Computational Models for Embedded Systems

Vescan Andreea, PHD, Assoc. Prof.

Faculty of Mathematics and Computer Science
Babeș-Bolyai University
Cluj-Napoca

2025-2026

Lecture 10a: Petri Nets (1)

Software Systems Verification and Validation

"Tell me and I forget, teach me and I may remember, involve me and I learn."

(Benjamin Franklin)

Outline

- PetriNets
- Why Petri nets?
- Petri nets – definition
- Transitions
- Reachability set
- Firing Vector, Incidence Matrix
- I/O Petri nets
- Compositions of Petri Nets
- Modeling Templates

- Next lecture: Petri Nets (cont.)
- Questions

PetriNets

- Introduced in 1962 by Carl Adam Petri in his PhD thesis: **“Communication with Automata”**

- Different “Types” of Petri nets known
    - Condition/event nets
    - Place/transition nets
    - Predicate/transition nets
    - Hierarchical Petri nets,
    - …

Why
Petri Nets?

- Models based on finite state machines are inherently focused on the state of a system and the **observable input-output behavior**.

- They are **not well suited to studying the interaction of concurrently active parts** of a system and the combined behavior of distributed parallel systems.

Why Petri Nets?

- To address issues of concurrency, synchronization and communication, we need to generalize state machines into a model of communicating, concurrent state machines.

- A first necessary step into that direction is to study the phenomenon of concurrency isolated from the internal complexities of the subparts of a system.

- Why? Mainly because Concurrency itself is an important source of complexity that can be rapidly overwhelming.

  → Petri Nets.

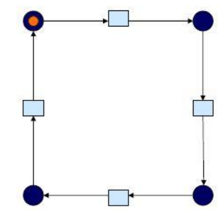
What is Petri Nets?

- 2 principal abstractions that render them so suitable for the study of concurrency:
  - They are concentrating of the act of communication, and are not concerned with the data being communicated. The mean of communication is a token which does not contain any data.
  - All details of behavior are omitted if they do not contribute to the consumption and emission of tokens.

PetriNets

- **Used for Modelling, Analysis, Verification of Distributed Systems**

Example 1: The four seasons



- (other) application areas:
  - automation engineering
  - business processes
- Focus on modeling causal dependencies;
- no global synchronization assumed (message passing only).
- **PNs describe explicitly and graphically:**
  - **sequencing/causality**
    - **conflict/non-deterministic choice**
    - **concurrency**
- **Basic PN model**
  - **Asynchronous model (partial ordering)**
  - **Main drawback: no hierarchy**

Petri net Definition

- A Petri net is a six-tuple N=(P,T,A, w, $\vec{x}_0$), where
  - P is a finite set of places
  - T is finite set of transitions
  - A is a set of arcs, $A \subseteq (P \times T) \cup (T \times P)$
  - W is a weight function, $w : A \to N$
  - $\vec{x}_0$ is an initial marking vector, $\vec{x}_0 \in N^{|P|}$
- The set $I(t) = \{p \in P \mid (p,t) \in A\}$ is the set of input places of transition t.
- The set $O(t) = \{p \in P \mid (t,p) \in A\}$ is the set of output places of transition t.
- A transition t is enabled in state $\vec{x}$ if

$$x(p) \geq w(p,t), \forall p \in I(t)$$

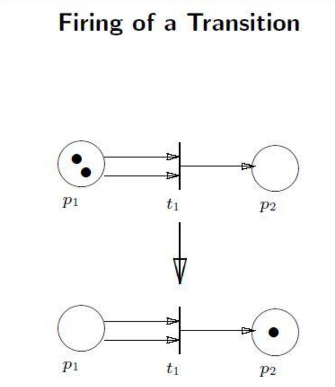
Petri net transition Definition

- Let N=(P,T,A, w, $\vec{x}_0$), be a petri net with P = {p_0, ..., p_{n-1}} and x = [x(p_0), ..., x(p_{n-1})] be a marking for the n places. The transition function

is defined as follows: $G:(N^n \times T) \to N^n$

$$G(\vec{x}, t) = \begin{cases} \vec{x}' & \text{if } x(p) \geq w(p,t) \forall p \in I(t) \\ \vec{x} & \text{otherwise} \end{cases}$$

$$\text{with} \quad \begin{array}{l} \vec{x}' = [x'(p_0), \ldots, x'(p_{n-1})] \\ x'(p_i) = x(p_i) - w(p_i, t) + w(t, p_i) \text{ for } 0 \leq i < n \end{array}$$

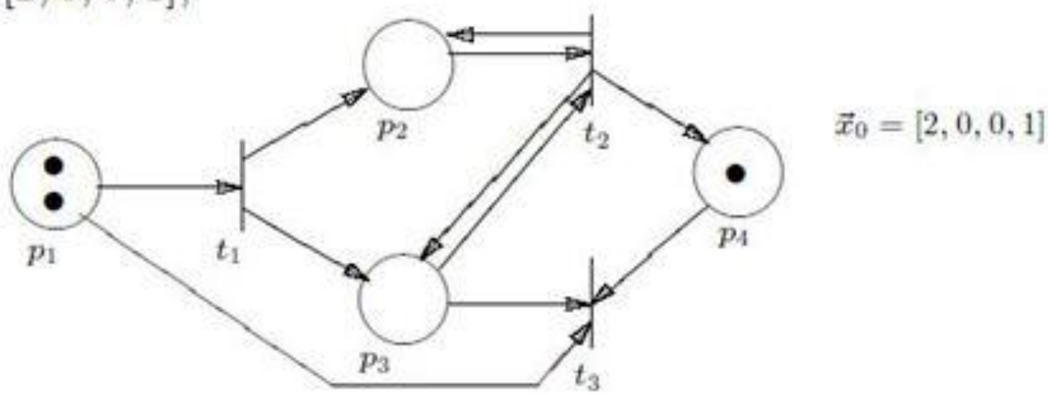


- **if the number of token in p is greater or equal to the sum of the transitions exiting the place, then take the transition**
- **otherwise nothing changes !**

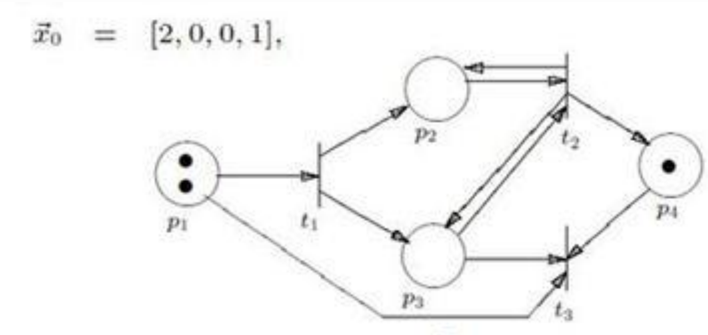
Petri net Dynamics (1)

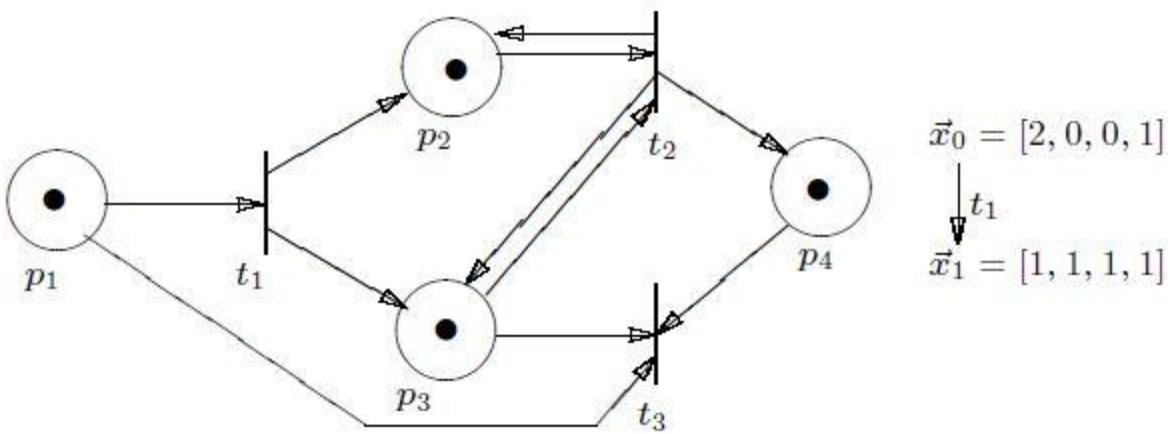
Petri net $N = (P, T, A, w, \vec{x}_0)$ with

$$\begin{aligned}
P &= \{p_1, p_2, p_3, p_4\} \\
T &= \{t_1, t_2, t_3\} \\
A &= \{(p_1, t_1), (p_1, t_3), (p_2, t_2), (p_3, t_2), (p_3, t_3), (p_4, t_3), \\
&\quad (t_1, p_2), (t_1, p_3), (t_2, p_2), (t_2, p_3), (t_2, p_4)\} \\
w(a) &= 1 \ \forall a \in A \\
\vec{x}_0 &= [2, 0, 0, 1],
\end{aligned}$$

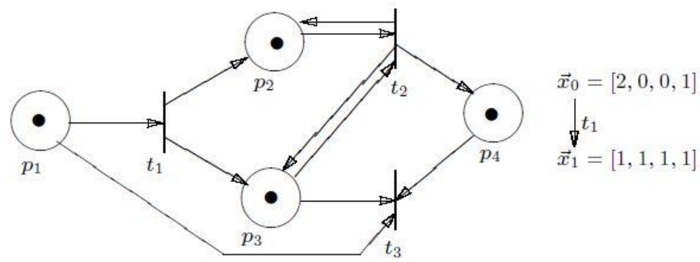


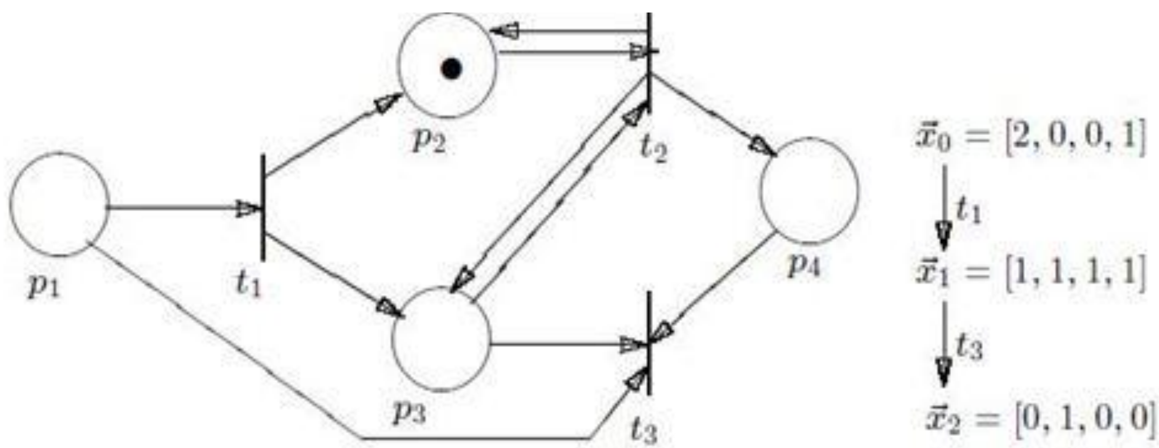
Petri net Dynamics (2)





Petri net Dynamics (3)





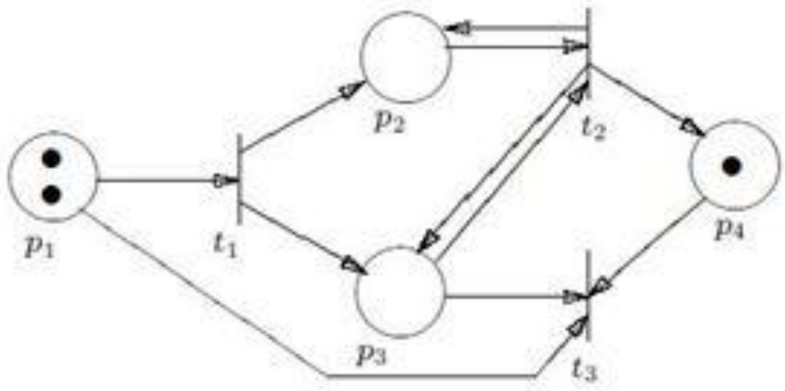
The Reachability Set

- For a Petri net N=(P,T,A, w, $\vec{x}_0$) and a given state $\vec{x}$ a state $\vec{y}$ is immediately reachable from $\vec{x}$ if there exists a transition $t \in T$ such that $G(\vec{x},t) = \vec{y}$
- The reachability set R($\vec{x}$) is the smallest set of states defined by:

1. $\vec{x} \in R(\vec{x})$
2. If $\vec{y} \in R(\vec{x})$ and $z = G(\vec{y}, t)$ for some $t \in T$, then $\vec{z} \in R(x)$.

- More intuitively, the reachability set of a state includes all the states that can eventually be reached by repeatedly firing transitions.

The Reachability Set



$$
\begin{aligned}
R(\vec{x}_0) &= R_1 \cup R_2 \cup R_3 \cup R_4 \\
R_1 &= \{\vec{x}_0\} \\
R_2 &= \{\vec{y} \mid \vec{y} = [1, 1, 1, n], n \geq 1\} \\
R_3 &= \{\vec{y} \mid \vec{y} = [0, 2, 2, n], n \geq 1\} \\
R_4 &= \{\vec{y} \mid \vec{y} = [0, 1, 0, n], n \geq 0\}
\end{aligned}
$$

Firing Vector, Incidence Matrix

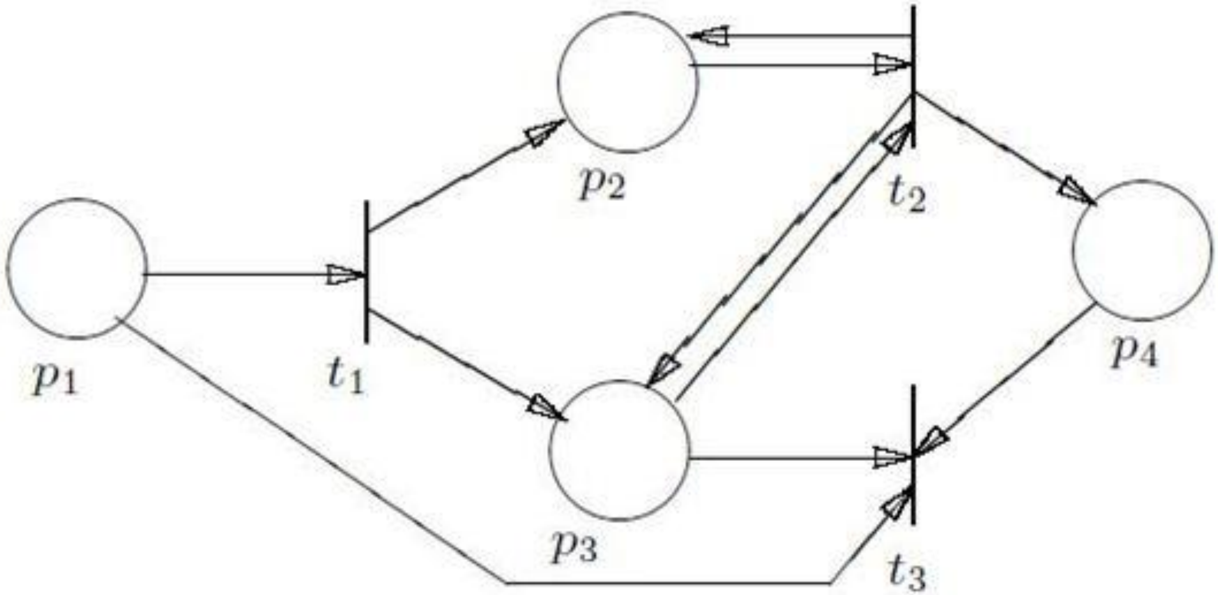
Let $N = (P, T, A, w, \vec{x_0})$ be a petri net with $P = p_1, ..., p_n$ and $T = t_1, ..., t_m$.
A firing vector $\vec{u} = [0, ..., 0, 1, 0, ..., 0]$ is a vector of length m where entry $j, 1 \geq j \geq m$ corresponds to transition t_j . All entries of the vector are 0 but one, where it has a value of 1. If entry j is 1, transition t_j fires.
The incidence matrix $\mathcal{A}$ is an $m \times n$ matrix whose (j, i) entry is:

$$a_{j,i} = w(t_j, p_i) - w(p_i, t_j)$$

- a $_{j,i}$ contains the information about the net effect of firing transition t_j from place p_i .
- A state equation can be written as:

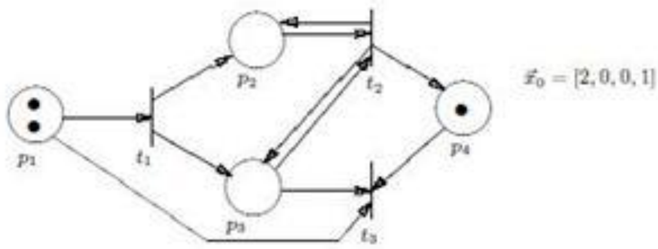
  x' = x + uA

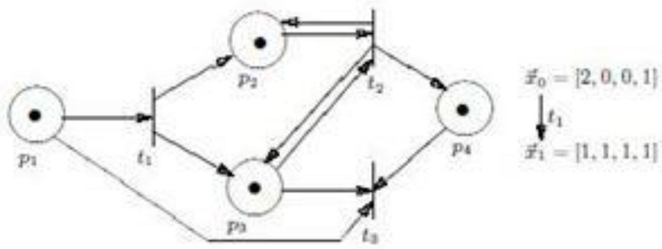
Building the incidence matrix



$$\mathcal{A} = \begin{bmatrix} -1 & 1 & 1 & 0 \\ 0 & 0 & 0 & 1 \\ -1 & 0 & -1 & -1 \end{bmatrix},$$

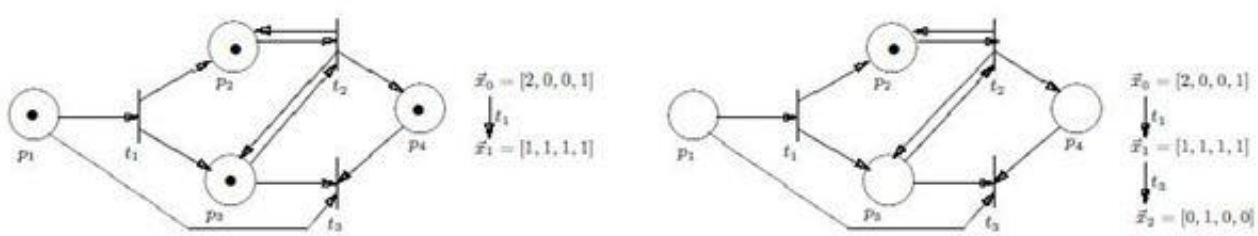
Petri nets - evaluation





$$
\begin{aligned}
\vec{x}_1 &= \vec{x}_0 + \vec{u}_1 \mathcal{A} \\
&= [2,0,0,1] + [1,0,0] \begin{bmatrix} -1 & 1 & 1 & 0 \\ 0 & 0 & 0 & 1 \\ -1 & 0 & -1 & -1 \end{bmatrix} \\
&= [2,0,0,1] + [-1+0+0, 1+0+0, 1+0+0, 0+0+0] \\
&= [2,0,0,1] + [-1,1,1,0] = [1,1,1,1]
\end{aligned}
$$

Petri nets - evaluation



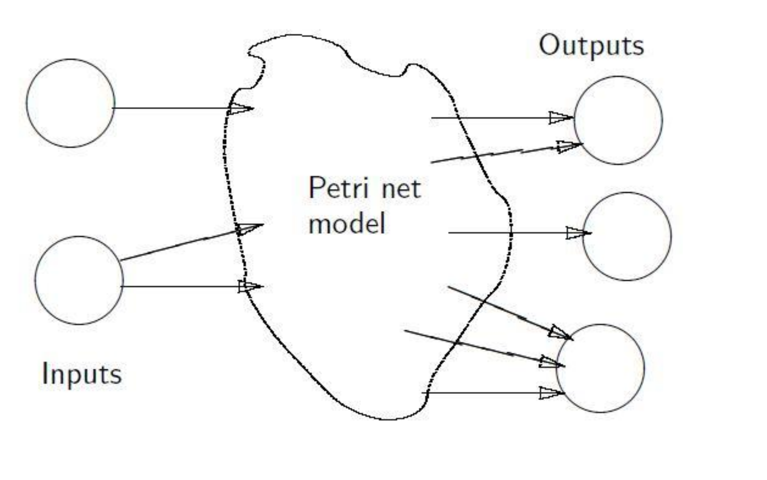
$$
\begin{aligned}
\vec{x}_2 &= \vec{x}_1 + \vec{u}_2 \mathcal{A} \\
&= [1,1,1,1] + [0,0,1] \begin{bmatrix} -1 & 1 & 1 & 0 \\ 0 & 0 & 0 & 1 \\ -1 & 0 & -1 & -1 \end{bmatrix} \\
&= [1,1,1,1] + [-1,0,-1,-1] = [0,1,0,0]
\end{aligned}
$$

I/O in Petri nets

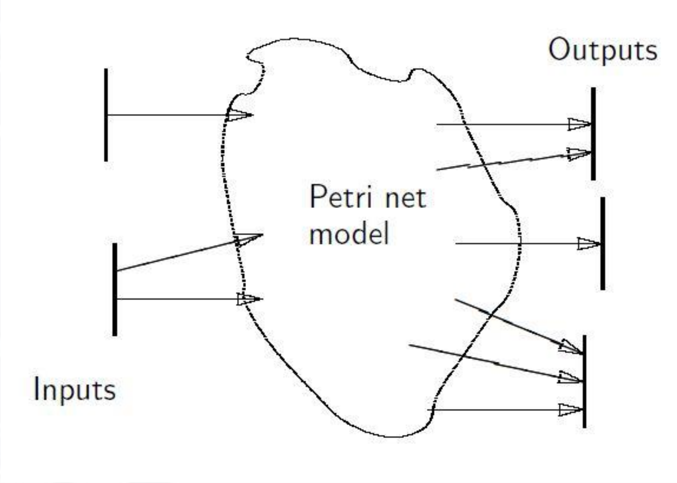
- So far, we have not considered input/outputs of petri nets.
- Many modeling tasks can be achieved with isolated nets without interfaces.
- It is however sometimes very useful to be able to model explicitly inputs and outputs.

I/O in Petri nets

I/O MODELED AS PLACES

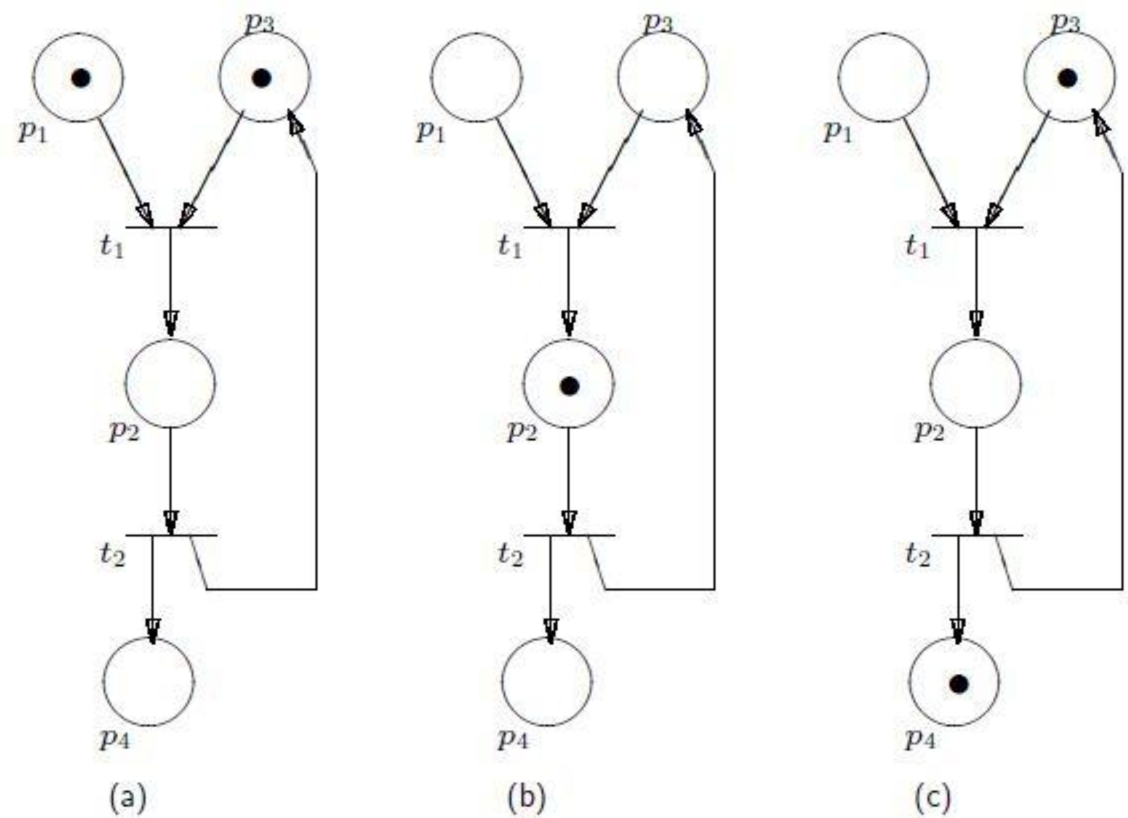


I/O MODELED AS TRANSITIONS



Both methods are equivalent (you can always model one by the other)
We'll use the "places" representation, because it seems slightly more intuitive

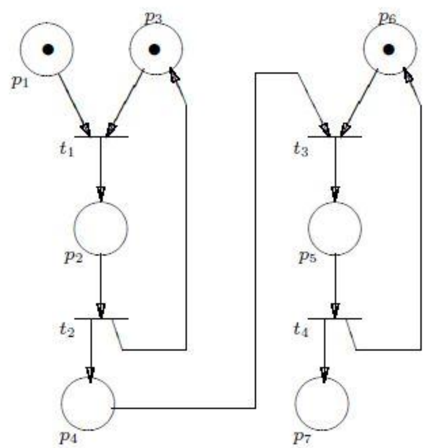
A Server Modeled as Petri Net



Customers arrive at input p_1 and depart at output p_4.

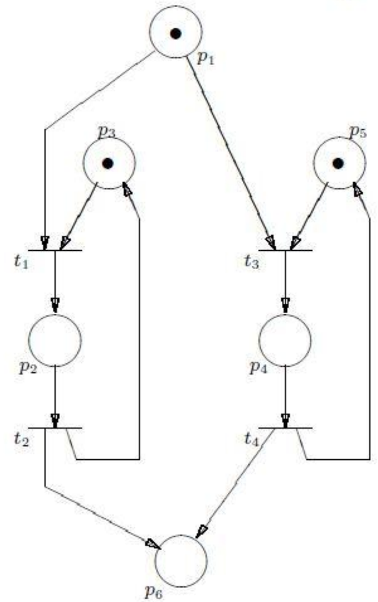
Compositions of Petri Nets

Sequential Composition of two Servers



Customers arrive at input p_1 and depart at output p_7.

Parallel Composition of two Servers



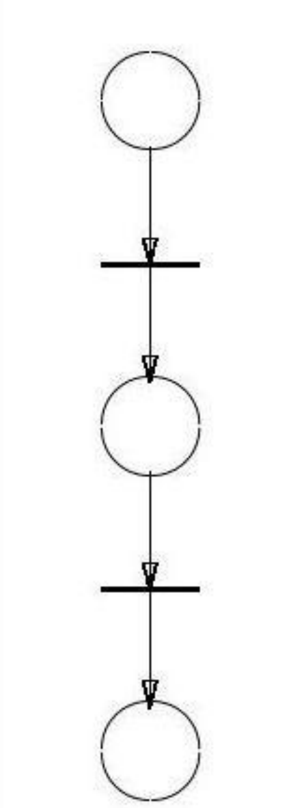
Customers arrive at input p_1 and depart at output p_6.

Modeling Templates

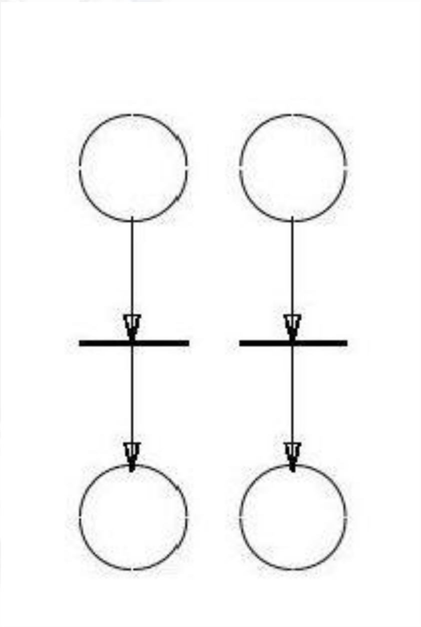
- Goal: illustrate Petri Nets
- We’ll look at several practical templates for standard modeling tasks
  - Sequence and Concurrency
  - Fork and Join
  - Conflict (e.g. for Mutual Exclusion)
  - Producer/Consumer

Sequence and Concurrency

- Sequence



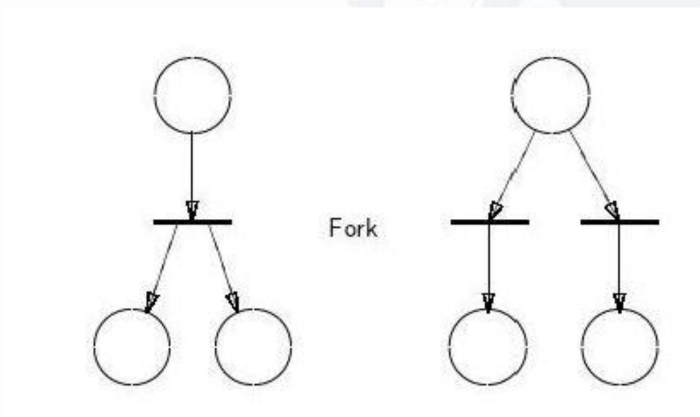
- Concurrency



Fork and Join

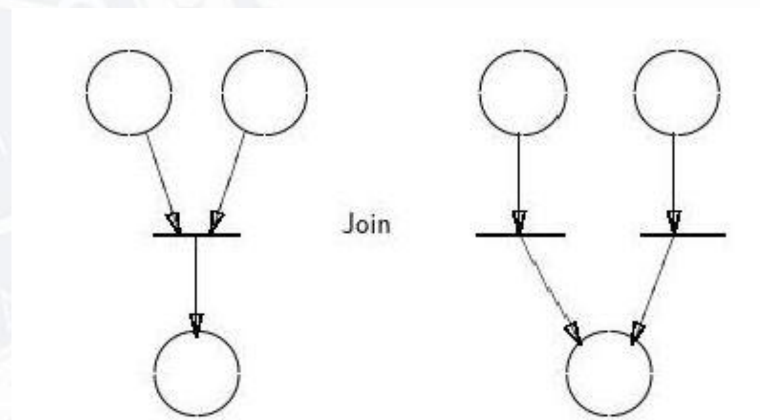
- Fork

doubles the control flow,
resulting in two parallel threads



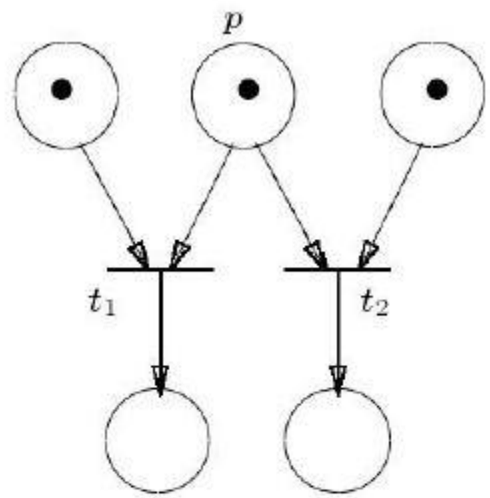
- Join

merges two control flows into one



Conflict

- Used to model situations where two processes compete for data or resources.



Conflict
Mutual Exclusion

```
read(x);
set x <- x + 1;
write(x);
```

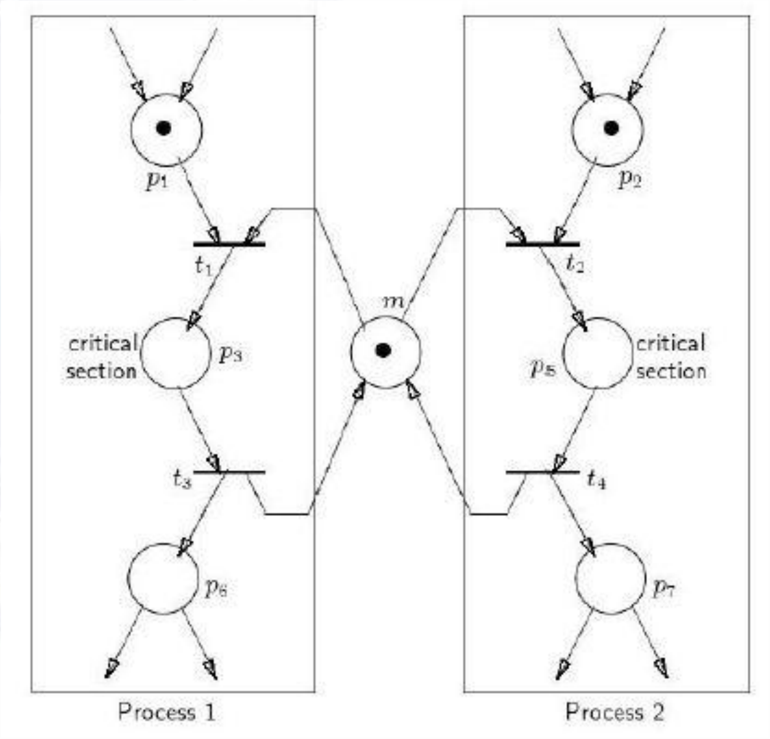
Process A

```
read(x);
set x <- x + 1;
write(x);
```

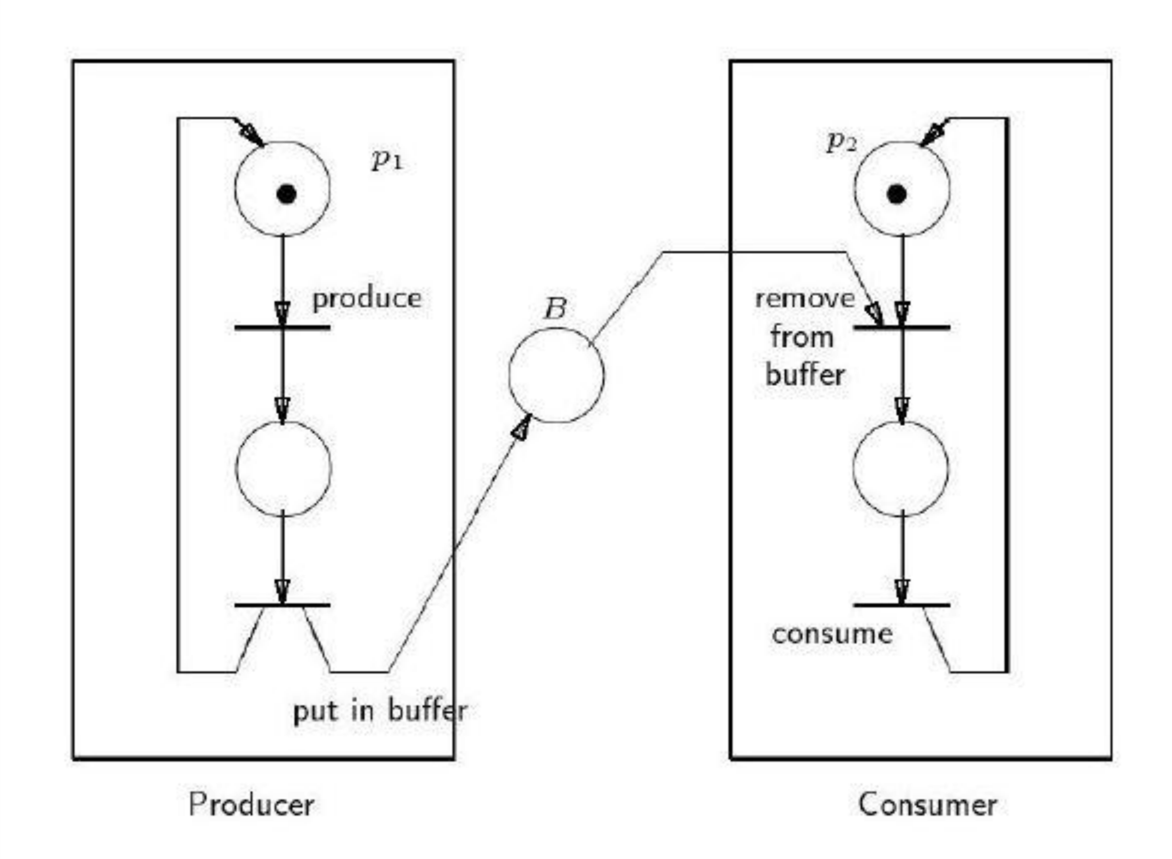
Process B

```
x <- 0;
A.read(x);
A.set x <- x + 1;
A.write(x);
B.read(x);
B.set x <- x + 1;
B.write(x);
x == 2
```

```
x <- 0;
A.read(x);
B.read(x);
A.set x <- x + 1;
A.write(x);
B.set x <- x + 1;
B.write(x);
x == 1
```



Producer/Consumer



Next lecture (today)
PetriNets

- Analysis Methods for Petri Nets
  - Boundedness
  - Conservation
  - Liveness
  - Reachability and Coverability
  - Persistence

  - A technique for addressing these questions on PN:

    The Coverability Tree

References

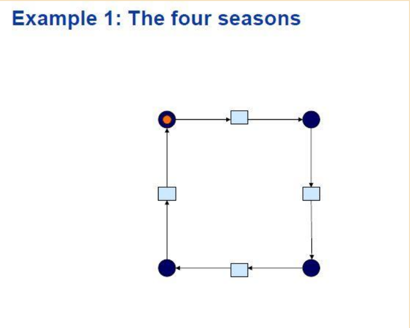
- **Models of Computations and Concurrency** or **Models of computation and their applications to Embedded systems modeling,** Lionel Morel, TUCS,
  http://users.abo.fi/lmorel/MoCs/
  - http://lionel.morel.ouvaton.org/wp/?page_id=13

- **Design of Embedded Systems: Models, Validation and Synthesis,** Alberto Sangiovanni-Vincentelli
  https://inst.eecs.berkeley.edu/~ee249/fa07/

CMES – Today Bring it All Together

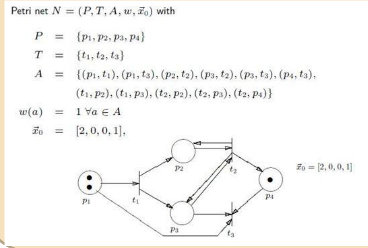
Petri nets

- They are concentrating of the act of communication, and are not concerned with the data being communicated. The mean of communication is a token which does not contain any data.
- All details of behavior are omitted if they do not contribute to the consumption and emission of tokens.

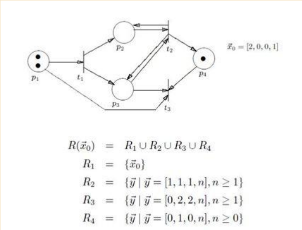


Petri nets

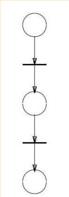
PN dynamics

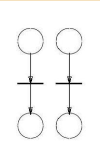


Reachability set

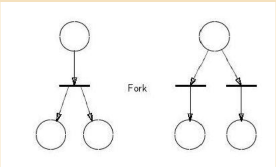


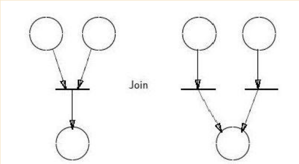
Sequence and Concurrency





Fork and Join





Next Lecture

- Petri nets (2)
- Today

Software Systems Verification and Validation

"Tell me and I forget, teach me and I may remember, involve me and I learn."

(Benjamin Franklin)