FORMATS 2016

Wednesday 24th of August, 2016
Québec, Québec

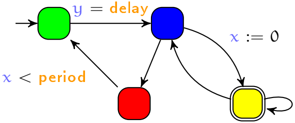
On the Expressiveness of Parametric Timed Automata

Étienne André[1,2], Didier Lime[2], Olivier H. Roux[2]

[1] LIPN, Université Paris 13, Sorbonne Paris Cité, CNRS, France
[2] École Centrale de Nantes, IRCCyN, CNRS, UMR 6597, France

Context: timed model checking

- Timed model checking



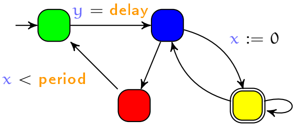
A model of the system

 is unreachable

A property to be satisfied

Context: timed model checking

- Timed model checking



$\stackrel{?}{\models}$

 is unreachable

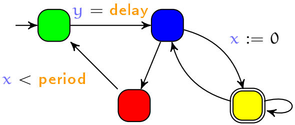
A model of the system

A property to be satisfied

- Question: does the model of the system satisfy the property?

Context: timed model checking

- Timed model checking



$\overset{?}{\models}$

 is unreachable

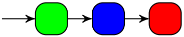
A model of the system

A property to be satisfied

- Question: does the model of the system satisfy the property?

Yes

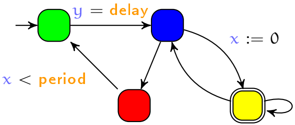
No



Counterexample

Context: parametric timed model checking

- Timed model checking



$\models^{?}$

 is unreachable

A model of the system

A property to be satisfied

- Question: for what values of the parameters does the model of the system satisfy the property?

Yes if. . .

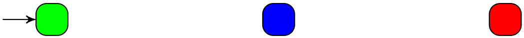
$$2\mathsf{delay} > \mathsf{period} \wedge \mathsf{period} < 20.46$$

Outline

Outline

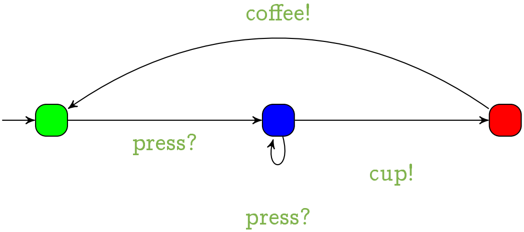
Timed automaton (TA)

- Finite state automaton (sets of locations)



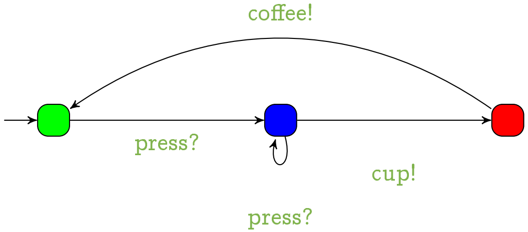
Timed automaton (TA)

- Finite state automaton (sets of locations and actions)



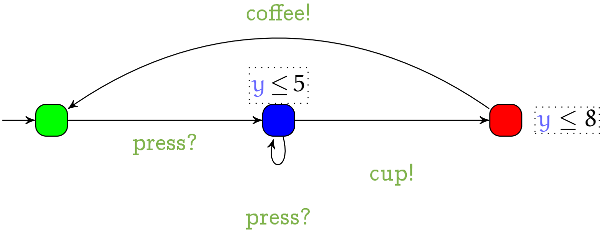
Timed automaton (TA)

- Finite state automaton (sets of locations and actions) augmented with a set X of clocks [Alur and Dill, 1994, Henzinger et al., 1994]
  - Real-valued variables evolving linearly at the same rate



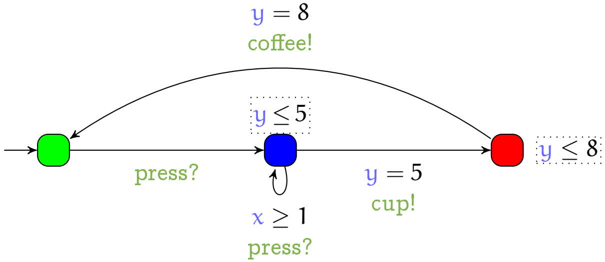
Timed automaton (TA)

- Finite state automaton (sets of locations and actions) augmented with a set X of clocks [Alur and Dill, 1994, Henzinger et al., 1994]
  - Real-valued variables evolving linearly at the same rate
- Features
  - Location invariant: constraint to be verified to stay at a location



Timed automaton (TA)

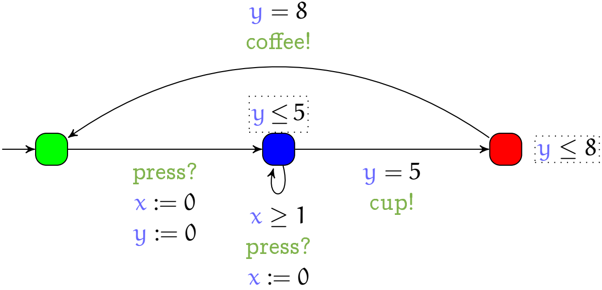
- Finite state automaton (sets of locations and actions) augmented with a set X of clocks [Alur and Dill, 1994, Henzinger et al., 1994]
  - Real-valued variables evolving linearly at the same rate
- Features
  - Location invariant: constraint to be verified to stay at a location
  - Transition guard: constraint to be verified to enable a transition



Timed automaton (TA)

- Finite state automaton (sets of locations and actions) augmented with a set $\mathbb{X}$ of clocks [Alur and Dill, 1994, Henzinger et al., 1994]
  - Real-valued variables evolving linearly at the same rate

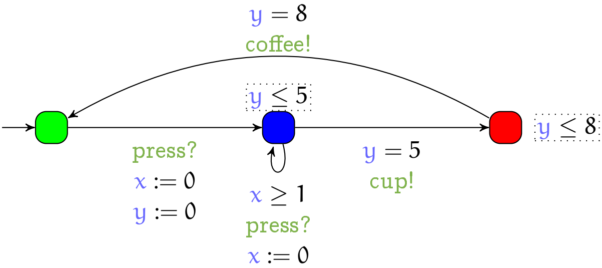
- Features
  - Location invariant: constraint to be verified to stay at a location
  - Transition guard: constraint to be verified to enable a transition
  - Clock reset: some of the clocks can be set to 0 at each transition



Concrete semantics of timed automata

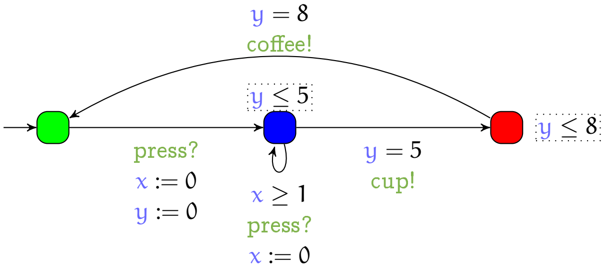
- Concrete state of a TA: pair (l, w), where
  - l is a location,
  - w is a valuation of each clock
- Concrete run: alternating sequence of concrete states and actions or time elapse

Examples of concrete runs



- Possible concrete runs for the coffee machine

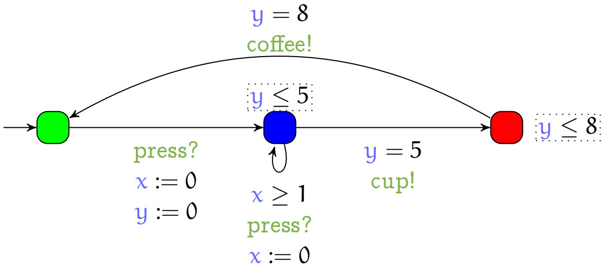
Examples of concrete runs



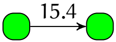
- Possible concrete runs for the coffee machine
  - Coffee with no sugar

x	0
y	0

Examples of concrete runs

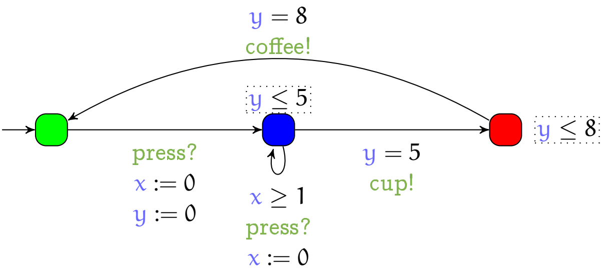


- Possible concrete runs for the coffee machine
  - Coffee with no sugar

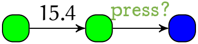


x	0	15.4
y	0	15.4

Examples of concrete runs

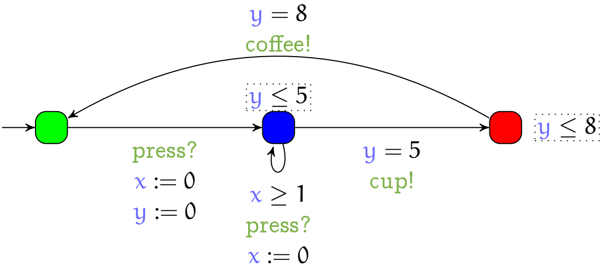


- Possible concrete runs for the coffee machine
  - Coffee with no sugar



x	0	15.4	0
y	0	15.4	0

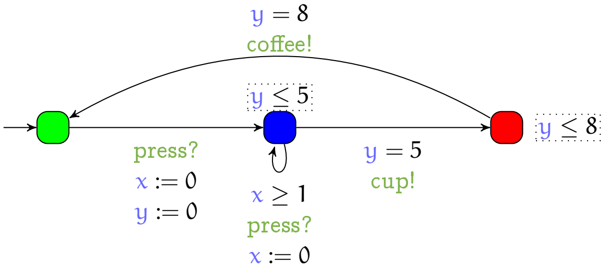
Examples of concrete runs



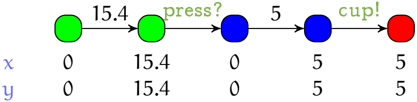
- Possible concrete runs for the coffee machine
  - Coffee with no sugar

x	0	15.4	0	5
y	0	15.4	0	5

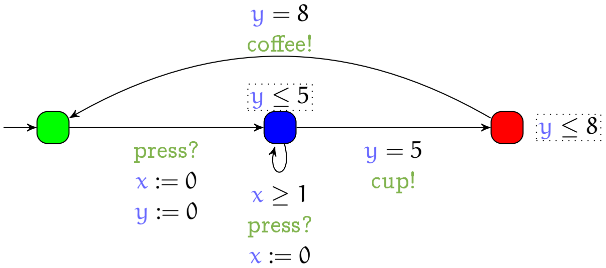
Examples of concrete runs



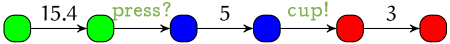
- Possible concrete runs for the coffee machine
  - Coffee with no sugar



Examples of concrete runs

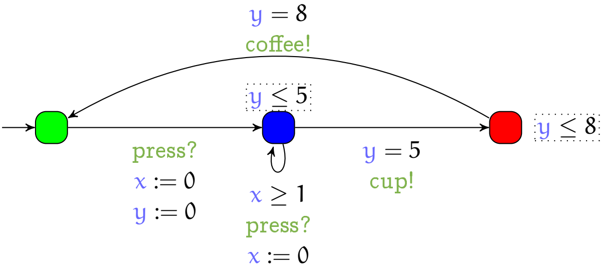


- Possible concrete runs for the coffee machine
  - Coffee with no sugar

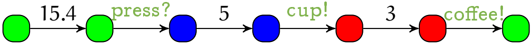


x	0	15.4	0	5	5	8
y	0	15.4	0	5	5	8

Examples of concrete runs

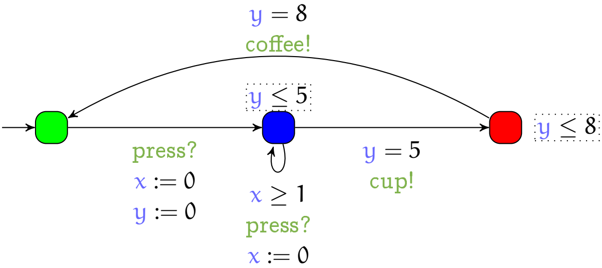


- Possible concrete runs for the coffee machine
  - Coffee with no sugar

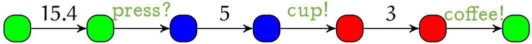


x	0	15.4	0	5	5	8	8
y	0	15.4	0	5	5	8	8

Examples of concrete runs



- Possible concrete runs for the coffee machine
  - Coffee with no sugar

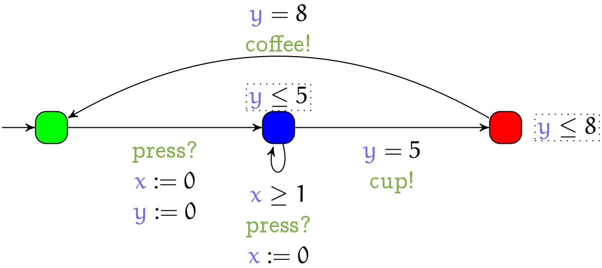


x	0	15.4	0	5	5	8	8
y	0	15.4	0	5	5	8	8

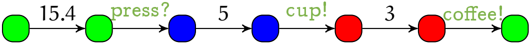
  - Coffee with 2 doses of sugar

x	0
y	0

Examples of concrete runs



- Possible concrete runs for the coffee machine
  - Coffee with no sugar

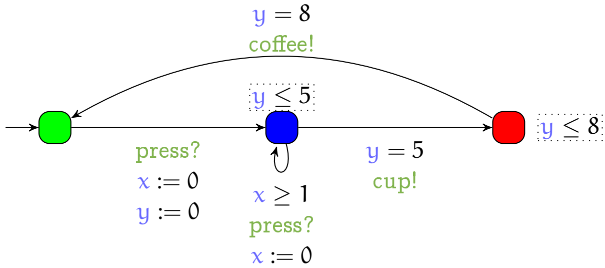


x	0	15.4	0	5	5	8	8
y	0	15.4	0	5	5	8	8

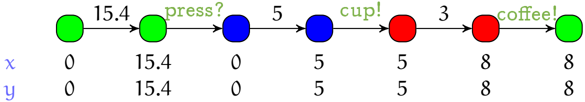
  - Coffee with 2 doses of sugar

x	0	0
y	0	0

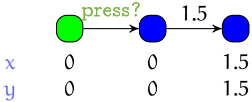
Examples of concrete runs



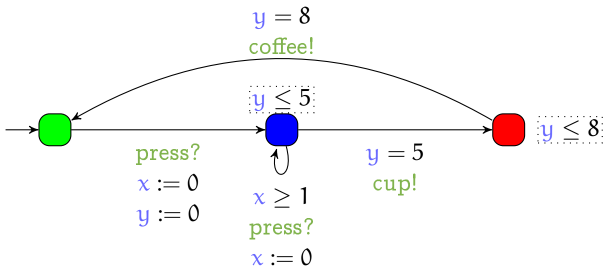
- Possible concrete runs for the coffee machine
  - Coffee with no sugar



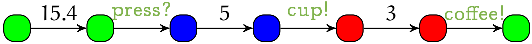
  - Coffee with 2 doses of sugar



Examples of concrete runs

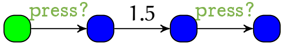


- Possible concrete runs for the coffee machine
  - Coffee with no sugar



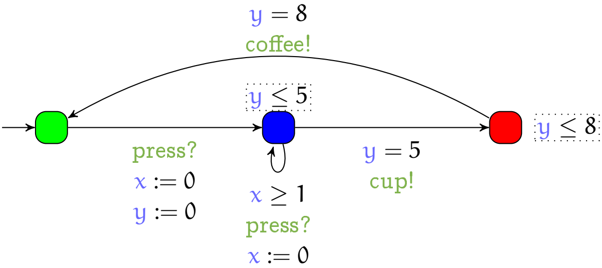
x	0	15.4	0	5	5	8	8
y	0	15.4	0	5	5	8	8

  - Coffee with 2 doses of sugar

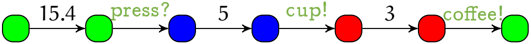


x	0	0	1.5	0
y	0	0	1.5	1.5

Examples of concrete runs

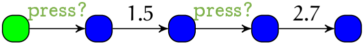


- Possible concrete runs for the coffee machine
  - Coffee with no sugar



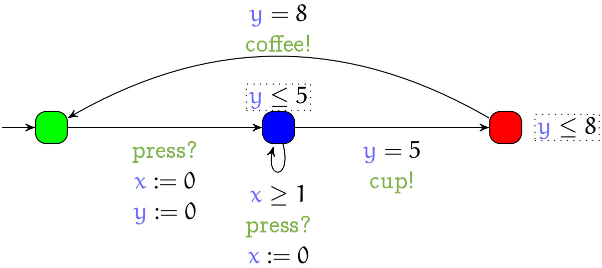
x	0	15.4	0	5	5	8	8
y	0	15.4	0	5	5	8	8

  - Coffee with 2 doses of sugar

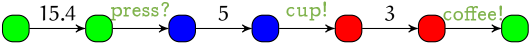


x	0	0	1.5	0	2.7
y	0	0	1.5	1.5	4.2

Examples of concrete runs



- Possible concrete runs for the coffee machine
  - Coffee with no sugar

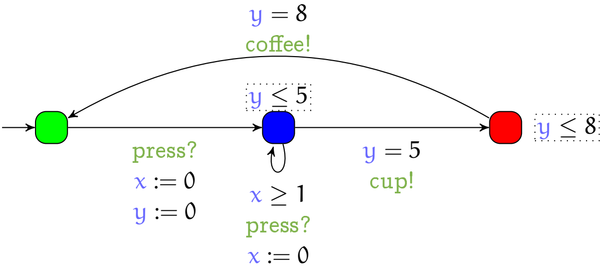


x	0	15.4	0	5	5	8	8
y	0	15.4	0	5	5	8	8

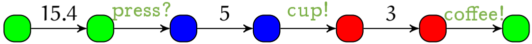
  - Coffee with 2 doses of sugar

x	0	0	1.5	0	2.7	0
y	0	0	1.5	1.5	4.2	4.2

Examples of concrete runs

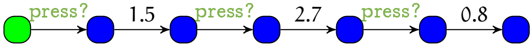


- Possible concrete runs for the coffee machine
  - Coffee with no sugar



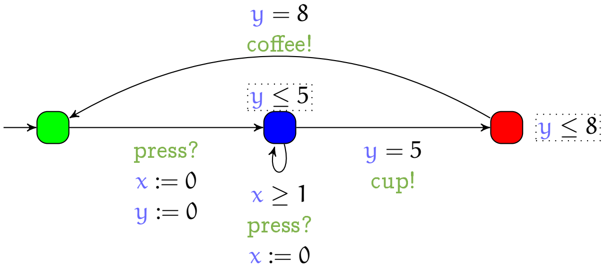
x	0	15.4	0	5	5	8	8
y	0	15.4	0	5	5	8	8

  - Coffee with 2 doses of sugar

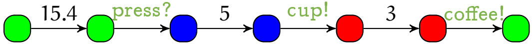


x	0	0	1.5	0	2.7	0	0.8
y	0	0	1.5	1.5	4.2	4.2	5

Examples of concrete runs

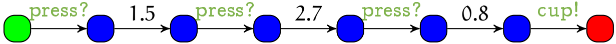


- Possible concrete runs for the coffee machine
  - Coffee with no sugar



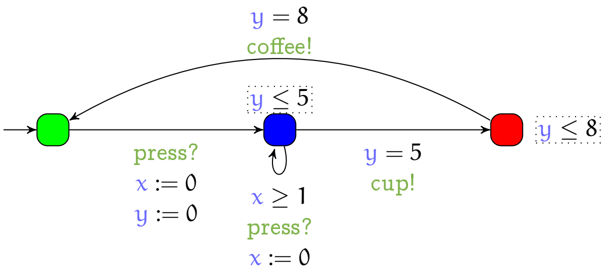
x	0	15.4	0	5	5	8	8
y	0	15.4	0	5	5	8	8

  - Coffee with 2 doses of sugar

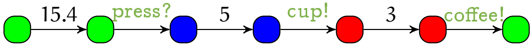


x	0	0	1.5	0	2.7	0	0.8	0.8
y	0	0	1.5	1.5	4.2	4.2	5	5

Examples of concrete runs

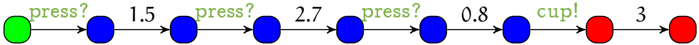


- Possible concrete runs for the coffee machine
  - Coffee with no sugar



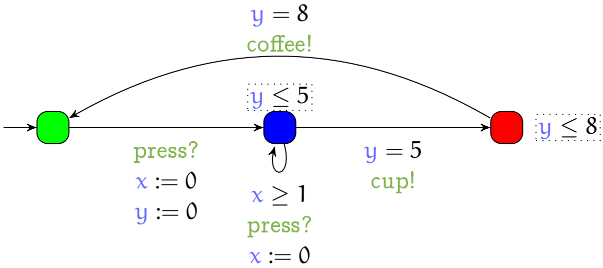
x	0	15.4	0	5	5	8	8
y	0	15.4	0	5	5	8	8

  - Coffee with 2 doses of sugar

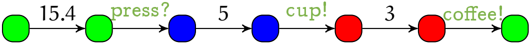


x	0	0	1.5	0	2.7	0	0.8	0.8	3.8
y	0	0	1.5	1.5	4.2	4.2	5	5	8

Examples of concrete runs



- Possible concrete runs for the coffee machine
  - Coffee with no sugar



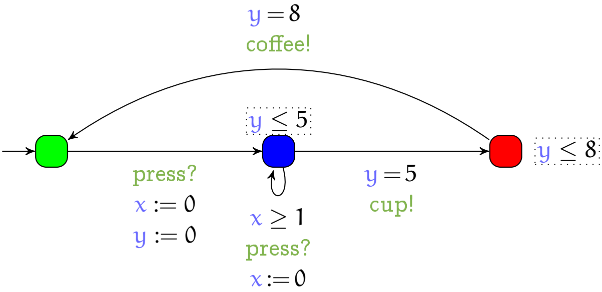
x	0	15.4	0	5	5	8	8
y	0	15.4	0	5	5	8	8

  - Coffee with 2 doses of sugar

x	0	0	1.5	0	2.7	0	0.8	0.8	3.8	3.8
y	0	0	1.5	1.5	4.2	4.2	5	5	8	8

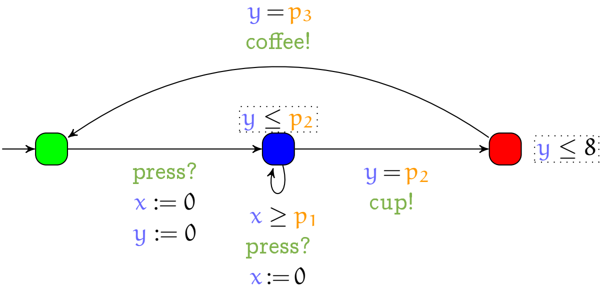
Parametric timed automaton (PTA)

- Timed automaton (sets of locations, actions and clocks)



Parametric timed automaton (PTA)

- Timed automaton (sets of locations, actions and clocks) augmented with a set P of parameters [Alur et al., 1993]
  - Unknown constants used in guards and invariants

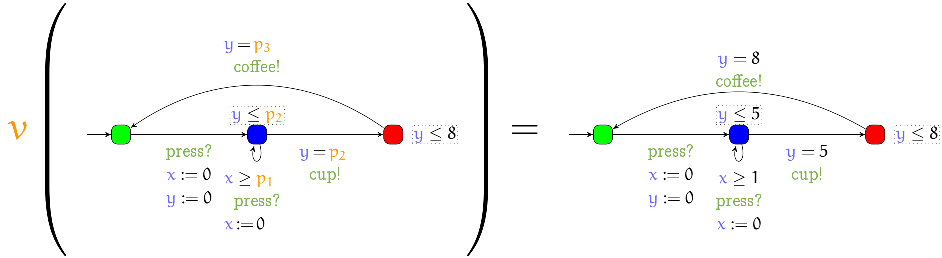


Valuation of a PTA

- Given a PTA $\mathcal{A}$ and a parameter valuation v, we denote by $v(\mathcal{A})$ the (non-parametric) timed automaton where all parameters are valuated by v

Valuation of a PTA

- Given a PTA $\mathcal{A}$ and a parameter valuation v, we denote by $v(\mathcal{A})$ the (non-parametric) timed automaton where all parameters are valuated by v



$$\text{with } v : \begin{cases} p_1 & \rightarrow & 1 \\ p_2 & \rightarrow & 5 \\ p_3 & \rightarrow & 8 \end{cases}$$

Integers or rationals?

In PTAs, both the clocks and the parameters can be either integer-valued or rational-valued.

This gives three possibilities:

	Clocks	Parameters
Discrete time	$\mathbb{N}$	$\mathbb{N}$
Dense time	$\mathbb{R}^+$	$\mathbb{N}$
Dense time	$\mathbb{R}^+$	$\mathbb{Q}^+$

Integers or rationals?

In PTAs, both the clocks and the parameters can be either integer-valued or rational-valued.
This gives three possibilities:

	Clocks	Parameters
Discrete time	$\mathbb{N}$	$\mathbb{N}$
Dense time	$\mathbb{R}^+$	$\mathbb{N}$
Dense time	$\mathbb{R}^+$	$\mathbb{Q}^+$

. . . and this has an impact on decidability

[Alur et al., 1993, Miller, 2000, Beneš et al., 2015]

Integers or rationals?

In PTAs, both the clocks and the parameters can be either integer-valued or rational-valued.
This gives three possibilities:

	Clocks	Parameters
Discrete time	$\mathbb{N}$	$\mathbb{N}$
Dense time	$\mathbb{R}^+$	$\mathbb{N}$
Dense time	$\mathbb{R}^+$	$\mathbb{Q}^+$

... and this has an impact on decidability

[Alur et al., 1993, Miller, 2000, Beneš et al., 2015]

Here: we consider (mainly) dense time with integer-valued parameters

Outline

The question of the syntax

Almost each work in the literature defines its own syntax:

- guards
- invariants
- accepting locations or not

Are these definitions equivalently expressive?

The question of the syntax

Almost each work in the literature defines its own syntax:

- guards
- invariants
- accepting locations or not

Are these definitions equivalently expressive?

Integer vs. rational valued parameters

- For a rational valued PTA, can we find an integer-valued PTA with the same expressiveness?

Subclasses of PTAs

Several subclasses were defined for PTAs, e. g.,:

- Lower-bound upper-bound PTAs

  [Hune et al., 2002, Bozzelli and La Torre, 2009]

- Bounded PTAs (i. e., with a bounded parameter domain)

Are these subclasses less expressive than PTAs?

Subclasses of PTAs

Several subclasses were defined for PTAs, e. g.,:

- Lower-bound upper-bound PTAs

  [Hune et al., 2002, Bozzelli and La Torre, 2009]

- Bounded PTAs (i. e., with a bounded parameter domain)

Are these subclasses less expressive than PTAs?

And are PTAs any more expressive than TAs. . . ?

Towards an expressiveness for PTAs

Problem

What is the expressiveness of a PTA?

Towards an expressiveness for PTAs

Problem

What is the expressiveness of a PTA?

Goal

Propose a definition of expressiveness for PTAs, and compare the syntax / subclasses of PTAs.

Outline

Definition (PTA with hidden parameters)

A parametric timed automaton with hidden parameters (hereafter hPTA) $\mathcal{A}$ is a tuple $(\Sigma, L, l_0, F, X, P, I, E)$, where:

1. Σ is a finite set of actions,
2. L is a finite set of locations,
3. $l_0 \in L$ is the initial location,
4. $F \subseteq L$ is a set of accepting locations,
5. X is a finite set of clocks,
6. $P = P_{\overline{v}} \uplus P_v$ is a finite set of parameters partitioned into hidden parameters $P_{\overline{v}}$ and visible parameters P_v,
7. I is the invariant, assigning to every $l \in L$ a guard $I(l)$,
8. E is a finite set of edges $e = (l, g, a, R, l')$ where $l, l' \in L$ are the source and target locations, $a \in \Sigma \cup \{\epsilon\}$ (ϵ: silent action), $R \subseteq X$ is a set of clocks to be reset, and g is a guard.

Lower-bound / Upper-bound PTAs

Definition (hL/U-PTA [Hune et al., 2002, Bozzelli and La Torre, 2009])

An hL/U-PTA is an hPTA where the set of parameters is partitioned into:

1. a set of lower-bound parameters P^- where each parameter can only be compared to a clock as a lower bound ("$p \leq x$") and,
2. a set of upper-bound parameters P^+ ("$x \leq p$").

L/U-PTAs (without hidden parameters) benefit from some decidability properties, in contrast to general PTAs

[Hune et al., 2002, Bozzelli and La Torre, 2009]

A first definition of expressiveness

Definition (untimed language of an hPTA)

Given an hPTA $\mathcal{A}$, the untimed language $\mathsf{UL}(\mathcal{A})$ of $\mathcal{A}$ is

$$\bigcup_{v \in \mathcal{V}(\mathsf{P})} \Big\{ w \mid w \text{ is an untimed word accepted by } v(\mathcal{A}) \Big\}$$

Untimed word accepted by a TA: sequence of actions associated with a finite (timed) run ending in an accepting location

A second definition of expressiveness

Definition (constrained untimed language of an hPTA)

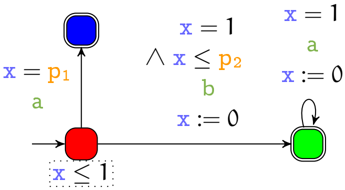
Given an hPTA $\mathcal{A}$, the constrained untimed language $\mathsf{CUL}(\mathcal{A})$ of $\mathcal{A}$ is

$$\bigcup_{v \in \mathcal{V}(\mathsf{P}_v)} \Big\{ (w, v) \mid \exists \bar{v} \in \mathcal{V}(\mathsf{P}_{\bar{v}}) \text{ s.t. } w \text{ is an untimed word of } v(\bar{v}(\mathcal{A})) \Big\}$$

Remarks:

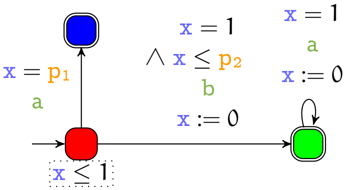
- Only the visible parameters are considered
- Constrained because an alternative way to represent $\mathsf{CUL}(\mathcal{A})$ is a set of pairs (w, K) (where K denotes all valuations accepting w)
- Only PTAs with the same number of visible parameters can be compared using CUL

Example



$\mathsf{UL}(\mathcal{A}) = \{a\} \cup \{ba^n \mid n \in \mathbb{N}\}$ (also written $\mathsf{UL}(\mathcal{A}) = a + ba^*$)

Example



$\mathsf{UL}(\mathcal{A}) = \{a\} \cup \{ba^n \mid n \in \mathbb{N}\}$ (also written $\mathsf{UL}(\mathcal{A}) = a + ba^*$)

$$\mathsf{CUL}(\mathcal{A}) = \Big\{(a, p_1 = i) \mid 0 \leq i \leq 1\Big\} \bigcup \Big\{(ba^n, p_1 = i) \mid i \in \mathbb{N}, n \in \mathbb{N}\Big\}$$

(also written $\mathsf{CUL}(\mathcal{A}) = \Big\{(a, p_1 \leq 1), (ba^*, p_1 \geq 0)\Big\}$, with $p_1 \in \mathbb{N}$)

Note: both the parameter p_2 and the fact that p_2 must be at least 1 to go to  *are hidden.*

Outline

Integers vs. Rationals

Proposition (Rationals and integers are equivalent for UL)

PTAs with rational parameters and PTAs with unbounded integer parameters are equivalent with respect to the untimed language.

Integers vs. Rationals

Proposition (Rationals and integers are equivalent for UL)

PTAs with rational parameters and PTAs with unbounded integer parameters are equivalent with respect to the untimed language.

Proof idea

- Any integer-valued PTA can be simulated by a rational-valued PTA using appropriate gadgets
- A rational-valued PTA can be simulated by an integer-valued by scaling up all expressions: since we do not know by how much we need to scale, we add... a fresh parameter.

  For example $x \leq 3p_1 + 2p_2 + 7$ becomes $x \leq 3p_1 + 2p_2 + 7p$.

Integers vs. Rationals

Proposition (Rationals and integers are equivalent for UL)

PTAs with rational parameters and PTAs with unbounded integer parameters are equivalent with respect to the untimed language.

Proof idea

- Any integer-valued PTA can be simulated by a rational-valued PTA using appropriate gadgets
- A rational-valued PTA can be simulated by an integer-valued by scaling up all expressions: since we do not know by how much we need to scale, we add... a fresh parameter.

  For example $x \leq 3p_1 + 2p_2 + 7$ becomes $x \leq 3p_1 + 2p_2 + 7p$.

A similar result is shown for L/U-PTAs *[see paper]*

Outline

General PTAs: type 0

Lemma

Turing-recognizable languages are also recognizable by PTAs.

General PTAs: type 0

Lemma

Turing-recognizable languages are also recognizable by PTAs.

Proof idea

- all proofs of undecidabiliy for PTAs reduce from the halting problem of a 2-counter machine.

  e. g., [Alur et al., 1993]

- 2-counter machines are Turing equivalent

Single-clock PTAs: type 3

Lemma

The untimed language recognized by a PTA with a single clock (and arbitrarily many rational-valued parameters) is regular.

Single-clock PTAs: type 3

Lemma

The untimed language recognized by a PTA with a single clock (and arbitrarily many rational-valued parameters) is regular.

Proof idea

From the fact that the parametric zone graph of a PTA with a single (necessarily parametric) clock and arbitrarily many parameters is finite.

[A. and Markey, 2015, Theorem 20]

Two clocks: at least type 1

Theorem

PTAs with 1 parametric clock, 1 non-parametric clock and 1 parameter can recognize languages that are context-sensitive.

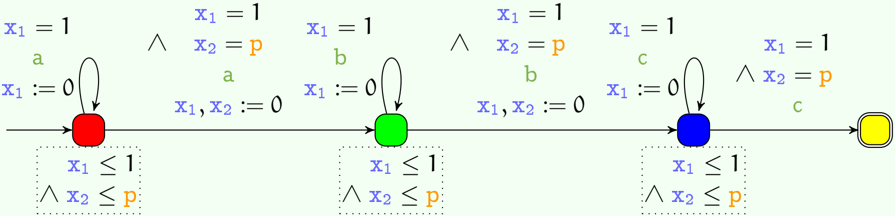
Two clocks: at least type 1

Theorem

PTAs with 1 parametric clock, 1 non-parametric clock and 1 parameter can recognize languages that are context-sensitive.

Proof idea

The following PTA recognizes $a^n b^n c^n$



Outline

TAs = L/U-PTAs

Proposition

Most subclasses of PTAs are not more expressive than timed automata w.r.t. the untimed language.

TAs = L/U-PTAs

Proposition

Most subclasses of PTAs are not more expressive than timed automata w.r.t. the untimed language.

Proof idea

- L/U-PTAs: union over all parameter valuations equivalent to the TA replacing lower-bound parameters with 0 and upper-bound parameter with a sufficiently large constant

[Bozzelli and La Torre, 2009]

TAs = L/U-PTAs = bounded PTAs

Proposition

Most subclasses of PTAs are not more expressive than timed automata w.r.t. the untimed language.

Proof idea

- L/U-PTAs: union over all parameter valuations equivalent to the TA replacing lower-bound parameters with 0 and upper-bound parameter with a sufficiently large constant

  [Bozzelli and La Torre, 2009]

- integer-valued bounded PTAs: from the closure of regular languages

TAs < PTAs

Proposition

PTAs are strictly more expressive than TAs w.r.t. the untimed language.

Proof idea

From the results on the hierarchy of Chomsky.

PTAs = fcp-PTAs

Proposition

Fully parametric constraints PTAs (fpc-PTAs), i. e., allowing parametric linear terms ($3p_1 - p_2 > 5p_3$) in guards, are not more expressive than PTAs w.r.t. the untimed language.

Proof idea

1. By translation to an equivalent PTA w.r.t. the untimed language, that follows the most restrictive syntax (that of [Alur et al., 1993])

PTAs = fcp-PTAs = hPTAs

Proposition

Fully parametric constraints PTAs (fpc-PTAs), i. e., allowing parametric linear terms ($3p_1 - p_2 > 5p_3$) in guards, are not more expressive than PTAs w.r.t. the untimed language.
hPTAs are equally expressive with PTAs w.r.t. the untimed language.

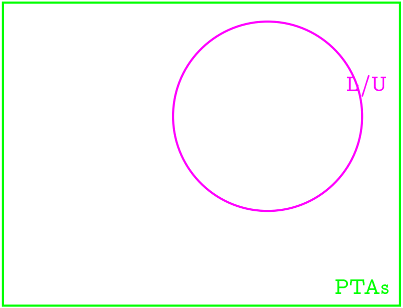
Proof idea

1. By translation to an equivalent PTA w.r.t. the untimed language, that follows the most restrictive syntax (that of [Alur et al., 1993])
2. By "de-hiding" parameters

Outline

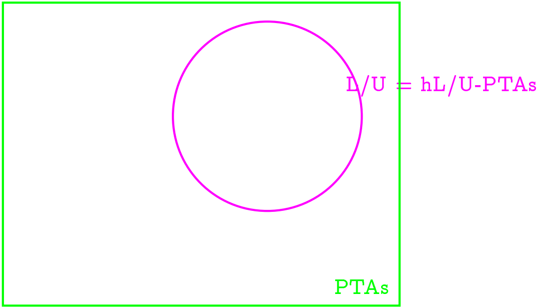
5 Comparison of the Expressiveness
- PTAs in the Chomsky hierarchy
- Comparison w.r.t. the untimed language
- Comparison w.r.t. the constrained untimed language

Expressiveness w.r.t. the CUL



Main results:
L/U-PTAs $<$ PTAs

Expressiveness w.r.t. the CUL

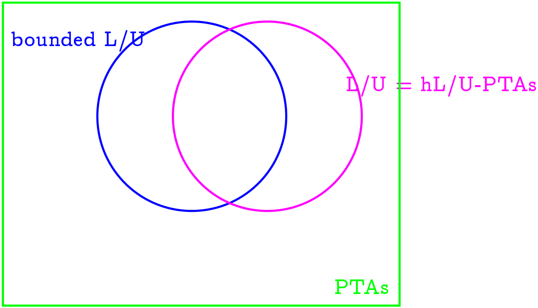


Main results:

L/U-PTAs $<$ PTAs

hL/U-PTAs $=$ L/U-PTAs

Expressiveness w.r.t. the CUL



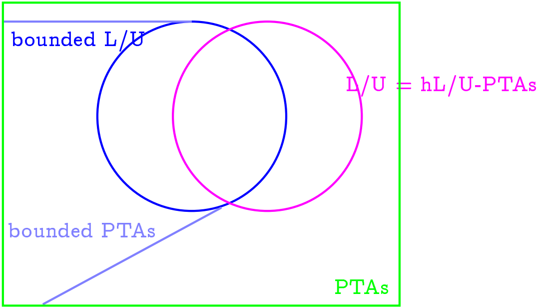
Main results:

L/U-PTAs < PTAs

hL/U-PTAs = L/U-PTAs

bL/U-PTA incomp. with L/U

Expressiveness w.r.t. the CUL



Main results:

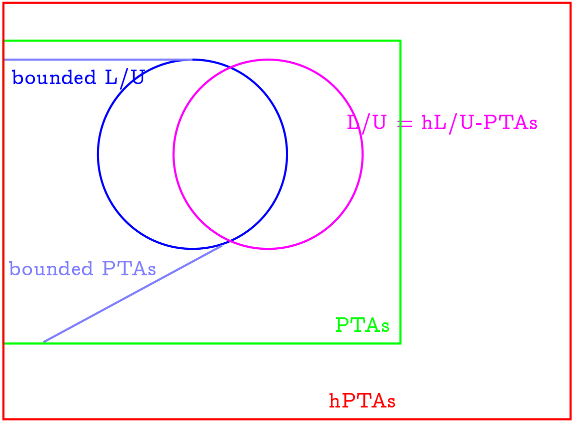
L/U-PTAs < PTAs

hL/U-PTAs = L/U-PTAs

bL/U-PTA incomp. with L/U

bounded PTAs < PTAs

Expressiveness w.r.t. the CUL



Main results: *(more in paper)*

- L/U-PTAs < PTAs
- hL/U-PTAs = L/U-PTAs
- bL/U-PTA incomp. with L/U
- bounded PTAs < PTAs
- hPTAs > PTAs

Outline

Summary

- First attempt to define the expressiveness of parametric timed automata and subclasses
  - Including by hiding parameters
- Untimed language: most formalisms are not more expressive than timed automata
- Constrained untimed language: proposed a first classification of the main subclasses of PTAs

Perspectives

Consider the timed language and constrained timed language as well

- (But in fact, results do not seem to differ from their untimed counterparts)

The expressiveness of two-clock PTAs remains open

- And close to well-known open decision problems

Power of silent transitions in our definitions?

Relationship between the guard syntax and the number of clocks and parameters

- Is using a syntax "$x \sim p + c$" equivalent to a syntax "$x \sim p \mid c$" at the cost of one extra clock in the PTA?

Bibliography

References I

Alur, R. and Dill, D. L. (1994).
A theory of timed automata.
Theoretical Computer Science, 126(2):183–235.

Alur, R., Henzinger, T. A., and Vardi, M. Y. (1993).
Parametric real-time reasoning.
In *STOC*, pages 592–601. ACM.

André, É. and Markey, N. (2015).
Language preservation problems in parametric timed automata.
In *FORMATS*, volume 9268 of *Lecture Notes in Computer Science*, pages 27–43. Springer.

Beneš, N., Bezděk, P., Larsen, K. G., and Srba, J. (2015).
Language emptiness of continuous-time parametric timed automata.
In *ICALP, Part II*, volume 9135 of *Lecture Notes in Computer Science*, pages 69–81. Springer.

Bozzelli, L. and La Torre, S. (2009).
Decision problems for lower/upper bound parametric timed automata.
Formal Methods in System Design, 35(2):121–151.

References II

Henzinger, T. A., Nicollin, X., Sifakis, J., and Yovine, S. (1994).
Symbolic model checking for real-time systems.
Information and Computation, 111(2):193–244.

Hune, T., Romijn, J., Stoelinga, M., and Vaandrager, F. W. (2002).
Linear parametric model checking of timed automata.
Journal of Logic and Algebraic Programming, 52-53:183–220.

Miller, J. S. (2000).
Decidability and complexity results for timed automata and semi-linear hybrid automata.
In *HSCC*, volume 1790 of *Lecture Notes in Computer Science*, pages 296–309. Springer.

Licensing

Source of the graphics used I

Title: Smiley green alien big eyes (aaah)
Author: LadyofHats
Source: https://commons.wikimedia.org/wiki/File:Smiley_green_alien_big_eyes.svg
License: public domain

Title: Smiley green alien big eyes (cry)
Author: LadyofHats
Source: https://commons.wikimedia.org/wiki/File:Smiley_green_alien_big_eyes.svg
License: public domain

License of this document