Petri nets:
Basic elements and extensions

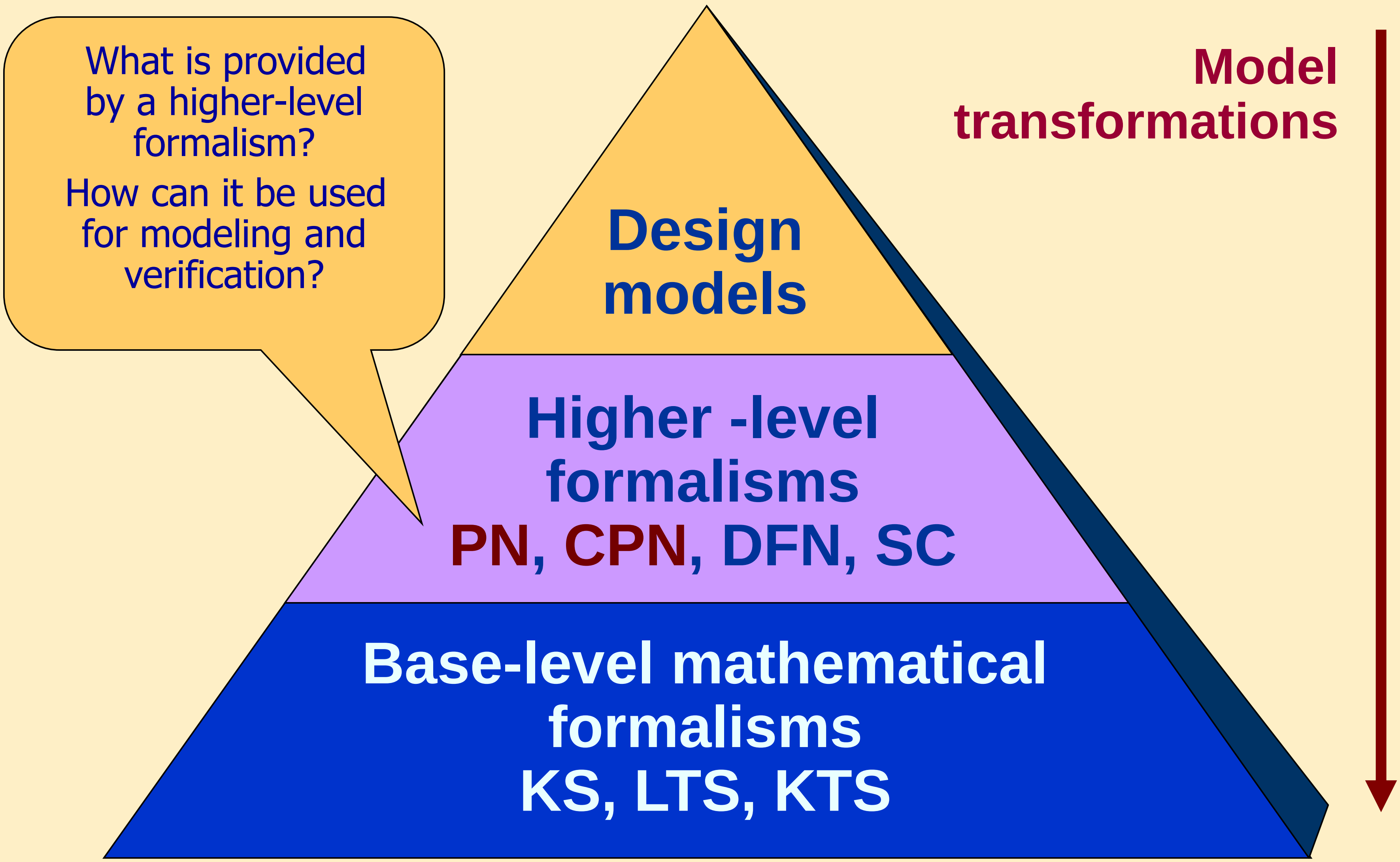
dr. Tamás Bartha

dr. András Pataricza

dr. István Majzik

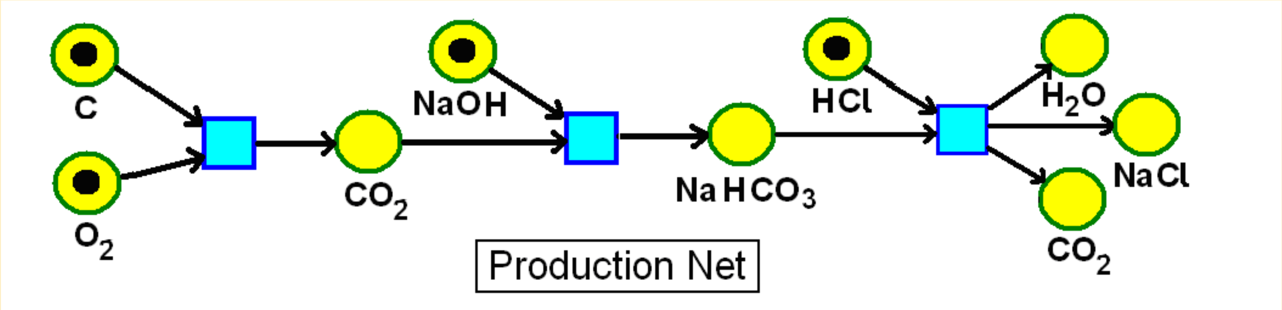
BME Department of Measurement and Information Systems

Formal models for verification



Petri nets: Origins

- Carl Adam Petri: German mathematician, 1926-2010
- Invented the notation in 1939 (as a 13 years old)
- Originally for describing chemical processes



- Mathematical foundations were developed later in his PhD dissertation (in two weeks in 1962)
  - C. A. Petri: Kommunikation mit Automaten. Schriften des Rheinisch-Westfälischen Institutes für Instrumentelle Mathematik an der Universität Bonn Nr. 2, 1962

Petri nets: Applications

Typical applications of Petri nets: modeling of

- concurrent,
- asynchronous,
- distributed,
- parallel,
- non-deterministic

systems

There are other formalisms for this purpose, e.g., network of automata
Why are Petri nets special?

- More compact representation of the state
- Clear expression of synchronization

$\Rightarrow$ Compact, clear models

Basic properties of Petri nets

- Provides both:
  - Graphical representation $\rightarrow$ Understandable (+hierarchy)
  - Mathematical formalism $\rightarrow$ Precise, unambiguous
- Structure expresses:
  - Control flow
  - Data structure
- Other advantages:
  - Easily extensible
    - E.g. timed, stochastic, colored, hierarchical Petri nets
  - Other formalisms can be translated to Petri nets
    - Some of its extensions is Turing-complete

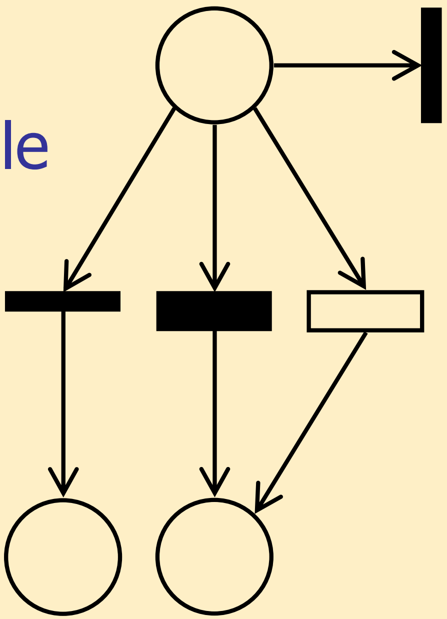
Structure and semantics of Petri nets

Structure of Petri nets

Structure: Directed, weighted, bipartite graph

- Two types of nodes:
  - Place: $p \in P$ Denoted by a circle
  - Transition: $t \in T$ Denoted by a rectangle
- Directed arcs:
  - Place $\rightarrow$ transition
  - Transition $\rightarrow$ place

  bipartite graph!
  - $e \in E, \quad E \subseteq (P \times T) \cup (T \times P)$

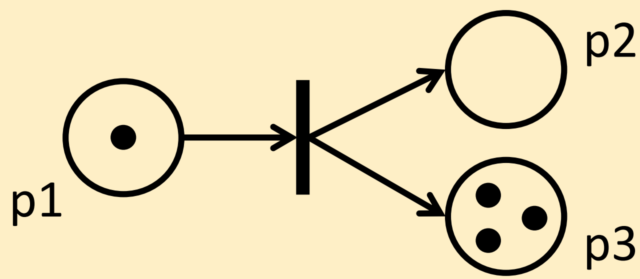


State of a Petri net

Places: Modeling of possible situations, conditions

A local situation or condition holds: The place is “marked”

- Marking a state: token Denoted by a black dot
  - E.g. marking place “Ready to start” if a process is ready to start
- “Marking” (state) of a place: number of its tokens
  - E.g. multiple tokens in place “Ready to start” means multiple processes are ready
- State of the net: marking of its places
  - Marking: token distribution vector M, one element for each place
  - Each m_i in M denotes the number of tokens in place p_i



$$M = \left[\begin{array}{c}1\\0\\3\end{array}\right]\begin{array}{l}\leftarrow p_1\\ \leftarrow p_2\\ \leftarrow p_3\end{array}$$

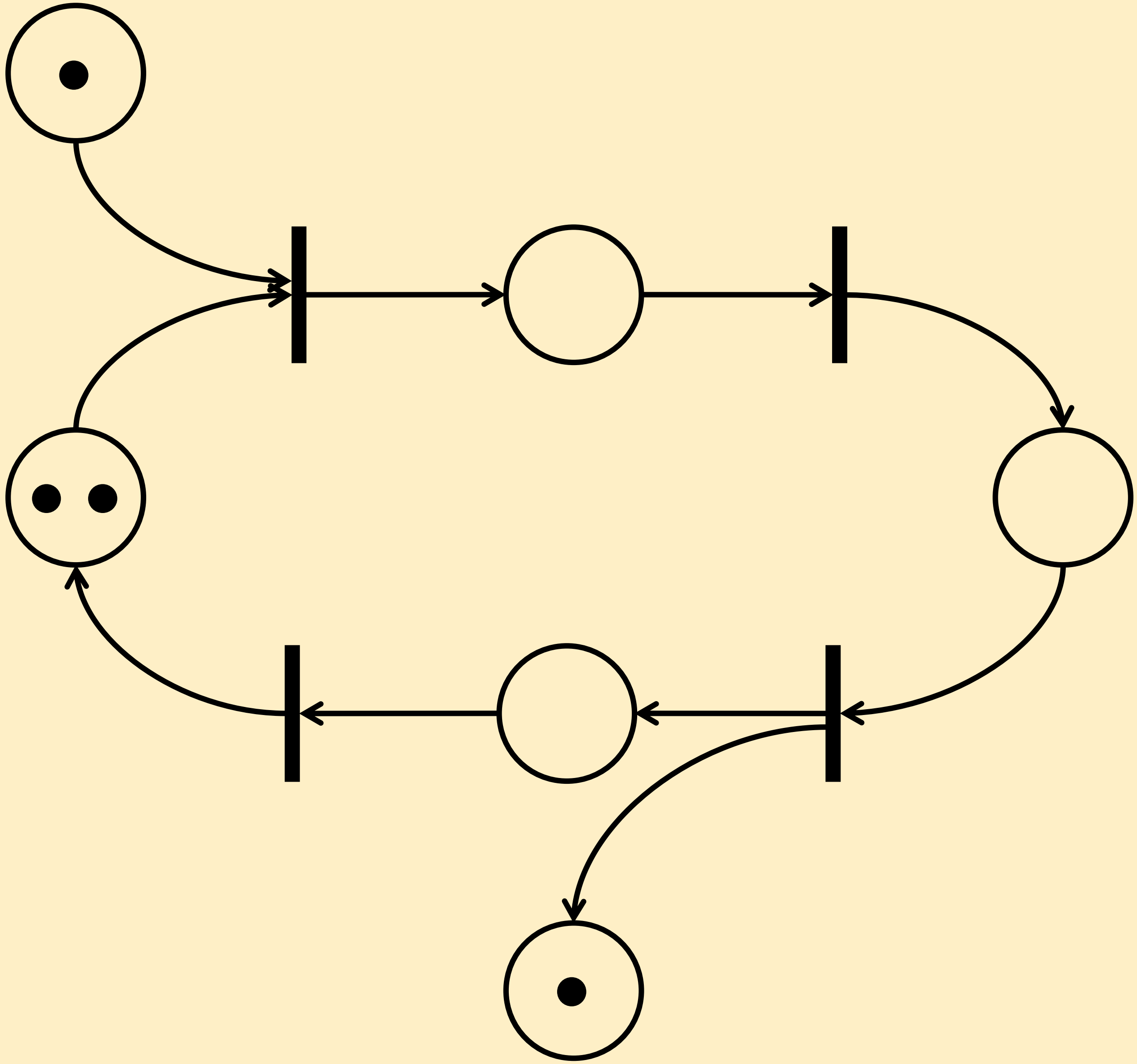
Semantics of Petri nets (dynamics)

Transitions: Modeling possible changes

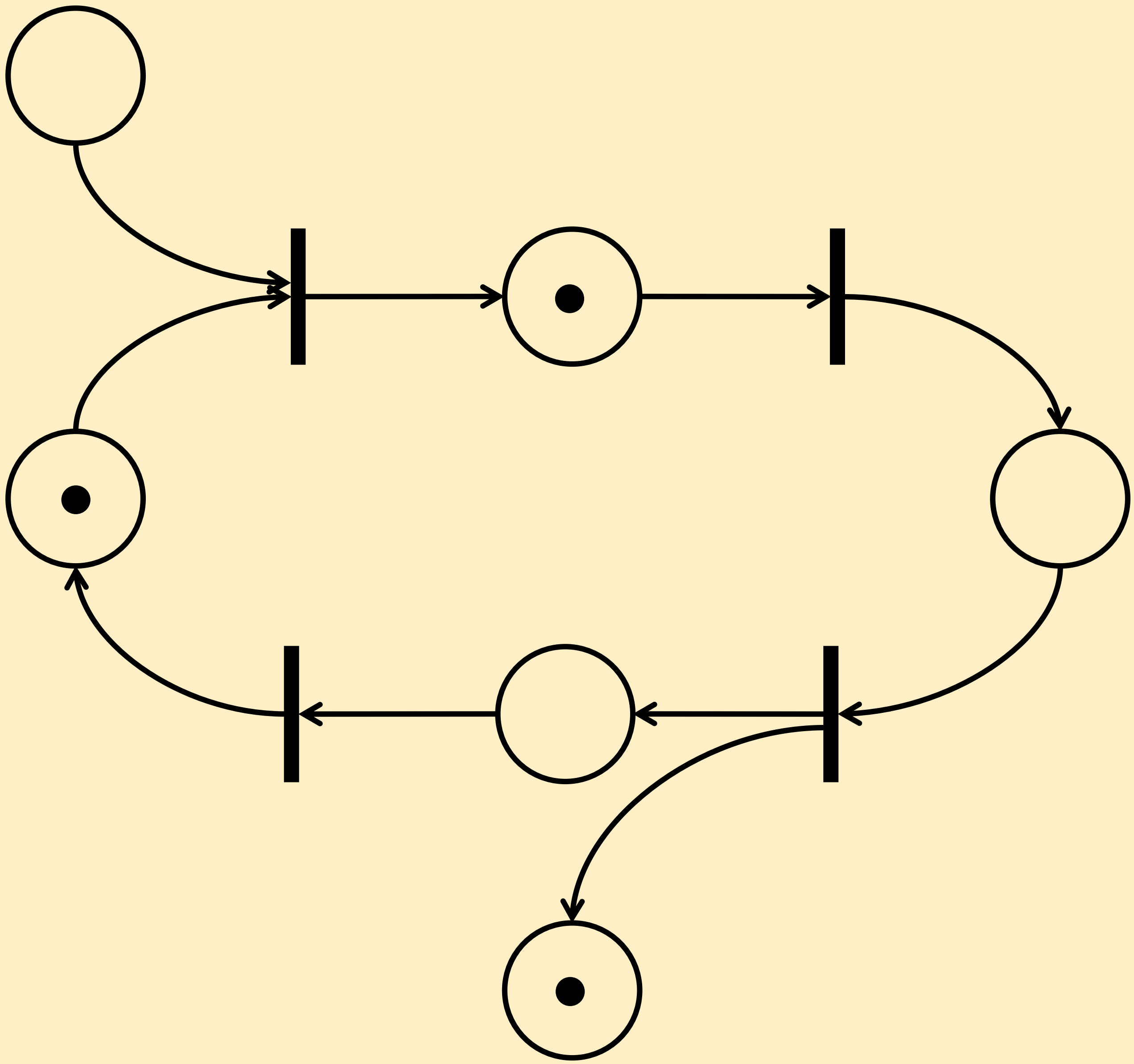
Change occurs: If a transition “fires”

- A transition can only fire if it is enabled
  - For each incoming arc of the transition:
    The place connected to the arc (input place) has a token
- Firing the transition
  - Removing a token from each input place
  - Putting a token to each output place
- Tokens are not “moved”, they are removed and put!
  - It is possible to “consume” and “generate” tokens
- Token distribution vector (marking) changes: New state

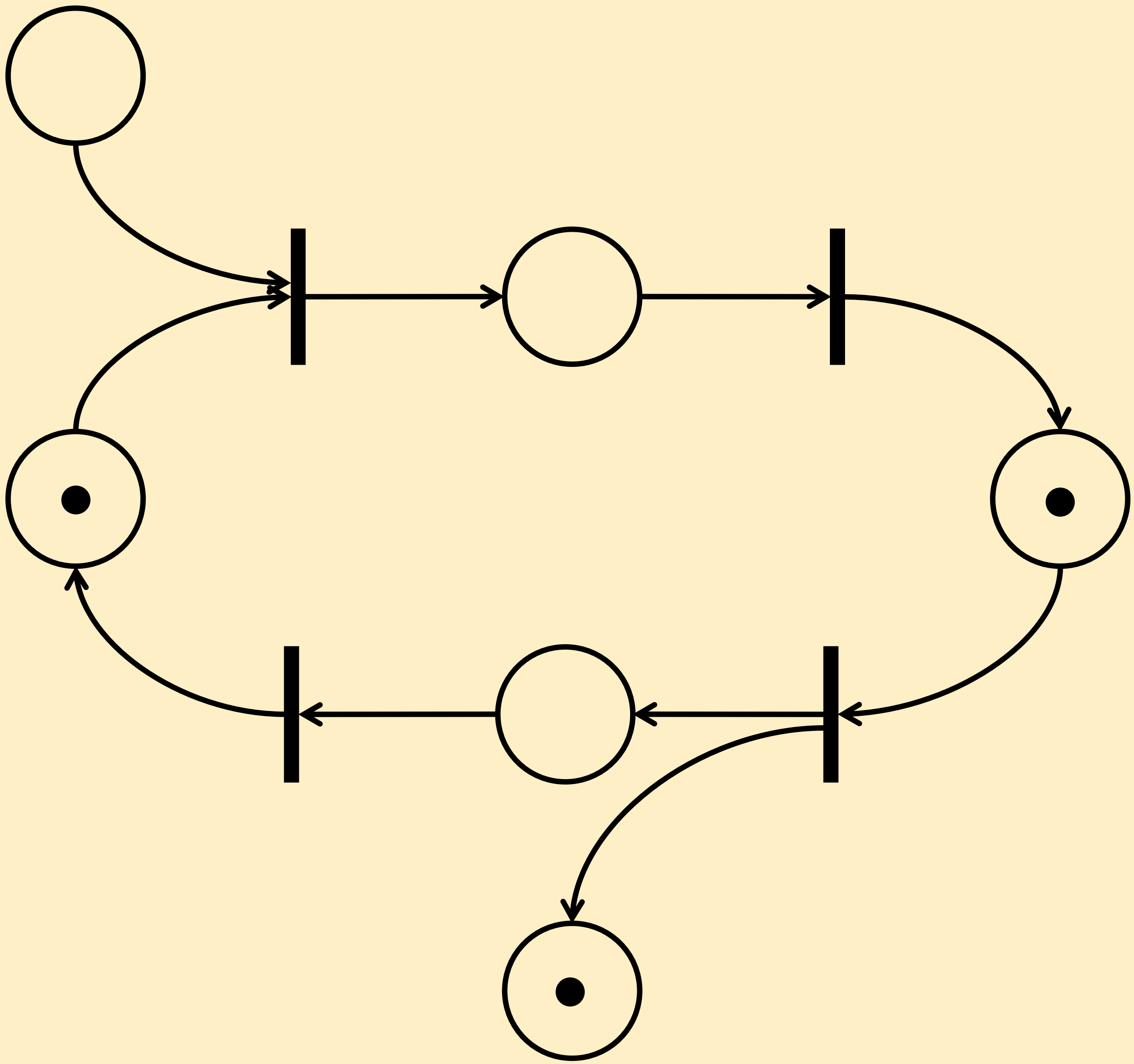
A simple example



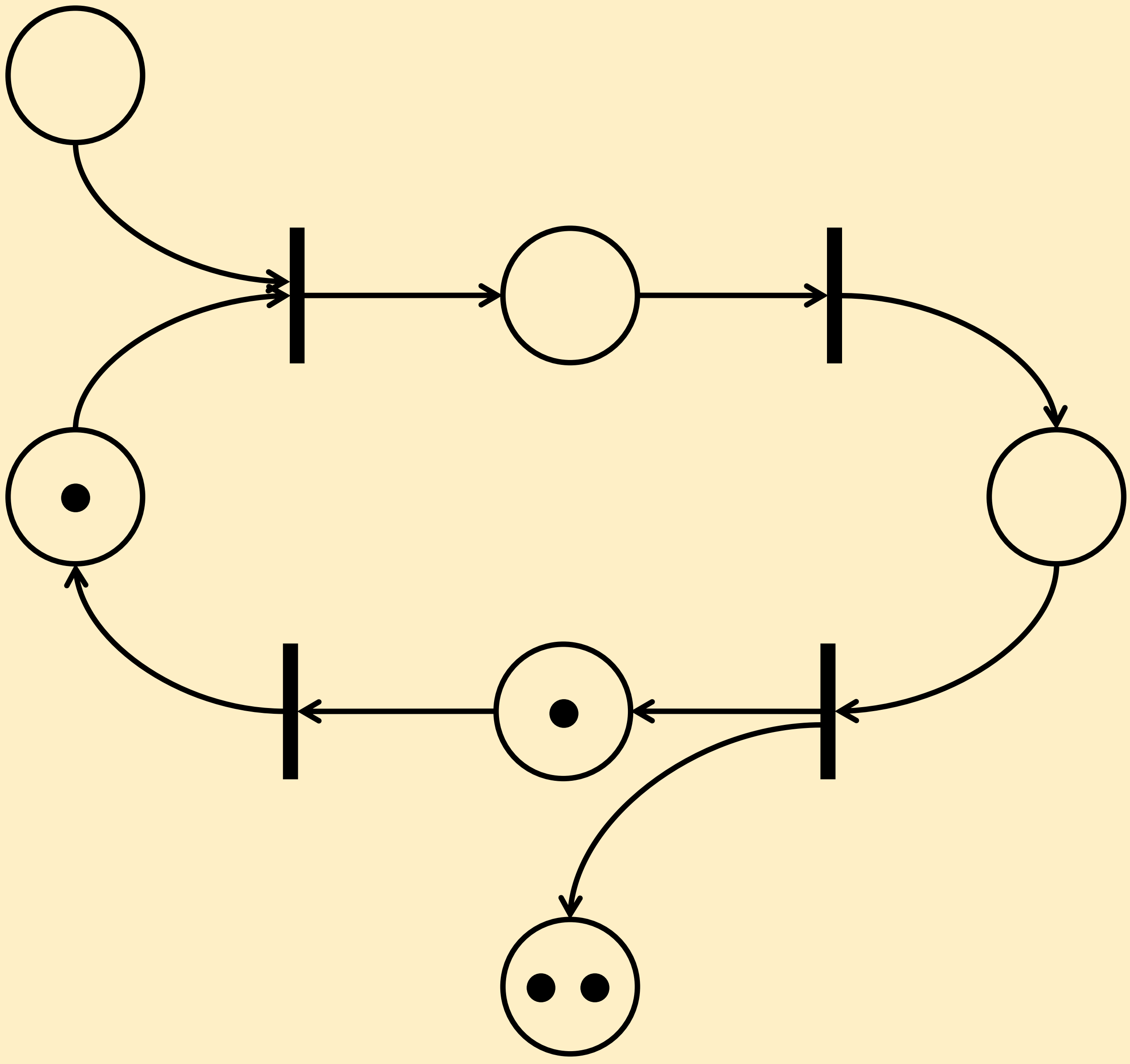
A simple example



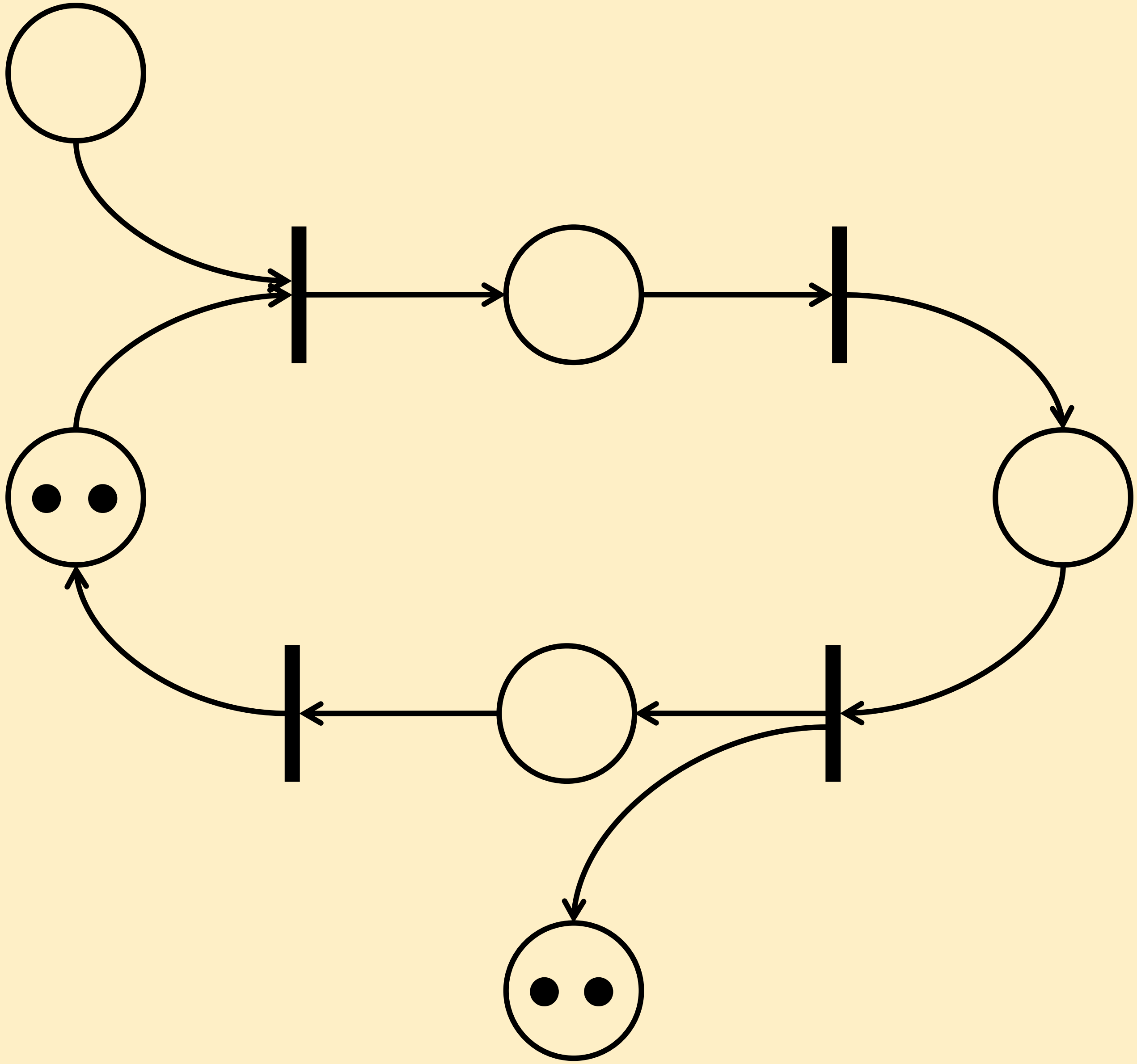
A simple example



A simple example



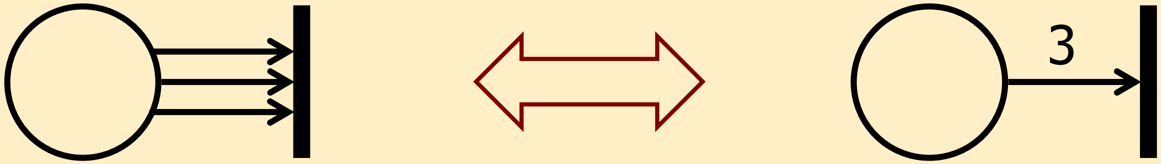
A simple example



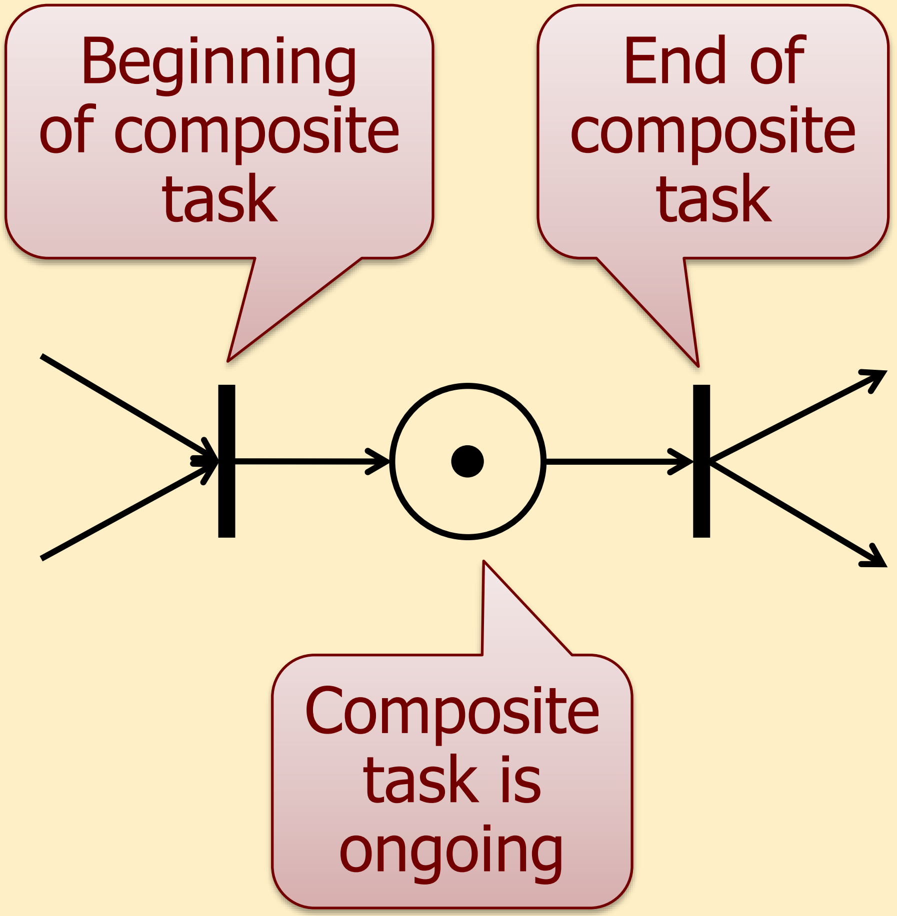
Multiple arcs

Arc weights:

- Each edge $e \in E$ can be associated with weight $w^*(e) \in \mathbf{N}^+$
- Edge e with weight $w^*(e)$ is equivalent to w_e parallel edges
- Parallel edges are not drawn, arc weight is used
- A weight of one is usually not denoted

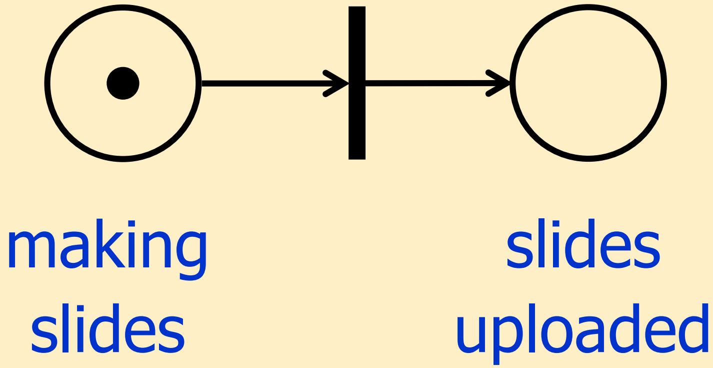


Properties of Petri nets

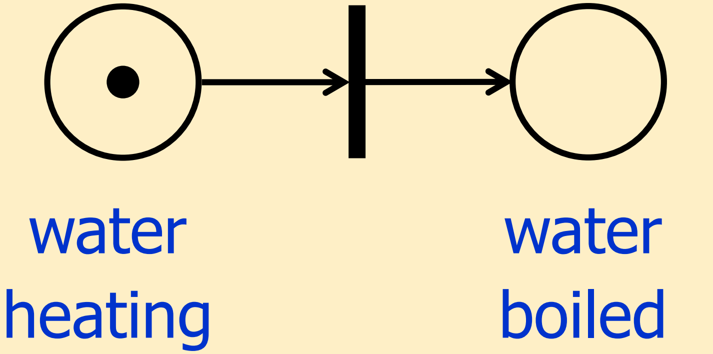


Petri Net Property	Modeling Property
„instant" firings	basic (atomic) events

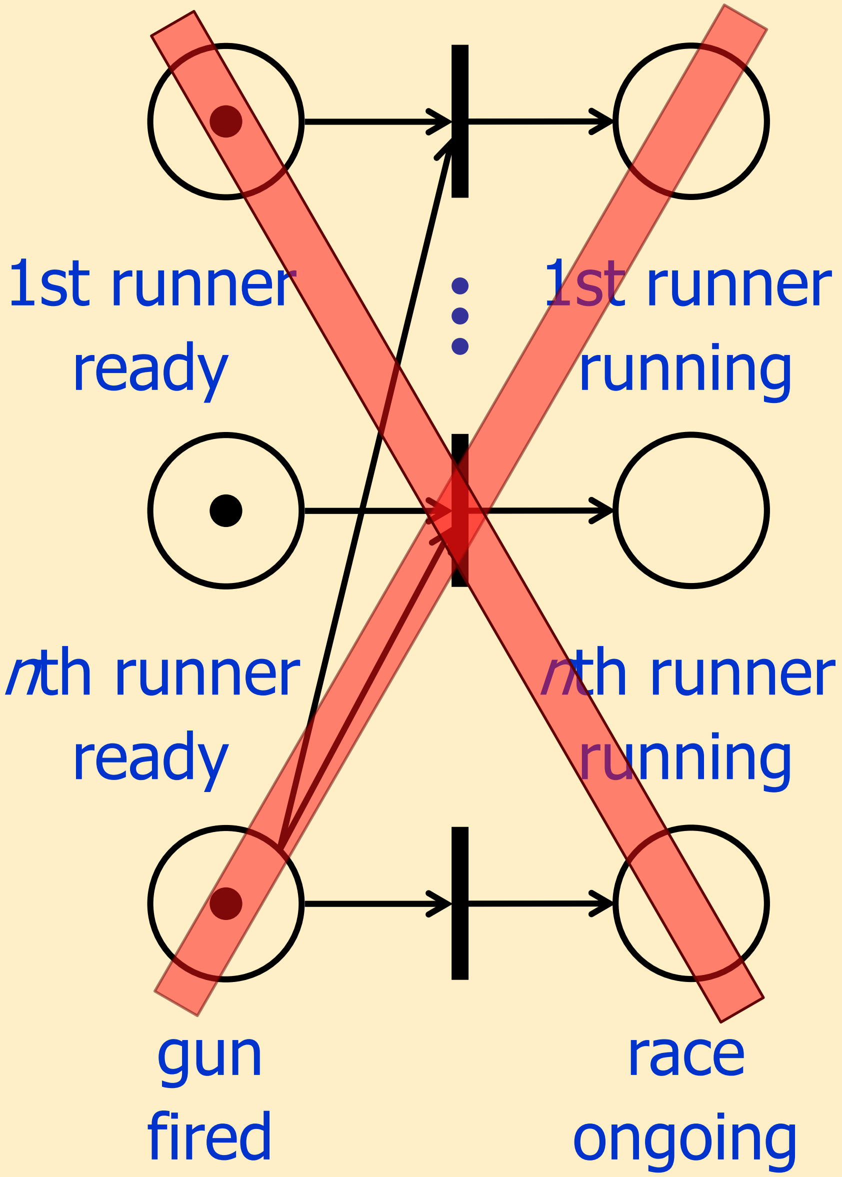
Properties of Petri nets

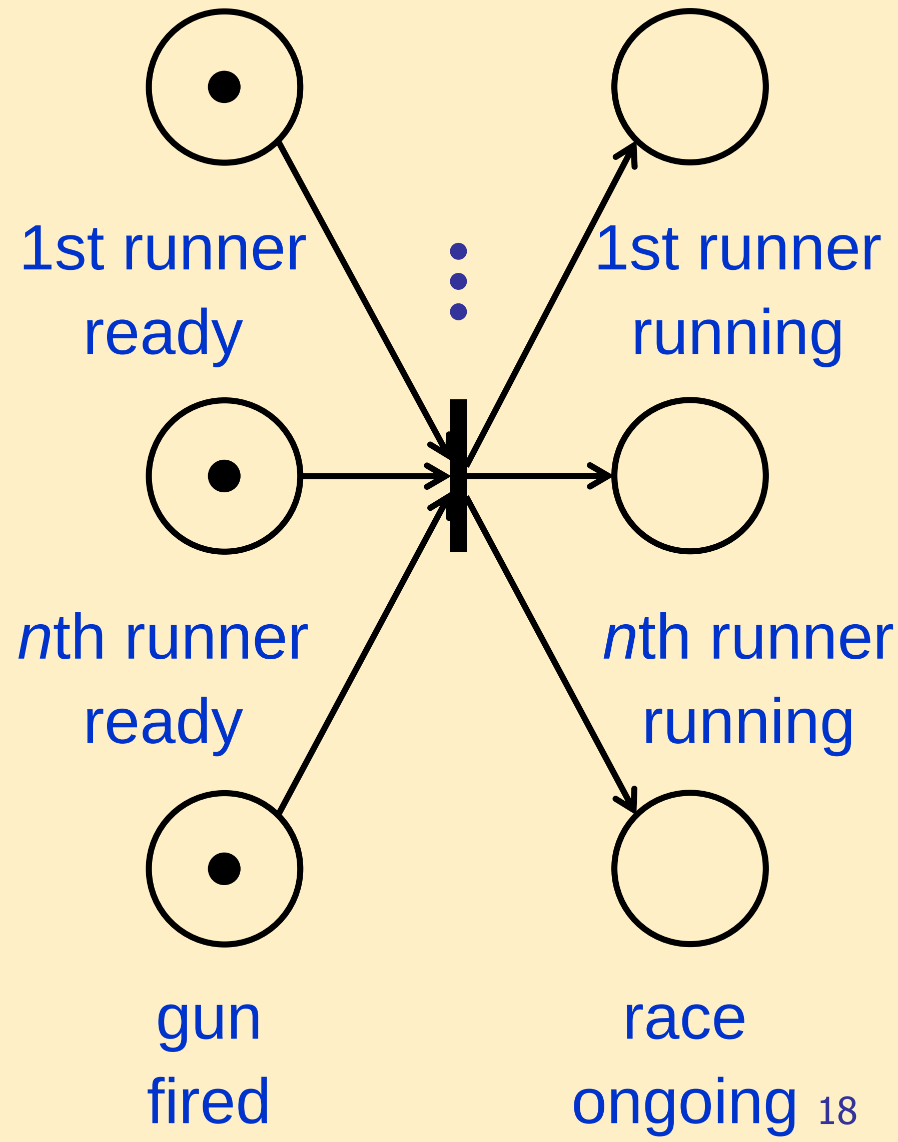


Petri Net Property	Modeling Property
„instant" firings	basic (atomic) events
asynchronous firings	concurrent / independent events

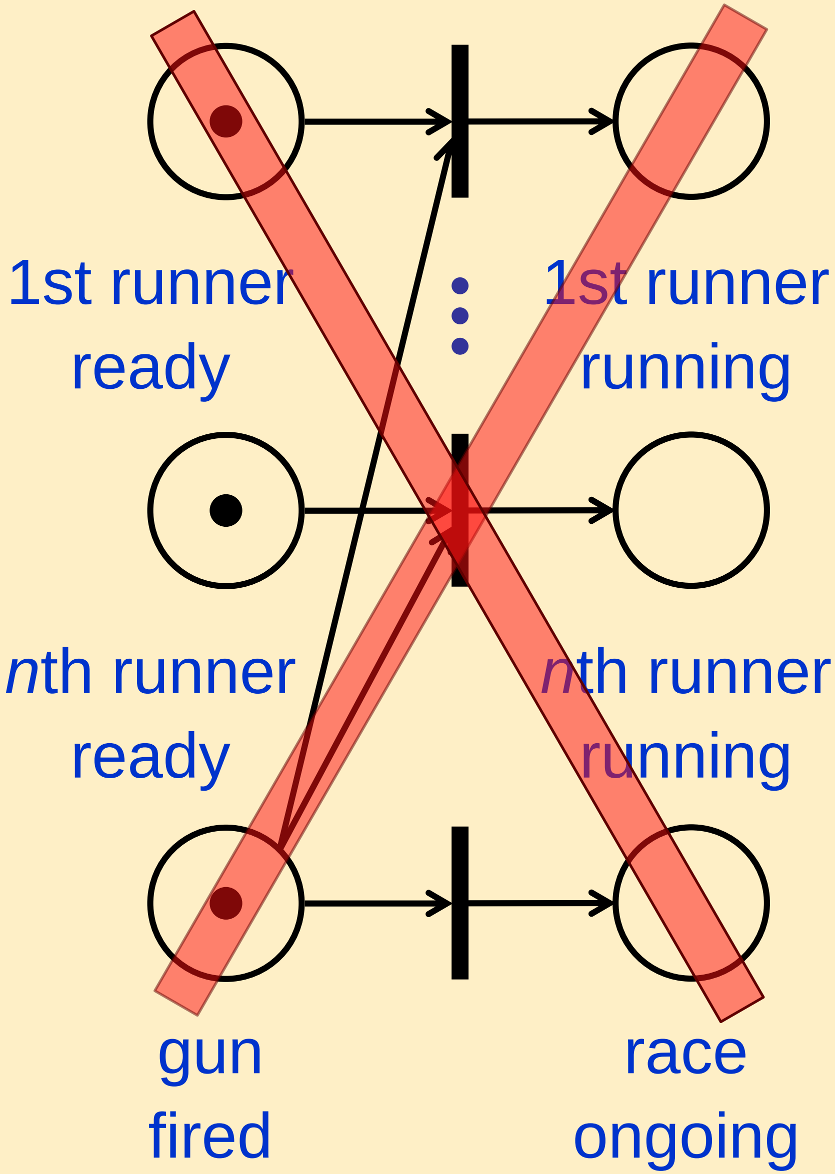


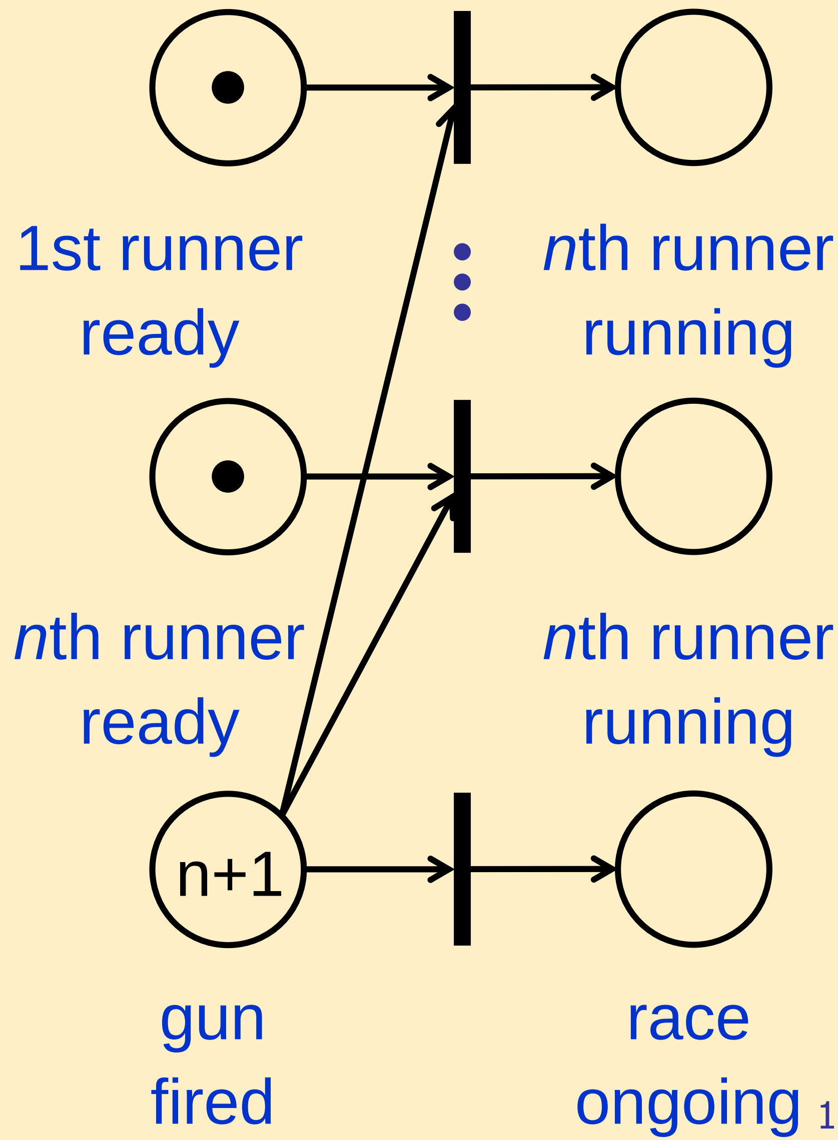
Simultaneousness, synchronization



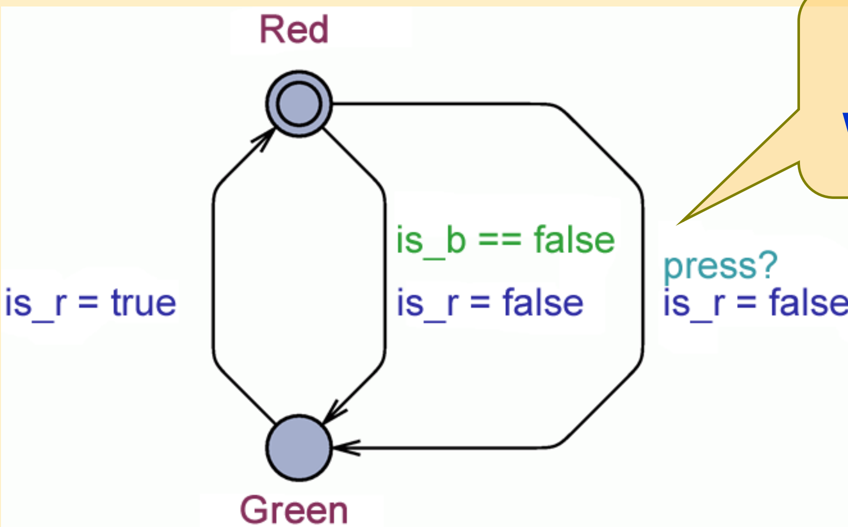


Simultaneousness, synchronization



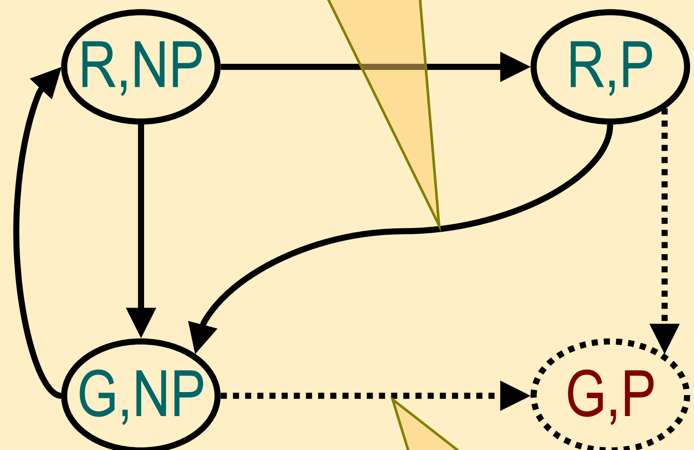


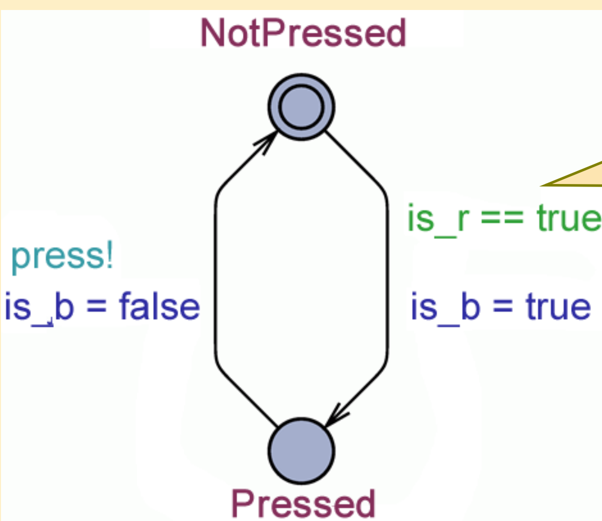
Example: Pedestrian light with button



Change with button

Synchronization (press!, press?)

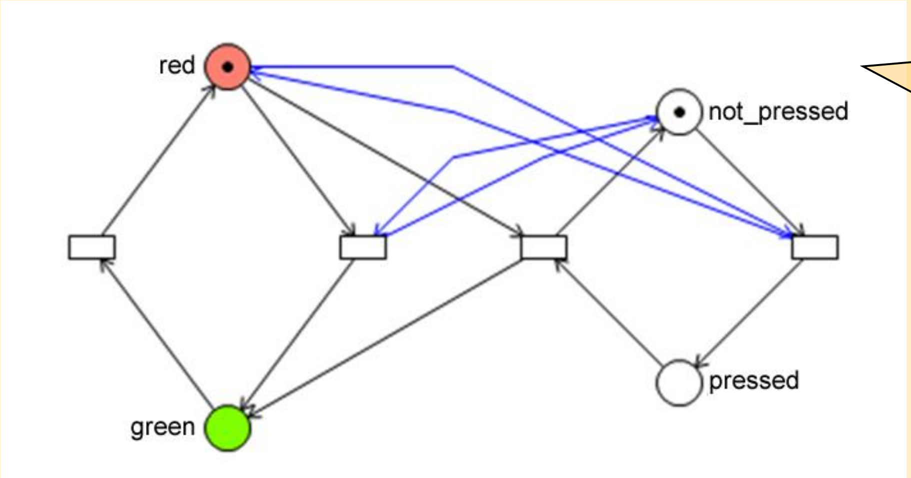




Presses when red

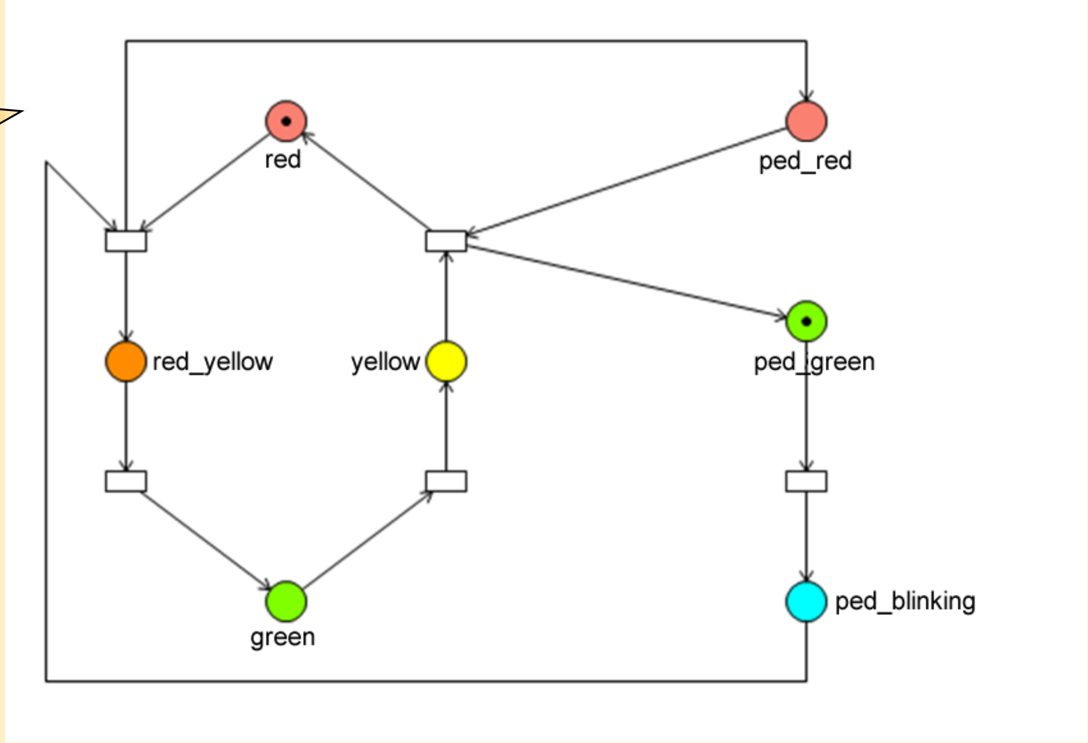
Constraint condition (is_r == true)

Example: Pedestrian light with button

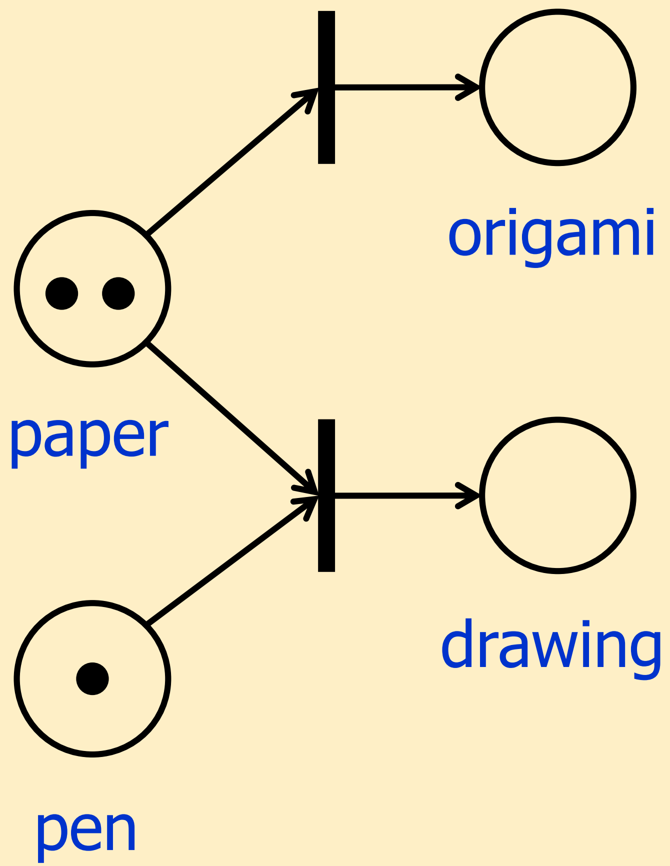


Pedestrian crossing with light and button

Crossing with traffic and pedestrian light

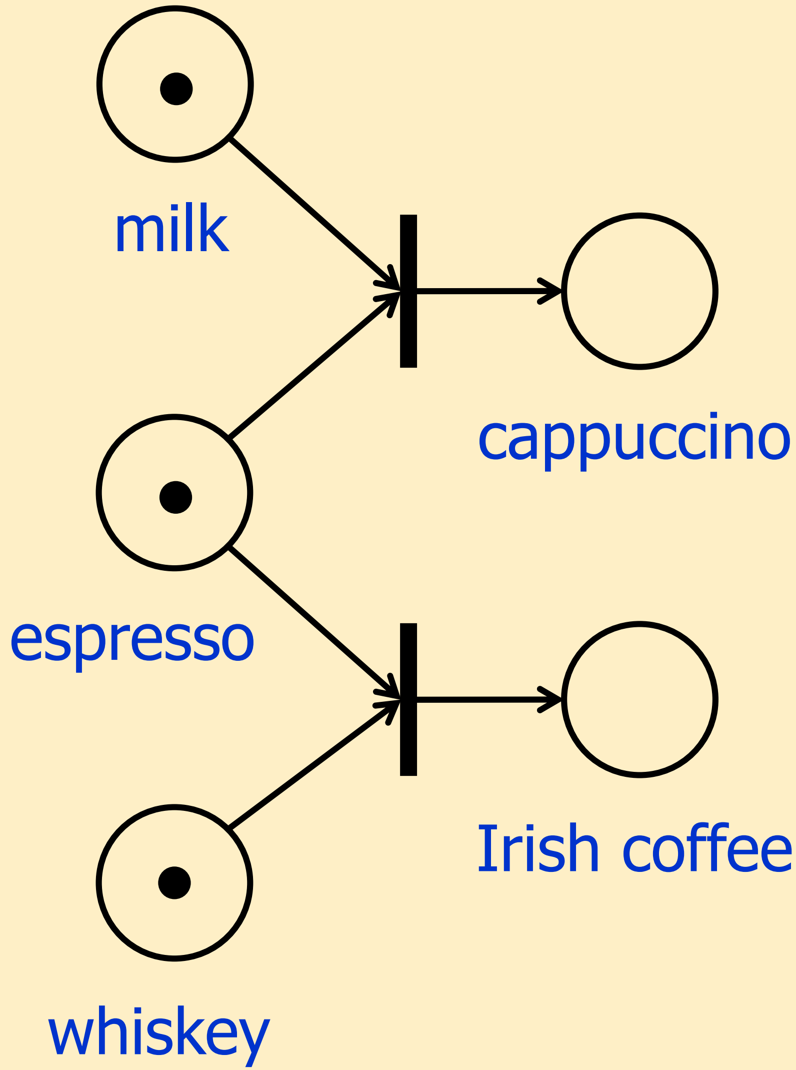


Properties of Petri nets



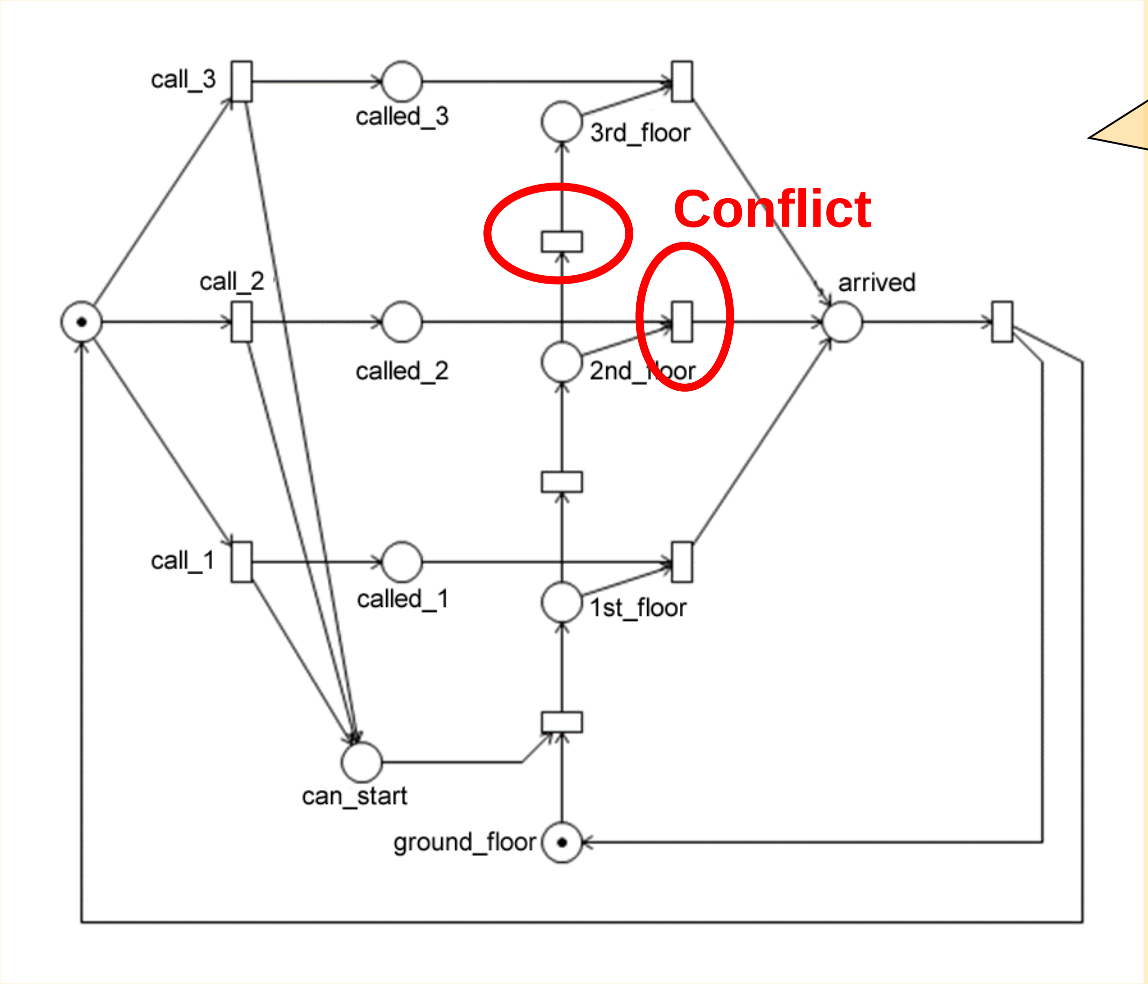
Petri Net Property	Modeling Property
„instant" firings	basic (atomic) events
asynchronous firings	concurrent / independent events
non-determinism	random events

Properties of Petri nets



Petri Net Property	Modeling Property
„instant" firings	basic (atomic) events
asynchronous firings	concurrent / independent events
non-determinism	random events
conflicting transitions	excluding events

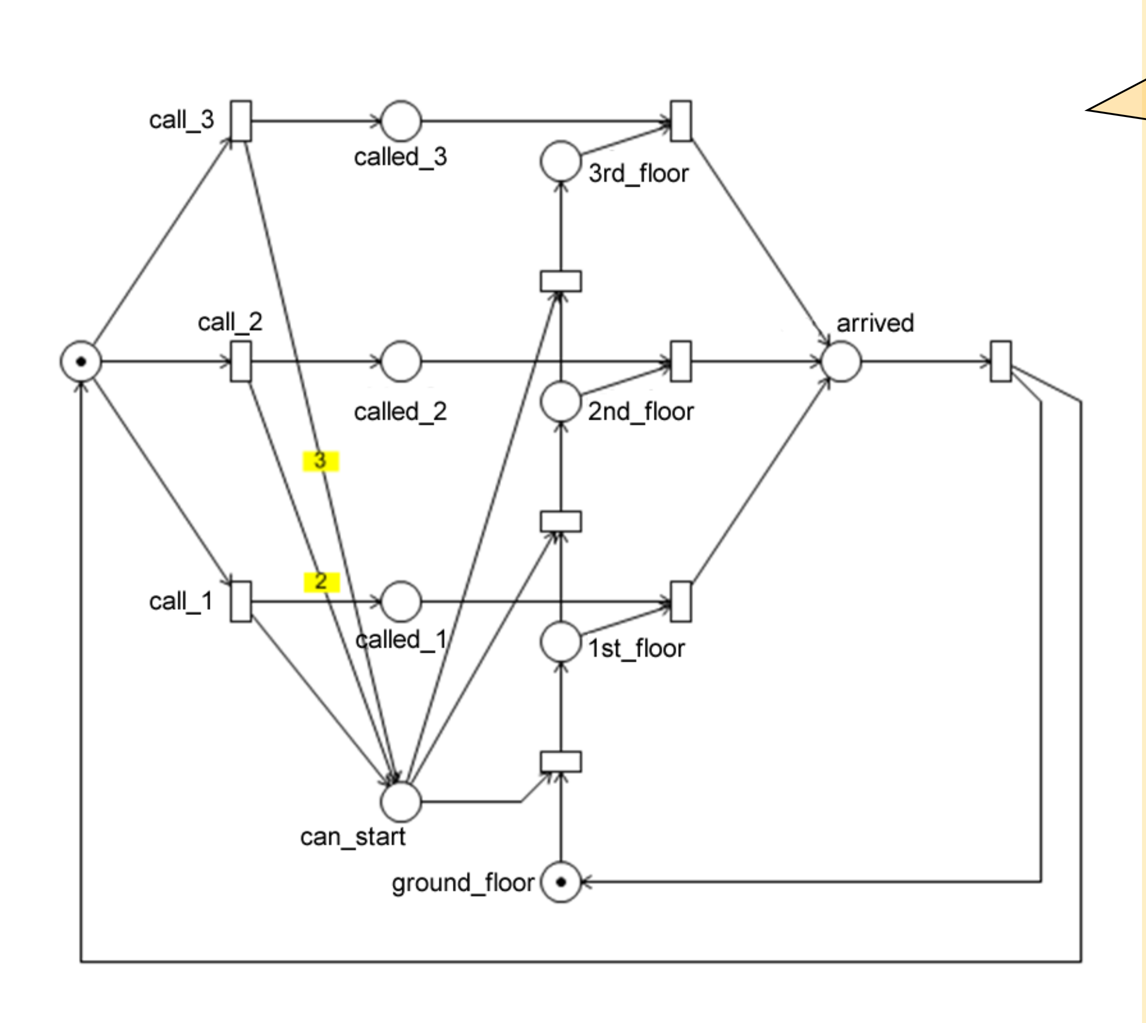
Simple models: Conflicts



Model of a food lift:

- Can be called from three levels, stops at desired level
- Model is incorrect

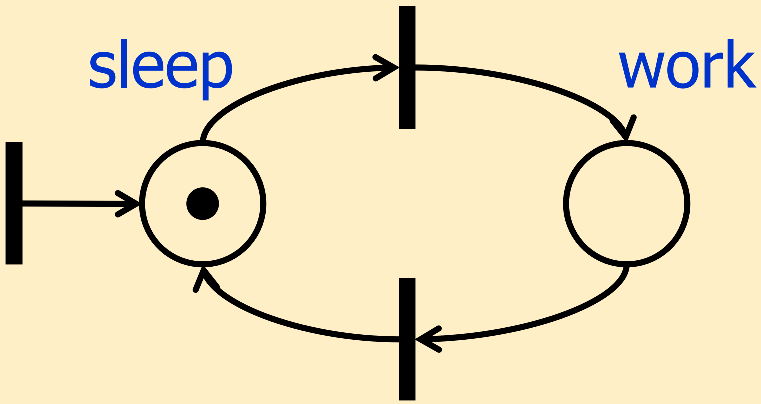
Simple models: Conflicts

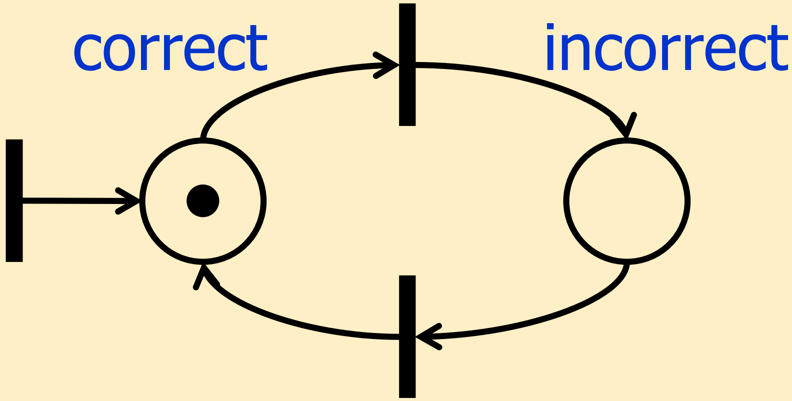


Correction:

- Condition for moving up and adding weights

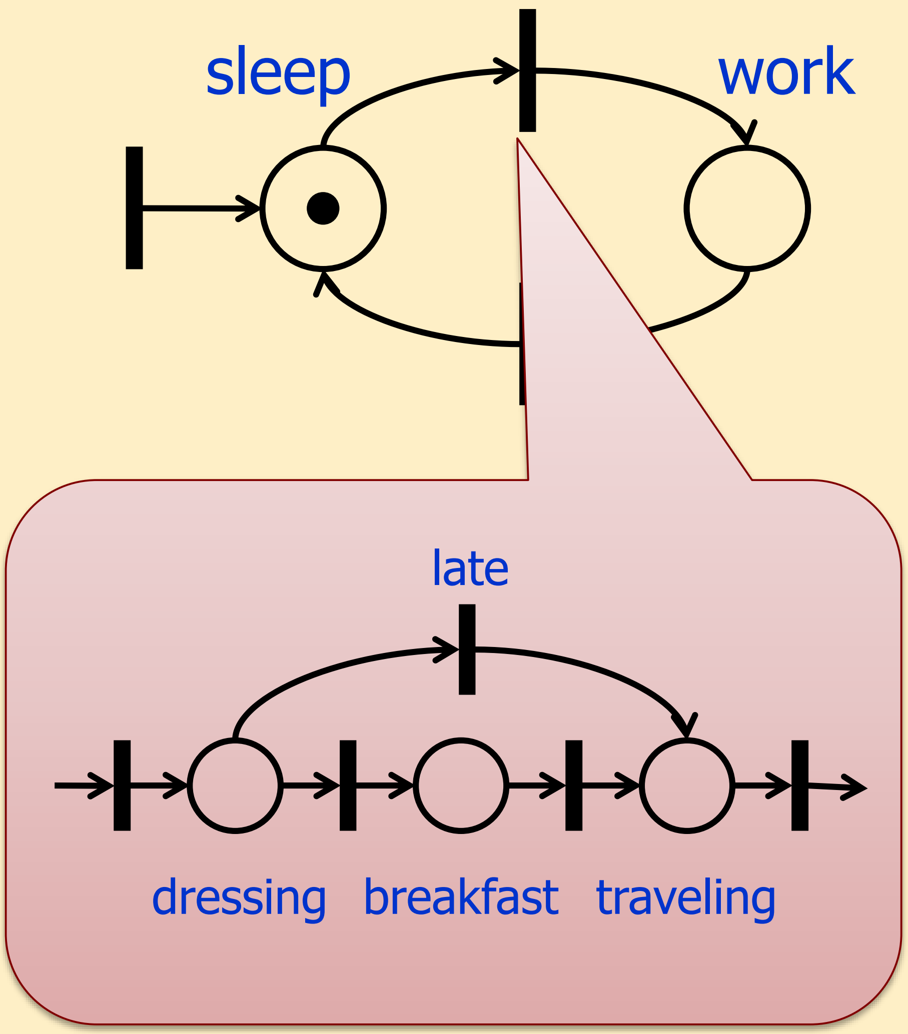
Properties of Petri nets





Petri Net Property	Modeling Property
„instant" firings	basic (atomic) events
asynchronous firings	concurrent / independent events
non-determinism	random events
conflicting transitions	excluding events
uninterpreted elements	abstract events

Properties of Petri nets



Petri Net Property	Modeling Property
„instant” firings	basic (atomic) events
asynchronous firings	concurrent / independent events
non-determinism	random events
conflicting transitions	excluding events
uninterpreted elements	abstract events
abstraction and refinement	hierarchical modeling

Summary of basic definitions

Petri net:

- Nondeterministic finite automaton
- State: token distribution vector
- Transition relation: transitions

Structure:

- Each place is a logical condition
- Structure of the net follows the decomposition of the modeled task

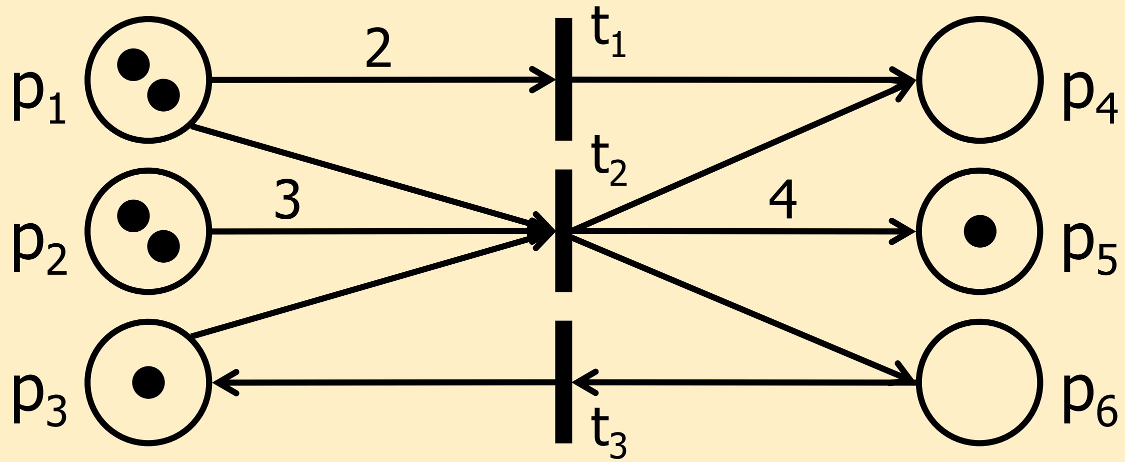
Topology and notations

- Input and output elements of places and transitions:
  - Input places of $t \in T$: $\bullet t = \{p \mid (p,t) \in E\}$
  - Output places of $t \in T$: $t\bullet = \{p \mid (t,p) \in E\}$
  - Input transitions of $p \in P$: $\bullet p = \{t \mid (t,p) \in E\}$
  - Output transitions of $p \in P$: $p\bullet = \{t \mid (p,t) \in E\}$
- For subsets of places $P' \subseteq P$ and transitions $T' \subseteq T$:

$$\bullet P' = \bigcup_{p \in P'} \bullet p \qquad \bullet T' = \bigcup_{t \in T'} \bullet t$$

$$P'\bullet = \bigcup_{p \in P'} p\bullet \qquad T'\bullet = \bigcup_{t \in T'} t\bullet$$

Topology example



$\bullet p_1 = \varnothing$

$\bullet p_2 = \varnothing$

$\bullet p_3 = \{t_3\}$

$\bullet p_4 = \{t_1, t_2\}$

$\bullet p_5 = \{t_2\}$

$\bullet p_6 = \{t_2\}$

$p_1 \bullet = \{t_1, t_2\}$

$p_2 \bullet = \{t_2\}$

$p_3 \bullet = \{t_2\}$

$p_4 \bullet = \varnothing$

$p_5 \bullet = \varnothing$

$p_6 \bullet = \{t_3\}$

$\bullet t_1 = \{p_1\}$

$\bullet t_2 = \{p_1, p_2, p_3\}$

$\bullet t_3 = \{p_6\}$

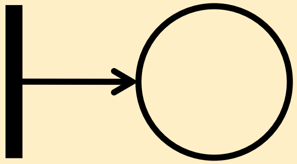
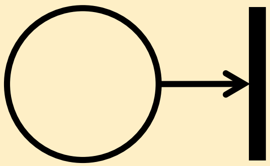
$t_1 \bullet = \{p_4\}$

$t_2 \bullet = \{p_4, p_5, p_6\}$

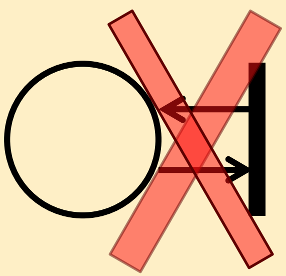
$t_3 \bullet = \{p_3\}$

Special nodes and nets

Source and sink transitions:

- A transition $t \in T$ is a source transition:
  
  - Has no input place, i.e., $\bullet t = \emptyset$
  - Source transitions can always fire
- A transition $t \in T$ is a sink transition:
  
  - Has no output place, i.e., $t \bullet = \emptyset$

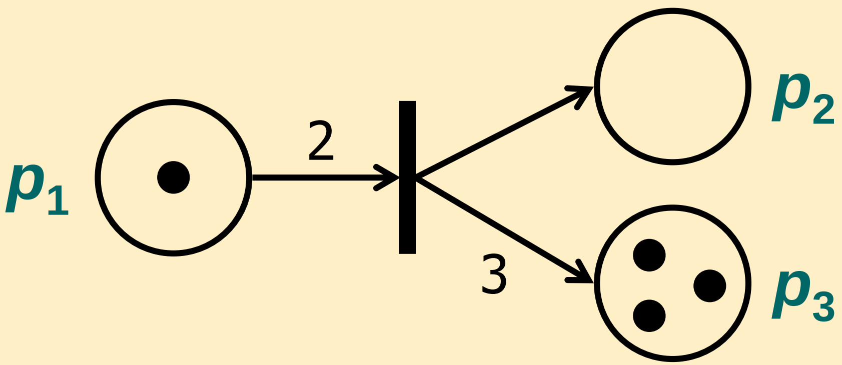
Pure Petri nets:

- A PN is pure, if it has no self-loops, i.e., $\forall t \in T: \bullet t \cap t \bullet = \emptyset$
  

State vector: Token distribution vector (marking)

$$M = \begin{bmatrix} m_1 \\ \vdots \\ m_\pi \end{bmatrix}$$

- Initial state: M_0 initial token distribution (marking)
- Example:



$$M = \begin{array}{c} \begin{bmatrix} 1 \\ 0 \\ 3 \end{bmatrix} \end{array} \begin{array}{l} \leftarrow p_1 \\ \leftarrow p_2 \\ \leftarrow p_3 \end{array}$$

Summary of structure

Petri net (PN): $PN = \langle P, T, E, W, M_0 \rangle$

- Places $P = \{p_1, p_2, \ldots, p_\pi\}$
- Transitions (firings) $T = \{t_1, t_2, \ldots, t_\tau\}$

  $P \cap T = \varnothing$
- Arcs $E \subseteq (P \times T) \cup (T \times P)$
- Weight function $\mathrm{W} : E \rightarrow \mathbf{N}^+$
- Initial state $M_0 : P \rightarrow \mathbf{N}$

PN structure: $N = \langle P, T, E, W \rangle$

PN with initial state: $PN = \langle N, M_0 \rangle$

Dynamic behavior: Enabling, firing, firing sequence

Dynamic behavior

A step in Petri nets (change of state):

"Firing" of a transition

- Original state: original token distribution
- Firing
    1. Transition is enabled
    2. Remove tokens from input places
    3. Put tokens to output places
- New state: new token distribution

Conditions for enabling a transition

- A transition $t \in T$ is enabled, if each input place is marked with at least as many tokens as the weight of the arc outgoing from the place
  - I.e., a transition $t \in T$ is enabled, if each input place is marked with at least $w^-(p, t)$ tokens
  - Here $w^-(p, t)$ is the weight $w^*(e)$ of the arc $e = (p, t)$ from p to t
- Formally:
  - Firing of a transition t is enabled, if

$$\boxed{\forall p \in \bullet t : m_p \geq w^-(p,t)}$$

Occurrence of a firing

- An enabled transition can fire
  - I.e., it may fire or not ("fire at will")
- A single transition can fire at once
- If multiple transitions are enabled:
  - One enabled transition has to be picked that can fire
  - Random choice $\Rightarrow$ Non-deterministic behavior

Non-determinism and timing

- Semantics of “fire at will”:
  - Implicit concept of time
  - No time scale
  - Firing can occur at any time in the time interval $[0, \infty)$
- Assigning concrete timestamps to firings:
  - A non-deterministic non-timed Petri net with the same structure and initial state covers all possible firing sequences of a timed Petri net

Change of state

Firing of a transition:

- Removes $w^{-}(p, t)$ tokens from input places $p \in \bullet t$
  - $w^{-}(p, t)$ is the weight of arc $p \rightarrow t$
- Puts $w^{+}(t, p)$ tokens to output places $p \in t\bullet$
  - $w^{+}(t, p)$ is the weight of arc $t \rightarrow p$

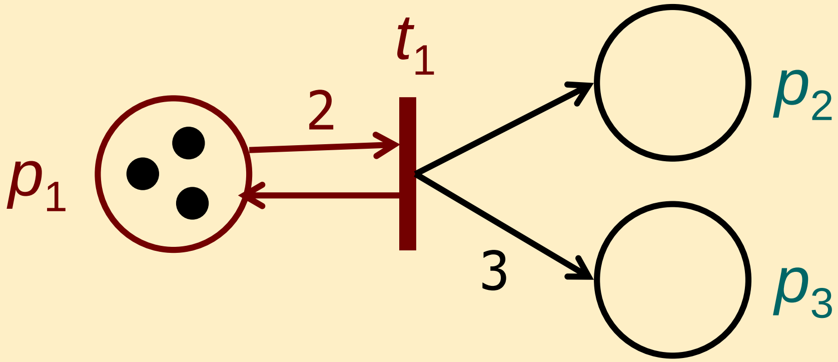
If transition t fires under marking M

- New marking: $\boxed{M' = M + \mathbf{W}^{T} \cdot e_t}$
  - where e_t is the unit vector for transition t
  - where $\mathbf{W}^{T}$ is the transposed weighted incidence matrix

Weighted incidence matrix

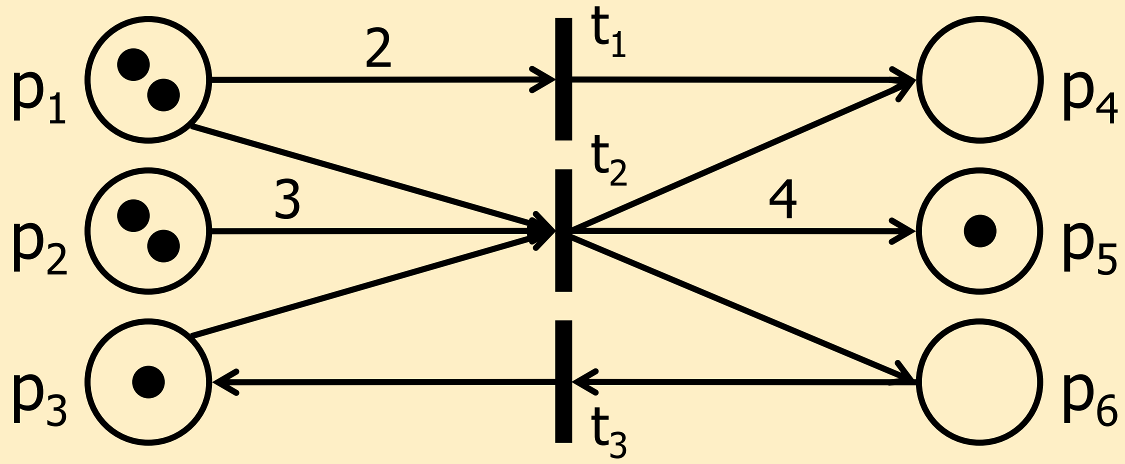
- Weighted incidence matrix: $\mathbf{W} = [\mathrm{w}(t, p)]$
- Dimensions: $\tau \times \pi = |T| \times |P|$
- $\mathrm{w}(t, p)$ denotes the change in the number of tokens in p if t fires:

$$\mathrm{w}(t, p) = \begin{cases} \mathrm{w}^{+}(t, p) - \mathrm{w}^{-}(p, t) & \text{if } (t, p) \in E \text{ or } (p, t) \in E \\ 0 & \text{if } (t, p) \notin E \text{ and } (p, t) \notin E \end{cases}$$



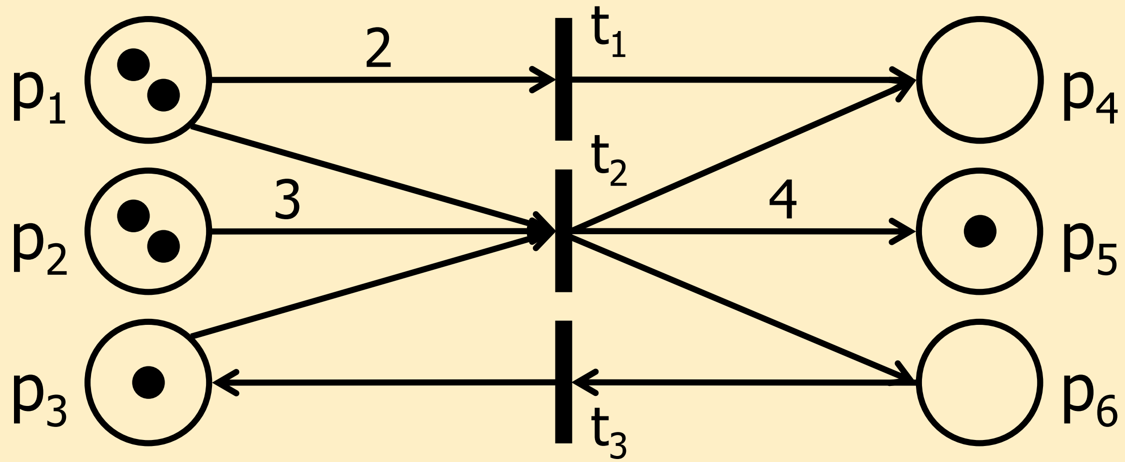
$\mathrm{w}(t_1, p_1) =$
$\mathrm{w}^{+}(t_1, p_1) - \mathrm{w}^{-}(p_1, t_1) =$
$1 - 2 = -1$

Weighted incidence matrix example



$$
\begin{array}{cc}
 & \begin{array}{cccccc} p_1 & p_2 & p_3 & p_4 & p_5 & p_6 \end{array} \\
W = & \begin{bmatrix} -2 & 0 & 0 & 1 & 0 & 0 \\ -1 & -3 & -1 & 1 & 4 & 1 \\ 0 & 0 & 1 & 0 & 0 & -1 \end{bmatrix} \begin{array}{c} t_1 \\ t_2 \\ t_3 \end{array}
\end{array}
$$

Weighted incidence matrix example



$$
\begin{array}{cccccc|c}
p_1 & p_2 & p_3 & p_4 & p_5 & p_6 & \\
\hline
-2 & 0 & 0 & 1 & 0 & 0 & t_1 \\
-1 & -3 & -1 & 1 & 4 & 1 & t_2 \\
0 & 0 & 1 & 0 & 0 & -1 & t_3
\end{array}
$$

$$W = \begin{bmatrix} -2 & 0 & 0 & 1 & 0 & 0 \\ -1 & -3 & -1 & 1 & 4 & 1 \\ 0 & 0 & 1 & 0 & 0 & -1 \end{bmatrix}$$

$$W^+ = \begin{bmatrix} 0 & 0 & 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 1 & 4 & 1 \\ 0 & 0 & 1 & 0 & 0 & 0 \end{bmatrix}$$

$$W^- = \begin{bmatrix} 2 & 0 & 0 & 0 & 0 & 0 \\ 1 & 3 & 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 1 \end{bmatrix}$$

$$\boldsymbol{W = W^+ - W^-}$$

Firing sequence

- State trajectory
  - States during a sequence of firings
- Firing sequence

$$\underline{\sigma} = \langle M_{i0}\ t_{i1}\ M_{i1} \ldots t_{in}\ M_{in} \rangle \text{ or } \underline{\sigma} = \langle t_{i1} \ldots t_{in} \rangle$$

- If all transitions satisfy the firing rules:

  State M_{in} is reachable from M_{i0} with firing sequence $\underline{\sigma}$:

$$M_{i0}\ [\underline{\sigma} >\ M_{in}$$

Extensions of Petri nets: Modified firing semantics

Extensions of Petri nets

Goals:

- Increase modeling power
- Restrict non-deterministic behavior

Extensions to the formalism of Petri nets:

- Finite capacity places
- Inhibitor arcs
- Transitions with priority

Finite capacity places

- Until now places had infinite capacity
  - Number of tokens in each place is unbounded
  - Modeling infinite capacity and resources
    - E.g. unbounded place “running” means that any number of processes can be running at the same time
- Finite capacity Petri net
  - A capacity K(p) can be assigned to each place p: Maximal number of tokens on that place
  - Modeling finite capacities
    - E.g. place “running” with finite capacity: maximal number of processes running at the same time

Firing rule in finite capacity Petri nets

- Firing a transition $t \in T$ is enabled, if
  1. There are enough tokens on input places:

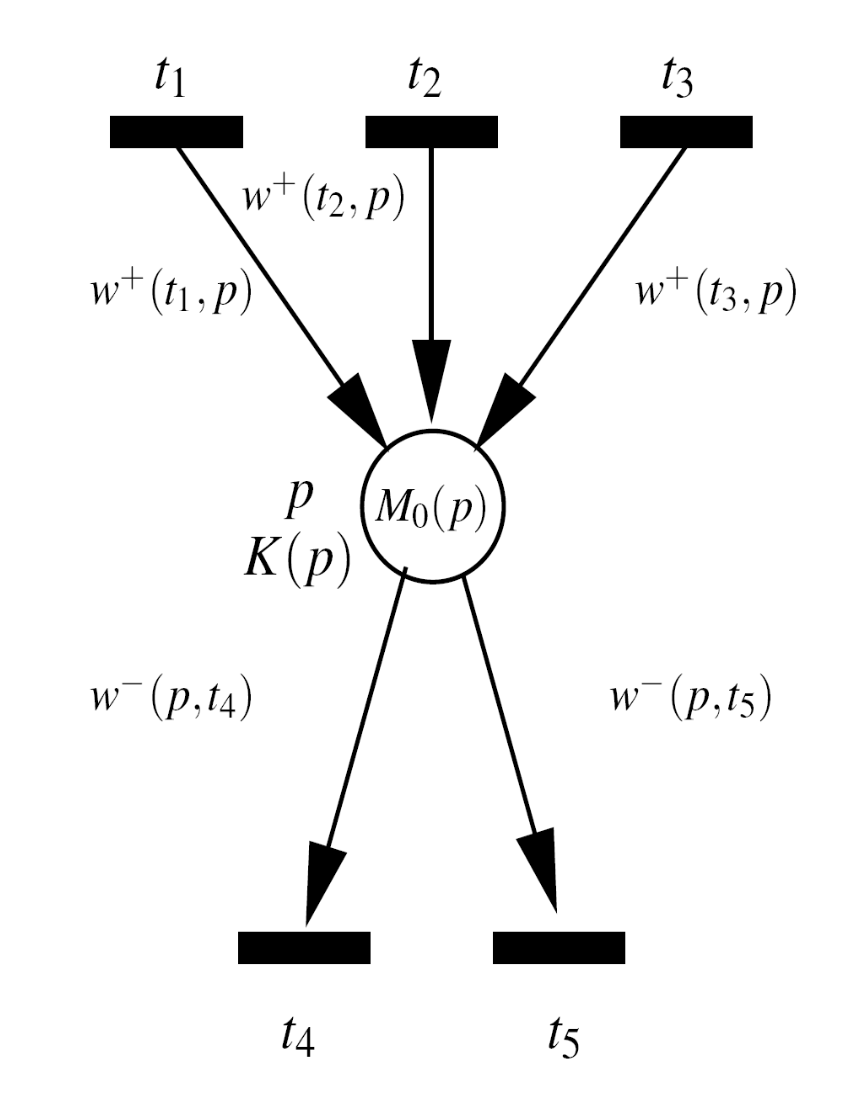
     $$\forall p \in \bullet t: \quad m_p \geq w^-(p,t)$$

  2. Capacity constraint holds after firing:

     $$\forall p \in t\bullet: \quad m'_p = m_p + w(t,p) \;\leq\; \boldsymbol{K(p)}$$

     i.e., firing the transition results in no more than K(p) tokens on each outgoing place p
- An enabled transition can fire at will
- After firing: $\forall p \in P: \; m'_p = m_p + w^+(t,p) - w^-(p,t)$

Place with finite capacity



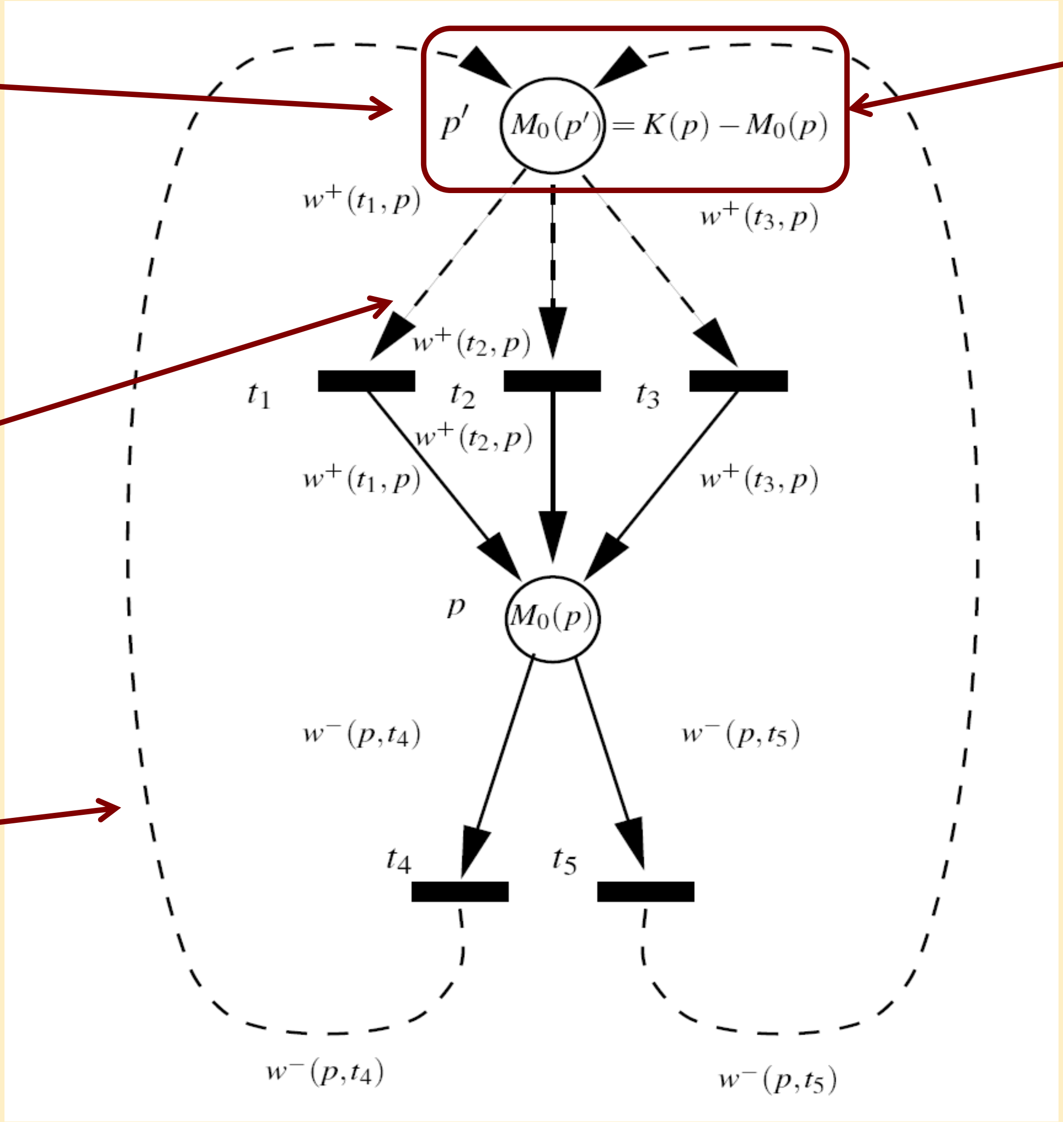
Can we avoid introducing a finite capacity for place p?

Equivalent infinite capacity net (pure PN)

Can only put tokens on p if there is free capacity.

Tokens put on p reduce the free capacity.

Tokens removed increase the free capacity.



Administrative place: Counting the free capacity

Only as many tokens can be put on p as the difference of the capacity and the initial marking (i.e., the free capacity).

Complementary place transformation 1/2

Complementary place transformation:

- Constructing an equivalent infinite capacity net from a finite capacity Petri net

Transformation process for pure Petri nets:

- For each finite capacity place p
  - Assign a complementary administrative place p’
  - The initial marking of p’ is

$$M_0(p') = K(p) - M_0(p)$$

i.e., the initial free capacity of p

Complementary place transformation 2/2

- Complementary arcs are drawn between place p′ and transitions $t \in \bullet p \cup p\bullet$
- Direction of the arcs depends on whether firing t increases or decreases the number of tokens on p:
  - If $w(t, p) < 0$, i.e., firing removes tokens from place p, then an arc (t, p′) with weight $|w(t, p)|$ is drawn between transition t and place p′
  - If $w(t, p) > 0$, i.e., firing puts tokens on place p, then an arc (p', t) with weight $w(t, p)$ is drawn between place p′ and transition t

Equivalence of the transformed net

- It can be shown that the complementary place transformation has the following properties:
  - If applying the strict firing rule (with capacity constraint) for a pure, finite capacity Petri net (N, M_0),
  - and applying the normal (weak) firing rule for the transformed net (N', M'_0),
  - then the firing sequences of the two nets will be identical.

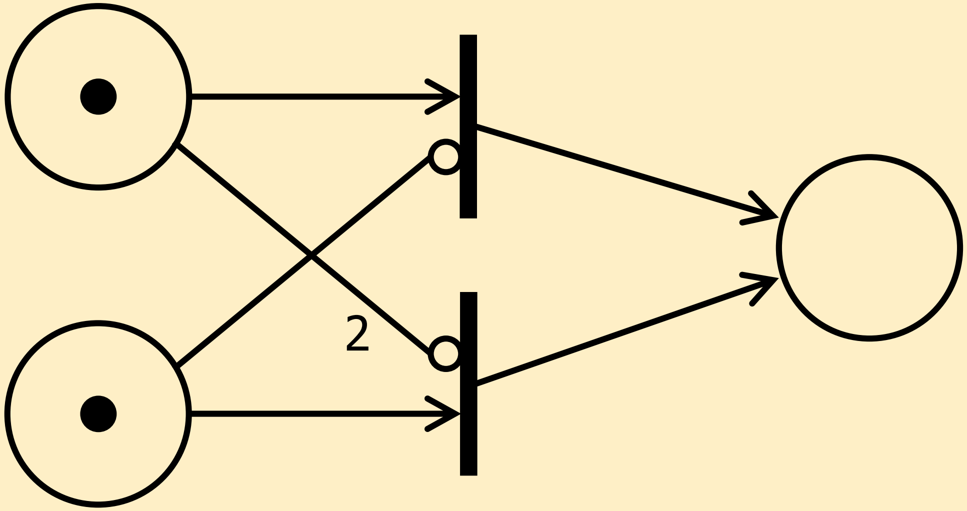
Prohibiting firing with inhibitor arcs

- Classic PN:
  - “Ponated” firing conditions: firing can occur if certain conditions hold for input places
- Expressing prohibition:
  - “Negated” firing conditions: Firing cannot occur under certain condition
  - Negated condition is checked on input places
  - Extension of the formalism: inhibitor arc

Firing rule with inhibitor arcs

- Extending the firing rule:

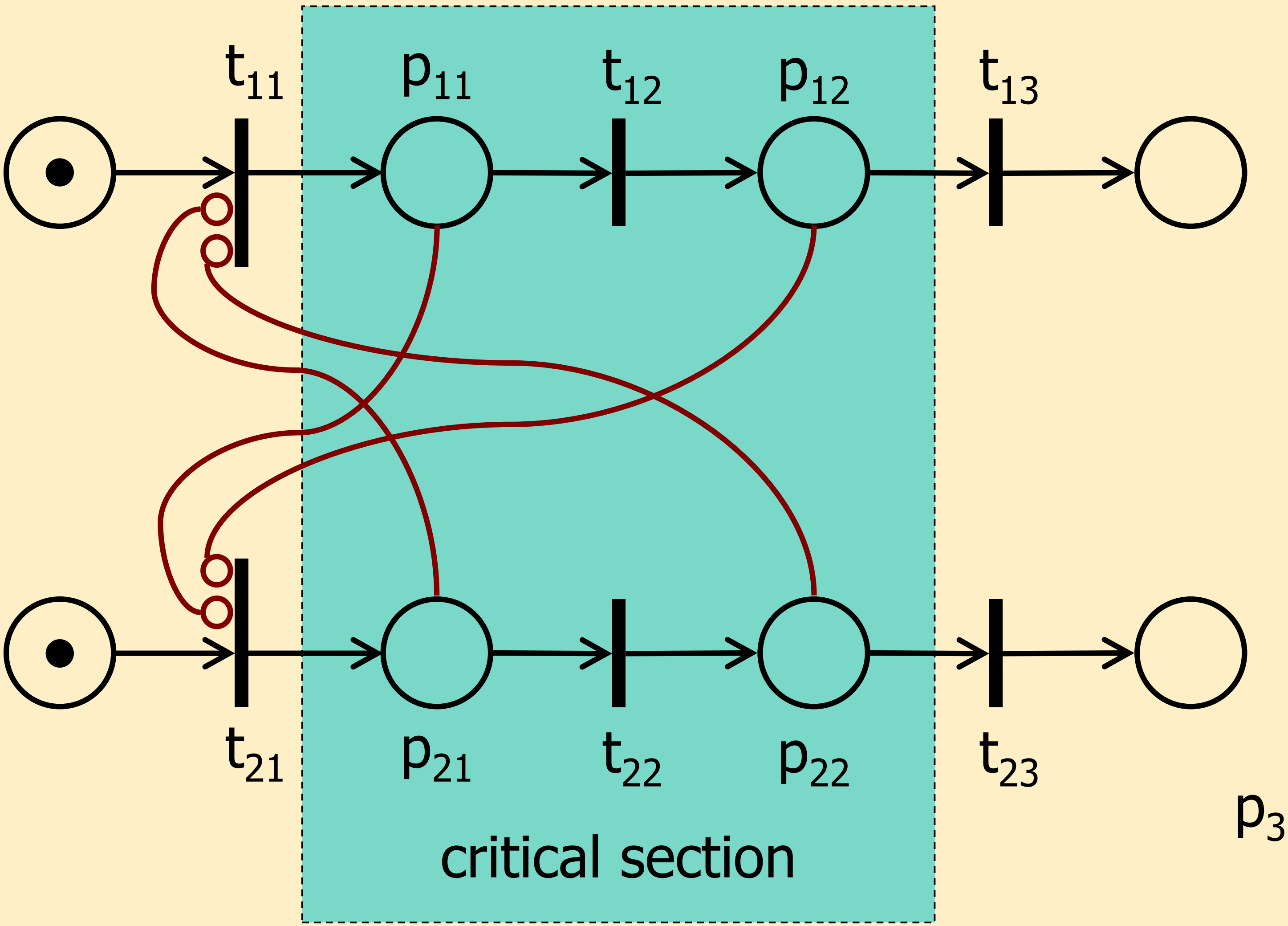
  If an input place p connected to a transition t with an inhibitor arc (p, t) is marked with at least $w^-(p, t)$ tokens, then the transition is not enabled



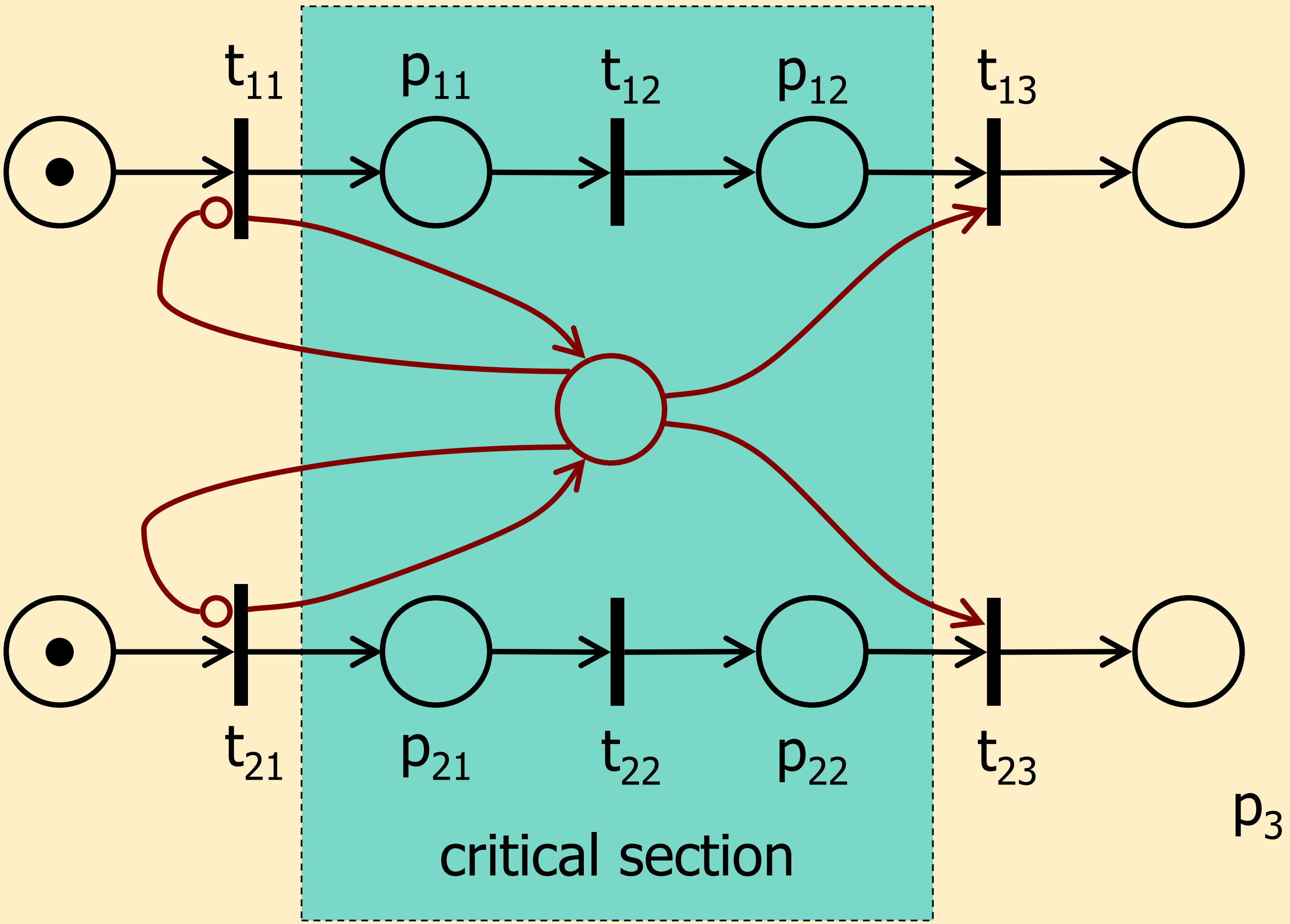
Using inhibitor arcs

- Advantage: Petri nets with inhibitor arcs are as expressive as Turing machines (Turing-complete)
- Disadvantage: many analysis methods cannot be applied to Petri nets with inhibitor arcs

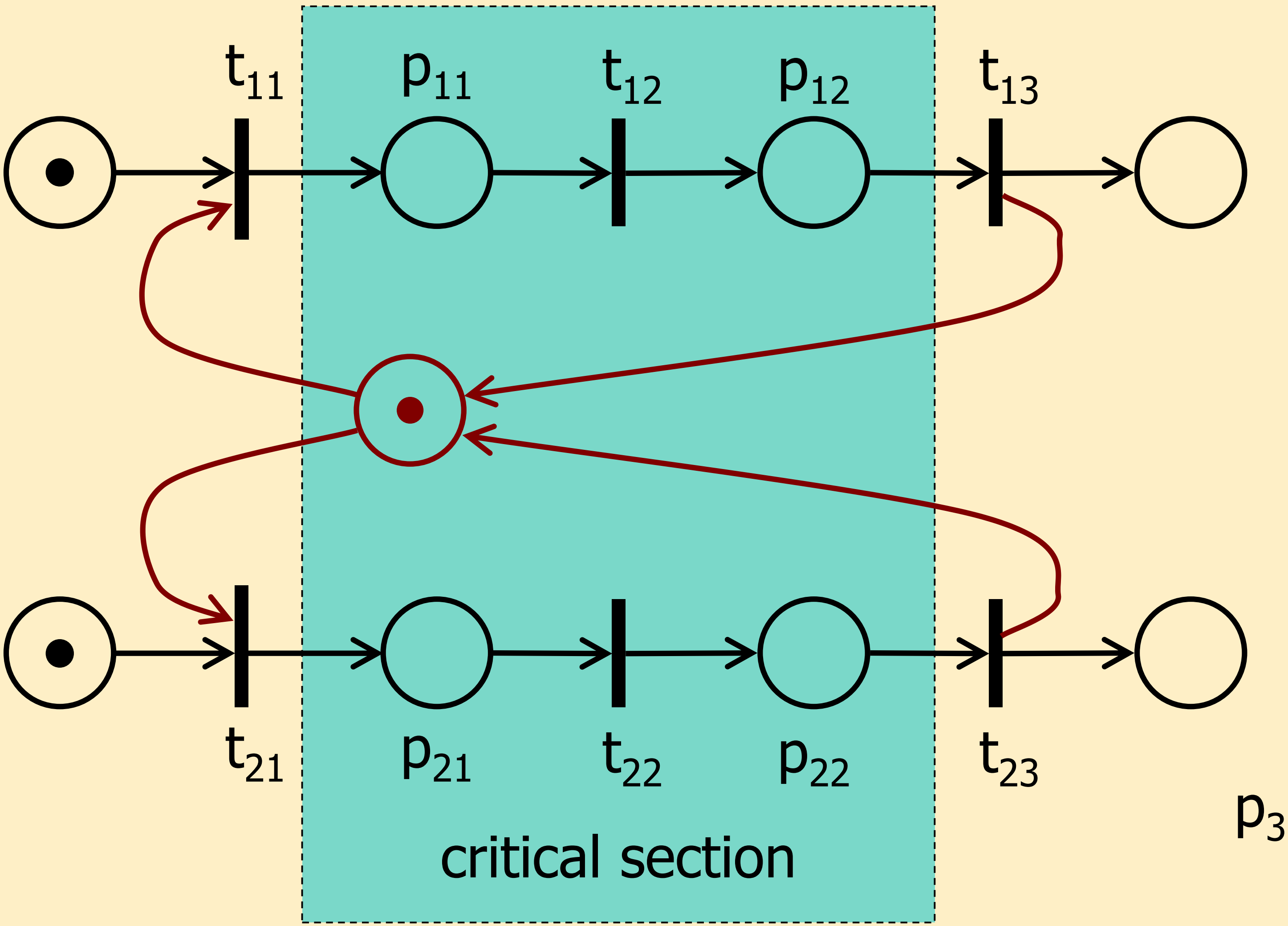
Example: Mutual exclusion with inhibitor arcs



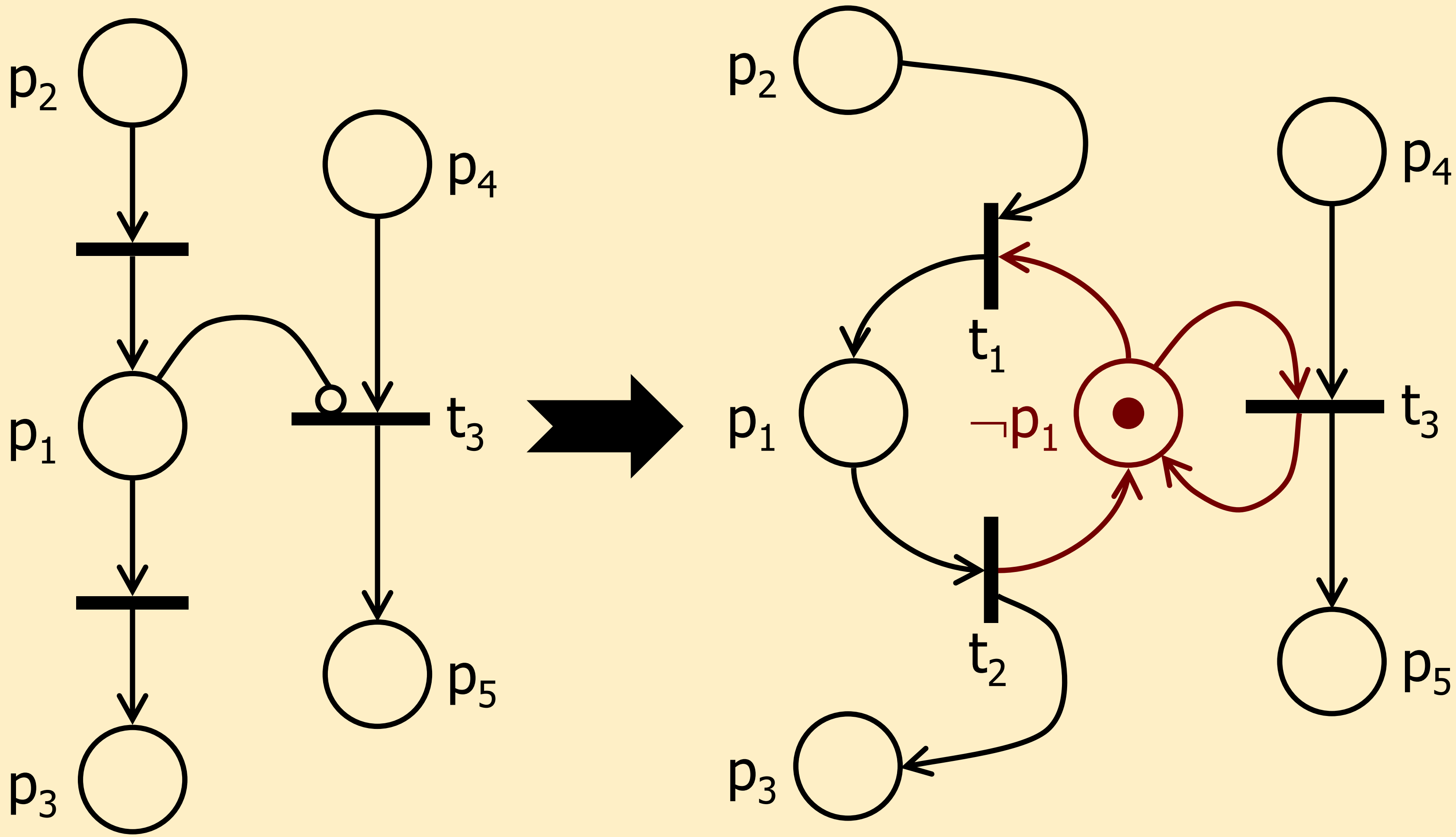
Example: Mutual exclusion with inhibitor arcs, improved



Example: Mutual exclusion without inhibitor arcs



Bypassing inhibitor arcs in a simple case (non-general)



Priority

- Multiple enabled transitions: which one to fire?
  - Priority instead of non-determinism
- Extension: priority assigned to transitions
- Modified firing rule:
  - An enabled transition with lower priority cannot fire, until there is a transition enabled AND having higher priority
  - Non-determinism still applies for transitions with the same priority!

Formal definition with priority

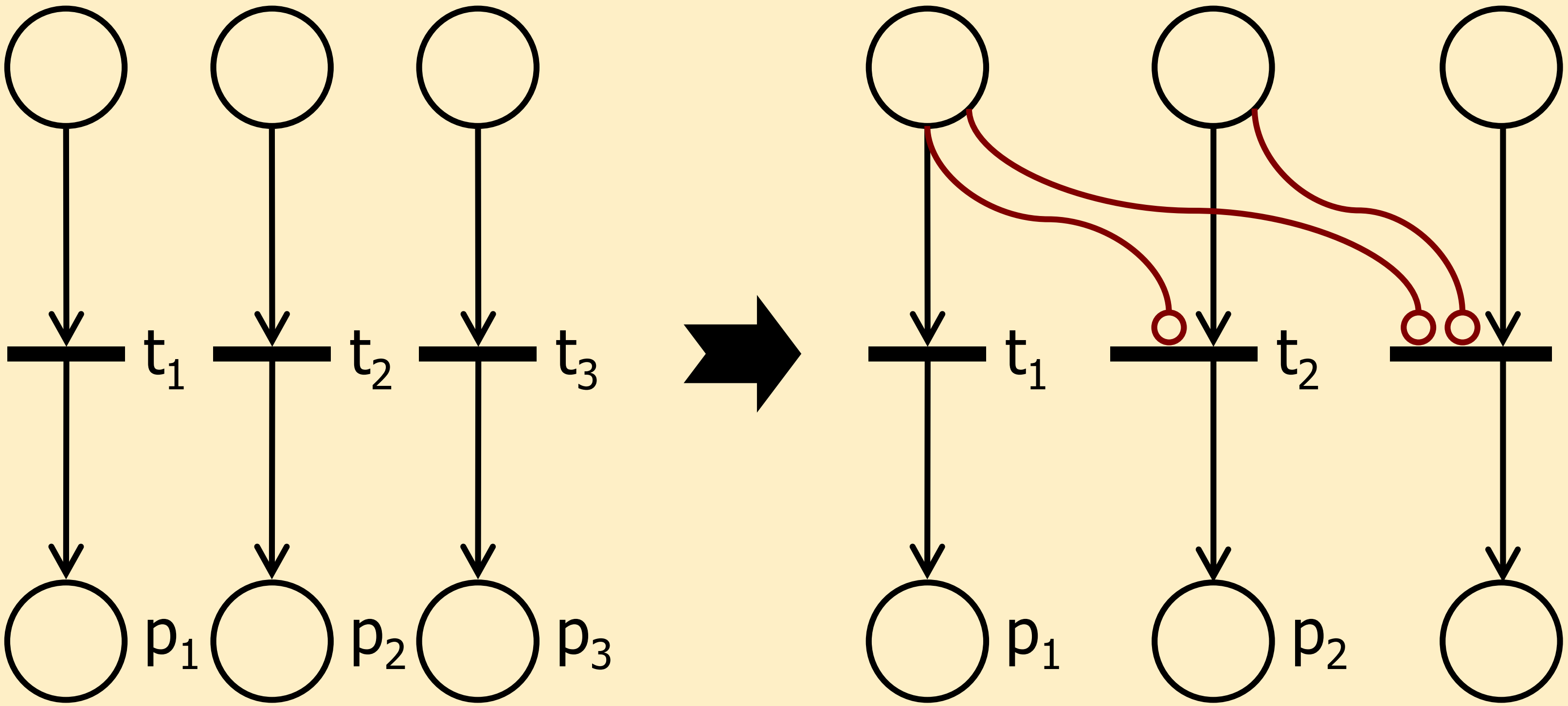
Petri net (PN): $PN = \langle P, T, \Pi, E, W, M_0 \rangle$

- Places $P = \{p_1, p_2, \ldots, p_\pi\}$
- Transitions (firings) $T = \{t_1, t_2, \ldots, t_\tau\}$

  $P \cap T = \varnothing$

- Priority $\Pi : T \rightarrow \mathbf{N}$
- Arcs $E \subseteq (P \times T) \cup (T \times P)$
- Weight function $\mathrm{W}: E \rightarrow \mathbf{N}^+$
- Initial marking $M_0 : P \rightarrow \mathbf{N}$

Inhibitor arcs instead of priority?

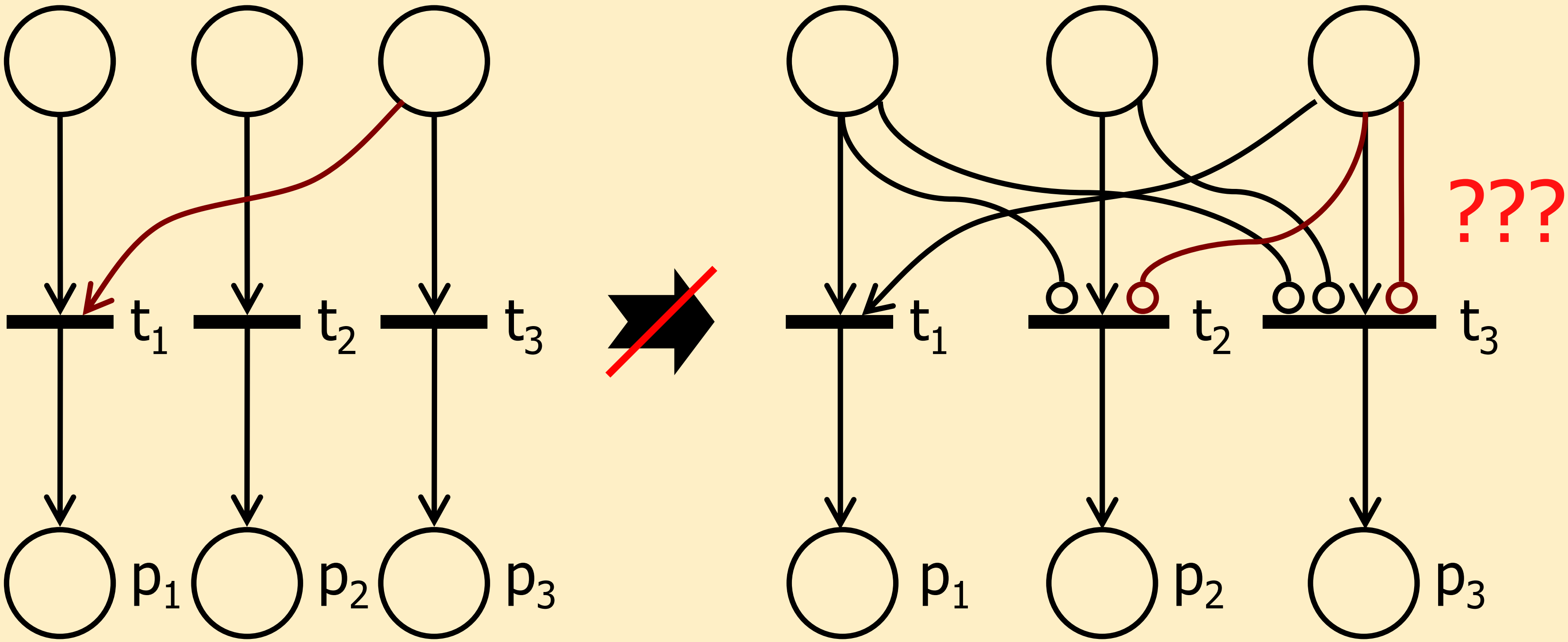


$\pi_1 > \pi_2 > \pi_3$

Idea: "Draw inhibitor arcs from input places of transition with higher priority to transitions with lower priority."

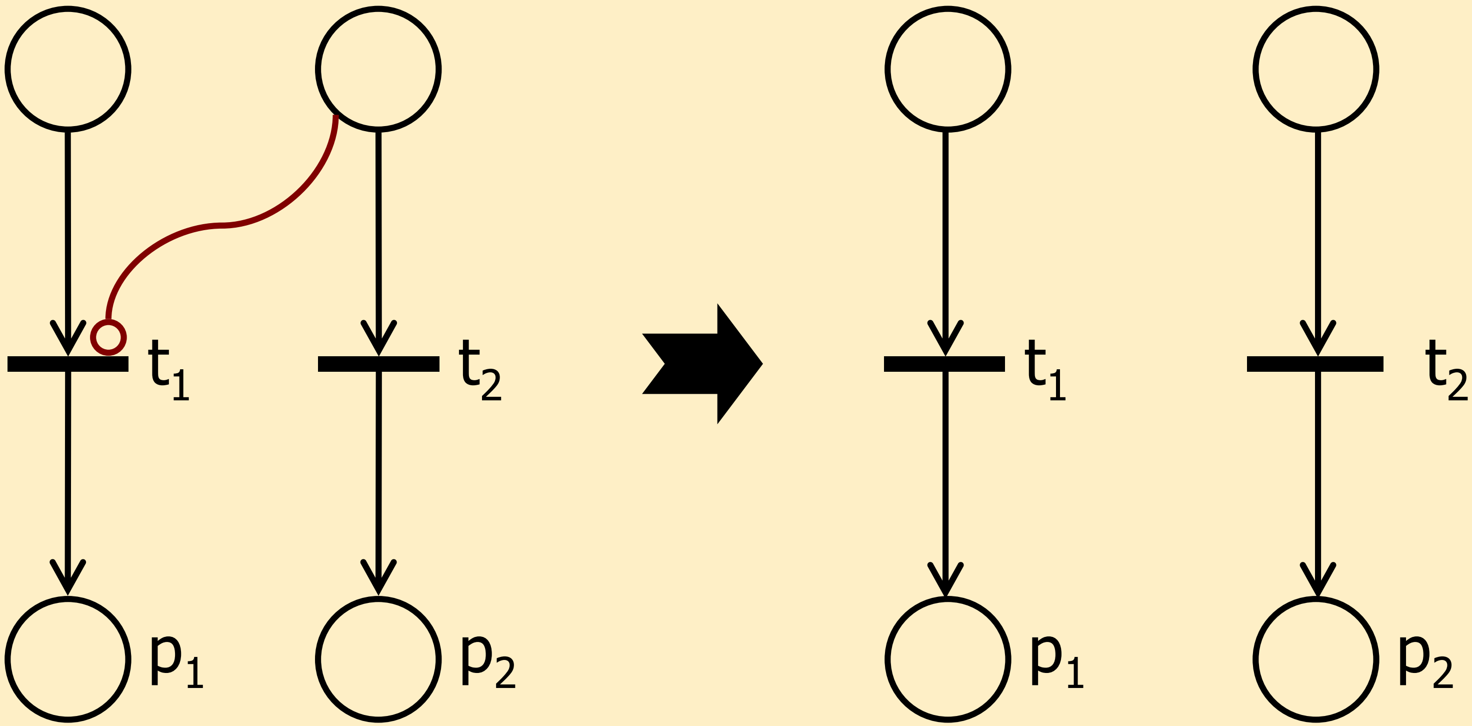
Can this idea be generalized?

Previous idea cannot be generalized



$\pi_1 > \pi_2 > \pi_3$

Priority instead of inhibitor arcs?

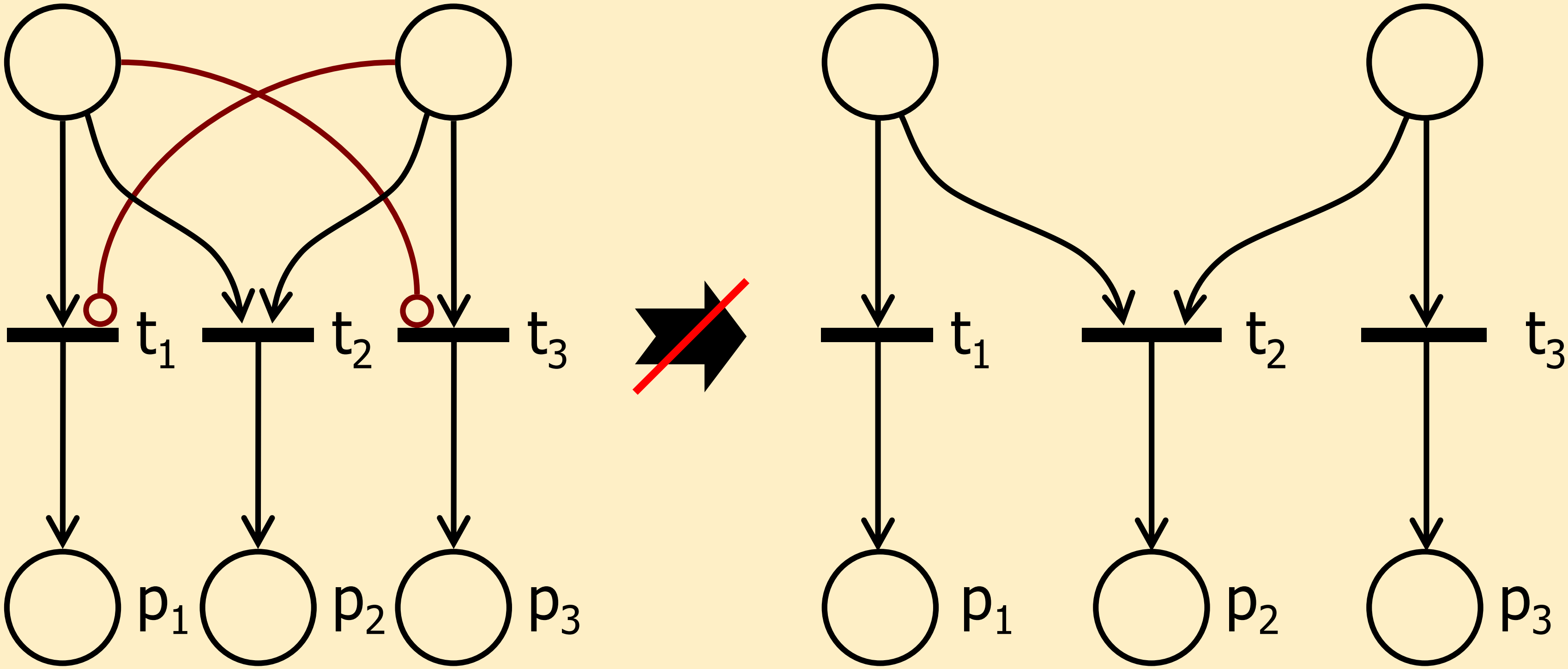


$\pi_1 < \pi_2$

Idea: A transition disabled by an inhibitor arc gets lower priority.

Can this idea be generalized?

Previous idea cannot be generalized



$\pi_1 < \pi_3$

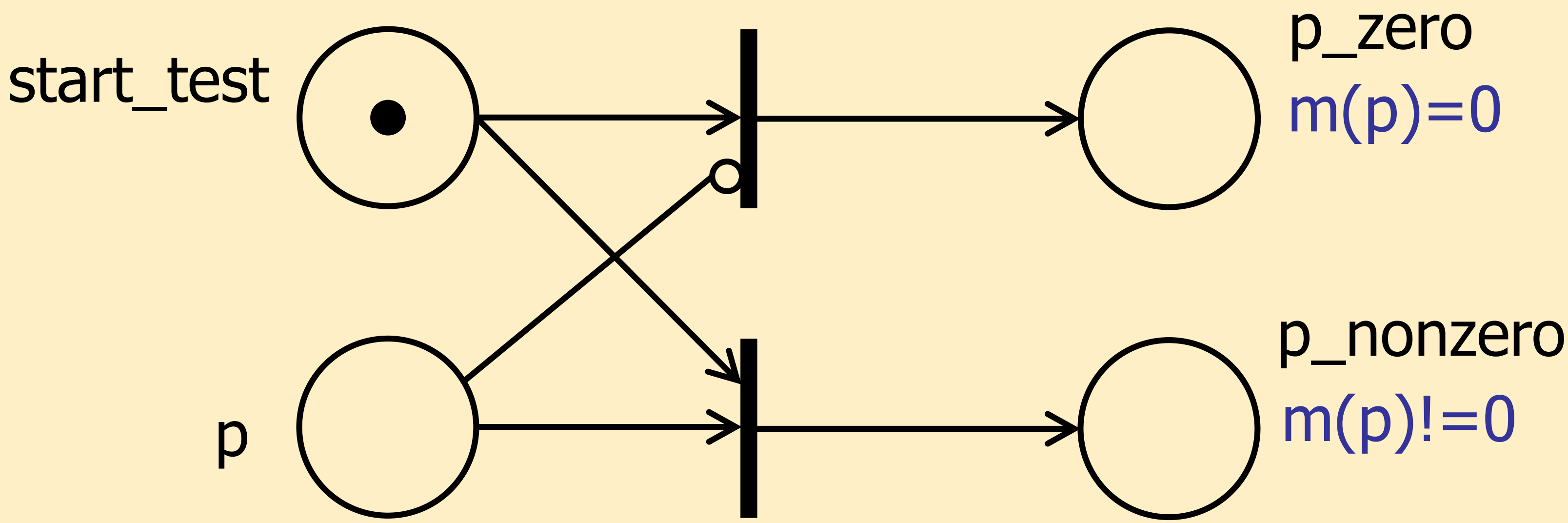
$\pi_3 < \pi_1$

???

Expressive power of inhibitor arcs

- Inhibitor arcs can be used for "zero testing"

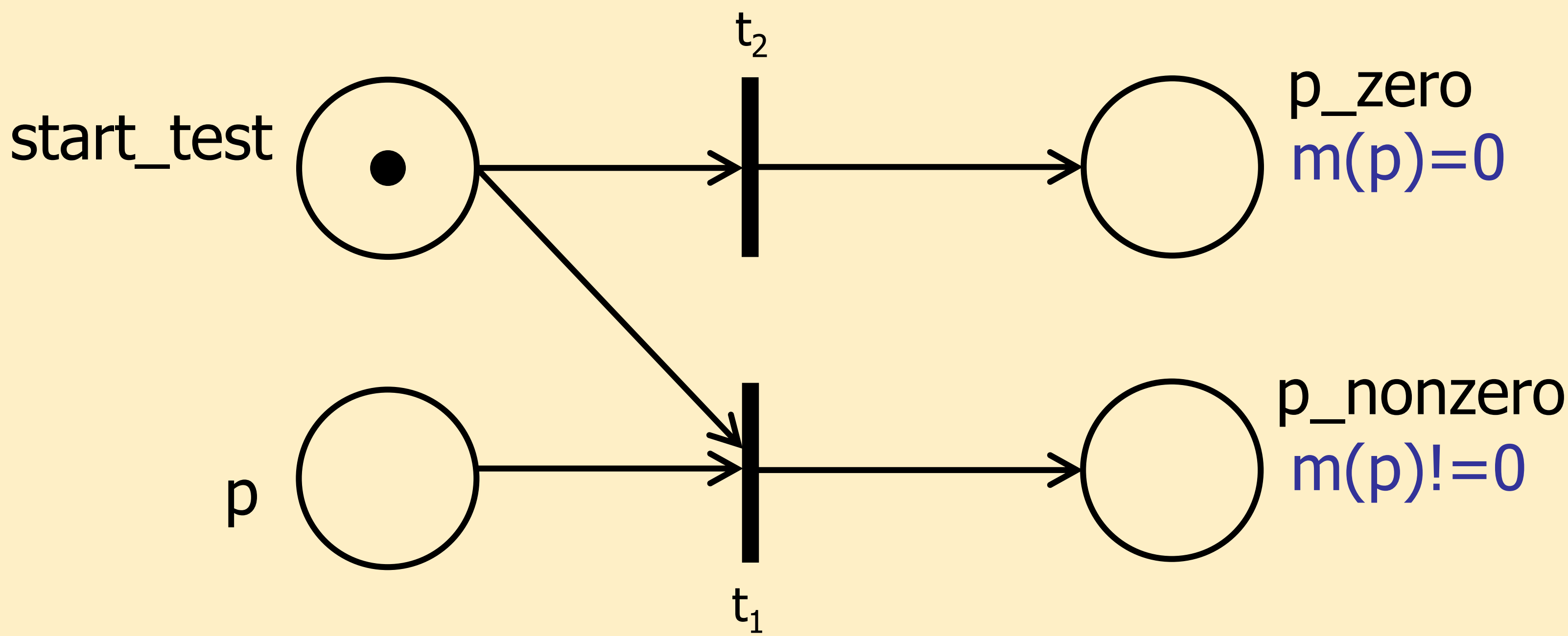
  p=0? (Marking places with tokens if m(p)=0 or m(p)!=0 holds for place p.)



Expressive power with priority

- Priority can be used for "zero testing"

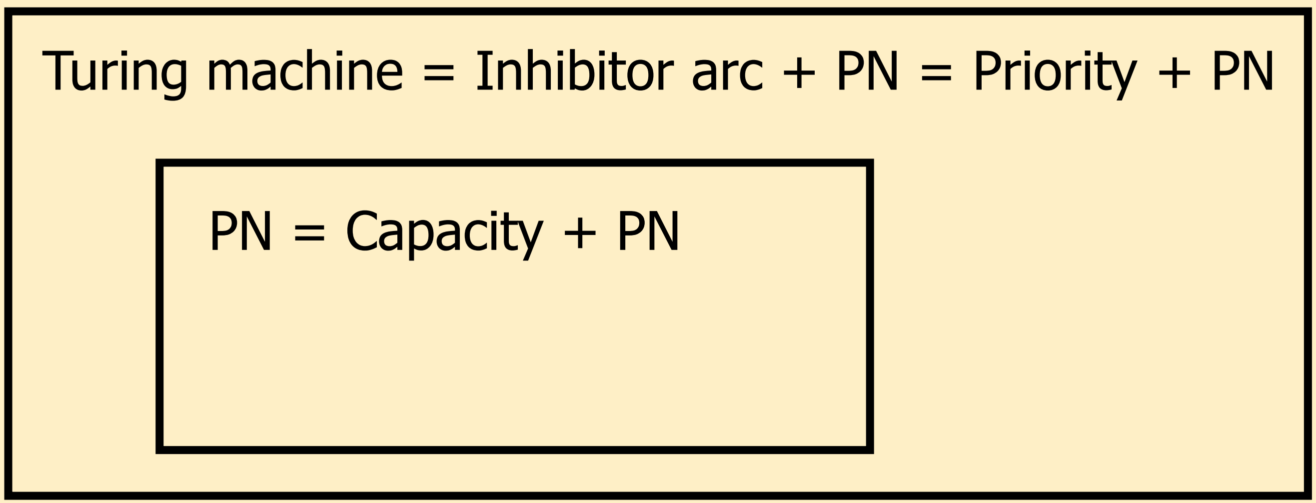
  p=0? (Marking places with tokens if m(p)=0 or m(p)!=0 holds for place p.)



$$\boxed{\pi_2 < \pi_1}$$

Summary of expressive power[P81]

- "Zero testing" enables Petri nets to simulate every Turing machine
  - Consequence: undecidable problems…
- Finite capacity is just a syntactical construct



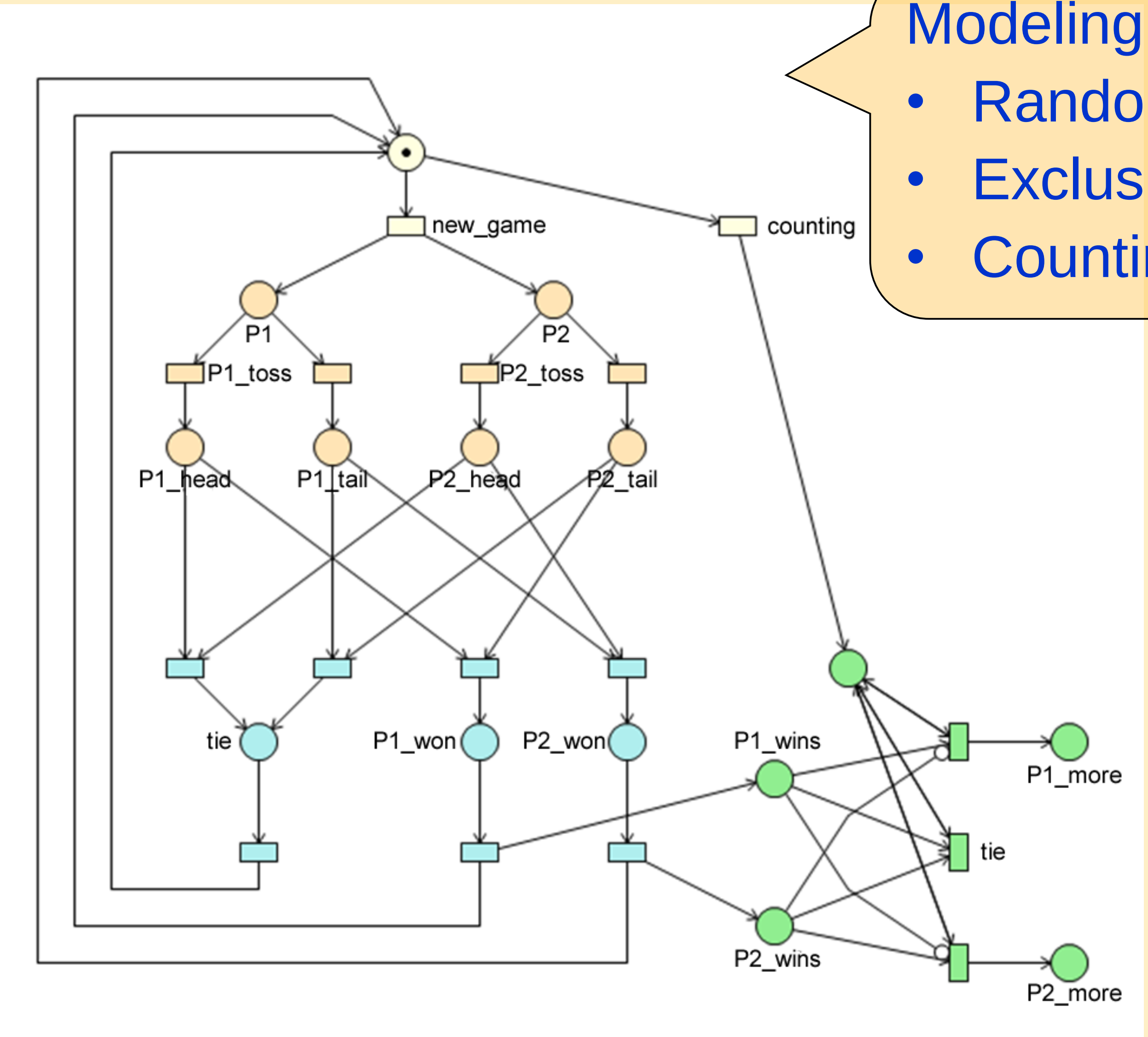
J.L. Peterson, *Petri Net Theory and the Modeling of Systems*, Prentice-Hall, 1981.

Expressive power of PNs without extensions

- Are there systems that cannot be modeled with Petri nets, if none of the extensions is used?
  - YES!
- The key for "non-modelability":
  - It cannot be checked if an infinite capacity place p is marked with k number of tokens or not
  - As a special case k=0, which is known as the "zero testing" problem
    - It can be shown that a solution for the "zero testing" problem yields a solution for the general case with an arbitrary k

Simple examples for building Petri nets

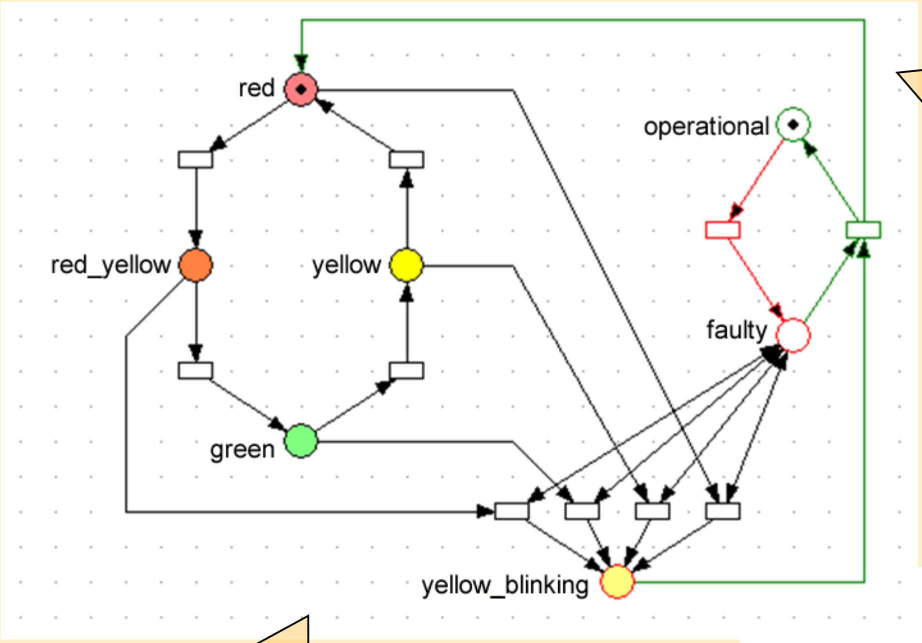
Simple example: Tossing a coin



Modeling constructions:

- Random choice
- Exclusions (alternatives)
- Counting (for the decision)

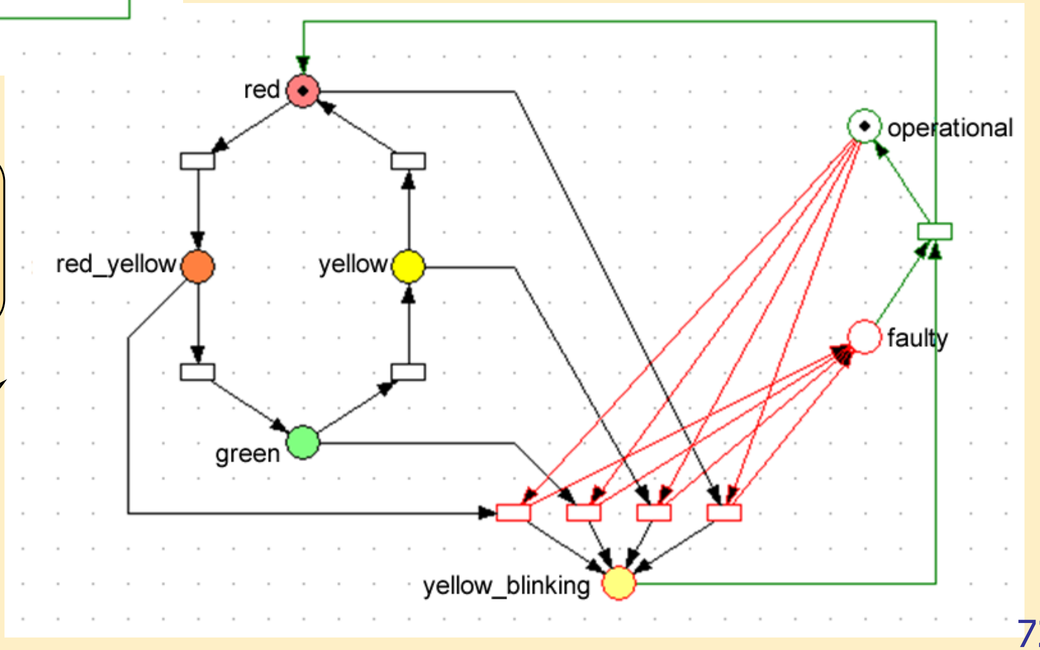
Simple example: Traffic light with failures



Modeling constructs:

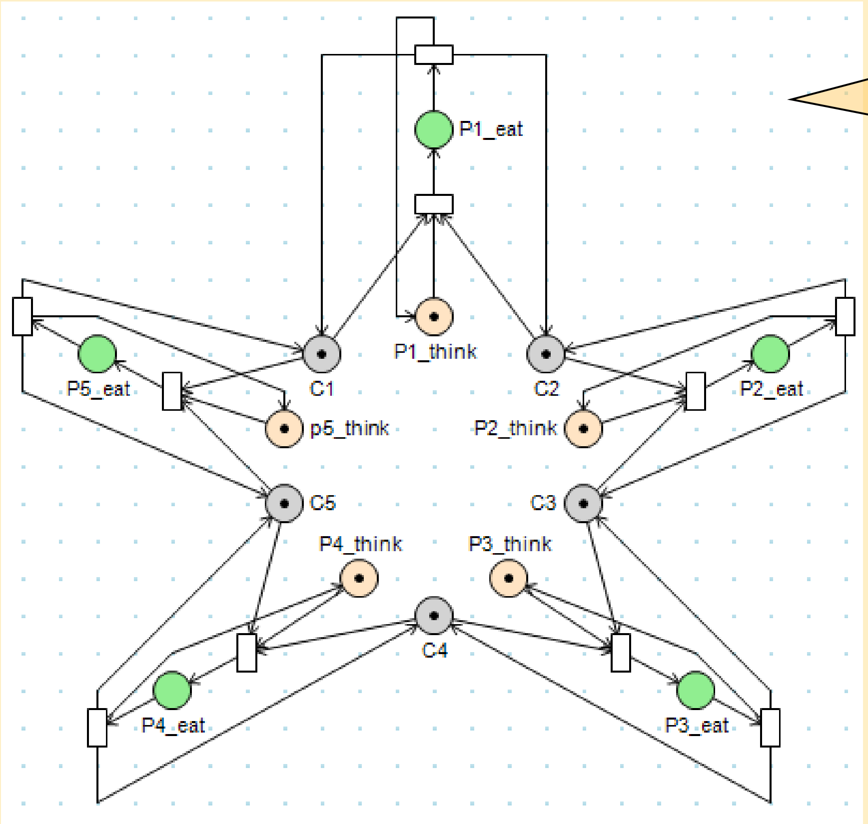
- Random event
- Synchronization
- State variable

Incorrect model: failure is only an alternative

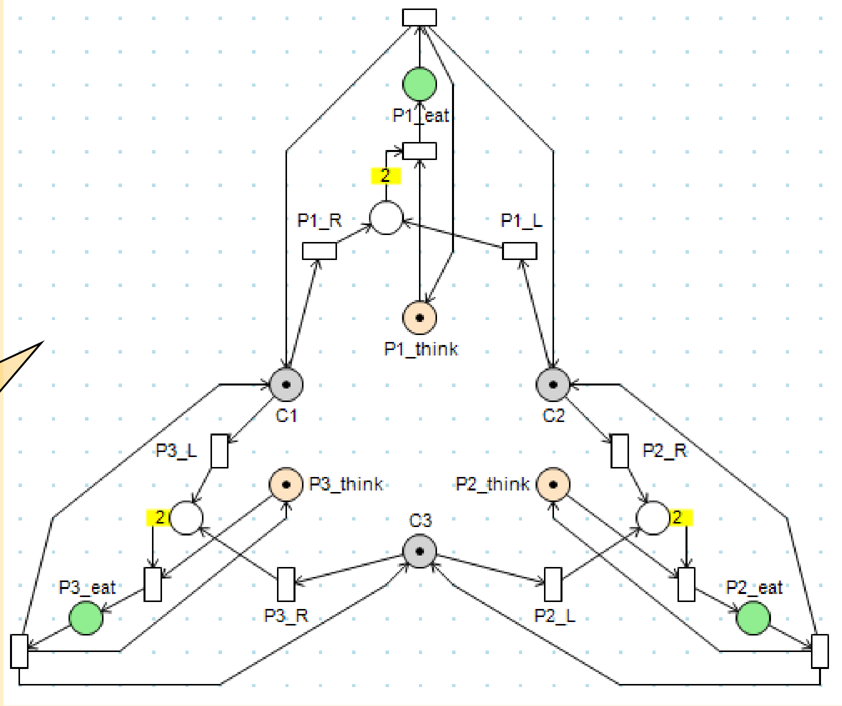


Correction: Event-driven approach

Simple example: Dining philosophers

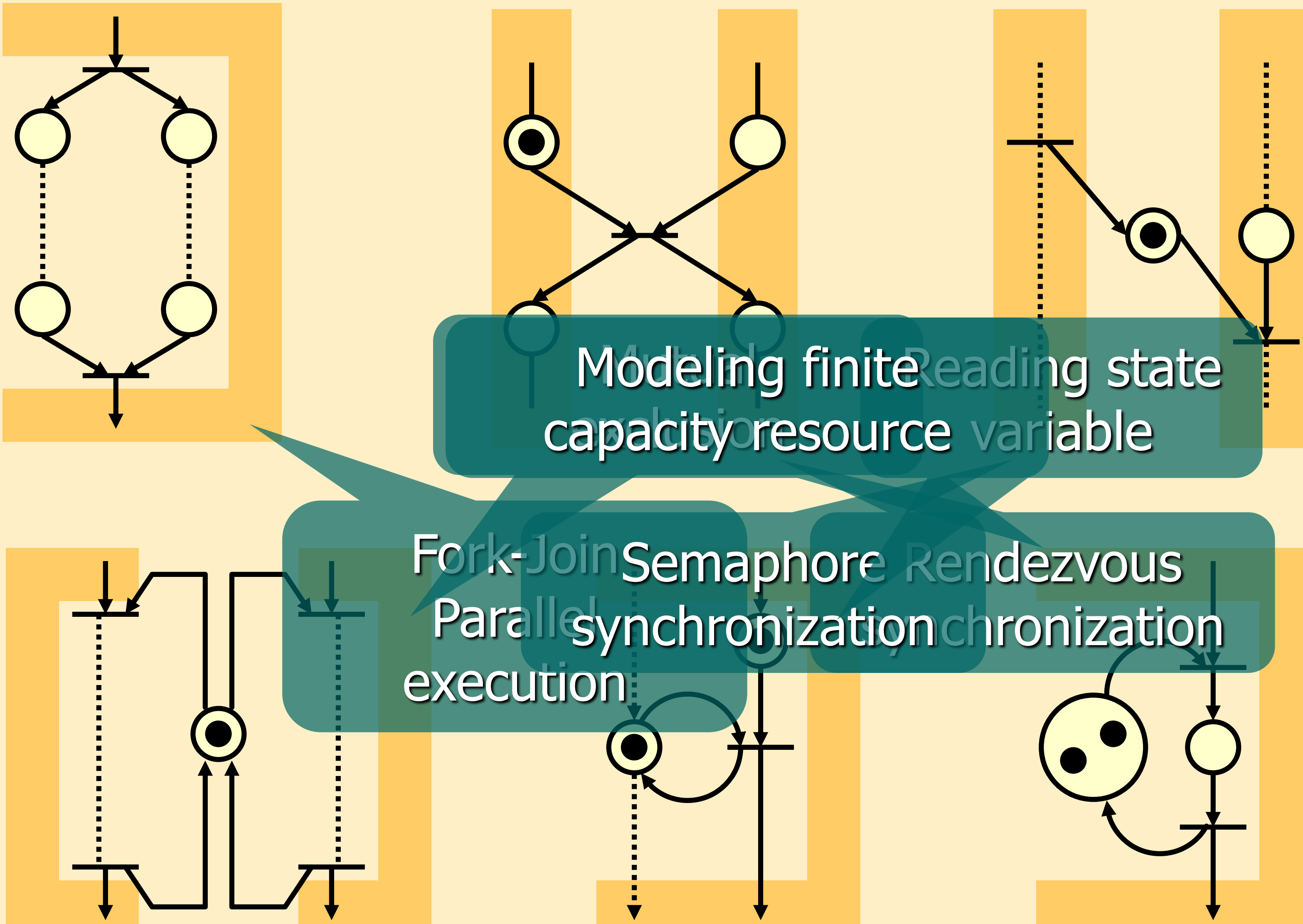


Modeling constructs:
- Atomic event: taking two forks

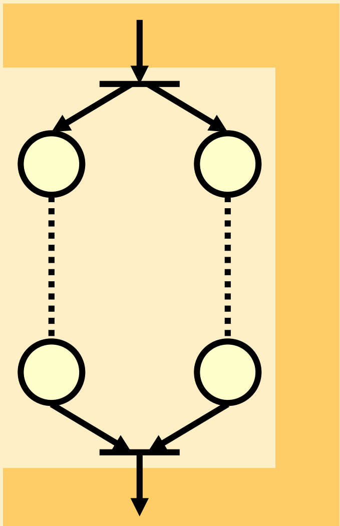


Modeling constructs:
- Atomic event: taking a single fork
- Possible deadlock

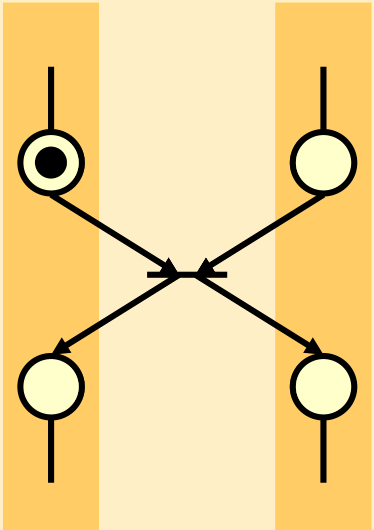
Typical modeling constructs



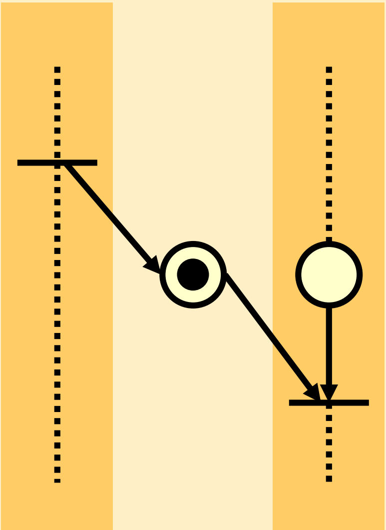
Typical modeling constructs



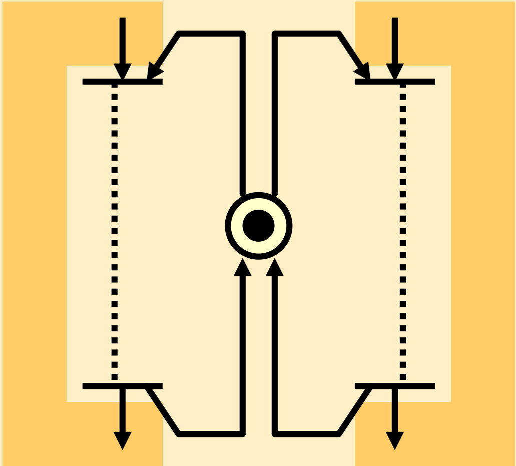
Fork-Join



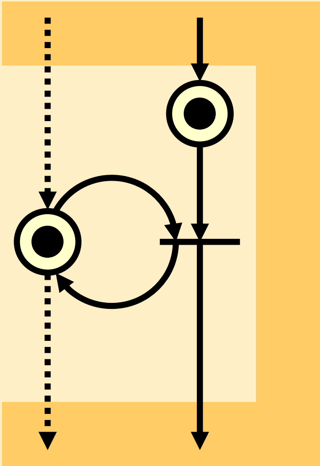
Rendezvous sync.



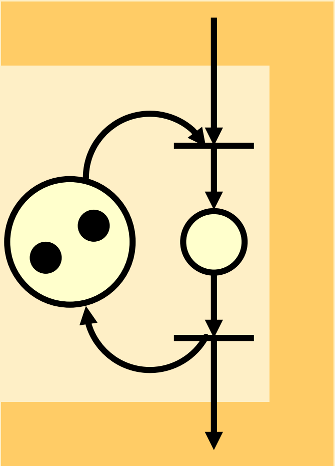
Semaphore sync.



Mutual exclusion



Reading state



Finite capacity