

# Robustness of Timed Automata: computing the maximal permissiveness.<sup>a</sup>

Emily Clement<sup>1, 2, 3</sup> Thierry Jéron<sup>1</sup> Nicolas Markey<sup>1</sup> David Menré<sup>2</sup>

<sup>1</sup>IRISA, Inria & CNRS & Univ. Rennes, France

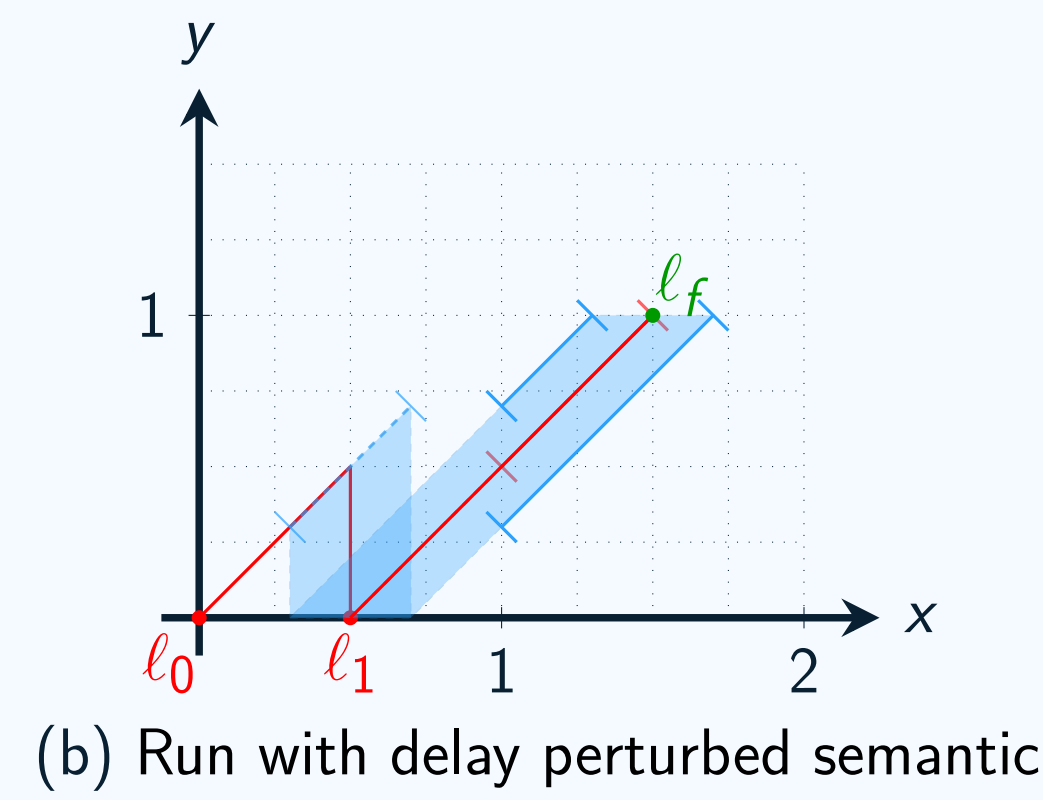
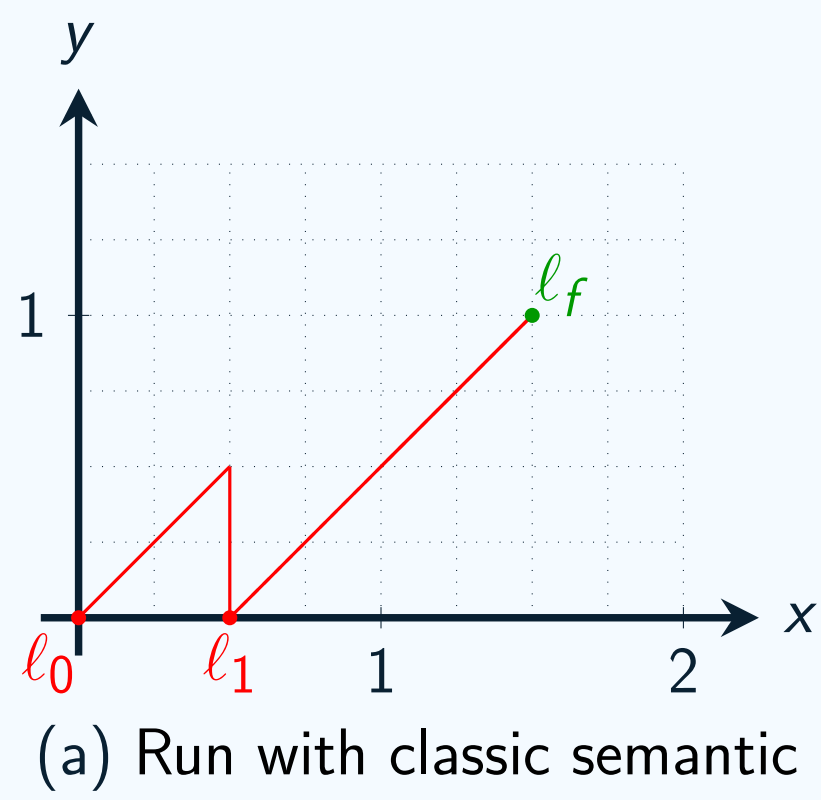
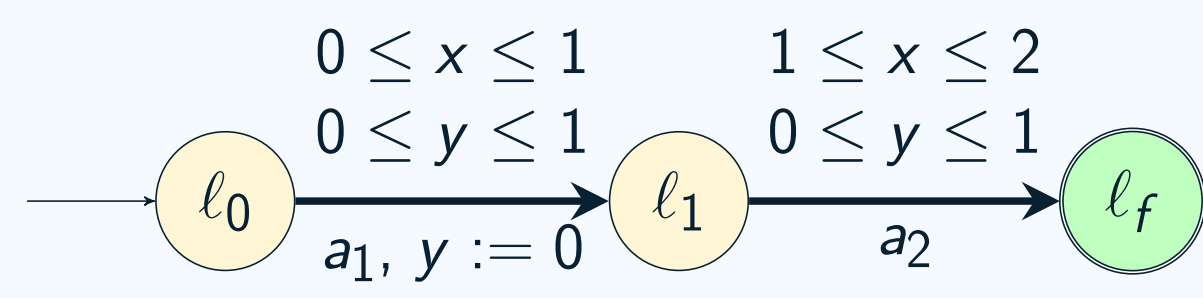
<sup>2</sup>Mitsubishi Electric R&D Centre Europe – Rennes, France: MERCE

<sup>3</sup>Sorbonne Université, CNRS, Institut des Systèmes Intelligents et de Robotique, ISIR, F-75005 Paris, France

<sup>a</sup>This work was partially funded by ANR project TickTac (ANR-18-CE40-0015).

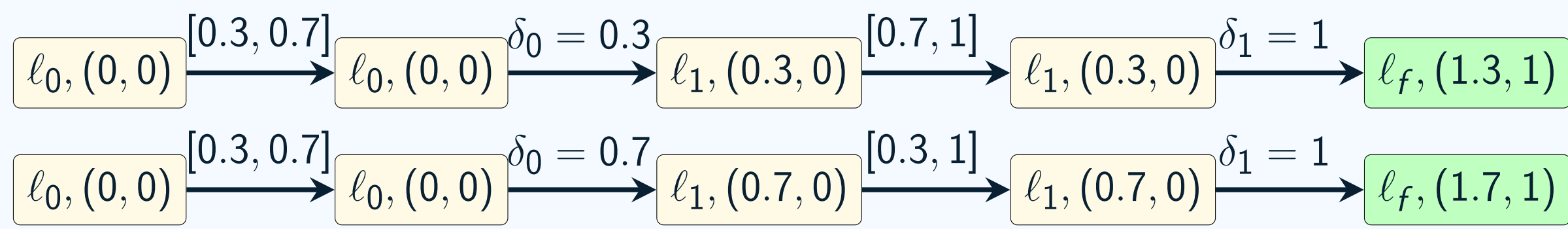
## Delay perturbation problem: Permissiveness

▪ **Context:** Given a Timed Automaton, how much can we perturb the applied delays while ensuring reachability?



▪ **Example of perturbed runs:**

▷ Classical move:  $(0.3, a_1)$ . (Permissive) p-move:  $([0.3, 0.7], a_1)$   
 ▷ Intervals of proposed delays are respectively at least 0.3 (resp. 0.4) permissive:

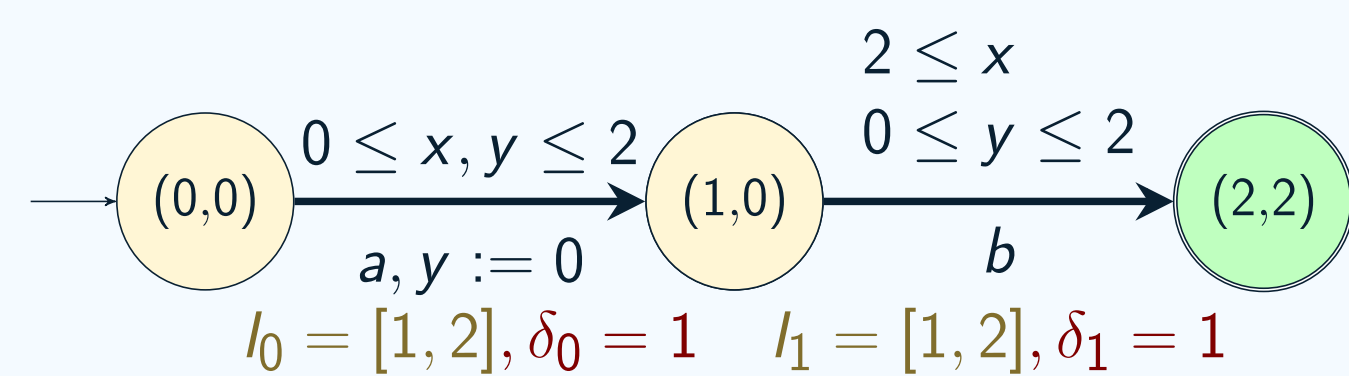


## Modelisation as turn-based game

▪ **The game:**

- ▷ **Player** maximises the permissiveness of runs. He chooses a p-move, i.e. **Interval** & **action**.
- ▷ **Opponent** minimises the permissiveness of runs. He chooses a **delay**  $\in$  **Interval**.
- ▷ **Permissiveness of the run** is the **minimum** of the size of the intervals of all the proposed p-moves.

▪ **Example of run:**



▷ **Permissiveness of the run** =  $\min(|l_0|, |l_1|) = \min(1, 1) = 1$ .

## Maximal Permissiveness problem

▪ **Maximal permissiveness:**

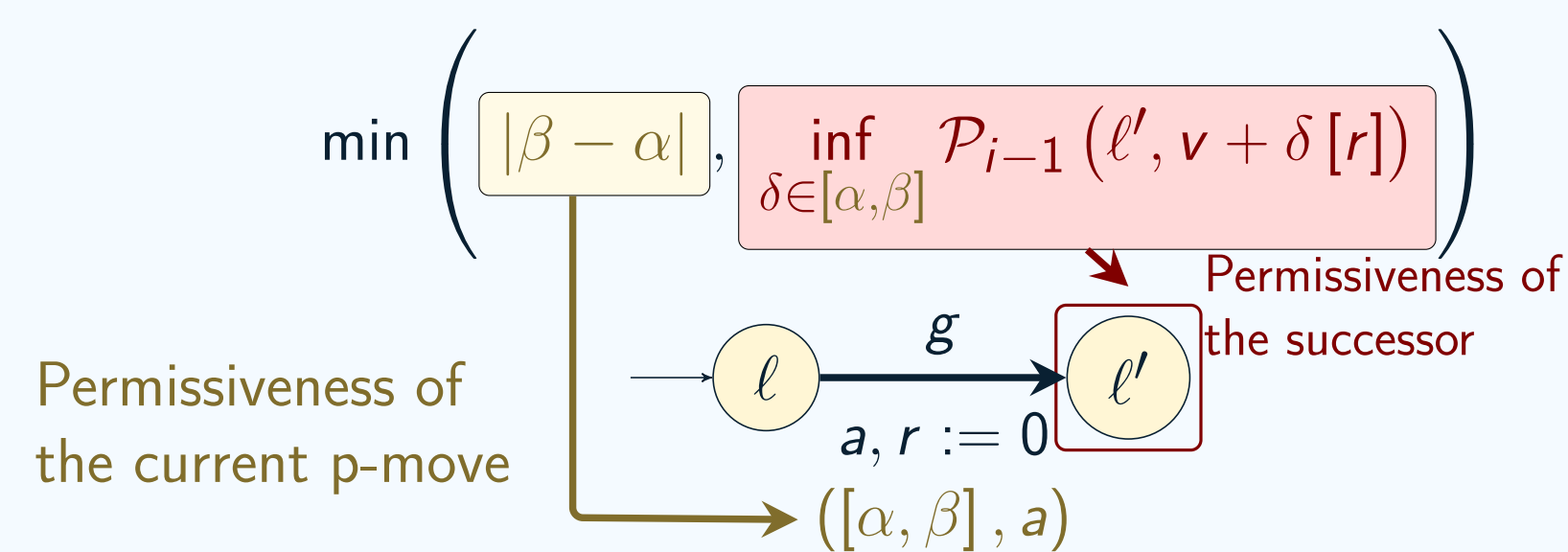
▷  $Perm(\ell, v)$ : permissiveness of a run from  $(\ell, v)$ , considering the **optimal strategy** of both **Player** and **Opponent**.

▪ **Sequence of Suboptimal Permissiveness Functions:**

$$P_i(\ell, v) = \sup_{([\alpha, \beta], a) \in \text{p-moves}(\ell, v)} \left( \min \left( \beta - \alpha, \inf_{\delta \in [\alpha, \beta]} P_{i-1}(\ell', v + \delta[r]) \right) \right)$$

▪ **Strategy of the player:**

▷ Find the enabled p-move  $([\alpha, \beta], a)$  that **maximises**:



## Goals

▪ **Compute the maximal permissiveness:**

▷ **Maximal:** Compute  $Perm(\ell, \cdot) : v \mapsto \lim_{i \rightarrow +\infty} P_i(\ell, v)$  if it exists. [2]

▪ **Compute the binary and levelled permissiveness [1]:**

▷ **Binary:** Given a **threshold**  $p \geq 0$ , compute the set of valuations  $v$  s.t.  $Perm(\ell, v) \geq p$ .

▷ **Levelled:** Let  $[p_0, p_1], [p_1, p_2], \dots, [p_m, p_{m+1}]$ , decide to which interval  $Perm(\ell, v)$  belongs.

## References

- [1] Emily Clement. *Robustness of timed automata : computing the maximally-permissive strategies*. PhD thesis, University of Rennes 1, France, 2022.
- [2] Emily Clement, Thierry Jéron, Nicolas Markey, and David Menré. *Computing maximally-permissive strategies in acyclic timed automata*. In *FORMATS 2020*, volume 12288, pages 111–126. Springer, 2020.



Gitlab



TickTac ANR

## Contribution: Algorithms that compute ...

▪ **Maximal Permissiveness:**

Automata	Complexity	Implemented
Linear	$\mathcal{O}(n^{3^d})$	✓
Acyclic	Non-elementary	Current work

$n$ =number of clocks,  $d$ =number of transitions,  $m$ =number of intervals

▪ **Binary & Levelled Permissiveness:**

Automata	Complexity	Implemented
Linear	$\mathcal{O}((m+1)(8 \cdot n)^{2^d})$	✓
Acyclic	Current work	Current work

## Maximal permissiveness computation

▪ **Linear case:**

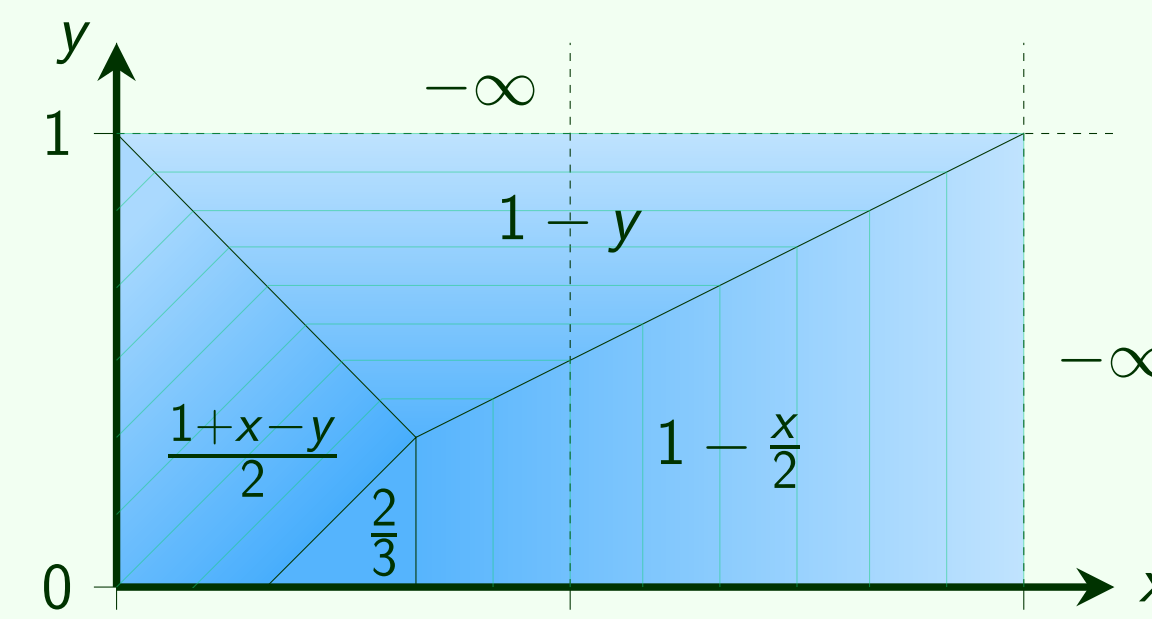
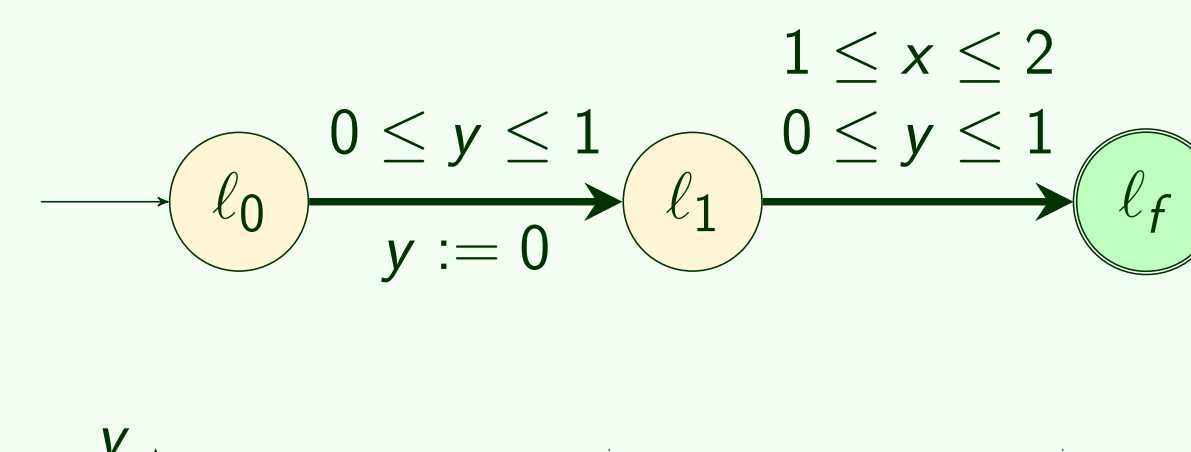


Figure 3. Perm of the linear TA

▪ **Acyclic case:**

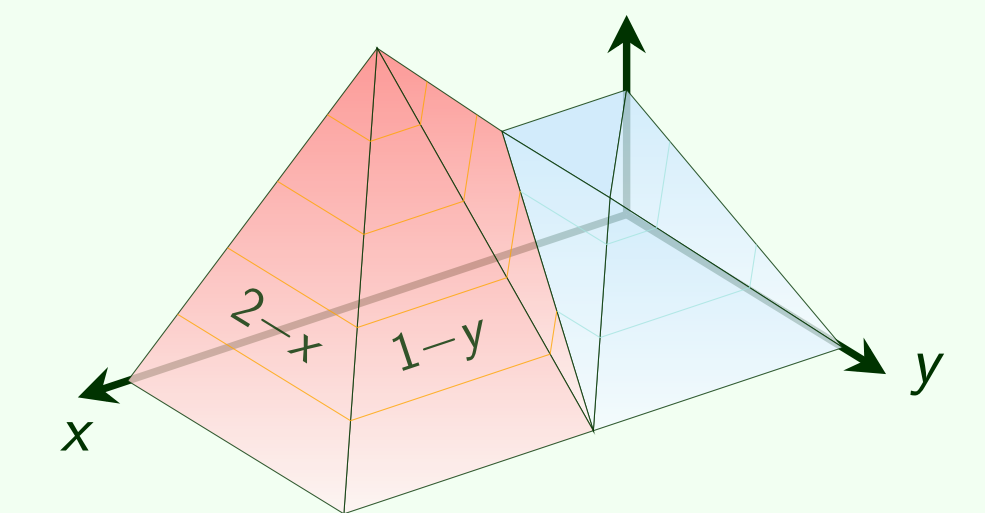
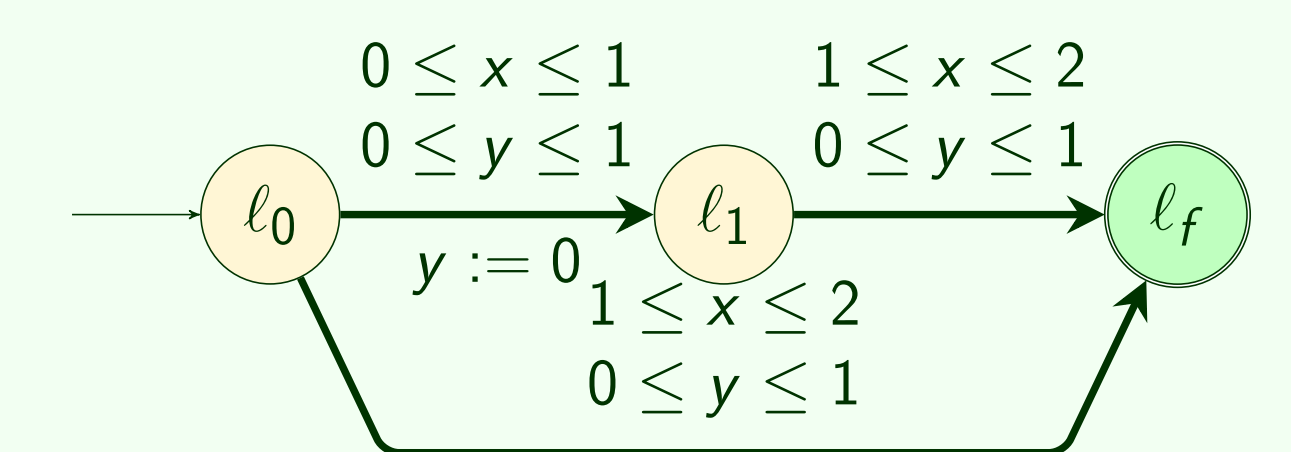


Figure 4. Perm of the acyclic TA

▪ **Results:**

- ▷ **Optimal strategy of the opponent in linear case:** choosing the bounds of the proposed interval.
- ▷ An algorithm to compute the optimal strategy of player in linear and acyclic case.
- ▷ A Python, Parma Polyhedra Library-based, implementation.
- ▷ An optimisation lemma to tackle the overfilling of polyhedra:

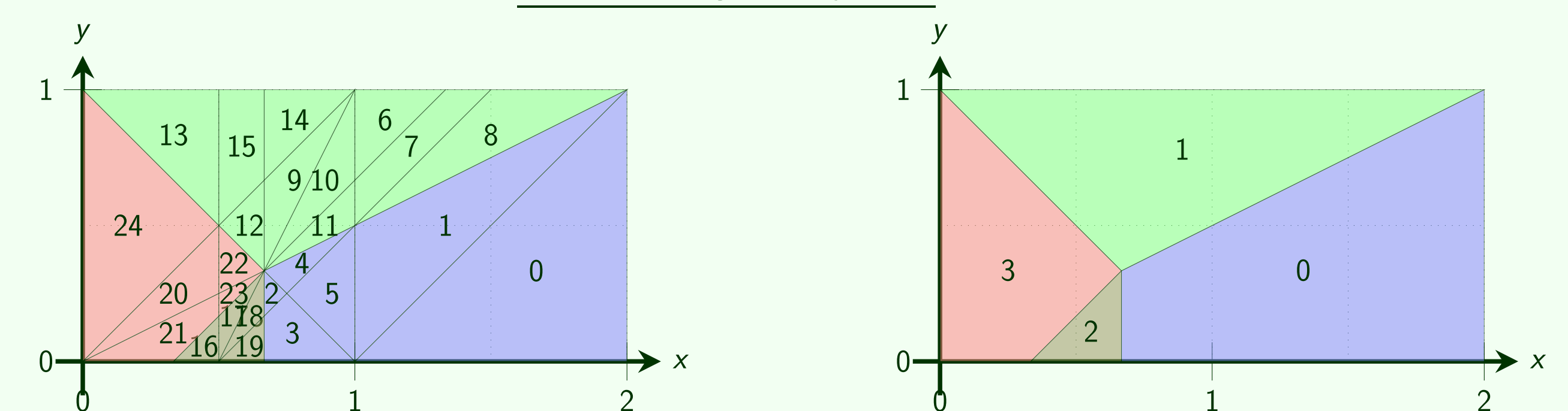


Figure 5. Before (left) and after (right) optimisation results of our tool.

For other three-clocks examples, we divided the number of cells by **60**.

▪ **What about cycles?**

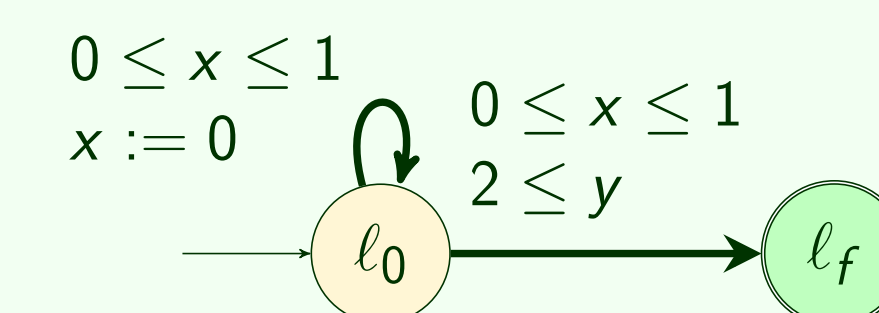


Figure 6. In this example, the permissiveness can (strictly) increase with the number of visits.

▪ **Future works:**

- ▷ Implement acyclic case.
- ▷ Find a subclasses of Timed automata with cycles where  $(P_i)_i$  becomes stationary.

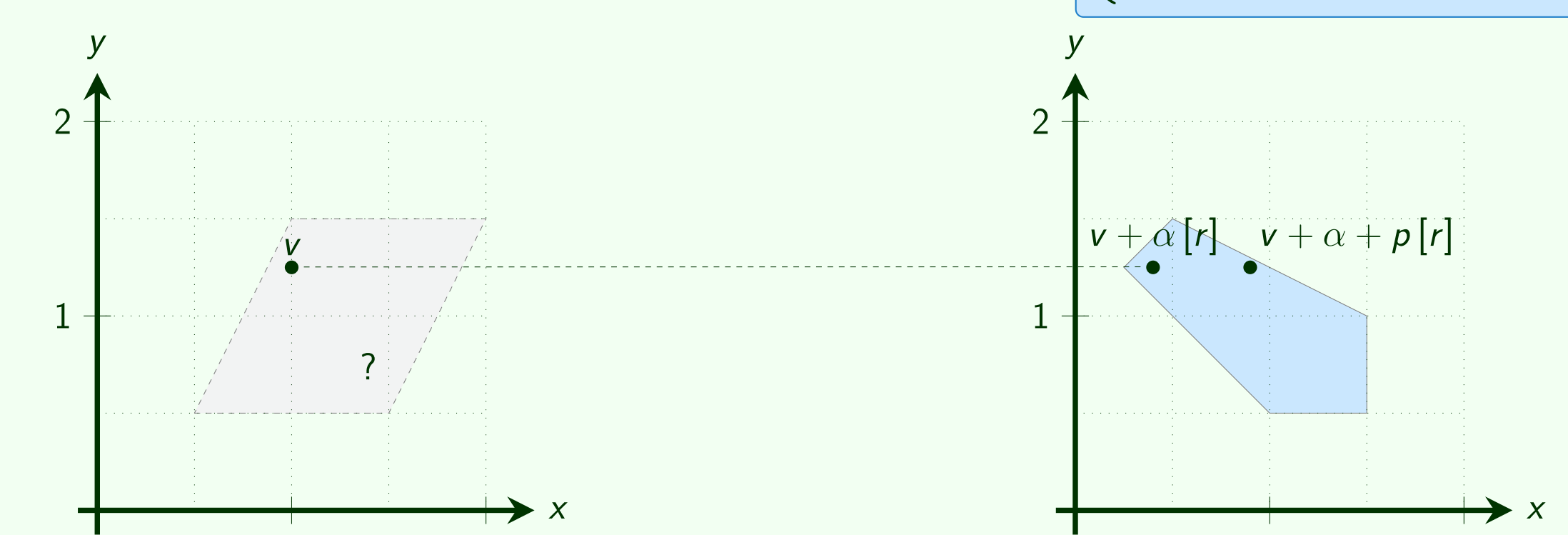
## Levelled and binary permissiveness computation

▪ **Goal - Compute the set of valuations  $S_i(\ell, p)$  such that  $P_i(\ell, \cdot)$  is greater than  $p$ :**

$$S_i(\ell, p) := \{v \in \mathbb{R}_+^n \mid P_i(\ell, v) \geq p\}$$

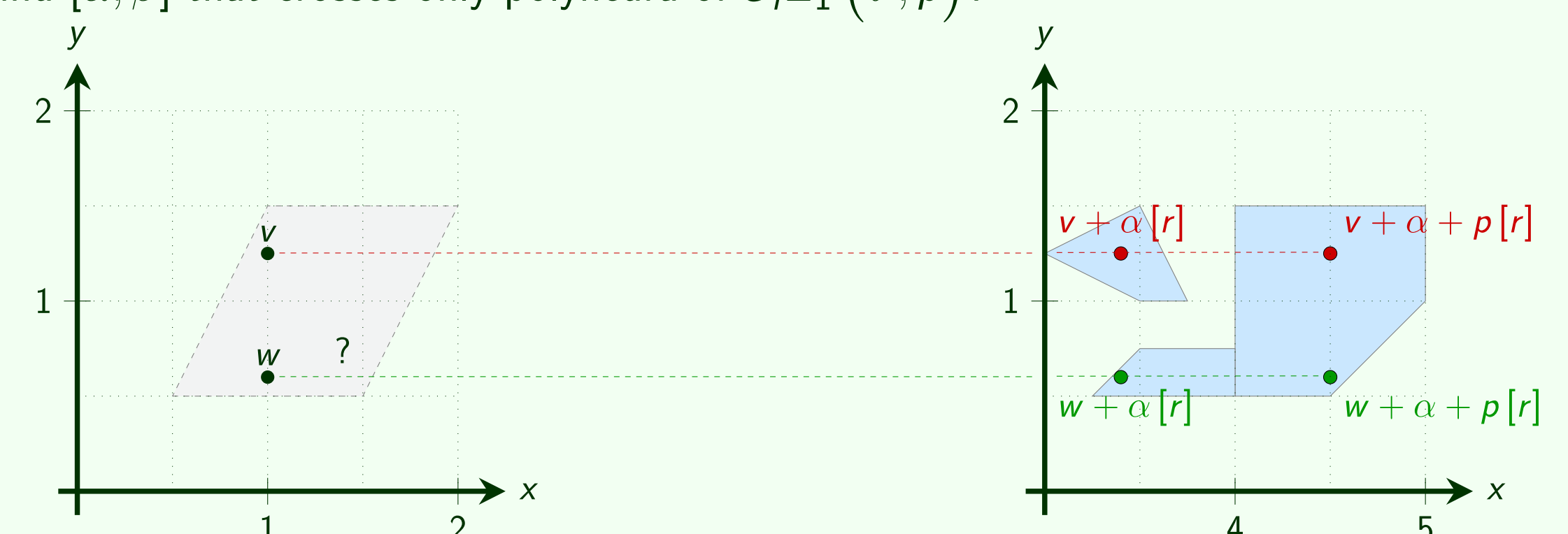
▪ **Results for Linear case:**

- ▷  $S_i(\ell, p)$  is a unique polyhedron.
- ▷ An algorithm to compute binary & levelled permissiveness. **Steps:**
  - Compute  $S_{i-1}(\ell', p)$
  - Fourier-Motzkin: compute the set of  $v$  s.t. there exists  $\alpha \geq 0$  s.t.  $\begin{cases} v + \alpha, v + \alpha + p \models g \\ v + \alpha[r], v + \alpha + p[r] \in S_{i-1}(\ell', p) \end{cases}$



▪ **Acyclic case:  $S_i(\ell, p)$  is no longer a unique polyhedron in general.**

▷ Can we find  $[\alpha, \beta]$  that crosses only polyhedra of  $S_{i-1}(\ell', p)$ ?



▪ **Future works:**

- ▷ Construct an adapted representation of polyhedra to merge them to solve the acyclic case.
- ▷ Implement an algorithm for acyclic case.

## Organisations

