TODAY'S GOAL:

Given T.A. $\mathcal{A}$ and $\mathcal{B}$, checking if

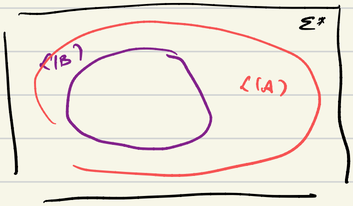
system $\rightarrow$ $\mathcal{L}(\mathcal{B}) \subseteq \mathcal{L}(\mathcal{A})$ $\leftarrow$ Property

is undecidable

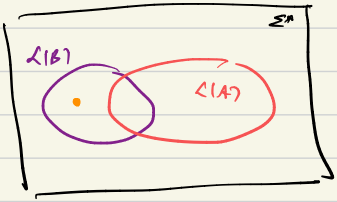
If $\mathcal{B}$ and $\mathcal{A}$ were NFA, how would we check:

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

→ untimed words over Σ^*



$\mathcal{L}(\mathcal{B}) \cap \mathcal{L}(\mathcal{A})^c = \phi$



$\mathcal{L}(\mathcal{B}) \cap \mathcal{L}(\mathcal{A})^c \neq \phi$

$\mathcal{L}(\mathcal{B}) \subseteq \mathcal{L}(\mathcal{A})$ iff $\mathcal{L}(\mathcal{B}) \cap \mathcal{L}(\mathcal{A})^c = \phi$

For NFA's we can effectively construct automaton $\mathcal{A}'$ for $\mathcal{L}(\mathcal{A})^c$.

$$\mathcal{L}(\mathcal{A}') = \mathcal{L}(\mathcal{A})^c$$

- We have seen earlier that there are timed automata for which the complement is not timed regular.

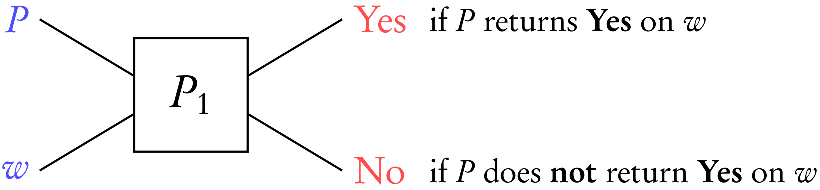
- So we cannot employ this technique for timed automata inclusion

Language inclusion is undecidable

Coming Next: Short recap of undecidability

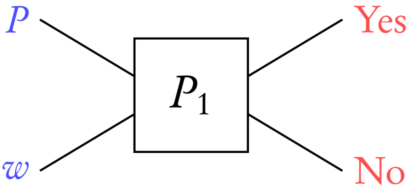
P : an arbitrary **boolean program** (string)

w : an arbitrary **string**



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

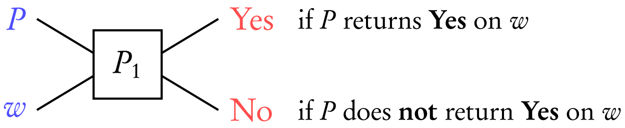
w : an arbitrary **string**

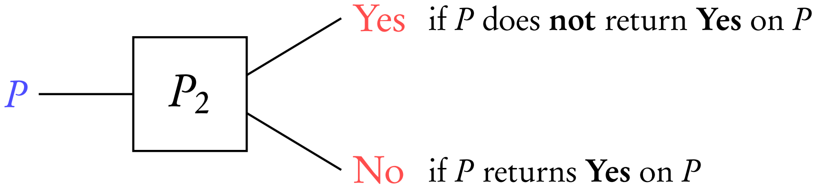


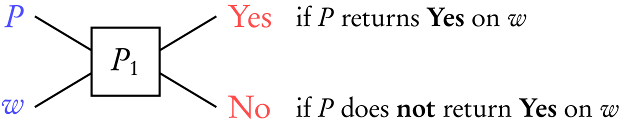
Yes if P returns **Yes** on w

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

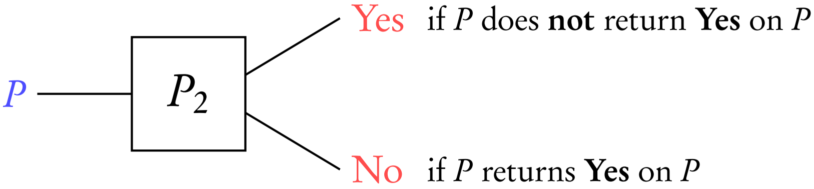
Can program P_1 **exist?**

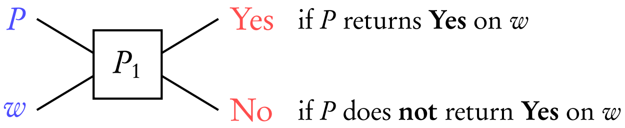

P
w
P1
Yes if P returns Yes on w
No if P does not return Yes on w


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

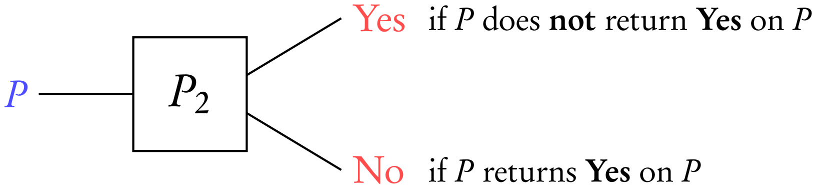


If P_1 exists, then P_2 exists

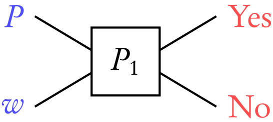




If P_1 exists, then P_2 exists



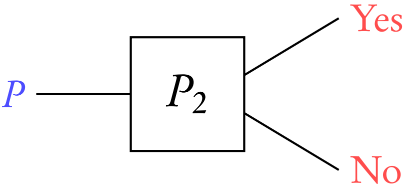
P_2 returns **Yes** on P_2



if P returns **Yes** on w

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

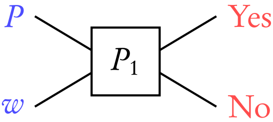
If P_1 exists, then P_2 exists



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

if P returns **Yes** on P

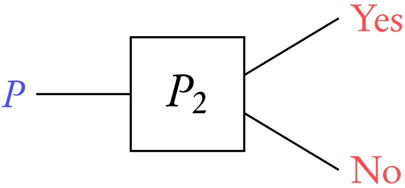
P_2 returns **Yes** on P_2 if P_2 does **not** return **Yes** on P_2



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

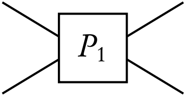
No if P returns **Yes** on P

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

P_2 returns **No** on P_2

P

w

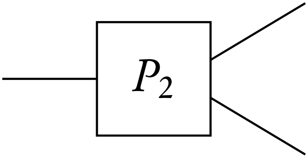


Yes if P returns **Yes** on w

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

If P_1 exists, then P_2 exists

P

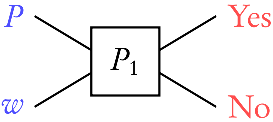


Yes if P does **not** return **Yes** on P

No if P returns **Yes** on P

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

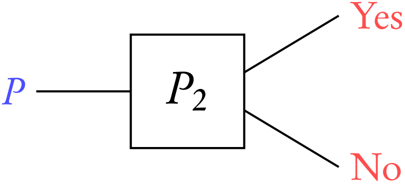
P_2 returns **No** on P_2 if P_2 returns **Yes** on P_2



Yes if P returns **Yes** on w

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

If P_1 exists, then P_2 exists



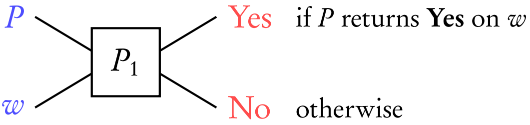
Yes if P does **not** return **Yes** on P

No if P returns **Yes** on P

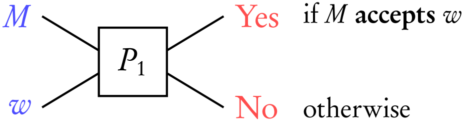
P_2 returns **Yes** on P_2 if P_2 does **not** return **Yes** on P_2

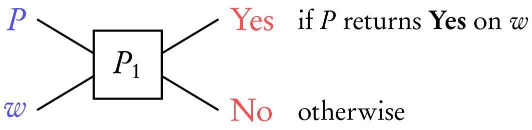
P_2 returns **No** on P_2 if P_2 returns **Yes** on P_2

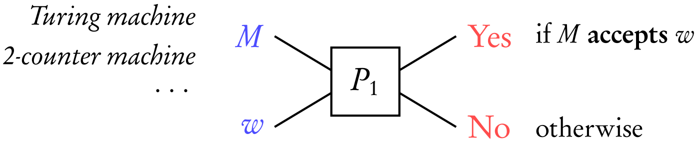
P_2 cannot exist $\Rightarrow$ **P_1 cannot exist**



Turing machine
2-counter machine
. . .







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**

deterministic

Goal of this lecture

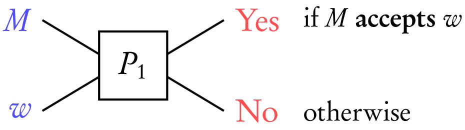
Timed regular languages are **powerful** enough to **encode** computations of **2-counter machine**

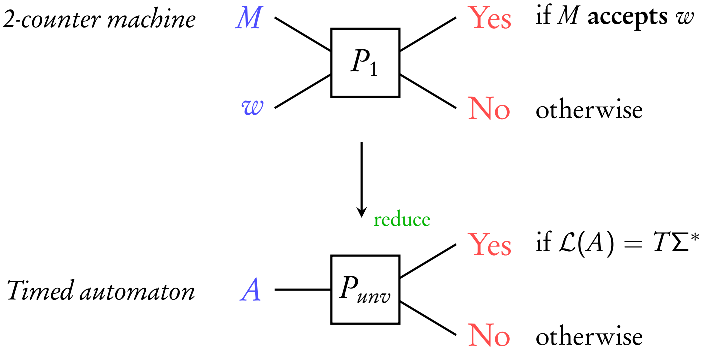
We will see:

If there is an algorithm for TA language inclusion,

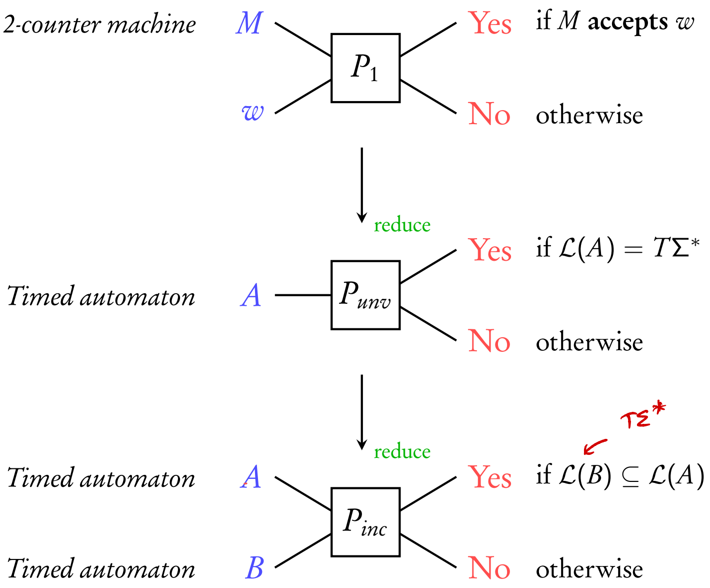
then there is an algorithm for MP

2-counter machine



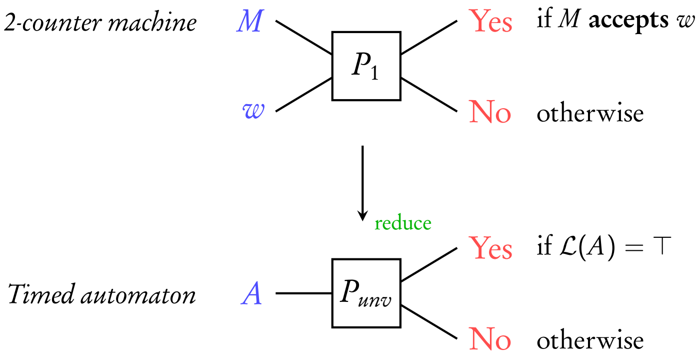


2-counter machine
M
w
P_1
Yes
if M **accepts** w
No
otherwise
reduce
Timed automaton
A
P_{unv}
Yes
if $\mathcal{L}(A) = T\Sigma^*$
No
otherwise



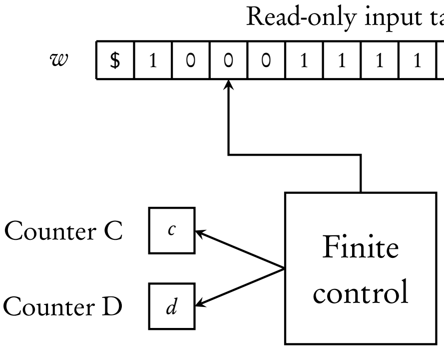
2-counter machine
M
w
P_1
Yes if M **accepts** w
No otherwise
reduce
Timed automaton
A
P_{unv}
Yes if $\mathcal{L}(A) = T\Sigma^*$
No otherwise
reduce
Timed automaton
A
Timed automaton
B
P_{inc}
$T\Sigma^*$
Yes if $\mathcal{L}(B) \subseteq \mathcal{L}(A)$
No otherwise

Coming next...



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

2-counter machines



$$q_1 \xrightarrow{C=0?,\ 1,\ L} q_2$$

$$q_1 \xrightarrow{0,\ R,\ D{+}{+}} q_2$$

$$q_1 \xrightarrow{1,\ L,\ C{-}{-}} q_2$$

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:

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

$\langle q_1, w_5, 3, 5 \rangle$

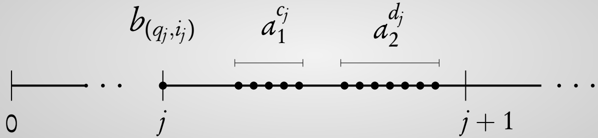
Encoding as a word over alphabet: $\{a_1, a_2, b_i\}$

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

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

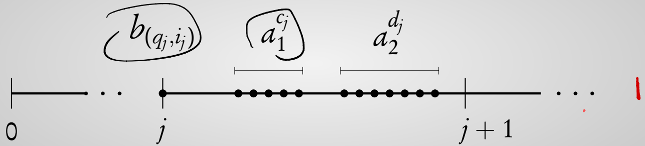
$b_{(q_1, w_5)}\, a_1 a_1 a_1\, a_2 a_2 a_2 a_2 a_2$

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



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

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

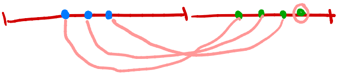


Encode the j^{th} configuration in $[\,j, j+1\,)$

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

$$\begin{bmatrix} b_{(q_0,0)} \\ 0 \end{bmatrix} \quad \begin{bmatrix} b_{(q_1,1)} & a_1 & a_2 \\ 1 & 1.5 & 1.7 \end{bmatrix} \quad \begin{bmatrix} b_{(q_2,2)} & a_1 & a_1 & a_2 \\ 2 & 2.5 & 2.6 & 2.7 \end{bmatrix}$$

$$\langle q_0, w_0, 0, 0 \rangle \quad \langle q_1, w_1, 1, 1 \rangle \quad \langle q_2, w_2, 2, 1 \rangle$$

- Notice that there are infinitely many timed words that encode one computation, This is due to the choice of time stamps.

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
- a_1's and a_2's occur at different time stamps.

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
  - a_1's and a_2's occur at different time stamps.
- 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 $\mathcal{L}(\mathcal{A}_{undec}) \neq T\Sigma^*$

M accepts w $\Leftrightarrow$ $L_{undec} \neq \phi$

$\Leftrightarrow$ $L_{undec}^c \neq T\Sigma^*$ (universal)

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

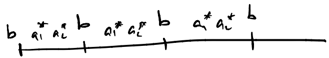
- either, there is **no** b**-symbol** at some **integer** point j

or, two ai's occur at the same time stamp.

$\overline{L_{undec}}$: 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, two a_i's occur at the same time stamp.
- 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, two a's occur at the same time stamp
- 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, two a's occur at the same time stamp
- 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, two a's occur at the same time stamp
- 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, two a's occur at the same time stamp 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**

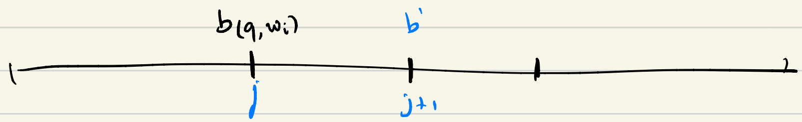
Timed word $(\sigma, \tau) \in \overline{L_{undec}}$ iff

- either, there is **no** b**-symbol** at some **integer** point j $\mathcal{A}_0$
- , two a's occur at the same time stamp A'
- 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 $\mathcal{A}_0, A', \mathcal{A}_1, \mathcal{A}_{init}, \mathcal{A}_{t_1}, \ldots, \mathcal{A}_{t_p}, \mathcal{A}_{acc}$

Main challenge:

- Coming up with an automaton A_t



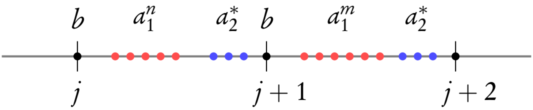
Assume t: $(q,\ 0,\ c{+}{+},\ \perp,\ q')$

There is a violation of t iff there exists a $b_{(q,w_i)}$ s.t. $w_i = 0$

and one of the following occurs:

- the letter at $j+1$ is not $b_{(q', w_{i-1})}$
- there exists an a_1 in ${}^{t\epsilon}(j+1, j+2)$, which is not the last for which there is no predecessor at $t-1$.

Crux

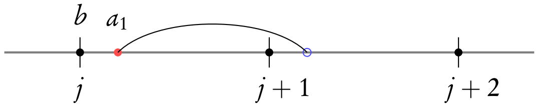


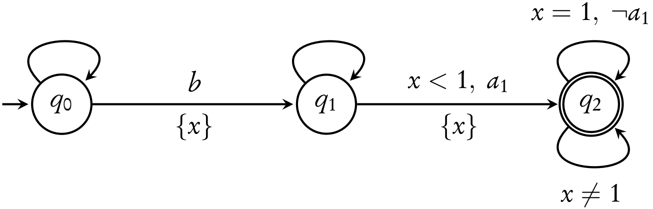
With our encoding, can timed automata express that $n \neq m$?

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 $t-1$

If we give automata for these two languages, then we can find automata for the transition violations (A_t).

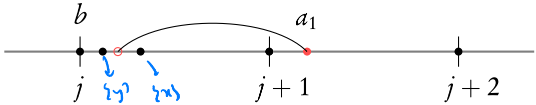
$\exists\, a_1$ at time $t \in (j,\, j+1)$ s.t there is no a_1 at $t+1$

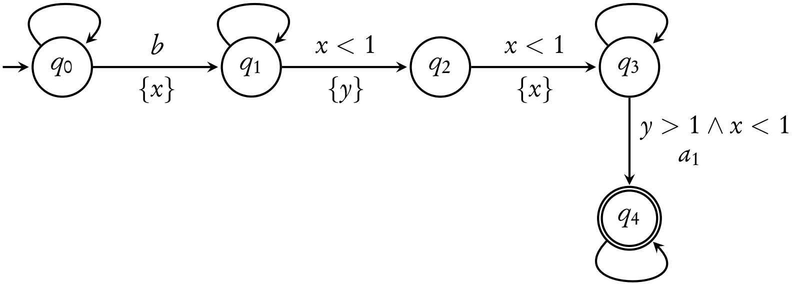




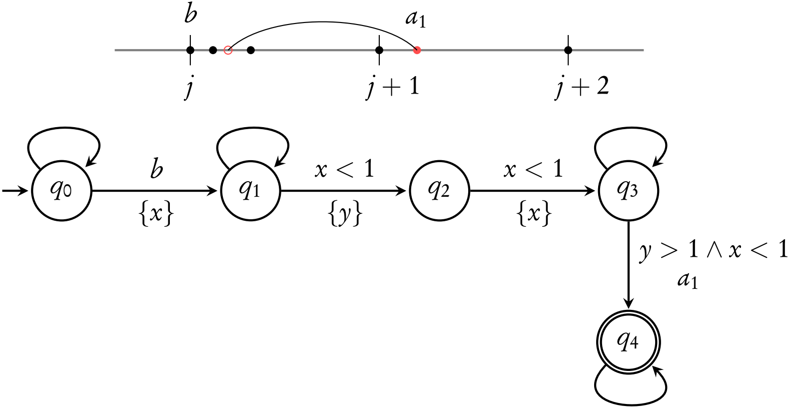
$\exists$ a 'b' and an 'a_1' at time t within 1 time unit of the 'b' s.t. there is no 'a_1' at $t+1$.

$\exists\, a_1$ 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 $\mathcal{A}_0$

  , two a's occur at the same time stamp A'
- 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

A_0 A' A_1 A_{init} A_{t_1} A_{t_2} ..... A_{t_k} , A_{acc}

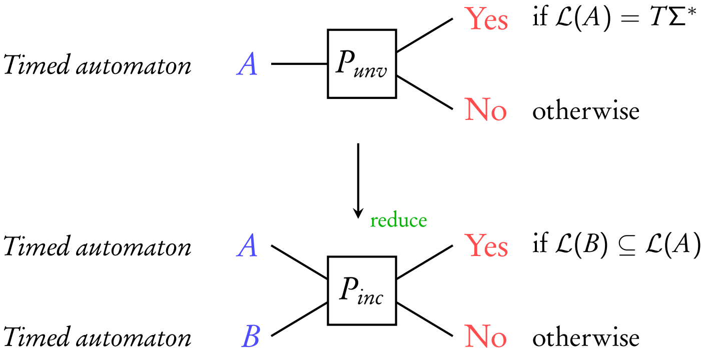
M **accepts** w iff $\mathcal{L}(A_{undec}) \neq T\Sigma^*$

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*