FORMATS 2015

September 2nd, 2015
Madrid, Spain

Language Preservation Problems in Parametric Timed Automata

Étienne André[1], Nicolas Marksey[2]

[1]LIPN, Université Paris 13, Sorbonne Paris Cité, CNRS, France
[2]LSV, ENS Cachan & CNRS, France

Beyond Model Checking: Parameter Synthesis

- Timed systems are characterized by a set of timing constants
  - "The packet transmission lasts for $50\,ms$"
  - "The sensor reads the value every $10\,s$"
- Verification for one set of constants does not usually guarantee the correctness for other values
- Challenges
  - Numerous verifications: is the system correct for any value within $[40; 60]$?
  - Optimization: until what value can we increase 10?
  - Robustness [Markey, 2011]: What happens if 50 is implemented with 49.99?

Beyond Model Checking: Parameter Synthesis

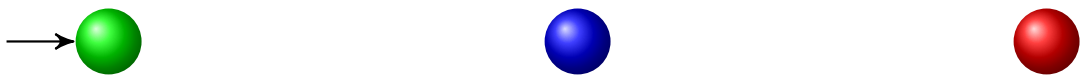
- Timed systems are characterized by a set of timing constants
  - "The packet transmission lasts for $50\,ms$"
  - "The sensor reads the value every $10\,s$"
- Verification for one set of constants does not usually guarantee the correctness for other values
- Challenges
  - Numerous verifications: is the system correct for any value within $[40; 60]$?
  - Optimization: until what value can we increase 10?
  - Robustness [Markey, 2011]: What happens if 50 is implemented with 49.99?
- Parameter synthesis
  - Consider that timing constants are unknown constants (parameters)
  - Find good values for the parameters

Outline

Outline

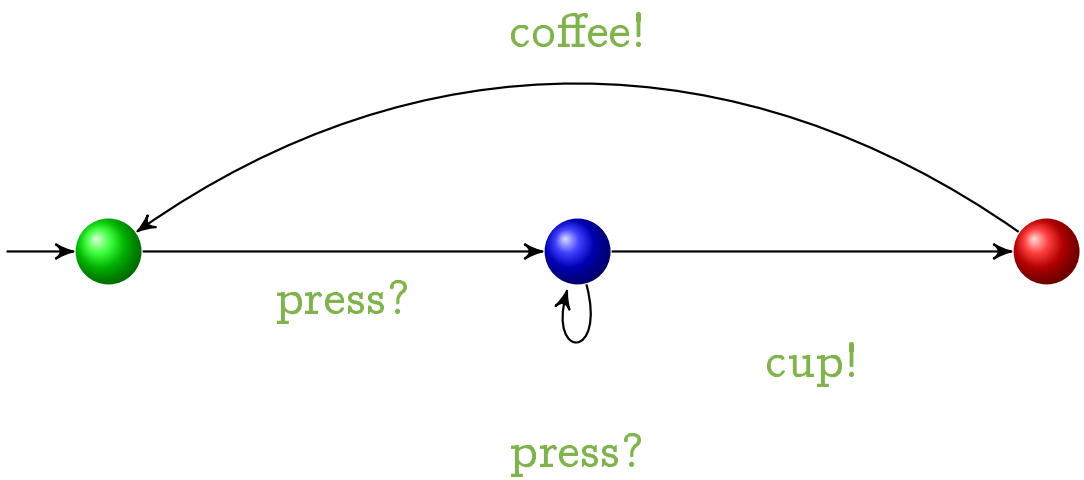
Timed automaton (TA)

- Finite state automaton (sets of locations)



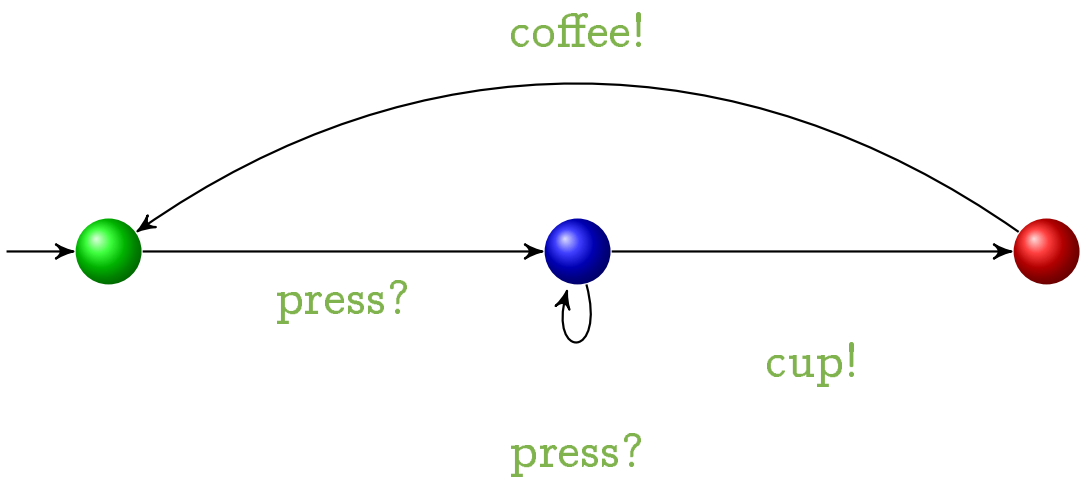
Timed automaton (TA)

- Finite state automaton (sets of locations and actions)



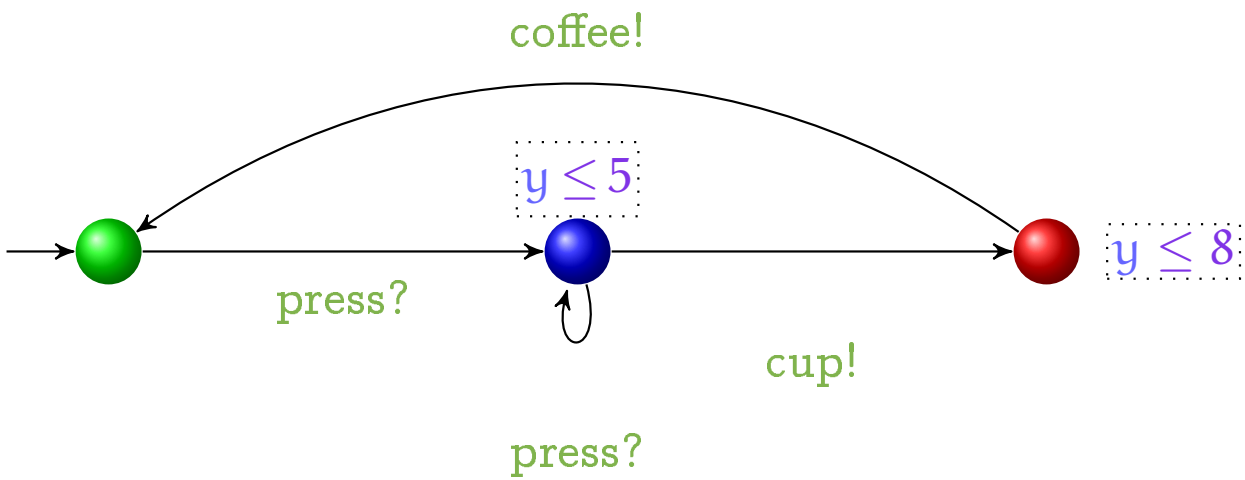
Timed automaton (TA)

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



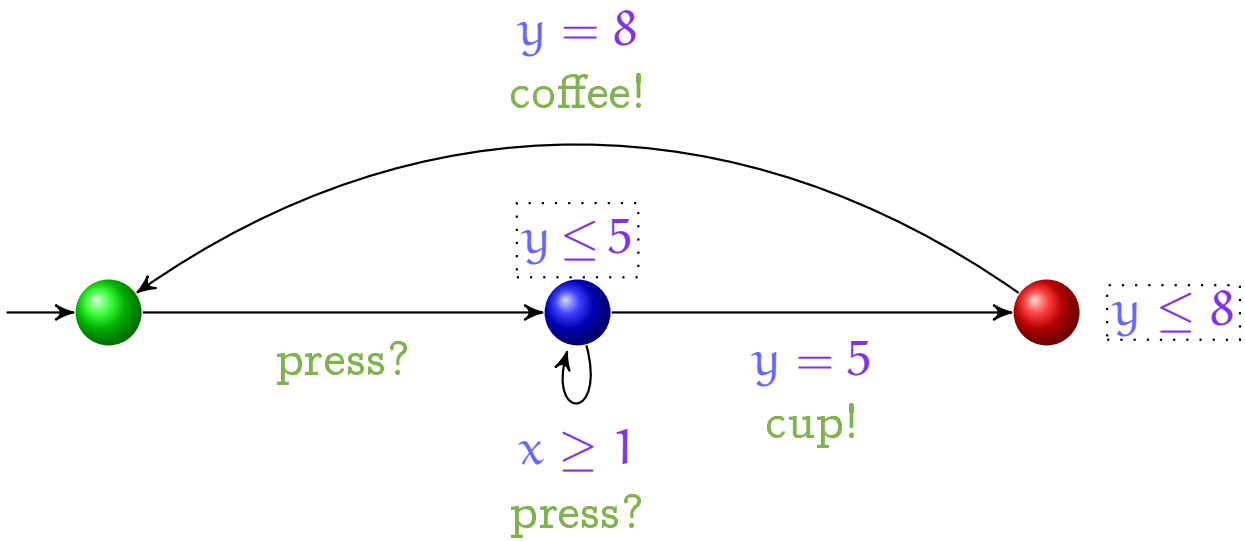
Timed automaton (TA)

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



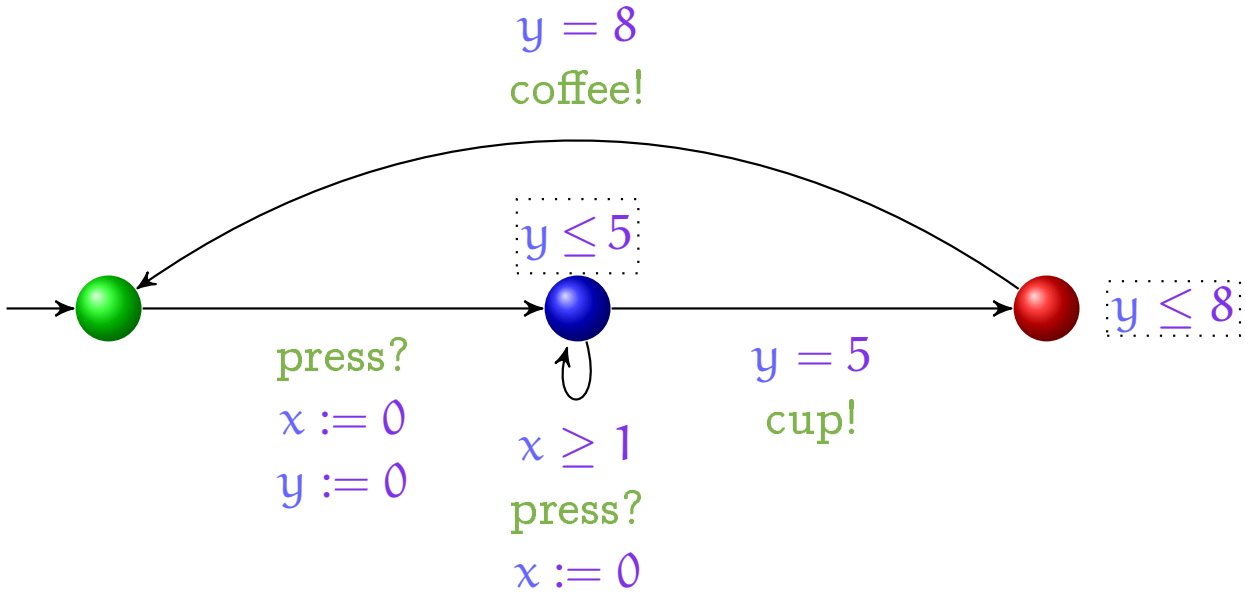
Timed automaton (TA)

- Finite state automaton (sets of locations and actions) augmented with a set X of clocks [Alur and Dill, 1994]
  - Real-valued variables evolving linearly at the same rate
- Features
  - Location invariant: property to be verified to stay at a location
  - Transition guard: property 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]
  - Real-valued variables evolving linearly at the same rate
- 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 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 elapsing of time
  - Possible concrete runs for the coffee machine

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

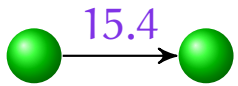
- Concrete run: alternating sequence of concrete states and actions or elapsing of time
  - Possible concrete runs for the coffee machine
    - Coffee with no sugar

x 0
y 0

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

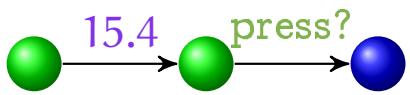


x	0	15.4
y	0	15.4

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

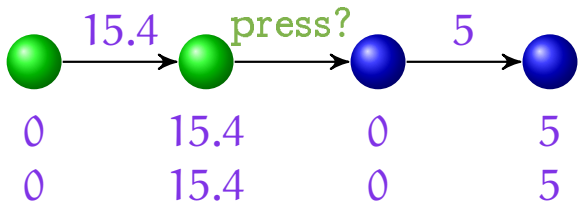


x	0	15.4	0
y	0	15.4	0

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

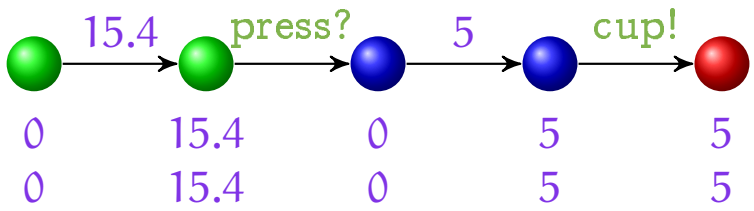
- Concrete run: alternating sequence of concrete states and actions or elapsing of time
  - Possible concrete runs for the coffee machine
    - Coffee with no sugar



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

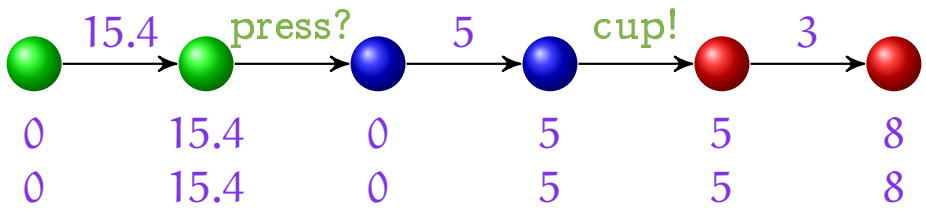
- Concrete run: alternating sequence of concrete states and actions or elapsing of time
  - Possible concrete runs for the coffee machine
    - Coffee with no sugar



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

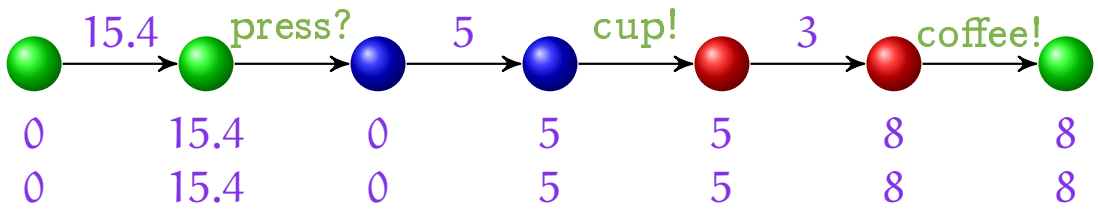
- Concrete run: alternating sequence of concrete states and actions or elapsing of time
  - Possible concrete runs for the coffee machine
    - Coffee with no sugar



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

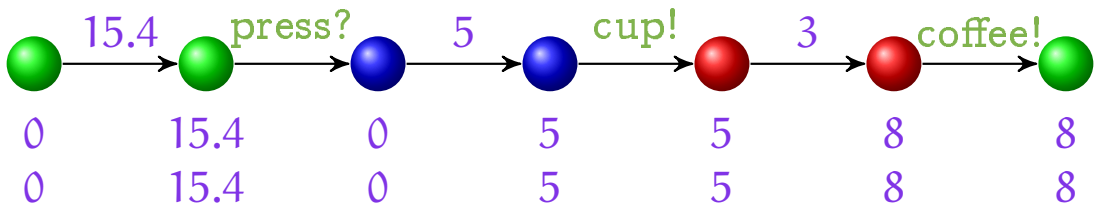
- Concrete run: alternating sequence of concrete states and actions or elapsing of time
  - Possible concrete runs for the coffee machine
    - Coffee with no sugar



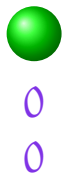
Concrete semantics of timed automata

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



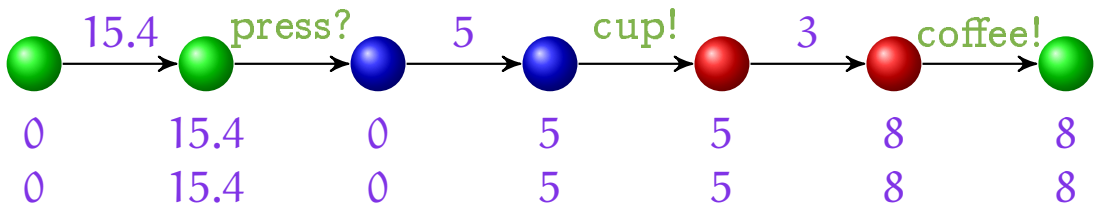
    - Coffee with 2 doses of sugar



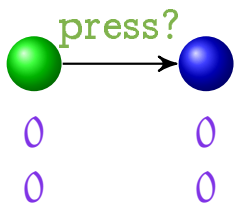
Concrete semantics of timed automata

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



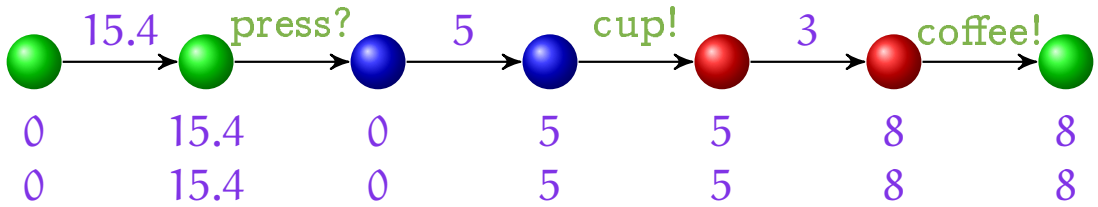
    - Coffee with 2 doses of sugar



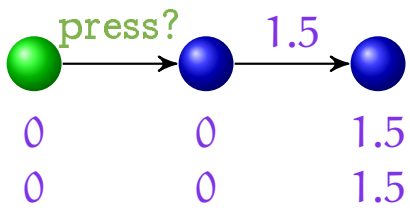
Concrete semantics of timed automata

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



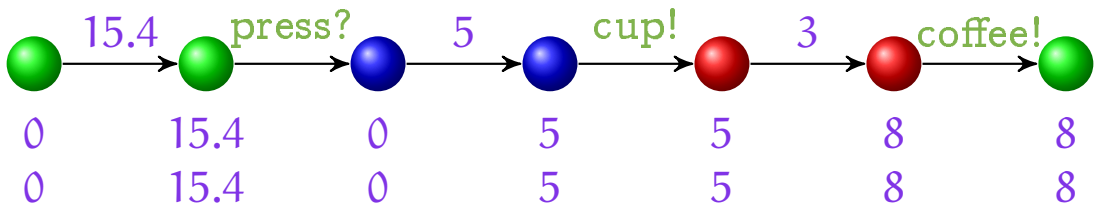
    - Coffee with 2 doses of sugar



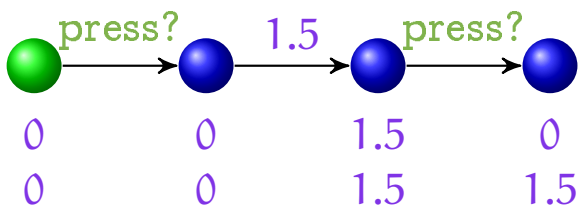
Concrete semantics of timed automata

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



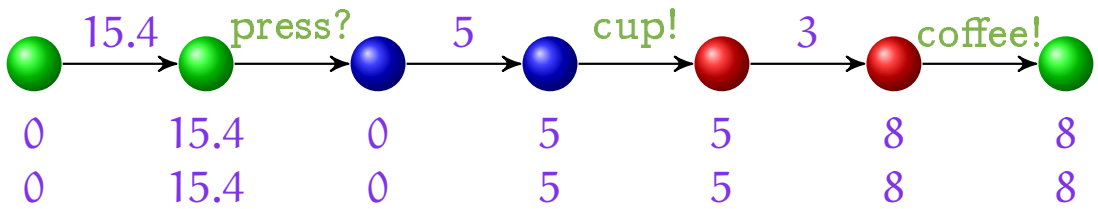
    - Coffee with 2 doses of sugar



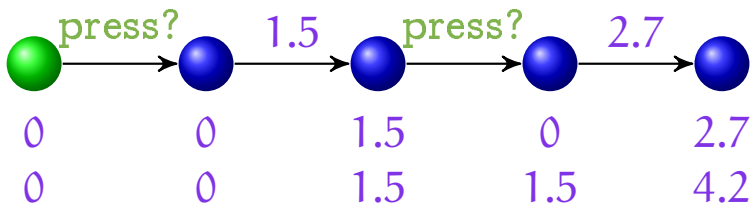
Concrete semantics of timed automata

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



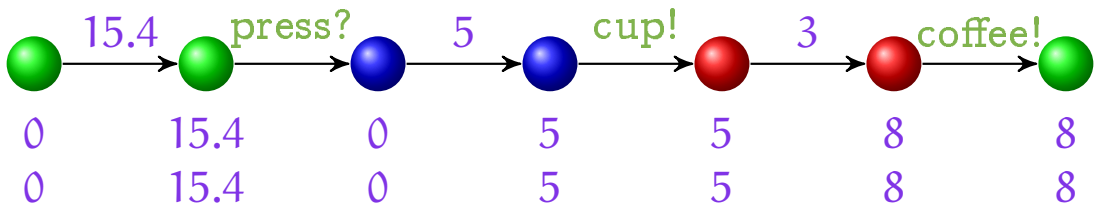
    - Coffee with 2 doses of sugar



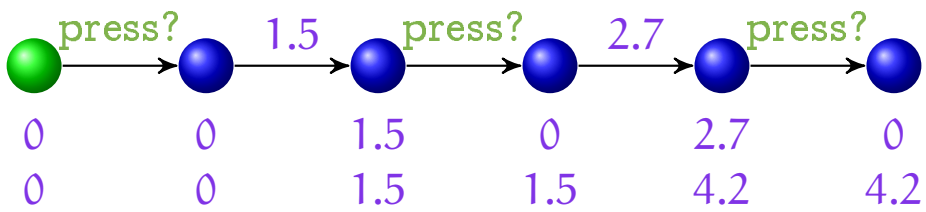
Concrete semantics of timed automata

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



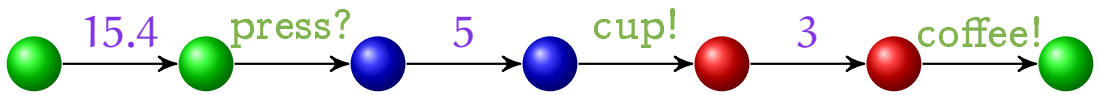
    - Coffee with 2 doses of sugar



Concrete semantics of timed automata

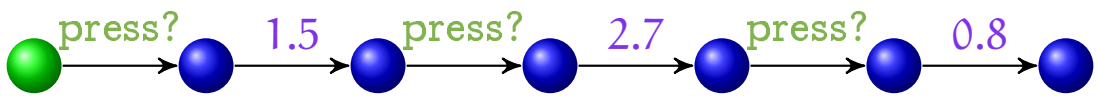
- Concrete state of a TA: pair (l, w), where
  - l is a location,
  - w is a valuation of each clock

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

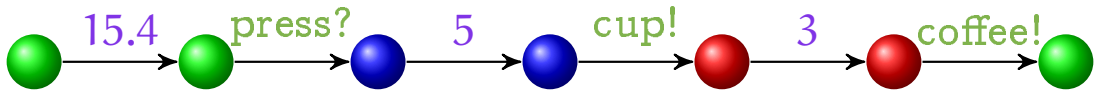


x	0	0	1.5	0	2.7	0	0.8
y	0	0	1.5	1.5	4.2	4.2	5

Concrete semantics of timed automata

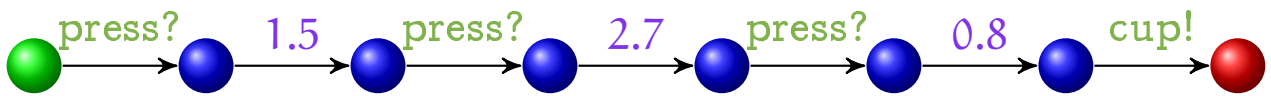
- Concrete state of a TA: pair (l, w), where
  - l is a location,
  - w is a valuation of each clock

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

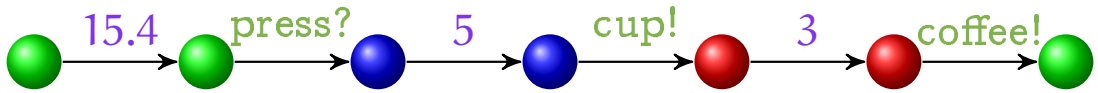


x	0	0	1.5	0	2.7	0	0.8	0.8
y	0	0	1.5	1.5	4.2	4.2	5	5

Concrete semantics of timed automata

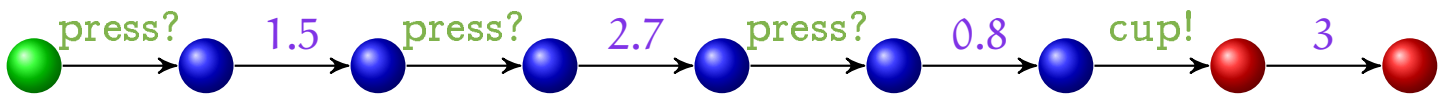
- Concrete state of a TA: pair (l, w), where
  - l is a location,
  - w is a valuation of each clock

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

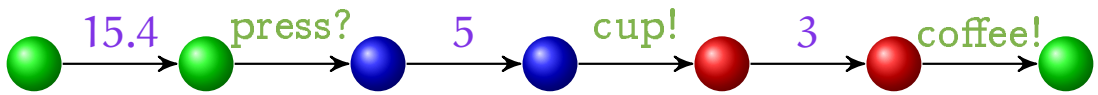


x	0	0	1.5	0	2.7	0	0.8	0.8	3.8
y	0	0	1.5	1.5	4.2	4.2	5	5	8

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



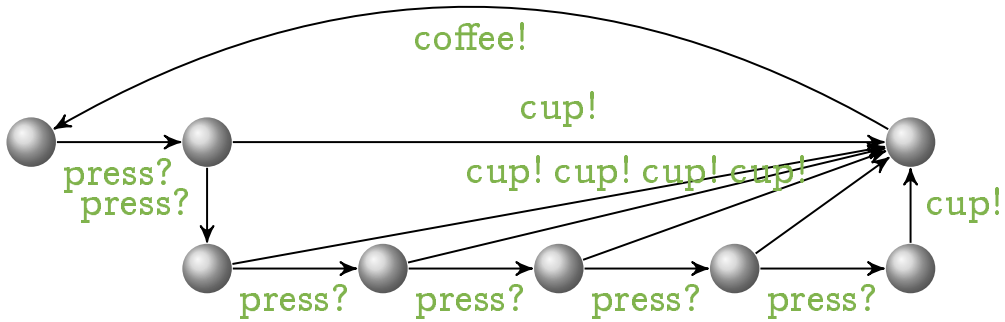
x	0	15.4	0	5	5	8	8
y	0	15.4	0	5	5	8	8

    - 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

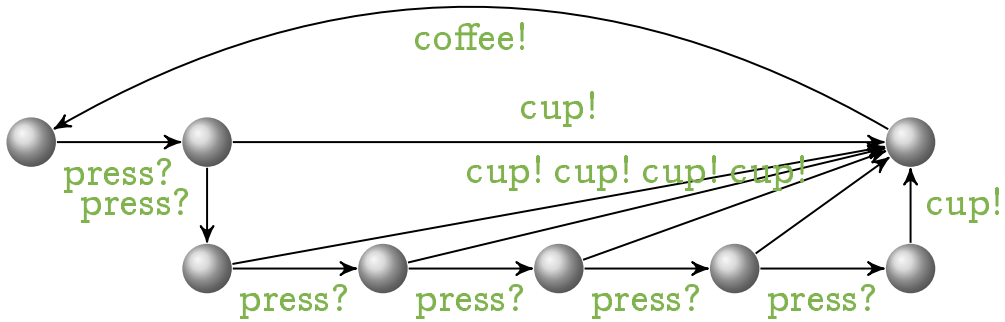
Untimed language and trace set

- Untimed language of a TA
  - Set over all maximal concrete runs of the TA of the corresponding sequences of actions

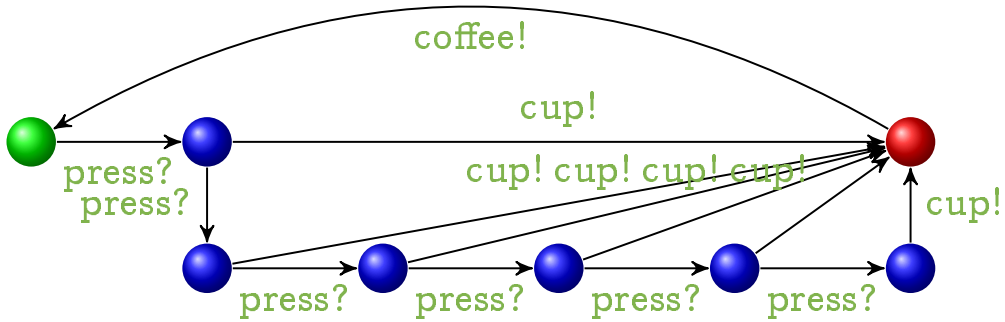


Untimed language and trace set

- Untimed language of a TA
  - Set over all maximal concrete runs of the TA of the corresponding sequences of actions

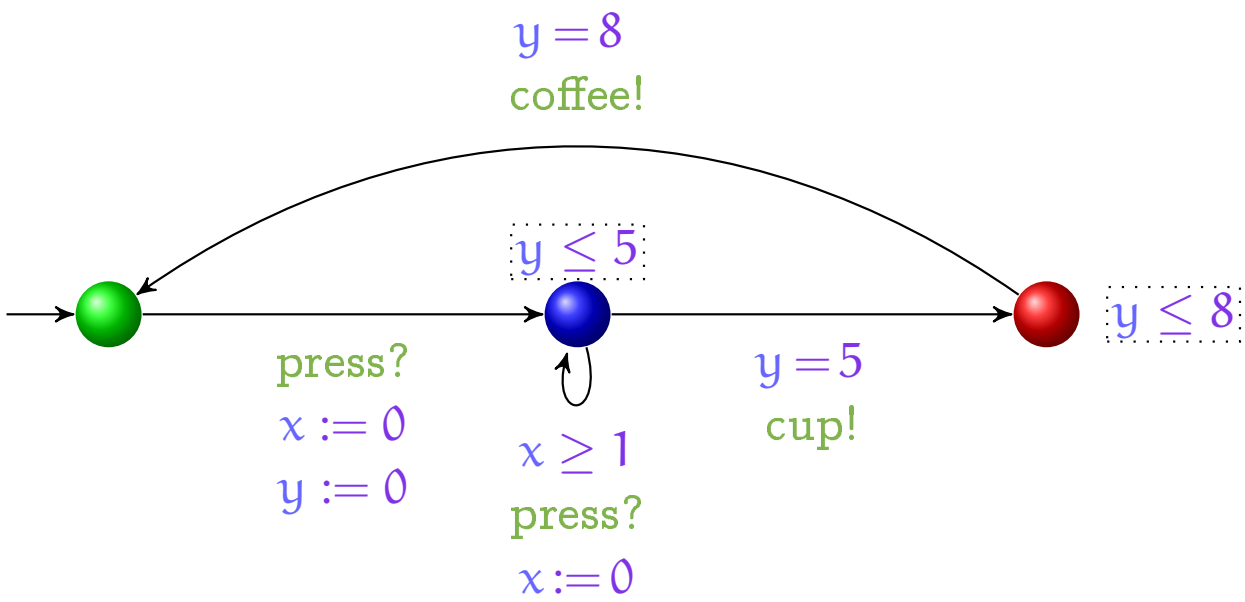


- Trace set of a TA
  - Set over all maximal concrete runs of the TA of the corresponding alternating sequences of locations and actions



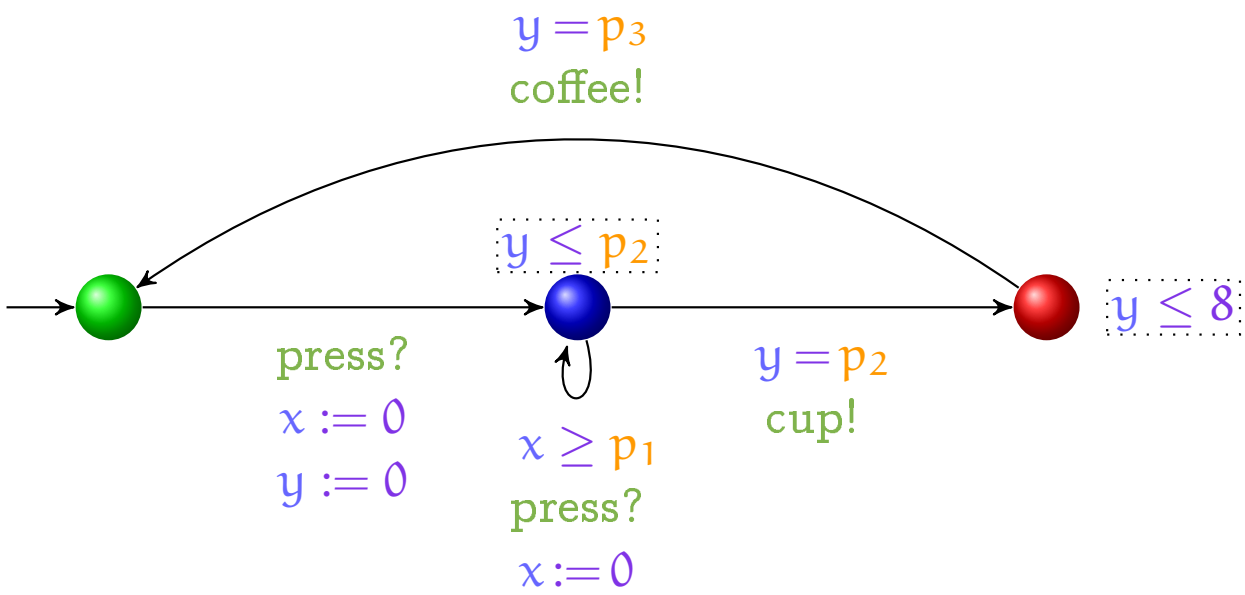
Parametric Timed Automaton (PTA)

- 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 ν, we denote by $\nu(\mathcal{A})$ the (non-parametric) timed automaton where all parameters are valuated by ν

Problems

Definition (language preservation)

Input: a PTA $\mathcal{A}$ and a parameter valuation ν

Question: does there exist another parameter valuation ν' giving rise to the same untimed language, i.e., $Lang(\nu(\mathcal{A})) = Lang(\nu'(\mathcal{A}))$?

Problems

Definition (language preservation)

Input: a PTA $\mathcal{A}$ and a parameter valuation ν

Question: does there exist another parameter valuation ν' giving rise to the same untimed language, i.e., $Lang(\nu(\mathcal{A})) = Lang(\nu'(\mathcal{A}))$?

Definition (trace preservation)

Input: a PTA $\mathcal{A}$ and a parameter valuation ν

Question: does there exist another parameter valuation ν' giving rise to the same trace set, i.e., $Traces(\nu(\mathcal{A})) = Traces(\nu'(\mathcal{A}))$?

Continuous version

Continuous version of those problems:

- additionally require that the language (resp. set of traces) is preserved under any intermediary valuation of the form $\lambda \cdot \nu + (1 - \lambda) \cdot \nu'$, for $\lambda \in [0, 1]$

Outline

Undecidability of language preservation

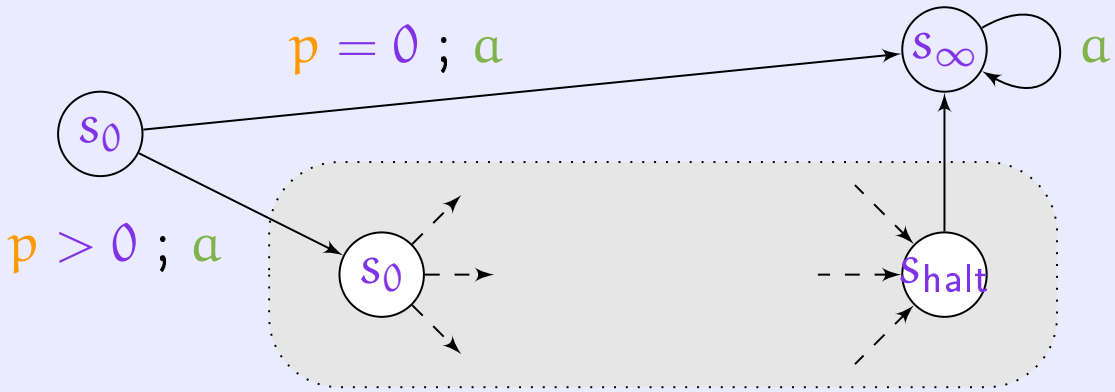
Result

The language preservation problem for PTA with one parameter is undecidable (both over discrete and continuous time, and for integer and rational parameter valuations).

Undecidability of language preservation: proof

Proof.

Reduction from a 2-counter machine, using an ad-hoc encoding requiring 4 clocks (the 4th clock is used to count the number of transitions taken) and 1 parameter



- For $p = 0$, the language is a^ω
- For $p \neq 0$, the PTA needs to mimic the 2CM: if the 2CM halts, the language is a^ω; otherwise, for any p, the PTA blocks at some point, and the language is a^n for some n

Undecidability of trace preservation

Two different proofs.

Result

The trace preservation problem for PTA with one parameter is undecidable (both over discrete and continuous time, and for integer and rational parameter valuations)

1. *for PTA with at least one parameter, 4 parametric clocks, unboundedly many non-parametric clocks, and diagonal constraints (of the form $x_1 - x_2 \prec c$).*
2. *for PTA with at least one parameter and unboundedly many parametric clocks.*

Unboundedly many clocks: because one clock per state of the 2-counter machine.

Undecidability of the continuous versions

Result

The language and trace preservation problems for PTA with one (possibly bounded) parameter are undecidable.

Undecidability of the continuous versions

Result

The language and trace preservation problems for PTA with one (possibly bounded) parameter are undecidable.

Proof.

Reduction from a 2-counter machine (using an encoding proposed by Didier Lime) □

Outline

Finitenes of the zone graph

1-clock PTA: PTA with a single clock, and possibly more complex parameter constraints (of the form $\sum_i \alpha_i p_i \sim 0$)

Finitenes of the zone graph

1-clock PTA: PTA with a single clock, and possibly more complex parameter constraints (of the form $\sum_i \alpha_i p_i \sim 0$)

Result

The parametric zone graph of a 1-clock PTA is finite.

Computability for 1-clock PTA

Result

It is possible to compute all parameter valuations preserving the trace set of a deterministic 1-clock PTA.

Computability for 1-clock PTA

Result

It is possible to compute all parameter valuations preserving the trace set of a deterministic 1-clock PTA.

Proof.

1. Semi-algorithm $\mathsf{TPS}(\mathcal{A}, \nu)$ synthesizing all valuations yielding the same trace set as $\nu(\mathcal{A})$

Computability for 1-clock PTA

Result

It is possible to compute all parameter valuations preserving the trace set of a deterministic 1-clock PTA.

Proof.

1. Semi-algorithm $\mathsf{TPS}(\mathcal{A}, \nu)$ synthesizing all valuations yielding the same trace set as $\nu(\mathcal{A})$
2. $\mathsf{TPS}(\mathcal{A}, \nu)$ is complete for deterministic PTA (whenever it terminates)

□

Computability for 1-clock PTA

Result

It is possible to compute all parameter valuations preserving the trace set of a deterministic 1-clock PTA.

Proof.

1. Semi-algorithm $\mathsf{TPS}(\mathcal{A}, \nu)$ synthesizing all valuations yielding the same trace set as $\nu(\mathcal{A})$
2. $\mathsf{TPS}(\mathcal{A}, \nu)$ is complete for deterministic PTA (whenever it terminates)
3. $\mathsf{TPS}(\mathcal{A}, \nu)$ terminates for PTA for which the parametric zone graph is finite

□

Computability for 1-clock PTA

Result

It is possible to compute all parameter valuations preserving the trace set of a deterministic 1-clock PTA.

Proof.

1. Semi-algorithm $\mathsf{TPS}(\mathcal{A}, \nu)$ synthesizing all valuations yielding the same trace set as $\nu(\mathcal{A})$
2. $\mathsf{TPS}(\mathcal{A}, \nu)$ is complete for deterministic PTA (whenever it terminates)
3. $\mathsf{TPS}(\mathcal{A}, \nu)$ terminates for PTA for which the parametric zone graph is finite
4. The parametric zone graph of 1-clock PTA is finite

□

Decidability for 1-clock PTA

Corollary

Trace preservation (emptiness) is decidable for deterministic 1-clock PTA.

Decidability for 1-clock PTA

Corollary

Trace preservation (emptiness) is decidable for deterministic 1-clock PTA.

Corollary

Language preservation (emptiness) is decidable for deterministic 1-clock PTA.

Proof.

Equivalence between trace and language for deterministic PTA. □

Lower-bound/upper-bound PTA

L/U-PTA restrict the use of parameters: partition of P into

- Lower-bound parameters: always compared with clocks as lower-bound (e.g., $p_1 \leq x$), and
- Upper-bound parameters: always compared with clocks as upper-bound (e.g., $x \leq p_2$)

Emptiness of the reachability problem is decidable for L/U-PTA

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

Undecidability for L/U-PTA

Result

All four preservation problems are undecidable for L/U-PTA.

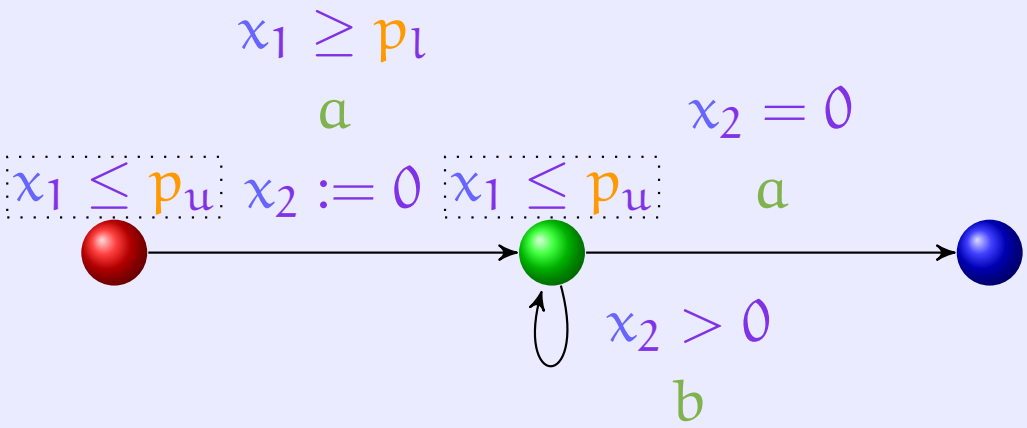
Undecidability for L/U-PTA

Result

All four preservation problems are undecidable for L/U-PTA.

Proof.

- Constraining parameter equality in L/U-PTA
  - The language of the following gadget is aa iff $p_l = p_u$



- Then, using $p_l = p_u$, we can encode a 2-counter machine

□

Decidability for 1-parameter U- and L-PTA

Result

The trace-preservation problem is decidable for deterministic U-PTA and deterministic L-PTA with a single integer-valued parameter.

- Non-deterministic: open
- More than one parameter: open
- One rational-valued parameter: open

Outline

Summary

Preservation	1ip-dL&U-PTA	L&U-PTA	bL/U-PTA	L/U-PTA	bPTA	PTA
Language	decidable	open	*undec.*	*undec.*	*undec.*	*undec.*
Trace	decidable	open	*undec.*	*undec.*	*undec.*	*undec.*
Robust language	(N/A)	open	*undec.*	*undec.*	*undec.*	*undec.*
Robust trace	(N/A)	open	*undec.*	*undec.*	*undec.*	*undec.*

b: bounded parameter domain

Perspectives

- Open problems
  - Trace preservation (general case): can we obtain undecidability with a bounded number of clocks?
  - Open classes: U-PTA and L-PTA
- Language preservation is close to a form of robustness
  - Decidable subclasses of the robust problems would improve the boundary between decidability and undecidability

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 Kosaraju, S. R., Johnson, D. S., and Aggarwal, A., editors, *STOC*, pages 592–601. ACM.

André, É. and Markey, N. (2015).
Language preservation problems in parametric timed automata.
In Sankaranarayanan, S. and Vicario, E., editors, *Proceedings of the 13th International Conference on Formal Modeling and Analysis of Timed Systems (FORMATS'15)*, volume 9268 of *Lecture Notes in Computer Science*, pages 27–43. 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.

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.

References II

Markey, N. (2011).
Robustness in real-time systems.
In *Proceedings of the 6th IEEE International Symposium on Industrial Embedded Systems (SIES'11)*, pages 28–34, Västerås, Sweden. IEEE Computer Society Press.

Licensing

License of this document

Authors: Étienne André and Nicolas Markey