UNIVERSITAT POLITÈCNICA DE CATALUNYA
BARCELONATECH

EPSRC
Pioneering research and skills

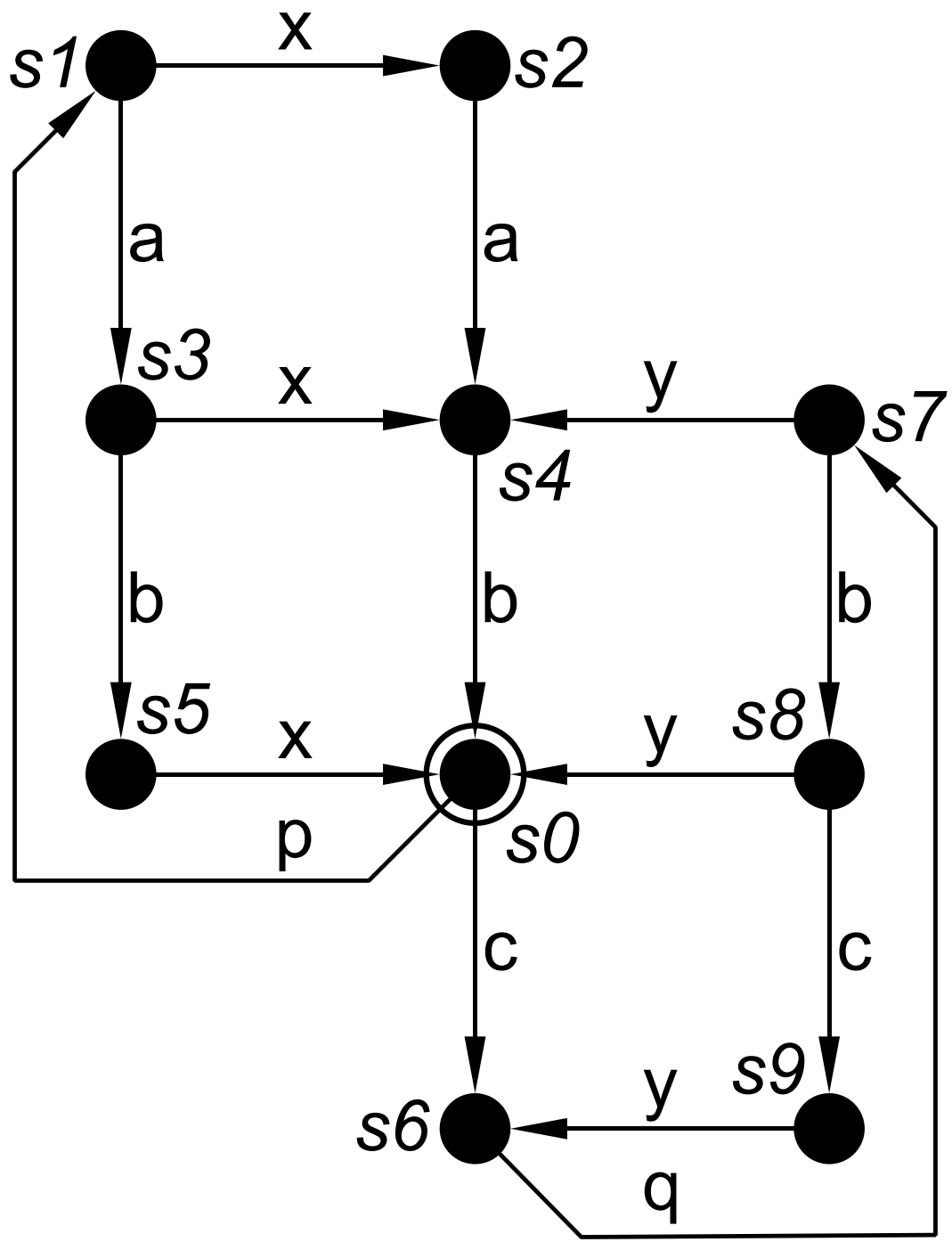
Newcastle University

Process Windows

Andrey Mokhov, Jordi Cortadella, Alessandro de Gennaro

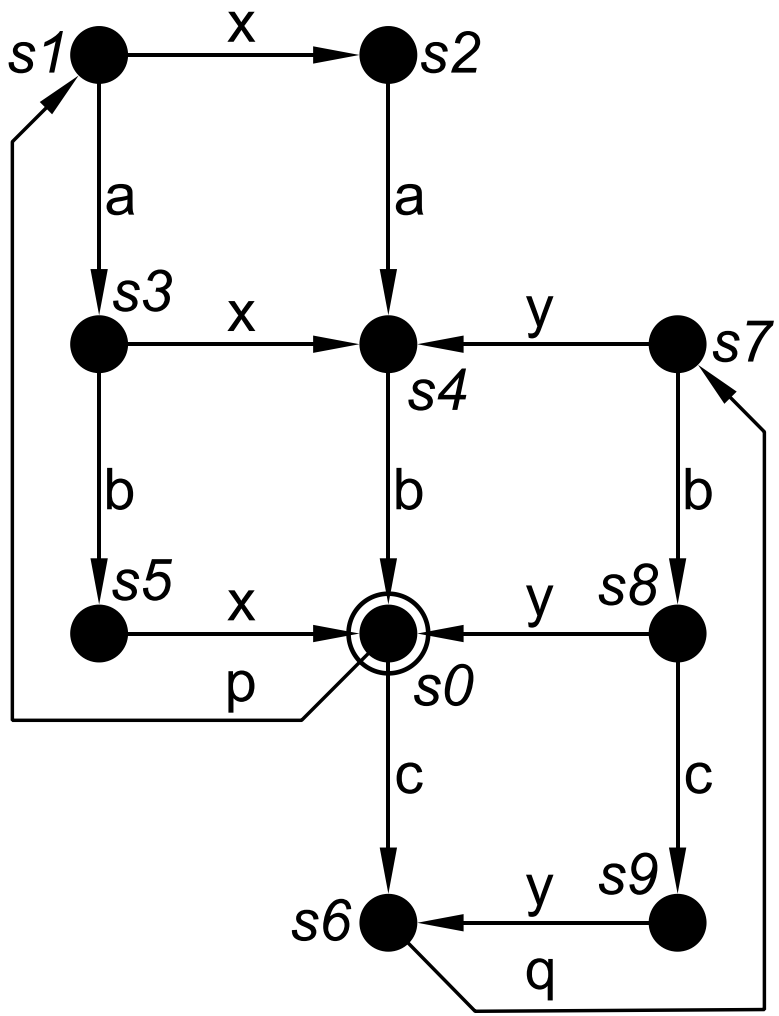
June 2017, Zaragoza

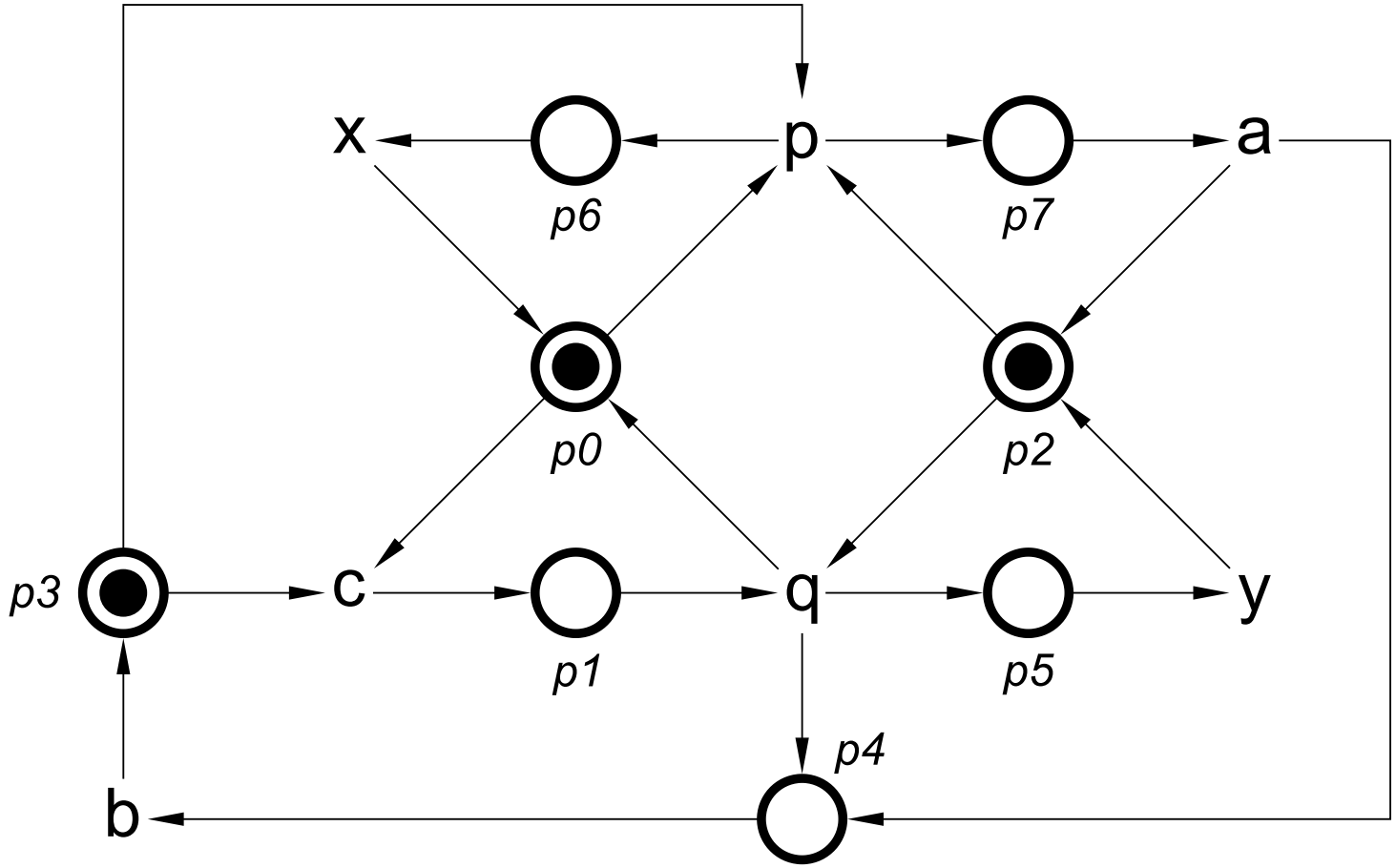
Part I:
Motivation and Main Idea



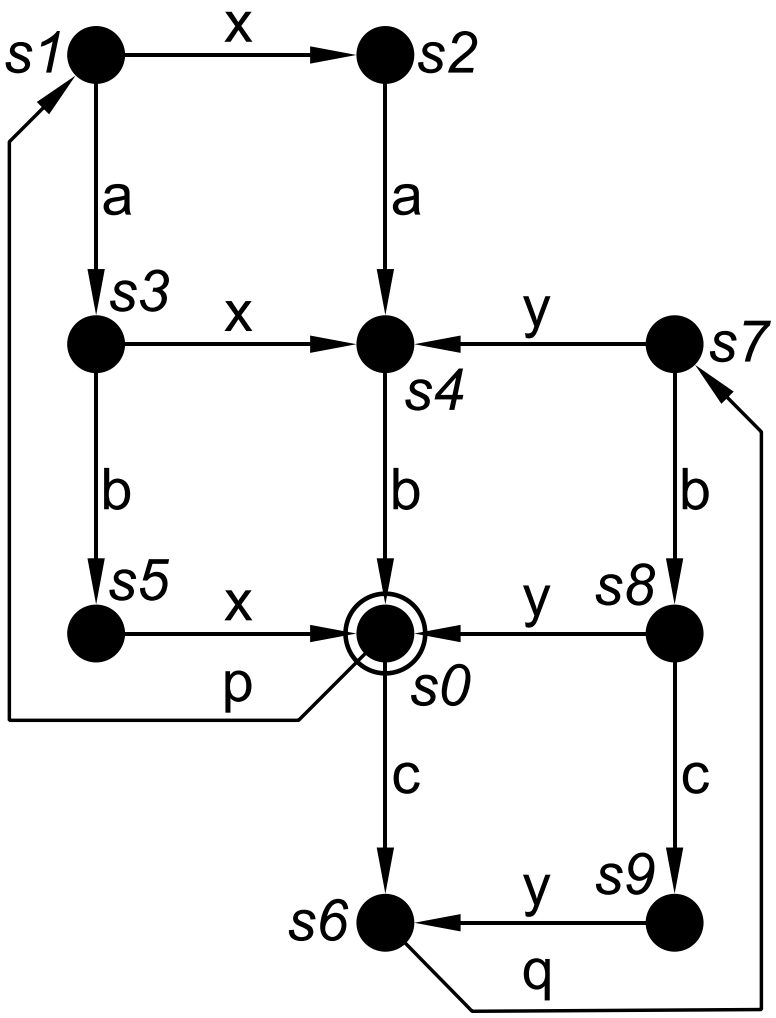
Once upon a time there was a transition system…

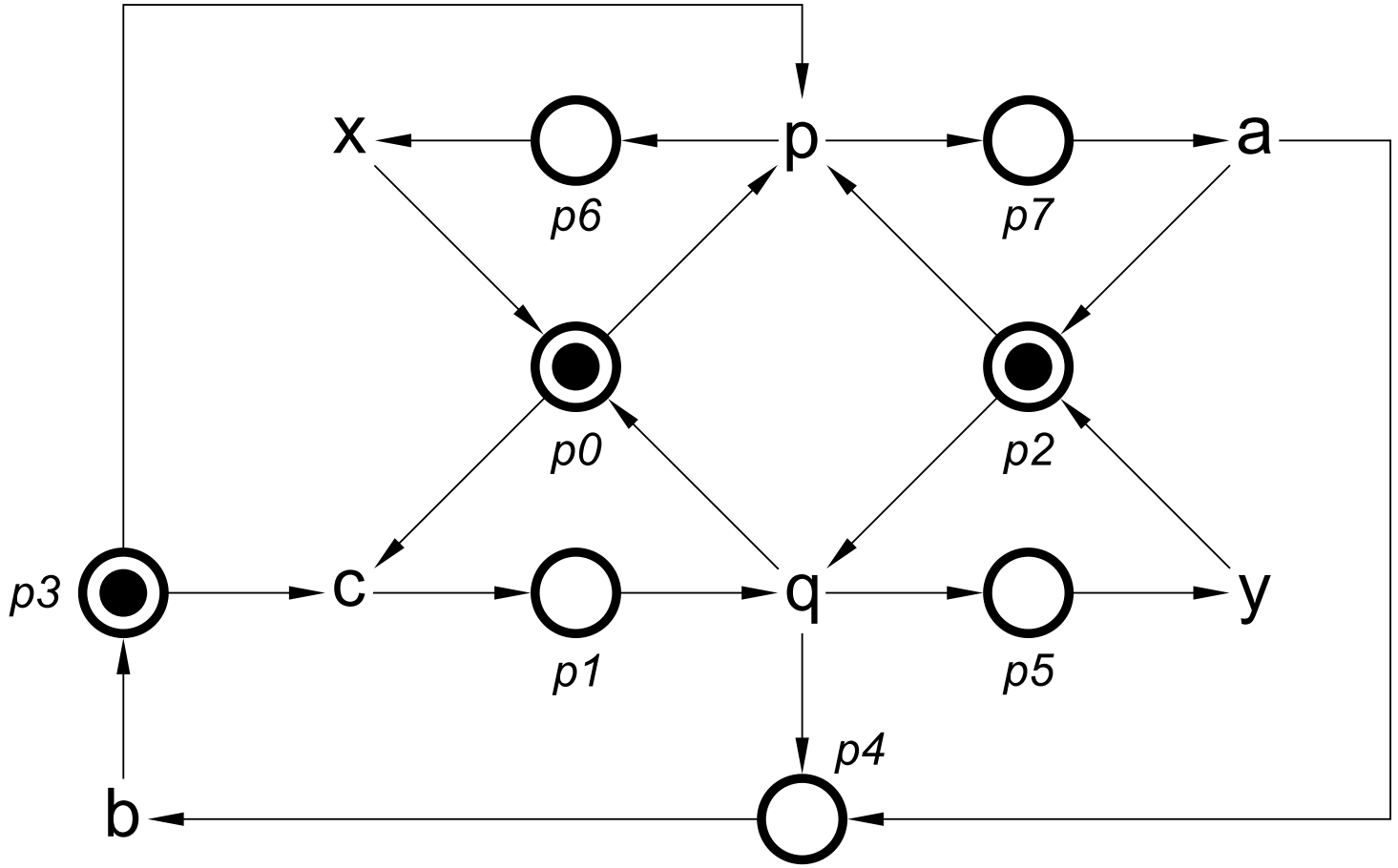
…it lacked persistency, but had a few nice diamonds.


s1
x
s2
a
a
s3
x
y
s7
s4
b
b
b
s5
x
y
s8
p
s0
c
c
y
s9
s6
q

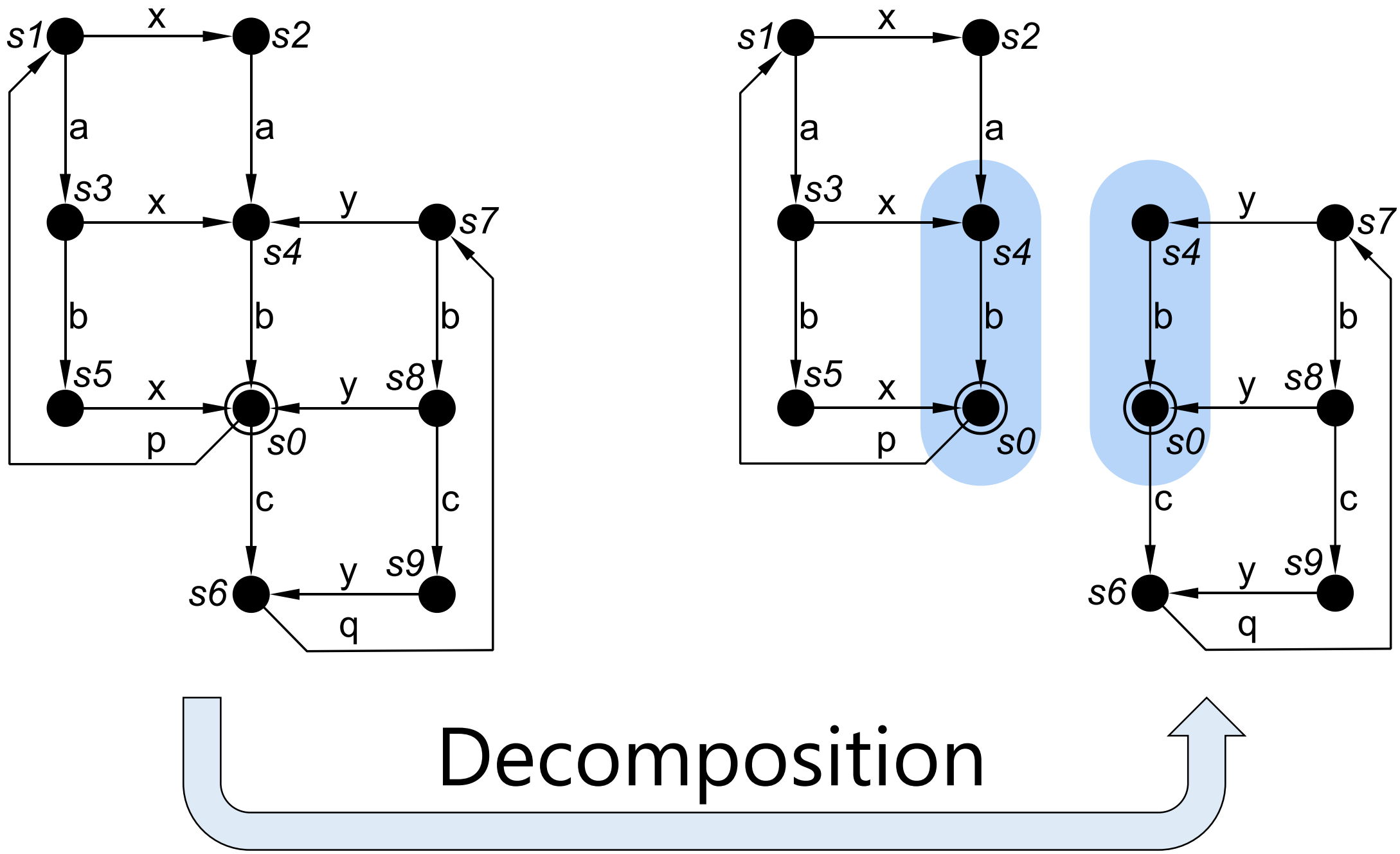

x
p
a
p6
p7
p0
p2
p3
c
q
y
p1
p5
p4
b

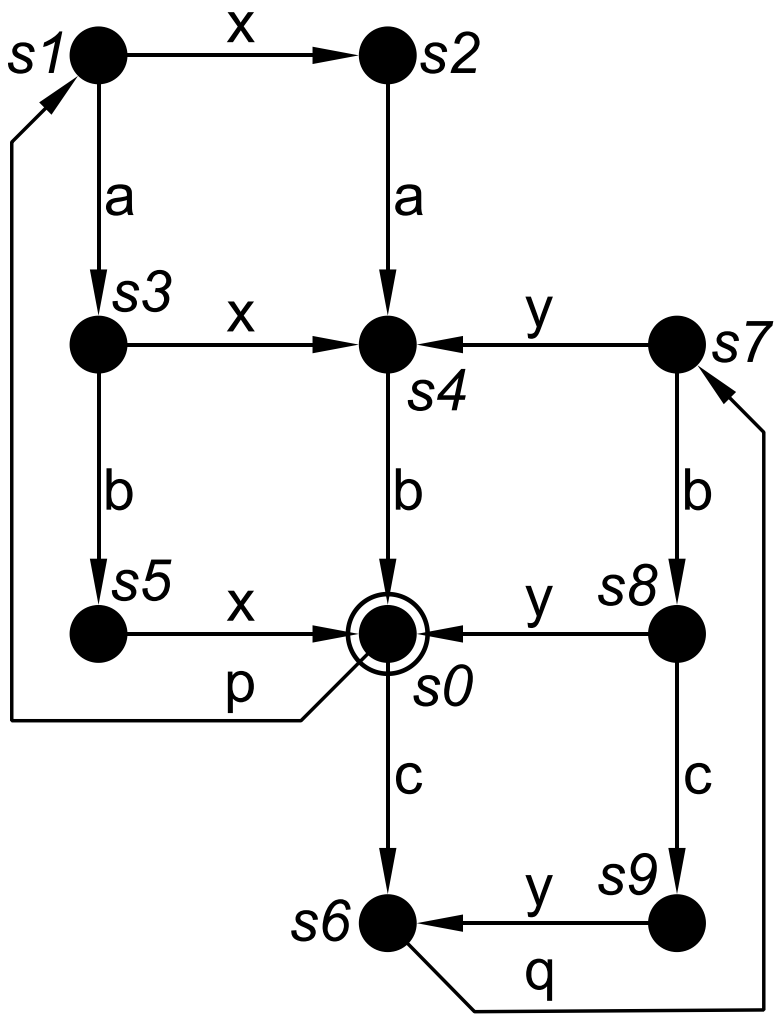
Synthesis

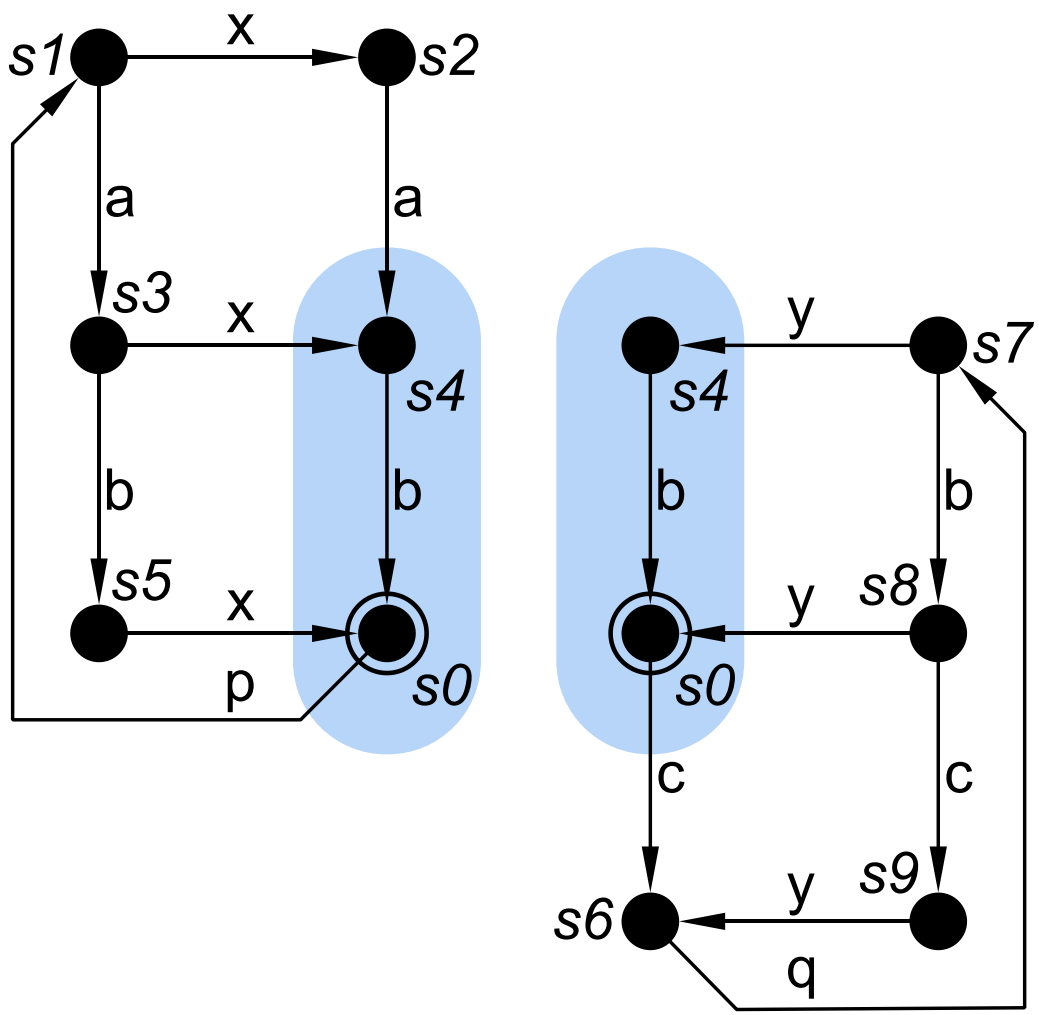




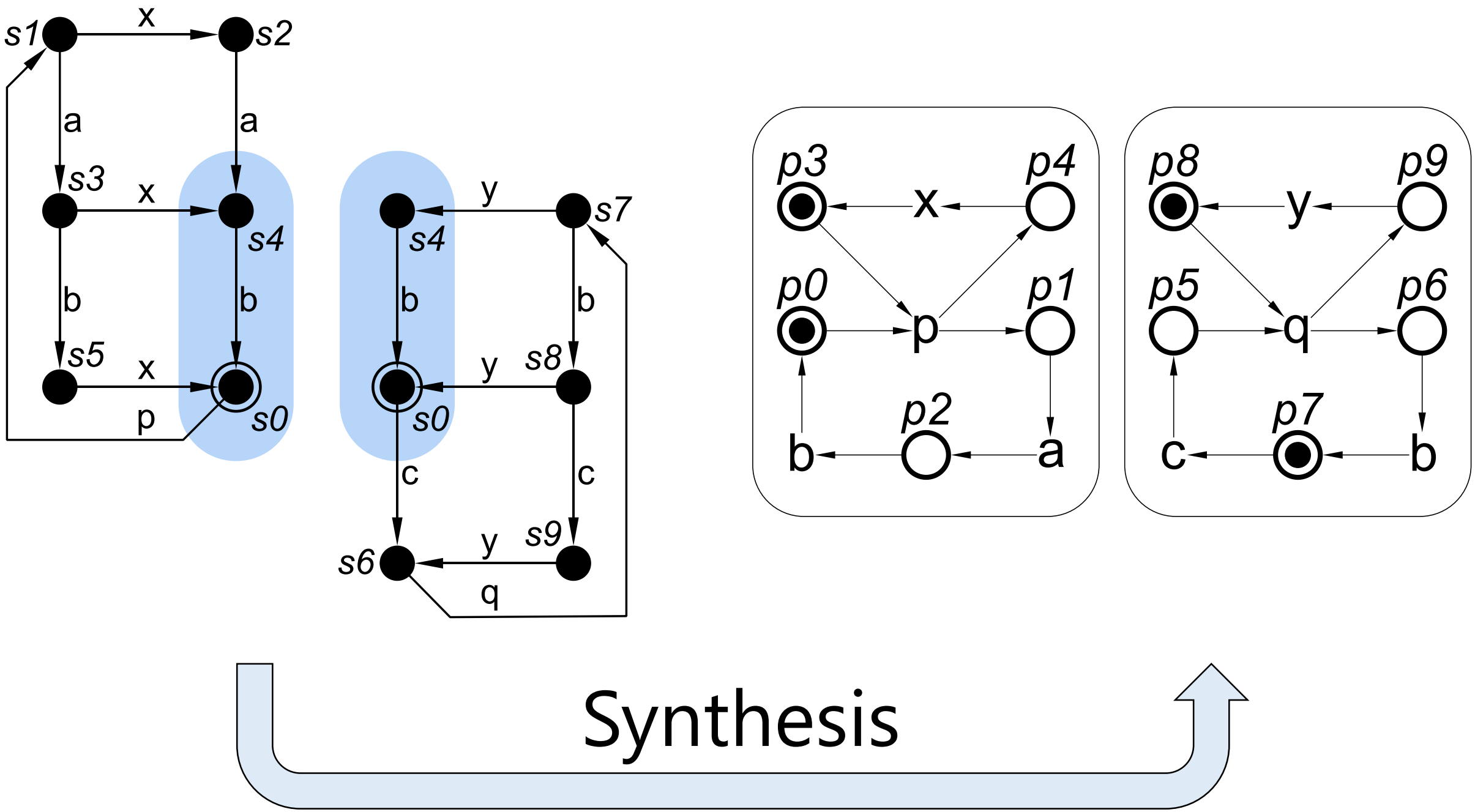
Ouch! Can you understand this Petri net?

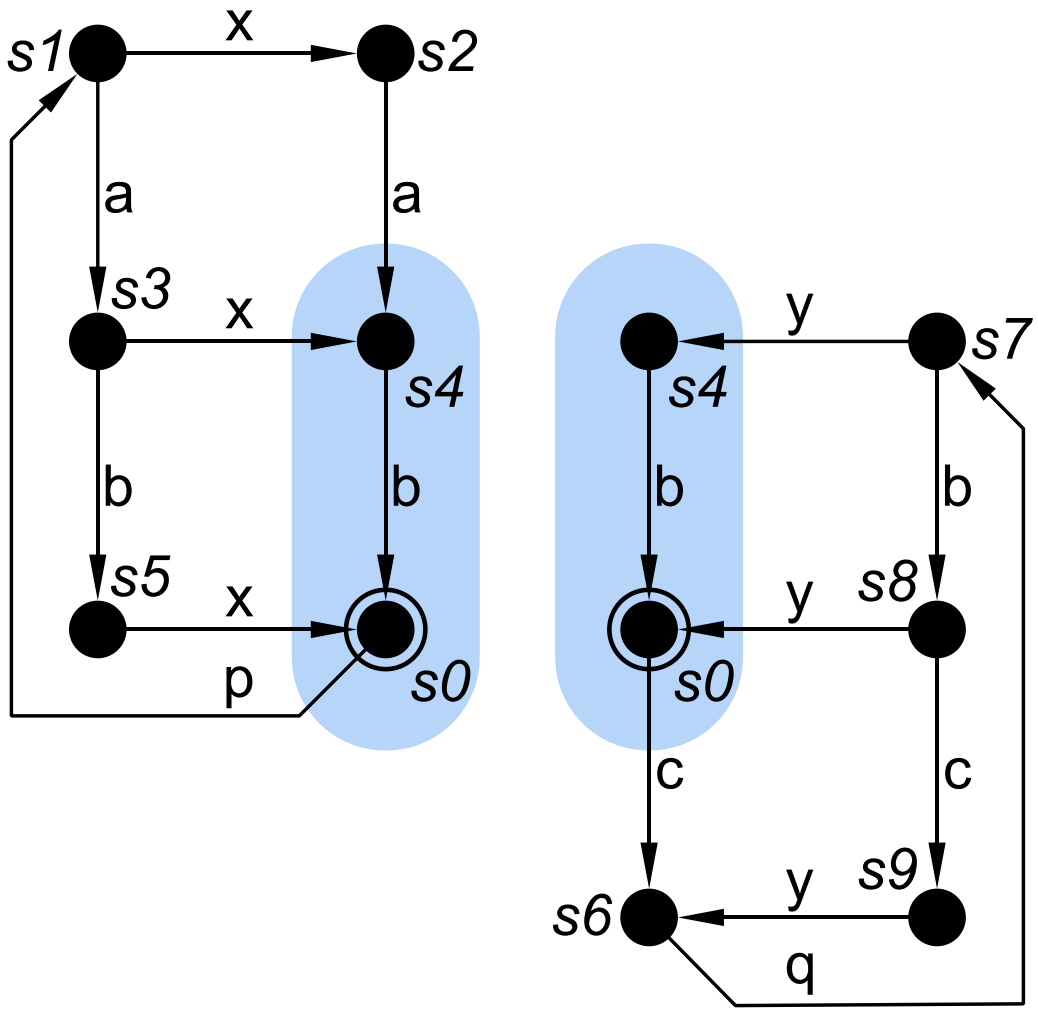

s1
x
s2
a
a
s3
x
y
s7
s4
b
b
b
s5
x
y
s8
p
s0
c
c
y
s9
s6
q
Decomposition

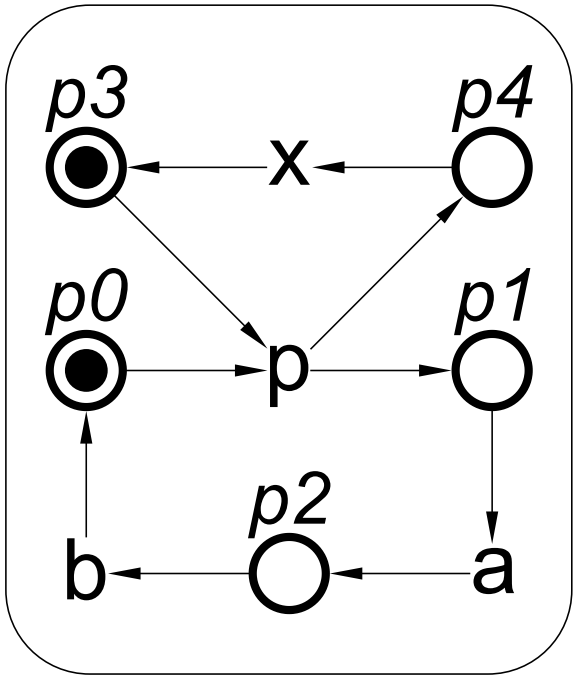


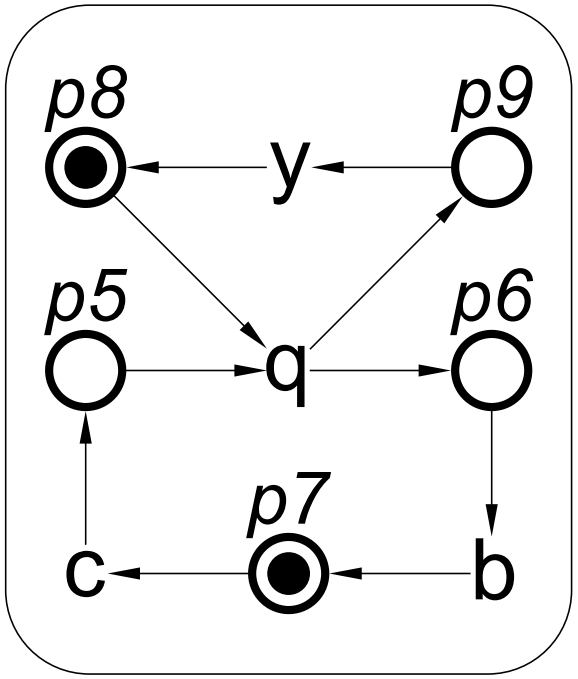


Idea: Keep the diamonds, remove non-persistency

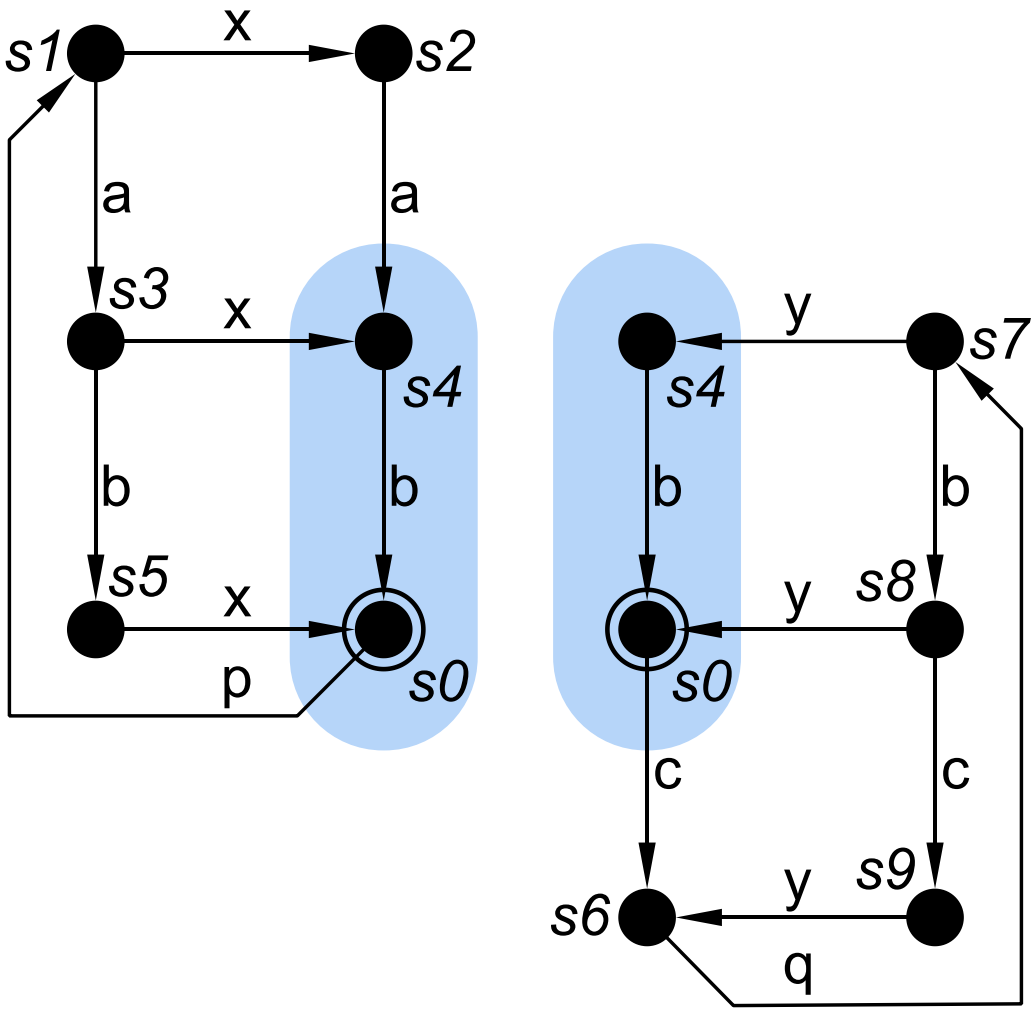

s1
x
s2
a
a
s3
x
s4
b
b
s5
x
p
s0
y
s7
s4
b
b
y
s8
s0
c
c
y
s9
s6
q
p3
x
p4
p0
p
p1
b
p2
a
p8
y
p9
p5
q
p6
c
p7
b
Synthesis

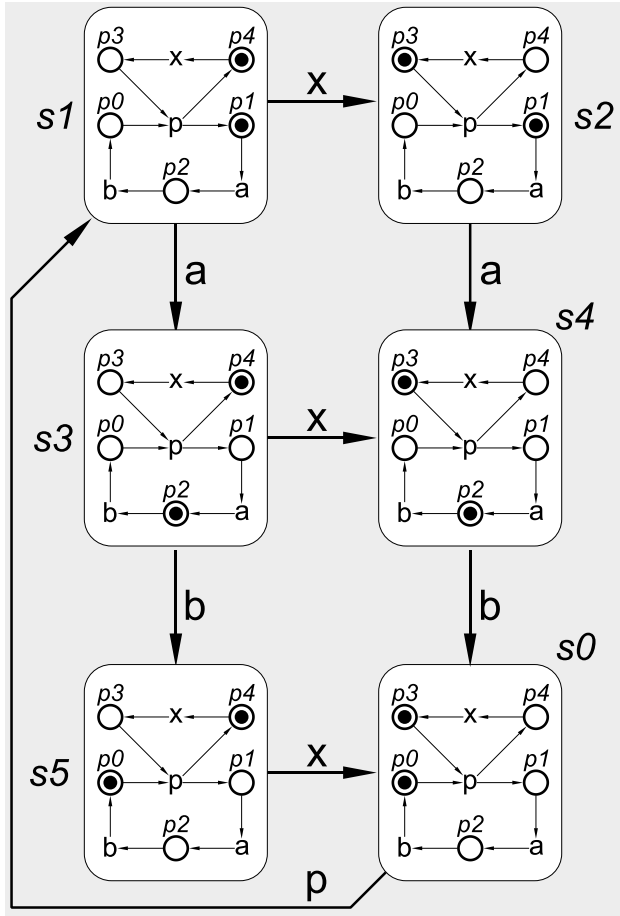


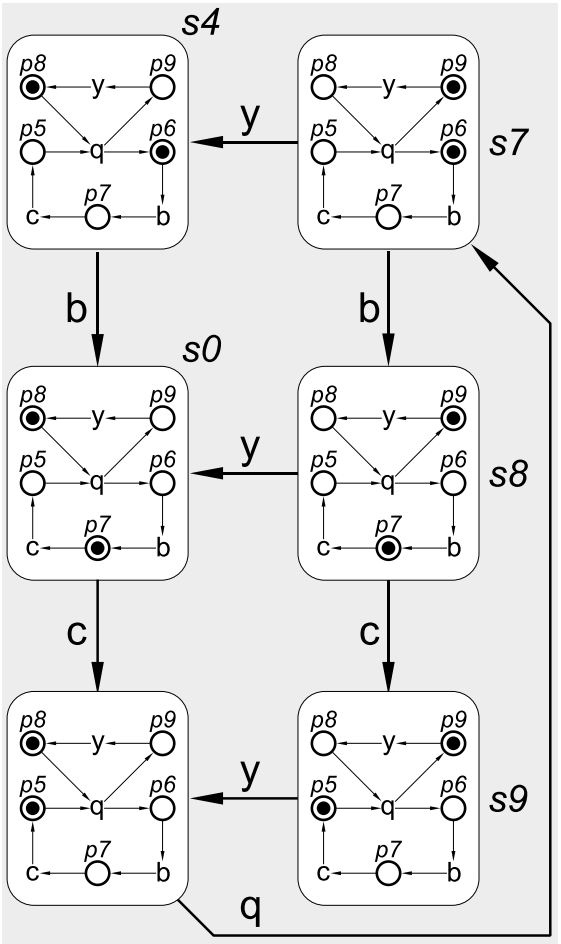




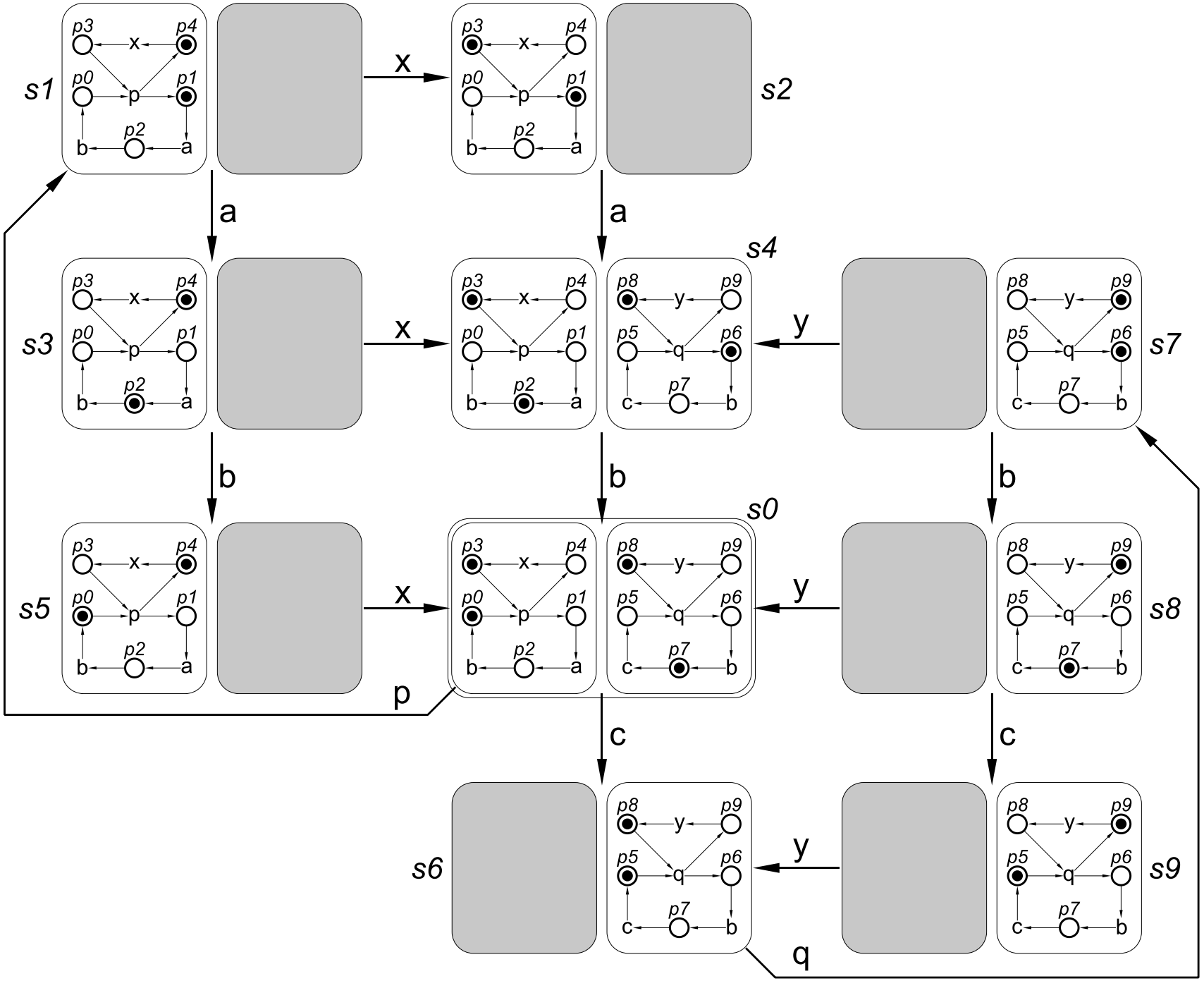
Aha, these are just marked graphs, but what do they mean?





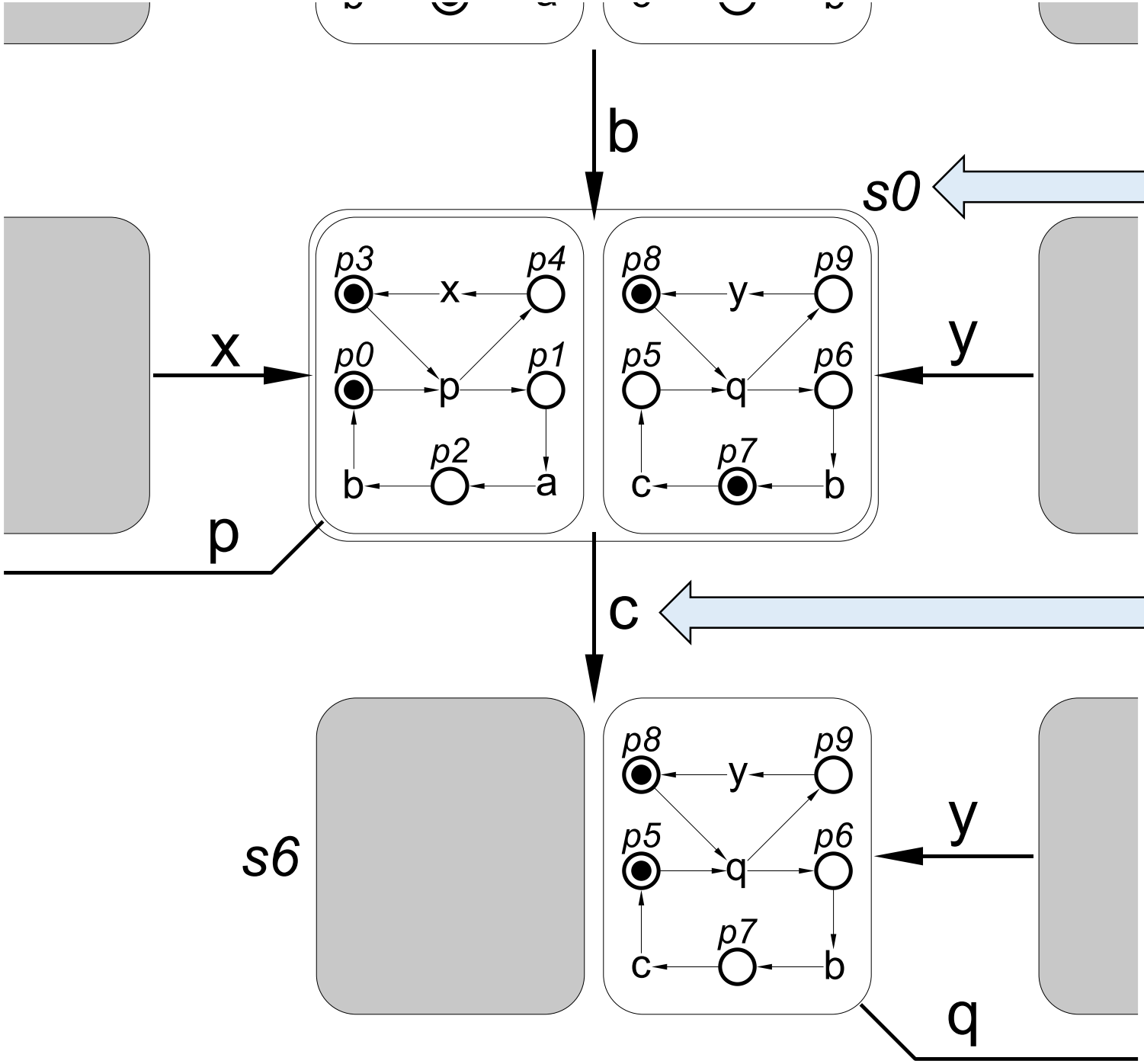


These are **windows** showing parts of the system behaviour.



Each **window** covers a part of the system behaviour, i.e. a **scenario**.

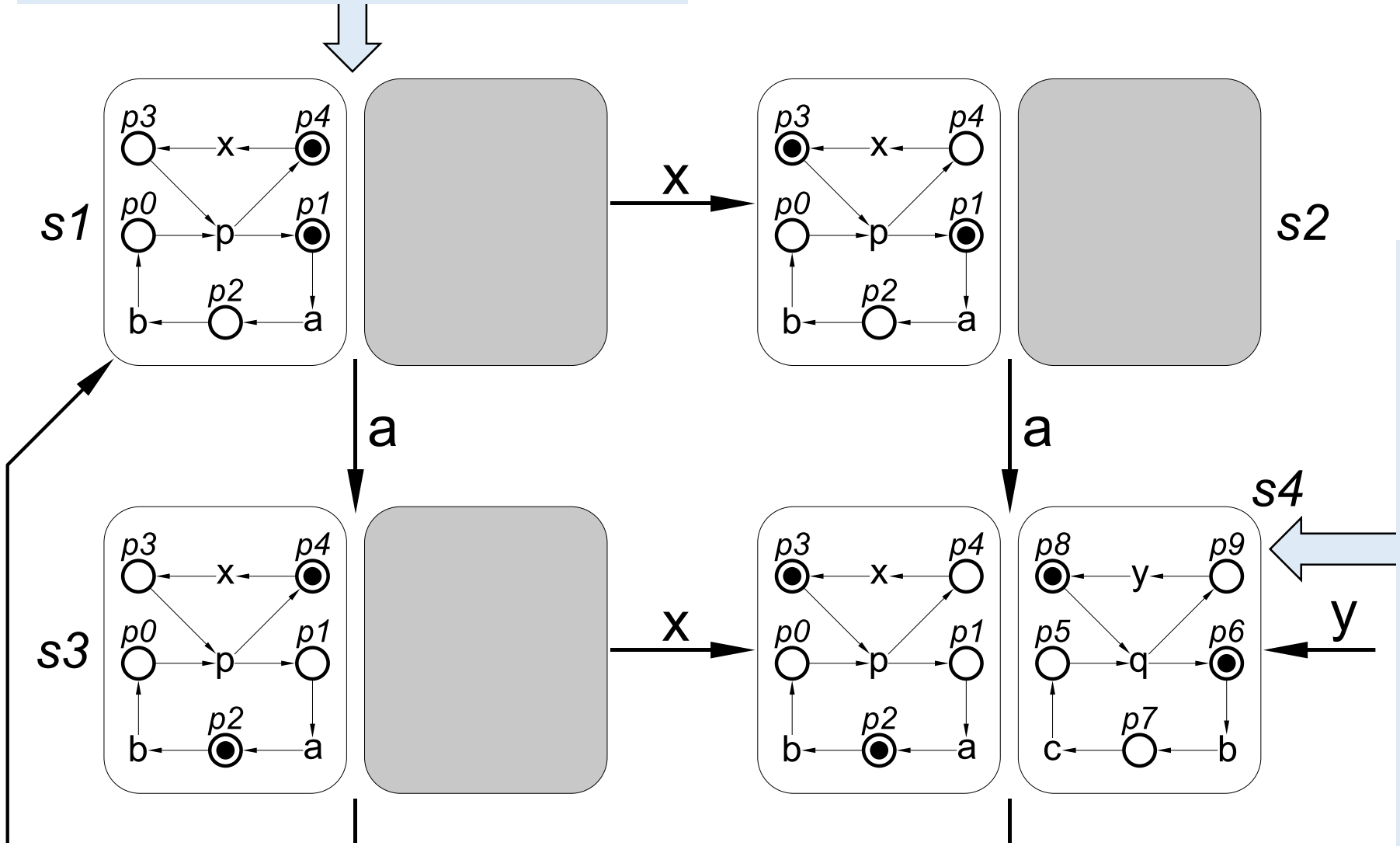
Each transition is covered by at least one window.



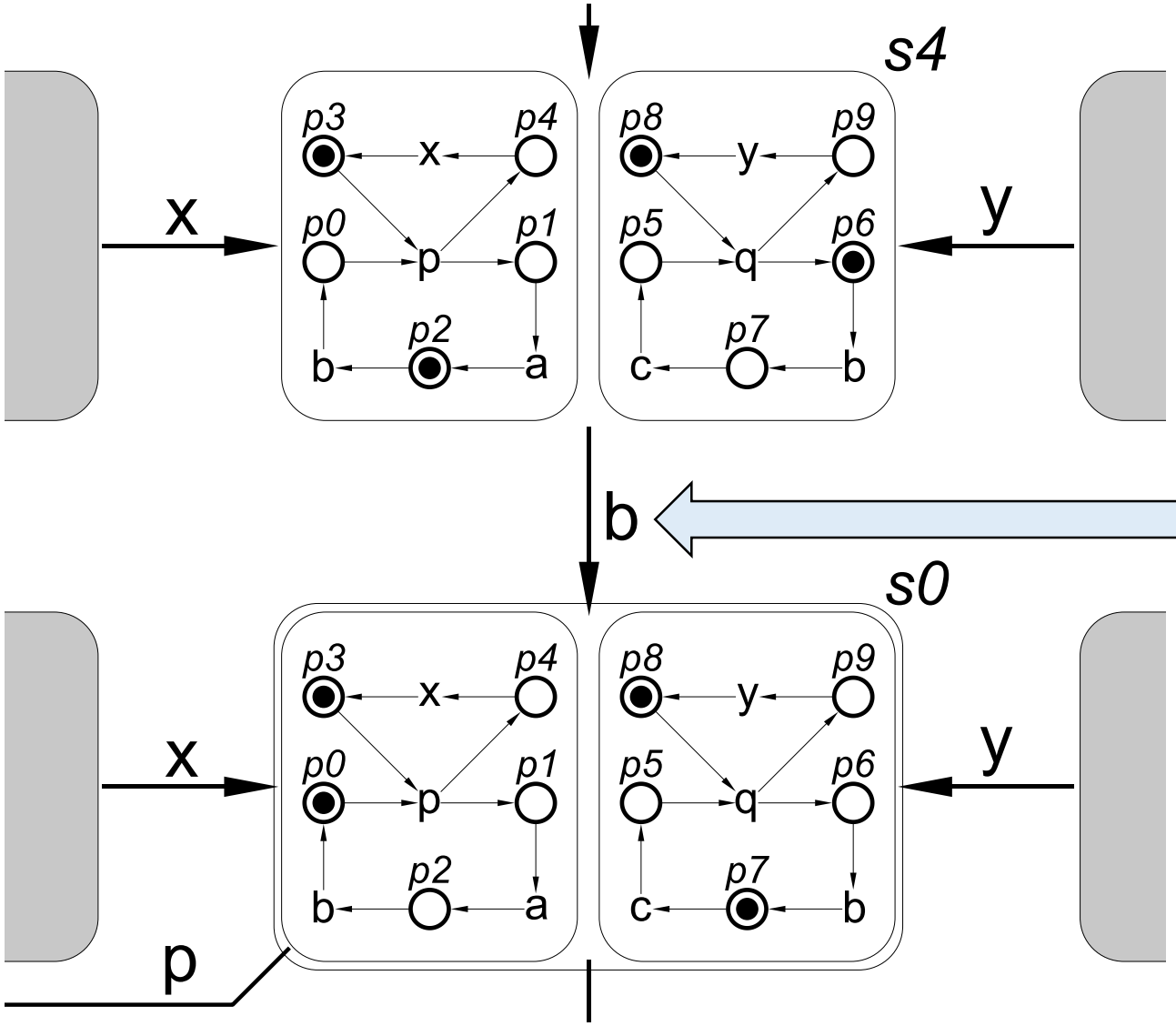
The initial state is covered by both windows.

Firing **c** is only possible in the second window, hence the first one becomes **inactive**.

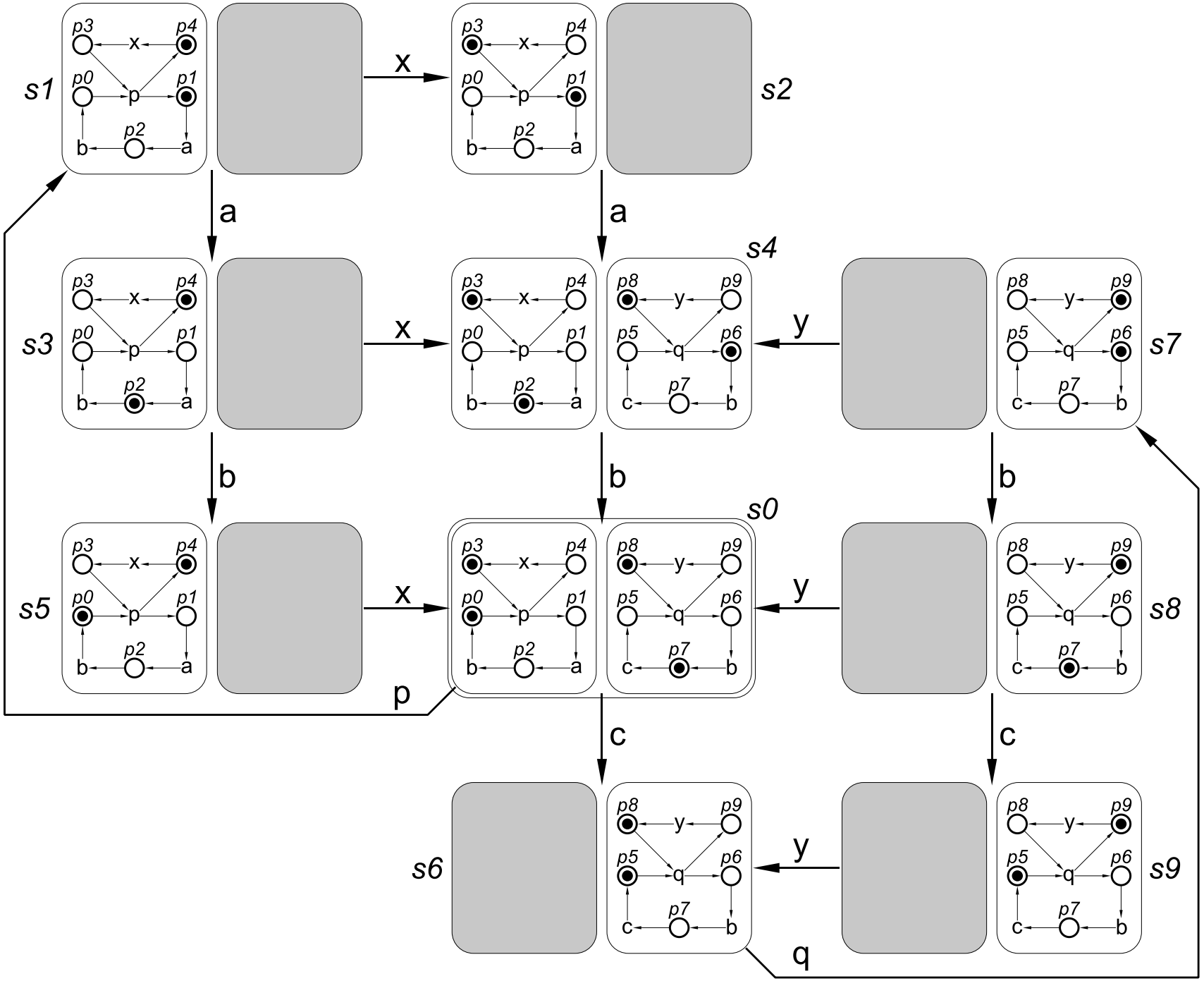
In **s1** the first window is **active**.



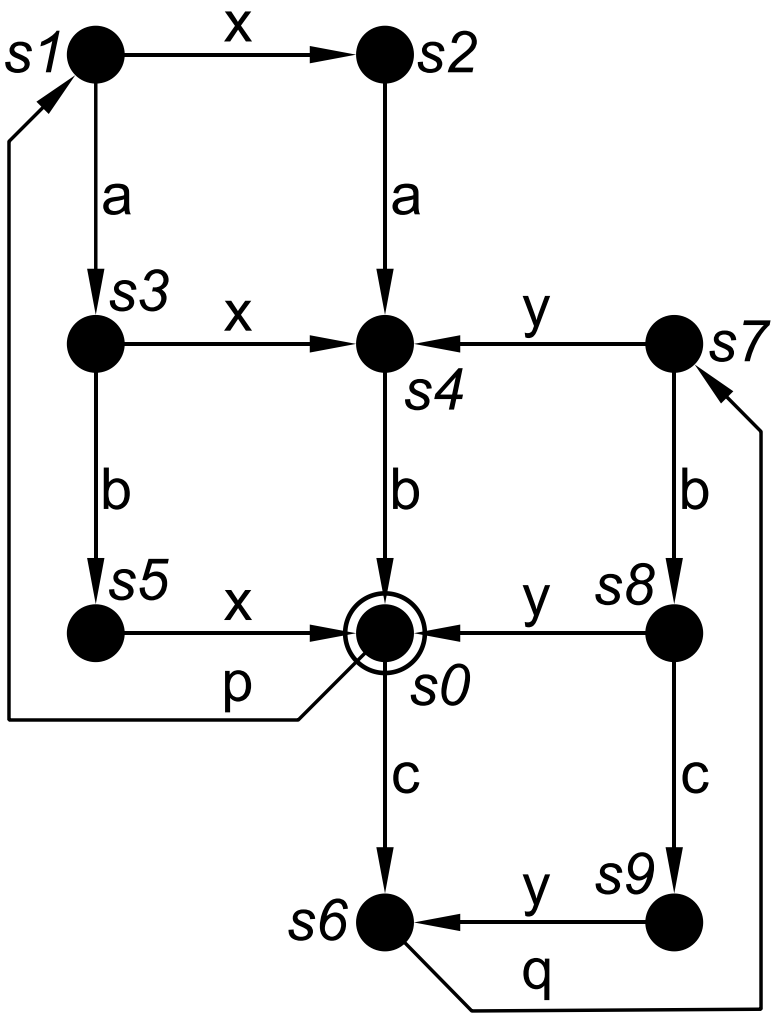
When **a** and **x** fire (in any order), the second window **wakes up**.

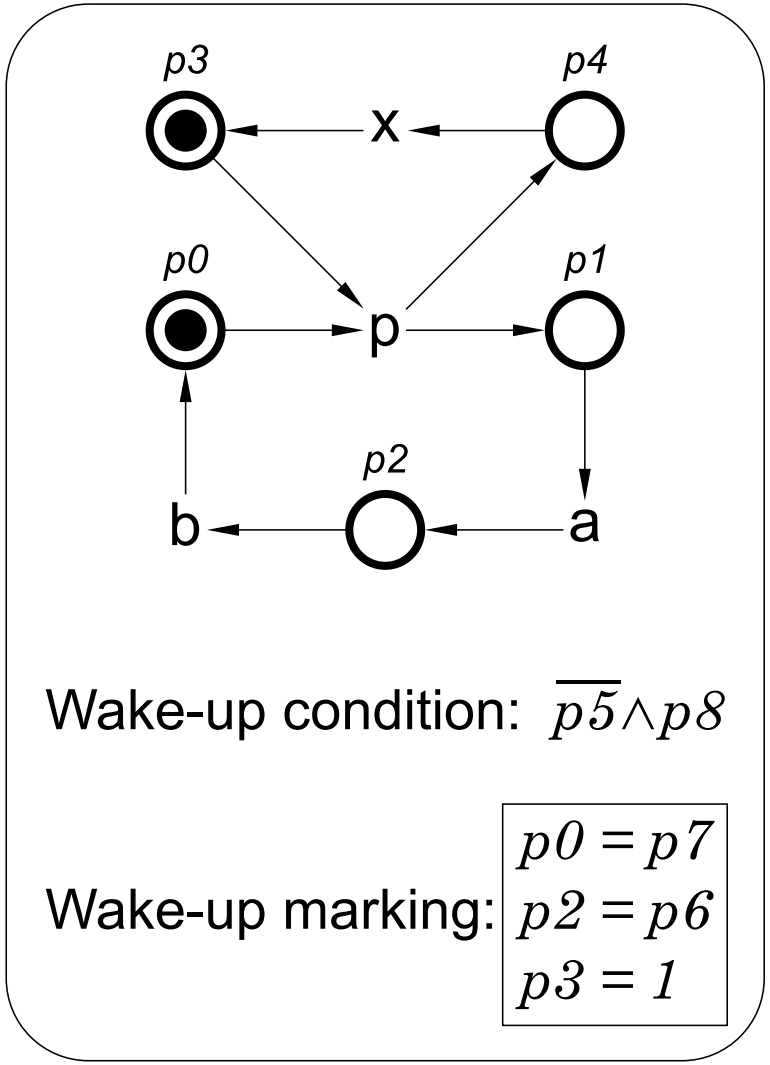


In **s4** both windows are active and **b** can be fired in both of them.



Note: when a window wakes up it must be correctly initialised with a **wake-up marking**.

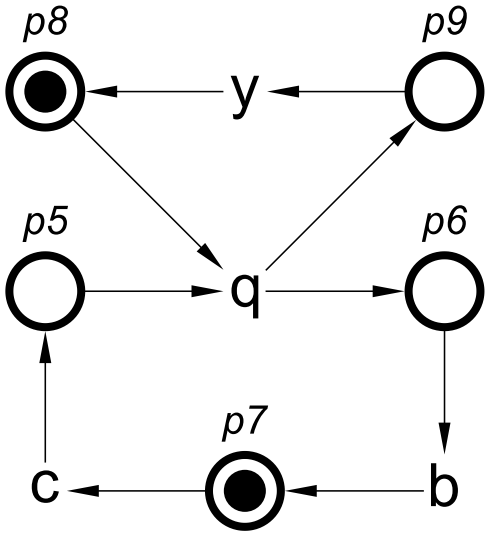




Wake-up condition: $\overline{p5} \wedge p8$

Wake-up marking:

$$p0 = p7$$
$$p2 = p6$$
$$p3 = 1$$



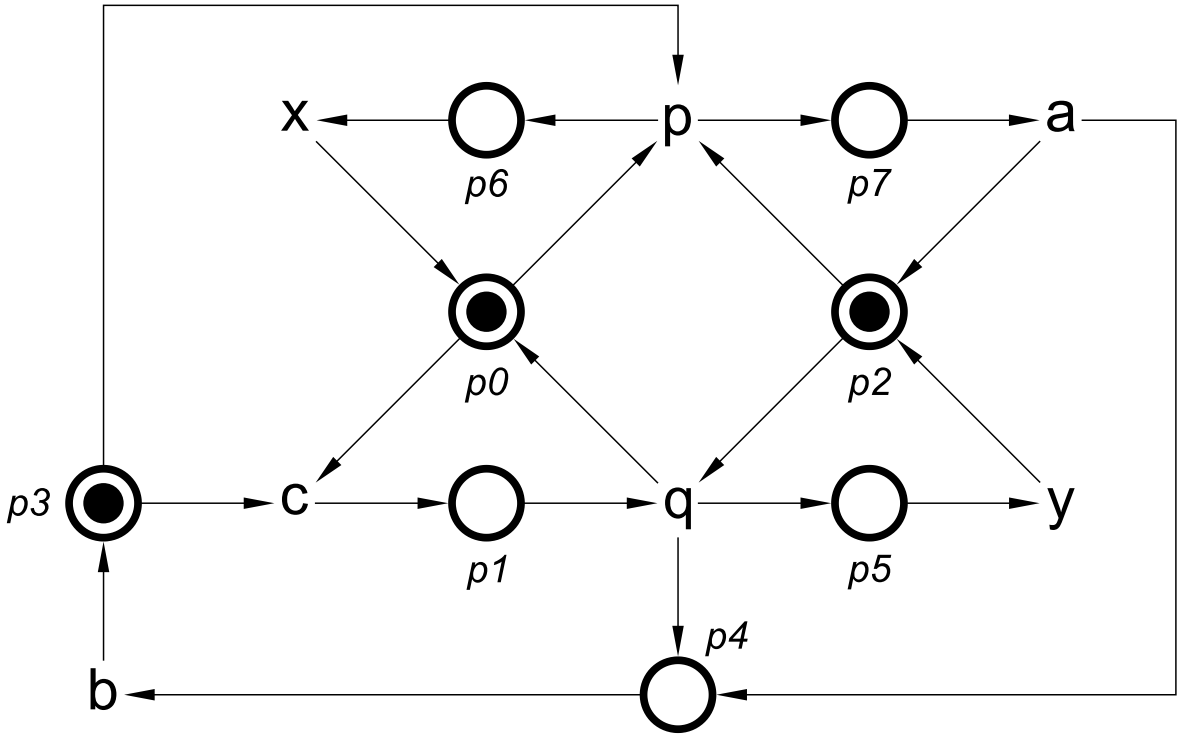
Wake-up condition: $\overline{p1} \wedge p3$

Wake-up marking:

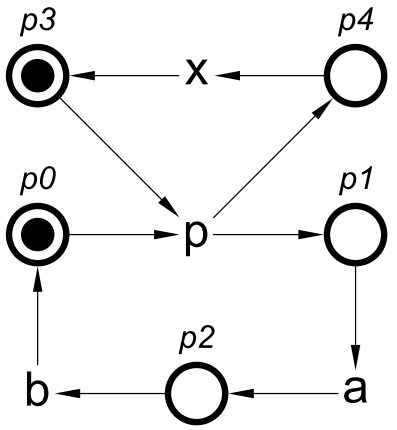
$$p6 = p2$$
$$p7 = p0$$
$$p8 = 1$$

Window decomposition

Direct synthesis



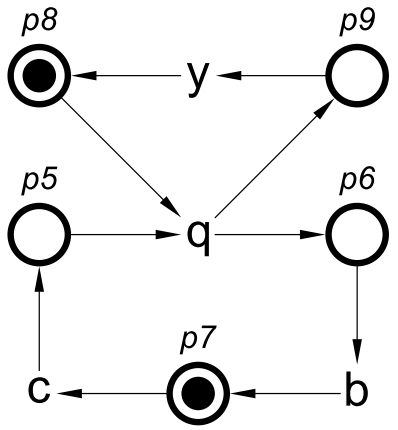
Window decomposition



Wake-up condition: $\overline{p5} \wedge p8$

Wake-up marking:

$$p0 = p7$$
$$p2 = p6$$
$$p3 = 1$$



Wake-up condition: $\overline{p1} \wedge p3$

Wake-up marking:

$$p6 = p2$$
$$p7 = p0$$
$$p8 = 1$$

Which description do you prefer?

Part II:
Automated Window Decomposition

Windows Decomposition Problem

Given:

- A labelled transition system **L**
- A set of desired structural properties, e.g. forward and backward **persistence**, **determinism**, **connectedness**…

Result:

- A set of windows $\mathbf{W_1}…\mathbf{W_n}$, such that $\mathbf{L} = \mathbf{W_1} \cup … \cup \mathbf{W_n}$
- Each window $\mathbf{W_k}$ satisfies the structural properties
- Wake-up condition **c** and marking **m** for each window

Implementation (sketch)

Discovering windows:

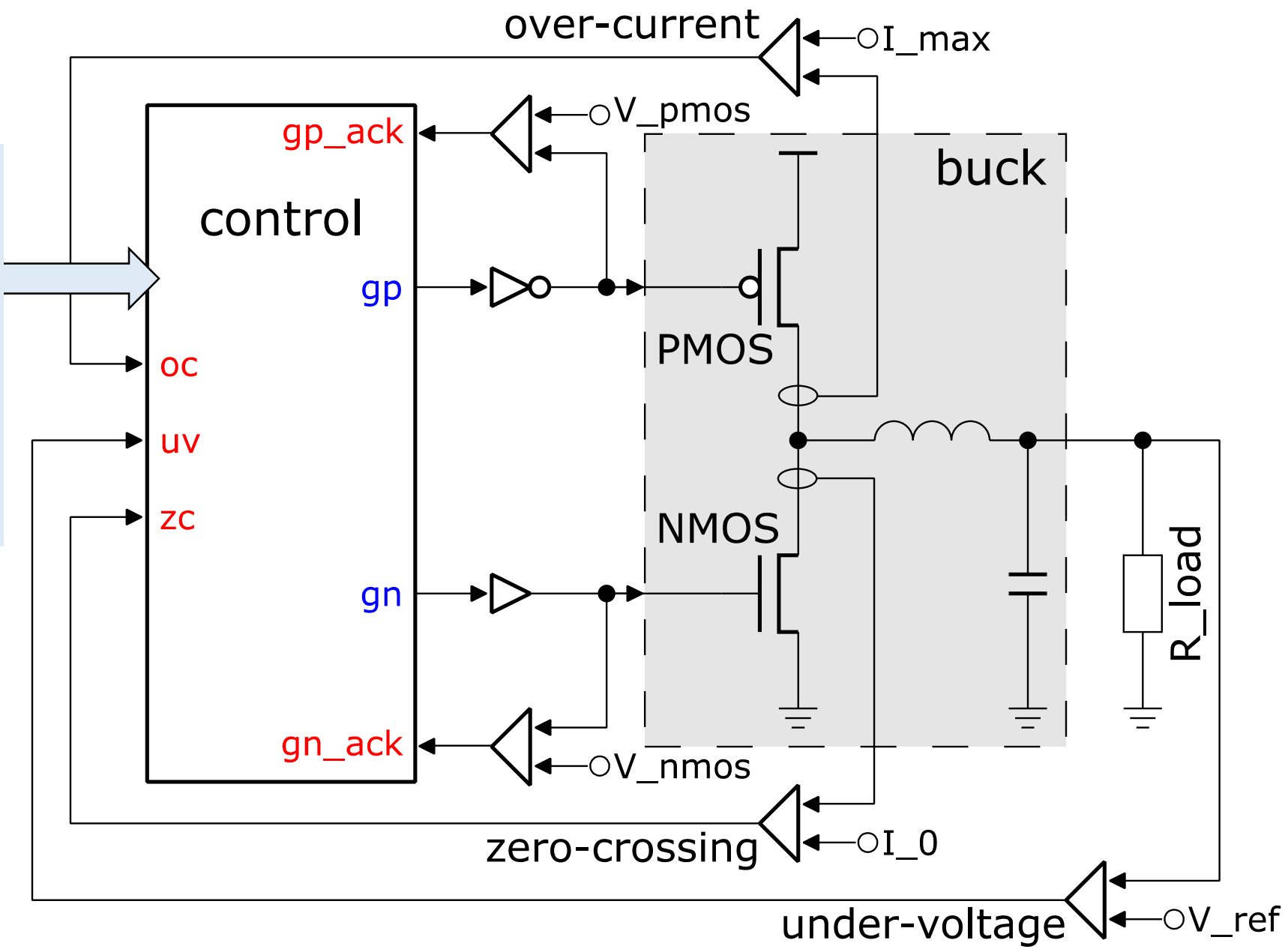
- Inspired by Javier de San Pedro and Jordi Cortadella (2016)
- SAT formulation: one Boolean variable per transition
- Desired structural properties are Boolean constraints
- Successively discover largest possible windows

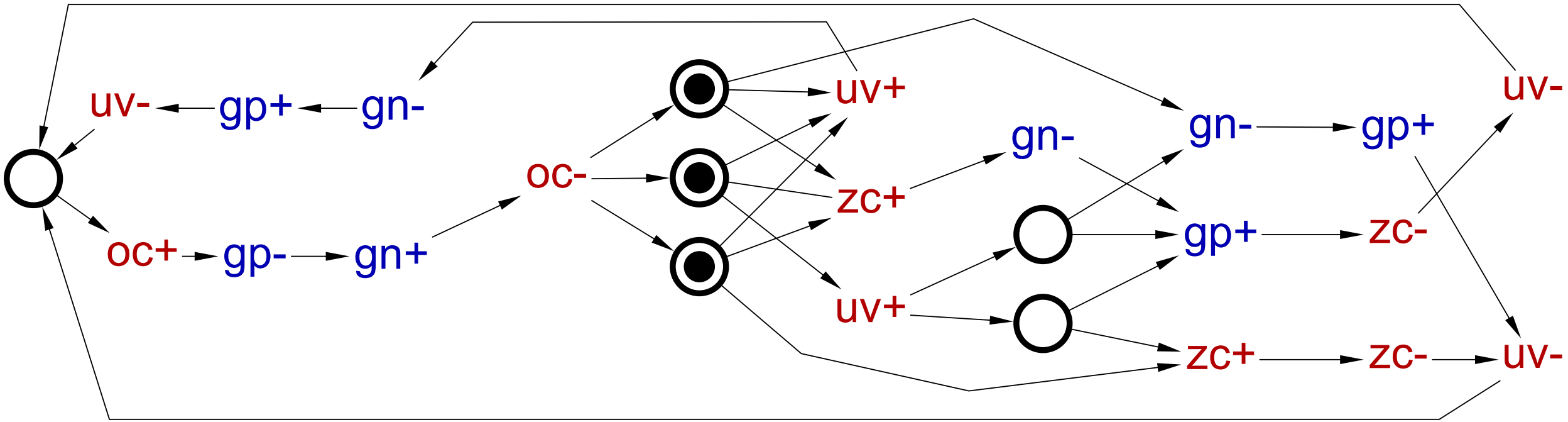
Deriving wake-up conditions and markings:

- Build a wake-up truth table: one row per state
- Perform Boolean minimisation (see paper for details)

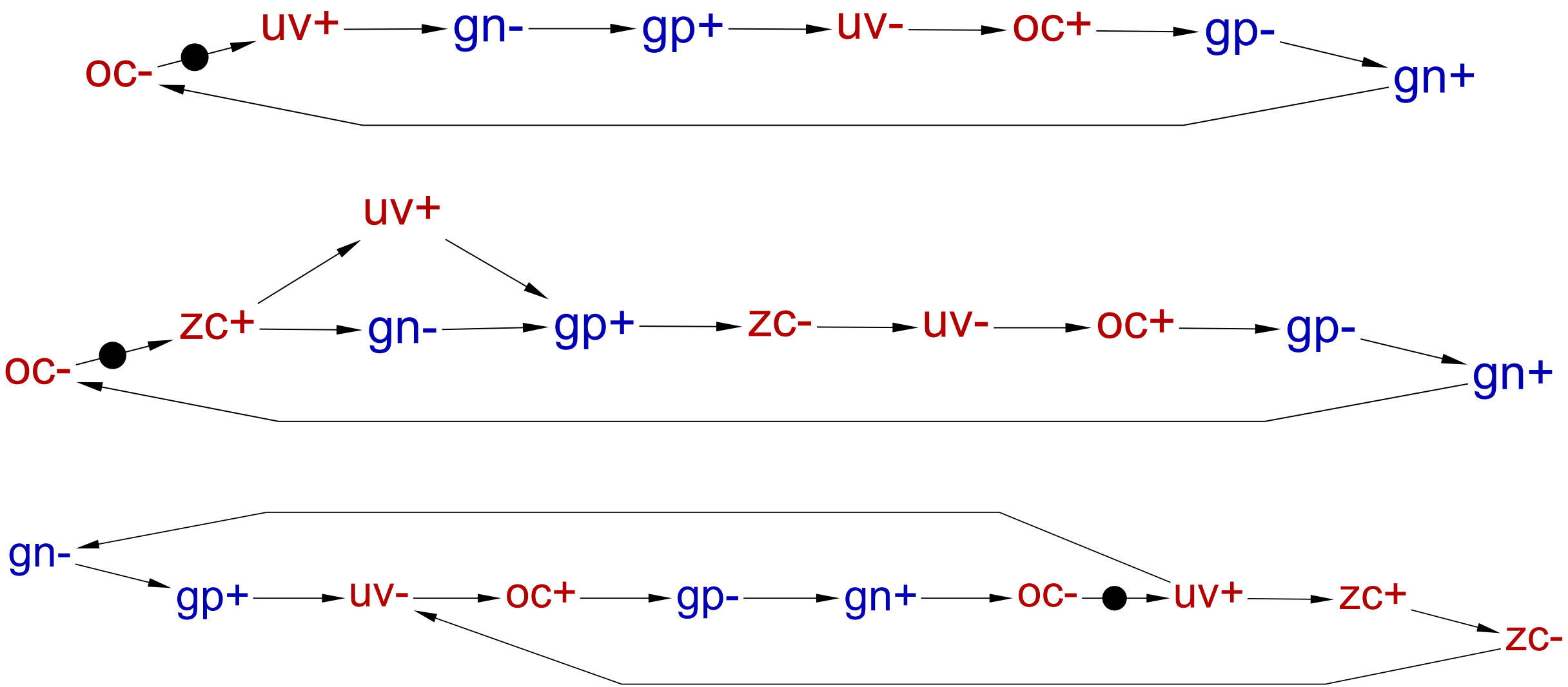
Part III: Applications

Asynchronous power management controller





Petri net model synthesised from the underlying transition system



Discovered process windows

Traces:

adeac

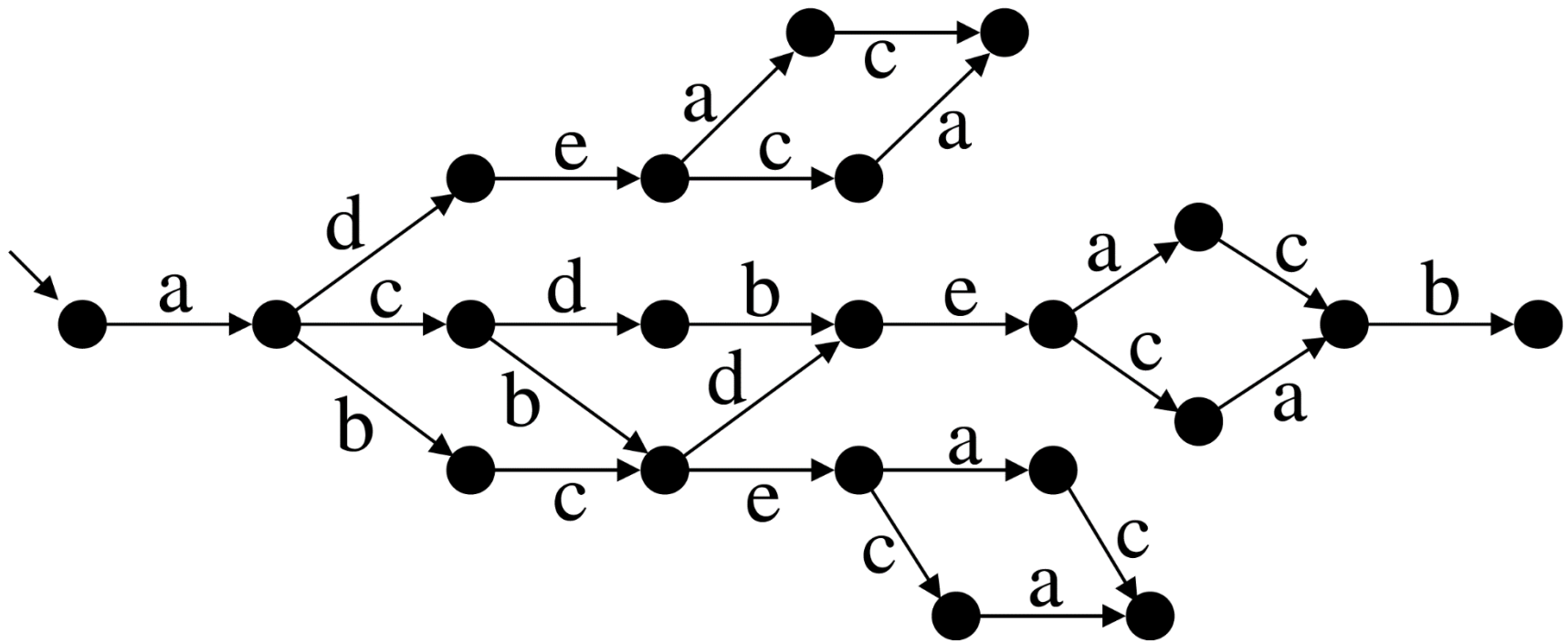
acdbeacb

abceac

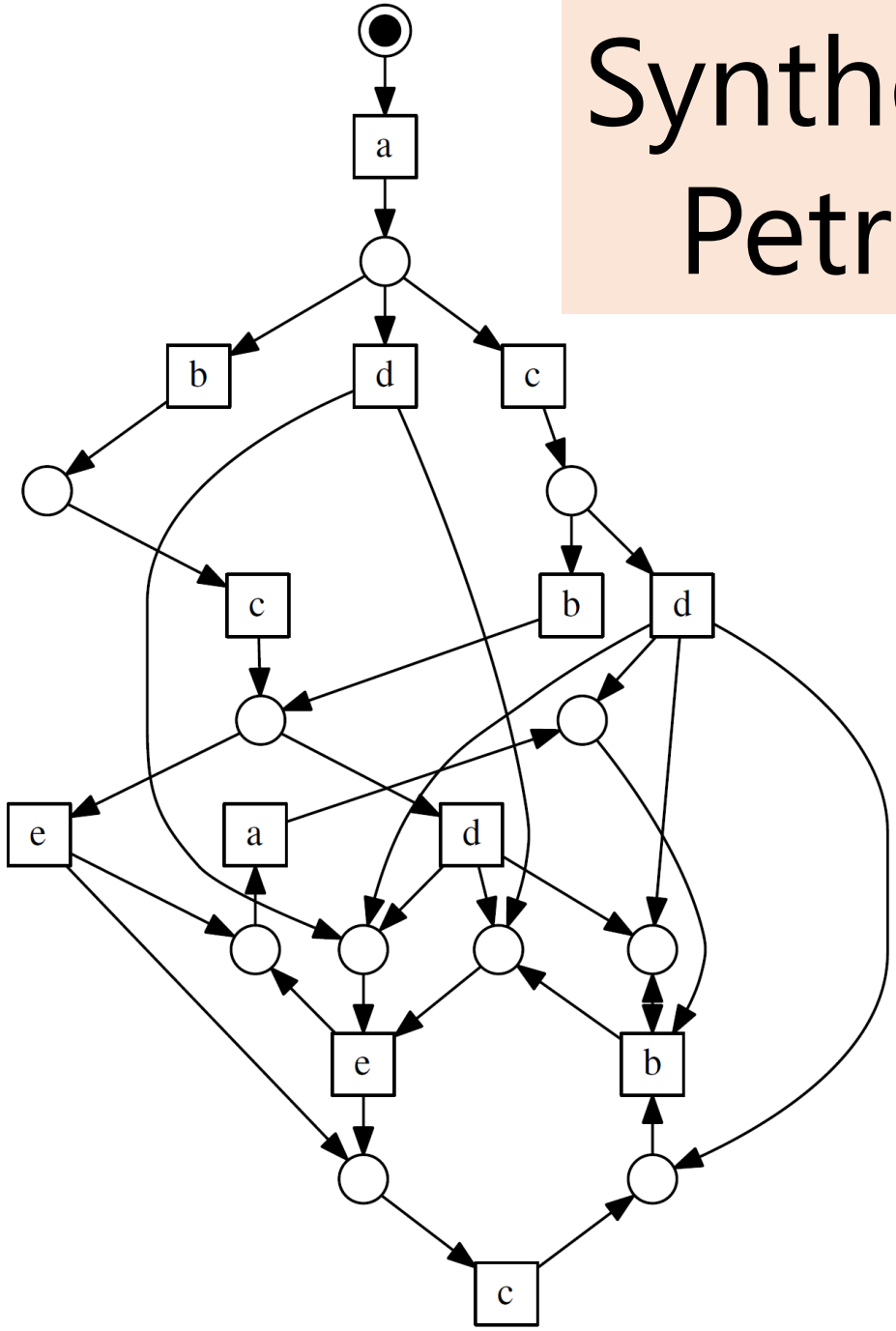
acbdecab

adeca

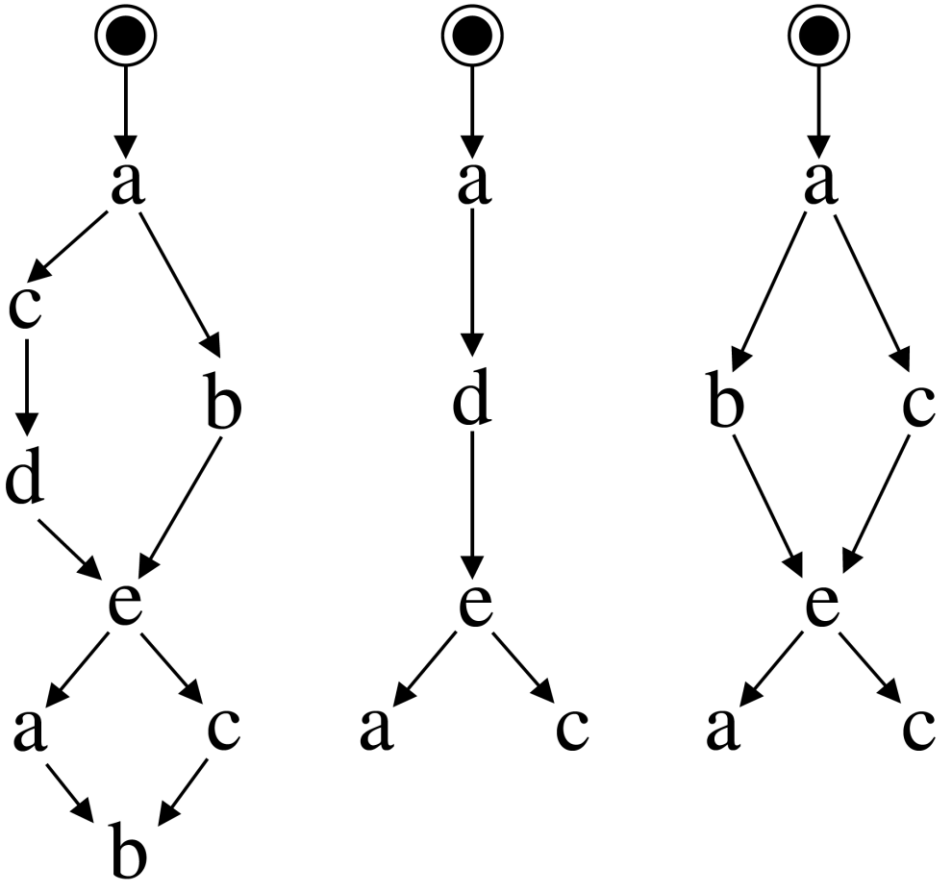
acbeca

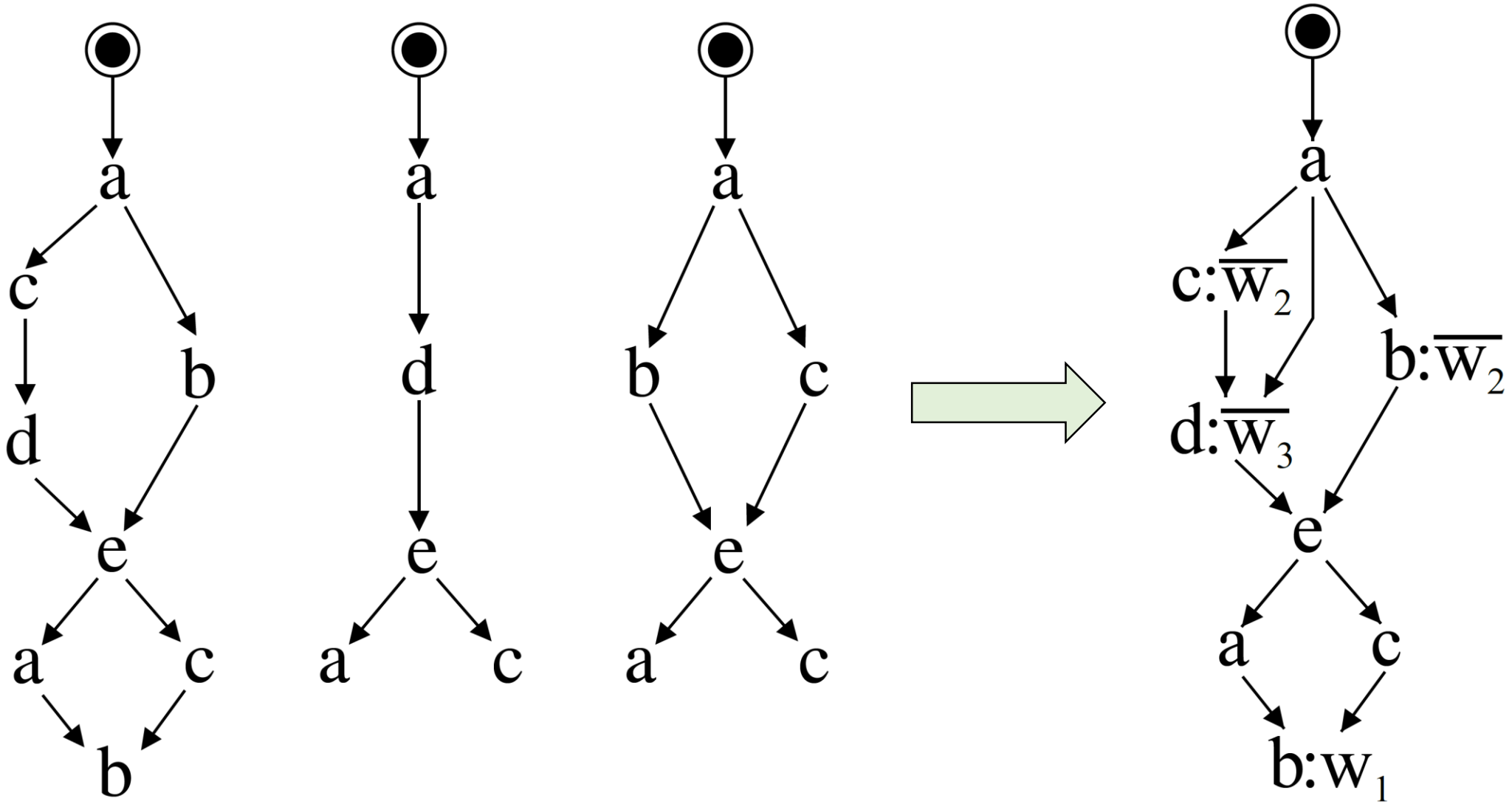


Synthesised Petri net



Discovered process windows





Discovering common patterns in scenarios

Summary

Process windows

- A new approach to representing complex processes
- Automated discovery of windows with desirable structural properties, such as marked graphs
- Implemented in **Workcraft** toolkit (ask for a demo!)

Future research

- Beyond choice-free scenarios
- Exploit the structure for efficient process analysis
- Circuit synthesis

Thank you!