TIMED AUTOMATA

LECTURE 10

GOALS OF TODAY'S LECTURE

- Emptiness problem revisited
- Introduction to the algorithm used in practice.

Networks of T.A.

$\boxed{\text{T.A.}} \parallel \boxed{\text{T.A.}} \parallel \boxed{\text{T.A.}}$

Tools:

- UPPAAL
- T.Checker
- PAT

Today: Given one T.A. $\mathcal{A}$ how to check if $\mathcal{L}(\mathcal{A}) =$ empty?

Proviso: Since we are interested only in the emptiness problem, we can forget the exact alphabet.

- All we need is to check if a given T.A. has an accepting run.

- So we will consider T.A. without Σ.

Edges will be of the form: $\boxed{q \xrightarrow[R]{g} q'}$

- Emptiness problem will be called as the:

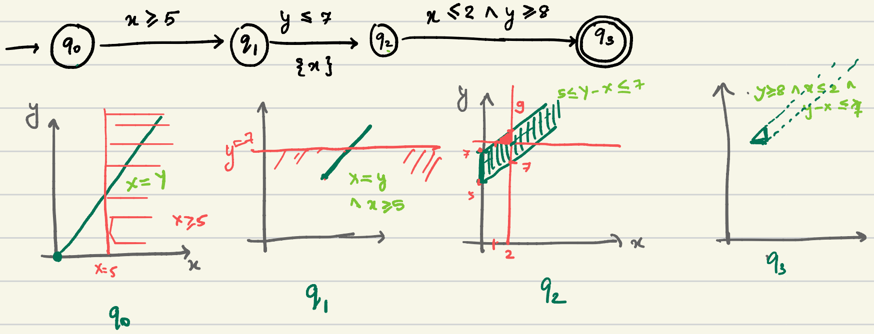
Reachability Problem

Given a T.A. $\mathcal{A}$, does there exist an accepting run?

---

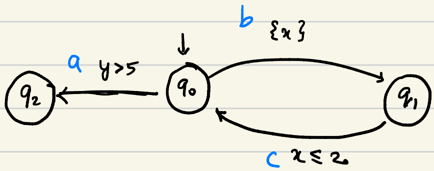
1. Motivating example (2 clocks, computation of the sets)
2. One more example (2 clocks)
3. Another example (3 clocks)
4. Zones and zone graph
5. Soundness and completeness
6. Zone graph could be infinite.
7. How to get a finite graph that preserves reachability?

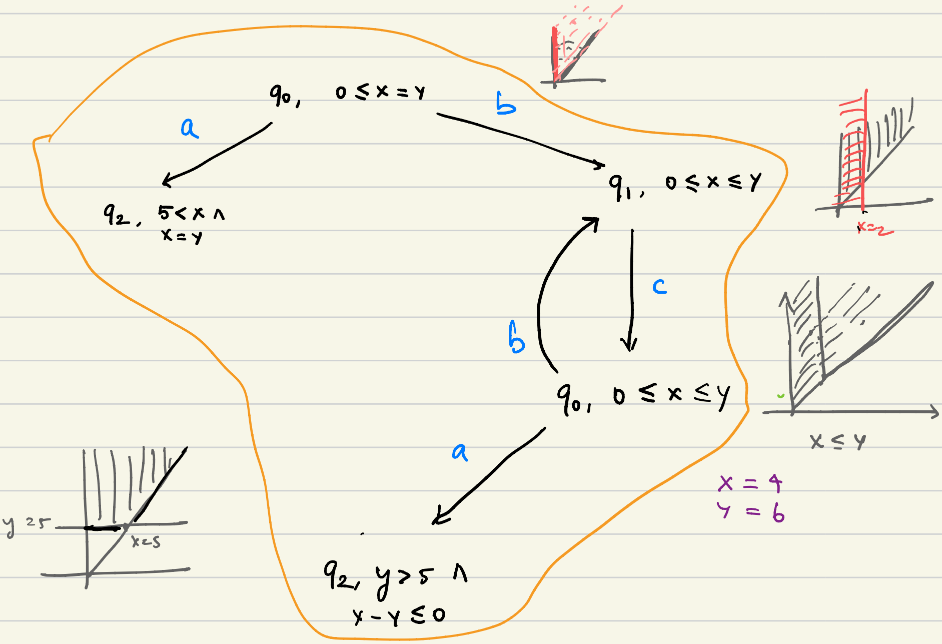
Example 1:



q0
x ≥ 5
q1
y ≤ 7
{x}
q2
x ≤ 2 ∧ y ≥ 8
q3
y
x
x = y
x ≥ 5
x = 5
q0
y = 7
x = y
∧ x ≥ 5
q1
5 ≤ y − x ≤ 7
q2
y ≥ 8 ∧ x ≤ 2 ∧
y − x ≤ 7
q3

Example 2:



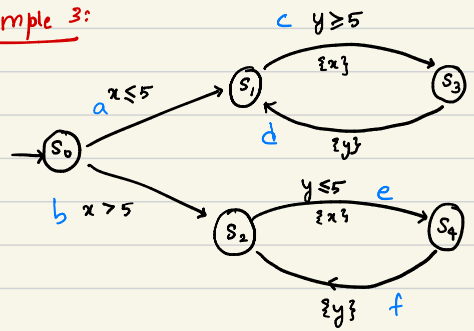


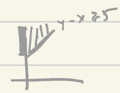
Question: $(q_0, \langle 0, 0\rangle) \xrightarrow{\delta_1} \xrightarrow{b} \xrightarrow{\delta_2} \xrightarrow{c} \xrightarrow{\delta_3} (q_0, \langle 4, 6\rangle)$

Find the $\delta_1, \delta_2, \delta_3$

$\delta_1 = 2$	$\delta_1 = 2$
$\delta_2 = 0$	$\delta_2 = 2$
$\delta_3 = 4$	$\delta_3 = 2$

Example 3:

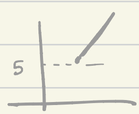




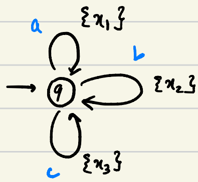
$s_0,\ 0 \leq x = y \xrightarrow{a} s_1,\ 0 \leq x = y \xrightarrow{c} s_3,\ y - x \geq 5 \xrightarrow{d} s_1,\ y - x \geq 5$

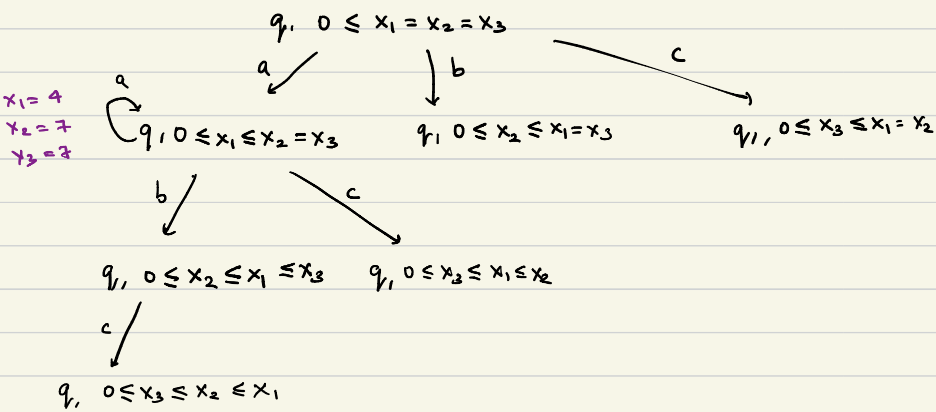
$s_1,\ y - x \geq 5 \xrightarrow{c} s_3,\ y - x \geq 5$

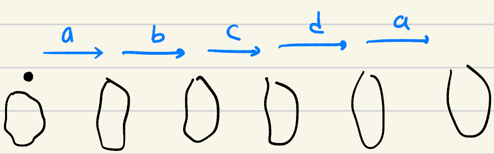
$s_0,\ 0 \leq x = y \xrightarrow{b} s_2,\ 5 < x = y \not\rightarrow$ disabled



Example 4:







- So far these sets were representable using simple constraints like $x - y \le 5 \wedge y \ge 2$, etc. that dealt with difference between two clocks or a comparison constraint with a single clock.

- We did not see constraints like $x + y < 2$ or $x_1 - x_2 + x_3 \ge 7$, etc.

Zones :

Let X be a set of clocks.

A zone is a set of valuations represented using a conjunction of constraints of the form:

$x - y \sim c$	$x, y \in X$
	$\sim \in \{<, \leq, =, >, \geq\}$
$x \sim c$	$c \in \mathbb{N}$

Remark: Zones can be efficiently represented using Difference-Bound-Matrices

$$x_1 - x_2 \leq 5 \;\wedge\; x_2 - x_3 \leq 3 \;\wedge\; x_1 - x_2 \leq 2$$

	x_1	x_2	x_3
x_1	≤ 0	≤ 5	–
x_2	–	≤ 0	≤ 3
x_3	≤ 2	–	≤ 0

(more in detail later)

Zone graph:

Nodes: (q, Z) where Z is a zone

Initial node: $\left(q_0, \bigwedge_{x,y \in X} (x = y \wedge x \geq 0)\right)$

Edges. For every $q \xrightarrow[R]{g} q'$ in the automaton,

there is an edge:

$$(q, Z) \longrightarrow (q', Z')$$

where:

$$Z' = \overrightarrow{[R][Z \cap g]}$$

$$Z \xrightarrow{g} Z \cap g \xrightarrow{R} [R](Z \cap g) \xrightarrow{\text{time elapse}} Z'$$

Remark: Each operation above preserves zones.

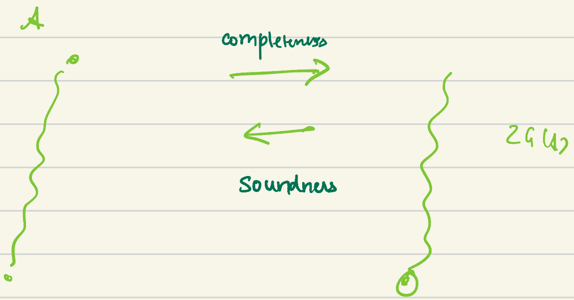
Accepting nodes: (q, Z) where q is an accepting state.

Zone graph is sound and complete for reachability:

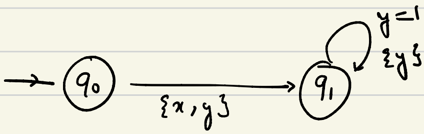
$$\mathcal{A} \longrightarrow ZG(\mathcal{A})$$

T.A.

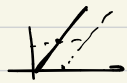
- Compute $ZG(\mathcal{A})$
- if there is a path to an accepting node, declare $\mathcal{L}(\mathcal{A}) \neq \phi$
- else declare $\mathcal{L}(\mathcal{A}) = \phi$



Infinite zone graphs:



$$\begin{array}{c} q_0,\ 0 \le x = y \\ \downarrow \\ q_1,\ 0 \le x = y \\ \downarrow \\ q_1,\ x - y = 1 \\ \downarrow \\ q_1,\ x - y = 2 \\ \vdots \end{array}$$



Summary: - Notion of zones and zone graphs.

- In general, reachable part of ZG can be infinite.

Next: How to get an algorithm based on zones?