Modeling & Control of Hybrid Systems

Chapter 7 — Model Checking and Timed Automata

Overview

1. Introduction
2. Transition systems
3. Bisimulation
4. Timed automata

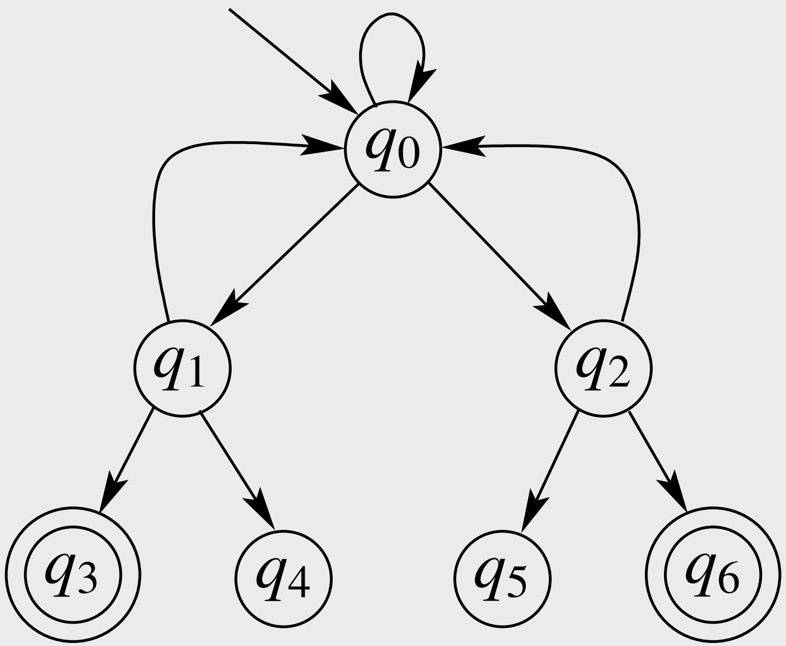
1. Introduction

- *Model checking* = process of automatically analyzing properties of systems by exploring their state space
- Finite state systems → properties can be investigated by systematically exploring states
  E.g., check whether particular set of states will be reached
- Not possible for hybrid systems since number of states is infinite
- However, for some hybrid systems one can find "equivalent" finite state system by partitioning state space into finite number of sets such that any two states in set exhibit similar behavior
  → analyze hybrid system by working with sets of partition
- Generation and analysis of finite partition can be carried out by computer

2. Transition systems

- *Transition system* $T = (S, \delta, S_0, S_F)$ consists of
  - set of states S (finite or infinite)
  - transition relation $\delta : S \to P(S)$
  - set of initial states $S_0 \subseteq S$
  - set of final states $S_F \subseteq S$
- *Trajectory* of transition system is (in)finite sequence of states $\{s_i\}_{i=0}^{N}$ such that
  - $s_0 \in S_0$
  - $s_{i+1} \in \delta(s_i)$ for all i

Example of finite state transition system



- States: $S = \{q_0, \ldots, q_6\}$;
- Transition relation: $\delta(q_0) = \{q_0, q_1, q_2\}$, $\delta(q_1) = \{q_0, q_3, q_4\}$, $\delta(q_2) = \{q_0, q_5, q_6\}$, $\delta(q_3) = \delta(q_4) = \delta(q_5) = \delta(q_6) = \varnothing$
- Initial states: $S_0 = \{q_0\}$
- Final states: $S_F = \{q_3, q_6\}$ (indicated by double circles)

Transition system of hybrid automaton

- Hybrid automaton can be transformed into transition system by abstracting away time
- Consider hybrid automaton $H = (Q, X, \mathsf{Init}, f, \mathsf{Inv}, E, G, R)$ and "final" set of states $F \subseteq Q \times X$
- Define
  - $S = Q \times X$, i.e., $s = (q, x)$
  - $S_0 = \mathsf{Init}$
  - $S_F = F$
  - transition relation δ consists of two parts:
    * discrete transition relation δ_e for each edge $e = (q, q') \in E$:

$$\delta_e(\hat{q}, \hat{x}) = \begin{cases} \{q'\} \times R(e, \hat{x}) & \text{if } \hat{q} = q \text{ and } \hat{x} \in G(e) \\ \emptyset & \text{if } \hat{q} \neq q \text{ or } \hat{x} \notin G(e) \end{cases}$$

Transition system of hybrid automaton (cont.)

* continuous transition relation δ_C:

$$\delta_C(\hat{q},\hat{x}) = \{(\hat{q}',\hat{x}') \mid \hat{q}' = \hat{q} \text{ and } \exists t_{\mathsf{f}} \geqslant 0,\ x(t_{\mathsf{f}}) = \hat{x}' \wedge \forall t \in [0,t_{\mathsf{f}}], x(t) \in \mathsf{Inv}(\hat{q})\}$$

where $x(\cdot)$ is solution of

$$\dot{x} = f(\hat{q},x) \text{ with } x(0) = \hat{x}$$

* Overall transition relation is then

$$\delta(s) = \delta_C(s) \cup \bigcup_{e \in E} \delta_e(s)$$

$\rightarrow$ transition from s to s' is possible if either discrete transition $e \in E$ of hybrid system brings s to s', or s can flow continuously to s' after *some* time

Transition system of hybrid automaton (cont.)

- Time has been abstracted away:
  we do not care how long it takes to get from s to s', we only care whether it is possible to get there eventually

$\rightarrow$ transition system captures sequence of events that hybrid system may experience, but *not* timing of these events

Reachability

- Transition system is *reachable* if there exists trajectory such that $s_i \in S_F$ for some i
- Predecessor operator $\mathsf{Pre} : P(S) \to P(S)$ defined as

$$\mathsf{Pre}(\hat{S}) = \{s \in S \mid \exists \hat{s} \in \hat{S} \text{ with } \hat{s} \in \delta(s)\}$$

$\to$ Pre gives set of states that can reach $\hat{S}$ in one transition

- **Algorithm 1 (Backwards Reachability)**

  initialization: $W_0 = S_F$, $i = 0$
  repeat
      if $W_i \cap S_0 \neq \varnothing$
          return "S_F reachable"
      end if
      $W_{i+1} = \mathsf{Pre}(W_i) \cup W_i$
      $i = i + 1$
  until $W_i = W_{i-1}$
  return "S_F not reachable"

Reachability (cont.)

- Problem: if new states get added to W_i each time we go around repeat-until loop $\rightarrow$ algorithm does not terminate

  Example:

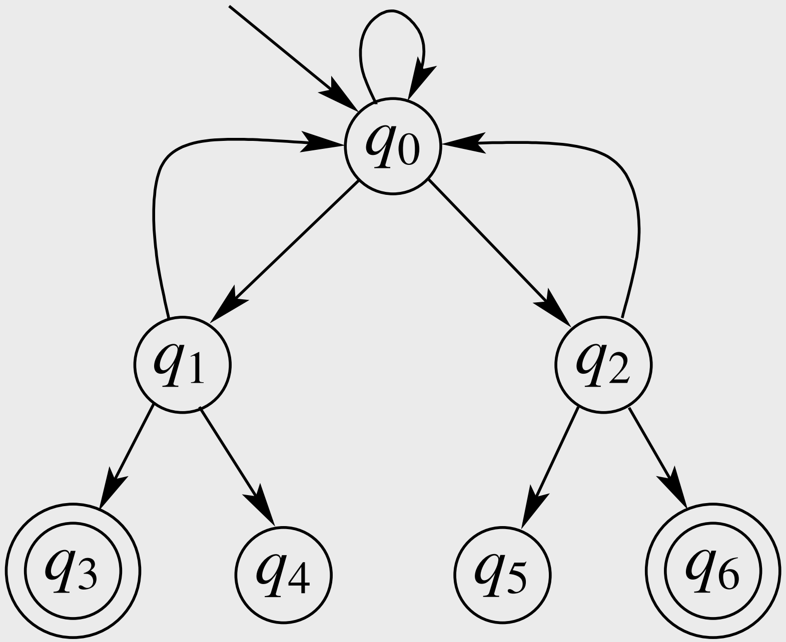
  - $T = (S, \delta, S_0, S_F)$ with $S = \mathbb{R}$, $\delta(x) = 2x$, $S_0 = \{-1\}$, $S_F = \{1\}$
  - Backwards Reachability algorithm produces

$$W_0 = \{1\}, W_1 = \{1, \frac{1}{2}\}, \ldots, W_i = \{1, \frac{1}{2}, \ldots, \left(\frac{1}{2}\right)^i\}, \ldots$$

  $\rightarrow$ algorithm will not terminate

- With finite state systems termination is not a problem

Example of finite state transition system (cont.)



- $W_0 = \{q_3, q_6\}, W_1 = \{q_1, q_2, q_3, q_6\}, W_2 = \{q_0, q_1, q_2, q_3, q_6\}$
- $W_2 \cap S_0 = \{q_0\} \neq \varnothing$
  $\rightarrow$ after 2 steps algorithm terminates with answer "S_F reachable"

3. Bisimulation

- Turn *infinite* state system into *finite* state system by grouping together states that have "similar" behavior $\rightarrow$ partition
- *Partition* is collection of sets of states $\{S_i\}_{i \in I}$ with $S_i \subseteq S$ and $S_i \neq \varnothing$ such that

  1. any two sets S_i and S_j in partition are disjoint
  2. union of all sets in partition is entire state space, i.e., $\bigcup_{i \in I} S_i = S$

- *Finite partition*: if I is finite set
- Examples:

  Partition: $\{q_0\}, \{q_1, q_2\}, \{q_3, q_6\}, \{q_4, q_5\}$

  No partitions: $\{q_1, q_3, q_4\}, \{q_2, q_5, q_6\}$

  $\{q_0, q_1, q_3, q_4\}, \{q_0, q_2, q_5, q_6\}$

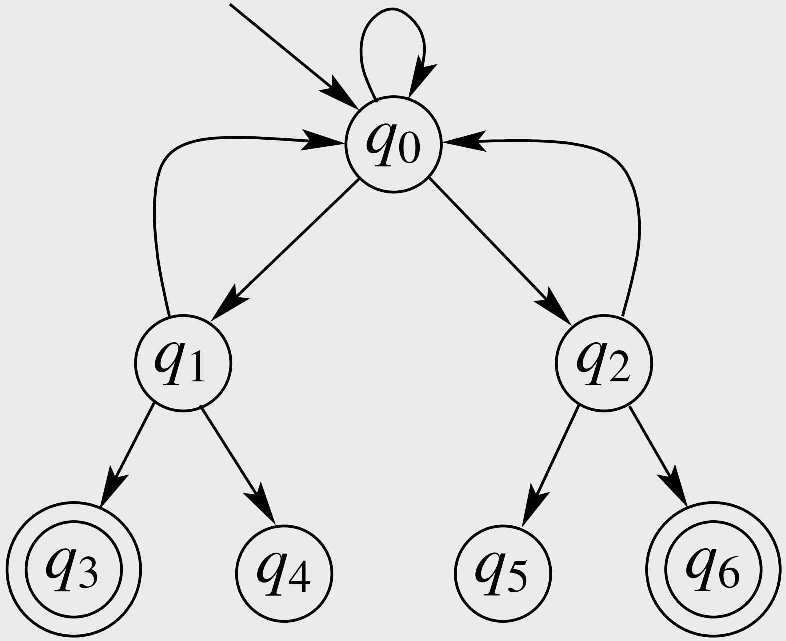
Quotient transition system

- Given transition system $T = (S, \delta, S_0, S_F)$, and partition $\{S_i\}_{i \in I}$
- *Quotient transition system* $\hat{T} = (\hat{S}, \hat{\delta}, \hat{S}_0, \hat{S}_F)$ is defined as
  - $\hat{S} = \{S_i\}_{i \in I}$, i.e., states are sets of partition
  - $\hat{\delta}$ allows transition from set S_i to S_j if and only if δ allows a transition from some state $s \in S_i$ to some state $s' \in S_j$
  - S_i is in initial set of $\hat{T}$ if and only if some element $s \in S_i$ is initial state of original transition system
  - S_i is final set of $\hat{T}$ if and only if some element of $s \in S_i$ is final state of original transition system
- If partition is finite, then quotient transition system $\hat{T}$ is finite state system $\rightarrow$ can be easily analyzed

Quotient transition system (cont.)

- Problem: for most partitions properties of quotient transition system do not allow to draw any useful conclusions about properties of original system
- However, special type of partition for which quotient system $\hat{T}$ is "equivalent" to original transition system T: *bisimulation*
- A bisimulation of transition system $T = (S, \delta, S_0, S_F)$ is partition $\{S_i\}_{i \in I}$ such that
  - S_0 is a union of elements of the partition
  - S_F is a union of elements of the partition
  - if one state s in some set S_i of the partition can transition to another set S_j in the partition, then *all* other states $\hat{s} \in S_i$ must be able to transition to some state in S_j

Example of bisimulation



- $\{q_0\}, \{q_1, q_2\}, \{q_3, q_6\}, \{q_4, q_5\}$ is bisimulation:
  - $S_0 = \{q_0\}$ which is an element of the partition
  - $S_F = \{q_3, q_6\}$ which is also an element of the partition
  - Consider, e.g., set $\{q_1, q_2\}$
    From q_1 one can jump to $\{q_0\}, \{q_3, q_6\}, \{q_4, q_5\}$
    From q_2 one can jump to exactly these same sets
    $\rightarrow$ third condition is satisfied for set $\{q_1, q_2\}$
    + also satisfied for other sets
- $\{q_0\}, \{q_1, q_3, q_4\}, \{q_2, q_5, q_6\}$ not bisimulation (S_F;$q_1 \rightarrow q_0$ but $q_3 \not\rightarrow q_0$)

Important property

If $\{S_i\}_{i \in I}$ is bisimulation of transition system T and $\hat{T}$ is quotient transition system, then S_F is reachable by T if and only if $\hat{S}_F$ is reachable by $\hat{T}$

- For finite state systems $\rightarrow$ computational efficiency
  Study reachability in quotient system instead of original system
  (quotient system usually much smaller than original)
- For infinite state systems:
  Even if original transition system has infinite number of states, sometimes bisimulation consisting of finite number of sets
  $\rightarrow$ answer reachability questions for infinite state system by studying equivalent finite state system
- For timed automata we can always find *finite* bisimulation

Bisimulation algorithm

Algorithm 2 (Bisimulation)

initialization: $\mathscr{P} = \{S_0, S_F, S \setminus (S_0 \cup S_F)\}$
while $\exists S_i, S_j \in \mathscr{P}$ such that $S_i \cap \mathsf{Pre}(S_j) \neq \varnothing$ and $S_i \cap \mathsf{Pre}(S_j) \neq S_i$ **do**
  $S_i' = S_i \cap \mathsf{Pre}(S_j)$
  $S_i'' = S_i \setminus \mathsf{Pre}(S_j)$
  $\mathscr{P} = (\mathscr{P} \setminus S_i) \cup \{S_i', S_i''\}$
end while
return $\mathscr{P}$

- Algorithm maintains partition $\mathscr{P}$ that gets refined progressively so that it looks more and more like a bisimulation
- From definition of bisimulation we deduce that bisimulation must at least allow us to “distinguish” initial and final states.
  $\rightarrow$ start with partition containing 3 sets: S_0, S_F, and everything else

Bisimulation algorithm (cont.)

Algorithm 3 (Bisimulation)

initialization: $\mathscr{P} = \{S_0, S_F, S \setminus (S_0 \cup S_F)\}$
while $\exists S_i, S_j \in \mathscr{P}$ such that $S_i \cap \mathsf{Pre}(S_j) \neq \emptyset$ and $S_i \cap \mathsf{Pre}(S_j) \neq S_i$ **do**
$S_i' = S_i \cap \mathsf{Pre}(S_j)$
$S_i'' = S_i \setminus \mathsf{Pre}(S_j)$
$\mathscr{P} = (\mathscr{P} \setminus S_i) \cup \{S_i', S_i''\}$
end while
return $\mathscr{P}$

- Assume we can find two sets $S_i, S_j \in \mathscr{P}$ such that $\mathsf{Pre}(S_j)$ contains some elements of S_i but not all of them
  $\rightarrow$ some states $s \in S_i$ may find themselves in S_j after one transition while others do not
  $\rightarrow$ not allowed if $\mathscr{P}$ is to be bisimulation
  $\rightarrow$ replace S_i by two sets: states in S_i that can transition to S_j
  states in S_i that cannot transition to S_j

Bisimulation algorithm (cont.)

- If bisimulation algorithm terminates, it will produce the coarsest bisimulation of the transition system (i.e., bisimulation containing *smallest* number of sets)
- For finite state systems bisimulation algorithm is easy to implement (by enumerating the states) and will always terminate
- Problem: it may be more work to find bisimulation than to investigate reachability of the original system
- For infinite state systems: sometimes, algorithm may never terminate (reason: not all infinite state transition systems have finite bisimulations)
- But for *timed automata*: bisimulation algorithm terminates in finite number of steps

4. Timed automata

- Timed automata involve simple continuous dynamics:
  - all differential equations of form $\dot{x} = 1$,
  - all invariants, guards, etc. involve comparison of real-valued states with constants (e.g., $x = 1$, $x < 2$, $x \geqslant 0$, etc.)
- Timed automata are limited for modeling physical systems
- However, very well suited for encoding timing constraints such as "event A must take place at least 2 seconds after event B and not more than 5 seconds before event C"
- Applications: multimedia, Internet, audio protocol verification

4.1 Rectangular sets

- Subset of $\mathbb{R}^n$ set is called rectangular if can be written as finite boolean combination of constraints of form

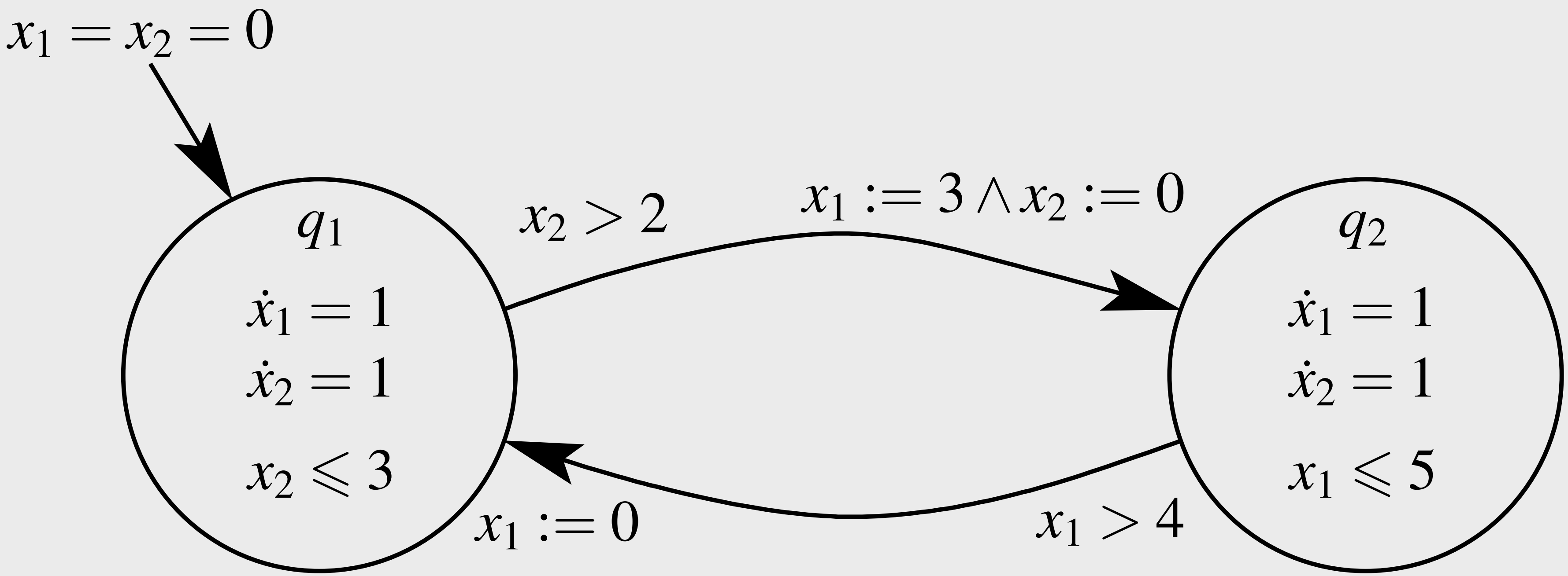
$$x_i \leqslant a, \;\; x_i < b, \;\; x_i = c, \;\; x_i \geqslant d, \;\; x_i > e$$

- Rectangular sets are “rectangles” or “boxes” in $\mathbb{R}^n$ whose sides are aligned with the axes, or unions of such rectangles/boxes
- Examples:
  - $\{(x_1,x_2) \mid (x_1 \geqslant 0) \wedge (x_1 \leqslant 2) \wedge (x_2 \geqslant 1) \wedge (x_2 \leqslant 2)\}$
  - $\{(x_1,x_2) \mid ((x_1 \geqslant 0) \wedge (x_2 = 0)) \vee ((x_1 = 0) \wedge (x_2 \geqslant 0))\}$
  - empty set (e.g., $\varnothing = \{(x_1,x_2) \mid (x_1 > 1) \wedge (x_1 \leqslant 0))\}$
- However, set $\{(x_1,x_2) \mid x_1 = 2x_2\}$ is not rectangular

4.2 Timed automaton

- Timed automaton is hybrid automaton with following characteristics:
  - automaton involves differential equations of form $\dot{x}_i = 1$; continuous variables governed by this differential equation are called “clocks” or “timers”
  - sets involved in definition of initial states, guards, and invariants are rectangular sets
  - reset maps involve either rectangular set, or may leave certain states unchanged

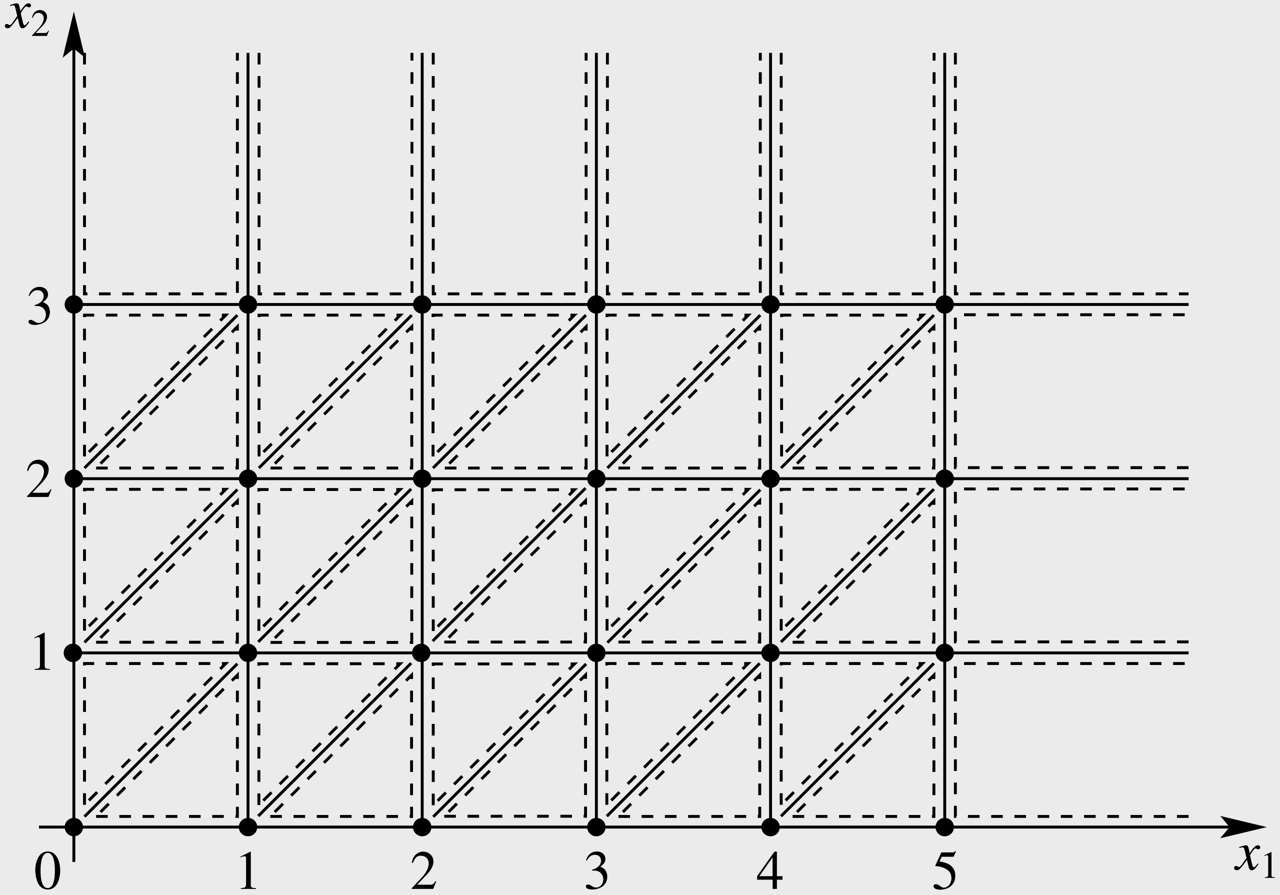
4.3 Example of timed automaton

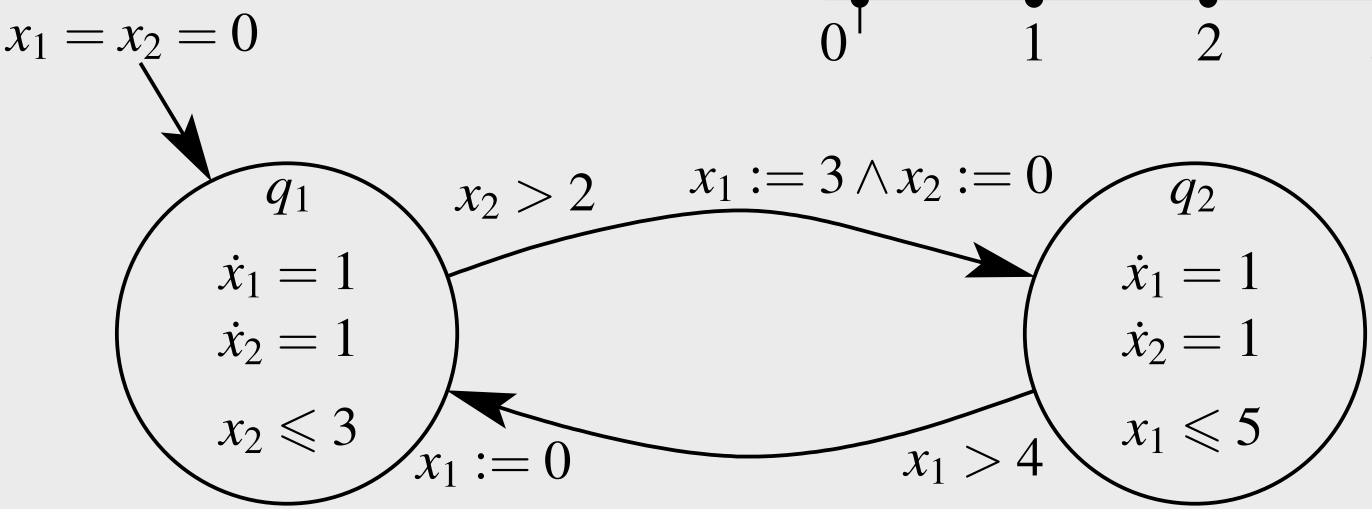


Timed automata (cont.)

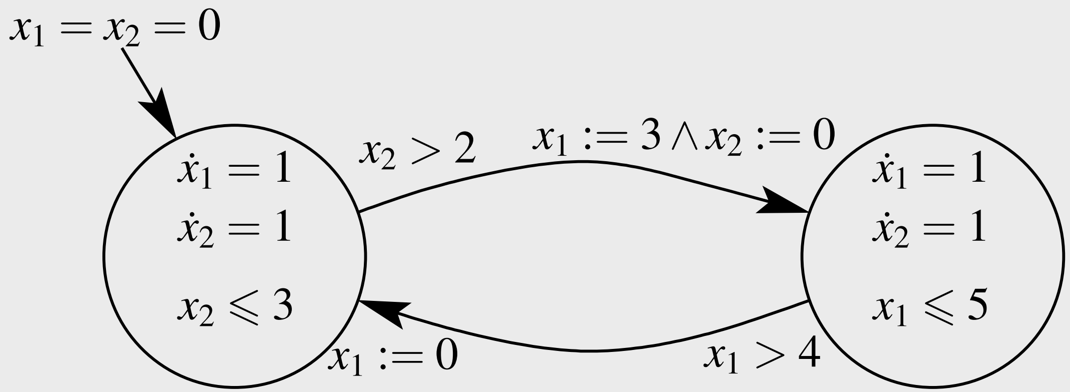
- For timed automaton of example: all constants are non-negative integers
  $\rightarrow$ can be generalized
- Given any timed automaton whose definition involves rational and/or negative constants, we can define an equivalent timed automaton whose definition involves only non-negative integers
  Done by “scaling” and “shifting” (adding appropriate integer) some of states
- Transformation into transition systems
  $\rightarrow$ transition system corresponding to timed automaton always has finite bisimulation
- Standard bisimulation for timed automata is *region graph*

Region graph



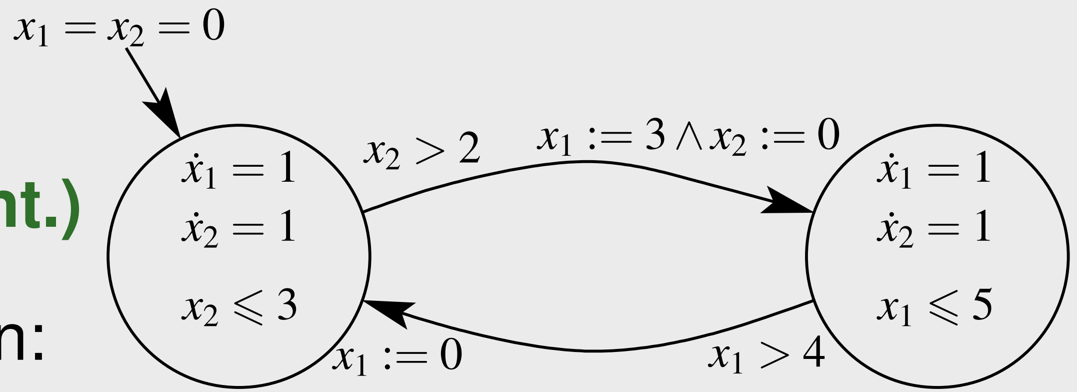


Construction of region graph



- Assume w.l.o.g. that all constants are non-negative integers
- Let C_i be largest constant with which x_i is compared in initial sets, guards, invariants and resets
  In example: $C_1 = 5$ and $C_2 = 3$
- If all we know about timed automaton is these bounds C_i, then x_i *could* be compared with any integer $M \in \{0, 1 \ldots, C_i\}$ in some guard, reset or initial condition set
- Hence, discrete transitions of timed automaton may be able to "distinguish" states with $x_i < M$ from states with $x_i = M$ and from states with $x_i > M$ (e.g., discrete transition may be possible from state with $x_i < M$ but not from state with $x_i > M$)

Construction of region graph (cont.)



- Add sets to candidate bisimulation:

for x_1: $x_1 \in (0,1), x_1 \in (1,2), x_1 \in (2,3), x_1 \in (3,4), x_1 \in (4,5), x_1 \in (5,\infty)$

$$x_1 = 0, x_1 = 1, x_1 = 2, x_1 = 3, x_1 = 4, x_1 = 5$$

for x_2: $x_2 \in (0,1), x_2 \in (1,2), x_2 \in (2,3), x_2 \in (3,\infty)$

$$x_2 = 0, x_2 = 1, x_2 = 2, x_2 = 3$$

- Products of all sets:

$\{x \in \mathbb{R}^2 \mid x_1 \in (0,1) \wedge x_2 \in (0,1)\}$ $\quad$ $\{x \in \mathbb{R}^2 \mid x_1 \in (0,1) \wedge x_2 = 1\}$

$\{x \in \mathbb{R}^2 \mid x_1 = 1 \wedge x_2 \in (0,1)\}$ $\quad$ $\{x \in \mathbb{R}^2 \mid x_1 = 1 \wedge x_2 = 1\}$

$\{x \in \mathbb{R}^2 \mid x_1 \in (1,2) \wedge x_2 \in (3,\infty)\}$, etc.

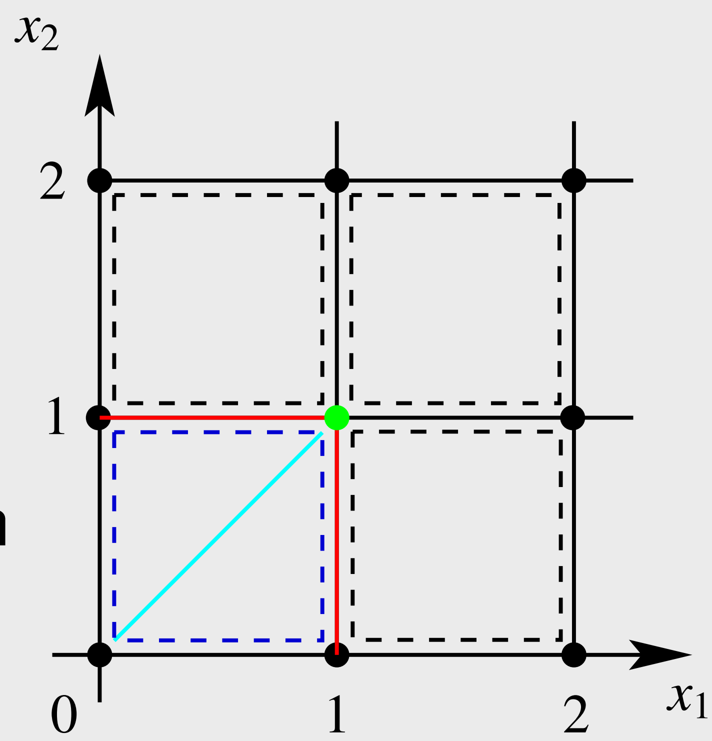
define all sets in $\mathbb{R}^2$ that discrete dynamics can distinguish

$\rightarrow$ open squares, open horizontal and vertical line segments, integer points, and open, unbounded rectangles

Construction of region graph (cont.)

- Since $\dot{x}_1 = \dot{x}_2 = 1$, continuous states move diagonally up along $45°$ lines

$\rightarrow$ by allowing time to flow timed automaton may distinguish points below diagonal of each square, points above diagonal, and points on the diagonal



- E.g., points above diagonal of square

$$\{x \in \mathbb{R}^2 \mid x_1 \in (0,1) \wedge x_2 \in (0,1)\}$$

will leave square through line $\{x \in \mathbb{R}^2 \mid x_1 \in (0,1) \wedge x_2 = 1\}$
Points below diagonal leave square through line

$$\{x \in \mathbb{R}^2 \mid x_1 = 1 \wedge x_2 \in (0,1)\}$$

Points on diagonal leave square through point $(1,1)$

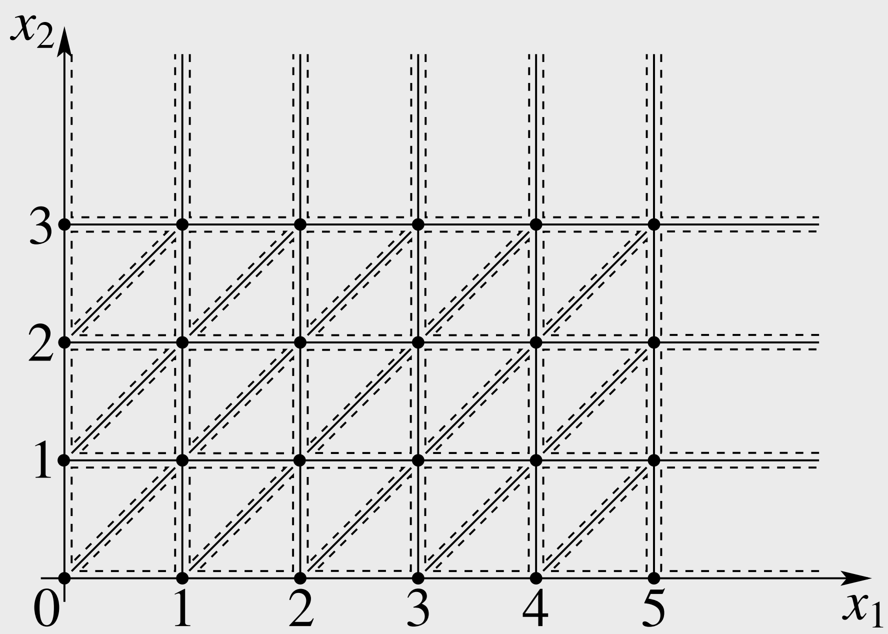
Construction of region graph (cont.)

- Split each open square in three: two open triangles and open diagonal line segment

→ is enough to generate bisimulation:

Theorem:

The region graph is finite bisimulation of timed automaton



- Disadvantage: total number of regions in the region graph grows very quickly (exponentially) as n increases

5. Summary

- Verification of hybrid systems → hard problem
- Transition systems
- Bisimulation & reachability
  → turn *infinite* state system into *finite* state system by grouping together states that have "similar" behavior
- Timed automata → finite bisimulation