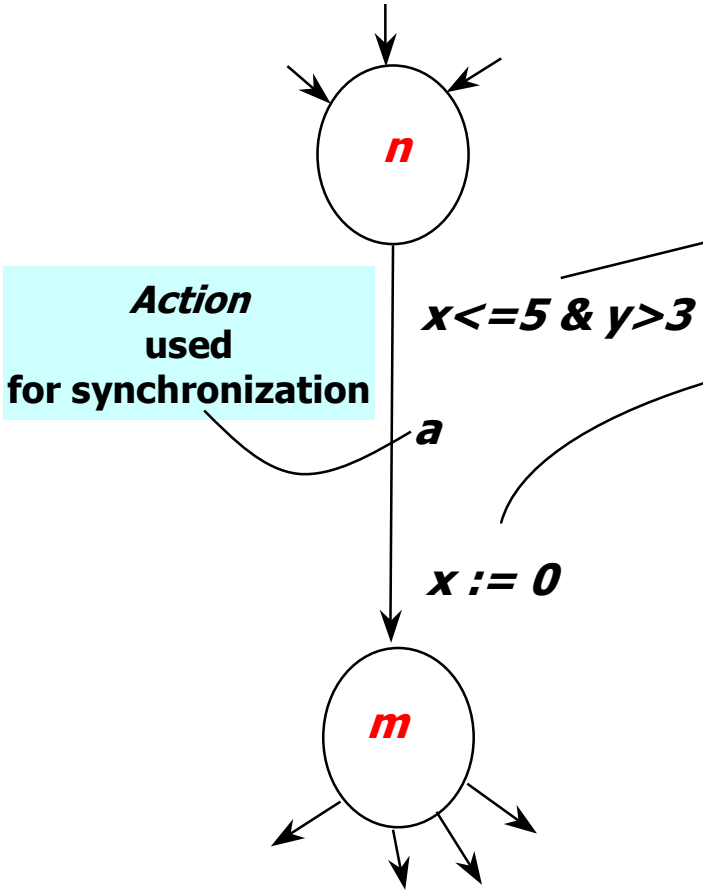
Timed Automata

Clocks: ***x, y***



Action
used
for synchronization

Guard
Boolean combination of comparisons with Integer/rational bounds

Reset
Action performed on clocks

State

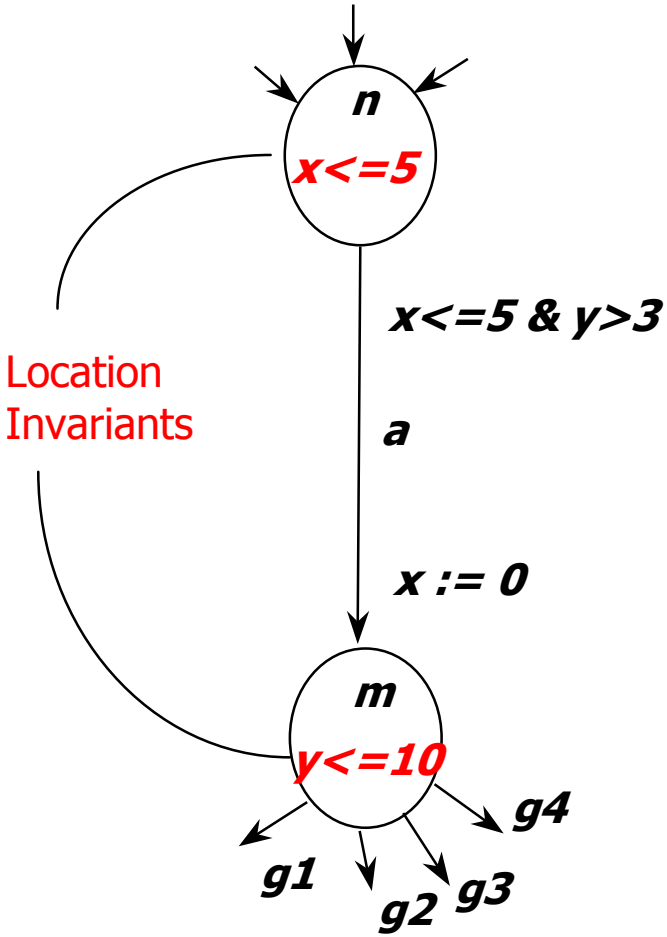
(***location*** , ***x***=v , ***y***=u) where v,u are in **R**

Transitions

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

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

Adding Invariants



Clocks: *x, y*

Transitions

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

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

Invariants ensure progress!!

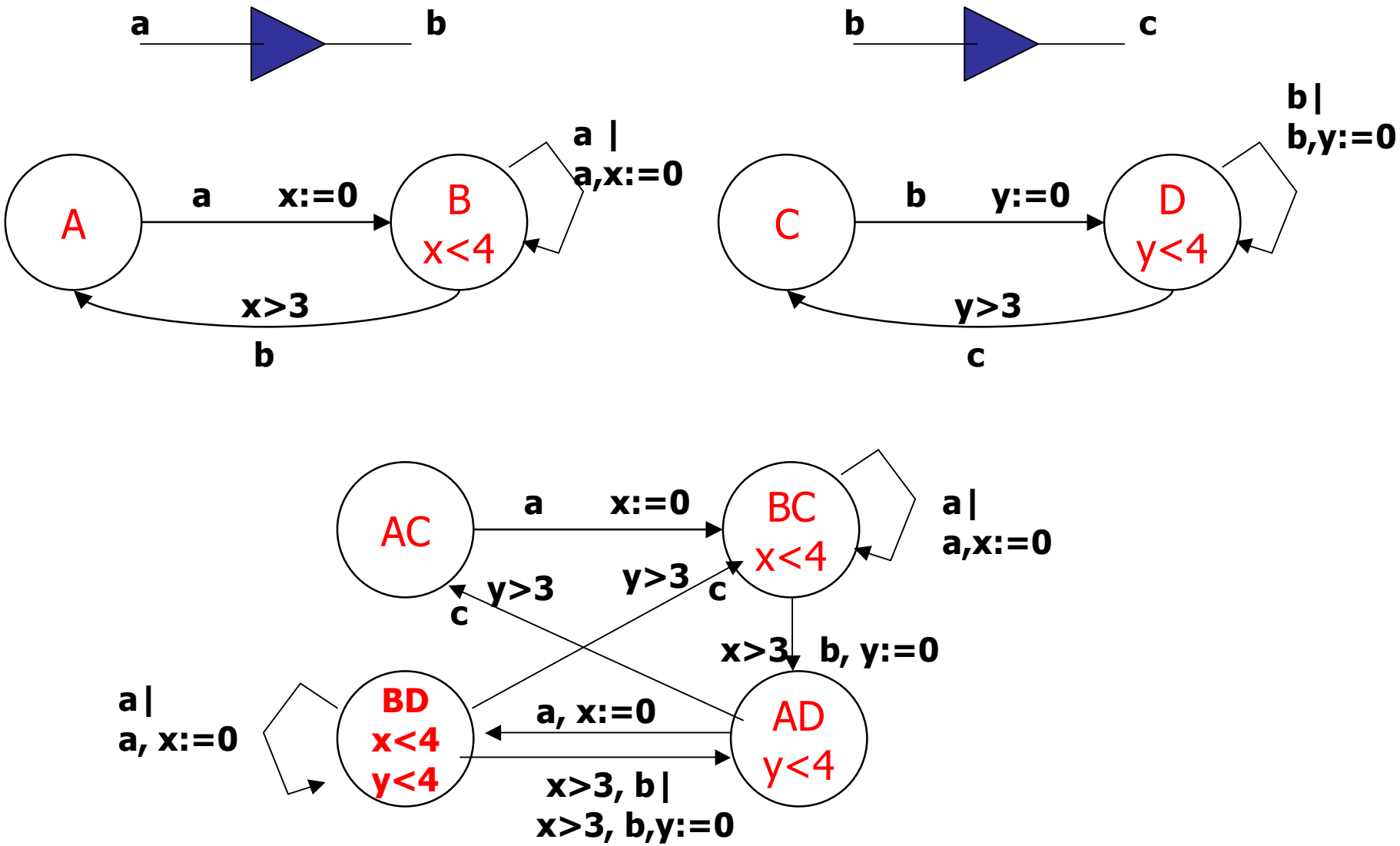
Timed Automata: Syntax

- A finite set V of locations
- A subset V^0 of initial locations
- A finite set Σ of labels (alphabet)
- A finite set X of clocks
- Invariant $Inv(l)$ for each location: (clock constraint over X)
- A finite set E of edges. Each edge has
  - source location l, target location l'
  - label a in Σ (ε labels also allowed)
  - guard g (a clock constraint over X)
  - a subset λ of clocks to be reset

Timed Automata: Semantics

- For a timed automaton A, define an infinite-state transition system $S(A)$
- States Q: a state q is a pair (l, v), where l is a location, and v is a clock vector, mapping clocks in X to R, satisfying $Inv(l)$
- (l, v) is initial state if l is in V^0 and $v(x)=0$
- Elapse of time transitions: for each nonnegative real number d, (l, v)-d->$(l, v+d)$ if both v and $v+d$ satisfy $Inv(l)$
- Location switch transitions: (l, v)-a->(l', v') if there is an edge (l, a, g, λ, l') such that v satisfies g and $v'=v[\lambda:=0]$

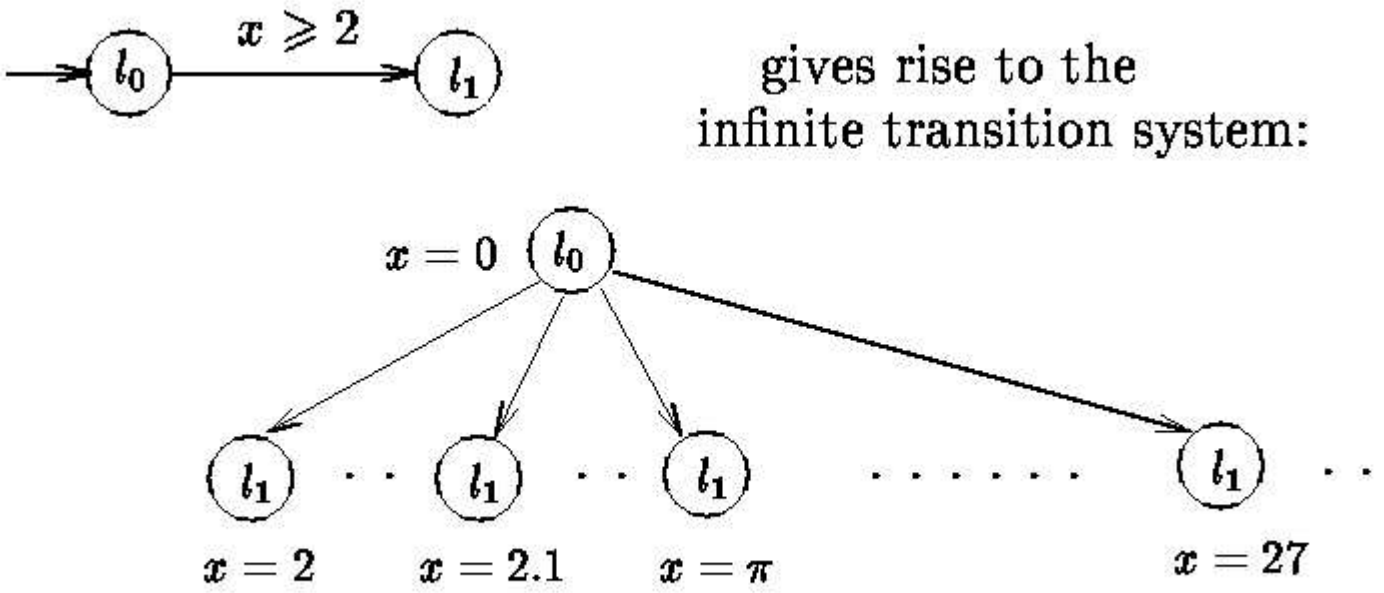
Product Construction



Verification

- System modeled as a product of timed automata
- Verification problem reduced to reachability or to temporal logic model checking
- Applications
  - Real-time controllers
  - Asynchronous timed circuits
  - Scheduling
  - Distributed timing-based algorithms

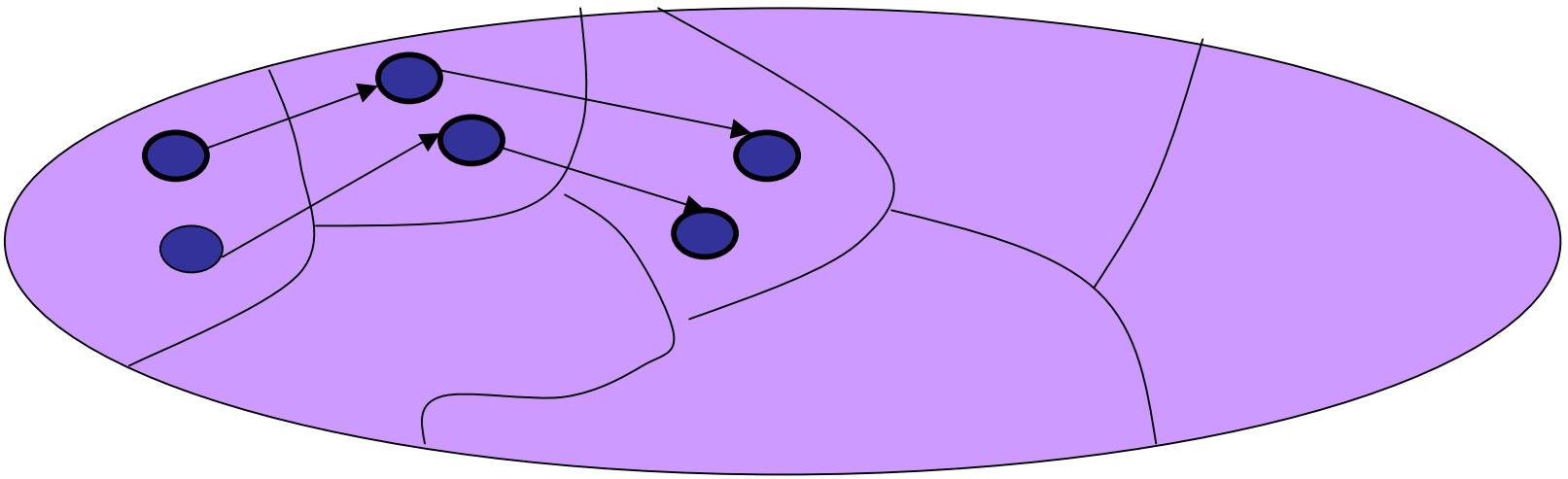
Reachability for Timed Automata



Is finite state analysis possible?
Is reachability problem decidable?

Finite Partitioning

Goal: To partition state-space into finitely many equivalence classes so that equivalent states exhibit similar behaviors



Labeled Transition System T

- Set Q of states
- Set I of initial states
- Set Σ of labels
- Set $\rightarrow$ of labeled transitions of the form

  $q \text{ -a-> } q'$

Partitions and Quotients

- Let $T=(Q, I, \Sigma, \rightarrow)$ be a transition system and $\cong$ be a partitioning of Q (i.e. an equivalence relation on Q)
- Quotient $T/\cong$ is transition system:
  1. States are equivalence classes of $\cong$
  2. A state P is initial if it contains a state in I
  3. Set of labels is Σ
  4. Transitions: P -a-> P' if q-a->q' for some q in P and some q' in P'

Language Equivalence

- Language of T: Set of possible finite strings over Σ that can be generated starting from initial states
- T and T' are language-equivalent iff they generate the same language
- Roughly speaking, language equivalent systems satisfy the same set of "safety" properties

Bisimulation

- Relation $\cong$ on QXQ' is a bisimulation iff whenever $q \cong q'$ then

  if $q\text{-}a\text{->}u$ then for some u', $u \cong u'$ and $q'\text{-}a\text{->}u'$, and

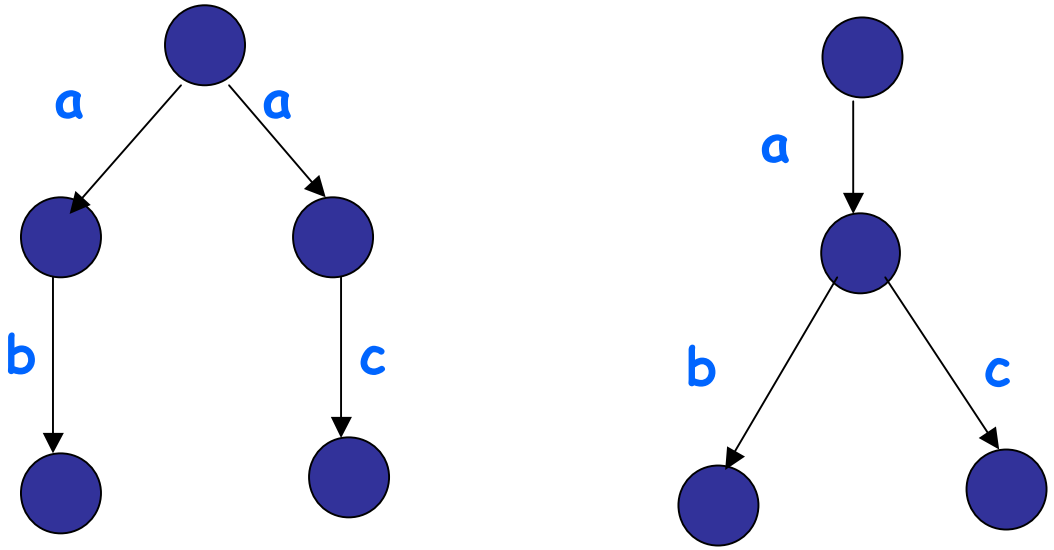
  if $q'\text{-}a\text{->}u'$ then for some u, $u \cong u'$ and $q\text{-}a\text{->}u$.

- Transition systems T and T' are bisimilar if there exists bisimulation $\cong$ on QXQ' such that

  For every q in I, there is q' in I', $q \cong q'$ and vice versa

- Many equivalent characterizations (e.g. game-theoretic)
- Roughly speaking, bisimilar systems satisfy the same set of branching-time properties (including safety)

Bisimulation Vs Language equivalence



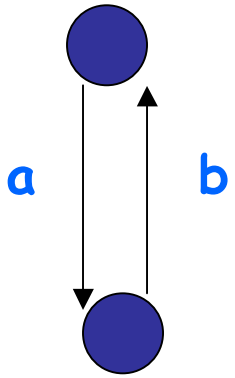
Language equivalent but not bisimilar

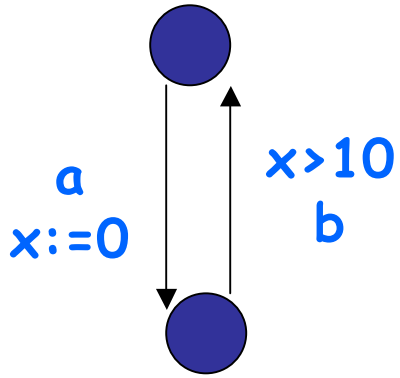
Bisimilarity -> Language equivalence

Timed Vs Time-Abstract Relations

- Transition system associated with a timed automaton:
  - Labels on continuous steps are delays in R: Timed
  - Actual delays are suppressed (all continuous steps have same label): **Time-abstract**
- Two versions of language equivalence and two versions of bisimulation
- Time-abstract relations enough to capture untimed properties (e.g. reachability, safety)

Time-abstract Vs Timed





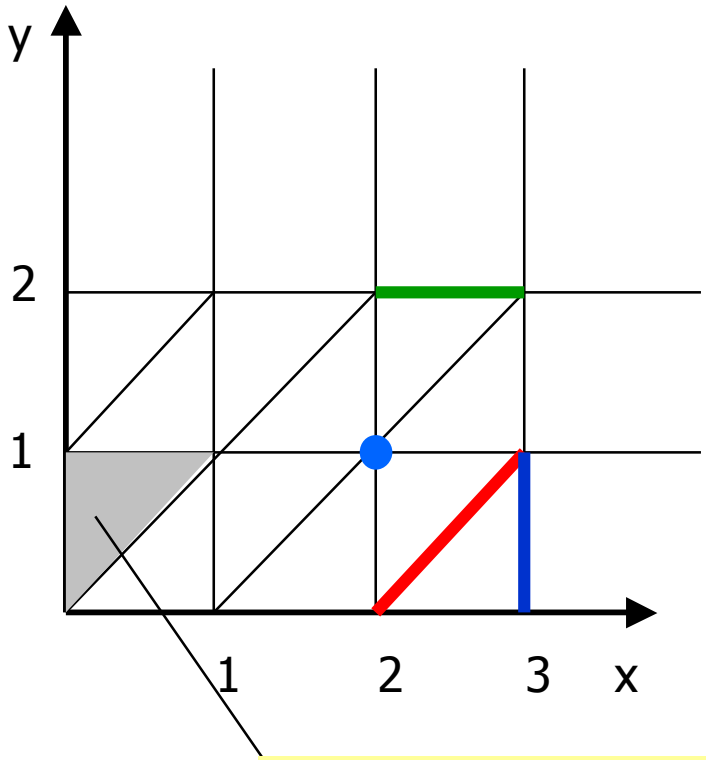
Time-abstract equivalent but not timed equivalent

Timed equivalence -> Time-abstract equivalence

Alur, Dill, 90

Regions

Finite partitioning of state space



Definition

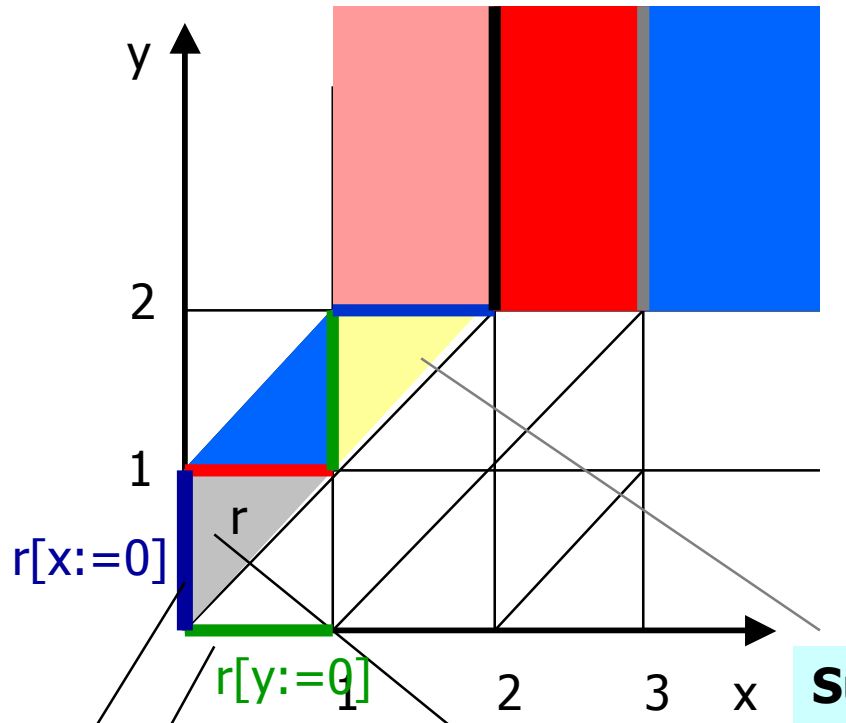
$w \cong w'$ iff they satisfy the same set of constraints of the form

$x_i < c$, $x_i = c$, $x_i - x_j < c$, $x_i - x_j = c$

for $c <=$ largest const relevant to x_i

An equivalence class (i.e. a *region*)
in fact there is only a *finite* number of regions!!

Region Operations



Successor regions, Succ(r)

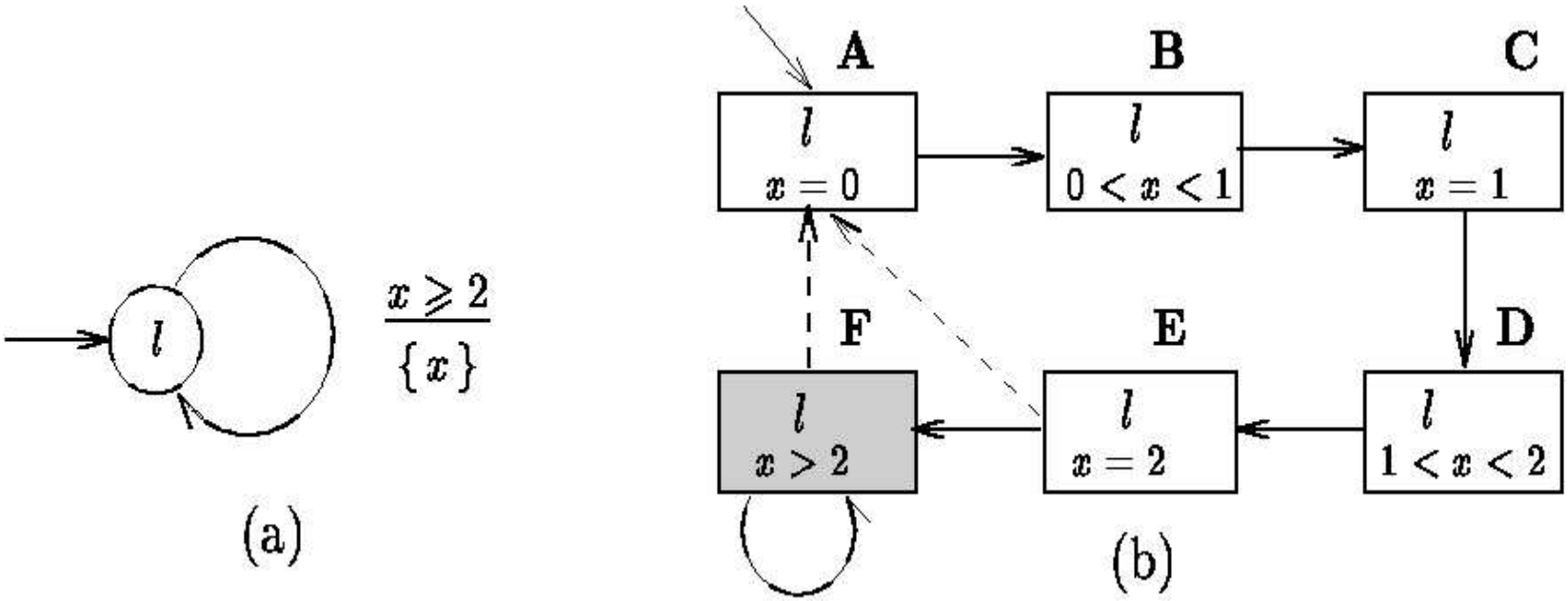
Reset regions

An equivalence class (i.e. a *region*)

Properties of Regions

- The region equivalence relation $\cong$ is a **time-abstract bisimulation**:
  - Action transitions: If $w \cong v$ and $(l,w) \xrightarrow{a} (l',w')$ for some w', then $\exists\ v' \cong w'$ s.t. $(l,v) \xrightarrow{a} (l',v')$
  - Delay transitions: If $w \cong v$ then for all real numbers d, there exists d' s.t. $w+d \cong v+d'$
- If $w \cong v$ then (l,w) and (l,v) satisfy the same temporal logic formulas

Region graph of a simple timed automata

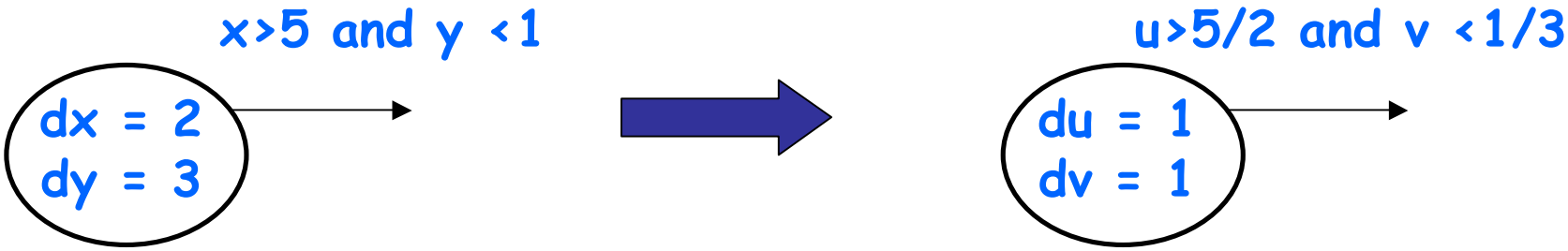


Region Graphs (Summary)

- Finite quotient of timed automaton that is time-abstract bisimilar
- Number of regions: (# of locations) times (product of all constants) times (factorial of number of clocks)
- Precise complexity class of reachability problem: PSPACE (basically, exponential dependence of clocks/constants unavoidable)
  - PSPACE-hard even for bounded constants or for bounded number of clocks

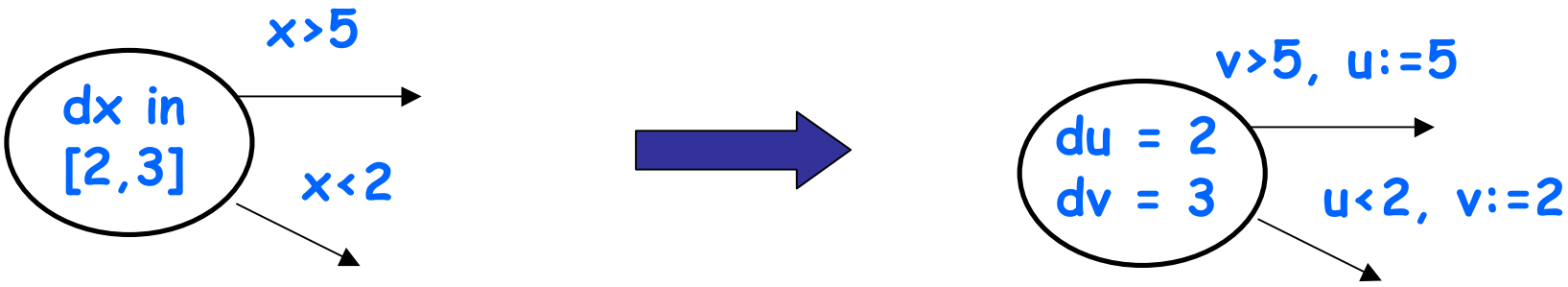
Multi-rate Automata

- Modest extension of timed automata
  - Dynamics of the form *dx = const* (rate of a clock is same in all locations)
  - Guards and invariants: *x < const, x > const*
  - Resets: *x := const*
- Simple translation to timed automata that gives time-abstract bisimilar system by scaling

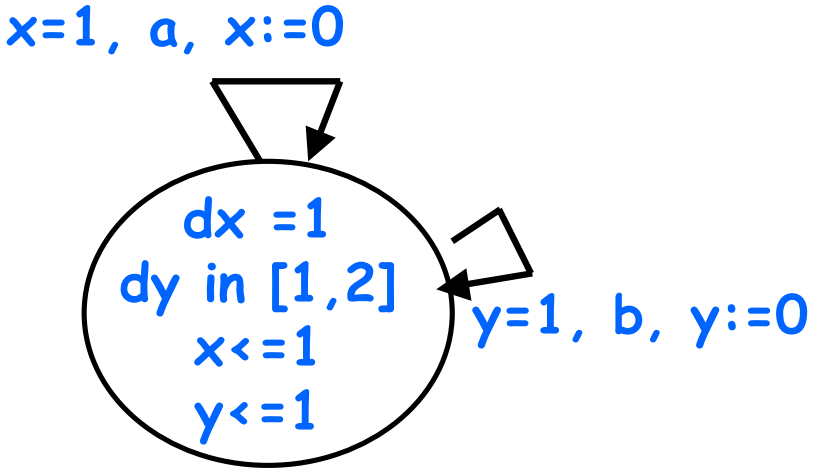


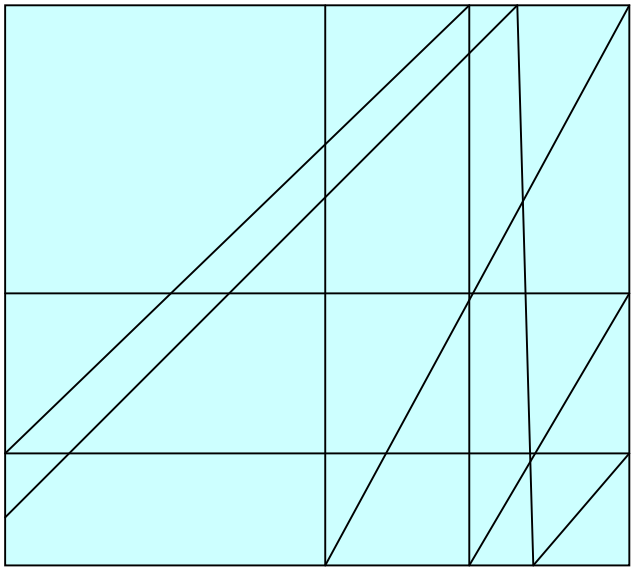
Rectangular Automata

- Interesting extension of timed automata
  - Dynamics of the form *dx* in const interval (rate-bounds of a clock same in all locations)
  - Guards/invariants/resets as before
- Translation to multi-rate automata that gives time-abstract language-equiv system



Rectangular Automata may not have finite bismilar quotients!

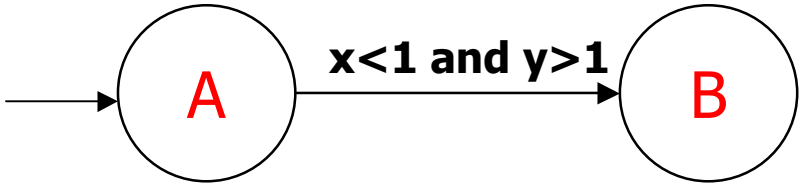




Decidable Problems

- Model checking branching-time properties (TCTL) of timed automata
- Reachability in rectangular automata
- Timed bisimilarity: are given two timed automata bisimilar?
- Optimization: Compute shortest paths (e.g. minimum time reachability) in timed automata with costs on locations and edges
- Controller synthesis: Computing winning strategies in timed automata with controllable and uncontrollable transitions

Limit Reachability



- Given A and error ε, define A^{ε} to be the rectangular automaton in which every clock x has rate in the interval $[1-\varepsilon, 1+\varepsilon]$
- A location l is limit reachable if l is reachable in A^{ε} for every $\varepsilon > 0$
- Limit reachability is decidable

Undecidable Reachability Problems

- Linear expressions as guards
- Guards that compare clocks with irrational constants
- Updates of the form x := x-1
- Multi-rate automata with comparisons among clocks as guards
- Timed automata + stop-watches (i.e. clocks that can have rates 0 or 1)