Topics in Timed Automata

B. Srivathsan

RWTH-Aachen

Software modeling and Verification group

System → $\mathcal{L}(B)$; *Specification* → $\mathcal{L}(A)$

$$\mathcal{L}(B) \subseteq \mathcal{L}(A)$$

$\updownarrow$

Is $\mathcal{L}(B) \cap \overline{\mathcal{L}(A)}$ empty?

If A is **deterministic**, inclusion can be solved

System *Specification*

$$\mathcal{L}(B) \subseteq \mathcal{L}(A)$$

$\updownarrow$

Is $\mathcal{L}(B) \cap \overline{\mathcal{L}(A)}$ empty?

If A is **deterministic**, inclusion can be solved

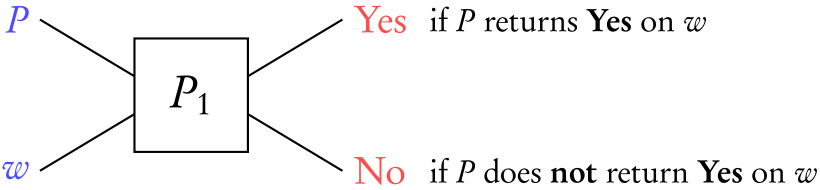
Q: Given general A and B, can we **decide** if $\mathcal{L}(B) \subseteq \mathcal{L}(A)$?

Lecture 3:

Language inclusion is undecidable

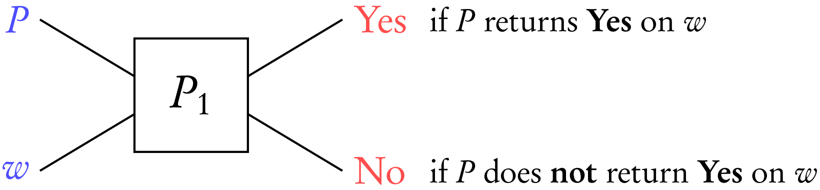
P : an arbitrary **boolean program** (string)

w : an arbitrary **string**

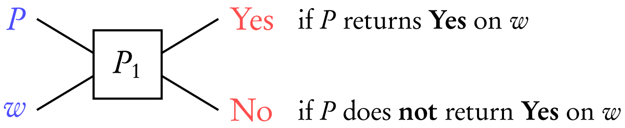


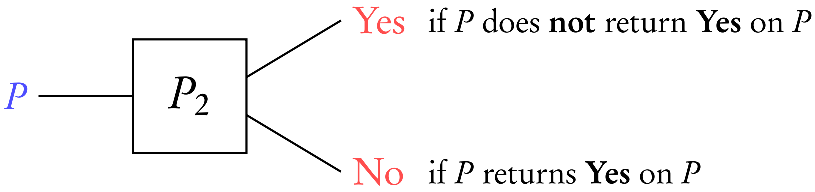
P : an arbitrary **boolean program** (string)

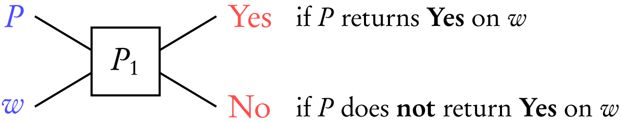
w : an arbitrary **string**



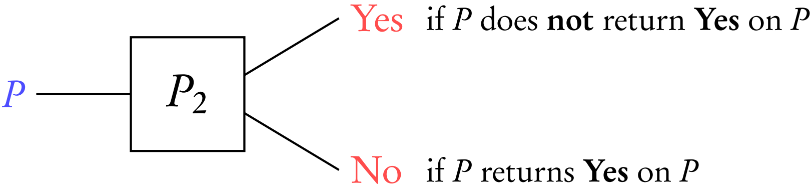
Can program P_1 **exist?**

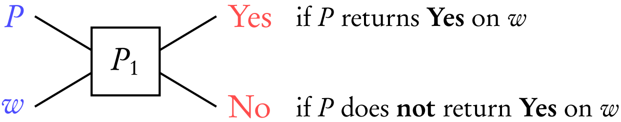

P
w
P1
Yes if P returns Yes on w
No if P does not return Yes on w


P
P2
Yes if P does not return Yes on P
No if P returns Yes on P

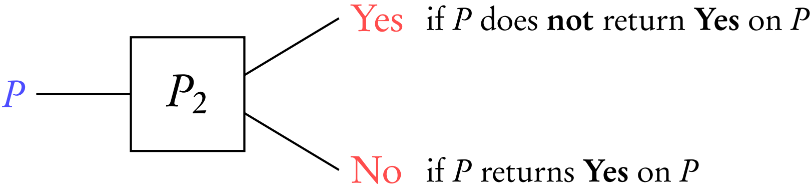


If P_1 exists, then P_2 exists

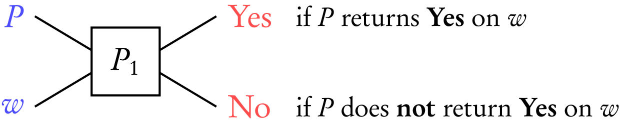




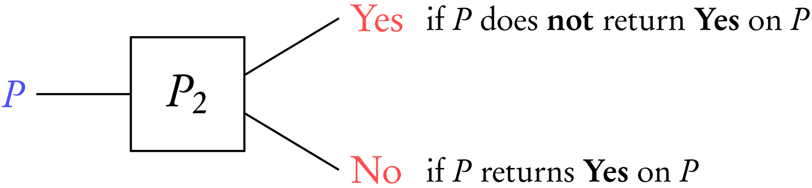
If P_1 exists, then P_2 exists



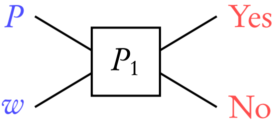
P_2 returns **Yes** on P_2



If P_1 exists, then P_2 exists



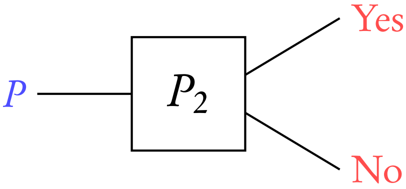
P_2 returns **Yes** on P_2 if P_2 does **not** return **Yes** on P_2



if P returns **Yes** on w

if P does **not** return **Yes** on w

If P_1 exists, then P_2 exists

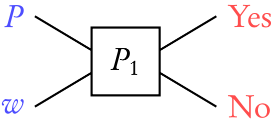


if P does **not** return **Yes** on P

if P returns **Yes** on P

P_2 returns **Yes** on P_2 if P_2 does **not** return **Yes** on P_2

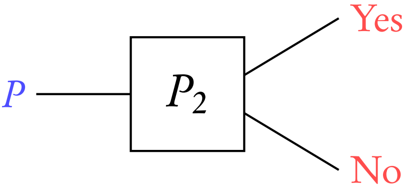
P_2 returns **No** on P_2



if P returns **Yes** on w

if P does **not** return **Yes** on w

If P_1 exists, then P_2 exists

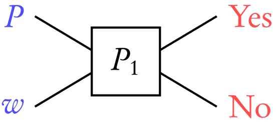


if P does **not** return **Yes** on P

if P returns **Yes** on P

P_2 returns **Yes** on P_2 if P_2 does **not** return **Yes** on P_2

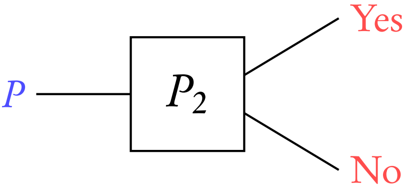
P_2 returns **No** on P_2 if P_2 returns **Yes** on P_2



Yes if P returns **Yes** on w

No if P does **not** return **Yes** on w

If P_1 exists, then P_2 exists



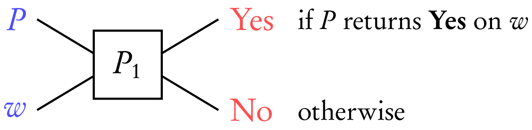
Yes if P does **not** return **Yes** on P

No if P returns **Yes** on P

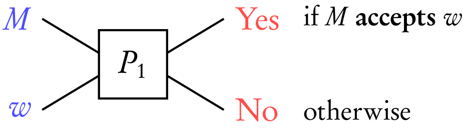
P_2 returns **Yes** on P_2 if P_2 does **not** return **Yes** on P_2

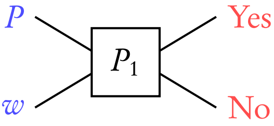
P_2 returns **No** on P_2 if P_2 returns **Yes** on P_2

P_2 cannot exist $\Rightarrow$ **P_1 cannot exist**



Turing machine
2-counter machine
. . .

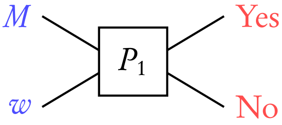




if P returns **Yes** on w

otherwise

Turing machine
2-counter machine
...



if M **accepts** w

otherwise

Membership problem for 2-counter machines (MP)

Given a **2-counter machine** M and an arbitrary string w, checking if M **accepts** w is **undecidable**

Goal of this lecture

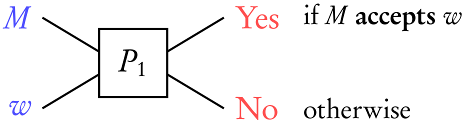
Timed regular languages are **powerful** enough to **encode** computations of **2-counter machine**

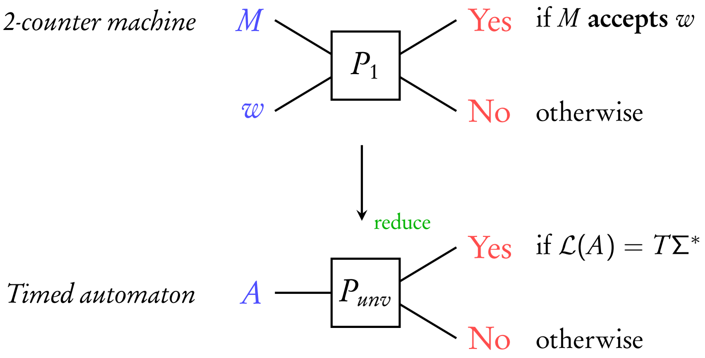
We will see:

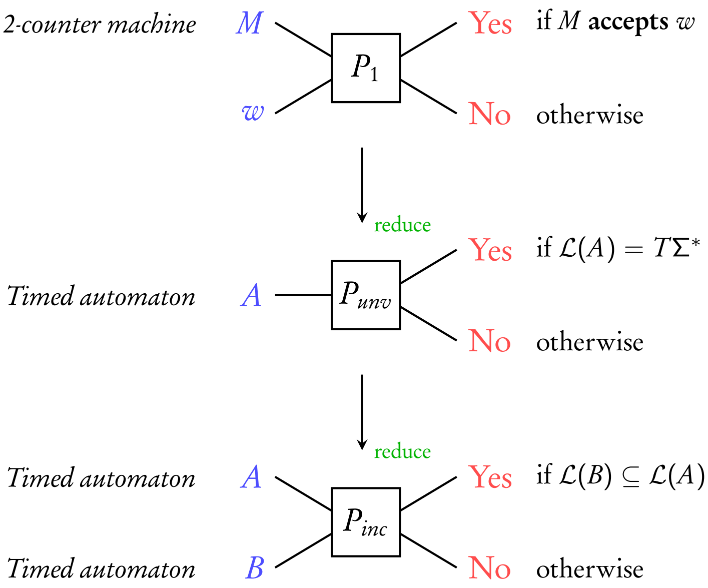
If there is an algorithm for TA language inclusion,

then there is an algorithm for MP

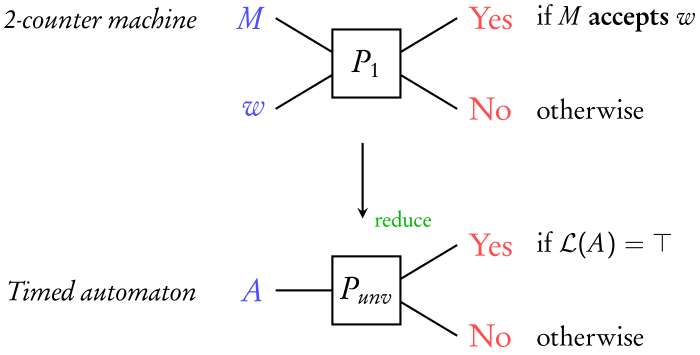
2-counter machine



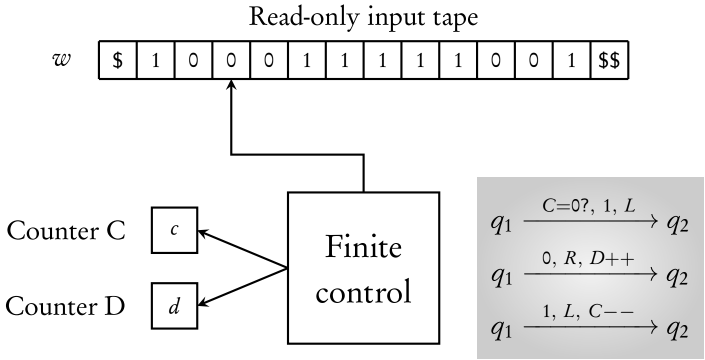

2-counter machine
M
w
P_1
Yes
if M **accepts** w
No
otherwise
reduce
Timed automaton
A
P_{unv}
Yes
if $\mathcal{L}(A) = T\Sigma^*$
No
otherwise


2-counter machine
M
w
P_1
Yes
No
if M **accepts** w
otherwise
reduce
Timed automaton
A
P_{unv}
Yes
No
if $\mathcal{L}(A) = T\Sigma^*$
otherwise
reduce
Timed automaton
A
Timed automaton
B
P_{inc}
Yes
No
if $\mathcal{L}(B) \subseteq \mathcal{L}(A)$
otherwise

Coming next...


2-counter machine
M
w
P1
Yes
if M accepts w
No
otherwise
reduce
Timed automaton
A
Punv
Yes
if L(A) = ⊤
No
otherwise

2-counter machines



Computation: $\langle q_0, w_0, 0, 0\rangle \; \langle q_1, w_{i_1}, c_1, d_1\rangle \; \cdots \langle q_i, w_i, c_i, d_i\rangle \; \cdots$

Accept: if **some** computation **ends** in $\langle q_F, \star, \star, \star\rangle$

Goal 1

Given M and w

define **timed language** L_{undec} s.t

M accepts w iff $L_{undec} \neq \emptyset$

Words in L_{undec} **encode accepting computations** of M on w

Configuration of a 2-counter machine:

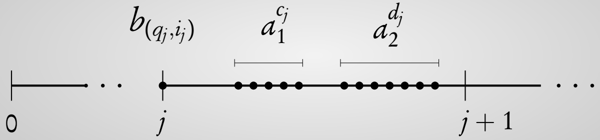
$$\langle q, w_k, c, d \rangle$$

Encoding as a word over alphabet: $\{a_1,\ a_2,\ b_i\}$

where $i \in Q \times \{0, \ldots, |w| + 1\}$

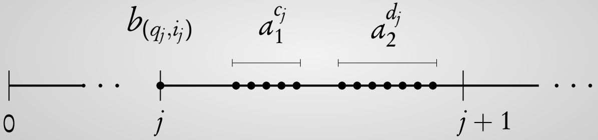
$$b_{(q,k)}\ a_1^c\ a_2^d$$

$$\langle q_0, w_{i_0}, 0, 0\rangle \cdots \langle q_j, w_{i_j}, c_j, d_j\rangle \cdots \langle q_m, w_{i_m}, c_m, d_m\rangle$$



Encode the j^{th} configuration in $[\, j, j+1\,)$

$$\langle q_0, w_{i_0}, 0, 0\rangle \cdots \langle q_j, w_{i_j}, c_j, d_j\rangle \cdots \langle q_m, w_{i_m}, c_m, d_m\rangle$$



Encode the j^{th} configuration in $[\,j, j+1\,)$

- if $c_{j+1} = c_j$, $\forall a_1$ at time t in $(j, j+1)$, $\exists a_1$ at time $t+1$
- if $c_{j+1} = c_j + 1$,

  $\forall a_1$ at time t in $(j+1, j+2)$ **except** the last one,

  $\exists a_1$ at time $t-1$
- if $c_{j+1} = c_j - 1$,

  $\forall a_1$ at time t in $(j, j+1)$ **except** the last one,

  $\exists a_1$ at time $t+1$

(same for counter d)

L_{undec} : encodes the **accepting computations**

Timed word $(\sigma, \tau) \in L_{undec}$ iff

L_{undec} : encodes the **accepting computations**

Timed word $(\sigma, \tau) \in L_{undec}$ iff

- $\sigma = b_{i_0} a_1^{c_0} a_2^{d_0} \; b_{i_1} a_1^{c_1} a_2^{c_2} \; \cdots \; b_{i_m} a_1^{c_m} a_2^{c_m}$ s.t. $\langle q_0, w_{i_0}, c_0, d_0 \rangle \langle q_1, w_{i_1}, c_1, d_1 \rangle \cdots \langle q_m, w_{i_m}, c_m, d_m \rangle$ is accepting

L_{undec} : encodes the **accepting computations**

Timed word $(\sigma, \tau) \in L_{undec}$ iff

- $\sigma = b_{i_0} a_1^{c_0} a_2^{d_0} \ b_{i_1} a_1^{c_1} a_2^{c_2} \ \cdots \ b_{i_m} a_1^{c_m} a_2^{c_m}$ s.t.
  $\langle q_0, w_{i_0}, c_0, d_0 \rangle \langle q_1, w_{i_1}, c_1, d_1 \rangle \cdots \langle q_m, w_{i_m}, c_m, d_m \rangle$ is accepting
- each b_{i_j} occurs at time j

L_{undec} : encodes the **accepting computations**

Timed word $(\sigma, \tau) \in L_{undec}$ iff

- $\sigma = b_{i_0} a_1^{c_0} a_2^{d_0} \; b_{i_1} a_1^{c_1} a_2^{c_2} \; \cdots \; b_{i_m} a_1^{c_m} a_2^{c_m}$ s.t.
  $\langle q_0, w_{i_0}, c_0, d_0 \rangle \langle q_1, w_{i_1}, c_1, d_1 \rangle \cdots \langle q_m, w_{i_m}, c_m, d_m \rangle$ is accepting
- each b_{i_j} occurs at time j
- if $c_{j+1} = c_j$, $\forall a_1$ at time t in $(j, j+1)$, $\exists a_1$ at time $t+1$
- if $c_{j+1} = c_j + 1$,

  $\forall a_1$ at time t in $(j+1, j+2)$ **except** the last one,

  $\exists a_1$ at time $t-1$
- if $c_{j+1} = c_j - 1$,

  $\forall a_1$ at time t in $(j, j+1)$ **except** the last one,

  $\exists a_1$ at time $t+1$

(same for counter d)

Goal 1

Given M and w

define **timed language** L_{undec} s.t

M accepts w iff $L_{undec} \neq \emptyset$

Words in L_{undec} **encode accepting computations** of M on w

Done!

Goal 2

Given M and w

construct a **timed automaton** $\mathcal{A}_{undec}$

for the **complement** language $\overline{L_{undec}}$

Goal 2

Given M and w

construct a **timed automaton** $\mathcal{A}_{undec}$

for the **complement** language $\overline{L_{undec}}$

M **accepts** w iff $\mathcal{L}(\mathcal{A}_{undec}) \neq T\Sigma^*$

Goal 2

Given M and w

construct a **timed automaton** $\mathcal{A}_{undec}$

for the **complement** language $\overline{L_{undec}}$

M **accepts** w iff $\mathcal{L}(\mathcal{A}_{undec}) \neq T\Sigma^*$

$\rightarrow$ reduction to universality of TA

$\overline{L_{undec}}$: words that **do not** encode **accepting computations**

Timed word $(\sigma, \tau) \in \overline{L_{undec}}$ iff

$\overline{L_{undec}}$: words that **do not** encode **accepting computations**

Timed word $(\sigma, \tau) \in \overline{L_{undec}}$ iff

- either, there is **no** b**-symbol** at some **integer** point j

$\overline{L_{undec}}$: words that **do not** encode **accepting computations**

Timed word $(\sigma, \tau) \in \overline{L_{undec}}$ iff

- either, there is **no** b**-symbol** at some **integer** point j
- or, there is a $(j, j+1)$ with a subsequence **not** of the form $a_1^* a_2^*$

$\overline{L_{undec}}$: words that **do not** encode **accepting computations**

Timed word $(\sigma, \tau) \in \overline{L_{undec}}$ iff

- either, there is **no *b*-symbol** at some **integer** point j
- or, there is a $(j, j+1)$ with a subsequence **not** of the form $a_1^* a_2^*$
- or, **initial** subsequence in $[0, 1)$ is wrong

$\overline{L_{undec}}$: words that **do not** encode **accepting computations**

Timed word $(\sigma, \tau) \in \overline{L_{undec}}$ iff

- either, there is **no** ***b*****-symbol** at some **integer** point j
- or, there is a $(j, j+1)$ with a subsequence **not** of the form $a_1^* a_2^*$
- or, **initial** subsequence in $[0, 1)$ is wrong
- or, some transition of M has been **violated** in the word

$\overline{L_{undec}}$: words that **do not** encode **accepting computations**

Timed word $(\sigma, \tau) \in \overline{L_{undec}}$ iff

- either, there is **no b-symbol** at some **integer** point j
- or, there is a $(j, j+1)$ with a subsequence **not** of the form $a_1^* a_2^*$
- or, **initial** subsequence in $[0, 1)$ is wrong
- or, some transition of M has been **violated** in the word
- or, final b-symbol denotes **non-accepting** state

$\overline{L_{undec}}$: words that **do not** encode **accepting computations**

Timed word $(\sigma, \tau) \in \overline{L_{undec}}$ iff

- either, there is **no b-symbol** at some **integer** point j $\mathcal{A}_0$
- or, there is a $(j, j+1)$ with a subsequence **not** of the form $a_1^* a_2^*$ $\mathcal{A}_1$
- or, **initial** subsequence in $[0, 1)$ is wrong $\mathcal{A}_{init}$
- or, some transition of M has been **violated** in the word $\mathcal{A}_t$ for each transition t of M
- or, final b-symbol denotes **non-accepting** state $\mathcal{A}_{acc}$

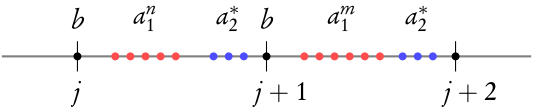
$\overline{L_{undec}}$: words that **do not** encode **accepting computations**

Timed word $(\sigma, \tau) \in \overline{L_{undec}}$ iff

- either, there is **no b-symbol** at some **integer** point j $\mathcal{A}_0$
- or, there is a $(j, j+1)$ with a subsequence **not** of the form $a_1^* a_2^*$ $\mathcal{A}_1$
- or, **initial** subsequence in $[0, 1)$ is wrong $\mathcal{A}_{init}$
- or, some transition of M has been **violated** in the word $\mathcal{A}_t$ for each transition t of M
- or, final b-symbol denotes **non-accepting** state $\mathcal{A}_{acc}$

Required $\mathcal{A}_{undec}$: **union** of $\mathcal{A}_0, \mathcal{A}_1, \mathcal{A}_{init}, \mathcal{A}_{t_1}, \ldots, \mathcal{A}_{t_p}, \mathcal{A}_{acc}$

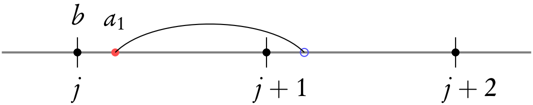
Crux

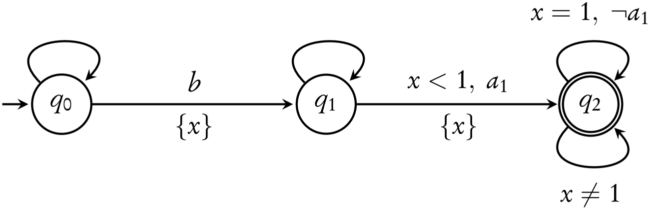


With our encoding, can timed automata express that $n \neq m$?

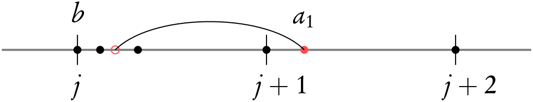
1. $\exists\, a_1$ at time $t \in (j,\ j+1)$ s.t there is no a_1 at $t+1$, or
2. $\exists\, a_1$ at time $t \in (j+1,\ j+2)$ s.t. there is no a_1 at $t-1$

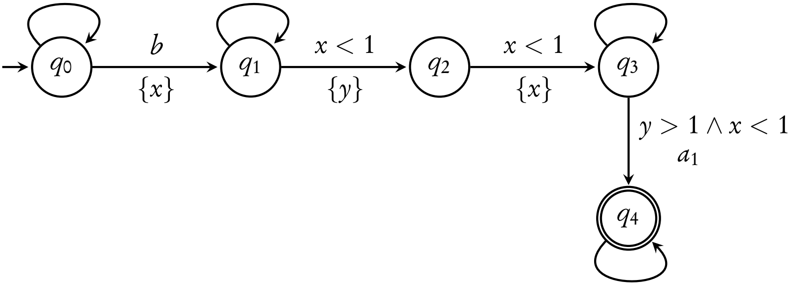
$\exists\, a_1$ at time $t \in (j,\ j+1)$ s.t there is no a_1 at $t+1$



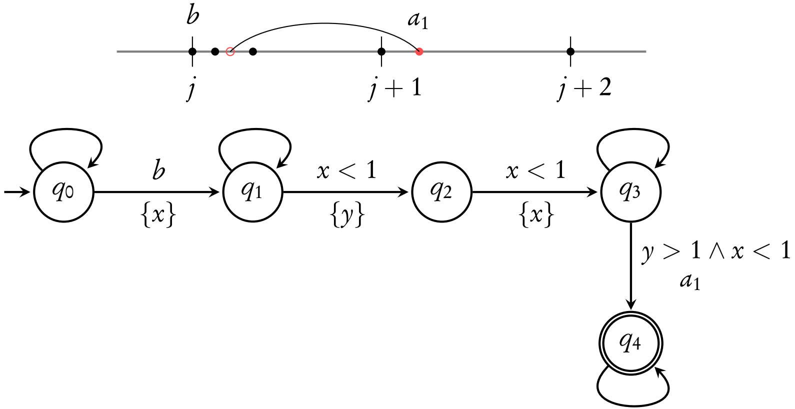


$\exists\, a_1$ at time $t \in (j+1,\, j+2)$ s.t. there is no a_1 at $t-1$





$\exists\, a_1$ at time $t \in (j+1,\ j+2)$ s.t. there is no a_1 at $t-1$



Need only **two clocks**!

$\overline{L_{undec}}$: words that **do not** encode **accepting computations**

Timed word $(\sigma, \tau) \in \overline{L_{undec}}$ iff

- either, there is **no** b**-symbol** at some **integer** point j $\mathcal{A}_0$
- or, there is a $(j, j+1)$ with a subsequence **not** of the form $a_1^* a_2^*$ $\mathcal{A}_1$
- or, **initial** subsequence in $[0, 1)$ is wrong $\mathcal{A}_{init}$
- or, some transition of M has been **violated** in the word $\mathcal{A}_t$ for each transition t of M
- or, final b-symbol denotes **non-accepting** state $\mathcal{A}_{acc}$

Required $\mathcal{A}_{undec}$ can be constructed using **two** clocks

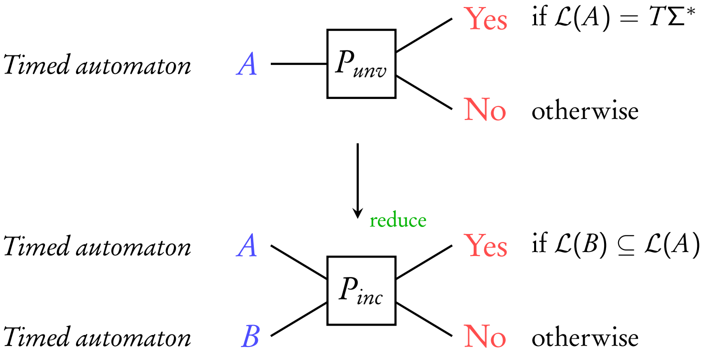
M **accepts** w iff $\mathcal{L}(A_{undec}) \neq T\Sigma^*$

Universality for TA

The universality problem is **undecidable** for TA with **two clocks or more**

A theory of timed automata

Alur and Dill. *TCS'94*



Put B as the **trivial** single state automaton **accepting** $T\Sigma *$

$$\mathcal{L}(A) = T\Sigma^* \quad \text{iff} \quad \mathcal{L}(B) \subseteq \mathcal{L}(A)$$

Language inclusion

The problem $\mathcal{L}(B) \subseteq \mathcal{L}(A)$ is **undecidable** when A has **two clocks or more**

A theory of timed automata

Alur and Dill. *TCS'94*

Next lecture...

- $\mathcal{L}(B) \subseteq \mathcal{L}(A)$ is decidable when A has **at most 1 clock**

- Further understanding as to why no algorithm when A has more than **two clocks**