Model Checking Timed Systems using Clock Constraints (1)

Reachability Analysis and Constraint solving

Problem: reachability analysis

- Give an automaton and a location n, or a local property F
- Question: does it exist an execution of the automaton, that leads to n (or a state where F holds)?
- This is the so called reachability problem.

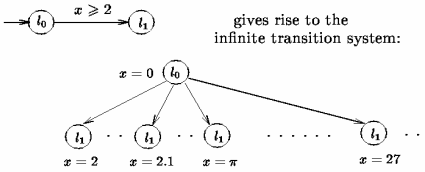
Formalizing requirements

- Reachability properties: E<> Q
  - E<> P.stop
  - E<> (y>200)
- Invariant properties: A[] Q (not E<> not Q)
  - A[] not (P1.CS and P2.cs)
  - A[] (i < 100)
  - A[] (x>10 imply i>100)
    - After 10:00AM, i should be larger than 100
- Bounded Liveness Properties: F1 $\rightarrow_{<=10}$ F2
  - A[](f1 and x>10 imply f2)

Other Verification Problems

- Timed Language Inclusion ☹
- 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?

Infinite State Space!

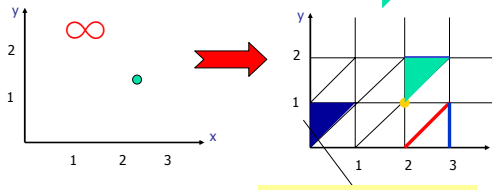


However, the reachability problem is decidable ☺ Alur&Dill 1991

Region: From infinite to finite

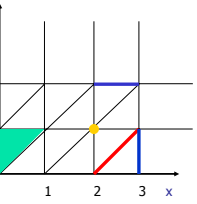
Concrete State
(n, x=2.2, y=1.5)

Symbolic state (region)
(n,)



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

Region equivalence (Intuition)



u ≅ v iff 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

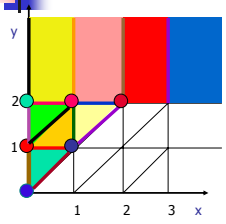
This is not quite correct; we need to consider the MAX more carefully

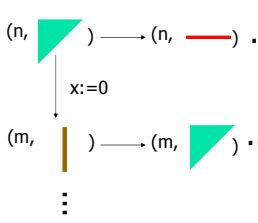
Region equivalence: Definition *[Alur and Dill 1990]*

- u,v are clock assignments
- u≈v iff
  - For all clocks x, either u(x)<=Cx and v(x)<=Cx or ⌊u(x)⌋=⌊v(x)⌋ (the same integer part)
  - 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)}

Regions

Finite partitioning of state space



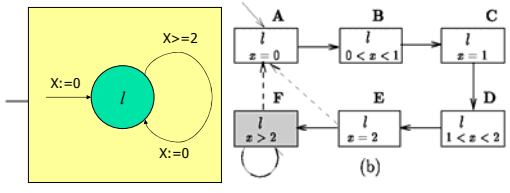


OBS: there are only Finite many regions

An Important Theorem for Region Equivalence

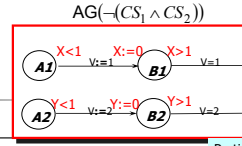
- u≈v implies
  - u(x:=0) ≈ v(x:=0)
  - u+n ≈ v+n for all natural number n
  - for all d<1:
  - u+d ≈ v+d' for some d'<1
- that is, 'region equivalence' is preserved by "addition" and reset
- in fact, it is also preserved by "subtraction" if clock values are 'bounded'

Region graph of a simple timed automata

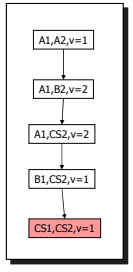


Fischers again

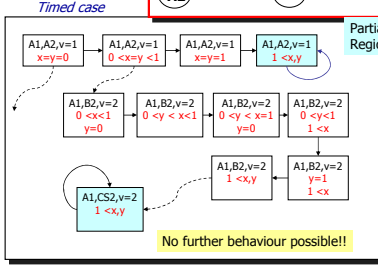
$AG(\neg(CS_1 \wedge CS_2))$



Untimed case



Timed case



Problems with Region Partitioning

- Too many 'regions'
- Sensitive to the maximal constants
  - e.g. x>1000000
- The number of regions is highly exponential in the number of clocks and the maximal constants (used to compare with clocks)

The more efficient solution *[1989 Dill ... 1994]*

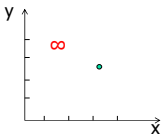
Symbolic Reachability
Using Clock Constraints

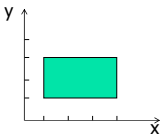
Zones: From infinite to finite

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

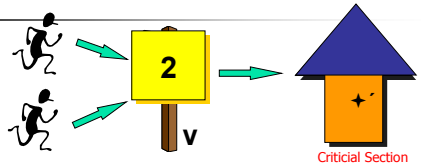
Symbolic state (zone)
(n, 1≤x≤4,1≤y≤3)

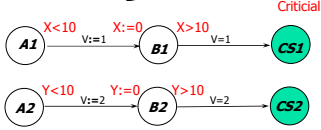
Zone:
conjuncti
x-y~n, x



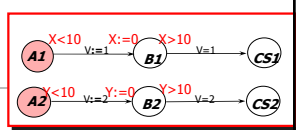


ischer's Protocol
nalysis using zones

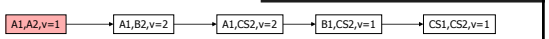




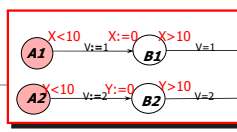
Fischers cont.



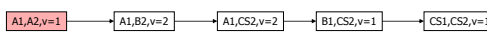
Untimed case



Fischers cont.

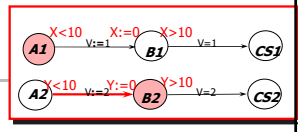


Untimed case



Taking time into account

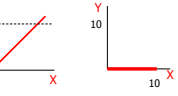
Fischers cont.



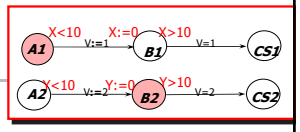
med case

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

ng time into account



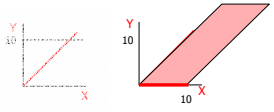
Fischers cont.



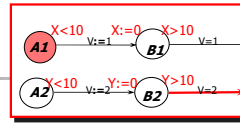
Untimed case

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

Taking time into account



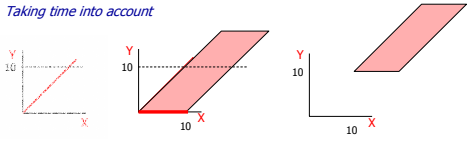
Fischers cont.



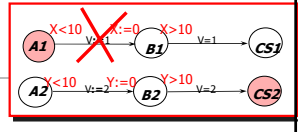
Untimed case

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

Taking time into account



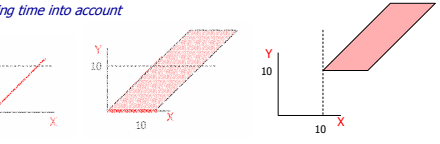
Fischers cont.



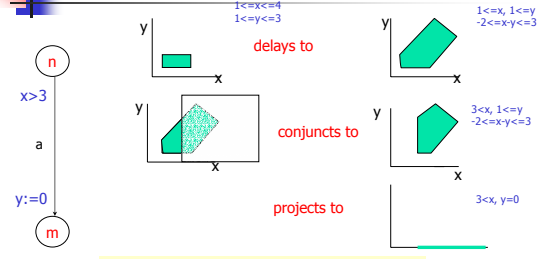
med case

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

ng time into account



Symbolic Transitions



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

Zones = Conjuctive constraints

- A zone Z is a conjunctive formula:
  g_1 & g_2 & ... & g_n
  where g_i is a clock constraint:
  $x_i \sim b_i$ or $x_i - x_j \sim b_{ij}$
- Use a zero clock x_0 (constant 0)
- A zone can be written as a set:
  $\{x_i - x_j \sim b_{ij} \mid \sim$ is $<$ 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}
  - The semantics

(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]

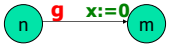
An important theorem on Zone

- The set of zones is closed under all constrain operations (including x:=x-c or x:=x+c)
  - That is, the result of the operations on a zone is a
  - That is, there will be a zone (a finite object i.e a zone/constraints) to represent the sets: [Z↑], [Z↓ [{x}Z]

One-step reachability: Si→Sj

Delay: (n,Z) → (n,Z') where Z'= Z↑ ∧ inv(n)

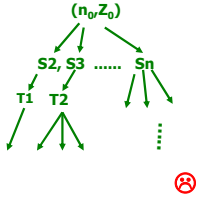
Action: (n,Z) → (m,Z') where Z'= {x}(Z ∧**g**)

if 

Successors(n,Z)={(m,Z') | (n,Z) →→(m,Z'), Z'≠Ø}

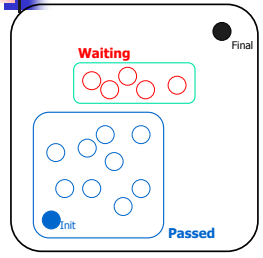
- Sometime we write: (n,Z)→(m,Z') if (m,Z') is a successor of (n,Z)

Now, we have a search problem



Forward Rechability

Init -> Final ?



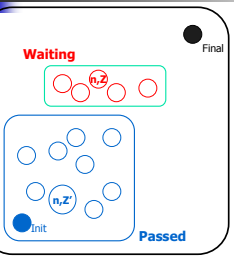
INITIAL **Passed** := Ø;
Waiting := {(n0,Z0)}

REPEAT
- pick (n,Z) in **Waiting**
- **if** for some Z' ⊇ **Z** (n,Z') in **Passed then STOP**
- **else** (explore) add successors(n,Z) to **Waiti** Add (n,Z) to **Passed**

UNTIL **Waiting** = Ø
or
Final is in **Waiting**

Forward Rechability

Init -> Final ?



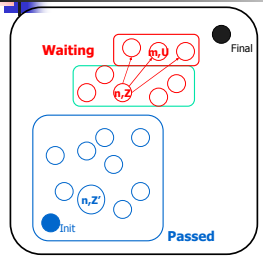
INITIAL **Passed** := Ø;
Waiting := {(n0,Z0)}

REPEAT
- pick (n,Z) in **Waiting**
- **if** for some Z' ⊇ Z
  (n,Z') in **Passed** **then** **STOP**

UNTIL **Waiting** = Ø
or
Final is in **Waiting**

Forward Rechability

Init -> Final ?



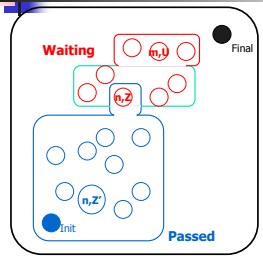
INITIAL **Passed** := Ø;
Waiting := {(n0,Z0)}

REPEAT
- pick (n,Z) in **Waiting**
- **if** for some Z' ⊇ Z
  (n,Z') in **Passed** **then** **STOP**
- **else** /explore/ add
  successors(n,Z) to **Waiting**;

UNTIL **Waiting** = Ø
or
Final is in **Waiting**

Forward Rechability

Init -> Final ?



INITIAL **Passed** := Ø;
Waiting := {(n0,Z0)}

REPEAT
- pick (n,Z) in **Waiting**
- **if** for some Z' ⊇ Z
  (n,Z') in **Passed** **then** **STOP**
- **else** /explore/ add
  successors(n,Z) to **Waiti**
  Add (n,Z) to **Passed**

UNTIL **Waiting** = Ø
or
Final is in **Waiting**

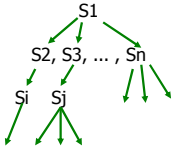
Two more operations on Zones

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

All Operations on Zones

(needed for reachability analysis)

- Transformation
  - Conjunction
  - Post condition (delay)
  - Reset
- Consistency Checking
  - Inclusion
  - Emptiness



EFFICIENT IMPLEMENTATION

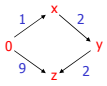
Canonical Datastructures for Zones

Difference Bounded Matrices Bellman 1958, Dill 1989

nclusion

1
x<=1
y-x<=2
z-y<=2
z<=9

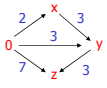
Graph



? ⊆ ?

2
x<=2
y-x<=3
y<=3
z-y<=3
z<=7

Graph



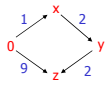
Canonical Dastructures for Zones

Difference Bounded Matrices Bellman 1958, Dill 1989

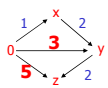
Inclusion

Z1
x<=1
y-x<=2
z-y<=2
z<=9

Graph



Shortest Path Closure

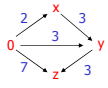


? ⊆ ?

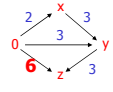
Z1 ⊆ Z2 !

Z2
x<=2
y-x<=3
y<=3
z-y<=3
z<=7

Graph



Shortest Path Closure



Canonical Datastructures for Zones

Difference Bounded Matrices Bellman 1958, Di

Emptiness

Z
x<=1
y>=5
y-x<=3

Graph

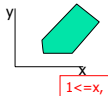
Negative Cycle
iff
empty solution set

Compact

Canonical Datastructures for Zones

Difference Bounded Matrices

Conjunction

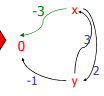


Z∧g

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

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

Add new edge for g



Canonical Dastructures for Zones

Difference Bounded Matrices

Delay

Z

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

Z ↑

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

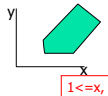
Shortest Path Closure

Remove upper bounds on clocks

Canonical Datastructures for Zones

Difference Bounded Matrices

Reset



Z

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

{y}Z

y=0, 1<=x

Remove all bounds involving y and set y to 0

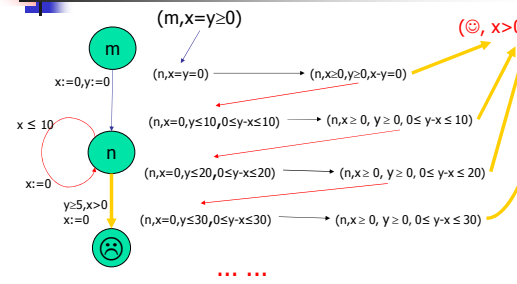
COMPLEXITY

- Computing the shortest path closure, the cannonical form of a zone: $O(n^3)$ [Dijkstra's alg.]
- Run-time complexity, mostly in $O(n)$
  (when we keep all zones in cannonical form)

How about termination?

We need the normalization operation according to the maximal constant

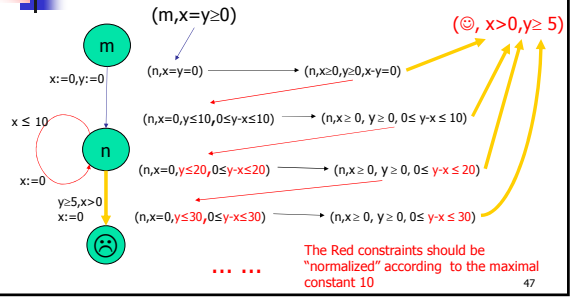
Example: is ☹ reachable?



The search process may never terminate

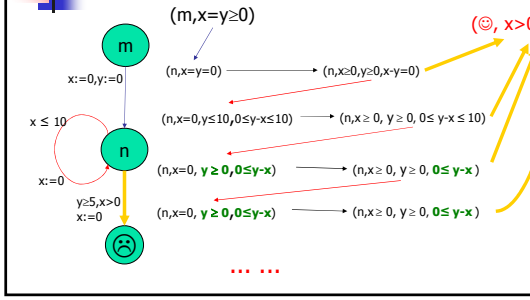
- The new zones created (due to the red edge/transitions) is getting larger and larger … …
- Note that in this example, Breadth first may terminate the search with result: yes (i.e. ☺ is reachable)
- But in general, there is no guarantee for termination even using Breadth first e.g.
  - Consider the same example and check the reachability of (☺,x>y) (it is NOT reachable; then the search will try to construct the whole state space)
- We need a solution !! (using the Maximal constant)

Example: is ☹ reachable?

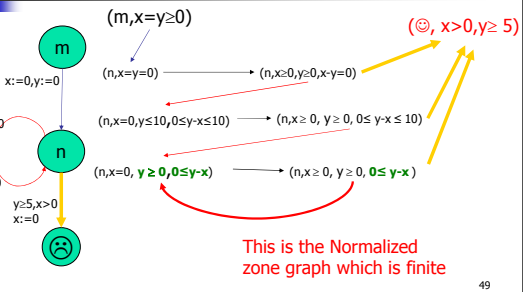


The Red constraints should be "normalized" according to the maximal constant 10

Example: is ☹ reachable?



Example: is ☹ reachable?



This is the Normalized zone graph which is finite

The normalization Operation

(For automata containing no constraints on clock differences, Only)

- First compute the shortest path closure of a zone
- Remove all constraints in the form:

  Xi <(≤) m or Xi-Xj <(≤) n

  where m, n>MAX
- Replace all constraints in the form:

  Xi >(≥) m or Xi-Xj >(≥) n

  where m, n>MAX

with Xi > MAX or Xi-Xj > MAX

The number of "Normalized Zones" is bounded

By the number of regions !

NOW, YOU CAN GO HOME
to MAKE YOUR OWN
MODEL CHECKER