Modeling & Control of Hybrid Systems

Chapter 7 — Model Checking and Timed Automata

Overview

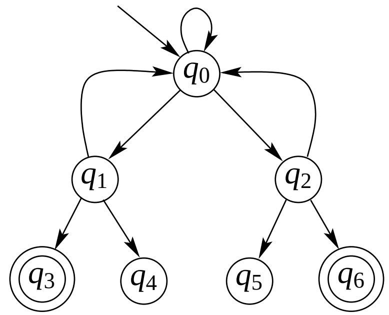
1. Introduction

- *Model checking* = process of automatically analyzing properties of systems by exploring their state space
- Finite state systems $\rightarrow$ 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
  $\rightarrow$ 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 $\delta : S \rightarrow P(S)$
  - 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 $e = (q, q') \in E$:

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

Transition system of hybrid automaton (cont.)

$*$ continuous transition relation δ_C:

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

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

3. Bisimulation

- Turn *infinite* state system into *finite* state system by grouping together states that have "similar" behavior $\rightarrow$ partition
- Yields so-called quotient transition system
  finite number of states $\rightarrow$ can be analyzed more easily
- 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*

Important property

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

Bisimulation algorithm

- For timed automata we can always find *finite* bisimulation
- Bisimulation algorithm (see lecture notes):
  - For finite state systems bisimulation algorithm 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)

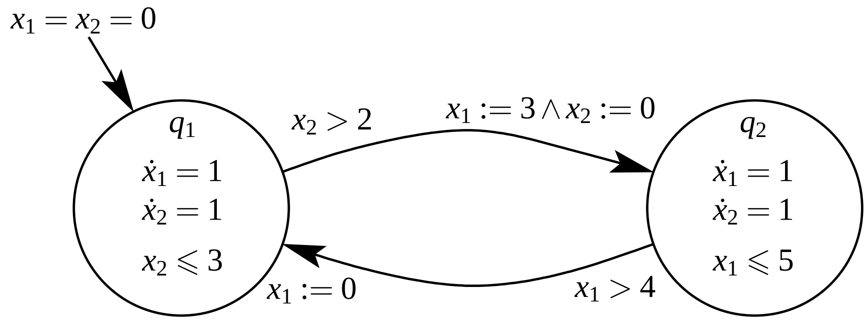
Bisimulation algorithm (continued)

- For *timed automata*: bisimulation algorithm terminates in finite number of steps
  Disadvantage: total number of states in the quotient transition system grows very quickly (exponentially) as number of timers n increases

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

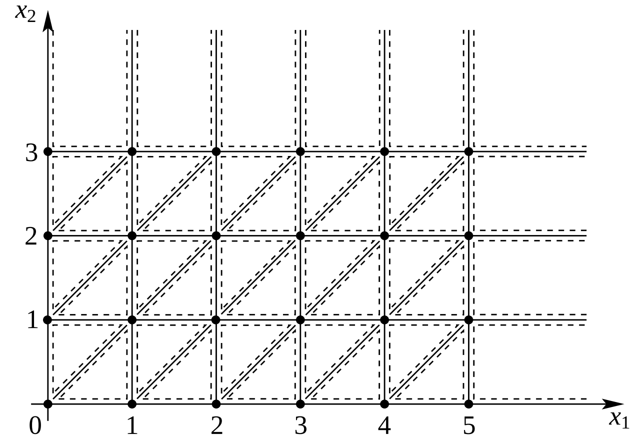
4.1 Example of timed automaton

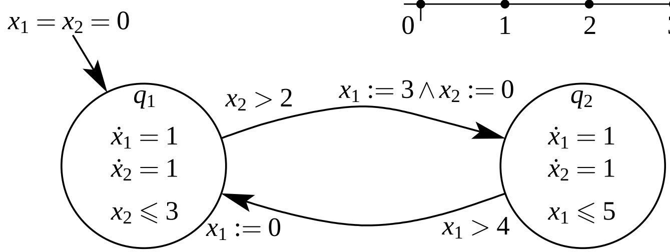


Timed automata (cont.)

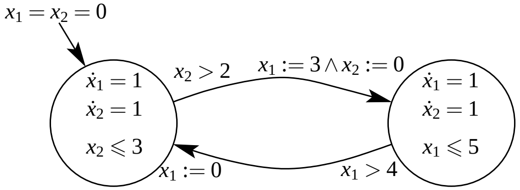
- For timed automaton of example: all constants are non-negative integers
  $\rightarrow$ can be generalized
- Given any timed automaton whose definition involves rational and/or negative constants, we can define an equivalent timed automaton whose definition involves only non-negative integers
  Done 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



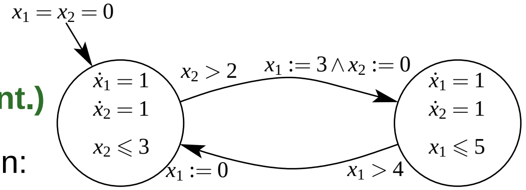


Construction of region graph



- 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:

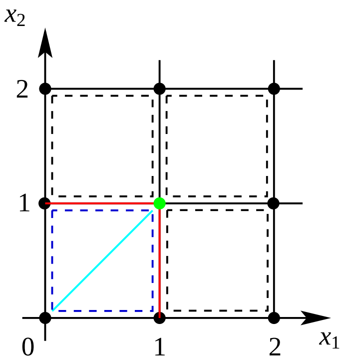
  for $x_1 : x_1 \in (0,1), x_1 \in (1,2), x_1 \in (2,3), x_1 \in (3,4), x_1 \in (4,5), x_1 \in (5,\infty)$
  $x_1 = 0, x_1 = 1, x_1 = 2, x_1 = 3, x_1 = 4, x_1 = 5$
  for $x_2 : x_2 \in (0,1), x_2 \in (1,2), x_2 \in (2,3), x_2 \in (3,\infty)$
  $x_2 = 0, x_2 = 1, x_2 = 2, x_2 = 3$

- Products of all sets:

  $\{x \in \mathbb{R}^2 \mid x_1 \in (0,1) \wedge x_2 \in (0,1)\}$ $\{x \in \mathbb{R}^2 \mid x_1 \in (0,1) \wedge x_2 = 1\}$
  $\{x \in \mathbb{R}^2 \mid x_1 = 1 \wedge x_2 \in (0,1)\}$ $\{x \in \mathbb{R}^2 \mid x_1 = 1 \wedge x_2 = 1\}$
  $\{x \in \mathbb{R}^2 \mid x_1 \in (1,2) \wedge x_2 \in (3,\infty)\}$, etc.

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

→ 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

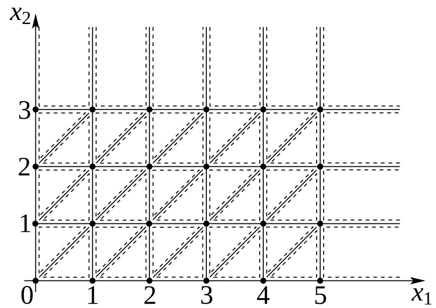
$$\{x \in \mathbb{R}^2 \mid x_1 \in (0,1) \wedge x_2 \in (0,1)\}$$

  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