FAKULTÄT
FÜR !NFORMATIK

Faculty of Informatics

Liveness of Parameterized Timed Networks

Florian Zuleger
Technische Universität Wien

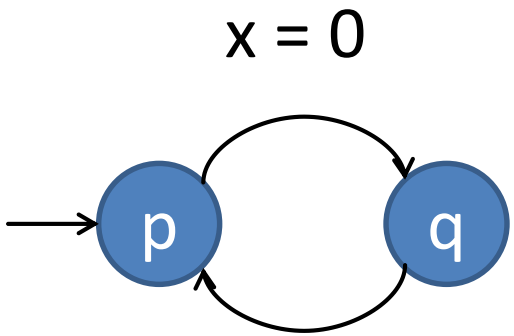
Joint work with Benjamin Aminof,
Sasha Rubin, Francesco Spegni

Timed Automata - Syntax

Labeled transition system:

- finite set of **states**

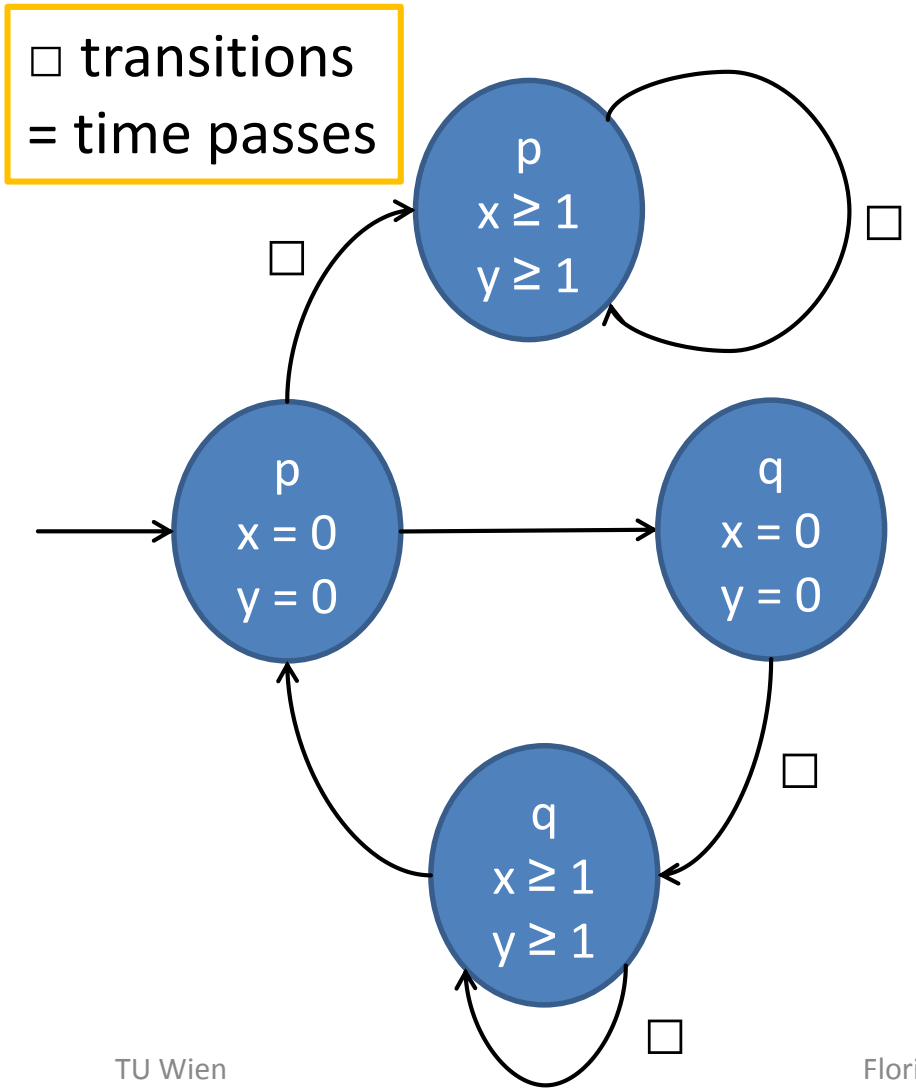
  (one **initial state**)
- finite set of **clocks**
- transitions labeled by **guards** and **resets**
- guard = **comparison** of a **clock** to a **constant**

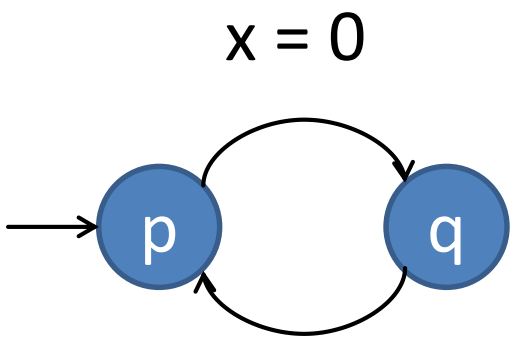


Time is either **continuous** or **discrete**.

this talk

Timed Automata - Semantics

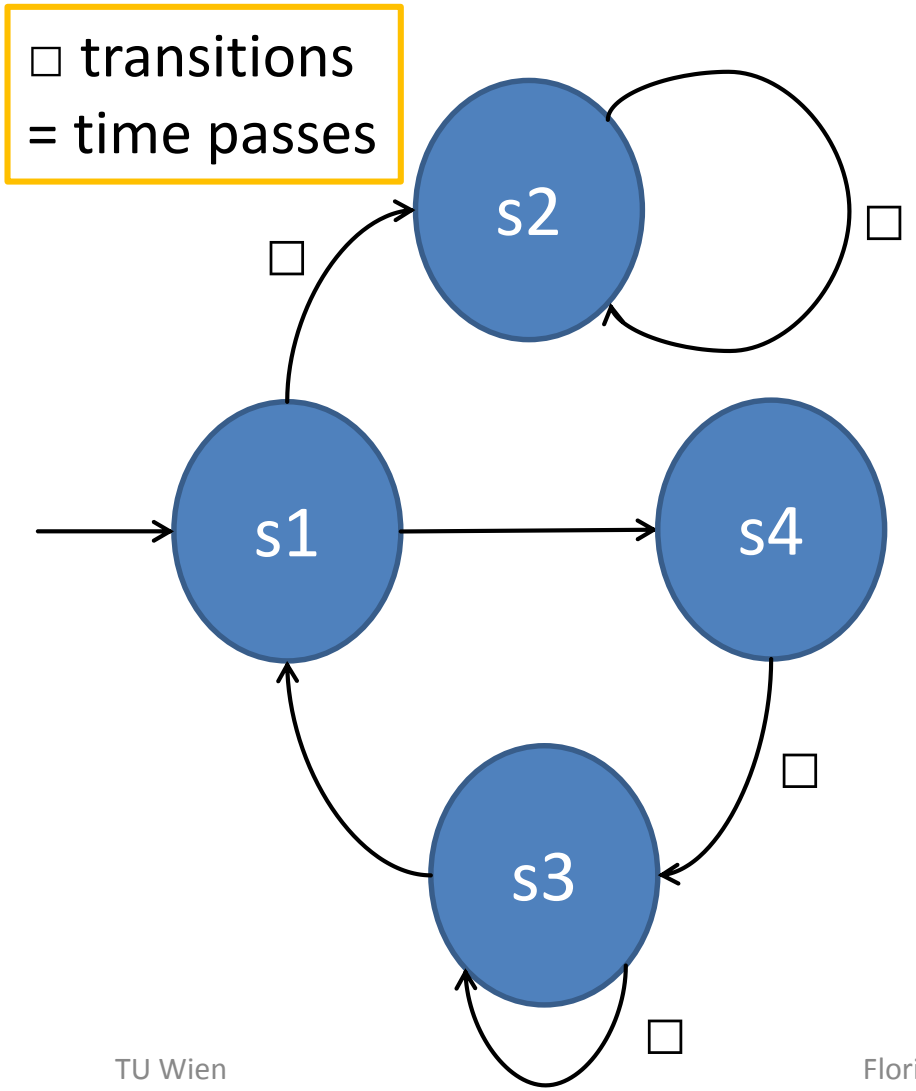




Alternative Representation:

- Explicit passage of time
- Clock values in states
- Finite number of clock values are sufficient

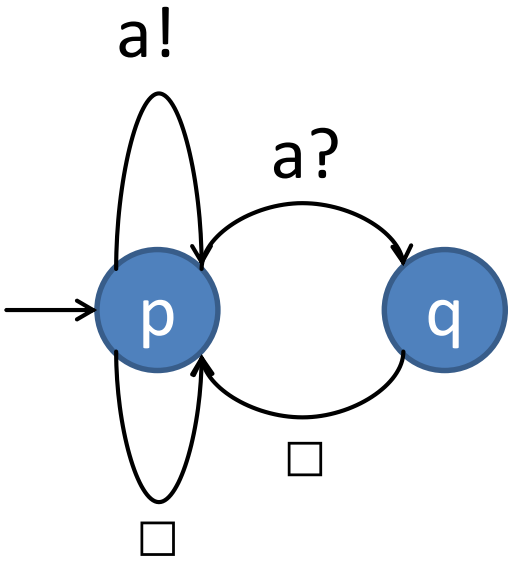
Timed Automata – Alternative Representation



Forget about clocks!

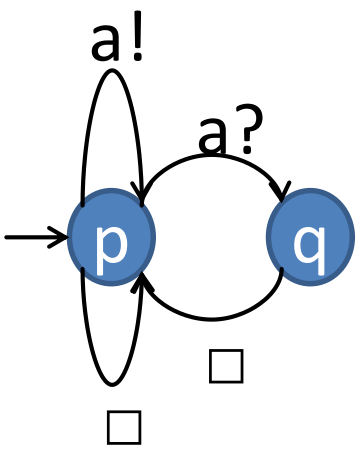
For the rest of the talk, we use this representation.

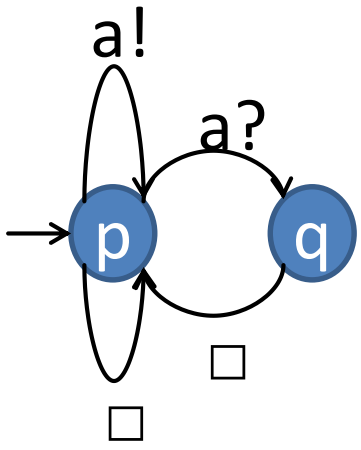
Timed Networks

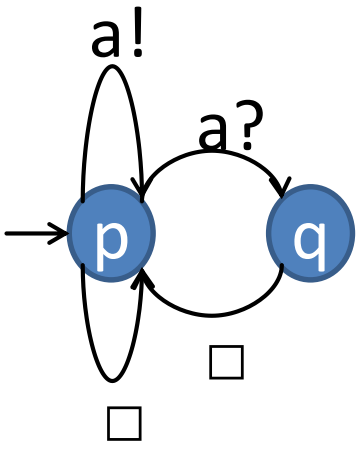


Timed Network =

finite number of copies of the same timed automaton + communication via rendezvous transitions





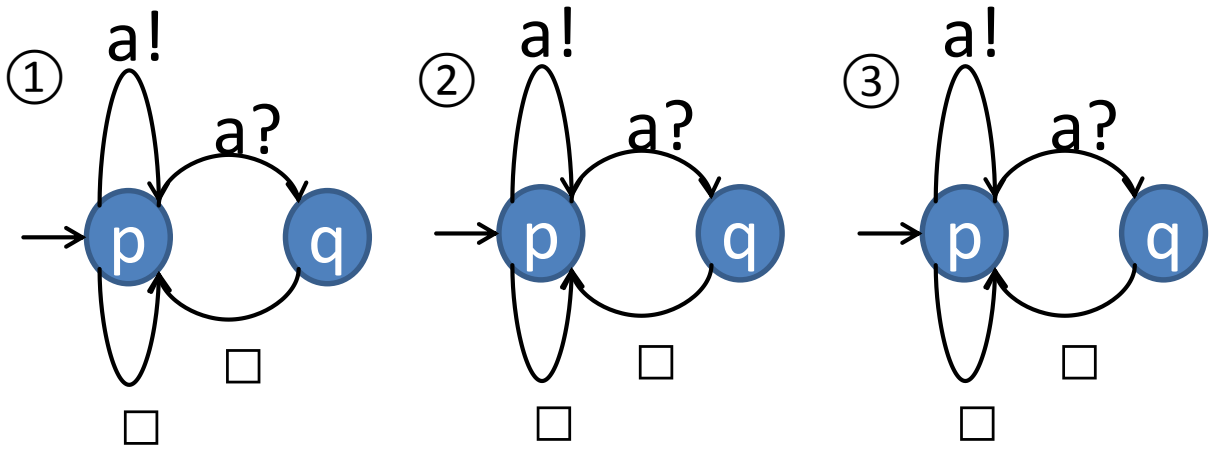


Timed Networks

Communication alphabet $\Sigma = \{a!, a?\} \cup \{\square\}$

Example run:

①	p									
②	p									
③	p									



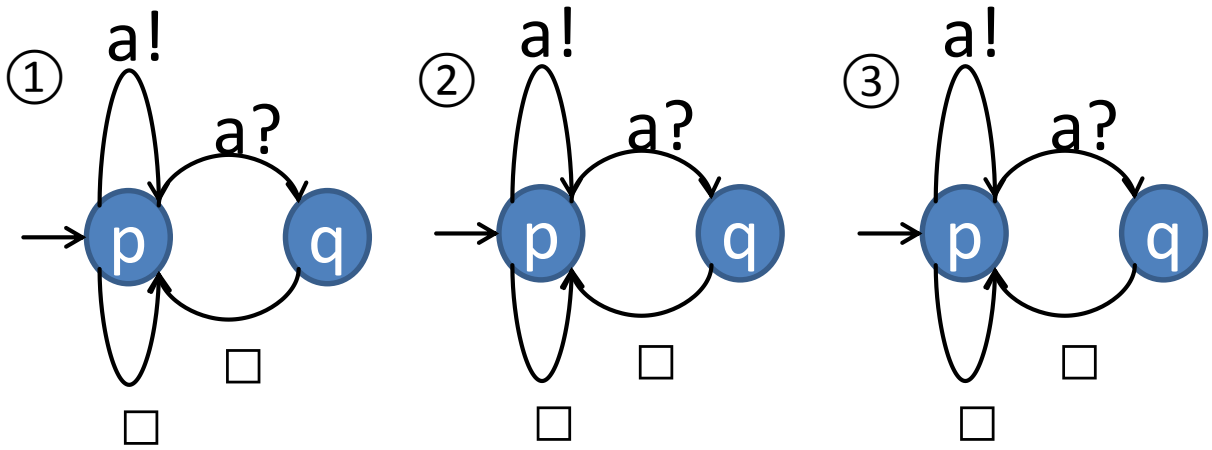
Timed Networks

Communication alphabet $\Sigma = \{a!, a?\} \cup \{\square\}$

Example run:

Rendezvous transition

①	p	a!	p							
②	p	a?	q							
③	p		p							

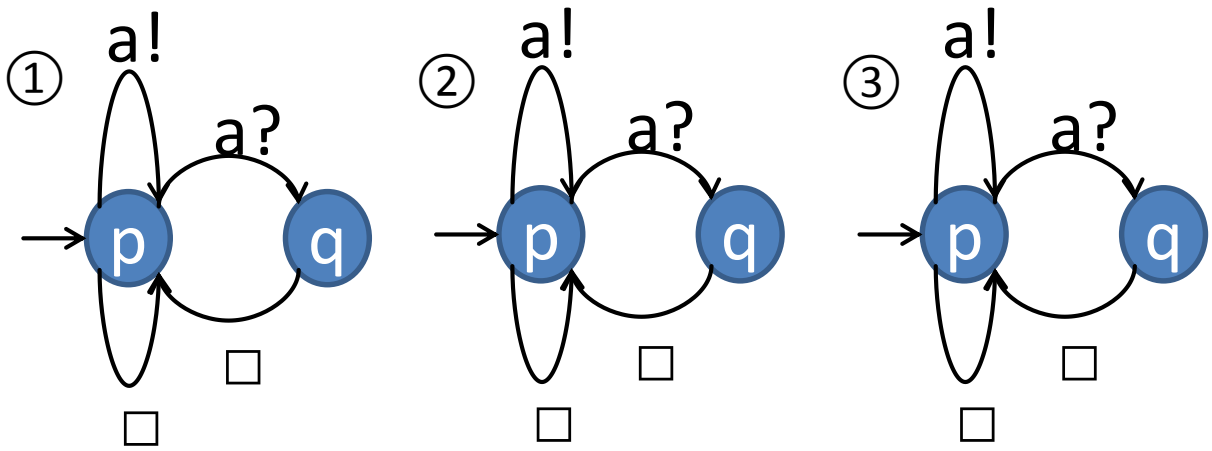


Timed Networks

Communication alphabet $\Sigma = \{a!, a?\} \cup \{\square\}$

Example run:

①	p	a!	p	a!	p					
②	p	a?	q		q					
③	p		p	a?	q					



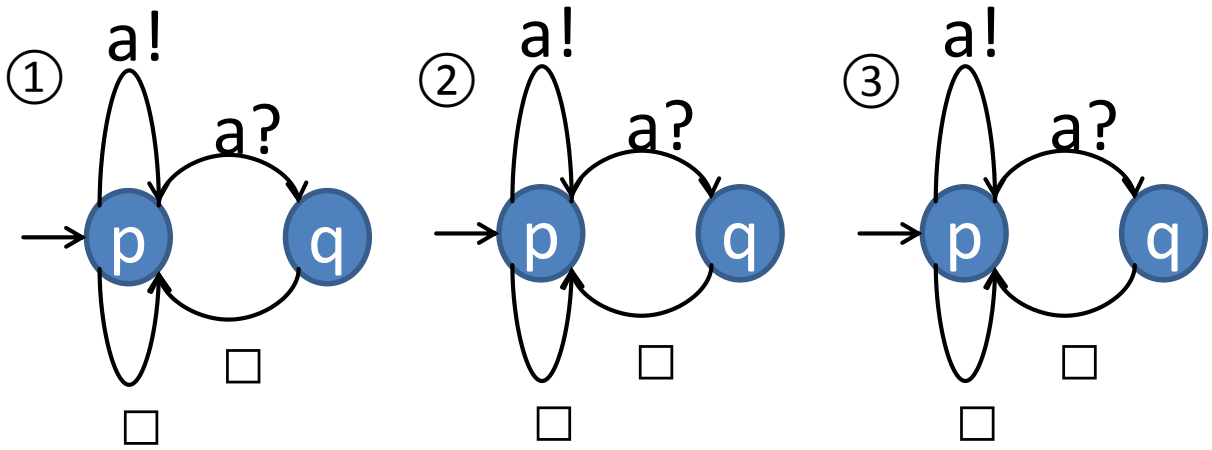
Timed Networks

Communication alphabet $\Sigma = \{a!, a?\} \cup \{\square\}$

Example run:

Time passing transition

①	p	a!	p	a!	p	□	p			
②	p	a?	q		q	□	p			
③	p		p	a?	q	□	p			

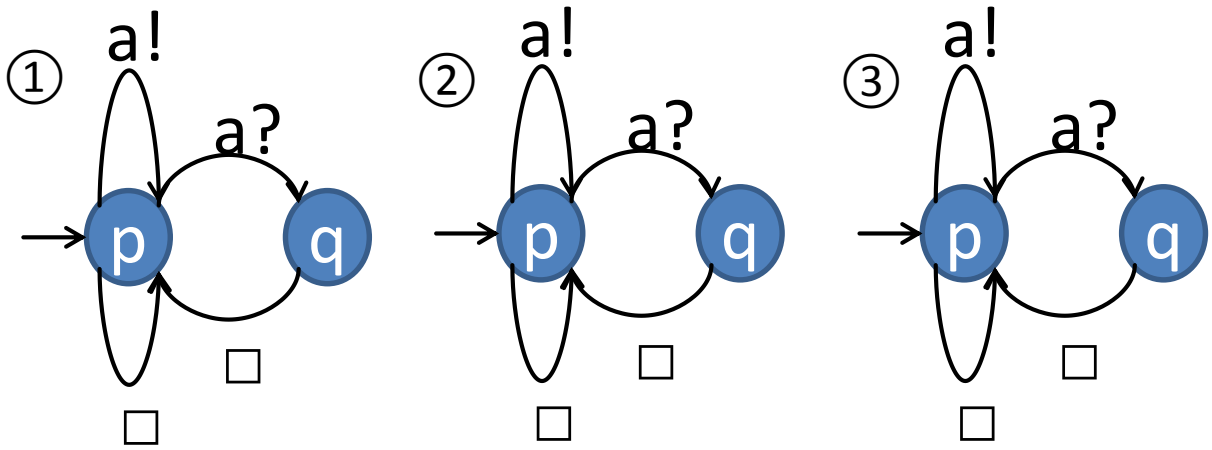


Timed Networks

Communication alphabet $\Sigma = \{a!, a?\} \cup \{\square\}$

Example run:

①	p	a!	p	a!	p	□	p	a?	q	
②	p	a?	q		q	□	p		p	
③	p		p	a?	q	□	p	a!	p	

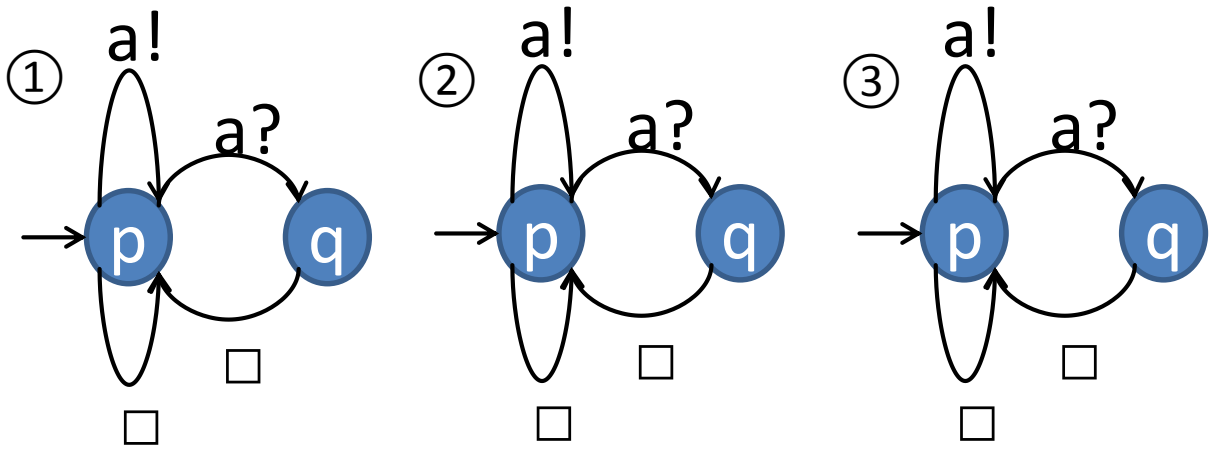


Timed Networks

Communication alphabet $\Sigma = \{a!, a?\} \cup \{\square\}$

Example run:

①	p	a!	p	a!	p	□	p	a?	q	…
②	p	a?	q		q	□	p		p	…
③	p		p	a?	q	□	p	a!	p	…



Timed Networks

Communication alphabet $\Sigma = \{a!, a?\} \cup \{\square\}$

Example run:

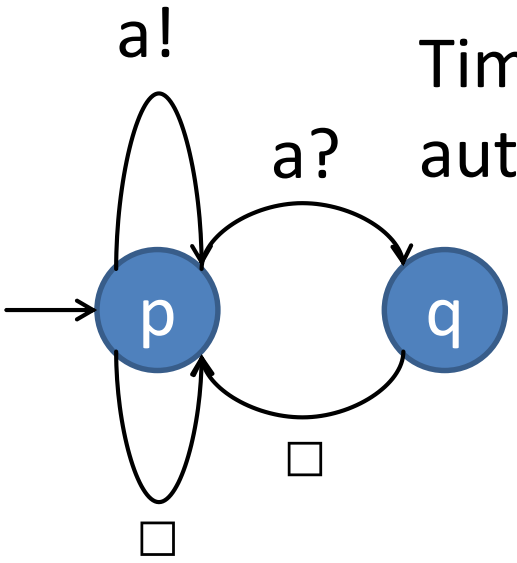
①	p	a!	p	a!	p	□	p	a?	q	…
②	p	a?	q		q	□	p		p	…
③	p		p	a?	q	□	p	a!	p	…

Execution of ③ in the run:

a?	□	a!	…

execution =
a sequence in Σ^{ω}

Parameterized Model Checking



Timed automaton A

Communication alphabet Σ

$Exec(A_n)$ = all executions of a timed network with n copies of automaton A

$Exec(A) = \bigcup_{n \geq 0} Exec(A_n)$

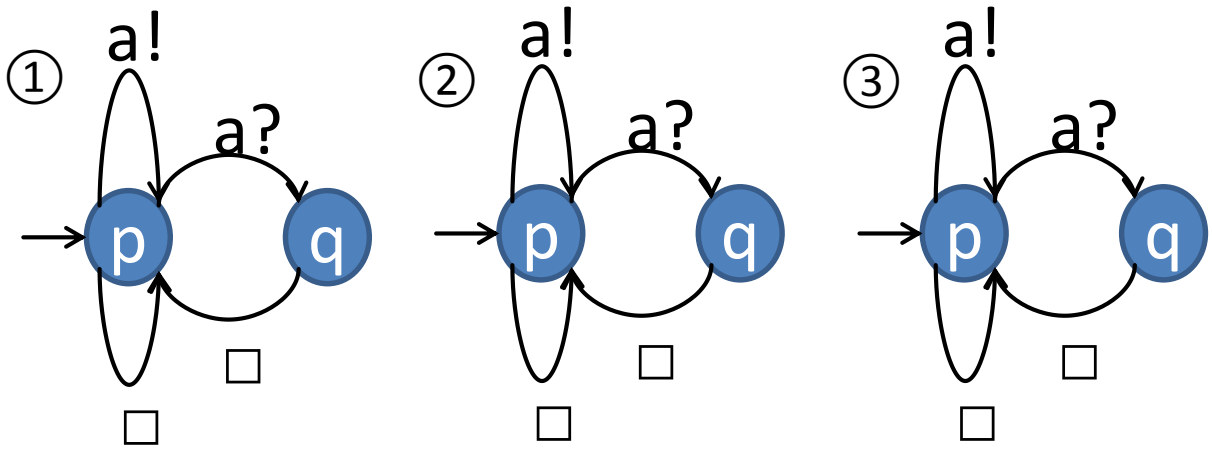
Parameterized Model Checking Problem (PMCP):
Given a language $L \subseteq \Sigma^\omega$, Liveness Property
decide $Exec(A) \subseteq L$?

Timed Networkds = RB-Systems

RB Systems = finite automata communicating via

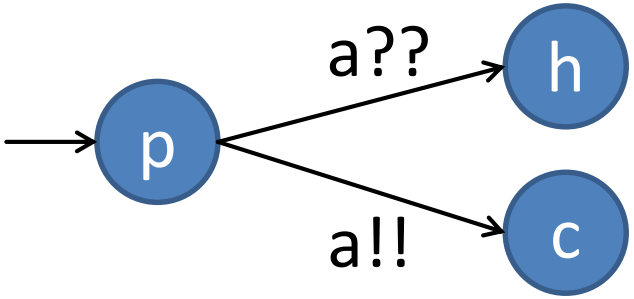
- rendezvous transitions
- symmetric broadcast transitions

①	p	a!	p	a!	p	□	p	a?	q	…
②	p	a?	q		q	□	p		p	…
③	p		p	a?	q	□	p	a!	p	…



(I) Why RB-Systems?

PMCP of liveness properties for finite automata communicating via **(asymmetric) broadcast** is undecidable (Esparza, Finkel, Mayr, LICS 1999)



Asymmetric broadcast is very powerful:

- allows to establish a controller process
- allows to simulate rendezvous transitions

(II) Why RB-Systems?

PMCP of liveness properties is undecidable (Abdulla, Jonsson, TCS 2003) for timed networks with

- continuous-time
- a distinguished controller process
- rendezvous transitions

Proof heavily relies on

- time being dense
- controller for coordination

Main Result

Theorem

Given a timed automaton A, we can compute a B-automaton B such that Exec(A) = L(B).

Corollary

PMCP is decdiable for specifications given by a BS-automaton*.

BS-automata (Bojanczyk, Colcombet LICS 2006):

- decidable emptiness
- closed under union, intersection
- not closed under complement
- subclasses B- and S-automata that are closed under complement
- strictly generalize ω-regular languages

Theo

Given

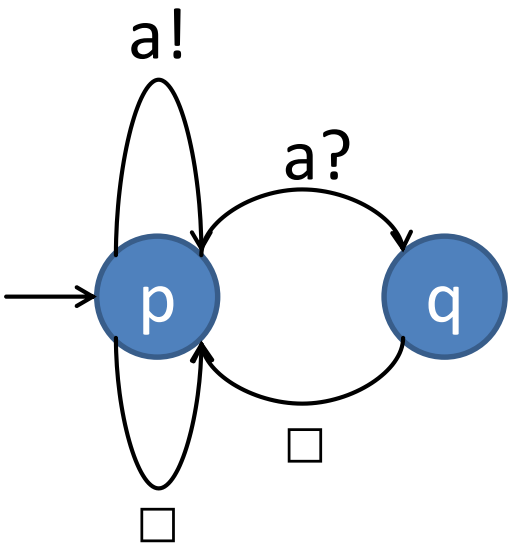
B-aut

Corollary

PMCP is decdiable for specifications given by a BS-automaton*.

Why BS-automata?

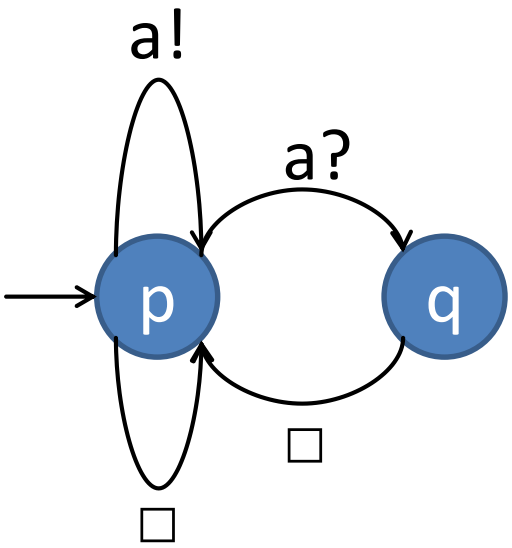
a!,a? may only boundedly often be taken between two □!



①	p	a!	p	a!	p	□	p	a?	q	…
②	p	a?	q		q	□	p		p	…
③	p		p	a?	q	□	p	a!	p	…

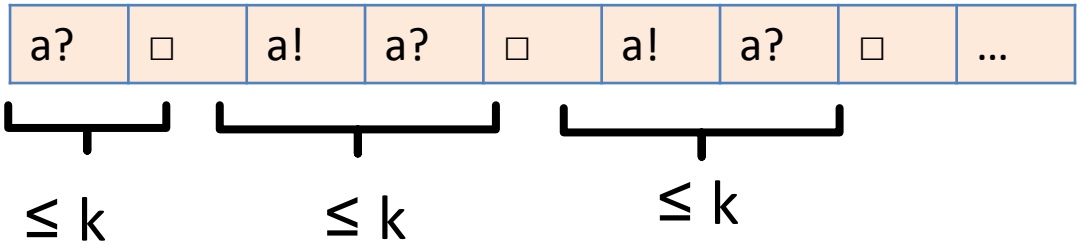
Why BS-automata?

a!,a? may only boundedly often be taken between two □!



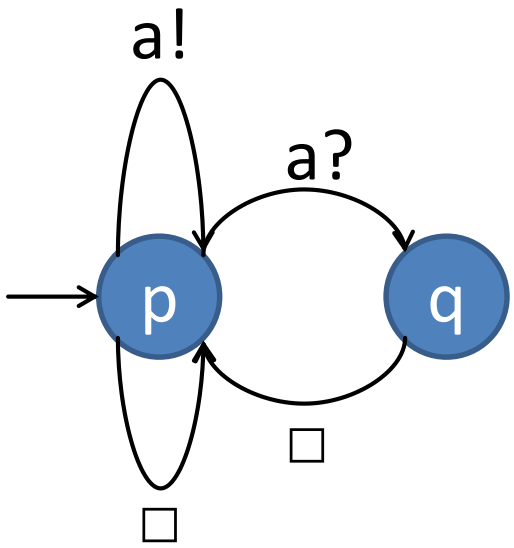
„boundedly often“ =

there is a $k \in \mathbb{N}$ with



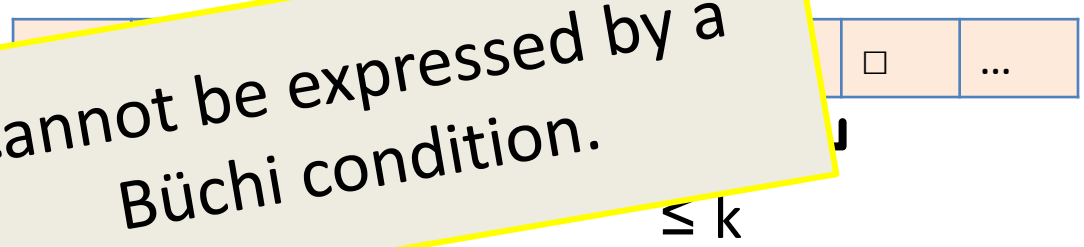
Why BS-automata?

a!,a? may only boundedly often be taken between two □!



„boundedly often“ =

there is a $k \in \mathbb{N}$ with

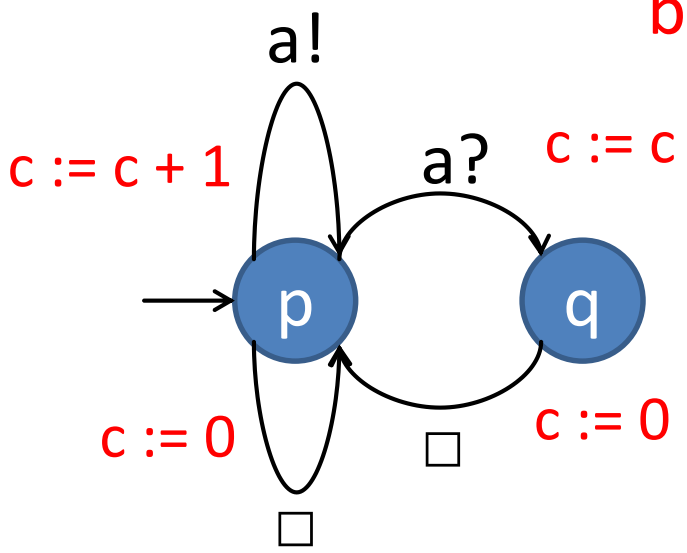


Cannot be expressed by a Büchi condition.

BS-automata

„counter c is bounded“

BS-automata have finite number of counters



Counters can be

1) reset,
2) incremented,
3) assigned to other counters

Acceptance condition =
positive boolean combination of Büchi condition + „counter is bounded“ + „counter goes to ∞“

4 Types of Automata Edges

Red: appears at most finitely often on any execution

Blue: appears infinitely times on some execution, but only finitely often on every execution with infinitely many broadcasts

Orange: appears infinitely times on some execution with infinitely many broadcasts, but only boundedly many times between two broadcasts

Green: otherwise

4 Types of Automata Edges

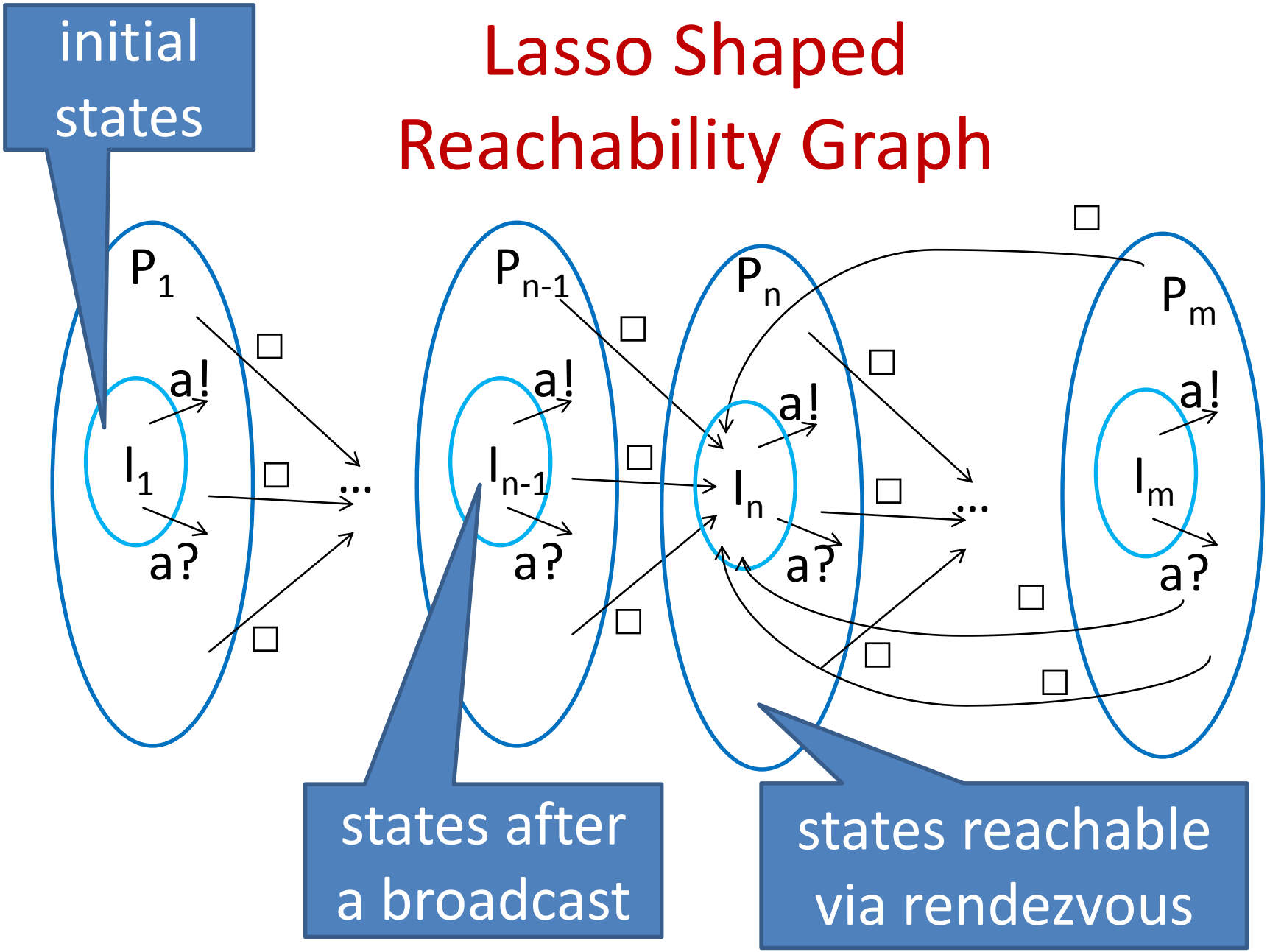
Red: appears at most finitely often in any execution

Proof strategy:
1) establish edge types
2) add a suitable counter update to each transition
3) equip the automaton with a suitable B-acceptance condition

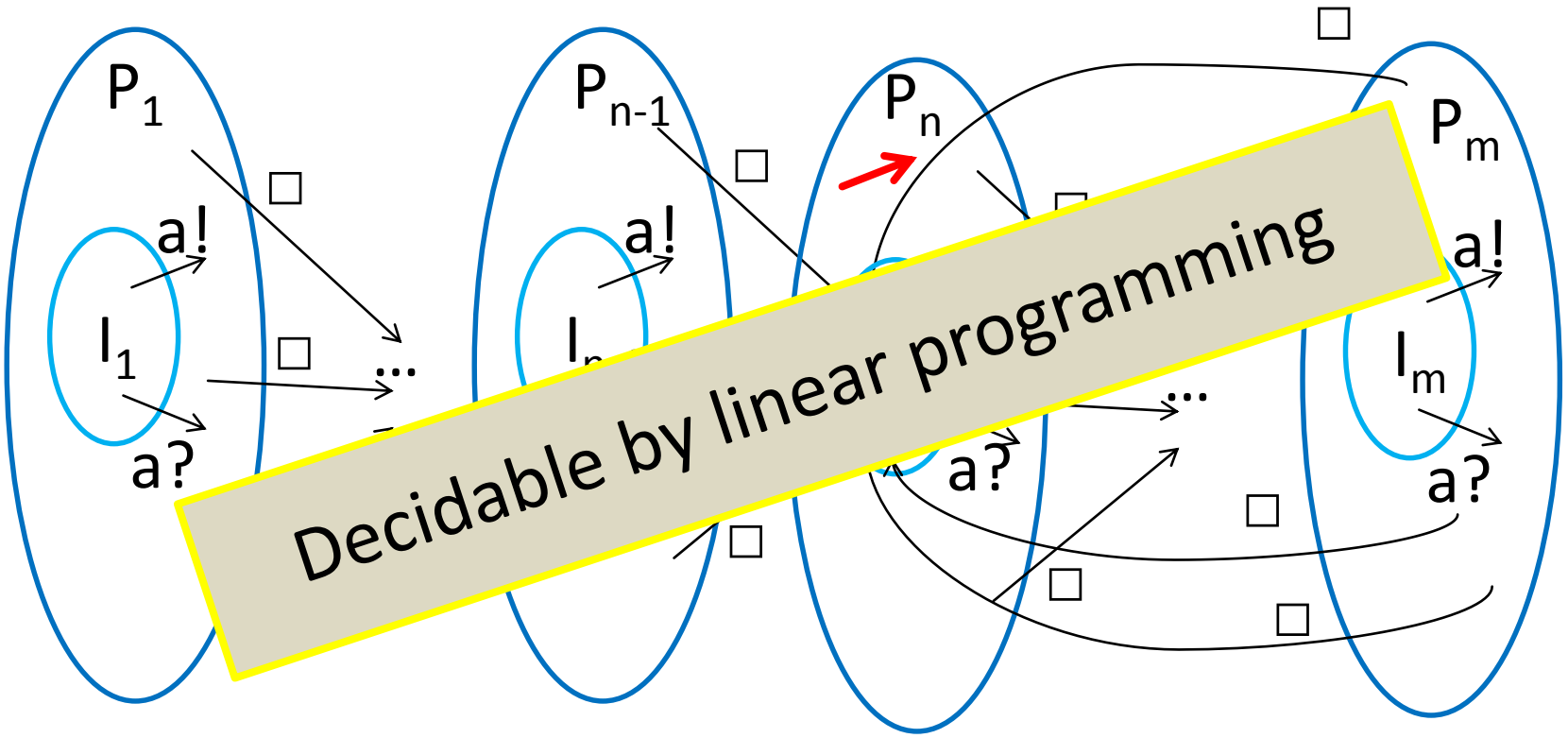
…times on some execution
…many broadcasts, but only boundedly many times between two broadcasts

Green: otherwise

Lasso Shaped Reachability Graph



Deciding Edge Types



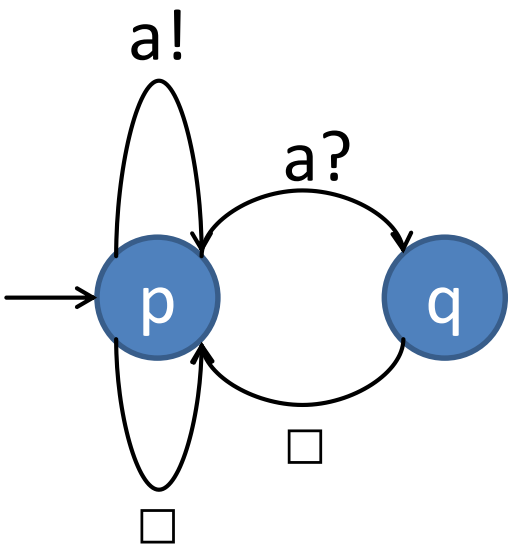
Essential question:

Is there a cyclic run of the lasso that uses edge ↗?

Linear Program by Example

$I_1 = I_2 = \{p\}$

$P_1 = P_2 = \{p,q\}$



variables $x_1,x_2,y_1,y_2 \in Q$ for the number of automata in state p resp. q at I_1 resp. P_1

$x_1,x_2,y_1,y_2 \geq 0$

executing rendezvous transitions (with $c \in Q$):

$y_1 = x_1 - c$
$y_2 = x_2 + c$

rendezvous transition is taken at least once:

$c \geq 1$

executing broadcast:

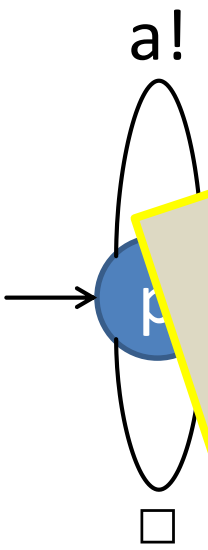
$x_1 = y_1 + y_2$
$x_2 = 0$

Linear Program by Example

$I_1 = I_2 = \{p\}$

$P_1 = P_2 = \{p,q\}$

variables $x_1,x_2,y_1,y_2 \in Q$ for the number of automata in state p resp. q at I_1 resp.



We can use LPs over Q because solutions are always scalable thanks to symmetry

$_1,y_2 \geq 0$

$x_1 - c$

$y_2 = x_2 + c$

…on is

…ast once:

$c \geq 1$

executing broadcast:

$x_1 = y_1 + y_2$

$x_2 = 0$

Linear Programs: A Complication

An assignment

$$y = x + c_1 \cdot t_1 + \ldots + c_n \cdot t_n$$

does not guarantee that there is a path from x to y, e.g.,

$$\begin{pmatrix} 3 \\ 0 \\ 0 \end{pmatrix} = \begin{pmatrix} 1 \\ 0 \\ 0 \end{pmatrix} + \begin{pmatrix} 1 \\ -1 \\ 1 \end{pmatrix} + \begin{pmatrix} 1 \\ 1 \\ -1 \end{pmatrix},$$

because coordinates can become **negative**.

Key Lemma:

If there is a path from $x \in Q^d$ to $y \in Q^d$, then there also is a path

$$x \xrightarrow{p_1} u \xrightarrow{q} v \xrightarrow{p_2} y$$

such that on q the vector components with a 0 do not change

and p_1, p_2 are of form $t_1^* \ldots t_d^*$ for some transitions $t_1, \ldots, t_d$.

Linear Programs: A Complication

An assignment

$$y = x + c_1 \cdot t_1 + \ldots + c_n \cdot t_n$$

does not guarantee that there is a path from x to y, e.g.,

$$\begin{pmatrix} 3 \\ 0 \\ 0 \end{pmatrix} = \begin{pmatrix} 1 \\ 0 \\ 0 \end{pmatrix} + \begin{pmatrix} 1 \\ -1 \\ 1 \end{pmatrix} + \begin{pmatrix} 1 \\ 1 \\ -1 \end{pmatrix},$$

because coordinates can become **negative**.

Key Lemma:

If there is a path from [illegible] a path

x [illegible] change

and p[illegible] some transitions $t_1, \ldots, t_d$.

→ Finite number of linear programs

Summary

- Decidability for liveness properties of timed networks
- New communication primitive „symmetric broadcast“
- New proof techniques: hopefully are useful in similar settings