Timed Automata and Time(d) Petri Nets

Serge Haddad

LSV
ENS Cachan & CNRS & INRIA
haddad@lsv.ens-cachan.fr

DISC'11, June 10th 2011

Outline

Timed Automata: Syntax

A timed automaton (TA) is:

- A finite automaton
- enlarged with a set of *clocks* (X) which evolve synchronously and continuously.

States (*locations*) of an automaton include *invariants* which restrict the way time may elapse in a location and whose syntax is:

$$\bigwedge_{x \in X} x \sim c$$

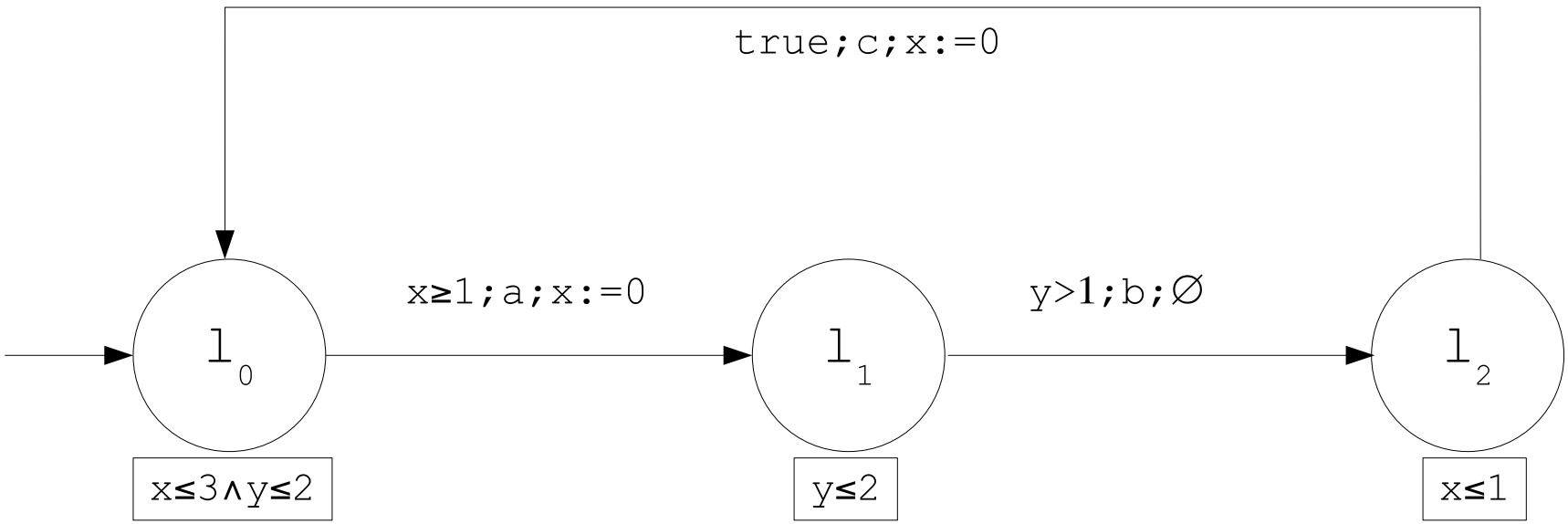
where $\sim \in \{<, \leq\}$ and c is some (possibly infinite) constant

Transitions of an automaton include *clock resets* and *guards* which restrict the temporal occurrence of a transition and whose syntax is:

$$\bigwedge_{x \in X} x \sim c$$

where $\sim \in \{>, <, \geq, \leq\}$ and c is some (possibly infinite) constant

Timed Automata: an Example



Timed Automata: Semantic

A configuration of timed automaton is given by:

- a location (l),
- a value $v(x)$ per clock satisfying the location invariant ($v \models Inv(l)$).

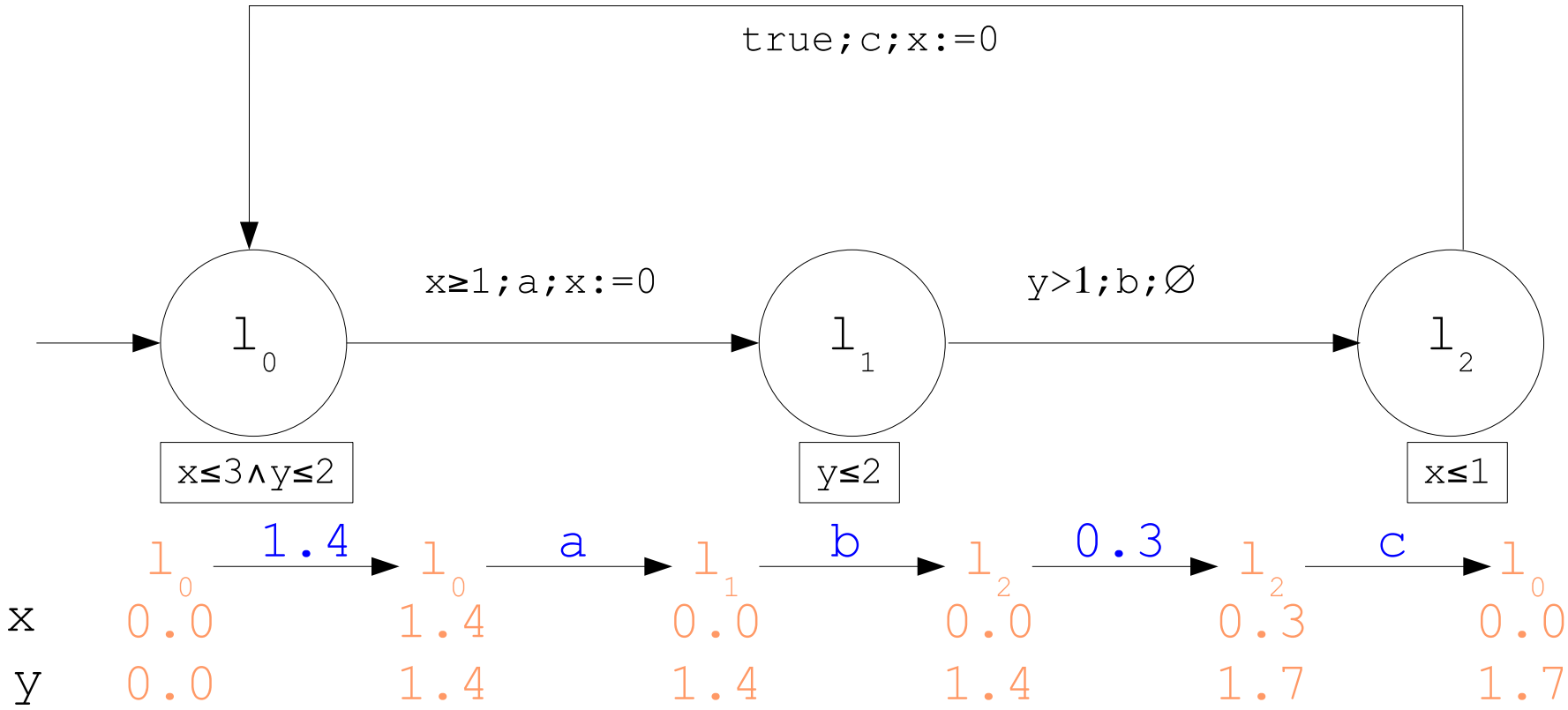
A configuration may evolve by a delay d such that the invariant is still satisfied:

$$v + d \models Inv(l) \text{ with } (v + d)(x) = v(x) + d$$

A configuration may change by a transition (l, g, a, R, l') to configuration (l', v') iff:

- $v \models g$
- If $x \in R$ then $v'(x) = 0$ else $v'(x) = v(x)$
- $v' \models Inv(l')$

Example Continued



What happens next?

Invariants in Timed Automata

Invariants may be deleted in timed automata:

- by adding the conjunct $Inv(l)$ to every guard of an edge e outgoing from l
- by adding the conjunct $\bigwedge_{x \in X \setminus R(e)} x \sim c$ to every guard of an edge e incoming to l with $Inv(l) = \bigwedge_{x \in X} x \sim c$

Nevertheless, this transformation

- is valid w.r.t. location reachability and linear temporal logics
- is invalid w.r.t. bisimulation and branching temporal logics

Extensions of TA

More elaborate guards and updates

- Comparing clocks: $x - x' \sim c$
- Updates rather than resets: $x := c$ or let x in $[a, b]$
- Adding finite counters in guards and updates

Composition of TA

A network of timed automata (NTA) is:

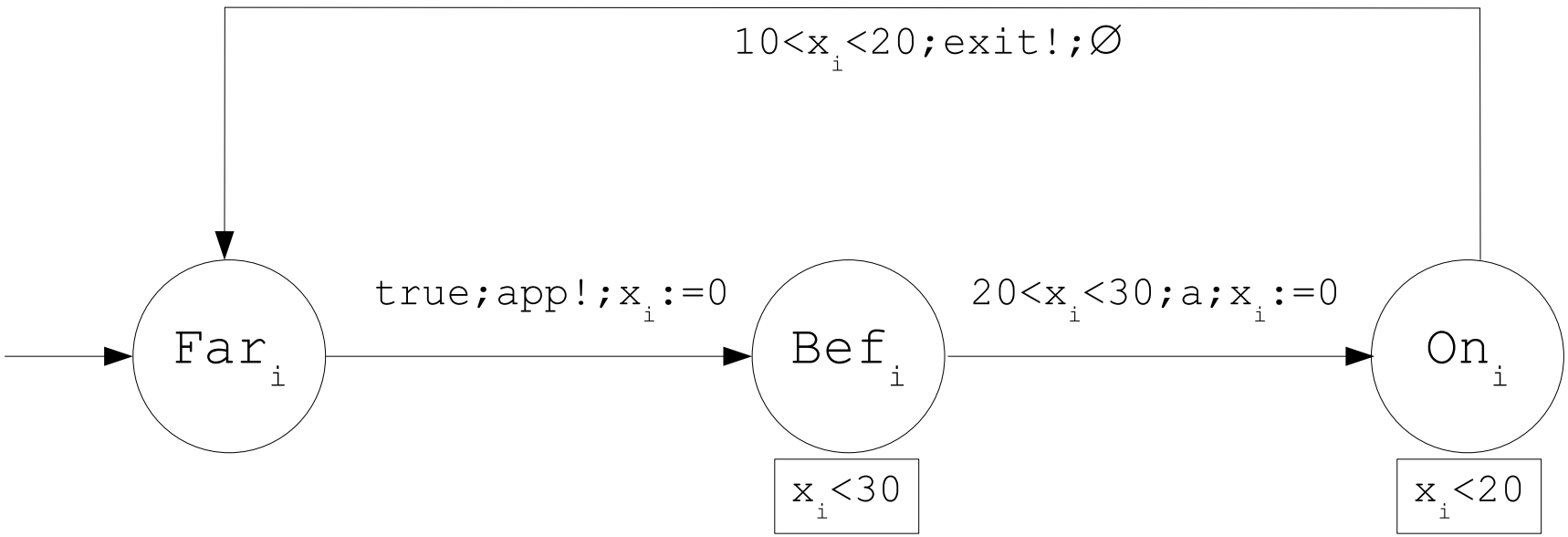
- a set of timed automata;
- a (partial) synchronization function from *vectors* of local actions to global actions.

A network of timed automata may always be transformed into a timed automaton equivalent w.r.t. bisimulation and reachability.

But this translation yields an exponential blow up.

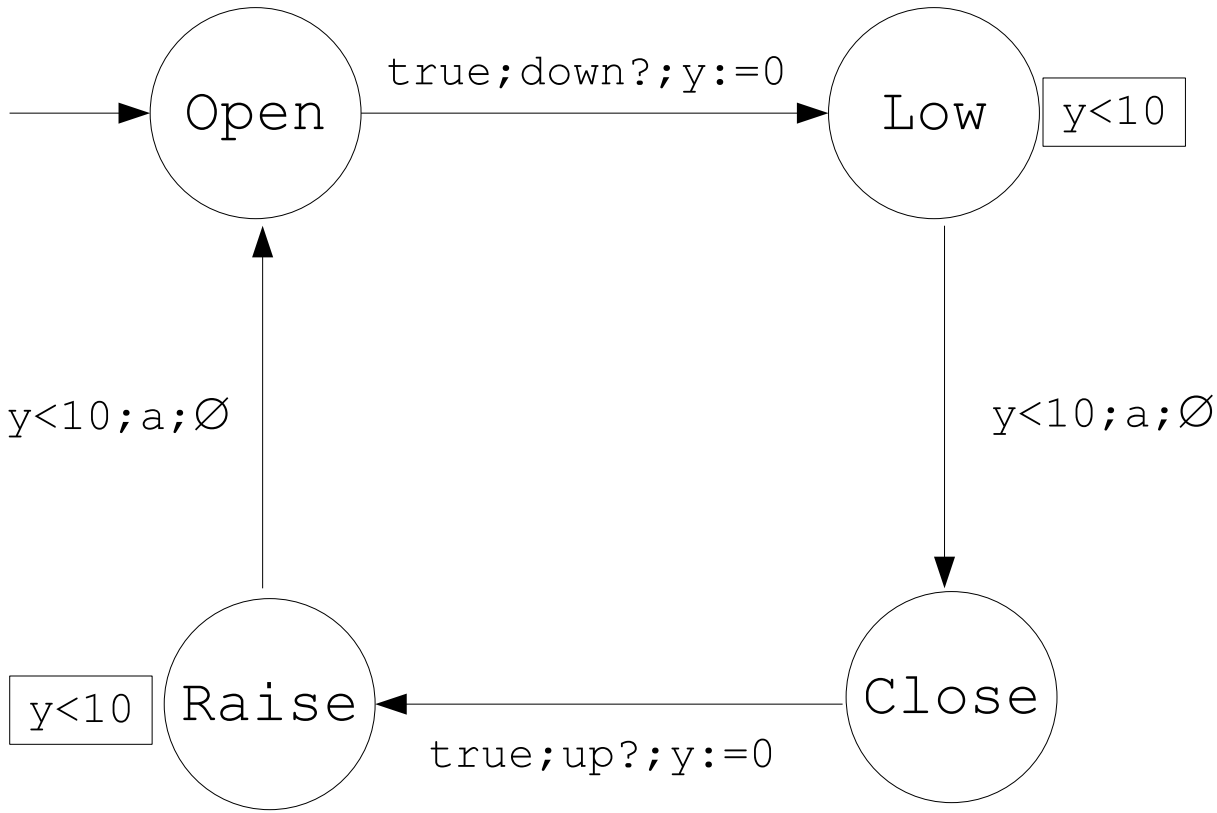
NTA: The Train Crossing Example

The trains $i = 1, 2, \ldots$



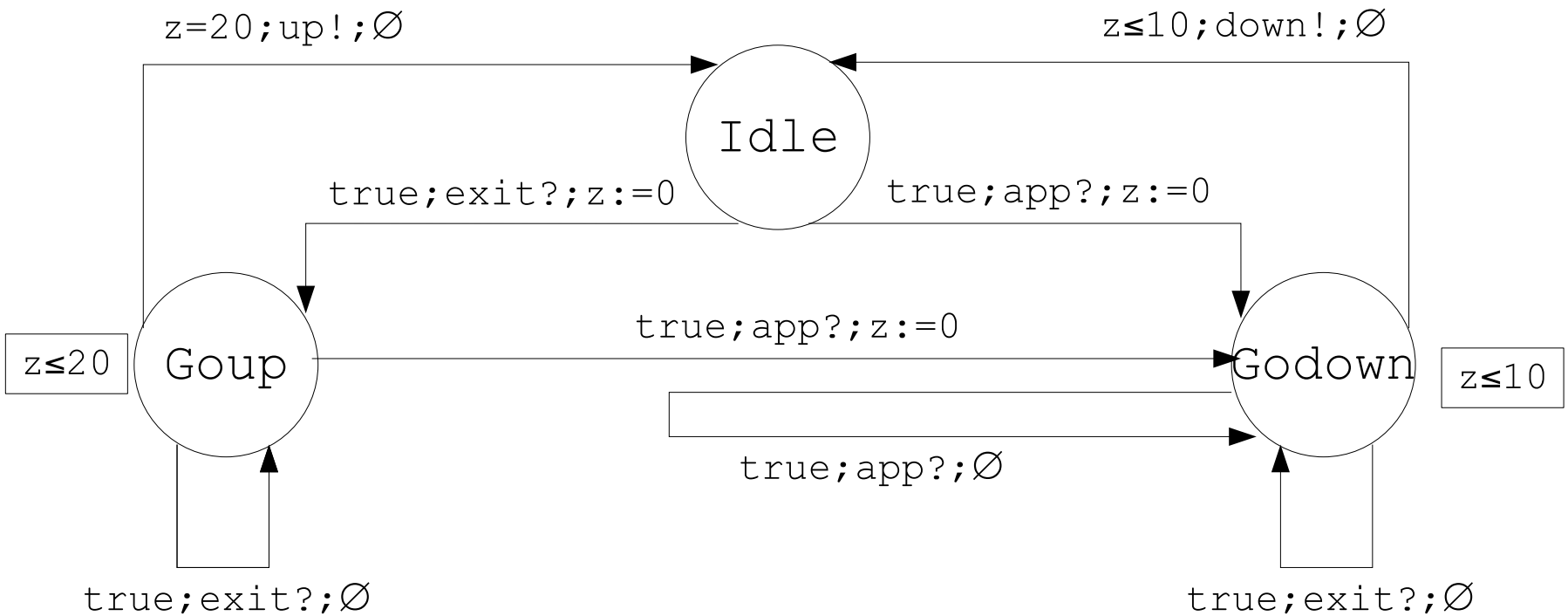
NTA: The Train Crossing Example

The gate



NTA: The Train Crossing Example

The controller



NTA: The Train Crossing Example

The synchronization function

$Train_1$	$Train_2$	Gate	Controller	
app!			app?	app
	app!		app?	app
exit!			exit?	exit
	exit!		exit?	exit
		down?	down!	down
		up?	up!	up
a				a
	a			a
		a		a

Outline

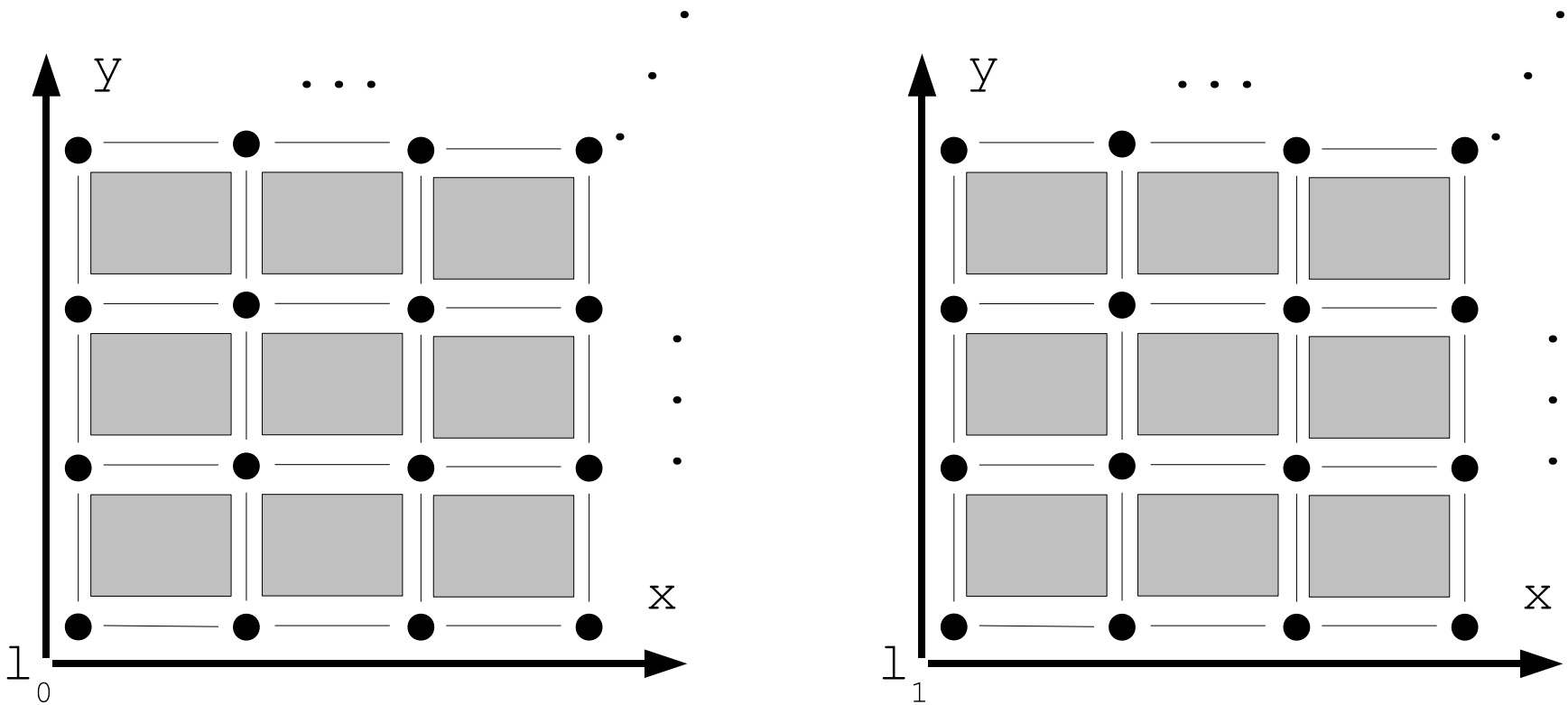
Reachability Analysis: the Key Idea

The number of (reachable) configurations is infinite (and even uncountable). So one wants to partition configurations into *regions* such that:

1. Two configurations in a region allow the same transitions and the new configurations belong to the same region.
2. If a configuration in a region letting time elapse reaches a new region every other configuration may reach the same region by time elapsing.
3. There is a finite representation of a region such that the discrete and time successors of the region are computable.
4. The number of regions is finite.

A First Partition (two clocks and two locations)

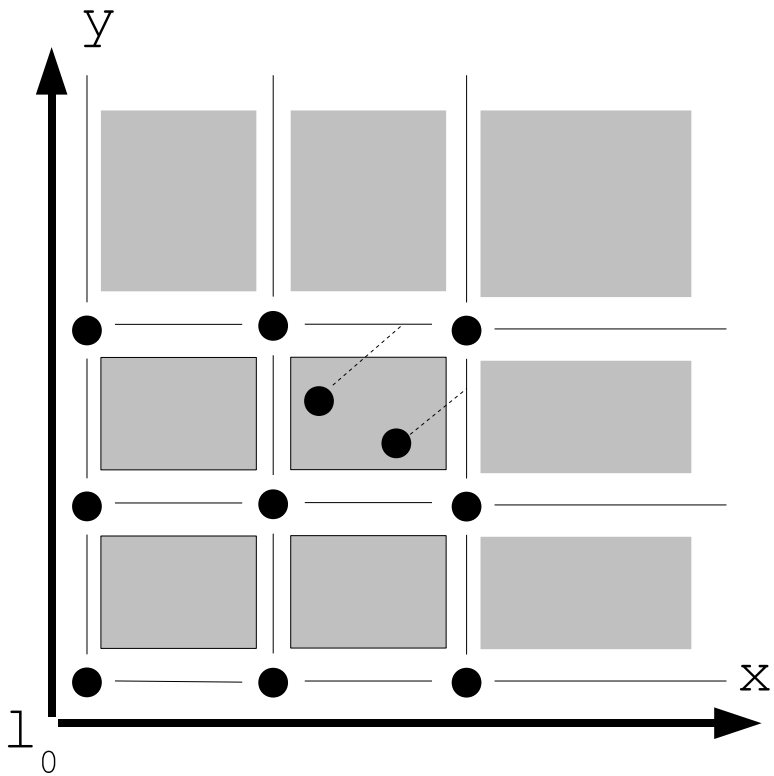
Guards and invariants check integer values

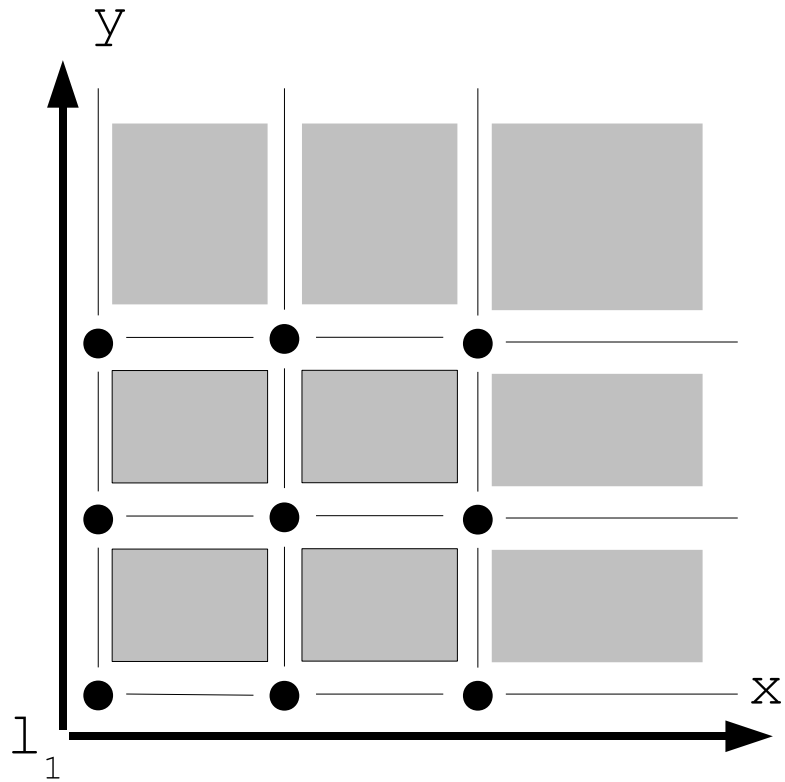


Why this partition is not appropriate?

A Second Partition (two clocks and two locations)

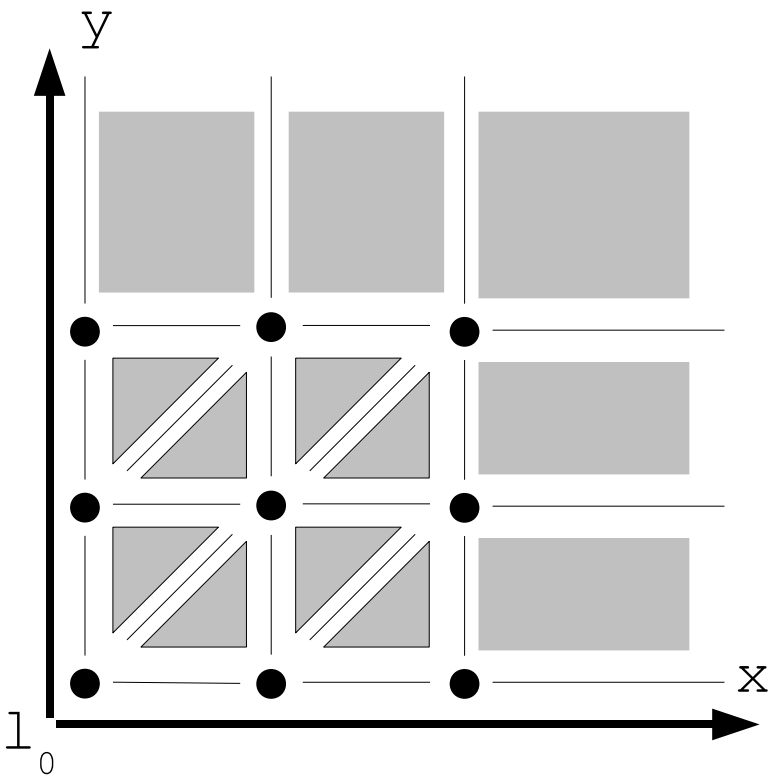
The exact value of a clock is irrelevant when it is beyond the maximal constant of the TA (here 2)

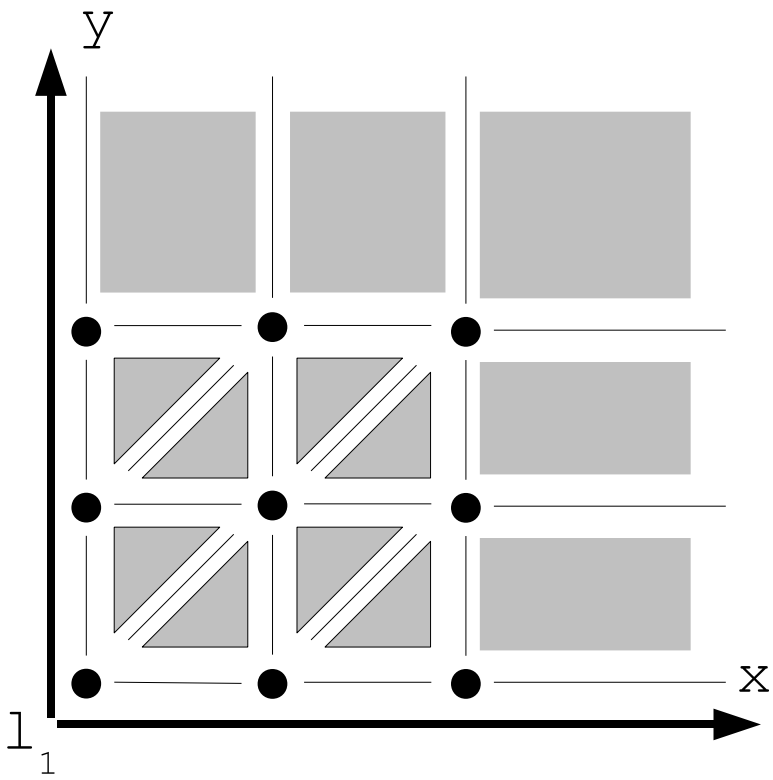




Why this partition is not appropriate?

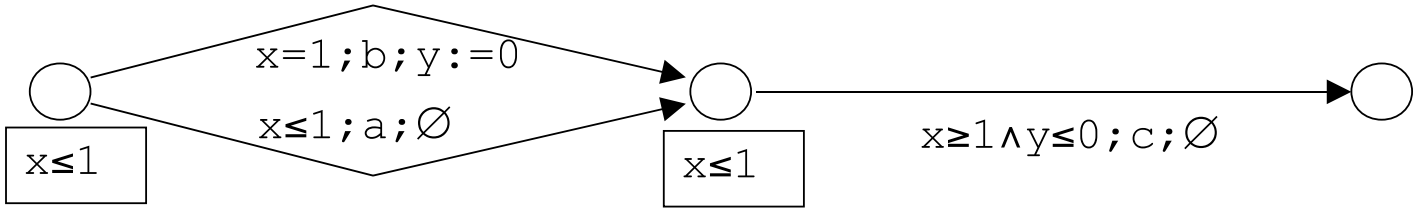
A Third Partition (two clocks and two locations)

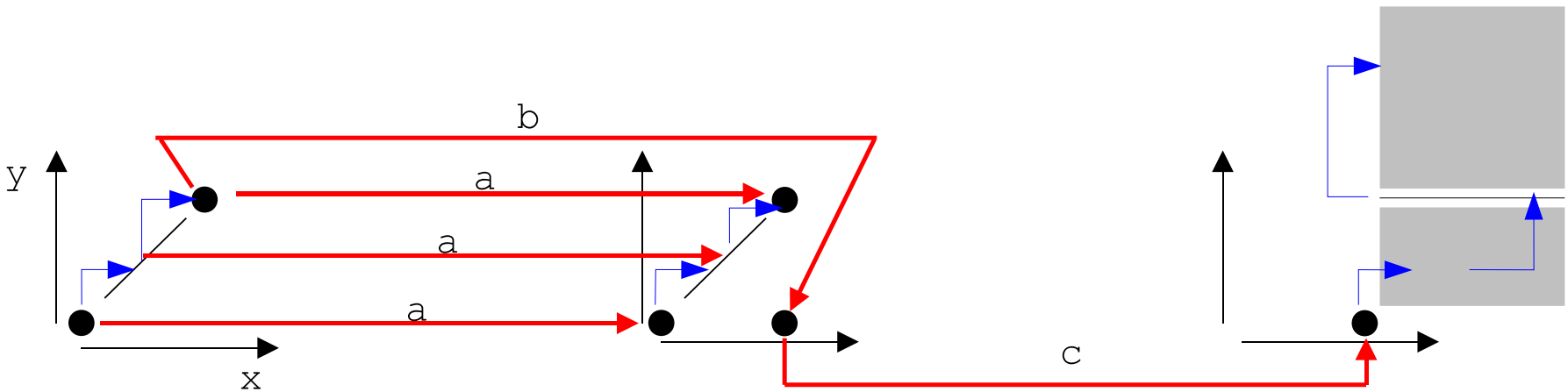




Check that this partition is appropriate

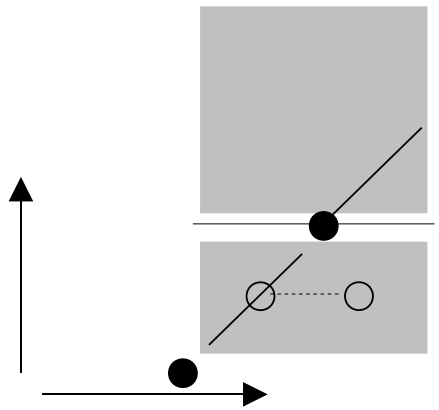
The Region Graph: Illustration





About Reachability in the Region Graph

Warning: when a region is "reachable", it does not mean that every configuration of the region is reachable.



However it means that there is another reachable configuration of the region which differs only on the values of irrelevant clocks.

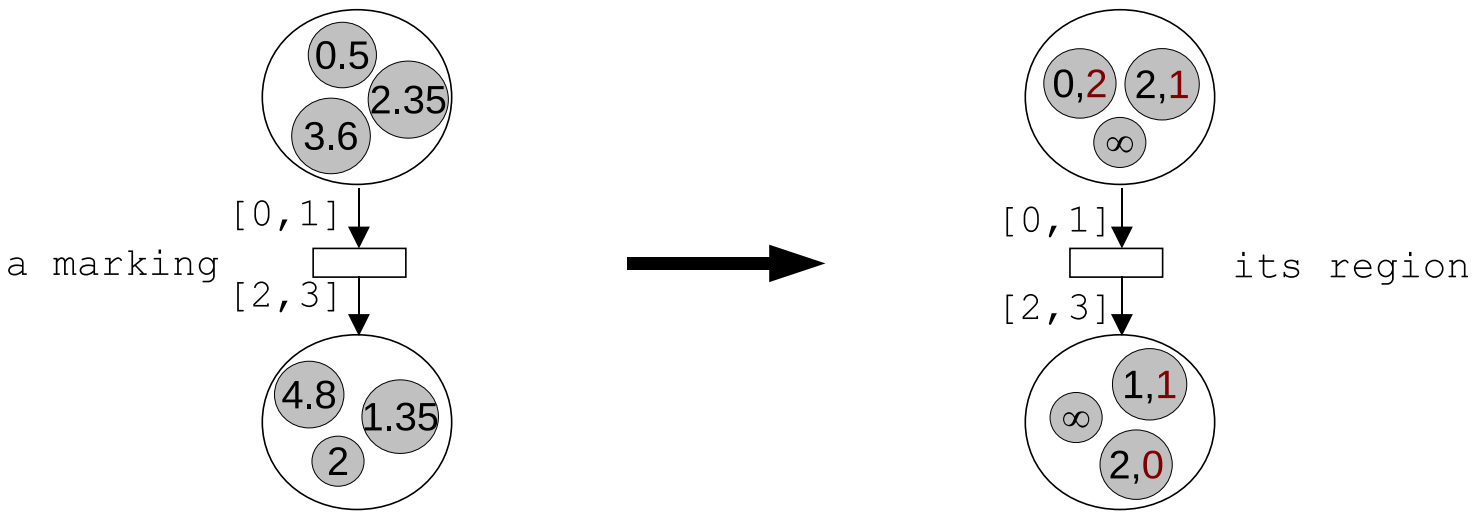
Hence, in order to check the reachability of a configuration, it is enough to increase the maximal constant.

Example: reachability of $(l_0, 1.7, 2.3)$ requires to choose at least 3 as maximal constant.

Regions for TdPNs

In TdPNs, the tokens are clocks. So a *region of a TdPN* is:

- $n+1$ different and ordered fractional parts of the token ages with the null fractional part.
- The distribution of tokens on places and their integer part (when less or equal than the maximal constant) for every fractional part.
- The distribution of tokens with age greater than the maximal constant on places.



The corresponding region graph is infinite ... but it is a *well-structured transition system* (out of the scope of the talk) and thus (for instance) coverability can be decided by a symbolic backward exploration.

Regions and Zones

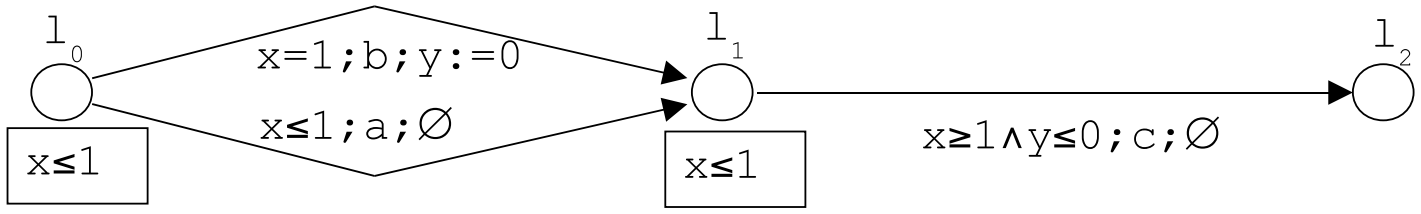
In practice, the building of the region graph is relatively inefficient since the partition is too finer w.r.t. a particular TA.

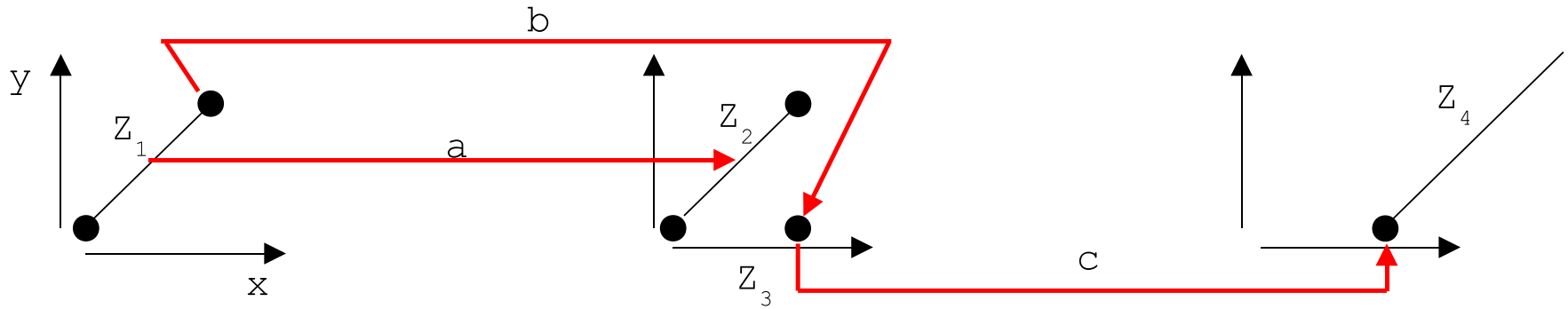
An alternative way is to work with zones.

A zone is defined by an *extended* DBM

- whose variables are clocks
- enlarged with a special clock x_0 representing the value 0.
- where the constraints are either $x_j - x_i \leq c_{ij}$ or $x_j - x_i < c_{ij}$.

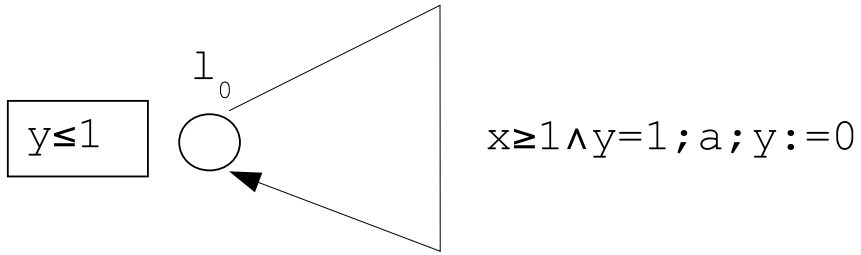
The Zone Graph: Illustration

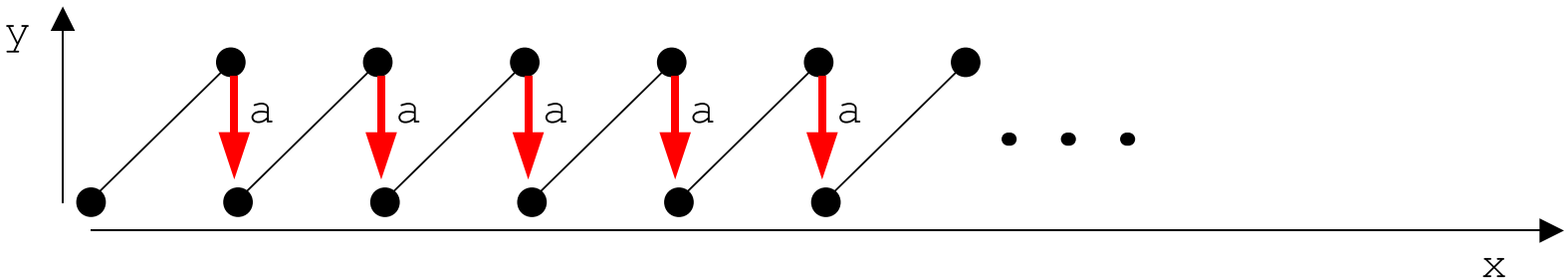




$$Z_0 = (l_0, \{x = y = 0\})$$
$$Z_1 = (l_0, Future(Z_0) \cap \{x \leq 1\}) = (l_0, \{0 \leq x = y \leq 1\})$$
$$Z_2 = (l_1, Future(Z_1 \cap \{x \leq 1\}) \cap \{x \leq 1\}) = (l_1, \{0 \leq x = y \leq 1\})$$
$$Z_3 = (l_1, Future(Reset(Z_1 \cap \{x = 1\}, \{y\})) \cap \{x \leq 1\}) = (l_1, \{x = 1 \wedge y = 0\})$$
$$Z_4 = (l_2, Future(Z_1 \cap \{x \geq 1 \wedge y \leq 0\})) = (l_2, \{0 \leq y = x - 1\})$$

An Infinite Zone Graph





The Zone Graph: Extrapolation

During the building of a zone examine every constraint $x - y \sim c$ with $\sim \in \{<, \leq\}$ *(K is the maximal constant)*

- and substitute to it the constraint $x - y \sim -K$ when $c < -K$
- Delete the constraint when $c > K$

Extrapolation enlarges the zone with *equivalent* configurations and ensures termination.

$Z_1 = (l_0, \{0 \leq y = x \leq 1\})$ no extrapolation

$Z_2 = (l_0, \{0 \leq y = x - 1 \leq 1\})$ no extrapolation

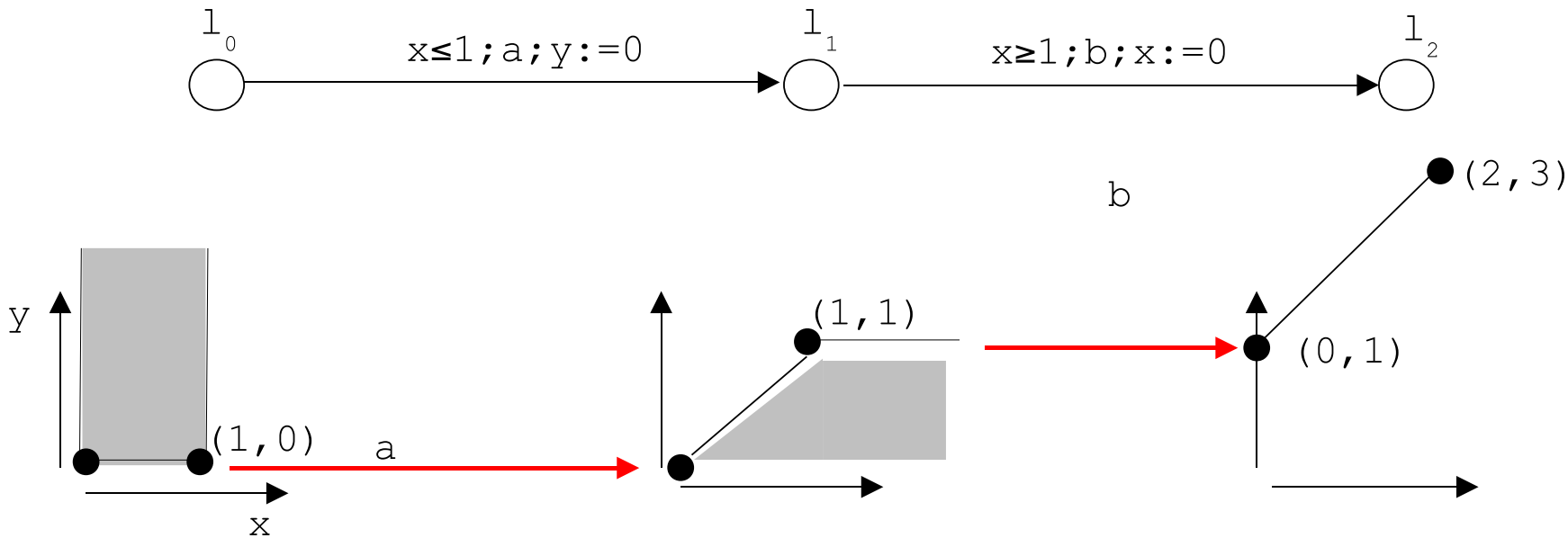
Before extrapolation $Z_3 = (l_0, \{0 \leq y = x - 2 \leq 1\})$
After extrapolation $Z_3 = (l_0, \{0 \leq y \leq 1 \wedge y \leq x - 1\})$

Before extrapolation $Z_4 = (l_0, \{0 \leq y \leq 1 \wedge y \leq x - 2\})$
After extrapolation $Z_4 = Z_3$

The Zone Graph: Backward Exploration

The "backward" zone graph always terminates since the coefficients of the zones remain bounded.

Example: Is $(l_2, \{x = 2 \wedge y = 3\})$ reachable in the following TA?



Outline

Expressiveness: A Language Point of View

Observation: There are non regular untimed languages of (unbounded) nets while all TA untimed languages are regular.

From TA to nets

- Timed Automata can be "simulated" by bounded TPNs.
- Timed Automata can be "simulated" by bounded TdPNs *with acceptance conditions*.

But these simulations are not valid w.r.t. bisimulation.

From bounded nets to TA

- Bounded TPNs can be simulated by Timed Automata.
- Bounded TdPNs can be simulated by Timed Automata.

And these simulations are valid w.r.t. bisimulation.

From TA to TPN

The structural part

- There is a place per location.
- There is an *untimed* net transition per automata transition (i.e. with interval $[0, \infty[$) which takes as input the source location.

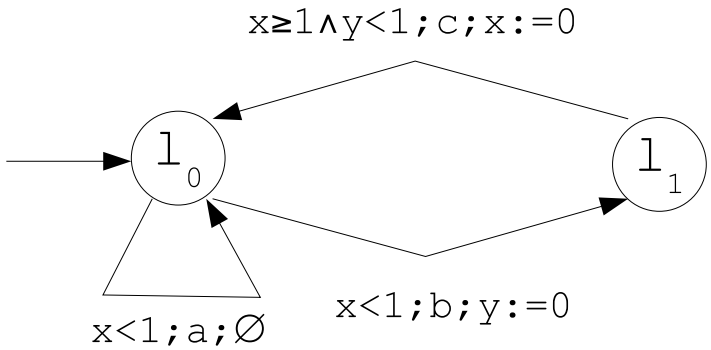
Time guards

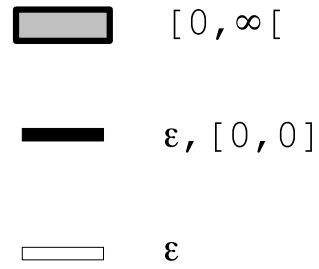
- There is a place $T_{x \sim c}$ per conjunct of a guard $x \sim c$ which is an input for the transitions where it occurs.
- The marking of such a place is ruled by an "independent" timed subnet.

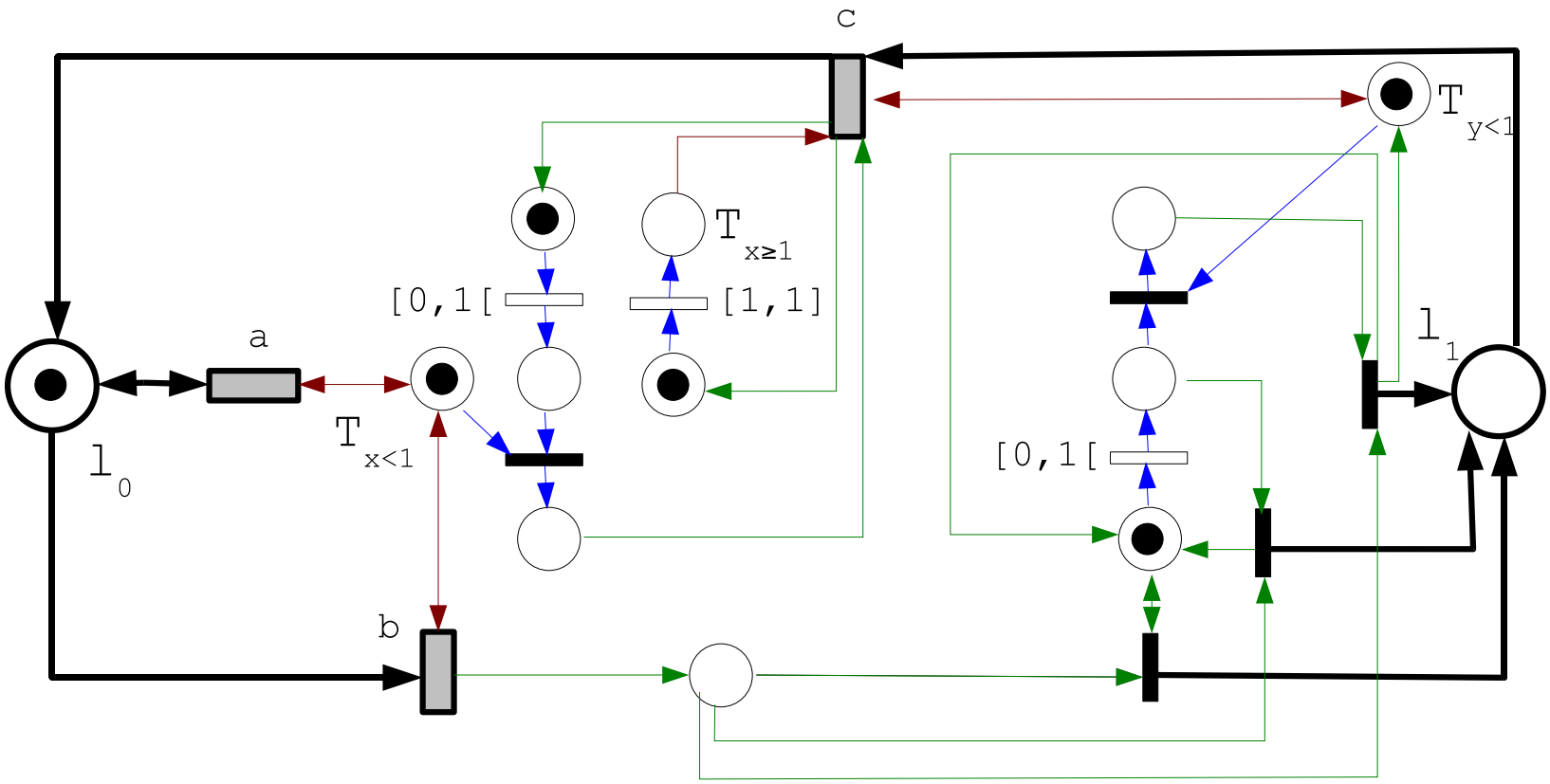
Clock Resets

- Clock resets consists (in zero time) to reinitialize all the timed subnets related to the clocks to be reset whatever the state of these subnets.
- The clock resets take place after the automata transition and before producing the token in the destination location.

From TA to TPN: an Example







From TA to TdPN

The structural part

- There is a place per location.
- There is a net transition per automata transition which takes as *untimed* input (i.e. with interval $[0, \infty[$) the source location.

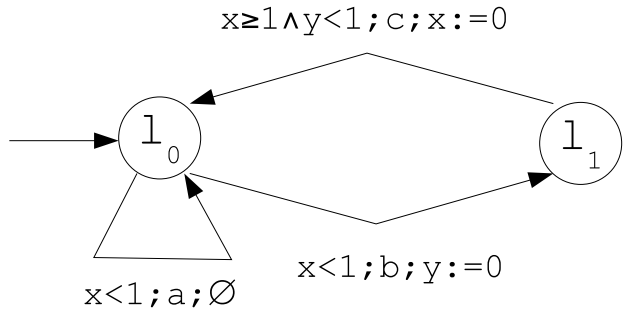
Time guards

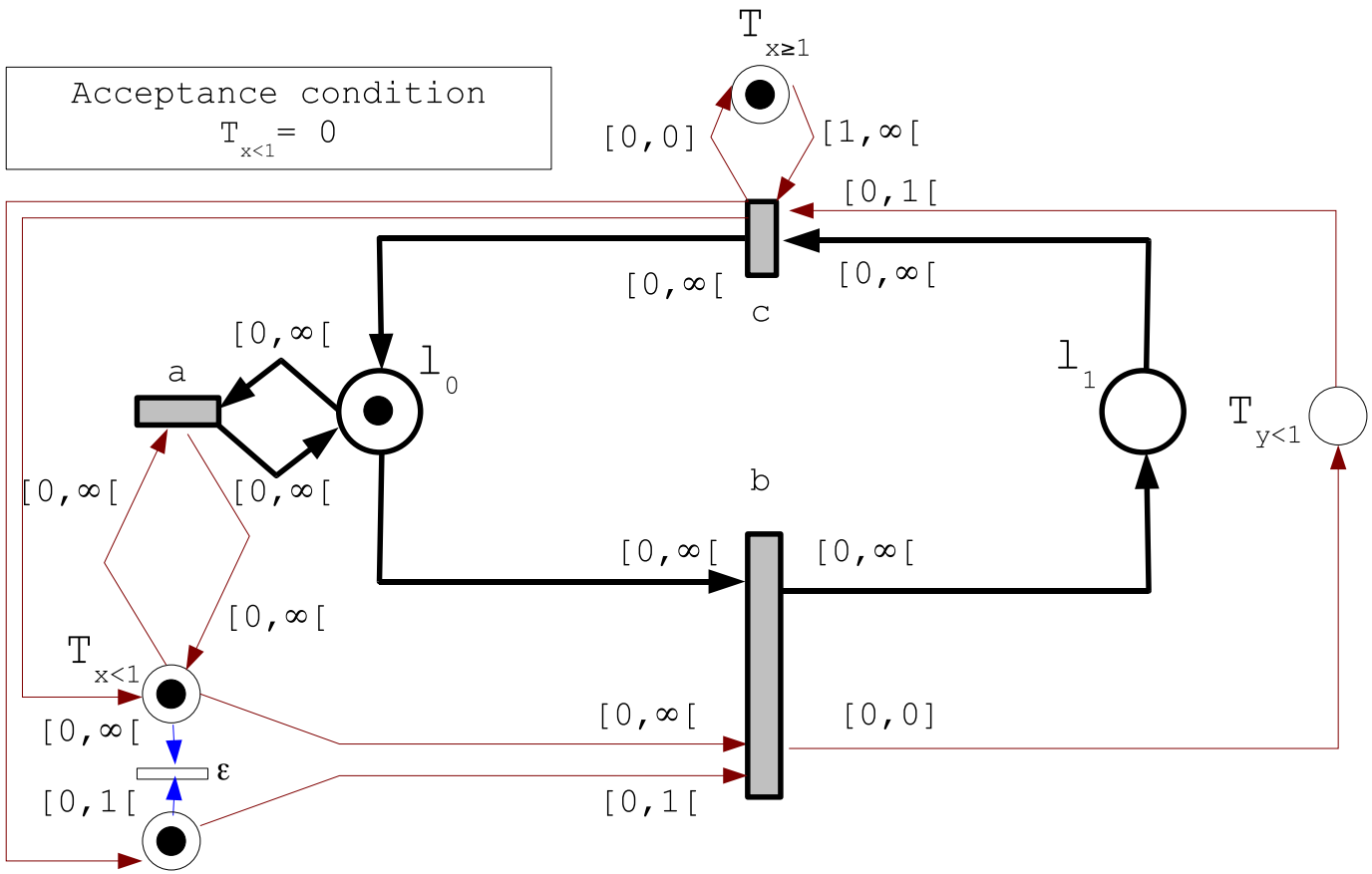
- There is a place $T_{x \sim c}$ per conjunct of a guard $x \sim c$ which is an untimed input for the transitions where it occurs.
- The control that a token of such a place has been checked at appropriate time is performed by an "independent" timed subnet *with an acceptance condition*.

Clock Resets

- Clock resets consists (in zero time) to reinitialize all the timed subnets related to the clocks to be reset whatever the state of these subnets.
- The clock resets take place after the automata transition and before producing the token in the destination location.

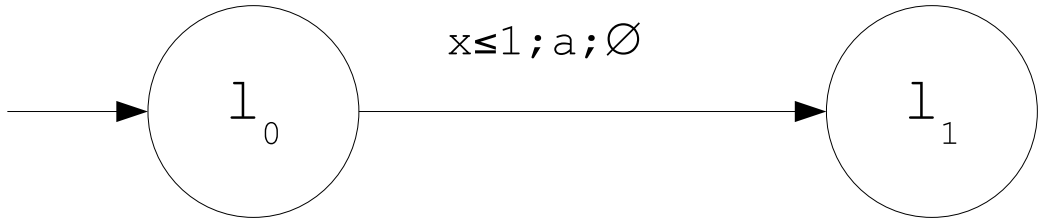
From TA to TdPN: an Example



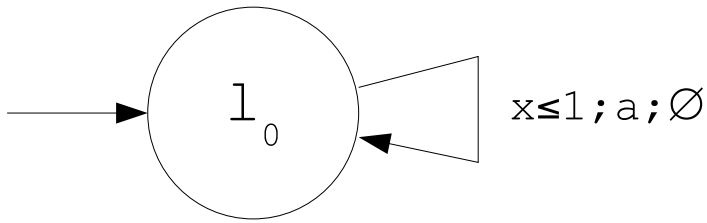


Limit of the simulations

A TA not bisimilar to any TPN



A TA whose infinite language is not the one of any TdPN



From Bounded TPN to TA

There is one clock x_t per transition t.

Build the reachability graph.

- The locations of the TA are the reachable markings.
- The transitions of the TA are the transitions of the reachability graph.

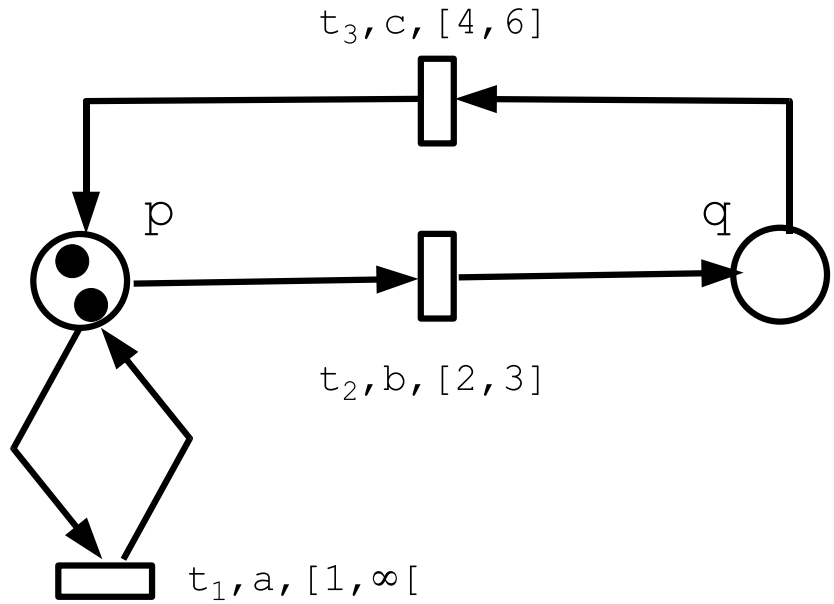
Define the invariants. Given T_m the set of transitions enabled at m
The invariant of m is: $\bigwedge_{t \in T_m} x_t \leq l(t)$.

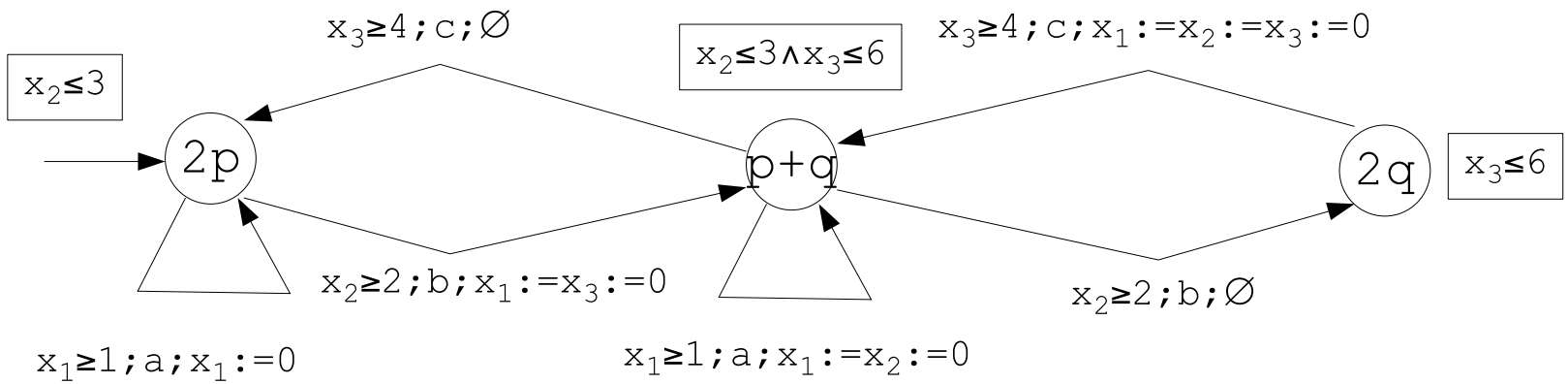
Define the guards and updates. Given a transition $m \xrightarrow{t} m'$,

- The guard is $x(t) \geq e(t)$
- The clocks to be reset are those associated with the newly enabled transitions.

Warning: There exists an alternative structural translation but it uses both networks of TA and finite counters.

From Bounded TPN to TA: an Example





From Bounded TdPN to TA

Transform the net such that all intervals of output arcs are $[0, 0]$.

Build the reachability graph with token identities p_i and instances of transitions

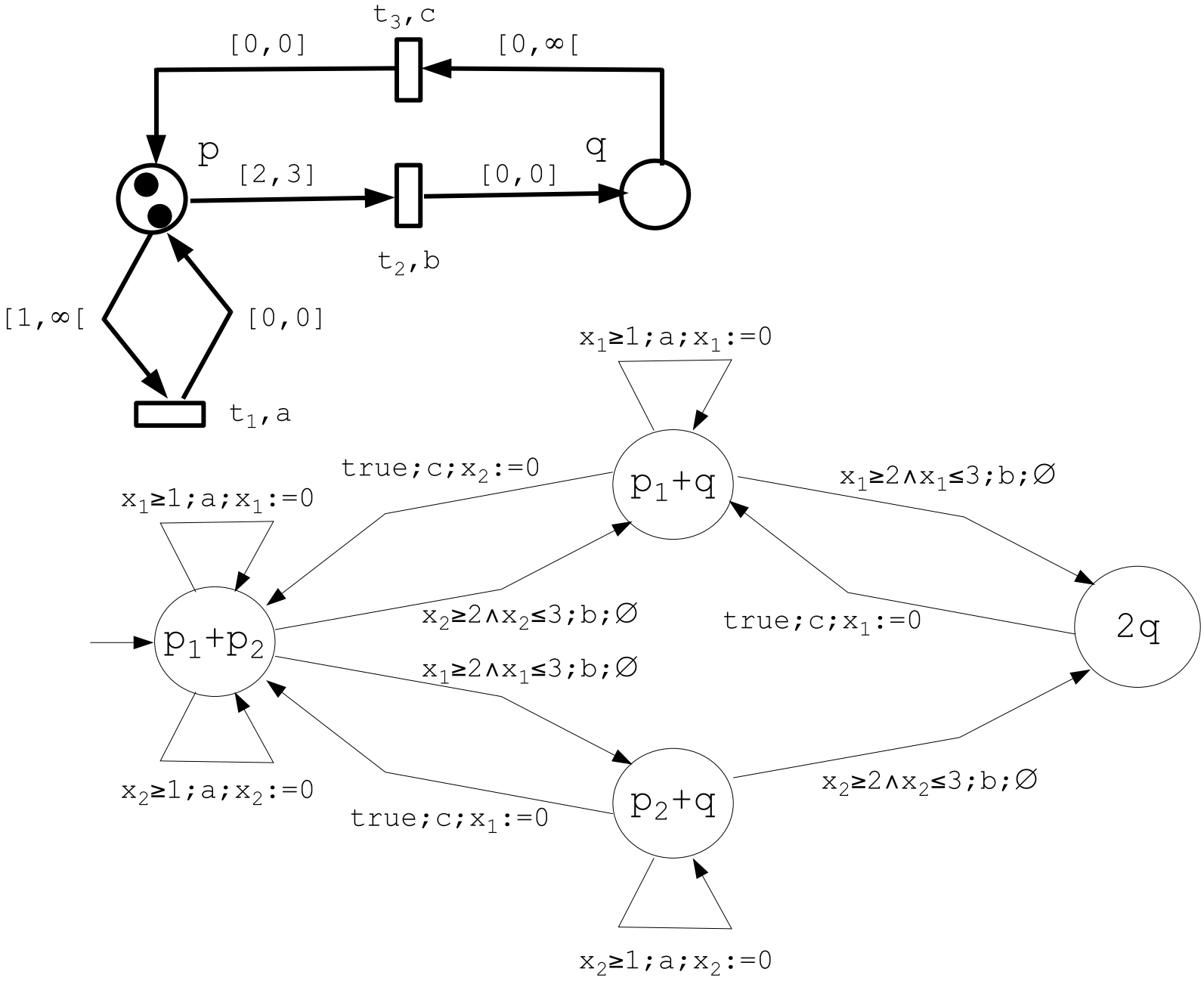
- The locations of the TA are the reachable markings.
- The transitions of the TA are the transitions of the reachability graph.

There is one clock x_{p_i} per token p_i in place p.

Define the guards and updates. Given a transition $m \xrightarrow{t} m'$,

- For every input arc (p, t) labelled by interval $[a, b]$ and consumed token p_i the guard is $a \leq x_{p_i} \leq b$
- The clocks to be reset are those associated with the tokens that are produced.

From Bounded TdPN to TA: an Example



Main References

On definition and analysis of TA R. Alur and D. Dill. A theory of timed automata. Theoretical Computer Science, 126(2):183-235, 1994

On relation between TA and TPN

- D. Lime, O. H. Roux. State class timed automaton of a time Petri net. In Proceedings of the 10th International Workshop on Petri Nets and Performance models (PNPM 2003) Urbana-Champaign, Illinois, USA IEEE Computer Society, p. 124-133, 2003
- F. Cassez, O. H. Roux. Structural Translation of Time Petri Nets into Timed Automata. In Michael Huth, editor, Workshop on Automated Verification of Critical Systems (AVoCS'04) London, UK, ENCS Elsevier, vol. 128, issue 6, 23 p. 145-160, 2004
- B. Bérard, F. Cassez, S. H., D. Lime, and O. H. Roux. Comparison of the expressiveness of timed automata and time Petri nets. FORMATS'05, vol. 3829 of LNCS, p. 211-225. Springer, 2005
- P. Bouyer, S. H., P-A. Reynier. Extended Timed Automata and Time Petri Nets. ACSD 2006 Sixth International Conference on Application of Concurrency to System Design, IEEE Computer Society, p. 91-100, 2006, Turku, Finland
- B. Bérard, F. Cassez, S. H., D. Lime and O.H. Roux. When are timed automata weakly timed bisimilar to time Petri nets? Theoretical Computer Science, 403(2-3): 202-220, 2008

On analysis of TdPN and relation between TA and TdPN P. Bouyer, S. H., P-A. Reynier, Timed Petri nets and timed automata: On the discriminating power of Zeno sequences, Information and Computation 206(1): 73-107, 2008