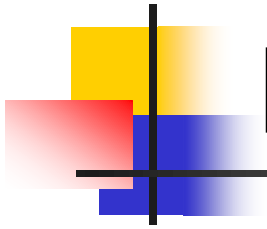
Verification of Real-Time Systems



Example: Petersson's algorithm

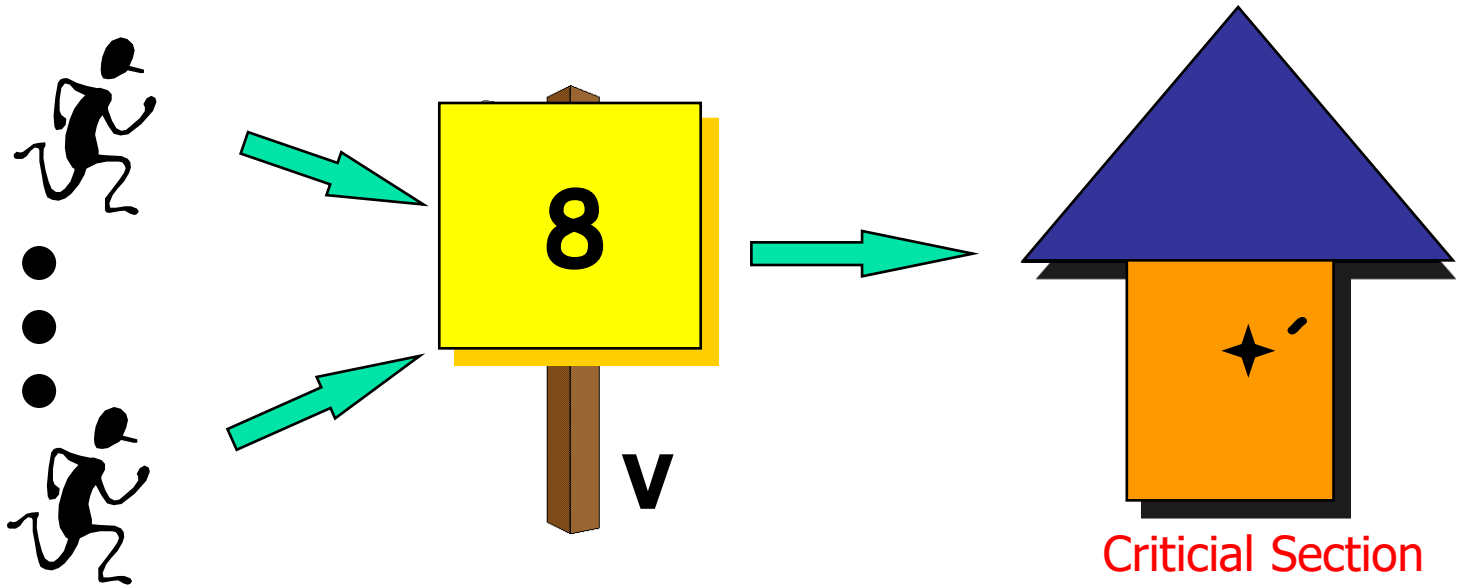
turn: shared variable

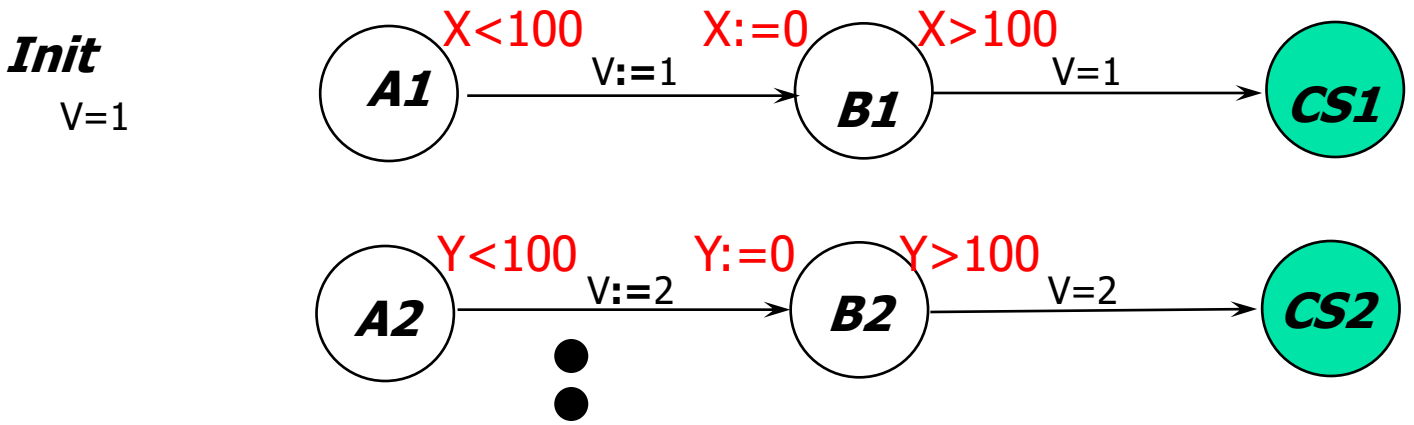
- Process 1
- Loop
- flag1:=1; turn:=2
- While (flag2 and turn=2) wait
- CS1
- flag1:=0
- End loop

- Process 2
- Loop
- flag2:=1; turn:=1
- While (flag1 and turn=1) wait
- CS2
- flag2:=0
- End loop

Question: no more than one process run in CS?

Example: Fischer's Protocol

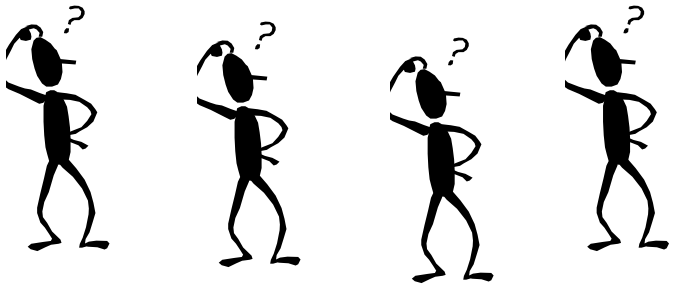




Example: the Vikings Problem

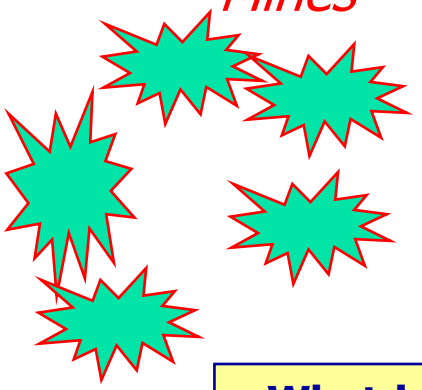
UNSAFE

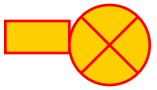
SAFE



5 10 20 25

Mines



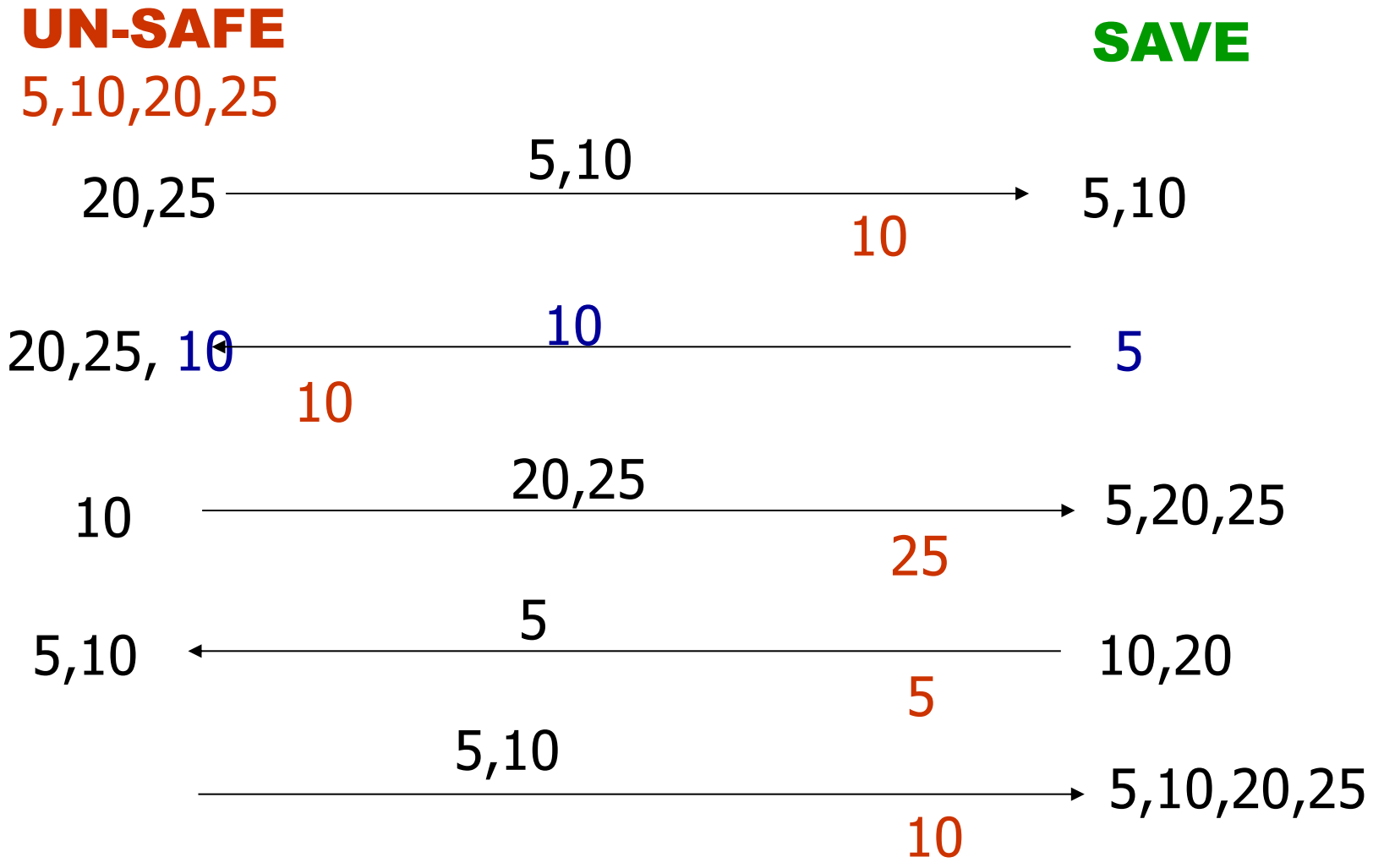


Torch

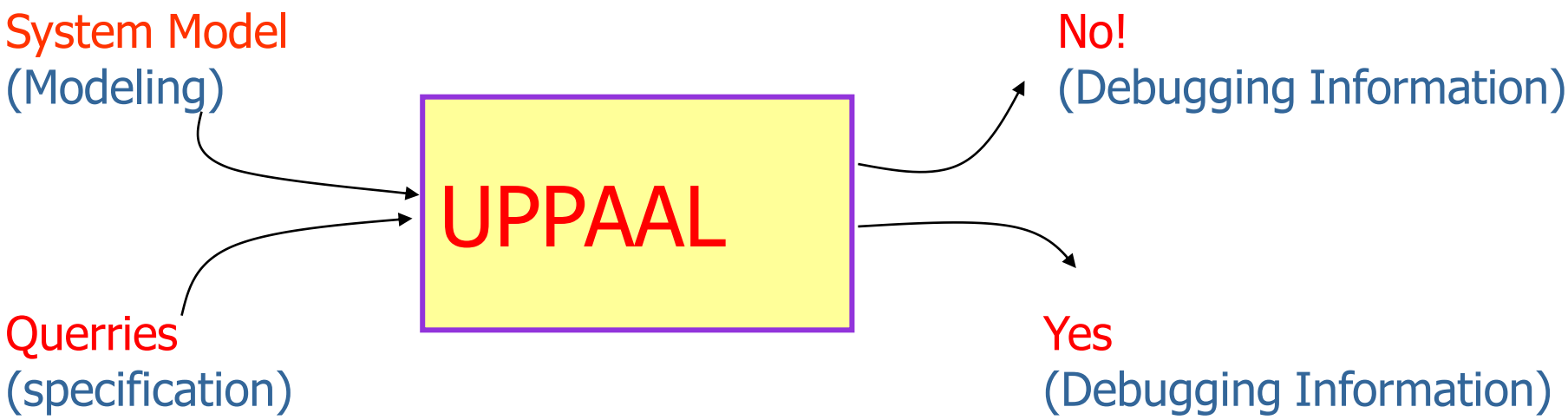
At most 2 crossing at a time
Need torch

What is the fastest time for getting all vikings on the safe side ?

Solution


UN-SAFE
5,10,20,25
SAVE
20,25
5,10
10
5,10
20,25, 10
10
10
5
10
20,25
5,20,25
25
5,10
5
10,20
5
5,10
5,10,20,25
10

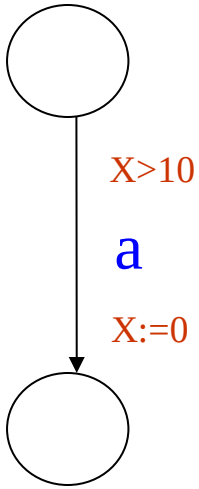
UPPAAL **A tool for verification of real-time systems**



MODELING

How to construct Model ?

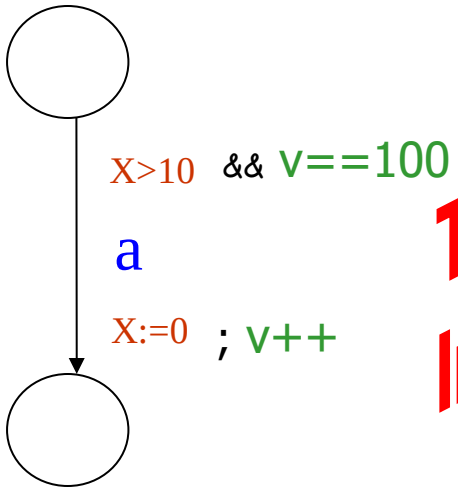
Modeling Real Time Systems



Timed Automaton

- Events
  - synchronization
  - interrupts
- Timing constraints
  - specifying event arrivals
  - e.g. Periodic and sporadic

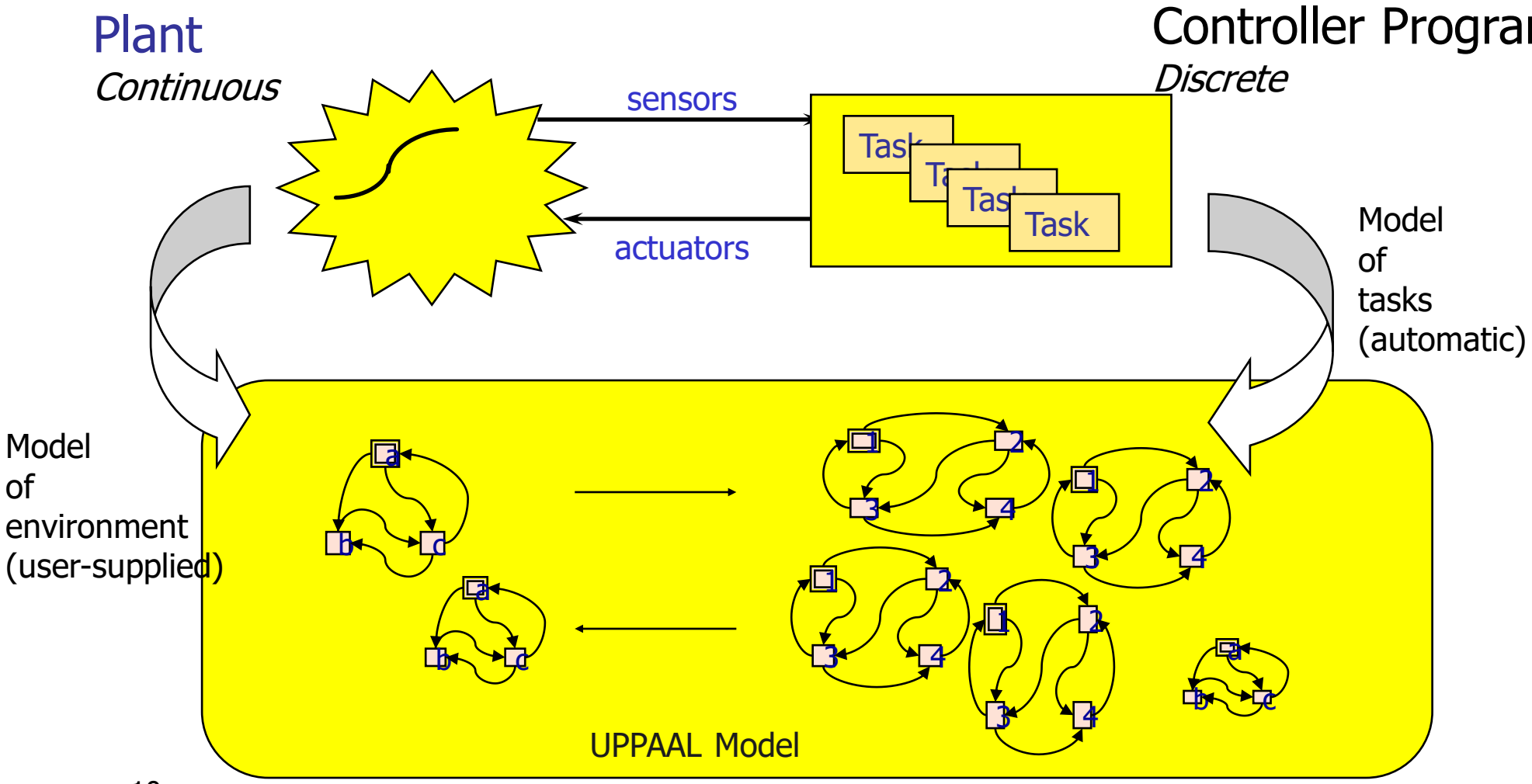
Modeling Real Time Systems



Timed Automaton In UPPAAL

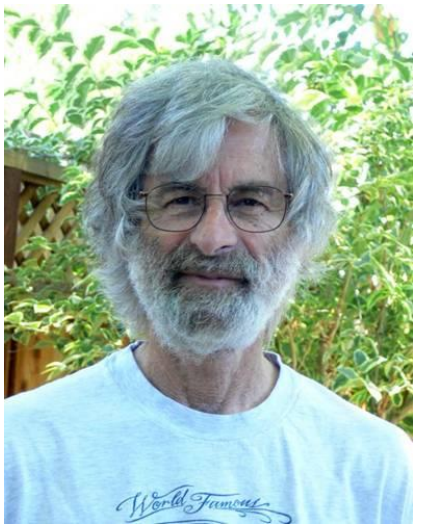
- Events
  - synchronization
  - interrupts
- Timing constraints
  - specifying event arrivals
  - e.g. Periodic and sporadic
- Data variables & C-subset
  - Guards
  - assignments

Construction of Models

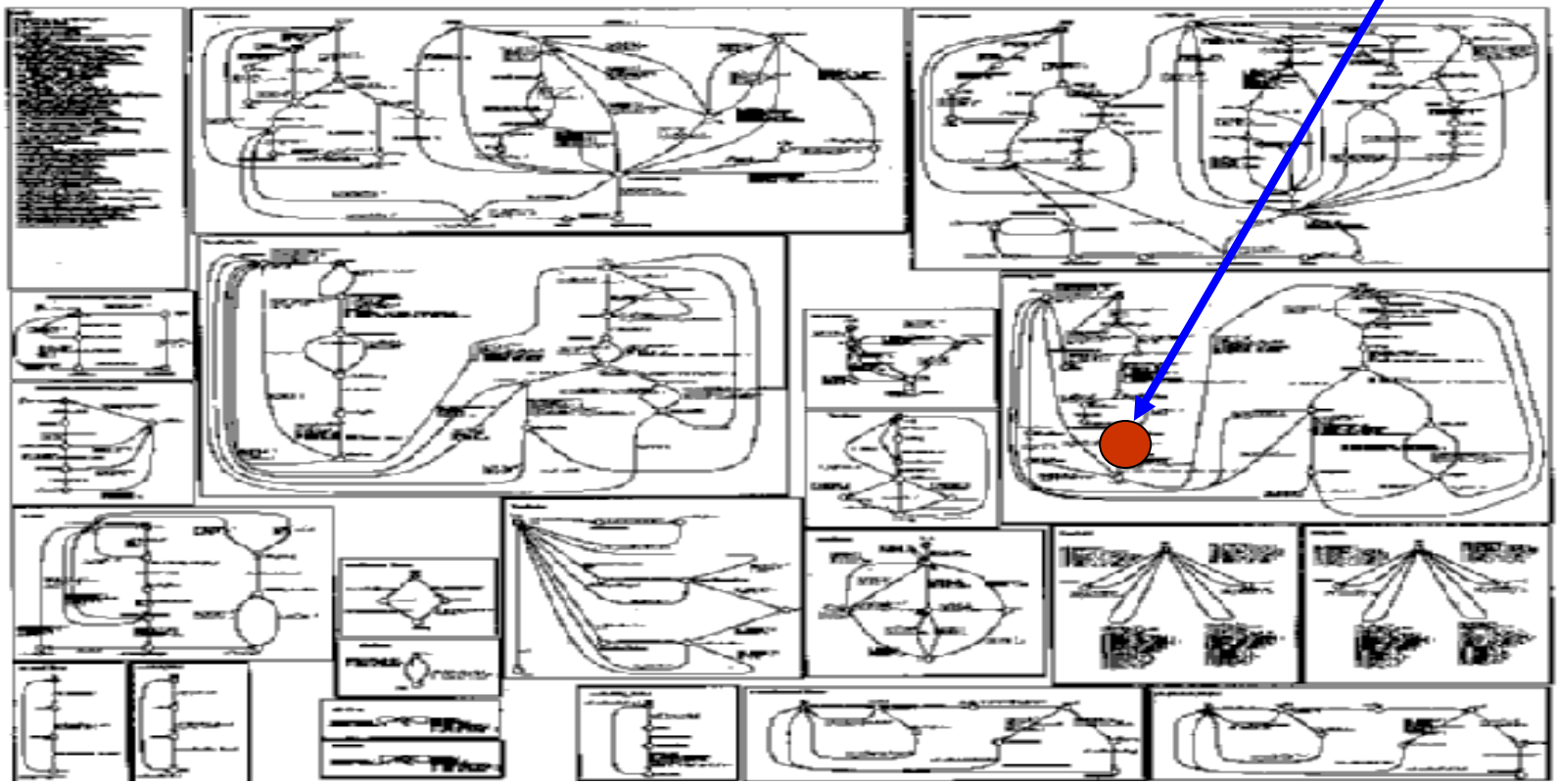


Specification=Requirement, Lamport 1977

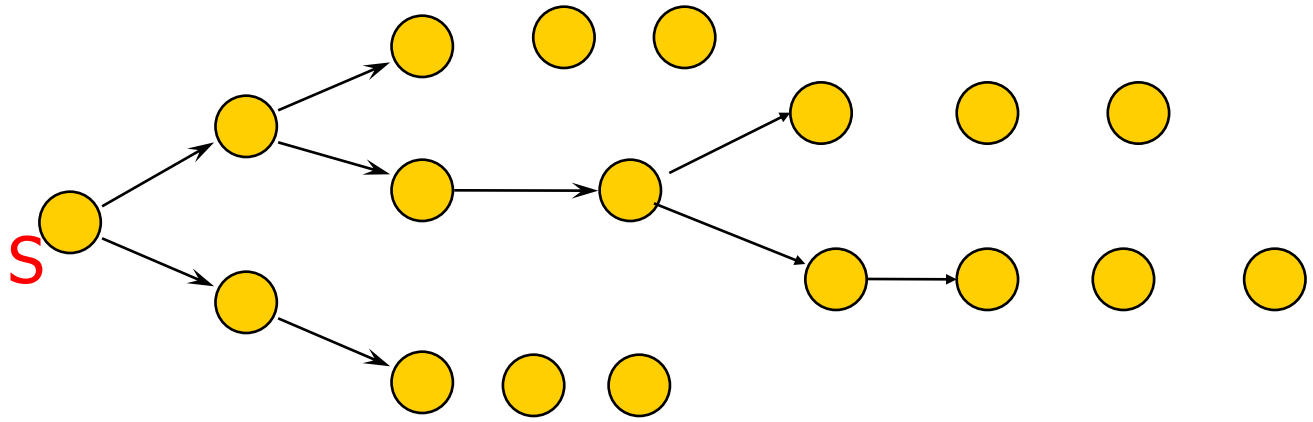
- Safety
  - Something (bad) should not happen
- Liveness
  - Something (good) must happen/should be repeated



An 'abstract' version of a fieled bus protocol



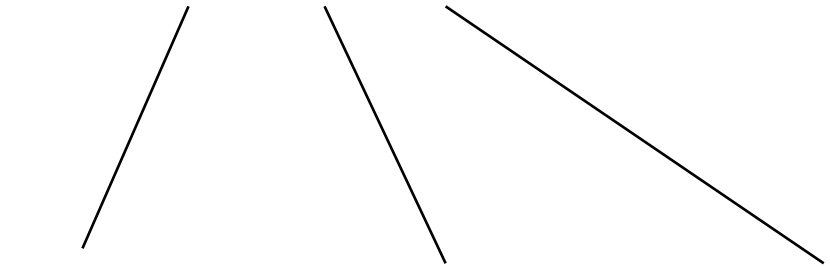
Computation Tree (of a system)



all possible executions of a system

Local properties of a state

P ::= A.n | g_c | g_d | not p | p or p | p and p | p imply p



a location in automaton A -- the current location of A is "n"

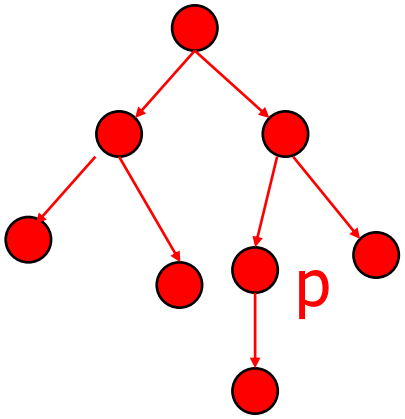
Clock Constraint X<=3 y>10 etc

predicate over data variables Or any logical expression in C e.g. i==100 && m<=168

Specifications/Querries in UPPAAL

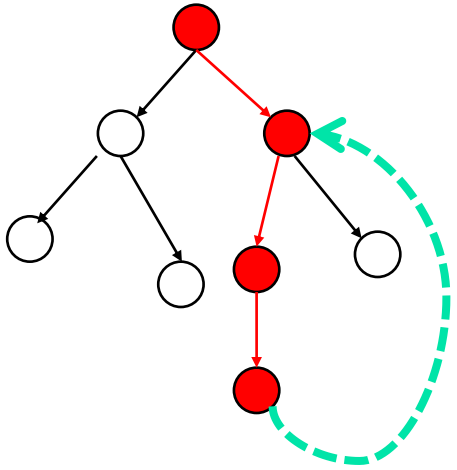
P is an invariant

A[] p



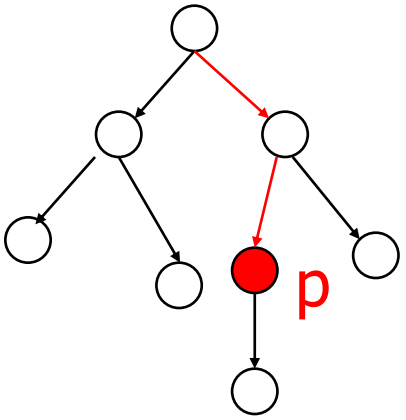
P may be true globally

E[] p



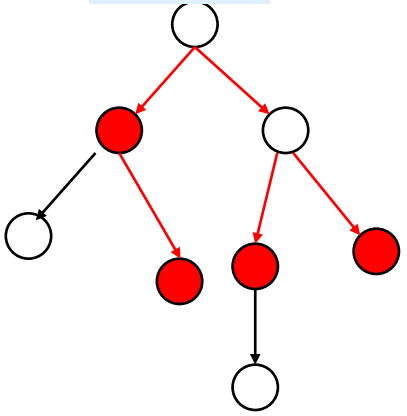
P is reachable/possible

E<> p



P is "guaranteed"

A<> p



Where p is a local property of the states; it can be a clock constraint, an automaton location or a predicate over data variables

Example querries in UPPAAL (safety properties)

- Reachability properties
  - E<> P.stop
  - E<> (y>200)
  - E<> (time>60 imply viking4.safe)
  - E<> (viking1.safe & viking2.safe & viking3.safe & viking4.safe)
  - A<> (y>200)
- Invariant properties
  - A[] not (P1.CS and P2.cs)
  - A[] (temp > 10 & speed < 120)
  - A[] (i < 100)
  - A[] (x>10 imply i>100) -- After 10, i should be larger than 100
- Deadlock-freedom
  - A[] !deadlock

VERIFICATION

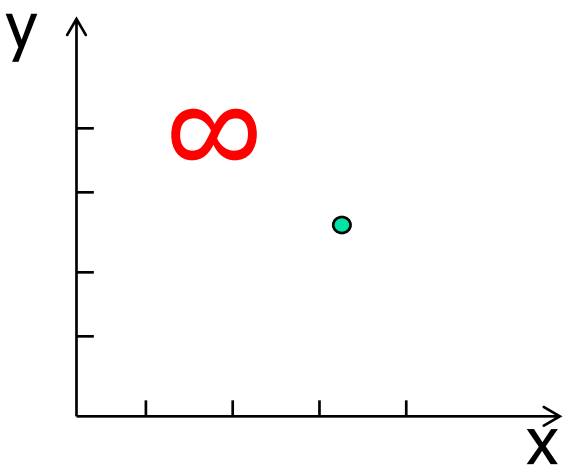
Model meets Specs ?

Two basic verification algorithms

- Reachability analysis
  - Checking safety properties
- Loop detection
  - Checking liveness properties

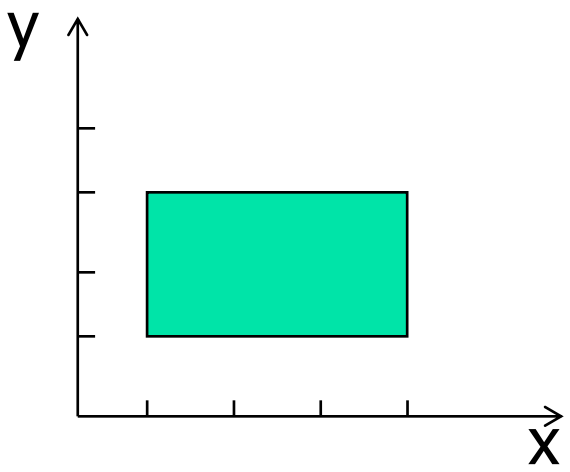
REACHABILITY ANALYSIS using ZONES

State
(n, x=3.2, y=2.5)

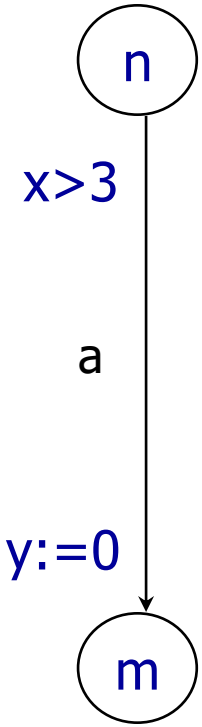


Symbolic state (zone)
(n, $1 \le x \le 4, 1 \le y \le 3$)

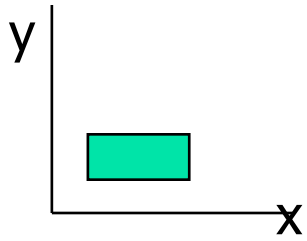
Zone:
conjunction of
x-y~n, x~n



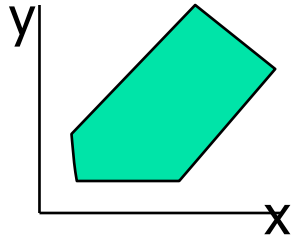
Symbolic Transitions



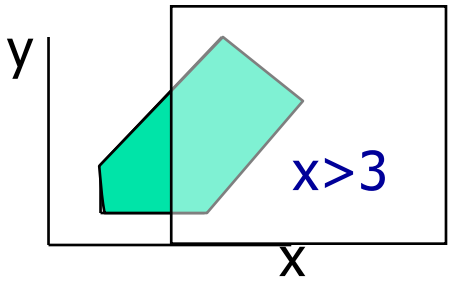
1<=x<=4
1<=y<=3



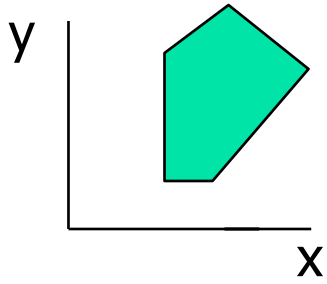
delays to



1<=x, 1<=y
-2<=x-y<=3

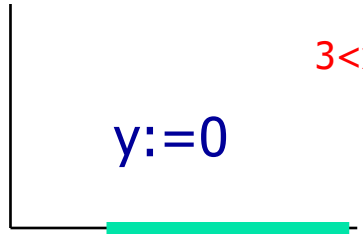


conjuncts to



3<x, 1<=y
-2<=x-y<=3

projects to

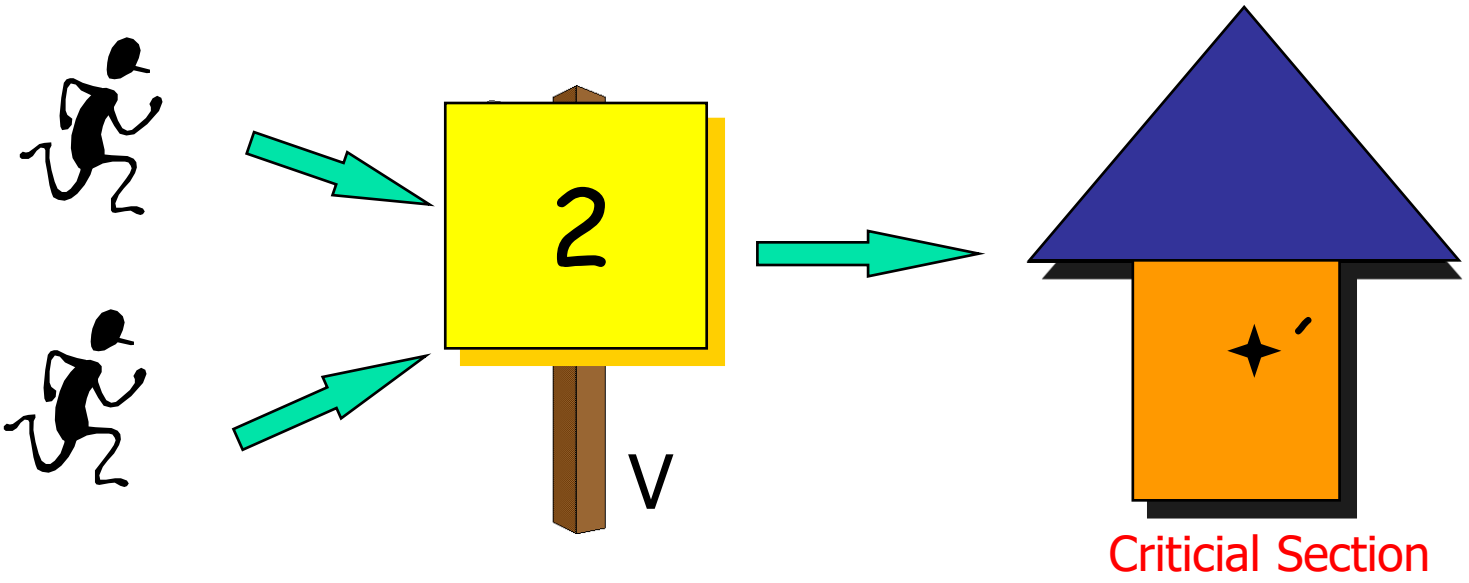


3<x, y=0

Thus (n, 1<=x<=4,1<=y<=3) =a=> (m, 3<x, y=0)

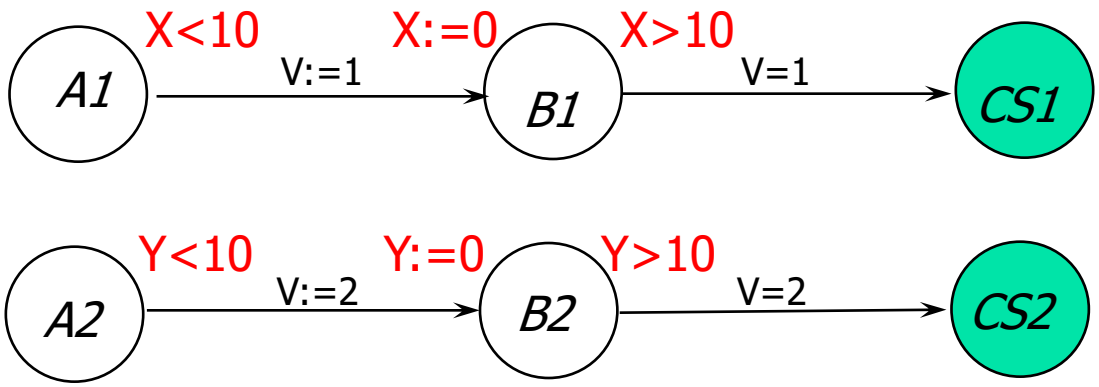
Fischer's Protocol

analysis using zones

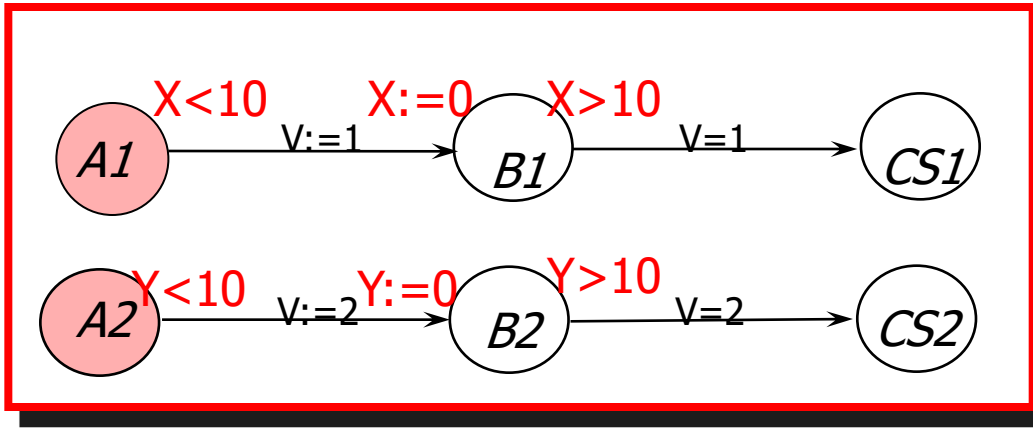


Initially

V=1

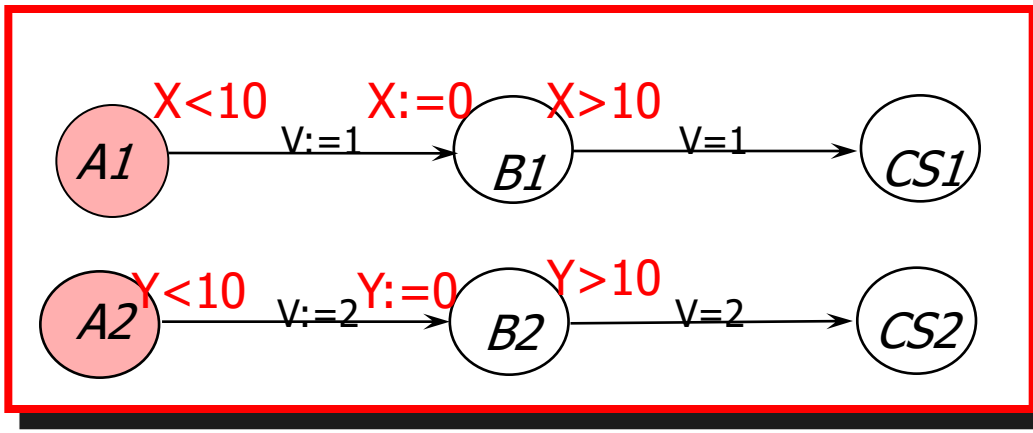


Fischers cont.



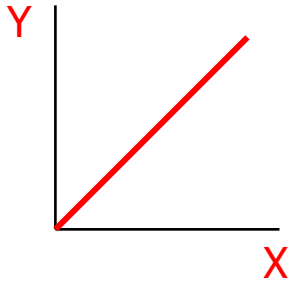
Untimed case

Fischers cont.

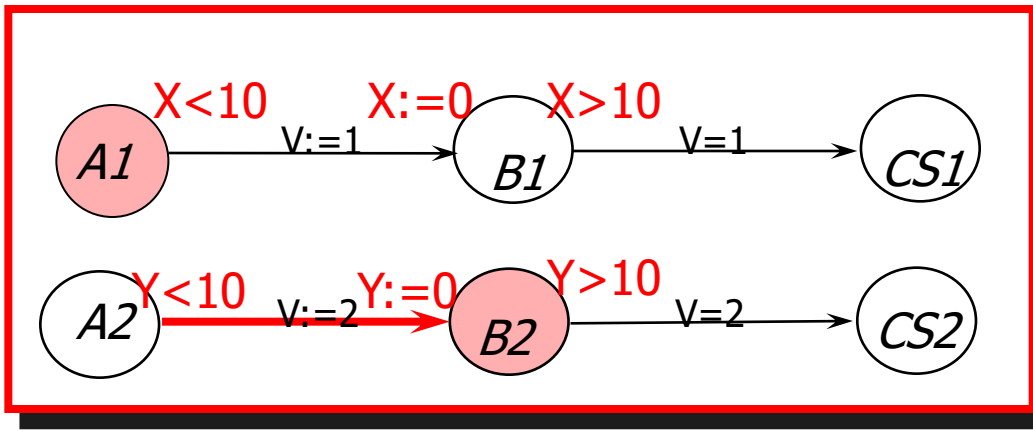


Untimed case

Taking time into account

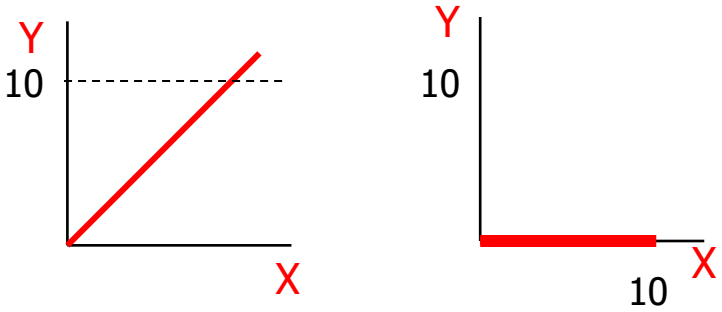


Fischers cont.

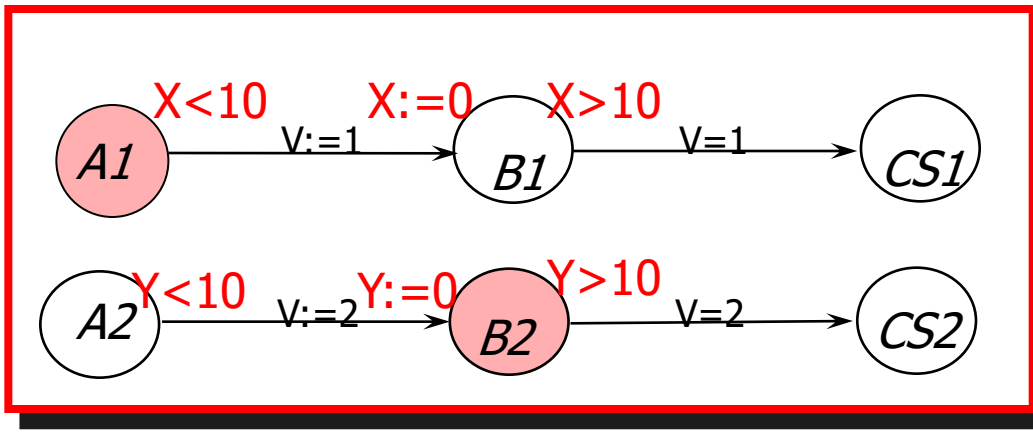


Untimed case

Taking time into account

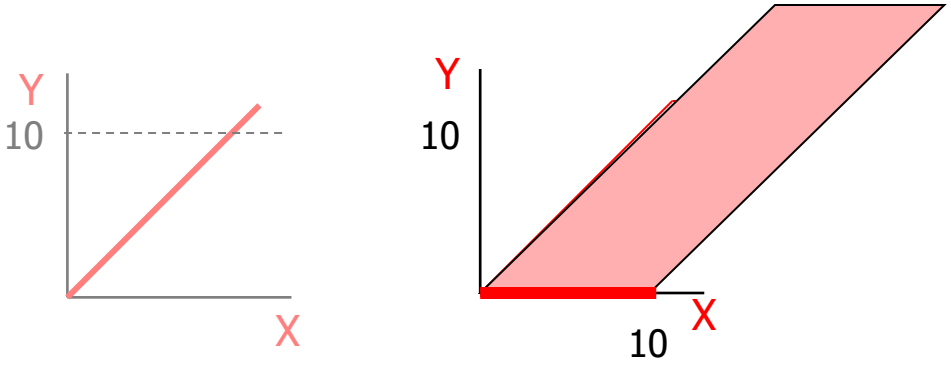


Fischers cont.

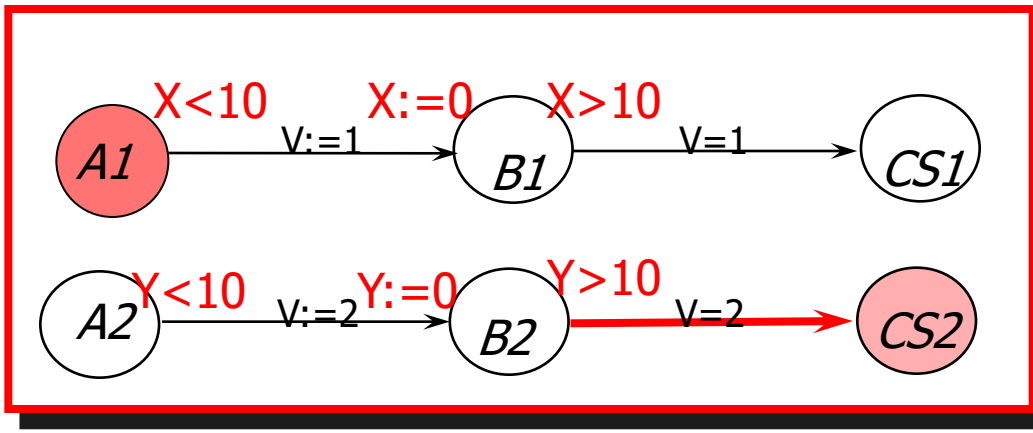


Untimed case

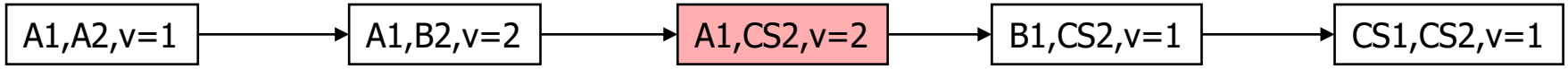
Taking time into account



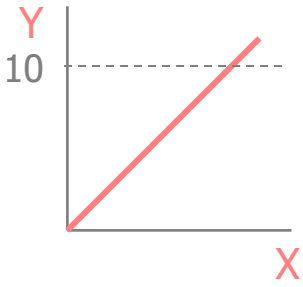
Fischers cont.

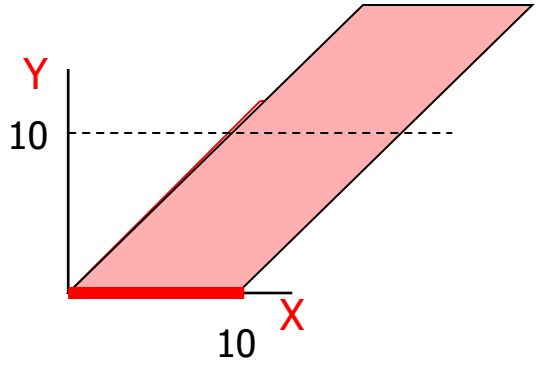


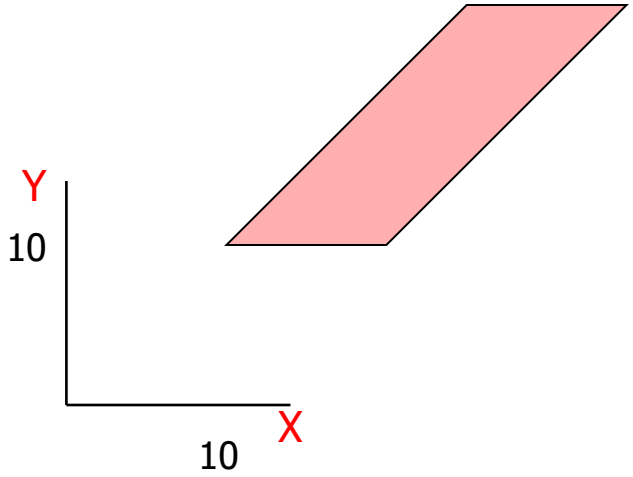
Untimed case



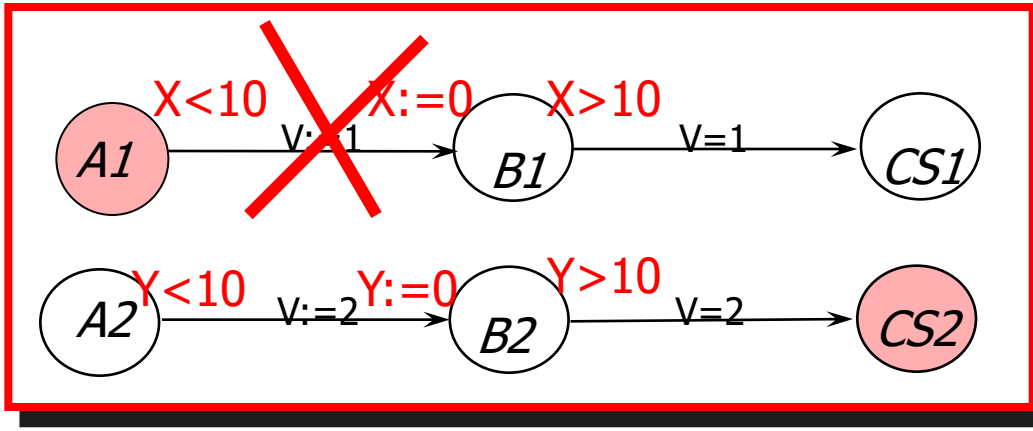
Taking time into account



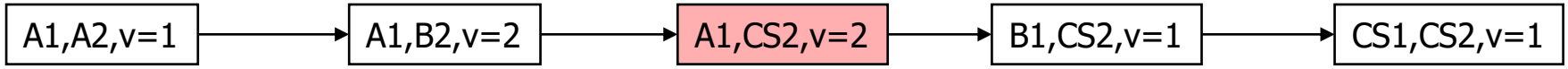




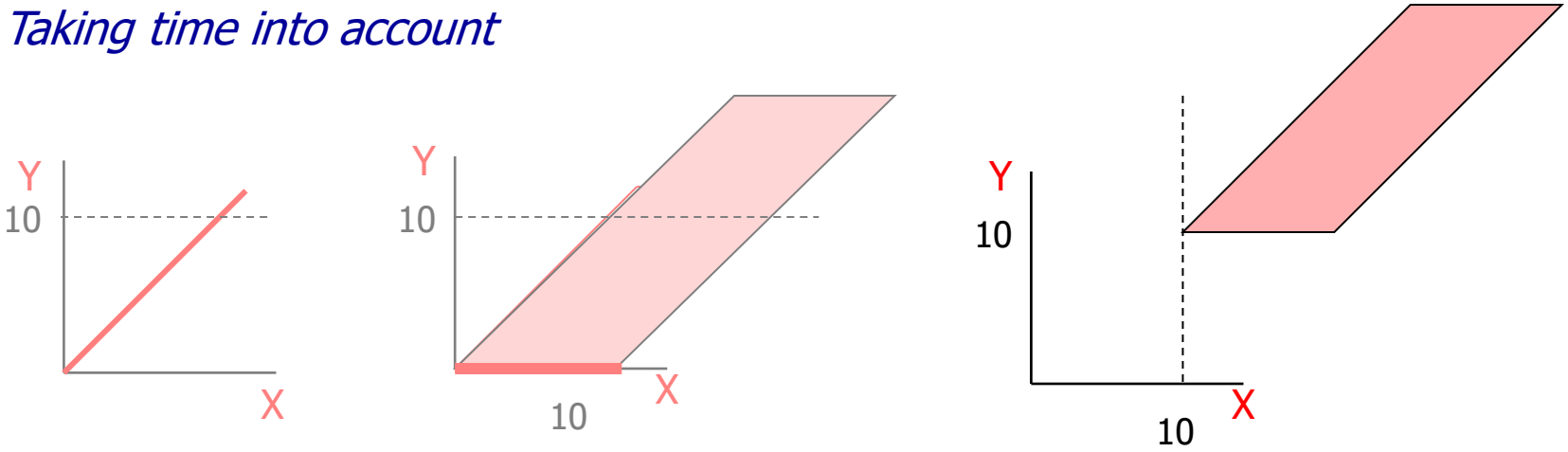
Fischers cont.



Untimed case



Taking time into account



Zones = Conjuctive constraints

- A zone Z is a conjunctive formula:

  g_1 & g_2 & ... & g_n

  where g_i may be $x_i \sim b_i$ or $x_i - x_j \sim b_{ij}$
- Use a zero-clock x_0 (constant 0), we have

  $\{x_i - x_j \sim b_{ij} \mid \sim \text{is} < \text{or} \leq, i,j \leq n\}$
- This can be represented as a MATRIX, DBM

  (Difference Bound Matrices)

Solution set as semantics

- Let Z be a zone (a set of constraints)
- Let [Z]={u | u is a solution of Z}

(We shall simply write Z instead [Z])

Operations on Zones

- Post-condition (Delay): $Z\uparrow$
  - $[Z\uparrow] = \{u+d \mid d \in R, u \in [Z]\}$
- Pre-condition: $Z\downarrow$ (the dual of $Z\uparrow$)
  - $[Z\downarrow] = \{u \mid u+d \in [Z] \text{ for some } d \in R\}$
- Reset: $\{x\}Z$ or $Z(x:=0)$
  - $[\{x\}Z] = \{u[0/x] \mid u \in [Z]\}$
- Conjunction
  - $[Z\&g] = [Z] \cap [g]$

Two more operations on Zones

- Inclusion checking: $Z_1 \subseteq Z_2$
  - solution sets
- Emptiness checking: $Z = \emptyset$
  - no solution

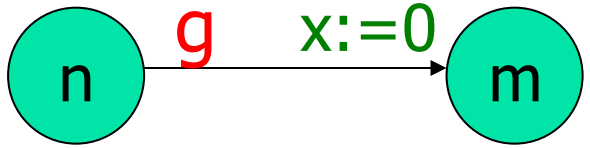
Theorem on Zones

The set of zones is closed under all zone operations

- That is, the result of the operations on a zone is a zone
- Thus, there will be a zone to represent the sets: $[Z^{\uparrow}]$, $[Z^{\downarrow}]$, $[\{x\}Z]$

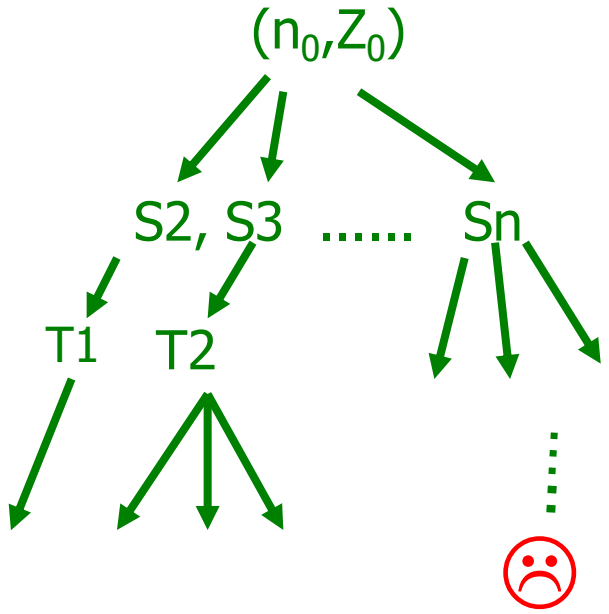
One-step reachability: Si ⇝ Sj

- Delay: (n,Z) → (n,Z′) where Z′= Z↑ ∧ inv(n)
- Action: (n,Z) → (m,Z′) where Z′= {x}(Z ∧**g**)

if 

- **Reach**: (n,Z) ⇝ (m,Z′) if (n,Z) →→(m,Z′)
- **Successors**(n,Z)={(m,Z′) | (n,Z) ⇝ (m,Z′), Z′≠Ø}

Now, we have a search problem



EF ☹