M2 P2S

2024-2025

Formal verification
Part 2: Timed automata

Étienne André

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

Version slides with holes: October 9, 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

Formalisms

A number of formalisms were proposed to model and verify timed systems

- time(d) Petri nets [Mer74]
- timed automata [AD94]
- timed process algebras [Sun+09b]
- etc.

- [Mer74] Philip Meir Merlin. « A study of the recoverability of computing systems. ». PhD thesis. University of California, Irvine, CA, USA, 1974
- [AD94] Rajeev Alur and David L. Dill. « A theory of timed automata ». In: *Theoretical Computer Science* 126.2 (Apr. 1994), pp. 183–235
- [Sun+09b] Jun Sun, Yang Liu, Jin Song Dong, and Xian Zhang. « Verifying Stateful Timed CSP Using Implicit Clocks and Zone Abstraction ». In: *ICFEM*. vol. 5885. Lecture Notes in Computer Science. Springer, 2009, pp. 581–600
- [Bér+05] Béatrice Bérard, Franck Cassez, Serge Haddad, Didier Lime, and Olivier H. Roux. « Comparison of the Expressiveness of Timed Automata and Time Petri Nets ». In: *FORMATS*. vol. 3829. Lecture Notes in Computer Science. Springer, 2005, pp. 211–225
- [Srb08] Jiří Srba. « Comparing the Expressiveness of Timed Automata and Timed Extensions of Petri Nets ». In: *FORMATS*. vol. 5215. Lecture Notes in Computer Science. Springer, 2008, pp. 15–32
- [Bér+13] Béatrice Bérard, Franck Cassez, Serge Haddad, Didier Lime, and Olivier H. Roux. « The expressive power of time Petri nets ». In: *Theoretical Computer Science* 474 (2013), pp. 1–20

Formalisms

A number of formalisms were proposed to model and verify timed systems

- time(d) Petri nets [Mer74]
- timed automata [AD94]
- timed process algebras [Sun+09b]
- etc.

We use here timed automata

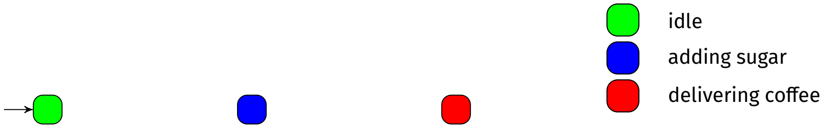
See [Bér+05] [Srb08] [Bér+13] for a comparison between timed Petri nets and timed automata

- [Mer74] Philip Meir Merlin. « A study of the recoverability of computing systems. ». PhD thesis. University of California, Irvine, CA, USA, 1974
- [AD94] Rajeev Alur and David L. Dill. « A theory of timed automata ». In: *Theoretical Computer Science* 126.2 (Apr. 1994), pp. 183–235
- [Sun+09b] Jun Sun, Yang Liu, Jin Song Dong, and Xian Zhang. « Verifying Stateful Timed CSP Using Implicit Clocks and Zone Abstraction ». In: *ICFEM*. vol. 5885. Lecture Notes in Computer Science. Springer, 2009, pp. 581–600
- [Bér+05] Béatrice Bérard, Franck Cassez, Serge Haddad, Didier Lime, and Olivier H. Roux. « Comparison of the Expressiveness of Timed Automata and Time Petri Nets ». In: *FORMATS*. vol. 3829. Lecture Notes in Computer Science. Springer, 2005, pp. 211–225
- [Srb08] Jiří Srba. « Comparing the Expressiveness of Timed Automata and Timed Extensions of Petri Nets ». In: *FORMATS*. vol. 5215. Lecture Notes in Computer Science. Springer, 2008, pp. 15–32
- [Bér+13] Béatrice Bérard, Franck Cassez, Serge Haddad, Didier Lime, and Olivier H. Roux. « The expressive power of time Petri nets ». In: *Theoretical Computer Science* 474 (2013), pp. 1–20

Outline

Timed automaton (TA)

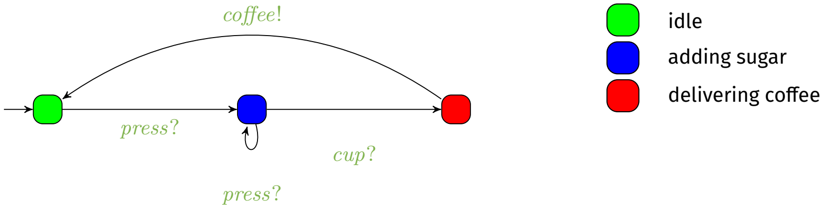
- Finite-state automaton (sets of locations)



• [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)

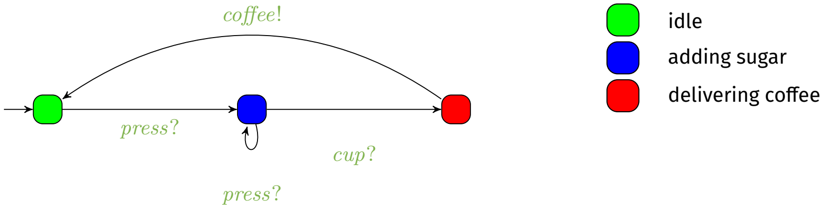
- Finite-state automaton (sets of locations and actions)



. [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

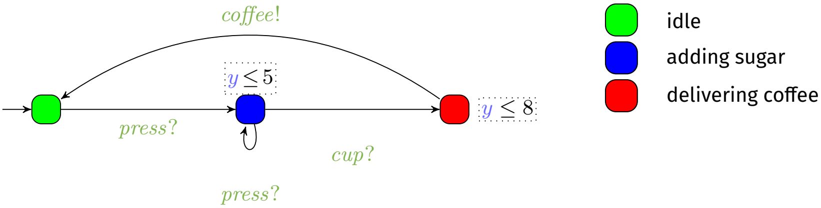


. [AD94] Rajeev Alur and David L. Dill. « A theory of timed automata ». In: *Theoretical Computer Science* 126.2 (Apr. 1994), pp. 183–235

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

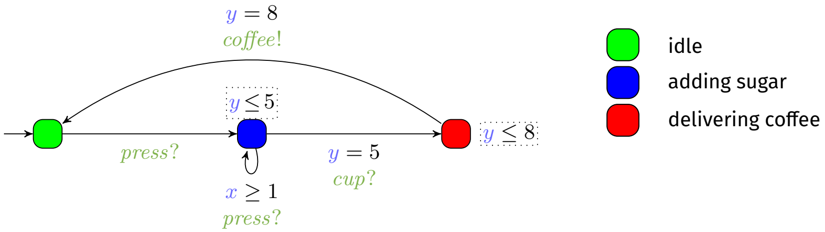


. [AD94] Rajeev Alur and David L. Dill. « A theory of timed automata ». In: *Theoretical Computer Science* 126.2 (Apr. 1994), pp. 183–235

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

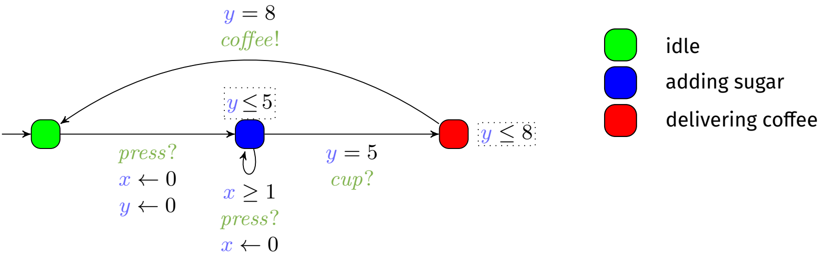


- [AD94] Rajeev Alur and David L. Dill. « A theory of timed automata ». In: *Theoretical Computer Science* 126.2 (Apr. 1994), pp. 183–235

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 finite set of clocks,
- I is the invariant, assigning to every $\ell \in L$ a clock constraint $I(\ell)$, and
- E is a 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 transition, and g (the transition guard) is clock constraint.

Clock constraints

Definition (clock constraint)

A clock constraint is a conjunction of atomic constraints

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

Clock constraints

Definition (clock constraint)

A clock constraint is a conjunction of atomic constraints

What is an atomic constraint?

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

Clock constraints

Definition (clock constraint)

A clock constraint is a conjunction of atomic constraints

What is an atomic constraint?

Various definitions in the literature:

- Originally [AD94]: $x \in [c_1, c_2]$ with $c_1 \in \mathbb{N}$ and $c_2 \in \mathbb{N} \cup \{\infty\}$
- Comparing clock values (diagonal constraints) $x_1 - x_2 \sim c$
  - $\sim \in \{<, \leq, =, \geq, >\}$

For now, we assume the following syntax:

- $x \sim c$, with $x \in X$ and $c \in \mathbb{N}$

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

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 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 1: questions

1. Are all the locations reachable?

Exercise 1: questions

1. Are all the locations reachable?
2. Can all transitions be taken?

Exercise 1: questions

1. Are all the locations reachable?
2. Can all transitions be taken?
3. What is the minimal time before taking the transition from ℓ_4 to ℓ_3?

Exercise 1: questions

1. Are all the locations reachable?
2. Can all transitions be taken?
3. What is the minimal time before taking the transition from ℓ_4 to ℓ_3?

Exercise 2

Give the formal TA corresponding to the timed coffee machine.

Exercise 2

Give the formal TA corresponding to the timed coffee machine.

Parallel composition of timed automata (1/2)

Just as finite-state automata, timed automata can be composed through parallel composition using synchronization actions

$\mathcal{A}_1 = (L_1, \Sigma_1, (\ell_0)_1, (F)_1, X_1, I_1, E_1)$
$\mathcal{A}_2 = (L_2, \Sigma_2, (\ell_0)_2, (F)_2, X_2, I_2, E_2)$

Then we define $\mathcal{A}_1 \parallel \mathcal{A}_2$ as

Parallel composition of timed automata (1/2)

Just as finite-state automata, timed automata can be composed through parallel composition using synchronization actions

$\mathcal{A}_1 = (L_1, \Sigma_1, (\ell_0)_1, (F)_1, X_1, I_1, E_1)$
$\mathcal{A}_2 = (L_2, \Sigma_2, (\ell_0)_2, (F)_2, X_2, I_2, E_2)$

Then we define $\mathcal{A}_1 \parallel \mathcal{A}_2$ as

Parallel composition of timed automata (1/2)

Just as finite-state automata, timed automata can be composed through parallel composition using synchronization actions

$\mathcal{A}_1 = (L_1, \Sigma_1, (\ell_0)_1, (F)_1, X_1, I_1, E_1)$
$\mathcal{A}_2 = (L_2, \Sigma_2, (\ell_0)_2, (F)_2, X_2, I_2, E_2)$

Then we define $\mathcal{A}_1 \parallel \mathcal{A}_2$ as

Parallel composition of timed automata (1/2)

Just as finite-state automata, timed automata can be composed through parallel composition using synchronization actions

$\mathcal{A}_1 = (L_1, \Sigma_1, (\ell_0)_1, (F)_1, X_1, I_1, E_1)$
$\mathcal{A}_2 = (L_2, \Sigma_2, (\ell_0)_2, (F)_2, X_2, I_2, E_2)$

Then we define $\mathcal{A}_1 \parallel \mathcal{A}_2$ as

Parallel composition of timed automata (1/2)

Just as finite-state automata, timed automata can be composed through parallel composition using synchronization actions

$\mathcal{A}_1 = (L_1, \Sigma_1, (\ell_0)_1, (F)_1, X_1, I_1, E_1)$
$\mathcal{A}_2 = (L_2, \Sigma_2, (\ell_0)_2, (F)_2, X_2, I_2, E_2)$

Then we define $\mathcal{A}_1 \parallel \mathcal{A}_2$ as

Parallel composition of timed automata (1/2)

Just as finite-state automata, timed automata can be composed through parallel composition using synchronization actions

$\mathcal{A}_1 = (L_1, \Sigma_1, (\ell_0)_1, (F)_1, X_1, I_1, E_1)$
$\mathcal{A}_2 = (L_2, \Sigma_2, (\ell_0)_2, (F)_2, X_2, I_2, E_2)$

Then we define $\mathcal{A}_1 \parallel \mathcal{A}_2$ as

Parallel composition of timed automata (1/2)

Just as finite-state automata, timed automata can be composed through parallel composition using synchronization actions

$\mathcal{A}_1 = (L_1, \Sigma_1, (\ell_0)_1, (F)_1, X_1, I_1, E_1)$
$\mathcal{A}_2 = (L_2, \Sigma_2, (\ell_0)_2, (F)_2, X_2, I_2, E_2)$

Then we define $\mathcal{A}_1 \parallel \mathcal{A}_2$ as

Parallel composition of timed automata (1/2)

Just as finite-state automata, timed automata can be composed through parallel composition using synchronization actions

$\mathcal{A}_1 = (L_1, \Sigma_1, (\ell_0)_1, (F)_1, X_1, I_1, E_1)$
$\mathcal{A}_2 = (L_2, \Sigma_2, (\ell_0)_2, (F)_2, X_2, I_2, E_2)$

Then we define $\mathcal{A}_1 \parallel \mathcal{A}_2$ as

Parallel composition of timed automata (2/2)

Parallel composition of timed automata (2/2)

Parallel composition of timed automata (2/2)

Parallel composition of timed automata (2/2)

Parallel composition of timed automata (2/2)

Parallel composition of timed automata (2/2)

Outline

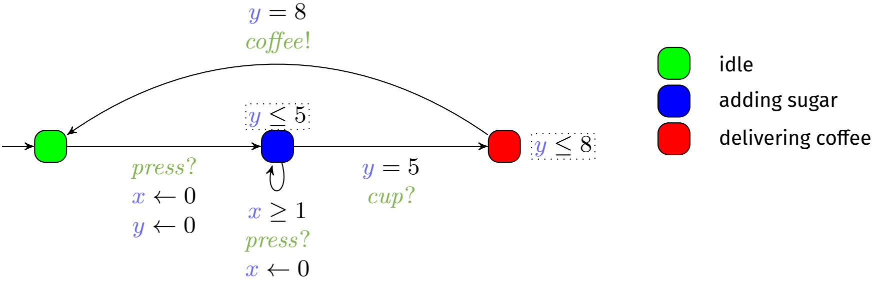
Concrete semantics of timed automata

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

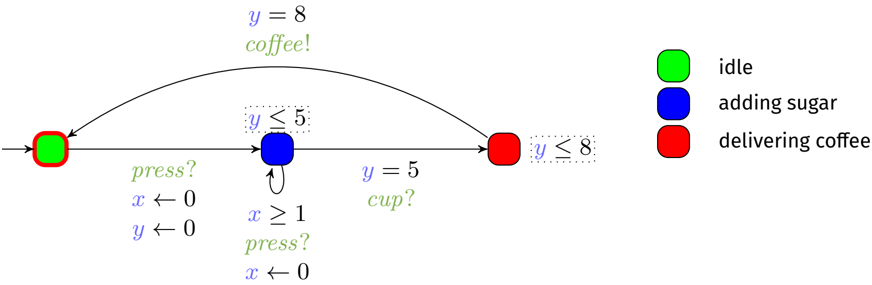
  Example: ( , $\binom{x=1.2}{y=3.7}$)

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

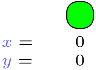
Examples of concrete runs



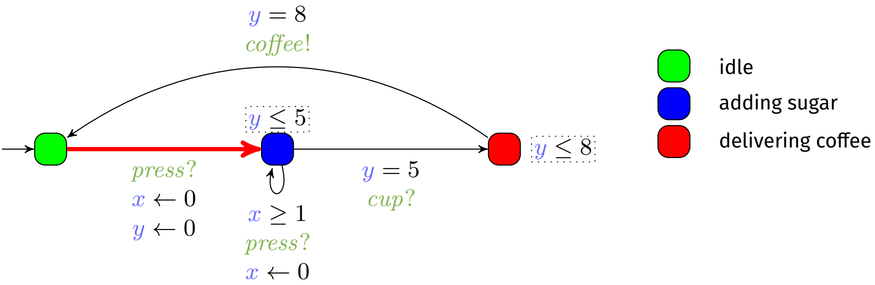
Examples of concrete runs



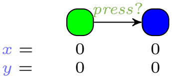
- Example of concrete run for the coffee machine
  - Coffee with 2 doses of sugar



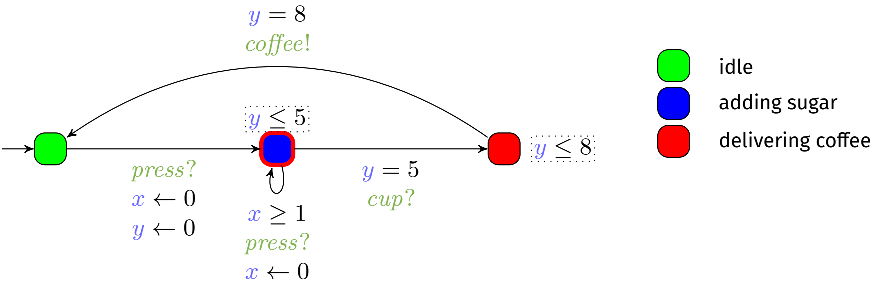
Examples of concrete runs



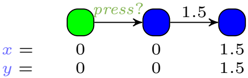
- Example of concrete run for the coffee machine
  - Coffee with 2 doses of sugar



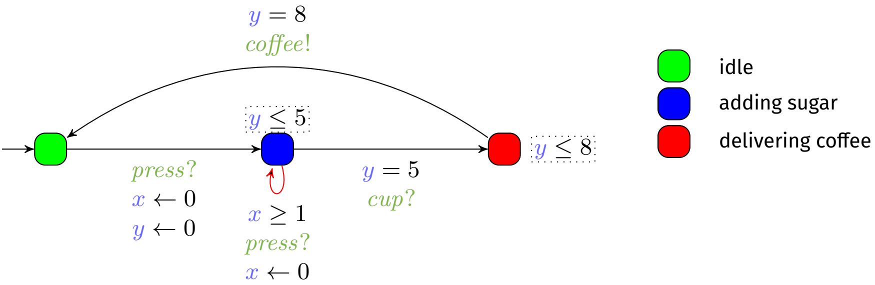
Examples of concrete runs



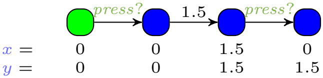
- Example of concrete run for the coffee machine
  - Coffee with 2 doses of sugar



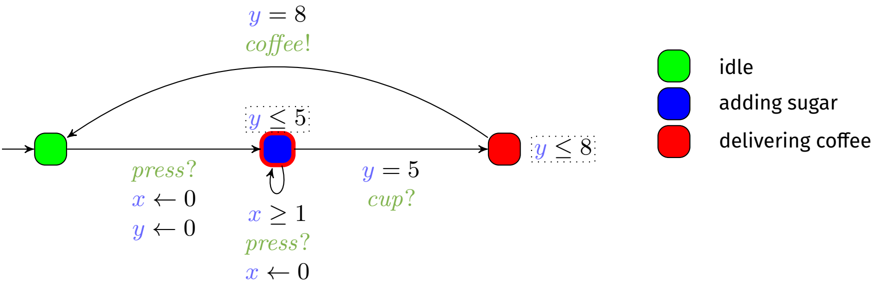
Examples of concrete runs



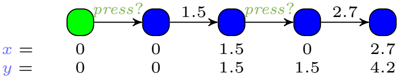
- Example of concrete run for the coffee machine
  - Coffee with 2 doses of sugar



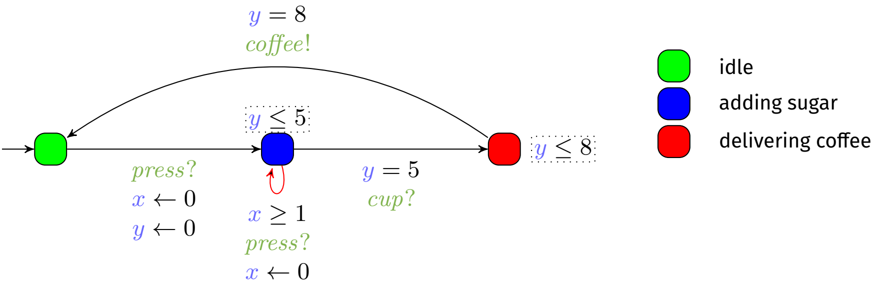
Examples of concrete runs



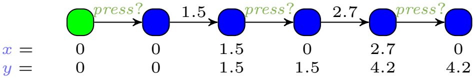
- Example of concrete run for the coffee machine
  - Coffee with 2 doses of sugar



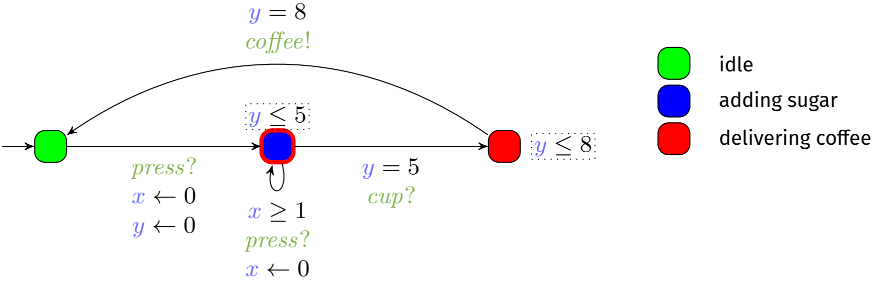
Examples of concrete runs



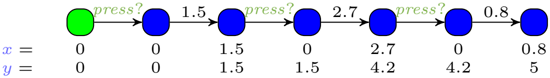
- Example of concrete run for the coffee machine
  - Coffee with 2 doses of sugar



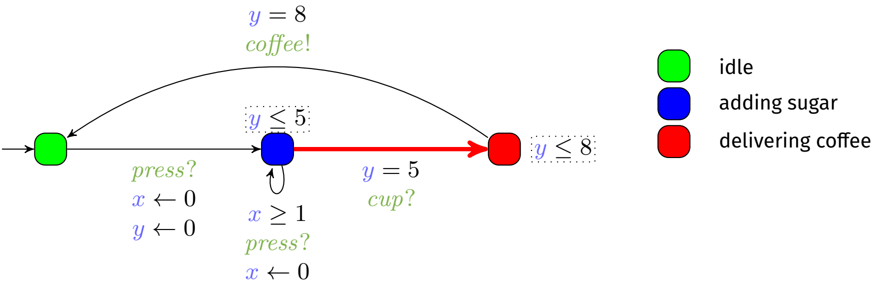
Examples of concrete runs



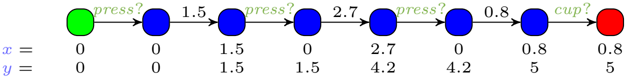
- Example of concrete run for the coffee machine
  - Coffee with 2 doses of sugar



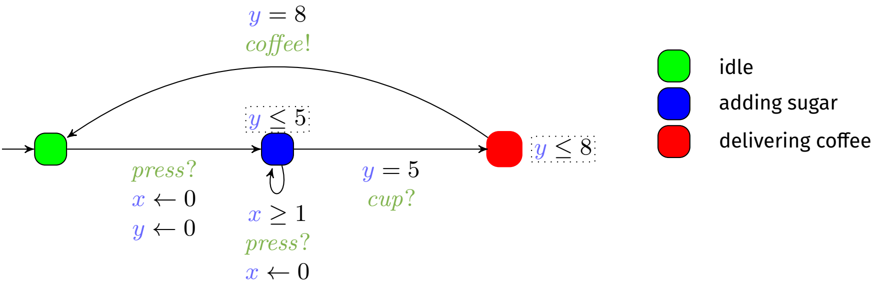
Examples of concrete runs



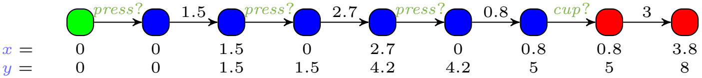
- Example of concrete run for the coffee machine
  - Coffee with 2 doses of sugar



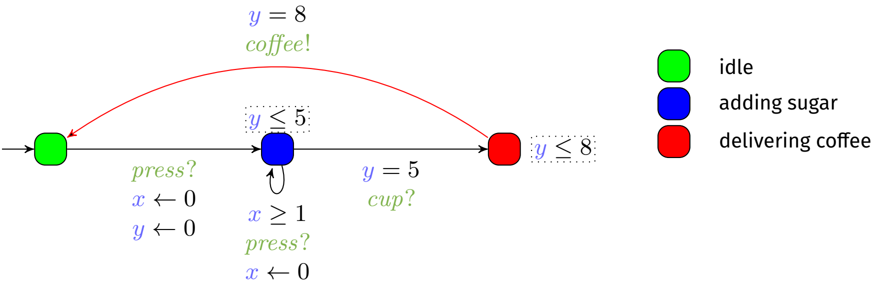
Examples of concrete runs



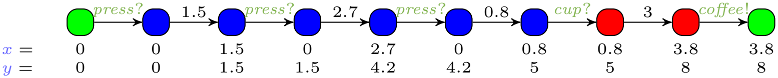
- Example of concrete run for the coffee machine
  - Coffee with 2 doses of sugar



Examples of concrete runs



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



Concrete semantics of timed automata: definition

Definition (Concrete 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. $\mathfrak{S} = \{(\ell, w) \in L \times \mathbb{R}_{\geq 0}^{|X|} \mid w \models I(\ell)\}$,
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 $\forall d' \in [0, d], (\ell, w + d') \in \mathfrak{S}$.

Notation:

$$[w]_R(x) = \left\{ \right.$$

Concrete semantics of timed automata: definition

Definition (Concrete 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 $\forall d' \in [0, d], (\ell, w + d') \in \mathfrak{S}$.

Notation:

$$[w]_R(x) = \begin{cases} & \text{if } x \in R \\ & \end{cases}$$

Concrete semantics of timed automata: definition

Definition (Concrete 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 $\forall d' \in [0, d], (\ell, w + d') \in \mathfrak{S}$.

Notation:

$$[w]_R(x) = \begin{cases} & \text{if } x \in R \\ & \text{otherwise} \end{cases}$$

Concrete semantics of timed automata: definition (cont.)

We write $(\ell, w) \xrightarrow{(d,e)} (\ell', w')$ or $\big((\ell, w), (d, e), (\ell', w')\big) \in \longrightarrow$ for a combination of a delay and discrete transitions if

$$\exists w'' : (\ell, w) \overset{d}{\mapsto} (\ell, w'') \overset{e}{\mapsto} (\ell', w')$$

Concrete semantics of timed automata: definition (cont.)

We write $(\ell, w) \xrightarrow{(d,e)} (\ell', w')$ or $\big((\ell, w), (d, e), (\ell', w')\big) \in \longrightarrow$ for a combination of a delay and discrete transitions if

$$\exists w'' : (\ell, w) \overset{d}{\mapsto} (\ell, w'') \overset{e}{\mapsto} (\ell', w')$$

Some remarks on the semantics of timed automata:

- Is $\mathfrak{T}_{\mathcal{A}}$ finite?

Concrete semantics of timed automata: definition (cont.)

We write $(\ell, w) \xrightarrow{(d,e)} (\ell', w')$ or $\big((\ell, w), (d, e), (\ell', w')\big) \in \longrightarrow$ for a combination of a delay and discrete transitions if

$$\exists w'' : (\ell, w) \overset{d}{\mapsto} (\ell, w'') \overset{e}{\mapsto} (\ell', w')$$

Some remarks on the semantics of timed automata:

- Is $\mathfrak{T}_{\mathcal{A}}$ finite?
- Is $\mathfrak{T}_{\mathcal{A}}$ finitely branching?

Concrete semantics of timed automata: definition (cont.)

We write $(\ell, w) \xrightarrow{(d,e)} (\ell', w')$ or $\big((\ell, w), (d, e), (\ell', w')\big) \in \longrightarrow$ for a combination of a delay and discrete transitions if

$$\exists w'' : (\ell, w) \overset{d}{\mapsto} (\ell, w'') \overset{e}{\mapsto} (\ell', w')$$

Some remarks on the semantics of timed automata:

- Is $\mathfrak{T}_{\mathcal{A}}$ finite?
- Is $\mathfrak{T}_{\mathcal{A}}$ finitely branching?

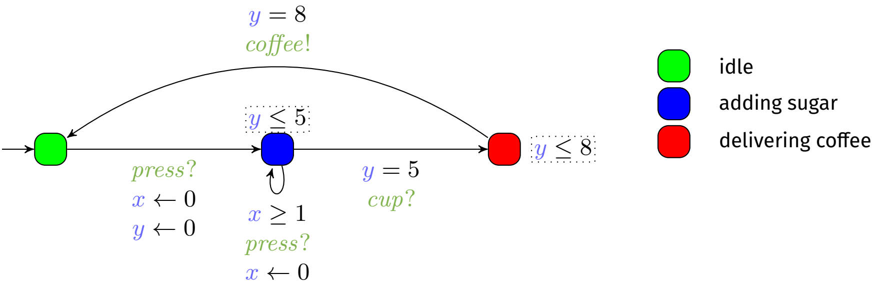
Timed words

Timed word: a sequence of pairs made of

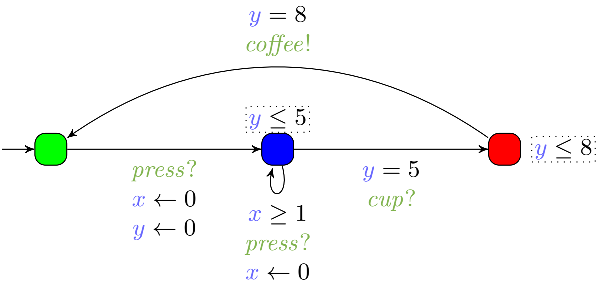
1. an action, and
2. an increasing timestamp in $\mathbb{R}_{\geq 0}$

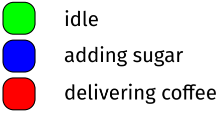
Given a run $(\ell_0, w_0), (d_0, e_0), (\ell_1, w_1), \cdots, (\ell_n, w_n)$ then it generates the timed word $(a_0, \sum_{i=0}^{0} d_i)(a_1, \sum_{i=0}^{1} d_i) \dots (a_n, \sum_{i=0}^{n} d_i)$

Examples of concrete runs

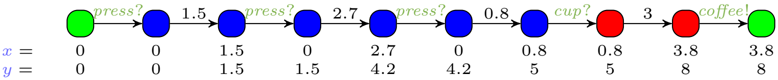


Examples of concrete runs



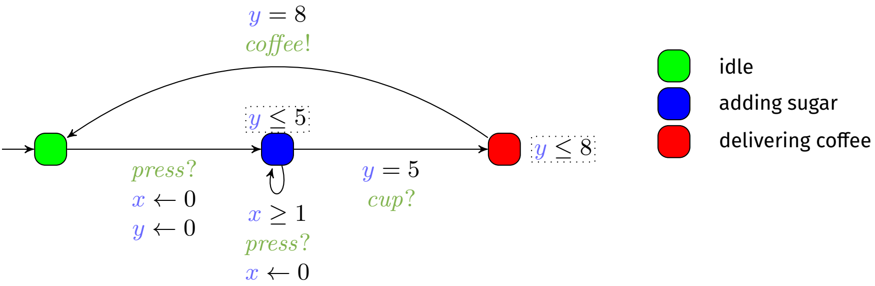


- Example of concrete run for the coffee machine
  - Coffee with 2 doses of sugar

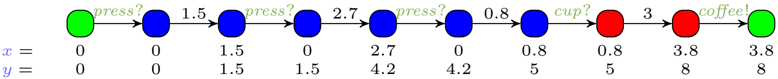


  - Associated timed word:

Examples of concrete runs



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



  - Associated timed word:

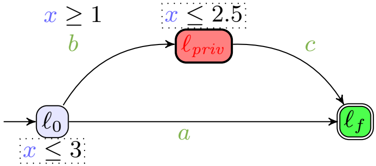
Timed language

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

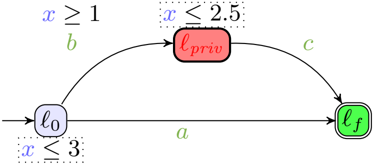
Definition (timed language of a TA)

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

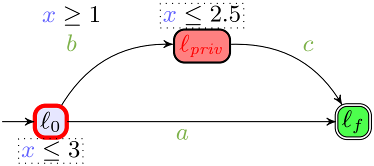


Examples of accepting runs

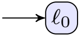


- Two examples of accepting runs

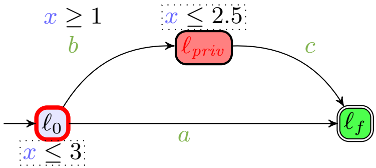
Examples of accepting runs



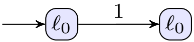
- Two examples of accepting runs



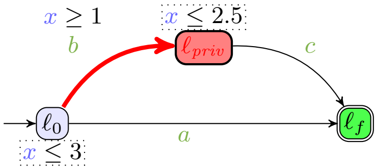
Examples of accepting runs



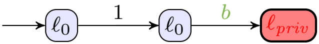
- Two examples of accepting runs



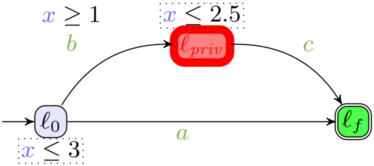
Examples of accepting runs



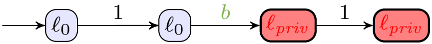
- Two examples of accepting runs



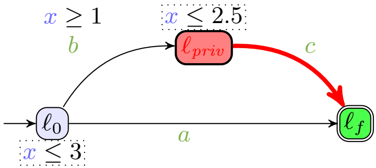
Examples of accepting runs



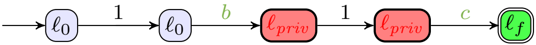
- Two examples of accepting runs



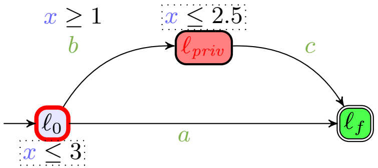
Examples of accepting runs



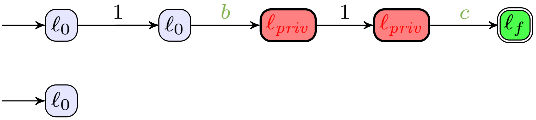
- Two examples of accepting runs



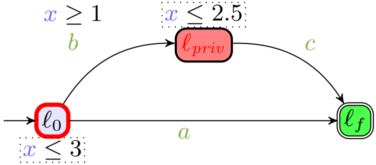
Examples of accepting runs



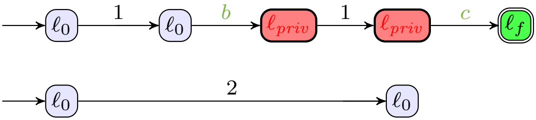
- Two examples of accepting runs



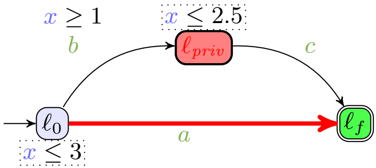
Examples of accepting runs



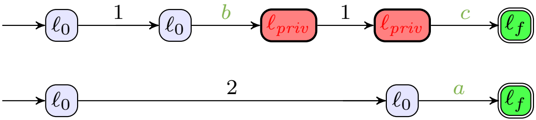
- Two examples of accepting runs



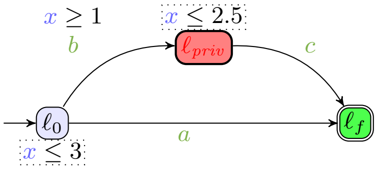
Examples of accepting runs



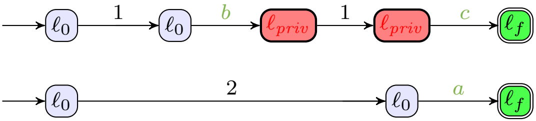
- Two examples of accepting runs



Examples of accepting runs

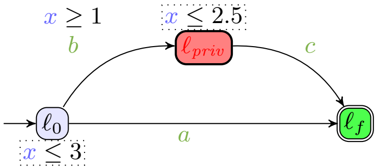


- Two examples of accepting runs

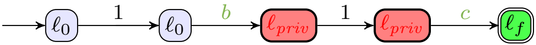


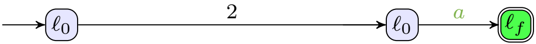
- Timed language of this TA $\mathcal{A}$:

Examples of accepting runs



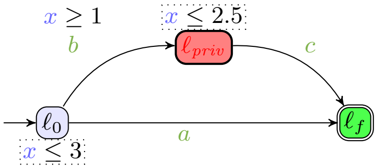
- Two examples of accepting runs



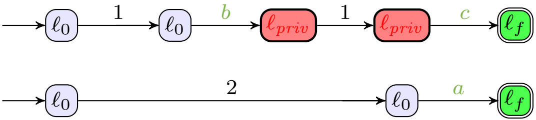


- Timed language of this TA $\mathcal{A}$:

Examples of accepting runs



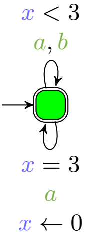
- Two examples of accepting runs



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

Timed language: Example 1

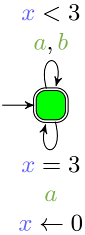
Give the timed language of the following automaton [AD94]



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

Timed language: Example 1

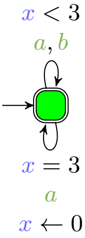
Give the timed language of the following automaton [AD94]



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

Timed language: Example 1

Give the timed language of the following automaton [AD94]



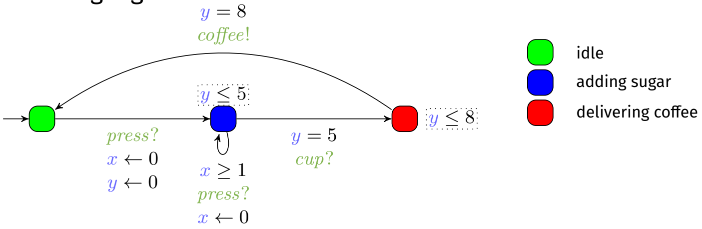
- [AD94] Rajeev Alur and David L. Dill. « A theory of timed automata ». In: *Theoretical Computer Science* 126.2 (Apr. 1994), pp. 183–235

Timed language: Example 2

Give the timed language of the following automaton

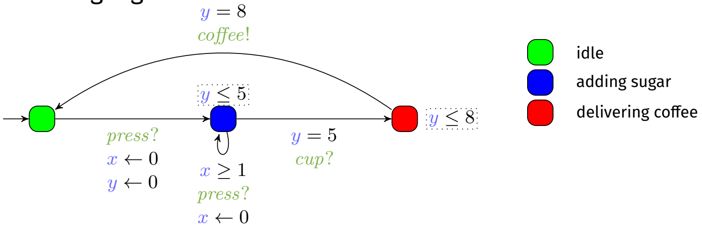
Timed language: Example 3

Give the timed language of the coffee machine



Timed language: Example 3

Give the timed language of the coffee machine



Accepting locations?

Timed automata may or may not be equipped with accepting locations

Often, timed automata with no accepting locations are called timed safety automata [Hen+94]

In that case the timed language can be defined as:

- All possible timed words read by the automaton
- All possible maximal timed words read by the automaton
  - Maximal: infinite or that cannot be extended
- All possible infinite timed words read by the automaton

- [Hen+94] Thomas A. Henzinger, Xavier Nicollin, Joseph Sifakis, and Sergio Yovine. « Symbolic Model Checking for Real-Time Systems ». In: *Information and Computation* 111.2 (1994), pp. 193–244
- [HKW95] Thomas A. Henzinger, Peter W. Kopke, and Howard Wong-Toi. « The Expressive Power of Clocks ». In: *ICALP*. vol. 944. Lecture Notes in Computer Science. Springer, 1995, pp. 417–428

Accepting locations?

Timed automata may or may not be equipped with accepting locations

Often, timed automata with no accepting locations are called timed safety automata [Hen+94]

In that case the timed language can be defined as:

- All possible timed words read by the automaton
- All possible maximal timed words read by the automaton
  - Maximal: infinite or that cannot be extended
- All possible infinite timed words read by the automaton

Theorem

The expressive power of timed safety automata is strictly less than timed automata with accepting locations *[HKW95]*

- [Hen+94] Thomas A. Henzinger, Xavier Nicollin, Joseph Sifakis, and Sergio Yovine. « Symbolic Model Checking for Real-Time Systems ». In: *Information and Computation* 111.2 (1994), pp. 193–244
- [HKW95] Thomas A. Henzinger, Peter W. Kopke, and Howard Wong-Toi. « The Expressive Power of Clocks ». In: *ICALP*. vol. 944. Lecture Notes in Computer Science. Springer, 1995, pp. 417–428

Deadlocks and timelocks

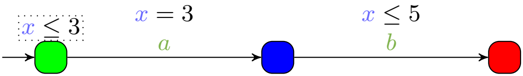
Timed automata can be subject to two annoying behaviors:

- Deadlock: similar to finite state automata
  - Can be a problem of

Deadlocks and timelocks

Timed automata can be subject to two annoying behaviors:

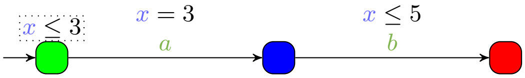
- Deadlock: similar to finite state automata
  - Can be a problem of



Deadlocks and timelocks

Timed automata can be subject to two annoying behaviors:

- Deadlock: similar to finite state automata
  - Can be a problem of

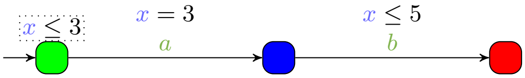


- Timelock: coming from the timed nature of TAs
  - Can

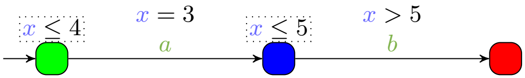
Deadlocks and timelocks

Timed automata can be subject to two annoying behaviors:

- Deadlock: similar to finite state automata
  - Can be a problem of



- Timelock: coming from the timed nature of TAs
  - Can



The Zeno problem (1/2)

Definition (Zeno run)

A run is Zeno if it contains an infinite number of actions in finite time.

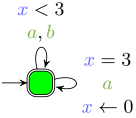
- [AD94] Rajeev Alur and David L. Dill. « A theory of timed automata ». In: *Theoretical Computer Science* 126.2 (Apr. 1994), pp. 183–235

The Zeno problem (1/2)

Definition (Zeno run)

A run is Zeno if it contains an infinite number of actions in finite time.

- Example of TA containing at least one Zeno run [AD94]



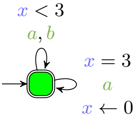
- [AD94] Rajeev Alur and David L. Dill. « A theory of timed automata ». In: *Theoretical Computer Science* 126.2 (Apr. 1994), pp. 183–235

The Zeno problem (1/2)

Definition (Zeno run)

A run is Zeno if it contains an infinite number of actions in finite time.

- Example of TA containing at least one Zeno run [AD94]



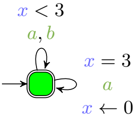
• [AD94] Rajeev Alur and David L. Dill. « A theory of timed automata ». In: *Theoretical Computer Science* 126.2 (Apr. 1994), pp. 183–235

The Zeno problem (1/2)

Definition (Zeno run)

A run is Zeno if it contains an infinite number of actions in finite time.

- Example of TA containing at least one Zeno run [AD94]



- Example of TA containing only non-Zeno runs

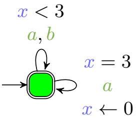
- [AD94] Rajeev Alur and David L. Dill. « A theory of timed automata ». In: *Theoretical Computer Science* 126.2 (Apr. 1994), pp. 183–235

The Zeno problem (1/2)

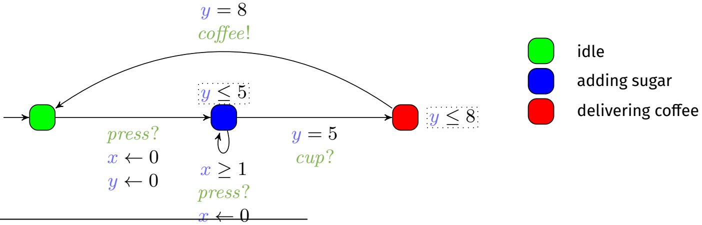
Definition (Zeno run)

A run is Zeno if it contains an infinite number of actions in finite time.

- Example of TA containing at least one Zeno run [AD94]



- Example of TA containing only non-Zeno runs
  - ☺ The coffee machine



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

The Zeno problem (2/2)

Problem (Zeno runs)

An infinite number of actions in finite time is impossible in practice

- *Processors have finite precision*

Zeno runs must be pruned when performing model checking

The Zeno problem: possible solutions

Some solutions:

- Transform the TA (with an additional clock) [Tri99] [TYB05] [BG06] [GB07]
- Transform the zone graph [HSW12]
- Consider a different but closely related formalism [Sun+13]
- Transform the TA on-the-fly [Wan+15]

- [Tri99] Stavros Tripakis. « Verifying Progress in Timed Systems ». In: *ARTS*. vol. 1601. Lecture Notes in Computer Science. Springer, 1999, pp. 299–314
- [TYB05] Stavros Tripakis, Sergio Yovine, and Ahmed Bouajjani. « Checking Timed Büchi Automata Emptiness Efficiently ». In: *Formal Methods in System Design* 26.3 (2005), pp. 267–292
- [BG06] Howard Bowman and Rodolfo Gómez. « How to stop time stopping ». In: *Formal Aspects of Computing* 18.4 (2006), pp. 459–493
- [GB07] Rodolfo Gómez and Howard Bowman. « Efficient Detection of Zeno Runs in Timed Automata ». In: *FORMATS*. vol. 4763. Lecture Notes in Computer Science. Springer, 2007, pp. 195–210
- [HSW12] Frédéric Herbreteau, B. Srivathsan, and Igor Walukiewicz. « Efficient emptiness check for timed Büchi automata ». In: *Formal Methods in System Design* 40.2 (2012), pp. 122–146
- [Sun+13] Jun Sun, Yang Liu, Jin Song Dong, Yan Liu, Ling Shi, and Étienne André. « Modeling and Verifying Hierarchical Real-time Systems using Stateful Timed CSP ». In: *ACM Transactions on Software Engineering and Methodology* 22.1 (Feb. 2013), pp. 3.1–3.29
- [Wan+15] Ting Wang, Jun Sun, Xinyu Wang, Yang Liu, Yuanjie Si, Jin Song Dong, Xiaohu Yang, and Xiaohong Li. « A Systematic Study on Explicit-State Non-Zenoness Checking for Timed Automata ». In: *IEEE Transactions on Software Engineering* 41.1 (2015), pp. 3–18

Outline

Example: Railroad gate controller [AHV93]

Design three timed automata in parallel:

1. The train: once it is approaching (action $approach$), it will come in (action in) after at least 5 time units, then go out (action out) and finally exit (action $exit$) after at most 6 time units
2. The gate: upon reception of a $lower$ signal, starts to lower; once it is down, and upon reception of a $raise$ signal, the gate raises again; the time to lower and to raise the gate is an interval $[1, 3]$
3. The controller: once a train approaches (action $approach$), it triggers the $lower$ signal within $[2, 3]$ time units; then, once the train exits (action $exit$), it triggers the $raise$ signal again within $[2, 4]$ time units

All TAs are cyclic, i. e., repeat the same behavior forever.

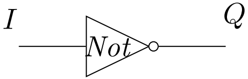
. [AHV93] Rajeev Alur, Thomas A. Henzinger, and Moshe Y. Vardi. « Parametric real-time reasoning ». In: *STOC*. ACM, 1993, pp. 592–601

Example: Railroad gate controller (gate)

Example: Railroad gate controller (controller)

Example: Railroad gate controller (train)

Example: A hardware gate

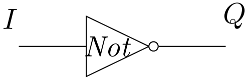


The output Q reacts to the change of the input I (actions $I^{\uparrow}$ and $I^{\downarrow}$) after a delay $[5, 9]$. Actions controlled by the gate are $Q^{\uparrow}$ and $Q^{\downarrow}$

[Che+09]

. [Che+09] Rémy Chevallier, Emmanuelle Encrenaz-Tiphène, Laurent Fribourg, and Weiwen Xu. « Timed Verification of the Generic Architecture of a Memory Circuit Using Parametric Timed Automata ». In: *Formal Methods in System Design* 34.1 (Feb. 2009), pp. 59–81

Example: A hardware gate

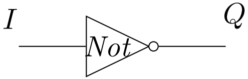


The output Q reacts to the change of the input I (actions $I^{\uparrow}$ and $I^{\downarrow}$) after a delay $[5, 9]$. Actions controlled by the gate are $Q^{\uparrow}$ and $Q^{\downarrow}$

[Che+09]

[Che+09] Rémy Chevallier, Emmanuelle Encrenaz-Tiphène, Laurent Fribourg, and Weiwen Xu. « Timed Verification of the Generic Architecture of a Memory Circuit Using Parametric Timed Automata ». In: *Formal Methods in System Design* 34.1 (Feb. 2009), pp. 59–81

Example: A hardware gate

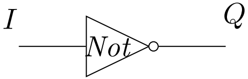


The output Q reacts to the change of the input I (actions $I^{\uparrow}$ and $I^{\downarrow}$) after a delay $[5, 9]$. Actions controlled by the gate are $Q^{\uparrow}$ and $Q^{\downarrow}$

[Che+09]

. [Che+09] Rémy Chevallier, Emmanuelle Encrenaz-Tiphène, Laurent Fribourg, and Weiwen Xu. « Timed Verification of the Generic Architecture of a Memory Circuit Using Parametric Timed Automata ». In: *Formal Methods in System Design* 34.1 (Feb. 2009), pp. 59–81

Example: A hardware gate



The output Q reacts to the change of the input I (actions $I^{\uparrow}$ and $I^{\downarrow}$) after a delay $[5, 9]$. Actions controlled by the gate are $Q^{\uparrow}$ and $Q^{\downarrow}$

[Che+09]

. [Che+09] Rémy Chevallier, Emmanuelle Encrenaz-Tiphène, Laurent Fribourg, and Weiwen Xu. « Timed Verification of the Generic Architecture of a Memory Circuit Using Parametric Timed Automata ». In: *Formal Methods in System Design* 34.1 (Feb. 2009), pp. 59–81

Example: A nuclear power plant

Design a TA modeling a nuclear power plant:

- At first, the plant is in normal mode.
- Suddenly, it may start to heat (action $startHeating$).
- At that point, a timer is set; after p_2 time units, the timer will trigger an alarm (action $alarm$).
- Then, p_3 time units later, a watering system (action $watering$) starts.
- This watering system lasts for at most p_4 time units, after which the plant is cool again (action $cool$) and goes back to the normal mode.
- However, p_1 time units after the plant starts to heat, the plant may explode at any time (action $boom$)—unless of course it is cool again.

Example: A nuclear power plant (solution)

Example: A real-time system

Design a (network of) timed automata modeling the following components:

1. a periodic task T_1 of period 5 with offset 2, best and worst case execution times in $[3, 4]$
2. a sporadic task T_2 of minimum interarrival time 20, best and worst case execution times in $[1, 2]$
3. a non-preemptive scheduler with fixed priority

Example: A real-time system (solution)

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).

“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

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

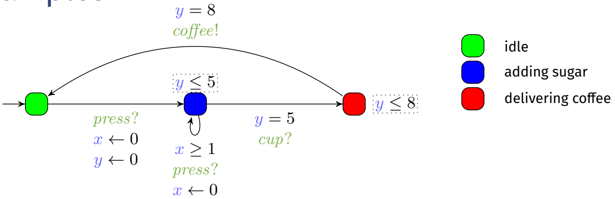
Definition (Region equivalence [AD94])

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: $\mathsf{fr}(w(x_i)) \leq \mathsf{fr}(w(x_j))$ iff $\mathsf{fr}(w'(x_i)) \leq \mathsf{fr}(w'(x_j))$; and
3. if $w(x_i) \leq c_i$, then: $\mathsf{fr}(w(x_i)) = 0$ iff $\mathsf{fr}(w'(x_i)) = 0$.

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

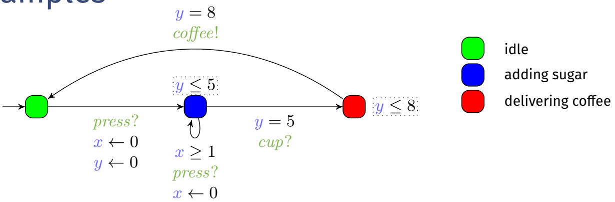
Regions: Examples



- 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



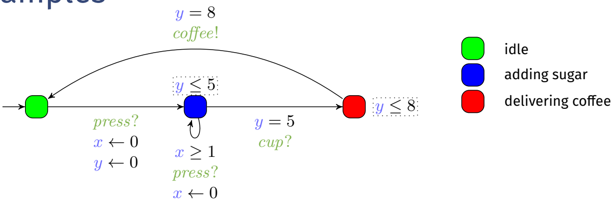
- Let w_1 be such that $w_1(x) = 0.5$ and $w_1(y) = 2.7$
- Let w_2 be such that $w_2(x) = 0.2$ and $w_2(y) = 2.8$

w_1 and w_2 are

- 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

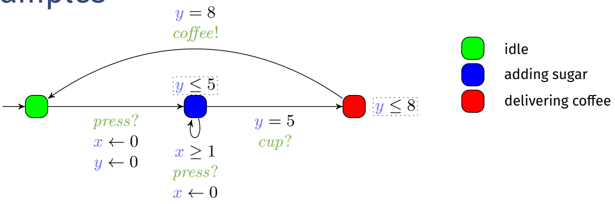
- Let w_3 be such that $w_3(x) = 1.3$ and $w_3(y) = 0.7$
- Let w_4 be such that $w_4(x) = 3.9$ and $w_4(y) = 0.2$

w_3 and w_4 are

- 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

- 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

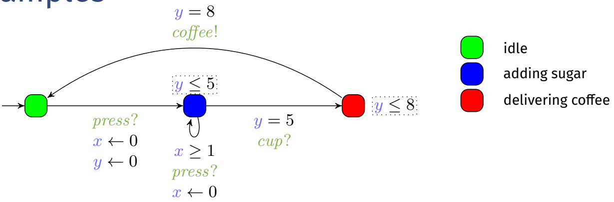
- Let w_5 be such that $w_5(x) = 0.8$ and $w_5(y) = 2.7$

w_1 and w_5 are

- 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

- 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

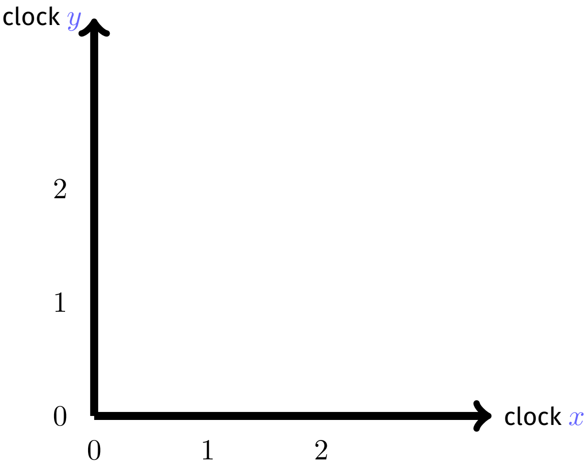
- Let w_5 be such that $w_5(x) = 0.8$ and $w_5(y) = 2.7$

w_1 and w_5 are

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

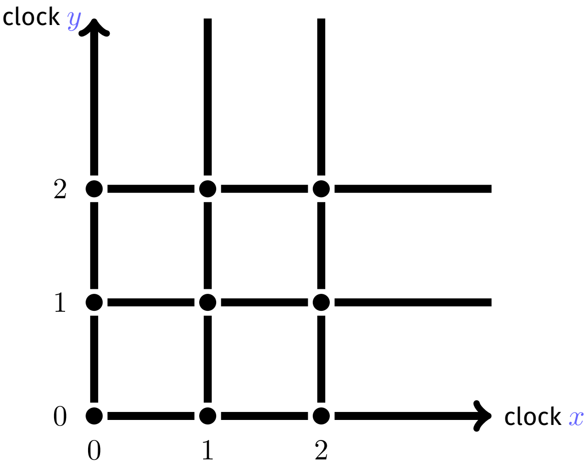
w_1 and w_6 are

Regions



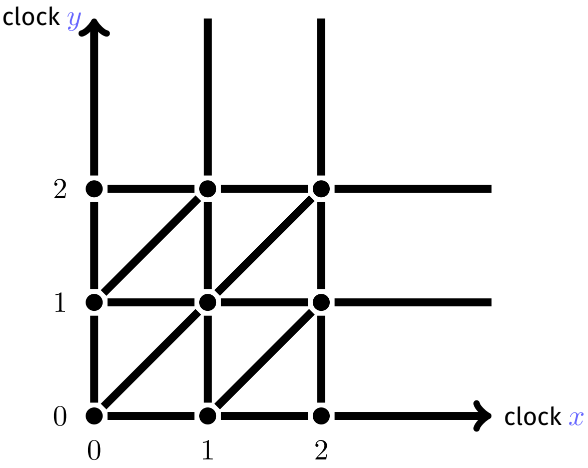
Inspired by a similar LaTeX illustration by Patricia Bouyer

Regions



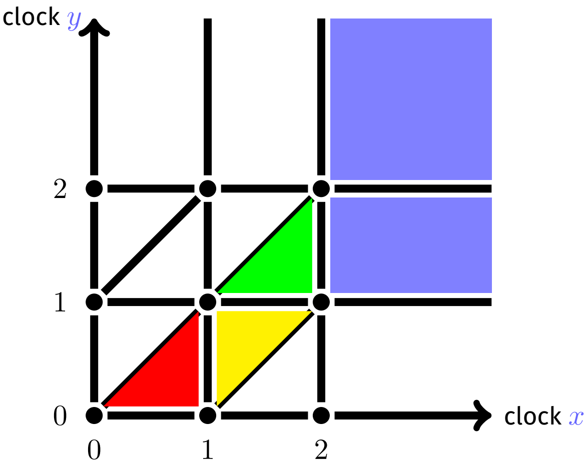
Inspired by a similar LaTeX illustration by Patricia Bouyer

Regions



Inspired by a similar LaTeX illustration by Patricia Bouyer

Regions



Inspired by a similar LaTeX illustration by Patricia Bouyer

Region graph construction

Two successors:

- time-elapsing
- clock reset

(see white board for the graph construction)

On the region graph finiteness

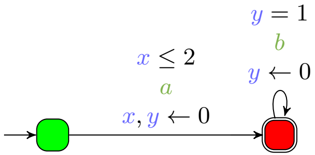
Is the region graph of TAs always finite?

On the region graph finiteness

Is the region graph of TAs always finite?

Region graph construction: exercise

Construct the region graph of the following TA:



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.

□

- [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.

□

- [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)

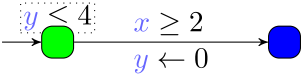
- Objective: group all concrete states reachable by the same sequence of discrete actions
- Symbolic state: a location ℓ and a (infinite) set of states Z
- For timed automata, Z can be represented by a convex polyhedron with a special form called zone, with constraints

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

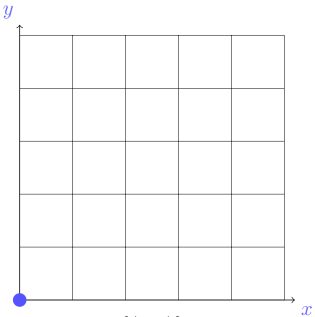
- Computation of successive reachable symbolic states can be performed symbolically with polyhedral operations: for edge $e = (\ell, a, g, R, \ell')$:

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

Symbolic states for timed automata (zones): Example

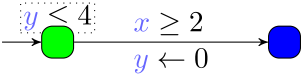


$$\text{Succ}\big((\square, Z), e\big) = \Big(\blacksquare, \big([(Z \cap g)]_R \cap I(\blacksquare)\big)^{\nearrow} \cap I(\blacksquare)\Big)$$

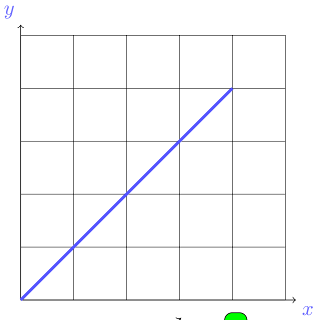


$\{(0,0)\}$

Symbolic states for timed automata (zones): Example

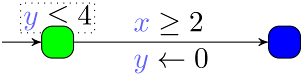


$$\mathsf{Succ}\big((\blacksquare, Z), e\big) = \Big(\blacksquare, \big([(Z \cap g)]_R \cap I(\blacksquare)\big)^{\nearrow} \cap I(\blacksquare)\Big)$$

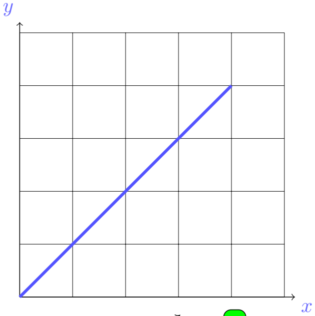


$Z_0 = \{(0,0)\}^{\nearrow} \cap I(\blacksquare)$

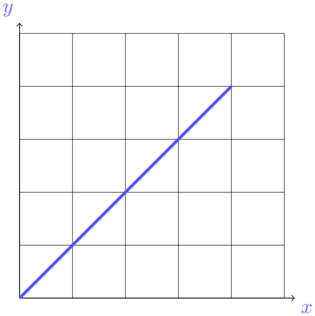
Symbolic states for timed automata (zones): Example



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

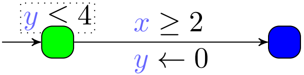


$Z_0 = \{(0,0)\}^{\nearrow} \cap I(\square)$

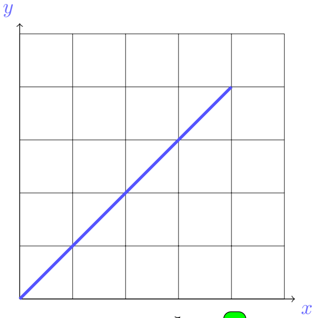


Z_0

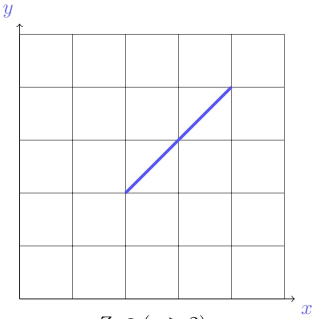
Symbolic states for timed automata (zones): Example



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

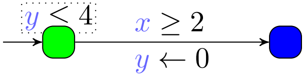


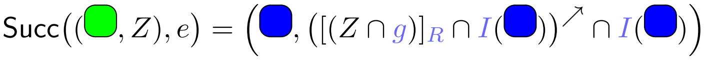
$Z_0 = \{(0,0)\}^{\nearrow} \cap I({\color{green}\blacksquare})$

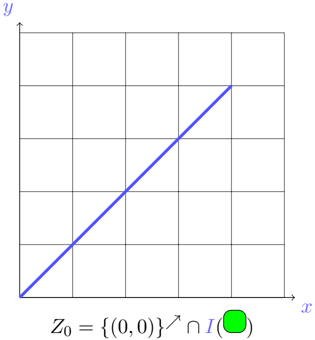


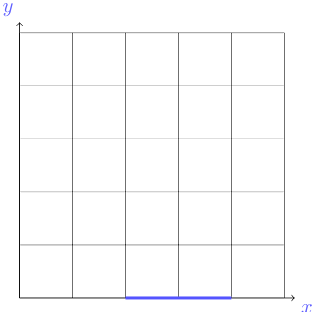
$Z_0 \cap (x \geq 2)$

Symbolic states for timed automata (zones): Example



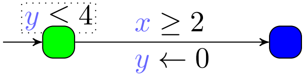




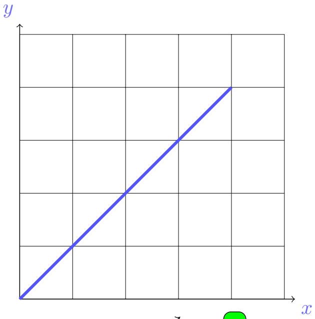


$[(Z_0 \cap (x \geq 2))]_{\{y\}}$

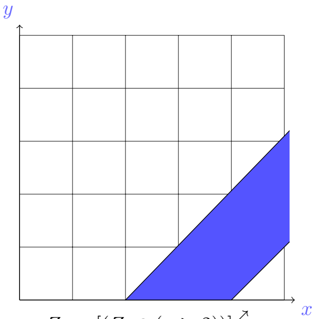
Symbolic states for timed automata (zones): Example



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



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



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

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

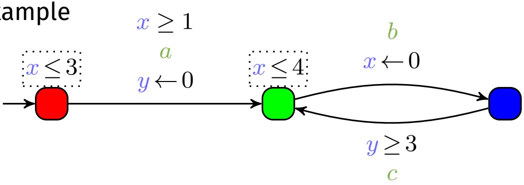
- Abstract state of a TA: pair (ℓ, C), where
  - ℓ is a location, and C is a constraint on the clocks ("zone")

Abstract semantics of timed automata

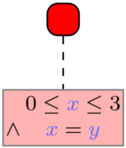
- Abstract state of a TA: pair (ℓ, C), where
  - ℓ is a location, and C is a constraint on the clocks ("zone")
- Abstract run: alternating sequence of abstract states and actions

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

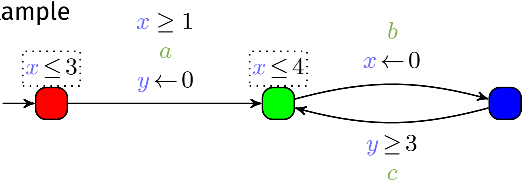


  - Possible abstract run for this TA

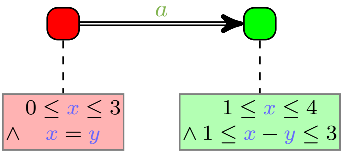


Abstract semantics of timed automata

- 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

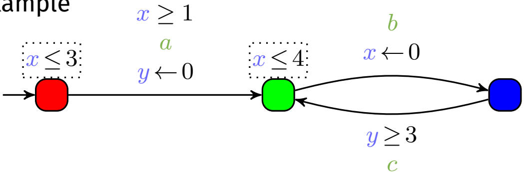


  - Possible abstract run for this TA

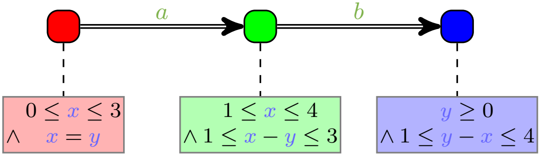


Abstract semantics of timed automata

- 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

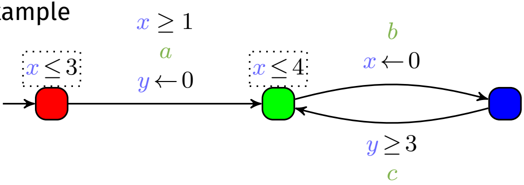


  - Possible abstract run for this TA

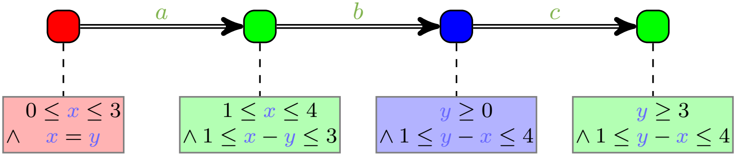


Abstract semantics of timed automata

- 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?

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

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?

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

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?

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

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?

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

• [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

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

Outline

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

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	$\rho, t \not\models \varphi$
$\rho, t \models \varphi \vee \psi$	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

- 

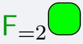
Continuous semantics of MTL

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

- 

- 

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

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

MTL: Extended syntax

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

MTL: Extended syntax

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

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

MTL: Extended syntax

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

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

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

MTL: Extended syntax

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

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

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

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

MTL: Extended syntax

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

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

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

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

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

MTL: Extended syntax

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

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

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

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

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

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

MTL: Extended syntax

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

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

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

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

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

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

(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"

MTL: Examples

Exercise

Express in MTL the following properties:

- “I will eventually get a job within a year”
- “The plane will never crash within the 8 hours of the flight”

MTL: Examples

Exercise

Express in MTL the following properties:

- “I will eventually get a job within a year”
- “The plane will never crash within the 8 hours of the flight”
- “Every time I ask a question, the teacher will answer me immediately”

MTL: Examples

Exercise

Express in MTL the following properties:

- “I will eventually get a job within a year”
- “The plane will never crash within the 8 hours of the flight”
- “Every time I ask a question, the teacher will answer me immediately”
- “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”
- “The plane will never crash within the 8 hours of the flight”
- “Every time I ask a question, the teacher will answer me immediately”
- “During the next 4 hours, I will starve unless I eventually get some food or 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”

MITL: Examples

Exercise

Express in MTL the following properties:

- “The plane will never crash within the 8 hours of the flight”

- “I will get a job in one year”

MITL: Examples

Exercise

Express in MTL the following properties:

- “The plane will never crash within the 8 hours of the flight”

- “I will get a job in one year”
  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"
- "I will get a job in one year"
  not expressible using MITL
- "During the next 4 hours, I will starve unless I eventually get some food or drinks"
- "Every time I ask a question, the teacher will answer me at least 2 and at most 4 minutes later"

MITL: Examples

Exercise

Express in MTL the following properties:

- "The plane will never crash within the 8 hours of the flight"
- "I will get a job in one year"
  not expressible using MITL
- "During the next 4 hours, I will starve unless I eventually get some food or drinks"
- "Every time I ask a question, the teacher will answer me at least 2 and at most 4 minutes later"
- "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”
- “I will get a job in one year”
  not expressible using MITL
- “During the next 4 hours, I will starve unless I eventually get some food or drinks”
- “Every time I ask a question, the teacher will answer me at least 2 and at most 4 minutes later”
- “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

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

. [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

- $\mathsf{E}\,\square\,\mathsf{U}_{\leq c}\,\square$

TCTL: Extended syntax

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

$$\varphi \land \psi \quad \equiv$$

TCTL: Extended syntax

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

$$\varphi \land \psi \quad \equiv$$

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

TCTL: Extended syntax

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

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

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

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

TCTL: Extended syntax

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

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

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

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

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

TCTL: Extended syntax

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

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

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

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

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

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

TCTL: Extended syntax

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

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

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

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

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

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

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

TCTL: Extended syntax

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

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

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

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

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

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

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

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

TCTL: Extended syntax

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

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

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

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

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

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

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

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

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

TCTL: Extended syntax

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

$$\begin{aligned}
\varphi \wedge \psi &\equiv \\
\varphi \implies \psi &\equiv \\
\varphi \iff \psi &\equiv \\
\mathsf{EF}_I \varphi &\equiv \\
\mathsf{AF}_I \varphi &\equiv \\
\mathsf{EG}_I \varphi &\equiv \\
\mathsf{AG}_I \varphi &\equiv \\
\mathsf{E}\varphi\mathsf{W}_I \psi &\equiv \\
\mathsf{A}\varphi\mathsf{W}_I \psi &\equiv
\end{aligned}$$

TCTL: Extended syntax

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

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

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

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

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

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

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

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

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

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

TCTL: Extended syntax

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

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

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

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

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

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

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

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

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

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

TCTL: Extended syntax

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

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

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

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

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

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

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

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

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

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

$$\equiv$$

TCTL: Extended syntax

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

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

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

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

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

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

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

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

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

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

$$\equiv$$

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"

- "I may get a job before next year"

TCTL: Examples

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

- “I may get a job before next year”

- “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”
- “I may get a job before next year”
- “Whenever a fire breaks, it is sure that the alarm will start ringing at least 5 seconds and at most 10 seconds later”
- “Whatever happens, I will love you for 2 years after we marry”

TCTL: Examples

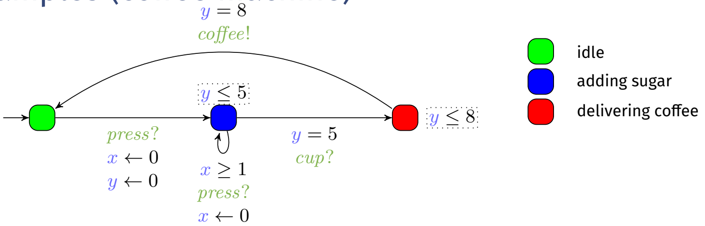
- "Whatever happens, the plane will never crash in the next 10 minutes"

- "I may get a job before next year"

- "Whenever a fire breaks, it is sure that the alarm will start ringing at least 5 seconds and at most 10 seconds later"

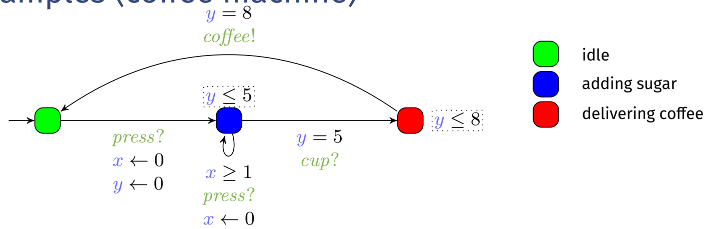
- "Whatever happens, I will love you for 2 years after we marry"

TCTL: Examples (coffee machine)



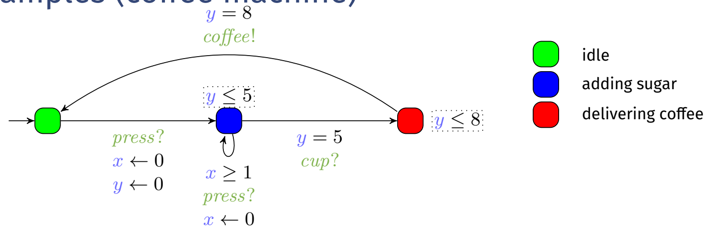
- “Whenever the button is pressed, a coffee is necessarily eventually delivered within 10 units of time.”

TCTL: Examples (coffee machine)



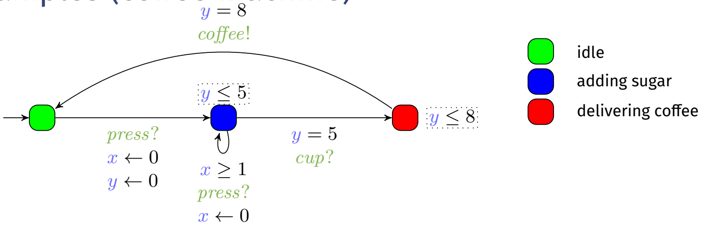
- "Whenever the button is pressed, a coffee is necessarily eventually delivered within 10 units of time."

TCTL: Examples (coffee machine)



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

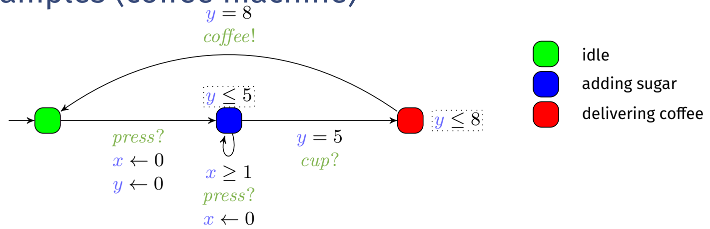
TCTL: Examples (coffee machine)



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

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

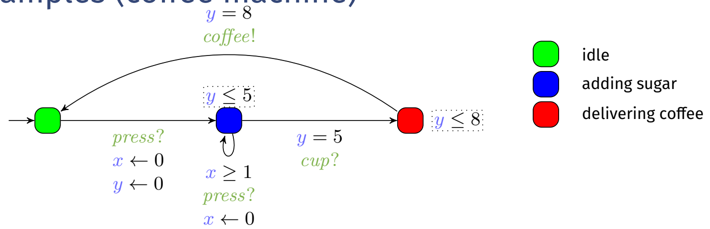
TCTL: Examples (coffee machine)



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

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

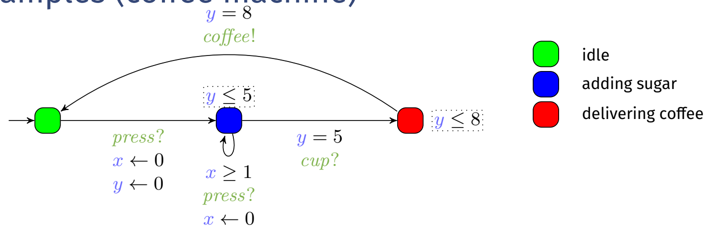
TCTL: Examples (coffee machine)



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

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

TCTL: Examples (coffee machine)

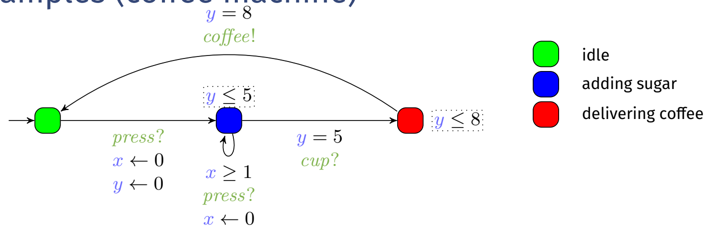


- "Whenever the button is pressed, a coffee is necessarily eventually delivered within 10 units of time."

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

- "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)

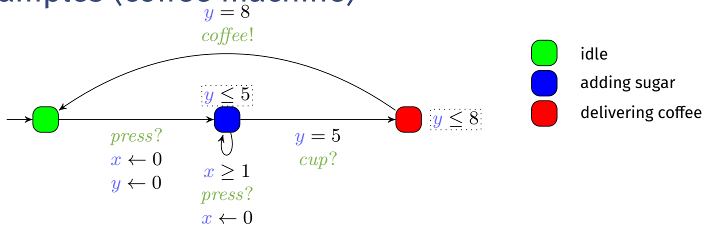


- "Whenever the button is pressed, a coffee is necessarily eventually delivered within 10 units of time."

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

- "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."

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

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

(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

Outline

Observers for timed automata

Observers (both untimed and timed) can be used for timed automata

Just as for FA:

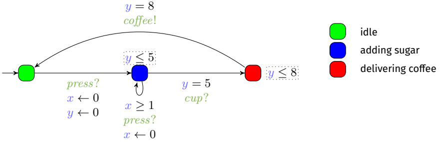
- A TA observer is an automaton that observes the system behavior
- It synchronizes with other automata's actions
- It can read the clocks of the system, and/or feature its own clock(s)
- It must be non-blocking
  - Pay attention to timelocks or deadlocks!
- Its location(s) give an indication on the system property

Then verifying the property reduces to a reachability condition on the observer (in parallel with the system)

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

- [ABL98; Ace+03] 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; 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

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+09a]
- 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+09a] 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

Some case studies and application domains (1/2)

- Scheduling and real-time systems

  [Feh99] [AM01] [AAM06] [AM12]

- Protocols
  - Bounded retransmission protocol [DAr+97]
  - Audio-video protocol [Hav+97]
  - Fast Reservation Protocol [TY98]
  - IEEE 1394a root contention protocol [SS01]

- [Feh99] Ansgar Fehnker. « Scheduling a Steel Plant with Timed Automata ». In: *RTCSA*. IEEE Computer Society, 1999, pp. 280–286
- [AM01] Yasmina Abdeddaïm and Oded Maler. « Job-Shop Scheduling Using Timed Automata ». In: *CAV*. vol. 2102. Lecture Notes in Computer Science. Springer, 2001, pp. 478–492. ISBN: 3-540-42345-1
- [AAM06] Yasmina Adbeddaïm, Eugene Asarin, and Oded Maler. « Scheduling with timed automata ». In: *Theoretical Computer Science* 354.2 (Mar. 2006), pp. 272–300
- [AM12] Yasmina Abdeddaïm and Damien Masson. « Real-Time Scheduling of Energy Harvesting Embedded Systems with Timed Automata ». In: *RTCSA*. IEEE Computer Society, 2012, pp. 31–40
- [DAr+97] Pedro R. D'Argenio, Joost-Pieter Katoen, Theo C. Ruys, and Jan Tretmans. « The Bounded Retransmission Protocol Must Be on Time! » In: *TACAS*. vol. 1217. Lecture Notes in Computer Science. Springer, 1997, pp. 416–431. ISBN: 3-540-62790-1
- [Hav+97] Klaus Havelund, Arne Skou, Kim Guldstrand Larsen, and K. Lund. « Formal modeling and analysis of an audio/video protocol: an industrial case study using UPPAAL ». In: *RTSS*. IEEE Computer Society, 1997, pp. 2–13
- [TY98] Stavros Tripakis and Sergio Yovine. « Verification of the Fast Reservation Protocol with Delayed Transmission using the Tool Kronos ». In: *RTAS*. IEEE Computer Society, 1998, pp. 165–170
- [SS01] David P. L. Simons and Mariëlle Stoelinga. « Mechanical verification of the IEEE 1394a root contention protocol using Uppaal2k ». In: *International Journal on Software Tools for Technology Transfer* 3.4 (2001), pp. 469–485

Some case studies and application domains (2/2)

- Hardware circuits [Boz+02] [Che+09]
- Health and biology [Sch+14]
- Monitoring [WAH16] [WHS18]
- Survey on the industrial use of UPPAAL [LLN18]

- [Boz+02] Marius Bozga, Jianmin Hou, Oded Maler, and Sergio Yovine. « Verification of Asynchronous Circuits using Timed Automata ». In: *Electronic Notes in Theoretical Computer Science* 65.6 (2002), pp. 47–59
- [Che+09] Rémy Chevallier, Emmanuelle Encrenaz-Tiphène, Laurent Fribourg, and Weiwen Xu. « Timed Verification of the Generic Architecture of a Memory Circuit Using Parametric Timed Automata ». In: *Formal Methods in System Design* 34.1 (Feb. 2009), pp. 59–81
- [Sch+14] Stefano Schivo, Jetse Scholma, Brend Wanders, Ricardo A. Urquidi Camacho, Paul E. van der Vet, Marcel Karperien, Rom Langerak, Jaco van de Pol, and Janine N. Post. « Modeling Biological Pathway Dynamics With Timed Automata ». In: *IEEE Journal of Biomedical and Health Informatics* 18.3 (2014), pp. 832–839
- [WAH16] Masaki Waga, Takumi Akazaki, and Ichiro Hasuo. « A Boyer-Moore Type Algorithm for Timed Pattern Matching ». In: *FORMATS*. vol. 9884. Lecture Notes in Computer Science. Springer, 2016, pp. 121–139
- [WHS18] Masaki Waga, Ichiro Hasuo, and Kohei Suenaga. « MONAA: A Tool for Timed Pattern Matching with Automata-Based Acceleration ». In: *MT@CPSWeek*. IEEE, 2018, pp. 14–15
- [LLN18] Kim Guldstrand Larsen, Florian Lorber, and Brian Nielsen. « 20 Years of UPPAAL Enabled Industrial Model-Based Validation and Beyond ». In: *ISoLA, Part IV*. vol. 11247. Lecture Notes in Computer Science. Springer, 2018, pp. 212–229

Outline

Further challenges: theory

- Timed language inclusion (using TA as a specification language)
  - Decidable subclasses [OW03] [OW04]
  - Practical algorithms [Wan+17]

- Robustness [De +04] [BMS13] [Bac+18]

- [OW03] Joël Ouaknine and James Worrell. « Universality and Language Inclusion for Open and Closed Timed Automata ». In: *HSCC*. vol. 2623. Lecture Notes in Computer Science. Springer, 2003, pp. 375–388
- [OW04] Joël Ouaknine and James Worrell. « On the Language Inclusion Problem for Timed Automata: Closing a Decidability Gap ». In: *LiCS*. IEEE Computer Society, 2004, pp. 54–63
- [Wan+17] Xinyu Wang, Jun Sun, Ting Wang, and Shengchao Qin. « Language Inclusion Checking of Timed Automata with Non-Zenoness ». In: *IEEE Transactions on Software Engineering* 43.11 (2017), pp. 995–1008
- [De +04] Martin De Wulf, Laurent Doyen, Nicolas Markey, and Jean-François Raskin. « Robustness and Implementability of Timed Automata ». In: *FORMATS and FTRTFT*. vol. 3253. Lecture Notes in Computer Science. Springer, 2004, pp. 118–133
- [BMS13] Patricia Bouyer, Nicolas Markey, and Ocan Sankur. « Robustness in timed automata ». In: *RP*. vol. 8169. Lecture Notes in Computer Science. Invited paper. Springer, Sept. 2013, pp. 1–18
- [Bac+18] Giovanni Bacci, Patricia Bouyer, Uli Fahrenberg, Kim Guldstrand Larsen, Nicolas Markey, and Pierre-Alain Reynier. « Optimal and Robust Controller Synthesis – Using Energy Timed Automata with Uncertainty ». In: *FM*. vol. 10951. Lecture Notes in Computer Science. Springer, 2018, pp. 203–221

Further challenges: algorithms and applications

- Controller synthesis [San+13] [Bac+18]
  - Game theory
- Distributed algorithms [Laa+13] [ZNL16]

- [San+13] Ocan Sankur, Patricia Bouyer, Nicolas Markey, and Pierre-Alain Reynier. « Robust Controller Synthesis in Timed Automata ». In: *CONCUR*. vol. 8052. Lecture Notes in Computer Science. Springer, 2013, pp. 546–560
- [Bac+18] Giovanni Bacci, Patricia Bouyer, Uli Fahrenberg, Kim Guldstrand Larsen, Nicolas Markey, and Pierre-Alain Reynier. « Optimal and Robust Controller Synthesis – Using Energy Timed Automata with Uncertainty ». In: *FM*. vol. 10951. Lecture Notes in Computer Science. Springer, 2018, pp. 203–221
- [Laa+13] Alfons Laarman, Mads Chr. Olesen, Andreas Engelbredt Dalsgaard, Kim Guldstrand Larsen, and Jaco van De Pol. « Multi-Core Emptiness Checking of Timed Büchi Automata using Inclusion Abstraction ». In: *CAV*. vol. 8044. Lecture Notes in Computer Science. Springer, July 2013, pp. 968–983
- [ZNL16] Zhengkui Zhang, Brian Nielsen, and Kim Guldstrand Larsen. « Time optimal reachability analysis using swarm verification ». In: *SAC*. ACM, 2016, pp. 1634–1640

Further challenges: algorithms and applications

- Controller synthesis [San+13] [Bac+18]
  - Game theory
- Distributed algorithms [Laa+13] [ZNL16]

Still a very active research field!

- [San+13] Ocan Sankur, Patricia Bouyer, Nicolas Markey, and Pierre-Alain Reynier. « Robust Controller Synthesis in Timed Automata ». In: *CONCUR*. vol. 8052. Lecture Notes in Computer Science. Springer, 2013, pp. 546–560
- [Bac+18] Giovanni Bacci, Patricia Bouyer, Uli Fahrenberg, Kim Guldstrand Larsen, Nicolas Markey, and Pierre-Alain Reynier. « Optimal and Robust Controller Synthesis – Using Energy Timed Automata with Uncertainty ». In: *FM*. vol. 10951. Lecture Notes in Computer Science. Springer, 2018, pp. 203–221
- [Laa+13] Alfons Laarman, Mads Chr. Olesen, Andreas Engelbredt Dalsgaard, Kim Guldstrand Larsen, and Jaco van De Pol. « Multi-Core Emptiness Checking of Timed Büchi Automata using Inclusion Abstraction ». In: *CAV*. vol. 8044. Lecture Notes in Computer Science. Springer, July 2013, pp. 968–983
- [ZNL16] Zhengkui Zhang, Brian Nielsen, and Kim Guldstrand Larsen. « Time optimal reachability analysis using swarm verification ». In: *SAC*. ACM, 2016, pp. 1634–1640

What's beyond timed automata...?

- Stopping clocks: stopwatch automata [CL00]
  - ☹ Undecidable
  - ☺ Interesting application domains
- Adding costs: energy [Beh+01] [ALP04]
- Enriching TA with tasks [Fer+07]
- Adding unknown parameters [AHV93]
- Allowing non-linear clocks: hybrid automata [Hen96] [Asa+12]
- Adding probabilities [Kwi+02]
  - Statistical model checking [LDB10]

- [CL00] Franck Cassez and Kim Guldstrand Larsen. « The Impressive Power of Stopwatches ». In: *CONCUR*. vol. 1877. Lecture Notes in Computer Science. Springer, 2000, pp. 138–152
- [Beh+01] Gerd Behrmann, Ansgar Fehnker, Thomas Hune, Kim Guldstrand Larsen, Paul Pettersson, Judi Romijn, and Frits W. Vaandrager. « Minimum-Cost Reachability for Priced Timed Automata ». In: *HSCC*. vol. 2034. Lecture Notes in Computer Science. Springer, 2001, pp. 147–161. ISBN: 3-540-41866-0
- [ALP04] Rajeev Alur, Salvatore La Torre, and George J. Pappas. « Optimal paths in weighted timed automata ». In: *Theoretical Computer Science* 318.3 (2004), pp. 297–322
- [Fer+07] Elena Fersman, Pavel Krcál, Paul Pettersson, and Wang Yi. « Task automata: Schedulability, decidability and undecidability ». In: *Information and Computation* 205.8 (2007), pp. 1149–1172
- [AHV93] Rajeev Alur, Thomas A. Henzinger, and Moshe Y. Vardi. « Parametric real-time reasoning ». In: *STOC*. ACM, 1993, pp. 592–601
- [Hen96] Thomas A. Henzinger. « The Theory of Hybrid Automata ». In: *LiCS*. IEEE Computer Society, 1996, pp. 278–292
- [Asa+12] Eugene Asarin, Venkatesh Mysore, Amir Pnueli, and Gerardo Schneider. « Low dimensional hybrid systems – Decidable, undecidable,

Towards a parametrization…

- Challenge 1: systems incompletely specified
  - Some delays may not be known yet, or may change

- Challenge 2: Robustness [Mar11]
  - What happens if 8 is implemented with 7.99?
  - Can I really get a coffee with 5 doses of sugar?

- Challenge 3: Optimization of timing constants
  - Up to which value of the delay between two actions $press?$ can I still order a coffee with 3 doses of sugar?

- Challenge 4: Avoiding numerous verifications
  - If one of the timing delays of the model changes, should I model check again the whole system?

. [Mar11] Nicolas Markey. « Robustness in Real-time Systems ». In: *SIES*. IEEE Computer Society Press, June 2011, pp. 28–34

Towards a parametrization…

- Challenge 1: systems incompletely specified
  - Some delays may not be known yet, or may change
- Challenge 2: Robustness [Mar11]
  - What happens if 8 is implemented with 7.99?
  - Can I really get a coffee with 5 doses of sugar?
- Challenge 3: Optimization of timing constants
  - Up to which value of the delay between two actions $press?$ can I still order a coffee with 3 doses of sugar?
- Challenge 4: Avoiding numerous verifications
  - If one of the timing delays of the model changes, should I model check again the whole system?
- A solution: Parametric analysis
  - Consider that timing constants are unknown (parameters)
  - Find good values for the parameters s.t. the system behaves well

. [Mar11] Nicolas Markey. « Robustness in Real-time Systems ». In: *SIES*. IEEE Computer Society Press, June 2011, pp. 28–34

Bibliography

References I

[AAM06] Yasmina Adbeddaïm, Eugene Asarin, and Oded Maler. « Scheduling with timed automata ». In: *Theoretical Computer Science* 354.2 (Mar. 2006), pp. 272–300. DOI: 10.1016/j.tcs.2005.11.018.

[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.

References II

[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.

[AHV93] Rajeev Alur, Thomas A. Henzinger, and Moshe Y. Vardi. « Parametric real-time reasoning ». In: *STOC* (May 16–18, 1993). Ed. by S. Rao Kosaraju, David S. Johnson, and Alok Aggarwal. San Diego, California, USA: ACM, 1993, pp. 592–601. DOI: 10.1145/167088.167242.

[ALP04] Rajeev Alur, Salvatore La Torre, and George J. Pappas. « Optimal paths in weighted timed automata ». In: *Theoretical Computer Science* 318.3 (2004), pp. 297–322. DOI: 10.1016/j.tcs.2003.10.038.

[AM01] Yasmina Abdeddaïm and Oded Maler. « Job-Shop Scheduling Using Timed Automata ». 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. 478–492. ISBN: 3-540-42345-1. DOI: 10.1007/3-540-44585-4_46.

[AM12] Yasmina Abdeddaïm and Damien Masson. « Real-Time Scheduling of Energy Harvesting Embedded Systems with Timed Automata ». In: *RTCSA* (Aug. 19–22, 2012). Seoul, South Korea: IEEE Computer Society, 2012, pp. 31–40. DOI: 10.1109/RTCSA.2012.21.

[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.

References III

[Asa+12] Eugene Asarin, Venkatesh Mysore, Amir Pnueli, and Gerardo Schneider. « Low dimensional hybrid systems – Decidable, undecidable, don't know ». In: *Information and Computation* 211 (2012), pp. 138–159. DOI: 10.1016/j.ic.2011.11.006.

[Bac+18] Giovanni Bacci, Patricia Bouyer, Uli Fahrenberg, Kim Guldstrand Larsen, Nicolas Markey, and Pierre-Alain Reynier. « Optimal and Robust Controller Synthesis – Using Energy Timed Automata with Uncertainty ». In: *FM* (July 15–17, 2018). Ed. by Klaus Havelund, Jan Peleska, Bill Roscoe, and Erik P. de Vink. Vol. 10951. Lecture Notes in Computer Science. Oxford, UK: Springer, 2018, pp. 203–221. DOI: 10.1007/978-3-319-95582-7_12.

[Beh+01] Gerd Behrmann, Ansgar Fehnker, Thomas Hune, Kim Guldstrand Larsen, Paul Pettersson, Judi Romijn, and Frits W. Vaandrager. « Minimum-Cost Reachability for Priced Timed Automata ». In: *HSCC* (Mar. 28–30, 2001). Ed. by Maria Domenica Di Benedetto and Alberto L. Sangiovanni-Vincentelli. Vol. 2034. Lecture Notes in Computer Science. Rome, Italy: Springer, 2001, pp. 147–161. ISBN: 3-540-41866-0. DOI: 10.1007/3-540-45351-2_15.

[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.

[Bér+05] Béatrice Bérard, Franck Cassez, Serge Haddad, Didier Lime, and Olivier H. Roux. « Comparison of the Expressiveness of Timed Automata and Time Petri Nets ». In: *FORMATS* (Sept. 26–28, 2005). Ed. by Paul Pettersson and Wang Yi. Vol. 3829. Lecture Notes in Computer Science. Uppsala, Sweden: Springer, 2005, pp. 211–225. DOI: 10.1007/11603009_17.

References IV

[Bér+13] Béatrice Bérard, Franck Cassez, Serge Haddad, Didier Lime, and Olivier H. Roux. « The expressive power of time Petri nets ». In: *Theoretical Computer Science* 474 (2013), pp. 1–20. DOI: 10.1016/j.tcs.2012.12.005.

[BG06] Howard Bowman and Rodolfo Gómez. « How to stop time stopping ». In: *Formal Aspects of Computing* 18.4 (2006), pp. 459–493. DOI: 10.1007/s00165-006-0010-7.

[BMS13] Patricia Bouyer, Nicolas Markey, and Ocan Sankur. « Robustness in timed automata ». In: *RP* (Sept. 25–27, 2013). Ed. by Parosh Aziz Abdulla and Igor Potapov. Vol. 8169. Lecture Notes in Computer Science. Invited paper. Uppsala, Sweden: Springer, Sept. 2013, pp. 1–18. DOI: 10.1007/978-3-642-41036-9_1.

[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.

[Boz+02] Marius Bozga, Jianmin Hou, Oded Maler, and Sergio Yovine. « Verification of Asynchronous Circuits using Timed Automata ». In: *Electronic Notes in Theoretical Computer Science* 65.6 (2002), pp. 47–59. DOI: 10.1016/S1571-0661(04)80468-7.

[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.

References V

[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* (Dec. 12–14, 2018). Ed. by Anthony Widjaja Lin and Jun Sun. Melbourne, Australia: IEEE Computer Society, 2018, pp. 180–184. DOI: 10.1109/ICECCS2018.2018.00027.

[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.

[Che+09] Rémy Chevallier, Emmanuelle Encrenaz-Tiphène, Laurent Fribourg, and Weiwen Xu. « Timed Verification of the Generic Architecture of a Memory Circuit Using Parametric Timed Automata ». In: *Formal Methods in System Design* 34.1 (Feb. 2009), pp. 59–81. DOI: 10.1007/s10703-008-0061-x.

[CL00] Franck Cassez and Kim Guldstrand Larsen. « The Impressive Power of Stopwatches ». In: *CONCUR* (Aug. 22–25, 2000). Ed. by Catuscia Palamidessi. Vol. 1877. Lecture Notes in Computer Science. University Park, PA, USA: Springer, 2000, pp. 138–152. DOI: 10.1007/3-540-44618-4_12.

[DAr+97] Pedro R. D'Argenio, Joost-Pieter Katoen, Theo C. Ruys, and Jan Tretmans. « The Bounded Retransmission Protocol Must Be on Time! » In: *TACAS*. Ed. by Ed Brinksma. Vol. 1217. Lecture Notes in Computer Science. Enschede, The Netherlands: Springer, 1997, pp. 416–431. ISBN: 3-540-62790-1. DOI: 10.1007/BFb0035403.

References VI

[De +04] Martin De Wulf, Laurent Doyen, Nicolas Markey, and Jean-François Raskin. « Robustness and Implementability of Timed Automata ». In: *FORMATS and FTRTFT* (Sept. 22–24, 2004). Ed. by Yassine Lakhnech and Sergio Yovine. Vol. 3253. Lecture Notes in Computer Science. Grenoble, France: Springer, 2004, pp. 118–133. DOI: 10.1007/978-3-540-30206-3_10.

[Feh99] Ansgar Fehnker. « Scheduling a Steel Plant with Timed Automata ». In: *RTCSA* (Dec. 13–16, 1999). Hong Kong, China: IEEE Computer Society, 1999, pp. 280–286. DOI: 10.1109/RTCSA.1999.811256.

[Fer+07] Elena Fersman, Pavel Krcál, Paul Pettersson, and Wang Yi. « Task automata: Schedulability, decidability and undecidability ». In: *Information and Computation* 205.8 (2007), pp. 1149–1172. DOI: 10.1016/j.ic.2007.01.009.

[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.

[GB07] Rodolfo Gómez and Howard Bowman. « Efficient Detection of Zeno Runs in Timed Automata ». In: *FORMATS* (Oct. 3–5, 2007). Ed. by Jean-François Raskin and P. S. Thiagarajan. Vol. 4763. Lecture Notes in Computer Science. Salzburg, Austria: Springer, 2007, pp. 195–210. DOI: 10.1007/978-3-540-75454-1_15.

[Hav+97] Klaus Havelund, Arne Skou, Kim Guldstrand Larsen, and K. Lund. « Formal modeling and analysis of an audio/video protocol: an industrial case study using UPPAAL ». In: *RTSS* (Dec. 3–5, 1997). San Francisco, CA, USA: IEEE Computer Society, 1997, pp. 2–13. DOI: 10.1109/REAL.1997.641264.

References VII

[Hen+94] Thomas A. Henzinger, Xavier Nicollin, Joseph Sifakis, and Sergio Yovine. « Symbolic Model Checking for Real-Time Systems ». In: *Information and Computation* 111.2 (1994), pp. 193–244. DOI: 10.1006/inco.1994.1045.

[Hen96] Thomas A. Henzinger. « The Theory of Hybrid Automata ». In: *LiCS* (July 27–30, 1996). Ed. by Moshe Y. Vardi and Edmund M. Clarke. New Brunswick, New Jersey, USA: IEEE Computer Society, 1996, pp. 278–292. DOI: 10.1109/LICS.1996.561342.

[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.

[HKW95] Thomas A. Henzinger, Peter W. Kopke, and Howard Wong-Toi. « The Expressive Power of Clocks ». In: *ICALP* (July 10–14, 1995). Ed. by Zoltán Fülöp and Ferenc Gécseg. Vol. 944. Lecture Notes in Computer Science. Szeged, Hungary: Springer, 1995, pp. 417–428. DOI: 10.1007/3-540-60084-1_93.

[HSW12] Frédéric Herbreteau, B. Srivathsan, and Igor Walukiewicz. « Efficient emptiness check for timed Büchi automata ». In: *Formal Methods in System Design* 40.2 (2012), pp. 122–146. DOI: 10.1007/s10703-011-0133-1.

[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.

References VIII

[Kwi+02] Marta Z. Kwiatkowska, Gethin Norman, Roberto Segala, and Jeremy Sproston. « Automatic Verification of Real-time Systems with Discrete Probability Distributions ». In: *Theoretical Computer Science* 282.1 (2002), pp. 101–150. DOI: 10.1016/S0304-3975(01)00046-9.

[Laa+13] Alfons Laarman, Mads Chr. Olesen, Andreas Engelbredt Dalsgaard, Kim Guldstrand Larsen, and Jaco van De Pol. « Multi-Core Emptiness Checking of Timed Büchi Automata using Inclusion Abstraction ». In: *CAV* (July 13–19, 2013). Ed. by Natasha Sharygina and Helmut Veith. Vol. 8044. Lecture Notes in Computer Science. Saint Petersburg, Russia: Springer, July 2013, pp. 968–983. DOI: 10.1007/978-3-642-39799-8_69.

[LDB10] Axel Legay, Benoît Delahaye, and Saddek Bensalem. « Statistical Model Checking: An Overview ». In: *RV* (Nov. 1–4, 2010). Ed. by Howard Barringer, Yliès Falcone, Bernd Finkbeiner, Klaus Havelund, Insup Lee, Gordon J. Pace, Grigore Rosu, Oleg Sokolsky, and Nikolai Tillmann. Vol. 6418. Lecture Notes in Computer Science. St. Julians, Malta: Springer, 2010, pp. 122–135. DOI: 10.1007/978-3-642-16612-9_11.

[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.

References IX

[LLN18] Kim Guldstrand Larsen, Florian Lorber, and Brian Nielsen. « 20 Years of UPPAAL Enabled Industrial Model-Based Validation and Beyond ». In: *ISoLA, Part IV* (Nov. 5–9, 2018). Ed. by Tiziana Margaria and Bernhard Steffen. Vol. 11247. Lecture Notes in Computer Science. Limassol, Cyprus: Springer, 2018, pp. 212–229. DOI: 10.1007/978-3-030-03427-6_18.

[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.

[Mar11] Nicolas Markey. « Robustness in Real-time Systems ». In: *SIES*. Ed. by Iain Bate and Roberto Passerone. Västerås, Sweden: IEEE Computer Society Press, June 2011, pp. 28–34. DOI: 10.1109/SIES.2011.5953652.

[Mer74] Philip Meir Merlin. « A study of the recoverability of computing systems. ». PhD thesis. University of California, Irvine, CA, USA, 1974.

[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.

[OW03] Joël Ouaknine and James Worrell. « Universality and Language Inclusion for Open and Closed Timed Automata ». In: *HSCC* (Apr. 3–5, 2003). Ed. by Oded Maler and Amir Pnueli. Vol. 2623. Lecture Notes in Computer Science. Prague, Czech Republic: Springer, 2003, pp. 375–388. DOI: 10.1007/3-540-36580-X_28.

References X

[OW04] Joël Ouaknine and James Worrell. « On the Language Inclusion Problem for Timed Automata: Closing a Decidability Gap ». In: *LiCS* (July 14–17, 2004). Turku, Finland: IEEE Computer Society, 2004, pp. 54–63. DOI: 10.1109/LICS.2004.1319600.

[San+13] Ocan Sankur, Patricia Bouyer, Nicolas Markey, and Pierre-Alain Reynier. « Robust Controller Synthesis in Timed Automata ». In: *CONCUR* (Aug. 27–30, 2013). Ed. by Pedro R. D'Argenio and Hernán C. Melgratti. Vol. 8052. Lecture Notes in Computer Science. Buenos Aires, Argentina: Springer, 2013, pp. 546–560. DOI: 10.1007/978-3-642-40184-8_38.

[Sch+14] Stefano Schivo, Jetse Scholma, Brend Wanders, Ricardo A. Urquidi Camacho, Paul E. van der Vet, Marcel Karperien, Rom Langerak, Jaco van de Pol, and Janine N. Post. « Modeling Biological Pathway Dynamics With Timed Automata ». In: *IEEE Journal of Biomedical and Health Informatics* 18.3 (2014), pp. 832–839. DOI: 10.1109/JBHI.2013.2292880.

[Srb08] Jiří Srba. « Comparing the Expressiveness of Timed Automata and Timed Extensions of Petri Nets ». In: *FORMATS* (Sept. 15–17, 2008). Ed. by Franck Cassez and Claude Jard. Vol. 5215. Lecture Notes in Computer Science. Saint-Malo, France: Springer, 2008, pp. 15–32. DOI: 10.1007/978-3-540-85778-5_3.

[SS01] David P. L. Simons and Mariëlle Stoelinga. « Mechanical verification of the IEEE 1394a root contention protocol using Uppaal2k ». In: *International Journal on Software Tools for Technology Transfer* 3.4 (2001), pp. 469–485. DOI: 10.1007/s100090100059.

References XI

[Sun+09a] 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.

[Sun+09b] Jun Sun, Yang Liu, Jin Song Dong, and Xian Zhang. « Verifying Stateful Timed CSP Using Implicit Clocks and Zone Abstraction ». In: *ICFEM* (Dec. 9–12, 2009). Ed. by Karin K. Breitman and Ana Cavalcanti. Vol. 5885. Lecture Notes in Computer Science. Rio de Janeiro, Brazil: Springer, 2009, pp. 581–600. DOI: 10.1007/978-3-642-10373-5_30.

[Sun+13] Jun Sun, Yang Liu, Jin Song Dong, Yan Liu, Ling Shi, and Étienne André. « Modeling and Verifying Hierarchical Real-time Systems using Stateful Timed CSP ». In: *ACM Transactions on Software Engineering and Methodology* 22.1 (Feb. 2013), pp. 3.1–3.29. DOI: 10.1145/2430536.2430537.

[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.

[Tri99] Stavros Tripakis. « Verifying Progress in Timed Systems ». In: *ARTS* (May 26–28, 1999). Ed. by Joost-Pieter Katoen. Vol. 1601. Lecture Notes in Computer Science. Bamberg, Germany: Springer, 1999, pp. 299–314.

[TY98] Stavros Tripakis and Sergio Yovine. « Verification of the Fast Reservation Protocol with Delayed Transmission using the Tool Kronos ». In: *RTAS* (June 3–5, 1998). Denver, Colorado, USA: IEEE Computer Society, 1998, pp. 165–170. DOI: 10.1109/RTTAS.1998.683200.

References XII

[TYB05] Stavros Tripakis, Sergio Yovine, and Ahmed Bouajjani. « Checking Timed Büchi Automata Emptiness Efficiently ». In: *Formal Methods in System Design* 26.3 (2005), pp. 267–292. DOI: 10.1007/s10703-005-1632-8.

[WAH16] Masaki Waga, Takumi Akazaki, and Ichiro Hasuo. « A Boyer-Moore Type Algorithm for Timed Pattern Matching ». In: *FORMATS* (Aug. 24–26, 2016). Ed. by Martin Fränzle and Nicolas Markey. Vol. 9884. Lecture Notes in Computer Science. Québec, QC, Canada: Springer, 2016, pp. 121–139. DOI: 10.1007/978-3-319-44878-7_8.

[Wan+15] Ting Wang, Jun Sun, Xinyu Wang, Yang Liu, Yuanjie Si, Jin Song Dong, Xiaohu Yang, and Xiaohong Li. « A Systematic Study on Explicit-State Non-Zenoness Checking for Timed Automata ». In: *IEEE Transactions on Software Engineering* 41.1 (2015), pp. 3–18. DOI: 10.1109/TSE.2014.2359893.

[Wan+17] Xinyu Wang, Jun Sun, Ting Wang, and Shengchao Qin. « Language Inclusion Checking of Timed Automata with Non-Zenoness ». In: *IEEE Transactions on Software Engineering* 43.11 (2017), pp. 995–1008. DOI: 10.1109/TSE.2017.2653778.

[WHS18] Masaki Waga, Ichiro Hasuo, and Kohei Suenaga. « MONAA: A Tool for Timed Pattern Matching with Automata-Based Acceleration ». In: *MT@CPSWeek* (Apr. 10, 2018). Porto, Portugal: IEEE, 2018, pp. 14–15. DOI: 10.1109/MT-CPS.2018.00014.

[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.

References XIII

[ZNL16] Zhengkui Zhang, Brian Nielsen, and Kim Guldstrand Larsen. « Time optimal reachability analysis using swarm verification ». In: *SAC* (Apr. 4–8, 2016). Ed. by Sascha Ossowski. Pisa, Italy: ACM, 2016, pp. 1634–1640. DOI: 10.1145/2851613.2851828.

License of this document

Author: **Étienne André**

(LaTeX source available to academic teachers upon request)

Version: October 9, 2024