Computing maximally-permissive strategies in acyclic timed automata

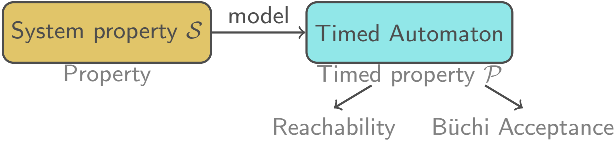
Emily Clement[1,2] Thierry Jéron[1] Nicolas Markey[1] David Mentré[2]

[1]IRISA, Inria & CNRS & Univ. Rennes, France
[2]Mitsubishi Electric R&D Centre Europe – Rennes, France

March 29, 2023

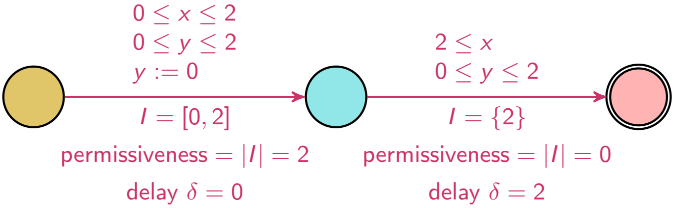
Context & Motivations - Verify properties despite perturbations

- Mathematical model with perfect clocks



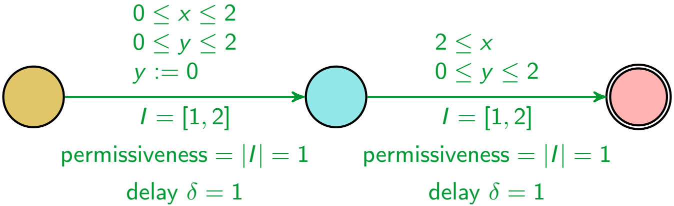
- Robustness
  - ▷ Clocks are **imperfects**
  - ▷ **Robustness**:
    (1) model these imperfections
    (2) verify $\mathcal{P}$ despite these imperfections.

- A run and its robustness



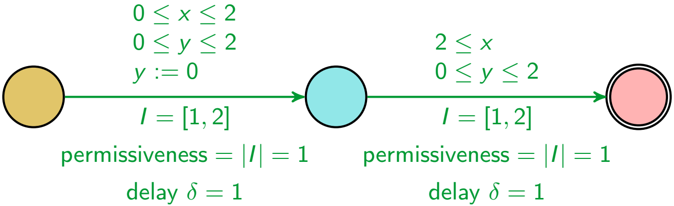
Permissiveness: $\min(0, 2) = 0$

- A run and its robustness



Permissiveness: $\min(1, 1) = 1$

- A run and its robustness



Permissiveness: $\min(1, 1) = 1$

- Our definition of robustness: the permissiveness function
  - ▷ The permissiveness function of a **run** is the **size** of the shortest interval that the player has proposed.
  - ▷ We introduce a **player** (choice of **intervals** I) and an **opponent** (choice of **delays** δ)
  - ▷ The permissiveness function of a configuration (l, v) is the permissiveness of the run where the **player maximizes** the permissiveness and the **opponent minimizes** it.

Introduction - State of the art of the robustness

- Topological robustness
  - ▷ **Gupta, Henzinger, Jagadeesan** "Robust Timed Automata", 1997
  - ▷ Tools: stability theorems.
- Guard enlargement
  - ▷ **Sankur** "Robustness in Timed Automata",PhD Thesis, 2013
  - ▷ Tools: game theory, parameterized DBM.
- Delay enlargement
  - ▷ **Bouyer, Fang, Markey** "Permissive strategies in timed automata and games", AVOCS'15
  - ▷ Tools: game theory
  - ▷ An algorithm: ✓
  - ▷ Multiple clocks: ✗.

Introduction - Our goal

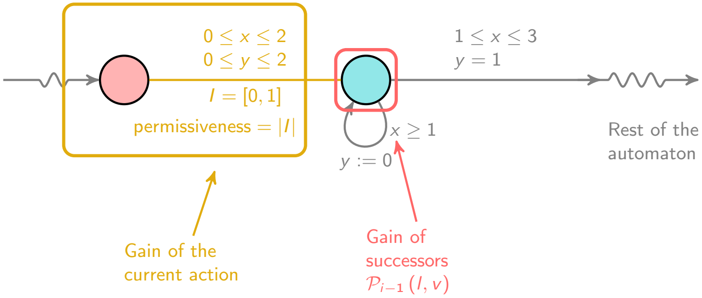
- Define our semantic of robustness:
  - ▷ We take a context of **reachability** and of **worst cases**.
  - ▷ We will call this robustness the **permissiveness function**.
- Construct an algorithm that answers the following question:

For a timed automaton $\mathcal{A}$ and a location l, compute the permissiveness function.

- Our Method
  - ▷ Construct an algorithm that computes **exactly** the robustness of any automaton/configuration.

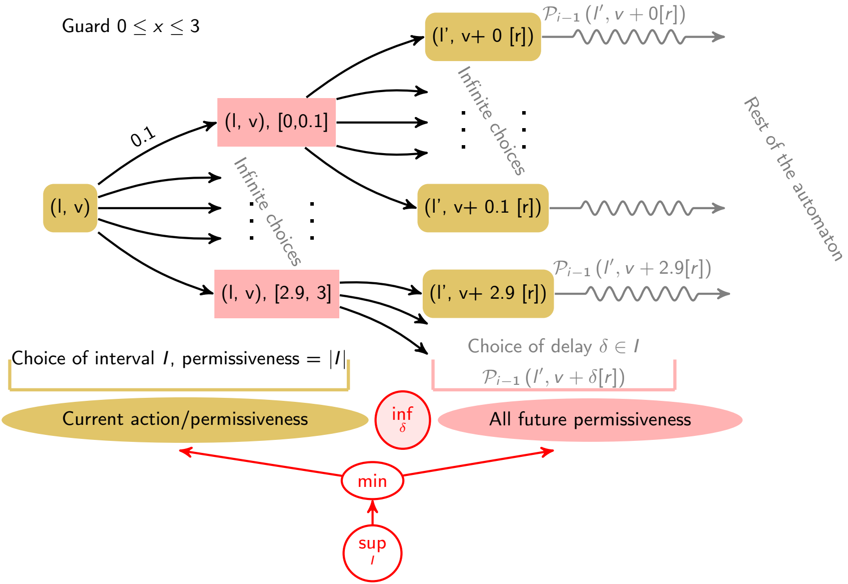
- The permissiveness: a way to quantify robustness
  - ▷ permissiveness $\searrow$ = robustness $\searrow$
  - ▷ A recursive calculus of a function $\mathcal{P}_i(I, v)$.

- A recursive algorithm to compute the permissiveness



Gain of the automaton: minimum of current permissiveness and the permissiveness of the successors

Permissiveness computation - What is the permissiveness?



Permissiveness computation - The formula to compute the permissiveness

- Algorithm by steps

We denote $moves(I, v)$ the set of available (interval,action):

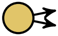
- ▷ Step 0, if $I = I_f$, $\mathcal{P}_0(I, v) = +\infty$, if not, 0
- ▷ Step i, if $moves(I, v) = \varnothing$, $\mathcal{P}_i(I, v) = 0$, if not

$$\mathcal{P}_i(I, v) = \sup_{(a,I) \in moves(I,v)} \min\left(|I|, \inf_{\delta \in I} \mathcal{P}_{i-1}(\text{succ}(v, I, \delta, a))\right).$$

- ▷ The sequence converges to the permissiveness function for acyclic automata **in a finite number of steps**

- Two player games
  - ▷ Player: choice of the moves $(a, I) \in moves(I, v)$
  - ▷ Opponent: choice of the delays $\delta \in I$

- Issues
  - ▷ inf / sup: **infinite** choices & **opposite** strategies: determine a finite number of strategies to test of the two players: inf ⇒ min and sup ⇒ max.
  - ▷ $\mathcal{P}_i(I, v)$ has to be computed for all v.

We consider only linear automata :no .

- Lemma for linear T.A

$v \mapsto \mathcal{P}_i(l, v)$ is a **concave** function over the set of valuations.

- Consequences

If the **player** proposes the interval $[\alpha, \beta]$, the best strategy of the opponent is to propose the delay α **or** β

$$\mathcal{P}_i(l, v) = \sup_{(a,I) \in moves(l,v)} \min\left(|I|, \inf_{\delta \in I} \mathcal{P}_{i-1}(\mathrm{succ}(v, l, \delta, a))\right) \text{ becomes}$$

$$\mathcal{P}_i(l, v) = \sup_{([\alpha,\beta],a) \in moves(l,v)} \min(|\beta - \alpha|, \min_{\delta=\alpha,\beta} \mathcal{P}_{i-1}(\mathrm{succ}(v, l, \delta, a))).$$

- Next step
  - ▷ sup → max
  - ▷ That means, determine the strategy of the player.

Strategy of the player for linear automata - the steps.

$$\mathcal{P}_i\left(l, v\right) = \sup_{([\alpha,\beta],a)\in moves(l,v)} \min(|\beta - \alpha|, \min_{\delta=\alpha,\beta} \mathcal{P}_{i-1}\left(\text{succ}\left(v, l, \delta, a\right)\right))$$

- Goal: Find the interval $[\alpha, \beta]$ that maximizes:

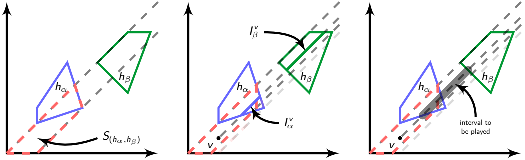
$$\min(|\beta - \alpha|, \mathcal{P}_{i-1}\left(\text{succ}\left(v, l, \alpha, a\right)\right), \mathcal{P}_{i-1}\left(\text{succ}\left(v, l, \beta, a\right)\right))$$

- Tool-Lemma: Proprerty of the permissiveness function

For **any** i **and any location** l, $v \mapsto \mathcal{P}_i\left(l, v\right)$ is an n-dim **piecewise-affine function**, with bounded number of pieces.

- Issue: How to optimize the minimum of three **piece-wise** affine functions?
  - ▷ (1) **"Fix"** the pieces where $v + \alpha[r]$ and $v + \beta[r]$ ends up: an algorithm
  - ▷ (2) **Optimize** the minimum of three **affine** functions: a technical lemma

Strategy of the player for linear automata - The algorithm.



- Goal: which interval $[\alpha, \beta]$ maximizes

$$\min(|\beta - \alpha|, \mathcal{P}_{i-1}(\mathrm{succ}(v, I, \alpha, a)), \mathcal{P}_{i-1}(\mathrm{succ}(v, I, \beta, a)))?$$

- Steps of the algorithm:
  - (1) **Fix** two arbitrary cells h_α, h_β s.t. $v + \alpha[r] \in h_\alpha$ and $v + \beta[r] \in h_\beta$
  - (2) Compute $S_{h_\alpha,h_\beta} = \{v \in \mathbb{R}^n | \exists \alpha, \beta, v + \alpha[r] \in h_\alpha, v + \beta[r] \in h_\beta\}$
  - (3) **Fix** $v \in S_{h_\alpha,h_\beta}$ and compute the intervals of enabled α, β: I_α^v, I_β^v
  - (4) The technical lemma: find such α and β in $I_\alpha^v \times I_\beta^v$ s.t $\alpha \leq \beta$ that maximizes

$$\min(\beta - \alpha, \mathcal{P}_i(I, v + \alpha[r]), \mathcal{P}_i(I, v + \beta[r])).$$

  - (5) Iterate for all pieces and compare

To maximize the quantity $\min(\beta - \alpha, a\alpha + b, c\beta + d)$ over α and β in $[m_\alpha, M_\alpha] \times [m_\beta, M_\beta]$ s.t $\alpha \leq \beta$:

- Detail of the case: $a \geq 0$ and $c \geq 0$

Condition	coordinates of maximal point	value of maximal point
$\frac{M_\beta - b}{a+1} \leq m_\alpha$	(m_α, M_β)	$\min\{M_\beta - m_\alpha, cM_\beta + d\}$
$m_\alpha \leq \frac{M_\beta - b}{a+1} \leq \min\{M_\alpha, M_\beta\}$	$(\frac{M_\beta - b}{a+1}, M_\beta)$	$\min\{\frac{aM_\beta + b}{a+1}, cM_\beta + d\}$
$\min\{M_\alpha, M_\beta\} \leq \frac{M_\beta - b}{a+1}$	$(\min\{M_\alpha, M_\beta\}, M_\beta)$	$\min\{aM_\alpha + b, aM_\beta + b, cM_\beta + d\}$

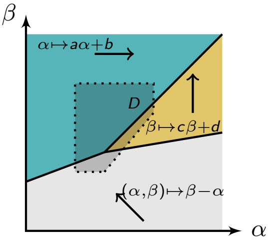
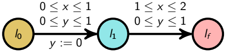


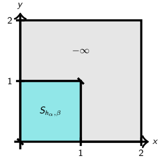
Figure: Value of $\min(\beta - \alpha, a\alpha + b, c\beta + d)$ over $\mathbb{R}^2$, where $D = \{\alpha \in [m_\alpha, M_\alpha], \beta \in [m_\beta, M_\beta] | \alpha \leq \beta\}$

- Other cases: similar.

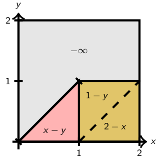
Example of this strategy



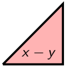
(a) A two-transitions automaton



(b) Gain in l_0

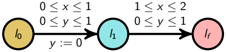


(c) Gain in l_1

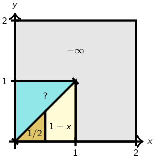
- Let's take $h_\alpha = h_\beta =$ 

. Then $S_{h_\alpha,h_\beta} =$ 

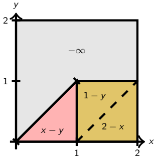
- For $v = (x, y)$, $I_\alpha^v = [0, min(1-x, 1-y)]$ and $I_\beta^v = [0, min(1-x, 1-y)]$
- Suppose that $1 - x < 1 - y$ then $I_\alpha^v = [0, 1-x]$ and $I_\beta^v = [0, 1-x]$
- Let's find $\alpha < \beta$ in $I_\alpha^v \times I_\beta^v$ that maximizes $\min(\beta - \alpha, 1 \cdot \alpha + x, 1 \cdot \beta + x)$
- The technical lemma application:
  $a = c = 1 \geq 0, \frac{M_\beta - b}{a+1} = \frac{1-x-1}{1+1} = x/2, m_\alpha = 0, \min\{M_\alpha, M_\beta\} = 1 - x.$
- If $x > 1/2$ then , $\mathcal{P}_2(l_0, v) = 1 - x$, otherwise $1/2$



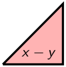
(a) A two-transitions automaton



(b) Gain in l_0



(c) Gain in l_1

- Let's take $h_\alpha = h_\beta =$ 

. Then $S_{h_\alpha, h_\beta} =$ 

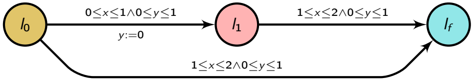
- For $v = (x, y)$, $I^v_\alpha = [0, min(1-x, 1-y)]$ and $I^v_\beta = [0, min(1-x, 1-y)]$
- Suppose that $1-x < 1-y$ then $I^v_\alpha = [0, 1-x]$ and $I^v_\beta = [0, 1-x]$
- Let's find $\alpha < \beta$ in $I^v_\alpha \times I^v_\beta$ that maximizes $\min(\beta - \alpha, 1 \cdot \alpha + x, 1 \cdot \beta + x)$
- The technical lemma application:
  $a = c = 1 \geq 0, \frac{M_\beta - b}{a+1} = \frac{1-x-1}{1+1} = x/2, m_\alpha = 0, \min\{M_\alpha, M_\beta\} = 1-x.$
- If $x > 1/2$ then , $\mathcal{P}_2\,(l_0, v) = 1-x$, otherwise $1/2$

- Linear automata

For a linear timed automaton, with d locations and n clocks, the permissiveness function is a **piecewise-affine concave** function and can be computed in time $\mathcal{O}(n+1)^{8^d}$, so in **double-exponential time.**

- Acyclic automata & timed games

For an acyclic timed automaton or for timed games the permissiveness function is a **piecewise-affine** function and can be computed **non-elementary time**



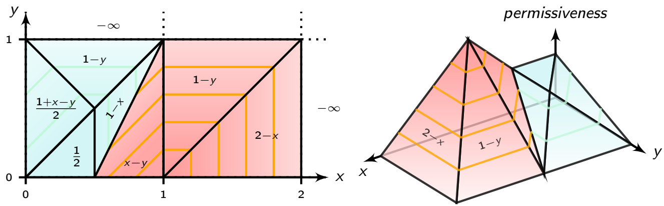


Figure: A timed automaton and its (non-concave) permissiveness function in l_0

Conclusion - Our contribution

A sequence $\mathcal{P}_i(l, v)$ — converges to → Permissiveness function

how to compute?

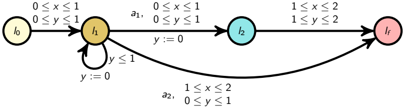
$$\mathcal{P}_i(l, v) = \sup_{(a,[\alpha,\beta]) \in moves(l,v)} \min \left[|I|, \inf_{\delta \in [\alpha,\beta]} \mathcal{P}_{i-1}(succ(v, I, \alpha, a)) \right]$$

Lemma

$$\mathcal{P}_i(l, v) = \sup_{(a,[\alpha,\beta]) \in moves(l,v)} \min \left[|\beta - \alpha|, \mathcal{P}_{i-1}(succ(v, I, \alpha, a)), \mathcal{P}_{i-1}(succ(v, I, \beta, a)) \right]$$

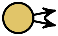
Optimization lemma & algorithm

Contribution



- Achieved works

Computation of the robustness:

- Operator: max.
- : ✓
- : ✓
- : ✓
- Timed games: ✓
- Constructive algorithm and worst-case complexity: ✓

- Future works

- 
- Implementation (Python)
- General permissiveness function
- Binary robustness
- Stochastic opponent