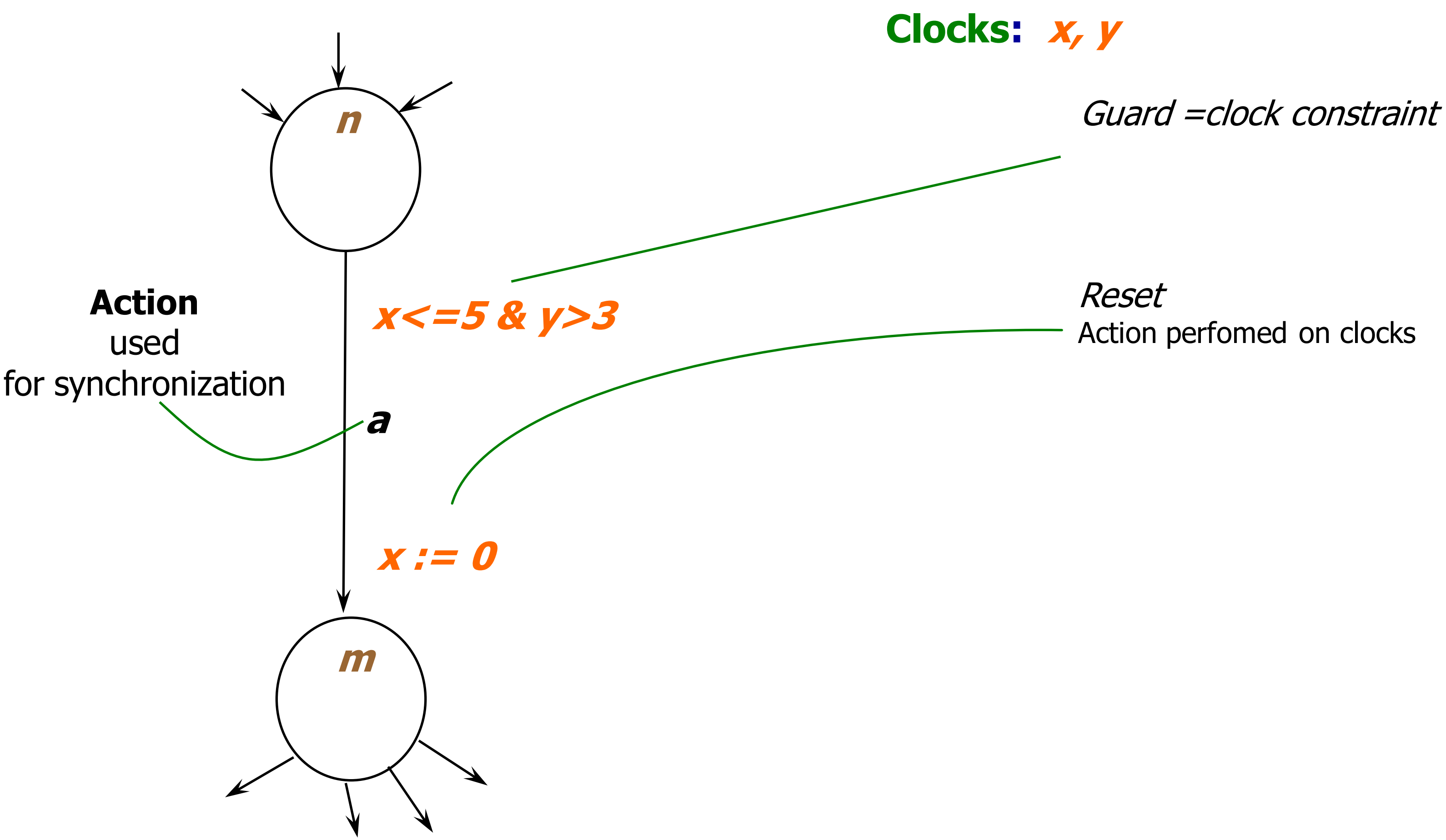
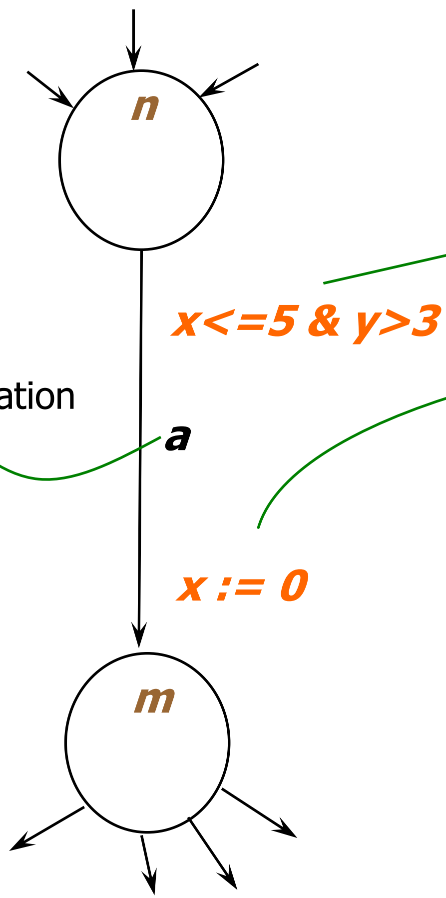
Timed Automata, TCTL & Verification Problems

Timed Automata: Syntax



Timed Automata: Semantics

Clocks: *x, y*



Action used for synchronization

Guard =clock constraint

Reset
Action perfomed on clocks

State

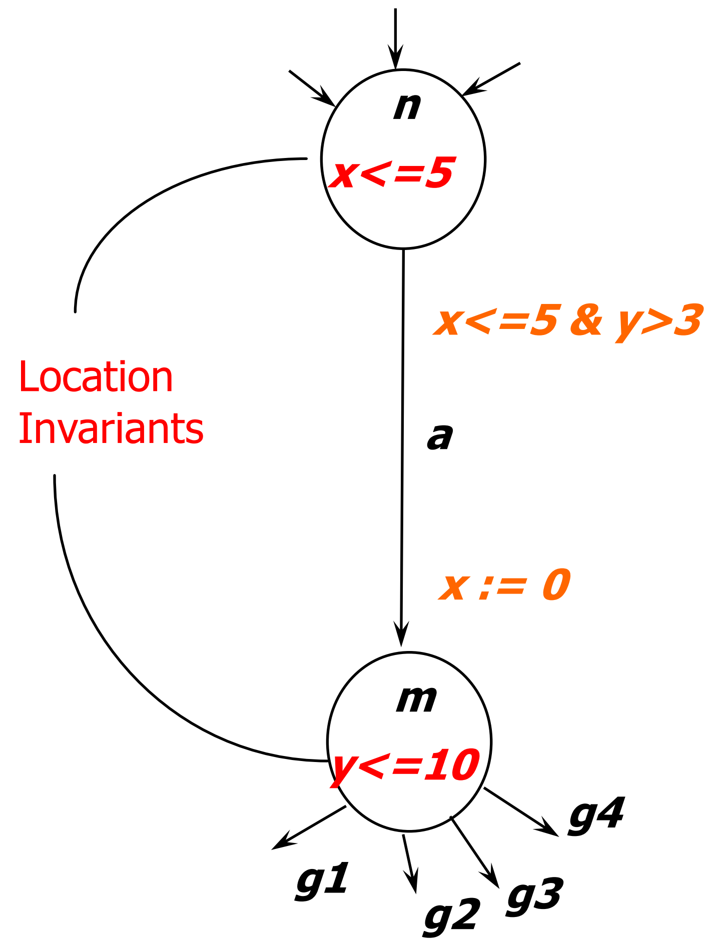
(***location*** , ***x***=v , ***y***=u)   where v,u are in **R**

Transitions

(***n*** , ***x***=2.4 , ***y***=3.1415) $\xrightarrow{a}$ (***m*** , ***x***=0 , ***y***=3.1415)

(***n*** , ***x***=2.4 , ***y***=3.1415) $\xrightarrow{1.1}$ (***n*** , ***x***=3.5 , ***y***=4.2415)

Timed Automata with *Invariants*



Clocks: ***x, y***

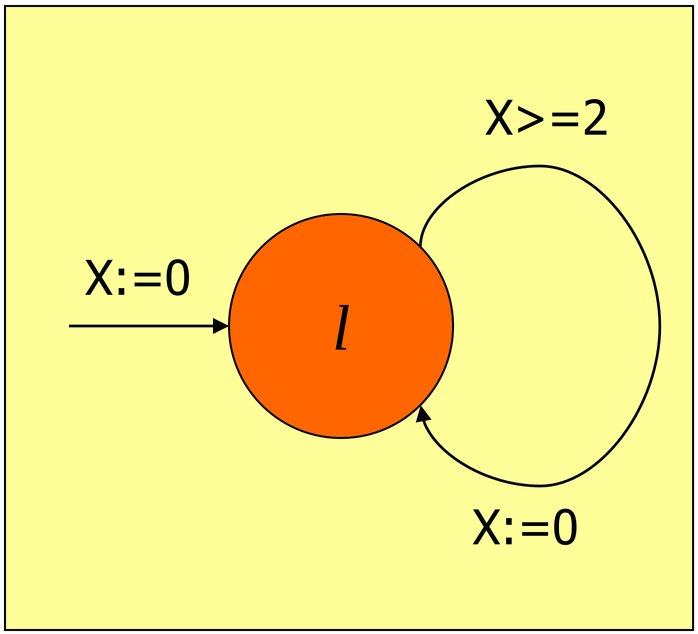
Transitions

(***n*** , ***x***=2.4 , ***y***=3.1415) —~~3.2~~→

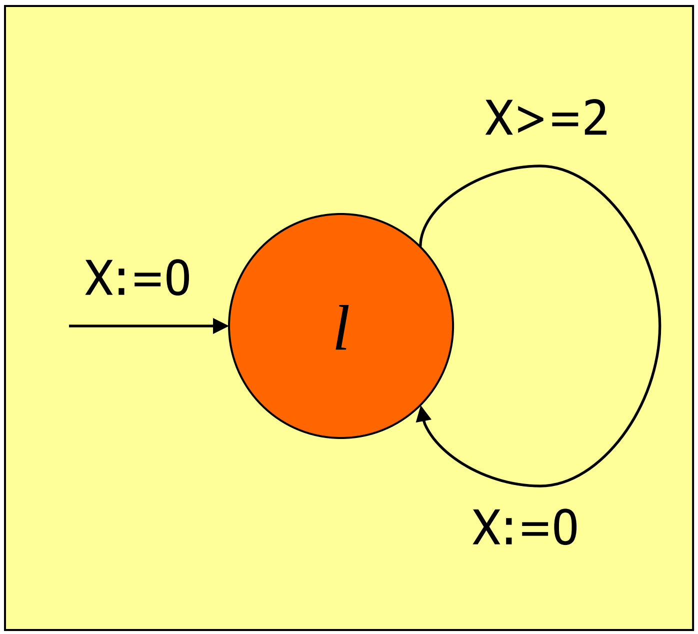
(***n*** , ***x***=2.4 , ***y***=3.1415) —*1.1*→ (***n*** , ***x***=3.5 , ***y***=4.2415)

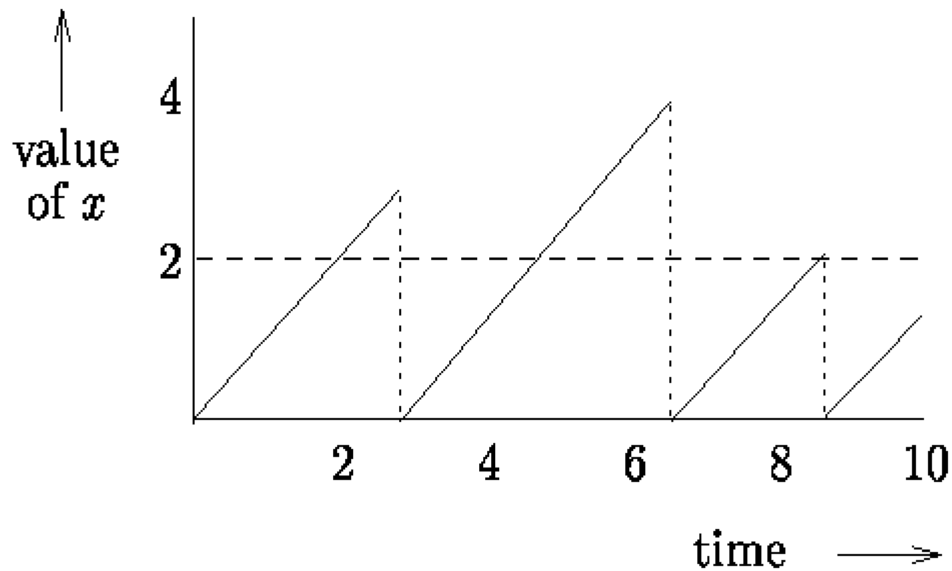
Invariants insure progress!!

Timed Automata: Example

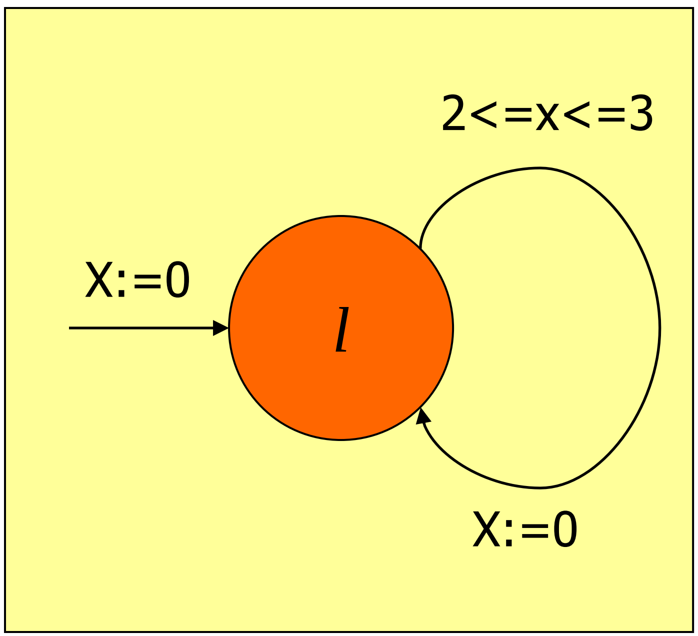


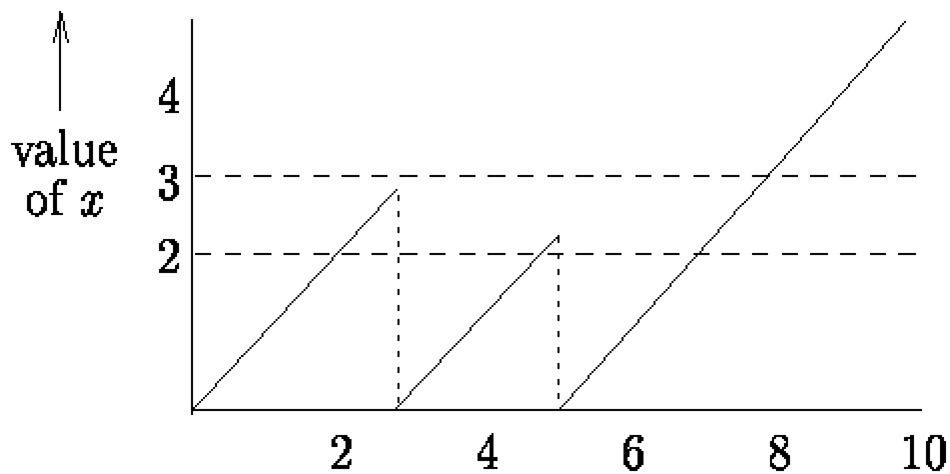
Timed Automata: Example



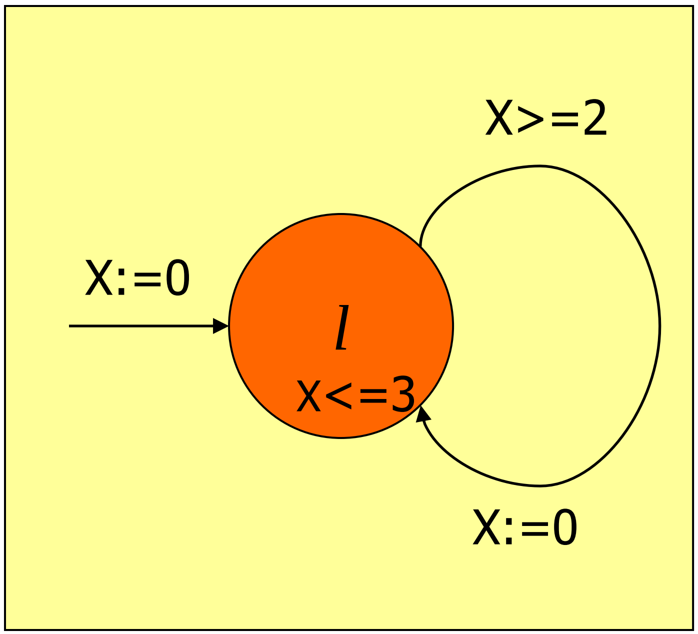


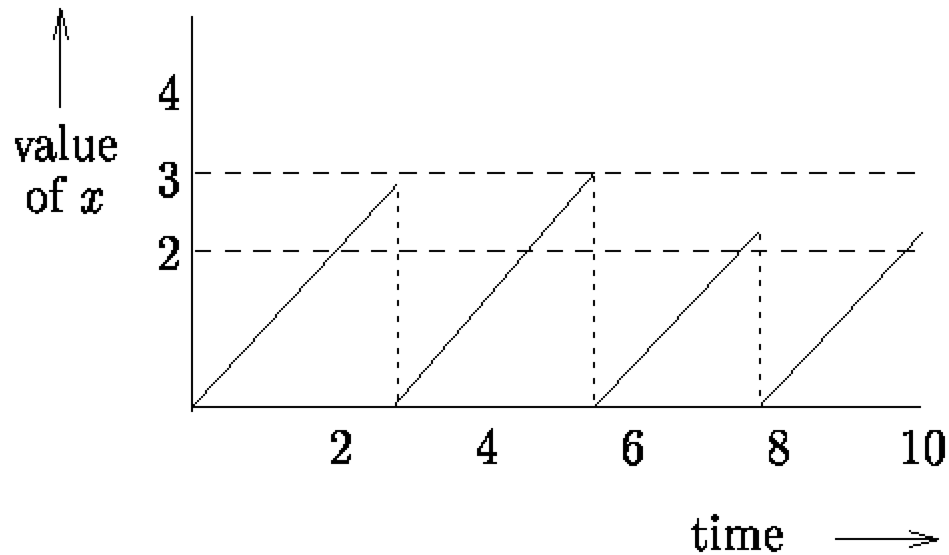
Timed Automata: Example





Timed Automata: Example





Timed Automata

=

Finite Automata + Clock Constraints + Clock resets

Clock Constraints

$$g ::= x \sim n \mid g \,\&\, g$$

where

- x is a clock variable
- $\sim \in \{<, >, \leq, \geq\}$
- n is a natural number

Semantics (definition)

- ***clock valuations***: $V(C) \quad v : C \rightarrow R_{\geq 0}$
- ***state***: $(l, v) \quad where \quad l \in L \;\; and \;\; v \in V(C)$

- ***action transition*** $(l,v) \xrightarrow{a} (l',v') \;\; iff$ 

  $g(v) \;\; and \;\; v' = v[r] \;\; and \;\; Inv(l')(v')$

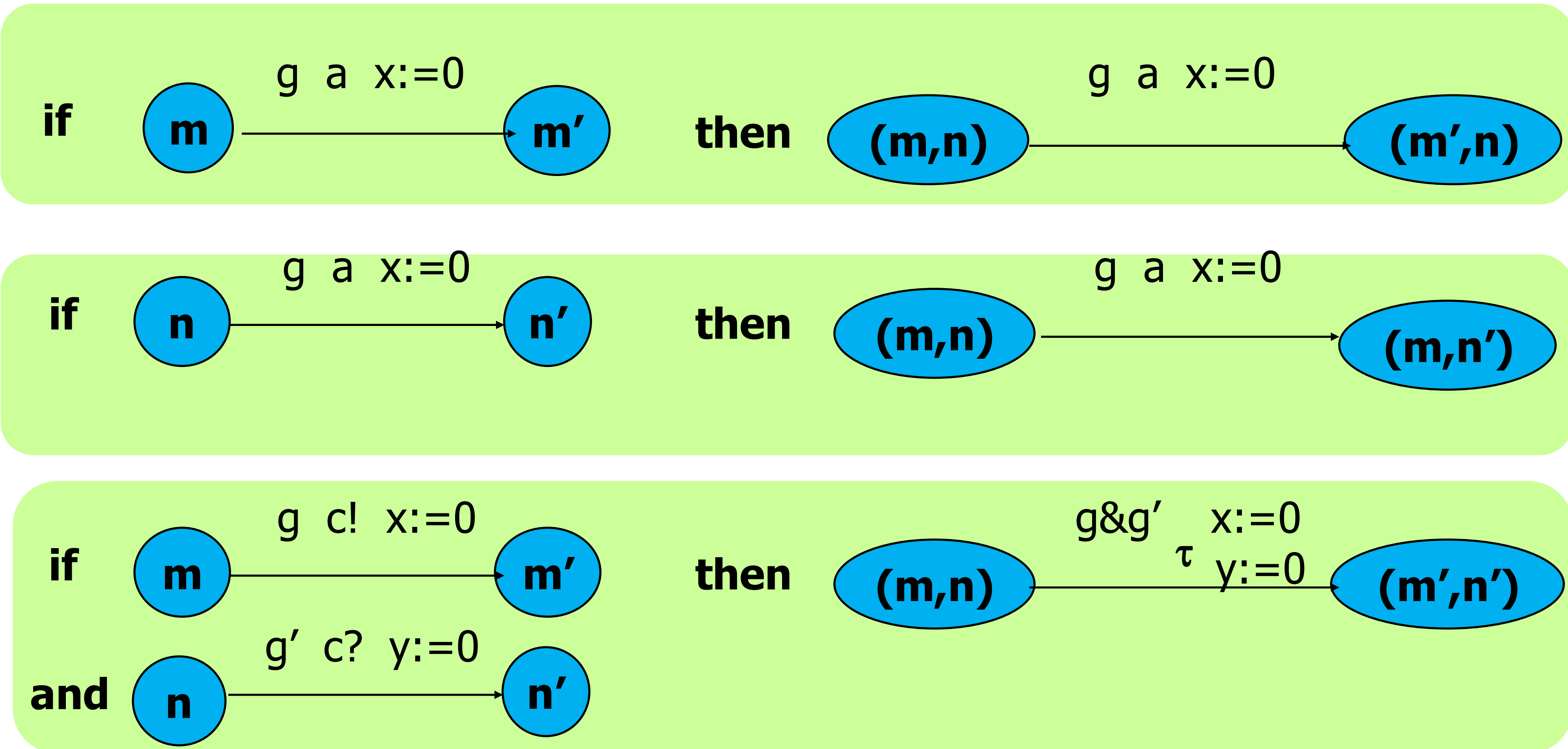
- ***delay Transition*** $(l,v) \xrightarrow{d} (l, v+d) \;\; iff$

  $Inv(l)(v+d') \;\; whenever \;\; d' \leq d \in R_{\geq 0}$

Modeling Concurrency

- Products of automata
- CCS Parallel composition
  - implemented in UPPAAL

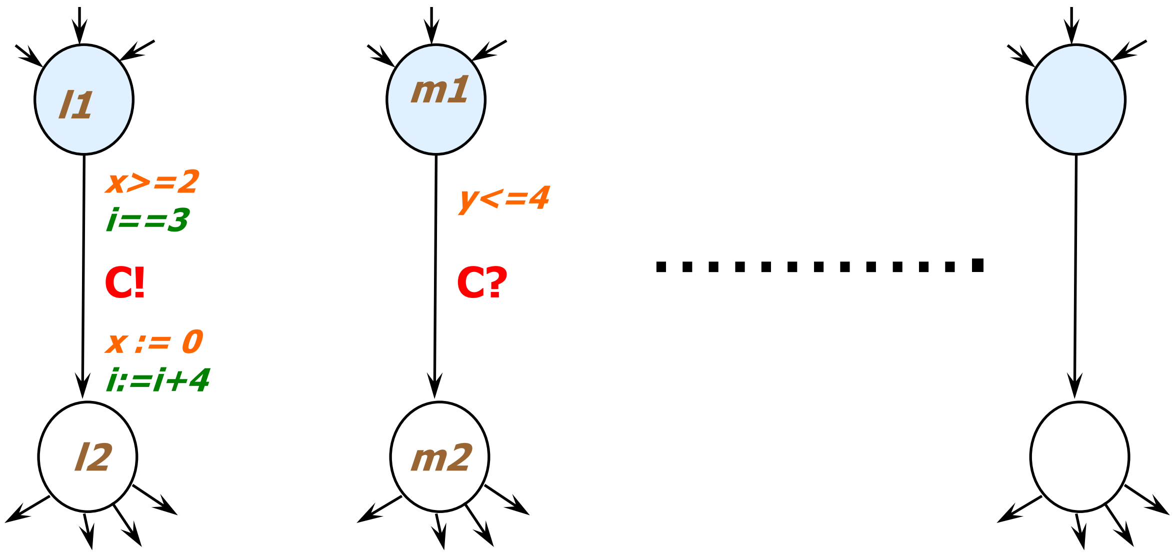
CCS Parallel Composition (implemented in UPPAAL)



where a is an action c! or c? or τ, and c is a channel name

The UPPAAL Model

= Networks of Timed Automata + Integer Variables +….



Two-way synchronization on *complementary* actions.

Closed Systems!

Example transitions

*(**l1**, **m1**,………, x=2, y=3.5, i=3,…..)* $\xrightarrow{\tau}$ *(**l2**,**m2**,……..,x=0, y=3.5, i=7,…..)*

Verification Problems

Location Reachability (def.)

n is reachable from m if there is a sequence of transitions:

$$(m, u) \longrightarrow^{*} (n, v)$$

(Timed) Language Inclusion, $L(A) \subseteq L(B)$

$$(a_0, t_0)\ (a_1, t_1)\ \ldots\ldots\ (a_n, t_n) \in L(A)$$

If

"A can perform a_0 at t_0, a_1 at t_1 … … a_n at t_n"

$$(l_0, u_0) \xrightarrow{t_0} (l_0, u_0+t_0) \xrightarrow{a_0} (l_1, u_1) \ldots\ldots$$

Verification Problems

- Timed Language Equivalence & Inclusion ☹
  - 1-clock, finite traces, decidable [Ouaknine & Worrell 04]
  - 1-clock, infinite traces & Buchi-conditions, undecidable [Abdulla et al 05]
- Universality ☹
- Untimed Language Inclusion ☺
- (Un)Timed (Bi)simulation ☺
- Reachability Analysis/Emptiness ☺
- Optimal Reachability (synthesis problem) ☺
  - If a location is reachable, what is the minimal delay before reaching the location?

Timed CTL = CTL + clock constraints

Note that the semantics of TA defines a transition system where each state has a Computation Tree

Computation Tree Logic, CTL

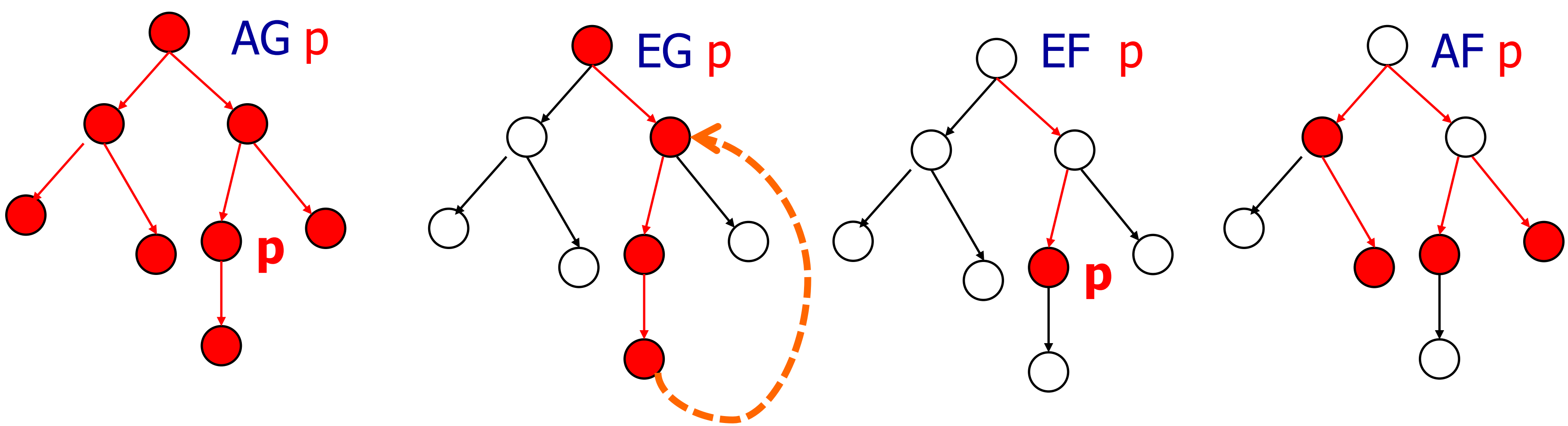
Clarke & Emerson 1980

Syntax

ϕ **::=** **P** | $\neg\,\phi$ | $\phi \vee \phi$ | **EX** ϕ | **E[**ϕ **U** ϕ**]** | **A[**ϕ **U** ϕ**]**

where **P** $\in$ AP (atomic propositions)

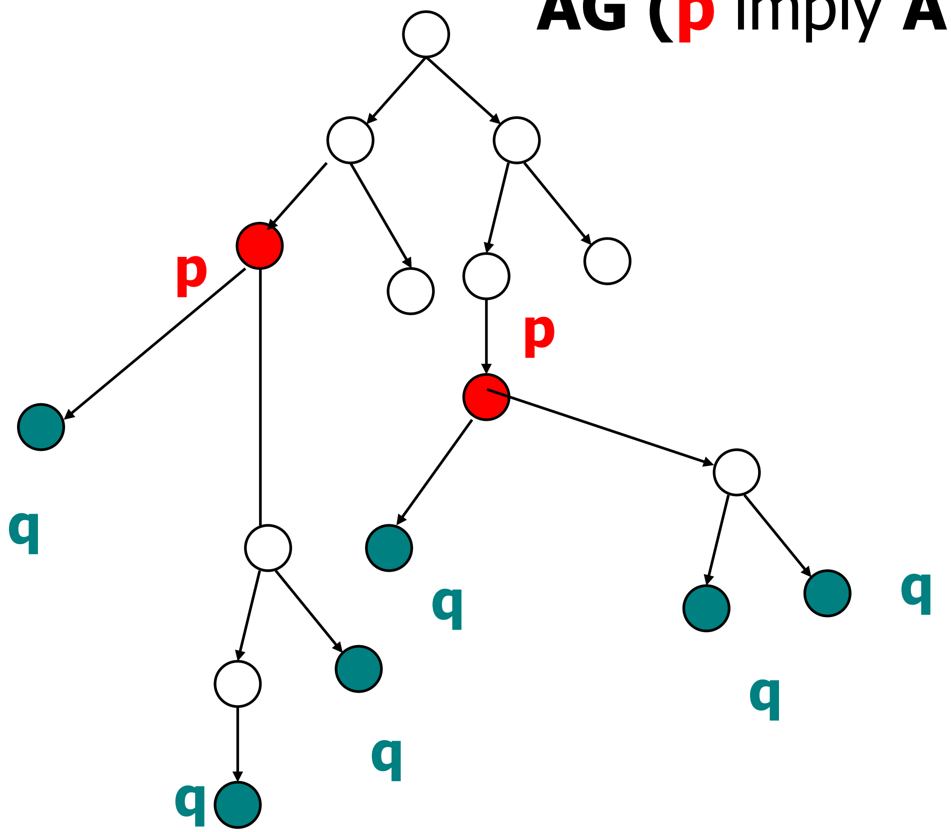
Derived Operators



Liveness: p - -> q

"p leads to q"

AG (p imply **AF q)**



Timed CTL (a simplified version)

Syntax

$$\phi ::= p \mid \neg \phi \mid \phi \vee \phi \mid EX\ \phi \mid E[\phi\ U\ \phi] \mid A[\phi\ U\ \phi]$$

where **p** $\in$ AP (atomic propositions) **or Clock constraint**

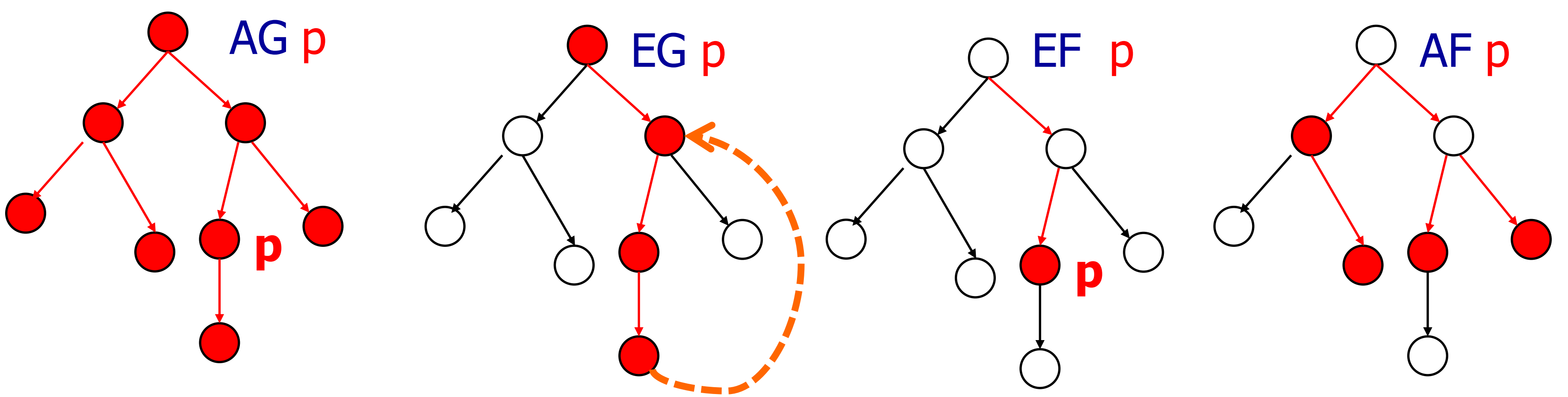
Timed CTL (a simplified version)

Syntax

$$\phi \ ::= \ p \mid \neg \phi \mid \phi \vee \phi \mid EX\ \phi \mid E[\phi\ U\ \phi] \mid A[\phi\ U\ \phi]$$

where **p** $\in$ AP (atomic propositions) **or Clock constraint**

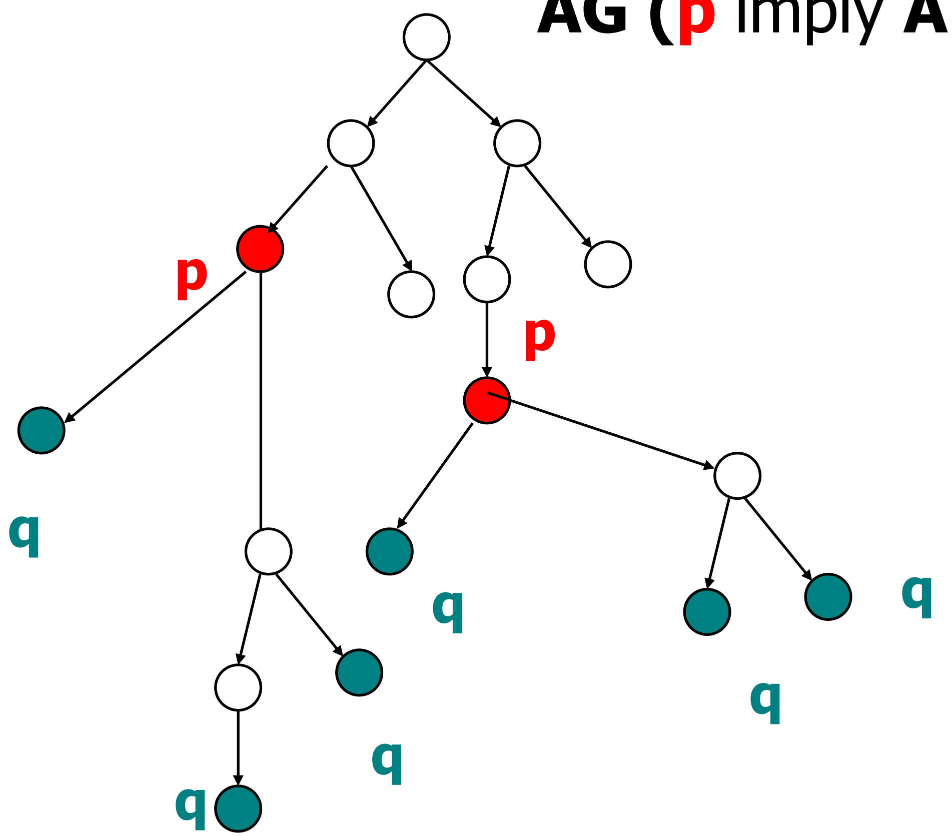
Derived Operators



A[] P in UPPAAL | **E[] P in UPPAAL** | **E<> P in UPPAAL** | **A<> P in UPPAAL**

Derived Operators (cont.)

AG (p imply **AF q)**



p - -> q in **UPPAAL**

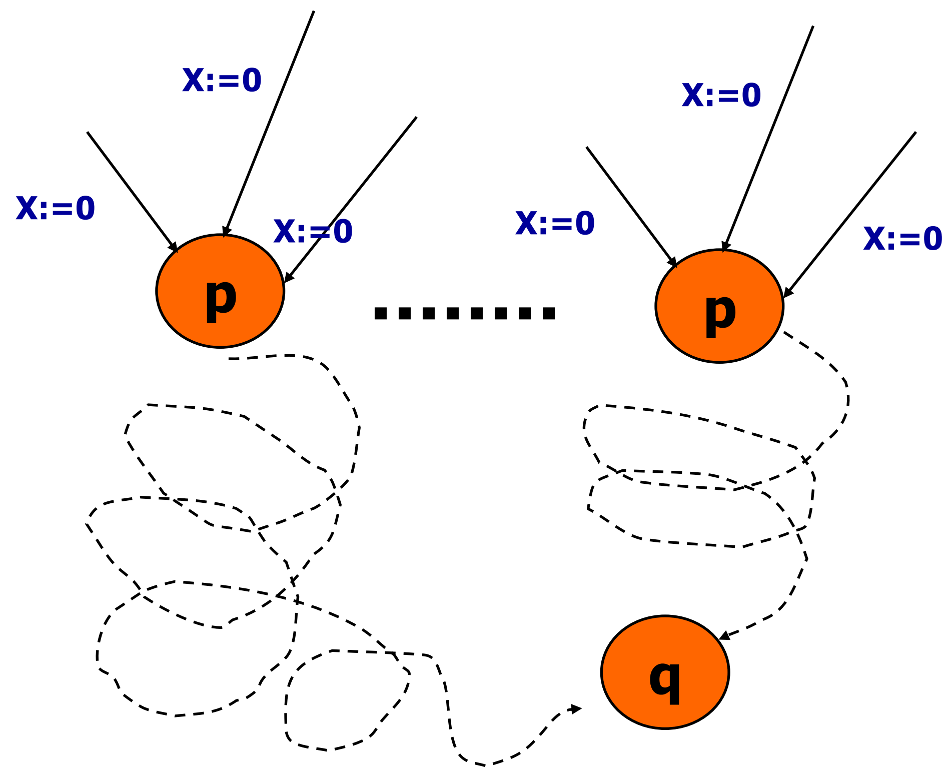
Bounded Liveness

[TACAS 98]

Verify: ”whenver p is true, q should be true within 10 sec

P - - > (q and x<10)

Use extra clock x
Add x:=0 on all edges leading to P



Bounded Liveness/Responsiveness

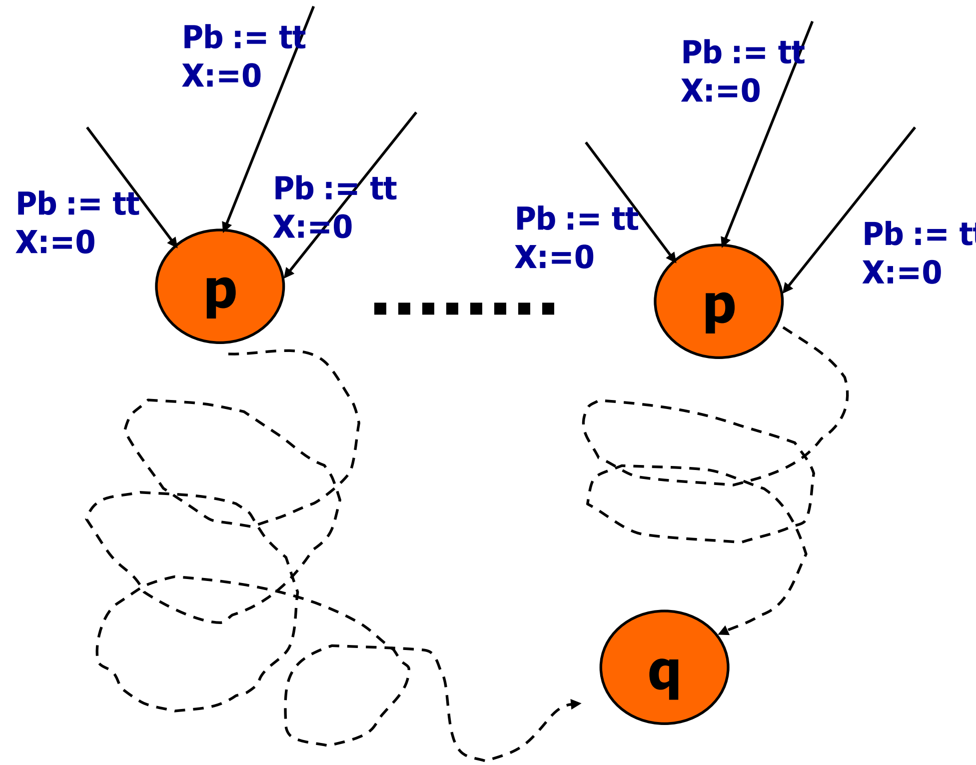
(reachability analysis, more efficient?)

[TACAS 98]

Verify: "whenver p is true,
q should be true within 10 sec

AG ((P_b and $x>10$) imply q)

Use extra clock x and boolean P_b
Add P_b := tt and x:=0 on all edges leading to location P



Bounded Liveness/Responsiveness

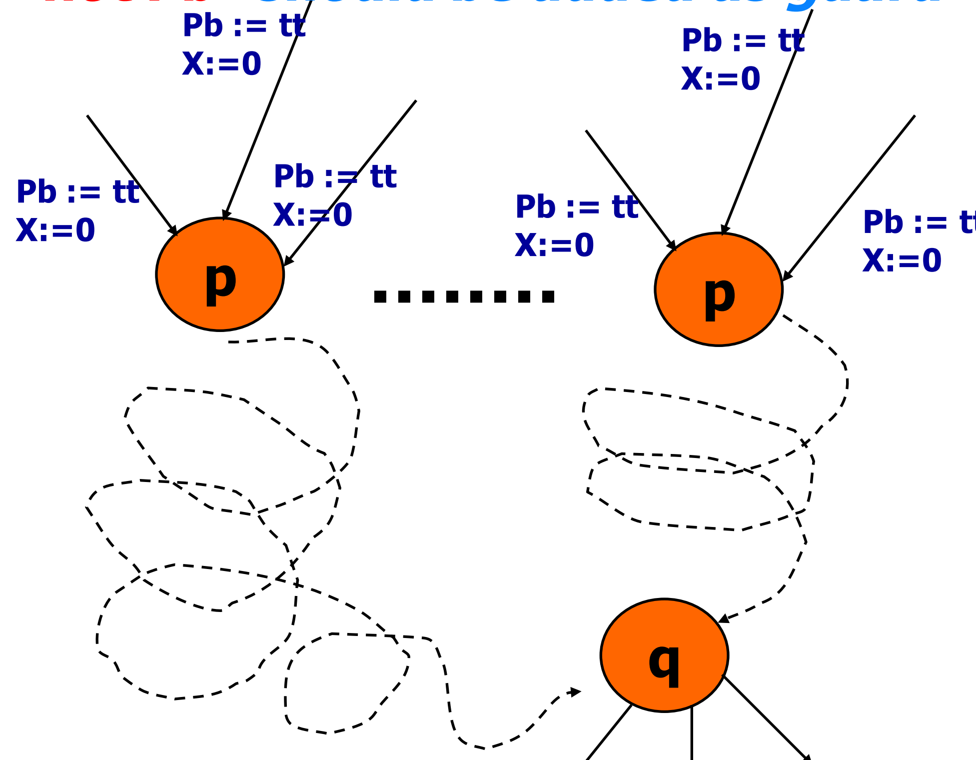
(reachability analysis, more efficient?)

[TACAS 98]

**Verify: "whenver p is true,
q should be true within 10 sec**

AG ((P_b and x>10) imply q)

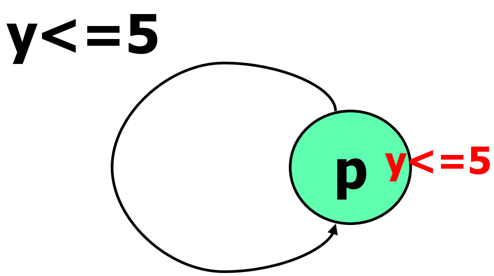
***This is not really correct;
"not Pb" should be added as guard***



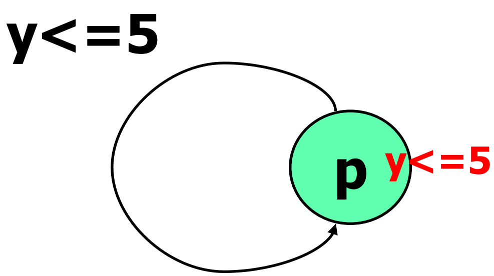
**Use extra clock x and boolean P_b
Add P_b := tt and x:=0 on all edges
leading to location P**

***Pb:=ff should be
On all eadges leaving q***

Problem with Zenoness/Time-stop

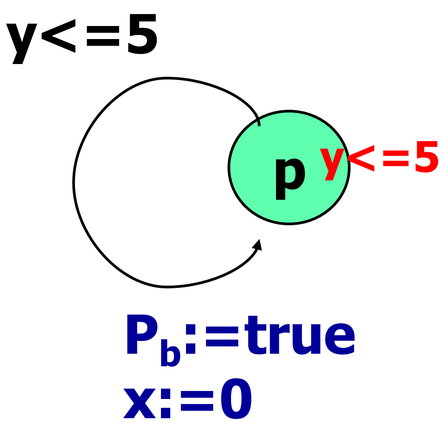


EXAMPLE



We want to specify **"whenever P is true, Q should be true within 10 time units**

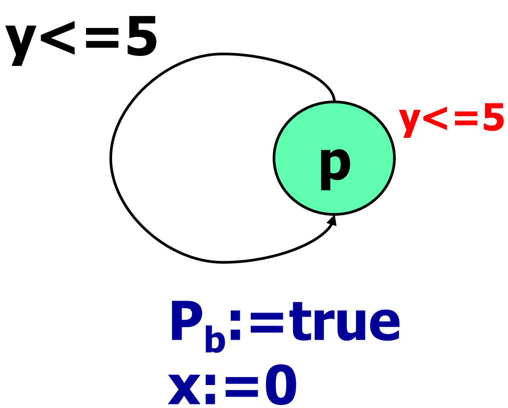
EXAMPLE



We want to specify "whenever P is true, Q should be true within 10 time units

AG ((P$_b$ and x>10) imply Q)

EXAMPLE



We want to specify "whenever P is true, Q should be true within 10 time units

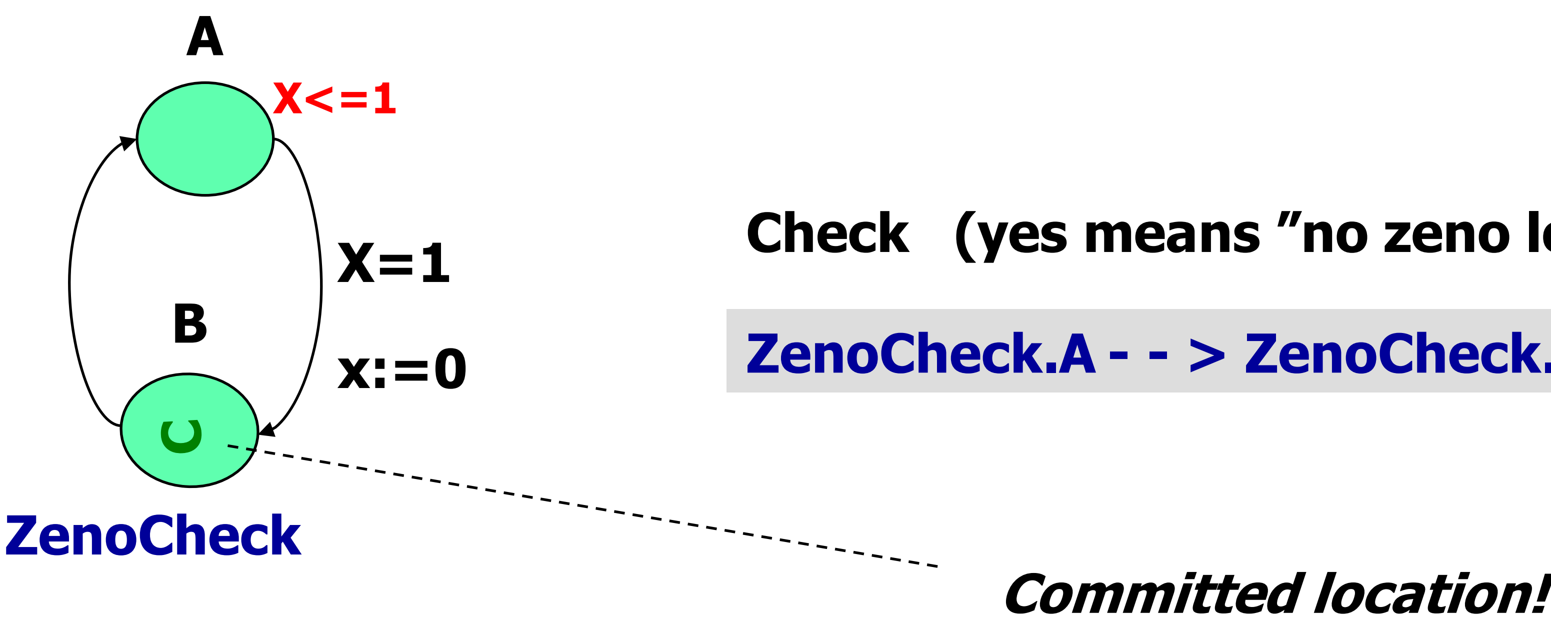
$$AG\ \ ((P_b \text{ and } x>10) \text{ imply } q)$$

is satisfied !!!

Solution with UPPAAL

Check Zeno-freeness by an extra observer

System || ZenoCheck

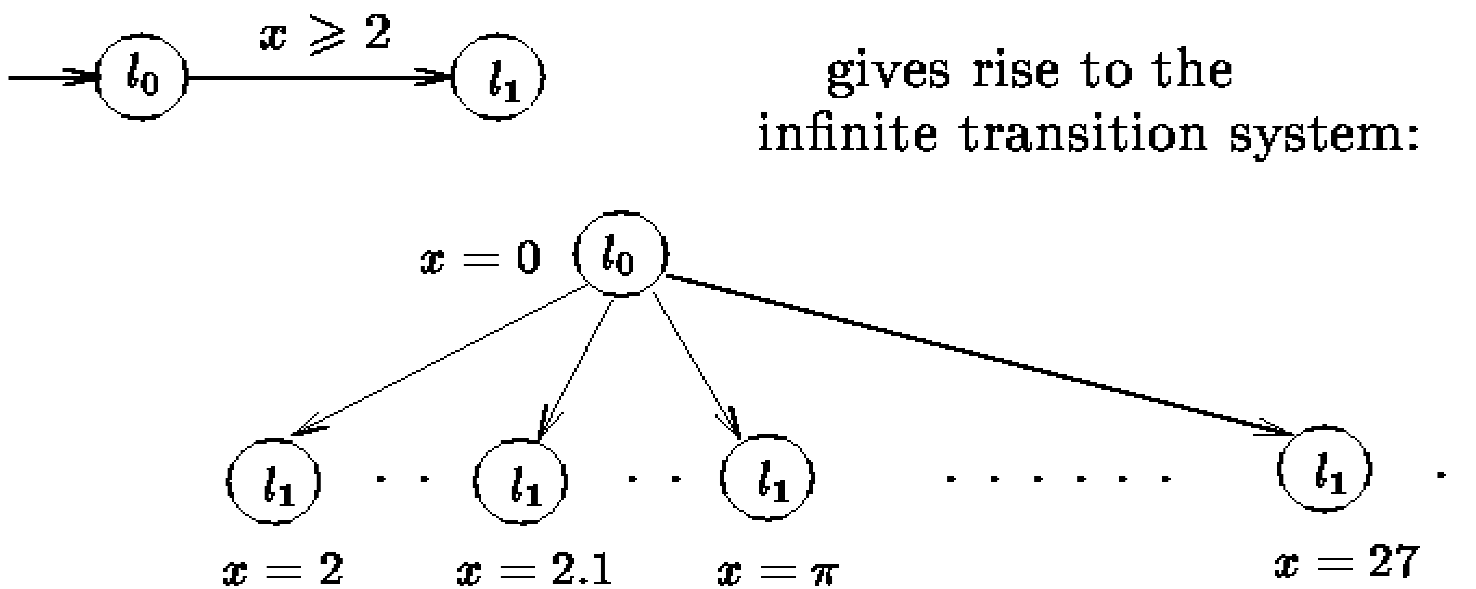


Check (yes means "no zeno loops")

ZenoCheck.A - - > ZenoCheck.B

REACHABILITY ANALYSIS using Regions

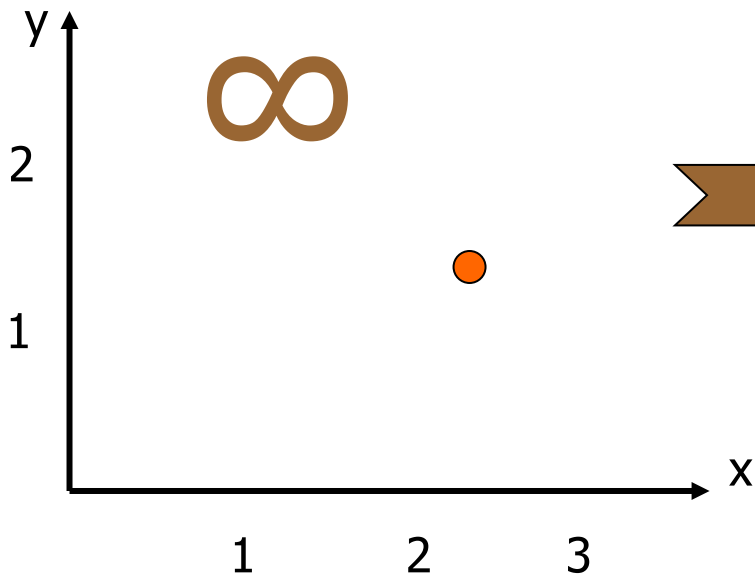
Infinite State Space!

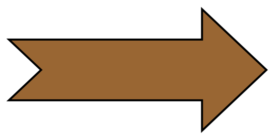


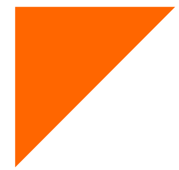
However, the reachability problem is decidable ☺ Alur&Dill 1991

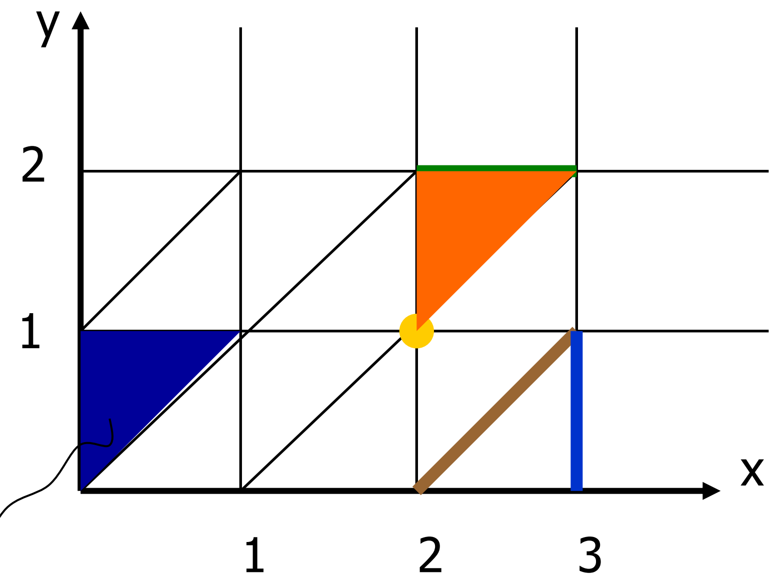
Region: From infinite to finite

Concrete State
(n, x=2.2, y=1.5)



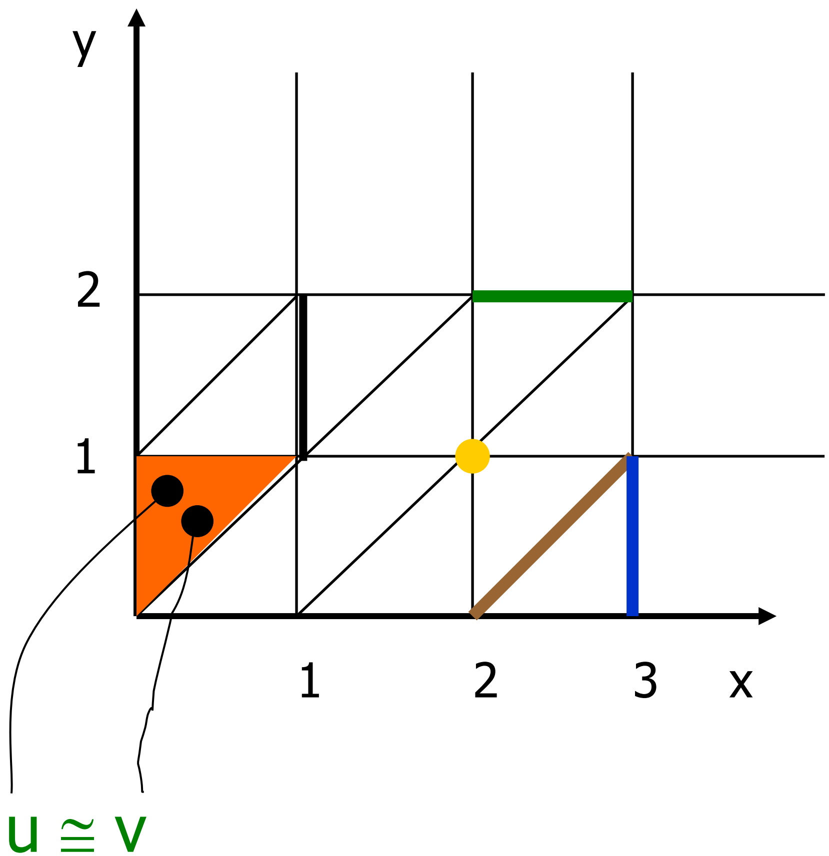


Symbolic state (region)
(n, )



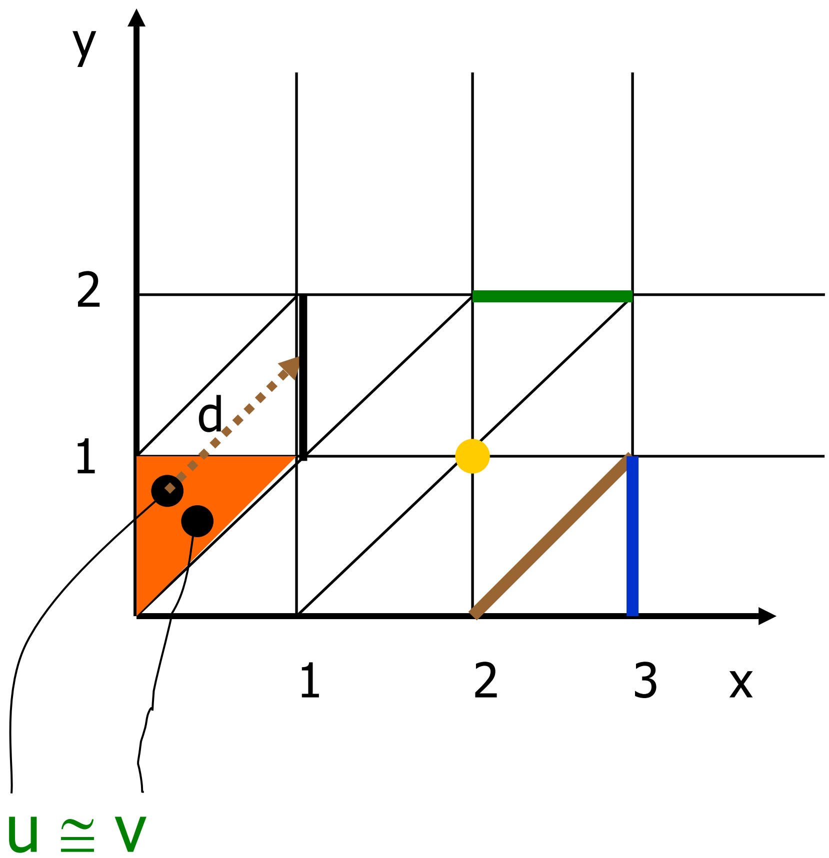
An equivalence class (i.e. a *region*)
There are only *finite* many such!!

Region equivalence (Intuition)



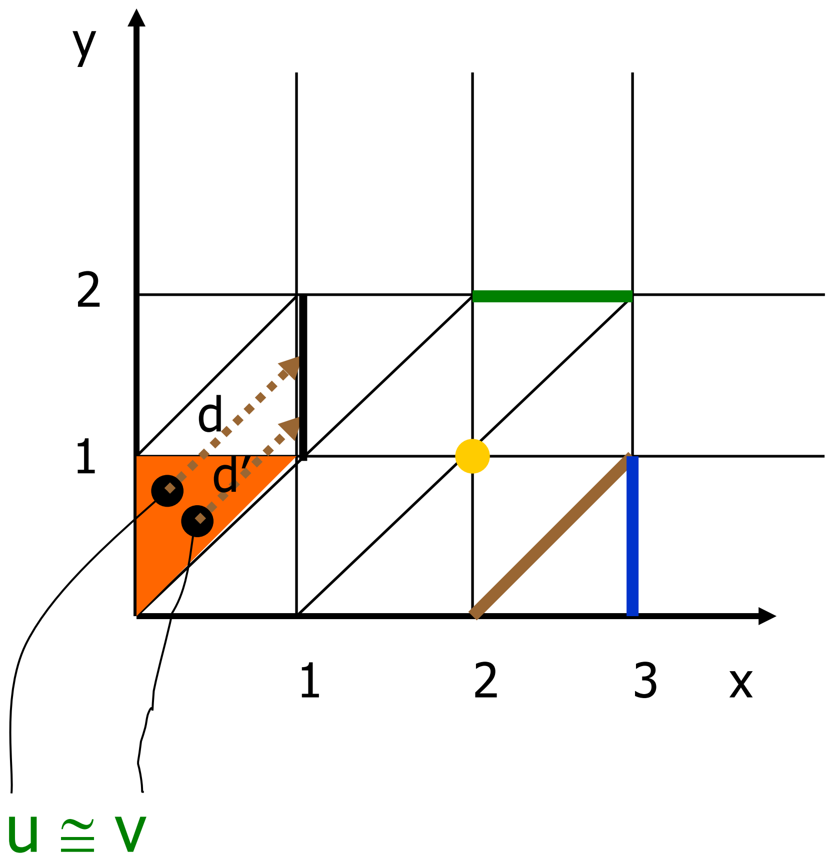
u ≅ v iff (l,u) and (l,v) may reach the same set of eqivalence classes

Region equivalence (Intuition)



u ≅ v iff (l,u) and (l,v) may reach the same set of eqivalence classes

Region equivalence (Intuition)



u ≅ v iff (l,u) and (l,v) may reach the same set of eqivalence classes

Region equivalence *[Alur and Dill 1990]*

- u,v are clock assignments
- u≈v iff
  - For all clocks x,

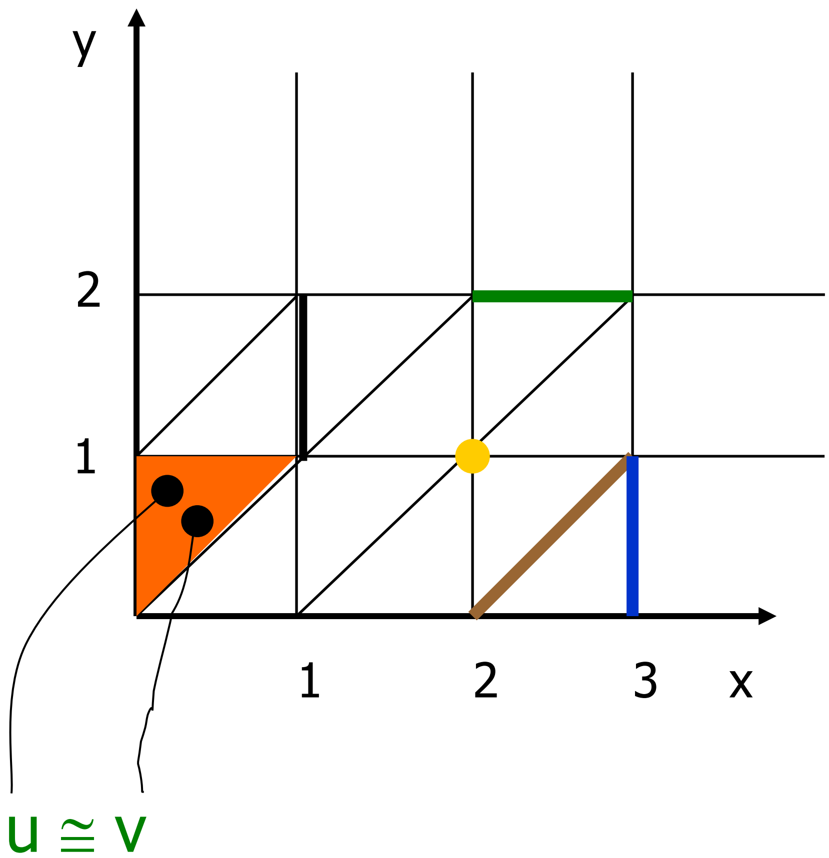
    either (1) u(x)>Cx and v(x)>Cx

    or (2) ⌊u(x)⌋=⌊v(x)⌋
  - For all clocks x, if u(x)<=Cx,

    {u(x)}=0 iff {v(x)}=0
  - For all clocks x, y, if u(x)<=Cx and u(y)<=Cy

    {u(x)}<= {u(y)} iff {v(x)}<= {v(y)}

Region equivalence (alternatively)



u ≅ v iff u and v satisfy exactly the same set of constraints in the form of

$x_i \sim m$ and $x_i - x_j \sim n$

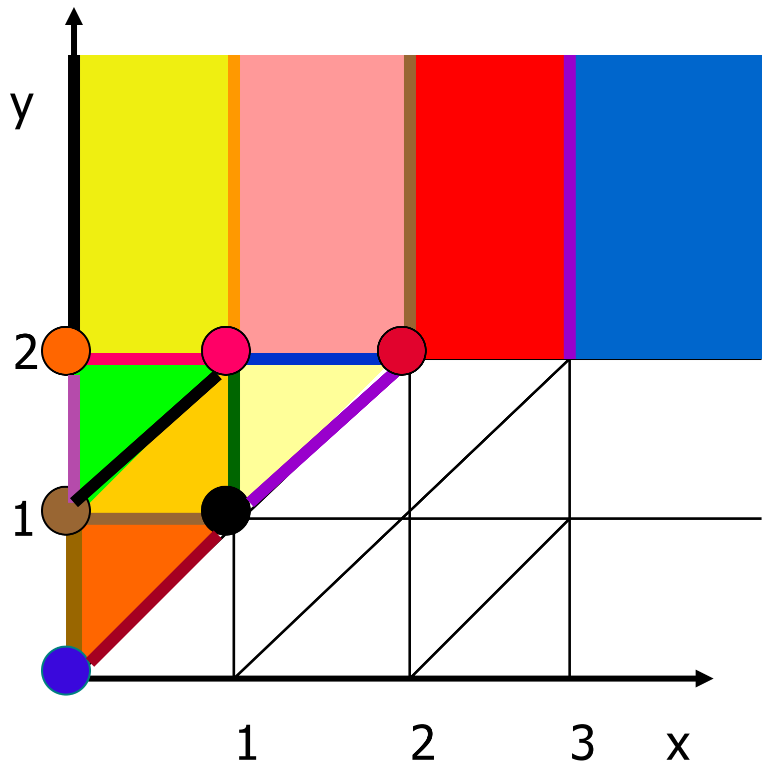
where ~ is in {<,>,≤,≥}

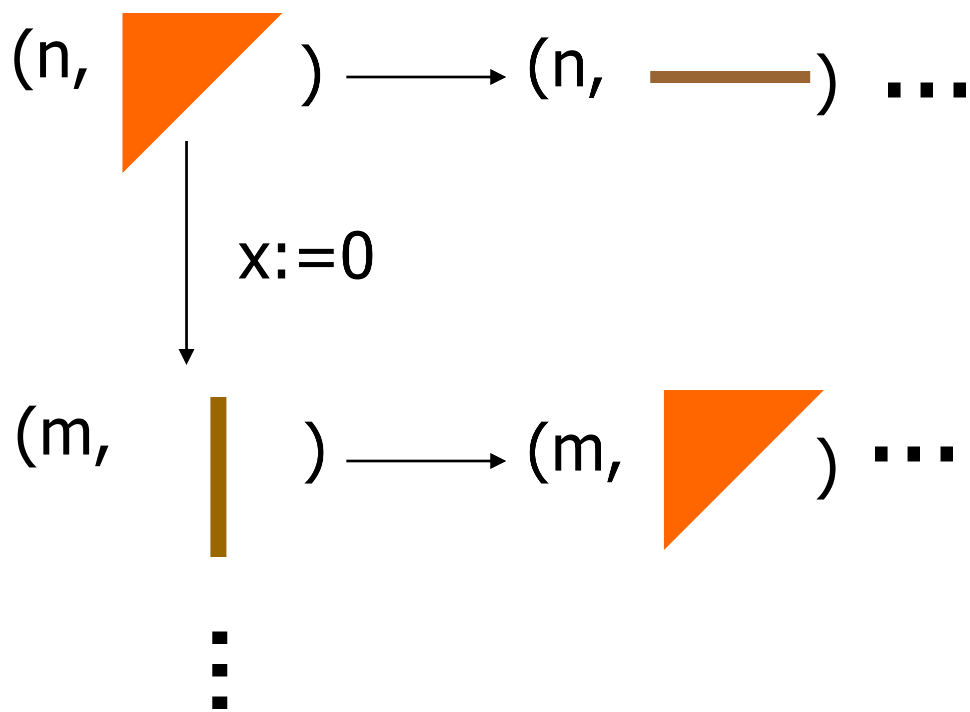
and m,n < MAX

This is not quite correct;
we need to consider the MAX more carefully

Region Graph

Finite-State Transition System!!





OBS: there are only
Finite many regions

(m, [u]) ⟶ (n, [v]) if (m, u) ⟶ (n,v)

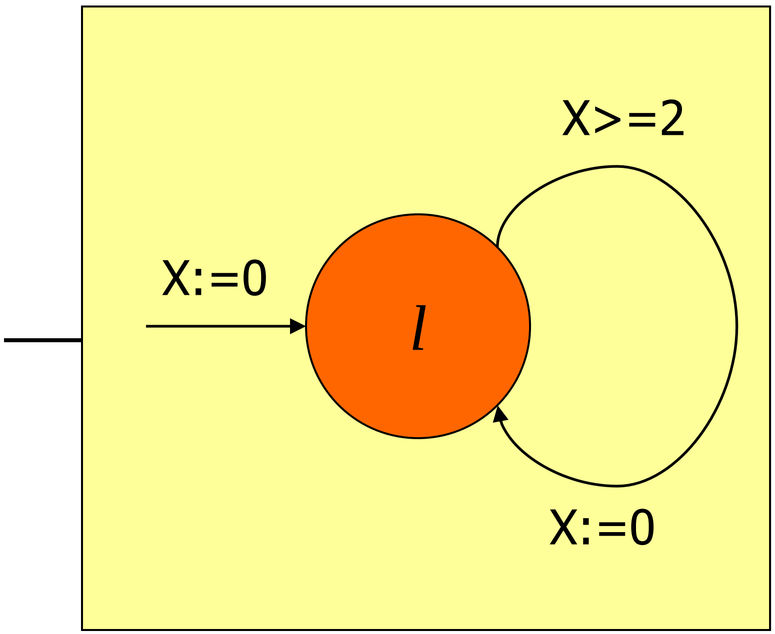
Theorem

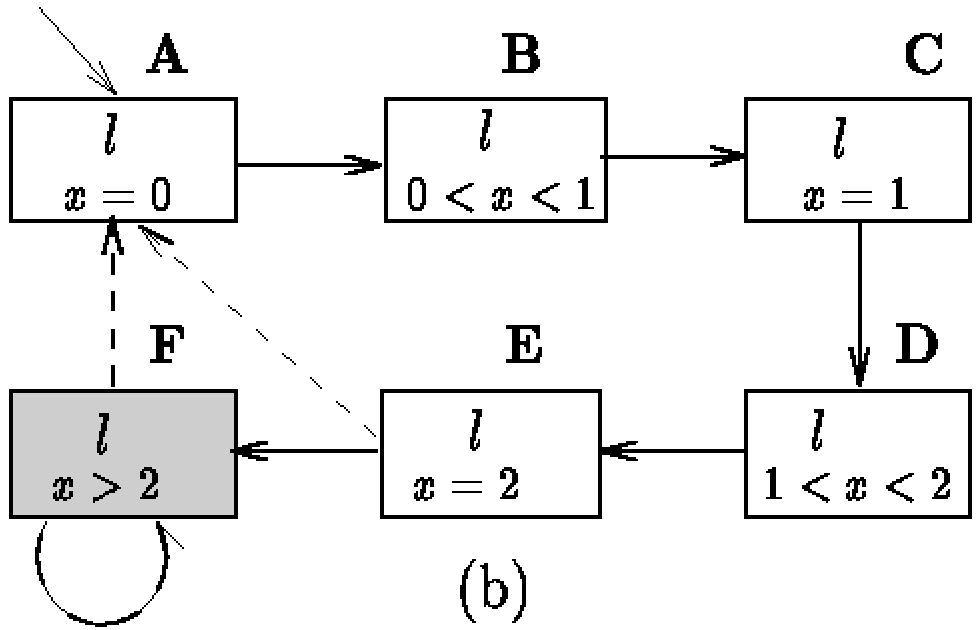
u≈v implies

- u(x:=0) ≈ v(x:=0)
- u+n ≈ v+n for all natural number n
- for all d<1: u+d ≈ v+d’ for some d’<1

”Region equivalence’ is preserved by ”addition” and reset.
(also preserved by ”subtraction” if clock values are ”bounded”)

Region graph of a simple timed automata

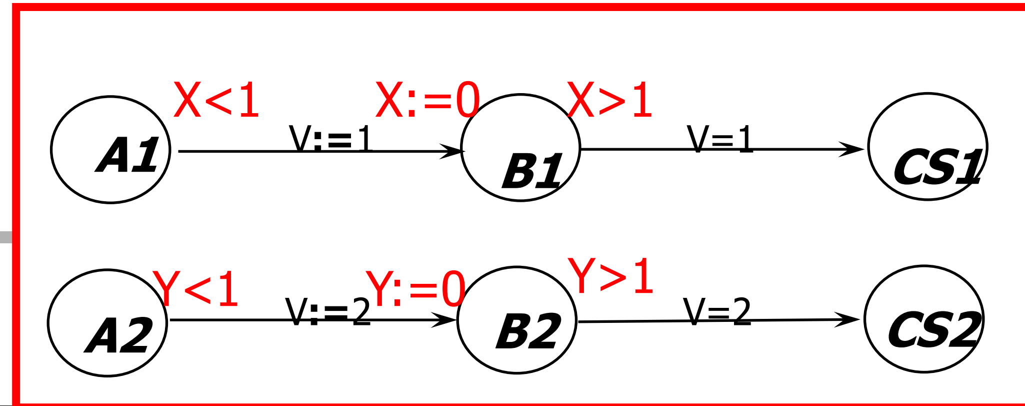




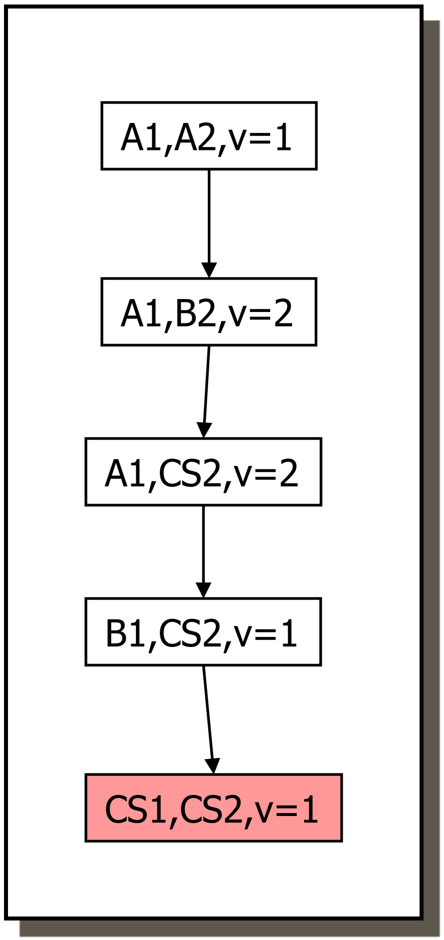
(b)

Fischers again

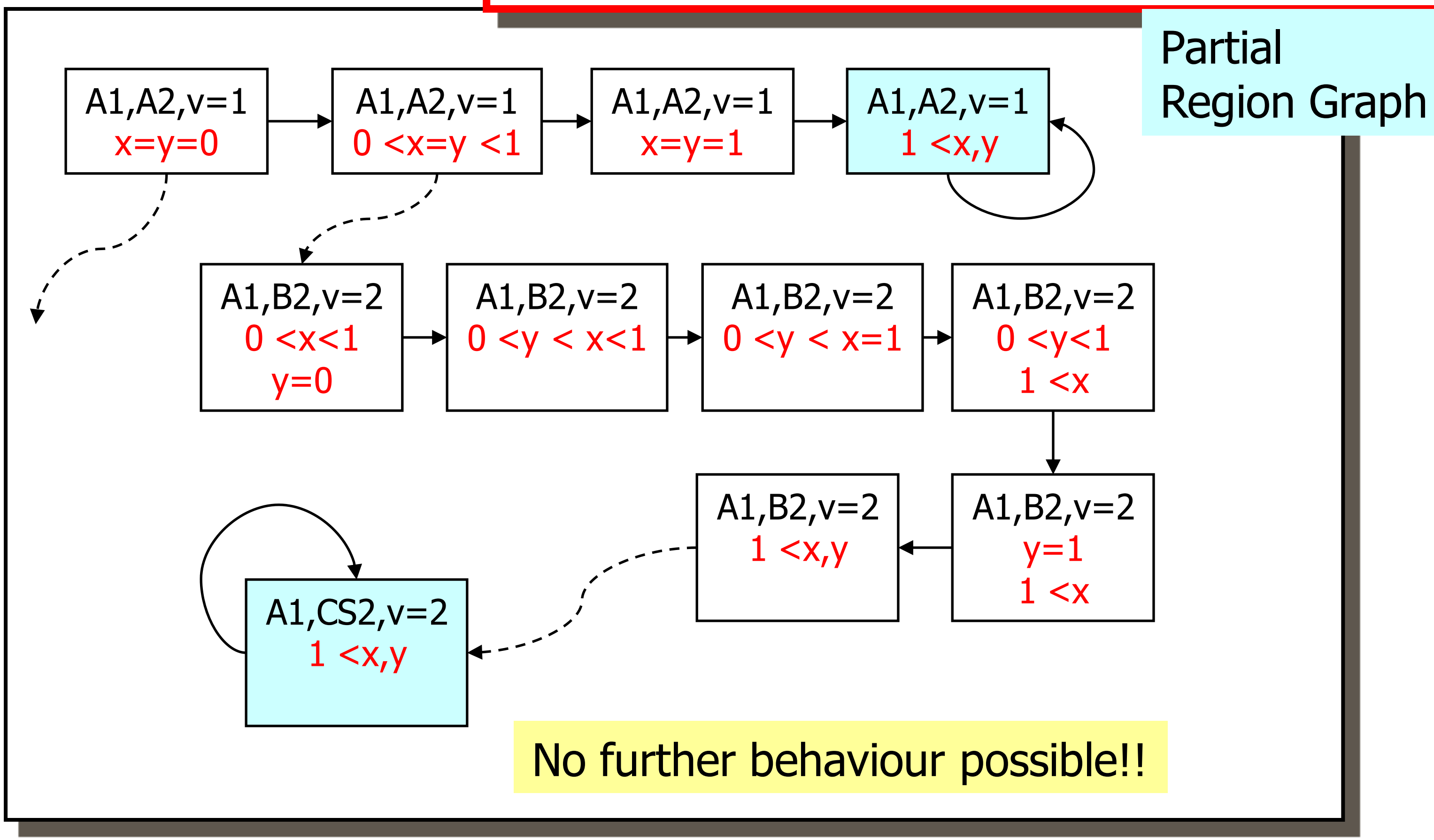
$$AG(\neg(CS_1 \wedge CS_2))$$



Untimed case



Timed case



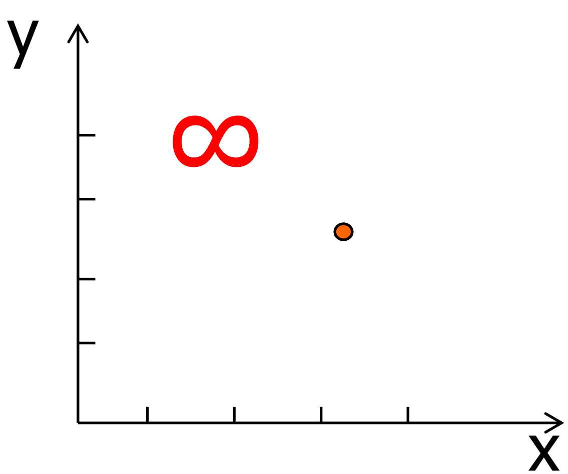
Problems with Region Construction

- Too many 'regions'
  - Sensitive to the maximal constants
  - e.g. x>1,000,000, y>1,000,000 as guards in TA
- The number of regions is highly exponential in the number of clocks and the maximal constants.

REACHABILITY ANALYSIS using ZONES

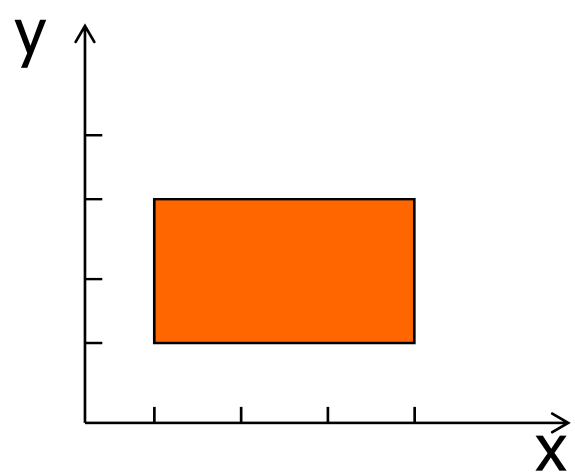
Zones: From infinite to finite

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

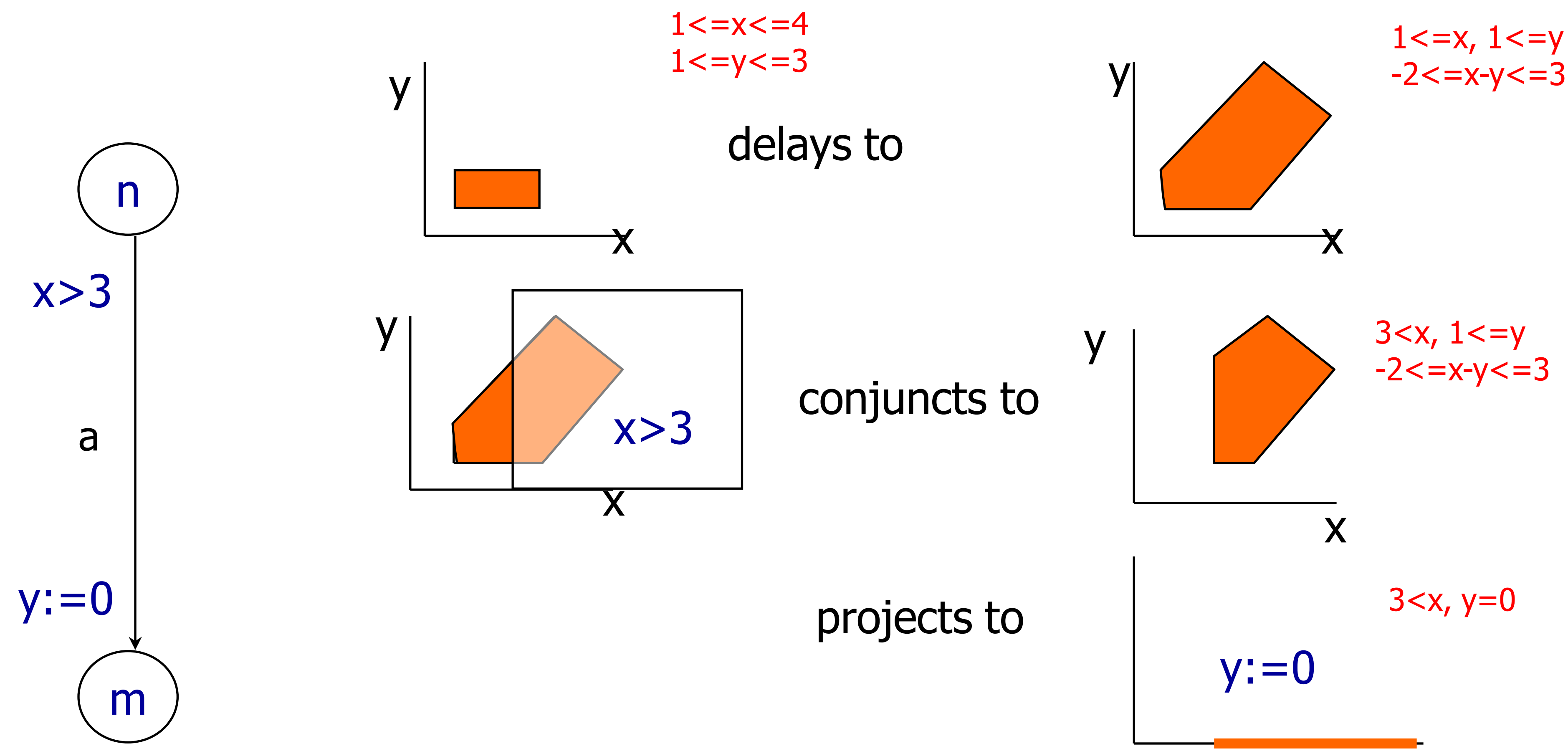


Symbolic state (zone)
(n, $1 \leq x \leq 4, 1 \leq y \leq 3$)

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



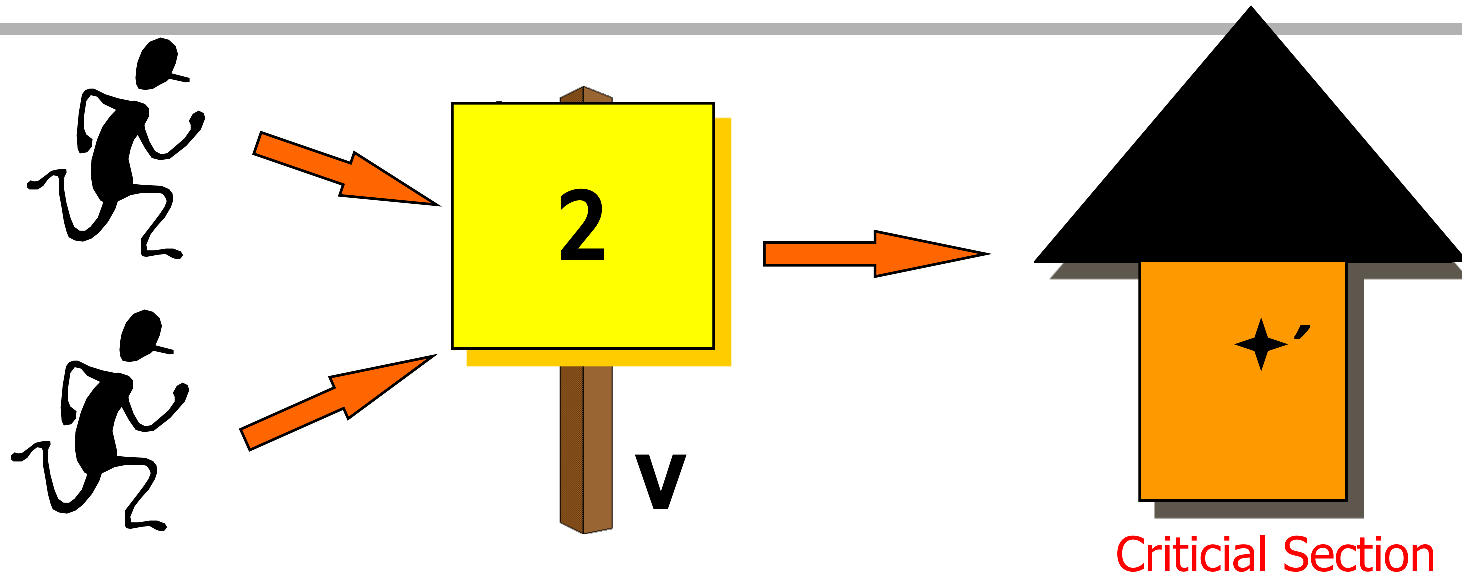
Symbolic Transitions



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

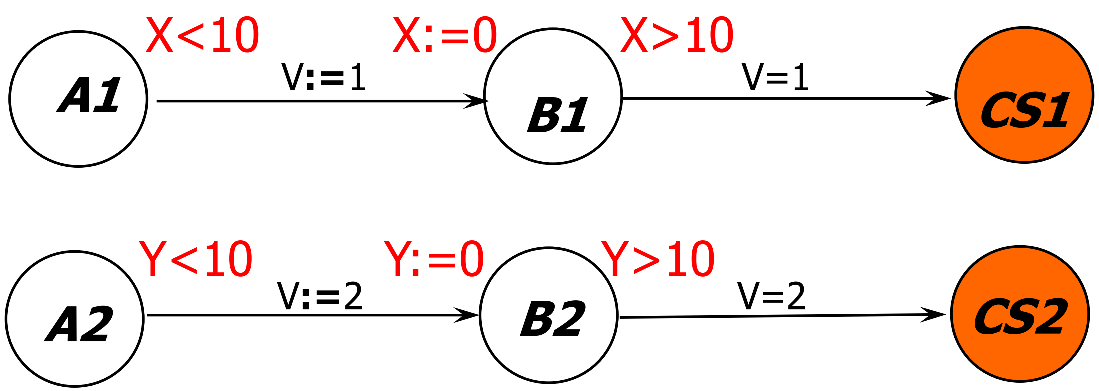
Fischer's Protocol

analysis using zones

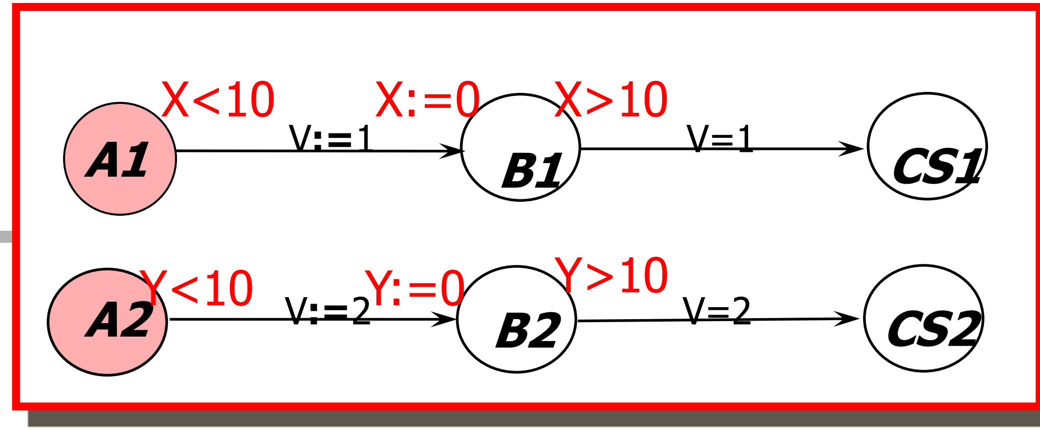


Initially

V=1

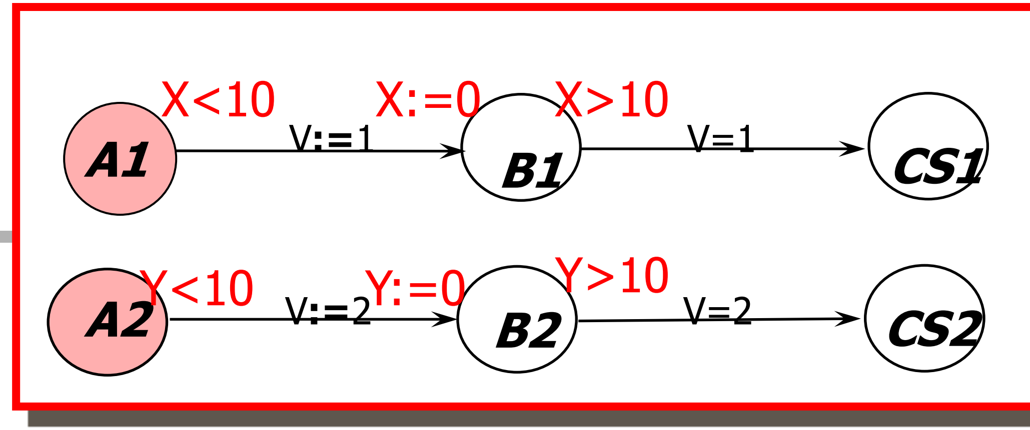


Fischers cont.



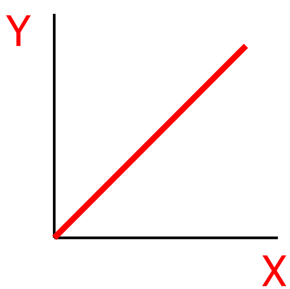
Untimed case

Fischers cont.

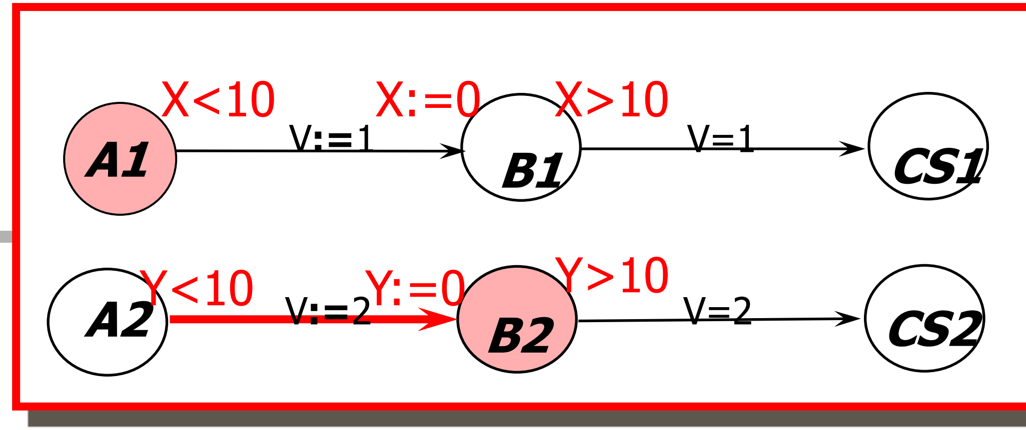


Untimed case

Taking time into account



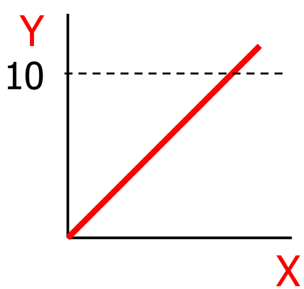
Fischers cont.

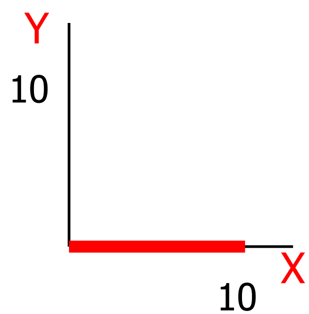


Untimed case

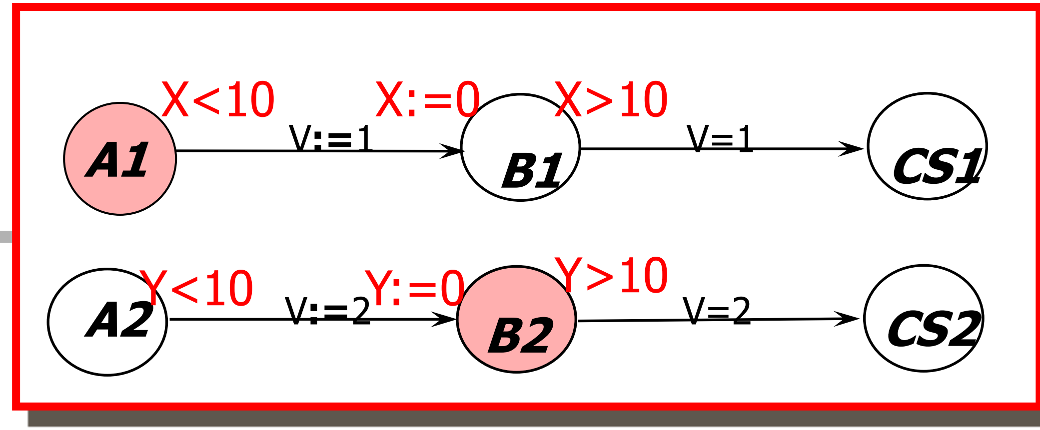
A1,A2,v=1 → A1,B2,v=2 → A1,CS2,v=2 → B1,CS2,v=1 → CS1,CS2,v=1

Taking time into account



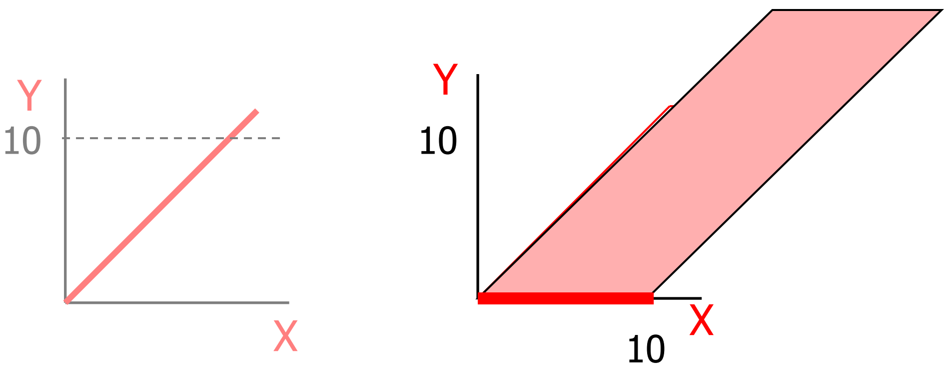


Fischers cont.

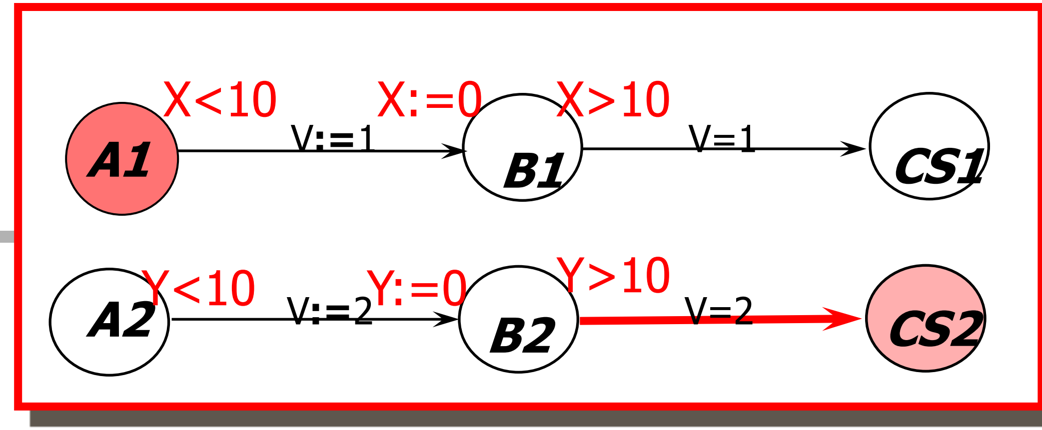


Untimed case

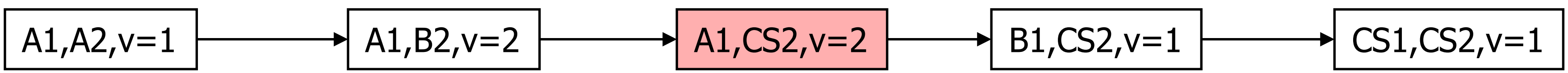
Taking time into account



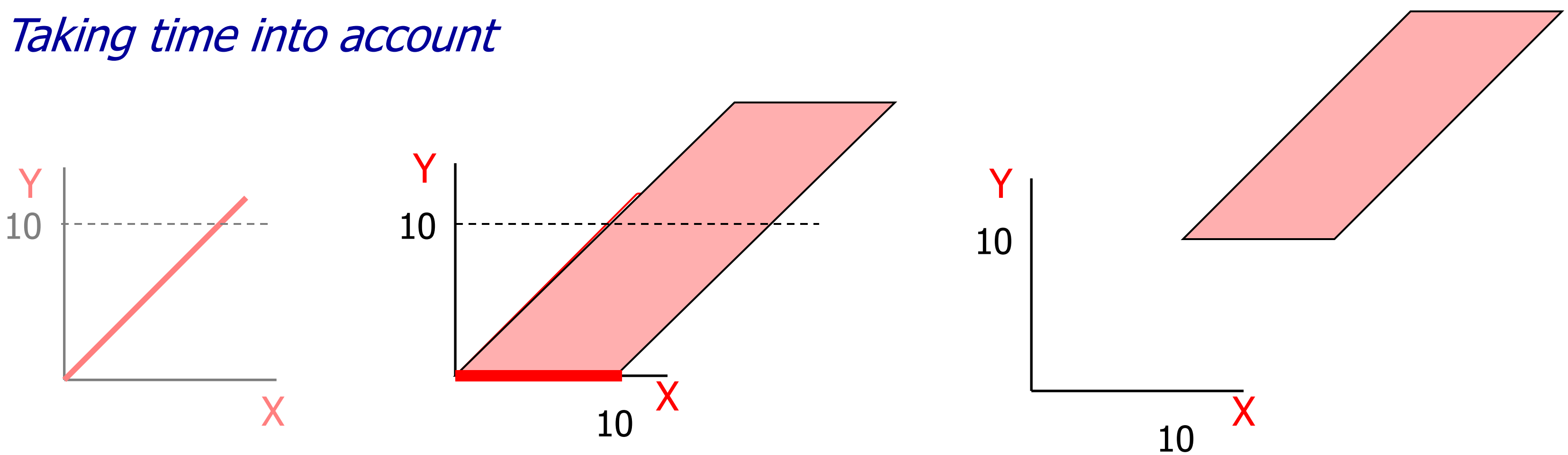
Fischers cont.



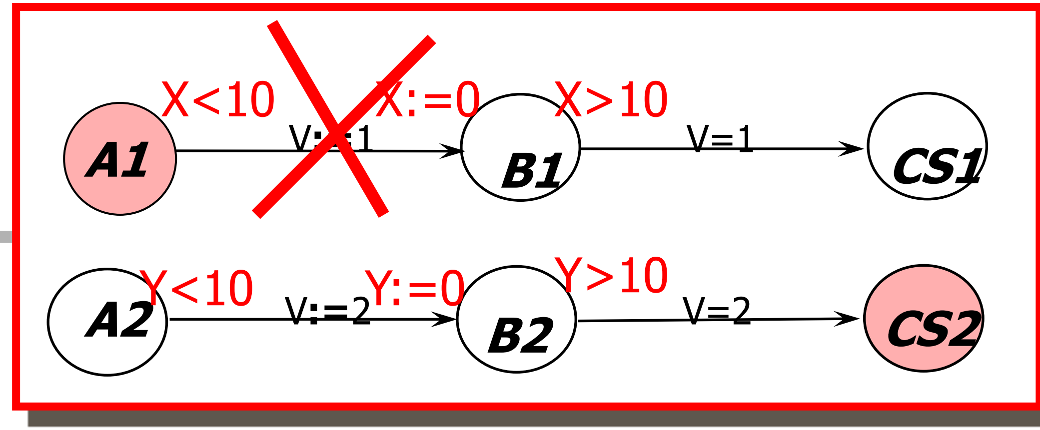
Untimed case



Taking time into account



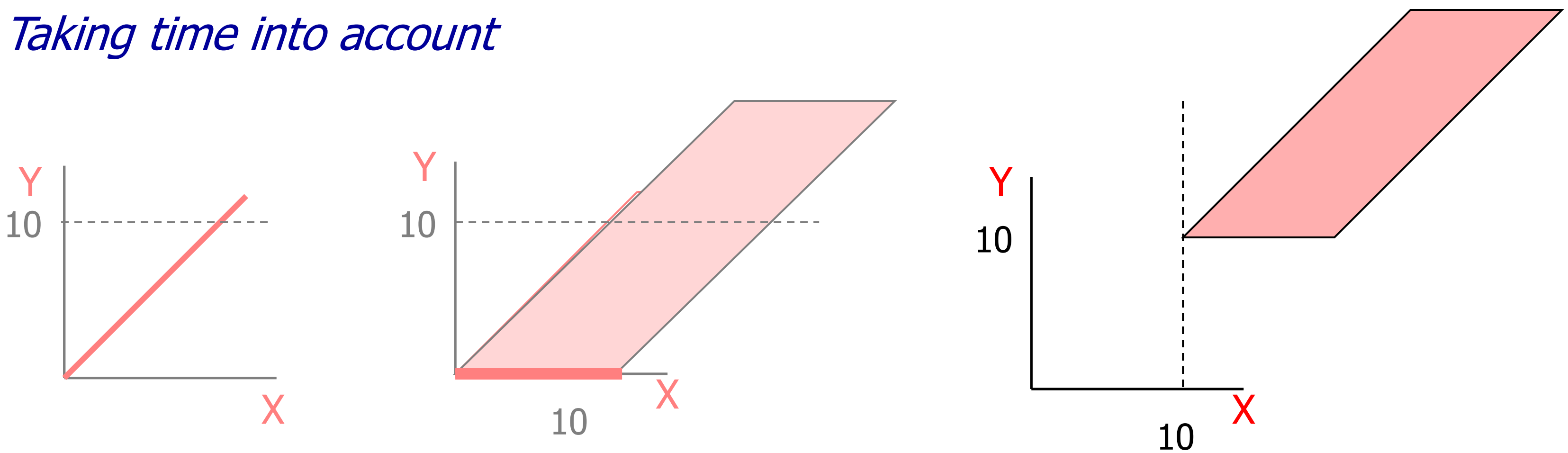
Fischers cont.



Untimed case

A1,A2,v=1 → A1,B2,v=2 → A1,CS2,v=2 → B1,CS2,v=1 → CS1,CS2,v=1

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): SP(Z) or $Z\uparrow$
  - $[Z\uparrow] = \{u+d \mid d \in R, u \in [Z]\}$
- Pre-condition: WP(Z) or $Z\downarrow$ (the dual of $Z\uparrow$)
  - $[Z\downarrow] = \{u \mid u+d \in [Z]$ 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↑], [Z↓], [{x}Z]

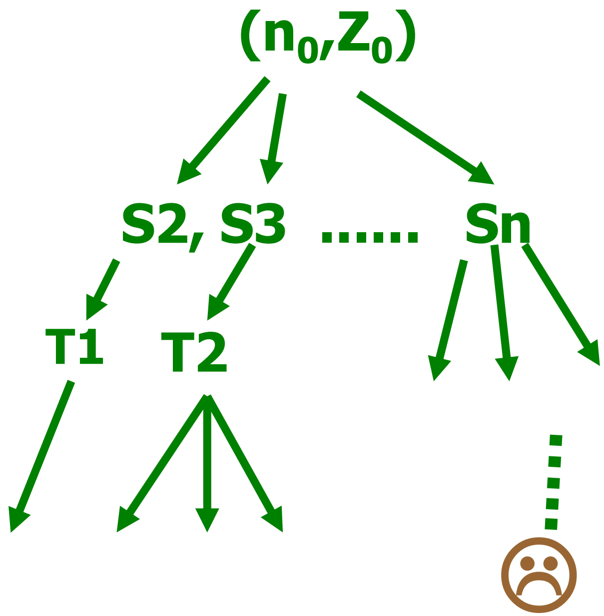
One-step reachability: $S_i \rightsquigarrow S_j$

- Delay: $(n,Z) \rightarrow (n,Z')$ where $Z' = Z\uparrow \wedge inv(n)$
- Action: $(n,Z) \rightarrow (m,Z')$ where $Z' = \{x\}(Z \wedge \mathbf{g})$

  if 

- **Reach**: $(n,Z) \rightsquigarrow (m,Z')$ if $(n,Z) \rightarrow\rightarrow (m,Z')$
- **Successors**$(n,Z) = \{(m,Z') \mid (n,Z) \rightsquigarrow (m,Z'),\ Z' \neq \emptyset\}$

Now, we have a search problem



EF ☹