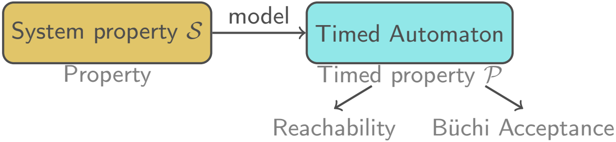
Computing maximally-permissive strategies in acyclic timed automata

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

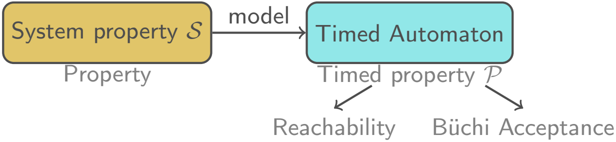
[1]IRISA, Inria & CNRS & Univ. Rennes, France
[2]Mitsubishi Electric R&D Centre Europe – Rennes, France

1-3 September 2020

- Mathematical model with perfect clocks

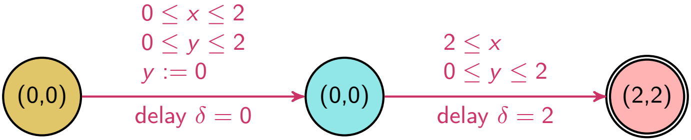


- Mathematical model with perfect clocks

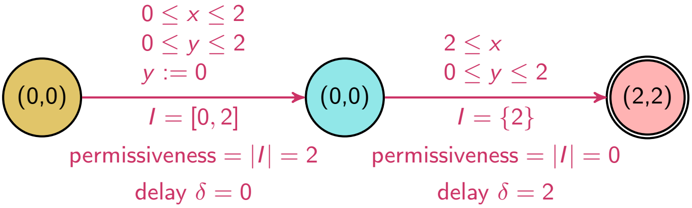


- Robustness
  - ▷ Clocks are **imperfects**
  - ▷ **Robustness**:
    (1) model these imperfections
    (2) verify $\mathcal{P}$ despite these imperfections.

- Classical semantics

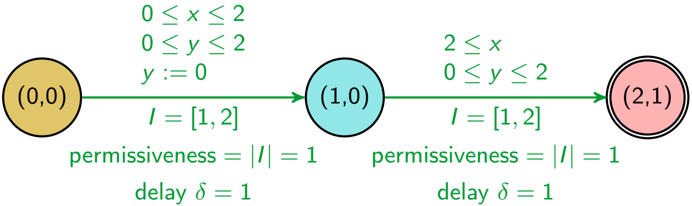


- An example (run)



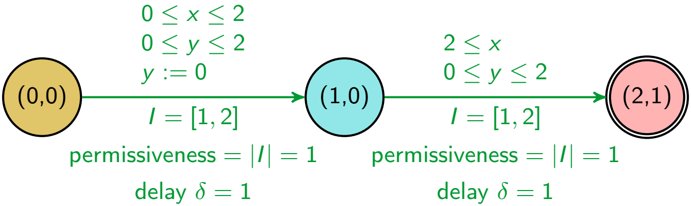
Permissiveness: $\min(0, 2) = 0$

- An example (run)



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

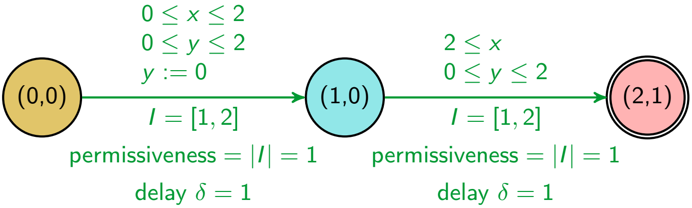
- An example (run)



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

- Our semantics:
  - ▷ Choice of **intervals** & **action**: player
  - ▷ Choice of **delays**: opponent
  - ▷ Permissiveness of the **run**: the **smallest** interval proposed

- An example (run)



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

- Our semantics:
  - ▷ Choice of **intervals** & **action**: player
  - ▷ Choice of **delays**: opponent
  - ▷ Permissiveness of the **configuration**: the **smallest** interval proposed **during the run when the player maximizes it/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: ✗.

- Define our semantics of robustness:
  - ▷ We take a context of **reachability** and of **worst cases**.
  - ▷ We will call this robustness the **permissiveness function**.

Introduction - Our goal

- Define our semantics 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:

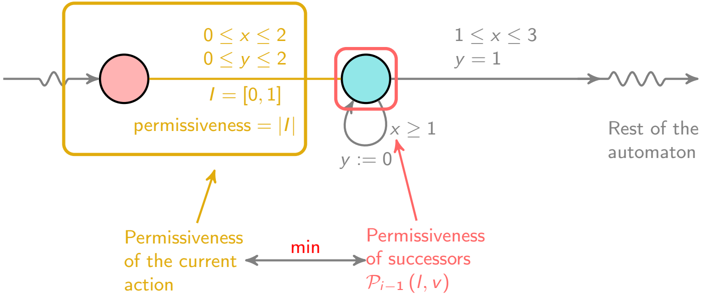
For a timed automaton $\mathcal{A}$ and a location l, compute the permissiveness function.

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

- The permissiveness: a way to quantify robustness
  - ▷ A sequence $\mathcal{P}_i\,(l, v)$ to compute the permissiveness function (**its limit**)

- The permissiveness: a way to quantify robustness
  - A sequence $\mathcal{P}_i(I, v)$ to compute the permissiveness function (**its limit**)

- A recursive algorithm to compute the permissiveness

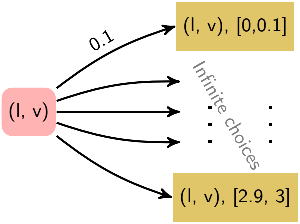


Permissiveness computation - What is the permissiveness?

Guard $0 \leq x \leq 3$

(l, v)

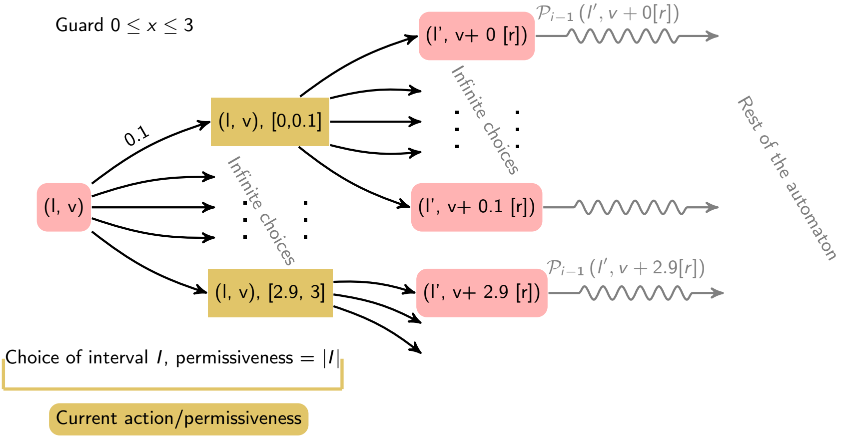
Guard $0 \leq x \leq 3$

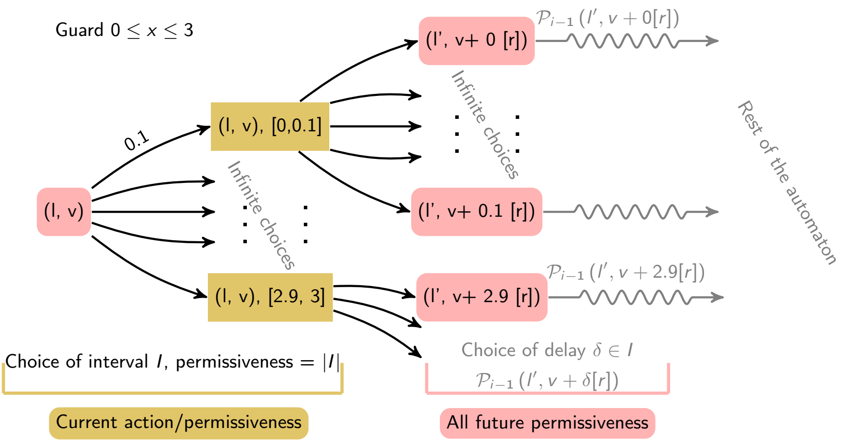


Choice of interval I, permissiveness $= |I|$

Current action/permissiveness

Permissiveness computation - What is the permissiveness?




Guard $0 \leq x \leq 3$
(l, v)
0.1
(l, v), [0,0.1]
(l, v), [2.9, 3]
Infinite choices
(l', v+ 0 [r])
(l', v+ 0.1 [r])
(l', v+ 2.9 [r])
Infinite choices
$\mathcal{P}_{i-1}(l', v + 0[r])$
$\mathcal{P}_{i-1}(l', v + 2.9[r])$
Rest of the automaton
Choice of interval I, permissiveness $= |I|$
Current action/permissiveness
Choice of delay $\delta \in I$
$\mathcal{P}_{i-1}(l', v + \delta[r])$
All future permissiveness

- Formula of $\mathcal{P}_i\,(l, v)$
  - ▷ $moves(l, v)$: set of available (interval,action).
  - ▷ If $l = l_f$:

$$\boxed{\mathcal{P}_i\,(l, v) = +\infty.}$$

- Formula of $\mathcal{P}_i(l, v)$
  - ▷ *moves*(l, v): set of available (interval,action).
  - ▷ If $i = 0$ and $l \neq l_f$:

$$\mathcal{P}_i(l, v) = 0.$$

- Formula of $\mathcal{P}_i\,(l, v)$
  - ▷ $moves(l, v)$: set of available (interval,action).
  - ▷ If $i > 0$, $l \neq l_f$ and $moves(l, v) = \varnothing$

$$\boxed{\mathcal{P}_i\,(l, v) = 0.}$$

- Formula of $\mathcal{P}_i(l, v)$
  - ▷ $moves(l, v)$: set of available (interval,action).
  - ▷ If $i > 0$, $l \neq l_f$ and if $moves(l, v) \neq \varnothing$:

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

- Formula of $\mathcal{P}_i(l, v)$
  - $moves(l, v)$: set of available (interval,action).
  - If $i > 0$, $l \neq l_f$ and if $moves(l, v) \neq \varnothing$:

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

- Convergence of the sequence
  - Convergence in a **finite number of steps** for acyclic automata.
  - Number of necessary steps: maximal distance $l \longleftrightarrow l_f$

- Formula of $\mathcal{P}_i\left(l, v\right)$
  - $moves(l, v)$: set of available (interval,action).
  - If $i > 0$, $l \neq l_f$ and if $moves(l, v) \neq \varnothing$:

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

- Convergence of the sequence
  - Convergence in a **finite number of steps** for acyclic automata.
  - Number of necessary steps: maximal distance $l \longleftrightarrow l_f$

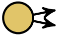
- Issues ⚠
  - inf / sup: **infinite** choices & **opposite** strategies.
  - $\mathcal{P}_i\left(l, v\right)$ has to be computed for all v.

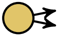
A sequence $\mathcal{P}_i(I, v)$ —converges to→ Permissiveness function

↓ how to compute?

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

- Next step
  - inf $\rightarrow$ min
  - That means, determine the strategy of the opponent.

We consider only linear automata :no .

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.

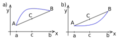
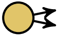


Figure: Example of a concave function

Strategy of the opponent for linear automata

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.

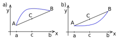


Figure: Example of a concave function

- Consequences

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

Summary

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

↓ how to compute?

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

Lemma

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

- Next step
  - sup $\to$ max
  - That means, determine the strategy of the player.

Strategy of the player for linear automata - the steps.

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

Strategy of the player for linear automata - the steps.

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

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

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

Strategy of the player for linear automata - the steps.

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

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

$\min(|\beta - \alpha|, \mathcal{P}_{i-1}(l', v + \alpha[r]), \mathcal{P}_{i-1}(l', v + \beta[r]))$

Strategy of the player for linear automata - the steps.

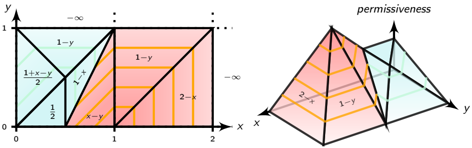
$$\mathcal{P}_i(l, v) = \sup_{([\alpha,\beta],a)\in moves(l,v)} \min(|\beta - \alpha|, \min_{\delta=\alpha,\beta} \mathcal{P}_{i-1}(\mathrm{succ}(v, l, \delta, a)))$$

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

$\min(|\beta - \alpha|, \mathcal{P}_{i-1}(l', v + \alpha[r]), \mathcal{P}_{i-1}(l', v + \beta[r]))$

- Tool-Lemma: Proprerty of the permissiveness function

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



Strategy of the player for linear automata - the steps.

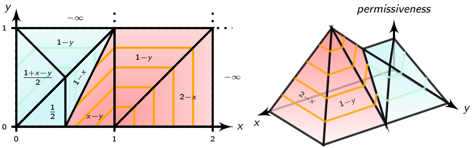
$$\mathcal{P}_i(l, v) = \sup_{([\alpha,\beta],a)\in moves(l,v)} \min(|\beta-\alpha|, \min_{\delta=\alpha,\beta} \mathcal{P}_{i-1}(\text{succ}(v, l, \delta, a)))$$

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

$\min(|\beta - \alpha|, \mathcal{P}_{i-1}(l', v + \alpha[r]), \mathcal{P}_{i-1}(l', v + \beta[r]))$

- Tool-Lemma: Proprerty of the permissiveness function

For **any** i **and any location** l, $v \mapsto \mathcal{P}_i(l, v)$ is a continuous 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 $v + \beta[r]$ end up: an algorithm

Strategy of the player for linear automata - the steps.

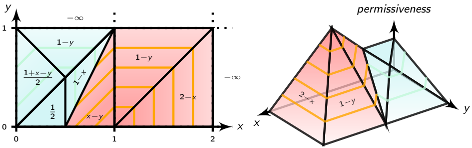
$$\mathcal{P}_i(l, v) = \sup_{([\alpha,\beta],a)\in moves(l,v)} \min(|\beta - \alpha|, \min_{\delta=\alpha,\beta} \mathcal{P}_{i-1}(\text{succ}(v, l, \delta, a)))$$

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

$\min(|\beta - \alpha|, \mathcal{P}_{i-1}(l', v + \alpha[r]), \mathcal{P}_{i-1}(l', v + \beta[r]))$

- Tool-Lemma: Proprerty of the permissiveness function

For **any** i **and any location** l, $v \mapsto \mathcal{P}_i(l, v)$ is a continuous 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 $v + \beta[r]$ end up: an algorithm
  - ▷ (2) **Optimize** the minimum of three **affine** functions: a technical lemma

Strategy of the player for linear automata - The algorithm.

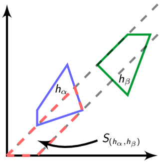
- Goal: which interval $[\alpha, \beta]$ maximizes

$$\min(|\beta - \alpha|, \mathcal{P}_{i-1}(l', v + \alpha[r]), \mathcal{P}_{i-1}(l', v + \beta[r]))?$$

- Steps of the algorithm:

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

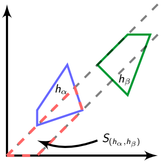
$$\min(|\beta - \alpha|, \mathcal{P}_{i-1}(l', v + \alpha[r]), \mathcal{P}_{i-1}(l', v + \beta[r]))?$$



- Steps of the algorithm:
  - ▷ (1) **Fix** two cells h_α, h_β s.t. $v + \alpha[r] \in h_\alpha$ and $v + \beta[r] \in h_\beta$

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

$$\min(|\beta - \alpha|, \mathcal{P}_{i-1}(l', v + \alpha[r]), \mathcal{P}_{i-1}(l', v + \beta[r]))?$$

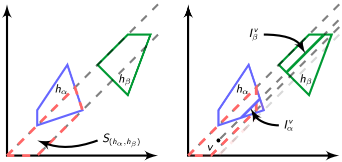


- Steps of the algorithm:
  - ▷ (1) **Fix** two cells h_α, h_β s.t. $v + \alpha[r] \in h_\alpha$ and $v + \beta[r] \in h_\beta$
  - ▷ (2) **Compute** $S_{h_\alpha, h_\beta} = \{v \in \mathbb{R}^n | \exists \alpha, \beta, v + \alpha[r] \in h_\alpha, v + \beta[r] \in h_\beta\}$

Strategy of the player for linear automata - The algorithm.

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

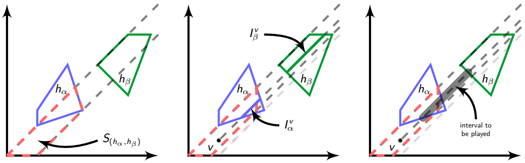
$$\min(|\beta - \alpha|, \mathcal{P}_{i-1}(l', v + \alpha[r]), \mathcal{P}_{i-1}(l', v + \beta[r]))?$$



- Steps of the algorithm:
  - ▷ (1) **Fix** two 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^v_α, I^v_β

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

$$\min(|\beta - \alpha|, \mathcal{P}_{i-1}(l', v + \alpha[r]), \mathcal{P}_{i-1}(l', v + \beta[r]))?$$



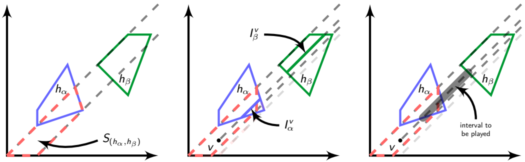
- Steps of the algorithm:
  - ▷ (1) **Fix** two cells h_α, h_β s.t. $v + \alpha[r] \in h_\alpha$ and $v + \beta[r] \in h_\beta$
  - ▷ (2) **Compute** $S_{h_\alpha, h_\beta} = \{v \in \mathbb{R}^n | \exists \alpha, \beta, v + \alpha[r] \in h_\alpha, v + \beta[r] \in h_\beta\}$
  - ▷ (3) **Fix** $v \in S_{h_\alpha, h_\beta}$ 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

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

Strategy of the player for linear automata - The algorithm.

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

$$\min(|\beta - \alpha|, \mathcal{P}_{i-1}(l', v + \alpha[r]), \mathcal{P}_{i-1}(l', v + \beta[r]))?$$



- Steps of the algorithm:
  - ▷ (1) **Fix** two 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^v_α, I^v_β
  - ▷ (4) The technical lemma: find such α and β in $I^v_\alpha \times I^v_\beta$ s.t $\alpha \leq \beta$ that maximizes

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

  - ▷ (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 $\alpha \leq \beta$:

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)	$\min\{M_\beta - m_\alpha, cM_\beta + d\}$
$m_\alpha \leq \frac{M_\beta - b}{a+1} \leq \min\{M_\alpha, M_\beta\}$	$(\frac{M_\beta - b}{a+1}, M_\beta)$	$\min\{\frac{aM_\beta + b}{a+1}, cM_\beta + d\}$
$\min\{M_\alpha, M_\beta\} \leq \frac{M_\beta - b}{a+1}$	$(\min\{M_\alpha, M_\beta\}, M_\beta)$	$\min\{aM_\alpha + b, aM_\beta + b, cM_\beta + d\}$

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)	$\min\{M_\beta - m_\alpha, cM_\beta + d\}$
$m_\alpha \leq \frac{M_\beta - b}{a+1} \leq \min\{M_\alpha, M_\beta\}$	$(\frac{M_\beta - b}{a+1}, M_\beta)$	$\min\{\frac{aM_\beta + b}{a+1}, cM_\beta + d\}$
$\min\{M_\alpha, M_\beta\} \leq \frac{M_\beta - b}{a+1}$	$(\min\{M_\alpha, M_\beta\}, M_\beta)$	$\min\{aM_\alpha + b, aM_\beta + b, cM_\beta + d\}$

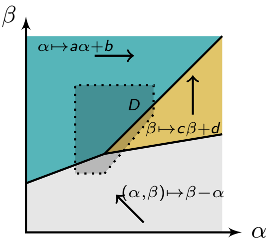
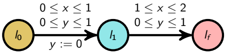
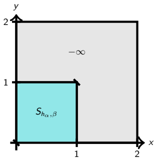


Figure: Value of $\min(\beta - \alpha, a\alpha + b, c\beta + d)$ over $\mathbb{R}^2$, where $D = \{\alpha \in [m_\alpha, M_\alpha], \beta \in [m_\beta, M_\beta] | \alpha \leq \beta\}$

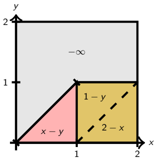
Example of this strategy



(a) A two-transitions automaton

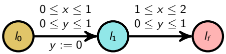


(b) Permissiveness in l_0

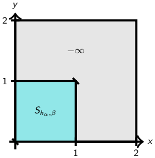


(c) Permissiveness in l_1

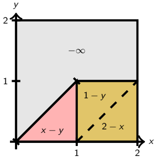
Example of this strategy



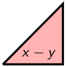
(a) A two-transitions automaton



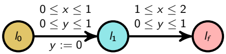
(b) Permissiveness in l_0



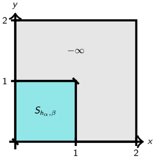
(c) Permissiveness in l_1

▷ Let's take $h_\alpha = h_\beta =$ 

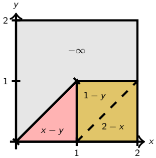
. Then $S_{h_\alpha,h_\beta} =$ 



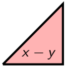
(a) A two-transitions automaton



(b) Permissiveness in l_0

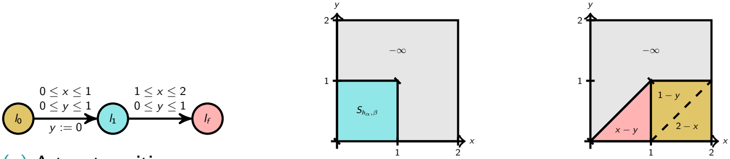


(c) Permissiveness in l_1

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

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

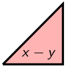
- For $v = (x, y)$, $I_\alpha^v = [0, min(1-x, 1-y)]$ and $I_\beta^v = [0, min(1-x, 1-y)]$



(a) A two-transitions automaton

(b) Permissiveness in l_0

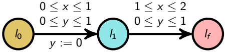
(c) Permissiveness in l_1

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

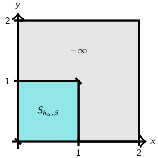
. Then $S_{h_\alpha, h_\beta} =$ 

- 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 $I^v_\beta = [0, 1 - x]$

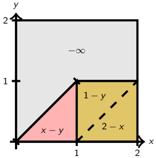
Example of this strategy



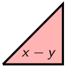
(a) A two-transitions automaton



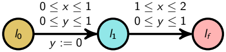
(b) Permissiveness in l_0



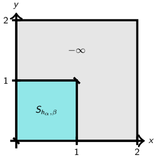
(c) Permissiveness in l_1

- 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 $\min(\beta - \alpha, 1 \cdot \alpha + x, 1 \cdot \beta + x)$

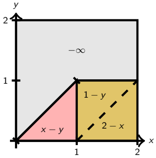
Example of this strategy



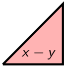
(a) A two-transitions automaton



(b) Permissiveness in l_0



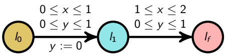
(c) Permissiveness in l_1

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

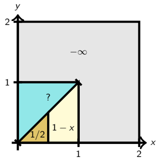
. Then $S_{h_\alpha, h_\beta} =$ 

- 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:
  $a = c = 1 \geq 0, \frac{M_\beta - b}{a+1} = \frac{1-x-1}{1+1} = x/2, m_\alpha = 0, \min\{M_\alpha, M_\beta\} = 1 - x.$

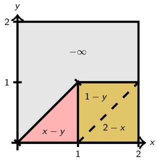
Example of this strategy



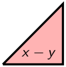
(a) A two-transitions automaton



(b) Permissiveness in l_0



(c) Permissiveness 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 $1/2$

Our contribution - Complexity of the algorithm for general cases

- 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.**

Our contribution - Complexity of the algorithm for general cases

- 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**

- 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**

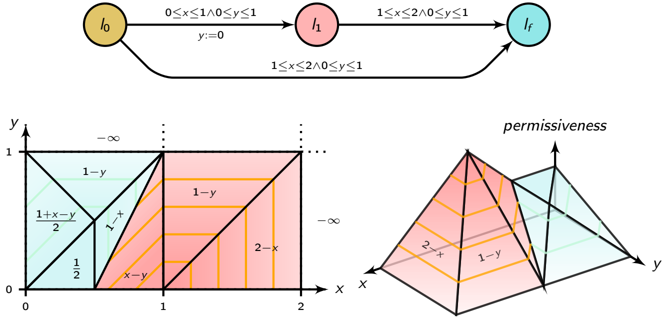
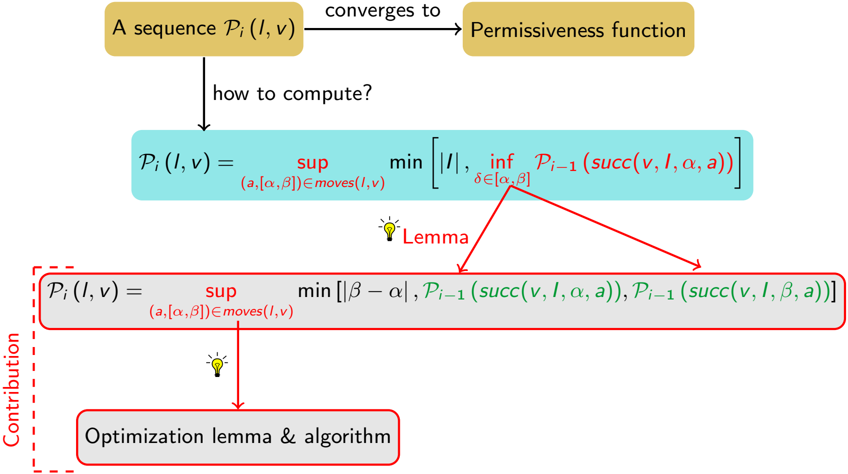
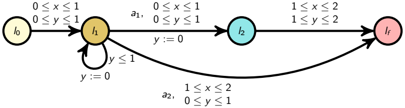


Figure: A timed automaton and its (non-concave) permissiveness function in l_0

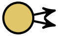
Conclusion - Our contribution



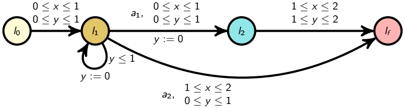


- Achieved works

Computation of the robustness:

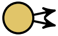
- ▷ Operator: min.
- ▷ : ✓
- ▷  ... : ✓
- ▷ : ✓
- ▷ Timed games: ✓
- ▷ Constructive algorithm and worst-case complexity: ✓

Conclusion - Achieved, ongoing and future works



- Achieved works

Computation of the robustness:

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

- Future works

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

Appendix - The technical lemma

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 $\alpha \leq \beta$:

- If $a \leq 0$ and $c \geq 0$

coordinates of maximal point	value of maximal point
(m_α, M_β)	$\min\{M_\beta - m_\alpha, am_\alpha + b, cM_\beta + d\}$

- If $a \geq 0$ 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)	$\min\{M_\beta - m_\alpha, cM_\beta + d\}$
$m_\alpha \leq \frac{M_\beta - b}{a+1} \leq \min\{M_\alpha, M_\beta\}$	$(\frac{M_\beta - b}{a+1}, M_\beta)$	$\min\{\frac{aM_\beta + b}{a+1}, cM_\beta + d\}$
$\min\{M_\alpha, M_\beta\} \leq \frac{M_\beta - b}{a+1}$	$(\min\{M_\alpha, M_\beta\}, M_\beta)$	$\min\{aM_\alpha + b, aM_\beta + b, cM_\beta + d\}$

- If $a \leq 0$ and $c \leq 0$

Symetric case of $a \geq 0$ and $c \geq 0$

- Otherwise:

Condition		coordinates of maximal point	value of maximal point
$f \leq g, h$ at $(\min\{M_\alpha, M_\beta\}, M_\beta)$		$(\min\{M_\alpha, M_\beta\}, M_\beta)$	$\min\{aM_\alpha + b, aM_\beta + b\})$
$g \leq f, h$ at $(m_\alpha, \max\{m_\alpha, m_\beta\})$		$(m_\alpha, \max\{m_\alpha, m_\beta\})$	$\min\{cm_\alpha + d, cm_\beta + d\}$
$h \leq f, g$ at (m_α, M_β)		(m_α, M_β)	$M_\beta - m_\alpha$

Otherwise, let $T_\alpha = \frac{d - b(1-c)}{(a+1)(1-c)-1}$ and $T_\beta = \frac{d(a+1) - b}{(a+1)(1-c)-1}$

Condition		coordinates of maximal point	value of maximal point
$T_\beta \geq M_\beta$		$(\frac{M_\beta - b}{a+1}, M_\beta)$	$\frac{aM_\beta + b}{a+1}$
$T_\alpha \leq m_\alpha$		$(m_\alpha, \frac{m_\alpha + d}{1-c})$	$\frac{cm_\alpha + d}{1-c}$
$ad \leq bc$	$g \leq f, h$ at $(\min\{m_\beta, M_\alpha\}, m_\beta)$	$(\frac{cm_\beta + d - b}{a}, m_\beta)$	$cm_\beta + d$
	$g \leq f, h$ at $(M_\alpha, \max\{m_\beta, M_\alpha\})$	$(\frac{d-b}{a-c}, \frac{d-b}{a-c})$	$\frac{ad - bc}{a-c}$
	otherwise	$(M_\alpha, \frac{aM_\alpha + b - d}{c})$	$aM_\alpha + b$
$ad \geq bc$	$T_\beta \leq m_\beta$	$((1-c)m_\beta - d, m_\beta)$	$cm_\beta + d$
	$T_\alpha \geq M_\alpha$	$(M_\alpha, (a+1)M_\alpha + b)$	$aM_\alpha + b$
	otherwise	(T_α, T_β)	$\frac{ad - bc}{(a+1)(1-c)-1}$