ICFEM 2016

Friday 18th November, 2016
東京，日本

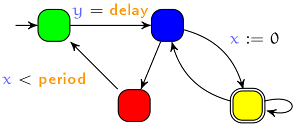
Decision Problems for Parametric Timed Automata

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



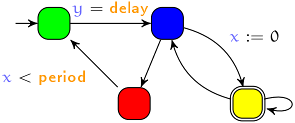
A model of the system


is unreachable

A property to be satisfied

Context: timed model checking

- Timed model checking



$\overset{?}{\models}$

 is unreachable

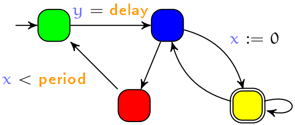
A model of the system

A property to be satisfied

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

Context: timed model checking

- Timed model checking



$\stackrel{?}{\models}$

 is unreachable

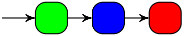
A model of the system

A property to be satisfied

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

Yes

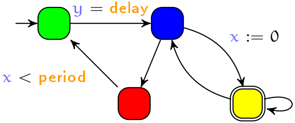
No



Counterexample

Context: parametric timed model checking

- Timed model checking



$\overset{?}{\models}$

 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...

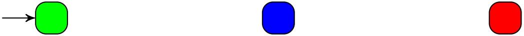
$$2\text{delay} > \text{period} \land \text{period} < 20.46$$

Outline

Outline

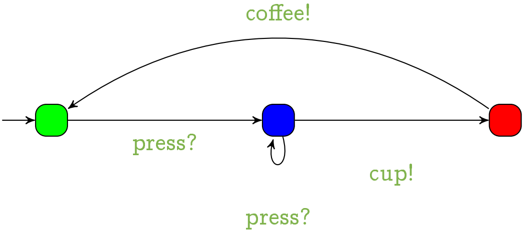
Timed automaton (TA)

- Finite state automaton (sets of locations)



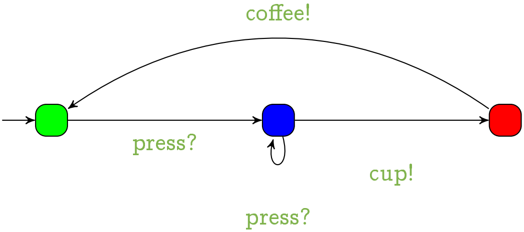
Timed automaton (TA)

- Finite state automaton (sets of locations and actions)



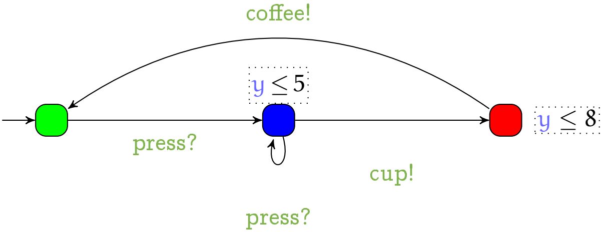
Timed automaton (TA)

- 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



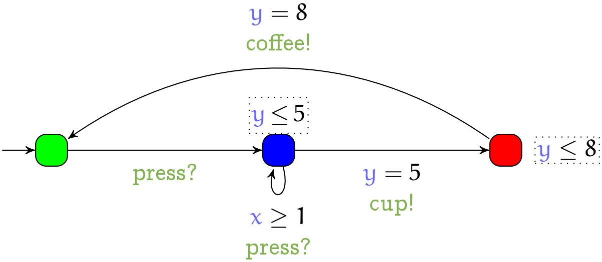
Timed automaton (TA)

- 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)

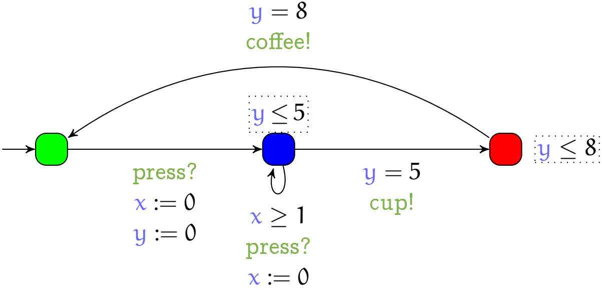
- Finite state automaton (sets of locations and actions) augmented with a set X of clocks [Alur and Dill, 1994, Henzinger et al., 1994]
  - Real-valued variables evolving linearly at the same rate
- Features
  - Location invariant: constraint to be verified to stay at a location
  - Transition guard: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



Timed automaton (TA)

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

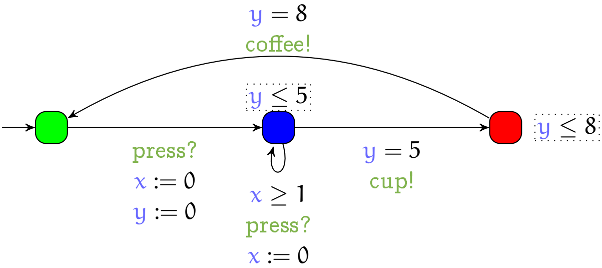
- Features
  - Location invariant: constraint to be verified to stay at a location
  - Transition guard: constraint to be verified to enable a transition
  - Clock reset: some of the clocks can be set to 0 at each transition



Concrete semantics of timed automata

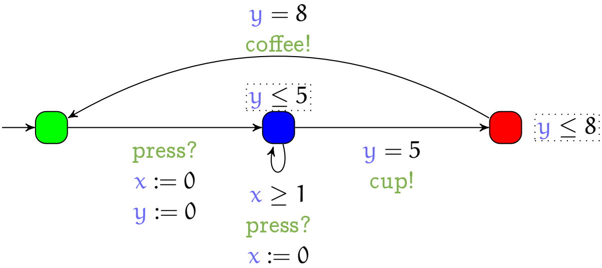
- Concrete state of a TA: pair (l, w), where
  - l is a location,
  - w is a valuation of each clock
- Concrete run: alternating sequence of concrete states and actions or time elapse

Examples of concrete runs

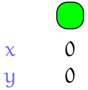


- Possible concrete runs for the coffee machine

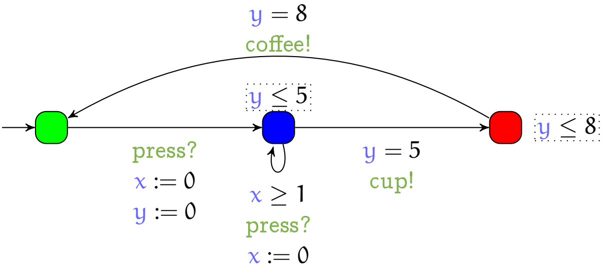
Examples of concrete runs



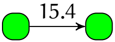
- Possible concrete runs for the coffee machine
  - Coffee with no sugar



Examples of concrete runs

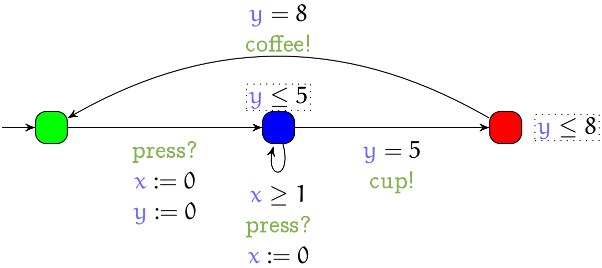


- Possible concrete runs for the coffee machine
  - Coffee with no sugar

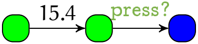


x	0	15.4
y	0	15.4

Examples of concrete runs

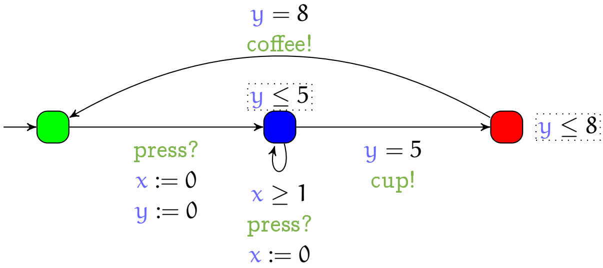


- Possible concrete runs for the coffee machine
  - Coffee with no sugar



x	0	15.4	0
y	0	15.4	0

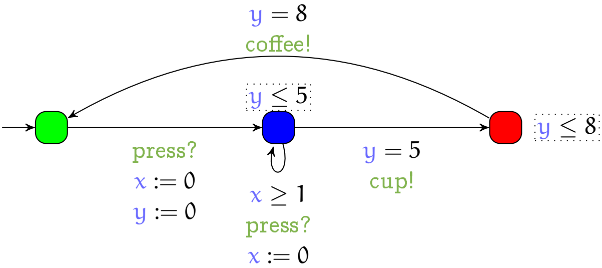
Examples of concrete runs



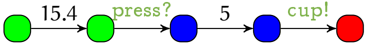
- Possible concrete runs for the coffee machine
  - Coffee with no sugar

x	0	15.4	0	5
y	0	15.4	0	5

Examples of concrete runs

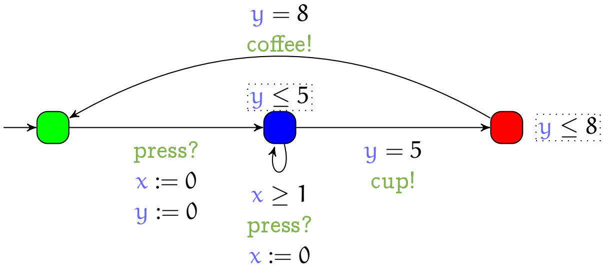


- Possible concrete runs for the coffee machine
  - Coffee with no sugar

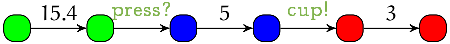


x	0	15.4	0	5	5
y	0	15.4	0	5	5

Examples of concrete runs

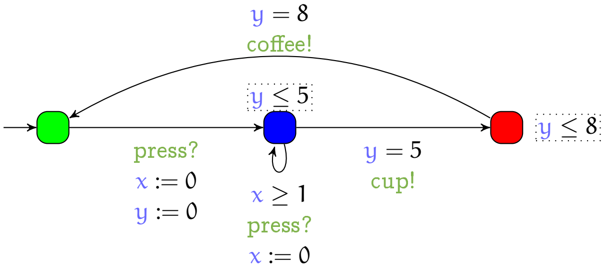


- Possible concrete runs for the coffee machine
  - Coffee with no sugar

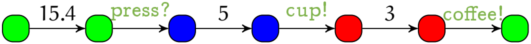


x	0	15.4	0	5	5	8
y	0	15.4	0	5	5	8

Examples of concrete runs

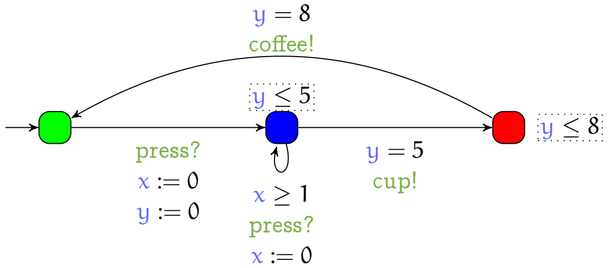


- Possible concrete runs for the coffee machine
  - Coffee with no sugar

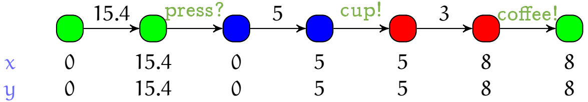


x	0	15.4	0	5	5	8	8
y	0	15.4	0	5	5	8	8

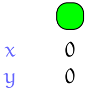
Examples of concrete runs



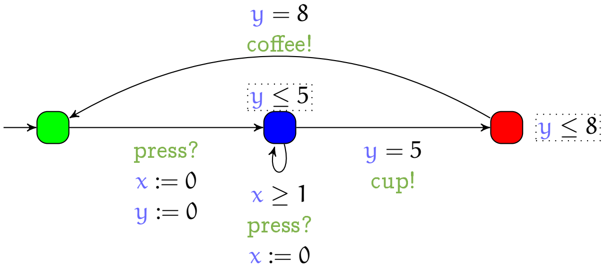
- Possible concrete runs for the coffee machine
  - Coffee with no sugar



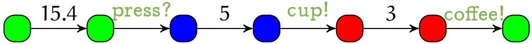
  - Coffee with 2 doses of sugar



Examples of concrete runs



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

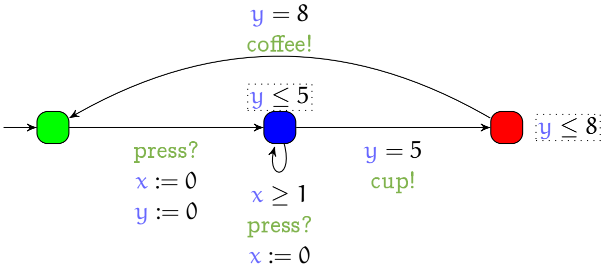


x	0	15.4	0	5	5	8	8
y	0	15.4	0	5	5	8	8

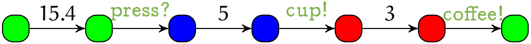
  - Coffee with 2 doses of sugar

x	0	0
y	0	0

Examples of concrete runs

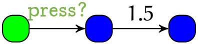


- Possible concrete runs for the coffee machine
  - Coffee with no sugar



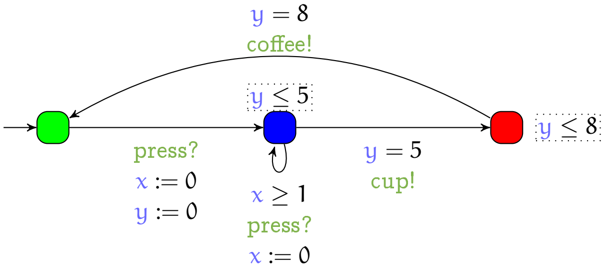
x	0	15.4	0	5	5	8	8
y	0	15.4	0	5	5	8	8

  - Coffee with 2 doses of sugar

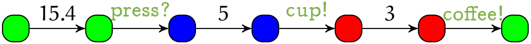


x	0	0	1.5
y	0	0	1.5

Examples of concrete runs

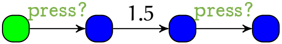


- Possible concrete runs for the coffee machine
  - Coffee with no sugar



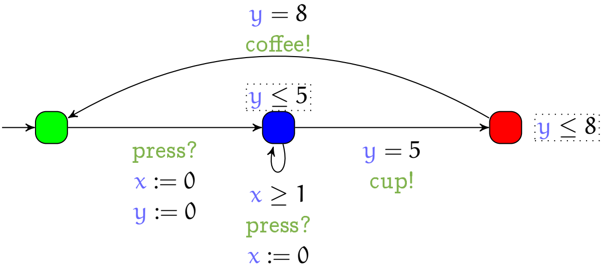
x	0	15.4	0	5	5	8	8
y	0	15.4	0	5	5	8	8

  - Coffee with 2 doses of sugar

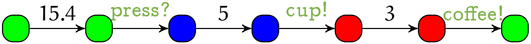


x	0	0	1.5	0
y	0	0	1.5	1.5

Examples of concrete runs

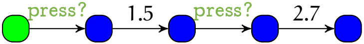


- Possible concrete runs for the coffee machine
  - Coffee with no sugar



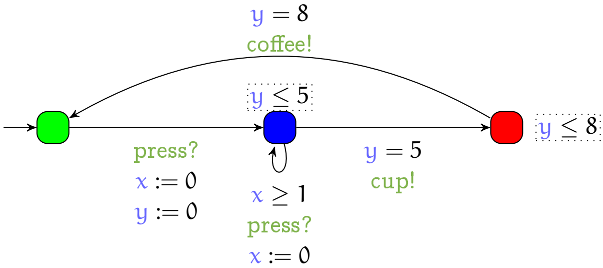
x	0	15.4	0	5	5	8	8
y	0	15.4	0	5	5	8	8

  - Coffee with 2 doses of sugar

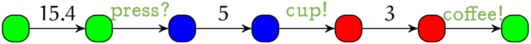


x	0	0	1.5	0	2.7
y	0	0	1.5	1.5	4.2

Examples of concrete runs



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

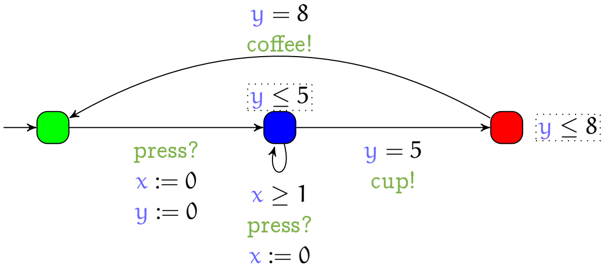


x	0	15.4	0	5	5	8	8
y	0	15.4	0	5	5	8	8

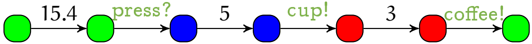
  - Coffee with 2 doses of sugar

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

Examples of concrete runs

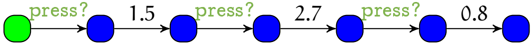


- Possible concrete runs for the coffee machine
  - Coffee with no sugar



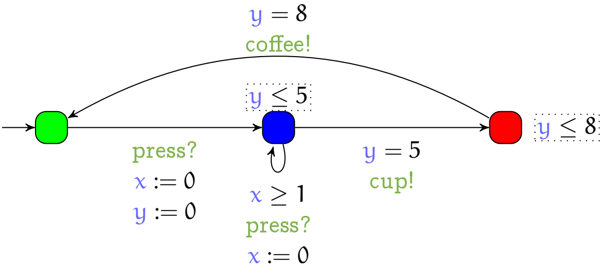
x	0	15.4	0	5	5	8	8
y	0	15.4	0	5	5	8	8

  - Coffee with 2 doses of sugar

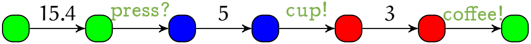


x	0	0	1.5	0	2.7	0	0.8
y	0	0	1.5	1.5	4.2	4.2	5

Examples of concrete runs

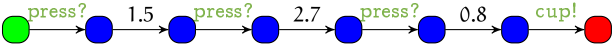


- Possible concrete runs for the coffee machine
  - Coffee with no sugar



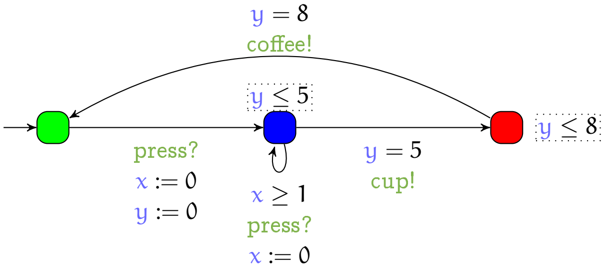
x	0	15.4	0	5	5	8	8
y	0	15.4	0	5	5	8	8

  - Coffee with 2 doses of sugar

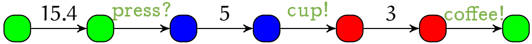


x	0	0	1.5	0	2.7	0	0.8	0.8
y	0	0	1.5	1.5	4.2	4.2	5	5

Examples of concrete runs

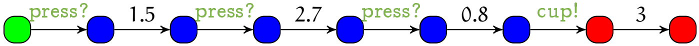


- Possible concrete runs for the coffee machine
  - Coffee with no sugar



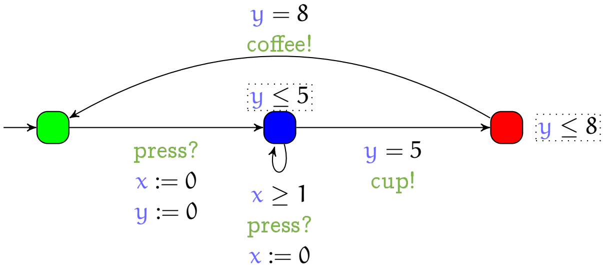
x	0	15.4	0	5	5	8	8
y	0	15.4	0	5	5	8	8

  - Coffee with 2 doses of sugar

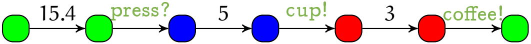


x	0	0	1.5	0	2.7	0	0.8	0.8	3.8
y	0	0	1.5	1.5	4.2	4.2	5	5	8

Examples of concrete runs



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



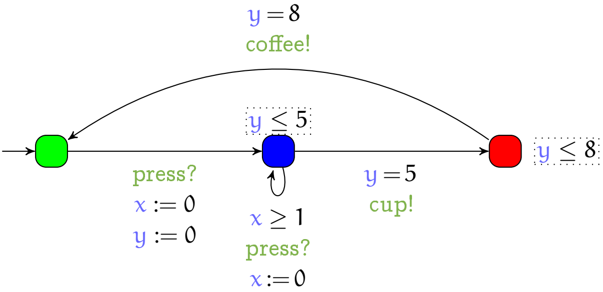
x	0	15.4	0	5	5	8	8
y	0	15.4	0	5	5	8	8

  - 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

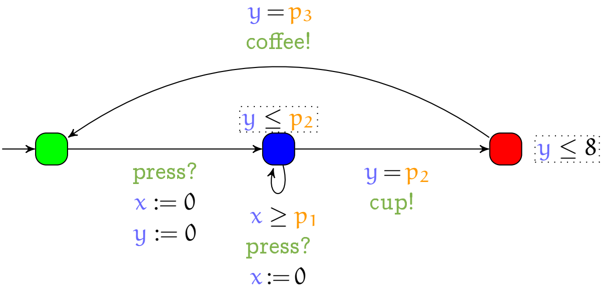
Parametric timed automaton (PTA)

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



Parametric timed automaton (PTA)

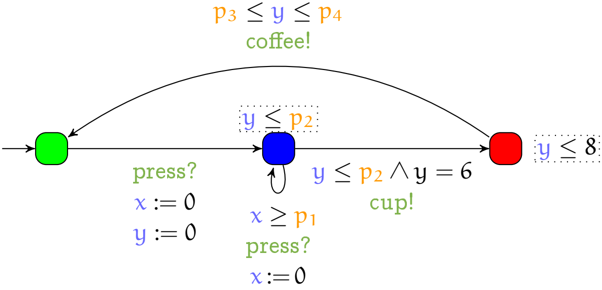
- Timed automaton (sets of locations, actions and clocks) augmented with a set P of parameters [Alur et al., 1993]
  - Unknown constants used in guards and invariants



L/U-PTAs

Definition

A lower/upper bound PTA (L/U-PTA) is a PTA in which each parameter p is always compared with clocks as an upper bound or always as a lower bound. [Hune et al., 2002, Bozzelli and La Torre, 2009]



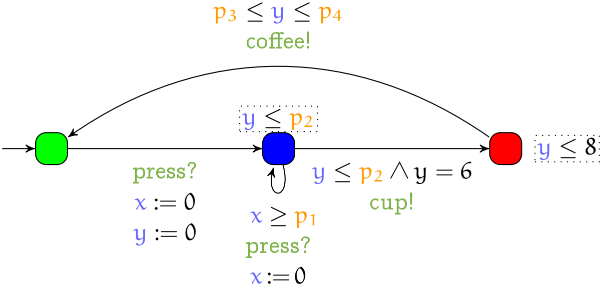
Lower-bound parameters:

Upped-bound parameters:

L/U-PTAs

Definition

A lower/upper bound PTA (L/U-PTA) is a PTA in which each parameter p is always compared with clocks as an upper bound or always as a lower bound. [Hune et al., 2002, Bozzelli and La Torre, 2009]



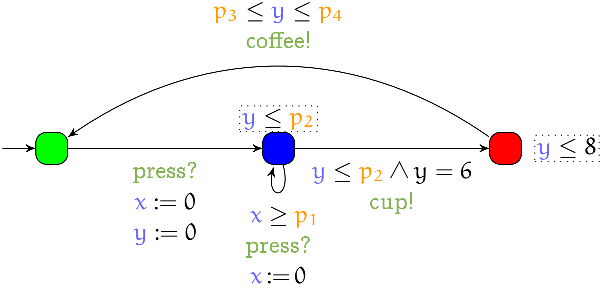
Lower-bound parameters: p_1, p_3

Upped-bound parameters:

L/U-PTAs

Definition

A lower/upper bound PTA (L/U-PTA) is a PTA in which each parameter p is always compared with clocks as an upper bound or always as a lower bound. [Hune et al., 2002, Bozzelli and La Torre, 2009]



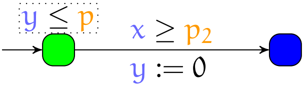
Lower-bound parameters: p_1, p_3

Upped-bound parameters: p_2, p_4

Symbolic semantics of PTAs

- Symbolic state $s = (l, Z)$: location + convex polyhedron constraining both clocks and parameters

  [Hune et al., 2002, André et al., 2009, Jovanović et al., 2015]
- Convex polyhedra obtained have a special form called parametric zone [Hune et al., 2002]



$$Z_0 = \begin{cases} & x = y \\ \wedge & 0 \leq y \leq p_1 \\ \wedge & p_1, p_2 \geq 0 \end{cases} \qquad Z_1 = \begin{cases} & p_2 \leq x - y \leq p_1 \\ \wedge & (p_2 \leq p_1) \\ \wedge & x, y, p_1, p_2 \geq 0 \end{cases}$$

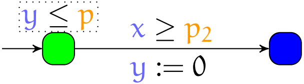
Symbolic semantics of PTAs

- Symbolic state $s = (l, Z)$: location + convex polyhedron constraining both clocks and parameters

  [Hune et al., 2002, André et al., 2009, Jovanović et al., 2015]

- Convex polyhedra obtained have a special form called parametric zone

  [Hune et al., 2002]



$$Z_0 = \begin{cases} & x = y \\ \wedge & 0 \leq y \leq p_1 \\ \wedge & p_1, p_2 \geq 0 \end{cases} \qquad Z_1 = \begin{cases} & p_2 \leq x - y \leq p_1 \\ \wedge & (p_2 \leq p_1) \\ \wedge & x, y, p_1, p_2 \geq 0 \end{cases}$$

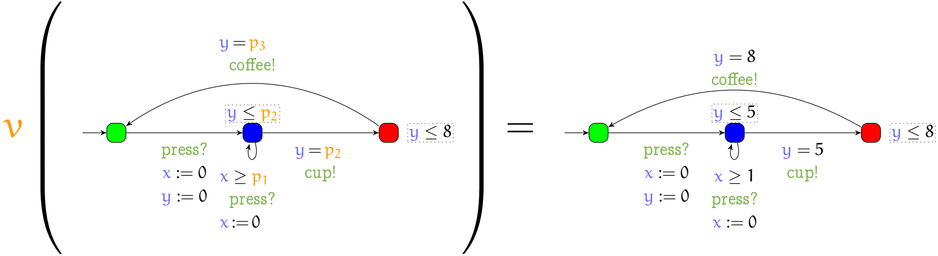
Note: there is a potentially infinite number of symbolic states

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



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

Outline

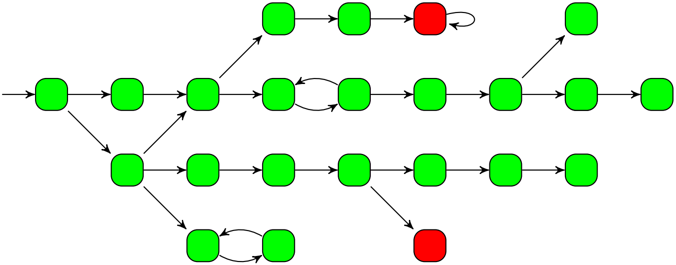
Decision problems (1/3)

Definition (reachability emptiness (EF-emptiness))

Input: a PTA $\mathcal{A}$ and a set of locations G

Problem: Is the set of parameter valuations v such that there exists a run of $v(\mathcal{A})$ reaching a location $l \in G$ empty?

$v(\mathcal{A})$:



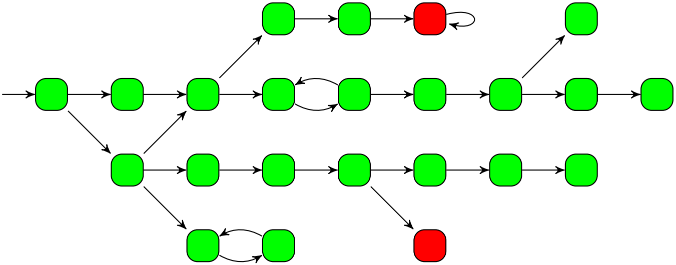
Decision problems (2/3)

Definition (reachability universality (EF-universality))

Input: a PTA $\mathcal{A}$ and a set of locations G

Problem: Are all parameter valuations v such that there exists a run of $v(\mathcal{A})$ reaching a location $l \in G$?

$v(\mathcal{A})$:



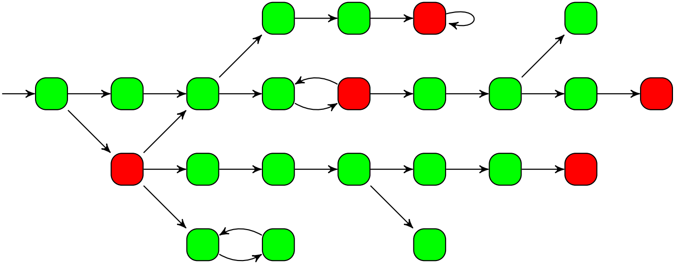
Decision problems (3/3)

Definition (unavoidability emptiness (AF-emptiness))

Input: a PTA $\mathcal{A}$ and a set of locations G

Problem: Is the set of parameter valuations v such that all runs of $v(\mathcal{A})$ eventually reach a location $l \in$ G empty?

$v(\mathcal{A})$:



Outline

EF-emptiness in the literature

EF-emptiness

Reachability emptiness ("is the set of parameter valuations reaching a given location l empty?") is undecidable for PTAs [Alur et al., 1993]

- even with a single parametric clock [Miller, 2000]
- even with a single real-valued parameter [Miller, 2000]
- even with only strict constraints [Doyen, 2007]
- even with a single integer-valued parameter [Beneš et al., 2015]

EF-emptiness in the literature

EF-emptiness

Reachability emptiness ("is the set of parameter valuations reaching a given location l empty?") is undecidable for PTAs [Alur et al., 1993]

- even with a single parametric clock [Miller, 2000]
- even with a single real-valued parameter [Miller, 2000]
- even with only strict constraints [Doyen, 2007]
- even with a single integer-valued parameter [Beneš et al., 2015]

Proof.

By reduction from the halting problem of a 2-counter machine, which is undecidable [Minsky, 1967] □

See [André, 2015] for an exhaustive survey

A new construction to prove the undecidability

Reduction from the halting problem of a 2-counter machine

Not a new theoretical result

Will be used for all subsequent undecidability proofs

Our new encoding of the 2-counter machine:

- Uses a single rational-valued parameter and three (parametric) clocks
- Matches the smallest known numbers of clocks and parameters for integer-valued parameters [Beneš et al., 2015] and rational-valued parameters [Miller, 2000]

2-counter machine

Finite program reading and modifying 2 non-negative integer counters C_1 and C_2

3 instructions

- when in state s_i, increment C_k and go to s_j;
- when in state s_i, decrement C_k and go to s_j;
- when in state s_i, if $C_k = 0$ then go to s_j, otherwise block.

Halting is undecidable [Minsky, 1967]

Encoding

Encoding the counters C_1 and C_2

- Three clocks x, y and z and one parameter a
- Let c_1, c_2 be the values of C_1 and C_2
- In any location s_i, when $x = 0$ we have
  - $y = 1 - ac_1$
  - $z = 1 - ac_2$

Encoding

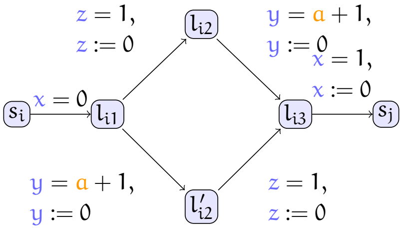
Encoding the counters C_1 and C_2

- Three clocks x, y and z and one parameter a
- Let c_1, c_2 be the values of C_1 and C_2
- In any location s_i, when $x = 0$ we have
  - $y = 1 - ac_1$
  - $z = 1 - ac_2$

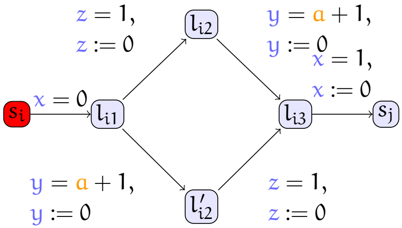
$$\longrightarrow l_0 \xrightarrow[]{\substack{x = 1 \\ x := 0}} s_0$$

Initializing the clocks

Encoding the instructions : Incrementing C_1

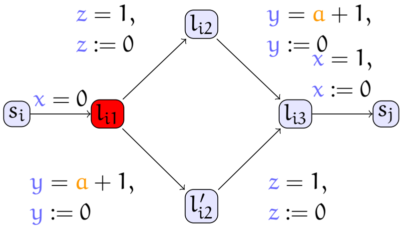


Encoding the instructions : Incrementing C_1



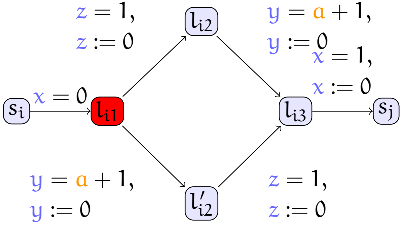
	s_i
x	0
y	$1 - ac_1$
z	$1 - ac_2$

Encoding the instructions : Incrementing C_1



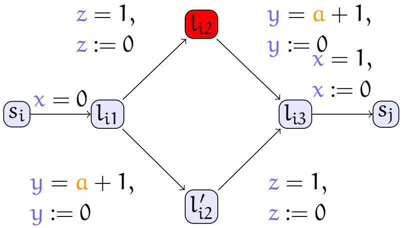
	l_{i1}
x	0
y	$1 - ac_1$
z	$1 - ac_2$

Encoding the instructions : Incrementing C_1



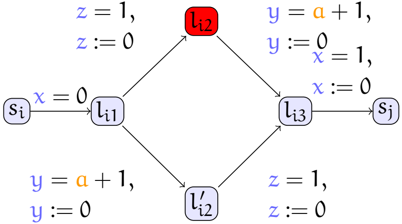
$$
\begin{array}{lll}
 & l_{i1} & & l_{i1} \\
x & 0 & \xrightarrow{ac_2} & ac_2 \\
y & 1 - ac_1 & & 1 - ac_1 + ac_2 \\
z & 1 - ac_2 & & 1
\end{array}
$$

Encoding the instructions : Incrementing C_1



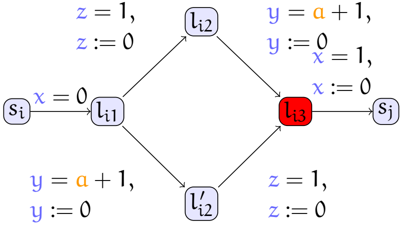
$$
\begin{array}{llcl}
 & l_{i1} & & l_{i2} \\
x & 0 & \xrightarrow{ac_2} & ac_2 \\
y & 1 - ac_1 & & 1 - ac_1 + ac_2 \\
z & 1 - ac_2 & & 0
\end{array}
$$

Encoding the instructions : Incrementing C_1



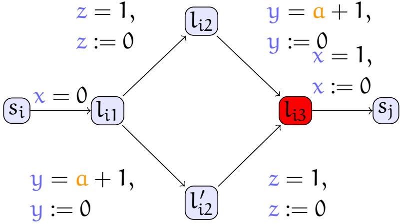
$$
\begin{array}{llclcl}
 & l_{i1} & & l_{i2} & & l_{i2} \\
x & 0 & \xrightarrow{ac_2} & ac_2 & \xrightarrow{a(c_1+1)-ac_2} & a(c_1+1) \\
y & 1 - ac_1 & & 1 - ac_1 + ac_2 & & a+1 \\
z & 1 - ac_2 & & 0 & & a(c_1+1) - ac_2
\end{array}
$$

Encoding the instructions : Incrementing C_1



	l_{i1}		l_{i2}		l_{i3}
x	0	$\xrightarrow{ac_2}$	ac_2	$\xrightarrow{a(c_1+1)-ac_2}$	$a(c_1+1)$
y	$1 - ac_1$		$1 - ac_1 + ac_2$		0
z	$1 - ac_2$		0		$a(c_1+1) - ac_2$

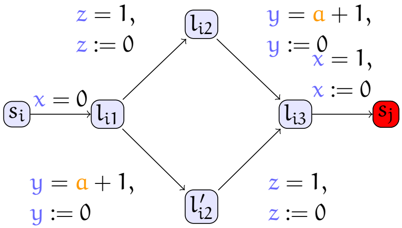
Encoding the instructions : Incrementing C_1



$$
\begin{array}{llclcl}
 & l_{i1} & & l_{i2} & & l_{i3} \\
x & 0 & \xrightarrow{ac_2} & ac_2 & \xrightarrow{a(c_1+1)-ac_2} & a(c_1+1) \\
y & 1 - ac_1 & & 1 - ac_1 + ac_2 & & 0 \\
z & 1 - ac_2 & & 0 & & a(c_1+1) - ac_2
\end{array}
$$

$$
\begin{array}{lcl}
 & & l_{i3} \\
x & \xrightarrow{1-a(c_1+1)} & 1 \\
y & & 1 - a(c_1+1) \\
z & & 1 - ac_2
\end{array}
$$

Encoding the instructions : Incrementing C_1

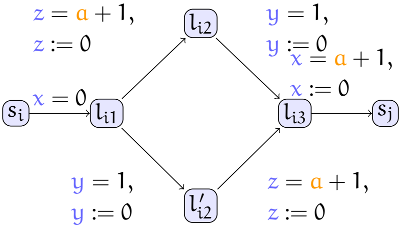


$$
\begin{array}{llcl cl}
 & l_{i1} & & l_{i2} & & l_{i3} \\
x & 0 & \xrightarrow{ac_2} & ac_2 & \xrightarrow{a(c_1+1)-ac_2} & a(c_1+1) \\
y & 1 - ac_1 & & 1 - ac_1 + ac_2 & & 0 \\
z & 1 - ac_2 & & 0 & & a(c_1+1) - ac_2
\end{array}
$$

$$
\begin{array}{lcl}
 & & s_j \\
x & \xrightarrow{1-a(c_1+1)} & 0 \\
y & & 1 - a(c_1+1) \\
z & & 1 - ac_2
\end{array}
$$

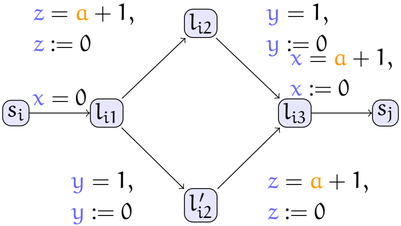
Encoding the instructions : Decrementing C_1

Replacing guards $y = a + 1$ with $y = 1$, and guards $x = 1$ and $z = 1$ with $x = a + 1$ and $z = a + 1$, respectively.



Encoding the instructions : Decrementing C_1

Replacing guards $y = a + 1$ with $y = 1$, and guards $x = 1$ and $z = 1$ with $x = a + 1$ and $z = a + 1$, respectively.



0-test: Straightforward (when $x = 0$, then y shall be 1)

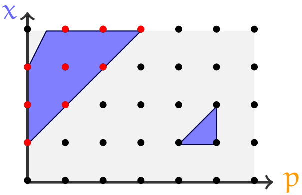
Outline

Definition of IP-PTAs

Definition (IP-PTA)

A PTA $\mathcal{A}$ is an integer points PTA (in short IP-PTA) if, in any reachable symbolic state (l, C) of $\mathcal{A}$, C contains at least one integer point.

C_1:
$p + 1 \leq x$
$\wedge x \leq 2p + 3$
contains integer points



C_2:
$x > 1$
$\wedge p < 5$
$\wedge x < p - 3$
contains no integer points

EF-emptiness for bounded IP-PTAs

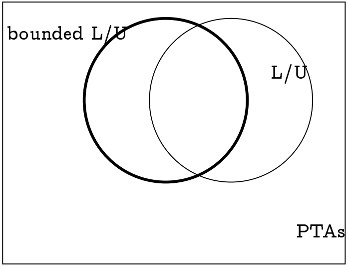
Theorem

EF-emptiness is decidable for bounded IP-PTAs.

Proof idea

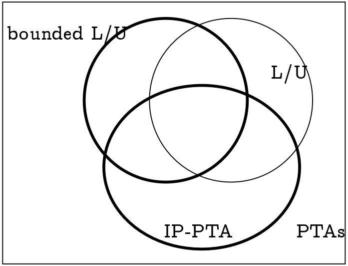
- There is a parameter valuation v and a path to l in $v(\mathcal{A})$ iff there is symbolic path π to some (l, C);
- There is a path π to (l, C) iff $C \neq \emptyset$;
- $C \neq \emptyset$ iff C contains an integer point (IP-PTA);
- C contains an integer point iff there is an integer parameter valuation v^* such that $v^*(\pi)$ is feasible in $v^*(\mathcal{A})$;
- There is a finite number of integer parameter valuations (bounded).

Expressiveness of IP-PTAs



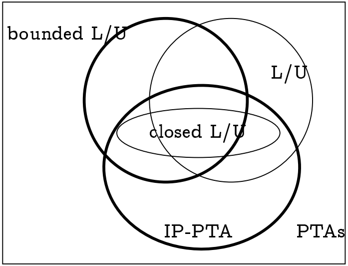
- L/U-PTAs incomparable with bounded L/U-PTA [André et al., 2016]

Expressiveness of IP-PTAs



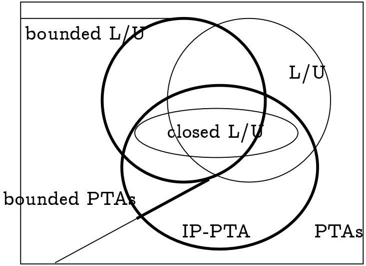
- L/U-PTAs incomparable with bounded L/U-PTA [André et al., 2016]
- IP-PTAs incomparable with L/U-PTAs

Expressiveness of IP-PTAs



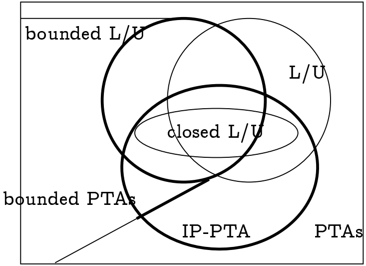
- L/U-PTAs incomparable with bounded L/U-PTA [André et al., 2016]
- IP-PTAs incomparable with L/U-PTAs
- IP-PTAs strictly larger than closed L/U-PTAs

Expressiveness of IP-PTAs



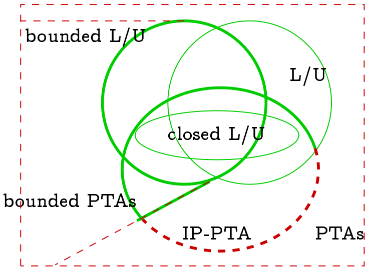
- L/U-PTAs incomparable with bounded L/U-PTA [André et al., 2016]
- IP-PTAs incomparable with L/U-PTAs
- IP-PTAs strictly larger than closed L/U-PTAs
- bounded IP-PTAs strictly larger than bounded closed L/U-PTAs

Expressiveness of IP-PTAs



- L/U-PTAs incomparable with bounded L/U-PTA [André et al., 2016]
- IP-PTAs incomparable with L/U-PTAs
- IP-PTAs strictly larger than closed L/U-PTAs
- bounded IP-PTAs strictly larger than bounded closed L/U-PTAs
- bounded IP-PTAs incomparable with bounded L/U-PTAs

Expressiveness of IP-PTAs



- L/U-PTAs incomparable with bounded L/U-PTA [André et al., 2016]
- IP-PTAs incomparable with L/U-PTAs
- IP-PTAs strictly larger than closed L/U-PTAs
- bounded IP-PTAs strictly larger than bounded closed L/U-PTAs
- bounded IP-PTAs incomparable with bounded L/U-PTAs

$\rightsquigarrow$ We strictly extend the class of PTAs for which the EF-emptiness problem is decidable

Intractability of exact synthesis

Proposition

The EF-synthesis for closed bounded L/U-PTAs (and therefore for IP-PTAs) is intractable in practice.

Proof idea

- Assume a closed bounded PTA $\mathcal{A}$.
- Split each parameter p_i into p_i^- and p_i^+.
- Assume complete synthesis is possible, giving constraint K.
- Then if one can test the emptiness of $K \wedge \left(\bigwedge_i p_i^- = p_i^+ \right)$ then this contradicts the undecidability of EF-emptiness for PTAs.

(following a reasoning from [Bozzelli and La Torre, 2009, Jovanović et al., 2015])

Undecidability of the membership

Theorem

It is undecidable whether a PTA is an IP-PTA.

Proof idea

From a reduction to the halting problem of a 2-counter machine: the model is IP iff the machine does not halt.

Undecidability of the membership

Theorem

It is undecidable whether a PTA is an IP-PTA.

Proof idea

From a reduction to the halting problem of a 2-counter machine: the model is IP iff the machine does not halt.

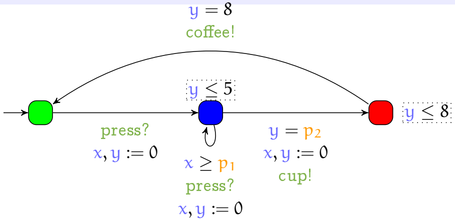
Consequence:

- Limited practical interest of IP-PTAs?
- ... unless we can exhibit sufficient syntactic conditions for membership

Reset-PTA

Definition (reset-PTA)

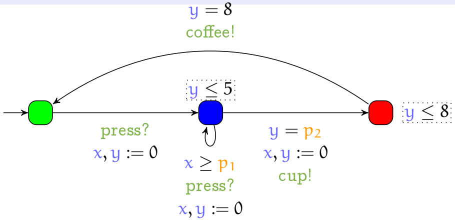
A PTA is a reset-PTA if, whenever a clock is compared to a parameter, all clocks are reset.



Reset-PTA

Definition (reset-PTA)

A PTA is a reset-PTA if, whenever a clock is compared to a parameter, all clocks are reset.



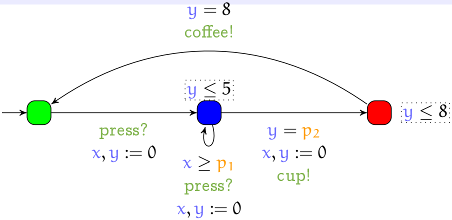
Theorem

A reset-PTA is an IP-PTA.

Reset-PTA

Definition (reset-PTA)

A PTA is a reset-PTA if, whenever a clock is compared to a parameter, all clocks are reset.



Theorem

A reset-PTA is an IP-PTA.

Corollary

The EF-emptiness problem is decidable for bounded reset-PTAs.

Outline

EF-universality and AF-emptiness

Theorem

EF-universality and AF-emptiness are undecidable for bounded IP-PTAs.

See paper for details

EF-universality and AF-emptiness

Theorem

EF-universality and AF-emptiness are undecidable for bounded IP-PTAs.

See paper for details

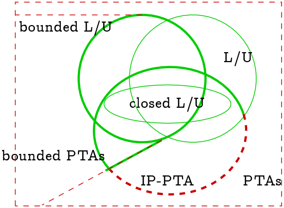
Corollary

EF-universality and AF-emptiness are undecidable for unbounded IP-PTAs, for bounded PTAs and for PTAs.

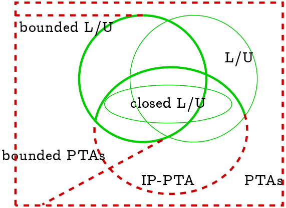
Outline

Summary

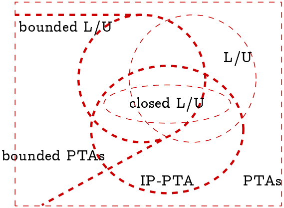
Class	bL/U-PTAs	bIP-PTAs	L/U-PTAs	IP-PTAs	bPTAs	PTAs
EF-empt.	√	√	**[HRSV02]**	×	[Miller00]	[AHV93]
EF-univ.	√	×	**[BlT09]**	×	×	×
AF-empt.	×	×	[JLR15]	×	×	[JLR15]



EF-emptiness



EF-universality



AF-emptiness

Conclusion

- PTAs extensively studied
- A new decidable subclass: bounded IP-PTAs... but of limited interest
  - ☹ Other problems than EF-emptiness are undecidable
  - ☹ Membership undecidable
  - ☹ Exact synthesis intractable
- A syntactic subclass of IP-PTAs: reset-PTAs
  - ☺ Promising decidability results

Perspectives

- Extend reset-PTAs
  - Using the same restrictions as in hybrid systems
    [Henzinger et al., 1998]
- AF-universality: studied in another paper with a subtle border between decidability and undecidability for L/U-PTAs
  [André and Lime, 2016]
- Study L-PTAs and U-PTAs
  - Entirely open classes

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. (2015).
What's decidable about parametric timed automata?
In *FTSCS*, volume 596 of *Communications in Computer and Information Science*, pages 1–17. Springer.

André, É., Chatain, T., Encrenaz, E., and Fribourg, L. (2009).
An inverse method for parametric timed automata.
International Journal of Foundations of Computer Science, 20(5):819–836.

André, É. and Lime, D. (2016).
Liveness in L/U-parametric timed automata.
Submitted.

References II

André, É., Lime, D., and Roux, O. H. (2016).
On the expressiveness of parametric timed automata.
In Fr‘ďnzle, M. and Markey, N., editors, *Proceedings of the 14th International Conference on Formal Modelling and Analysis of Timed Systems (FORMATS'16)*, volume 9984 of *Lecture Notes in Computer Science*, pages 19–34. 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.

Doyen, L. (2007).
Robust parametric reachability for timed automata.
Information Processing Letters, 102(5):208–213.

Henzinger, T., Kopke, P., Puri, A., and Varaiya, P. (1998).
What's decidable about hybrid automata?
Journal of Computer and System Sciences, 57:94–124.

References III

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.

Jovanović, A., Lime, D., and Roux, O. H. (2015).
Integer parameter synthesis for timed automata.
Transactions on Software Engineering, 41(5):445–461.

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.

Minsky, M. L. (1967).
Computation: finite and infinite machines.
Prentice-Hall, Inc., NJ, USA.

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

Author: Étienne André, Didier Lime, Olivier H. Roux