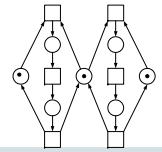
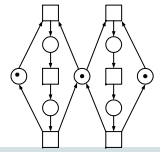


Concurrent Systems Modeling using Petri Nets – Part II

Marlon Dumas

(Based on lecture material by Wil van der Aalst
Eindhoven University of Technology, The Netherlands
http://www,workflowcourse.com)

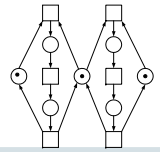


Classical Petri nets - Recap

- Place: passive element
- Transition: active element
- Arc: causal relation
- Token: elements subject to change

The state (space) of a Petri net is a distribution of tokens across its places.

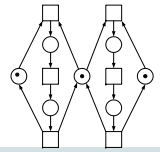
Transition firing move the net from one state to another.



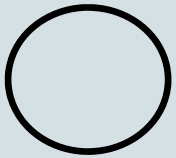
Role of a token

Tokens can play the following roles:

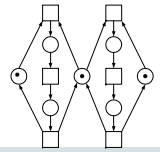
- a **physical object**, for example a product, a part, a drug, a person;
- an **information object**, for example a message, a signal, a report;
- a **collection of objects**, for example a truck with products, a warehouse with parts, or an address file;
- an **indicator of a state**, for example the indicator of the state in which a process is, or the state of an object;
- an **indicator of a condition**: the presence of a token indicates whether a certain condition is fulfilled.



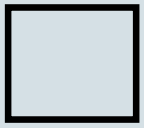
Role of a place



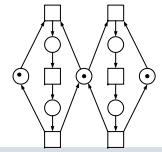
- a type of **communication medium**, like a telephone line, a middleman, or a communication network;
- a **buffer**: for example, a depot, a queue or a post bin;
- a **geographical location**, like a place in a warehouse, office or hospital;
- a possible **state or state condition**: for example, the floor where an elevator is, or the condition that a specialist is available.



Role of a transition

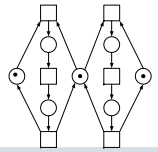


- an **event**: for example, starting an operation, the death of a patient, a change seasons or the switching of a traffic light from red to green;
- a **transformation of an object**, like adapting a product, updating a database, or updating a document;
- a **transport of an object**: for example, transporting goods, or sending a file.

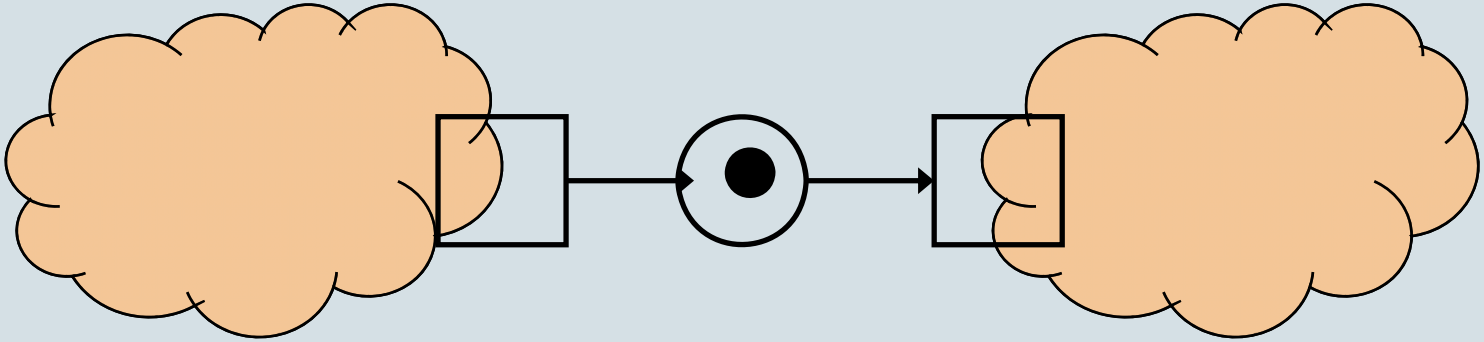


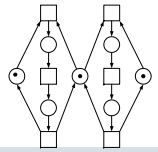
Typical network structures

- Causality
- Parallelism (AND-split - AND-join)
- Choice (XOR-split – XOR-join)
- Iteration (XOR-join - XOR-split)
- Capacity constraints
  - Feedback loop
  - Mutual exclusion
  - Alternating

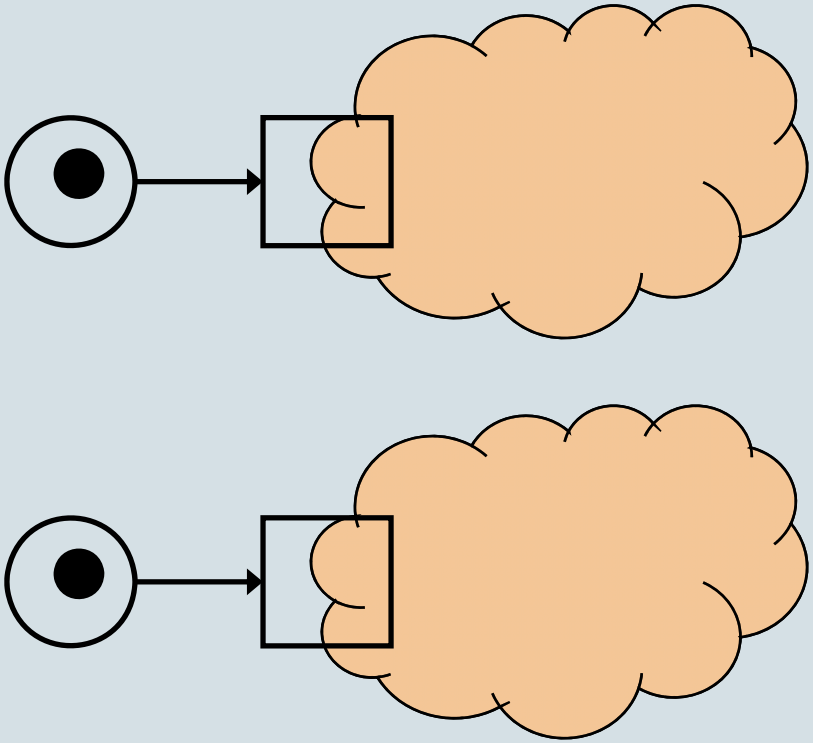


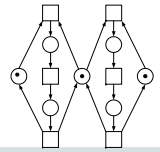
Causality



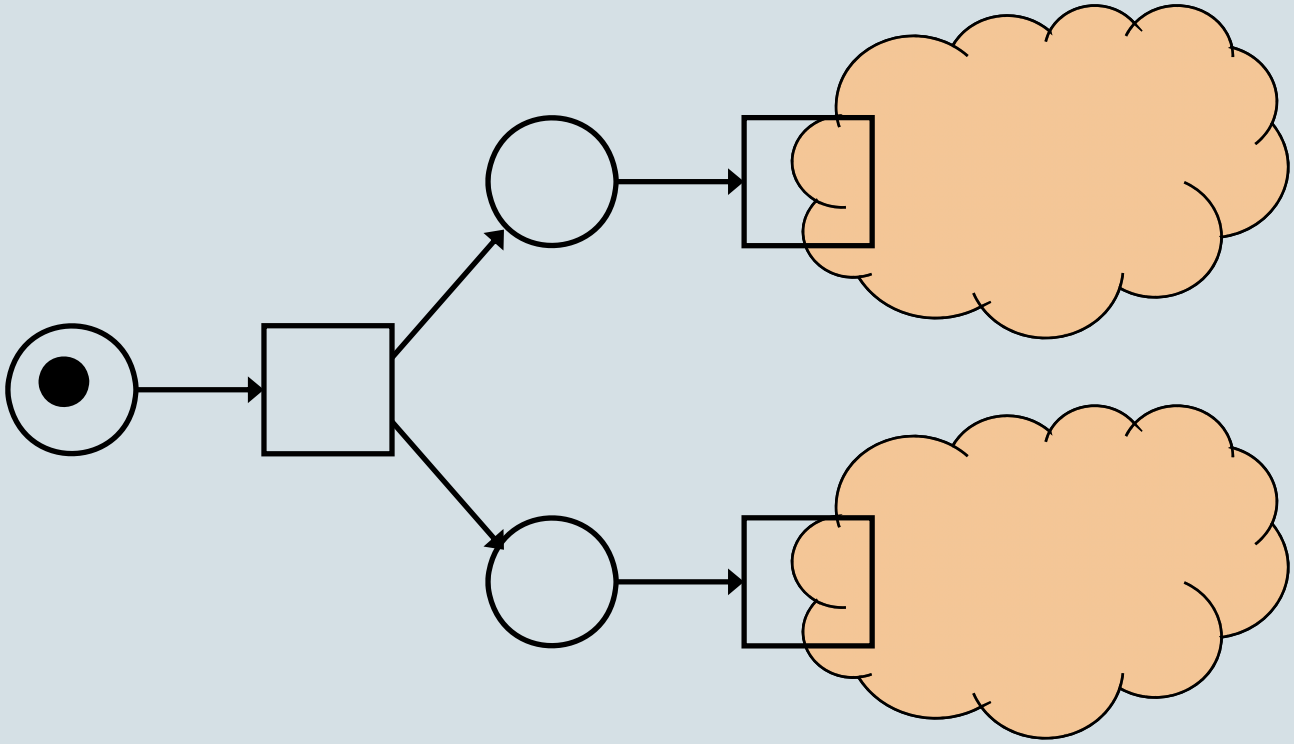


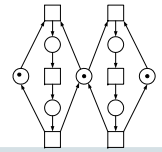
Parallelism



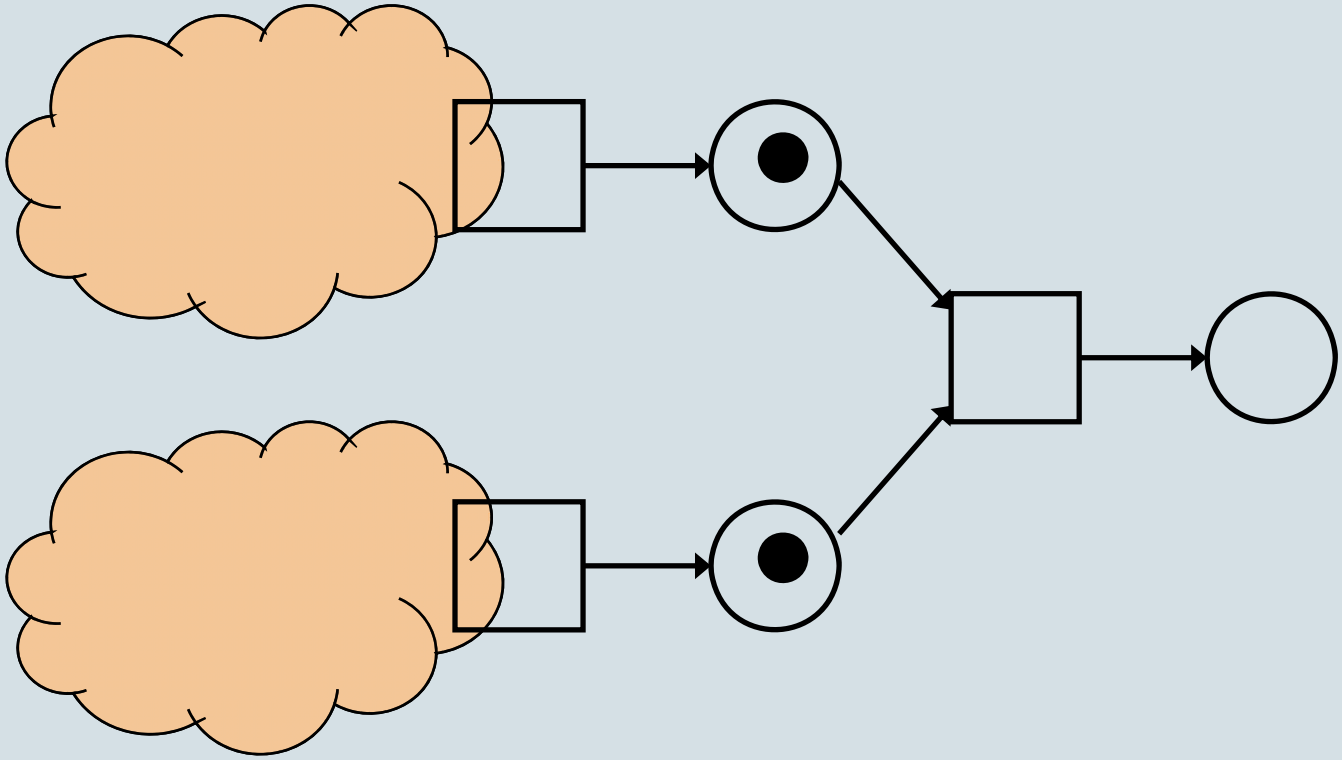


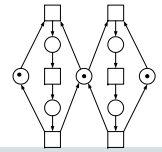
Parallelism: AND-split



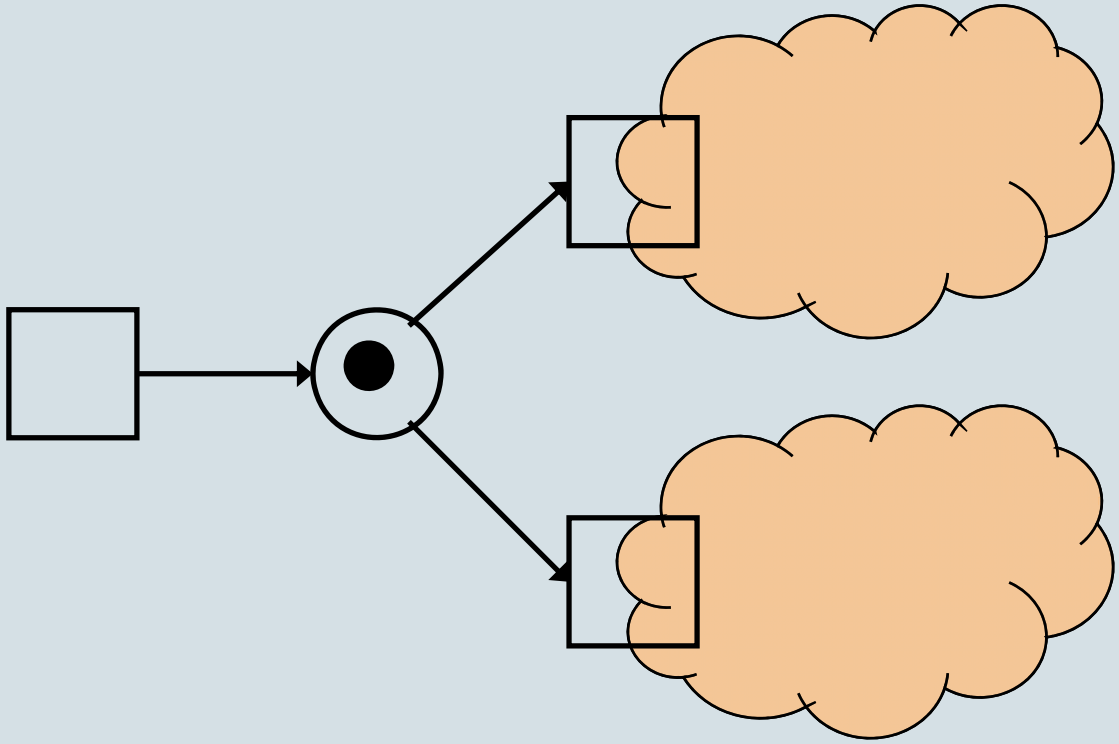


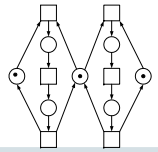
Parallelism: AND-join



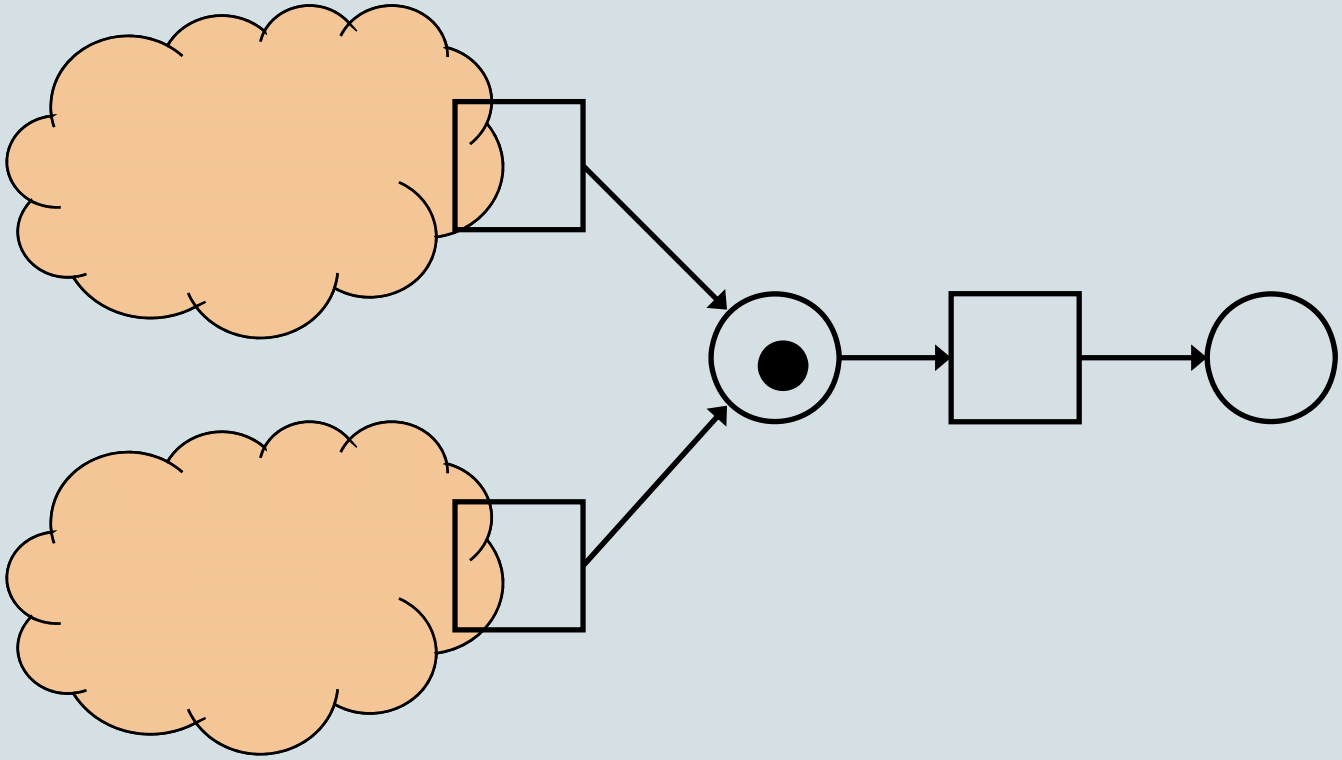


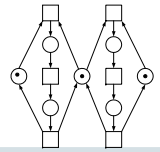
Choice: XOR-split



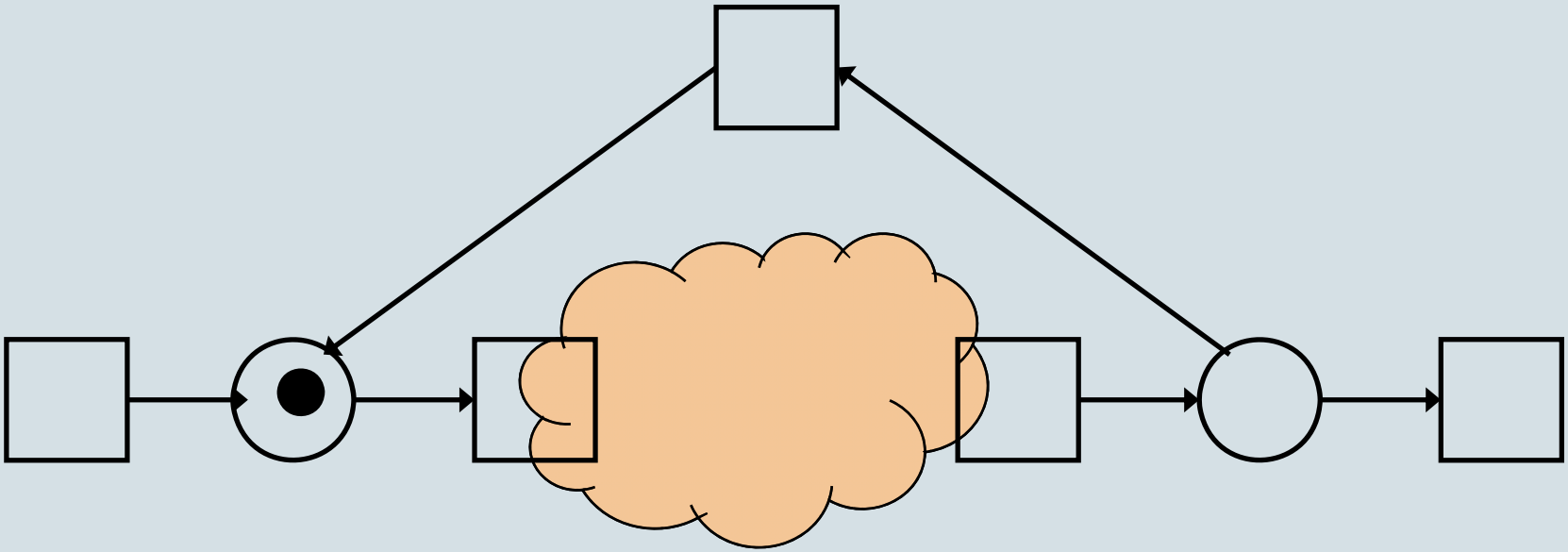


Choice: XOR-join

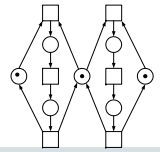




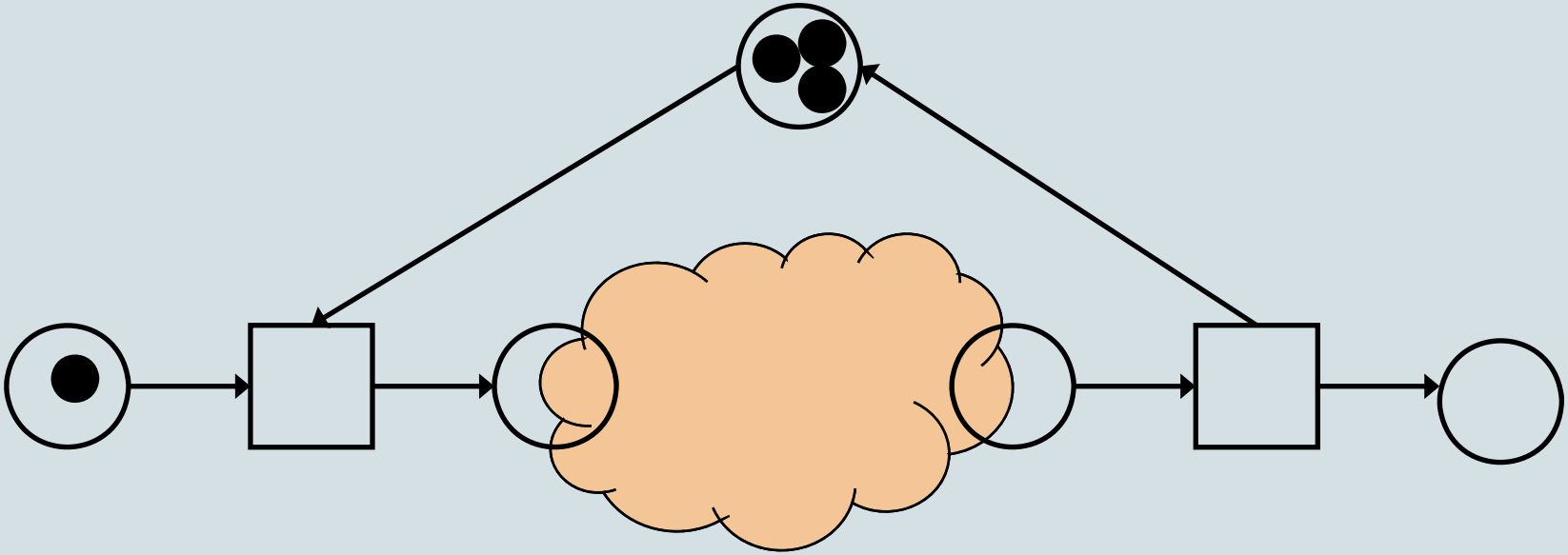
Iteration: 1 or more times



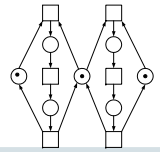
XOR-join before XOR-split



Capacity constraints: feedback loop

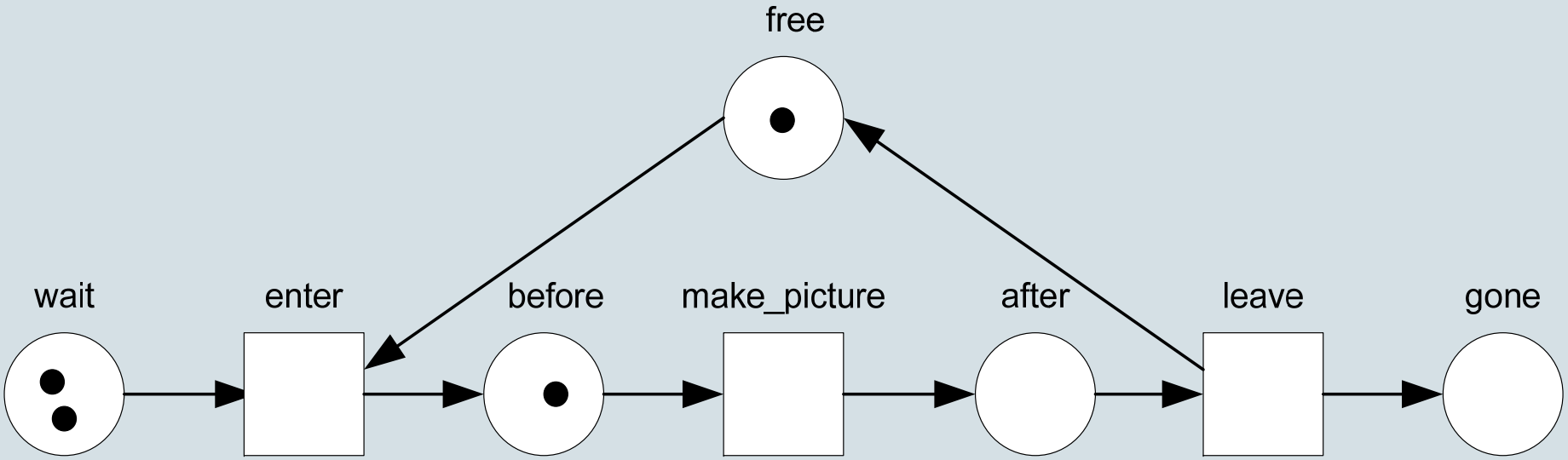


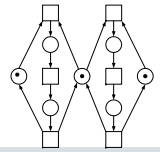
AND-join before AND-split



Capacity constraint: example

The room has a capacity of 2, 1 person is in the room

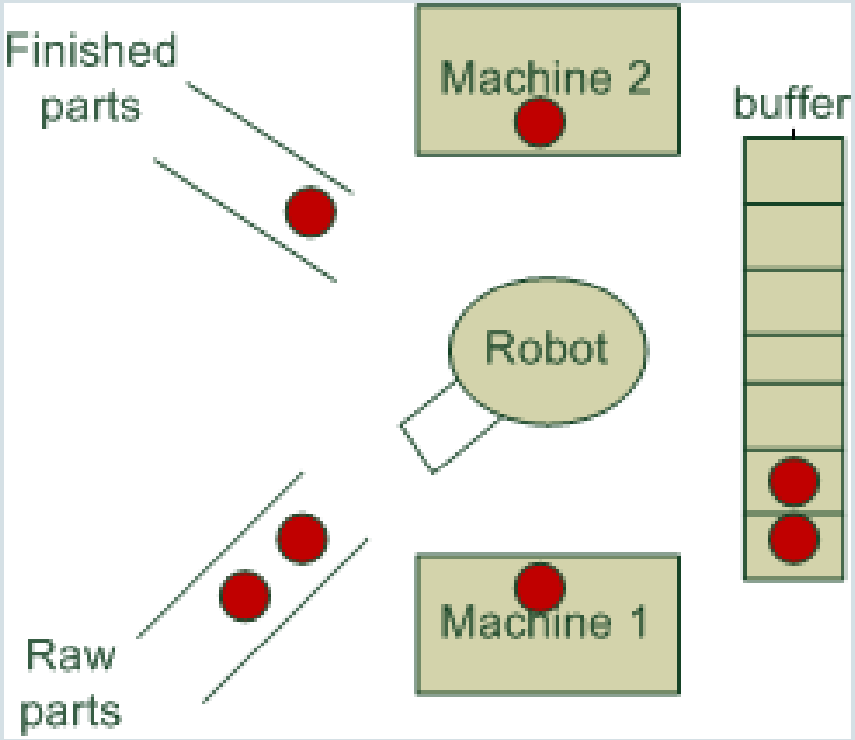




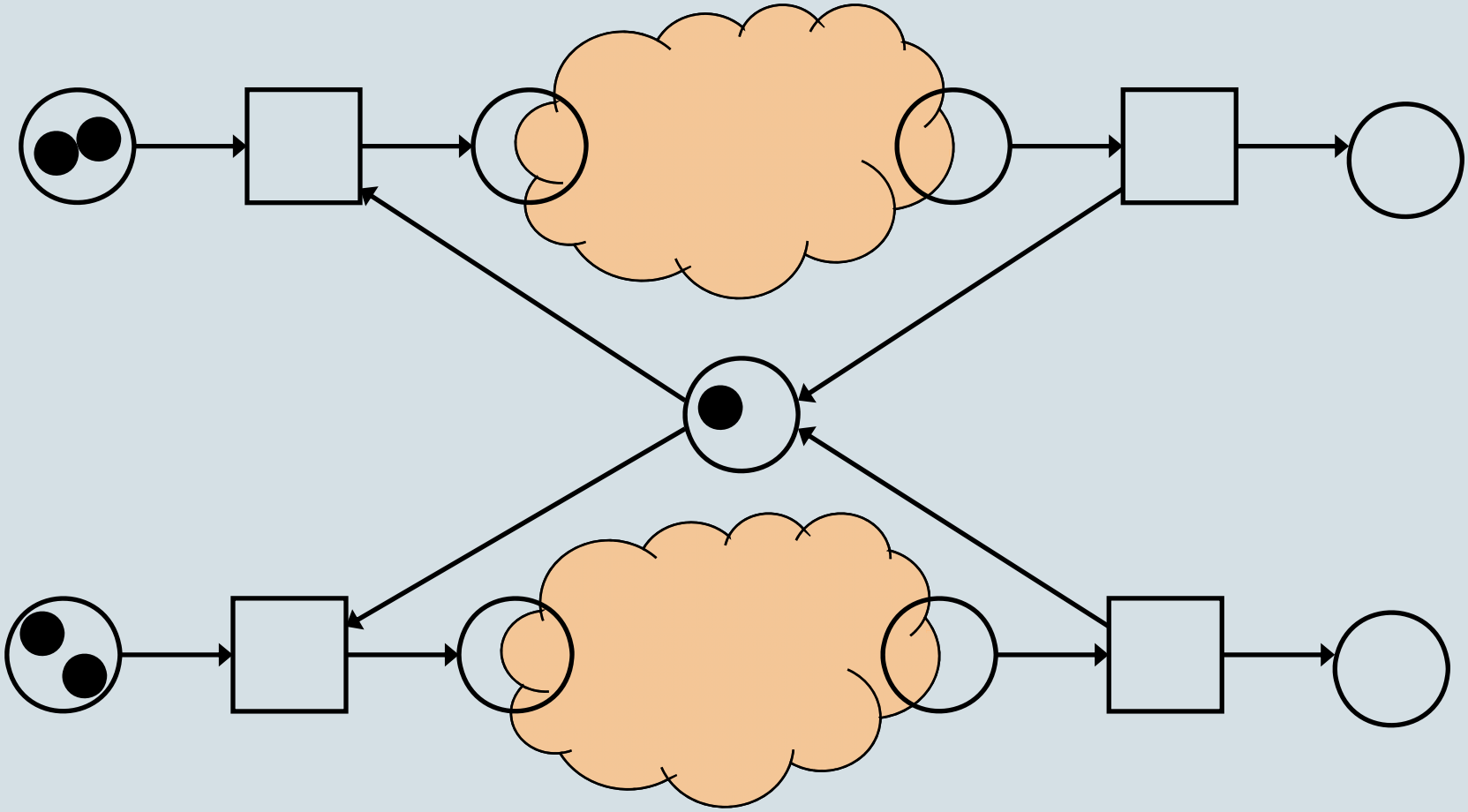
Inspired by an exercise of Prof. R. Wattenhofer, ETH Zürich

Capacity constraint: exercise

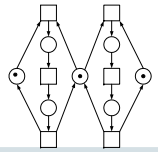
- Pipelined factory
- Two machines, one robot, one buffer
- Robot moves parts from raw line to M1 to buffer, to M2 and finally to finished line
- Buffer can hold maximum 7 parts



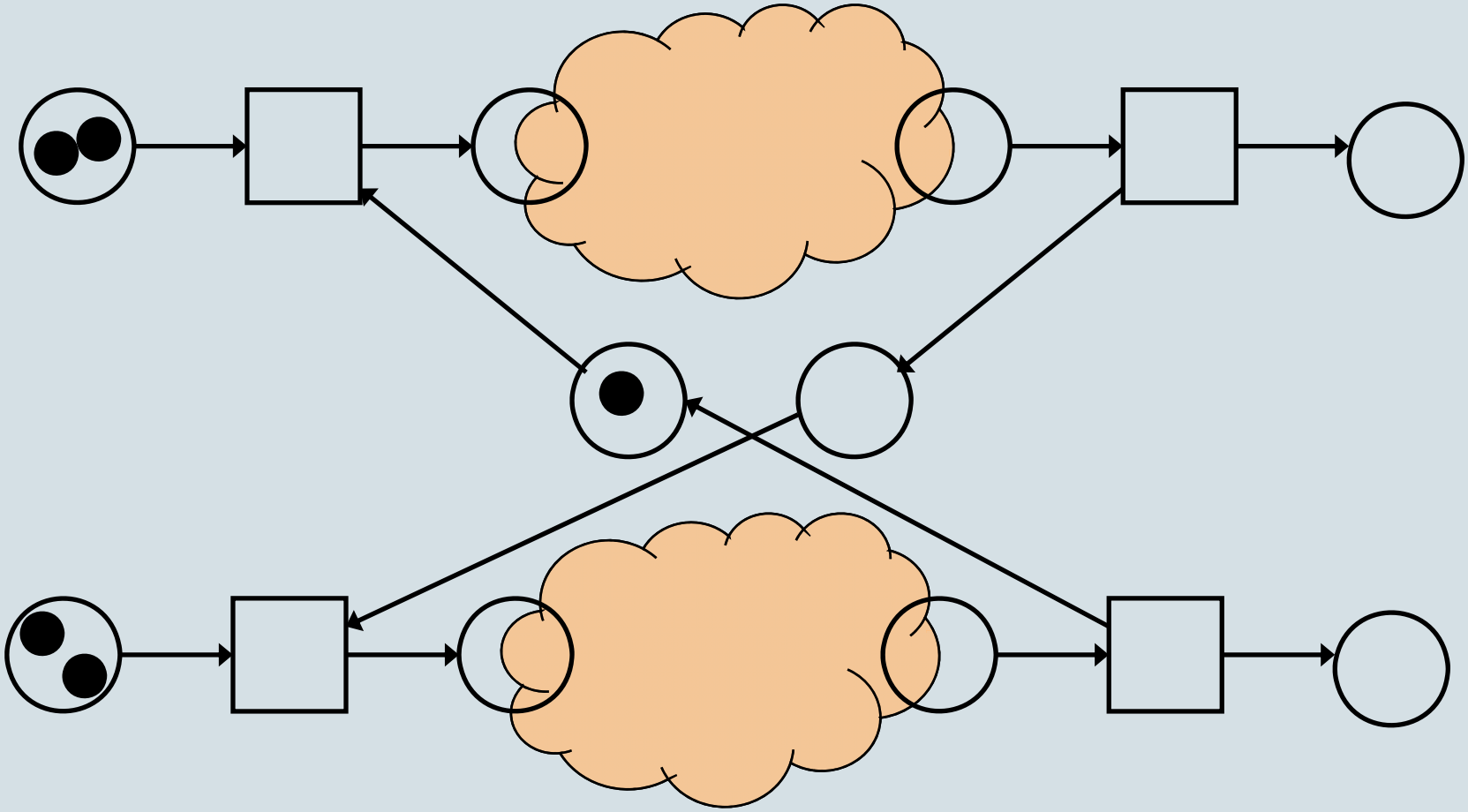
Capacity constraints: mutual exclusion



AND-join before AND-split

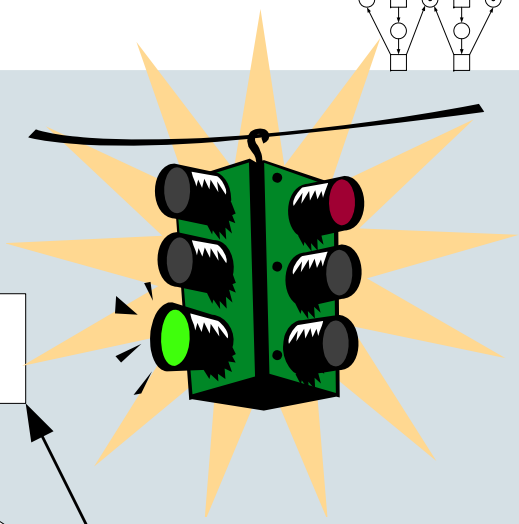


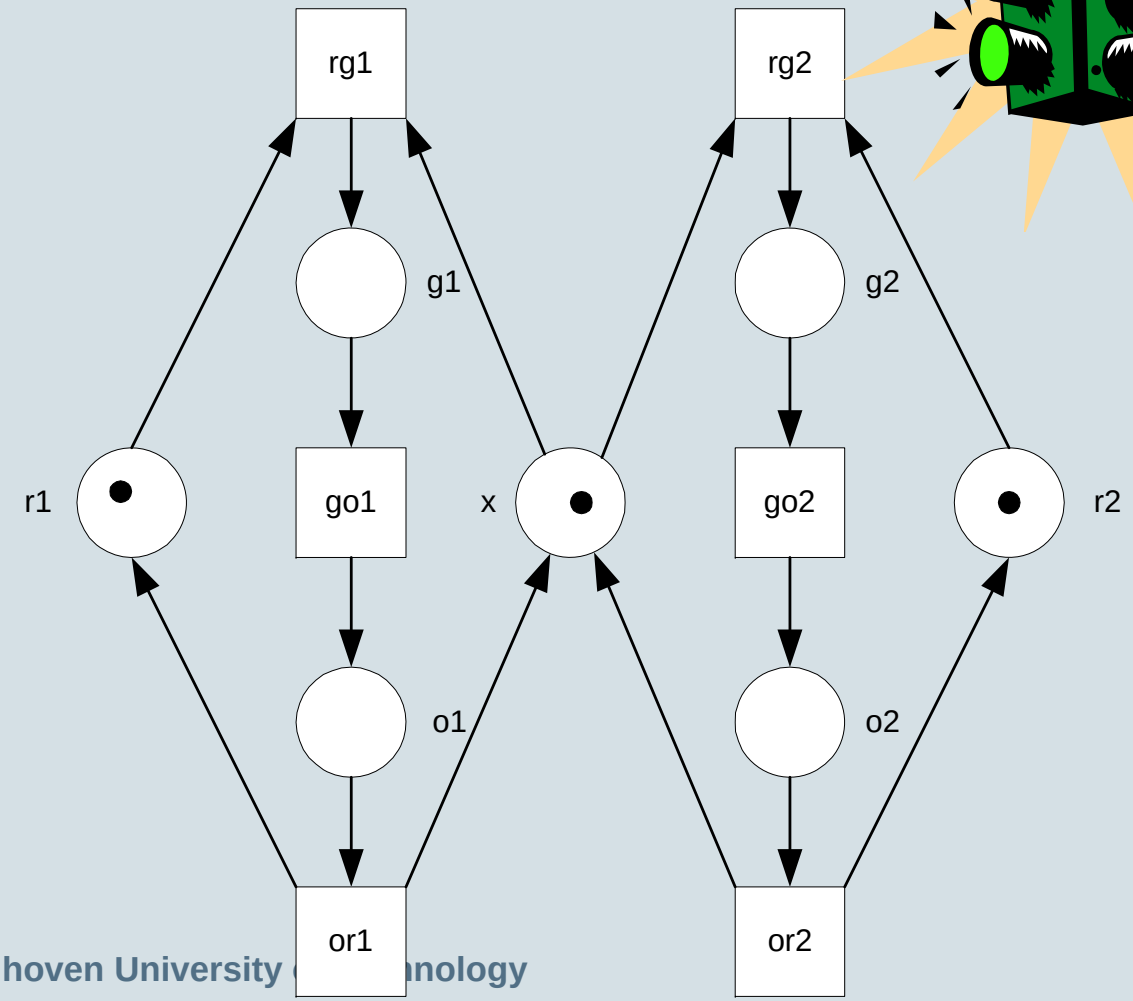
Capacity constraints: alternating



AND-join before AND-split

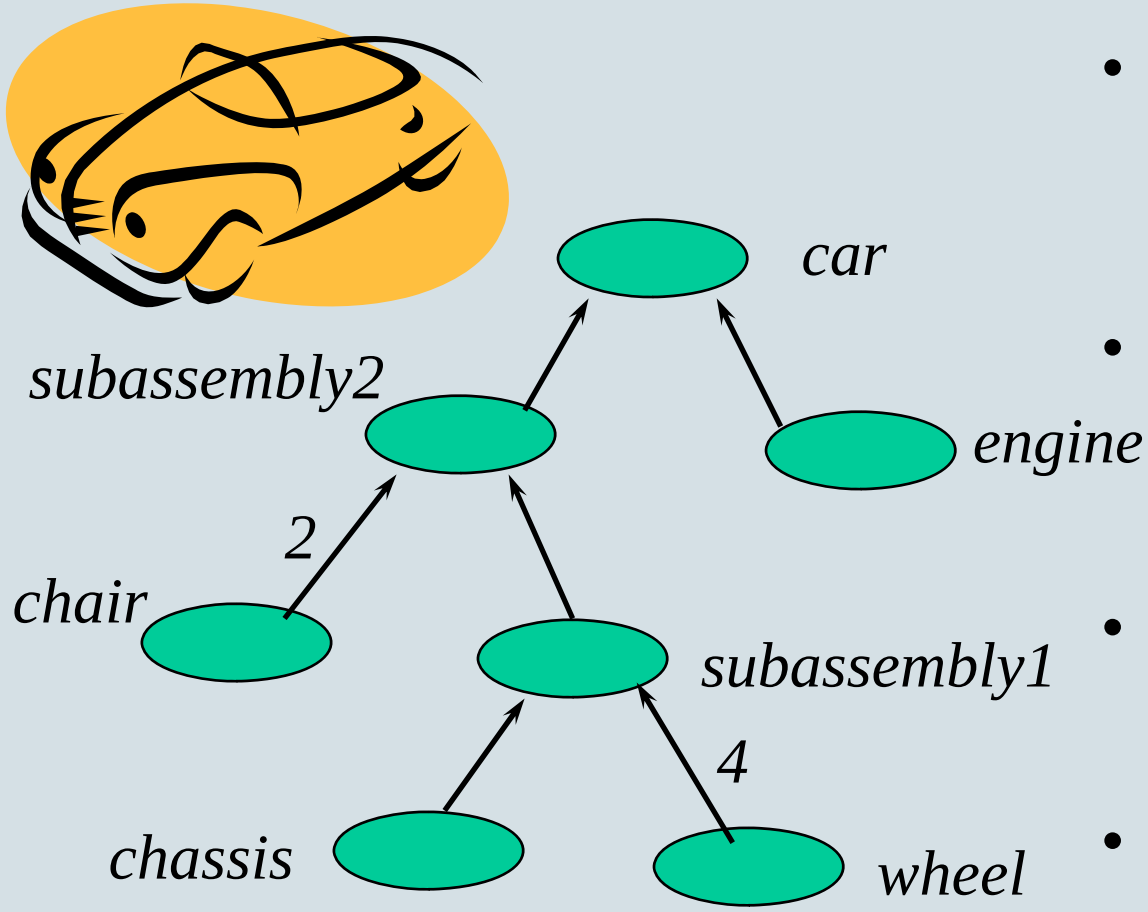
Mutual exclusion pattern: example



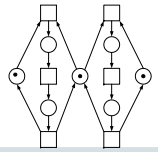


How to make them alternate?

Exercise: Manufacturing a car (cont.)

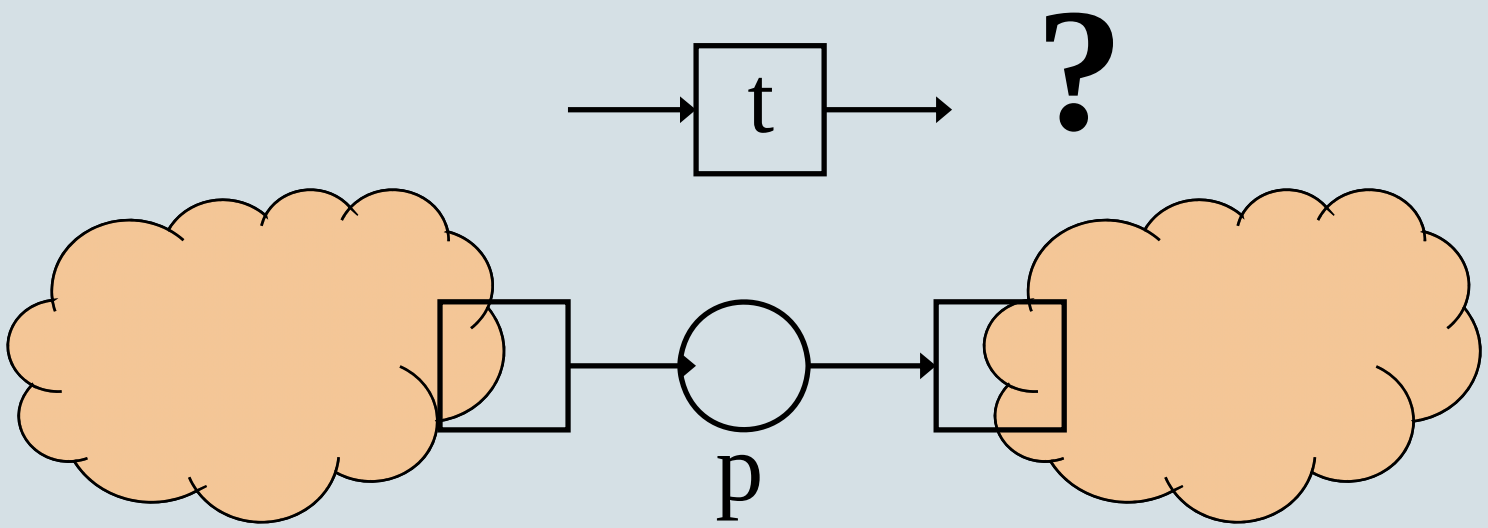


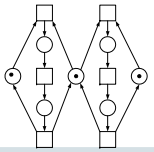
- Model the production process shown in the Bill-Of-Materials **with resources**.
- Each assembly step requires a dedicated machine and an operator.
- There are two operators and one machine of each type.
- Capture both the start and completion of each assembly step.



Modeling problem (1): Zero testing

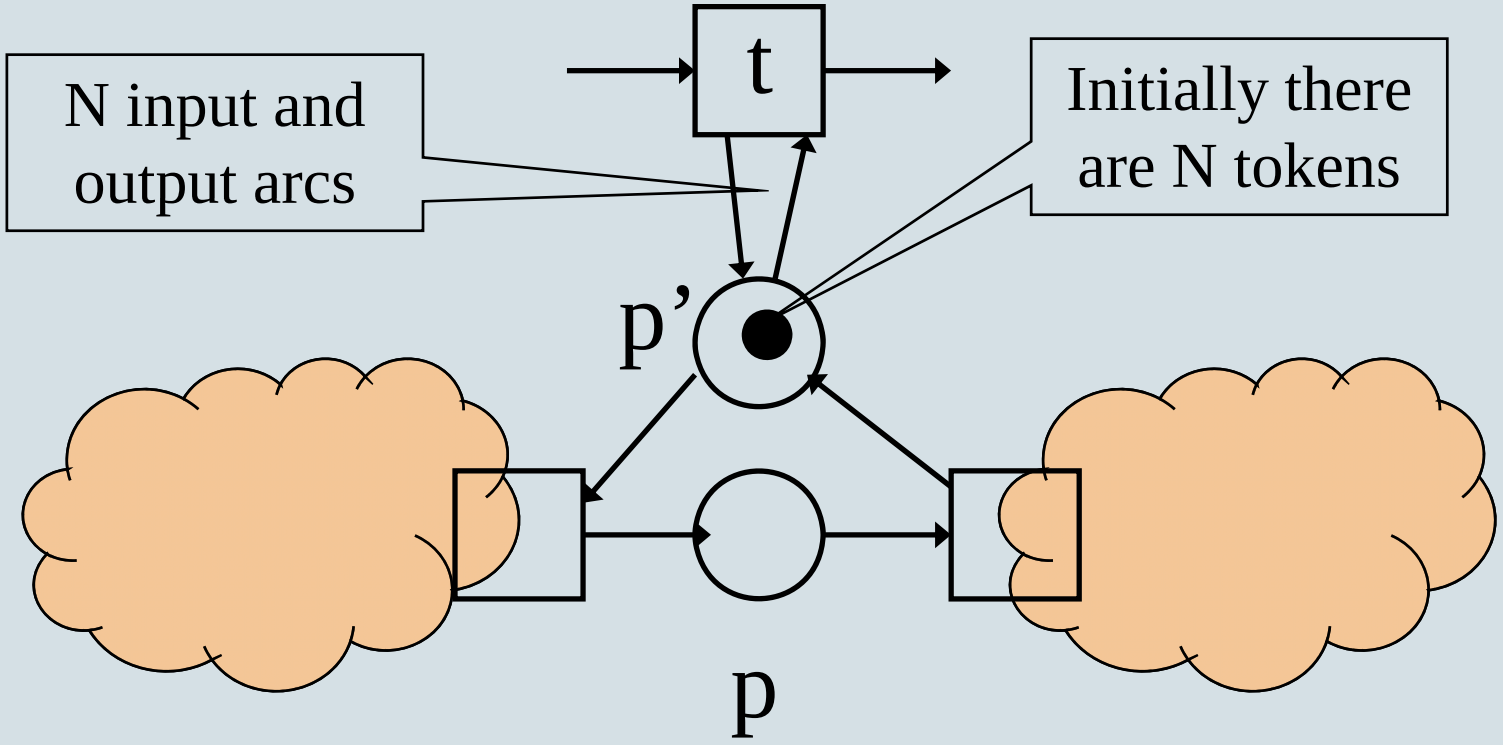
- Transition *t* should fire if place *p* is empty.





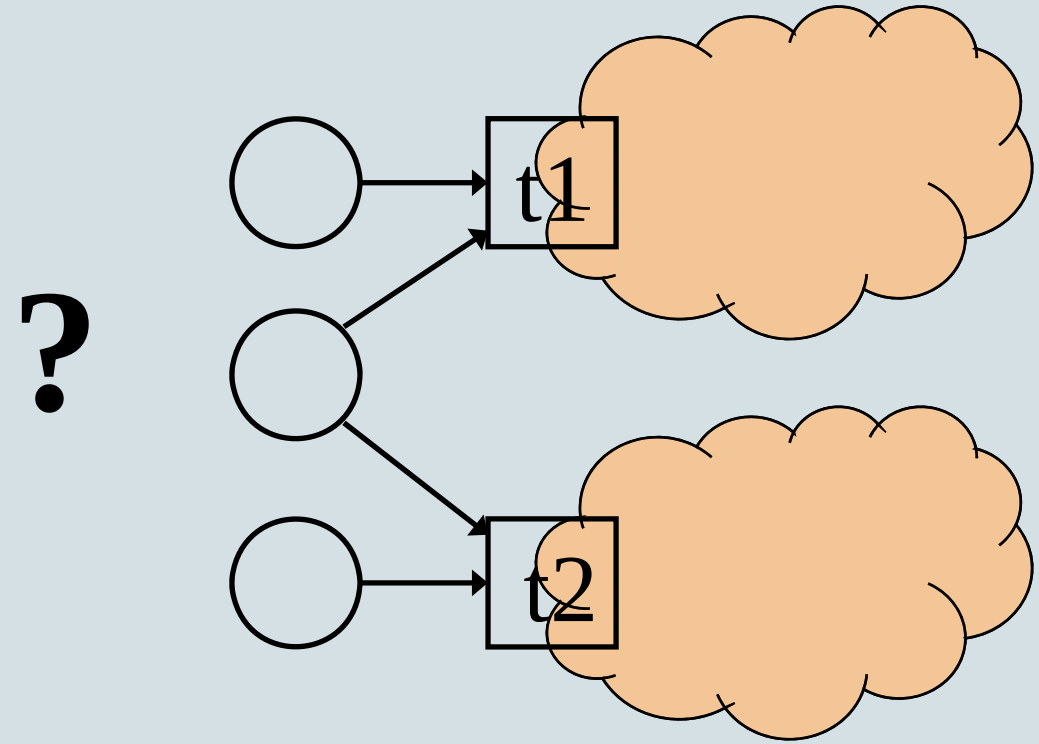
Solution

- Only works if place is N-bounded

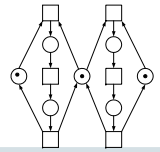


Modeling problem (2): Priority

- Transition t1 has priority over t2



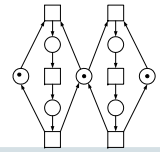
Hint: similar to Zero testing!



Theoretical Results

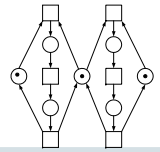
- Extensions have been proposed to tackle these problems, e.g., inhibitor arcs.
- These extensions extend the modeling power (Turing completeness*).
- Without such an extension not Turing complete.
- Still certain questions are difficult/expensive to answer or even undecidable (e.g., equivalence of two nets).

* Turing completeness corresponds to the ability to execute any computation.



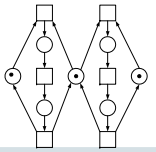
Exercise: Witness statements

- As part of the process of handling insurance claims, we need to handle witness statements.
- There may any number of witnesses per claim. The number is determined when the claim is lodged. After an initialization step (one per claim), each of the witnesses is registered and contacted (N witnesses per claim in parallel). Only after all witnesses have contacted a report is made (one report per claim).
- Model this in terms of a Petri net.

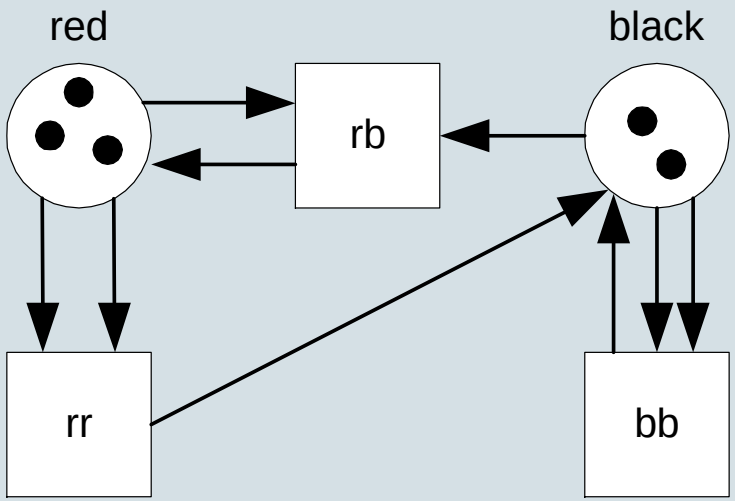


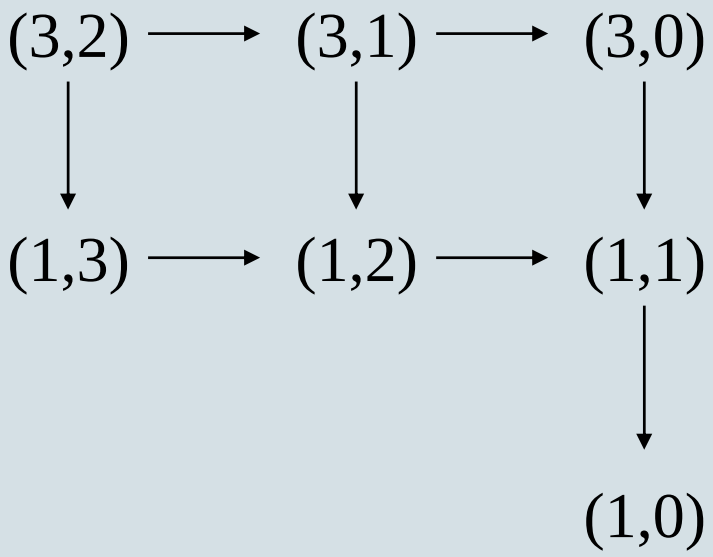
Quick Intro to Petri net analysis: **Reachability Graph**

- Graph containing one node for each reachable state; an edge indicates that the system can move from the source state to the target state through one transition firing
- The reachability graph can be calculated as follows:
    1. Let X be the set containing just the initial state and let Y be the empty set.
    2. Take an element x of X and add this to Y. Calculate all states reachable for x by firing some enabled transition. Each successor state that is not in Y is added to X.
    3. If X is empty stop, otherwise goto 2.



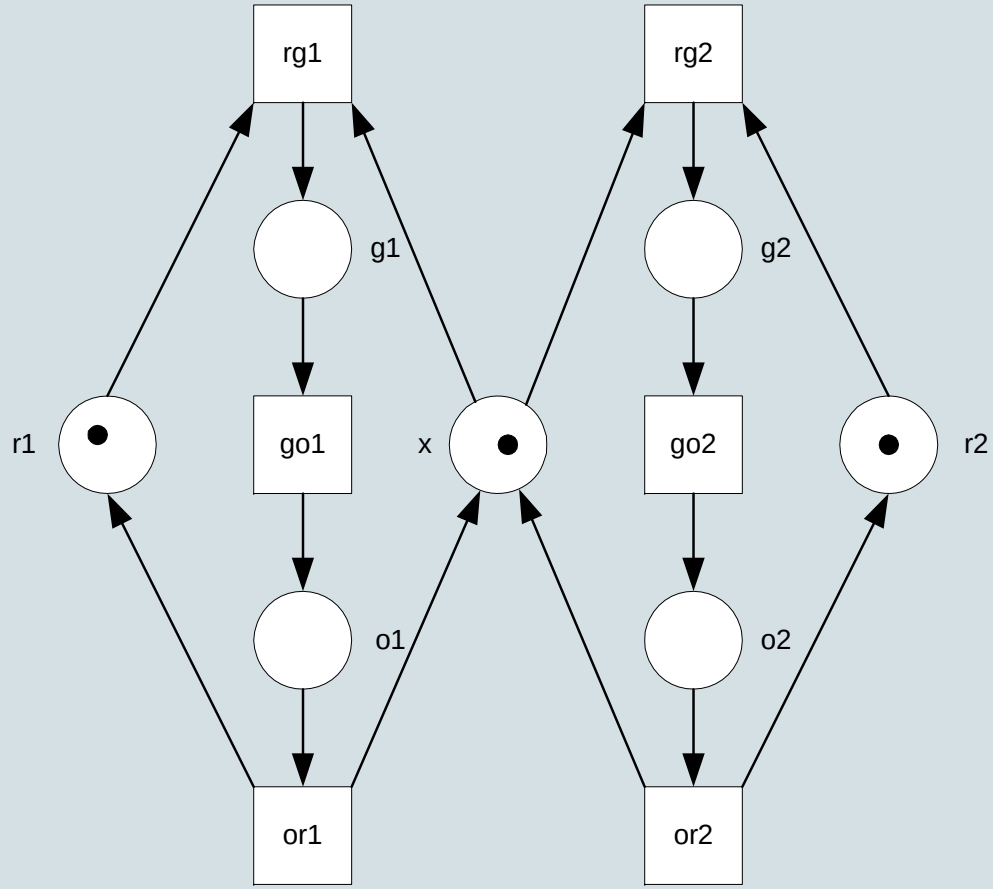
Example

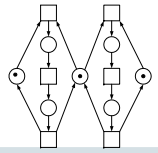




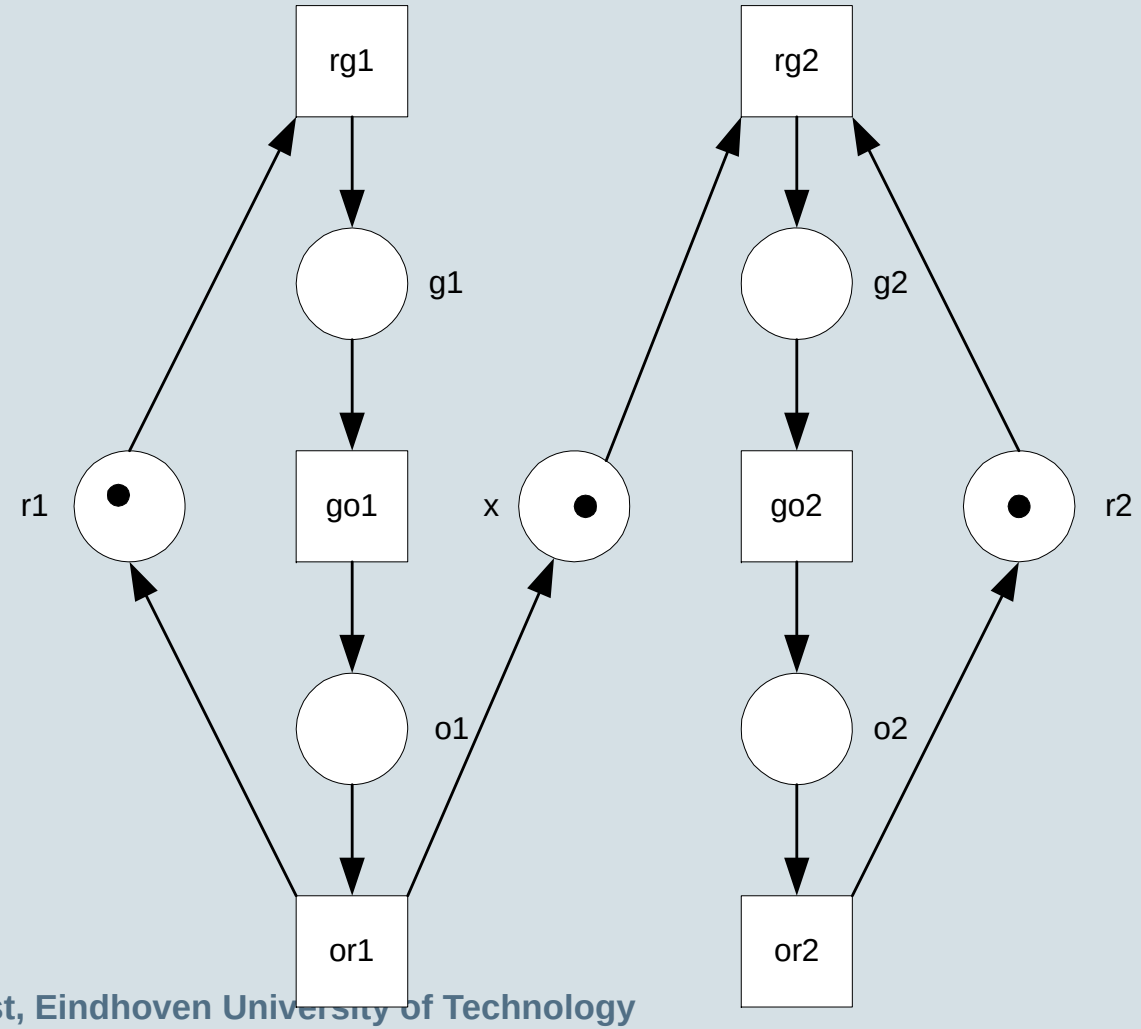
Nodes in the reachability graph can be represented by a vector “(3,2)” or as “3 red + 2 black”. The latter is useful for “sparse states” (i.e., few places are marked).

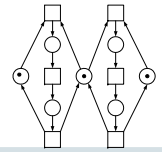
Exercise: Give the reachability graph using both notations





Infinite reachability graph

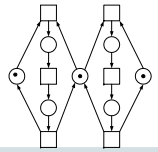




Different types of states

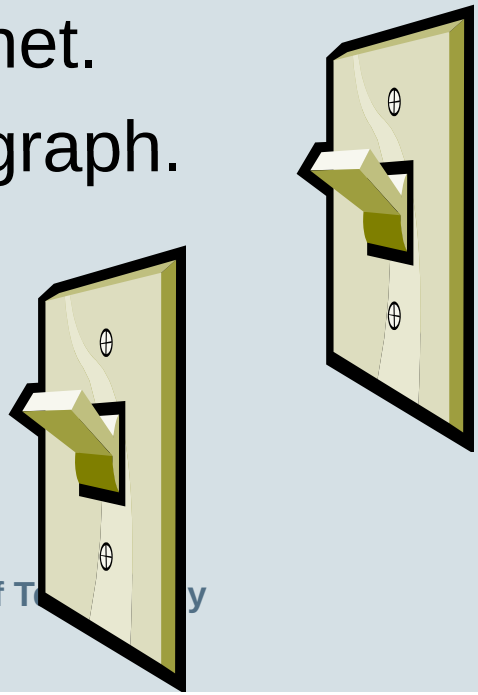
- **Initial state**: Initial distribution of tokens.
- **Reachable state**: Reachable from initial state.
- **Final state** (also referred to as “dead states”): No transition is enabled.
- **Home state** (also referred to as home marking): It is always possible to return (i.e., it is reachable from any reachable state).

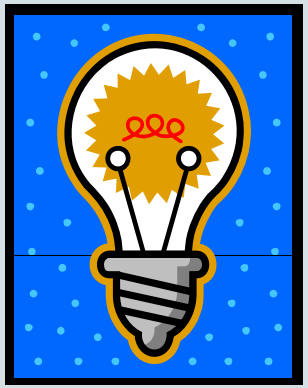
How to recognize these states in the reachability graph?

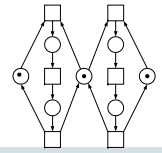


Exercise: Two switches

- Consider a room with two switches and one light. The light is **on** or **off**. The switches are in state **up** or **down**. At any time any of the switches can be used to turn the light on or off.
- Model this as a Petri net.
- Give the reachability graph.

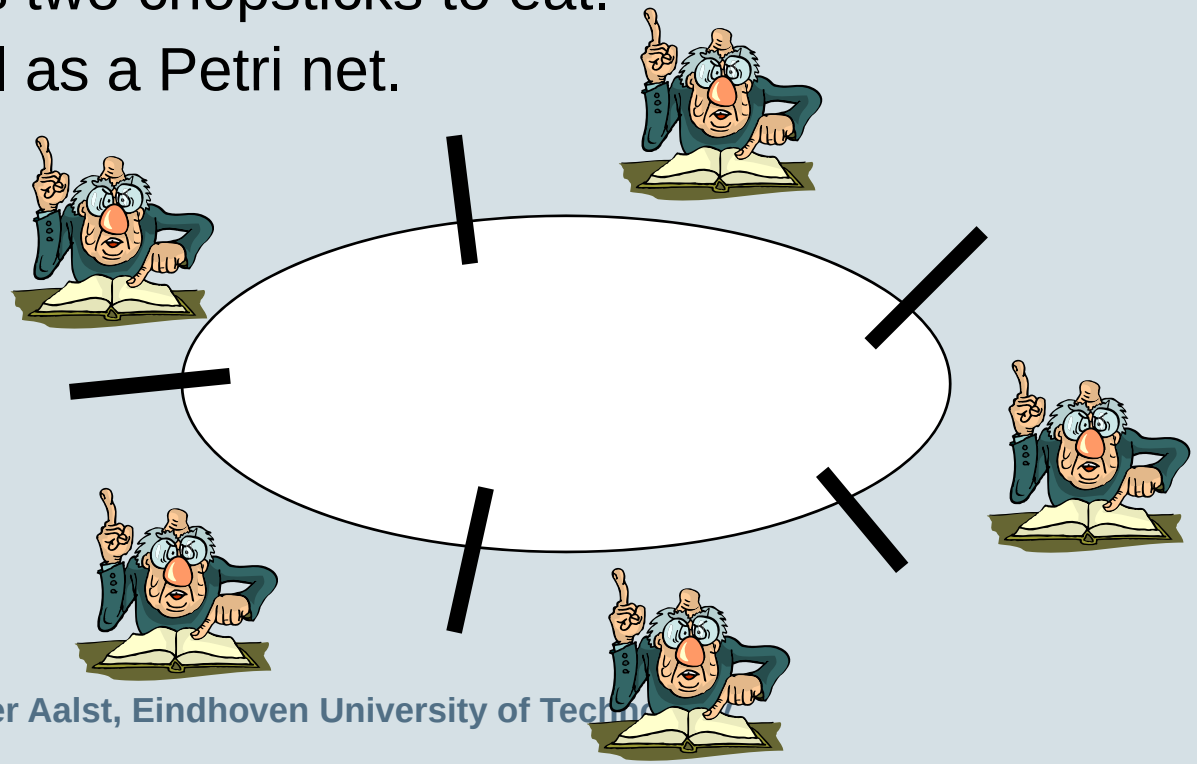


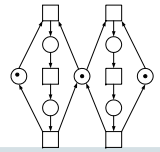




Exercise: Dining philosophers (basic version)

- 5 philosophers sharing 5 chopsticks: chopsticks are located in-between philosophers
- A philosopher is either in state eating or thinking and needs two chopsticks to eat.
- Model as a Petri net.





Want to know more about Petri nets?

- Lecture on Petri nets by Karsten Wolf (University of Rostock) at the Basoti Summer School, 7-21 August 2009

  http://www.ief.uni-rostock.de/?id=basoti
- Lecture on Colored Petri nets by Kurt Jensen (Aarhus University) at the Estonian Summer School in Computer Science, 24-28 August 2009