VTSA 2024

2023-2024

Formal verification
Part 2: Timed automata

Étienne André

Université Sorbonne Paris Nord
Etienne.Andre@univ-paris13.fr

Version slides: July 12, 2024

Objectives of this part of the module

- introduce formal models for timed critical systems specification
  - timed automata
- use model checking to verify their timed properties
  - properties expressed in MITL and TCTL logics

Beyond finite state automata

Finite State Automata give a simple syntax and a formal semantics to model qualitative aspects of systems

- Executions, sequence of actions
- Modular definitions (parallelism)
- Powerful checking (reachability, safety, liveness…)

Beyond finite state automata

Finite State Automata give a simple syntax and a formal semantics to model qualitative aspects of systems

- Executions, sequence of actions
- Modular definitions (parallelism)
- Powerful checking (reachability, safety, liveness…)

But what about quantitative aspects:

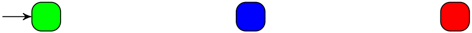
- Time (“the airbag always eventually inflates after a crash”, but maybe 10 seconds after the crash)
- Temperature (“the alarm always eventually ring after the temperature is high”, but maybe when the temperature is above 200 degrees)
- etc.

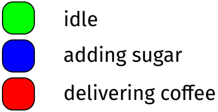
Outline

Outline

Timed automaton (TA)

- Finite-state automaton (sets of locations)

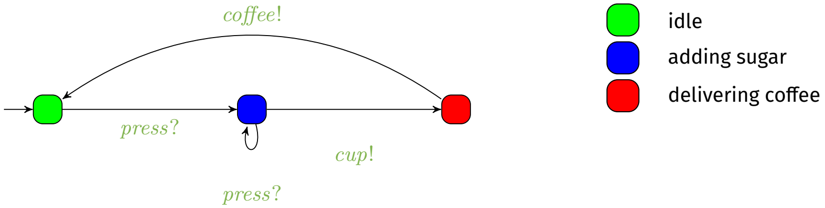




• [AD94] Rajeev Alur and David L. Dill. « A theory of timed automata ». In: *Theoretical Computer Science* 126.2 (Apr. 1994), pp. 183–235

Timed automaton (TA)

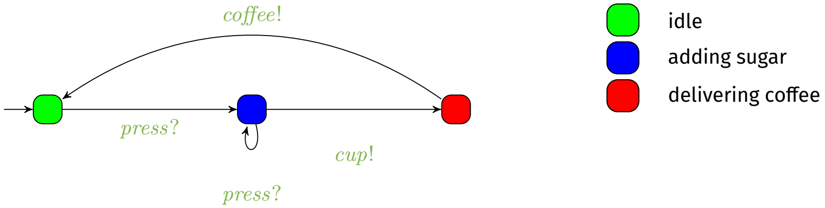
- Finite-state automaton (sets of locations and actions)



. [AD94] Rajeev Alur and David L. Dill. « A theory of timed automata ». In: *Theoretical Computer Science* 126.2 (Apr. 1994), pp. 183–235

Timed automaton (TA)

- Finite-state automaton (sets of locations and actions) augmented with a set X of clocks [AD94]
  - Real-valued variables evolving linearly at the same rate

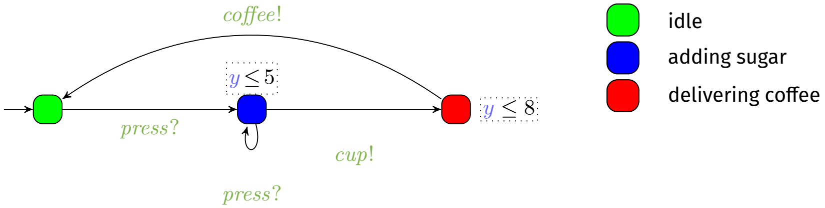


. [AD94] Rajeev Alur and David L. Dill. « A theory of timed automata ». In: *Theoretical Computer Science* 126.2 (Apr. 1994), pp. 183–235

Timed automaton (TA)

- Finite-state automaton (sets of locations and actions) augmented with a set X of clocks [AD94]
  - Real-valued variables evolving linearly at the same rate
  - Can be compared to integer constants in invariants

- Features
  - Location invariant: property to be verified to stay at a location

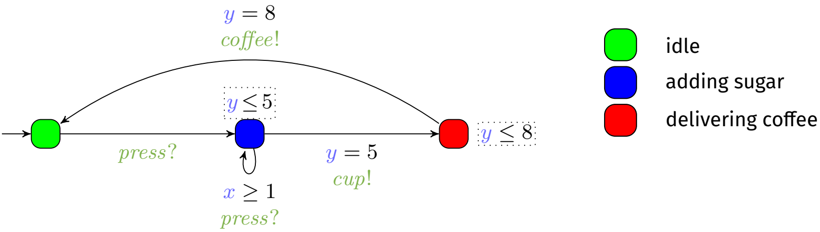


. [AD94] Rajeev Alur and David L. Dill. « A theory of timed automata ». In: *Theoretical Computer Science* 126.2 (Apr. 1994), pp. 183–235

Timed automaton (TA)

- Finite-state automaton (sets of locations and actions) augmented with a set X of clocks [AD94]
  - Real-valued variables evolving linearly at the same rate
  - Can be compared to integer constants in invariants and guards

- Features
  - Location invariant: property to be verified to stay at a location
  - Transition guard: property to be verified to enable a transition

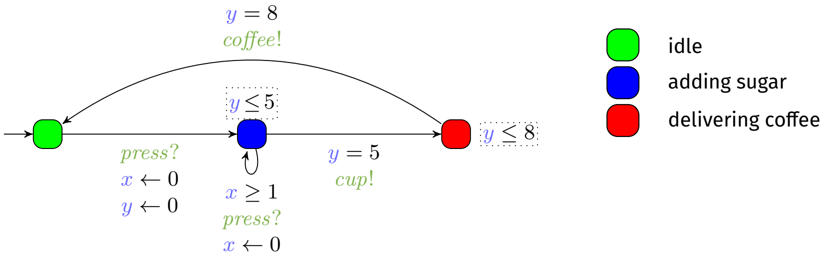


- [AD94] Rajeev Alur and David L. Dill. « A theory of timed automata ». In: *Theoretical Computer Science* 126.2 (Apr. 1994), pp. 183–235

Timed automaton (TA)

- Finite-state automaton (sets of locations and actions) augmented with a set X of clocks [AD94]
  - Real-valued variables evolving linearly at the same rate
  - Can be compared to integer constants in invariants and guards

- Features
  - Location invariant: property to be verified to stay at a location
  - Transition guard: property to be verified to enable a transition
  - Clock reset: some of the clocks can be set to 0 along transitions



. [AD94] Rajeev Alur and David L. Dill. « A theory of timed automata ». In: *Theoretical Computer Science* 126.2 (Apr. 1994), pp. 183–235

Formal definition of timed automata

Definition (Timed automaton)

A timed automaton (TA) $\mathcal{A}$ is a 7-tuple of the form $\mathcal{A} = (L, \Sigma, \ell_0, F, X, I, E)$, where

- L is a finite set of locations,
- $\ell_0 \in L$ is the initial location,
- $F \subseteq L$ is a set of final (or accepting) locations,
- Σ is a finite set of actions,
- X is a set of clocks,
- I is the invariant, assigning to every $\ell \in L$ a constraint $I(\ell)$ on the clocks, and
- E is a step (or “transition”) relation consisting of elements of the form $e = (\ell, g, a, R, \ell')$, where $\ell, \ell' \in L$, $a \in \Sigma$, $R \subseteq X$ is a set of clock variables to be reset by the step, and g (the step guard) is a constraint on the clocks.

Exercise 1

Draw the TA $\mathcal{A} = (L, \Sigma, \ell_1, F, X, I, E)$ such that

- $L = \{\ell_1, \ell_2, \ell_3, \ell_4\}$,
- $F = \{\ell_2, \ell_4\}$,
- $\Sigma = \{a_1, a_2, a_3\}$,
- $X = \{x_1, x_2\}$,
- $I(\ell_1) = x_1 \leq 3$, and $I(\ell_3) = x_2 \geq 2$,
- $E = \Big\{(\ell_1, x_1 \geq 2, a_1, \{x_1\}, \ell_2),$
  $(\ell_1, x_2 \leq 1, a_2, \emptyset, \ell_3),$
  $(\ell_2, x_2 = 1, a_3, \{x_2\}, \ell_2),$
  $(\ell_2, \top, a_1, \emptyset, \ell_3),$
  $(\ell_3, \top, a_2, \{x_1, x_2\}, \ell_4),$
  $(\ell_4, x_2 > 2, a_3, \emptyset, \ell_3)\Big\}$

Exercise 2

Give the formal TA corresponding to the timed coffee machine.

Exercise 2

Give the formal TA corresponding to the timed coffee machine.

$\mathcal{A} = (L, \Sigma,$  $, \{\}, X, I, E)$, with:

- $L = \{$  ,  ,  $\}$,
- $\Sigma = \{press?, cup?, coffee!\}$,
- $X = \{x, y\}$,
- $I($  $) = \top$, $I($  $) = y \leq 5$, and $I($  $) = y \leq 8$,
- $E = \Big\{ ($  $, \top, press?, \{x, y\},$  $),$
  $($  $, x \geq 1, press?, \{x\},$  $),$
  $($  $, y = 5, cup?, \emptyset,$  $),$
  $($  $, y = 8, coffee!, \emptyset,$  $) \Big\}$

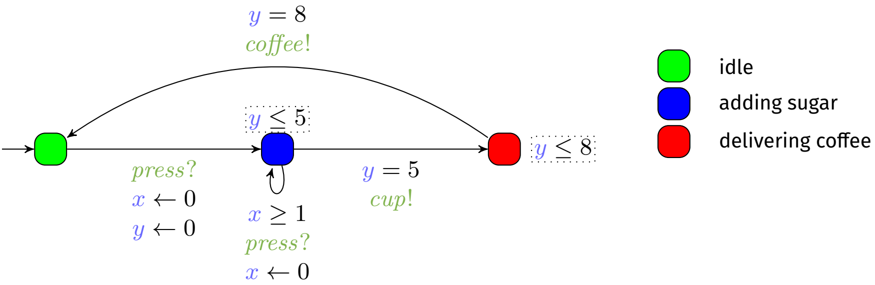
Concrete semantics of timed automata

- Concrete state of a TA: pair (ℓ, w), where
  - ℓ is a location,
  - w is a valuation of each clock

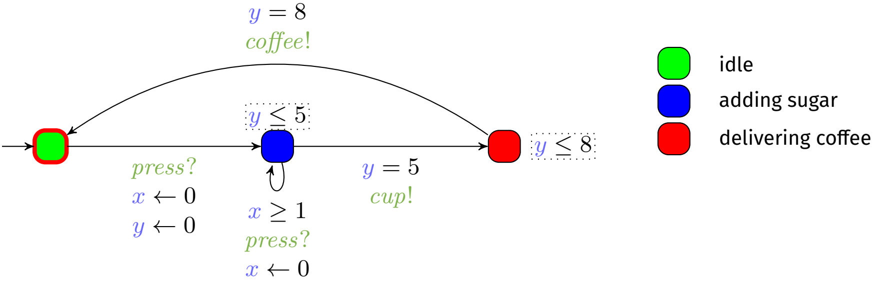
  Example: ( , $\binom{x=1.2}{y=3.7}$)

- Concrete run: alternating sequence of concrete states and actions or time elapse

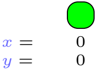
Examples of concrete runs



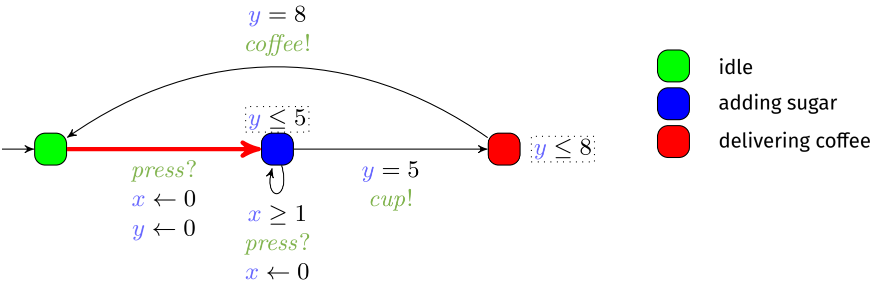
Examples of concrete runs



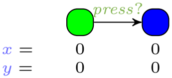
- Example of concrete run for the coffee machine
  - Coffee with 2 doses of sugar



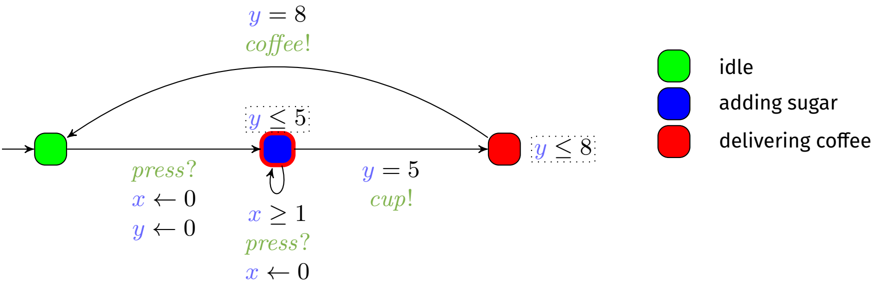
Examples of concrete runs



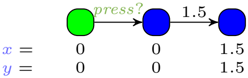
- Example of concrete run for the coffee machine
  - Coffee with 2 doses of sugar



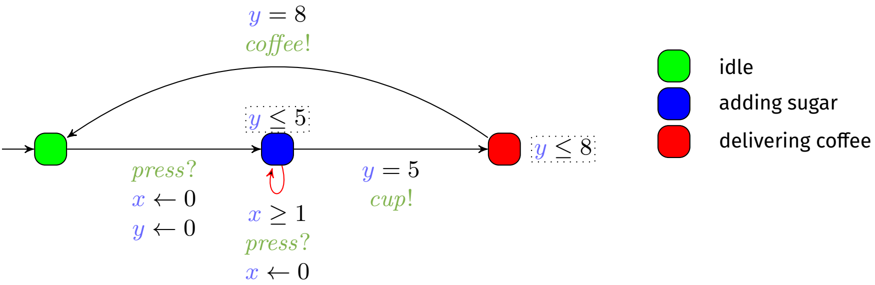
Examples of concrete runs



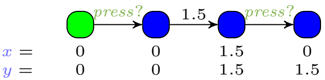
- Example of concrete run for the coffee machine
  - Coffee with 2 doses of sugar



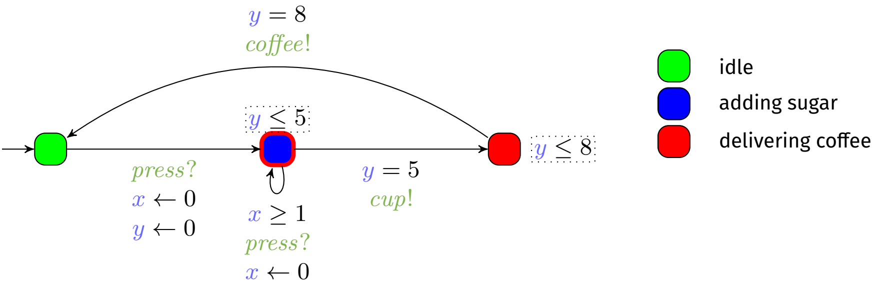
Examples of concrete runs



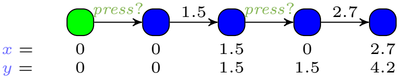
- Example of concrete run for the coffee machine
  - Coffee with 2 doses of sugar



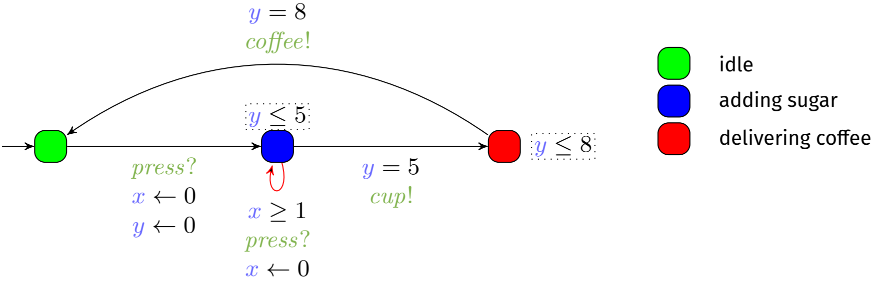
Examples of concrete runs



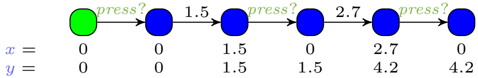
- Example of concrete run for the coffee machine
  - Coffee with 2 doses of sugar



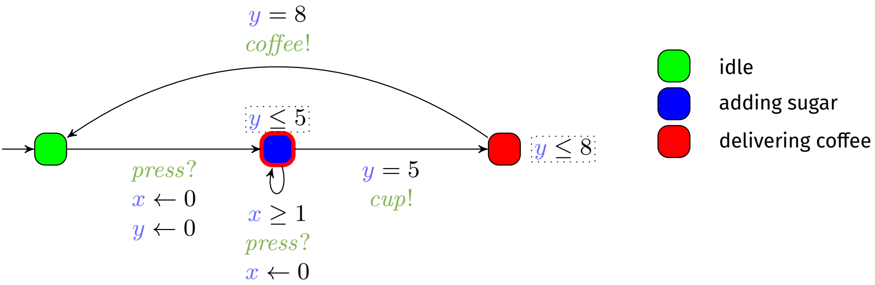
Examples of concrete runs



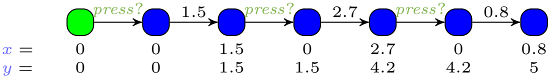
- Example of concrete run for the coffee machine
  - Coffee with 2 doses of sugar



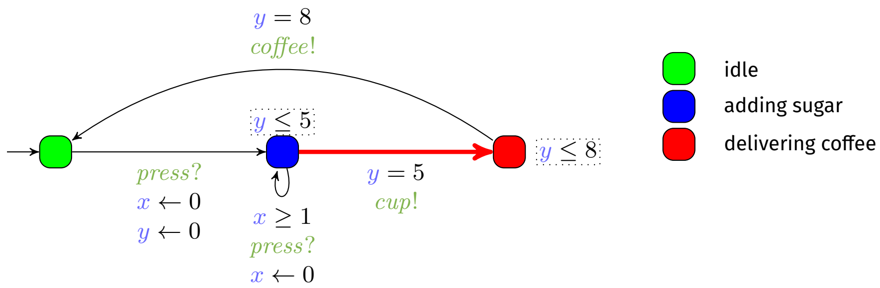
Examples of concrete runs



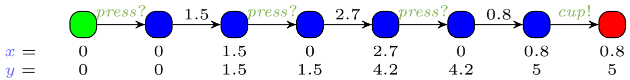
- Example of concrete run for the coffee machine
  - Coffee with 2 doses of sugar



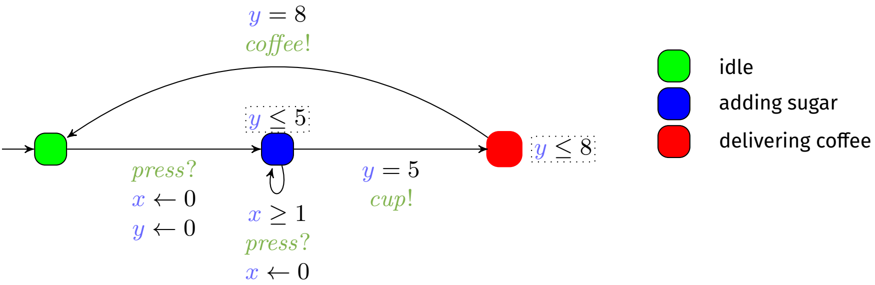
Examples of concrete runs



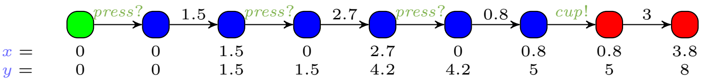
- Example of concrete run for the coffee machine
  - Coffee with 2 doses of sugar



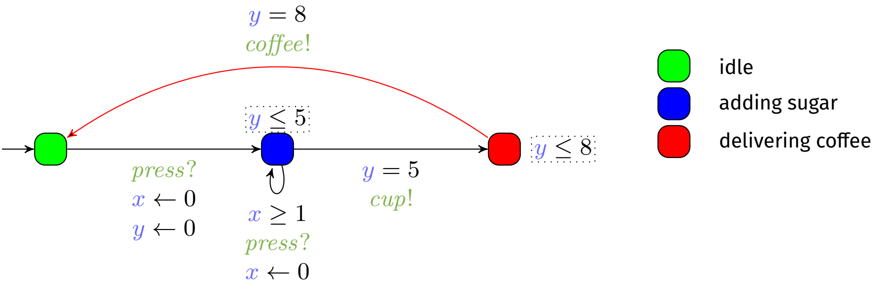
Examples of concrete runs



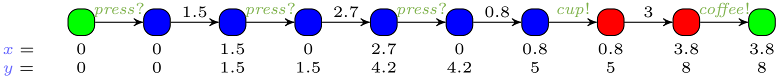
- Example of concrete run for the coffee machine
  - Coffee with 2 doses of sugar



Examples of concrete runs



- Example of concrete run for the coffee machine
  - Coffee with 2 doses of sugar



Formal concrete semantics of timed automata

Definition (Semantics of a timed automaton)

Given a TA $\mathcal{A} = (L, \Sigma, \ell_0, X, I, E)$, the semantics of $\mathcal{A}$ is given by the timed transition system $\mathfrak{T}_{\mathcal{A}} = (\mathfrak{S}, \mathfrak{s}_0, \Sigma \cup \mathbb{R}_{\geq 0}, \rightarrow)$, with

1. $\mathfrak{S} = \big\{(\ell, w) \in L \times \mathbb{R}_{\geq 0}^{|X|} \mid w \models I(\ell)\big\}$,
2. $\mathfrak{s}_0 = (\ell_0, \vec{0})$,
3. $\rightarrow$ consists of the discrete and (continuous) delay transition relations:
   - discrete transitions: $(\ell, w) \overset{e}{\mapsto} (\ell', w')$, if $(\ell, w), (\ell', w') \in \mathfrak{S}$, and there exists $e = (\ell, g, a, R, \ell') \in E$, such that $w' = [w]_R$, and $w \models g$.
   - delay transitions: $(\ell, w) \overset{d}{\mapsto} (\ell, w + d)$, with $d \in \mathbb{R}_{\geq 0}$, if $\forall d' \in [0, d], (\ell, w + d') \in \mathfrak{S}$.

Formal concrete semantics of timed automata

Definition (Semantics of a timed automaton)

Given a TA $\mathcal{A} = (L, \Sigma, \ell_0, X, I, E)$, the semantics of $\mathcal{A}$ is given by the timed transition system $\mathfrak{T}_{\mathcal{A}} = (\mathfrak{S}, \mathfrak{s}_0, \Sigma \cup \mathbb{R}_{\geq 0}, \rightarrow)$, with

1. $\mathfrak{S} = \left\{ (\ell, w) \in L \times \mathbb{R}_{\geq 0}^{|X|} \mid w \models I(\ell) \right\}$,
2. $\mathfrak{s}_0 = (\ell_0, \vec{0})$,
3. $\rightarrow$ consists of the discrete and (continuous) delay transition relations:
   - discrete transitions: $(\ell, w) \overset{e}{\mapsto} (\ell', w')$, if $(\ell, w), (\ell', w') \in \mathfrak{S}$, and there exists $e = (\ell, g, a, R, \ell') \in E$, such that $w' = [w]_R$, and $w \models g$.
   - delay transitions: $(\ell, w) \overset{d}{\mapsto} (\ell, w + d)$, with $d \in \mathbb{R}_{\geq 0}$, if $\forall d' \in [0, d], (\ell, w + d') \in \mathfrak{S}$.

Given two transitions $(\ell, w) \overset{d}{\mapsto} (\ell, w + d) \overset{e}{\mapsto} (\ell', w')$, we usually write $(\ell, w)(d, a)(\ell', w')$ (where a is the action of edge e)

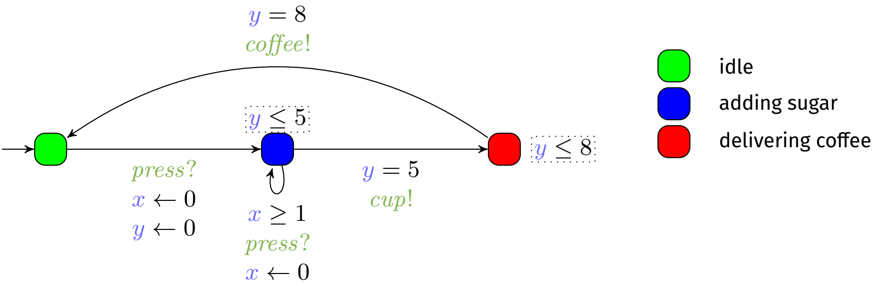
Timed words

Timed word: a sequence of pairs made of

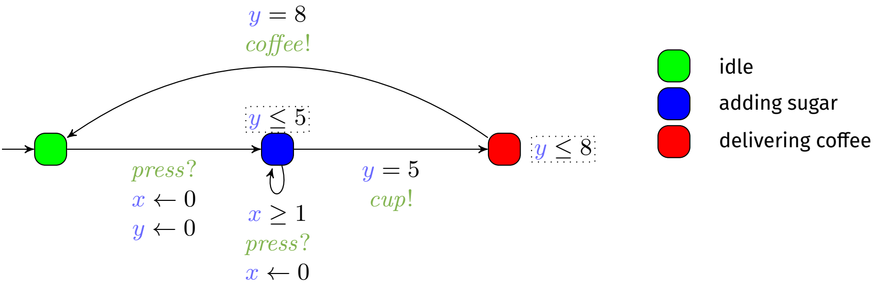
1. an action, and
2. an increasing timestamp in $\mathbb{R}_{\geq 0}$

Given a run $(\ell_0, w_0), (d_0, e_0), (\ell_1, w_1), \cdots, (\ell_n, w_n)$ then it generates the timed word $(a_0, \sum_{i=0}^{0} d_i)(a_1, \sum_{i=0}^{1} d_i) \dots (a_n, \sum_{i=0}^{n} d_i)$

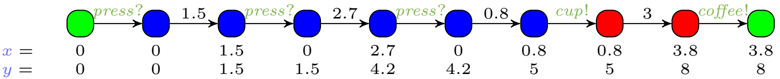
Examples of concrete runs



Examples of concrete runs

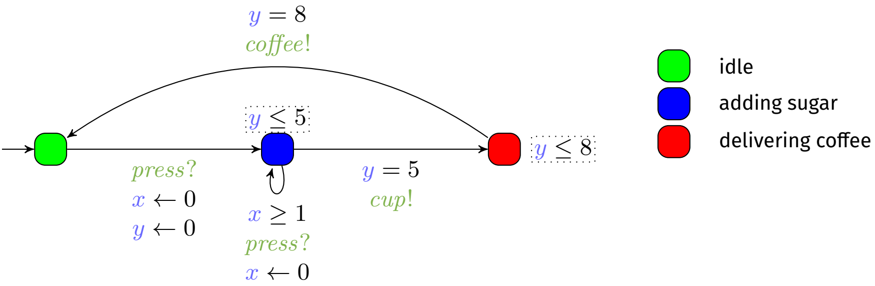


- Example of concrete run for the coffee machine
  - Coffee with 2 doses of sugar

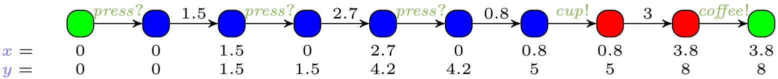


  - Associated timed word:

Examples of concrete runs



- Example of concrete run for the coffee machine
  - Coffee with 2 doses of sugar



  - Associated timed word:
    $(press?, 0)(press?, 1.5)(press?, 4.2)(cup!, 5)(coffee!, 8)$

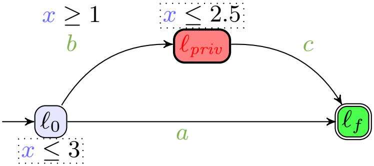
Timed language

Accepting run: run ending in an accepting location (in F)

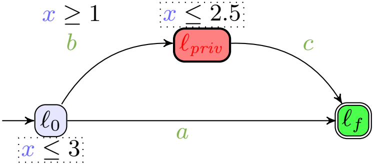
Definition (timed language of a TA)

The timed language of a TA $\mathcal{A}$, denoted by $\mathcal{L}(\mathcal{A})$, is the set of timed words associated to all accepting runs

Examples of accepting runs

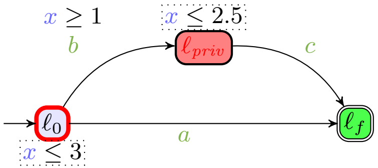


Examples of accepting runs

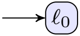


- Two examples of accepting runs

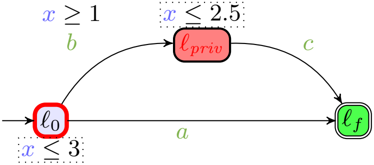
Examples of accepting runs



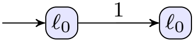
- Two examples of accepting runs



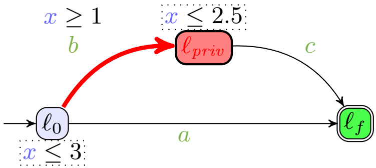
Examples of accepting runs



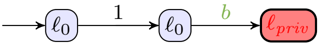
- Two examples of accepting runs



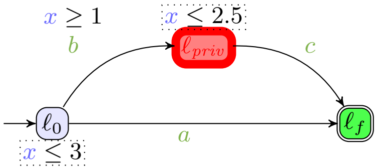
Examples of accepting runs



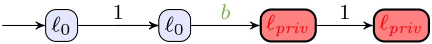
- Two examples of accepting runs



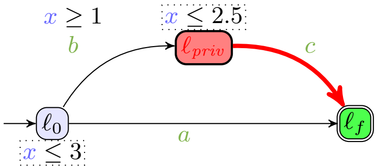
Examples of accepting runs



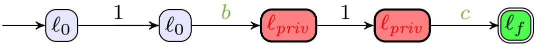
- Two examples of accepting runs



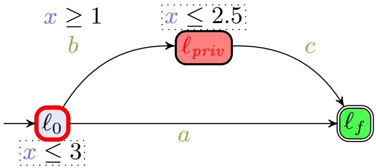
Examples of accepting runs



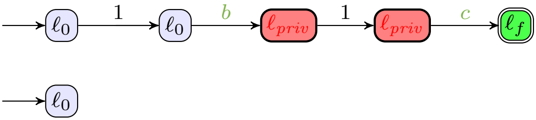
- Two examples of accepting runs



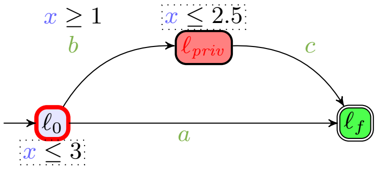
Examples of accepting runs



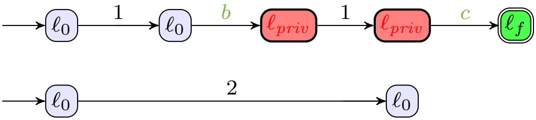
- Two examples of accepting runs



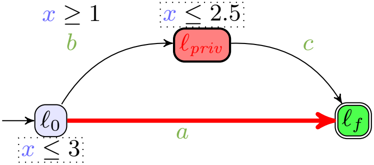
Examples of accepting runs



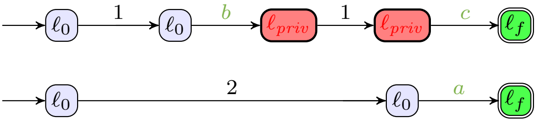
- Two examples of accepting runs



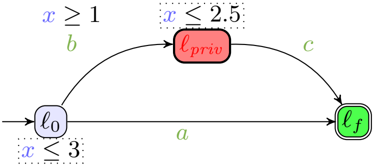
Examples of accepting runs



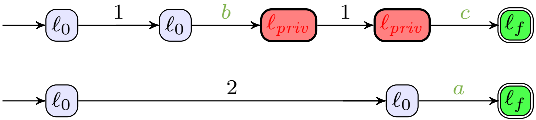
- Two examples of accepting runs



Examples of accepting runs

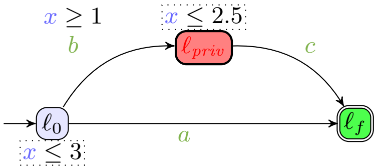


- Two examples of accepting runs

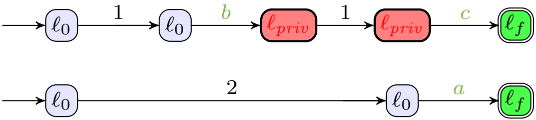


- Timed language of this TA $\mathcal{A}$:

Examples of accepting runs

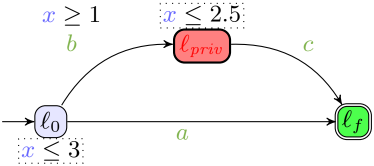


- Two examples of accepting runs

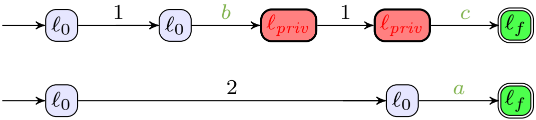


- Timed language of this TA $\mathcal{A}$:
  $\mathcal{L}(\mathcal{A}) = \{(a, i) \mid i \leq 3\}$

Examples of accepting runs



- Two examples of accepting runs



- Timed language of this TA $\mathcal{A}$:
  $\mathcal{L}(\mathcal{A}) = \{(a, i) \mid i \leq 3\} \cup \{(b, i)(c, j) \mid 1 \leq i \leq j \leq 2.5\}$

Outline

What is decidability?

Definition

A decision problem is decidable if one can design an algorithm that, for any input of the problem, can answer yes or no (in a finite time, with a finite memory).

What is decidability?

Definition

A decision problem is decidable if one can design an algorithm that, for any input of the problem, can answer yes or no (in a finite time, with a finite memory).

“given three integers, is one of them the product of the other two?”

“given a context-free grammar, does it generate all strings?”

“given a Turing machine, will it eventually halt?”

“given a timed automaton, does there exist a run from the initial state to a given location ℓ?”

What is decidability?

Definition

A decision problem is decidable if one can design an algorithm that, for any input of the problem, can answer yes or no (in a finite time, with a finite memory).

- $\surd$ "given three integers, is one of them the product of the other two?"
- "given a context-free grammar, does it generate all strings?"
- "given a Turing machine, will it eventually halt?"
- "given a timed automaton, does there exist a run from the initial state to a given location ℓ?"

What is decidability?

Definition

A decision problem is decidable if one can design an algorithm that, for any input of the problem, can answer yes or no (in a finite time, with a finite memory).

- √ "given three integers, is one of them the product of the other two?"
- × "given a context-free grammar, does it generate all strings?"
- "given a Turing machine, will it eventually halt?"
- "given a timed automaton, does there exist a run from the initial state to a given location ℓ?"

What is decidability?

Definition

A decision problem is decidable if one can design an algorithm that, for any input of the problem, can answer yes or no (in a finite time, with a finite memory).

- √ "given three integers, is one of them the product of the other two?"
- × "given a context-free grammar, does it generate all strings?"
- × "given a Turing machine, will it eventually halt?"
- "given a timed automaton, does there exist a run from the initial state to a given location ℓ?"

What is decidability?

Definition

A decision problem is decidable if one can design an algorithm that, for any input of the problem, can answer yes or no (in a finite time, with a finite memory).

- ✓ "given three integers, is one of them the product of the other two?"
- × "given a context-free grammar, does it generate all strings?"
- × "given a Turing machine, will it eventually halt?"
- ✓ "given a timed automaton, does there exist a run from the initial state to a given location ℓ?"

Why studying decidability?

If a decision problem is undecidable, it is hopeless to look for algorithms yielding exact solutions (because that is impossible)

Why studying decidability?

If a decision problem is undecidable, it is hopeless to look for algorithms yielding exact solutions (because that is impossible)

However, one can:

- design semi-algorithms: if the algorithm halts, then its result is correct
- design algorithms yielding over- or under-approximations

Outline

1. Timed automata
   - Syntax and semantics
   - Studying decidability
   - Regions
   - Decision problems
   - Zones

Dense time

- Time is dense: transitions can be taken anytime
  - Infinite number of (concrete) states
  - Infinite number of timed runs
  - Model checking needs a finite structure!

- Some runs are equivalent
  - Taking the $press?$ action at $t = 1.5$ or $t = 1.57$ can be seen as equivalent
  - Good news: clocks evolve at the same speed

- Idea: reason with abstractions
  - region automaton [AD94], and
  - zone automaton [BY03]

- [AD94] Rajeev Alur and David L. Dill. « A theory of timed automata ». In: *Theoretical Computer Science* 126.2 (Apr. 1994), pp. 183–235
- [BY03] Johan Bengtsson and Wang Yi. « Timed Automata: Semantics, Algorithms and Tools ». In: *Lectures on Concurrency and Petri Nets, Advances in Petri Nets*. Vol. 3098. Lecture Notes in Computer Science. Springer, 2003, pp. 87–124

Regions: Intuition

Main idea: given two clock valuations, the exact value of clocks does not matter as long as…

- their integral part is identical
- the relative order of the fractional part is identical
- …or both clock valuations exceed the largest constant of the TA

Regions: Formal definition

Let c_i denote the maximal constant compared to x_i in the TA

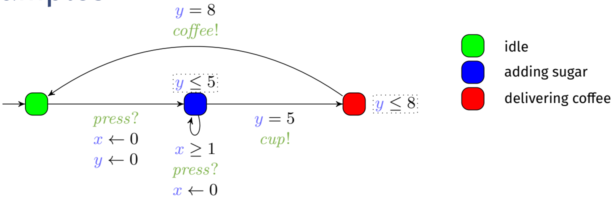
Definition (Region equivalence [AD94])

Two clocks valuations w, w' are *equivalent*, denoted by $w \approx w'$, when the following three conditions hold for any clocks $x_i, x_j \in X$:

1. $\lfloor w(x_i) \rfloor = \lfloor w'(x_i) \rfloor$ or $w(x_i) > c_i$ and $w'(x_i) > c_i$;
2. if $w(x_i) \leq c_i$ and $w(x_j) \leq c_j$, then: fr$(w(x_i)) \leq$ fr$(w(x_j))$ iff fr$(w'(x_i)) \leq$ fr$(w'(x_j))$; and
3. if $w(x_i) \leq c_i$, then: fr$(w(x_i)) = 0$ iff fr$(w'(x_i)) = 0$.

- [AD94] Rajeev Alur and David L. Dill. « A theory of timed automata ». In: *Theoretical Computer Science* 126.2 (Apr. 1994), pp. 183–235

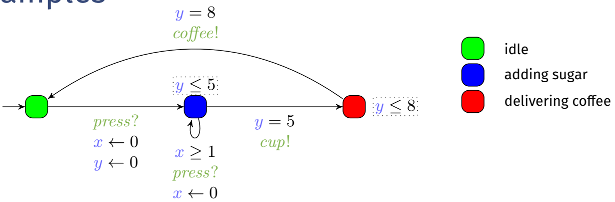
Regions: Examples



- Let w_1 be such that $w_1(x) = 0.5$ and $w_1(y) = 2.7$
- Let w_2 be such that $w_2(x) = 0.2$ and $w_2(y) = 2.8$

w_1 and w_2 are

Regions: Examples



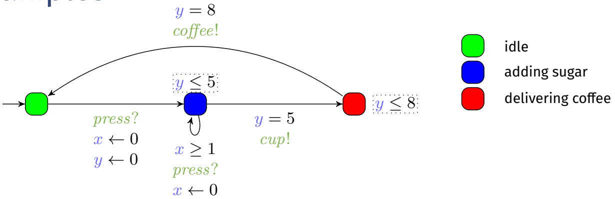
- Let w_1 be such that $w_1(x) = 0.5$ and $w_1(y) = 2.7$
- Let w_2 be such that $w_2(x) = 0.2$ and $w_2(y) = 2.8$

w_1 and w_2 are equivalent

- Let w_3 be such that $w_3(x) = 1.3$ and $w_3(y) = 0.7$
- Let w_4 be such that $w_4(x) = 3.9$ and $w_4(y) = 0.2$

w_3 and w_4 are

Regions: Examples



- Let w_1 be such that $w_1(x) = 0.5$ and $w_1(y) = 2.7$
- Let w_2 be such that $w_2(x) = 0.2$ and $w_2(y) = 2.8$

w_1 and w_2 are equivalent

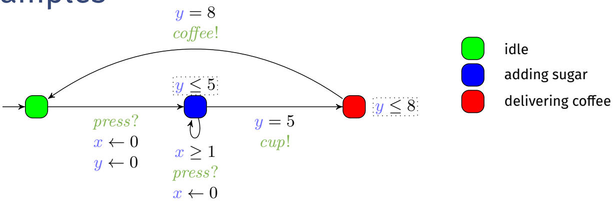
- Let w_3 be such that $w_3(x) = 1.3$ and $w_3(y) = 0.7$
- Let w_4 be such that $w_4(x) = 3.9$ and $w_4(y) = 0.2$

w_3 and w_4 are equivalent (note that the maximum constant of x is 1)

- Let w_5 be such that $w_5(x) = 0.8$ and $w_5(y) = 2.7$

w_1 and w_5 are

Regions: Examples



- Let w_1 be such that $w_1(x) = 0.5$ and $w_1(y) = 2.7$
- Let w_2 be such that $w_2(x) = 0.2$ and $w_2(y) = 2.8$

w_1 and w_2 are equivalent

- Let w_3 be such that $w_3(x) = 1.3$ and $w_3(y) = 0.7$
- Let w_4 be such that $w_4(x) = 3.9$ and $w_4(y) = 0.2$

w_3 and w_4 are equivalent (note that the maximum constant of x is 1)

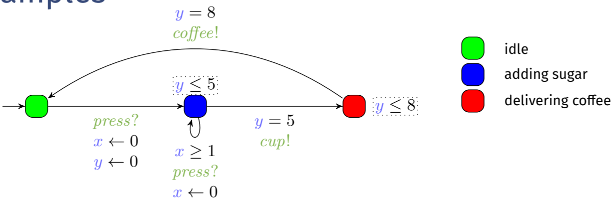
- Let w_5 be such that $w_5(x) = 0.8$ and $w_5(y) = 2.7$

w_1 and w_5 are not equivalent

- Let w_6 be such that $w_6(x) = 0$ and $w_6(y) = 2.8$

w_1 and w_6 are

Regions: Examples



- Let w_1 be such that $w_1(x) = 0.5$ and $w_1(y) = 2.7$
- Let w_2 be such that $w_2(x) = 0.2$ and $w_2(y) = 2.8$

w_1 and w_2 are equivalent

- Let w_3 be such that $w_3(x) = 1.3$ and $w_3(y) = 0.7$
- Let w_4 be such that $w_4(x) = 3.9$ and $w_4(y) = 0.2$

w_3 and w_4 are equivalent (note that the maximum constant of x is 1)

- Let w_5 be such that $w_5(x) = 0.8$ and $w_5(y) = 2.7$

w_1 and w_5 are not equivalent

- Let w_6 be such that $w_6(x) = 0$ and $w_6(y) = 2.8$

w_1 and w_6 are not equivalent

Region graph

Region graph: A nice property

- ☹ The region graph is exponential in the number of clocks (which is not too good)
- ☺ …but at least it is finite, thus allowing for model checking

Outline

Language emptiness

Theorem (reachability [AD94])

Given a TA, deciding whether there exists an accepting run is PSPACE-complete.

Proof.

- PSPACE-membership: region automaton is exponential in the number of clocks, but it is possible to guess a path using only polynomial space
- PSPACE-completeness: by reducing from the question whether a given linear bounded automaton accepts a given input string (which is PSPACE-complete)

□

Alternative formulations:

- Deciding whether the language is empty
- Deciding whether a given location is reachable

This result still holds over discrete time.

- [AD94] Rajeev Alur and David L. Dill. « A theory of timed automata ». In: *Theoretical Computer Science* 126.2 (Apr. 1994), pp. 183–235

Liveness

Theorem (liveness [AD94])

Given a timed Büchi automaton (i. e., a TA with a Büchi acceptance condition), deciding whether there exists an accepting run is PSPACE-complete.

- [AD94] Rajeev Alur and David L. Dill. « A theory of timed automata ». In: *Theoretical Computer Science* 126.2 (Apr. 1994), pp. 183–235

Language universality

Theorem (universality [AD94])

Given a timed automaton, deciding whether the language is universal (i. e., accept all timed words) is undecidable.

Proof.

By reducing from the problem asking whether a nondeterministic 2-counter machine has a recurring computation, which is undecidable. □

- [AD94] Rajeev Alur and David L. Dill. « A theory of timed automata ». In: *Theoretical Computer Science* 126.2 (Apr. 1994), pp. 183–235

Language inclusion

Theorem (inclusion [AD94])

Given two TAs $\mathcal{A}_1$ and $\mathcal{A}_2$, deciding whether the language of $\mathcal{A}_1$ is included in the language of $\mathcal{A}_2$ is undecidable.

- [AD94] Rajeev Alur and David L. Dill. « A theory of timed automata ». In: *Theoretical Computer Science* 126.2 (Apr. 1994), pp. 183–235

Language inclusion

Theorem (inclusion [AD94])

Given two TAs $\mathcal{A}_1$ and $\mathcal{A}_2$, deciding whether the language of $\mathcal{A}_1$ is included in the language of $\mathcal{A}_2$ is undecidable.

Proof.

By reducing from universality.
Let $\mathcal{A}$ be a TA. Let $\mathcal{A}_{univ}$ be a TA accepting every timed word. Then $\mathcal{A}$ is universal (which is undecidable) iff the language of $\mathcal{A}_{univ}$ is included in the language of $\mathcal{A}$. □

- [AD94] Rajeev Alur and David L. Dill. « A theory of timed automata ». In: *Theoretical Computer Science* 126.2 (Apr. 1994), pp. 183–235

Language equivalence

Theorem (equivalence [AD94])

Given two TAs $\mathcal{A}_1$ and $\mathcal{A}_2$, deciding whether the language of $\mathcal{A}_1$ is identical to the language of $\mathcal{A}_2$ is undecidable.

. [AD94] Rajeev Alur and David L. Dill. « A theory of timed automata ». In: *Theoretical Computer Science* 126.2 (Apr. 1994), pp. 183–235

Language equivalence

Theorem (equivalence [AD94])

Given two TAs $\mathcal{A}_1$ and $\mathcal{A}_2$, deciding whether the language of $\mathcal{A}_1$ is identical to the language of $\mathcal{A}_2$ is undecidable.

Proof.

By reducing from universality. □

- [AD94] Rajeev Alur and David L. Dill. « A theory of timed automata ». In: *Theoretical Computer Science* 126.2 (Apr. 1994), pp. 183–235

Complementability

Theorem (complementability [Tri06] [Fin06])

Given a TA $\mathcal{A}$, whether the complement of $\mathcal{L}(\mathcal{A})$ can be accepted by a TA is undecidable.

- [Tri06] Stavros Tripakis. « Folk theorems on the determinization and minimization of timed automata ». In: *Information Processing Letters* 99.6 (2006), pp. 222–226
- [Fin06] Olivier Finkel. « Undecidable Problems About Timed Automata ». In: *FORMATS*. vol. 4202. Lecture Notes in Computer Science. Springer, 2006, pp. 187–199

Outline

Symbolic states for timed automata (zones)

- Objective: group all concrete states reachable by the same sequence of discrete actions
- Symbolic state: a location ℓ and a (infinite) set of states Z
- For timed automata, Z can be represented by a convex polyhedron with a special form called zone, with constraints

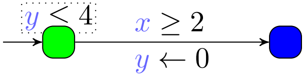
$$-d_{0i} \leq x_i \leq d_{i0} \text{ and } x_i - x_j \leq d_{ij}$$

- Computation of successive reachable symbolic states can be performed symbolically with polyhedral operations: for edge $e = (\ell, a, g, R, \ell')$:

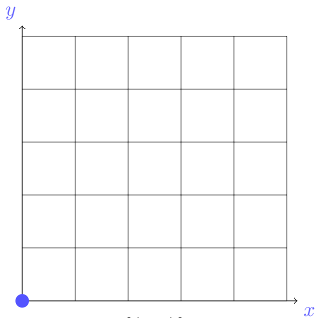
$$\mathsf{Succ}((\ell, Z), e) = (\ell', [(Z \cap g)]_R \cap I(\ell'))^{\nearrow} \cap I(\ell'))$$

- With an additional technicality, there is a finite number of reachable zones in a TA.

Symbolic states for timed automata (zones): Example

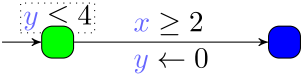


$$\text{Succ}((\square, Z), e) = (\blacksquare, [(Z \cap g)]_R \cap I(\blacksquare)))^{\nearrow} \cap I(\blacksquare))$$

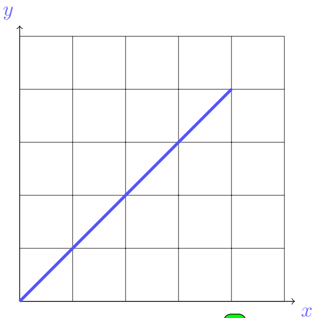


$\{(0, 0)\}$

Symbolic states for timed automata (zones): Example

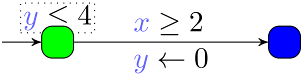


$$\mathsf{Succ}((\color{green}\blacksquare\color{black}, Z), e) = (\color{blue}\blacksquare\color{black}, [(Z \cap g)]_R \cap I(\color{blue}\blacksquare\color{black}))^{\nearrow} \cap I(\color{blue}\blacksquare\color{black}))$$

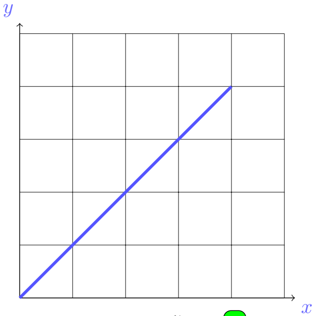


$Z_0 = \{(0,0)\}^{\nearrow} \cap I(\color{green}\blacksquare\color{black})$

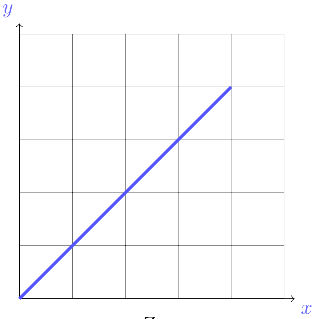
Symbolic states for timed automata (zones): Example



$$\mathsf{Succ}((\text{green}, Z), e) = (\text{blue}, [(Z \cap g)]_R \cap I(\text{blue}))^{\nearrow} \cap I(\text{blue}))$$

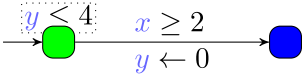


$Z_0 = \{(0,0)\}^{\nearrow} \cap I(\text{green})$

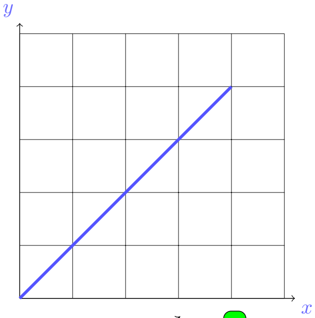


Z_0

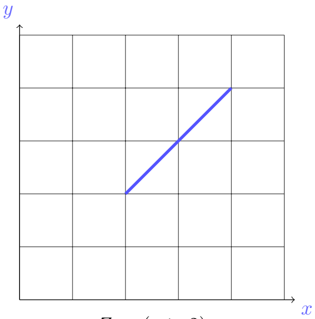
Symbolic states for timed automata (zones): Example



$$\mathsf{Succ}((\text{green location}, Z), e) = (\text{blue location}, [(Z \cap g)]_R \cap I(\text{blue location}))^{\nearrow} \cap I(\text{blue location}))$$

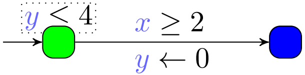


$Z_0 = \{(0,0)\}^{\nearrow} \cap I(\text{green location})$

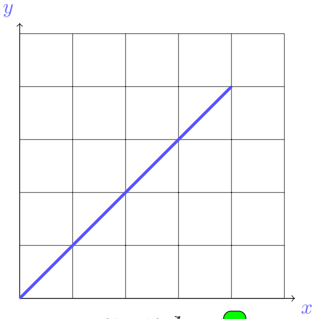


$Z_0 \cap (x \geq 2)$

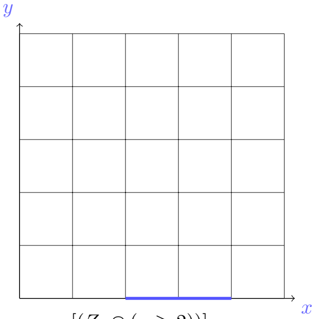
Symbolic states for timed automata (zones): Example



$$\mathsf{Succ}((\;, Z), e) = (\;, [(Z \cap g)]_R \cap I(\;))^{\nearrow} \cap I(\;))$$

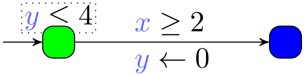


$Z_0 = \{(0,0)\}^{\nearrow} \cap I(\;)$

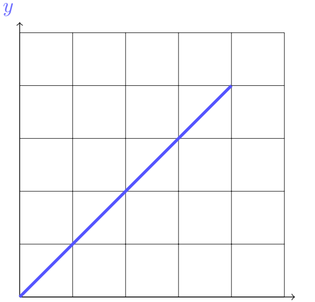


$[(Z_0 \cap (x \geq 2))]_{\{y\}}$

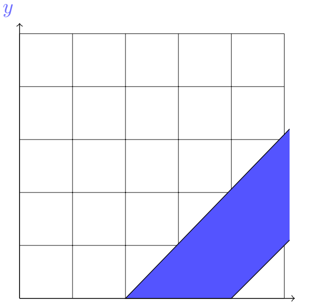
Symbolic states for timed automata (zones): Example



$$\mathsf{Succ}((\blacksquare, Z), e) = (\blacksquare, [(Z \cap g)]_R \cap I(\blacksquare))^{\nearrow} \cap I(\blacksquare))$$



$Z_0 = \{(0,0)\}^{\nearrow} \cap I(\blacksquare)$



$Z_1 = [(Z_0 \cap (x \geq 2))]_{\{y\}}^{\nearrow}$

Finiteness of the zone graph

- With an additional technicality, there is a finite number of reachable zones in a TA
  - See zone-based abstractions [Beh+06] [HSW16] [Bou+22]

. [Beh+06] Gerd Behrmann, Patricia Bouyer, Kim Guldstrand Larsen, and Radek Pelánek. « Lower and upper bounds in zone-based abstractions of timed automata ». In: *International Journal on Software Tools for Technology Transfer* 8.3 (2006), pp. 204–215

. [HSW16] Frédéric Herbreteau, B. Srivathsan, and Igor Walukiewicz. « Better abstractions for timed automata ». In: *Information and Computation* 251 (2016), pp. 67–90

. [Bou+22] Patricia Bouyer, Paul Gastin, Frédéric Herbreteau, Ocan Sankur, and B. Srivathsan. « Zone-Based Verification of Timed Automata: Extrapolations, Simulations and What Next? » In: *FORMATS*. vol. 13465. Lecture Notes in Computer Science. Springer, 2022, pp. 16–42

Abstract semantics of timed automata

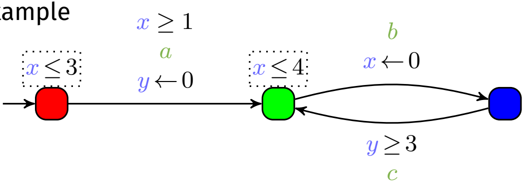
- Abstract state of a TA: pair (ℓ, C), where
  - ℓ is a location, and C is a constraint on the clocks ("zone")

Abstract semantics of timed automata

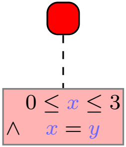
- Abstract state of a TA: pair (ℓ, C), where
  - ℓ is a location, and C is a constraint on the clocks ("zone")
- Abstract run: alternating sequence of abstract states and actions

Abstract semantics of timed automata

- Abstract state of a TA: pair (ℓ, C), where
  - ℓ is a location, and C is a constraint on the clocks ("zone")
- Abstract run: alternating sequence of abstract states and actions
- Example

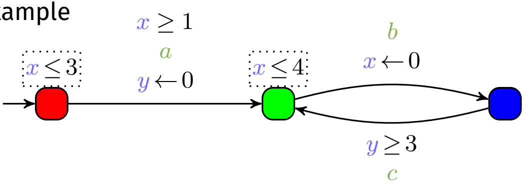


  - Possible abstract run for this TA

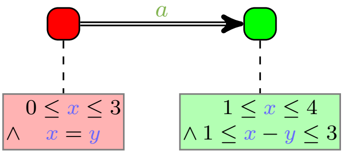


Abstract semantics of timed automata

- Abstract state of a TA: pair (ℓ, C), where
  - ℓ is a location, and C is a constraint on the clocks ("zone")
- Abstract run: alternating sequence of abstract states and actions
- Example

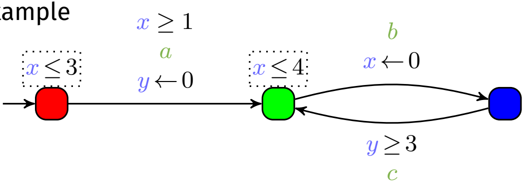


  - Possible abstract run for this TA

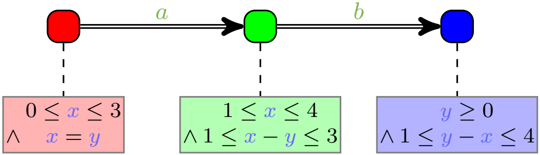


Abstract semantics of timed automata

- Abstract state of a TA: pair (ℓ, C), where
  - ℓ is a location, and C is a constraint on the clocks ("zone")
- Abstract run: alternating sequence of abstract states and actions
- Example

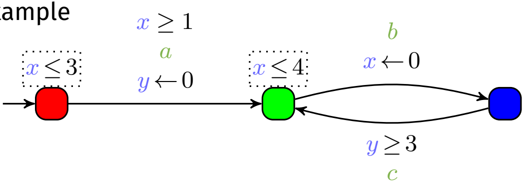


  - Possible abstract run for this TA

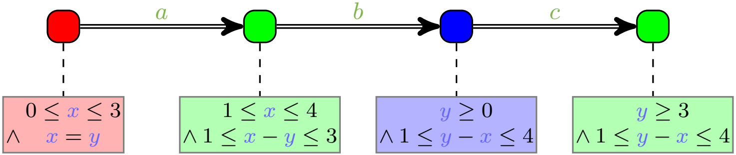


Abstract semantics of timed automata

- Abstract state of a TA: pair (ℓ, C), where
  - ℓ is a location, and C is a constraint on the clocks ("zone")
- Abstract run: alternating sequence of abstract states and actions
- Example



  - Possible abstract run for this TA



Difference bound matrices (DBMs)

Objectives:

- Represent zones using a canonical representation
- Allow for efficient zone operations
  - Much faster than normal polyhedra!

Principle:

- Matrix of size $|X| + 1$
- Includes a special “clock” of value 0

Difference bound matrices: Principle

$$\begin{pmatrix} 0 & c_{01} & c_{02} \\ c_{10} & 0 & c_{12} \\ c_{20} & c_{21} & 0 \end{pmatrix}$$

Each cell c_{ij} represents a constraint of the form $x_i - x_j \leq c_{ij}$ (with $x_0 = 0$)

Difference bound matrices: Example

Exercise

What is the polyhedron encoded by the following DBM?

$$\begin{pmatrix} 0 & 3 & 5 \\ -2 & 0 & 1 \\ -4 & -1 & 0 \end{pmatrix}$$

Difference bound matrices: Example

Exercise

What is the polyhedron encoded by the following DBM?

$$\begin{pmatrix} 0 & 3 & 5 \\ -2 & 0 & 1 \\ -4 & -1 & 0 \end{pmatrix}$$

$$x_1 \leq -2$$

Difference bound matrices: Example

Exercise

What is the polyhedron encoded by the following DBM?

$$\begin{pmatrix} 0 & 3 & 5 \\ -2 & 0 & 1 \\ -4 & -1 & 0 \end{pmatrix}$$

$$\begin{array}{lrcl} & x_1 & \leq & -2 \\ \wedge & x_2 & \leq & -4 \end{array}$$

Difference bound matrices: Example

Exercise

What is the polyhedron encoded by the following DBM?

$$\begin{pmatrix} 0 & 3 & 5 \\ -2 & 0 & 1 \\ -4 & -1 & 0 \end{pmatrix}$$

$$\begin{array}{rrcl} & x_1 & \leq & -2 \\ \wedge & x_2 & \leq & -4 \\ \wedge & -x_1 & \leq & 3 \\ \wedge & x_2 - x_1 & \leq & -1 \\ \wedge & -x_2 & \leq & 5 \\ \wedge & x_1 - x_2 & \leq & 1 \end{array}$$

Difference bound matrices: Strict constraints

To differentiate between strict and non-strict constraints, one considers in fact a pair $(c_{ij}, \triangleleft)$, with $\triangleleft \in \{<, \leq\}$

$$\begin{pmatrix} 0 & (c_{01}, \triangleleft_{01}) & (c_{02}, \triangleleft_{02}) \\ (c_{10}, \triangleleft_{10}) & 0 & (c_{12}, \triangleleft_{12}) \\ (c_{20}, \triangleleft_{20} & (c_{21}, \triangleleft_{21}) & 0 \end{pmatrix}$$

Each cell c_{ij} represents a constraint of the form $x_i - x_j \triangleleft_{ij} c_{ij}$ (with $x_0 = 0$)

Difference bound matrices: Example

Exercise

What is the polyhedron encoded by the following DBM?

$$\begin{pmatrix} 0 & (3, <) & \infty \\ (-2, \leq) & 0 & (1, \leq) \\ (-4, <) & (-1, \leq) & 0 \end{pmatrix}$$

Difference bound matrices: Example

Exercise

What is the polyhedron encoded by the following DBM?

$$\begin{pmatrix} 0 & (3, <) & \infty \\ (-2, \leq) & 0 & (1, \leq) \\ (-4, <) & (-1, \leq) & 0 \end{pmatrix}$$

$$\begin{array}{rrcl} & x_1 & \leq & -2 \\ \wedge & x_2 & < & -4 \\ \wedge & -x_1 & < & 3 \\ \wedge & x_2 - x_1 & \leq & -1 \\ \wedge & x_1 - x_2 & \leq & 1 \end{array}$$

Difference bound matrices: Example

Exercise ([BY03])

What is the DBM encoding the following polyhedron?

$$x_1 < 20 \land x_2 \leq 20 \land x_2 - x_1 \leq 10 \land x_1 - x_2 \leq -10 \land -x_3 < 5$$

[BY03] Johan Bengtsson and Wang Yi. « Timed Automata: Semantics, Algorithms and Tools ». In: *Lectures on Concurrency and Petri Nets, Advances in Petri Nets*. Vol. 3098. Lecture Notes in Computer Science. Springer, 2003, pp. 87–124

Difference bound matrices: Example

Exercise ([BY03])

What is the DBM encoding the following polyhedron?

$$x_1 < 20 \wedge x_2 \leq 20 \wedge x_2 - x_1 \leq 10 \wedge x_1 - x_2 \leq -10 \wedge -x_3 < 5$$

$$\begin{pmatrix} 0 & (0, \leq) & (0, \leq) & (5, <) \\ (20, <) & 0 & (-10, \leq) & \infty \\ (20, \leq) & (10, \leq) & 0 & \infty \\ \infty & \infty & \infty & 0 \end{pmatrix}$$

- [BY03] Johan Bengtsson and Wang Yi. « Timed Automata: Semantics, Algorithms and Tools ». In: *Lectures on Concurrency and Petri Nets, Advances in Petri Nets*. Vol. 3098. Lecture Notes in Computer Science. Springer, 2003, pp. 87–124

Operations on DBMs: Time elapsing

Time elapsing: unconstraining a zone when time elapses

For DBMs:

Operations on DBMs: Time elapsing

Time elapsing: unconstraining a zone when time elapses

For DBMs: simply replace the c_{i0} cells with ∞

Example

Before time elapsing

$$\begin{pmatrix} 0 & (3, \leq) & (5, <) \\ (2, \leq) & 0 & (1, \leq) \\ (4, <) & (-1, \leq) & 0 \end{pmatrix}$$

After time elapsing

$$\begin{pmatrix} 0 & (3, \leq) & (5, <) \\ \infty & 0 & (1, \leq) \\ \infty & (-1, \leq) & 0 \end{pmatrix}$$

Operations on DBMs: other operations

- time backwards
- conjunction with a guard
- variable elimination
- clock reset
- copying a clock to another one
- shifting a clock (with an integer value)

Some operations need zone normalization [BY03]

- needs a shortest path algorithm for graphs (typically Floyd-Warshall)

[BY03] Johan Bengtsson and Wang Yi. « Timed Automata: Semantics, Algorithms and Tools ». In: *Lectures on Concurrency and Petri Nets, Advances in Petri Nets*. Vol. 3098. Lecture Notes in Computer Science. Springer, 2003, pp. 87–124

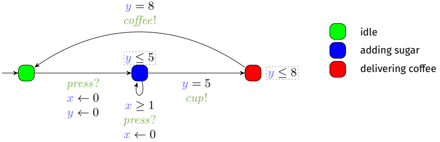
Remarks on timed automata

- Timed automata can be composed just as finite-state automata
- Symbolic states can be efficiently computed using Difference Bound Matrices (DBMs)
- Observers (both untimed and timed) can be used for timed automata
  - Checking a property modeled using an observer reduces to reachability

The expressive power of observers for timed automata has been studied in [ABL98] [Ace+03]

- [ABL98] Luca Aceto, Augusto Burgueño, and Kim Guldstrand Larsen. « Model Checking via Reachability Testing for Timed Automata ». In: *TACAS*. vol. 1384. Lecture Notes in Computer Science. Springer, 1998, pp. 263–280. ISBN: 3-540-64356-7
- [Ace+03] Luca Aceto, Patricia Bouyer, Augusto Burgueño, and Kim Guldstrand Larsen. « The power of reachability testing for timed automata ». In: *Theoretical Computer Science* 300.1-3 (2003), pp. 411–475

Exercise: An observer for the coffee machine



1. Design an observer for the coffee machine verifying that it must never happen that the button can be pressed twice within a time strictly less than 1 unit of time.
2. What is the reachability property?

Outline

Metric Temporal Logics

- Extension of LTL with timing constraints on modalities
- Specify properties on the order and the delay between atomic propositions
- No X modality because

Metric Temporal Logics

- Extension of LTL with timing constraints on modalities
- Specify properties on the order and the delay between atomic propositions
- No X modality because of dense time

Syntax of MTL

MTL = Metric Temporal Logics

Definition (Syntax of MTL)

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

where I is an interval with bounds in $\mathbb{Q}_+ \cup \{\infty\}$

Syntax of MTL

MTL = Metric Temporal Logics

Definition (Syntax of MTL)

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

where I is an interval with bounds in $\mathbb{Q}_+ \cup \{\infty\}$

Two semantics:

- pointwise semantics
- continuous semantics

Continuous semantics of MTL

Definition (Continuous semantics of MTL)

$\rho, t \models p$	if	$p \in lab\big(\rho(t)\big)$
$\rho, t \models \neg\varphi$	if	$\rho, t \not\models \varphi$
$\rho, t \models \varphi \vee \psi$	if	$\rho, t \models \varphi$ or $\rho, t \models \psi$
$\rho, t \models \varphi \mathsf{U}_I \psi$	if	$\exists u$ s.t. $u > 0 : \rho, t{+}u \models \psi$ and $\forall 0 < v < u : \rho, t{+}v \models \varphi$ and $u \in I$

Continuous semantics of MTL

Definition (Continuous semantics of MTL)

$\rho, t \models p$	if	$p \in lab\big(\rho(t)\big)$
$\rho, t \models \neg\varphi$	if	$\rho, t \not\models \varphi$
$\rho, t \models \varphi \lor \psi$	if	$\rho, t \models \varphi$ or $\rho, t \models \psi$
$\rho, t \models \varphi \mathsf{U}_I \psi$	if	$\exists u$ s.t. $u > 0 : \rho, t{+}u \models \psi$ and $\forall 0 < v < u : \rho, t{+}v \models \varphi$ and $u \in I$

Example

- 

Continuous semantics of MTL

Definition (Continuous semantics of MTL)

$\rho, t \models p$	if	$p \in lab\big(\rho(t)\big)$
$\rho, t \models \neg\varphi$	if	$\rho, t \not\models \varphi$
$\rho, t \models \varphi \vee \psi$	if	$\rho, t \models \varphi$ or $\rho, t \models \psi$
$\rho, t \models \varphi \mathsf{U}_I \psi$	if	$\exists u$ s.t. $u > 0 : \rho, t{+}u \models \psi$ and $\forall 0 < v < u : \rho, t{+}v \models \varphi$ and $u \in I$

Example

- ( $\vee$ ) $\mathsf{U}_{\leq 2}$  valid
- $\mathsf{F}_{=2}$ 

Continuous semantics of MTL

Definition (Continuous semantics of MTL)

$\rho, t \models p$	if	$p \in lab\big(\rho(t)\big)$
$\rho, t \models \neg\varphi$	if	$\rho, t \not\models \varphi$
$\rho, t \models \varphi \lor \psi$	if	$\rho, t \models \varphi$ or $\rho, t \models \psi$
$\rho, t \models \varphi \mathsf{U}_I \psi$	if	$\exists u$ s.t. $u > 0 : \rho, t+u \models \psi$ and $\forall 0 < v < u : \rho, t+v \models \varphi$ and $u \in I$

Example

- $(\;\lor\;)\mathsf{U}_{\leq 2}$ valid
- $\mathsf{F}_{=2}$ valid

MTL: Extended syntax

$$\varphi \wedge \psi \quad \equiv$$

MTL: Extended syntax

$$\varphi \wedge \psi \equiv \neg(\neg\varphi \vee \neg\psi)$$

$$\varphi \implies \psi \equiv$$

MTL: Extended syntax

$$\varphi \wedge \psi \equiv \neg(\neg\varphi \vee \neg\psi)$$

$$\varphi \implies \psi \equiv \neg\varphi \vee \psi$$

$$\varphi \iff \psi \equiv$$

MTL: Extended syntax

$$\varphi \wedge \psi \equiv \neg(\neg\varphi \vee \neg\psi)$$

$$\varphi \implies \psi \equiv \neg\varphi \vee \psi$$

$$\varphi \iff \psi \equiv (\varphi \implies \psi) \wedge (\psi \implies \varphi)$$

$$\mathsf{F}_I \varphi \equiv$$

MTL: Extended syntax

$$\varphi \wedge \psi \equiv \neg(\neg\varphi \vee \neg\psi)$$

$$\varphi \implies \psi \equiv \neg\varphi \vee \psi$$

$$\varphi \iff \psi \equiv (\varphi \implies \psi) \wedge (\psi \implies \varphi)$$

$$\mathsf{F}_I \varphi \equiv \mathtt{true} \mathsf{U}_I \varphi$$

$$\mathsf{G}_I \varphi \equiv$$

MTL: Extended syntax

$$\varphi \land \psi \equiv \neg(\neg\varphi \lor \neg\psi)$$

$$\varphi \implies \psi \equiv \neg\varphi \lor \psi$$

$$\varphi \iff \psi \equiv (\varphi \implies \psi) \land (\psi \implies \varphi)$$

$$\mathsf{F}_I \varphi \equiv \mathtt{true}\mathsf{U}_I \varphi$$

$$\mathsf{G}_I \varphi \equiv \neg(\mathsf{F}_I \neg\varphi)$$

$$\varphi \mathsf{W}_I \psi \equiv$$

MTL: Extended syntax

$$\varphi \wedge \psi \equiv \neg(\neg\varphi \vee \neg\psi)$$

$$\varphi \implies \psi \equiv \neg\varphi \vee \psi$$

$$\varphi \iff \psi \equiv (\varphi \implies \psi) \wedge (\psi \implies \varphi)$$

$$\mathsf{F}_I \varphi \equiv \mathtt{true} \mathsf{U}_I \varphi$$

$$\mathsf{G}_I \varphi \equiv \neg(\mathsf{F}_I \neg\varphi)$$

$$\varphi \mathsf{W}_I \psi \equiv (\varphi \mathsf{U}_I \psi) \vee \mathsf{G}_{\leq I} \varphi \ \textbf{(?)}$$

(where $\leq I$ denotes the downward-closed interval of I intersected with $\mathbb{Q}_+$)

MTL: Examples

Exercise

Express in MTL the following properties:

- "I will eventually get a job within a year" (liveness property)

MTL: Examples

Exercise

Express in MTL the following properties:

- "I will eventually get a job within a year" (liveness property)

  $\mathsf{F}_{\leq 365}\mathrm{job}$

- "The plane will never crash within the 8 hours of the flight" (safety property)

MTL: Examples

Exercise

Express in MTL the following properties:

- "I will eventually get a job within a year" (liveness property)
  $\mathsf{F}_{\leq 365}\mathrm{job}$
- "The plane will never crash within the 8 hours of the flight" (safety property)
  $\mathsf{G}_{\leq 8}\neg\mathrm{crash}$
- "Every time I ask a question, the teacher will answer me immediately"
  ((immediate) response property)

MTL: Examples

Exercise

Express in MTL the following properties:

- "I will eventually get a job within a year" (liveness property)
  $\mathsf{F}_{\leq 365}\mathrm{job}$
- "The plane will never crash within the 8 hours of the flight" (safety property)
  $\mathsf{G}_{\leq 8}\neg\mathrm{crash}$
- "Every time I ask a question, the teacher will answer me immediately" ((immediate) response property)
  $\mathsf{G}(\mathrm{ask} \implies \mathsf{F}_{=0}\mathrm{answer})$
- "During the next 4 hours, I will starve unless I eventually get some food or drinks"

MTL: Examples

Exercise

Express in MTL the following properties:

- "I will eventually get a job within a year" (liveness property)
  $\mathsf{F}_{\leq 365}\mathrm{job}$
- "The plane will never crash within the 8 hours of the flight" (safety property)
  $\mathsf{G}_{\leq 8}\neg\mathrm{crash}$
- "Every time I ask a question, the teacher will answer me immediately" ((immediate) response property)
  $\mathsf{G}(\mathrm{ask} \implies \mathsf{F}_{=0}\mathrm{answer})$
- "During the next 4 hours, I will starve unless I eventually get some food or drinks"
  $\mathrm{hungry}\,\mathsf{W}_{\leq 4}(\mathrm{food} \vee \mathrm{drinks})$

MTL model checking

Theorem (undecidability [AH93])

MTL model checking and satisfiability are undecidable under the continuous semantics.

Proof idea.

By reduction from the halting problem of a Turing machine. □

- [AH93] Rajeev Alur and Thomas A. Henzinger. « Real-Time Logics: Complexity and Expressiveness ». In: *Information and Computation* 104.1 (1993), pp. 35–77

Syntax of MITL

MTL = Metric Interval Temporal Logics

Definition (Syntax of MITL [AFH96])

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

where I is a non-punctual interval with bounds in $\mathbb{Q}_+ \cup \{\infty\}$

- [AFH96] Rajeev Alur, Tomás Feder, and Thomas A. Henzinger. « The Benefits of Relaxing Punctuality ». In: *Journal of the ACM* 43.1 (1996), pp. 116–146

Syntax of MITL

MTL = Metric Interval Temporal Logics

Definition (Syntax of MITL [AFH96])

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

where I is a non-punctual interval with bounds in $\mathbb{Q}_+ \cup \{\infty\}$

Example

- ☺ $\mathsf{G}(P \implies \mathsf{F}_{[2024,2025]} Q)$ is an MITL formula
- ☹ $\mathsf{G}(P \implies \mathsf{F}_{[2024,2024]} Q)$ is not

. [AFH96] Rajeev Alur, Tomás Feder, and Thomas A. Henzinger. « The Benefits of Relaxing Punctuality ». In: *Journal of the ACM* 43.1 (1996), pp. 116–146

Model checking MITL

Theorem (decidability of MITL [AFH96])

MITL model checking and satisfiability are EXPSACE-complete.

. [AFH96] Rajeev Alur, Tomás Feder, and Thomas A. Henzinger. « The Benefits of Relaxing Punctuality ». In: *Journal of the ACM* 43.1 (1996), pp. 116–146

Model checking MITL: Method

Similar to LTL:

Principle for checking whether $\mathcal{A} \models \varphi$

1. Construct the timed automaton $\mathcal{B}_{\neg\varphi}$ recognizing all executions not satisfying φ
2. Construct the synchronized product $\mathcal{A} \times \mathcal{B}_{\neg\varphi}$
3. If its timed language is empty, then $\mathcal{A} \models \varphi$

Note: translating an MITL formula to a timed automaton isn't that easy!

[AFH96] [MNP06] [Bri+17] [Bri+18]

- [AFH96] Rajeev Alur, Tomás Feder, and Thomas A. Henzinger. « The Benefits of Relaxing Punctuality ». In: *Journal of the ACM* 43.1 (1996), pp. 116–146
- [MNP06] Oded Maler, Dejan Ničković, and Amir Pnueli. « From MITL to Timed Automata ». In: *FORMATS*. vol. 4202. Lecture Notes in Computer Science. Springer, 2006, pp. 274–289
- [Bri+17] Thomas Brihaye, Gilles Geeraerts, Hsi-Ming Ho, and Benjamin Monmege. « MightyL: A Compositional Translation from MITL to Timed Automata ». In: *CAV, Part I*. vol. 10426. Lecture Notes in Computer Science. Springer, 2017, pp. 421–440
- [Bri+18] Thomas Brihaye, Gilles Geeraerts, Hsi-Ming Ho, Arthur Milchior, and Benjamin Monmege. « Efficient Algorithms and Tools for MITL Model-Checking and Synthesis ». In: *ICECCS*. IEEE Computer Society, 2018, pp. 180–184

MITL: Examples

Exercise

Express in MTL the following properties:

- “The plane will never crash within the 8 hours of the flight” (safety property)

MITL: Examples

Exercise

Express in MTL the following properties:

- "The plane will never crash within the 8 hours of the flight" (safety property)
  $\mathsf{G}_{\leq 8} \neg \mathrm{crash}$
- "I will get a job in one year" (liveness property)

MITL: Examples

Exercise

Express in MTL the following properties:

- "The plane will never crash within the 8 hours of the flight" (safety property)
  $G_{\leq 8} \neg \mathrm{crash}$
- "I will get a job in one year" (liveness property)
  not expressible using MITL
- "During the next 4 hours, I will starve unless I eventually get some food or drinks"

MITL: Examples

Exercise

Express in MTL the following properties:

- "The plane will never crash within the 8 hours of the flight" (safety property)
  $G_{\leq 8} \neg \mathrm{crash}$
- "I will get a job in one year" (liveness property)
  not expressible using MITL
- "During the next 4 hours, I will starve unless I eventually get some food or drinks"
  $\mathrm{hungry} W_{\leq 4} (\mathrm{food} \vee \mathrm{drinks})$
- "Every time I ask a question, the teacher will answer me at least 2 and at most 4 minutes later" (response property)

MITL: Examples

Exercise

Express in MTL the following properties:

- "The plane will never crash within the 8 hours of the flight" (safety property)
  $\mathsf{G}_{\leq 8} \neg \text{crash}$
- "I will get a job in one year" (liveness property)
  not expressible using MITL
- "During the next 4 hours, I will starve unless I eventually get some food or drinks"
  $\text{hungry} \mathsf{W}_{\leq 4} (\text{food} \lor \text{drinks})$
- "Every time I ask a question, the teacher will answer me at least 2 and at most 4 minutes later" (response property)
  $\mathsf{G}(\text{ask} \implies \mathsf{F}_{[2,4]} \text{answer})$
- "Every time I ask a question, the teacher will answer me immediately"

MITL: Examples

Exercise

Express in MTL the following properties:

- "The plane will never crash within the 8 hours of the flight" (safety property)
  $G_{\leq 8} \neg \mathrm{crash}$
- "I will get a job in one year" (liveness property)
  not expressible using MITL
- "During the next 4 hours, I will starve unless I eventually get some food or drinks"
  $\mathrm{hungry} W_{\leq 4} (\mathrm{food} \vee \mathrm{drinks})$
- "Every time I ask a question, the teacher will answer me at least 2 and at most 4 minutes later" (response property)
  $G(\mathrm{ask} \implies F_{[2,4]} \mathrm{answer})$
- "Every time I ask a question, the teacher will answer me immediately"
  not expressible using MITL

Outline

TCTL (Timed CTL) [ACD93]

TCTL expresses formulas on the order and the time between the future atomic propositions for some or for all paths, over a set of atomic propositions AP

Definition (Syntax of TCTL)

$$TCTL \ni \varphi ::= p \mid \neg\varphi \mid \varphi \vee \varphi \mid \mathsf{E}\varphi\mathsf{U}_{\sim c}\psi \mid \mathsf{A}\varphi\mathsf{U}_{\sim c}\psi$$

where $\sim \in \{<, \leq, =, \geq, >\}$ and $c \in \mathbb{Q}_+$

Example

- $\mathsf{AG}($  $\implies \mathsf{EF}_{\leq 5}$  $)$
- $\mathsf{AF}(\mathsf{AG}_{\leq 5}$  $)$

[ACD93] Rajeev Alur, Costas Courcoubetis, and David L. Dill. « Model-Checking in Dense Real-Time ». In: *Information and Computation* 104.1 (May 1993), pp. 2–34

Semantics of TCTL

Definition (Semantics of TCTL)

$(\ell, w) \models p$	if	$p \in lab(\ell)$
$(\ell, w) \models \neg\varphi$	if	$(\ell, w) \not\models \varphi$
$(\ell, w) \models \varphi \vee \psi$	if	$(\ell, w) \models \varphi$ or $(\ell, w) \models \psi$
$(\ell, w) \models \mathsf{E}\varphi\mathsf{U}_{\sim c}\psi$	if	there is a run from (ℓ, w) to (ℓ', w') s.t. $t(w') - t(w) \sim c$, for all (ℓ'', w'') between (ℓ, w) and (ℓ', w'), we have $(\ell'', w'') \models \varphi$, and $(\ell', w') \models \psi$
$(\ell, w) \models \mathsf{A}\varphi\mathsf{U}_{\sim c}\psi$	if	for all runs such that, etc.

Example

- EU$_{\leq c}$

TCTL: Extended syntax

- As for CTL and MTL, additional operators can be defined:

$$\varphi \land \psi \qquad \equiv$$

TCTL: Extended syntax

- As for CTL and MTL, additional operators can be defined:

$$\varphi \land \psi \equiv \neg(\neg\varphi \lor \neg\psi)$$

$$\varphi \implies \psi \equiv$$

TCTL: Extended syntax

- As for CTL and MTL, additional operators can be defined:

$$\varphi \land \psi \equiv \neg(\neg\varphi \lor \neg\psi)$$

$$\varphi \implies \psi \equiv \neg\varphi \lor \psi$$

$$\varphi \iff \psi \equiv$$

TCTL: Extended syntax

- As for CTL and MTL, additional operators can be defined:

$$\varphi \wedge \psi \equiv \neg(\neg\varphi \vee \neg\psi)$$

$$\varphi \implies \psi \equiv \neg\varphi \vee \psi$$

$$\varphi \iff \psi \equiv (\varphi \implies \psi) \wedge (\psi \implies \varphi)$$

$$\mathsf{EF}_I \varphi \equiv$$

TCTL: Extended syntax

- As for CTL and MTL, additional operators can be defined:

$$\varphi \land \psi \equiv \neg(\neg\varphi \lor \neg\psi)$$

$$\varphi \implies \psi \equiv \neg\varphi \lor \psi$$

$$\varphi \iff \psi \equiv (\varphi \implies \psi) \land (\psi \implies \varphi)$$

$$\mathsf{EF}_I\varphi \equiv \mathsf{E}\,\mathtt{true}\,\mathsf{U}_I\varphi$$

$$\mathsf{AF}_I\varphi \equiv$$

TCTL: Extended syntax

- As for CTL and MTL, additional operators can be defined:

$$\varphi \land \psi \equiv \neg(\neg\varphi \lor \neg\psi)$$

$$\varphi \implies \psi \equiv \neg\varphi \lor \psi$$

$$\varphi \iff \psi \equiv (\varphi \implies \psi) \land (\psi \implies \varphi)$$

$$\mathsf{EF}_I\varphi \equiv \mathsf{E}\texttt{true}\mathsf{U}_I\varphi$$

$$\mathsf{AF}_I\varphi \equiv \mathsf{A}\texttt{true}\mathsf{U}_I\varphi$$

$$\mathsf{EG}_I\varphi \equiv$$

TCTL: Extended syntax

- As for CTL and MTL, additional operators can be defined:

$$\varphi \wedge \psi \equiv \neg(\neg\varphi \vee \neg\psi)$$

$$\varphi \implies \psi \equiv \neg\varphi \vee \psi$$

$$\varphi \iff \psi \equiv (\varphi \implies \psi) \wedge (\psi \implies \varphi)$$

$$\mathsf{EF}_I\varphi \equiv \mathsf{E}\texttt{true}\mathsf{U}_I\varphi$$

$$\mathsf{AF}_I\varphi \equiv \mathsf{A}\texttt{true}\mathsf{U}_I\varphi$$

$$\mathsf{EG}_I\varphi \equiv \neg(\mathsf{AF}_I\neg\varphi)$$

$$\mathsf{AG}_I\varphi \equiv$$

TCTL: Extended syntax

- As for CTL and MTL, additional operators can be defined:

$$\varphi \land \psi \equiv \neg(\neg\varphi \lor \neg\psi)$$

$$\varphi \implies \psi \equiv \neg\varphi \lor \psi$$

$$\varphi \iff \psi \equiv (\varphi \implies \psi) \land (\psi \implies \varphi)$$

$$\mathsf{EF}_I\varphi \equiv \mathsf{E}\texttt{true}\mathsf{U}_I\varphi$$

$$\mathsf{AF}_I\varphi \equiv \mathsf{A}\texttt{true}\mathsf{U}_I\varphi$$

$$\mathsf{EG}_I\varphi \equiv \neg(\mathsf{AF}_I\neg\varphi)$$

$$\mathsf{AG}_I\varphi \equiv \neg(\mathsf{EF}_I\neg\varphi)$$

$$\mathsf{E}\varphi\mathsf{W}_I\psi \equiv$$

TCTL: Extended syntax

- As for CTL and MTL, additional operators can be defined:

$$\varphi \wedge \psi \equiv \neg(\neg\varphi \vee \neg\psi)$$

$$\varphi \implies \psi \equiv \neg\varphi \vee \psi$$

$$\varphi \iff \psi \equiv (\varphi \implies \psi) \wedge (\psi \implies \varphi)$$

$$\mathsf{EF}_I\varphi \equiv \mathsf{E}\mathtt{true}\mathsf{U}_I\varphi$$

$$\mathsf{AF}_I\varphi \equiv \mathsf{A}\mathtt{true}\mathsf{U}_I\varphi$$

$$\mathsf{EG}_I\varphi \equiv \neg(\mathsf{AF}_I\neg\varphi)$$

$$\mathsf{AG}_I\varphi \equiv \neg(\mathsf{EF}_I\neg\varphi)$$

$$\mathsf{E}\varphi\mathsf{W}_I\psi \equiv (\mathsf{E}\varphi\mathsf{U}_I\psi) \vee \mathsf{EG}_{\leq I}\varphi \text{ (?)}$$

$$\mathsf{A}\varphi\mathsf{W}_I\psi \equiv$$

TCTL: Extended syntax

- As for CTL and MTL, additional operators can be defined:

$$\varphi \land \psi \equiv \neg(\neg\varphi \lor \neg\psi)$$

$$\varphi \implies \psi \equiv \neg\varphi \lor \psi$$

$$\varphi \iff \psi \equiv (\varphi \implies \psi) \land (\psi \implies \varphi)$$

$$\mathsf{EF}_I\varphi \equiv \mathsf{E}\,\mathtt{true}\,\mathsf{U}_I\varphi$$

$$\mathsf{AF}_I\varphi \equiv \mathsf{A}\,\mathtt{true}\,\mathsf{U}_I\varphi$$

$$\mathsf{EG}_I\varphi \equiv \neg(\mathsf{AF}_I\neg\varphi)$$

$$\mathsf{AG}_I\varphi \equiv \neg(\mathsf{EF}_I\neg\varphi)$$

$$\mathsf{E}\varphi\mathsf{W}_I\psi \equiv (\mathsf{E}\varphi\mathsf{U}_I\psi) \lor \mathsf{EG}_{\leq I}\varphi \textbf{ (?)}$$

$$\mathsf{A}\varphi\mathsf{W}_I\psi \equiv \mathsf{A}(\varphi\mathsf{U}_I\psi) \lor \mathsf{AG}_I\varphi$$

TCTL: Extended syntax

- As for CTL and MTL, additional operators can be defined:

$$\varphi \land \psi \equiv \neg(\neg\varphi \lor \neg\psi)$$

$$\varphi \implies \psi \equiv \neg\varphi \lor \psi$$

$$\varphi \iff \psi \equiv (\varphi \implies \psi) \land (\psi \implies \varphi)$$

$$\mathsf{EF}_I\varphi \equiv \mathsf{E}\,\mathtt{true}\,\mathsf{U}_I\varphi$$

$$\mathsf{AF}_I\varphi \equiv \mathsf{A}\,\mathtt{true}\,\mathsf{U}_I\varphi$$

$$\mathsf{EG}_I\varphi \equiv \neg(\mathsf{AF}_I\neg\varphi)$$

$$\mathsf{AG}_I\varphi \equiv \neg(\mathsf{EF}_I\neg\varphi)$$

$$\mathsf{E}\varphi\mathsf{W}_I\psi \equiv (\mathsf{E}\varphi\mathsf{U}_I\psi) \lor \mathsf{EG}_{\leq I}\varphi \text{ (?)}$$

$$\mathsf{A}\varphi\mathsf{W}_I\psi \equiv \mathsf{A}(\varphi\mathsf{U}_I\psi) \lor \mathsf{AG}_I\varphi \text{ (according to ChatGPT}$$

TCTL: Extended syntax

- As for CTL and MTL, additional operators can be defined:

$$\begin{aligned}
\varphi \land \psi &\equiv \neg(\neg\varphi \lor \neg\psi) \\
\varphi \implies \psi &\equiv \neg\varphi \lor \psi \\
\varphi \iff \psi &\equiv (\varphi \implies \psi) \land (\psi \implies \varphi) \\
\mathsf{EF}_I\varphi &\equiv \mathsf{E}\mathtt{true}\mathsf{U}_I\varphi \\
\mathsf{AF}_I\varphi &\equiv \mathsf{A}\mathtt{true}\mathsf{U}_I\varphi \\
\mathsf{EG}_I\varphi &\equiv \neg(\mathsf{AF}_I\neg\varphi) \\
\mathsf{AG}_I\varphi &\equiv \neg(\mathsf{EF}_I\neg\varphi) \\
\mathsf{E}\varphi\mathsf{W}_I\psi &\equiv (\mathsf{E}\varphi\mathsf{U}_I\psi) \lor \mathsf{EG}_{\leq I}\varphi \text{ (?)} \\
\mathsf{A}\varphi\mathsf{W}_I\psi &\equiv \mathsf{A}(\varphi\mathsf{U}_I\psi) \lor \mathsf{AG}_I\varphi \text{ (according to ChatGPT (which is wrong))} \\
&\equiv
\end{aligned}$$

TCTL: Extended syntax

- As for CTL and MTL, additional operators can be defined:

$$\varphi \wedge \psi \equiv \neg(\neg\varphi \vee \neg\psi)$$

$$\varphi \implies \psi \equiv \neg\varphi \vee \psi$$

$$\varphi \iff \psi \equiv (\varphi \implies \psi) \wedge (\psi \implies \varphi)$$

$$\mathsf{EF}_I\varphi \equiv \mathsf{E}\,\mathtt{true}\,\mathsf{U}_I\varphi$$

$$\mathsf{AF}_I\varphi \equiv \mathsf{A}\,\mathtt{true}\,\mathsf{U}_I\varphi$$

$$\mathsf{EG}_I\varphi \equiv \neg(\mathsf{AF}_I\neg\varphi)$$

$$\mathsf{AG}_I\varphi \equiv \neg(\mathsf{EF}_I\neg\varphi)$$

$$\mathsf{E}\varphi\mathsf{W}_I\psi \equiv (\mathsf{E}\varphi\mathsf{U}_I\psi) \vee \mathsf{EG}_{\leq I}\varphi \text{ (?)}$$

$$\mathsf{A}\varphi\mathsf{W}_I\psi \equiv \mathsf{A}(\varphi\mathsf{U}_I\psi) \vee \mathsf{AG}_I\varphi \text{ (according to ChatGPT (which is wrong))}$$

$$\equiv \neg\big(\mathsf{E}(\neg\psi\mathsf{U}_I(\neg\psi \wedge \neg\varphi))\big) \text{ (??)}$$

TCTL: Examples

- "Whatever happens, the plane will never crash in the next 10 minutes"

TCTL: Examples

- “Whatever happens, the plane will never crash in the next 10 minutes”
  $\mathsf{AG}_{\leq 10 minutes} \neg \mathrm{crash}$

- “I may get a job before next year”

TCTL: Examples

- "Whatever happens, the plane will never crash in the next 10 minutes"
  $\mathsf{AG}_{\leq 10 minutes} \neg \mathrm{crash}$
- "I may get a job before next year"
  $\mathsf{EF}_{\leq 1 year} \mathrm{job}$
- "Whenever a fire breaks, it is sure that the alarm will start ringing at least 5 seconds and at most 10 seconds later"

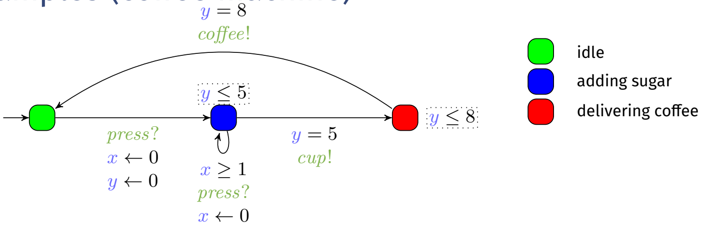
TCTL: Examples

- "Whatever happens, the plane will never crash in the next 10 minutes"
  $\mathsf{AG}_{\leq 10 minutes} \neg \mathrm{crash}$
- "I may get a job before next year"
  $\mathsf{EF}_{\leq 1 year} \mathrm{job}$
- "Whenever a fire breaks, it is sure that the alarm will start ringing at least 5 seconds and at most 10 seconds later"
  $\mathsf{AG}\big(\mathrm{fire} \implies (\mathsf{AF}_{[5,10]} \mathrm{alarm})\big)$
- "Whatever happens, I will love you for 2 years after we marry"

TCTL: Examples

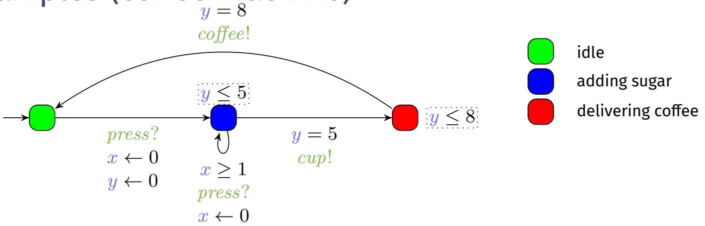
- "Whatever happens, the plane will never crash in the next 10 minutes"
  $\mathsf{AG}_{\leq 10 minutes} \neg \mathrm{crash}$
- "I may get a job before next year"
  $\mathsf{EF}_{\leq 1 year} \mathrm{job}$
- "Whenever a fire breaks, it is sure that the alarm will start ringing at least 5 seconds and at most 10 seconds later"
  $\mathsf{AG}\big(\mathrm{fire} \implies (\mathsf{AF}_{[5,10]} \mathrm{alarm})\big)$
- "Whatever happens, I will love you for 2 years after we marry"
  $\mathsf{AG}\big(\mathrm{marry} \implies (\mathsf{AG}_{\leq 2} \mathrm{love})\big)$

TCTL: Examples (coffee machine)



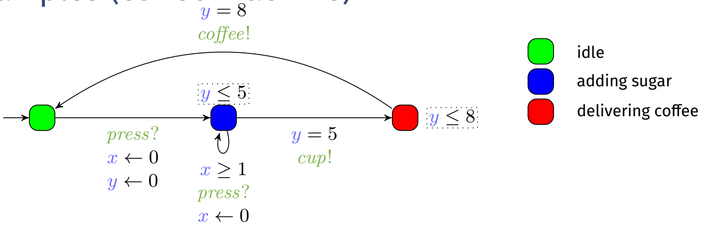
- "Whenever the button is pressed, a coffee is necessarily eventually delivered within 10 units of time."

TCTL: Examples (coffee machine)



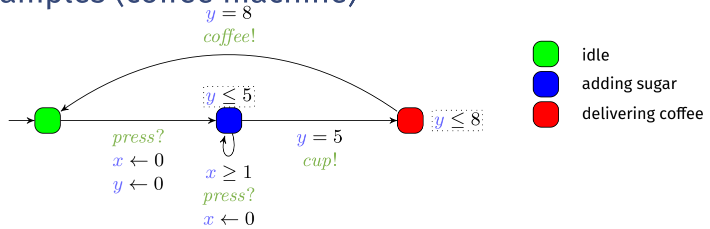
- "Whenever the button is pressed, a coffee is necessarily eventually delivered within 10 units of time."
  $\mathsf{AG}\Big(\mathit{press} \implies (\mathsf{AF}_{\leq 10}\,\mathit{coffee})\Big)$

TCTL: Examples (coffee machine)



- "Whenever the button is pressed, a coffee is necessarily eventually delivered within 10 units of time."
  $\mathsf{AG}\big(\mathit{press} \implies (\mathsf{AF}_{\leq 10}\, \mathit{coffee})\big)$ ($\checkmark$)

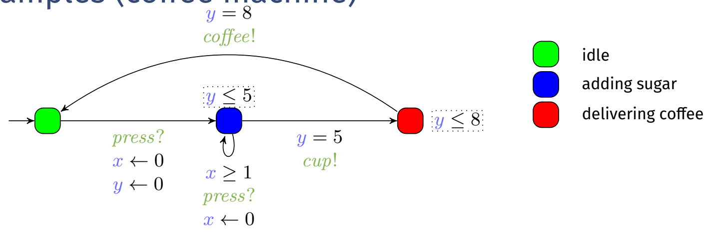
TCTL: Examples (coffee machine)



- "Whenever the button is pressed, a coffee is necessarily eventually delivered within 10 units of time."
  $\mathsf{AG}\big(press \implies (\mathsf{AF}_{\leq 10}\, coffee)\big)$ ($\surd$)

- "It must never happen that the button can be pressed twice within 1 unit of time."

TCTL: Examples (coffee machine)



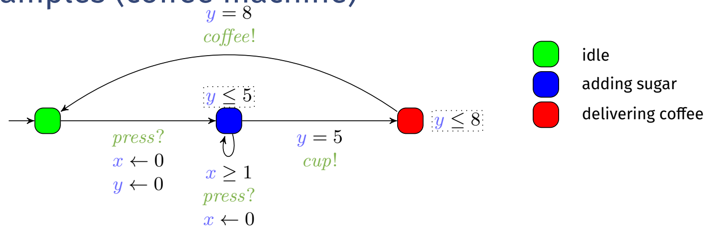
- "Whenever the button is pressed, a coffee is necessarily eventually delivered within 10 units of time."

  $\mathsf{AG}\big(press \implies (\mathsf{AF}_{\leq 10}\, coffee)\big)$ ($\surd$)

- "It must never happen that the button can be pressed twice within 1 unit of time."

  $\mathsf{AG}\big(press \implies (\mathsf{AG}_{\leq 1} \neg press)\big)$

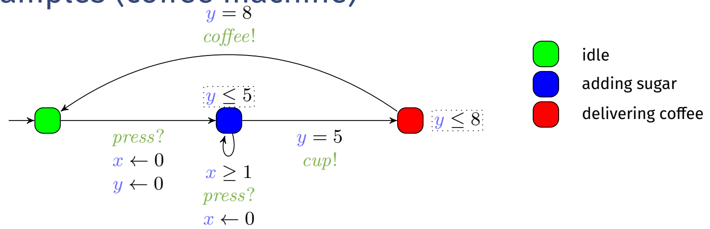
TCTL: Examples (coffee machine)



- "Whenever the button is pressed, a coffee is necessarily eventually delivered within 10 units of time."
  $\mathsf{AG}\big(\mathit{press} \implies (\mathsf{AF}_{\leq 10}\,\mathit{coffee})\big)$ ($\checkmark$)

- "It must never happen that the button can be pressed twice within 1 unit of time."
  $\mathsf{AG}\big(\mathit{press} \implies (\mathsf{AG}_{\leq 1}\neg\mathit{press})\big)$ ($\times$)

TCTL: Examples (coffee machine)

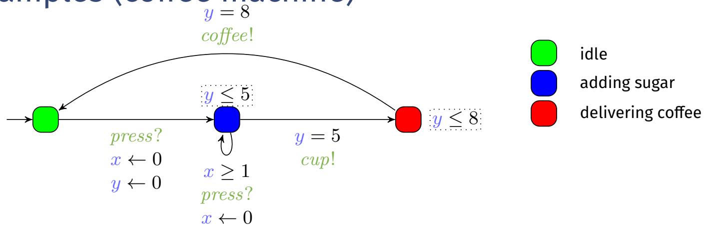


- "Whenever the button is pressed, a coffee is necessarily eventually delivered within 10 units of time."
  $\mathsf{AG}\big(press \implies (\mathsf{AF}_{\leq 10}\, coffee)\big)$ ($\surd$)

- "It must never happen that the button can be pressed twice within 1 unit of time."
  $\mathsf{AG}\big(press \implies (\mathsf{AG}_{\leq 1} \neg press)\big)$ ($\times$)

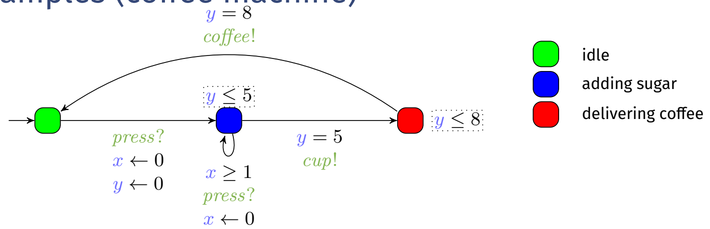
- "It must never happen that the button can be pressed twice within a time strictly less than 1 unit of time."

TCTL: Examples (coffee machine)



- "Whenever the button is pressed, a coffee is necessarily eventually delivered within 10 units of time."
  $\mathsf{AG}\big(press \implies (\mathsf{AF}_{\leq 10}\, coffee)\big)$ ($\surd$)
- "It must never happen that the button can be pressed twice within 1 unit of time."
  $\mathsf{AG}(press \implies (\mathsf{AG}_{\leq 1} \neg press))$ ($\times$)
- "It must never happen that the button can be pressed twice within a time strictly less than 1 unit of time."
  $\mathsf{AG}(press \implies (\mathsf{AG}_{< 1} \neg press))$

TCTL: Examples (coffee machine)



- "Whenever the button is pressed, a coffee is necessarily eventually delivered within 10 units of time."
  $\mathsf{AG}\big(press \implies (\mathsf{AF}_{\leq 10}\, coffee)\big)$ ($\surd$)

- "It must never happen that the button can be pressed twice within 1 unit of time."
  $\mathsf{AG}(press \implies (\mathsf{AG}_{\leq 1} \neg press))$ ($\times$)

- "It must never happen that the button can be pressed twice within a time strictly less than 1 unit of time."
  $\mathsf{AG}(press \implies (\mathsf{AG}_{< 1} \neg press))$ ($\surd$)

(NB: we use here actions instead of atomic propositions in locations)

TCTL model checking

Lemma (region equivalence)

Let ℓ be a location and φ be a TCTL formula. For any two valuations w and w' that belong to the same region,

$$(\ell, w) \models \varphi \iff (\ell, w') \models \varphi$$

Theorem (decidability [ACD93])

TCTL model checking is PSPACE-complete.

. [ACD93] Rajeev Alur, Costas Courcoubetis, and David L. Dill. « Model-Checking in Dense Real-Time ». In: *Information and Computation* 104.1 (May 1993), pp. 2–34

Software supporting timed automata

Tools for modeling and verifying models specified using timed automata

- HyTech (also hybrid, parametric timed automata) [HHW97]
- Kronos [Yov97]
- TReX (also parametric timed automata) [ABS01]
- UPPAAL [LPY97]
- Roméo (parametric time Petri nets) [Lim+09]
- PAT (also other formalisms) [Sun+09]
- IMITATOR (also parametric timed automata) [And21]

- [HHW97] Thomas A. Henzinger, Pei-Hsin Ho, and Howard Wong-Toi. « HyTech: A Model Checker for Hybrid Systems ». In: *International Journal on Software Tools for Technology Transfer* 1.1-2 (1997), pp. 110–122
- [Yov97] Sergio Yovine. « KRONOS: A Verification Tool for Real-Time Systems ». In: *International Journal on Software Tools for Technology Transfer* 1.1-2 (1997), pp. 123–133
- [ABS01] Aurore Annichini, Ahmed Bouajjani, and Mihaela Sighireanu. « TReX: A Tool for Reachability Analysis of Complex Systems ». In: *CAV*. vol. 2102. Lecture Notes in Computer Science. Springer, 2001, pp. 368–372
- [LPY97] Kim Guldstrand Larsen, Paul Pettersson, and Wang Yi. « UPPAAL in a Nutshell ». In: *International Journal on Software Tools for Technology Transfer* 1.1-2 (1997), pp. 134–152
- [Lim+09] Didier Lime, Olivier H. Roux, Charlotte Seidner, and Louis-Marie Traonouez. « Romeo: A Parametric Model-Checker for Petri Nets with Stopwatches ». In: *TACAS*. vol. 5505. Lecture Notes in Computer Science. Springer, Mar. 2009, pp. 54–57
- [Sun+09] Jun Sun, Yang Liu, Jin Song Dong, and Jun Pang. « PAT: Towards Flexible Verification under Fairness ». In: *CAV*. vol. 5643. Lecture Notes in Computer Science. Springer, 2009, pp. 709–714. ISBN: 978-3-642-02657-7
- [And21] Étienne André. « IMITATOR 3: Synthesis of timing parameters beyond decidability ». In: *CAV*. vol. 12759. Lecture Notes in Computer Science. Springer, 2021, pp. 1–14

Bibliography

References I

[ABL98] Luca Aceto, Augusto Burgueño, and Kim Guldstrand Larsen. « Model Checking via Reachability Testing for Timed Automata ». In: *TACAS* (Mar. 28–Apr. 4, 1998). Ed. by Bernhard Steffen. Vol. 1384. Lecture Notes in Computer Science. Lisbon, Portugal: Springer, 1998, pp. 263–280. ISBN: 3-540-64356-7. DOI: 10.1007/BFb0054177.

[ABS01] Aurore Annichini, Ahmed Bouajjani, and Mihaela Sighireanu. « TReX: A Tool for Reachability Analysis of Complex Systems ». In: *CAV* (July 18–22, 2001). Ed. by Gérard Berry, Hubert Comon, and Alain Finkel. Vol. 2102. Lecture Notes in Computer Science. Paris, France: Springer, 2001, pp. 368–372. DOI: 10.1007/3-540-44585-4_34.

[ACD93] Rajeev Alur, Costas Courcoubetis, and David L. Dill. « Model-Checking in Dense Real-Time ». In: *Information and Computation* 104.1 (May 1993), pp. 2–34. DOI: 10.1006/inco.1993.1024.

[Ace+03] Luca Aceto, Patricia Bouyer, Augusto Burgueño, and Kim Guldstrand Larsen. « The power of reachability testing for timed automata ». In: *Theoretical Computer Science* 300.1-3 (2003), pp. 411–475. DOI: 10.1016/S0304-3975(02)00334-1.

[AD94] Rajeev Alur and David L. Dill. « A theory of timed automata ». In: *Theoretical Computer Science* 126.2 (Apr. 1994), pp. 183–235. DOI: 10.1016/0304-3975(94)90010-8.

[AFH96] Rajeev Alur, Tomás Feder, and Thomas A. Henzinger. « The Benefits of Relaxing Punctuality ». In: *Journal of the ACM* 43.1 (1996), pp. 116–146. DOI: 10.1145/227595.227602.

[AH93] Rajeev Alur and Thomas A. Henzinger. « Real-Time Logics: Complexity and Expressiveness ». In: *Information and Computation* 104.1 (1993), pp. 35–77. DOI: 10.1006/INCO.1993.1025.

References II

[And21] Étienne André. « IMITATOR 3: Synthesis of timing parameters beyond decidability ». In: *CAV* (July 18–23, 2021). Ed. by Rustan Leino and Alexandra Silva. Vol. 12759. Lecture Notes in Computer Science. virtual: Springer, 2021, pp. 1–14. DOI: 10.1007/978-3-030-81685-8_26.

[Beh+06] Gerd Behrmann, Patricia Bouyer, Kim Guldstrand Larsen, and Radek Pelánek. « Lower and upper bounds in zone-based abstractions of timed automata ». In: *International Journal on Software Tools for Technology Transfer* 8.3 (2006), pp. 204–215. DOI: 10.1007/s10009-005-0190-0.

[Bou+22] Patricia Bouyer, Paul Gastin, Frédéric Herbreteau, Ocan Sankur, and B. Srivathsan. « Zone-Based Verification of Timed Automata: Extrapolations, Simulations and What Next? » In: *FORMATS* (Sept. 13–15, 2022). Ed. by Sergiy Bogomolov and David Parker. Vol. 13465. Lecture Notes in Computer Science. Warsaw, Poland: Springer, 2022, pp. 16–42. DOI: 10.1007/978-3-031-15839-1_2.

[Bri+17] Thomas Brihaye, Gilles Geeraerts, Hsi-Ming Ho, and Benjamin Monmege. « MightyL: A Compositional Translation from MITL to Timed Automata ». In: *CAV, Part I* (July 24–28, 2017). Ed. by Rupak Majumdar and Viktor Kuncak. Vol. 10426. Lecture Notes in Computer Science. Heidelberg, Germany: Springer, 2017, pp. 421–440. DOI: 10.1007/978-3-319-63387-9_21.

[Bri+18] Thomas Brihaye, Gilles Geeraerts, Hsi-Ming Ho, Arthur Milchior, and Benjamin Monmege. « Efficient Algorithms and Tools for MITL Model-Checking and Synthesis ». In: *ICECCS*. Ed. by Anthony Widjaja Lin and Jun Sun. Melbourne, Australia: IEEE Computer Society, 2018, pp. 180–184. DOI: 10.1109/ICECCS2018.2018.00027.

References III

[BY03] Johan Bengtsson and Wang Yi. « Timed Automata: Semantics, Algorithms and Tools ». In: *Lectures on Concurrency and Petri Nets, Advances in Petri Nets* (Sept. 2003). Ed. by Jörg Desel, Wolfgang Reisig, and Grzegorz Rozenberg. Vol. 3098. Lecture Notes in Computer Science. Eichstätt, Germany: Springer, 2003, pp. 87–124. DOI: 10.1007/978-3-540-27755-2_3.

[Fin06] Olivier Finkel. « Undecidable Problems About Timed Automata ». In: *FORMATS* (Sept. 25–27, 2006). Ed. by Eugene Asarin and Patricia Bouyer. Vol. 4202. Lecture Notes in Computer Science. Paris, France: Springer, 2006, pp. 187–199. DOI: 10.1007/11867340_14.

[HHW97] Thomas A. Henzinger, Pei-Hsin Ho, and Howard Wong-Toi. « HyTech: A Model Checker for Hybrid Systems ». In: *International Journal on Software Tools for Technology Transfer* 1.1-2 (1997), pp. 110–122. DOI: 10.1007/s100090050008.

[HSW16] Frédéric Herbreteau, B. Srivathsan, and Igor Walukiewicz. « Better abstractions for timed automata ». In: *Information and Computation* 251 (2016), pp. 67–90. DOI: 10.1016/j.ic.2016.07.004.

[Lim+09] Didier Lime, Olivier H. Roux, Charlotte Seidner, and Louis-Marie Traonouez. « Romeo: A Parametric Model-Checker for Petri Nets with Stopwatches ». In: *TACAS* (Mar. 22–29, 2009). Ed. by Stefan Kowalewski and Anna Philippou. Vol. 5505. Lecture Notes in Computer Science. York, United Kingdom: Springer, Mar. 2009, pp. 54–57. DOI: 10.1007/978-3-642-00768-2_6.

[LPY97] Kim Guldstrand Larsen, Paul Pettersson, and Wang Yi. « UPPAAL in a Nutshell ». In: *International Journal on Software Tools for Technology Transfer* 1.1-2 (1997), pp. 134–152. DOI: 10.1007/s100090050010.

References IV

[MNP06] Oded Maler, Dejan Ničković, and Amir Pnueli. « From MITL to Timed Automata ». In: *FORMATS* (Sept. 25–27, 2006). Ed. by Eugene Asarin and Patricia Bouyer. Vol. 4202. Lecture Notes in Computer Science. Paris, France: Springer, 2006, pp. 274–289. DOI: 10.1007/11867340_20.

[Sun+09] Jun Sun, Yang Liu, Jin Song Dong, and Jun Pang. « PAT: Towards Flexible Verification under Fairness ». In: *CAV* (June 26–July 2, 2009). Ed. by Ahmed Bouajjani and Oded Maler. Vol. 5643. Lecture Notes in Computer Science. Grenoble, France: Springer, 2009, pp. 709–714. ISBN: 978-3-642-02657-7. DOI: 10.1007/978-3-642-02658-4_59.

[Tri06] Stavros Tripakis. « Folk theorems on the determinization and minimization of timed automata ». In: *Information Processing Letters* 99.6 (2006), pp. 222–226. DOI: 10.1016/J.IPL.2006.04.015.

[Yov97] Sergio Yovine. « KRONOS: A Verification Tool for Real-Time Systems ». In: *International Journal on Software Tools for Technology Transfer* 1.1-2 (1997), pp. 123–133. DOI: 10.1007/s100090050009.

License of this document

Authors: **Étienne André**

(LaTeX source available to academic teachers upon request)

Version: July 12, 2024