Topics in Timed Automata

B. Srivathsan

RWTH-Aachen
Software modeling and Verification group
If $A$ is \textbf{deterministic}, inclusion can be solved.
Q: Given general $A$ and $B$, can we decide if $\mathcal{L}(B) \subseteq \mathcal{L}(A)$?

If $A$ is deterministic, inclusion can be solved.

System

\[ \mathcal{L}(B) \subseteq \mathcal{L}(A) \]

Specification

Is $\mathcal{L}(B) \cap \overline{\mathcal{L}(A)}$ empty?
Lecture 3:

Language inclusion is undecidable
$P$ : an arbitrary boolean program (string)

$w$ : an arbitrary string

Can program $P_1$ exist?

- **Yes** if $P$ returns Yes on $w$
- **No** if $P$ does not return Yes on $w$
$P$: an arbitrary **boolean program** (string)

$w$: an arbitrary **string**

Can program $P_1$ exist?

$P$ $\quad$ $P_1$ $\quad$ Yes $\quad$ $P$ returns Yes on $w$

$w$ $\quad$ $P_1$ $\quad$ No $\quad$ if $P$ does not return Yes on $w$
If $P_1$ exists, then $P_2$ exists.

$P_2$ returns Yes if $P_2$ does not return Yes on $P_2$.

$P_2$ returns No if $P_2$ returns Yes on $P_2$.

$P_1$ returns Yes if $P$ returns Yes on $w$.

$P_1$ returns No if $P$ does not return Yes on $w$. 

$P$
If $P_1$ exists, then $P_2$ exists

If $P$ returns $\text{Yes}$ on $w$

If $P$ returns $\text{No}$ on $w$

If $P$ does not return $\text{Yes}$ on $w$
If $P_1$ exists, then $P_2$ exists

$P_2$ returns Yes on $P_2$

$P$ returns Yes on $P$ if $P$ does not return Yes on $P$

$P$ returns No on $P$ if $P$ returns Yes on $P$

$P_1$ returns Yes if $P$ returns Yes on $w$

$P_1$ returns No if $P$ does not return Yes on $w$
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 $P_1$ exists, then $P_2$ exists

$P_1$ returns\n
- **Yes** if $P$ returns **Yes** on $w$
- **No** if $P$ does not return **Yes** on $w$

$P_2$ returns\n
- **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 $P_1$ exists, then $P_2$ exists

$P_2$ 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 $P_2$
If $P_1$ exists, then $P_2$ exists

- If $P$ returns Yes on $w$, then $P_1$ returns Yes.
- If $P$ does not return Yes on $w$, then $P_1$ returns No.

- If $P$ returns Yes on $P$, then $P_2$ returns Yes.
- If $P$ returns No on $P$, then $P_2$ returns No.

- $P_2$ returns Yes on $P$, if $P_2$ does not return Yes on $P_2$.
- $P_2$ returns No on $P$, if $P_2$ returns Yes on $P_2$.

$P_2$ cannot exist $\Rightarrow$ $P_1$ cannot exist
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.
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.

Turing machine
2-counter machine

...
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

$P_1$

$M$

Yes

If $M$ accepts $w$

$w$

No

Otherwise

$\Sigma^*$
2-counter machine

\[ M \]
\[ w \] Yes if \( M \) accepts \( w \)
No otherwise

\[ P_1 \]

Timed automaton

\[ A \]
\[ P_{unv} \]

Yes if \( L(A) = T\Sigma^* \)
No otherwise
2-counter machine

$M$

$P_1$

$w$

Yes if $M$ accepts $w$

No otherwise

Timed automaton

$A$

$P_{unv}$

Yes if $\mathcal{L}(A) = T\Sigma^*$

No otherwise

Timed automaton

$A$

$P_{inc}$

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

No otherwise
2-counter machine \( M \)

\( w \)

Yes if \( M \) accepts \( w \)

No otherwise

reduce

Timed automaton \( A \)

\( P_{unv} \)

Yes if \( L(A) = \top \)

No otherwise
## 2-counter machines

**Read-only input tape**

$$w \quad \$ \quad 1 \quad 0 \quad 0 \quad 0 \quad 1 \quad 1 \quad 1 \quad 1 \quad 1 \quad 0 \quad 0 \quad 1 \quad \$\$$

**Counter C**

$$c$$

**Counter D**

$$d$$

**Finite control**

### 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 $$\langle q_F, *, *, * \rangle$$
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 \[ i \in Q \times \{ 0, \ldots, |w| + 1 \} \]

\[ b(q,k) a_1^c a_2^d \]
Encode the $j^{th}$ configuration in $[j, j+1)$
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
\( L_{\text{undec}} \) encodes the \textbf{accepting computations}

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

\[
\sigma = b_{i_0}a_{\sigma_0}^d a_{\tau_0}^d b_{i_1}a_{\sigma_1}^d a_{\tau_1}^d \cdots b_{i_m}a_{\sigma_m}^d a_{\tau_m}^d \quad \text{s.t.} \\
\langle q_0, w_{i_0}, c_0, d_0 \rangle \langle q_1, w_{i_1}, c_1, d_1 \rangle \cdots \langle q_m, w_{i_m}, c_m, d_m \rangle \text{ is accepting}
\]
$L_{\text{undec}}$: encodes the **accepting computations**

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

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

- each $b_{i_j}$ occurs at time $j$
$L_{undec}$: encodes the accepting computations

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

- $\sigma = b_0a_1^0a_2^0 b_1a_1^1a_2^2 \cdots b_ma_1^m a_2^m$ 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 $b_i$ 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 $A_{\text{undec}}$

for the complement language $L_{\text{undec}}$
Goal 2

Given $M$ and $w$

**construct** a timed automaton $A_{\text{undec}}$

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

$M$ accepts $w$  iff  $L(A_{\text{undec}}) \neq T\Sigma^*$
Goal 2

Given $M$ and $w$

**construct** a timed automaton $A_{\text{undec}}$

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

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

$\rightarrow$ reduction to universality of TA
\( L_{\text{undec}} \): words that do not encode accepting computations

Timed word \((\sigma, \tau) \in L_{\text{undec}}\) iff
$L_{\text{undec}}$: words that do not encode accepting computations

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

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

Required $L_{\text{undec}}$: union of $A_0$, $A_1$, $A_{\text{init}}$, $A_{t_1}$,..., $A_{t_p}$, $A_{\text{acc}}$
$L_{\text{undec}}$: words that do not encode accepting computations

Timed word $(\sigma, \tau) \in L_{\text{undec}}$ 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 $a_1^*a_2^*$
$\overline{L_{\text{undec}}}$: words that do not encode accepting computations

Timed word $(\sigma, \tau) \in \overline{L_{\text{undec}}}$ 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
$L_{\text{undec}}$: words that do not encode accepting computations

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

- either, there is no \textit{b-symbol} at some \textit{integer} point $j$
- or, there is a $(j, j + 1)$ with a subsequence \textit{not} of the form $a_1^*a_2^*$
- or, initial subsequence in $[0, 1)$ is \textit{wrong}
- or, some transition of $M$ has been \textit{violated} in the word
\( \overline{L_{\text{undec}}}: \text{words that do not encode accepting computations} \)

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

- either, there is no \textit{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 \textit{violated} in the word
- or, final \textit{b}-symbol denotes \textit{non-accepting} state
\( L_{\text{undec}} \): words that do not encode accepting computations

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

- either, there is no \(b\)-symbol at some integer point \(j\) \(A_0\)
- or, there is a \((j, j + 1)\) with a subsequence not of the form \(a_1^*a_2^*\) \(A_1\)
- or, initial subsequence in \([0, 1)\) is wrong \(A_{\text{init}}\)
- or, some transition of \(M\) has been violated in the word \(A_t\) for each transition \(t\) of \(M\)
- or, final \(b\)-symbol denotes non-accepting state \(A_{\text{acc}}\)
\( \overline{L_{\text{undec}}} \): words that do not encode accepting computations

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

- either, there is no \( b \)-symbol at some integer point \( j \) \( A_0 \)
- or, there is a \((j, j + 1)\) with a subsequence not of the form \( a_1^*a_2^* \) \( A_1 \)
- or, initial subsequence in \([0, 1)\) is wrong \( A_{\text{init}} \)
- or, some transition of \( M \) has been violated in the word \( A_t \) for each transition \( t \) of \( M \)
- or, final \( b \)-symbol denotes non-accepting state \( A_{\text{acc}} \)

Required \( \mathcal{A}_{\text{undec}} \): union of \( A_0, A_1, A_{\text{init}}, A_t, \ldots, A_t, A_{\text{acc}} \)
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 \( t - 1 \)
\( \exists a_1 \text{ at time } t \in (j, j+1) \text{ s.t there is no } a_1 \text{ at } t+1 \)
\[ \exists a_1 \text{ at time } t \in (j + 1, j + 2) \text{ s.t. there is no } a_1 \text{ at } t - 1 \]
\[ \exists a_1 \text{ at time } t \in (j + 1, j + 2) \text{ s.t. there is no } a_1 \text{ at } t - 1 \]

Need only **two clocks**!
$\overline{L_{unde}}$: words that do not encode accepting computations

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

- either, there is no $b$-symbol at some integer point $j \ A_0$
- or, there is a $(j, j + 1)$ with a subsequence not of the form $a_1^*a_2^* \ A_1$
- or, initial subsequence in $[0, 1)$ is wrong $A_{init}$
- or, some transition of $M$ has been violated in the word $A_t$ for each transition $t$ of $M$
- or, final $b$-symbol denotes non-accepting state $A_{acc}$

Required $A_{unde}$ can be constructed using two clocks
The universality problem is undecidable for TA with two clocks or more.
Timed automaton $A$  

$P_{unv}$  

Yes  

if $L(A) = TΣ^*$

No  

otherwise

Timed automaton $A$  

$P_{inc}$  

Yes  

if $L(B) \subseteq L(A)$

No  

otherwise

Timed automaton $B$  

reduce

Put $B$ as the trivial single state automaton accepting $TΣ^*$

$L(A) = TΣ^*$  iff  $L(B) \subseteq L(A)$
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
$L(B) \subseteq L(A)$ is decidable when $A$ has at most 1 clock

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