# Petri nets

## Classical Petri nets: The basic model

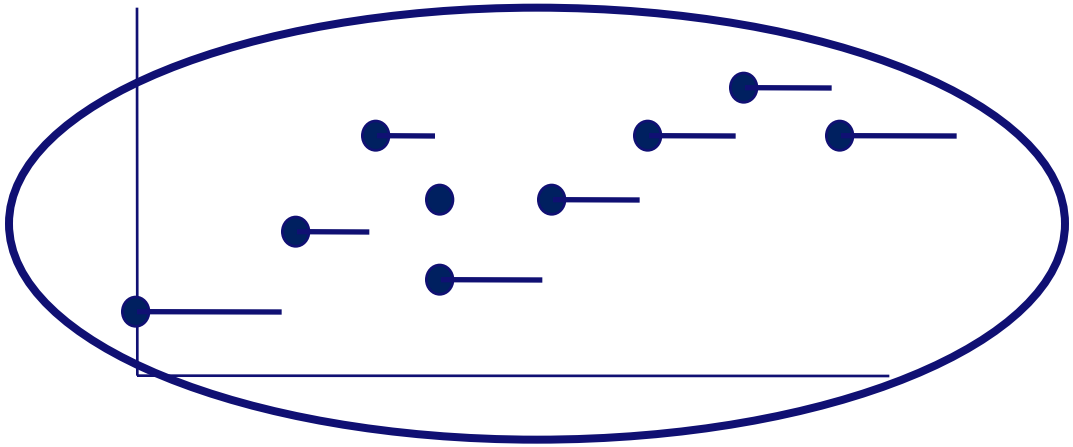
prof.dr.ir. Wil van der Aalst

TU/e Technische Universiteit Eindhoven University of Technology

Where innovation starts

# Process modeling

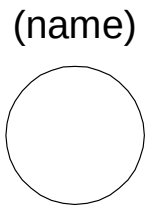
- Emphasis on dynamic behavior rather than structuring the state space
- Transition system is too low level
- We start with the classical Petri net
- Then we extend it with:
  - Color
  - Time
  - Hierarchy



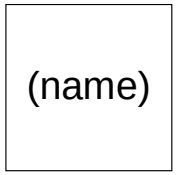
# Classical Petri net

- **Simple process model**
  - **Just three elements: places, transitions and arcs.**
  - **Graphical and mathematical description.**
  - **Formal semantics and allows for analysis.**
- **History:**
  - **Carl Adam Petri (1926-2010)**
    - **PhD thesis 1962, start of concurrency research**
    - **e.g. Commandeur in de Orde van de Nederlandse Leeuw**
  - **In sixties and seventies focus mainly on theory.**
  - **Since eighties also focus on tools and applications (cf. CPN work by Kurt Jensen).**
  - **“Hidden” in many diagramming techniques and systems (“token game semantics”).**

# Elements



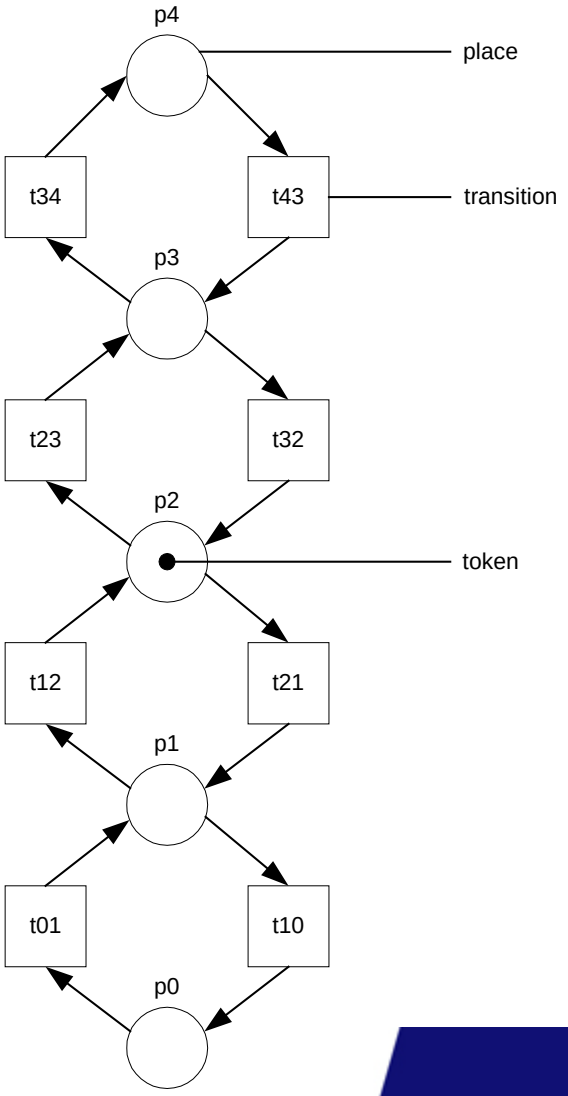

place



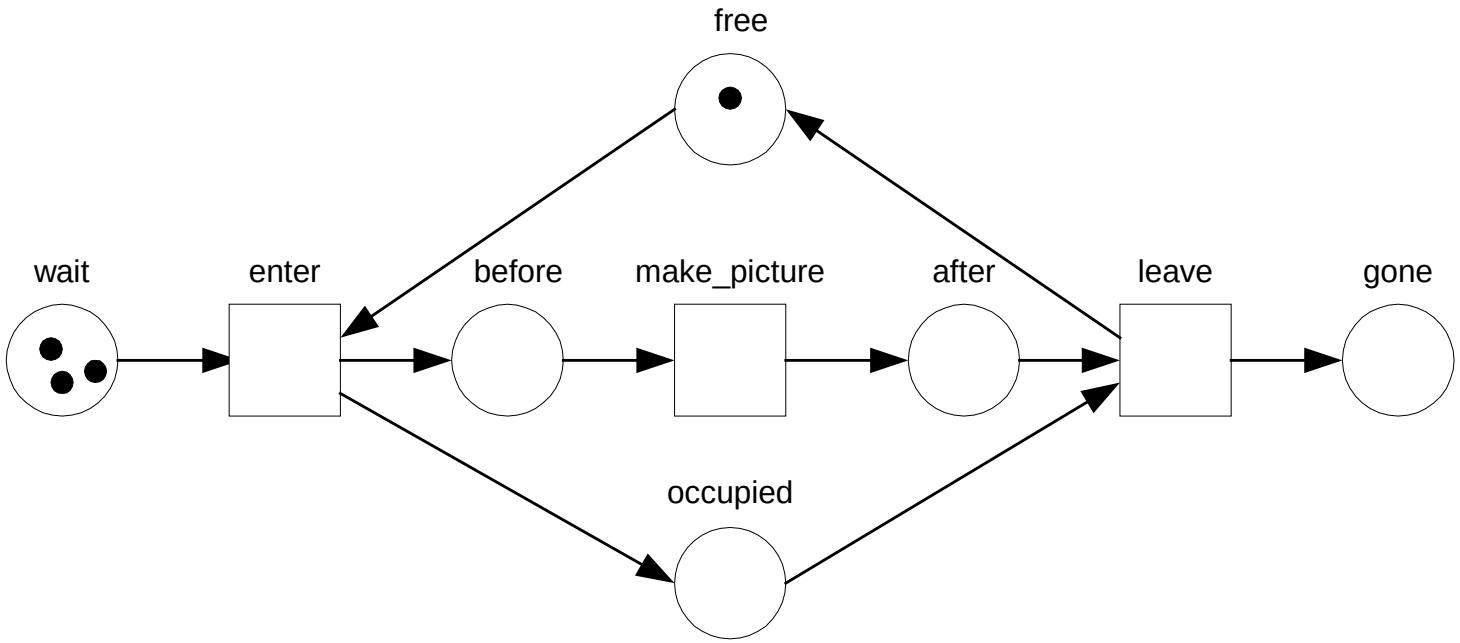

transition

arc (directed connection)

token



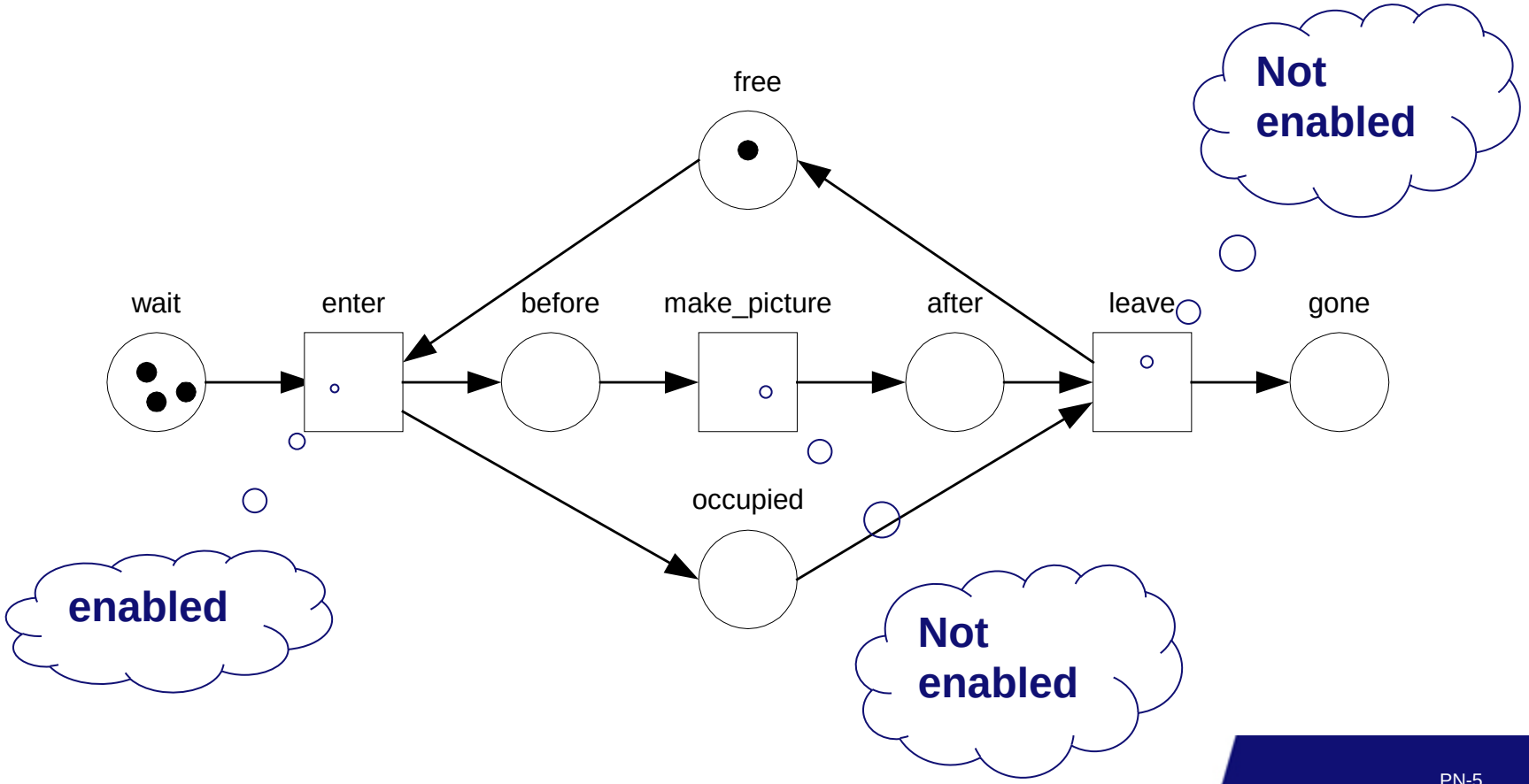
# Rules




- **Connections are directed.**
- **No connections between two places or two transitions.**
- **Places may hold zero or more tokens.**
- **First, we consider the case of at most one arc between two nodes.**

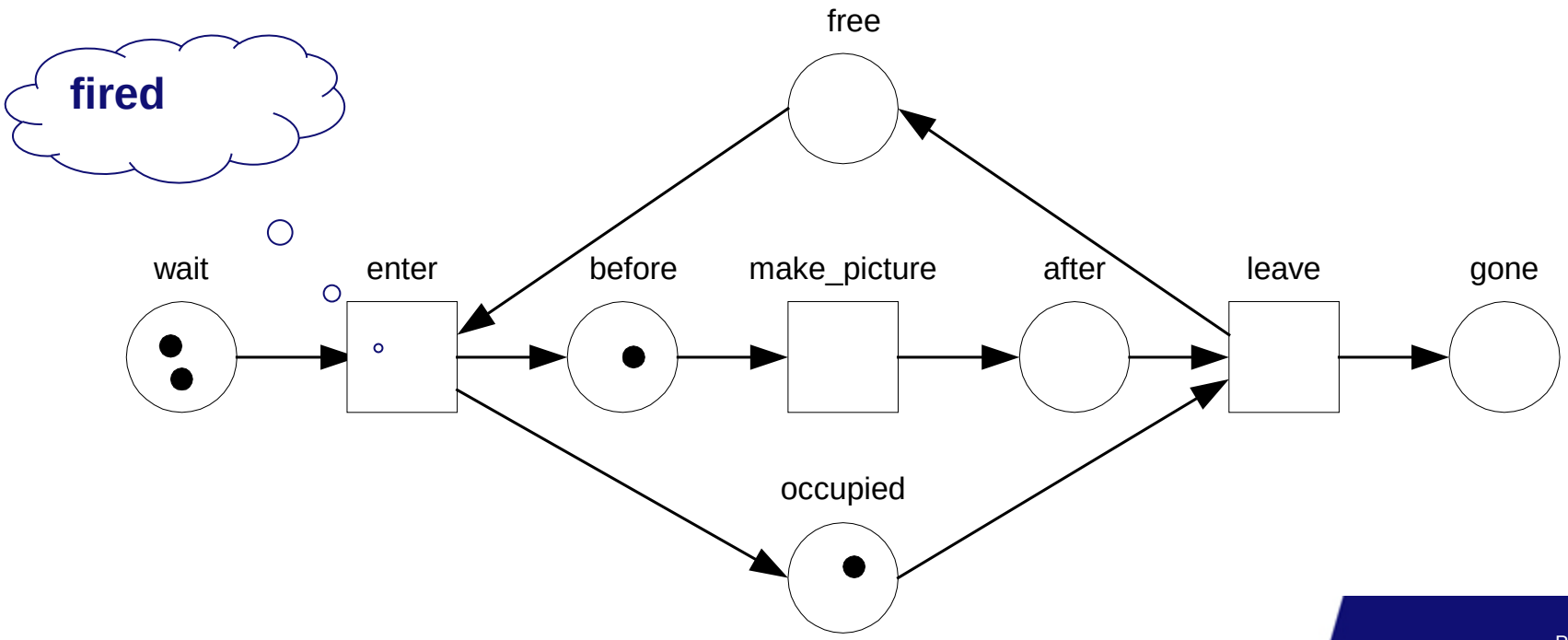
# Enabled

- **A transition is enabled if each of its input places contains at least one token.**



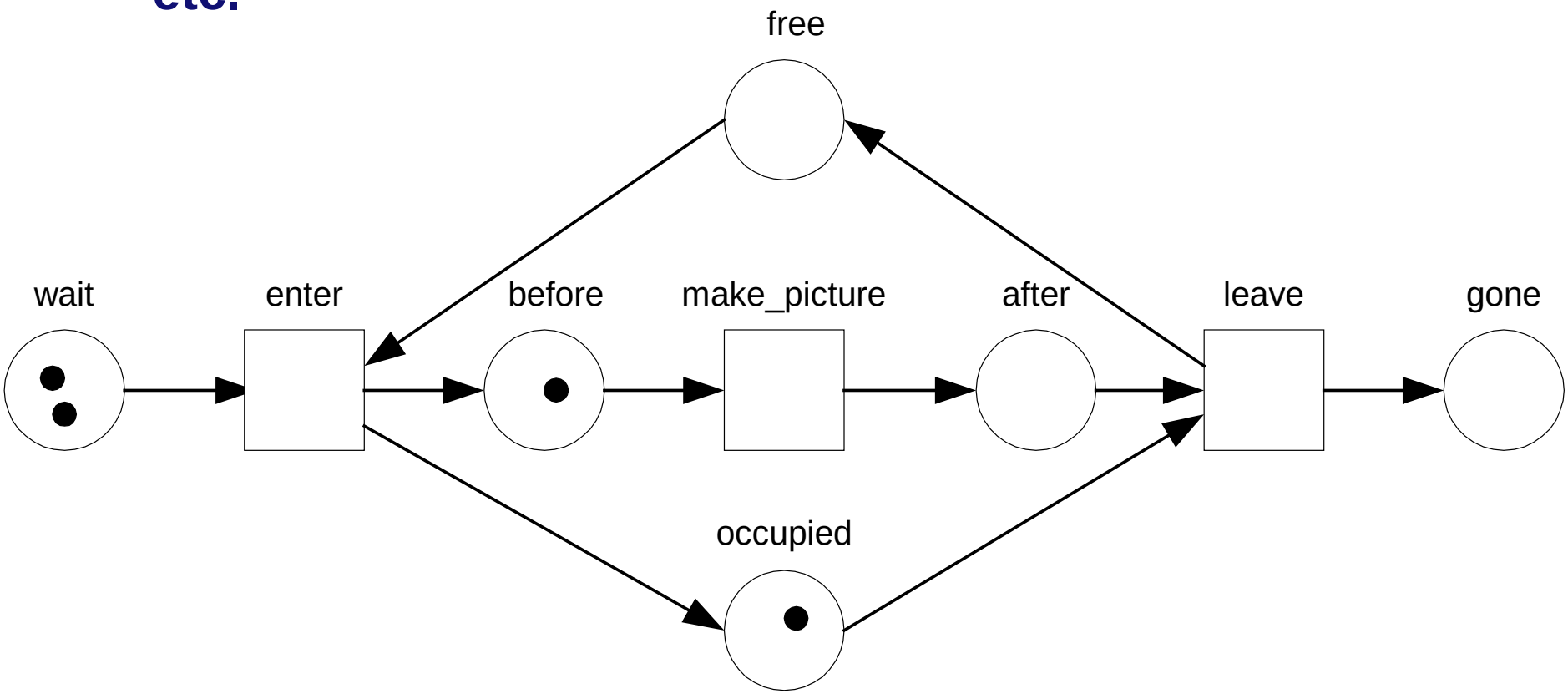
# Firing

- **An enabled transition can fire (i.e., it occurs).**
- **When it fires it consumes a token from each input place and produces a token for each output place.**



# Play “Token Game”

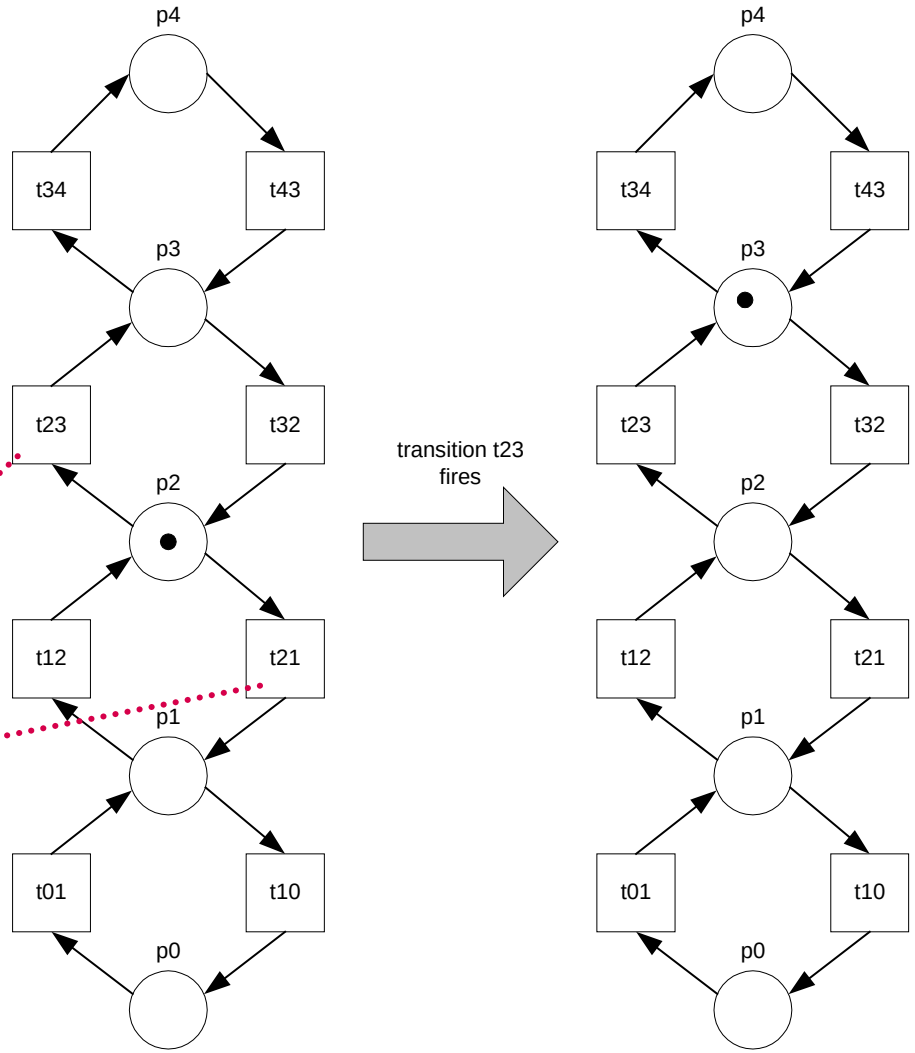
- **In the new state, *make_picture* is enabled. It will fire, etc.**



# Remarks

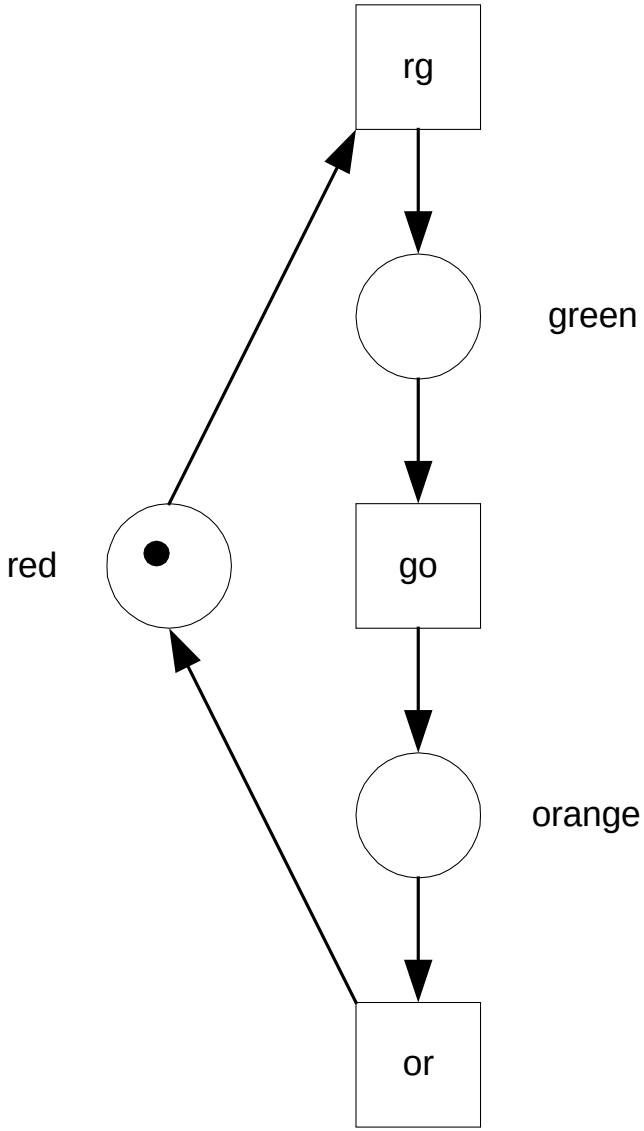
- **Firing is atomic.**
- **Multiple transitions may be enabled, but only one fires at a time, i.e., we assume interleaving semantics (cf. diamond rule).**
- **The number of tokens may vary if there are transitions for which the number of input places is not equal to the number of output places.**
- **The network is static.**
- **The state is represented by the distribution of tokens over places (also referred to as marking).**

# Non-determinism

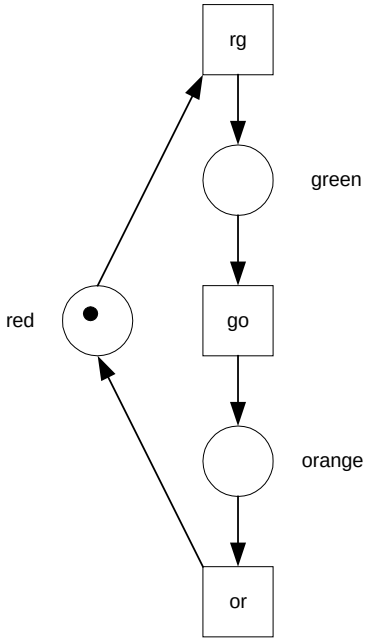


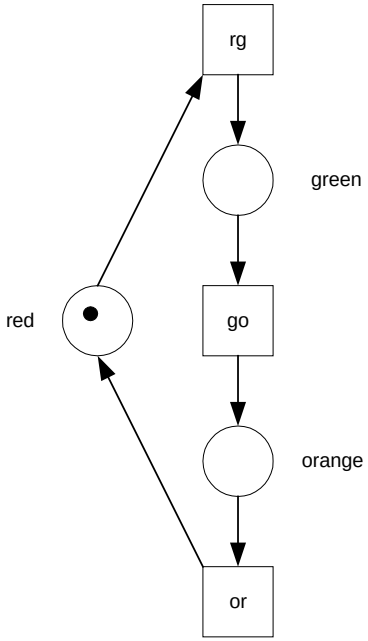

**Two transitions are enabled but only one can fire**

# Example: Single traffic light

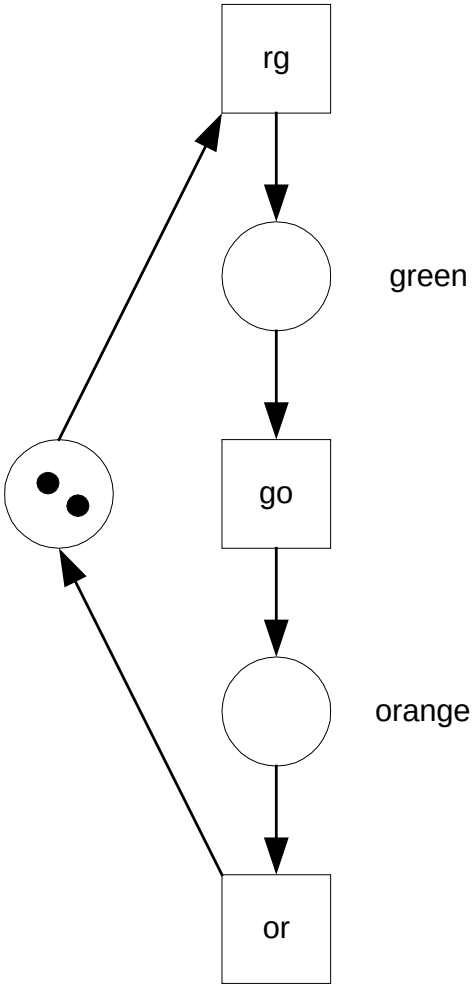


# Two traffic lights

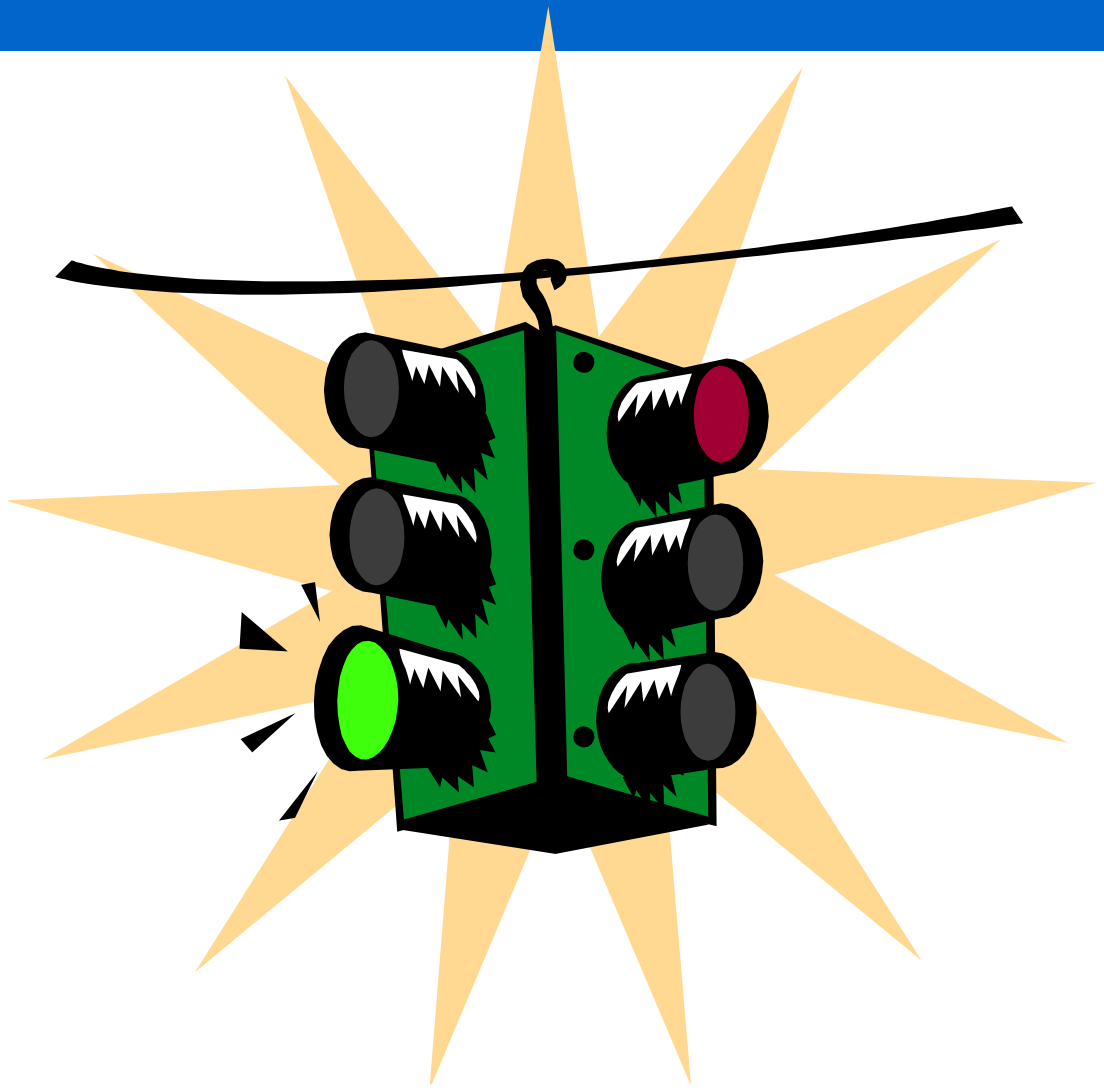


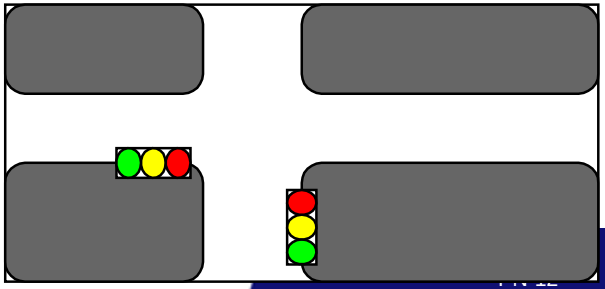




**OR**

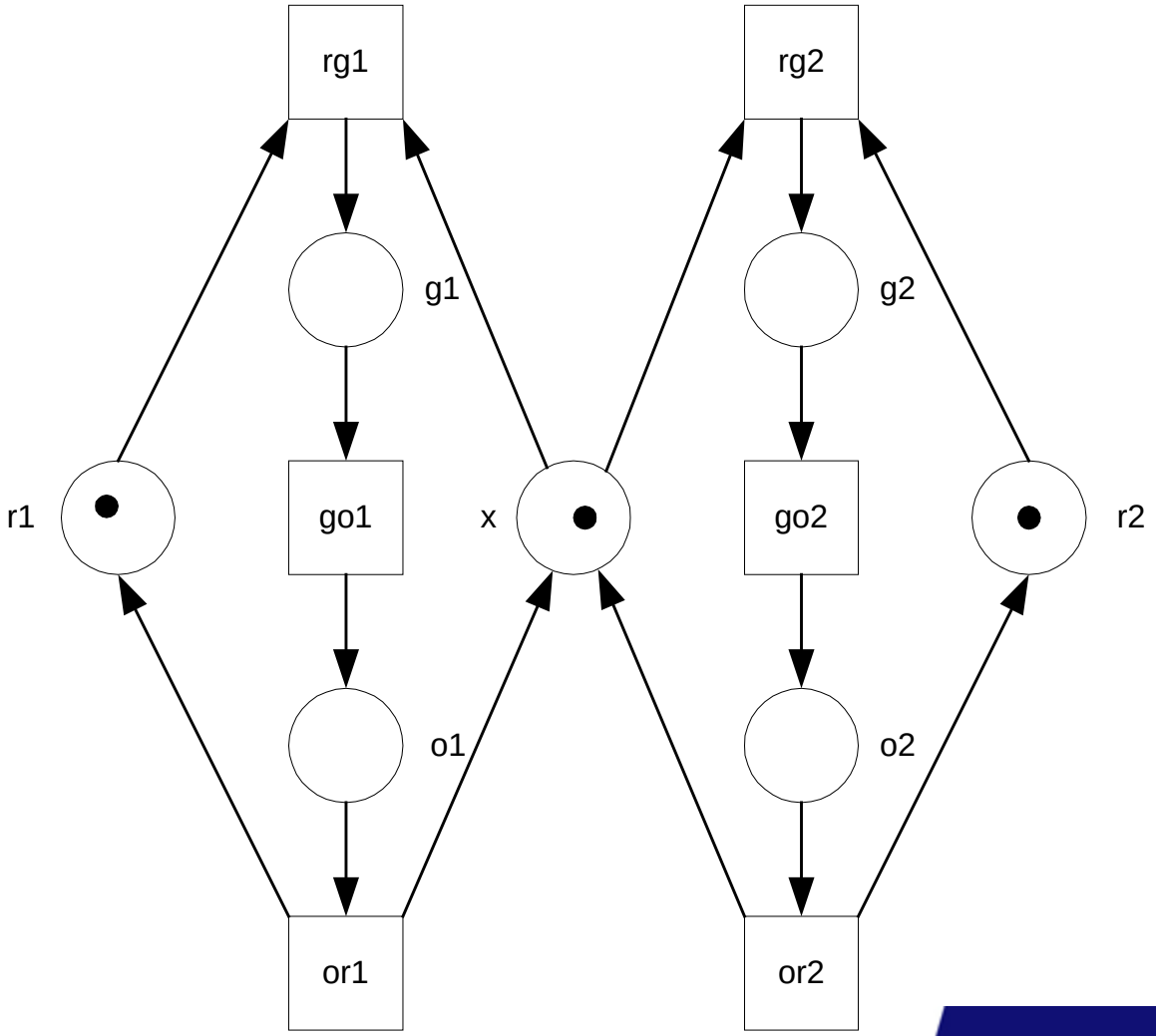


# Problem





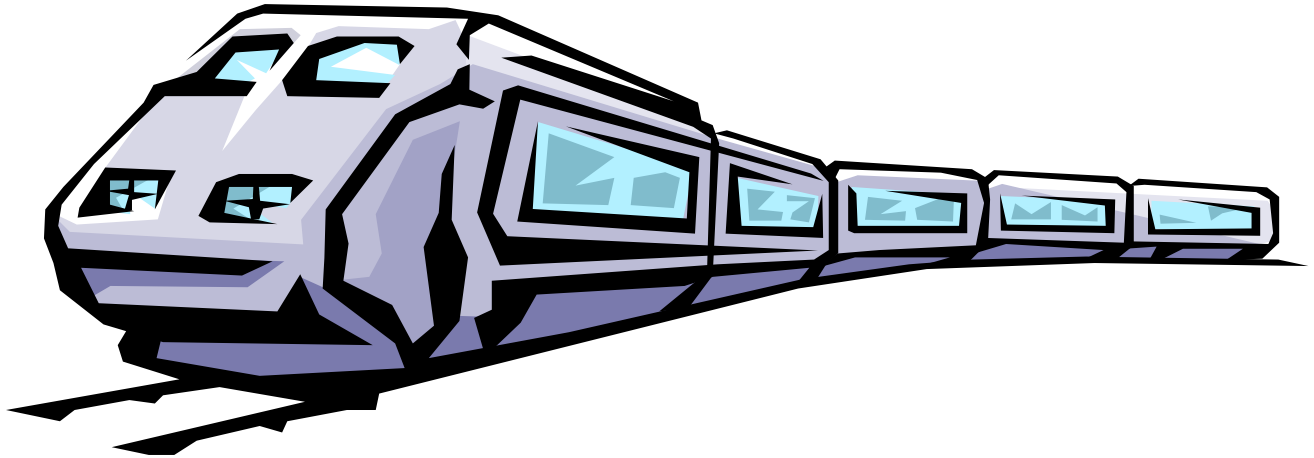
# Solution




**How to make them alternate?**

# Exercise: Train system (1)

- **Consider a circular railroad system with 4 (one-way) tracks (1,2,3,4) and 2 trains (A,B). No two trains should be at the same track at the same time and we do not care about the identities of the two trains.**

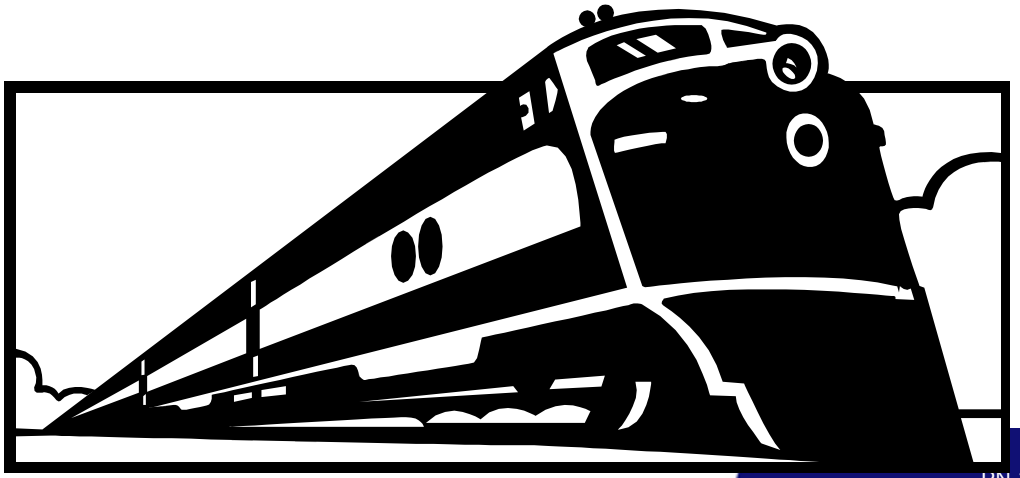


# Exercise: Train system (2)

- **Consider a railroad system with 4 tracks (1,2,3,4) and 2 trains (A,B). No two trains should be at the same track at the same time and we want to distinguish the two trains.**

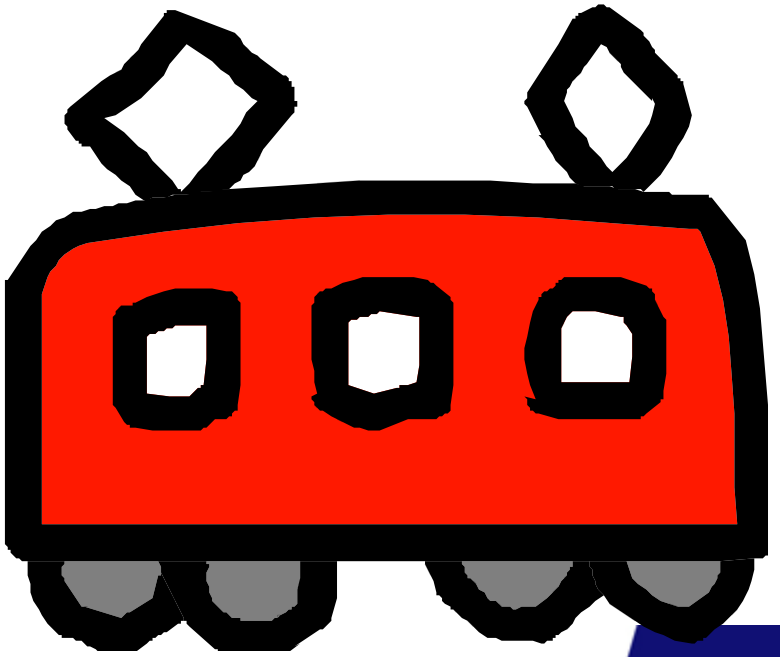
# Exercise: Train system (3)

- **Consider a railroad system with 4 tracks (1,2,3,4) and 2 trains (A,B). No two trains should be at the same track at the same time. Moreover the next track should also be free to allow for a safe distance. (We do not care about train identities.)**



# Exercise: Train system (4)

- **Consider a railroad system with 4 tracks (1,2,3,4) and 2 trains. Tracks are free, busy or claimed. Trains need to claim the next track before entering.**

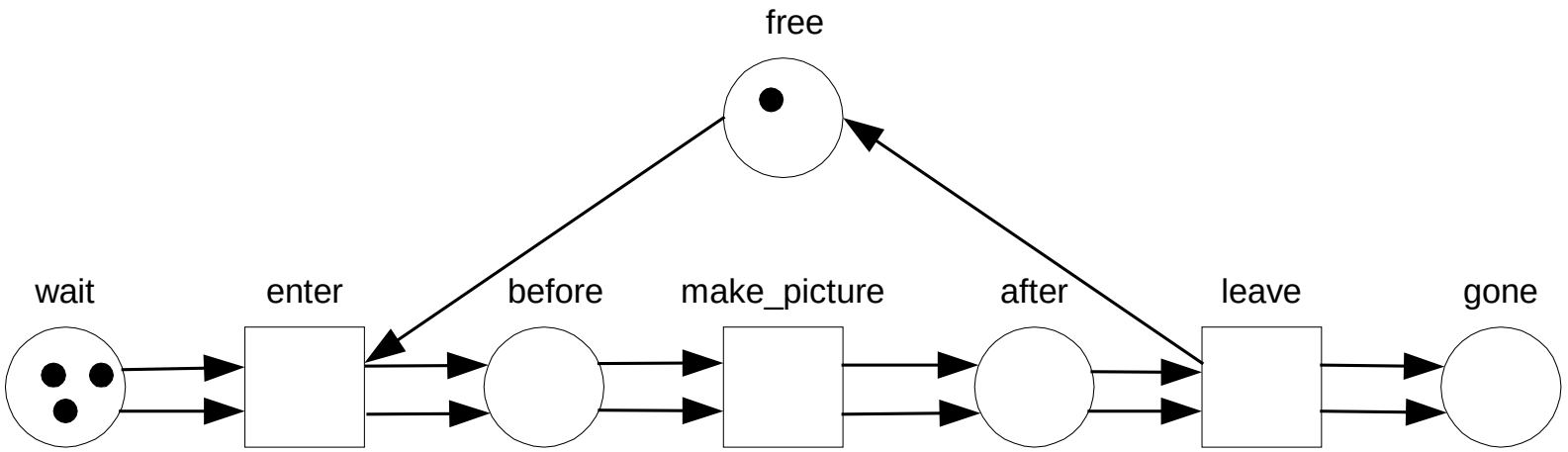


# WARNING

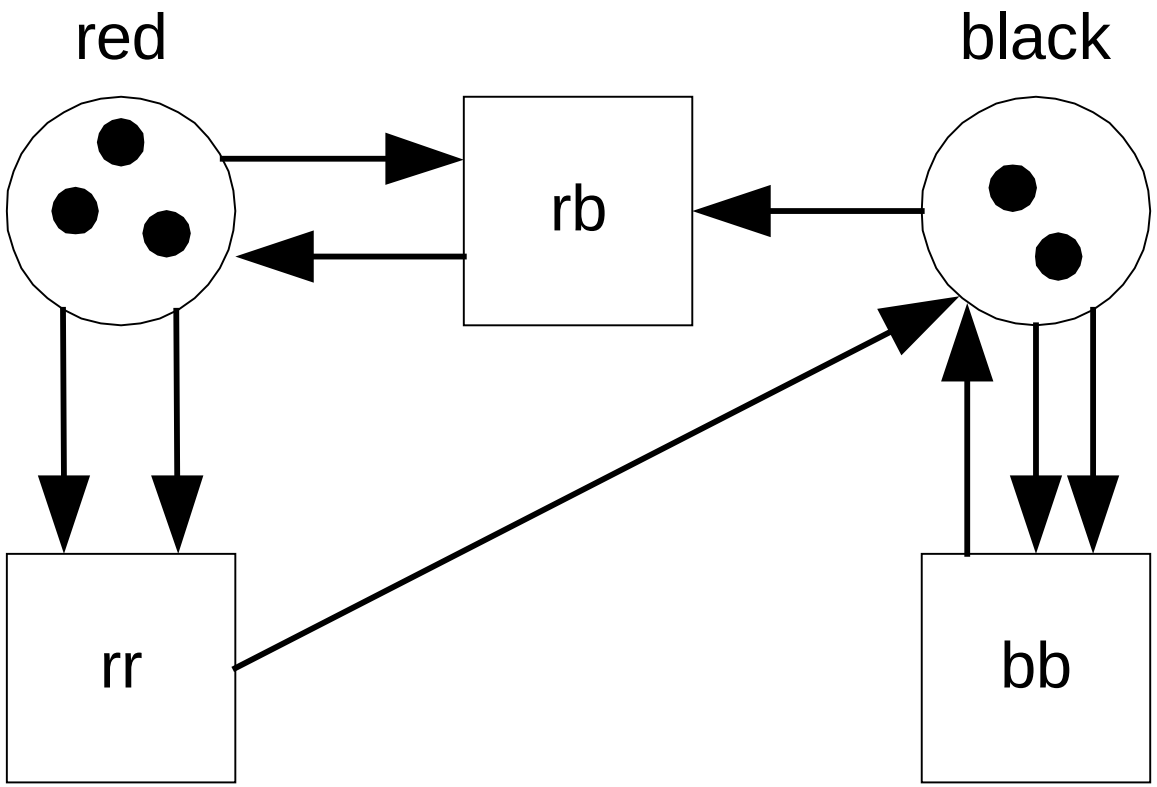
**It is not sufficient to understand the (process) models. You have to be able to design them yourself !**

# Multiple arcs connecting two nodes

- **The number of arcs between an input place and a transition determines the number of tokens required to be enabled.**
- **The number of arcs determines the number of tokens to be consumed/produced.**

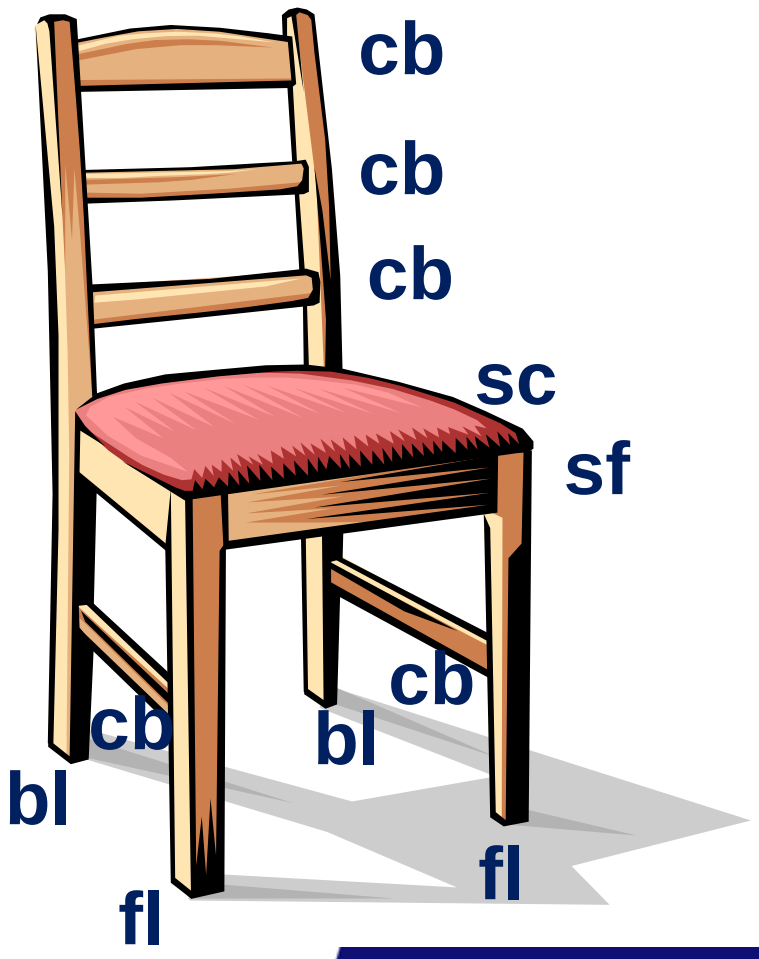


# Example: Ball game



# Exercise: Manufacturing a chair

- Model the manufacturing of a chair from its components: 2 front legs, 2 back legs, 5 cross bars, 1 seat frame, and 1 seat cushion as a Petri net.
- Select some sensible assembly order.
- Reverse logistics?

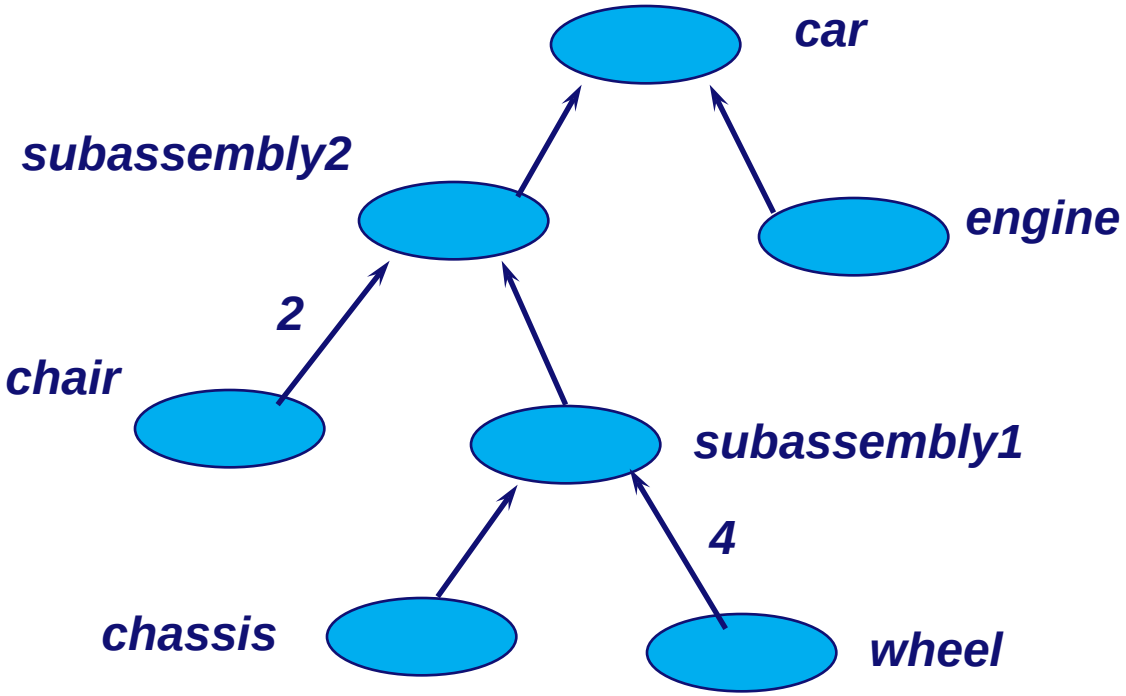


# Exercise: Burning alcohol.

- Model $C_2H_5OH + 3 * O_2 => 2 * CO_2 + 3 * H_2O$
- Assume that there are two steps: first each molecule is disassembled into its atoms and then these atoms are assembled into other molecules.

# Exercise: Manufacturing a car

- **Model the production process shown in the Bill-Of-Materials.**




# Formal definition

**A classical Petri net is a four-tuple (P,T,I,O) where:**

- **P is a finite set of places,**
- **T is a finite set of transitions,**
- **I : P x T -> N is the input function, and**
- **O : T x P -> N is the output function.**

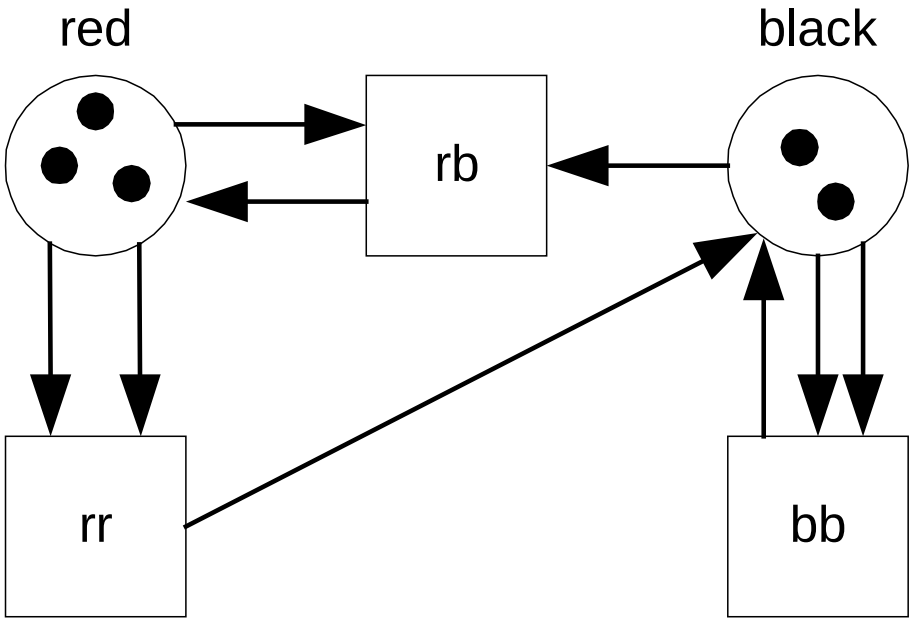
**Any diagram can be mapped onto such a four tuple and vice versa.**

# Formal definition (2)

**The state (marking) of a Petri net (P,T,I,O) is defined as follows:**

- **s: P-> N, i.e., a function mapping the set of places onto {0,1,2, … }.**

# Exercise: Map onto (P,T,I,O) and s



# Exercise: Draw diagram

**Petri net (P,T,I,O):**

- **P = {a,b,c,d}**
- **T = {e,f}**
- **I(a,e)=1, I(b,e)=2, I(c,e)=0, I(d,e)=0, I(a,f)=0, I(b,f)=0, I(c,f)=1, I(d,f)=0.**
- **O(e,a)=0, O(e,b)=0, O(e,c)=1, O(e,d)=0, O(f,a)=0, O(f,b)=2, O(f,c)=0, O(f,d)=3.**

**State s:**

- **s(a)=1, s(b)=2, s(c)=0, s(d) = 0.**

# Enabling formalized

**Transition *t* is enabled in state $s_1$ if and only if:**

$$\forall p \in P : s_1(p) \geq I(p,t)$$

# Firing formalized

**If transition *t* is enabled in state $s_1$, it can fire and the resulting state is $s_2$ :**

$$\forall p \in P : s_2(p) = s_1(p) - I(p,t) + O(t,p)$$

# Mapping Petri nets onto transition systems

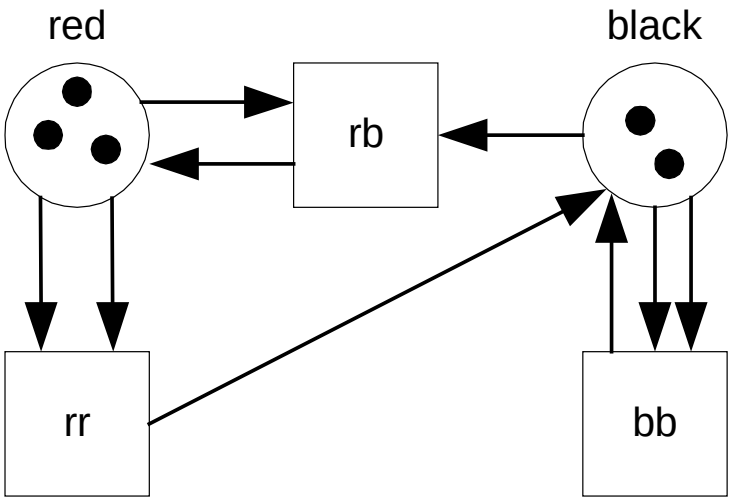
**A Petri net (P,T,I,O) defines the following transition system (S,TR):**

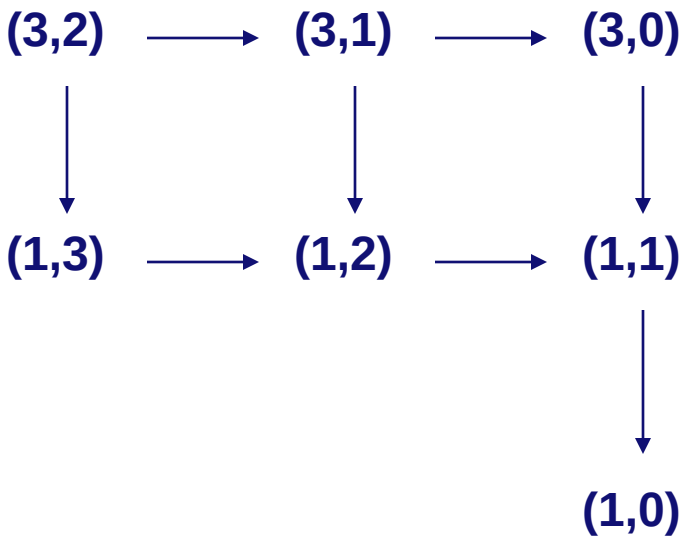
$$
\begin{aligned}
S &= P \to \mathbf{N} \\
TR &= \{\langle s_1, s_2 \rangle \in S \times S \mid \exists t \in T(\forall p \in P(s_1(p) \geq I(p,t)) \wedge \\
&\qquad (s_2(p) = s_1(p) - I(p,t) + O(t,p))\}
\end{aligned}
$$

# Reachability graph

- **The reachability graph of a Petri net is the part of the transition system reachable from the initial state in graph-like notation.**
- **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.**

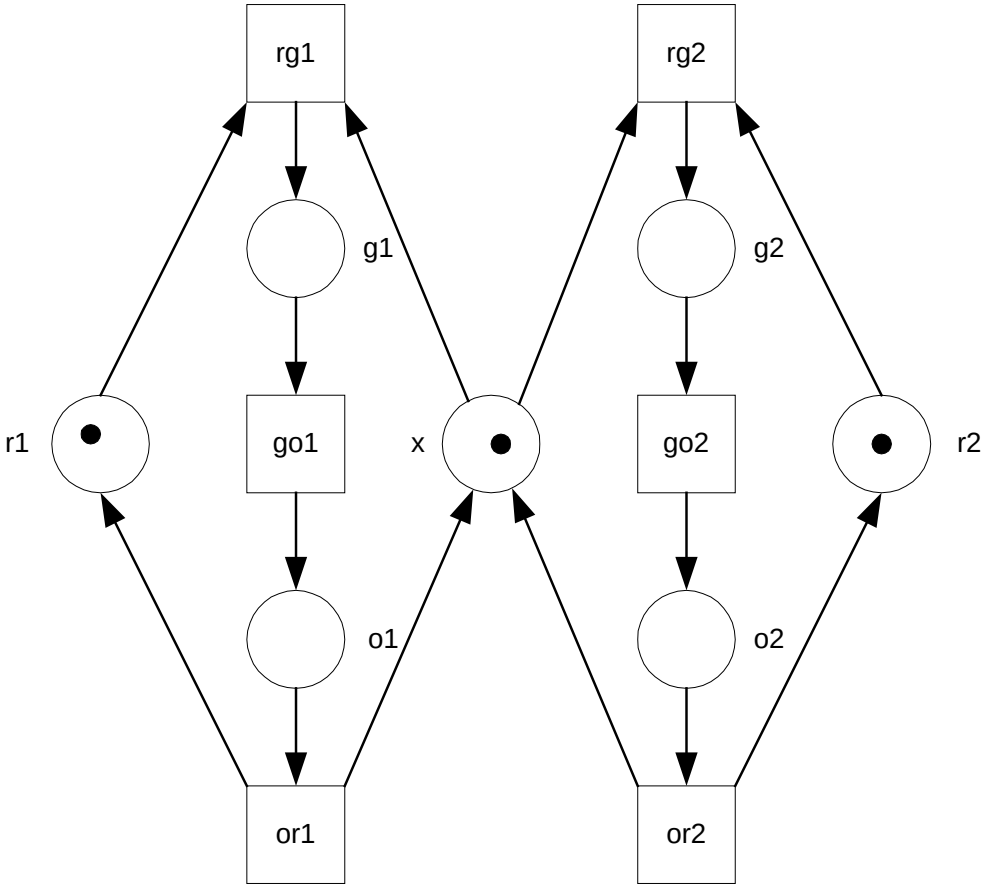
# Example




**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



# 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).**

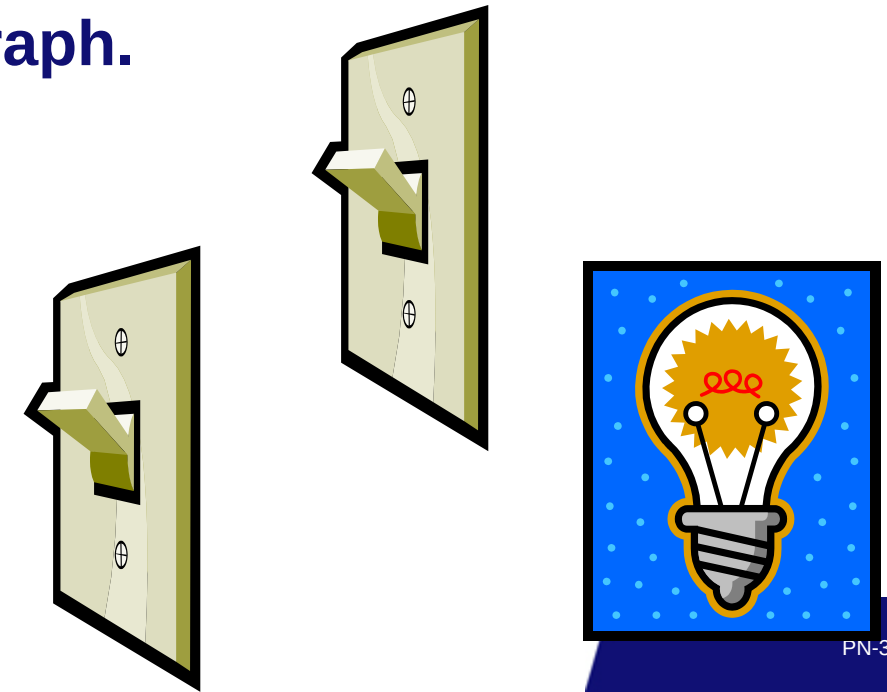
***How to recognize these states in the reachability graph?***

# Exercise: Producers and consumers

- **Model a process with one producer and one consumer, both are either busy or free and alternate between these two states. After every production cycle the producer puts a product in a buffer. The consumer consumes one product from this buffer per cycle.**
- **Give the reachability graph and indicate the final states.**
- **How to model 4 producers and 3 consumers connected through a single buffer?**
- **How to limit the size of the buffer to 4?**

# 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.



# Modeling

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

***The state (space) of a process/system is modeled by places and tokens and state transitions are modeled by transitions (cf. transition systems).***

# Role of a token

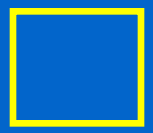
**Tokens can play the following roles:**

- **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

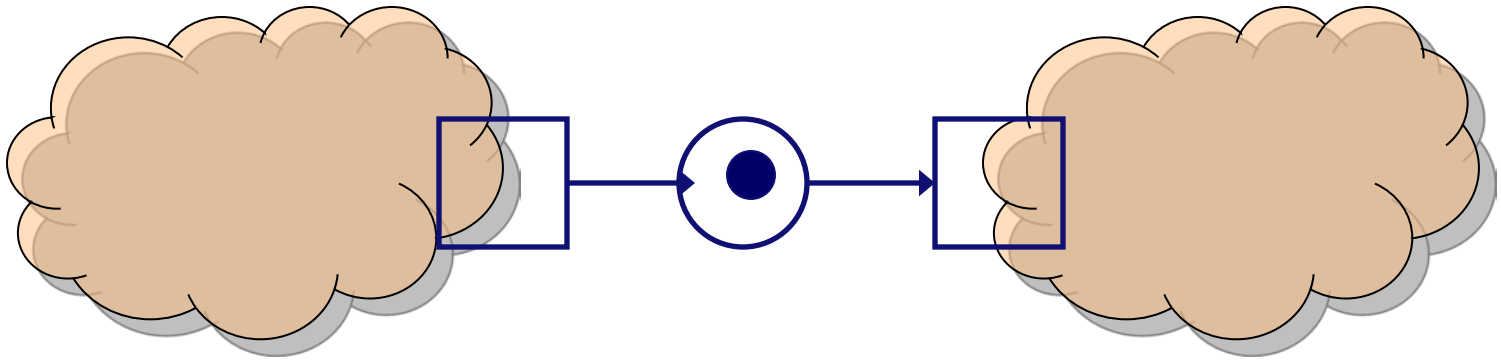


- an event: for example, starting an operation, the death of a patient, a change seasons or the switching of a traffic light from red to green;
- a transformation of an object, like adapting a product, updating a database, or updating a document;
- a transport of an object: for example, transporting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

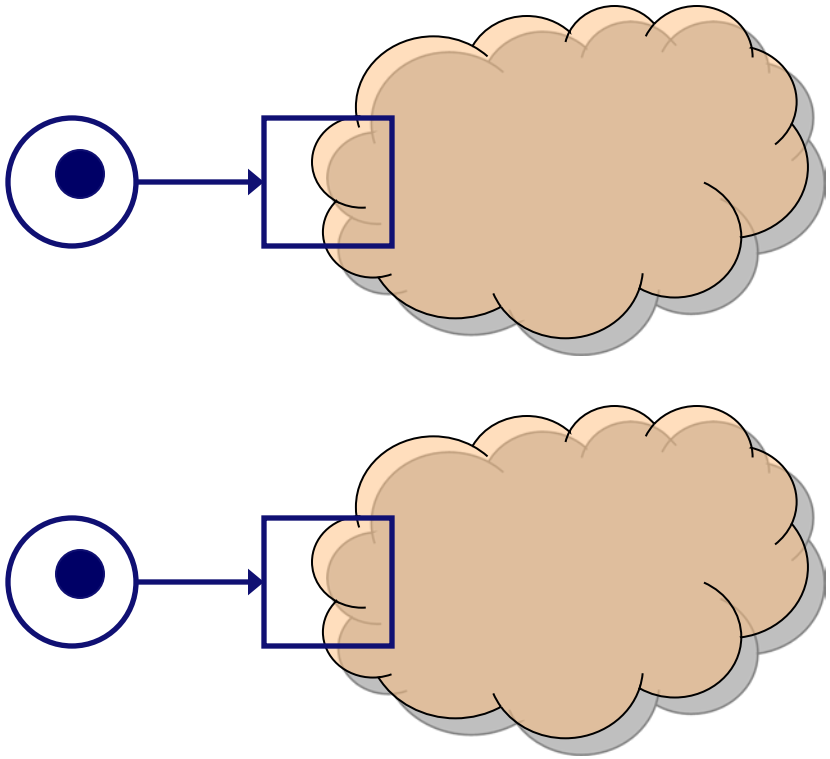
# Typical network structures

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

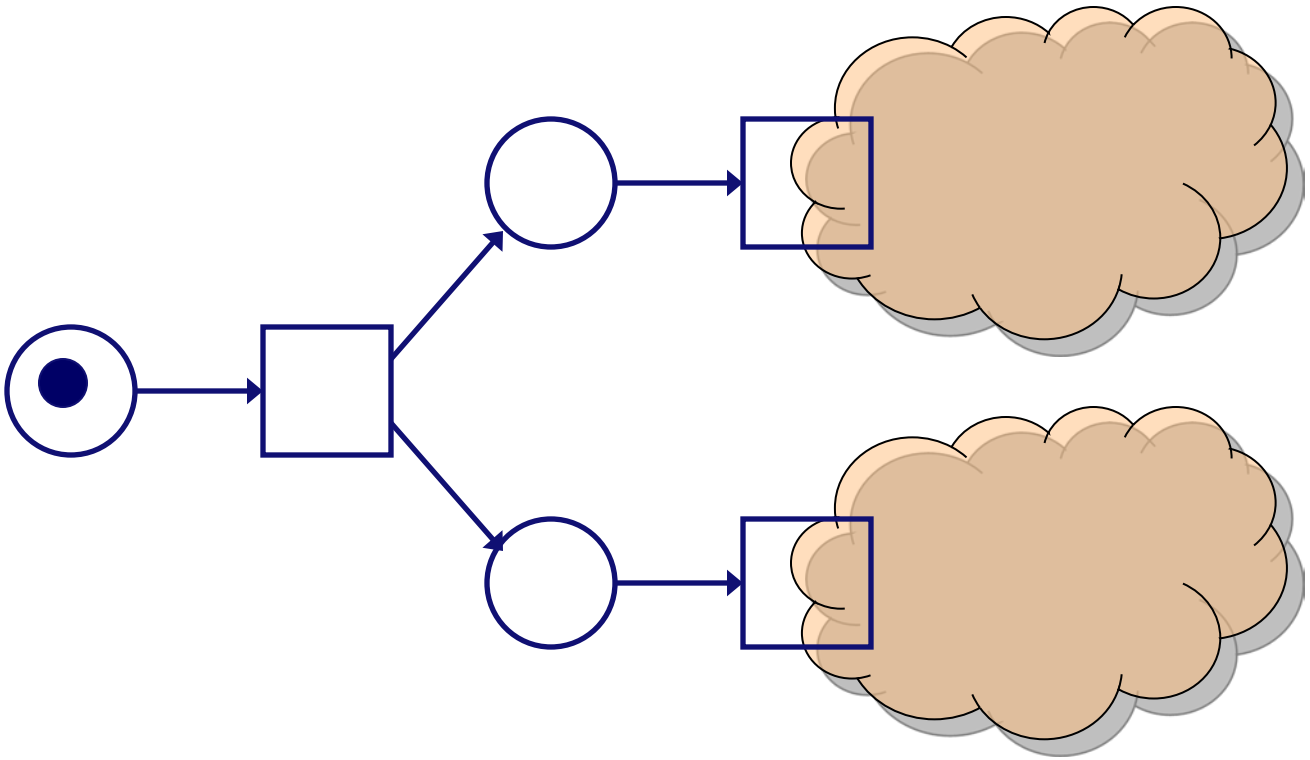
# Causality



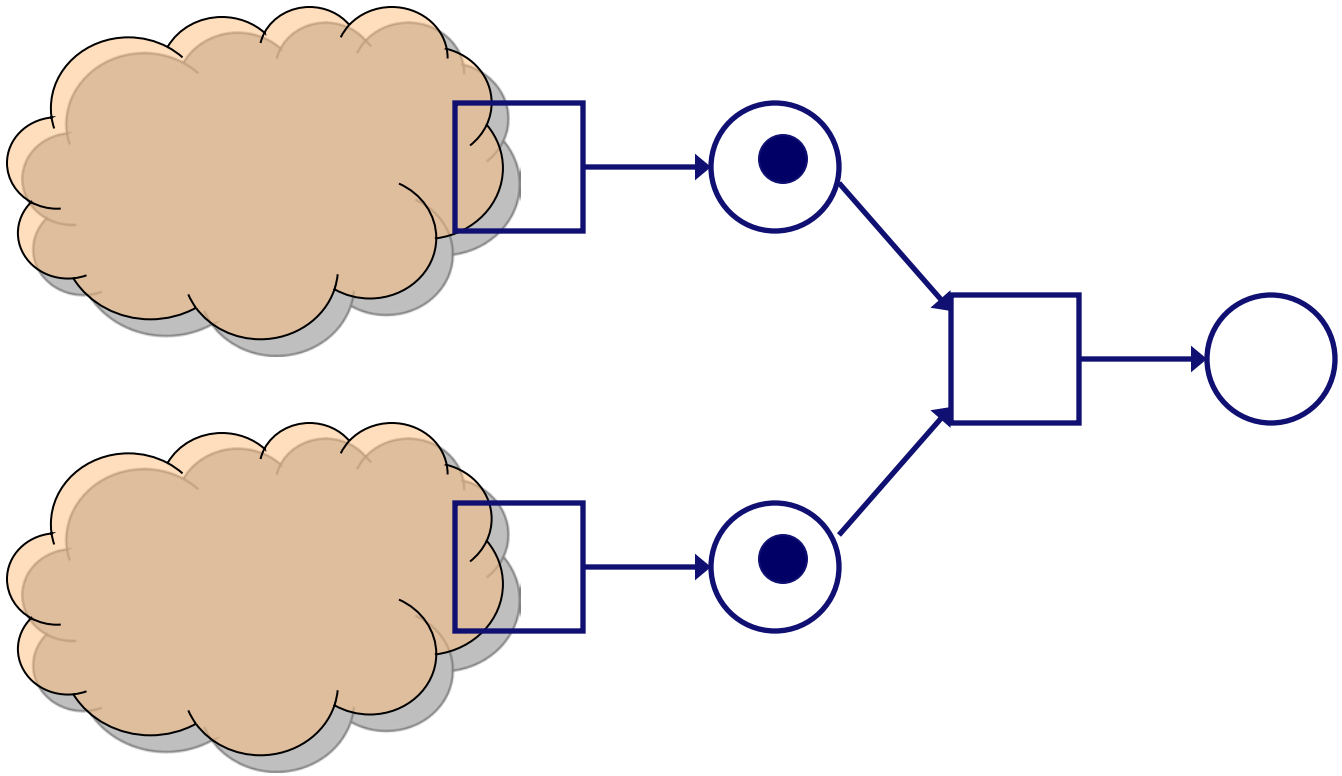
# Parallelism



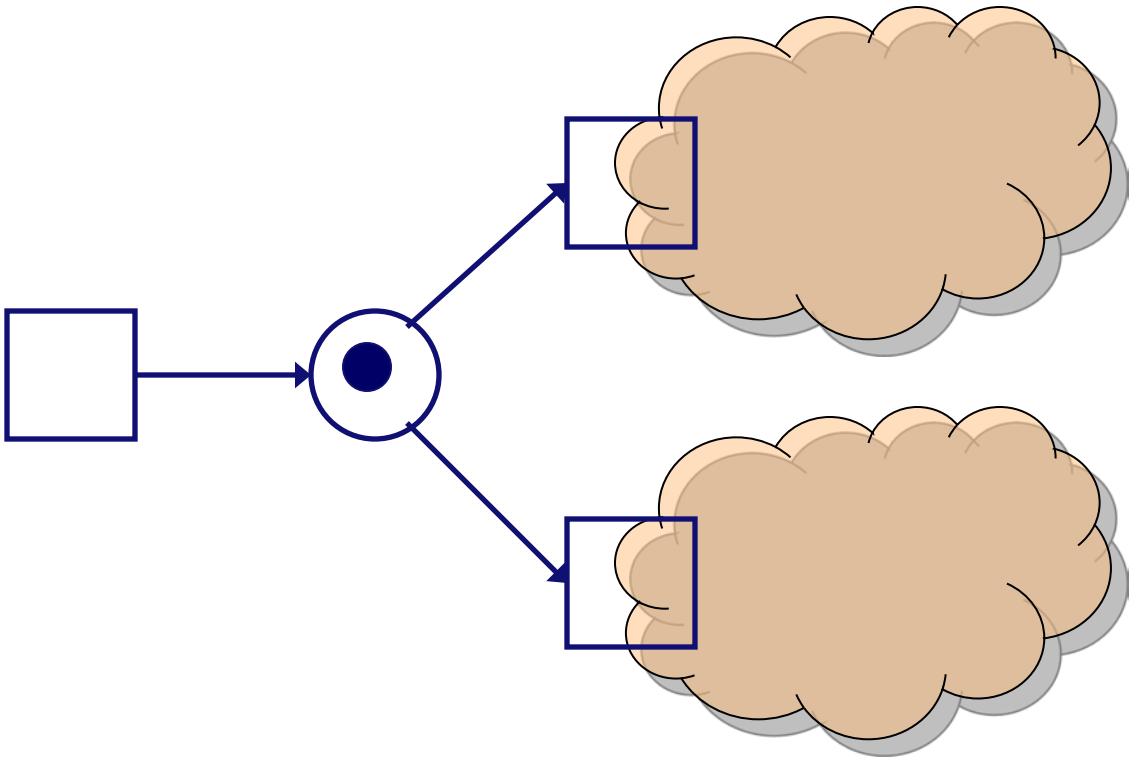
# Parallelism: AND-split



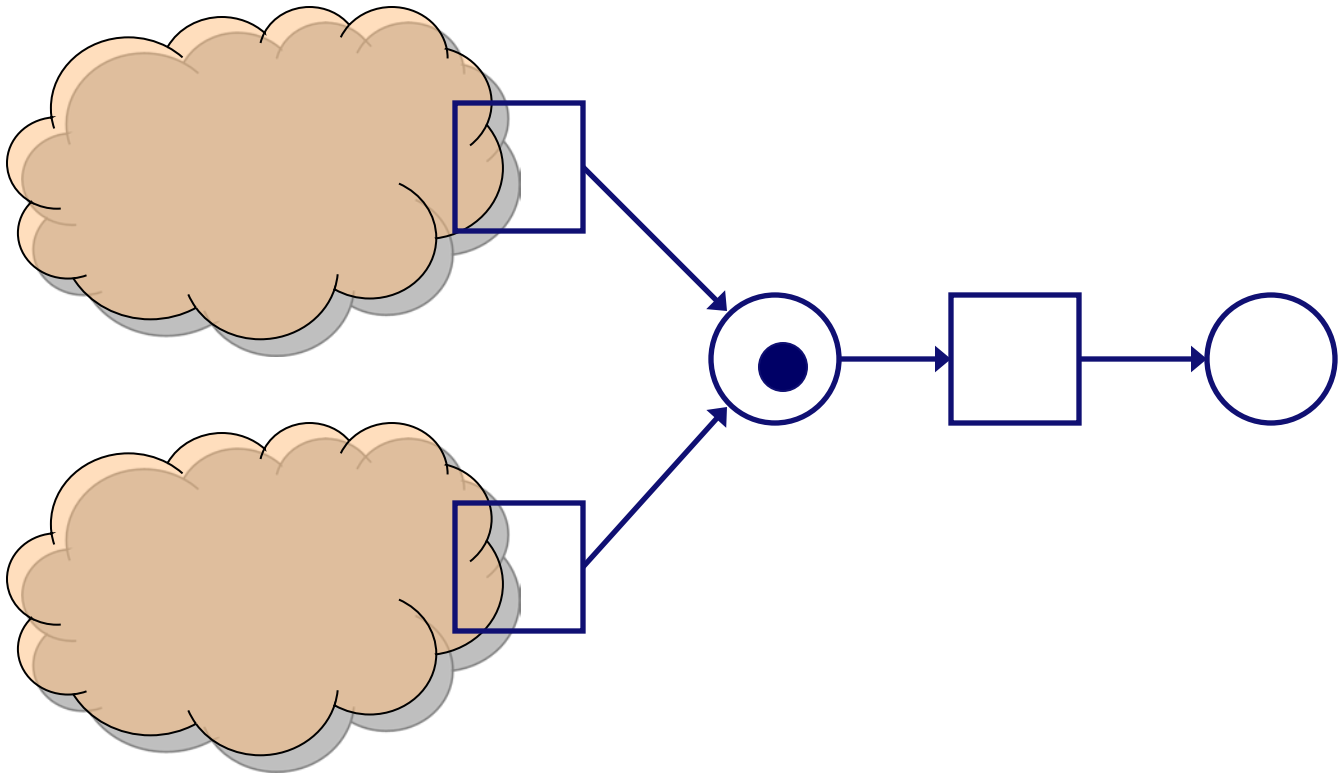
# Parallelism: AND-join



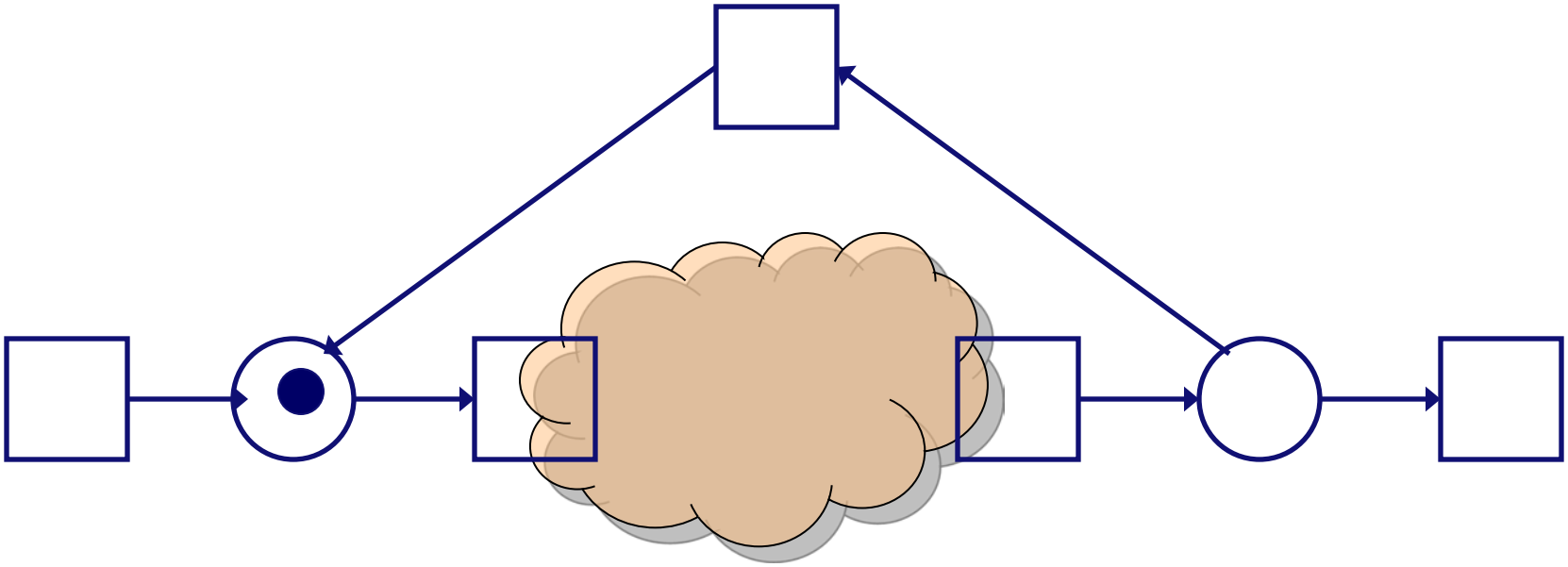
# Choice: XOR-split



# Choice: XOR-join

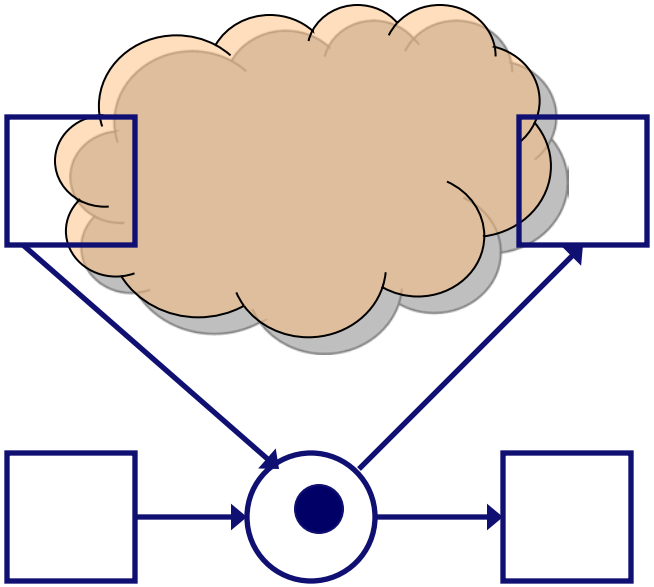


# Iteration: 1 or more times



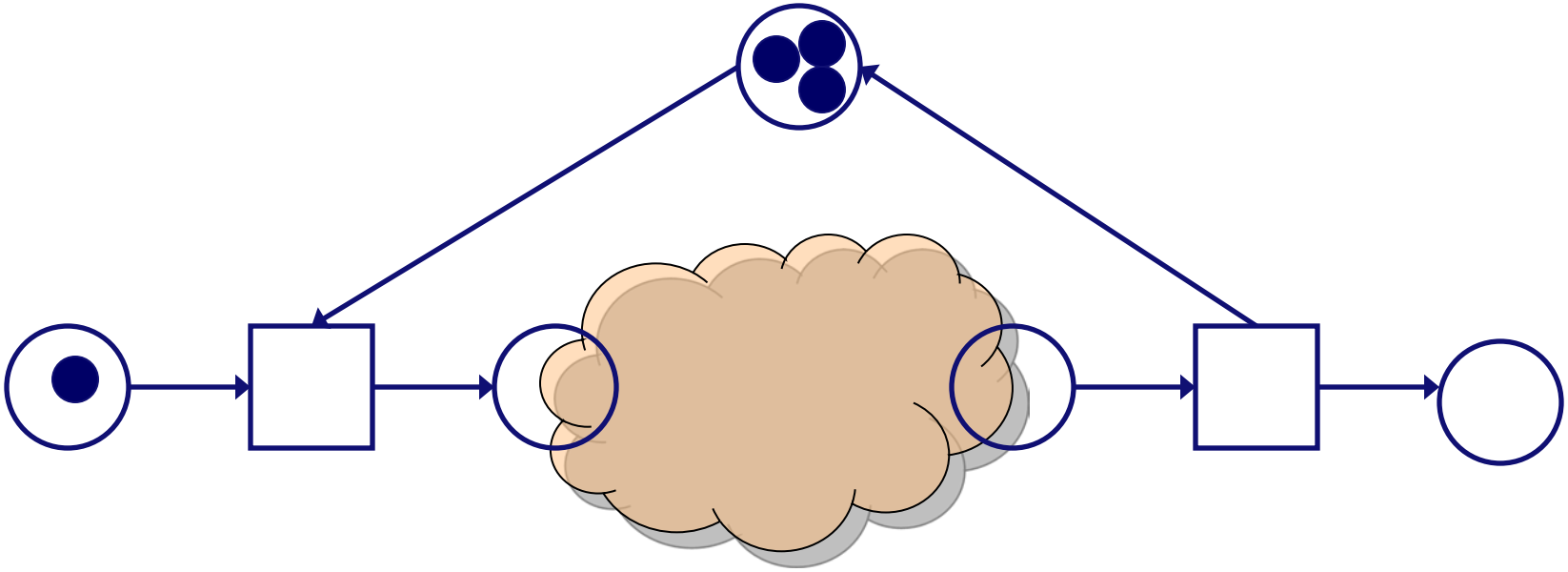
***XOR-join before XOR-split***

# Iteration: 0 or more times



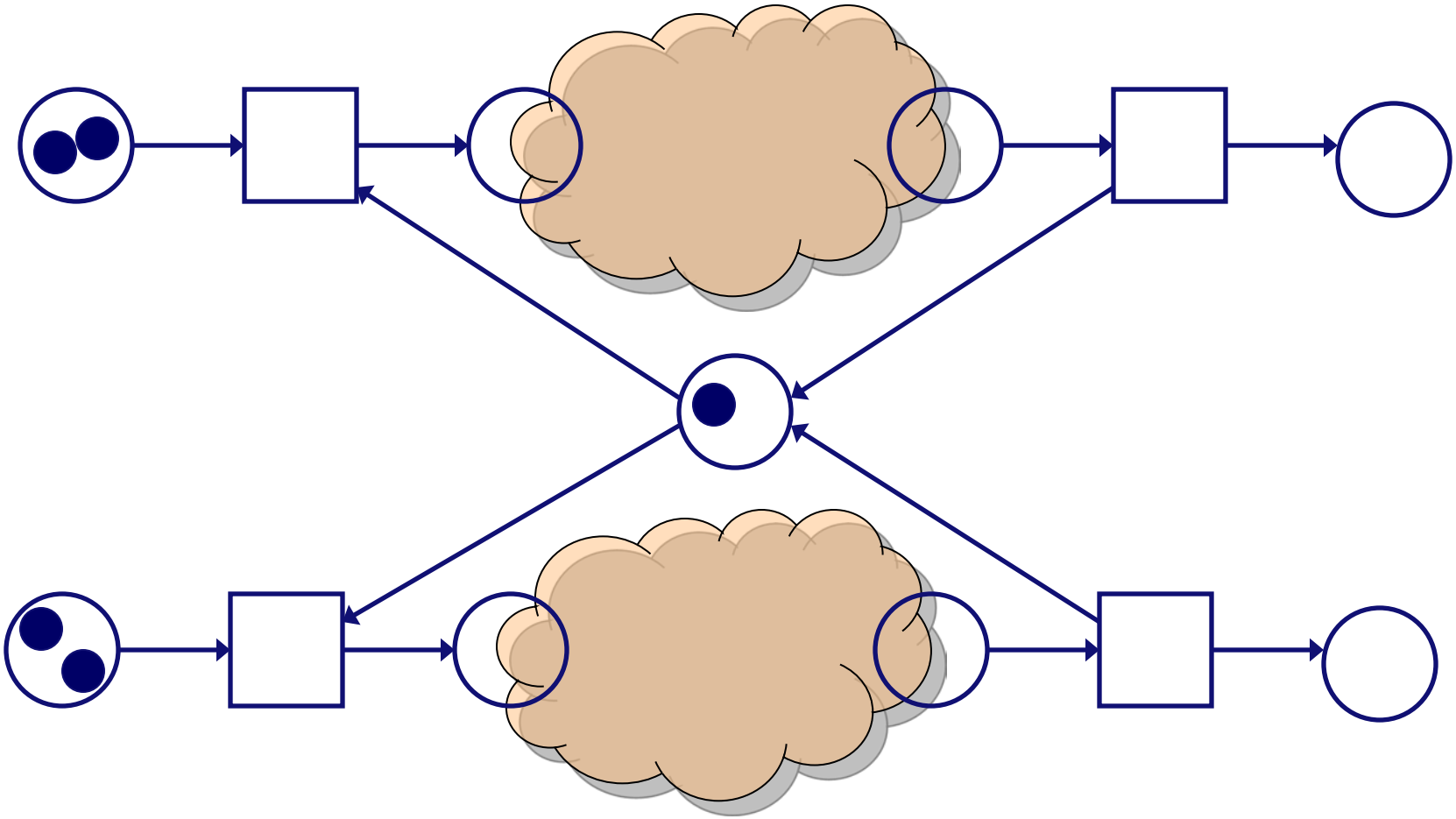
***XOR-join before XOR-split***

# Capacity constraints: feedback loop



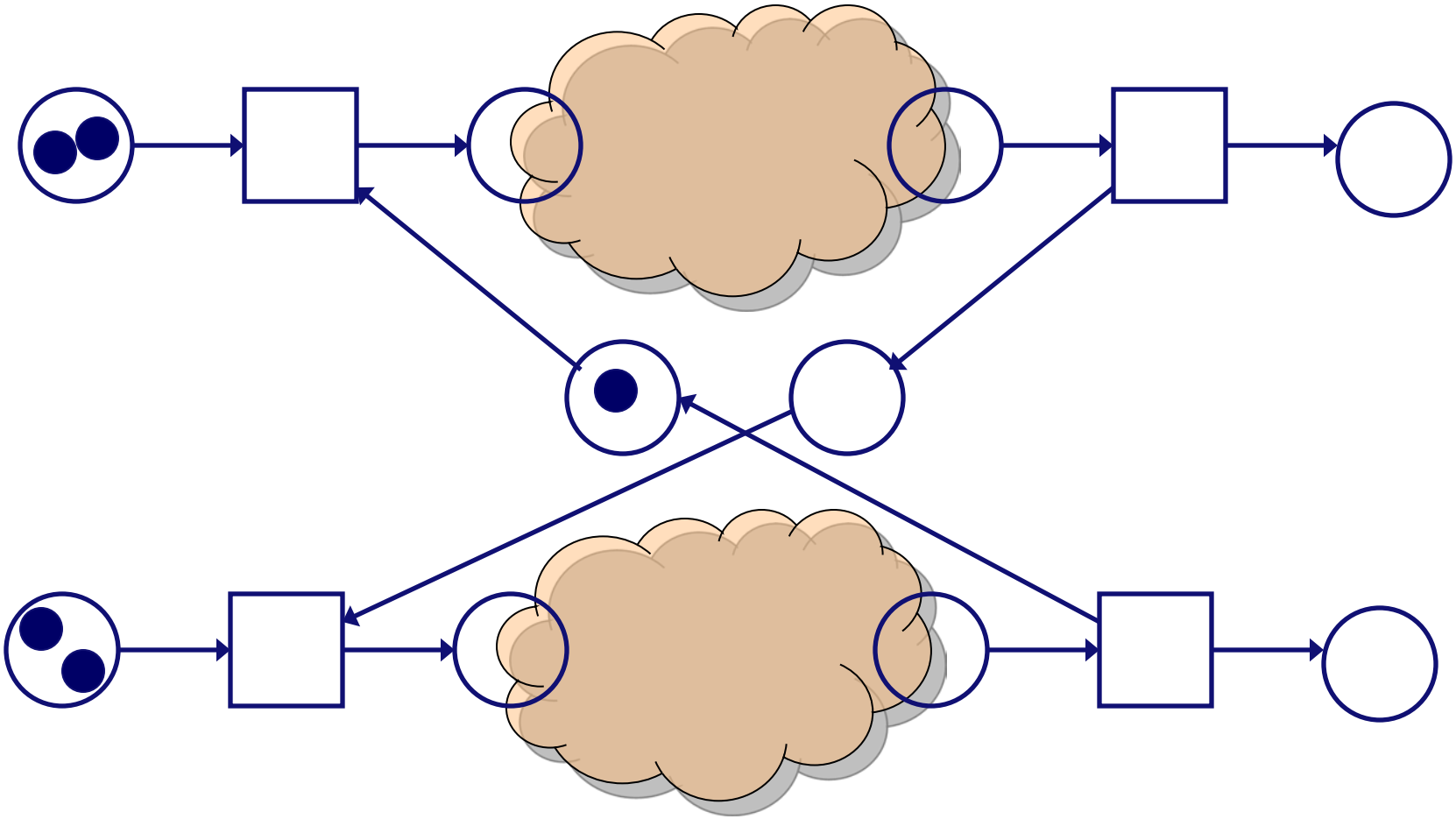
***AND-join before AND-split***

# Capacity constraints: mutual exclusion



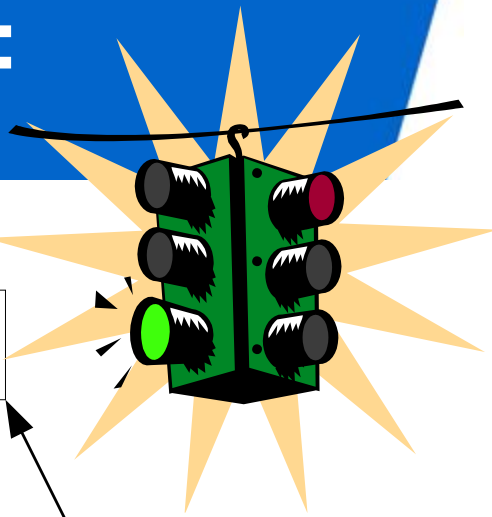
*AND-join before AND-split*

# Capacity constraints: alternating

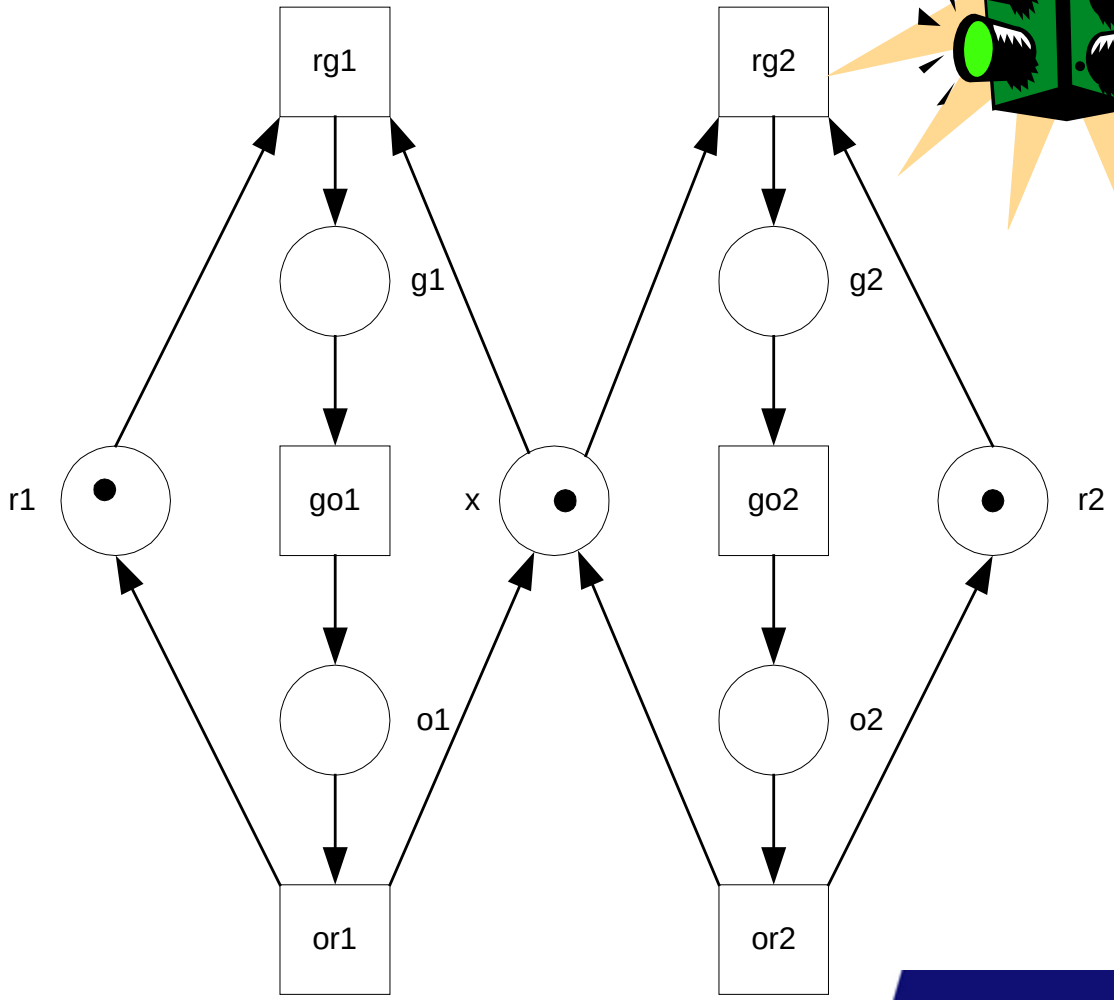


*AND-join before AND-split*

# We have seen most patterns, e.g.:

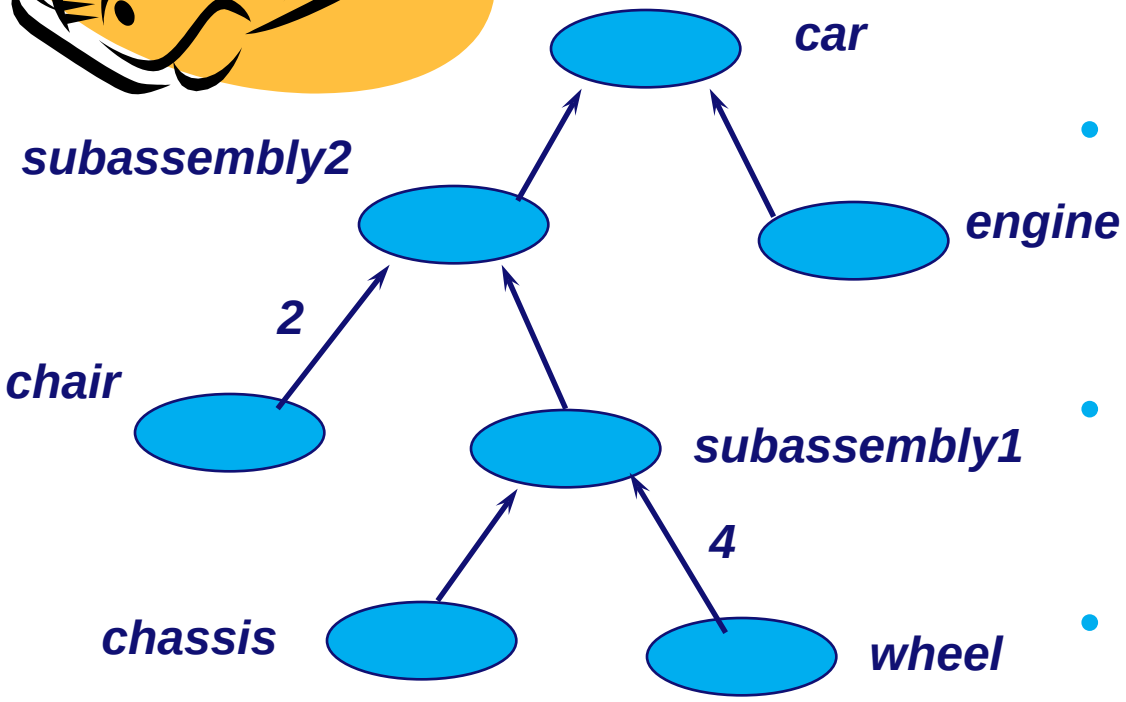


**Example of mutual exclusion**




**How to make them alternate?**

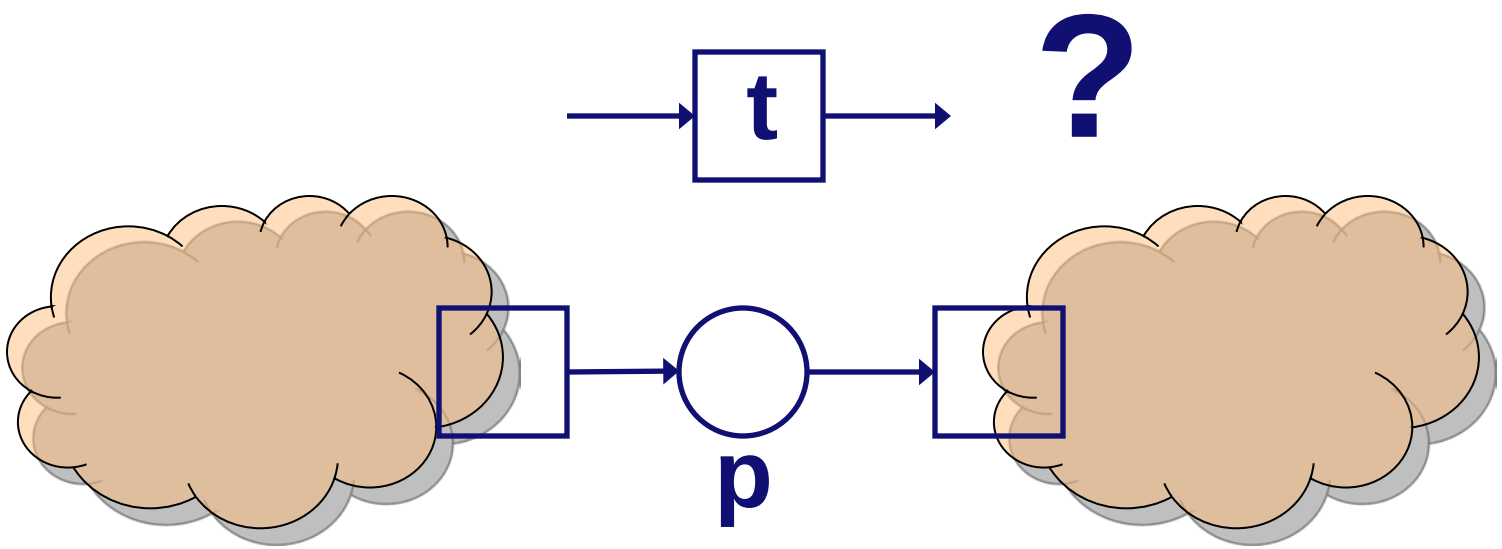
# Exercise: Manufacturing a car (2)




- 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.
- Hint: model both the start and completion of an assembly step.

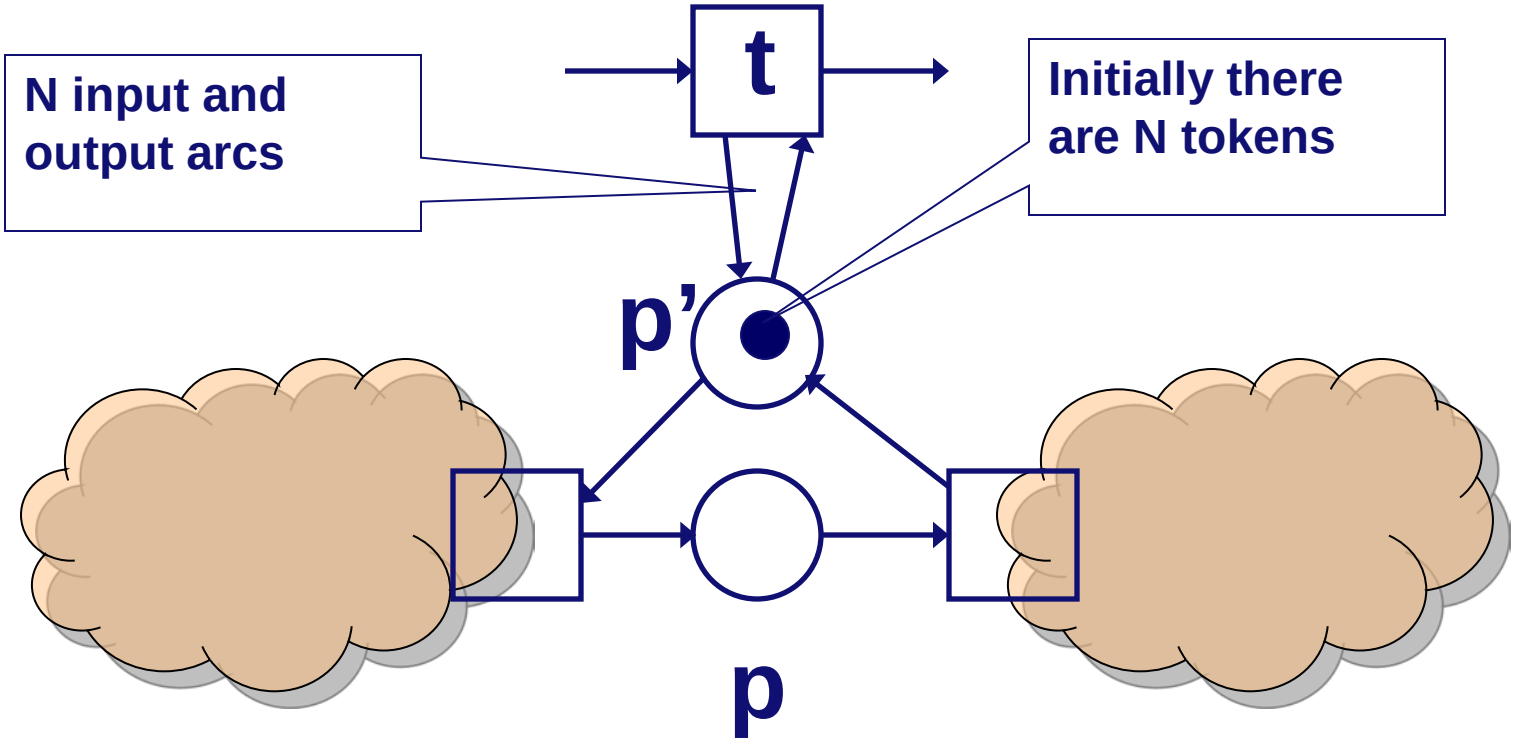
# Modeling problem (1): Zero testing

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



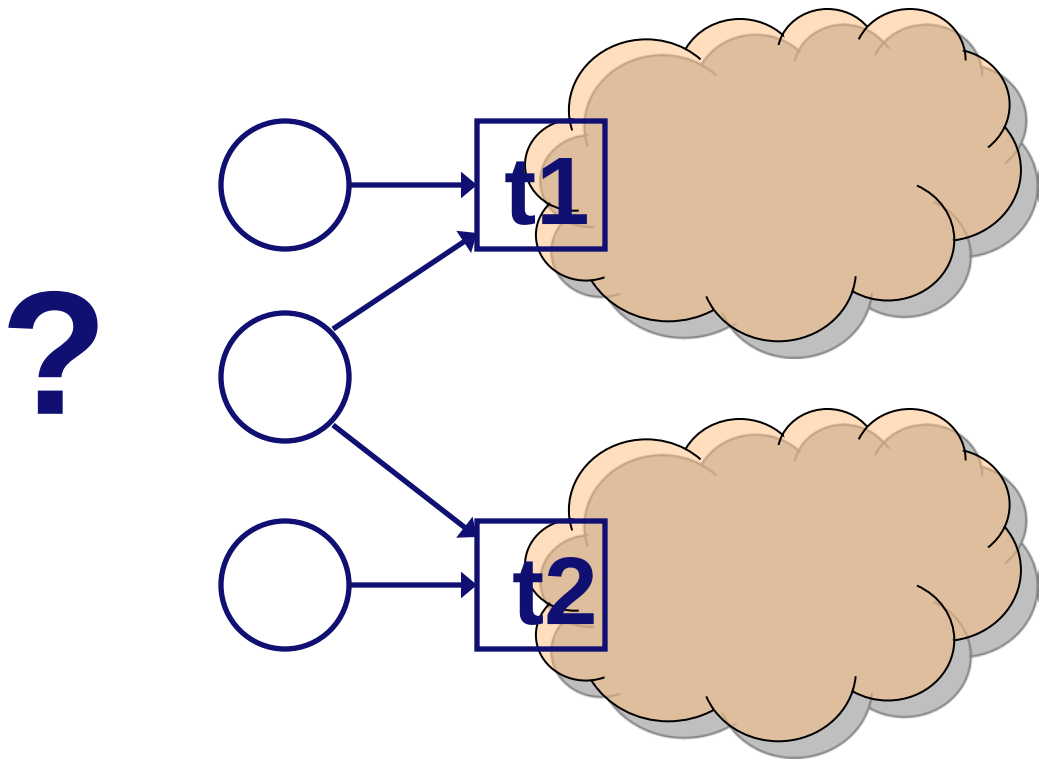
# Solution

- **Only works if place is N-bounded**



# Modeling problem (2): Priority

- **Transition t1 has priority over t2**




***Hint: similar to Zero testing!***

# A bit of theory

- **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).**

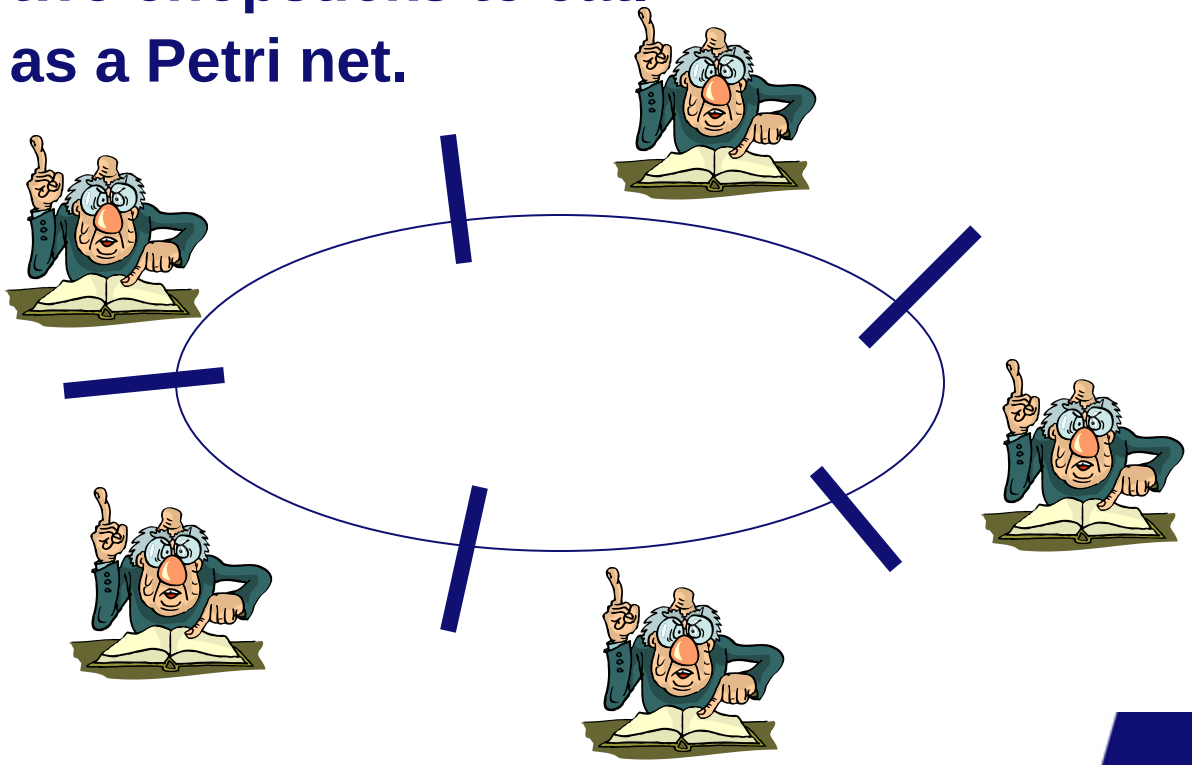
*** Turing completeness corresponds to the ability to execute any computation.**

# Exercise: Witness statements

- As part of the process of handling insurance claims there is the handling of witness statements.
- There may be 0-10 witnesses per claim. After an initialization step (one per claim), each of the witnesses is registered, contacted, and informed (i.e., 0-10 per claim in parallel). Only after all witness statements have been processed a report is made (one per claim).
- Model this in terms of a Petri net.

# Exercise: Dining philosophers

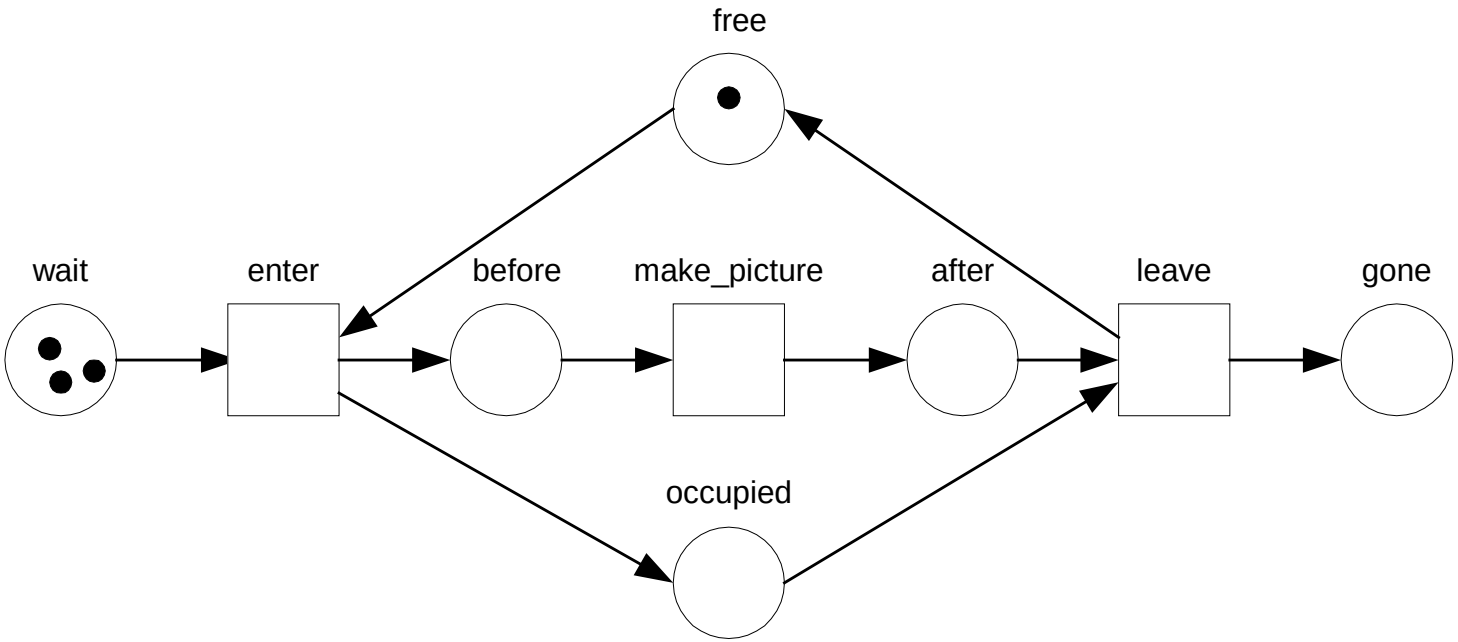
- 5 philosophers sharing 5 chopsticks: chopsticks are located in-between philosophers
- A philosopher is either in state eating or thinking and needs two chopsticks to eat.
- Model as a Petri net.



# Preview: Analysis (Chapter 8)

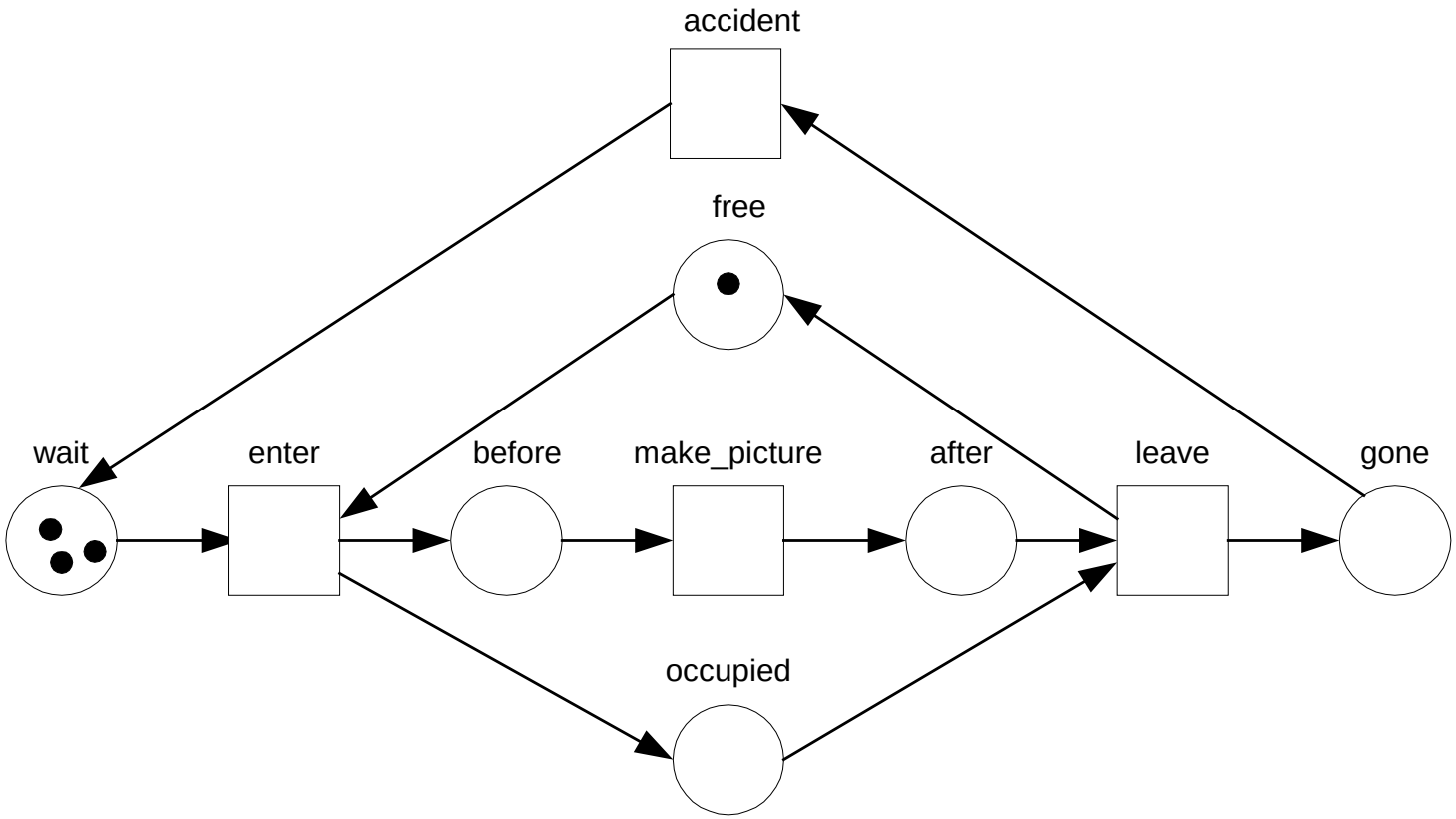
- **Various types of analysis techniques:**
  - ***Simulation*** **(repeatedly playing the token game)**
  - ***Reachability analysis*** **(constructing the reachability graph)**
  - ***Markovian analysis*** **(reachability graph with transition probabilities)**
  - ***Invariants*****: place invariants and transition invariants (conservation of tokens and sequences without effect)**
- **Role of models: (1)** ***insight*****, (2)** ***analysis*****, and (3)** ***specification*****.**

# Place invariant: Example




**wait+before+after+gone**
**free+occupied**

# Transition invariant: Example




**enter+make_picture+leave+accident**