OUTLINE

- A Brief Introduction
  - Motivation ... what are the problems to solve
  - CTL, LTL and basic model-checking algorithms
- Timed Systems
  - Timed automata and verification problems
  - UPPAAL tutorial (1): data stuctures & algorithms
  - UPPAAL tutorial (2): input languages
  - TIMES: From models to code "guaranteeing" timing constraints
- Further topics/Recent Work
  - Systems with buffers/queues [CAV 2006]

Lecture 3

Timed Automata and TCTL

syntax, semantics, and verification problems

Timed Automata

=

Finite Automata + Clock Constraints + Clock resets

Clock Constraints

For set C of clocks with $x, y \in C$, the set of *clock constraints* over C, $\Psi(C)$, is defined by

$$\alpha ::= x \prec c \;\big|\; x - y \prec c \;\big|\; \neg\alpha \;\big|\; (\alpha \wedge \alpha)$$

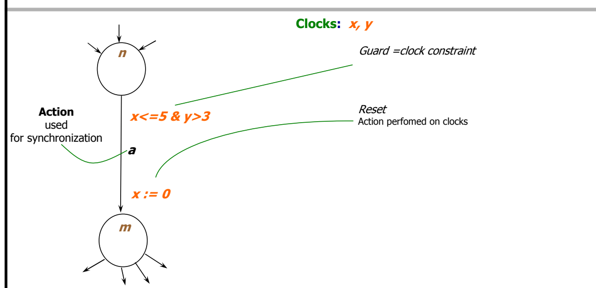
where $c \in \mathbb{N}$ and $\prec \in \{<, \leqslant\}$.

Timed Automata

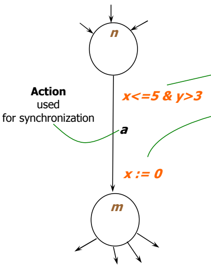
A *timed automaton* $\mathcal{A}$ is a tuple $(L, l_0, E, \mathit{Label}, C, \mathit{clocks}, \mathit{guard}, \mathit{inv})$ with

- L, a non-empty, finite set of locations with initial location $l_0 \in L$
- $E \subseteq L \times L$, a set of edges
- $\mathit{Label} : L \longrightarrow 2^{AP}$, a function that assigns to each location $l \in L$ a set $\mathit{Label}(l)$ of atomic propositions
- C, a finite set of clocks
- $\mathit{clocks} : E \longrightarrow 2^C$, a function that assigns to each edge $e \in E$ a set of clocks $\mathit{clocks}(e)$
- $\mathit{guard} : E \longrightarrow \Psi(C)$, a function that labels each edge $e \in E$ with a clock constraint $\mathit{guard}(e)$ over C, and
- $\mathit{inv} : L \longrightarrow \Psi(C)$, a function that assigns to each location an *invariant*.

Timed Automata: Syntax



Timed Automata: Semantics



Clocks: *x, y*

Guard =clock constraint

Reset
Action perfomed on clocks

State

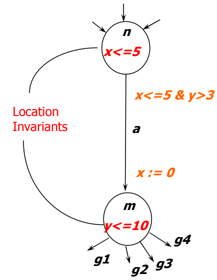
(*location* , *x*=v , *y*=u) where v,u are in $\mathbb{R}$

Transitions

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

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

Timed Automata with *Invariants*



Clocks: *x, y*

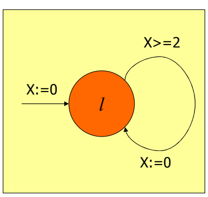
Transitions

(*n* , *x*=2.4 , *y*=3.1415) ~~$\xrightarrow{3.2}$~~

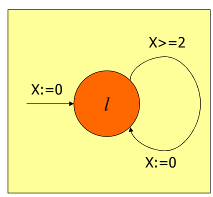
(*n* , *x*=2.4 , *y*=3.1415) $\xrightarrow{1.1}$ (*n* , *x*=3.5 , *y*=4.2415)

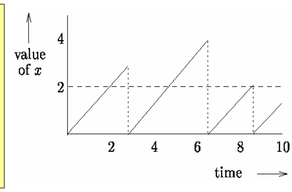
Invariants insure progress!!

Timed Automata: Example

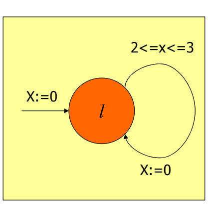


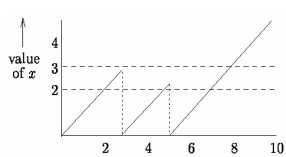
Timed Automata: Example



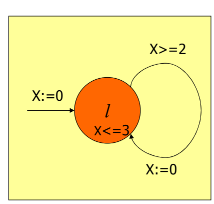


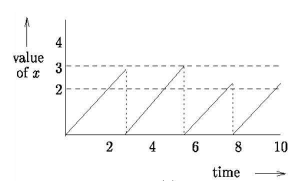
Timed Automata: Example



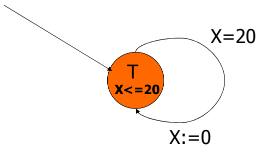


Timed Automata: Example

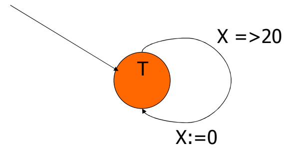




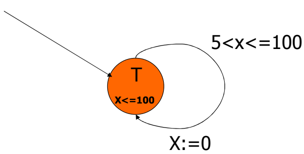
Timed Automata: Example
(periodic task)



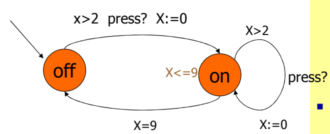
Timed Automata: Example
(sporadic task)



Timed Automata: Example
(aperiodic task)



Timed Automata: Light Switch

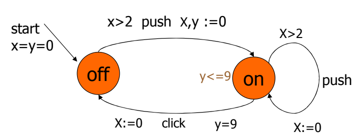


- Switch may be turned on whenever at least 2 time units has elapsed since last "turn off"
- Light automatically switches off after 9 time units if it is not pressed.

Semantics (definition)

- *clock valuations*: $V(C) \quad v : C \rightarrow R_{\geq 0}$
- *state*: $(l, v) \quad where \quad l \in L \quad and \quad v \in V(C)$
- *action transition* $(l,v) \xrightarrow{a} (l',v') \text{ iff } \; l \xrightarrow{g\;a\;r} l'$
  $g(v) \;\; and \;\; v' = v[r] \;\; and \;\; Inv(l')(v')$
- *delay Transition* $(l,v) \xrightarrow{d} (l, v+d) \text{ iff}$
  $Inv(l)(v+d') \text{ whenever } d' \leq d \in R_{\geq 0}$

Timed Automata: Example



$(off, x=y=0) \xrightarrow{3.5} (off, x=y=3.5) \xrightarrow{push}$

$(on, x=y=0) \xrightarrow{\pi} (on, x=y=\pi) \xrightarrow{push}$

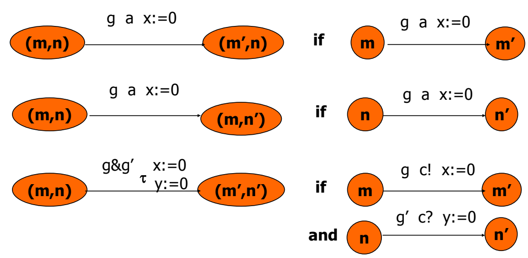
$(on, x=0, y=\pi) \xrightarrow{3} (on, x=3, y=\pi+3) \xrightarrow{9-(\pi+3)}$

$(on, x=9-(\pi+3), y=9) \xrightarrow{click} (off, x=0, y=9) \ldots$

Modeling Concurrency

- Products of automata
- Parallel composition

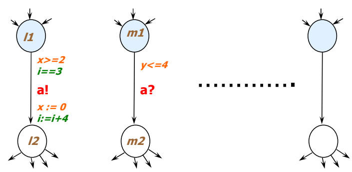
CCS Parallel Composition (implemented in UPPAAL)



Where a is an action c! or c? or τ
c is a channel name

The UPPAAL Model

= Networks of Timed Automata + Integer Variables +….



Two-way synchronization on *complementary* actions.

Closed Systems!

Example transitions

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

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 Inclusion ☹
  - 1-clock, finite traces, decidable [Ouaknine & Worrell 04]
  - 1-clock, infinite traces & Buchi-conditions, undecidable [Abdulla et al 05]
- Untimed Language Inclusion ☺
- (Un)Timed Bisimulation ☺
- Reachability Analysis ☺
- 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 | ¬ ϕ | ϕ ∨ ϕ | EX ϕ | E[ϕ U ϕ] | A[ϕ U ϕ]

where **P** ∈ AP (atomic propositions)

TCTL

Henzinger, Sifakis et al 1992

Syntax

ϕ ::= P | g | ¬ ϕ | ϕ ∨ ϕ | z.ϕ | E[ϕ U ϕ] | A[ϕ U ϕ]

where **P** ∈ AP (atomic propositions) and g is a Clock constraint

AG (P imply z.(z<10 or q))

Timed CTL (a simplified version of TCTL)

Syntax

ϕ ::= p | ¬ ϕ | ϕ ∨ ϕ | EX ϕ | E[ϕ U ϕ] | A[ϕ U ϕ]

where **p** ∈ AP (atomic propositions) **or** a Clock constraint

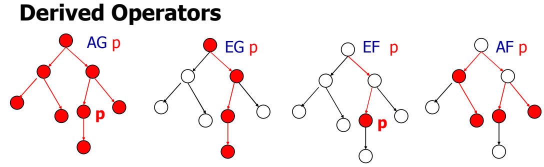
Timed CTL

Syntax

ϕ ::= p | ¬ ϕ | ϕ ∨ ϕ | EX ϕ | E[ϕ U ϕ] | A[ϕ U ϕ]

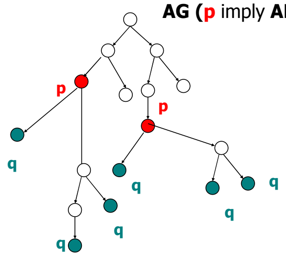
where **p** ∈ AP (atomic propositions) **or** Clock constraint

Derived Operators



Liveness: p - -> q *"p leads to q"*

AG (p imply AF q)



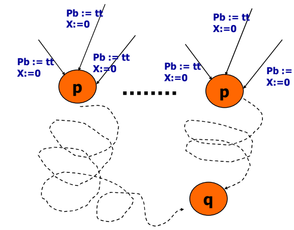
Bounded Liveness/Response

[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/Response

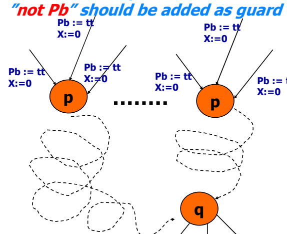
[TACAS 98]

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

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



Pb:=ff should be On all eadges leaving q

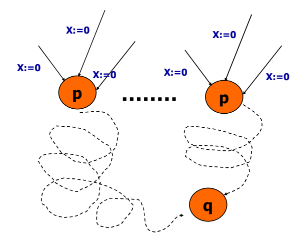
Bounded Liveness

[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



Timed CTL in UPPAAL

EF p | AG p | EG p | AF p | p - -> q

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

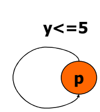
Process Location (a location in automaton A)

Clock constraint

predicate over data variables

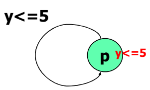
p leads to q *denotes* **AG (p imply AF q)**

Problem with Zenoness



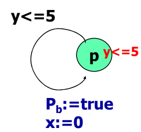
A Zeno-automaton may satisfy the formula Without containing a state where q is true

EXAMPLE



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

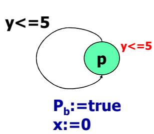
EXAMPLE



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

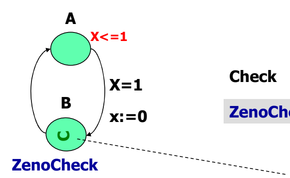
AG ((P_b and x>10) imply q)

is satisfied !!!

Solution with UPPAAL

Check Zeno-freeness by an extra observer

System | | ZenoCheck



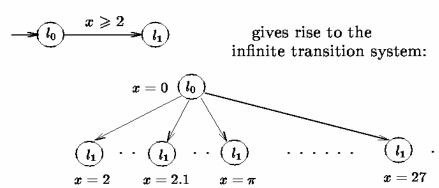
Check

ZenoCheck.A - - > ZenoCheck.B

Committed location!

REACHABILITY ANALYSIS
using Regions

Infinite State Space!

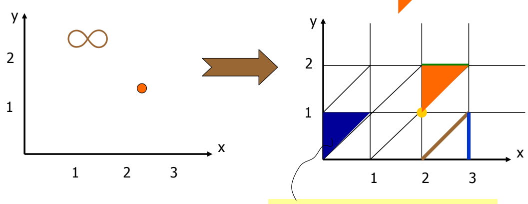


However, the reachability problem is decidable ☺ Alur&Dill 1991

Region: From infinite to finite

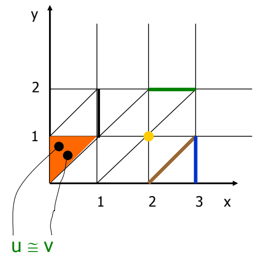
Concrete State
(n, x=2.2, y=1.5)

Symbolic state (region)
(n, )



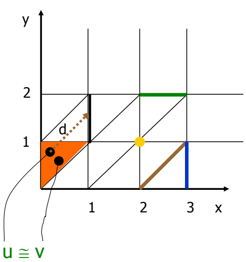
An equivalence class (i.e. a *region*)
There are only *finite* many such!!

Region equivalence (Intuition)



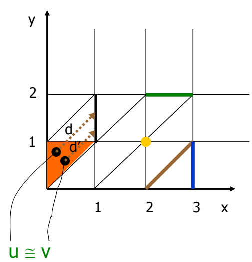
u ≅ v iff (l,u) and (l,v) may reach the same set of eqivalence classes

Region equivalence (Intuition)



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

Region equivalence (Intuition)

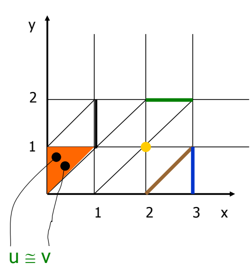


u ≅ v iff (l,u) and (l,v) may reach the same set of eqivalence classes

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)}

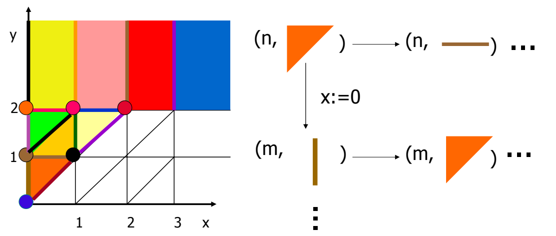
Region equivalence (alternatively)



u ≅ v iff u and v satisfy exactly the same set of constraints in the form of
xi ~ m and xi-xj ~ 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

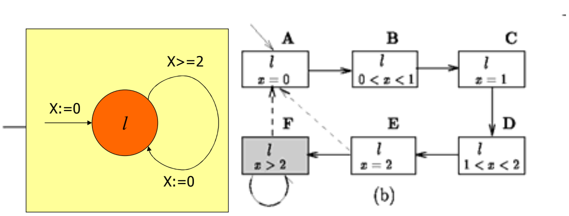
(m, [u]) ⟶ (n, [v]) if (m, u) ⟶ (n,v)

Theorem

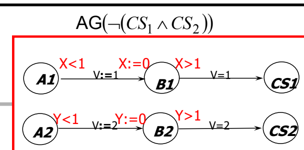
u≈v implies
- u(x:=0) ≈ v(x:=0)
- u+n ≈ v+n for all natural number n
- for all d<1: u+d ≈ v+d' for some d'<1

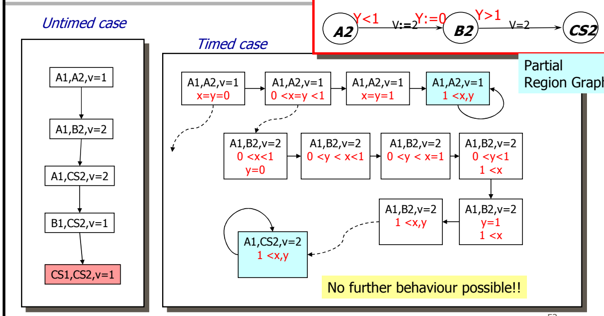
"Region equivalence' is preserved by "addition" and reset.
(also preserved by "subtraction" if clock values are "bounded")

Region graph of a simple timed automata



Fischers again



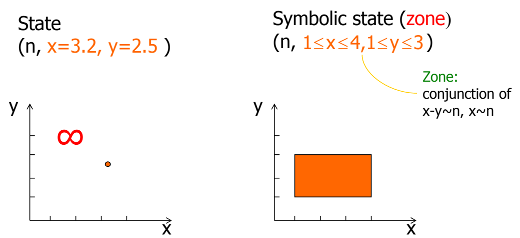


Problems with Region Construction

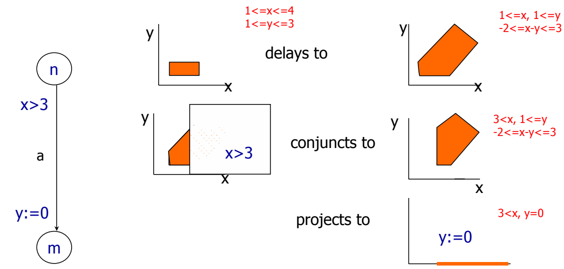
- Too many 'regions'
  - Sensitive to the maximal constants
  - e.g. x>1,000,000, y>1,000,000 as guards in TA
- The number of regions is highly exponential in the number of clocks and the maximal constants.

REACHABILITY ANALYSIS
using ZONES

Zones: From infinite to finite

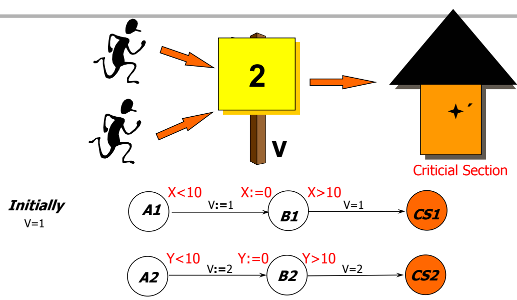


Symbolic Transitions

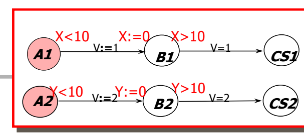


Thus (n, 1<=x<=4,1<=y<=3) =a=> (m, 3<x, y=0)

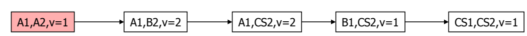
Fischer's Protocol
analysis using zones



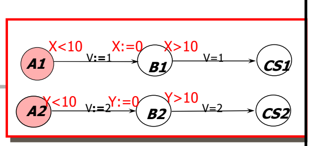
Fischers cont.



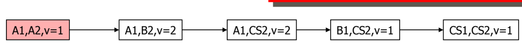
Untimed case



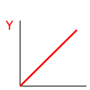
Fischers cont.



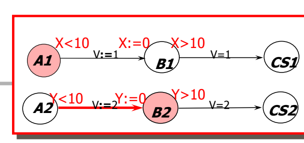
Untimed case



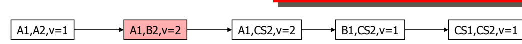
Taking time into account



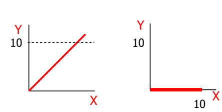
Fischers cont.



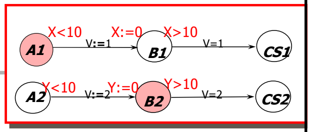
Untimed case



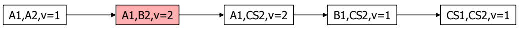
Taking time into account



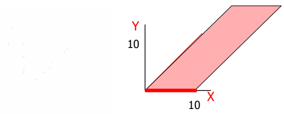
Fischers cont.



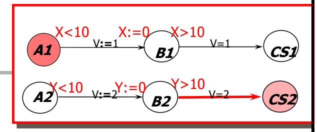
Untimed case



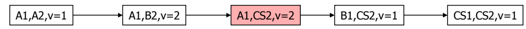
Taking time into account



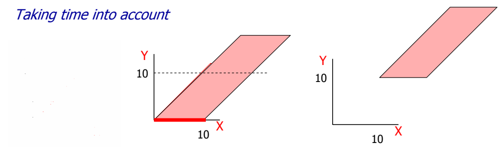
Fischers cont.



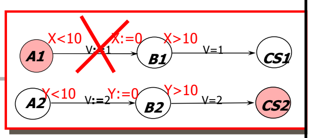
Untimed case



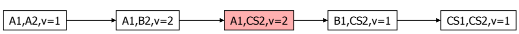
Taking time into account



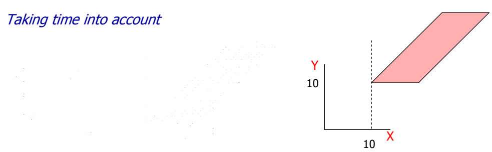
Fischers cont.



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

- Strongest post-condition (Delay): SP(Z) or Z↑
  - [Z↑] = {u+d| d ∈ R, u∈[Z]}
- Weakest pre-condition: WP(Z) or Z↓ (the dual of Z↑)
  - [Z↓] = {u| u+d∈[Z] for some d∈R}
- Reset: {x}Z or Z(x:=0)
  - [{x}Z] = {u[0/x] | u ∈[Z]}
- Conjunction
  - [Z&g]= [Z]∩[g]

Two more operations on Zones

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

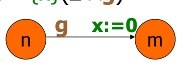
Theorem on Zones

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: $[Z\uparrow]$, $[Z\downarrow]$, $[\{x\}Z]$

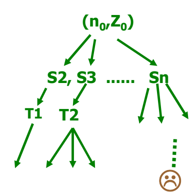
One-step reachability: Si ⁓ Sj

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

if 

- **Reach**: (n,Z) ⁓ (m,Z') if (n,Z) →→(m,Z')
- **Successors**(n,Z)={(m,Z') | (n,Z) ⁓ (m,Z'), $Z' \neq \emptyset$}

Now, we have a search problem



EF ☹