

Three exercises:

- 1) scheduling of operations on a machine
- 2) series connection of queueing systems
- 3) queueing system with parallel servers

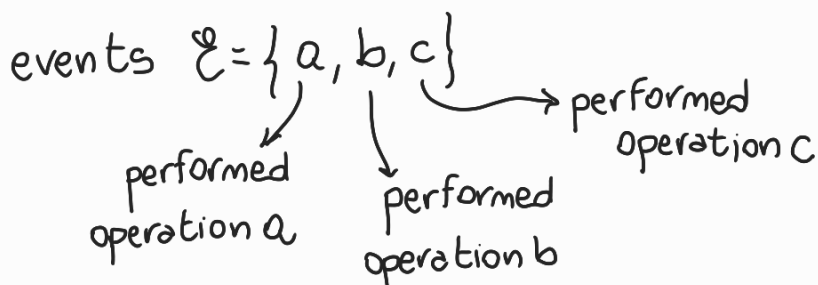
EXERCISE 1

A machine performs operations of three types, denoted a , b and c . For technical reasons, an operation of type c cannot be performed immediately after two consecutive operations both of type a , or both of type b . At initialization, no operation has been performed.

1. Model the logic of the machine.
2. Model a system designed to support the scheduling of the operations on the machine: given a sequence of operations, the system returns whether the sequence is feasible for the machine, or not.

1. We model the system logic with a state automaton

$$(\mathcal{E}, \mathcal{X}, \Gamma, f, x_0)$$

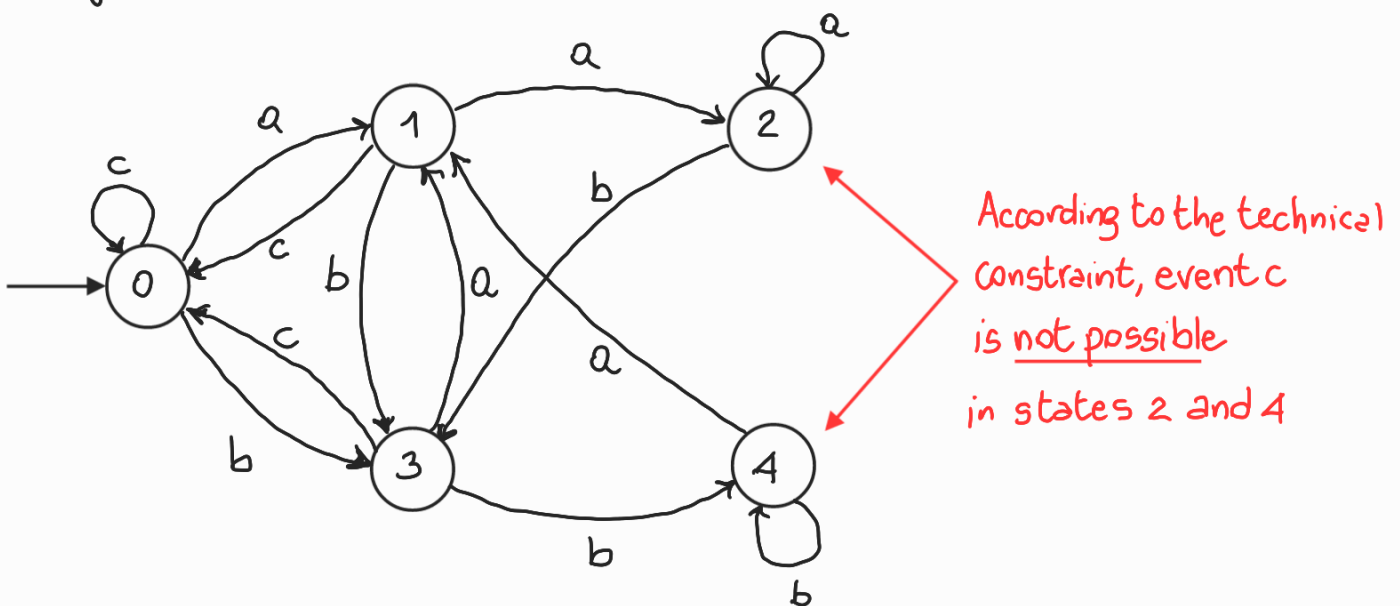


The definition of state for this system should take into account the last two operations performed

↓
information needed to figure out whether operation c can be performed next or not

$$x = \begin{cases} 1: \text{last operation } a, \text{ second last } \underline{\text{not}} a \\ 2: \text{last two operations } a \\ 3: \text{last operation } b, \text{ second last } \underline{\text{not}} b \\ 4: \text{last two operations } b \\ 0: \text{otherwise} \end{cases}$$

Γ , f and x_0 are defined by the following graph:



This model is useful, e.g., for monitoring :

a monitoring software receives information from the machine (operation performed) and updates the estimated state of the machine, which is visualized on the screen of an operator.



~> The sequence of operations received from the "field" is always feasible (except for possible communication errors...)

2. We modify the previous model :

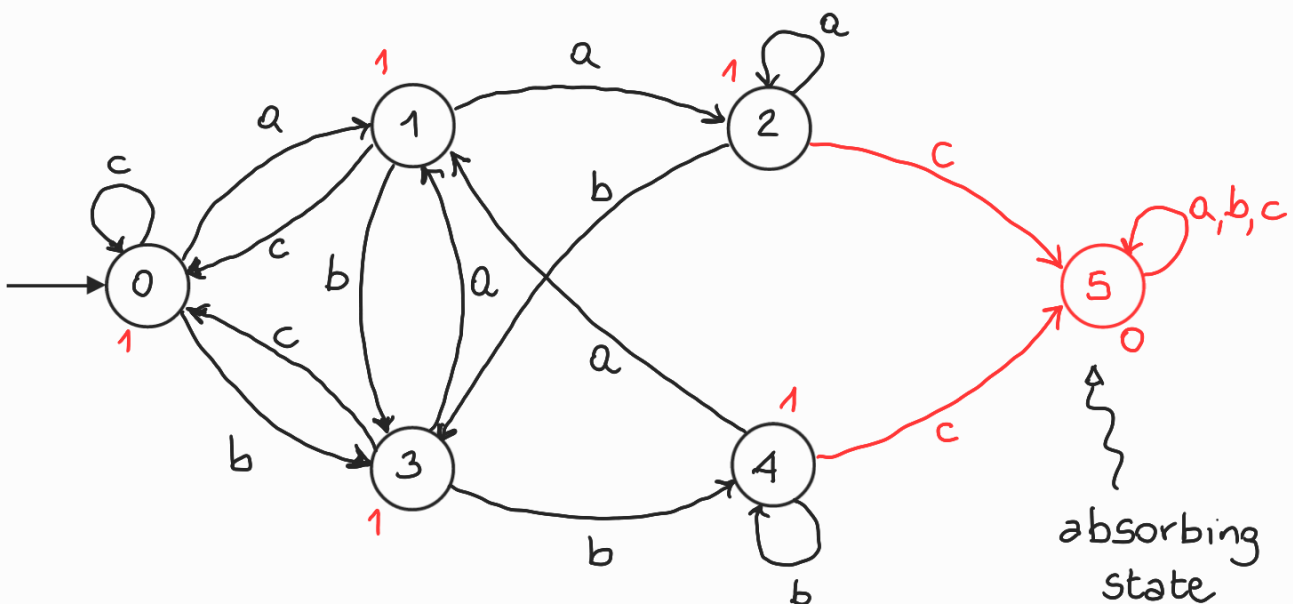
- we add a dummy state collecting error situations

state 5: error

- We define the following output:

$$y = \begin{cases} 0 & \text{if the sequence of operations is infeasible} \\ 1 & \text{otherwise} \end{cases}$$

Resulting model:



This model is useful, e.g., for planning:

given a sequence of operations
generated in a planning process,
if the final output is 0,
the sequence is infeasible,
and feasible otherwise.



example: determine whether the sequence of operations

caabcbbcab

is feasible for the machine

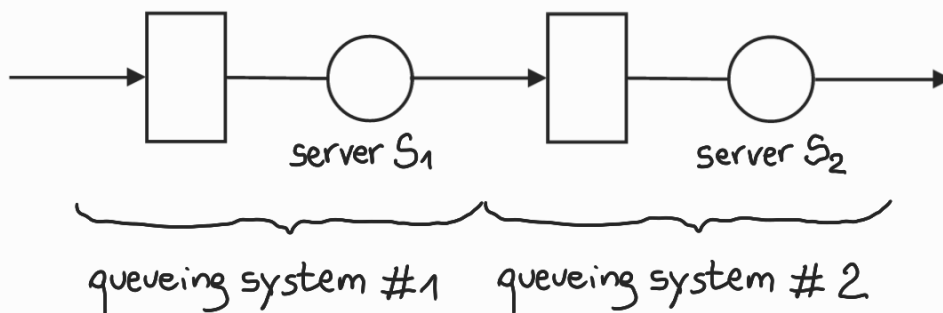
event	c	a	a	b	c	b	b	c	a	b	
state	0	0	1	2	3	0	3	4	5	5	5
output	1	1	1	1	1	1	1	1	0	0	0

the final output is 0;

the sequence is infeasible

EXERCISE 2

Series connection of two queueing systems:



Technical constraint: if the second queueing system is full
when S_1 terminates a service, S_1 keeps the customer,
and is not available for a new service until the customer
can move forward (the server is **blocked**).

We model the system with a state automaton $(\mathcal{E}, \mathcal{X}, \Gamma, f, x_0)$:

events $\mathcal{E} = \{a, d_1, d_2\}$

- a → arrival of a customer
- d_1 → termination of a service in S_1
- d_2 → termination of a service in S_2

state $x = \begin{cases} x_1 \\ x_2 \end{cases} \rightarrow \text{queueing system \#1: } x_1 \in \{0, 1, 2, 3, 4\}$
 $\rightarrow \text{queueing system \#2: } x_2 \in \{0, 1, 2\}$

where:

0: empty

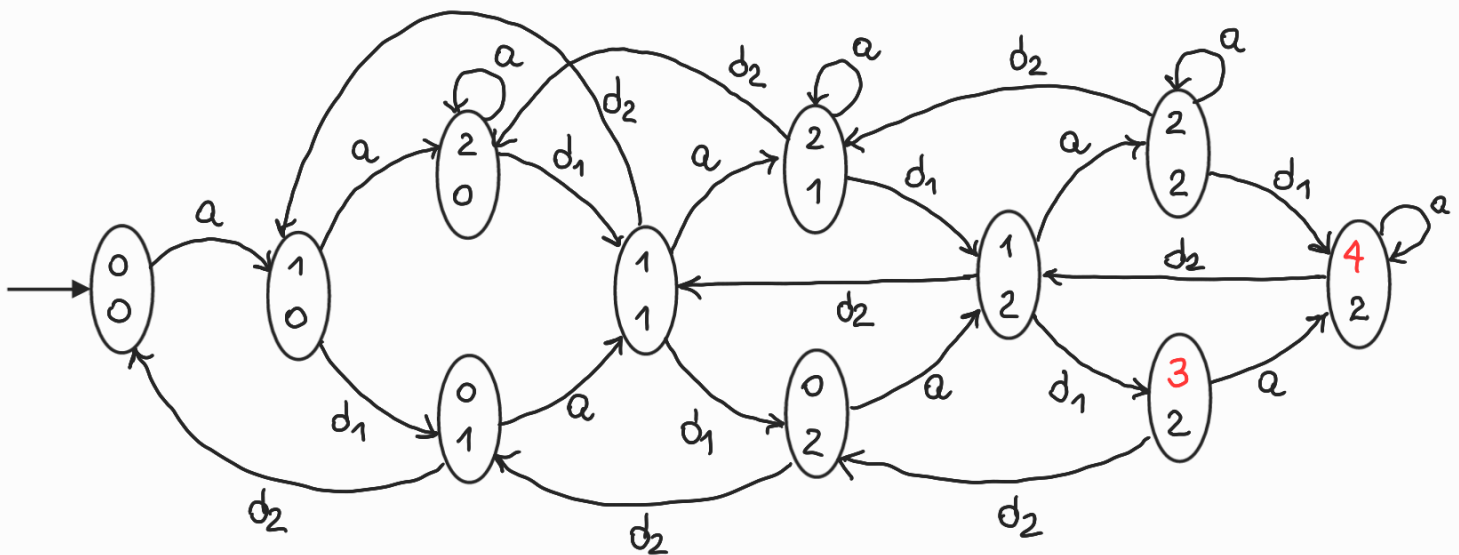
1: one customer, server working

2: two customers, server working

3: one customer, server blocked

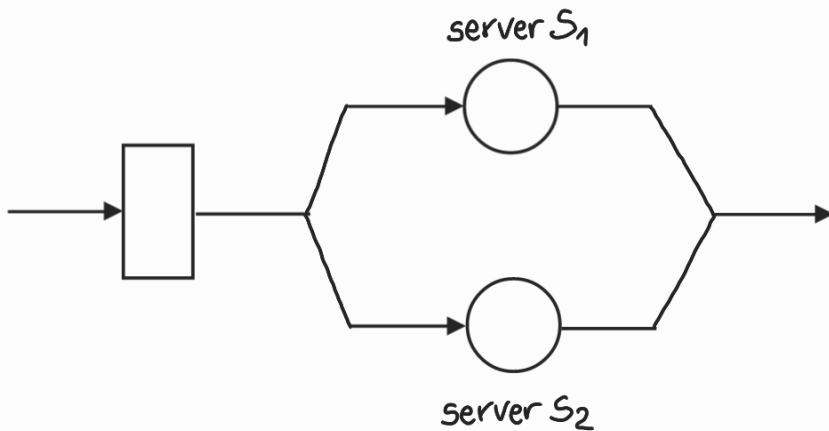
4: two customers, server blocked

Γ , f and x_0 are defined by the following graph
 (we assume that the system is initially empty):



EXERCISE 3

Queueing system with parallel servers:



The two servers perform the same service, but server S_1 is faster than server $S_2 \Rightarrow$ the two servers are **different**

Rule for routing the customers: if both servers are available, the next arriving customer is served by S_1

We model the system with a state automaton $(\mathcal{E}, \mathcal{X}, \Gamma, f, x_0)$:

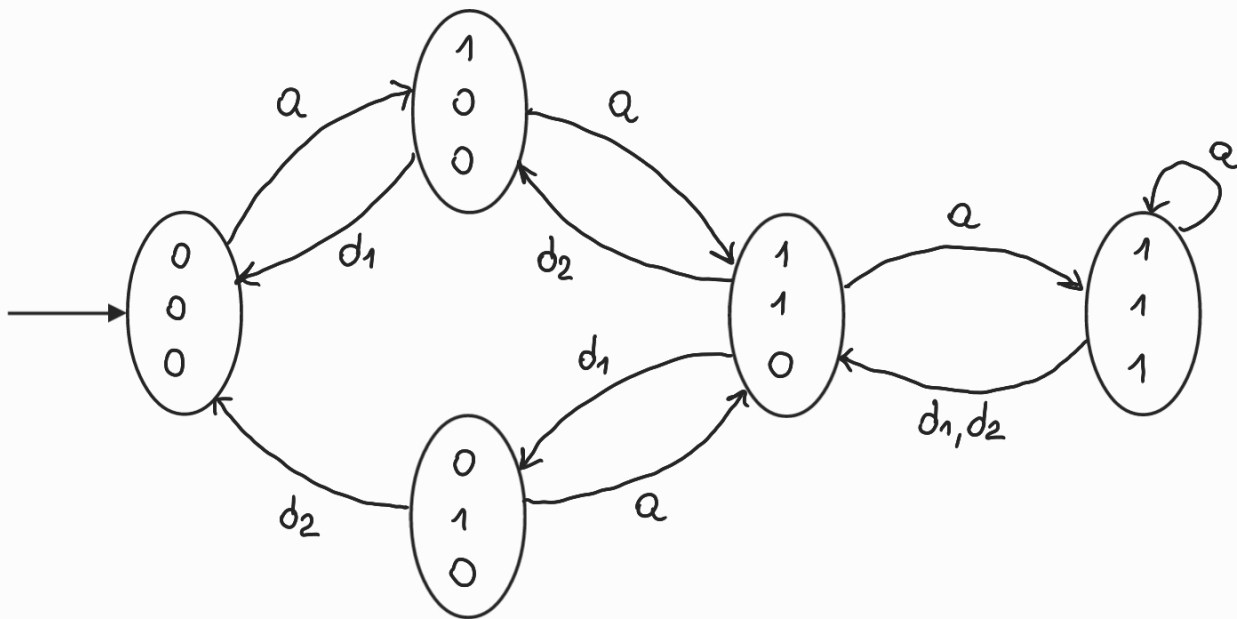
events $\mathcal{E} = \{a, d_1, d_2\}$

arrival of a customer $\swarrow$ $\searrow$ $\searrow$

termination of a service in S_1 $\searrow$ termination of a service in S_2

state $\mathcal{X} = \begin{cases} x_1 \rightarrow \text{server } S_1: 0 \text{ (idle), } 1 \text{ (working)} \\ x_2 \rightarrow \text{server } S_2: 0 \text{ (idle), } 1 \text{ (working)} \\ x_3 \rightarrow \text{queue: } 0 \text{ (empty), } 1 \text{ (full)} \end{cases}$

Γ, f and x_0 are defined by the following graph
(we assume that the system is initially empty):



REMARK: If the two servers are **identical**, the model can be simplified

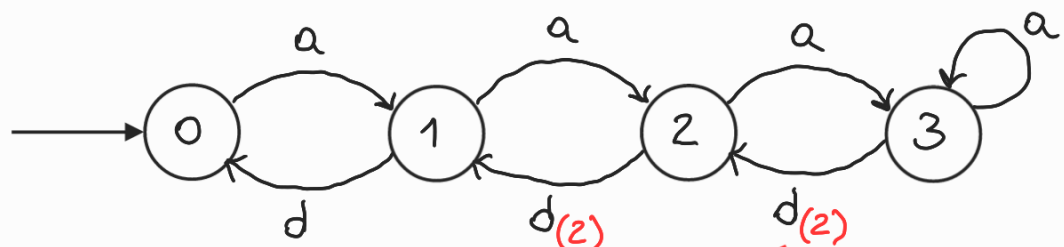


|| We don't need to distinguish which of the two servers is working

events $\mathcal{E} = \{a, d\}$
 ↓ ↓
 arrival termination of
 of a customer a service (generic)

state $x = \# \text{ customers in the system} \in \{0, 1, 2, 3\}$

graph:



We use this subscript to remind us that event d may be originated by one of **two** sources $\Rightarrow$ useful in the future...