Topics in Timed Automata

B. Srivathsan

RWTH-Aachen

Software modeling and Verification group

Theorem (Lecture 2)

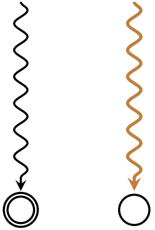
Deterministic timed automata are **closed under complement**

Theorem (Lecture 2)

Deterministic timed automata are **closed under complement**

1. **Unique** run for every timed word

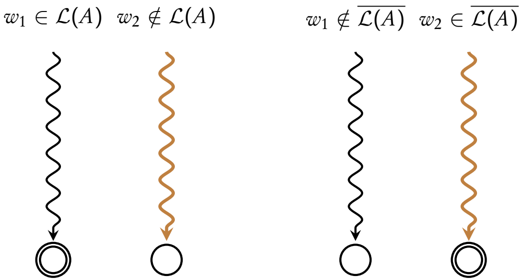
$w_1 \in \mathcal{L}(A)$ $w_2 \notin \mathcal{L}(A)$



Theorem (Lecture 2)

Deterministic timed automata are **closed under complement**

1. **Unique** run for every timed word
2. **Complementation:** **Interchange** acc. and non-acc. states

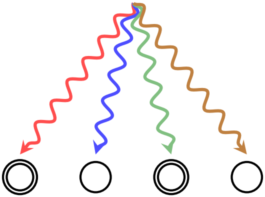


Theorem (Lecture 1)

Non-deterministic timed automata are **not closed under complement**

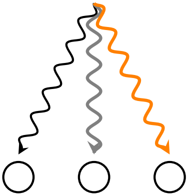
Many runs for a timed word

$w_1 \in \mathcal{L}(A)$



Exists an acc. run

$w_2 \notin \mathcal{L}(A)$

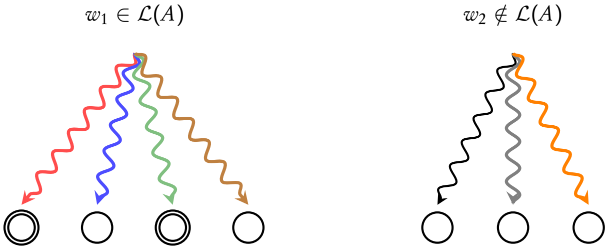


All runs non-acc.

Theorem (Lecture 1)

Non-deterministic timed automata are **not closed under complement**

Many runs for a timed word



Exists an acc. run

All runs non-acc.

Complementation: interchange acc/non-acc + ask are **all runs acc.** ?

A timed automaton model with **existential** and **universal** semantics for acceptance

Lecture 5:
Alternating timed automata

Lasota and Walukiewicz. *FoSSaCS'05*, *ACM TOCL'2008*

Section 1:

Introduction to ATA

- X : set of **clocks**
- $\Phi(X)$: set of clock constraints σ (**guards**)

$$\sigma : \; x < c \mid x \leq c \mid \sigma_1 \wedge \sigma_2 \mid \neg\sigma$$

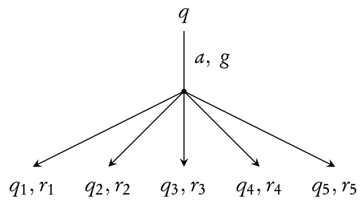
  c is a non-negative **integer**

- Timed automaton A: $(Q, Q_0, \Sigma, X, T, F)$

$$T \;\subseteq\; Q \times \Sigma \times \Phi(X) \times Q \times \mathcal{P}(X)$$

$$T \subseteq Q \times \Sigma \times \Phi(X) \times Q \times \mathcal{P}(X)$$

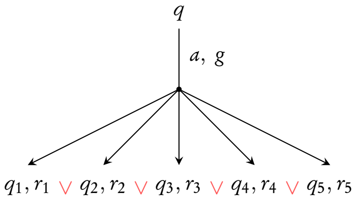
$$T : Q \times \Sigma \times \Phi(X) \mapsto \mathcal{P}(Q \times \mathcal{P}(X))$$



$$T \subseteq Q \times \Sigma \times \Phi(X) \times Q \times \mathcal{P}(X)$$

$$\downarrow$$

$$T : Q \times \Sigma \times \Phi(X) \mapsto \mathcal{P}(Q \times \mathcal{P}(X))$$



$$T : Q \times \Sigma \times \Phi(X) \mapsto \mathcal{P}(Q \times \mathcal{P}(X))$$

$$T : Q \times \Sigma \times \Phi(X) \mapsto \mathcal{P}(Q \times \mathcal{P}(X))$$

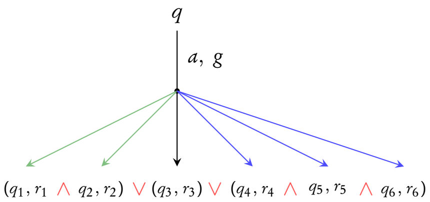
$$\Big\downarrow \; \mathcal{B}^+(S) \text{ is all } \phi ::= S \mid \phi_1 \wedge \phi_2 \mid \phi_1 \vee \phi_2$$

$$T : Q \times \Sigma \times \Phi(X) \;\mapsto\; \mathcal{B}^+(Q \times \mathcal{P}(X))$$

$$T : Q \times \Sigma \times \Phi(X) \mapsto \mathcal{P}(Q \times \mathcal{P}(X))$$

$\big\downarrow$ $\mathcal{B}^+(S)$ is all $\phi ::= S \mid \phi_1 \wedge \phi_2 \mid \phi_1 \vee \phi_2$

$$\boxed{T : Q \times \Sigma \times \Phi(X) \ \mapsto \ \mathcal{B}^+(Q \times \mathcal{P}(X))}$$



Alternating Timed Automata

An **ATA** is a tuple $A = (Q, q_0, \Sigma, X, T, F)$ where:

$$T : Q \times \Sigma \times \Phi(X) \mapsto \mathcal{B}^+(Q \times \mathcal{P}(X))$$

is a **finite partial function.**

Alternating Timed Automata

An **ATA** is a tuple $A = (Q, q_0, \Sigma, X, T, F)$ where:

$$T : Q \times \Sigma \times \Phi(X) \mapsto \mathcal{B}^+(Q \times \mathcal{P}(X))$$

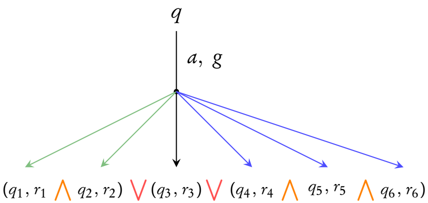
is a **finite partial function.**

Partition: For every q, a the set

$$\{ [\sigma] \mid T(q, a, \sigma) \text{ is defined } \}$$

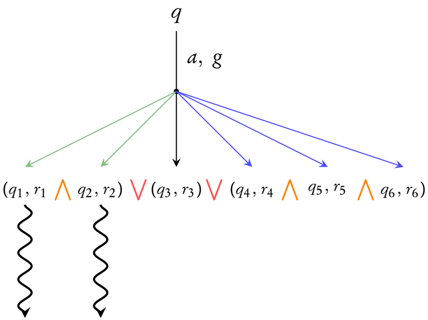
gives a finite partition of $\mathbb{R}_{\geq 0}^X$

Acceptance



Accepting run from q iff:

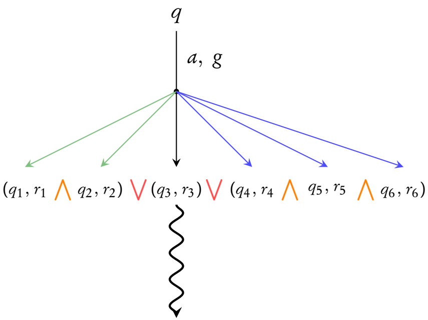
Acceptance



Accepting run from q iff:

- accepting run from q_1 **and** q_2,

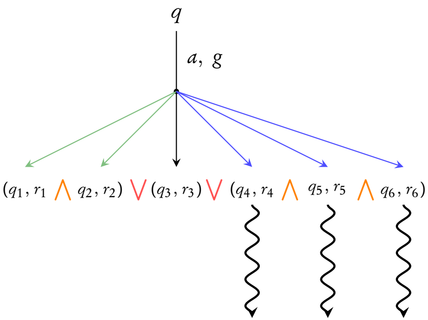
Acceptance



Accepting run from q iff:

- accepting run from q_1 **and** q_2,
- **or** accepting run from q_3,

Acceptance



Accepting run from q iff:

- accepting run from q_1 **and** q_2,
- **or** accepting run from q_3,
- **or** accepting run from q_4 **and** q_5 **and** q_6

L : timed words over $\{a\}$ containing **no two** $a's$ at distance 1

(Not expressible by non-deterministic TA)

L : timed words over $\{a\}$ containing **no two** $a's$ at distance 1

(Not expressible by non-deterministic TA)

ATA:

$$q_0, a, tt \;\mapsto\; (q_0, \emptyset) \wedge (q_1, \{x\})$$

$$q_1, a, x = 1 \;\mapsto\; (q_2, \emptyset)$$

$$q_1, a, x \neq 1 \;\mapsto\; (q_1, \emptyset)$$

$$q_2, a, tt \;\mapsto\; (q_2, \emptyset)$$

q_0, q_1 are acc., q_2 is non-acc.

Closure properties

- Union, intersection: use disjunction/conjunction
- Complementation: **interchange**
  1. acc./non-acc.
  2. conjunction/disjunction

Closure properties

- Union, intersection: use disjunction/conjunction
- Complementation: **interchange**
  1. acc./non-acc.
  2. conjunction/disjunction

No change in the number of clocks!

Section 2:

The 1-clock restriction

- Emptiness: given A, is $\mathcal{L}(A)$ empty
- Universality: given A, does $\mathcal{L}(A)$ contain all timed words
- Inclusion: given A, B, is $\mathcal{L}(A) \subseteq \mathcal{L}(B)$

- Emptiness: given A, is $\mathcal{L}(A)$ empty
- Universality: given A, does $\mathcal{L}(A)$ contain all timed words
- Inclusion: given A, B, is $\mathcal{L}(A) \subseteq \mathcal{L}(B)$

Undecidable for **two clocks or more** (via Lecture 3)

- Emptiness: given A, is $\mathcal{L}(A)$ empty
- Universality: given A, does $\mathcal{L}(A)$ contain all timed words
- Inclusion: given A, B, is $\mathcal{L}(A) \subseteq \mathcal{L}(B)$

Undecidable for **two clocks or more** (via Lecture 3)

Decidable for **one clock** (via Lecture 4)

- Emptiness: given A, is $\mathcal{L}(A)$ empty
- Universality: given A, does $\mathcal{L}(A)$ contain all timed words
- Inclusion: given A, B, is $\mathcal{L}(A) \subseteq \mathcal{L}(B)$

Undecidable for **two clocks or more** (via Lecture 3)

Decidable for **one clock** (via Lecture 4)

Restrict to one-clock ATA

Theorem

Languages recognizable by 1-clock ATA and (many clock) TA are **incomparable**

$\rightarrow$ proof on the board

Section 3:

Complexity

Lower bound

Complexity of emptiness of **purely universal** 1-clock ATA is **not** bounded by a **primitive recursive** function

Lower bound

Complexity of emptiness of **purely universal** 1-clock ATA is **not** bounded by a **primitive recursive** function

$\Rightarrow$ complexity of Ouaknine-Worrell algorithm for **universality** of 1-clock TA is **non-primitive recursive**

Primitive recursive functions

Functions $f : \mathbb{N} \mapsto \mathbb{N}$

Basic primitive recursive functions:

- **Zero function:** $Z() = \mathtt{0}$
- **Successor function:** $Succ(n) = n + 1$
- **Projection function:** $P_i(x_1, \ldots, x_n) = x_i$

Operations:

- **Composition**
- **Primitive recursion:** if f and g are p.r. of arity k and $k + 2$, there is a p.r. h of arity $k + 1$:

$$\begin{aligned} h(\mathtt{0}, x_1, \ldots, x_k) &= f(x_1, \ldots, x_k) \\ h(n + 1, x_1, \ldots, x_k) &= g(h(n, x_1, \ldots, x_k), n, x_1, \ldots, x_k) \end{aligned}$$

Addition:

$$\begin{aligned} Add(\mathtt{0}, y) &= y \\ Add(n+1, y) &= Succ(Add(n, y)) \end{aligned}$$

Addition:

$$
\begin{aligned}
Add(\mathtt{0}, y) &= y \\
Add(n+1, y) &= Succ(Add(n, y))
\end{aligned}
$$

Multiplication:

$$
\begin{aligned}
Mult(\mathtt{0}, y) &= Z() \\
Mult(n+1, y) &= Add(Mult(n, y), y)
\end{aligned}
$$

Addition:

$$\begin{aligned} Add(0, y) &= y \\ Add(n+1, y) &= Succ(Add(n, y)) \end{aligned}$$

Multiplication:

$$\begin{aligned} Mult(0, y) &= Z() \\ Mult(n+1, y) &= Add(Mult(n, y), y) \end{aligned}$$

Exponentiation 2^n:

$$\begin{aligned} Exp(0) &= Succ(Z()) \\ Exp(n+1) &= Mult(Exp(n), 2) \end{aligned}$$

Addition:

$$\begin{aligned} Add(\mathtt{0}, y) &= y \\ Add(n+1, y) &= Succ(Add(n, y)) \end{aligned}$$

Multiplication:

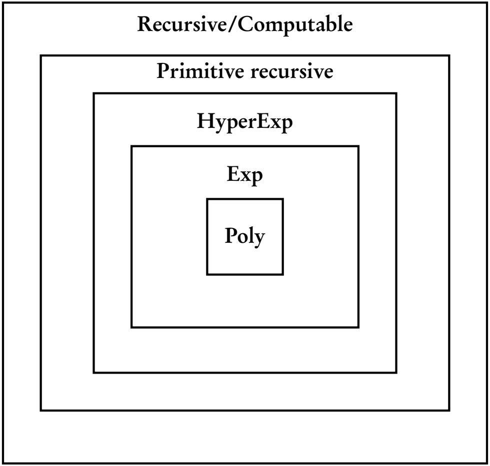
$$\begin{aligned} Mult(\mathtt{0}, y) &= Z() \\ Mult(n+1, y) &= Add(Mult(n, y), y) \end{aligned}$$

Exponentiation 2^n:

$$\begin{aligned} Exp(\mathtt{0}) &= Succ(Z()) \\ Exp(n+1) &= Mult(Exp(n), 2) \end{aligned}$$

Hyper-exponentiation (tower of n two-s):

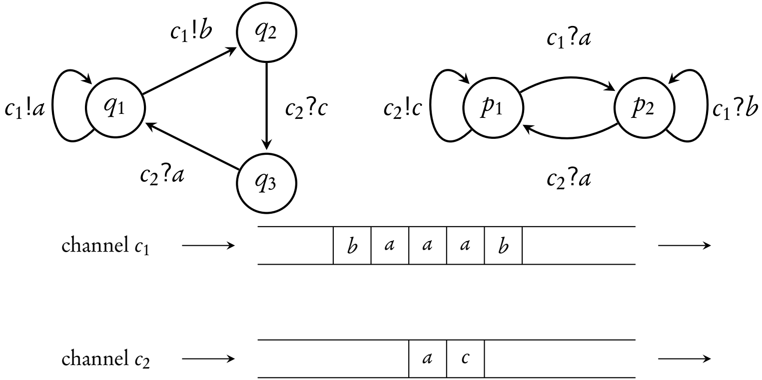
$$\begin{aligned} HyperExp(\mathtt{0}) &= Succ(Z()) \\ HyperExp(n+1) &= Exp(HyperExp(n)) \end{aligned}$$



Recursive but not primitive rec.: Ackermann function, Sudan function

Coming next: a problem that has complexity non-primitive recursive

Channel systems



Finite state description of communication protocols

G. von Bochmann. 1978

On communicating finite-state machines

D. Brand and P. Zafiropulo. 1983

Example from Schnoebelen'2002

Theorem [BZ'83]

Reachability in channel systems is **undecidable**

Coming next: modifying the model for decidability

Lossy channel systems

Finkel'94, Abdulla and Jonsson'96

Messages stored in channel can be **lost** during transition

Lossy channel systems

Finkel'94, Abdulla and Jonsson'96

Messages stored in channel can be **lost** during transition

Theorem [Schnoebelen'2002]

Reachability for **lossy one-channel** systems is **non-primitive recursive**

Reachability problem for **lossy one-channel** systems can be reduced to emptiness problem for **purely universal 1-clock ATA**

1-clock ATA

- **closed** under boolean operations
- **decidable** emptiness problem
- expressivity **incomparable** to many clock TA
- **non-primitive recursive** complexity for emptiness

1-clock ATA

- **closed** under boolean operations
- **decidable** emptiness problem
- expressivity **incomparable** to many clock TA
- **non-primitive recursive** complexity for emptiness

- Other results: **Undecidability** of:
  - 1-clock ATA + ε-transitions
  - 1-clock ATA over infinite words

Summary of Part 1 of the course

- Lecture 1: Expressiveness, ε-transitions
- Lecture 2: Determinization
- Lecture 3: Universality and inclusion
- Lecture 4: Restriction to one-clock
- Lecture 5: Alternating timed automata