Timed automata with parametric updates

ACSD 2018

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

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

June 27, 2018

Introduction

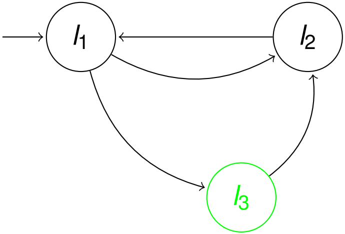
- Discovering a bug during a test of a system can be very expensive
- Can have dramatical consequences in critical embedded system: autonomous car, in aeronautics...

Introduction

- Discovering a bug during a test of a system can be very expensive
- Can have dramatical consequences in critical embedded system: autonomous car, in aeronautics...
- Need for formal verification to ensure ahead the good behavior of a system

Model checking

- Model of a system:

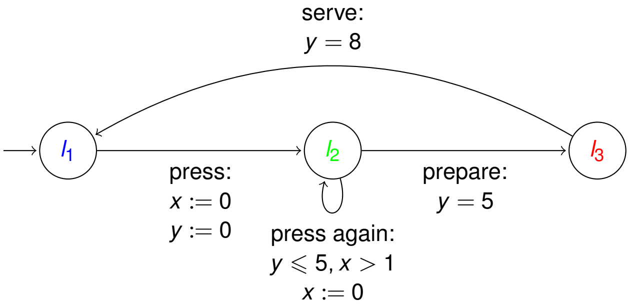


- A property of the system:  *is reachable*

- Check whether the system satisfies the property

Example of timed automaton

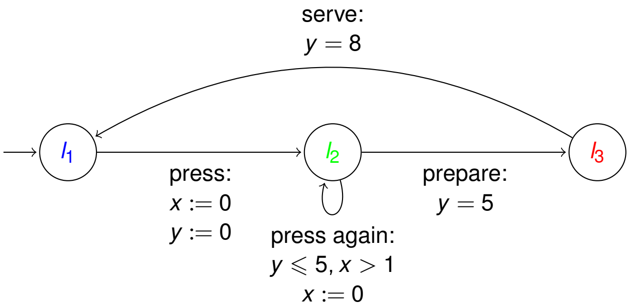
A **timed automaton** [AD94] which models a coffee machine



- 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** [AD94] which models a coffee machine



- 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.

Common decision problems for timed automata

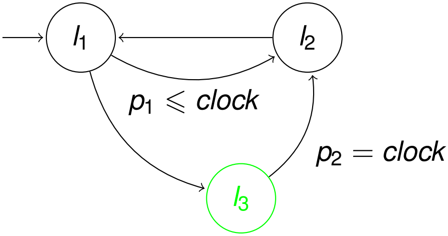
- *Reachability*: Is there a run such that the location l is reachable?
  Unavoidability: For all runs, is the location l reachable?

Common decision problems for timed automata

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

Model checking with unknown constants

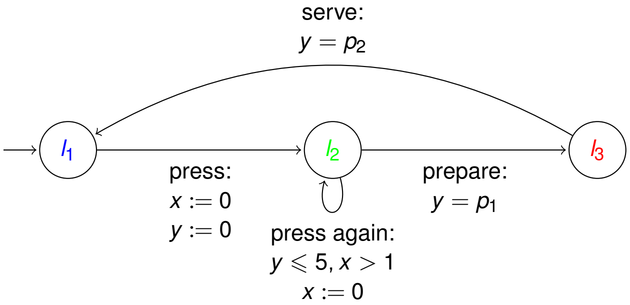
- *What if all constants are not specified ahead?*
- Model of a system with parameters:



- A property of the system: l_3 *is reachable*
- Compute the values of p_1, p_2 such that the system satisfies the property

Example of parametric timed automaton

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 $p_1 < 1$.

Challenges for parametric timed automata

- *EF-emptiness* (decision problem): is the set of parameter valuations s.t. there exists a run reaching l in the instantiated TA empty ?

  EF-synthesis (computation problem): Compute all parameter valuations s.t. there exists a run reaching l in the instantiated TA

Challenges for parametric timed automata

- *EF-emptiness* (decision problem): is the set of parameter valuations s.t. there exists a run reaching l in the instantiated TA empty ?
  EF-synthesis (computation problem): 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]

Challenges for parametric timed automata

- *EF*-*emptiness* (decision problem): is the set of parameter valuations s.t. there exists a run reaching *l* in the instantiated TA empty ?
  EF-*synthesis* (computation problem): 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]
- To recover decidability, we need to add restrictions on parameters, or restrain the PTA syntax

Where to start from ?

- Almost everything is undecidable for PTAs [And17]—especially EF-emptiness, AF-emptiness (is there a parameter valuation such that all runs reach a given location).
- Therefore, we go back to TAs.
- The reachability problem is PSPACE-complete for timed automata with updates to rational constants [BDFP04].

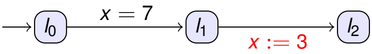


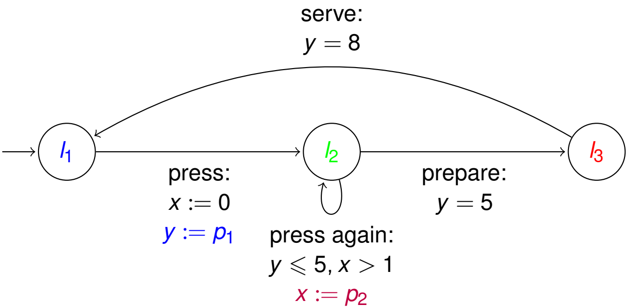
Figure: An updatable TA

Contributions

- New formalism with parametric updates of clocks: update-to-parameter TA (U2P-TA)
- Undecidability result for EF-emptiness and universality (are all parameter valuations such that there is a run reaching a given location) and AF-emptiness and universality (are all runs reaching a given location) for rational-valued parameters
- Decidability result for the same problems (in PSPACE) for integer-valued parameters, and synthesis of parameters

U2P-TA

Update-to-parameter TA (U2P-TA): TA extended with updates to rational-valued parameters.



Parametric clock updates: $y := p_1$, $x := p_2$.
Bounded parameters p_1, p_2 i.e. $p_1, p_2 \in [a, b]$ with $a, b \in \mathbb{N}$.

U2P-TA

Theorem

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

Proof sketch: we prove that a bounded PTA can be simulated by a bounded U2P-TA.

U2P-TA

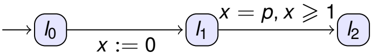


Figure: A PTA A

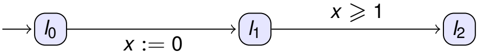


Figure: A U2P-TA obtained from A

U2P-TA

Duplicate x.

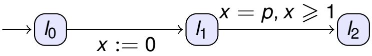


Figure: A PTA A

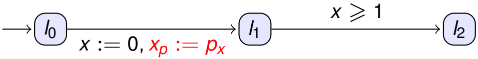


Figure: A U2P-TA obtained from A

U2P-TA

Compare x_p with C_{MAX} (maximum value between constants and parameters appearing in guards) where x is compared to p.

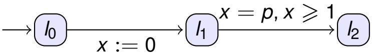


Figure: A PTA A

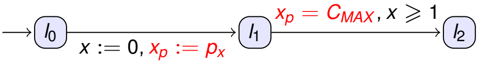


Figure: A U2P-TA obtained from A

U2P-TA

As we can simulate (w.r.t. reachability) any bounded rational-valued U2P-TA using an unbounded rational-valued U2P-TA:

Theorem

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

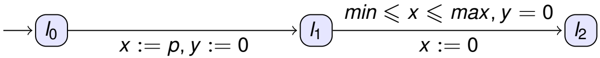


Figure: A gadget that ensures a parameter p is bounded by min and max

Integer-valued U2P-TA

U2P-TAs with integer-valued parameters over dense time.

Integer-valued U2P-TA

U2P-TAs with integer-valued parameters over dense time.

Theorem

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

Integer-valued U2P-TA

Corollary

the EF-emptiness problem is PSPACE-complete for unbounded integer-valued U2P-TAs

and *unlike integer-valued PTAs* for which EF-emptiness is undecidable [AHV93,BBLS15].

Integer-valued U2P-TA

Corollary

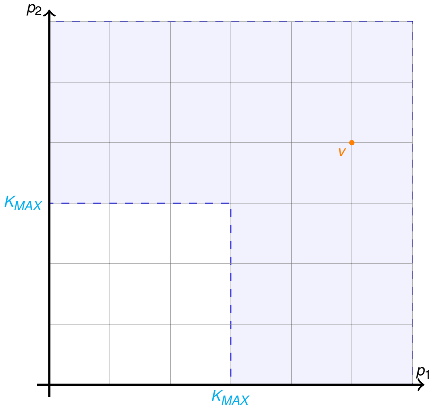
the EF-emptiness problem is PSPACE-complete for unbounded integer-valued U2P-TAs

and *unlike integer-valued PTAs* for which EF-emptiness is undecidable [AHV93,BBLS15].

Proof sketch: using equivalence between parameter valuations if $> K_{MAX}$ (the maximum constant value), we enumerate parameter valuations $\leqslant K_{MAX} + 1$ as they are bounded integers.

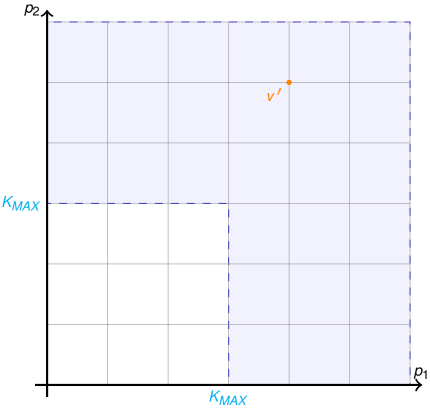
Integer-valued U2P-TA

v and v' are equivalent.



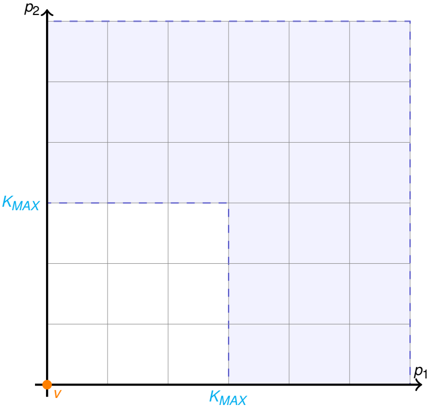
Integer-valued U2P-TA

v and v' are equivalent.



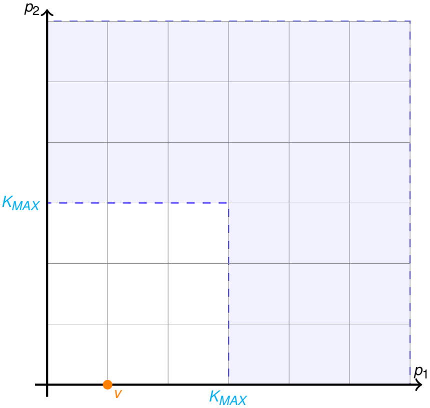
Integer-valued U2P-TA

Enumeration below $K_{MAX} + 1$.



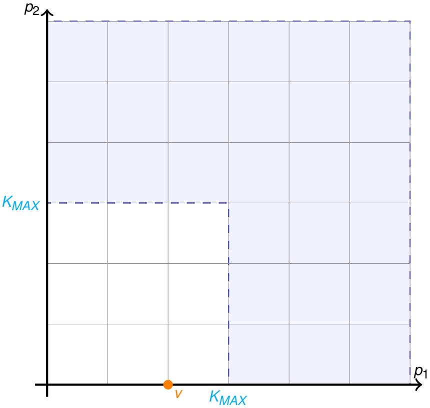
Integer-valued U2P-TA

Enumeration below $K_{MAX} + 1$.



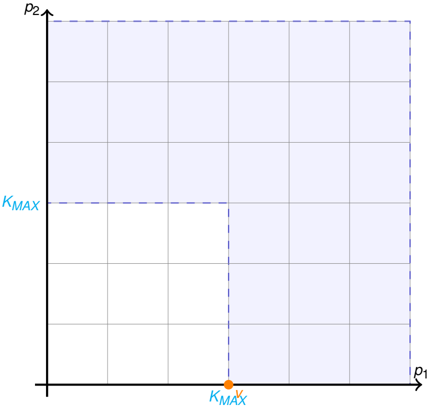
Integer-valued U2P-TA

Enumeration below $K_{MAX} + 1$.



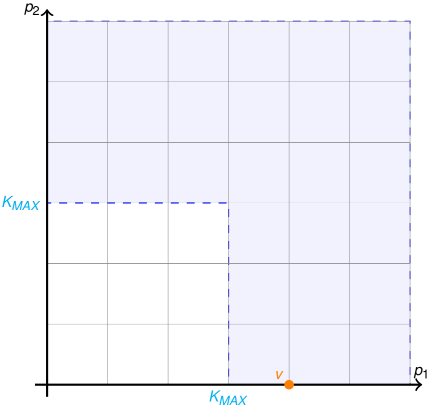
Integer-valued U2P-TA

Enumeration below $K_{MAX} + 1$.



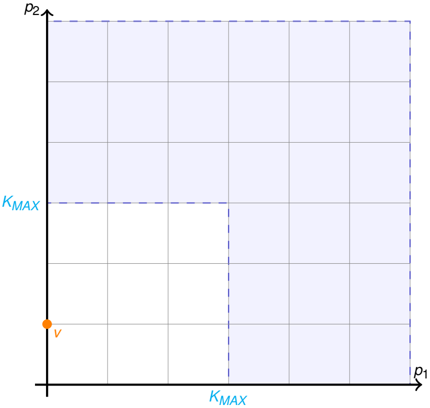
Integer-valued U2P-TA

Enumeration below $K_{MAX} + 1$.



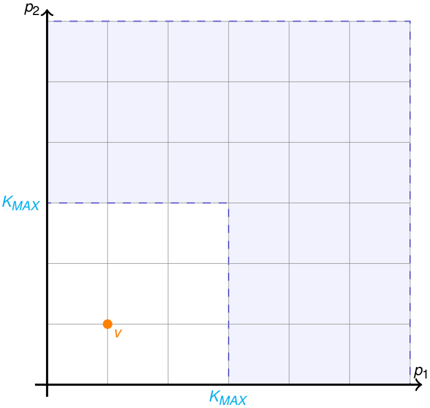
Integer-valued U2P-TA

Enumeration below $K_{MAX} + 1$.



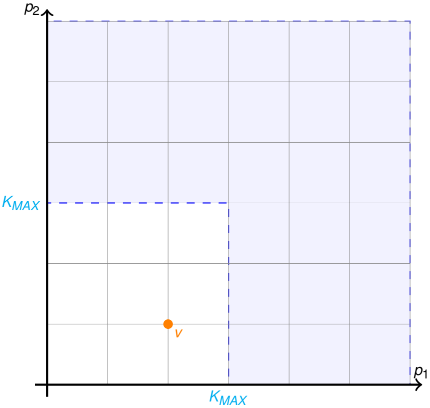
Integer-valued U2P-TA

Enumeration below $K_{MAX} + 1$.



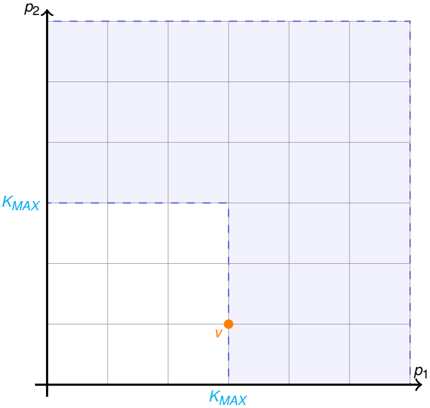
Integer-valued U2P-TA

Enumeration below $K_{MAX} + 1$.



Integer-valued U2P-TA

Enumeration below $K_{MAX} + 1$.



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*.
- In fact we have the same results for *EF*-universality, *AF*-emptiness/universality.
- We also can perform *parameter synthesis*.

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*.
- In fact we have the same results for *EF*-universality, *AF*-emptiness/universality.
- We also can perform *parameter synthesis*.

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.

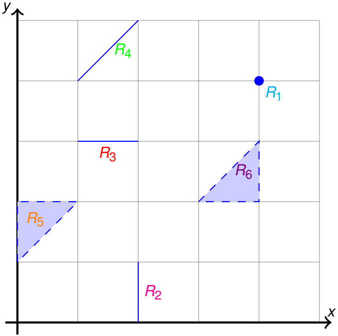
Rajeev Alur, Thomas A. Henzinger, and Moshe Y. Vardi.
Parametric real-time reasoning.
In S. Rao Kosaraju, David S. Johnson, and Alok Aggarwal, editors, *STOC*, pages 592–601, New York, NY, USA, 1993. ACM.

Étienne André.
What's decidable about parametric timed automata?
International Journal on Software Tools for Technology Transfer, 2017.
To appear.

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.

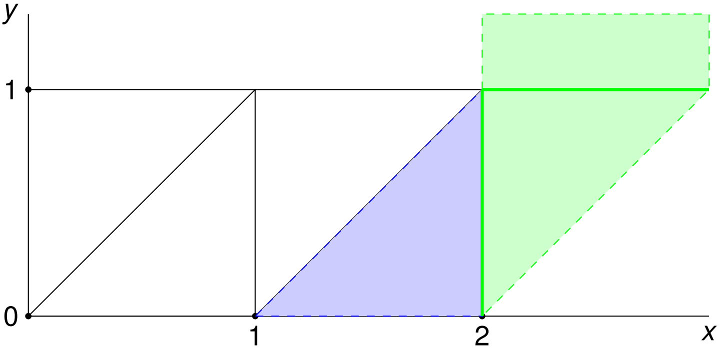
Clock regions



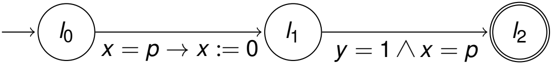
- The corner point: $R_1 = \{(4,4)\}$
- The vertical line: $R_2 = \{(x,y) \mid x = 2, 0 < y < 1\}$
- The horizontal line: $R_3 = \{(x,y) \mid y = 3, 1 < x < 2\}$
- The diagonal: $R_4 = \{(x,y) \mid x = y - 3, 4 < y < 5\}$
- The upward triangle: $R_5 = \{(x,y) \mid 0 < x < y - 1, 1 < y < 2\}$
- The downward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

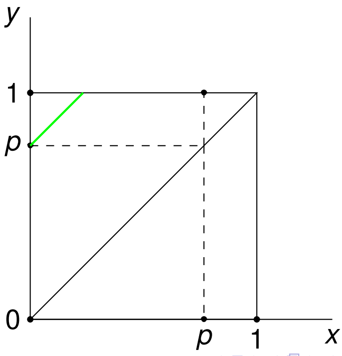


Using regions for parametric timed automata ?

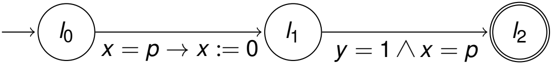


In l_1: $(x, y) = (0, p)$
But after letting some time elapse, depending on the value of $0 < p < 1$ we reach different regions:

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

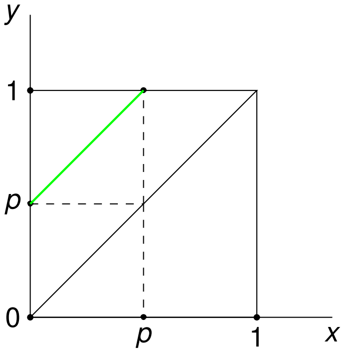


Using regions for parametric timed automata ?

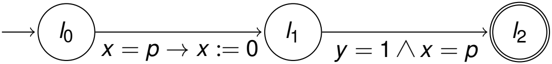


In l_1: $(x, y) = (0, p)$
But after letting some time elapse, depending on the value of $0 < p < 1$ we access different regions:

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

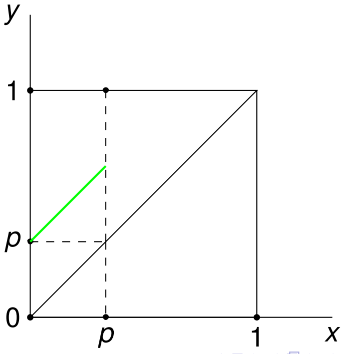


Using regions for parametric timed automata ?

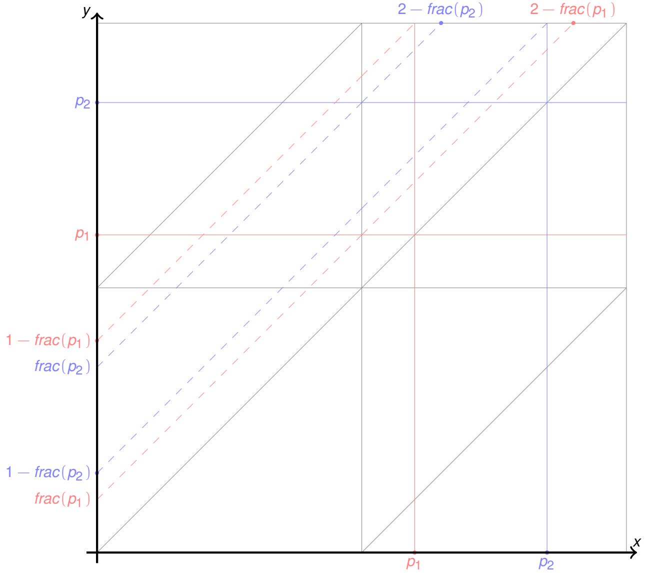


In l_1: $(x, y) = (0, p)$
But after letting some time elapse, depending on the value of $0 < p < 1$ we access different regions:

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

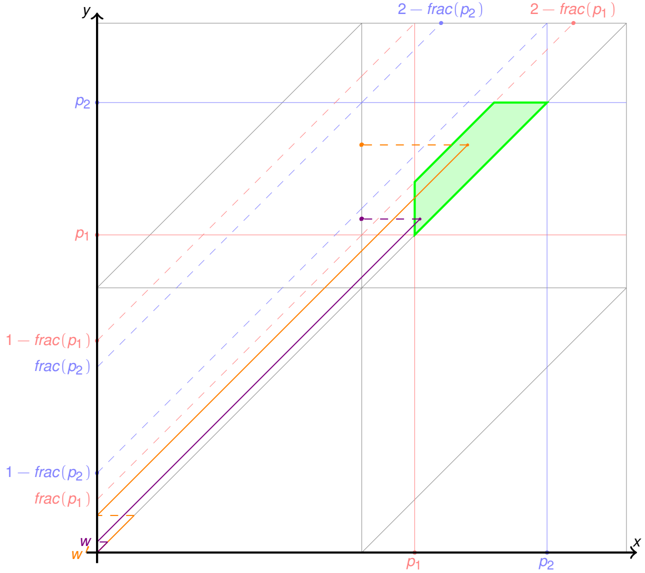


U2P-TA



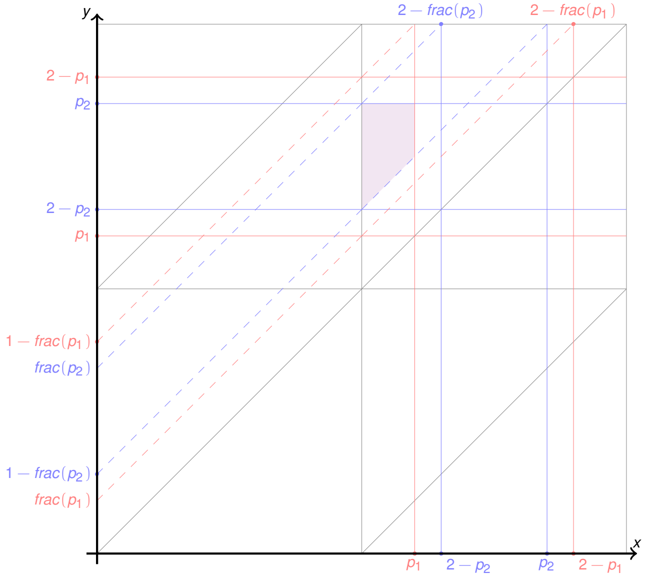
2 — frac(p2)
2 — frac(p1)
y
p2
p1
1 — frac(p1)
frac(p2)
1 — frac(p2)
frac(p1)
x
p1
p2

U2P-TA



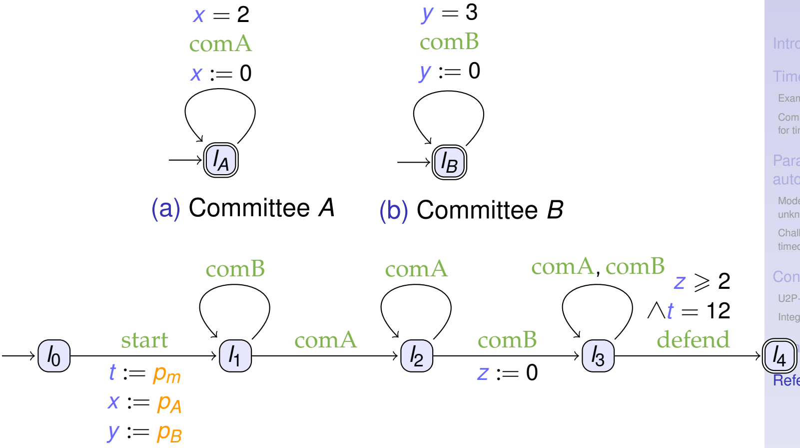
y
x
$2 - frac(p_2)$
$2 - frac(p_1)$
p_2
p_1
$1 - frac(p_1)$
$frac(p_2)$
$1 - frac(p_2)$
$frac(p_1)$
w
w'
p_1
p_2

U2P-TA



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

Example



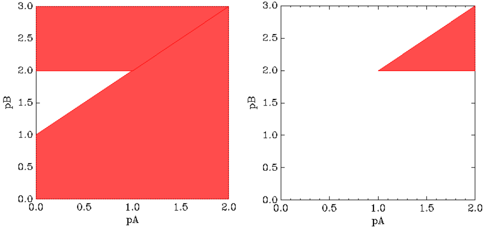
(a) Committee A

(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 $p_m = 9$