TIMED AUTOMATA

LECTURE 15

TIMED AUTOMATA

Determinism

- DTA $\subset$ NTA
- Event-clock automata

Emptiness

- Region automaton
- Zone graph + simulations

Universality / Inclusion

- DTA : use the complement
- NTA : undecidable with ≥ 2 clocks

Advanced topics

Let $T\Sigma^*$ denote the set of **all timed words**

Universality: Given A, is $\mathcal{L}(A) = T\Sigma^*$?

Inclusion: Given A, B, is $\mathcal{L}(B) \subseteq \mathcal{L}(A)$? $\Leftrightarrow$ $\mathcal{L}(B) \cap \mathcal{L}(A)^c \neq \phi$

Universality and inclusion are **undecidable** when A has **two clocks** or more

A theory of timed automata

Alur and Dill. *TCS'94*

A decidable case of the inclusion problem

Universality: Given A, is $\mathcal{L}(A) = T\Sigma^*$?

Inclusion: Given A, B, is $\mathcal{L}(B) \subseteq \mathcal{L}(A)$? $\Leftrightarrow$ $\mathcal{L}(B) \cap \mathcal{L}(A)^c \neq \phi$

One-clock restriction

Universality and inclusion are **decidable** when A has at most **one clock**

On the language inclusion problem for timed automata: Closing a decidability gap

Ouaknine and Worrell. *LICS'05*

We still cannot hope to solve these problems by taking the complement of A, because there are 1-clock T.A. that cannot be complemented.

Universality: Given A, is $\mathcal{L}(A) = T\Sigma^*$?

Inclusion: Given A, B, is $\mathcal{L}(B) \subseteq \mathcal{L}(A)$?

One-clock restriction

Universality and inclusion are **decidable** when A has at most **one clock**

On the language inclusion problem for timed automata: Closing a decidability gap

Ouaknine and Worrell. *LICS'05*

In this lecture: universality for one clock TA

Step 0:

Well-quasi orders and Higman's Lemma

Quasi-order

Given a set $\mathcal{Q}$, a quasi-order is a **reflexive** and **transitive** relation:

$$\sqsubseteq \quad \subseteq \quad \mathcal{Q} \times \mathcal{Q}$$

- $(\mathbb{N}, \leq)$
- $(\mathbb{Z}, \leq)$

$$\text{Let } \Lambda = \{A, B, \ldots, Z\}, \quad \Lambda^* = \{\text{set of words}\}$$

- $(\Lambda^*,$ lexicographic order $\sqsubseteq_L)$: $AAAB \sqsubseteq_L AAB \sqsubseteq_L AB$
- $(\Lambda^*,$ prefix order $\subseteq_P)$: $AB \subseteq_P ABA \subseteq_P ABAA$
- $(\Lambda^*,$ subword order $\preccurlyeq)$ $HIGMAN \preccurlyeq HIGHMOUNTAIN$ [OW'05]

Well-quasi-order

$q_1 \quad q_2 \ldots\ldots q_i \overset{\sqsubseteq}{\cdots} q_j$

An infinite sequence $\langle q_1, q_2, \ldots \rangle$ in $(Q, \sqsubseteq)$ is saturating if $\exists\, i < j : q_i \sqsubseteq q_j$

A quasi-order $\sqsubseteq$ is a well-quasi-order (wqo) if **every** infinite sequence is saturating

- $(\mathbb{N}, \leq)$ Wqo. No infinite decreasing seq.
- $(\mathbb{Z}, \leq)$ ✗ $\quad 0 \quad -1 \quad -2 \quad -3 \ldots\ldots$
- $(\Lambda^*$, lexicographic order $\sqsubseteq_L)$:
- $(\Lambda^*$, prefix order $\subseteq_P)$:
- $(\Lambda^*$, subword order $\preccurlyeq)$

Well-quasi-order

An infinite sequence $\langle q_1, q_2, \ldots \rangle$ in $(\mathcal{Q}, \sqsubseteq)$ is saturating if $\exists\, i < j : q_i \sqsubseteq q_j$

A quasi-order $\sqsubseteq$ is a well-quasi-order (wqo) if **every** infinite sequence is saturating

- $(\mathbb{N}, \leq)$ $\surd$
- $(\mathbb{Z}, \leq)$ $\times$ $-1 \geq -2 \geq -3, \ldots$
- $(\Lambda^*,$ lexicographic order $\sqsubseteq_L)$: $\times$ $B \sqsupseteq_L AB \sqsupseteq_L AAB \ldots$
- $(\Lambda^*,$ prefix order $\subseteq_P)$: $\times$ $B, AB, AAB, \ldots$
- $(\Lambda^*,$ subword order $\preccurlyeq)$

$B \not\subseteq_P AB$

$AB \not\subseteq_P B$

$A^i B \not\subseteq_P A^j B$

Well-quasi-order

An infinite sequence $\langle q_1, q_2, \ldots \rangle$ in $(\mathcal{Q}, \sqsubseteq)$ is saturating if $\exists\, i < j : q_i \sqsubseteq q_j$

A quasi-order $\sqsubseteq$ is a well-quasi-order (wqo) if **every** infinite sequence is saturating

- $(\mathbb{N}, \leq)$ $\surd$
- $(\mathbb{Z}, \leq)$ $\times$ $-1 \geq -2 \geq -3, \ldots$
- $(\Lambda^*,$ lexicographic order $\sqsubseteq_L)$: $\times$ $B \sqsupseteq_L AB \sqsupseteq_L AAB \ldots$
- $(\Lambda^*,$ prefix order $\subseteq_P)$: $\times$ $B,\ AB,\ AAB, \ldots$
- $(\Lambda^*,$ subword order $\preccurlyeq)$?

Higman's lemma

Let $\sqsubseteq$ be a quasi-order on Λ

Define the induced **monotone domination order** $\preccurlyeq$ on Λ^* as follows:

$a_1 \ldots a_m \ \preccurlyeq\ b_1 \ldots b_n$ if there exists a **strictly increasing** function

$$f : \{1, \ldots, m\} \mapsto \{1, \ldots, n\} \text{ s.t}$$

$$\forall\, 1 \leq i \leq m :\ a_i \ \sqsubseteq\ b_{f(i)}$$

$a_i \ \sqsubseteq\ b_{f(i)}$

Higman's lemma

Let $\sqsubseteq$ be a quasi-order on Λ

Define the induced **monotone domination order** $\preccurlyeq$ on Λ^* as follows:

$a_1 \dots a_m \preccurlyeq b_1 \dots b_n$ if there exists a **strictly increasing** function $f : \{1, \dots, m\} \mapsto \{1, \dots, n\}$ s.t $\forall\, 1 \leq i \leq m : a_i \sqsubseteq b_{f(i)}$

Higman'52

If $\sqsubseteq$ is a wqo on Λ, then the induced monotone domination order $\preccurlyeq$ is a wqo on Λ^*

Subword order

$$\begin{aligned} \Lambda &:= \{A, B, \ldots, Z\} \\ \sqsubseteq &:= x \sqsubseteq y \text{ if } x = y \end{aligned}$$

Subword order

$$\Lambda \;:=\; \{A, B, \ldots, Z\}$$
$$\sqsubseteq \;:=\; x \sqsubseteq y \text{ if } x = y$$

$\sqsubseteq$ is a **wqo** as Λ is **finite**

Subword order

$$\Lambda := \{A, B, \ldots, Z\}$$
$$\sqsubseteq := x \sqsubseteq y \text{ if } x = y$$

$\sqsubseteq$ is a **wqo** as Λ is **finite**

Induced monotone domination order $\preccurlyeq$ is the subword order

$$HIGMAN \preccurlyeq HIGHMOUNTAIN$$

f:

$1 \to 1$

$2 \to 2$

$3 \to 3$

$4 \to 5$

$5 \to 10$

$6 \to 12$

Subword order

$$\Lambda := \{A, B, \ldots, Z\}$$
$$\sqsubseteq := x \sqsubseteq y \text{ if } x = y$$

$\sqsubseteq$ is a **wqo** as Λ is **finite**

Induced monotone domination order $\preccurlyeq$ is the subword order

$$HIGMAN \preccurlyeq HIGHMOUNTAIN$$

By Higman's lemma, $\preccurlyeq$ is a wqo too

If we start writing an **infinite sequence** of words, we will **eventually** write down a **superword** of an earlier word in the sequence

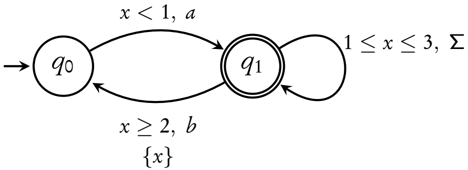
Step 1:

A naive procedure for universality of one-clock TA

Terminology

Let $A = (Q, \Sigma, Q_0, \{x\}, T, F)$ be a timed automaton with one clock

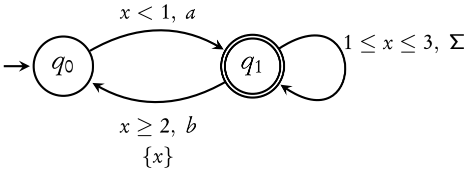
- **Location:** $q_0, q_1, \dots \in Q$ q_1
- **State:** (q, u) where $u \in \mathbb{R}_{\geq 0}$ gives value of the clock $(q_1, 2.5)$
- **Configuration:** **finite** set of states $\{(q_1, 2.5), (q_0, 1.5), (q_0, 7.7)\}$



Terminology

Let $A = (Q, \Sigma, Q_0, \{x\}, T, F)$ be a timed automaton with one clock

- **Location:** $q_0, q_1, \dots \in Q$
- **State:** (q, u) where $u \in \mathbb{R}_{\geq 0}$ gives value of the clock
- **Configuration:** **finite** set of states $\{(q_1, 2.3), (q_0, 0)\}$



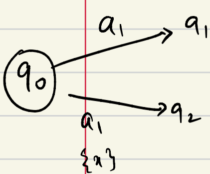
Goal: We want to check if $\mathcal{L}(A)$ is universal.

- Suppose A was deterministic.

  ↳ complement and check for emptiness.

- Else, A is non-deterministic:

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



$\{(q_0, 0)\}$

δ_1, a_1

$\{(q_1, \delta_1)(q_2, 0)\}$

δ_2, a_2

$\{(\quad)(\quad)(\quad)(\quad)\}$

$\delta_i = t_i - t_{i-1}$

$t_0 = 0$

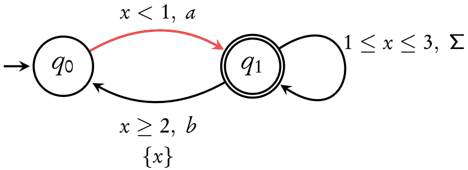
If neither q_1 nor q_2 is accepting, then we can be sure that (a_1, t_1) is not accepted by A, since we have accumulated all the possible states to which A goes on reading (a_1, t_1).

- If we have such a graph, then checking whether a timed word is accepted boils down to moving down to the unique node in this graph that represents the timed word. That node contains all states reached by A on the word. If the node contain no accepting location, then the word is rejected. Else accepted.

- Does there exist a word which is rejected?

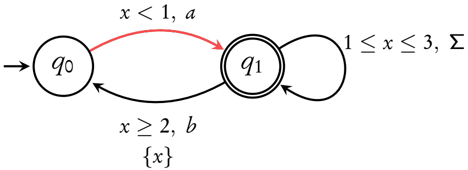
Transition between configurations:

$\{(q_0, 0)\} \xrightarrow{0.2,\ a}$



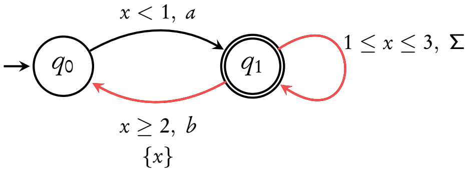
Transition between configurations:

$$\{(q_0, 0)\} \xrightarrow{0.2,\ a} \{(q_1, 0.2)\}$$



Transition between configurations:

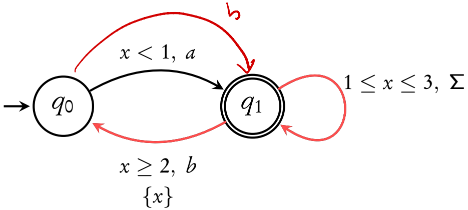
$$\{(q_0, 0)\} \xrightarrow{0.2,\ a} \{(q_1, 0.2)\} \xrightarrow{2.1,\ b}$$



Transition between configurations:

$$\{(q_0, 0)\} \xrightarrow{0.2,\ a} \{(q_1, 0.2)\} \xrightarrow{2.1,\ b} \{(q_1, 2.3), (q_0, 0)\} \ldots \xrightarrow{0.1,\ b} \{(q_1, 2.4), (q_0, 0), (q_1, 0.1)\}$$

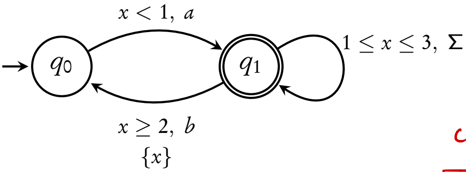
$$\big\downarrow\ 0.01, b$$

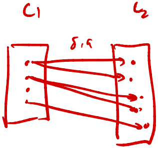


Transition between configurations:

$$\{(q_0,0)\} \xrightarrow{0.2,\ a} \{(q_1,0.2)\} \xrightarrow{2.1,\ b} \{(q_1,2.3),(q_0,0)\} \ldots$$

$C_1 \xrightarrow{2.1,\ b} C_2$



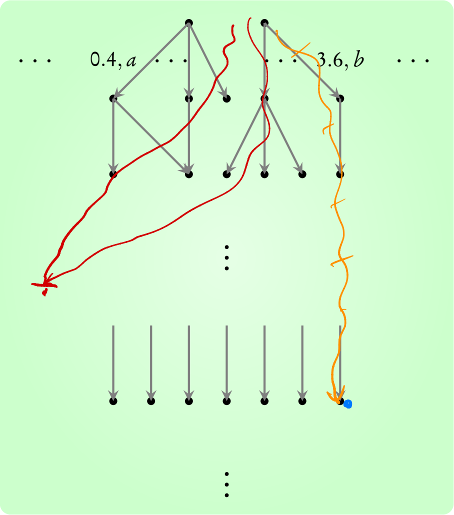


$C_1 \xrightarrow{\delta_1, a_1} C_2 \xrightarrow{\delta_2, a_2} C_3$

$$C_1 \xrightarrow{\delta,\ a} C_2 \text{ if}$$

$$C_2 = \{\ (q_2,u_2) \mid \exists (q_1,u_1) \in C_1 \text{ s. t. } (q_1,u_1) \xrightarrow{\delta,\ a} (q_2,u_2)\}$$

Labeled transition system of **configurations**

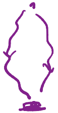


There is a correspondence between nodes of this transition system and timed words.

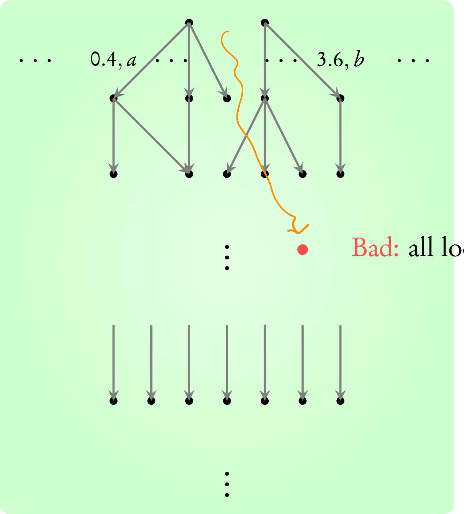
Every time word can be mapped to a unique node.

- Every node with a non-empty set of states can be mapped to a unique timed word.

Not true!



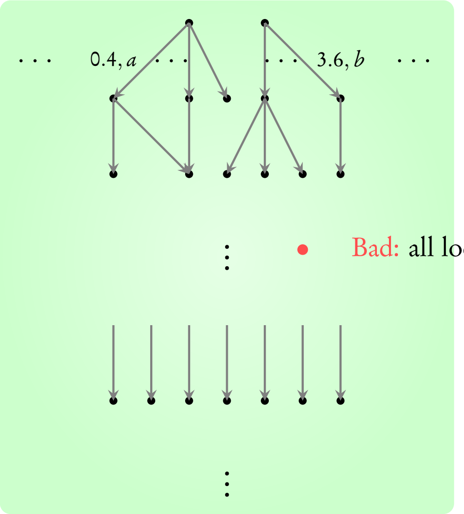
Labeled transition system of **configurations**



Bad: all locations **non-accepting**

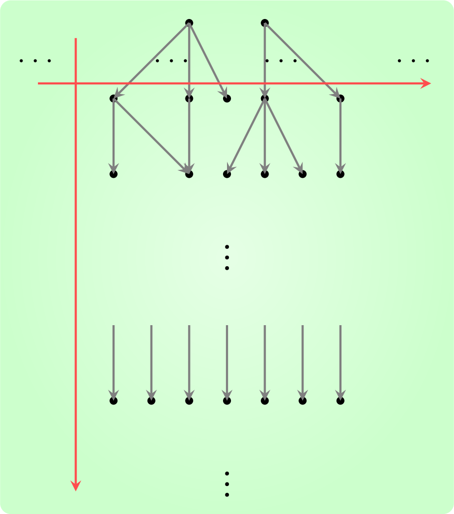
$\Rightarrow$ The word reaching this node is not accepted by $\mathcal{A}$.

Labeled transition system of **configurations**

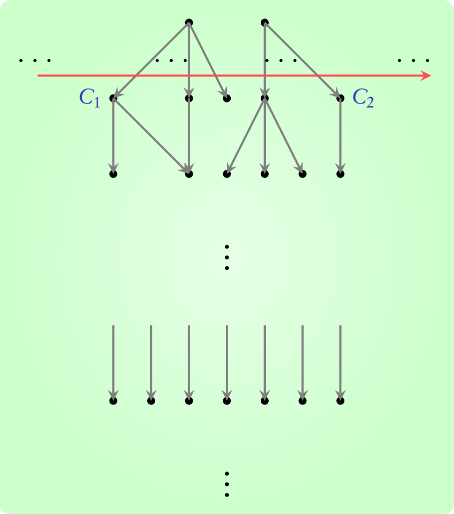


● Bad: all locations **non-accepting**

Is a bad configuration **reachable** from some **initial** configuration?



Need to handle **two dimensions** of infinity!

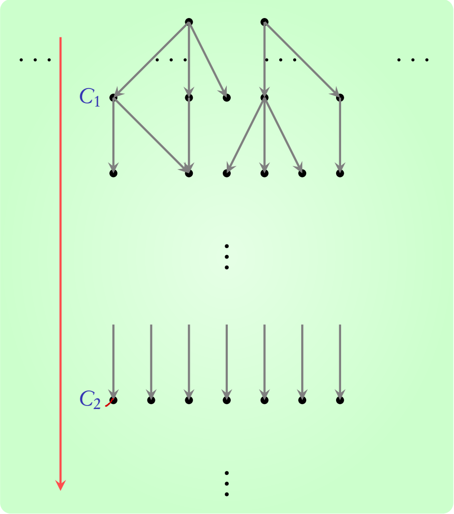


abstraction by **equivalence** $\sim$

$C_1 \sim C_2$ ~~iff~~ will imply

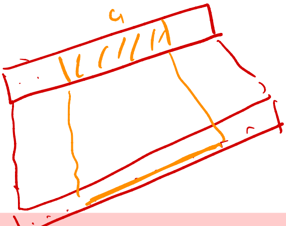
C_1 goes to a **bad** config. $\Leftrightarrow$ C_2 goes to a **bad** config.

finite **domination** order $\preccurlyeq$



suppose $c_1 \subseteq c_2$

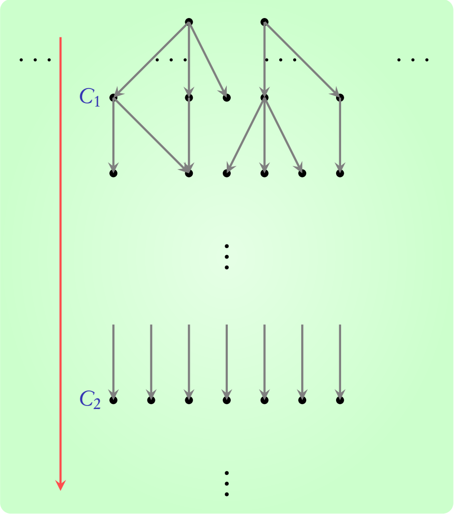
c_2



all non-acc.

$C_1 \preccurlyeq C_2$ ~~iff~~ should imply

C_2 goes to a **bad** config $\Rightarrow$ C_1 goes to a **bad** config. too finite **domination** order $\preccurlyeq$



$C_1 \preccurlyeq C_2$ ~~iff~~: should imply

C_2 goes to a **bad** config $\quad \Rightarrow \quad$ C_1 goes to a **bad** config. too

No need to explore C_2!

$\mathcal{A}$: one-dca:

Infinite labeled transition system
of configurations

Horizontal infinity

$\sim$ equivalence

Vertical infinity

$\precsim$ finite domination order.

$\sim$ and $\precsim$ will result
in a finite graph.

$\sim$: next class.