Timed automata with parametric updates

Mefosyloma 2018[1]

Étienne André *, Didier Lime **& **Mathias Ramparison***

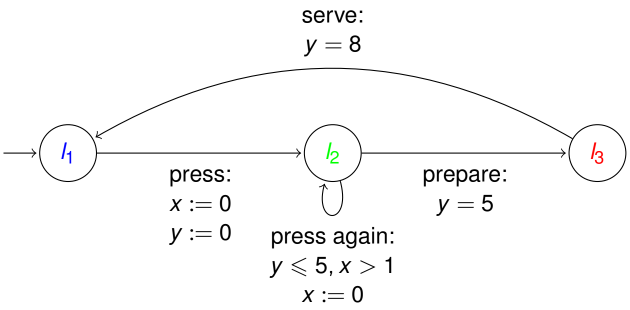
*LIPN, Université Paris 13
**LS2N, École Centrale de Nantes

May 18, 2018

[1] To be presented at ACSD 2018

Example of timed automaton

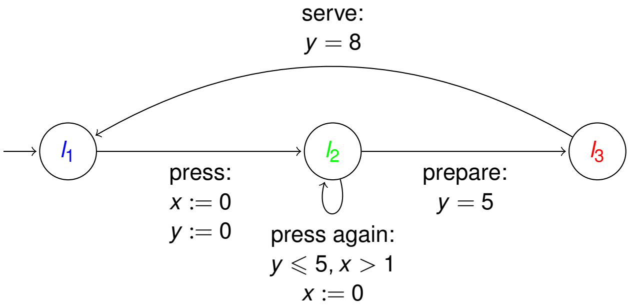
A **timed automaton** [AD94] which models a coffee machine



- Locations : $\{l_1, l_2, l_3\}$, clocks : $\{x, y\}$, action : {press, press again, prepare, serve}
- $Guard(\text{press again}) = \{y \leqslant 5 \wedge x \geqslant 0\}$, $Guard(\text{prepare}) = \{y = 5\}$, $Guard(\text{serve}) = \{y = 8\}$
- $Reset(\text{press}) = \{x, y := 0\}$, $Reset(\text{press again}) = \{x := 0\}$

Example of timed automaton

A **timed automaton** which models a coffee machine [Alur and Dill, 1994]

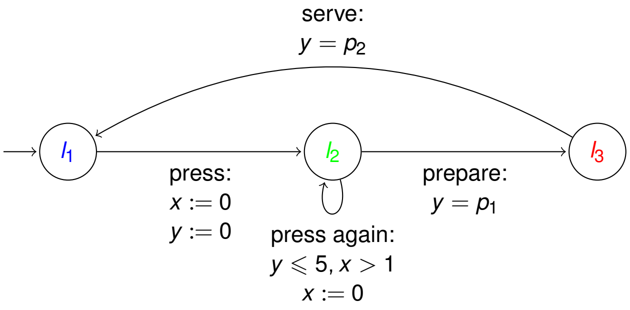


- A run : $(l_1, (0, 0)) \xrightarrow[2.1]{\text{press}} (l_2, (0, 0)) \xrightarrow[1.2]{\text{press again}} (l_2, (0, 1.2)) \xrightarrow[3.8]{\text{prepare}} (l_3, (3.8, 5)) \xrightarrow[3]{\text{serve}} (l_1, (6.8, 8))$
- triple (location, (value of x, value of y)) and $\xrightarrow[\delta]{\text{name}}$ discrete transition "name" after a delay δ.

Example of parametric timed automaton

What if all constants are not specified ahead?

A **parametric timed automaton** [AHV93] which models a parametric coffee machine



- A possible run if $p_1 = 2, p_2 = 3$: $\big(l_1, (0,0)\big) \xrightarrow[2]{\text{press}} \big(l_2, (0,0)\big) \xrightarrow[1]{\text{press again}} \big(l_2, (0,1)\big) \xrightarrow[1]{\text{prepare}} \big(l_3, (1,2)\big) \xrightarrow[1]{\text{serve}} \big(l_1, (2,3)\big)$
- The same run is impossible if $p_1 = 5, p_2 = 2$, or $p_1 < 1$.

Common decision problems for timed automata

- *Reachability*: Is there a run such that the location l is reachable?
  Universality: For all runs, is the location l reachable?

Common decision problems for timed automata

- *Reachability*: Is there a run such that the location l is reachable?
  Universality: For all runs, is the location l reachable?
- Proved decidable [AD94]. Strategy: construct a finite automaton using an abstraction of clock valuations (clock regions)

Common decision problems for parametric timed automata

- *EF-emptiness*: Is there a parameter valuation s.t. there exists a run reaching l in the instantiated TA?
  EF-synthesis: Compute all parameter valuations s.t. there exists a run reaching l in the instantiated TA?

Common decision problems for parametric timed automata

- *EF-emptiness*: Is there a parameter valuation s.t. there exists a run reaching *l* in the instantiated TA?
  EF-synthesis: Compute all parameter valuations s.t. there exists a run reaching *l* in the instantiated TA?
- *EF*-emptiness problem: proved undecidable in general case [AHV93], unbounded integer-valued parameters, (un)bounded rational valued parameters and even with only one bounded parameter [Mil00]

Common decision problems for parametric timed automata

- *EF-emptiness*: Is there a parameter valuation s.t. there exists a run reaching l in the instantiated TA?
  EF-synthesis: Compute all parameter valuations s.t. there exists a run reaching l in the instantiated TA?
- *EF*-emptiness problem: proved undecidable in general case [AHV93], unbounded integer-valued parameters, (un)bounded rational valued parameters and even with only one bounded parameter [Mil00]
- Need to add restrictions on parameters, or restrain the PTA syntax

Contributions

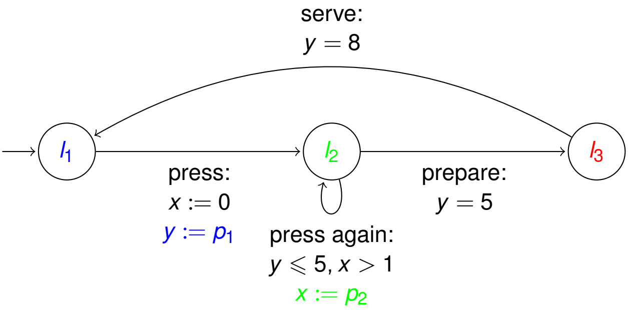
- As almost everything is undecidable, we try to remove parameters from guards and invariants.
- The reachability problem is decidable for timed automata with updates to integer constants [BDFP04].

Contributions

- As almost everything is undecidable, we try to remove parameters from guards and invariants.
- The reachability problem is decidable for timed automata with updates to integer constants [BDFP04].
- New formalism with parametric updates of clocks: update-to-parameter TA (U2P-TA)
- One undecidability result and one decidability result

U2P-TA

Update-to-parameter TA (U2P-TA) with two rational-valued parameters p_1, p_2:



Parametric clock updates: $y := p_1, x := p_2$.

U2P-TA

Theorem

The EF-emptiness problem is undecidable for bounded rational-valued U2P-TAs

Proof sketch: we prove that a PTA can be transformed into an U2P-TA, using K_{MAX} the maximum value between constants and parameters appearing in guards.

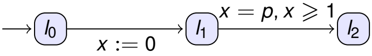


Figure: A PTA A

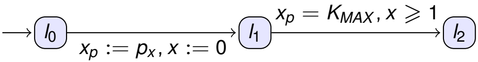


Figure: A U2P-TA obtained from A

U2P-TA

As we can transform any unbounded rational-valued U2P-TA into a bounded rational-valued U2P-TA:

Theorem

The EF-emptiness problem is undecidable for unbounded rational-valued U2P-TAs

Integer-valued U2P-TA

U2P-TAs with integer-valued parameters.

Integer-valued U2P-TA

U2P-TAs with integer-valued parameters.

Theorem

EF-synthesis is computable for unbounded integer-valued U2P-TAs.

Integer-valued U2P-TA

U2P-TAs with integer-valued parameters.

Theorem

EF-synthesis is computable for unbounded integer-valued U2P-TAs.

... and the *EF*-emptiness problem is decidable, *unlike integer-valued PTAs* [AHV93,BBLS15].

Proof sketch: using equivalence between parameter valuations if $> K_{MAX}$, we enumerate parameter valuations $\leqslant K_{MAX}$.

Conclusion

- Two new subclasses of PTAs: rational-valued U2P-TAs for which the *EF*-emptiness problem is undecidable, and integer-valued U2P-TAs for which it is decidable.

Conclusion

- Two new subclasses of PTAs: rational-valued U2P-TAs for which the *EF*-emptiness problem is undecidable, and integer-valued U2P-TAs for which it is decidable.

Future work:

- Find syntactic restrictions in order to find a decidability result for rational parameter valuations
- Adapt our formalism to hybrid systems, in which clocks can evolve at different rates

References

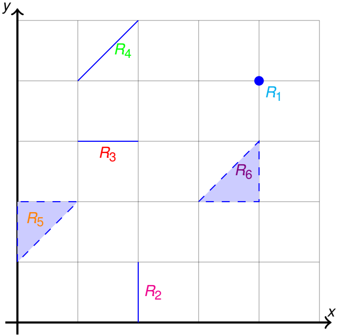
Rajeev Alur and David L. Dill.
A theory of timed automata.
Theoretical Computer Science, 126(2):183–235, April 1994.

Rajeev Alur, Thomas A. Henzinger, and Moshe Y. Vardi.
Parametric real-time reasoning.
In S. Rao Kosaraju, David S. Johnson, and Alok Aggarwal, editors, *STOC*, pages 592–601, New York, NY, USA, 1993. ACM.

Patricia Bouyer, Catherine Dufourd, Emmanuel Fleury, and Antoine Petit.
Updatable timed automata.
Theoretical Computer Science, 321(2-3):291–345, August 2004.

Joseph S. Miller.
Decidability and complexity results for timed automata and semi-linear hybrid automata.
In Nancy A. Lynch and Bruce H. Krogh, editors, *HSCC*, volume 1790 of *Lecture Notes in Computer Science*, pages 296–309. Springer, 2000.

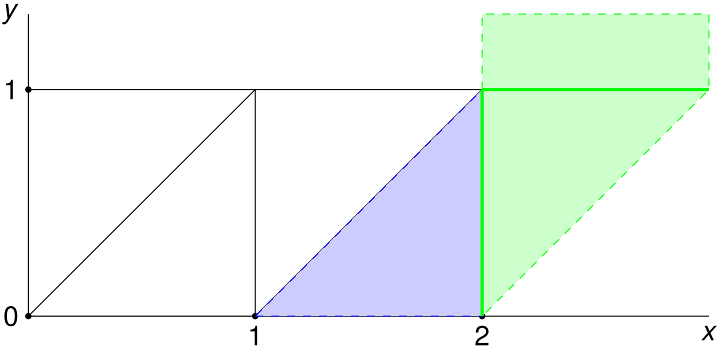
Clock regions



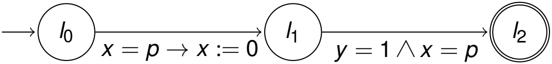
- The corner point: $R_1 = \{(4, 4)\}$
- The vertical line: $R_2 = \{(x, y) \mid x = 2, 0 < y < 1\}$
- The horizontal line: $R_3 = \{(x, y) \mid y = 3, 1 < x < 2\}$
- The diagonal: $R_4 = \{(x, y) \mid x = y - 3, 4 < y < 5\}$
- The upward triangle: $R_5 = \{(x, y) \mid 0 < x < y - 1, 1 < y < 2\}$
- The downward triangle: $R_6 = \{(x, y) \mid y + 1 < x < 4, 2 < y < 3\}$

Clock regions

Two clocks x, y, max constants $c_x = 2, c_y = 1$.
Time successors of the blue region $\{0 < y < 1, 0 < y < x - 1\}$ different of itself: four regions in green: $\{0 < y < 1, x = 2\}, \{0 < y < 1, x > 2\}, \{y = 1, x > 2\}$ and $\{y > 1, x > 2\}$

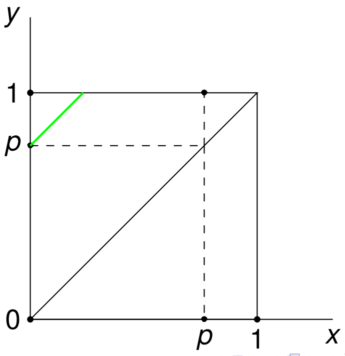


Using regions for parametric timed automata ?

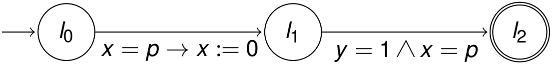


In l_1: $(x, y) = (0, p)$
But after letting some time elapse, depending on the value of $0 < p < 1$ we reach different regions:

- region $y = 1, 0 < x < p$ if $1 > p > \frac{1}{2}$



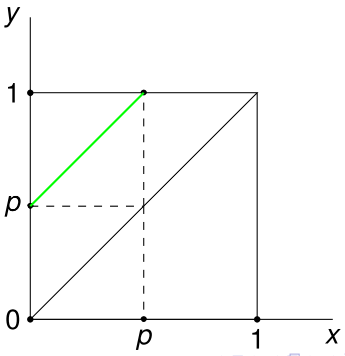
Using regions for parametric timed automata ?



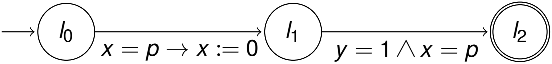
In l_1: $(x, y) = (0, p)$

But after letting some time elapse, depending on the value of $0 < p < 1$ we access different regions:

- region $y = 1, x = p$ if $p = \frac{1}{2}$

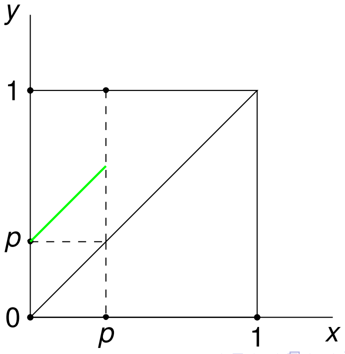


Using regions for parametric timed automata ?

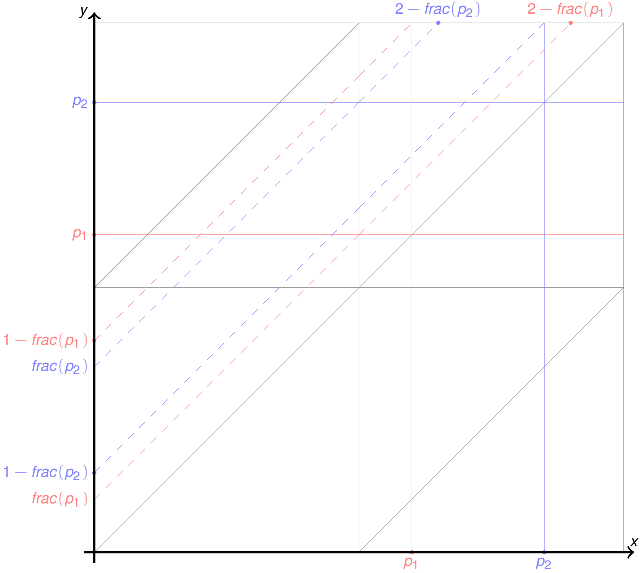


In l_1: $(x, y) = (0, p)$
But after letting some time elapse, depending on the value of $0 < p < 1$ we access different regions:

- region $p < y < 1, x = p$ if $p < \frac{1}{2}$

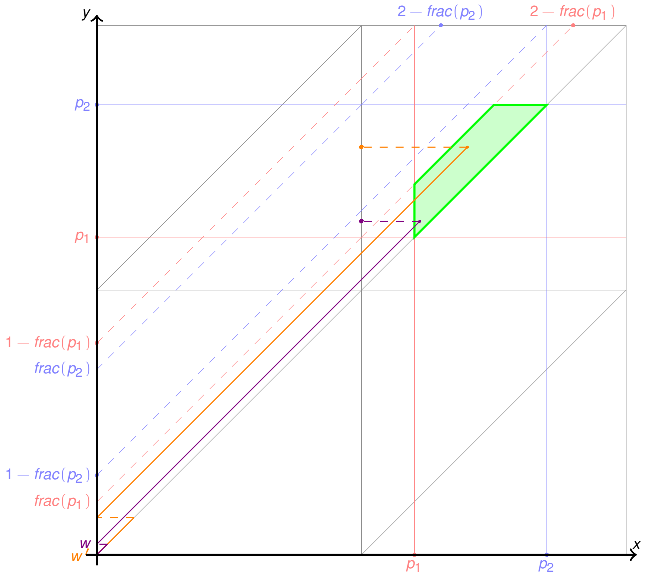


U2P-TA



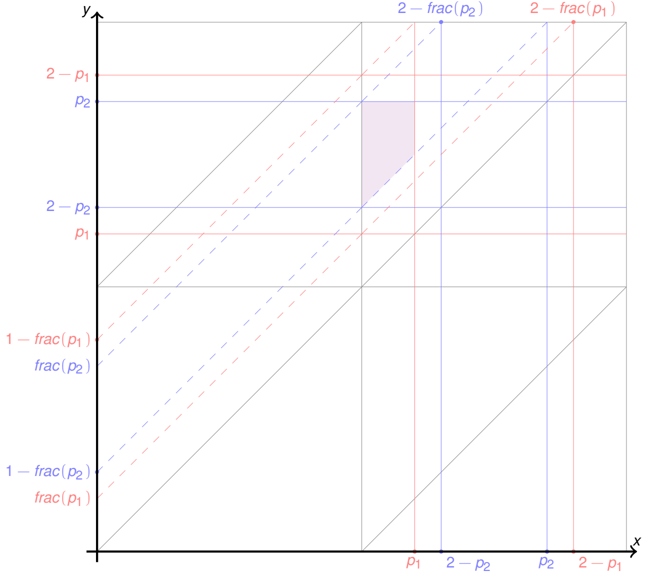
y
x
$2 - frac(p_2)$
$2 - frac(p_1)$
p_2
p_1
$1 - frac(p_1)$
$frac(p_2)$
$1 - frac(p_2)$
$frac(p_1)$
p_1
p_2

U2P-TA



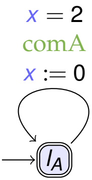
y
x
$2 - frac(p_2)$
$2 - frac(p_1)$
p_2
p_1
$1 - frac(p_1)$
$frac(p_2)$
$1 - frac(p_2)$
$frac(p_1)$
w
w
p_1
p_2

U2P-TA

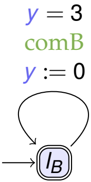


y
x
2 — frac(p2)
2 — frac(p1)
2 — p1
p2
2 — p2
p1
1 — frac(p1)
frac(p2)
1 — frac(p2)
frac(p1)
p1
2 — p2
p2
2 — p1

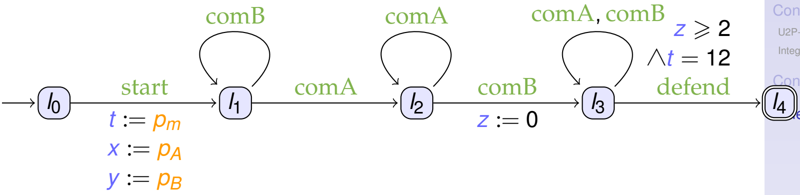
Example



(a) Committee A



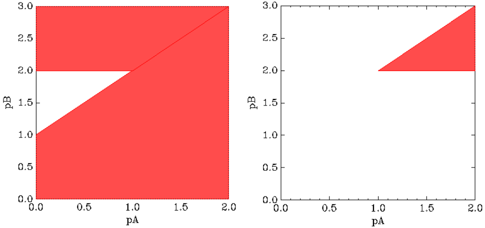
(b) Committee B



(c) A PhD student's defense workflow

Figure: A motivating example of integer-valued U2P-TA

Example



Graphical visualization in two dimensions of the parameter synthesis of with $p_m = 6$ (left) and $p_m = 9$ (right) provided by IMITATOR. Constraints are:

$$p_A \leqslant 2 \wedge p_B \leqslant p_A + 1$$
$$\vee$$
$$p_B \geqslant 2 \wedge p_B \leqslant 3 \wedge p_B \geqslant p_A + 1$$

with $p_m = 6$

$$p_B \geqslant 2 \wedge p_A \leqslant 2 \wedge p_A + 1 \geqslant p_B$$

with $p_m = 9$