07_Model Checking and Timed Automata

1. Introduction

Model Checking

Process of automatically analyzing properties of systems by exploring their state space

Note:

- 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

2. Transition Systems

Conceptions

Transition System

Transition system $T=(S, \delta, S_0, S_F)$ consists of

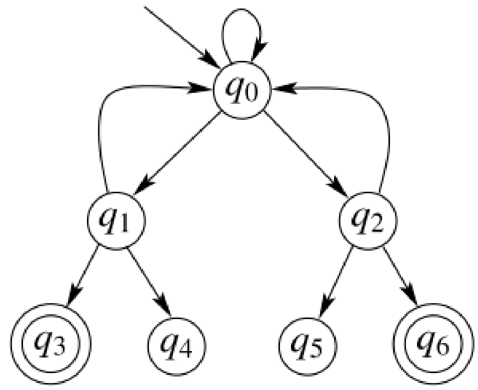
- set of states S (finite or infinite)
- transition relation $\delta: S \rightarrow P(S)$
- set of initial states $S_0 \subseteq S$
- set of final states $S_F \subseteq S$

Trajectory

Trajectory of transition system is (in)finite sequence of states $\{s_i\}_{i=0}^N$ such that

- $s0 \in S_0$
- $s_{i+1} \in \delta\left(s_i\right)$ for all i

Example of finite state transition system



- States: $S = \{q_0, \dots, q_6\}$;
- 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)

hs_check.4

Reachability

Transition system is reachable if there exists trajectory such that $s_i \in S_F$ for some i

Transformation between Hybrid Automaton and Transition System

- Hybrid automaton can be transformed into transition system by **abstracting away time**
  - 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

Method

Consider hybrid automaton $H = (Q, X, \text{Init}, f, \text{Inv}, E, G, R)$ and "final" set of states $F \subseteq Q \times X$

- $S = Q \times X$, i.e., $s = (q, x)$
- $S_0 = \text{Init}$
- $S_F = F$
- Transition δ consists of two parts:
  - discrete transition relation δ_e for each edge $e = (q, q') \in E$

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

- continuous transition relation δ_C

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

- Overall Transition Relation is then

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

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

3. Bisimulation

Conceptions

Definition: Partition

A **partition** is a collection of non-empty 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 the partition are **disjoint**
2. The **union** of all sets in the partition is the **entire state space**, i.e.

$$\bigcup_{i \in I} S_i = S$$

(A family of sets with this property is said to **cover** the state space).

The index set, I, of the partition may be **either finite or infinite**. If I is a finite set (e.g., $I = \{1, 2, \ldots, M\}$ for $M < \infty$) then we say that the partition $\{S_i\}_{i \in I}$ is a **finite partition**.

Definition: Bisimulation

A **bisimulation** of a transition system $T = (S, \delta, S_0, S_\mathrm{f})$ is a partition $\{S_i\}_{i \in I}$ of the state space S of T such that

1. S_0 is a **union of elements** of the partition,
2. S_f is a **union of elements** of the partition,
3. if one state (say s) in some set of the partition (say S_i) can transition to another set in the partition (say S_j), then all other states, $\hat{s}$ **in** S_i **must be able to transition to some state in** S_j. More formally, for all $i, j \in I$ and for all states $s, \hat{s} \in S_i$, if $\delta(s) \cap S_j \neq \varnothing$, then $\delta(\hat{s}) \cap S_j \neq \varnothing$.

i.e., states in the same set have the same transition relations

Notes:

- Turn infinite state system into finite state system by **grouping together states** that have “similar” behavior → <u>partition</u>
- Yields so-called <u>**quotient transition system**</u>
- for most partitions properties of quotient transition system do not allow to draw any useful conclusions about properties of original system
- special type of partition for which quotient system $\hat{T}$ is “equivalent” to original transition system T: <u>**bisimulation**</u>

Bisimulation Property

<u>**Theorem**</u>

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 $\hat{T}$

Note:

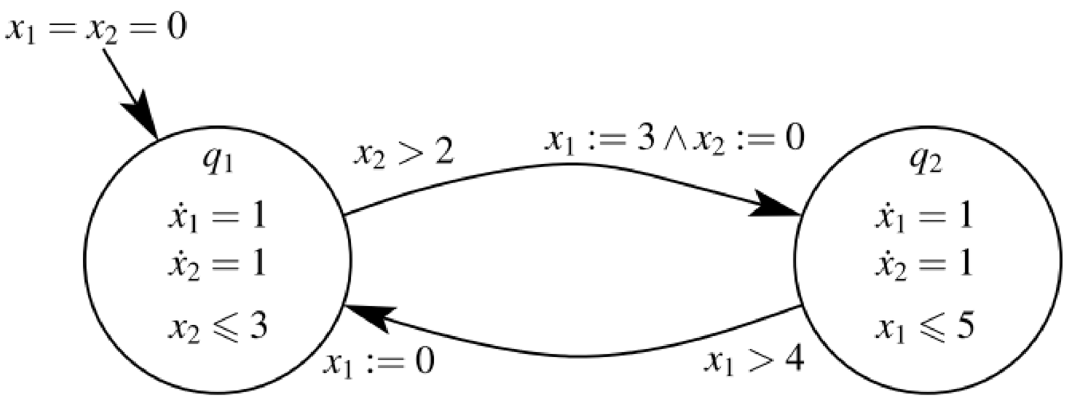
- For finite state systems, use quotient system will have higher computational efficiency
- For infinite state system, we can sometimes bisimulation consisting of finite number of sets

Bisimulation Algorithm

- For **timed automata** we can **always find** finite bisimulation
- For infinite state systems: sometimes, algorithm may never terminate (reason: **not all infinite state transition systems have finite bisimulations**)
- total number of states in the quotient transition system grows very quickly (exponentially) as **number of timers n increases**

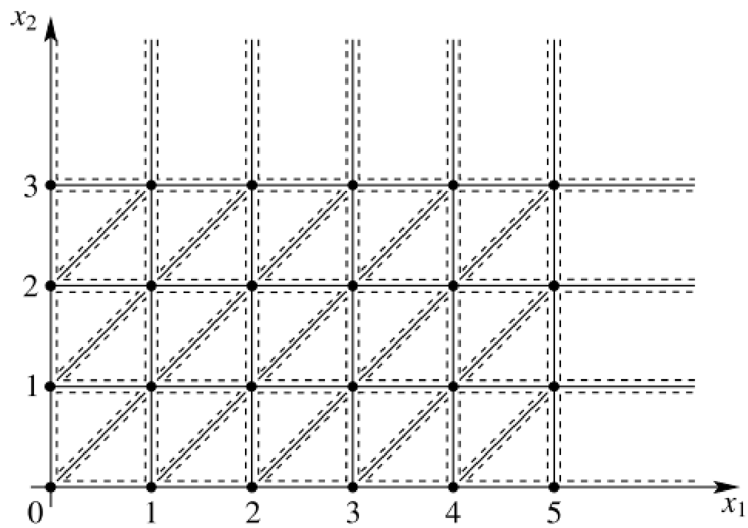
4. Analysis Example of Timed Automata

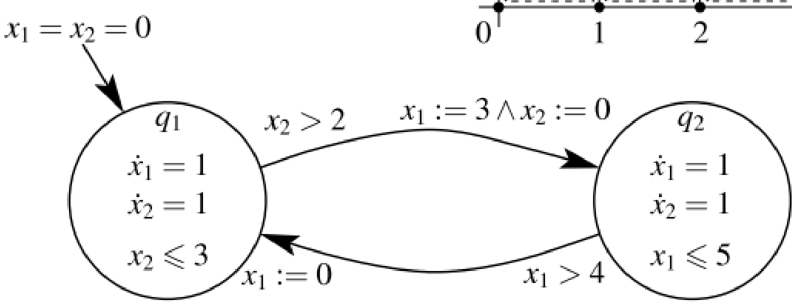
4.1 Example of timed automaton



hs_check.14

Region graph





hs_check.16

- 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 $x_i > M$)

hs_check.17

- Add sets to candidate bisimulation:

$$
\begin{aligned}
\text{for } x_1 : \;& x_1 \in (0,1), x_1 \in (1,2), x_1 \in (2,3), x_1 \in (3,4), x_1 \in (4,5), x_1 \in (5,\infty) \\
& x_1 = 0, x_1 = 1, x_1 = 2, x_1 = 3, x_1 = 4, x_1 = 5 \\
\text{for } x_2 : \;& x_2 \in (0,1), x_2 \in (1,2), x_2 \in (2,3), x_2 \in (3,\infty) \\
& x_2 = 0, x_2 = 1, x_2 = 2, x_2 = 3
\end{aligned}
$$

- Products of all sets:

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

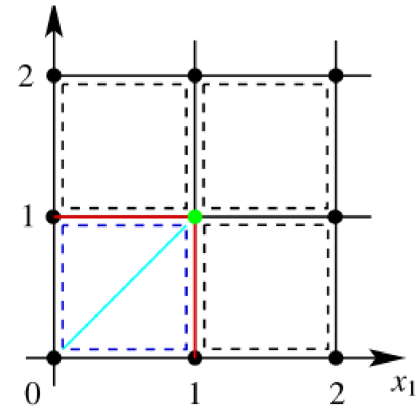
  define all sets in $\mathbb{R}^2$ that discrete dynamics can distinguish
  $\rightarrow$ open squares, open horizontal and vertical line segments, integer points, and open, unbounded rectangles

hs_check.18

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 $(1,1)$

hs_check.19

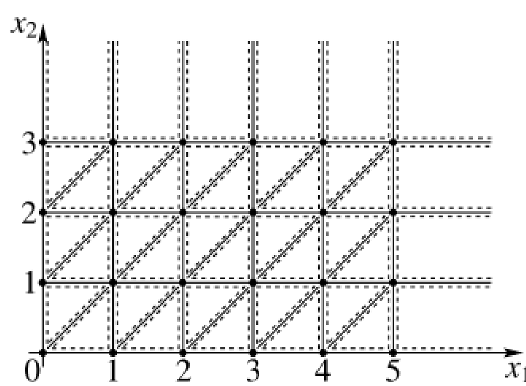
Construction of region graph (cont.)

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

$\rightarrow$ 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

Summary

- Verification of Hybrid System: **reachable problem** $\rightarrow$ hard problem

- Transition Systems: Hybrid Automata → Transition Systems
    - transition/edge transformation
- Bisimulation & Reachability
    - bisimulation → terminal state same reachability
    - turn infinite state system into finite state system by grouping together states that have “similar” behavior
    - **Timed automata → finite bisimulation**