Modeling and verifying reactive systems

Temporal logics

Nicolas Markey

Lab. Specification et Verification
ENS Cachan & CNRS, France

Outline of the course

1. Branching-time temporal logics
   - Complexity
   - Alternating-time Temporal Logic

2. Timed temporal logics
   - Timed models
   - Timed logics
   - Undecidability

CTL* model-checking

Theorem

CTL model-checking is* PSPACE*-complete.*

CTL* model-checking

Theorem

CTL* model-checking is PSPACE*-complete.*

Proof.

- hardness in PSPACE: CTL* subsumes LTL.
- membership in PSPACE: labeling algorithm involving LTL model-checking algorithm.

□

ECTL^{+} model-checking

Theorem

ECTL^{+} model-checking is Δ_2^{P}-complete.

ECTL$^+$ model-checking

Theorem

ECTL$^+$ model-checking is Δ_2^P-complete.

Proof.

- hardness in NP: easy encoding of SAT as a CTL$^+$ model-checking problem.
  Hardness in Δ_2^P is an intricate extension of that encoding.
- membership in Δ_2^P: using an oracle for deciding LTL_1-subformulas;

□

Outline of the course

1. Branching-time temporal logics
   - Complexity
   - Alternating-time Temporal Logic
2. Timed temporal logics
   - Timed models
   - Timed logics
   - Undecidability

Multi-agent systems

Problem

The CTL formula

$$\mathbf{A}\,\mathbf{G}(\mathbf{E}\,\mathbf{F}\,\textit{cabin.ground floor})$$

is not exactly what we mean with

it is always possible to reach the ground floor.

Multi-agent systems

Problem

The CTL formula

$$\mathbf{A}\,\mathbf{G}(\mathbf{E}\,\mathbf{F}\,\mathit{cabin.ground\ floor})$$

is not exactly what we mean with

it is always possible to reach the ground floor.

We rather mean that there is a strategy that makes the cabin eventually reach the ground floor. Moreover, we'd prefer that this strategy only involves the button at the third floor (say) and the buttons in the cabin.

Multi-agent systems

Definition

A *CGS* $\mathcal{C}$ is a 6-tuple $(Q, \mathrm{AP}, \ell, \mathbb{A}, \mathrm{Mv}, \mathrm{Edg})$ s.t:

- Q: a finite set of *locations*;
- AP: a finite set of *atomic propositions*;
- $\ell \colon Q \to 2^{\mathrm{AP}}$: a labeling function;
- $\mathbb{A} = \{A_1, ..., A_k\}$: a set of *agents* (or *players*);
- $\mathrm{Mv} \colon Q \times \mathbb{A} \to \mathcal{P}(\mathbb{Z}^+)$ the choice function.
  $\mathrm{Mv}(\ell, A_i)$ = set of possible moves for player A_i from ℓ.
- $\mathrm{Edg} \colon Q \times \mathbb{Z}^{+k} \to Q$: the transition table.

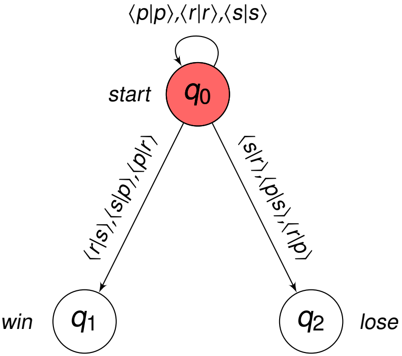
Semantics of CGSs

- From a location ℓ, each agent A_i chooses some m_{A_i} with

$$m_{A_i} \in \mathrm{Mv}(\ell, A_i).$$

- $\mathrm{Edg}(\ell, m_{A_1}, \cdots, m_{A_k})$ gives the new location.

Example

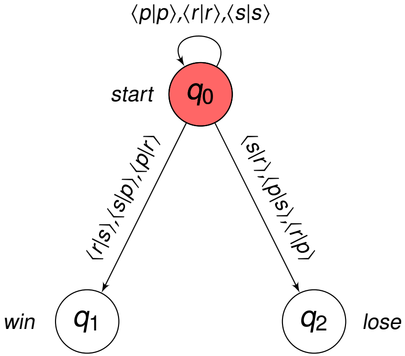


Player 2

Player 1

q_0	p	r	s
p	q_0	q_1	q_2
r	q_2	q_0	q_1
s	q_1	q_2	q_0

Example

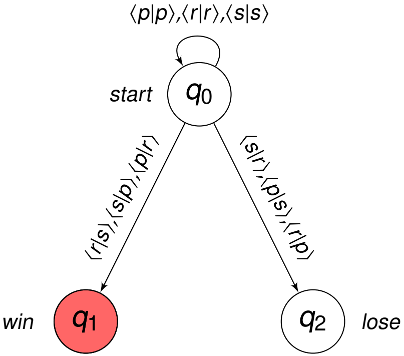


Player 2

Player 1

q_0	p	r	s
p	q_0	q_1	q_2
r	q_2	q_0	q_1
s	q_1	q_2	q_0

Example



Player 2

Player 1

q_0	p	r	s
p	q_0	q_1	q_2
r	q_2	q_0	q_1
s	q_1	q_2	q_0

Semantics of CGSs

- From a location ℓ, each agent A_i chooses some m_{A_i} with
$$m_{A_i} \in \mathrm{Mv}(\ell, A_i).$$
- $\mathrm{Edg}(\ell, m_{A_1}, \cdots, m_{A_k})$ gives the new location.

Notations:

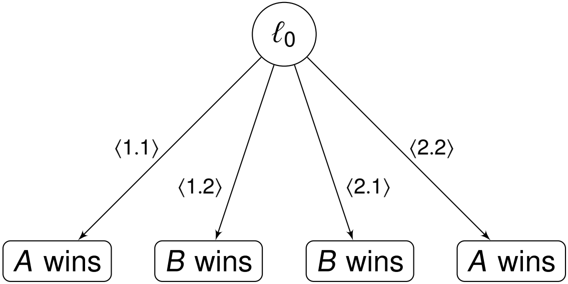
- $\mathrm{Next}(\ell) = \left\{\mathrm{Edg}(\ell, \cdots m_{A_i} \cdots) \mid \forall m_{A_i} \cdot 1 \leq i \leq k\right\}$
- $\mathrm{Next}(\ell, A_j, m) = \left\{\mathrm{Edg}(\ell, \cdots, m_{A_{j-1}}, m, m_{A_{j+1}}, \cdots)\right\}$

Strategies and outcomes

Definition

- A computation is an infinite sequence $\rho = \ell_0 \ell_1 \cdots$ such that $\forall i, \ell_{i+1} \in \text{Next}(\ell_i)$.
- A strategy for agent A_i is a function f_{A_i} s.t.
  $$f_{A_i}(\ell_0, \cdots, \ell_m) \in \text{Mv}(\ell_m, A_i).$$
- The outcomes $\text{Out}(\ell, f_{A_i})$ are the set of computations from ℓ that agree with the strategy f_{A_i} of A_i.
- Those notions extend to coallitions of agents: given $A \subseteq \mathbb{A}$, we write
  - $F_A = \{f_{A_i} | A_i \in A\}$
  - $\text{Out}(\ell, F_A)$

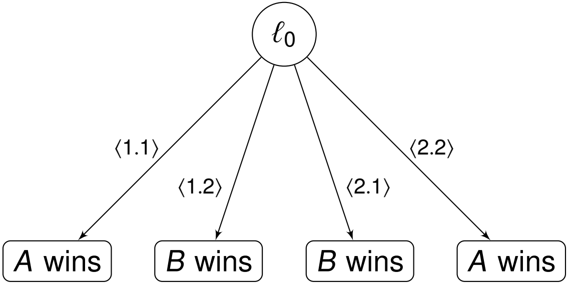
Another example



- player A has no strategy to win.
- player B has no strategy to win.

Synchronous games are not determined.

Another example



- player A has no strategy to win.
- player B has no strategy to win.

Synchronous games are not determined.

Theorem (Martin, 1975)

Turn-based games (with reasonnable winning conditions) are determined.

Syntax of ATL

Definition

The syntax of ATL is defined by the following grammar:

$$\begin{aligned} \text{ATL} \ni \varphi_s, \psi_s &::= p \mid \neg \varphi_s \mid \varphi_s \vee \psi_s \mid \langle\!\langle A \rangle\!\rangle \varphi_p \mid [\![A]\!] \varphi_p \\ \varphi_p &::= \mathbf{X}\, \varphi_s \mid \varphi_s \,\mathbf{U}\, \psi_s. \end{aligned}$$

where p ranges over the set AP and A over the subsets of $\mathbb{A}$.

ATL subsumes CTL, since we have:

$$\mathbf{E}\varphi_p \equiv \langle\!\langle \mathbb{A} \rangle\!\rangle \varphi_p,$$

$$\mathbf{A}\varphi_p \equiv \langle\!\langle \varnothing \rangle\!\rangle \varphi_p.$$

Semantics of ATL

Definition

- Semantics

$$\ell \models \langle\!\langle A \rangle\!\rangle \varphi_p \quad \text{iff} \quad \exists F_A \in \text{Strat}(A).\ \forall \rho \in \text{Out}(\ell, F_A).\ \rho \models \varphi_p$$

$$\rho \models \varphi_s \;\mathbf{U}\; \psi_s \quad \text{iff} \quad \exists i. \rho[i] \models \psi_s \text{ and } \forall 0 \leq j < i. \rho[j] \models \varphi_s$$

$$\rho \models \mathbf{X}\, \varphi_s \quad \text{iff} \quad \rho[1] \models \varphi_s$$

- We have $\langle\!\langle A \rangle\!\rangle \varphi \Rightarrow \neg \langle\!\langle \mathbb{A} \setminus A \rangle\!\rangle \neg \varphi$,
  but

$$\neg \langle\!\langle A \rangle\!\rangle \varphi \not\Rightarrow \langle\!\langle \mathbb{A} \setminus A \rangle\!\rangle \neg\varphi.$$

- The semantics of $[\![A]\!]\varphi$ is that of $\neg \langle\!\langle A \rangle\!\rangle \neg\varphi$

Model checking ATL

Theorem

Model-checking ATL is PTIME*-complete.*

Model checking ATL

Theorem

Model-checking ATL is PTIME*-complete.*

Proof.

- hardness in PTIME: ATL subsumes CTL.
- membership in PTIME: extend CTL labeling algorithm to handle “multi-agent” transitions.

□

Model checking ATL

Theorem

Model-checking ATL is PTIME*-complete.*

Proof.

- hardness in PTIME: ATL subsumes CTL.
- membership in PTIME: extend CTL labeling algorithm to handle “multi-agent” transitions.

□

- we cannot restrict to modalities $\langle\langle A\rangle\rangle\mathbf{X}$, $\langle\langle A\rangle\rangle\mathbf{G}$ and $\langle\langle A\rangle\rangle\mathbf{U}$: modality $[[A]]\,\mathbf{U}$ cannot be expressed from those three modalities;

Model checking ATL

Theorem

Model-checking ATL is PTIME*-complete.*

Proof.

- hardness in PTIME: ATL subsumes CTL.
- membership in PTIME: extend CTL labeling algorithm to handle “multi-agent” transitions.

□

- we cannot restrict to modalities $\langle\langle A\rangle\rangle\mathbf{X}$, $\langle\langle A\rangle\rangle\mathbf{G}$ and $\langle\langle A\rangle\rangle\mathbf{U}$: modality $[[A]]\,\mathbf{U}$ cannot be expressed from those three modalities;
- this algorithm runs in time $O(|\varphi| \cdot |{\rightarrow}|)$.

Outline of the course

1. Branching-time temporal logics
   - Complexity
   - Alternating-time Temporal Logic

2. Timed temporal logics
   - Timed models
   - Timed logics
   - Undecidability

Timed temporal logics

Temporal logics = qualitative requirements

Timed temporal logics adds quantitative requirements.

Timed temporal logics

Temporal logics = qualitative requirements

Timed temporal logics adds quantitative requirements.

Example

Any request is granted in at most 1 minute.

An alarm rings if the doors are open for more than 30 seconds.

Timed temporal logics

Temporal logics = qualitative requirements

Timed temporal logics adds quantitative requirements.

Example

Any request is granted in at most 1 minute.

An alarm rings if the doors are open for more than 30 seconds.

Requires explicit timing constraints in the model.

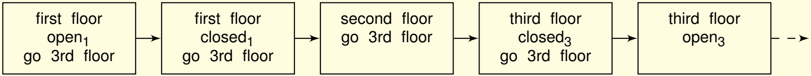
Adding “time” in Kripke structures

- basic idea: counting the number of transitions:

Adding “time” in Kripke structures

- basic idea: counting the number of transitions:

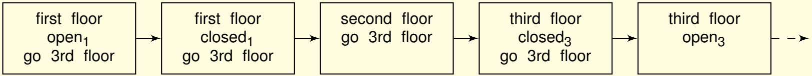
Examples



Adding "time" in Kripke structures

- basic idea: counting the number of transitions:

Examples

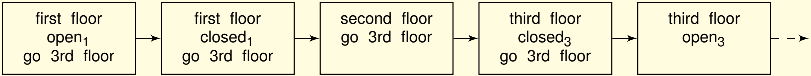


$$\mathbf{G}(\text{go 3rd floor} \Rightarrow \mathbf{F}_{\leq 4}\, \text{cabin.open}_3)$$

Adding “time” in Kripke structures

- basic idea: counting the number of transitions:

Examples



$$\mathbf{A}\,\mathbf{G}(\mathbf{E}\,\mathbf{F}_{\leq 10}\ \text{cabin.open}_1)$$

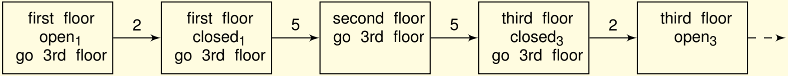
Adding “time” in Kripke structures

- basic idea: counting the number of transitions:
- slightly more involved: adding timing informations in Kripke structures:

Adding "time" in Kripke structures

- basic idea: counting the number of transitions:
- slightly more involved: adding timing informations in Kripke structures:

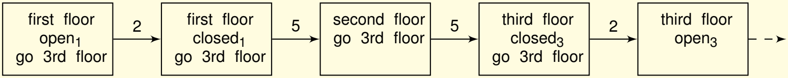
Examples



Adding "time" in Kripke structures

- basic idea: counting the number of transitions:
- slightly more involved: adding timing informations in Kripke structures:

Examples

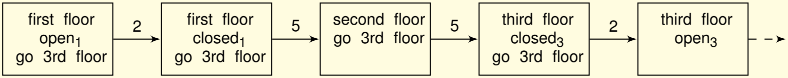


$$\mathbf{G}(\text{go 3rd floor} \Rightarrow \mathbf{F}_{\leq 14}\, \text{open}_3)$$

Adding "time" in Kripke structures

- basic idea: counting the number of transitions:
- slightly more involved: adding timing informations in Kripke structures:

Examples



$$\mathbf{A}\,\mathbf{G}(\mathbf{E}\,\mathbf{F}_{\leq 25}\,\text{open}_1)$$

Adding “time” in Kripke structures

- basic idea: counting the number of transitions:
- slightly more involved: adding timing informations in Kripke structures:

$\rightsquigarrow$ those models are not very expressive (only more succinct);

Adding "time" in Kripke structures

- basic idea: counting the number of transitions:
- slightly more involved: adding timing informations in Kripke structures:

$\leadsto$ those models are not very expressive (only more succinct);

$\leadsto$ in this settings, the logics also are not more expressive:

$$\mathbf{A\,G}(\mathbf{E\,F}_{\leq 25}\,\text{open}_1) \equiv \mathbf{A\,G}(\mathbf{E\,X}(\text{open}_1 \vee \mathbf{E\,X}(\text{open}_1 \vee \mathbf{E\,X}(\text{open}_1 \vee \mathbf{E\,X}(\text{open}_1 \ldots)))))$$

Adding "time" in Kripke structures

- basic idea: counting the number of transitions:
- slightly more involved: adding timing informations in Kripke structures:

$\rightsquigarrow$ those models are not very expressive (only more succinct);

$\rightsquigarrow$ in this settings, the logics also are not more expressive:

$$\mathbf{A\,G}(\mathbf{E\,F}_{\leq 25}\,\text{open}_1) \equiv \mathbf{A\,G}(\mathbf{E\,X}(\text{open}_1 \vee \mathbf{E\,X}(\text{open}_1 \vee \mathbf{E\,X}(\text{open}_1 \vee \mathbf{E\,X}(\text{open}_1 \ldots)))))$$

Theorem

Model-checking TCTL on timed Kripke structures is PSPACE*-complete.*
Model-checking TLTL on timed Kripke structures is EXPSPACE*-complete.*

Timed automata

Definition

A *timed automaton* is a tuple $\mathcal{A} = \langle Q, Q_0, C, \rightarrow, \Sigma, \ell \rangle$ s.t.:

- Q is the set of locations, of which Q_0 are initial;
- C is a (finite) set of *clock variables*;
- $\rightarrow$ is the set of transitions
- Σ is the alphabet;
- ℓ labels either the states or the transitions.

Timed automata

Definition

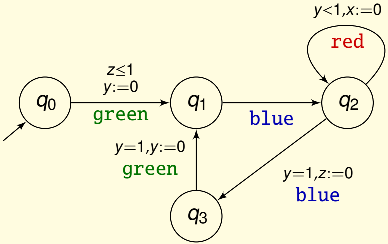
A *timed automaton* is a tuple $\mathcal{A} = \langle Q, Q_0, \mathcal{C}, \rightarrow, \Sigma, \ell \rangle$ s.t.:

- Q is the set of locations, of which Q_0 are initial;
- $\mathcal{C}$ is a (finite) set of *clock variables*;
- $\rightarrow$ is the set of transitions
- Σ is the alphabet;
- ℓ labels either the states or the transitions.

Clocks are used on transitions: a transition is labeled with a *guard*, i.e., a list of constraints $x \sim n$ where $x \in \mathcal{C}$, $n \in \mathbb{Z}^+$ and $\sim \in \{<, \leq, =, \geq, >\}$.

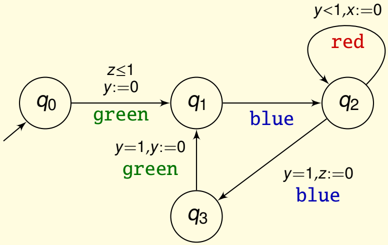
Timed automata

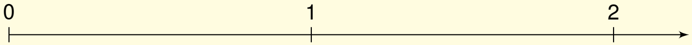
Example



Timed automata

Example





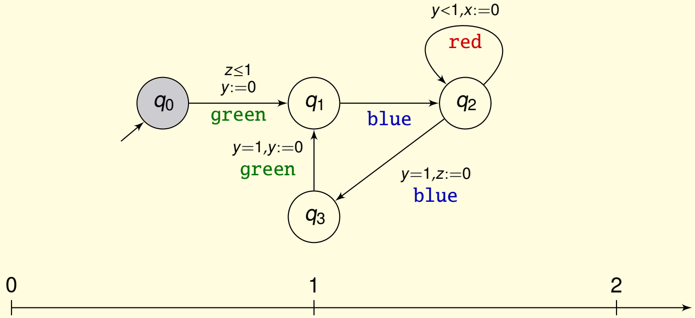
$x=$

$y=$

$z=$

Timed automata

Example



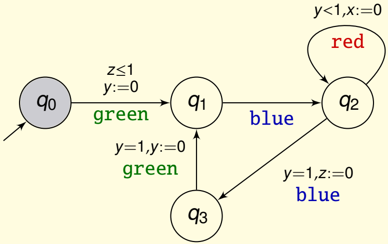
$x=$ 0

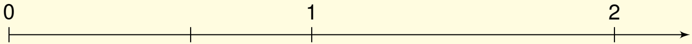
$y=$ 0

$z=$ 0

Timed automata

Example

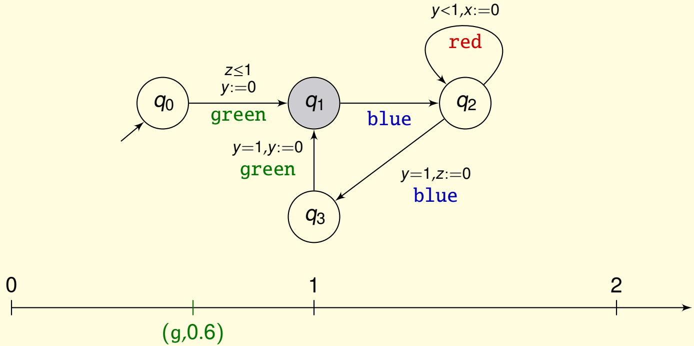




$x=$	0	0.6
$y=$	0	0.6
$z=$	0	0.6

Timed automata

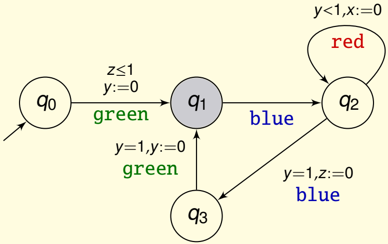
Example

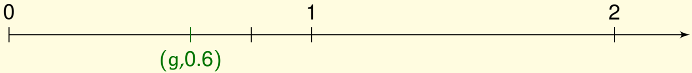


$x=$	0	0.6
$y=$	0	0
$z=$	0	0.6

Timed automata

Example

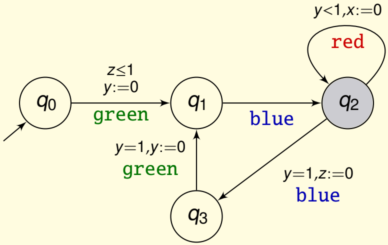


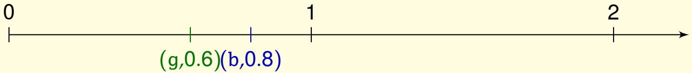


$x=$	0	0.6	0.8
$y=$	0	0	0.2
$z=$	0	0.6	0.8

Timed automata

Example

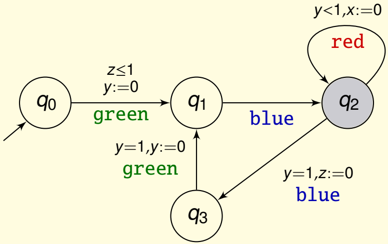


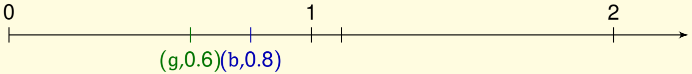


$x=$	0	0.6	0.8
$y=$	0	0	0.2
$z=$	0	0.6	0.8

Timed automata

Example

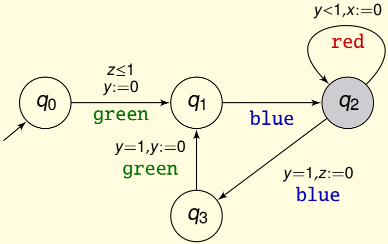


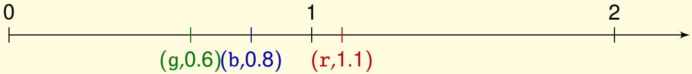


$x=$	0	0.6	0.8	1.1
$y=$	0	0	0.2	0.5
$z=$	0	0.6	0.8	1.1

Timed automata

Example

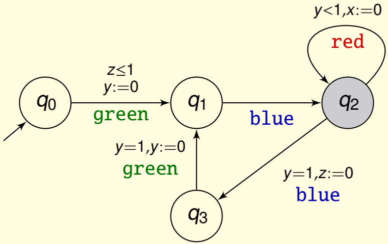


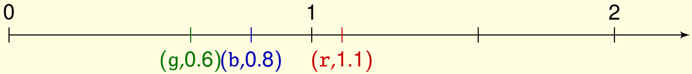


$x=$	0	0.6	0.8	0
$y=$	0	0	0.2	0.5
$z=$	0	0.6	0.8	1.1

Timed automata

Example

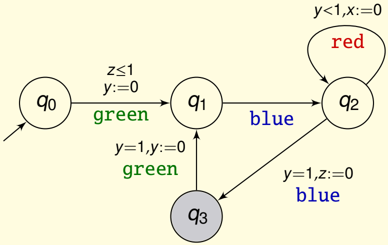


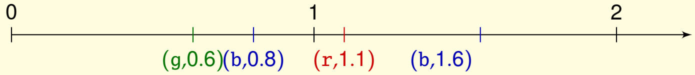


$x=$	0	0.6	0.8	0	0.5
$y=$	0	0	0.2	0.5	1
$z=$	0	0.6	0.8	1.1	1.6

Timed automata

Example

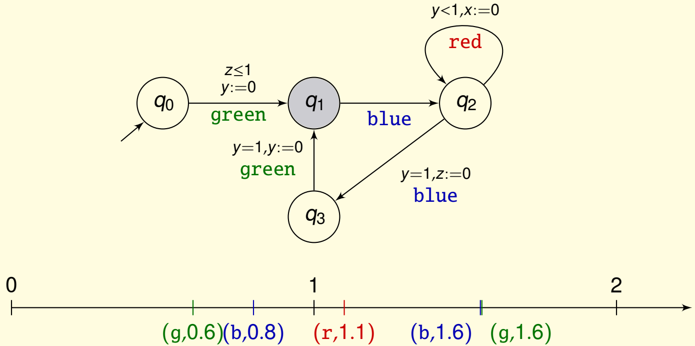




$x=$	0	0.6	0.8	0	0.5
$y=$	0	0	0.2	0.5	1
$z=$	0	0.6	0.8	1.1	0

Timed automata

Example



$x=$	0	0.6	0.8	0	0.5	0.5
$y=$	0	0	0.2	0.5	1	0
$z=$	0	0.6	0.8	1.1	0	0

Timed automata

Definition

A *timed word* is a function $w\colon \mathbb{Z}^+ \to 2^{\mathsf{AP}} \times \mathbb{R}^+$ s.t. w_2 is nondecreasing and diverges.

Timed automata

Definition

A *timed word* is a function $w\colon \mathbb{Z}^+ \to 2^{\mathsf{AP}} \times \mathbb{R}^+$ s.t. w_2 is nondecreasing and diverges.

Example

$$w = (\texttt{green}, 0.6)\,(\texttt{blue}, 0.8)\,(\texttt{red}, 1.1)\,(\texttt{blue}, 1.6)\,(\texttt{green}, 1.6)\ldots$$

Timed automata

Definition

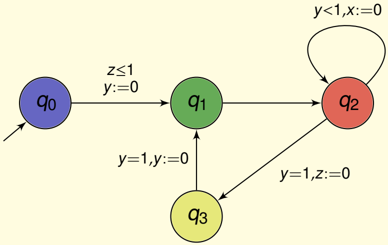
A *timed word* is a function $w\colon \mathbb{Z}^+ \to 2^{\mathsf{AP}} \times \mathbb{R}^+$ s.t. w_2 is nondecreasing and diverges.

Example

$$w = (\epsilon, 0)\,(\mathtt{green}, 0.6)\,(\mathtt{blue}, 0.8)\,(\mathtt{red}, 1.1)\,(\mathtt{blue}, 1.6)\,(\mathtt{green}, 1.6)\ldots$$

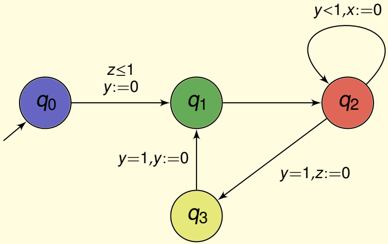
Timed automata

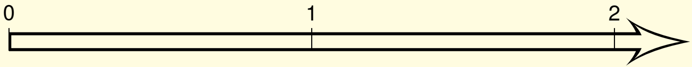
Example



Timed automata

Example





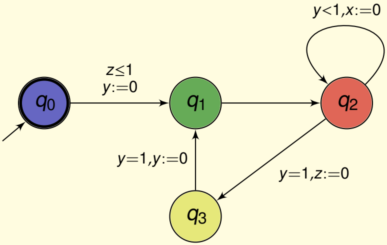
$x=$

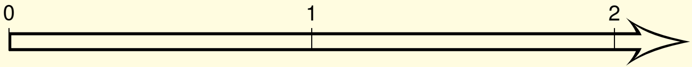
$y=$

$z=$

Timed automata

Example





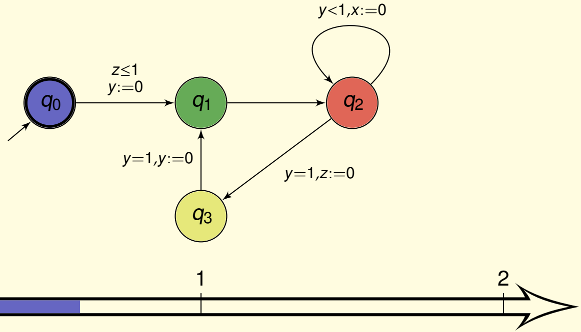
$x =$ 0

$y =$ 0

$z =$ 0

Timed automata

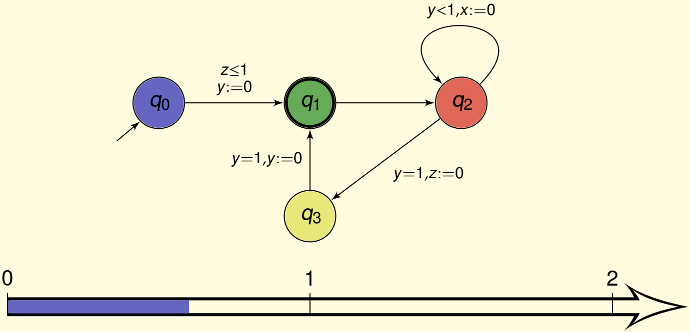
Example



$x=$	0	0.6
$y=$	0	0.6
$z=$	0	0.6

Timed automata

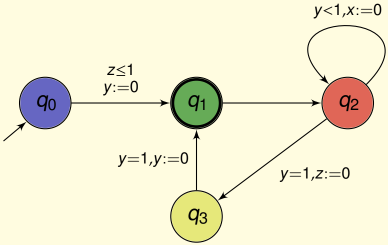
Example

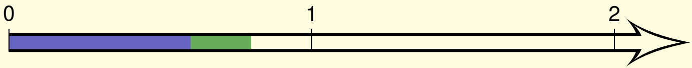


$x=$	0	0.6
$y=$	0	0
$z=$	0	0.6

Timed automata

Example

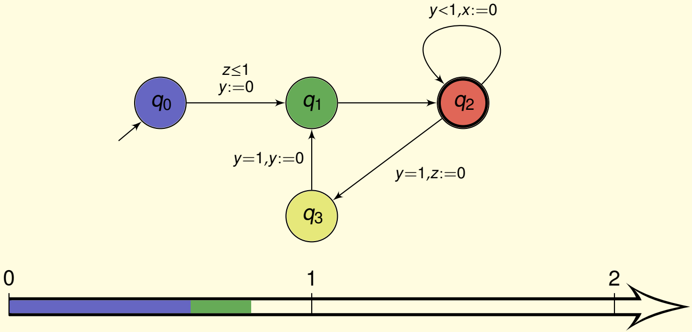




$x=$	0	0.6	0.8
$y=$	0	0	0.2
$z=$	0	0.6	0.8

Timed automata

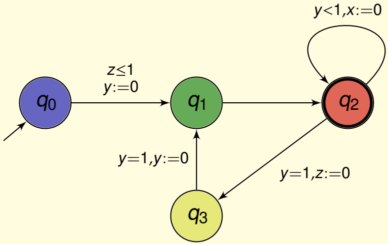
Example

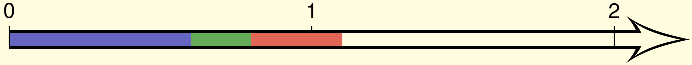


$x=$	0	0.6	0.8
$y=$	0	0	0.2
$z=$	0	0.6	0.8

Timed automata

Example

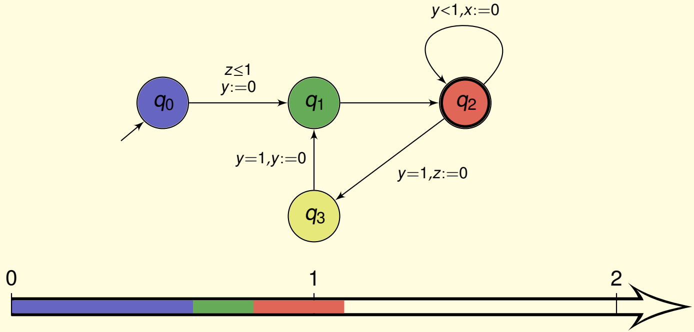




$x=$	0	0.6	0.8	1.1
$y=$	0	0	0.2	0.5
$z=$	0	0.6	0.8	1.1

Timed automata

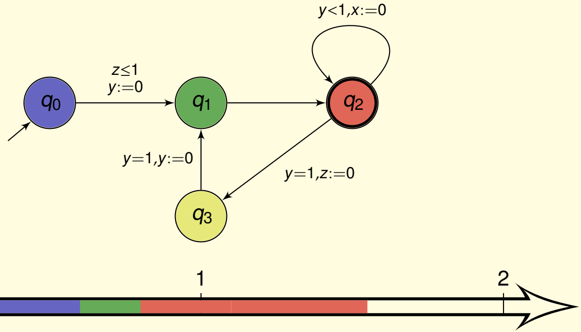
Example



$x=$	0	0.6	0.8	0
$y=$	0	0	0.2	0.5
$z=$	0	0.6	0.8	1.1

Timed automata

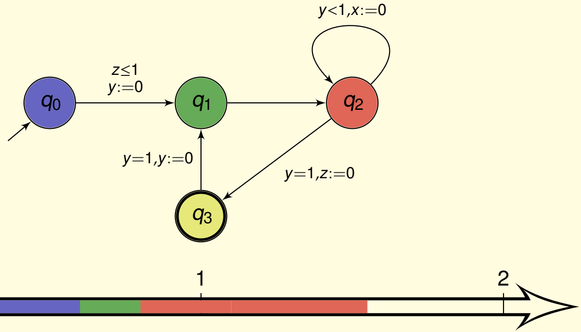
Example



$x=$	0	0.6	0.8	0	0.5
$y=$	0	0	0.2	0.5	1
$z=$	0	0.6	0.8	1.1	1.6

Timed automata

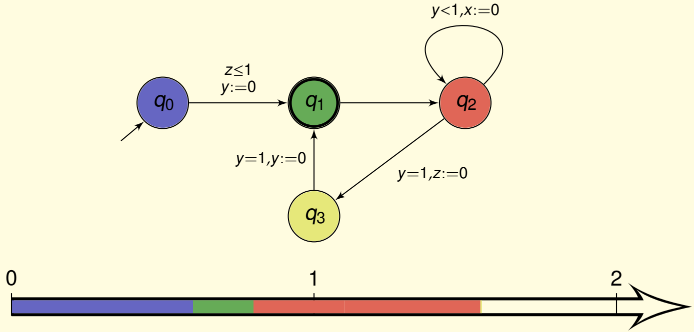
Example



$x=$	0	0.6	0.8	0	0.5
$y=$	0	0	0.2	0.5	1
$z=$	0	0.6	0.8	1.1	0

Timed automata

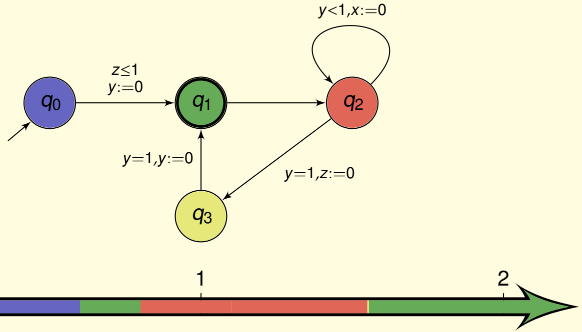
Example



$x=$	0	0.6	0.8	0	0.5	0.5
$y=$	0	0	0.2	0.5	1	0
$z=$	0	0.6	0.8	1.1	0	0

Timed automata

Example



$x=$	0	0.6	0.8	0	0.5	0.5
$y=$	0	0	0.2	0.5	1	0
$z=$	0	0.6	0.8	1.1	0	0

Timed automata

Definition

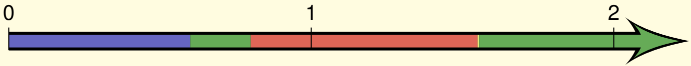
A *timed state sequence* is a function $\pi \colon \mathbb{R}^+ \to 2^{\mathsf{AP}}$.

Timed automata

Definition

A *timed state sequence* is a function $\pi \colon \mathbb{R}^+ \to 2^{\mathsf{AP}}$.

Example



Outline of the course

1. Branching-time temporal logics
   - Complexity
   - Alternating-time Temporal Logic

2. Timed temporal logics
   - Timed models
   - Timed logics
   - Undecidability

Extending temporal logics with time

Two different ways of extending temporal logics:

- by associating intervals with modalities: those intervals (having rational bounds) indicate e.g. the moment at which an eventuality is to be fulfilled.

Extending temporal logics with time

Two different ways of extending temporal logics:

- by associating intervals with modalities: those intervals (having rational bounds) indicate e.g. the moment at which an eventuality is to be fulfilled.

Examples

$$\mathbf{G}(\texttt{call}_3 \Rightarrow \mathbf{F}_{[0,1]}\, \texttt{open}_3)$$

Extending temporal logics with time

Two different ways of extending temporal logics:

- by associating intervals with modalities: those intervals (having rational bounds) indicate e.g. the moment at which an eventuality is to be fulfilled.

Examples

$$\mathbf{G}(\mathtt{call}_3 \Rightarrow \mathbf{F}_{[0,1]}\, \mathtt{open}_3)$$

$$\mathbf{A}\,\mathbf{G}(\mathbf{E}\,\mathbf{F}_{[0,3]}\, \mathtt{open}_1)$$

Extending temporal logics with time

Two different ways of extending temporal logics:

- by associating intervals with modalities: those intervals (having rational bounds) indicate e.g. the moment at which an eventuality is to be fulfilled.
- by using real clocks in the formula: the clocks can be reset at some point during the evaluation of the formula, and then compared to rationals.

Extending temporal logics with time

Two different ways of extending temporal logics:

- by associating intervals with modalities: those intervals (having rational bounds) indicate e.g. the moment at which an eventuality is to be fulfilled.
- by using real clocks in the formula: the clocks can be reset at some point during the evaluation of the formula, and then compared to rationals.

Examples

$$\mathbf{G}(\mathtt{call}_3 \Rightarrow x.\mathbf{F}(\mathtt{open}_3 \wedge x \leq 1))$$

Extending temporal logics with time

Two different ways of extending temporal logics:

- by associating intervals with modalities: those intervals (having rational bounds) indicate e.g. the moment at which an eventuality is to be fulfilled.
- by using real clocks in the formula: the clocks can be reset at some point during the evaluation of the formula, and then compared to rationals.

Examples

$$\mathbf{G}(\mathtt{call}_3 \Rightarrow x.\mathbf{F}(\mathtt{open}_3 \wedge x \leq 1))$$

$$\mathbf{A}\,\mathbf{G}(x.\mathbf{E}\,\mathbf{F}(\mathtt{open}_1 \wedge x \leq 3))$$

Timed logics in the pointwise framework

- Syntax of MTL:

$$\mathsf{MTL} \ni \varphi ::= p \mid \neg\,\varphi \mid \varphi \vee \varphi \mid \varphi\ \mathbf{U}_I\ \varphi$$

where p ranges over AP and I is an interval with bounds in $\mathbb{Q}^+ \cup \{+\infty\}$.

Timed logics in the pointwise framework

- Syntax of MTL:

$$\mathsf{MTL} \ni \varphi ::= p \mid \neg\varphi \mid \varphi \vee \varphi \mid \varphi \,\mathbf{U}_I\, \varphi$$

  where p ranges over AP and I is an interval with bounds in $\mathbb{Q}^+ \cup \{+\infty\}$.

- Pointwise semantics of MTL: over $\pi = ((w_i)_i, (t_i)_i)$:
  - $\pi, i \models \varphi \,\mathbf{U}_I\, \psi$ iff there exists some $j > 0$ s.t.
    - $\pi, i + j \models \psi$,
    - $\pi, i + k \models \varphi$ for all $0 < k < j$,
    - $t_{i+j} - t_i \in I$.

Timed logics in the pointwise framework

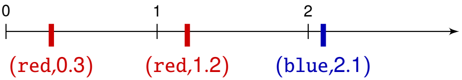
- Syntax of MTL:

$$\mathsf{MTL} \ni \varphi ::= p \mid \neg\varphi \mid \varphi \vee \varphi \mid \varphi \,\mathbf{U}_I\, \varphi$$

  where p ranges over AP and I is an interval with bounds in $\mathbb{Q}^+ \cup \{+\infty\}$.

- Pointwise semantics of MTL: over $\pi = ((w_i)_i, (t_i)_i)$:
  - $\pi, i \models \varphi \,\mathbf{U}_I\, \psi$ iff there exists some $j > 0$ s.t.
    - $\pi, i + j \models \psi$,
    - $\pi, i + k \models \varphi$ for all $0 < k < j$,
    - $t_{i+j} - t_i \in I$.

- Examples:



$\texttt{red}\ \mathbf{U}_{[2,3]}\ \texttt{blue}$

Timed logics in the pointwise framework

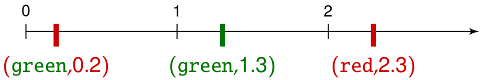
- Syntax of MTL:

$$\mathsf{MTL} \ni \varphi ::= p \mid \neg\varphi \mid \varphi \vee \varphi \mid \varphi\ \mathbf{U}_I\ \varphi$$

  where p ranges over AP and I is an interval with bounds in $\mathbb{Q}^+ \cup \{+\infty\}$.

- Pointwise semantics of MTL: over $\pi = ((w_i)_i, (t_i)_i)$:
  - $\pi, i \models \varphi\ \mathbf{U}_I\ \psi$ iff there exists some $j > 0$ s.t.
    - $\pi, i + j \models \psi$,
    - $\pi, i + k \models \varphi$ for all $0 < k < j$,
    - $t_{i+j} - t_i \in I$.

- Examples:



$\mathbf{F}(\texttt{green} \wedge \bot\ \mathbf{U}_{[1,1]}\ \texttt{red})$

Timed logics in the pointwise framework

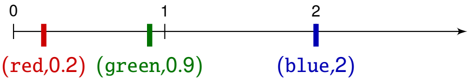
- Syntax of MTL:

$$\mathsf{MTL} \ni \varphi ::= p \mid \neg\varphi \mid \varphi \vee \varphi \mid \varphi\ \mathbf{U}_I\ \varphi$$

  where p ranges over AP and I is an interval with bounds in $\mathbb{Q}^+ \cup \{+\infty\}$.

- Pointwise semantics of MTL: over $\pi = ((w_i)_i, (t_i)_i)$:
  - $\pi, i \models \varphi\ \mathbf{U}_I\ \psi$ iff there exists some $j > 0$ s.t.
    - $\pi, i + j \models \psi$,
    - $\pi, i + k \models \varphi$ for all $0 < k < j$,
    - $t_{i+j} - t_i \in I$.

- Examples:



$\mathbf{F}_{[2,2]}\ \mathtt{blue}$

Timed logics in the pointwise framework

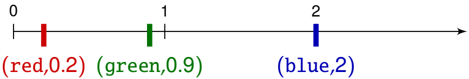
- Syntax of MTL:

$$\mathsf{MTL} \ni \varphi ::= p \mid \neg\varphi \mid \varphi \vee \varphi \mid \varphi \,\mathbf{U}_I\, \varphi$$

  where p ranges over AP and I is an interval with bounds in $\mathbb{Q}^+ \cup \{+\infty\}$.

- Pointwise semantics of MTL: over $\pi = ((w_i)_i, (t_i)_i)$:
  - $\pi, i \models \varphi \,\mathbf{U}_I\, \psi$ iff there exists some $j > 0$ s.t.
    - $\pi, i + j \models \psi$,
    - $\pi, i + k \models \varphi$ for all $0 < k < j$,
    - $t_{i+j} - t_i \in I$.

- Examples:



$\mathbf{F}_{[2,2]}\,\texttt{blue} \stackrel{\text{def}}{=} \mathbf{F}_{=2}\,\texttt{blue}$

Timed logics in the pointwise framework

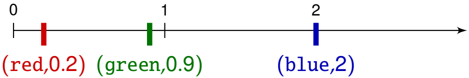
- Syntax of MTL:

$$\mathsf{MTL} \ni \varphi ::= p \mid \neg\varphi \mid \varphi \vee \varphi \mid \varphi\ \mathbf{U}_I\ \varphi$$

  where p ranges over AP and I is an interval with bounds in $\mathbb{Q}^+ \cup \{+\infty\}$.

- Pointwise semantics of MTL: over $\pi = ((w_i)_i, (t_i)_i)$:
  - $\pi, i \models \varphi\ \mathbf{U}_I\ \psi$ iff there exists some $j > 0$ s.t.
    - $\pi, i + j \models \psi$,
    - $\pi, i + k \models \varphi$ for all $0 < k < j$,
    - $t_{i+j} - t_i \in I$.

- Examples:



$$\mathbf{F}_{[2,2]}\,\texttt{blue} \not\equiv \mathbf{F}_{=1}\,\mathbf{F}_{=1}\,\texttt{blue}$$

Timed logics in the pointwise framework

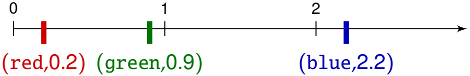
- Syntax of MTL:

$$\mathsf{MTL} \ni \varphi ::= p \mid \neg\varphi \mid \varphi \vee \varphi \mid \varphi\ \mathbf{U}_I\ \varphi$$

  where p ranges over AP and I is an interval with bounds in $\mathbb{Q}^+ \cup \{+\infty\}$.

- Pointwise semantics of MTL: over $\pi = ((w_i)_i, (t_i)_i)$:
  - $\pi, i \models \varphi\ \mathbf{U}_I\ \psi$ iff there exists some $j > 0$ s.t.
    - $\pi, i+j \models \psi$,
    - $\pi, i+k \models \varphi$ for all $0 < k < j$,
    - $t_{i+j} - t_i \in I$.

- Examples:



$\mathbf{F}(\mathtt{blue} \wedge \mathbf{G}^{-1}_{[-1,0]} \bot)$

Timed logics in the pointwise framework

- Syntax of TPTL:

$$\mathsf{TPTL} \ni \varphi ::= p \mid x \sim c \mid \neg\varphi \mid \varphi \vee \varphi \mid \varphi \,\mathbf{U}\, \varphi \mid x.\,\varphi$$

where p ranges over AP, x ranges over a set of formula clocks, $c \in \mathbb{Q}^+$ and $\sim \in \{<, \leq, =, \geq, >\}$.

Timed logics in the pointwise framework

- Syntax of TPTL:

$$\text{TPTL} \ni \varphi ::= p \mid x \sim c \mid \neg\varphi \mid \varphi \vee \varphi \mid \varphi\ \mathbf{U}\ \varphi \mid x.\ \varphi$$

  where p ranges over AP, x ranges over a set of formula clocks, $c \in \mathbb{Q}^+$ and $\sim\ \in \{<, \leq, =, \geq, >\}$.

- Pointwise semantics of TPTL: over $\pi = ((w_i)_i, (t_i)_i)$:
  - $\pi, i, \tau \models x \sim c$ iff $\tau(x) \sim c$

Timed logics in the pointwise framework

- Syntax of TPTL:

$$\mathsf{TPTL} \ni \varphi ::= p \mid x \sim c \mid \neg\varphi \mid \varphi \vee \varphi \mid \varphi \, \mathbf{U} \, \varphi \mid x.\,\varphi$$

  where p ranges over AP, x ranges over a set of formula clocks, $c \in \mathbb{Q}^+$ and $\sim \in \{<, \leq, =, \geq, >\}$.

- Pointwise semantics of TPTL: over $\pi = ((w_i)_i, (t_i)_i)$:
  - $\pi, i, \tau \models x \sim c$ iff $\tau(x) \sim c$
  - $\pi, i, \tau \models x.\,\varphi$ iff $\pi, i, \tau[x \leftarrow 0] \models \varphi$

Timed logics in the pointwise framework

- Syntax of TPTL:

$$\text{TPTL} \ni \varphi ::= p \mid x \sim c \mid \neg\varphi \mid \varphi \vee \varphi \mid \varphi \;\mathbf{U}\; \varphi \mid x.\,\varphi$$

  where p ranges over AP, x ranges over a set of formula clocks, $c \in \mathbb{Q}^+$ and $\sim \in \{<, \leq, =, \geq, >\}$.

- Pointwise semantics of TPTL: over $\pi = ((w_i)_i, (t_i)_i)$:
  - $\pi, i, \tau \models x \sim c$ iff $\tau(x) \sim c$
  - $\pi, i, \tau \models x.\,\varphi$ iff $\pi, i, \tau[x \leftarrow 0] \models \varphi$
  - $\pi, i, \tau \models \varphi \;\mathbf{U}\; \psi$ iff there exists some $j > 0$ s.t.
    - $\pi, i+j, \tau + t_{i+j} - t_i \models \psi$,
    - $\pi, i+k, \tau + t_{i+k} - t_i \models \varphi$ for all $0 < k < j$.

Timed logics in the pointwise framework

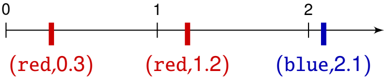
- Syntax of TPTL:

$$\text{TPTL} \ni \varphi ::= p \mid x \sim c \mid \neg \varphi \mid \varphi \vee \varphi \mid \varphi \,\mathbf{U}\, \varphi \mid x.\, \varphi$$

  where p ranges over AP, x ranges over a set of formula clocks, $c \in \mathbb{Q}^+$ and $\sim \in \{<, \leq, =, \geq, >\}$.

- Pointwise semantics of TPTL: over $\pi = ((w_i)_i, (t_i)_i)$:
  - $\pi, i, \tau \models x \sim c$ iff $\tau(x) \sim c$
  - $\pi, i, \tau \models x.\, \varphi$ iff $\pi, i, \tau[x \leftarrow 0] \models \varphi$
  - $\pi, i, \tau \models \varphi \,\mathbf{U}\, \psi$ iff there exists some $j > 0$ s.t.
    - $\pi, i + j, \tau + t_{i+j} - t_i \models \psi$,
    - $\pi, i + k, \tau + t_{i+k} - t_i \models \varphi$ for all $0 < k < j$.

- Examples:



$x.(\texttt{red}\ \mathbf{U}\ (\texttt{blue} \wedge x \in [2, 3]))$

Timed logics in the pointwise framework

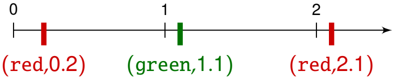
- Syntax of TPTL:

$$\text{TPTL} \ni \varphi ::= p \mid x \sim c \mid \neg\varphi \mid \varphi \vee \varphi \mid \varphi \mathbf{U} \varphi \mid x.\, \varphi$$

  where p ranges over AP, x ranges over a set of formula clocks, $c \in \mathbb{Q}^+$ and $\sim \in \{<, \leq, =, \geq, >\}$.

- Pointwise semantics of TPTL: over $\pi = ((w_i)_i, (t_i)_i)$:
  - $\pi, i, \tau \models x \sim c$ iff $\tau(x) \sim c$
  - $\pi, i, \tau \models x.\, \varphi$ iff $\pi, i, \tau[x \leftarrow 0] \models \varphi$
  - $\pi, i, \tau \models \varphi \mathbf{U} \psi$ iff there exists some $j > 0$ s.t.
    - $\pi, i+j, \tau + t_{i+j} - t_i \models \psi$,
    - $\pi, i+k, \tau + t_{i+k} - t_i \models \varphi$ for all $0 < k < j$.

- Examples:



$$\mathbf{F}(\texttt{green} \wedge x.(\bot \mathbf{U} (\texttt{red} \wedge x = 1)))$$

Timed logics in the pointwise framework

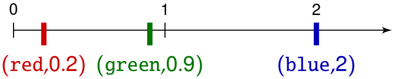
- Syntax of TPTL:

$$\text{TPTL} \ni \varphi ::= p \mid x \sim c \mid \neg\varphi \mid \varphi \vee \varphi \mid \varphi \,\mathbf{U}\, \varphi \mid x.\,\varphi$$

  where p ranges over AP, x ranges over a set of formula clocks, $c \in \mathbb{Q}^+$ and $\sim \in \{<, \leq, =, \geq, >\}$.

- Pointwise semantics of TPTL: over $\pi = ((w_i)_i, (t_i)_i)$:
  - $\pi, i, \tau \models x \sim c$ iff $\tau(x) \sim c$
  - $\pi, i, \tau \models x.\,\varphi$ iff $\pi, i, \tau[x \leftarrow 0] \models \varphi$
  - $\pi, i, \tau \models \varphi \,\mathbf{U}\, \psi$ iff there exists some $j > 0$ s.t.
    - $\pi, i + j, \tau + t_{i+j} - t_i \models \psi$,
    - $\pi, i + k, \tau + t_{i+k} - t_i \models \varphi$ for all $0 < k < j$.

- Examples:



$$x.\,\mathbf{F}(\texttt{red} \wedge \mathbf{F}(\texttt{green} \wedge x \leq 1))$$

Timed logics in the continuous framework

- Syntax of MTL:

$$\mathsf{MTL} \ni \varphi ::= p \mid \neg\varphi \mid \varphi \vee \varphi \mid \varphi \,\mathbf{U}_I\, \varphi$$

Timed logics in the continuous framework

- Syntax of MTL:

$$\mathsf{MTL} \ni \varphi ::= p \mid \neg \varphi \mid \varphi \vee \varphi \mid \varphi \ \mathbf{U}_I \ \varphi$$

- Continuous semantics of MTL: over $\pi : \mathbb{R}^+ \to 2^{\mathsf{AP}}$:
  - $\pi, t \models \varphi \ \mathbf{U}_I \ \psi$ iff there exists some $u > 0$ s.t.
    - $\pi, t + u \models \psi$,
    - $\pi, t + v \models \varphi$ for all $0 < v < u$,
    - $u \in I$.

Timed logics in the continuous framework

- Syntax of MTL:

$$\mathsf{MTL} \ni \varphi ::= p \mid \neg\varphi \mid \varphi \vee \varphi \mid \varphi \,\mathbf{U}_I\, \varphi$$

- Continuous semantics of MTL: over $\pi : \mathbb{R}^+ \to 2^{\mathsf{AP}}$:
  - $\pi, t \models \varphi \,\mathbf{U}_I\, \psi$ iff there exists some $u > 0$ s.t.
    - $\pi, t + u \models \psi$,
    - $\pi, t + v \models \varphi$ for all $0 < v < u$,
    - $u \in I$.
  - $\pi, t \models p$ iff $p \in \pi(t)$

Timed logics in the continuous framework

- Syntax of MTL:

$$\mathsf{MTL} \ni \varphi ::= p \mid \neg \varphi \mid \varphi \vee \varphi \mid \varphi \, \mathbf{U}_I \, \varphi$$

- Continuous semantics of MTL: over $\pi \colon \mathbb{R}^+ \to 2^{\mathsf{AP}}$:
  - $\pi, t \models \varphi \, \mathbf{U}_I \, \psi$ iff there exists some $u > 0$ s.t.
    - $\pi, t + u \models \psi$,
    - $\pi, t + v \models \varphi$ for all $0 < v < u$,
    - $u \in I$.
  - $\pi, t \models p$ iff $p \in \pi(t)$

- Examples:

$(\mathtt{red} \vee \mathtt{blue}) \, \mathbf{U}_{\leq 2} \, \mathtt{green}$

Timed logics in the continuous framework

- Syntax of MTL:

$$\mathsf{MTL} \ni \varphi ::= p \mid \neg\varphi \mid \varphi \vee \varphi \mid \varphi\ \mathbf{U}_I\ \varphi$$

- Continuous semantics of MTL: over $\pi\colon \mathbb{R}^+ \to 2^{AP}$:
  - $\pi, t \models \varphi\ \mathbf{U}_I\ \psi$ iff there exists some $u > 0$ s.t.
    - $\pi, t+u \models \psi$,
    - $\pi, t+v \models \varphi$ for all $0 < v < u$,
    - $u \in I$.
  - $\pi, t \models p$ iff $p \in \pi(t)$

- Examples:

$\mathbf{F}_{=2}\ \texttt{green}$

Timed logics in the continuous framework

- Syntax of MTL:

$$\text{MTL} \ni \varphi ::= p \mid \neg\varphi \mid \varphi \vee \varphi \mid \varphi\ \mathbf{U}_I\ \varphi$$

- Continuous semantics of MTL: over $\pi\colon \mathbb{R}^+ \to 2^{\mathsf{AP}}$:
  - $\pi, t \models \varphi\ \mathbf{U}_I\ \psi$ iff there exists some $u > 0$ s.t.
    - $\pi, t + u \models \psi$,
    - $\pi, t + v \models \varphi$ for all $0 < v < u$,
    - $u \in I$.
  - $\pi, t \models p$ iff $p \in \pi(t)$

- Examples:

$\mathbf{F}_{=2}\,\mathtt{green} \equiv \mathbf{F}_{=1}(\mathbf{F}_{=1}\,\mathtt{green})$

Timed logics in the continuous framework

- Syntax of TPTL:

$$\text{TPTL} \ni \varphi ::= p \mid x \sim c \mid \neg\varphi \mid \varphi \vee \varphi \mid \varphi \,\mathbf{U}\, \varphi \mid x.\,\varphi$$

Timed logics in the continuous framework

- Syntax of TPTL:

$$\mathsf{TPTL} \ni \varphi ::= p \mid x \sim c \mid \neg\varphi \mid \varphi \vee \varphi \mid \varphi\ \mathbf{U}\ \varphi \mid x.\ \varphi$$

- Continuous semantics of TPTL: over $\pi\colon \mathbb{R}^+ \to 2^{\mathsf{AP}}$:
  - $\pi, t, \tau \models x \sim c$ iff $\tau(x) \sim c$

Timed logics in the continuous framework

- Syntax of TPTL:

$$\mathsf{TPTL} \ni \varphi ::= p \mid x \sim c \mid \neg\varphi \mid \varphi \vee \varphi \mid \varphi \,\mathbf{U}\, \varphi \mid x.\ \varphi$$

- Continuous semantics of TPTL: over $\pi : \mathbb{R}^+ \to 2^{AP}$:
  - $\pi, t, \tau \models x \sim c$ iff $\tau(x) \sim c$
  - $\pi, t, \tau \models x.\ \varphi$ iff $\pi, i, \tau_{[x \leftarrow 0]} \models \varphi$

Timed logics in the continuous framework

- Syntax of TPTL:

$$\text{TPTL} \ni \varphi ::= p \mid x \sim c \mid \neg\varphi \mid \varphi \vee \varphi \mid \varphi \mathbf{U} \varphi \mid x.\, \varphi$$

- Continuous semantics of TPTL: over $\pi \colon \mathbb{R}^+ \to 2^{\mathsf{AP}}$:
  - $\pi, t, \tau \models x \sim c$ iff $\tau(x) \sim c$
  - $\pi, t, \tau \models x.\, \varphi$ iff $\pi, i, \tau[x \leftarrow 0] \models \varphi$
  - $\pi, t, \tau \models \varphi \mathbf{U} \psi$ iff there exists some $u > 0$ s.t.
    - $\pi, t + u, \tau + u - t \models \psi$,
    - $\pi, i + k, \tau + v - t \models \varphi$ for all $0 < v < u$.

Timed logics in the continuous framework

- Syntax of TPTL:

$$\text{TPTL} \ni \varphi ::= p \mid x \sim c \mid \neg\varphi \mid \varphi \vee \varphi \mid \varphi \mathbf{U} \varphi \mid x.\,\varphi$$

- Continuous semantics of TPTL: over $\pi\colon \mathbb{R}^+ \to 2^{\mathsf{AP}}$:
  - $\pi, t, \tau \models x \sim c$ iff $\tau(x) \sim c$
  - $\pi, t, \tau \models x.\,\varphi$ iff $\pi, i, \tau[x\leftarrow 0] \models \varphi$
  - $\pi, t, \tau \models \varphi \mathbf{U} \psi$ iff there exists some $u > 0$ s.t.
    - $\pi, t+u, \tau + u - t \models \psi$,
    - $\pi, i+k, \tau + v - t \models \varphi$ for all $0 < v < u$.

- Examples:

$x.((\texttt{red} \vee \texttt{blue}) \mathbf{U} (\texttt{green} \wedge x \leq 2))$

Timed logics in the continuous framework

- Syntax of TPTL:

$$\mathsf{TPTL} \ni \varphi ::= p \mid x \sim c \mid \neg\varphi \mid \varphi \vee \varphi \mid \varphi\ \mathbf{U}\ \varphi \mid x.\ \varphi$$

- Continuous semantics of TPTL: over $\pi\colon \mathbb{R}^+ \to 2^{\mathsf{AP}}$:
  - $\pi, t, \tau \models x \sim c$ iff $\tau(x) \sim c$
  - $\pi, t, \tau \models x.\ \varphi$ iff $\pi, i, \tau[x\leftarrow 0] \models \varphi$
  - $\pi, t, \tau \models \varphi\ \mathbf{U}\ \psi$ iff there exists some $u > 0$ s.t.
    - $\pi, t + u, \tau + u - t \models \psi$,
    - $\pi, i + k, \tau + v - t \models \varphi$ for all $0 < v < u$.

- Examples:

$x.\ \mathbf{F}(\mathtt{blue} \wedge \mathbf{F}(\mathtt{green} \wedge x \leq 2))$

Outline of the course

1. Branching-time temporal logics
   - Complexity
   - Alternating-time Temporal Logic
2. Timed temporal logics
   - Timed models
   - Timed logics
   - Undecidability

MTL and TPTL are very expressive

Lemma

The halting problem for a Turing machine can be encoded in TPTL and MTL (with past) in both (pointwise and continuous) frameworks.

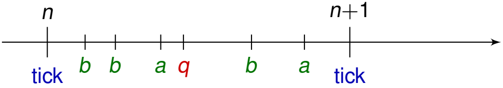
MTL and TPTL are very expressive

Lemma

The halting problem for a Turing machine can be encoded in TPTL and MTL (with past) in both (pointwise and continuous) frameworks.

Proof (sketch).

- the successive configurations of the Turing machine are encoded on a one-time-unit-long segment;



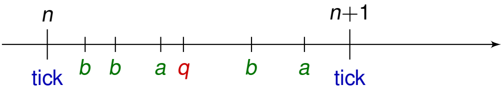
MTL and TPTL are very expressive

Lemma

The halting problem for a Turing machine can be encoded in TPTL and MTL (with past) in both (pointwise and continuous) frameworks.

Proof (sketch).

- the successive configurations of the Turing machine are encoded on a one-time-unit-long segment;
- a transition of the Turing machine is applied between one configuration and its successor;



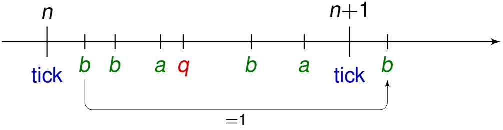
MTL and TPTL are very expressive

Lemma

The halting problem for a Turing machine can be encoded in TPTL and MTL (with past) in both (pointwise and continuous) frameworks.

Proof (sketch).

- the successive configurations of the Turing machine are encoded on a one-time-unit-long segment;
- a transition of the Turing machine is applied between one configuration and its successor;



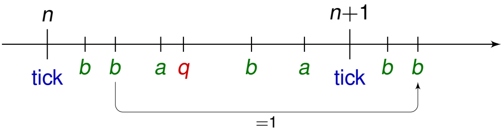
MTL and TPTL are very expressive

Lemma

The halting problem for a Turing machine can be encoded in TPTL and MTL (with past) in both (pointwise and continuous) frameworks.

Proof (sketch).

- the successive configurations of the Turing machine are encoded on a one-time-unit-long segment;
- a transition of the Turing machine is applied between one configuration and its successor;



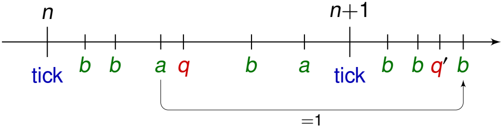
MTL and TPTL are very expressive

Lemma

The halting problem for a Turing machine can be encoded in TPTL and MTL (with past) in both (pointwise and continuous) frameworks.

Proof (sketch).

- the successive configurations of the Turing machine are encoded on a one-time-unit-long segment;
- a transition of the Turing machine is applied between one configuration and its successor;



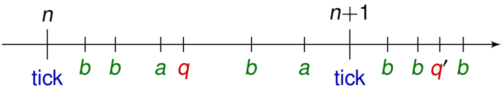
MTL and TPTL are very expressive

Lemma

The halting problem for a Turing machine can be encoded in TPTL and MTL (with past) in both (pointwise and continuous) frameworks.

Proof (sketch).

- the successive configurations of the Turing machine are encoded on a one-time-unit-long segment;
- a transition of the Turing machine is applied between one configuration and its successor;
- the final state of the Turing machine is eventually reached.



MTL and TPTL are very expressive

Theorem

Satisfiability of an MTL- or TPTL-formula is undecidable.

MTL and TPTL are very expressive

Theorem

Satisfiability of an MTL- or TPTL-formula is undecidable.

Definition

MITL is a (syntactic) fragment of MTL where *punctuality* is not allowed: intervals cannot be singletons.

MTL and TPTL are very expressive

Theorem

Satisfiability of an MTL- or TPTL-formula is undecidable.

Definition

MITL is a (syntactic) fragment of MTL where *punctuality* is not allowed: intervals cannot be singletons.

Theorem (Alur, Feder, Henzinger, 1991)

In the continuous semantics, with any MITL formula, we can associate a timed automaton that accepts exactly the same set of timed state sequences.

MTL and TPTL are very expressive

Theorem

Satisfiability of an MTL- or TPTL-formula is undecidable.

Definition

MITL is a (syntactic) fragment of MTL where *punctuality* is not allowed: intervals cannot be singletons.

Theorem (Alur, Feder, Henzinger, 1991)

In the continuous semantics, satisfiability of an MITL formula is EXPSPACE-complete.