Updatable Timed Automata

- Bouyer, Dufourd, Fleury, Petit
  Theoretical Computer Science (2004)

Our goals:

- Syntax, semantics, examples (today)
- Emptiness problem
- Expressive power

Update:

$$q \xrightarrow[R]{a,\ g} q'$$

$x := 0$
$y := 0$ } Reset

$x := 5$
$x := y$
$x := y + 5$
$x := y - 2$

Let X be a set of clocks. An "update" is an expression generated by the following grammar:

$$\boxed{x := c \quad | \quad x := y + d}$$ → an update to x.

$c \in \mathbb{N}$ $\qquad$ $d \in \mathbb{Z}$

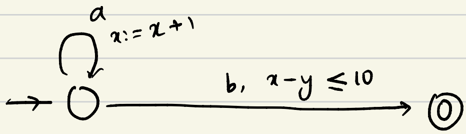
An update function maps each clock 'x' to an "update to x".

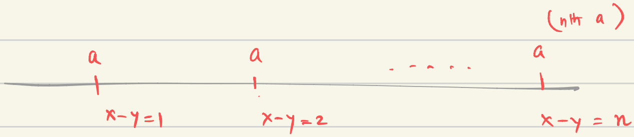
Eg: $X = \{x, y, z\}$ $\qquad$ update functions

$x := 2$	$z := x + 2$	$x := x - 1$
$y := y$	$y := y$	$y := y + 1$
$z := x + 2$	$x := 2$	$z := z$

same

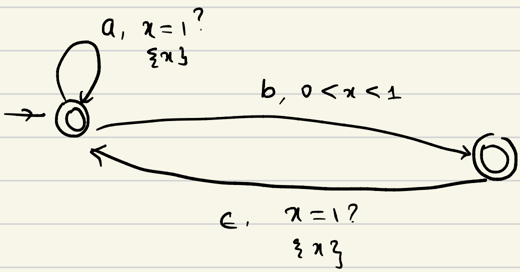
Example:





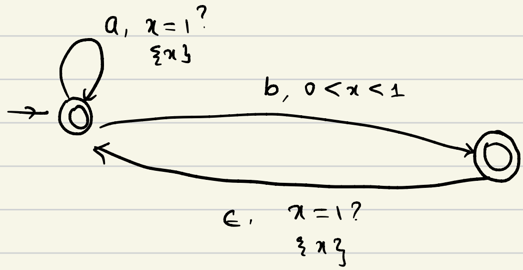
$$= \{ (a^k b, \tau) \mid k \le 10, \; \tau_1 \le \tau_2 \dots \tau_k \le \tau_{k+1} \}$$

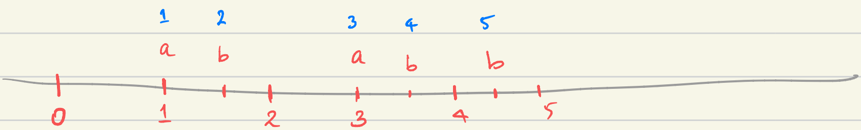
Example:



classical timed automaton.

Consider the above automaton with ϵ-transitions. What is its language?





$(w_1 w_2 \dots w_n,\ \tau_1 \tau_2 \dots \tau_n)$ is accepted if:

if $w_1 = a$, then $\tau_1 = 1$

$w_1 = b$, then $\tau_1 \in (0, 1)$

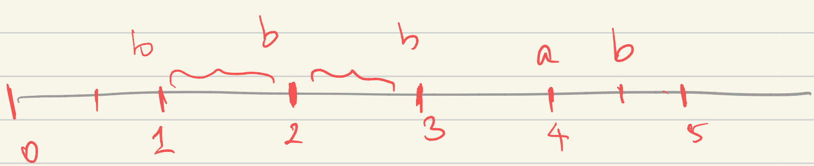
if $w_2 = a$, then $\tau_2 = 2$

$w_2 = b$, then $\tau_2 \in (1, 2)$

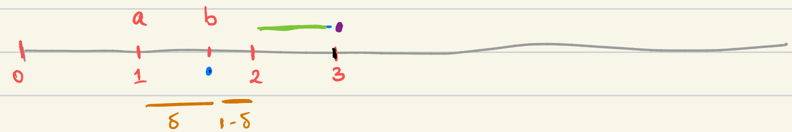
⋮

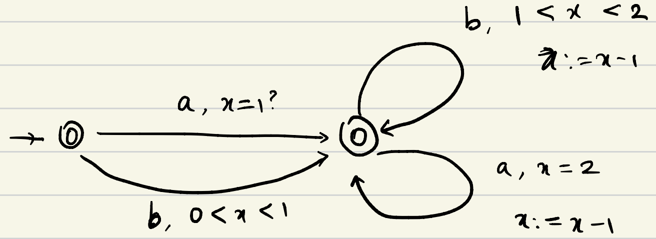
$\forall 1 \leq i \leq n$, if $w_i = a$, then $\tau_i = i$

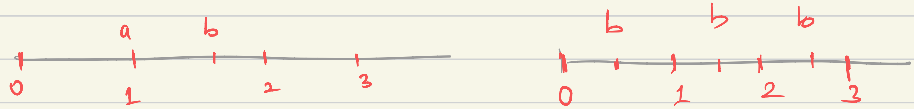
$w_i = b$, then $\tau_i \in (i-1, i)$



- Construct an updatable T.A for the previous language that has no ϵ-transitions.

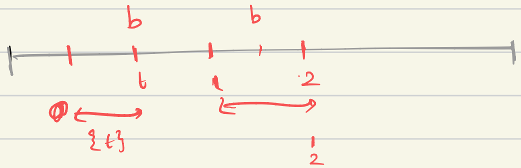






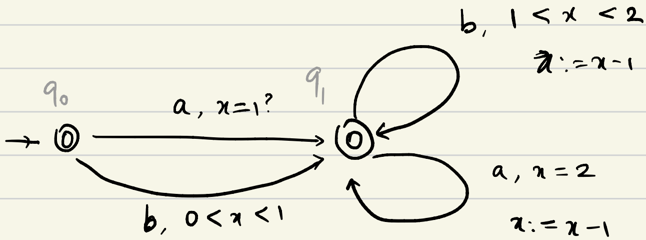
Invariant maintained by above automaton:

- After reading 'a' at 't', value of $x = 1$
- After reading 'b' at 't', value of $x = \{t\}$



Because of above invariant: the next 'a' is read at $x = 2$

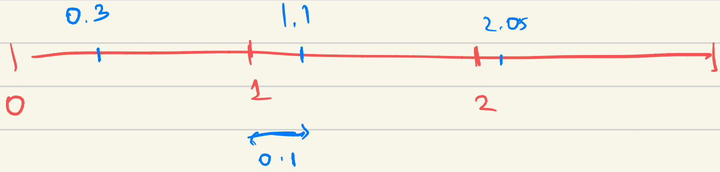
the next 'b' is read at $1 < x < 2$



b	b	b
0.3	1.1	2.05

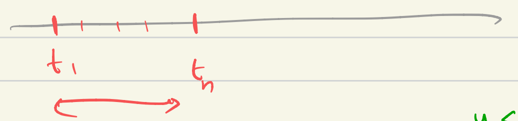
$$q_0 \longrightarrow q_1 \xrightarrow[1.1]{1} q_1 \longrightarrow q_1$$

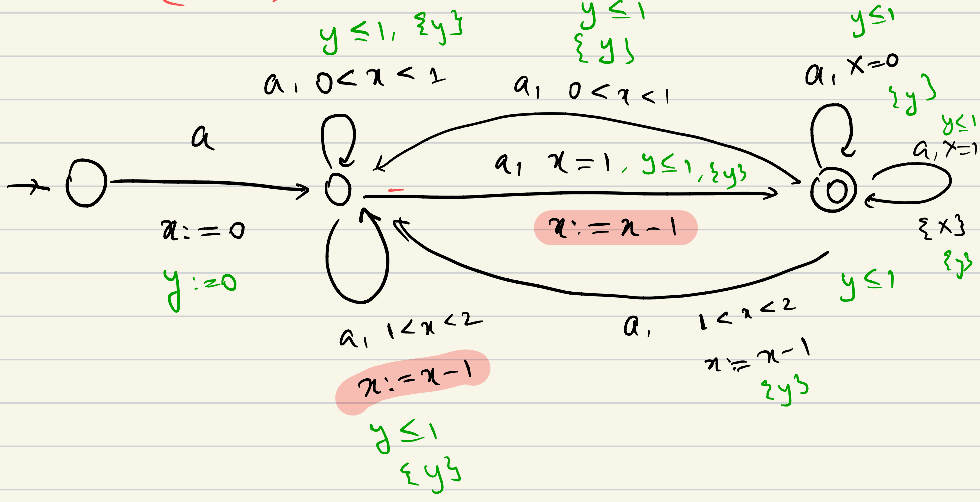
0   0.3   0.1   0.05

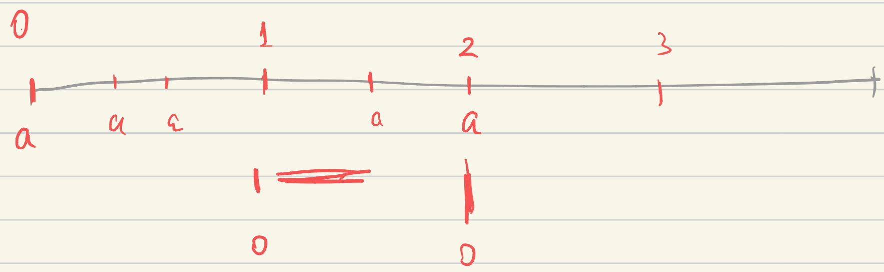


Example:

$$\{ (a^n, t_1 t_2 \ldots t_n) \mid t_{i+1} - t_i \leq 1$$
$$t_n - t_1 \text{ is an integer} \}$$

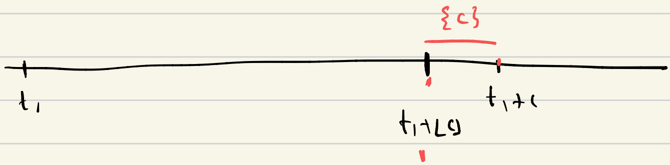






If the last 'a' seen is at $t_1 + c$, then

$t_1 + \lfloor c \rfloor$ becomes the new reference point.



Summary:

- introduction to updatable T.A.
- Some examples

TODAY'S LECTURE

- Reachability problem for Updatable Timed Automata

Semantics of UTA:

When does UTA $\mathcal{A}$ accept a timed word $(a_1, t_1)\,(a_2, t_2) \cdots (a_n, t_n)$?

Valuations: $v : X \longmapsto \mathbb{R}_{\geq 0}$

Operations on valuations: $v + \delta$

$up(v)$ ← New operation

Example: Suppose $X = \{x, y, z\}$

up is:

$x := x + 2$
$y := z - 5$
$z := x$

$$v_1 = \begin{array}{c} x \\ y \\ z \end{array} \begin{bmatrix} 12 \\ 2 \\ 7.2 \end{bmatrix} \qquad up(v_1) = \begin{bmatrix} 14 \\ 2.2 \\ 12 \end{bmatrix}$$

$$v_2 = \begin{bmatrix} 0 \\ 3 \\ 2.5 \end{bmatrix}$$

$up(v_2)$ not defined since $y := z - 5$ results in a negative value

---

$$up(v)(x) = \begin{cases} v(y) + d & \text{if } x := y + d \text{ and } v(y) + d \geq 0 \\ c & \text{if } x := c, \ c \geq 0 \end{cases}$$

$up(v)$ is undefined otherwise

Run of a UTA on a timed word:

$w: \quad (a_1, t_1) \ (a_2, t_2) \ \ldots \ (a_n, t_n)$

Run: $(q_0, v_0) \longrightarrow (q_1, v_1) \longrightarrow \ldots \longrightarrow (q_n, t_n)$

if: $\exists \quad q_i \xrightarrow[up_i]{a_i, g_i} q_{i+1}$

s.t. $v_i + (\underbrace{t_{i+1} - t_i}_{\delta_i}) \models g_i$

$up_i(v_i + \delta_i)$ is defined

$v_{i+1} = up_i(v_i + \delta_i)$

Accepting run: q_n is accepting

Emptiness problem:

- Given UTA $\mathcal{A}$, is language of $\mathcal{A}$ empty?

Theorem: Emptiness problem is undecidable for UTA.

Proof of undecidability:

Reducing emptiness problem of 2-counter machines.

2- Counter Machines

$$(Q, Q_0, \Sigma, \{c, d\}, \Delta)$$

Counters

Operations on counters:
1) increment $c{+}{+}$, $d{+}{+}$
2) decrement $c{-}{-}$, $d{-}{-}$
3) zero test $c==0$, $d==0$?

Ex. of a transition: $q \xrightarrow[c++]{a,\ d==0} q'$

- Counter values are always ≥ 0

- A transition with a decrement $c{-}{-}$ can be taken only when $c \geq 1$

Simulating a 2-counter machine using a UTA:

Run of the counter machine:

$$(q_0, 0, 0) \longrightarrow (q_1, 1, 0) \longrightarrow \cdots \longrightarrow (q_i, c_i, d_i) \longrightarrow \cdots$$

2-Counter machine $\mathcal{A}$ $\longrightarrow$ 3-clock UTA $\mathcal{B}$

clocks $\{x, y, z\}$

$$q \xrightarrow[c++]{} q' \quad \longrightarrow \quad q \xrightarrow[x := x+1]{z = 0\,?} q'$$

$$q \xrightarrow[d--]{} q' \quad \longrightarrow \quad q \xrightarrow[y := y-1]{z = 0\,?} q'$$

$$q \xrightarrow{c == 0} q' \quad \longrightarrow \quad q \xrightarrow{z = 0 \,\wedge\, x = 0\,?} q'$$

- There is zero time elapse in UTA $\mathcal{B}$, ensured by $z = 0$.

  Clock x gives the value of counter c,
  Clock y counter d

- For every run of 2-counter machine; there is a zero-time run of UTA:

$$(q_0, x=0, y=0, z=0) \longrightarrow (q_1, x = c_1, y = d_1, z = 0) \longrightarrow \cdots$$

Decidable subclasses:

General idea to show decidability : Region automaton

Recall region equivalence: for diagonal-free

M: biggest constant appearing in the automaton.

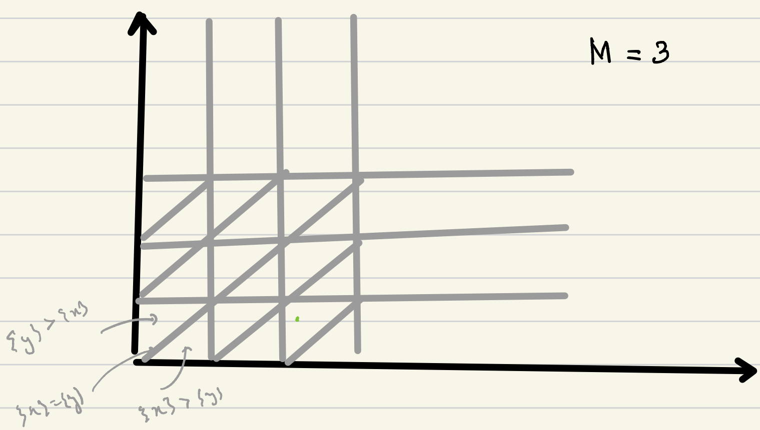
$v \equiv_M v'$ if

1) $\lfloor v(x) \rfloor = \lfloor v'(x) \rfloor$ or $v(x), v'(x) > M$

2) $\{v(x)\} = 0$ iff $\{v'(x)\} = 0$ $\quad \forall x : v(x) \leq M$

3) $\{v(x)\} \leq \{v(y)\} \iff \{v'(x)\} \leq \{v'(y)\}$ $\quad \forall x, y : v(x), v(y) \leq M$

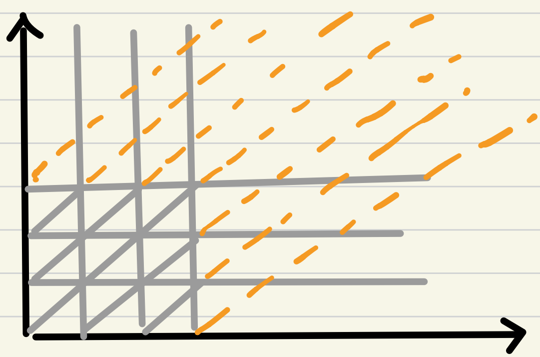
$M = 3$



$v \equiv_M v'$ does not work in the presence of diagonal constraints in guards. Add the following conditions in the presence of diagonals:

$$\boxed{\begin{array}{l} 4) \quad \lfloor v(x) - v(y) \rfloor = \lfloor v'(x) - v'(y) \rfloor \quad \text{or} \\ \qquad\qquad v(x) - v(y),\ v'(x) - v'(y) \text{ are both } > M \\ \qquad\qquad\qquad\qquad\qquad\qquad\qquad\qquad\qquad \text{or } < -M \\ 5) \quad \{ v(x) - v(y) \} = 0 \iff \{ v'(x) - v(y) \} = 0 \\ \qquad\qquad \forall x, y \text{ s.t. } -M \leq v(x) - v(y) \leq M \end{array}}$$

$M = 3$



orange lines in the presence of diagonals

- Call the equivalence given by the 5 conditions as $v \equiv_M^d v'$

The region equivalences $v \equiv_M^d v'$ and $v \equiv_M v'$ satisfy the following conditions:

Lemma 1: $v \equiv v' \implies \forall \delta \geq 0 \; \exists \delta' \geq 0$ s.t. $v + \delta \equiv v' + \delta'$

Lemma 2: $v \equiv_M v' \implies$ v and v' satisfy the same set of diagonal-free guards having constant $\leq M$

$v \equiv_M^d v' \implies$ v and v' satisfy the same set of diagonal-free and diagonal guards having constant $\leq M$

Lemma 3: $v \equiv v' \implies [R]v \equiv [R]v'$

Reset of R

We now want the region-equivalence to work when resets are replaced with updates.

Goal: For what subclasses of updates will the region equivalence work?

$$v \equiv v' \implies up(v) \equiv up(v')$$

Subclass 1: $x := c$, $x := y$, diagonal-free guards

Let M be max constant occurring among all guards in the automaton.

Problem: Show that region-equivalence $\equiv_M$ satisfies Lemma 3 with resets replaced with updates of the above form.

$$v \equiv_M v' \implies up(v) \equiv_M up(v')$$

for all updates of the form $x := c$, $x := y$ $x, y \in X$, $c \geq 0$

Proof:

$v \equiv_M v'$ if
1) $\lfloor v(x) \rfloor = \lfloor v'(x) \rfloor$ or $v(x), v'(x) > M$
2) $\{v(x)\} = 0 \iff \{v'(x)\} = 0$ $\forall x : v(x) \leq M$
3) $\{v(x)\} \leq \{v(y)\} \iff \{v'(x)\} \leq \{v'(y)\}$ $\forall x, y$ $v(x), v(y) \leq M$

$up(v) \equiv_M up(v')$

i) up maps $x := c$
$z := z$ $\forall z \neq x$

$up(v)(x) = c$ $up(v')(x) = c$

$up(v)(z) = v(z)$ $\forall z \neq x$ $up(v')(z) = v'(z)$

Conditions 1 & 2: $\lfloor up(v)(x) \rfloor = \lfloor up(v')(x) \rfloor = c$

$\lfloor up(v)(z) \rfloor = \lfloor v(z) \rfloor = \lfloor v'(z) \rfloor = \lfloor up(v')(z) \rfloor$

<u>Condition 3:</u> For all pairs both different from 'x', the condition holds due to $v \equiv_M v'$.

$$up(v)(x) = c$$

$$\therefore \{up(v)(x)\} = 0 \qquad \{up(v')(x)\} = 0$$

i) $\therefore \{up(v)(x)\} - \{up(v)(z)\} = -\{v(z)\} \le 0$ always

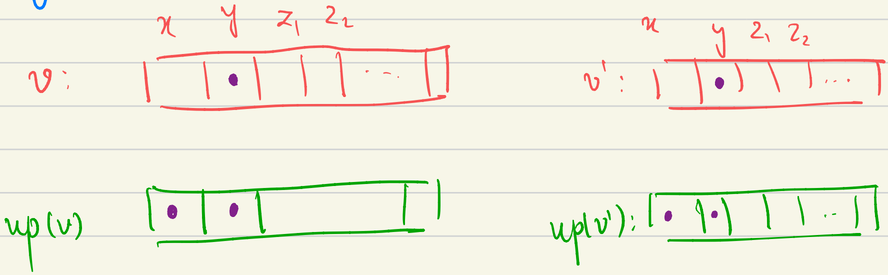
$|||^{ly}$ $\{up(v')(x)\} - \{up(v')(z)\} = -\{v'(z)\} \le 0$

ii) $\{up(v)(z)\} - \{up(v)(x)\} = \{v(z)\} \le 0$

$\{up(v')(z)\} - \{up(v')(x)\} = \{v'(z)\} \le 0$

Use the fact that $\{v(z)\} = 0$ iff $\{v'(z)\} = 0$.

ii) $x := y$



$x := c$

$x := y$

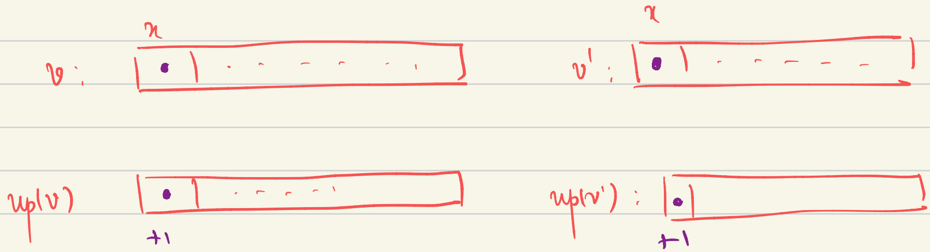
$+$

Subclass 2: $x := x+1$, diagonal-free guards

Let M be max constant occurring among all guards in the automaton.

Problem: Show that region-equivalence $\equiv_M$ satisfies Lemma 3 with resets replaced with updates of the above form.

To show: $v \equiv_M v' \Rightarrow up(v) \equiv_M up(v')$



$\{up(v)(x)\} = \{v(x)\}$ $\qquad$ $\{up(v'(x))\} = \{v'(x)\}$

$\lfloor up(v)(x) \rfloor = \lfloor v(x) \rfloor + 1$ $\qquad$ $\lfloor up(v')(x) \rfloor = \lfloor v'(x) \rfloor + 1$

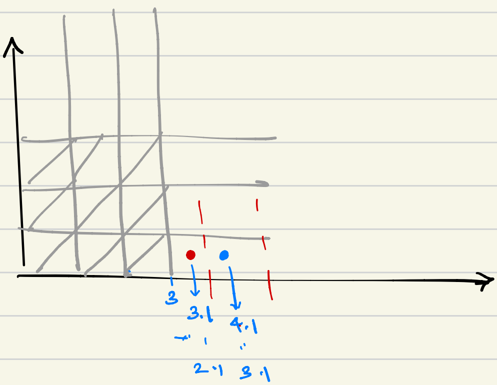
Problem: Consider subclass with $x := x-1$, diagonal-free guards.

Is there an 'M', in general, for which $\equiv_M$ satisfies Lemma 3?

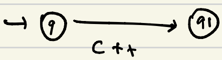
Can we give an M s.t. $v \equiv_M v' \Rightarrow \mathrm{upd}(v) \equiv_M \mathrm{upd}(v')$

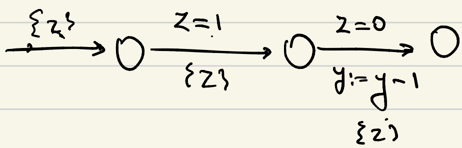
for decrement update.

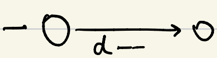
No.

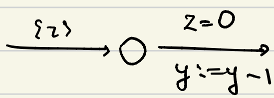


Idea of undecidability of this subclass: x, y, z









When $z=0$, value of x gives counter 'c'

y ——— 'd'

Summary:

- Emptiness problem for UTA
- Undecidable
- Some sublasses with decidability. Proof based on regions
- More decidable classes in the paper:

  Bouyer et al: Updatable Timed Automata.

Exercise:

1. $x := c$, $x := y$, guards can include diagonals → decidable

2. $x := c$, $x := y$ diagonals → undecidable
   $x := x+1$

Motivation for updates: preemptive scheduling

convenient models for scheduling problems.

TODAY'S GOALS

- Expressiveness of Updatable T.A.

(Recall):

Updatable Timed automaton (UTA):

Resets generalized to updates

Updates:

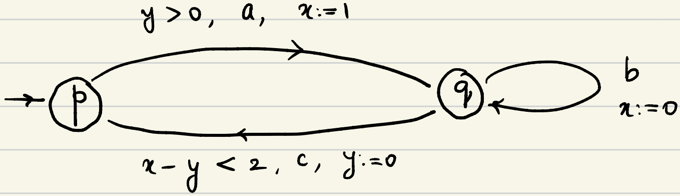
X: a set of clocks

For each clock $x \in X$, an update on x takes the following form:

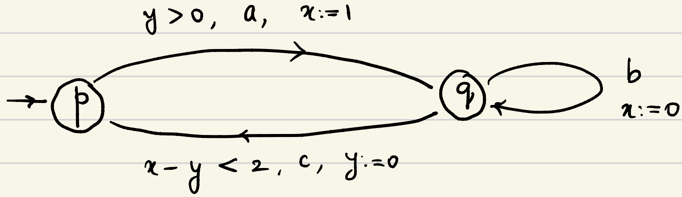
$$\boxed{x := c \quad | \quad x := y + d \qquad c \in \mathbb{N},\ d \in \mathbb{Z},\ y \in X}$$

Examples: $x := 5$, $x := x - 1$, $x := y + 2$, $x := y$, $x := 2 - 1$

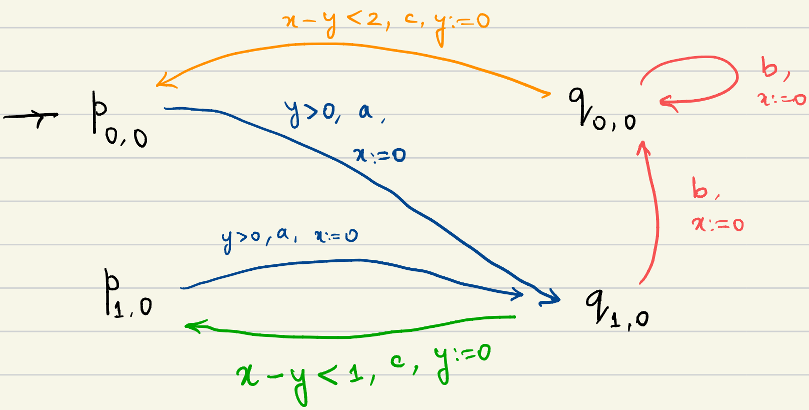
Question 1: Convert the following UTA into a TA:



Solution:



Idea: Remember the last update constant of each clock in the state.
Modify the guards depending on the appropriate constant in the state.



$(x+1) - y < 2$

Question 2: Consider UTA restricted to updates of the form $x := c \ (c \in \mathbb{N})$. Construct an equivalent timed automaton with only resets for this class of UTA.

Solution: Generalizing the previous construction:

Convert a UTA with $x := c$ updates to a T.A.

UTA $A := (Q, Q_0, X, \Sigma, \Delta, F)$

We will construct a TA B.

States: Let $S_x := \{ c \mid x := c \text{ appears in some transition} \} \cup \{0\}$

$$S = \prod_{x \in X} S_x$$

$x = \{3, 5\} \cup \{0\}$

$y = \{4, 7, 8\} \cup \{0\}$

$S = (3, 4)\ (3, 7) \ldots$

States of B are of the form: $Q \times S$

Transitions: For every $q \xrightarrow[up]{a,\ g} q'$ in A

we have a transition from every (q, σ)

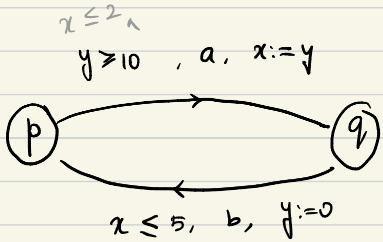
$$(q, \sigma) \xrightarrow[R_{up}]{a,\ g'} (q', \sigma')$$

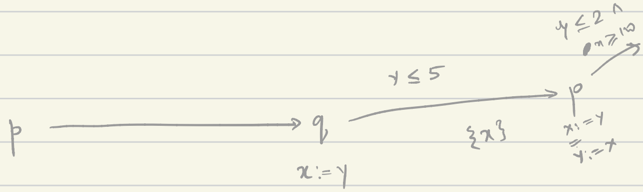
where $g' := g[x \mapsto x + \sigma(x)]$ (replace x with $x + \sigma(x)$)

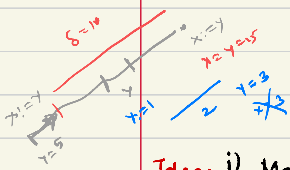
$R_{up} :=$ Set of clocks updated in up

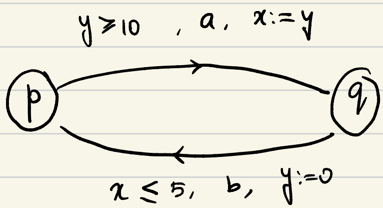
$\sigma'(x) = c$ if $x := c \in up$, else $\sigma'(x) = \sigma(x)$.

Question 3: Convert the following UTA into a TA:



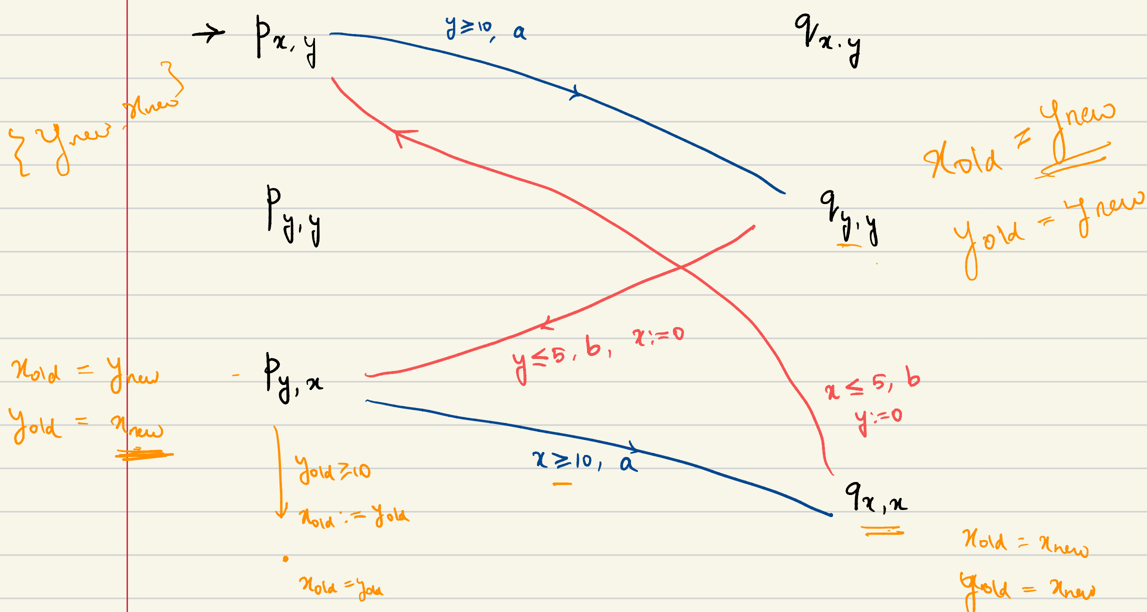




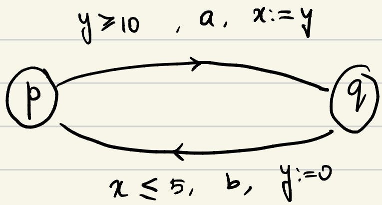


<u>Idea:</u> i) Maintain the last clock to which each clock was reset to.

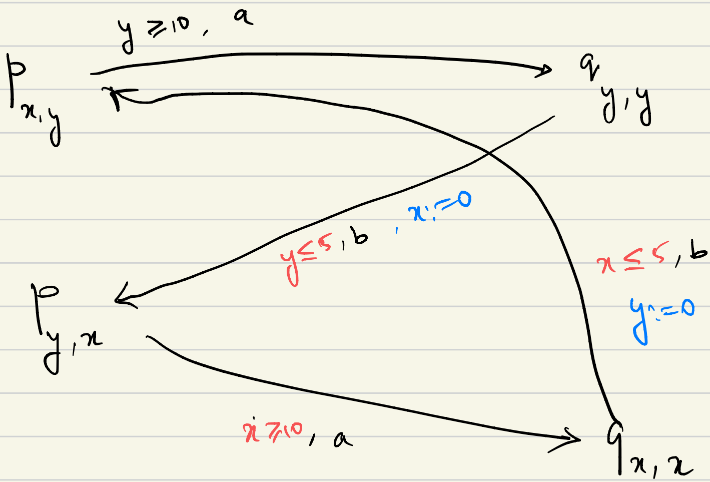
ii) If value of some clock is stored in some other clock, then that clock should not be updated to a constant.



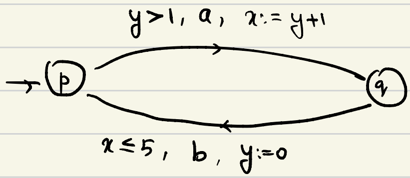
UTA:

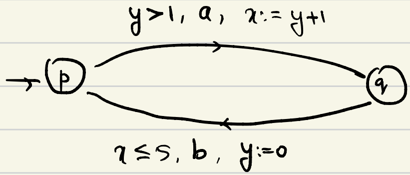

$y \geq 10$, a, $x := y$
p
q
$x \leq 5$, b, $y := 0$

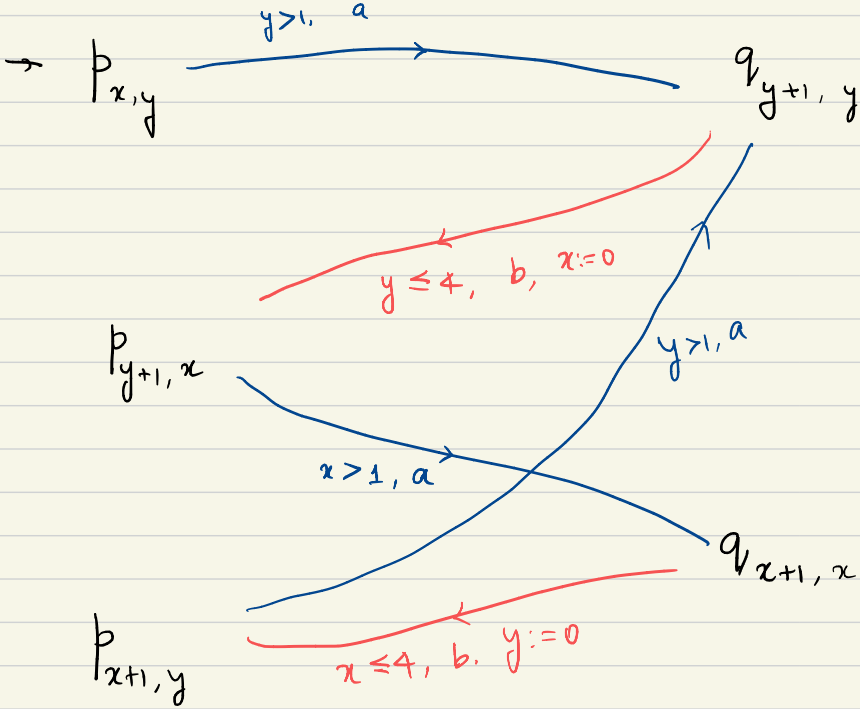
TA: x, y


$y \geq 10$, a
$p_{x,y}$
$q_{y,y}$
$y \leq 5$, b , $x := 0$
$x \leq 5$, b
$y := 0$
$p_{y,x}$
$x \geq 10$, a
$q_{x,x}$

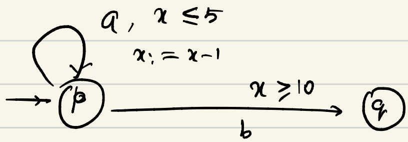
Question 4: Convert the following UTA into a TA:



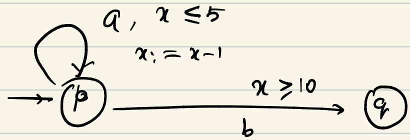

y>1, a, x:=y+1
p
q
x≤5, b, y:=0


y>1, a
p x,y
q y+1, y
y≤4, b, x:=0
p y+1, x
y>1, a
x>1, a
q x+1, x
p x+1, y
x≤4, b. y:=0

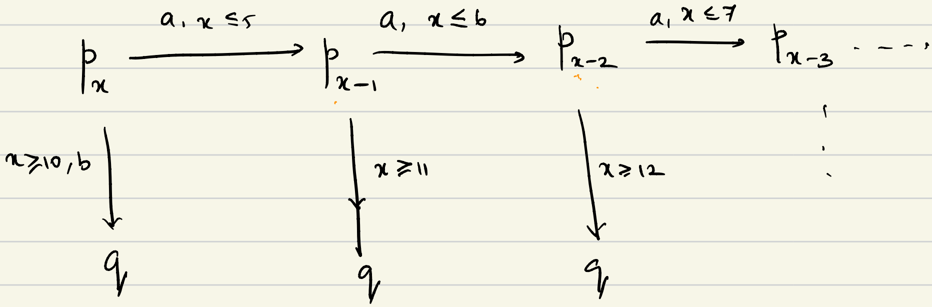
Question 5: Apply the construction of previous question to the following UTA:



$$p_x \xrightarrow{a,\, x \le 5} p_{x-1} \xrightarrow{x \le 6} p_{x-2} \xrightarrow{x \le 7} \cdots$$



$x_{old} := x_{old} - 1$
$= x_{new} - 1$



Remark 1: This construction does not work always.

The paper on Updatable Timed Automata by Bouyer et al. gives a termination criterion: it generates a system of inequalities; if the system has a solution, then the above construction gives finitely many states.

Remark 2: In our first lecture on UTA we have given an example of a timed language that is accepted by a UTA, but not by TA.

Summary:

	Diagonal-free	Diagonals
$x := c$, $x := y$	TA	TA
$x := y + c$	TA	More expressive
$x := x - 1$	More expressive	More expressive

$x := x+1$

$y - x$

$y - (x+1)$

$y - x - 1$