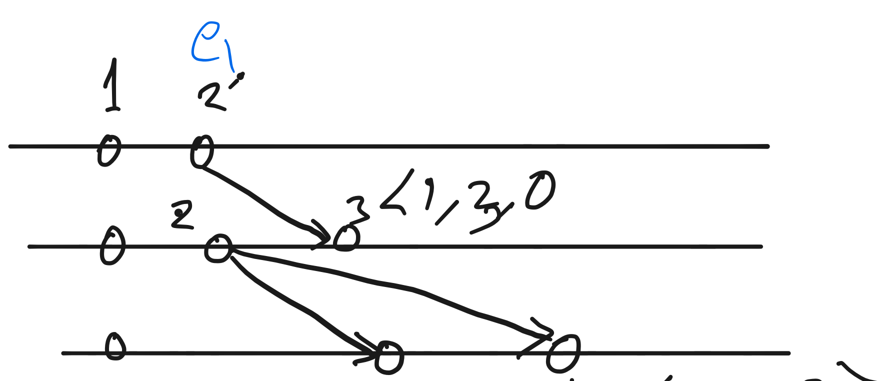
- Lamport's approach

- Vector clocks

Where Chandy-Lamport fits in?

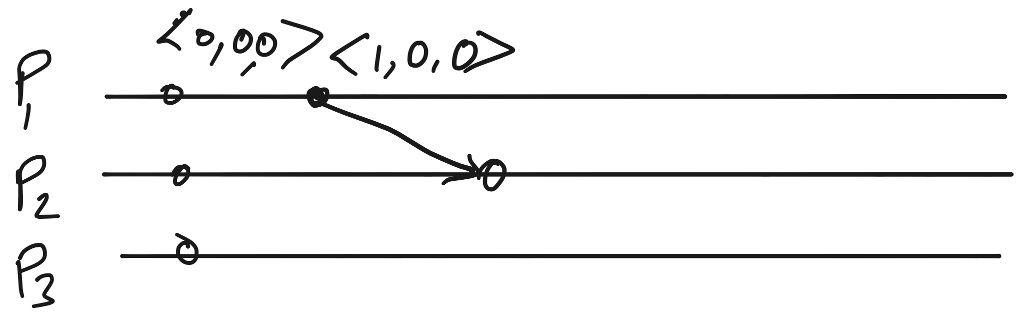
assert

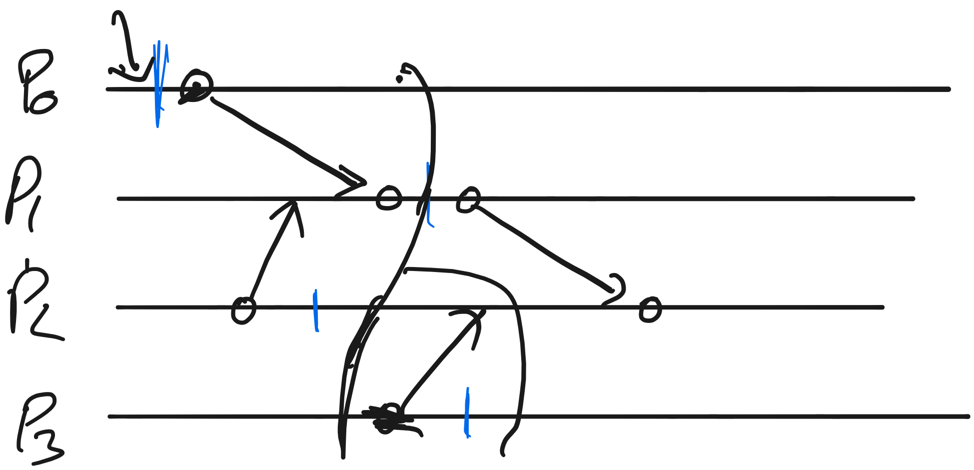
```
int x
recv m
    x++
```



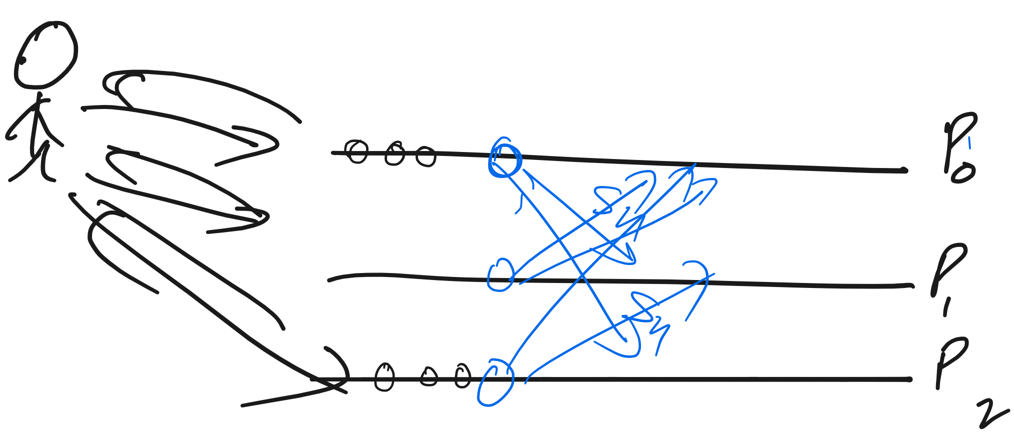
3 4 <0, 2, 2>
e_2

1





Causally Consistent Snapshot



m1
m2

... are ordered

Channels with