Uniform Sampling for Timed Automata with Application to Language Inclusion Measurement

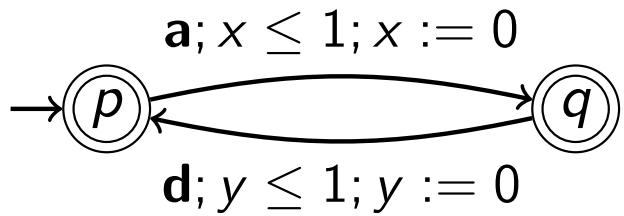
Nicolas Basset (Université libre de Bruxelles)

joint work with
Benoît Barbot, Marc Beunardeau and Marta Kwiatkowska

AVERTS, 14 July 2017

The context of timed automata theory

Introduced by Alur and Dill in 1990 to model and verify **real-time** properties of **embedded systems** (cars, phones, pacemakers...). Build by adding real time *clocks* to finite state automaton.



Since then many people worked on

- ▷ adding real-time to classical automata and verification theory.
- ▷ implementing model checking tools (e.g. UPPAAL).

Model checking timed regular properties.

- ▷ Model encoded as a timed automaton A.
- ▷ Specification encoded as a timed automaton B.
- ▷ Problem : is $L(A) \subseteq L(B)$?

Model checking timed regular properties. Undecidable!

- ▷ Model encoded as a timed automaton A.
- ▷ Specification encoded as a timed automaton B.
- ▷ Problem : is $L(A) \subseteq L(B)$?

Model checking timed regular properties. Undecidable!

- ▷ Model encoded as a timed automaton A.
- ▷ Specification encoded as a timed automaton B.
- ▷ Problem : is $L(A) \subseteq L(B)$?

Our statistical model checking method

- ▷ Estimate statistically the proportion of runs of $L(A)$ that are in $L(B)$.
- ▷ If the estimation is 1 then answer **Yes** with high confidence;
- ▷ else answer **No** and exhibit a counter-example.

Model checking timed regular properties. Undecidable!

- Model encoded as a timed automaton A.
- Specification encoded as a timed automaton B.
- Problem : is $L(A) \subseteq L(B)$?

Our statistical model checking method

- Estimate statistically the proportion of runs of $L(A)$ that are in $L(B)$.
- If the estimation is 1 then answer **Yes** with high confidence;
- else answer **No** and exhibit a counter-example.

Our method is based on

- volumetry of timed languages described in [Asarin, B., Degorre, Information & Computation 2015];
- uniform sampling of timed words (wrt. volumes);

Simulation of a hybrid system $\mathcal{H}$ whose inputs are given by a TA $\mathcal{A}$

- Draw inputs $w \in \mathcal{A}$ at random and check properties on the corresponding outputs.

Example

- $\mathcal{H}$ models a boiler with two modes Heating and Non-Heating
- $\mathcal{A}$ models the following input behaviours:

- the boiler is no more than 20 minutes in Heating mode before being switch to Non-Heating mode
- it is switched to Heating mode at least twice per hour.

- Possible question one may ask:
  - will the boiler have a problem?
  - will a problem appears in less than 0.01% of possible inputs?
  - what is the average energy consumption of the boiler?

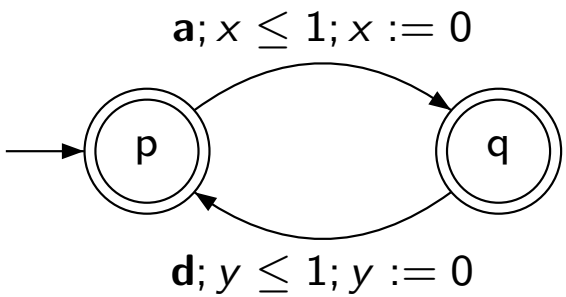
Outline

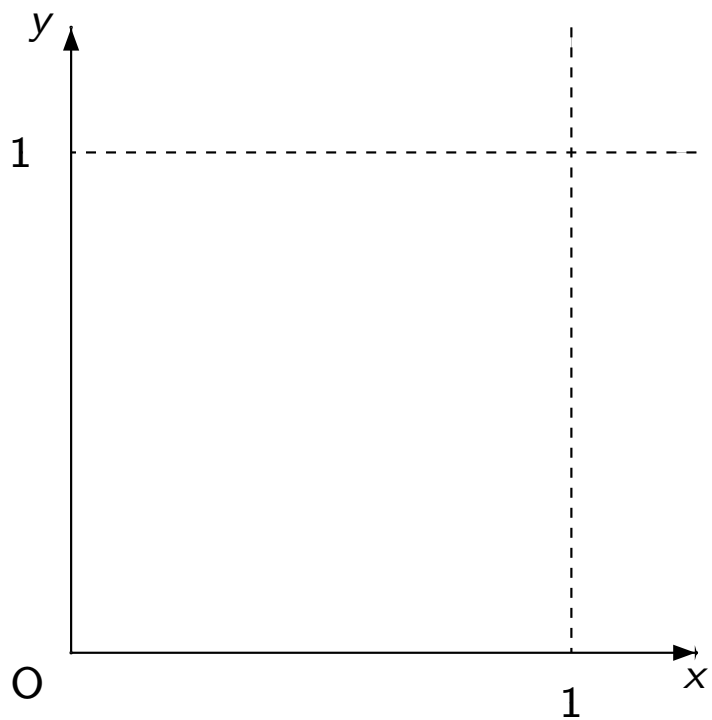
Outline

Timed automata [Alur and Dill 1990]

Timed automata = finite automata + constraints on timings on edges using clocks.

Recognise timed words e.g. (0.4,**a**)(0.3,**d**)(0.6,**a**).

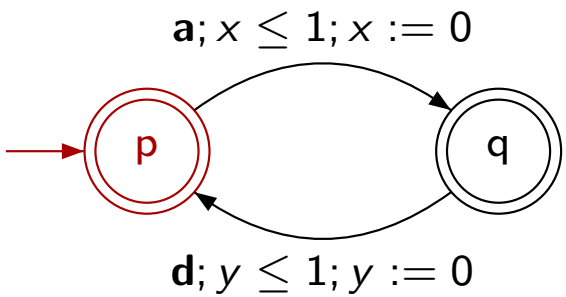


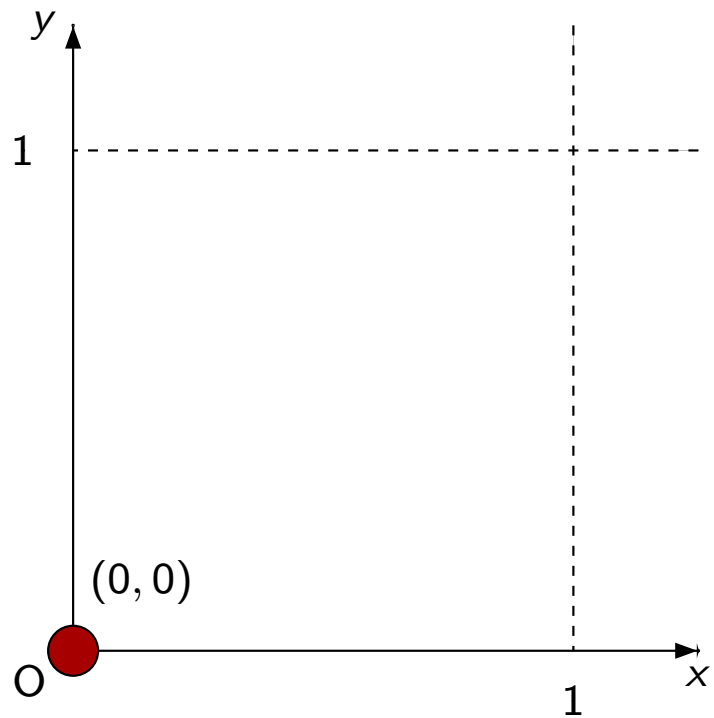


Timed automata [Alur and Dill 1990]

Timed automata = finite automata + constraints on timings on edges using clocks.

Recognise timed words e.g. (0.4,**a**)(0.3,**d**)(0.6,**a**).

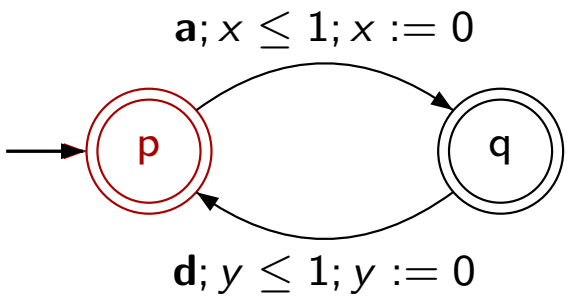


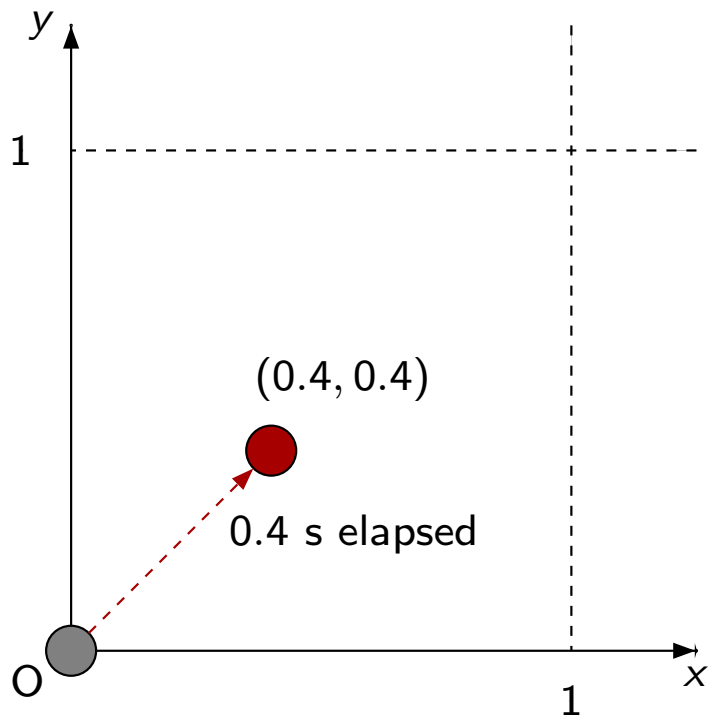


Timed automata [Alur and Dill 1990]

Timed automata = finite automata + constraints on timings on edges using clocks.

Recognise timed words e.g. (0.4,**a**)(0.3,**d**)(0.6,**a**).

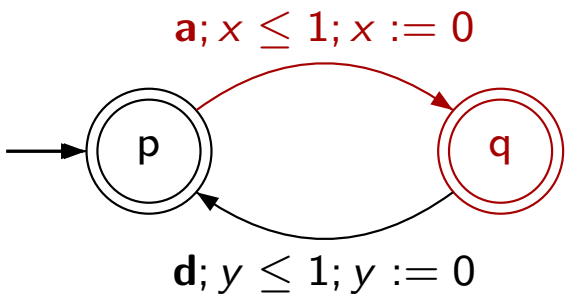


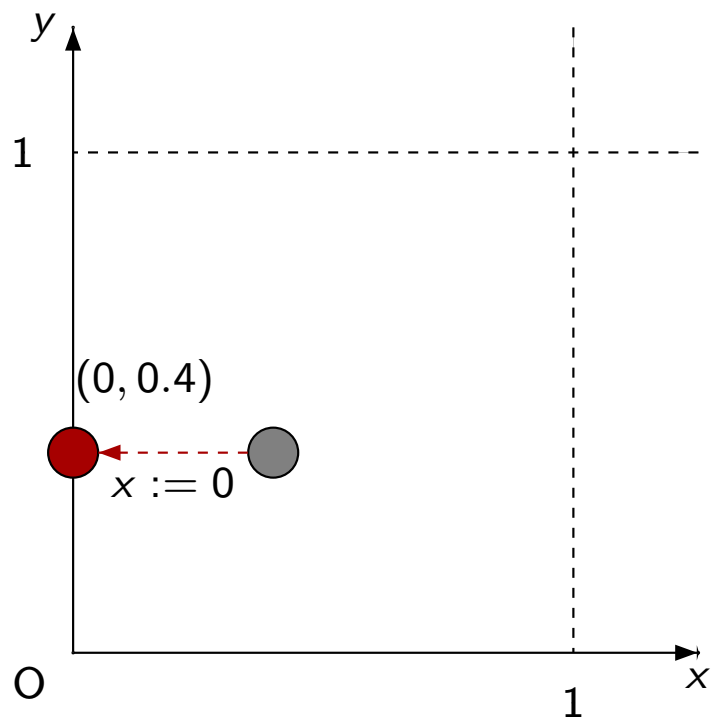


Timed automata [Alur and Dill 1990]

Timed automata = finite automata + constraints on timings on edges using clocks.

Recognise timed words e.g. (0.4,**a**)(0.3,**d**)(0.6,**a**).

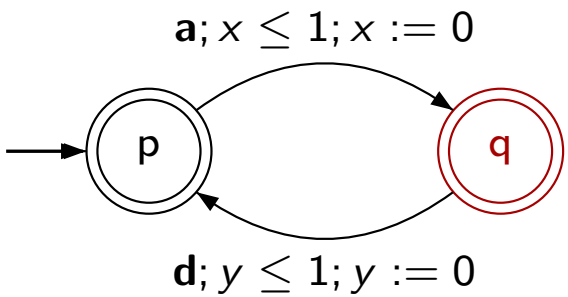


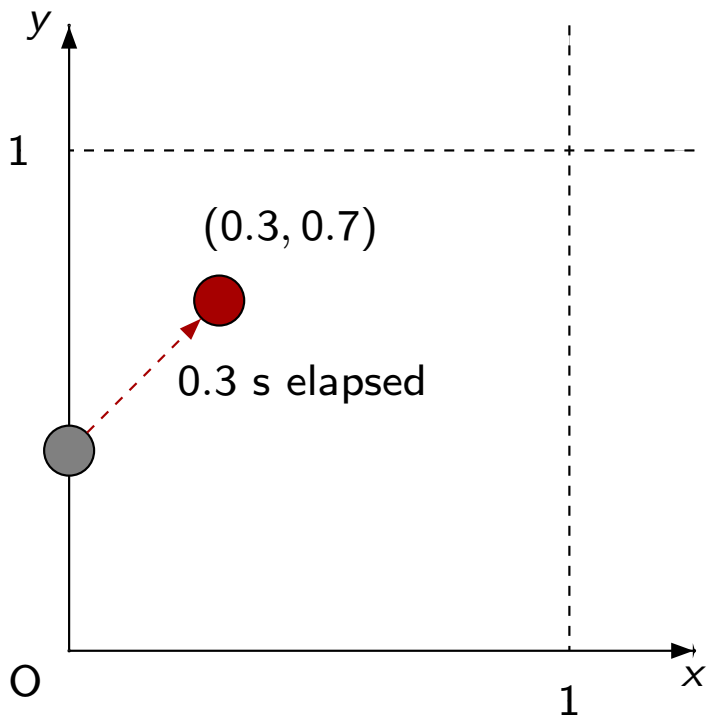


Timed automata [Alur and Dill 1990]

Timed automata = finite automata + constraints on timings on edges using clocks.

Recognise timed words e.g. (0.4,**a**)(0.3,**d**)(0.6,**a**).

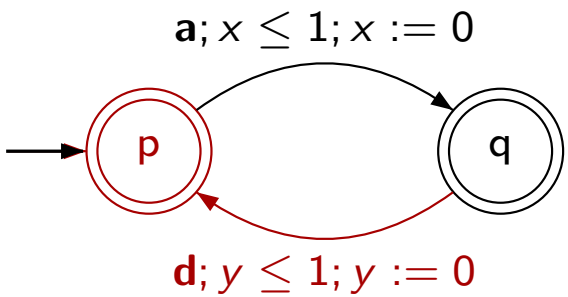


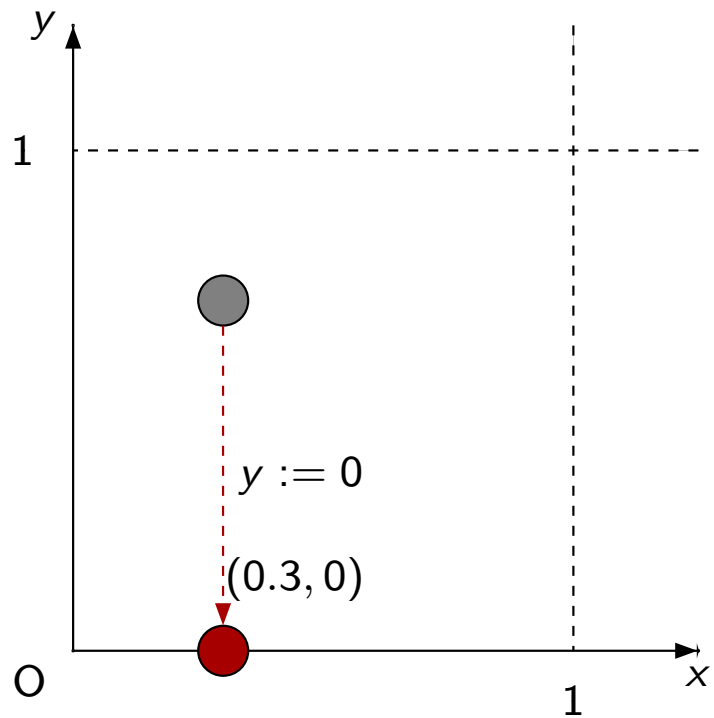


Timed automata [Alur and Dill 1990]

Timed automata = finite automata + constraints on timings on edges using clocks.

Recognise timed words e.g. (0.4,**a**)(0.3,**d**)(0.6,**a**).

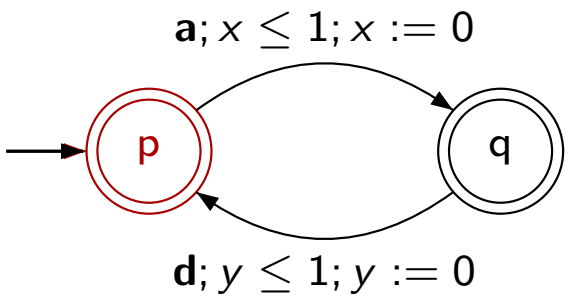


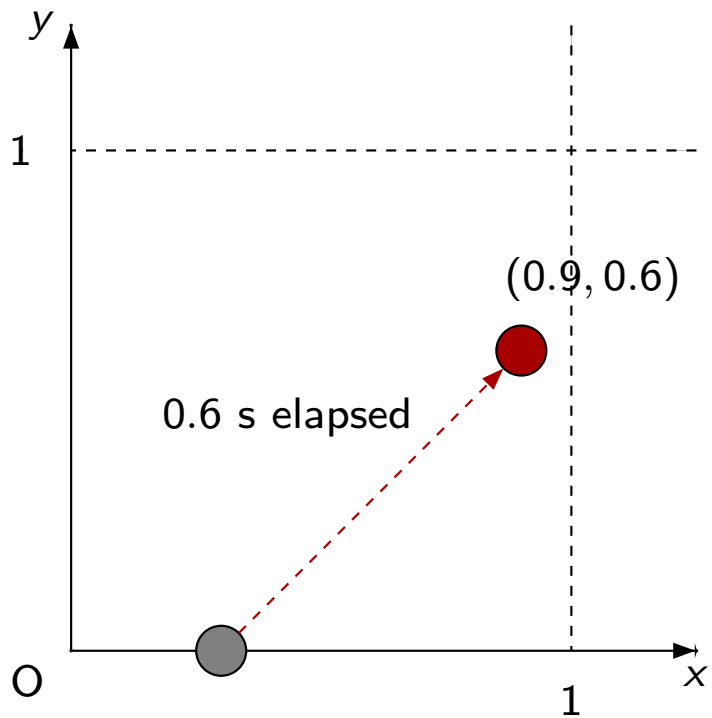


Timed automata [Alur and Dill 1990]

Timed automata = finite automata + constraints on timings on edges using clocks.

Recognise timed words e.g. (0.4,**a**)(0.3,**d**)(0.6,**a**).

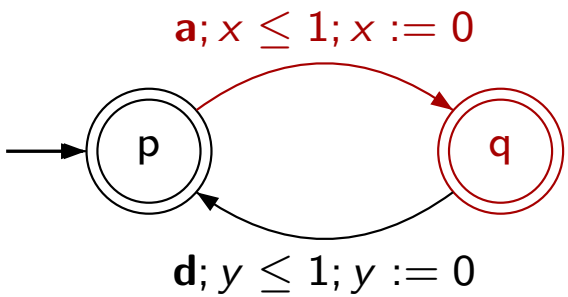


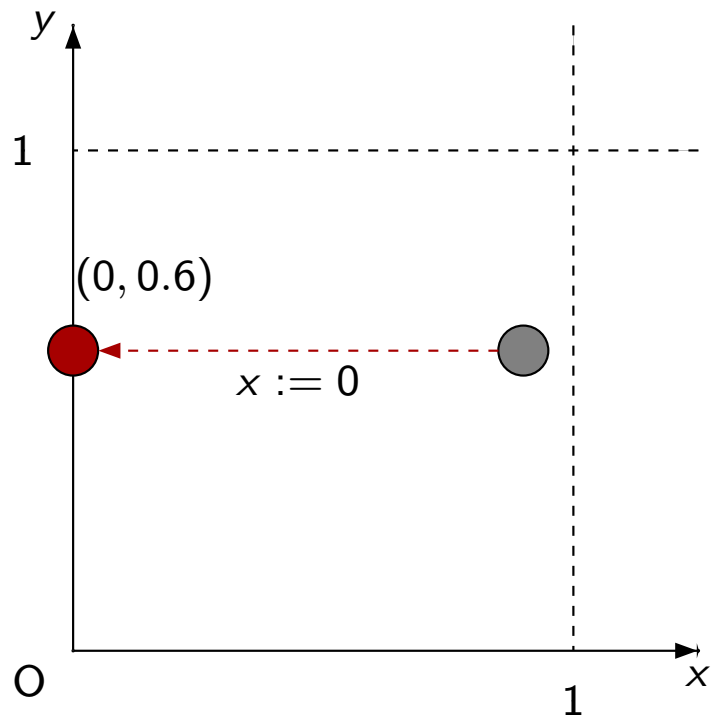


Timed automata [Alur and Dill 1990]

Timed automata = finite automata + constraints on timings on edges using clocks.

Recognise timed words e.g. (0.4,**a**)(0.3,**d**)(0.6,**a**).

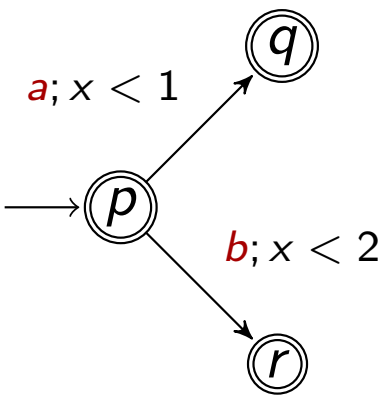




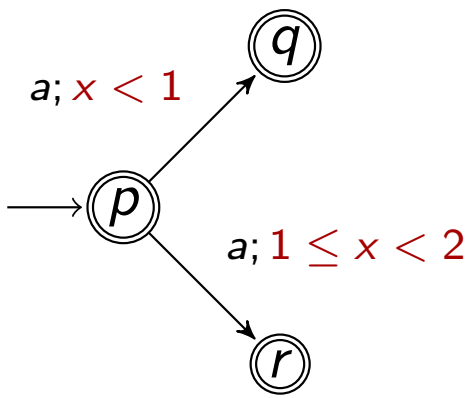
Some (restrictive) hypotheses we make

Restrictive hypotheses

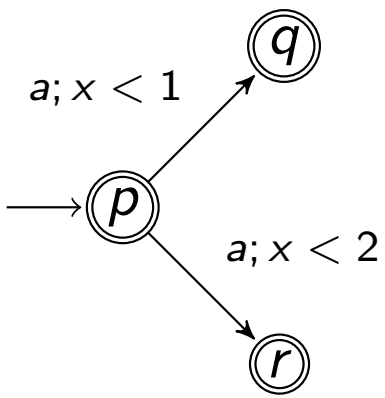
- The clocks and the delays are bounded by a constant M.
- The timed automaton is deterministic (DTA)
  (we address non-deterministic timed automata (NTA) later).



Deterministic



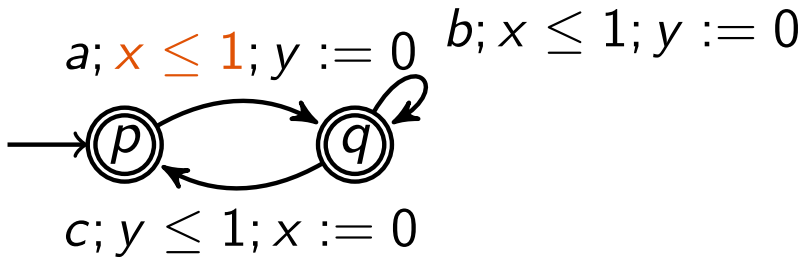
Deterministic

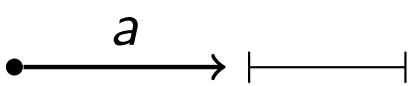


Non-deterministic

Measuring timed languages wrt. volumes

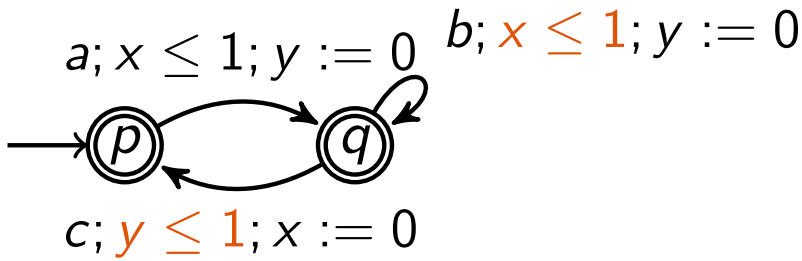
Constraints of timings along a path w = polytope P_w^L

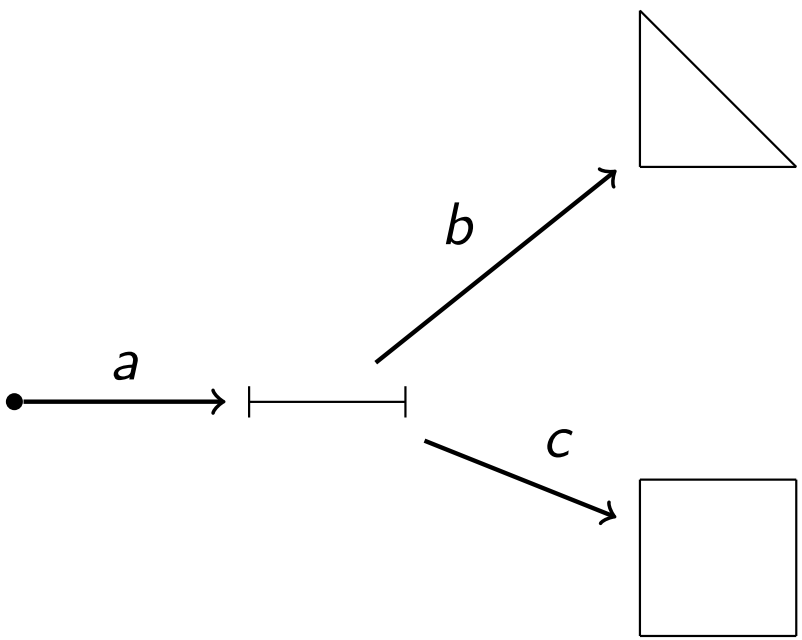




Measuring timed languages wrt. volumes

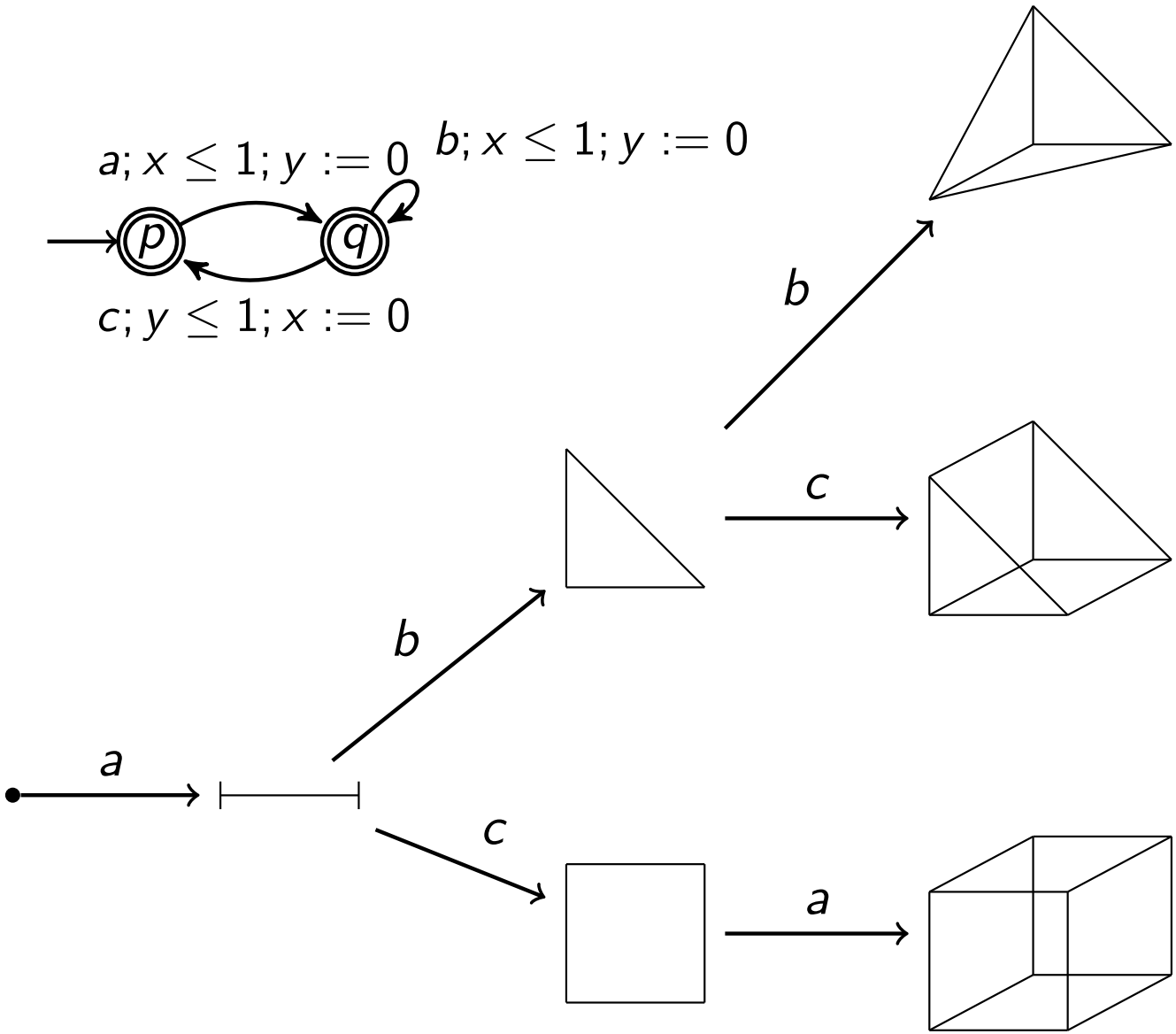
Constraints of timings along a path w = polytope P_w^L





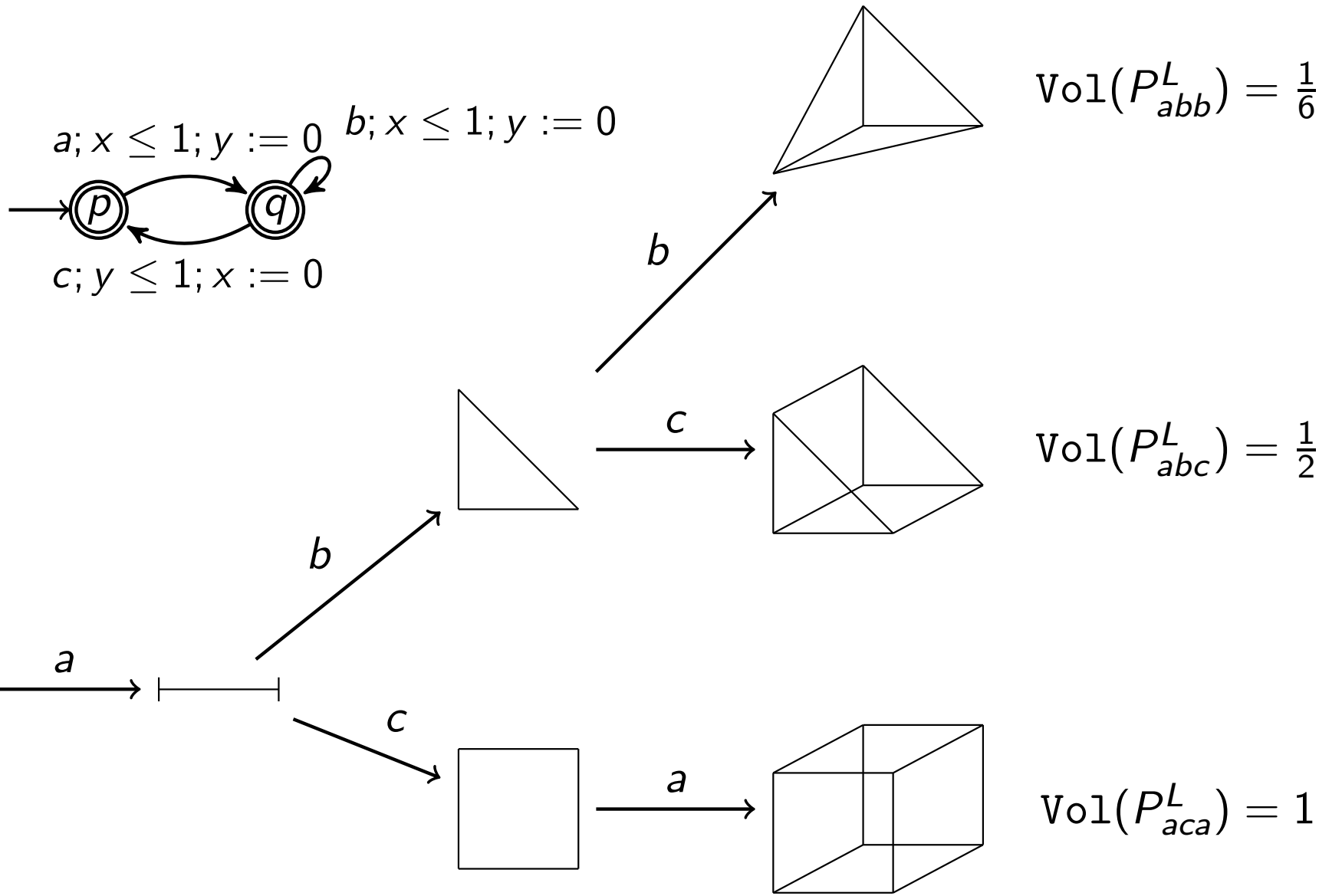
Measuring timed languages wrt. volumes

Constraints of timings along a path w = polytope P_w^L



Measuring timed languages wrt. volumes

Constraints of timings along a path w = polytope P_w^L



Computing the volumes

A recipe

- ▷ Define for $n \geq 0$ and reachable state $(q, \vec{x})$ the volume $v_n(q, \vec{x})$ of timed words of length n starting from $(q, \vec{x})$;
- ▷ write recursive equations on volume functions ;
- ▷ compute $v_n(q_0, \vec{0})$ the volume of the language from the initial state.

Problem : recursive equations are difficult to write and use a priori

Computing the volumes

A recipe

- Define for $n \geq 0$ and reachable state $(q, \vec{x})$ the volume $v_n(q, \vec{x})$ of timed words of length n starting from $(q, \vec{x})$;
- write recursive equations on volume functions ;
- compute $v_n(q_0, \vec{0})$ the volume of the language from the initial state.

Problem : recursive equations are difficult to write and use a priori

Previous solution : decompose the state space into regions. [Asarin, B., Degorre, Information & Computation 2015]

- the equations are easy to write ;
- the volume functions have nice form (polynomial functions) ;
- v_n can be computed in polynomial time wrt. n.

Computing the volumes

A recipe

- Define for $n \geq 0$ and reachable state $(q, \vec{x})$ the volume $v_n(q, \vec{x})$ of timed words of length n starting from $(q, \vec{x})$;
- write recursive equations on volume functions ;
- compute $v_n(q_0, \vec{0})$ the volume of the language from the initial state.

Problem : recursive equations are difficult to write and use a priori

Previous solution : decompose the state space into regions. [Asarin, B., Degorre, Information & Computation 2015]

- the equations are easy to write ;
- the volume functions have nice form (polynomial functions) ;
- v_n can be computed in polynomial time wrt. n.

Problem : too many regions!!!

Computing the volumes

A recipe

- Define for $n \geq 0$ and reachable state $(q, \vec{x})$ the volume $v_n(q, \vec{x})$ of timed words of length n starting from $(q, \vec{x})$;
- write recursive equations on volume functions ;
- compute $v_n(q_0, \vec{0})$ the volume of the language from the initial state.

Problem : recursive equations are difficult to write and use a priori

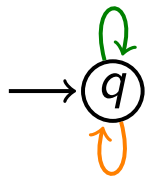
Previous solution : decompose the state space into regions. [Asarin, B., Degorre, Information & Computation 2015]

- the equations are easy to write ;
- the volume functions have nice form (polynomial functions) ;
- v_n can be computed in polynomial time wrt. n.

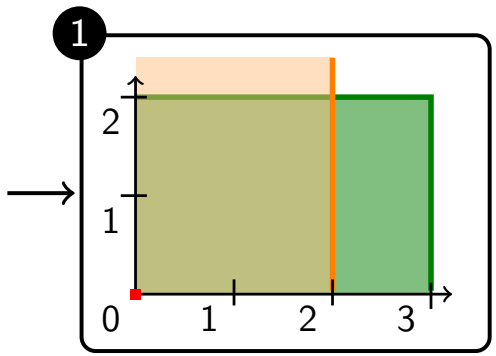
Problem : too many regions!!!
Our solution : use zones instead of regions.

A first decomposition into zones (the forward reachability timed automaton).

b,
$0 < x < 3$,
$0 < y < 2$,
$x := 0$

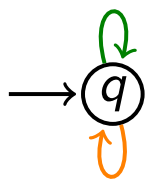


a,
$0 < x < 2$,
$0 < y < 4$,
$y := 0$

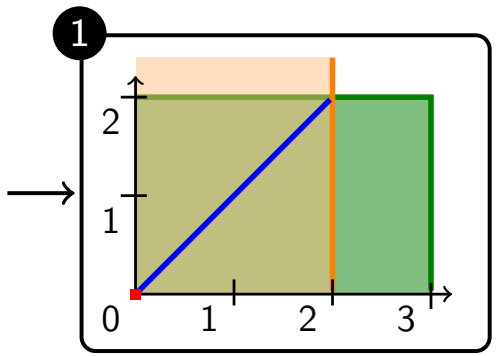


A first decomposition into zones (the forward reachability timed automaton).

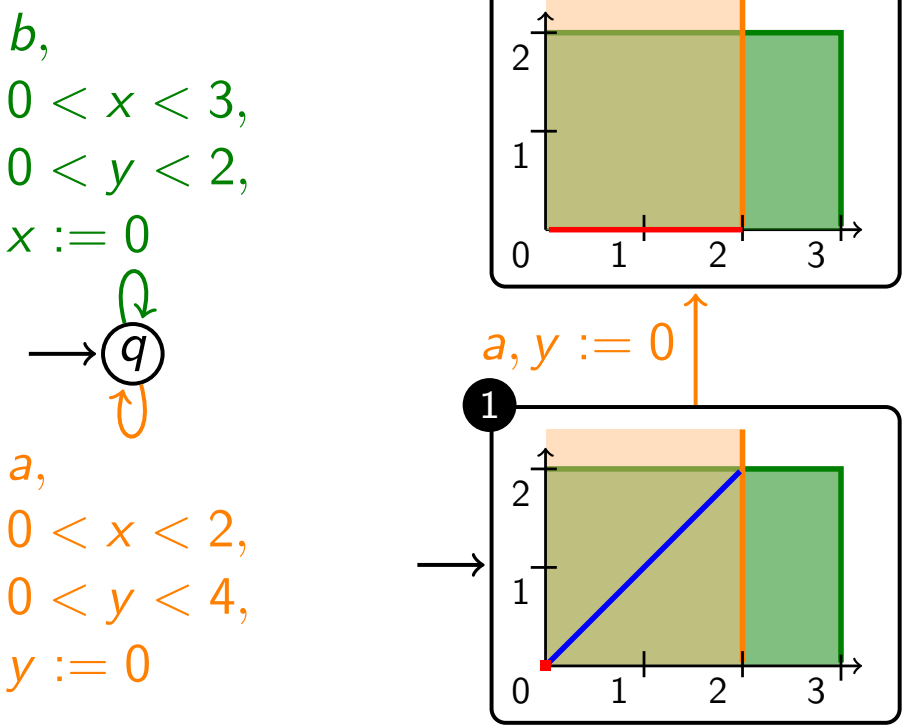
b,
$0 < x < 3$,
$0 < y < 2$,
$x := 0$



a,
$0 < x < 2$,
$0 < y < 4$,
$y := 0$

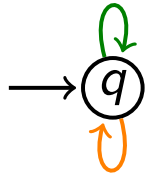


A first decomposition into zones (the forward reachability timed automaton).

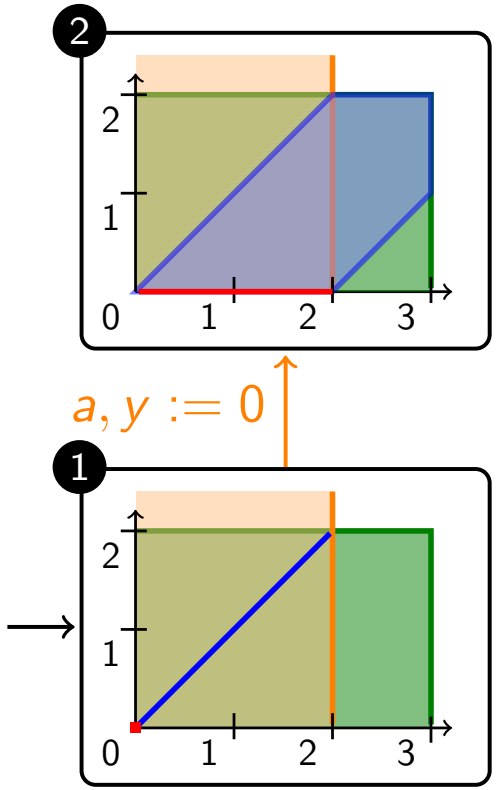


A first decomposition into zones (the forward reachability timed automaton).

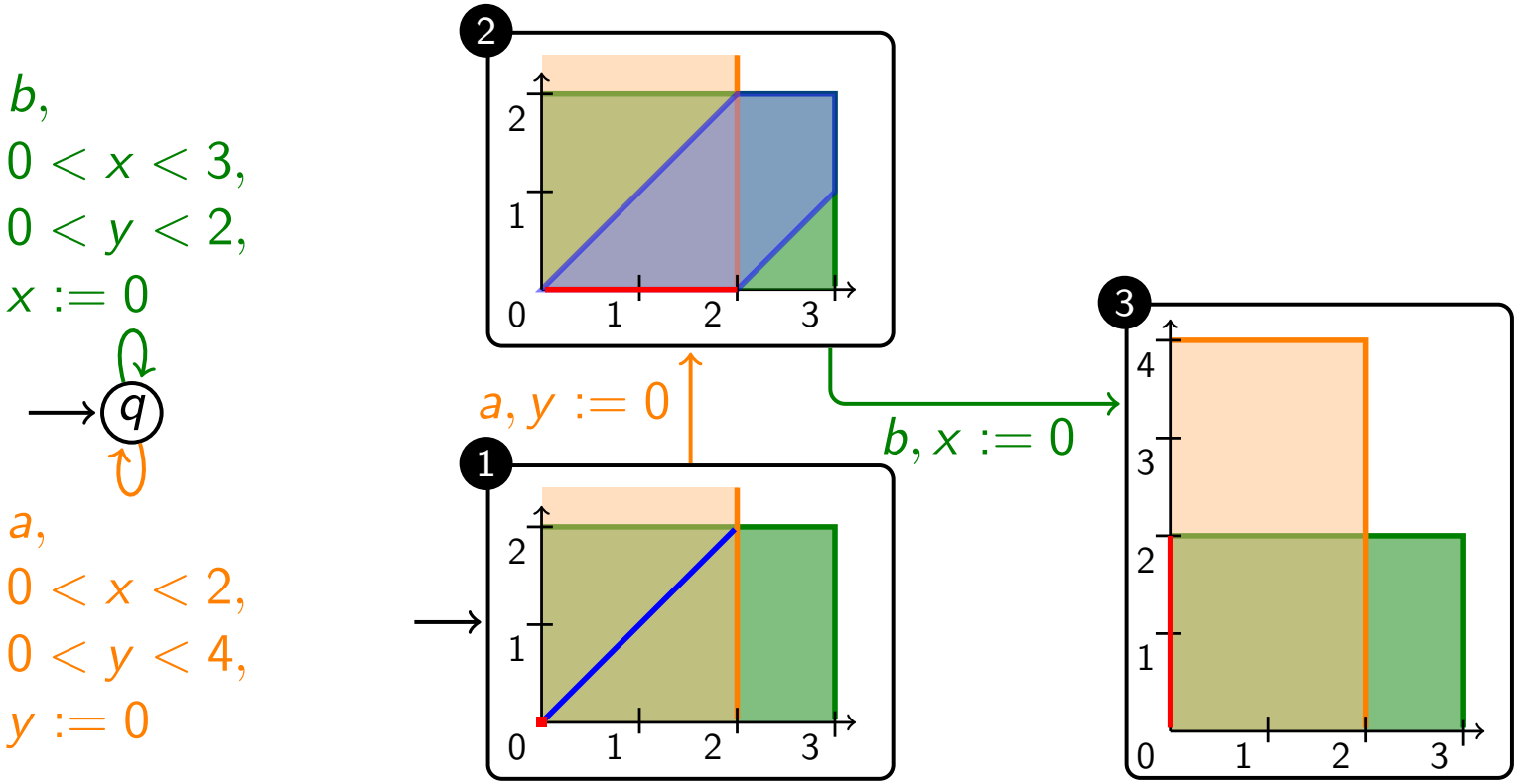
b,
$0 < x < 3$,
$0 < y < 2$,
$x := 0$



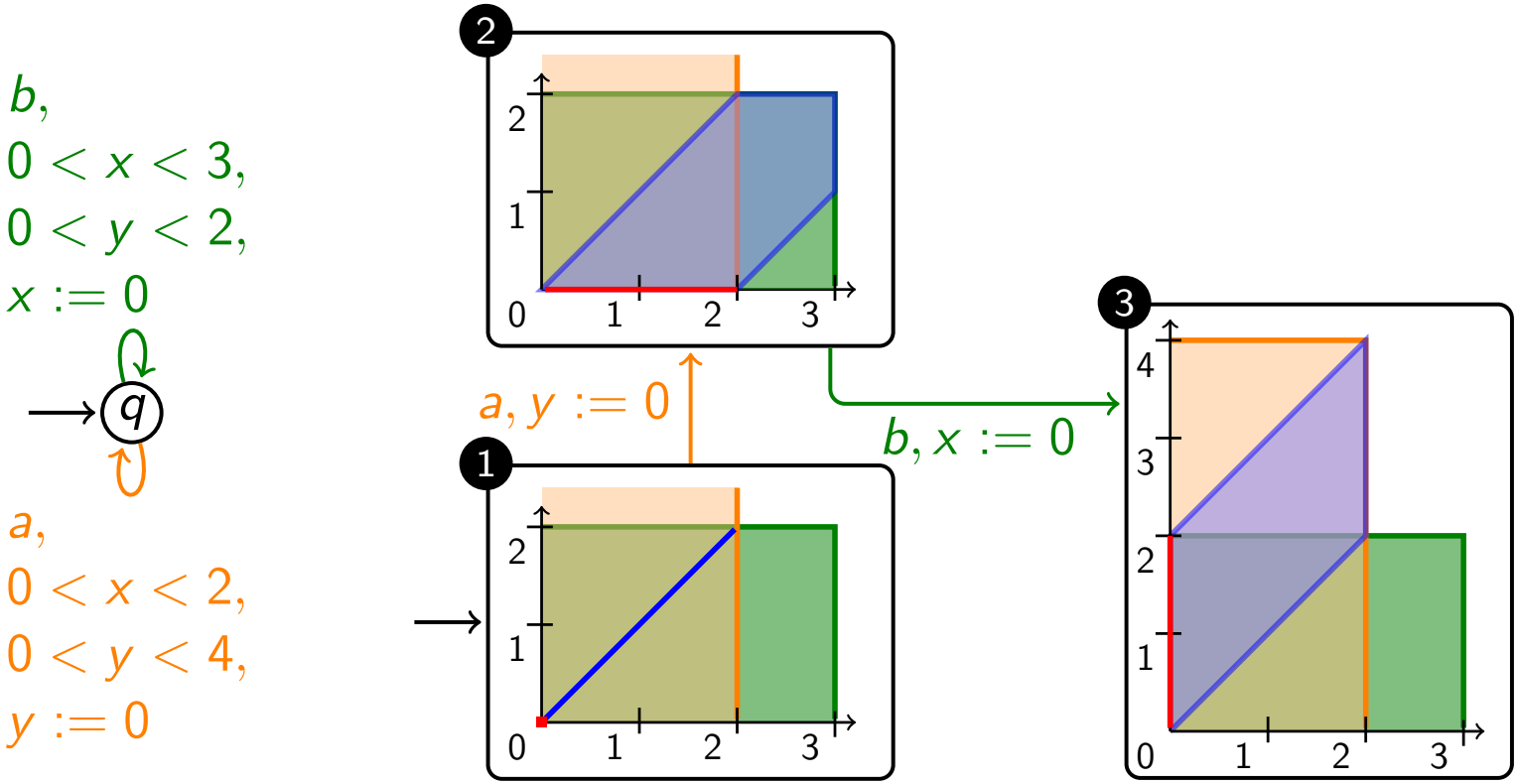
a,
$0 < x < 2$,
$0 < y < 4$,
$y := 0$



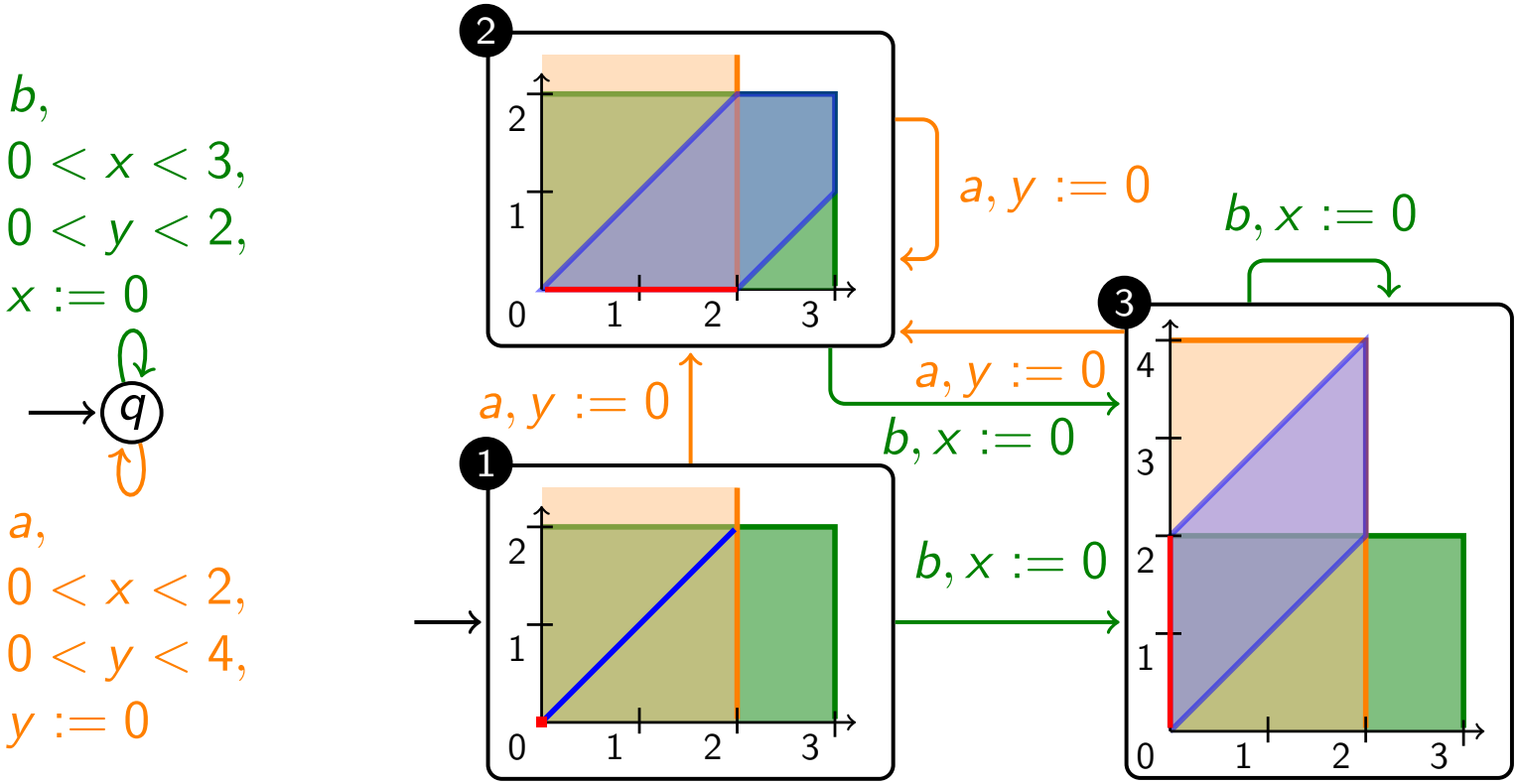
A first decomposition into zones (the forward reachability timed automaton).



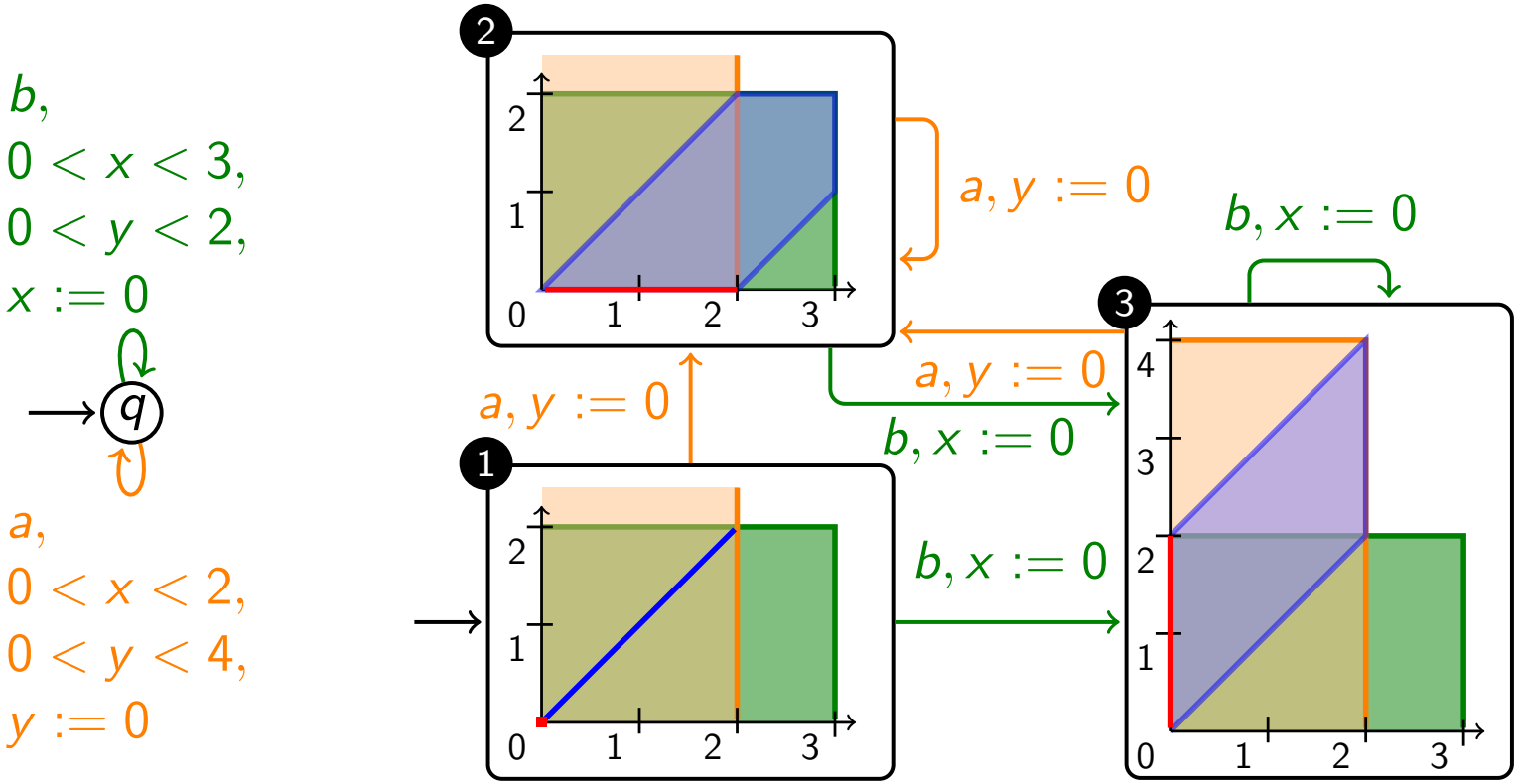
A first decomposition into zones (the forward reachability timed automaton).



A first decomposition into zones (the forward reachability timed automaton).



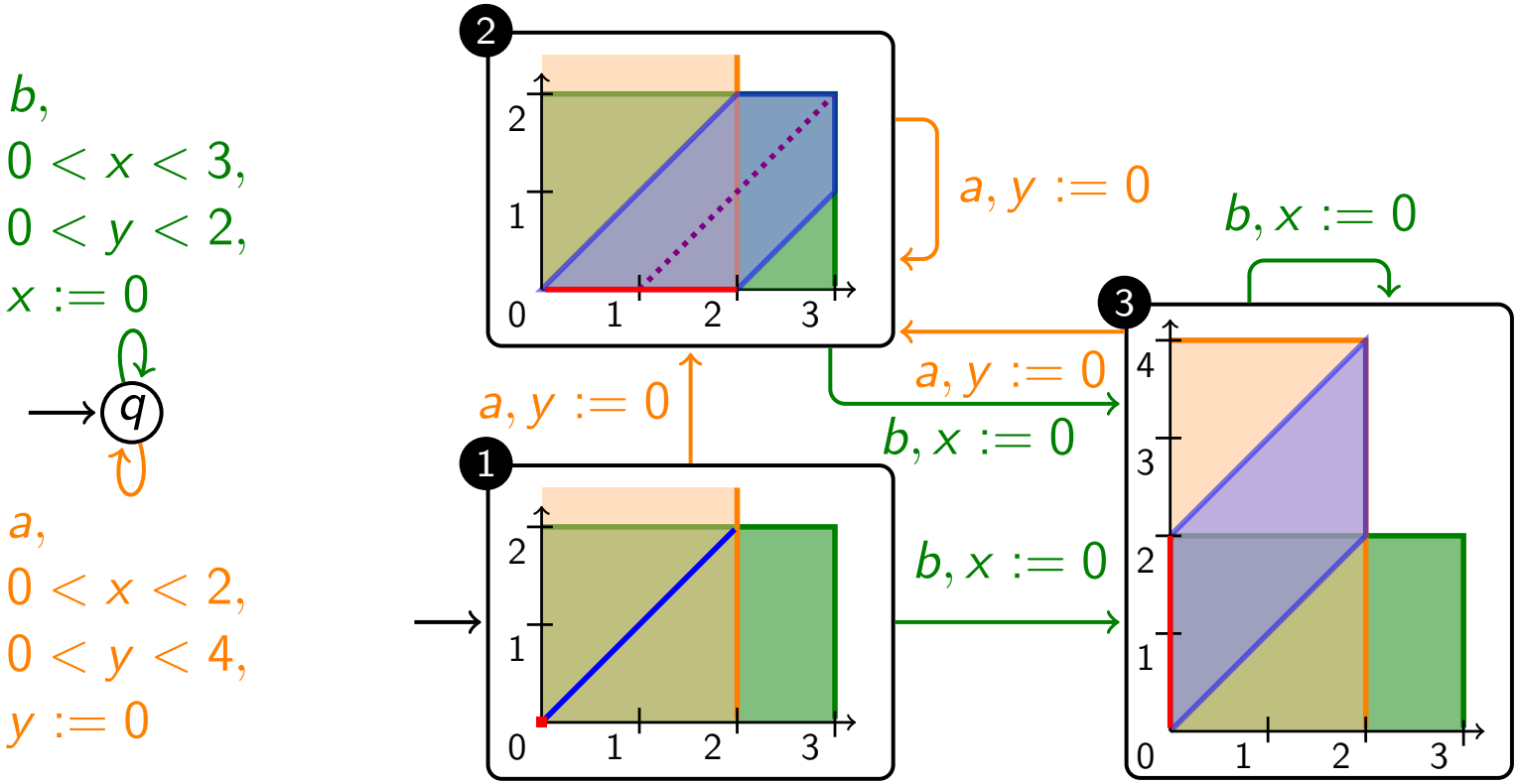
A first decomposition into zones (the forward reachability timed automaton).



One of the recursive equations to compute volumes

$$v_{n+1}[❷,(x,0)] = \int_0^{2-x} v_n(❷,(x+t,0))\mathrm{d}t + \int_0^{\min(2,3-x)} v_n(❸,(0,t))\mathrm{d}t.$$

A first decomposition into zones (the forward reachability timed automaton).

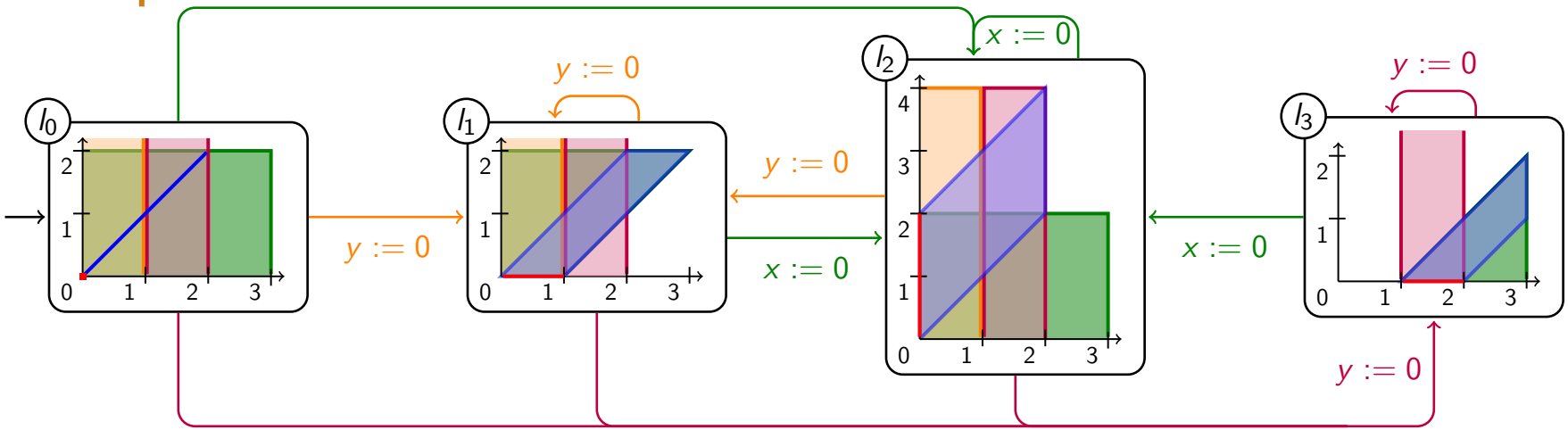


One of the recursive equations to compute volumes

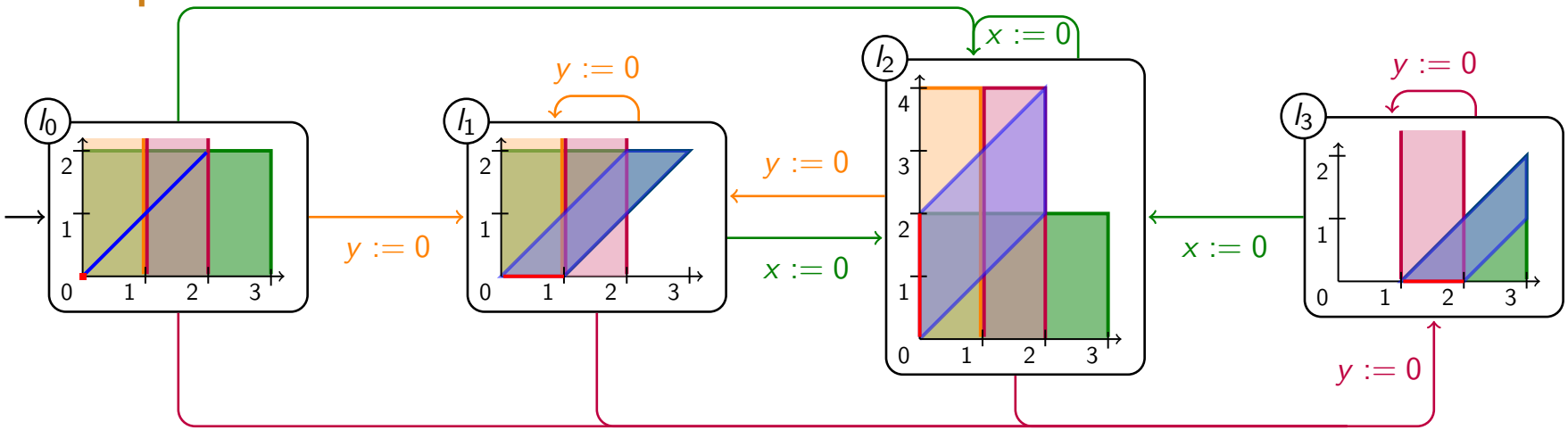
$$v_{n+1}[❷,(x,0)] = \int_0^{2-x} v_n(❷,(x+t,0))\mathrm{d}t + \int_0^{\min(2,3-x)} v_n(❸,(0,t))\mathrm{d}t.$$

A further split is needed to simplify $\min(2, 3-x)$.

The split timed automaton



The split timed automaton



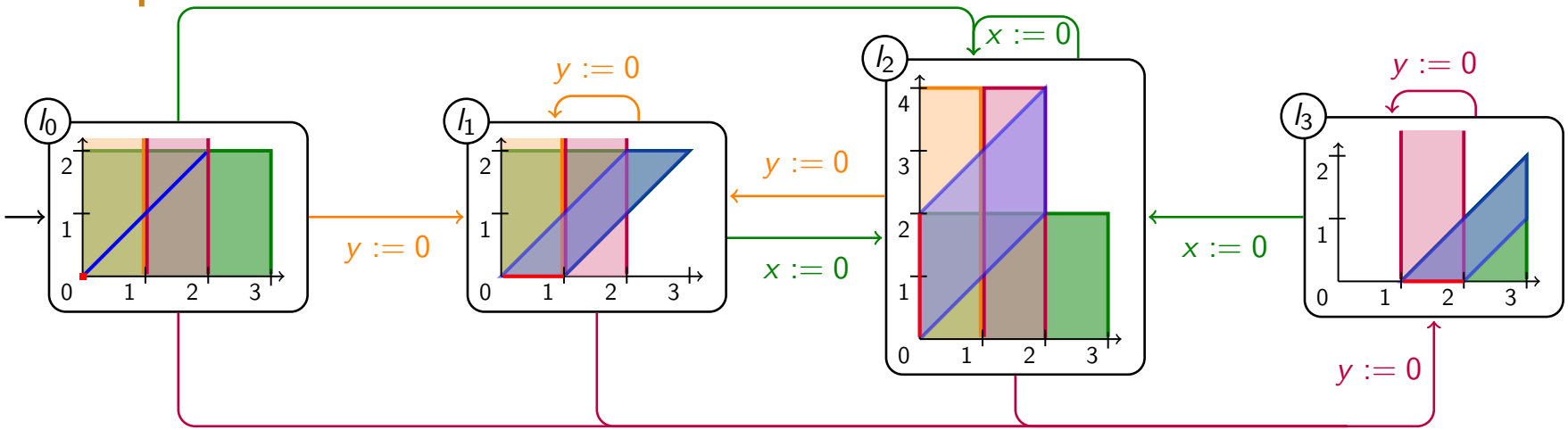
The previous equation is now split into two simpler equations

$$v_{n+1}[\textcircled{l_1}, (x, 0)] =$$

$$\int_0^{1-x} v_n(\textcircled{l_1}, (x+t, 0))\mathrm{d}t + \int_0^{2} v_n(\textcircled{l_2}, (0, t))\mathrm{d}t + \int_{1-x}^{2-x} v_n(\textcircled{l_3}, (x+t, 0))\mathrm{d}t;$$

$$v_{n+1}[\textcircled{l_3}, (x, 0)] = \int_0^{3-x} v_n(\textcircled{l_2}, (0, t))\mathrm{d}t + \int_0^{2-x} v_n(\textcircled{l_3}, (x+t, 0))\mathrm{d}t.$$

The split timed automaton



The previous equation is now split into two simpler equations

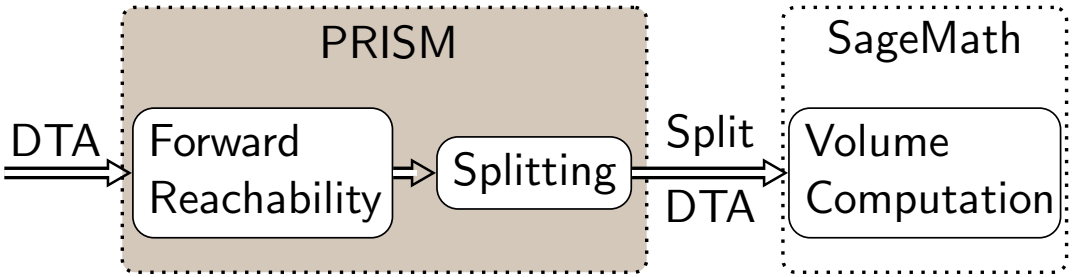
$$v_{n+1}[\textcircled{l_1}, (x,0)] =$$

$$\int_0^{1-x} v_n(\textcircled{l_1}, (x+t,0))\mathrm{d}t + \int_0^{2} v_n(\textcircled{l_2}, (0,t))\mathrm{d}t + \int_{1-x}^{2-x} v_n(\textcircled{l_3}, (x+t,0))\mathrm{d}t;$$

$$v_{n+1}[\textcircled{l_3}, (x,0)] = \int_0^{3-x} v_n(\textcircled{l_2}, (0,t))\mathrm{d}t + \int_0^{2-x} v_n(\textcircled{l_3}, (x+t,0))\mathrm{d}t.$$

Volume functions are polynomials computable in polynomial time wrt. n.
e.g. $v_3[\textcircled{$l_3$}, (x,0)] = -\frac{1}{6}x^3 - \frac{1}{2}x^2 - 25x + \frac{133}{2}$.

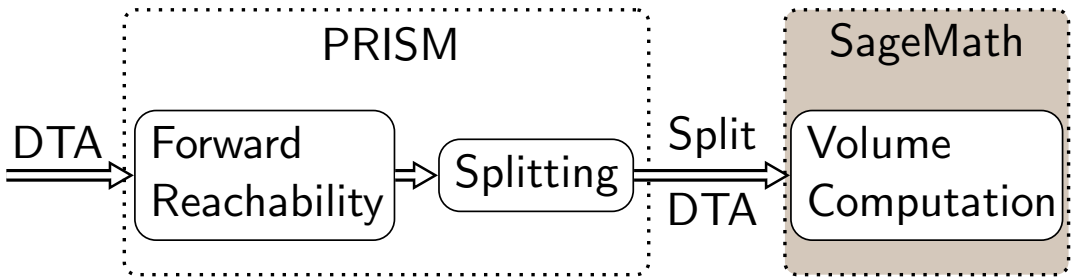
A first glimpse at our tool chain



PRISM (probabilistic model checker)

- ▷ Compute forward reachability zone graph
- ▷ Split the zone graph

A first glimpse at our tool chain



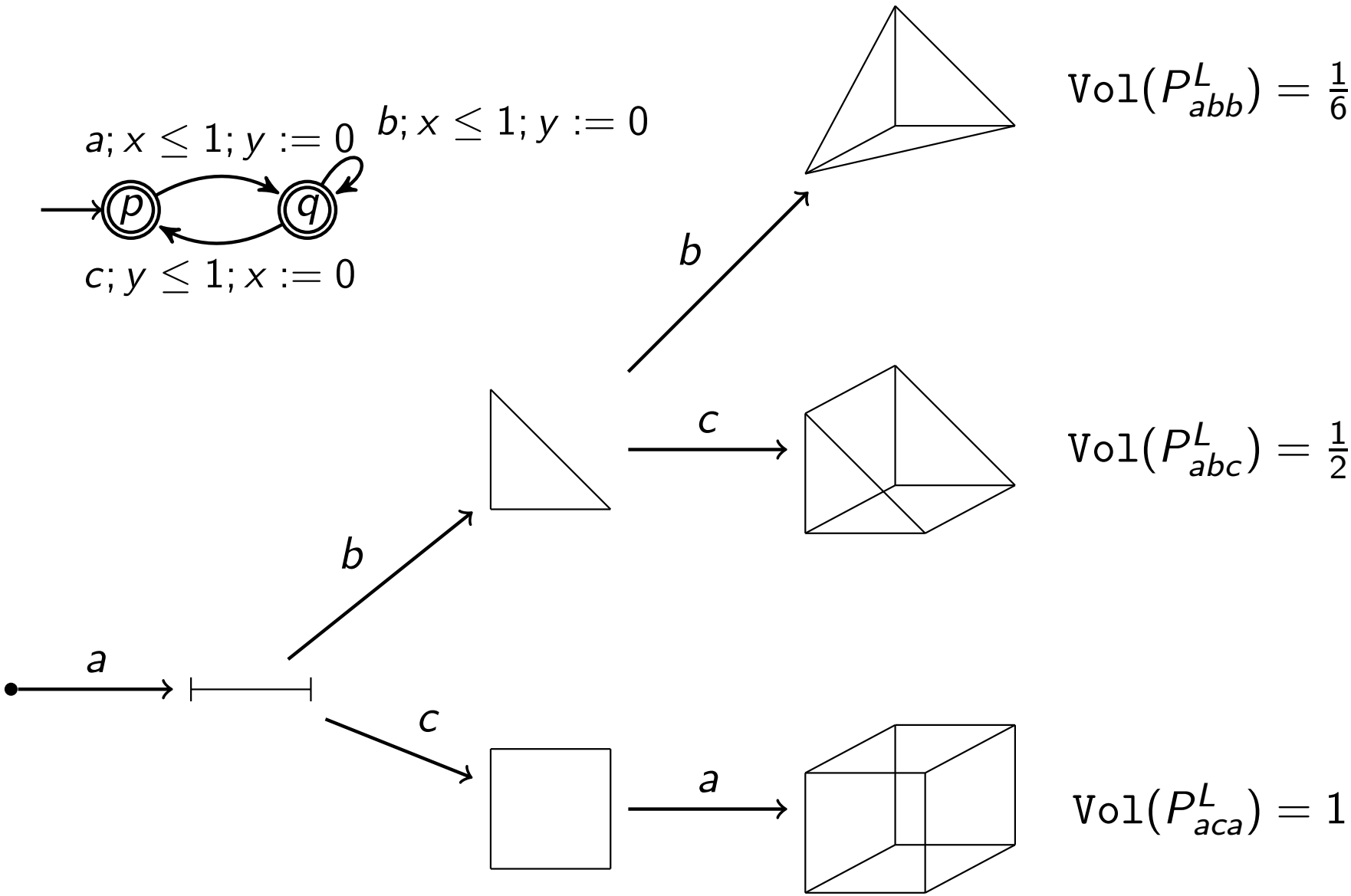
SageMath (open-source mathematics software)

- Compute volumes

Outline

Exact uniform sampling

Assign the same weight to every timed word of length n.



Several methods of (quasi)-uniform sampling

Recursive method

- ▷ At step k, the timed transition $s_k \xrightarrow{t_k, a_k} s_{k+1}$ is randomly picked with weight proportional to the volume of timed words of length $n - k$ from s_{k+1}.
- ▷ Drawback, necessitate to compute volume functions up to v_n.

Several methods of (quasi)-uniform sampling

Recursive method

- ▷ At step k, the timed transition $s_k \xrightarrow{t_k, a_k} s_{k+1}$ is randomly picked with weight proportional to the volume of timed words of length $n - k$ from s_{k+1}.
- ▷ Drawback, necessitate to compute volume functions up to v_n.

Receding horizon method (the one we implemented)

- ▷ Replace $n - k$ by a constant $m << n$.

Several methods of (quasi)-uniform sampling

Recursive method

- ▷ At step k, the timed transition $s_k \xrightarrow{t_k, a_k} s_{k+1}$ is randomly picked with weight proportional to the volume of timed words of length $n - k$ from s_{k+1}.
- ▷ Drawback, necessitate to compute volume functions up to v_n.

Receding horizon method (the one we implemented)

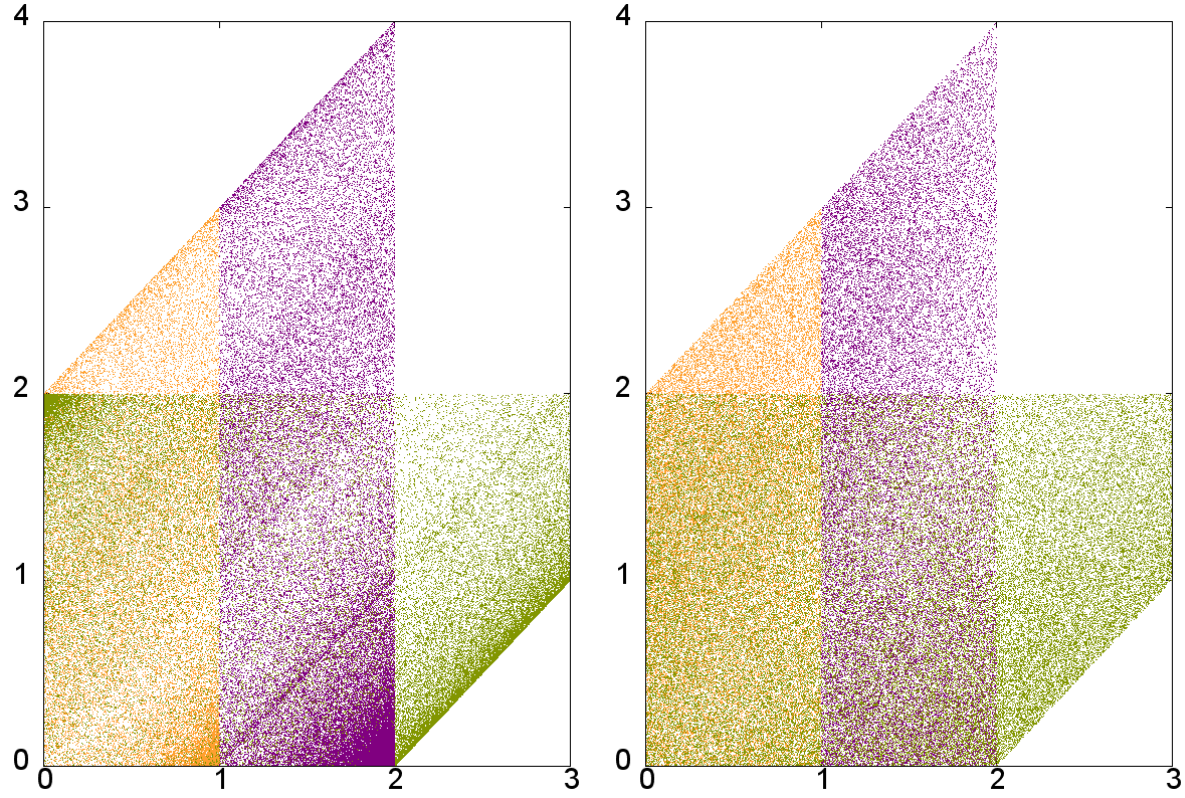
- ▷ Replace $n - k$ by a constant $m << n$.

Maximal entropy (infinite receding horizon)

- ▷ When $n \to \infty$ (or $m \to \infty$), we obtain a maximal entropy stochastic process [B., Information and Computation 2015].

Comparing isotropic sampling and receding horizon sampling with m=9

Isotropic="by default"= every discrete transition available has the same weight, every delay available has the same weight.



Isotropic $m = 9$

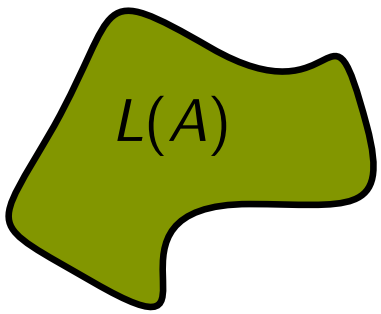
Generate a 200,000 steps trajectory

Outline

Statistical measure of general timed languages (1/2)

$L(A)$ a "complex" timed language

- A is a timed model, *possibly non-deterministic timed automaton*, *possibly stop-watch automaton*
- We only require to be able to check the membership of a word in the language



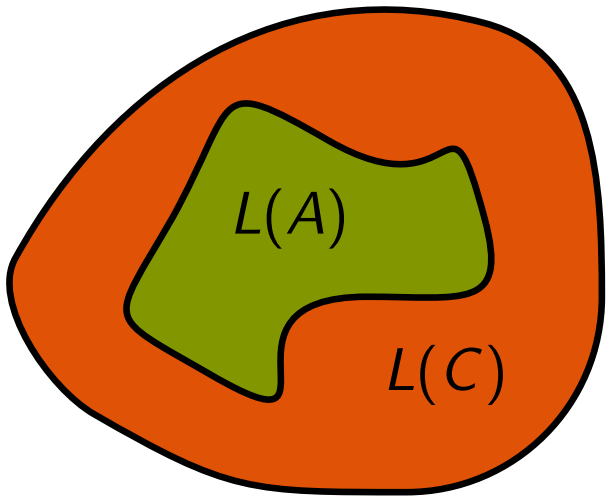
Statistical measure of general timed languages (1/2)

$L(A)$ a "complex" timed language

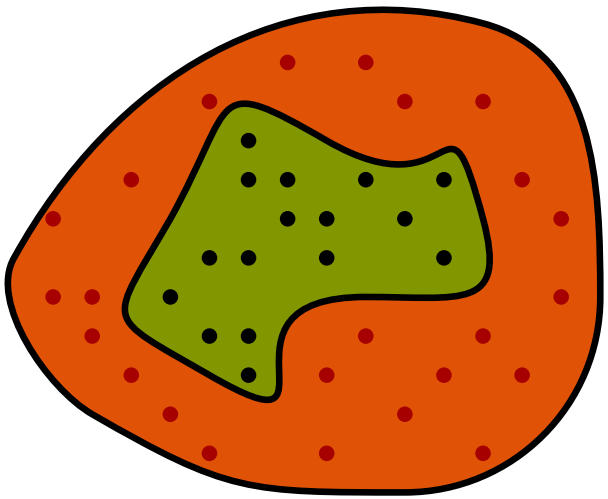
- A is a timed model, *possibly non-deterministic timed automaton*, *possibly stop-watch automaton*
- We only require to be able to check the membership of a word in the language

$L(C)$ a "simple" over-approximation: of the language $L(A) \subseteq L(C)$

- C is a deterministic time automaton
- We can compute the volume of its language
- We can sample uniformly from its language



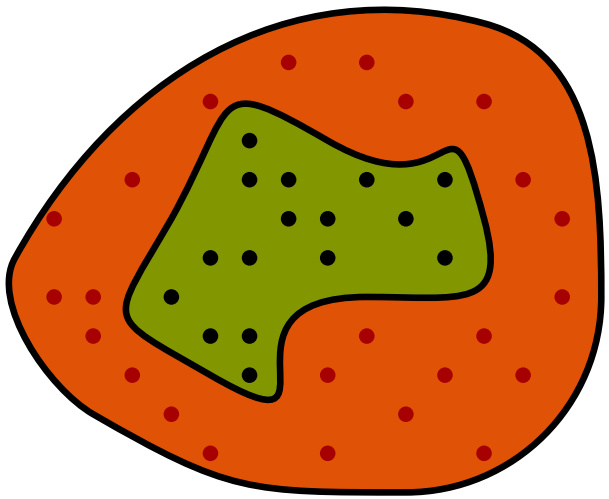
Statistical measure of general timed languages (2/2)



Sampling in the over-approximation

- ▷ Sample uniformly timed words of length n in the language $L_n(C)$ (by previous methods for DTAs).

Statistical measure of general timed languages (2/2)



Sampling in the over-approximation

- ▷ Sample uniformly timed words of length n in the language $L_n(C)$ (by previous methods for DTAs).

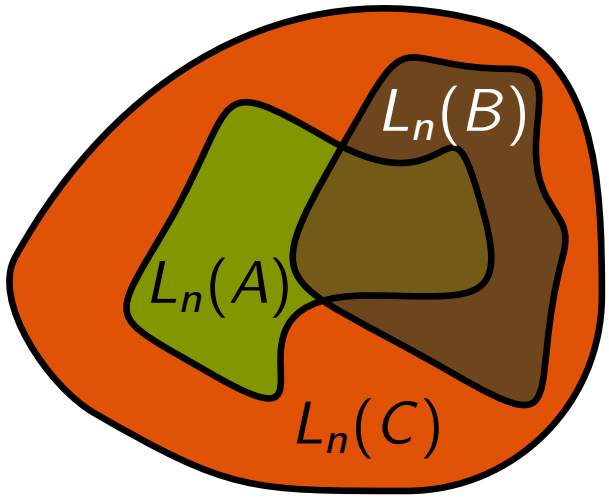
Filtering

- ▷ For each trajectory in $L_n(C)$ check the membership in $L_n(A)$
- ▷ Estimate the volume of $\texttt{Vol}_n(L_n(A))$ as $\frac{\#\text{trajectory in } L_n(A)}{\#\text{trajectory in } L_n(C)} \cdot \texttt{Vol}_n(L_n(C))$

Statistical language inclusion measurement

Two languages

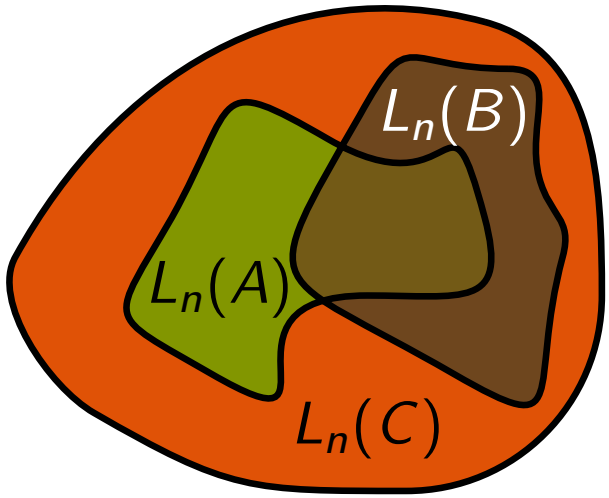
- $L(A)$ and $L(B)$ are "complex" languages
- Use an over-approximation $L(C)$ with C a DTA.
- Using membership of automaton estimate
  $\mathtt{Vol}_n\left(L_n(A) \cap L_n(B)\right)$,
  $\mathtt{Vol}_n\left(L_n(A) \setminus L_n(B)\right)$,
  $\mathtt{Vol}_n\left(L_n(A) \,\Delta\, L_n(B)\right) \ldots$



Statistical language inclusion measurement

Two languages

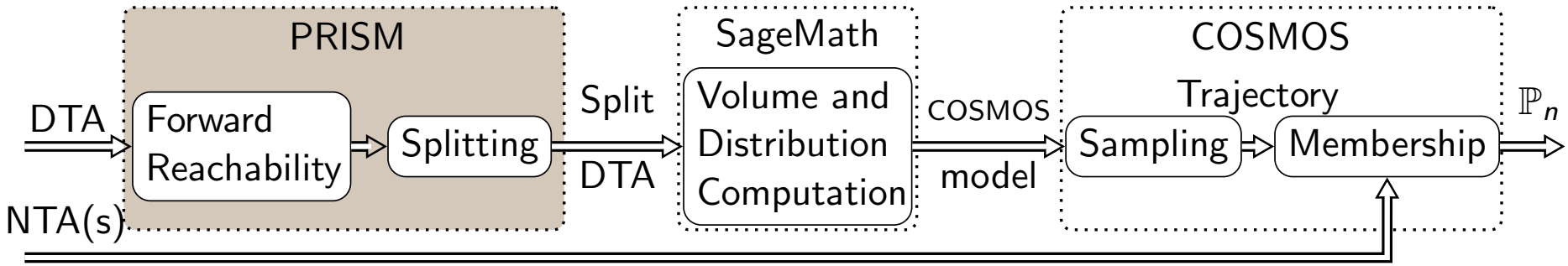
- $L(A)$ and $L(B)$ are "complex" languages
- Use an over-approximation $L(C)$ with C a DTA.
- Using membership of automaton estimate
  $\mathtt{Vol}_n\left(L_n(A) \cap L_n(B)\right)$,
  $\mathtt{Vol}_n\left(L_n(A) \setminus L_n(B)\right)$,
  $\mathtt{Vol}_n\left(L_n(A) \Delta L_n(B)\right) \ldots$



Checking $L_n(A) \subset L_n(B)$ (up to null volume measure)

$$\mathbb{P}_n(B|A) = \frac{\mathtt{Vol}_n\left(L_n(A) \cap L_n(B)\right)}{\mathtt{Vol}_n(L_n(A))} = 1$$

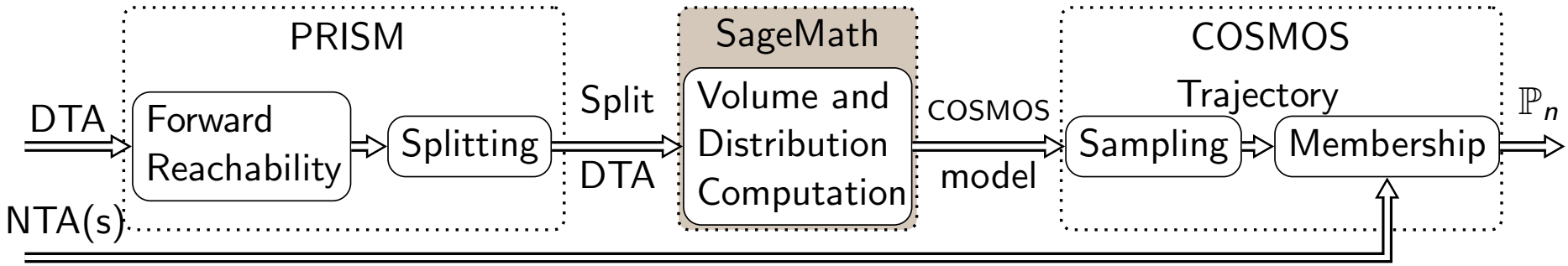
Toolchain



PRISM (probabilistic model checker)

- ▷ Compute forward reachability zone graph.
- ▷ Split the zone graph.

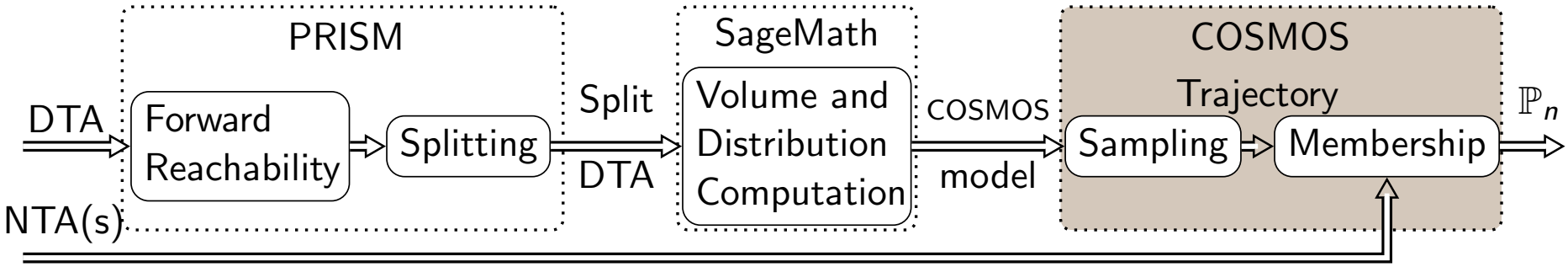
Toolchain



SageMath (open-source mathematics software)

- ▷ Compute volumes and *distributions*.
- ▷ Output a fully probabilistic COSMOS model.

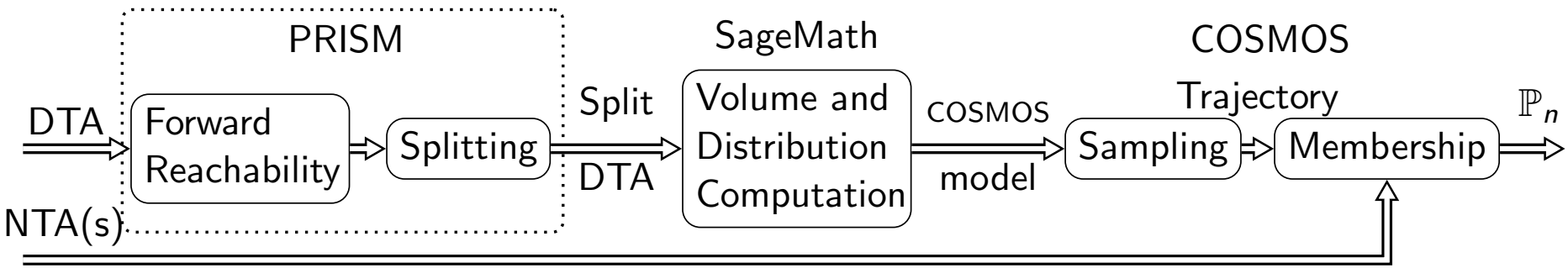
Toolchain



COSMOS (statistical model checker)

- Sampling of trajectories of the probabilistic model from SageMath.
- Output estimation of $\frac{\mathtt{Vol}_n(L_n(A))}{\mathtt{Vol}_n(L_n(C))}$ or estimation of $\frac{\mathtt{Vol}_n(L_n(A) \cap L_n(B))}{\mathtt{Vol}_n(L_n(A))}$ (for language inclusion).

Toolchain



COSMOS (statistical model checker)

- ▷ Sampling of trajectories of the probabilistic model from SageMath.
- ▷ Output estimation of $\frac{\mathtt{Vol}_n(L_n(A))}{\mathtt{Vol}_n(L_n(C))}$ or estimation of $\frac{\mathtt{Vol}_n(L_n(A) \cap L_n(B))}{\mathtt{Vol}_n(L_n(A))}$ (for language inclusion).

A case study of a repair failure model is available in [Barbot B. Beunardeau Kwiatkowska, Qest'16].

Outline

We have seen how to

- measure timed languages in terms of volumes;
- adopt a zone-based framework to compute efficiently such volumes;
- sample (quasi-)uniformly timed words for DTA;
- apply this to measure and sampling for general timed languages.

We have seen how to

- measure timed languages in terms of volumes;
- adopt a zone-based framework to compute efficiently such volumes;
- sample (quasi-)uniformly timed words for DTA;
- apply this to measure and sampling for general timed languages.

What we plan to do:

- experiment more with the same theory
  - implement membership for more expressive timed languages (stopwatch, hybrid);
  - develop bigger case studies;
- extend the theory
  - develop random generation mehods uniform on timed words of same duration (as opposed to uniform on timed words of same length).
  - develop uniform random generation for networks of TAs.