System Modeling

Introduction
Rugby Meta-Model
Finite State Machines
Petri Nets
Untimed Model of Computation
Synchronous Model of Computation
Timed Model of Computation
Integration of Computational Models
Tightly Coupled Process Networks
Nondeterminism and Probability
Applications

Petri Nets

Definition: A Petri net is a six-tuple $N = (P, T, A, w, x_0)$, where

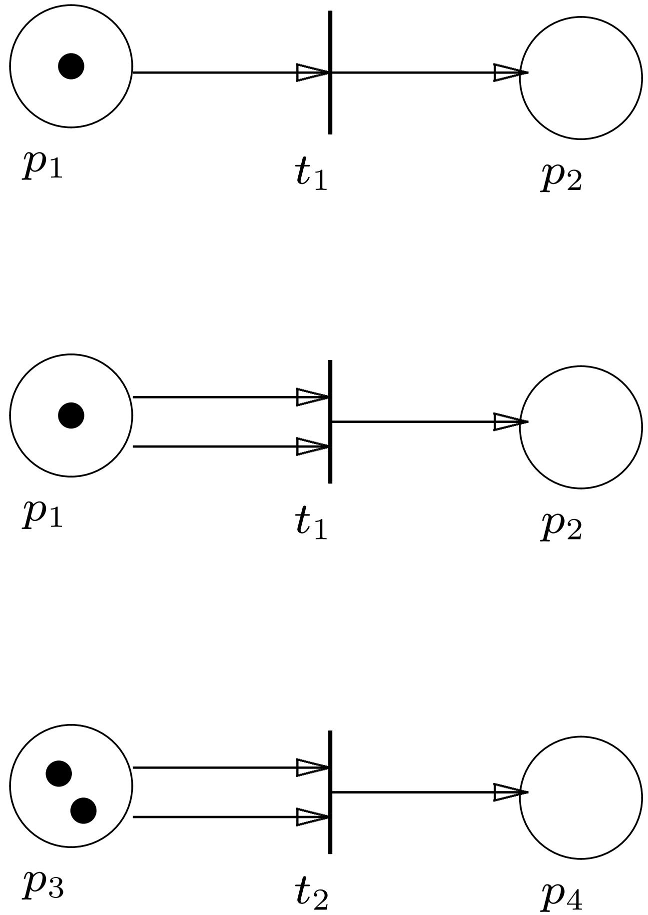
P	is a finite set of places
T	is a finite set of transitions
A	is a set of arcs, $A \subseteq (P \times T) \cup (T \times P)$
w	is a weight function, $w : A \rightarrow \mathbb{N}$
$\vec{x}_0$	is an initial marking vector, $\vec{x}_0 \in \mathbb{N}^{\|P\|}$

Definition: Let $N = (P, T, A, w, \vec{x}_0)$ be a petri net. The set $I(t) = \{p \in P | (p, t) \in A\}$ is the set of input places of transition t. The set $O(t) = \{p \in P | (t, p) \in A\}$ 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 Examples



Petri Net Transition

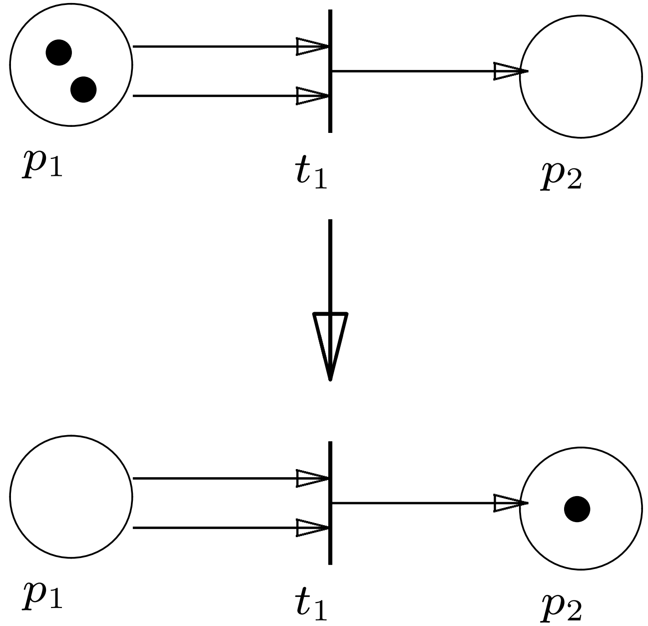
Definition: Let $N = (P, T, A, w, \vec{x}_0)$ be a petri net with $P = \{p_0, \ldots, p_{n-1}\}$ and $\vec{x} = [x(p_0), \ldots, x(p_{n-1})]$ be a marking for the n places. The the transition function $G : (\mathbb{N}^n \times T) \rightarrow \mathbb{N}^n$ is defined as follows

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

$$\text{with} \quad \vec{x}' = [x'(p_0), \ldots, x'(p_{n-1})]$$

$$x'(p_i) = x(p_i) - w(p_i, t) + w(t, p_i) \text{ for } 0 \leq i < n$$

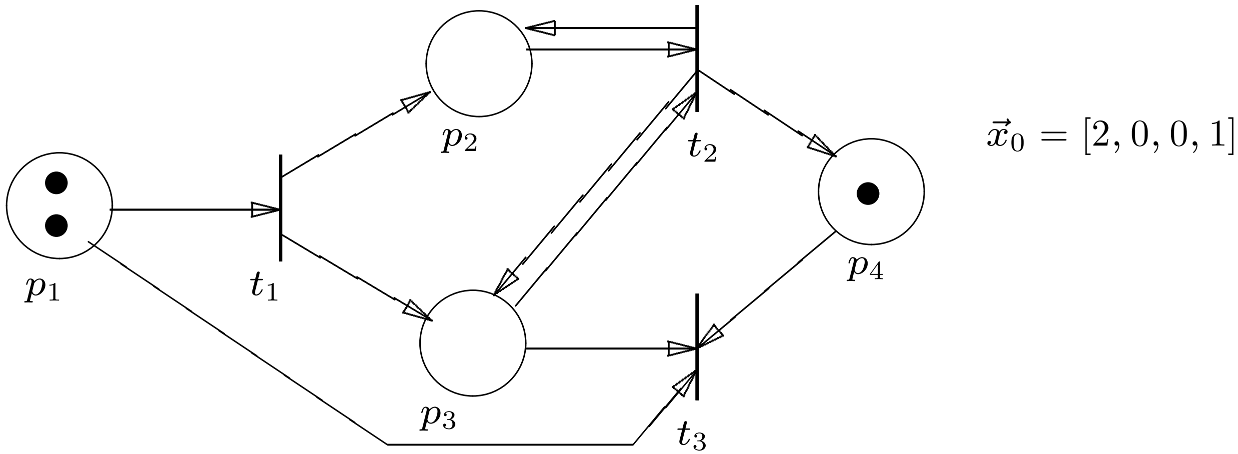
Firing of a Transition



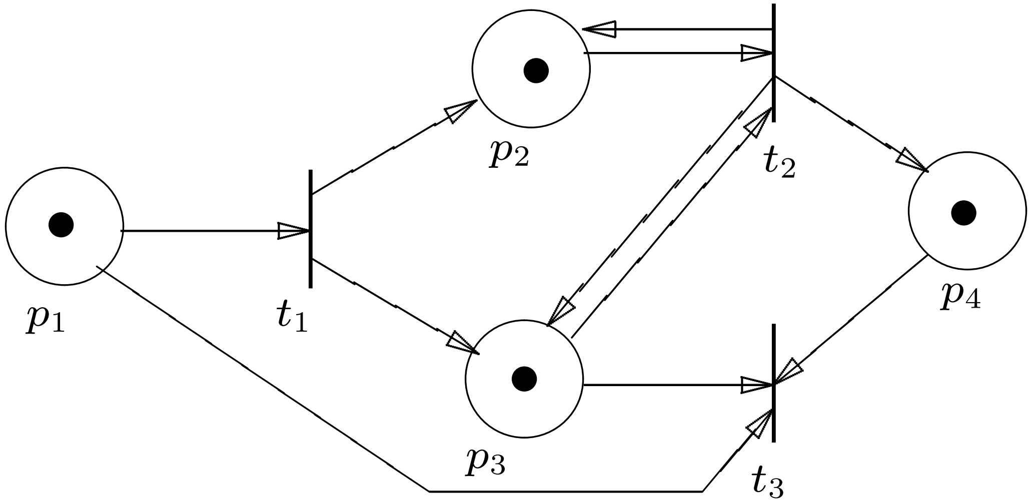
Petri Net Dynamics Example - 1

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

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

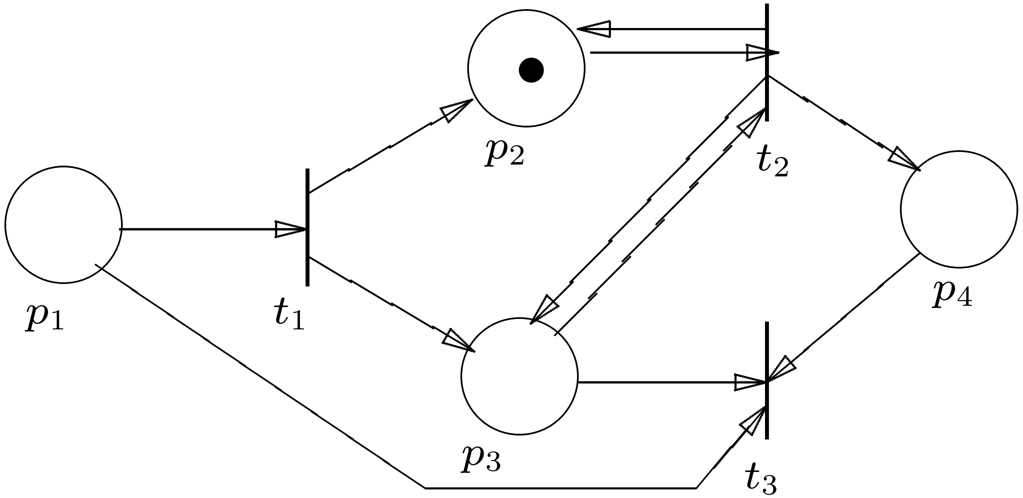


Petri Net Dynamics Example - 2



$$\vec{x}_0 = [2, 0, 0, 1]$$
$$\downarrow t_1$$
$$\vec{x}_1 = [1, 1, 1, 1]$$

Petri Net Dynamics Example - 3



$$\vec{x}_0 = [2, 0, 0, 1]$$

$$\downarrow t_1$$

$$\vec{x}_1 = [1, 1, 1, 1]$$

$$\downarrow t_3$$

$$\vec{x}_2 = [0, 1, 0, 0]$$

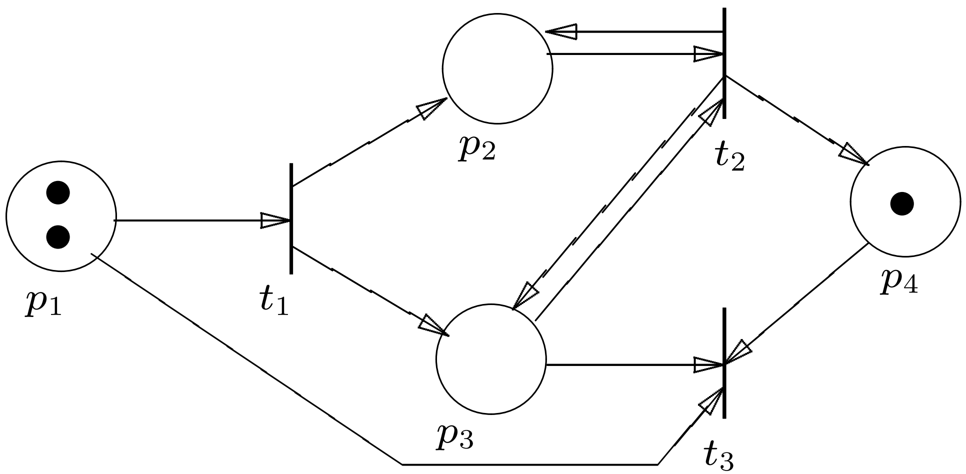
The Reachability Set

Definition: 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 $\vec{z} \in R(\vec{x})$.

Reachability Set Example



$\vec{x}_0 = [2, 0, 0, 1]$

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

Firing Vector and Incidence Matrix

Definition: Let $N = (P, T, A, w, \vec{x}_0)$ be a petri net with $P = \{p_1, \ldots, p_n\}$ and $T = \{t_1, \ldots, t_m\}$. A firing vector $\vec{u} = [0, \ldots, 0, 1, 0, \ldots, 0]$ 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.

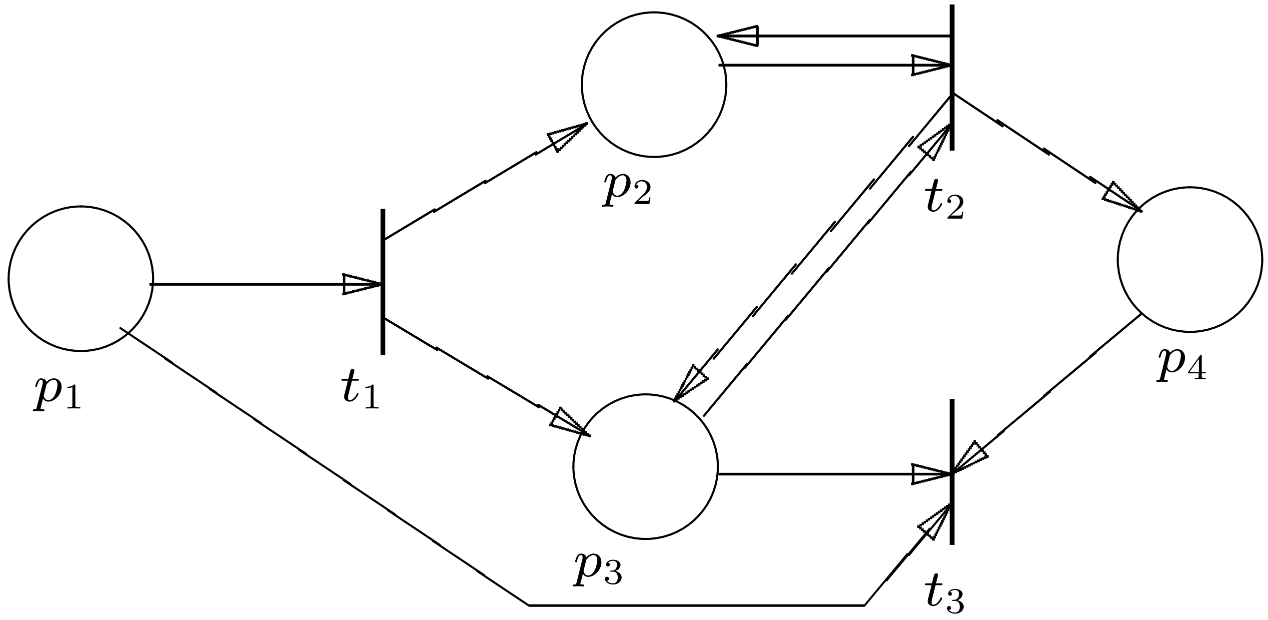
The incidence matrix $\boldsymbol{\mathcal{A}}$ is an $m \times n$ matrix whose (j, i) entry is

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

A state equation can be written as

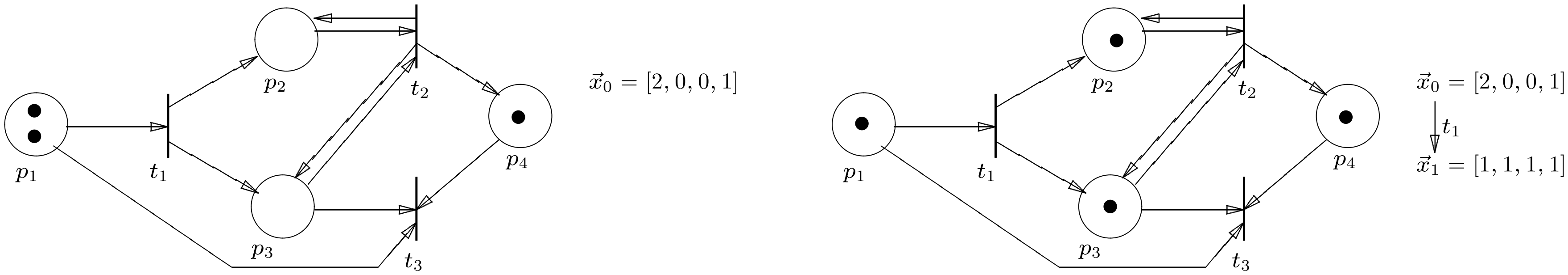
$$\vec{x}' = \vec{x} + \vec{u}\boldsymbol{\mathcal{A}}$$

Incidence Matrix Example



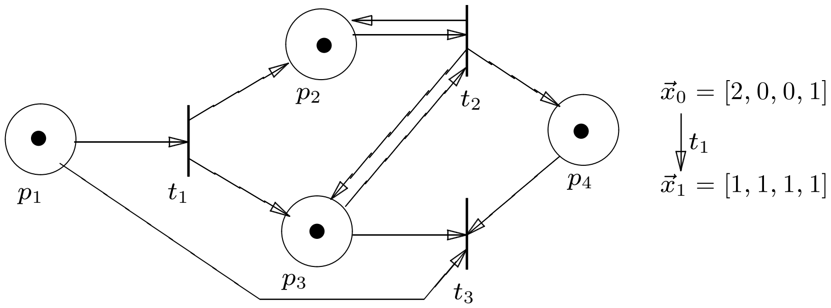
$$\boldsymbol{\mathcal{A}} = \begin{bmatrix} -1 & 1 & 1 & 0 \\ 0 & 0 & 0 & 1 \\ -1 & 0 & -1 & -1 \end{bmatrix},$$

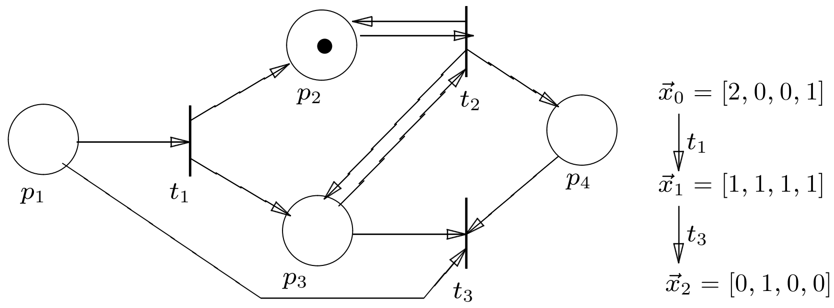
The Evaluation of State Equations



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

The Evaluation of State Equations - cont'd





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

The Evaluation of a Transition Sequence

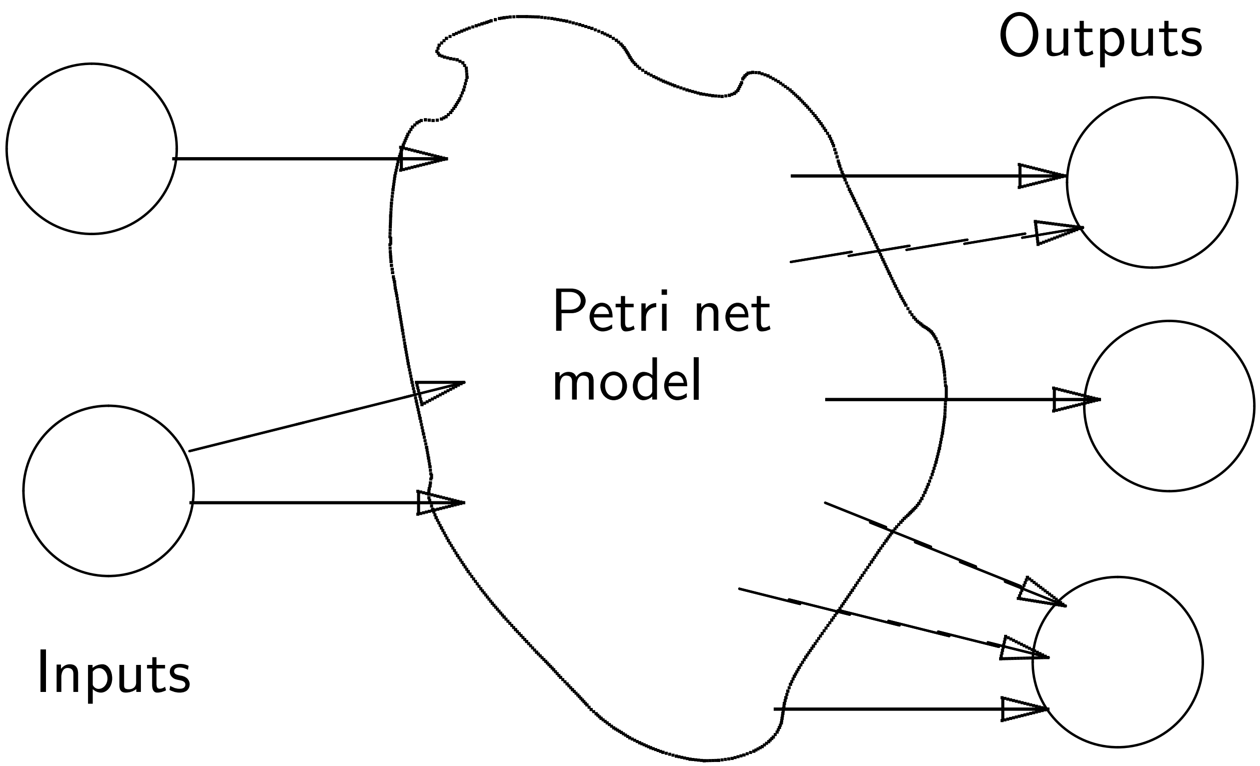
$N = (P, T, A, w, \vec{x}_0)$ is a Petri net;
$T' = \langle t_1, t_2, \ldots, t_i, \ldots, t_n \rangle, t_i \in T$ a sequence of n transitions with $\vec{u}_t$ the transition vector for t.
The state after firing of all transitions in T' is

$$\vec{x}_n = \vec{x}_0 + (\sum_{t \in T'} \vec{u}_t)\boldsymbol{\mathcal{A}}$$

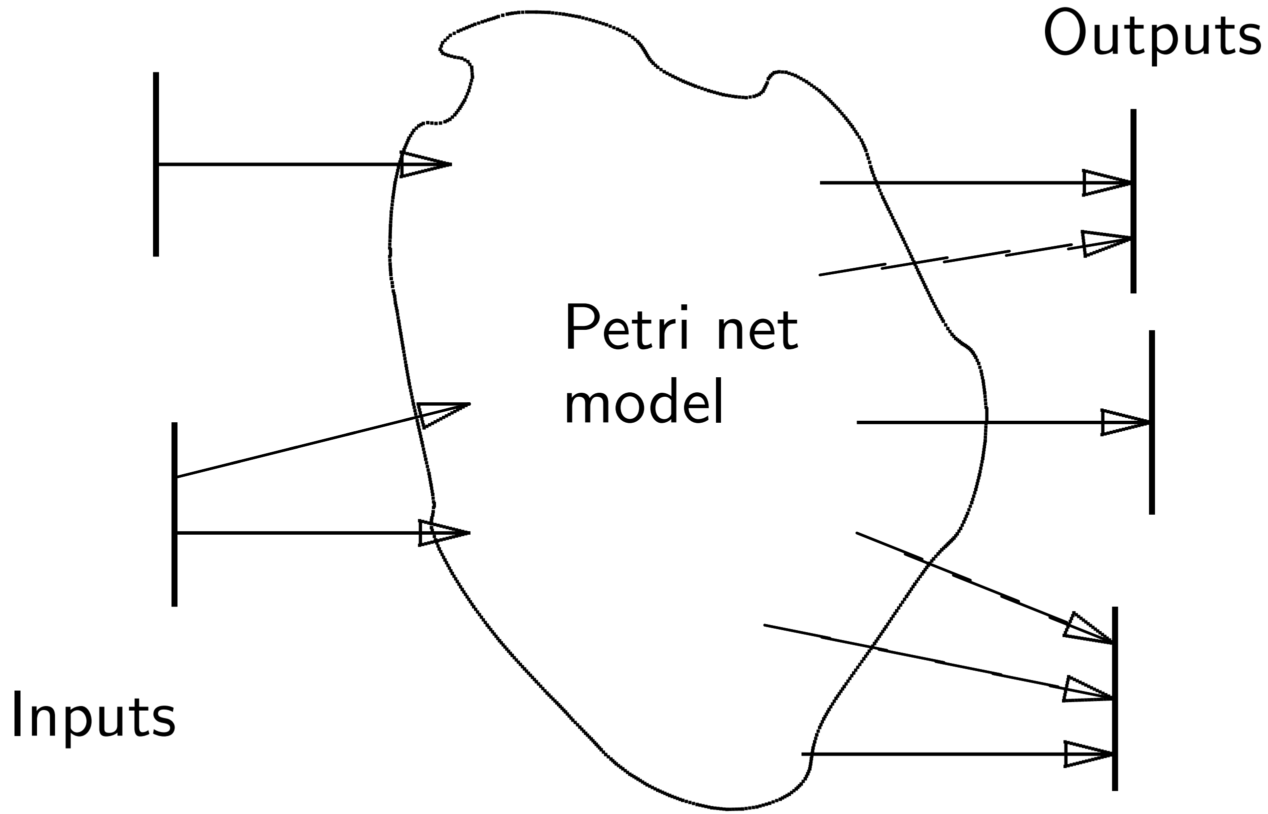
provided that for all $t_i \in T'$, t_i is enabled in state

$$\vec{x}_{i-1} = \vec{x}_0 + (\sum_{t \in \langle t_1, \ldots, t_{i-1} \rangle} \vec{u}_t)\boldsymbol{\mathcal{A}}.$$

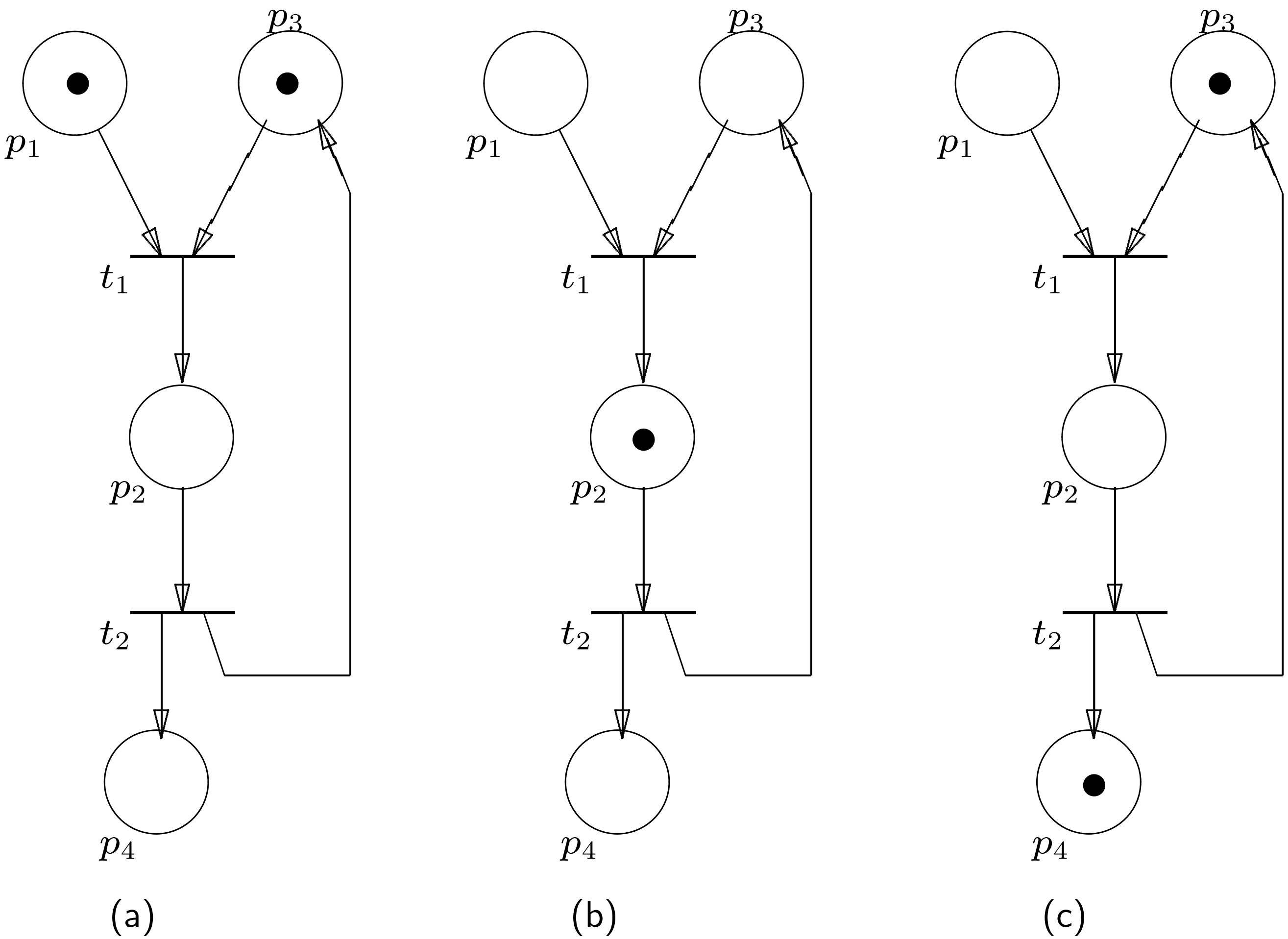
I/O Modeled as Places



I/O Modeled as Transitions

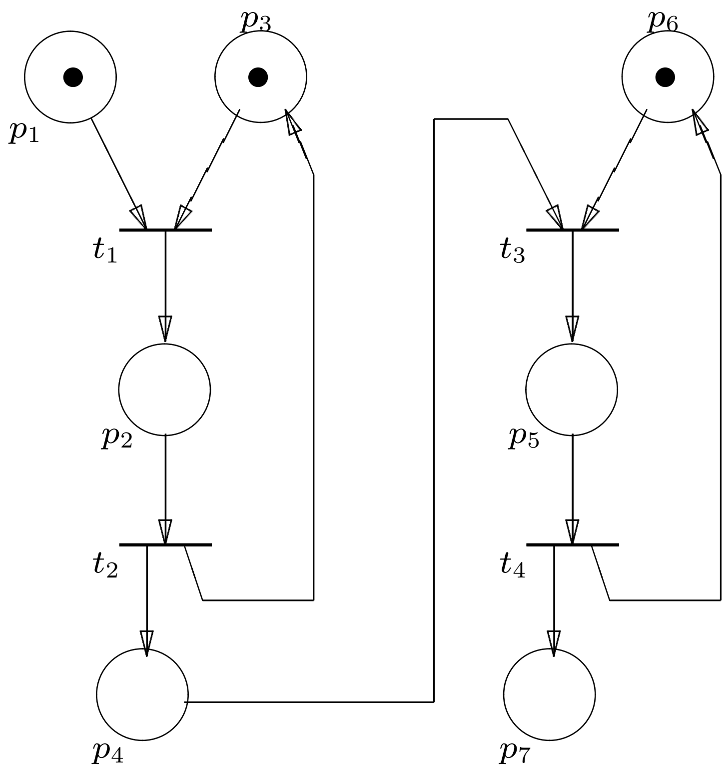


A Server Modeled as Petri Net



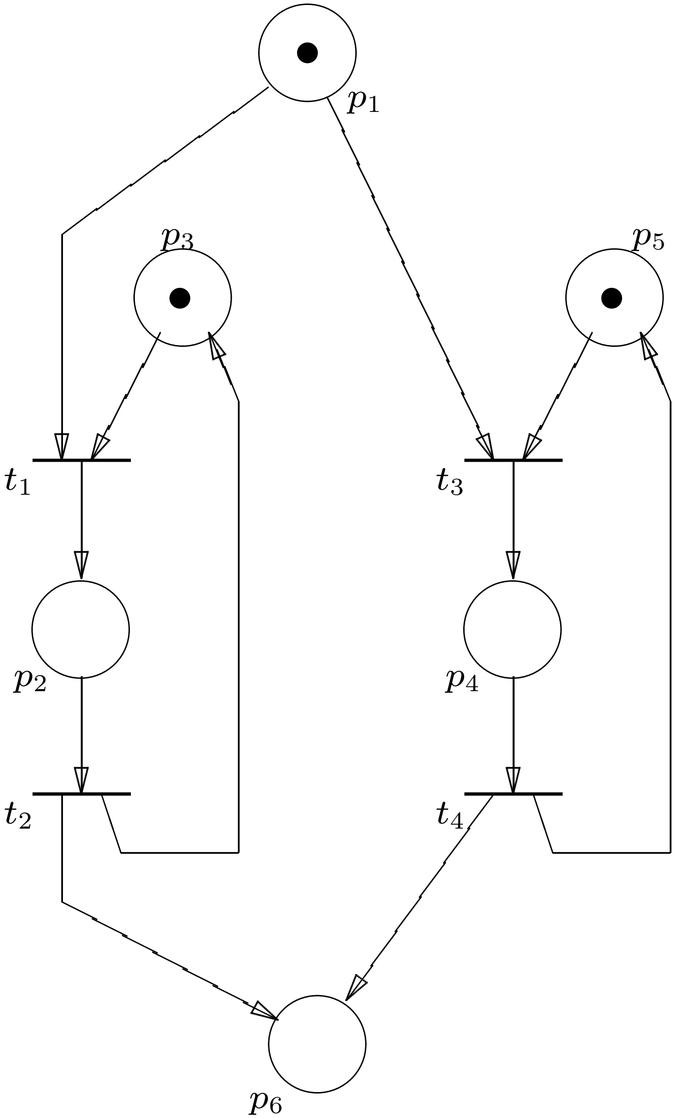
Customers arrive at input p_1 and depart at output p_4.

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.

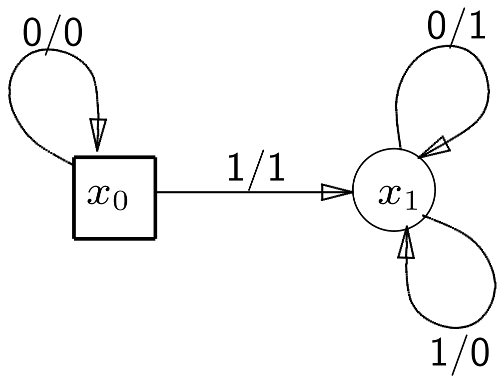
A Finite State Machine Modeled as Petri Net

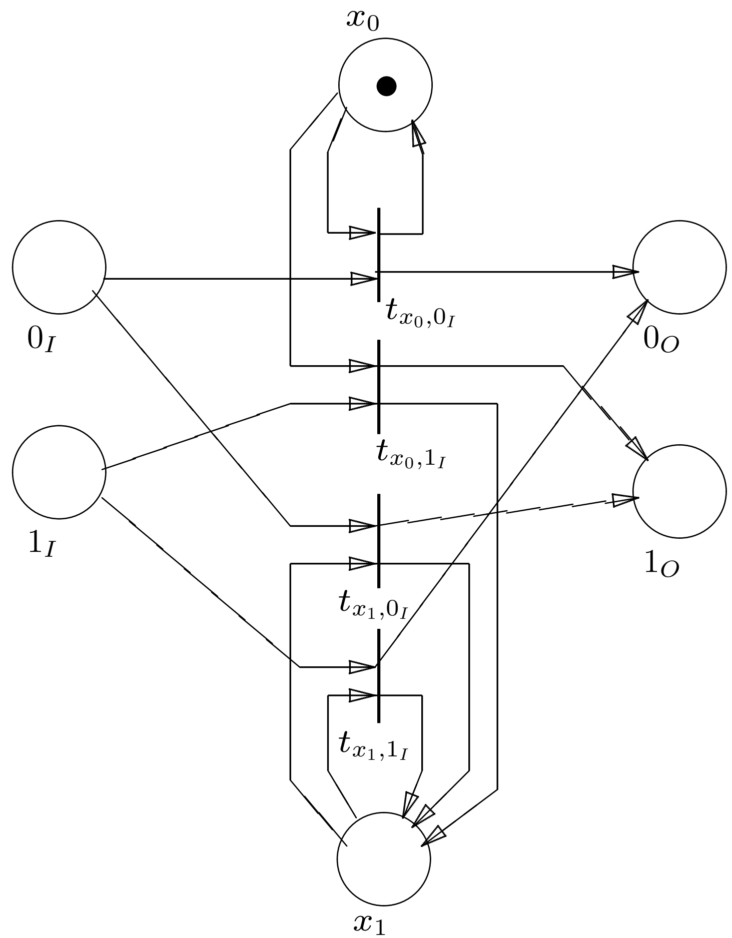
FSM $M = (\Sigma, \Delta, X, x_0, g, f)$ with mutually exclusive sets Σ and Δ.
An equivalent Petri net is $N = (P, T, A, \vec{y}_0)$ with

$$
\begin{aligned}
P &= X \cup \Sigma \cup \Delta \\
T &= \{t_{x,a} | x \in X, a \in \Sigma\} \\
A &= I(t_{x,a}) \cup O(t_{x,a}) \ \ \forall t_{x,a} \in T \\
I(t_{x,a}) &= \{x, a\} \\
O(t_{x,a}) &= \{g(x, a), f(x, a)\} \\
\vec{y}_0 &= [1, 0, \ldots, 0]
\end{aligned}
$$

- Σ are input places;
- Δ are output places;
- X are internal places;
- Each (state,input) pair in M becomes a transition in N;
- Initial marking represents state x_0 and no input;

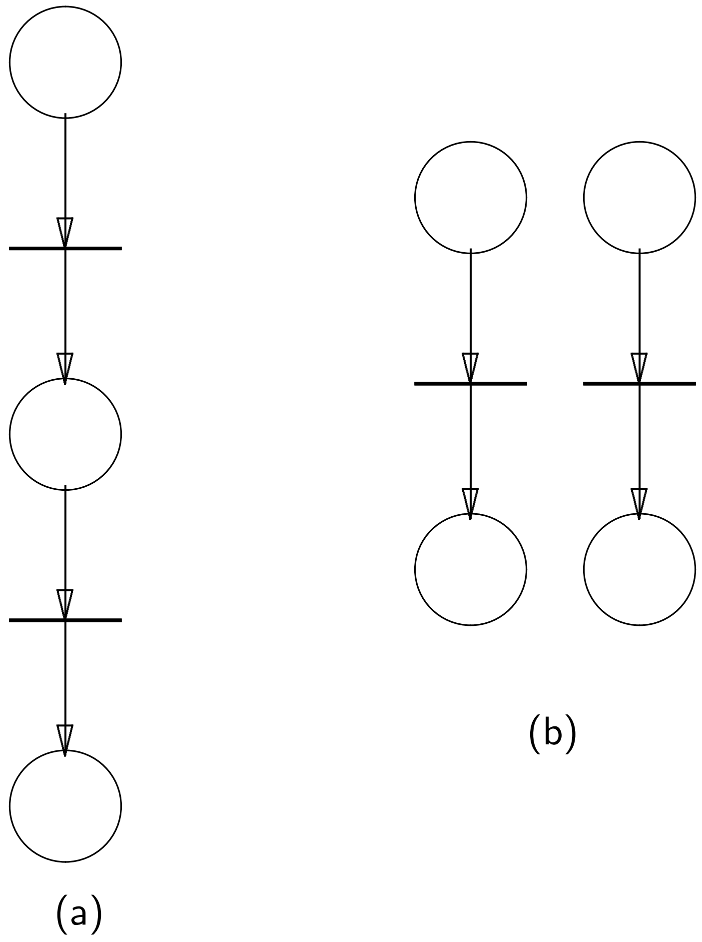
A FSM Modeled as Petri Net - Example



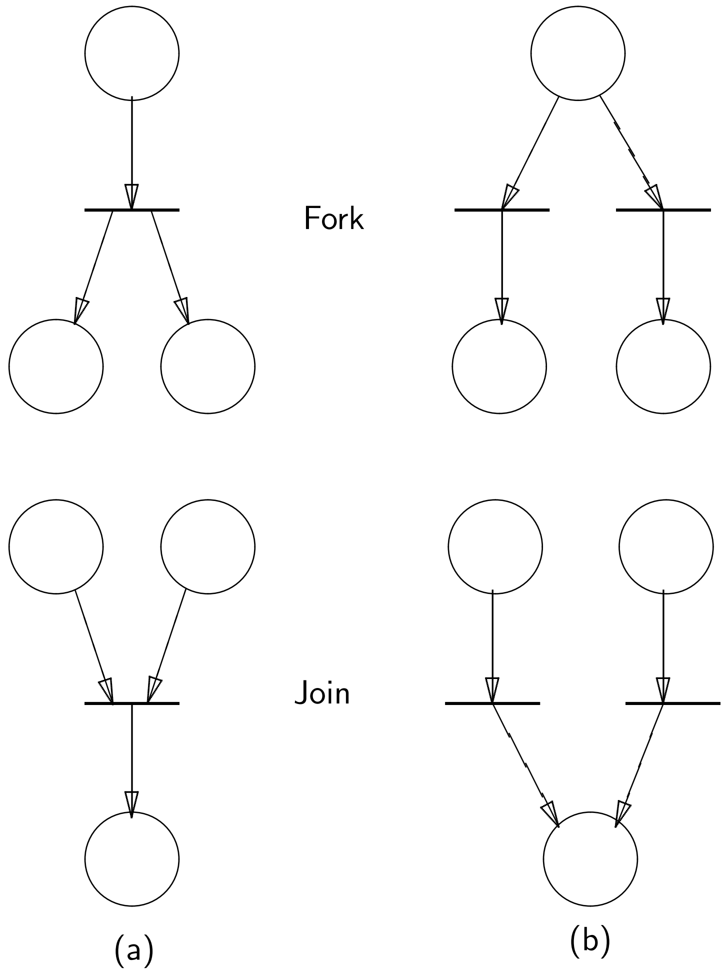


Computation of the two's complement of a binary number represented with the least significant bit first.

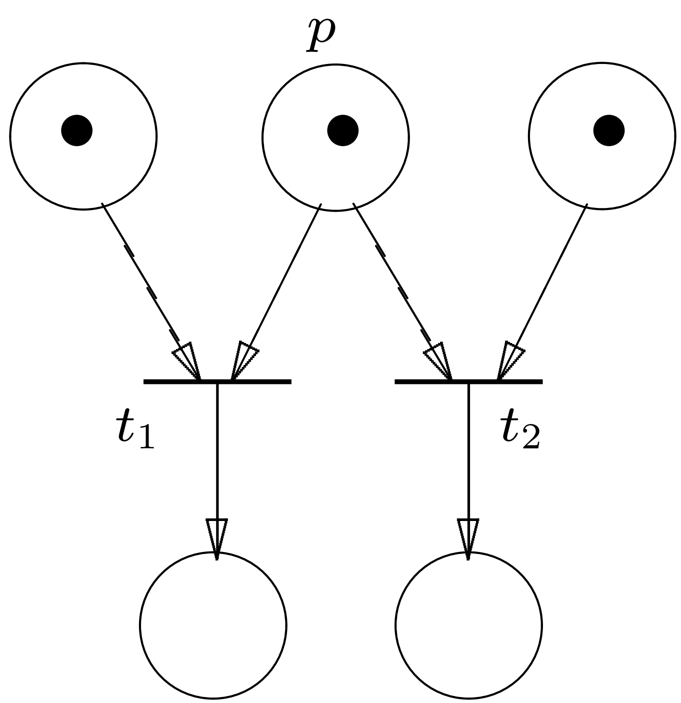
Sequence and Concurrency



Fork and Join



Conflict



The Mutual Exclusion Problem

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

Process A

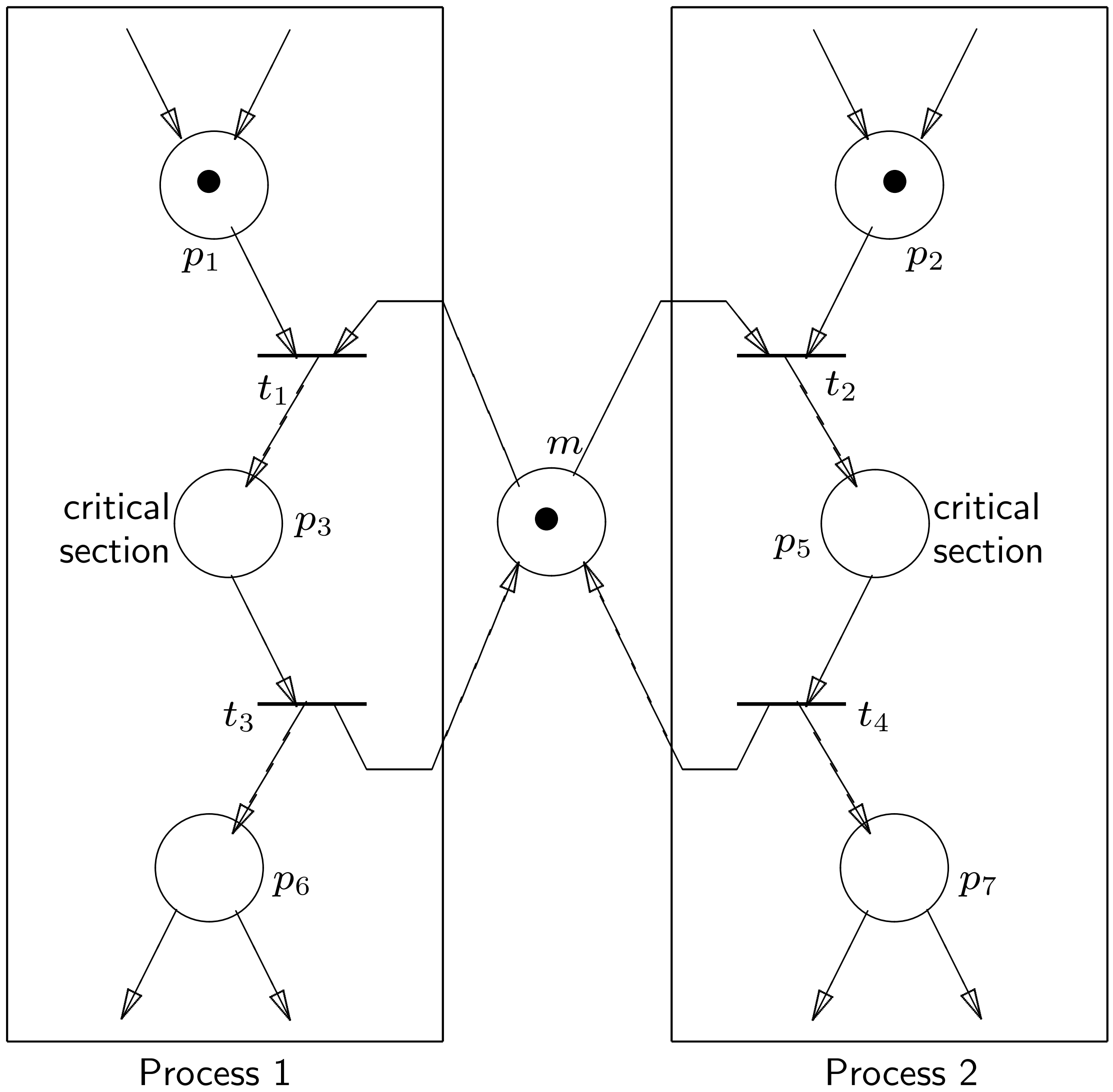
```
read(x);
set x <- x + 1;
write(x);
```

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
```

Mutual Exclusion Modeled with a Petri Net



The Mutual Exclusion Problem Solved

The TestSet instruction atomically tests a variable and, if successful, sets the variable.

```
while (not (TestSet(S==0,S<-1));
read(x);
set x <- x + 1;
write(x);
TestSet(True,S<-0);
```

Process A

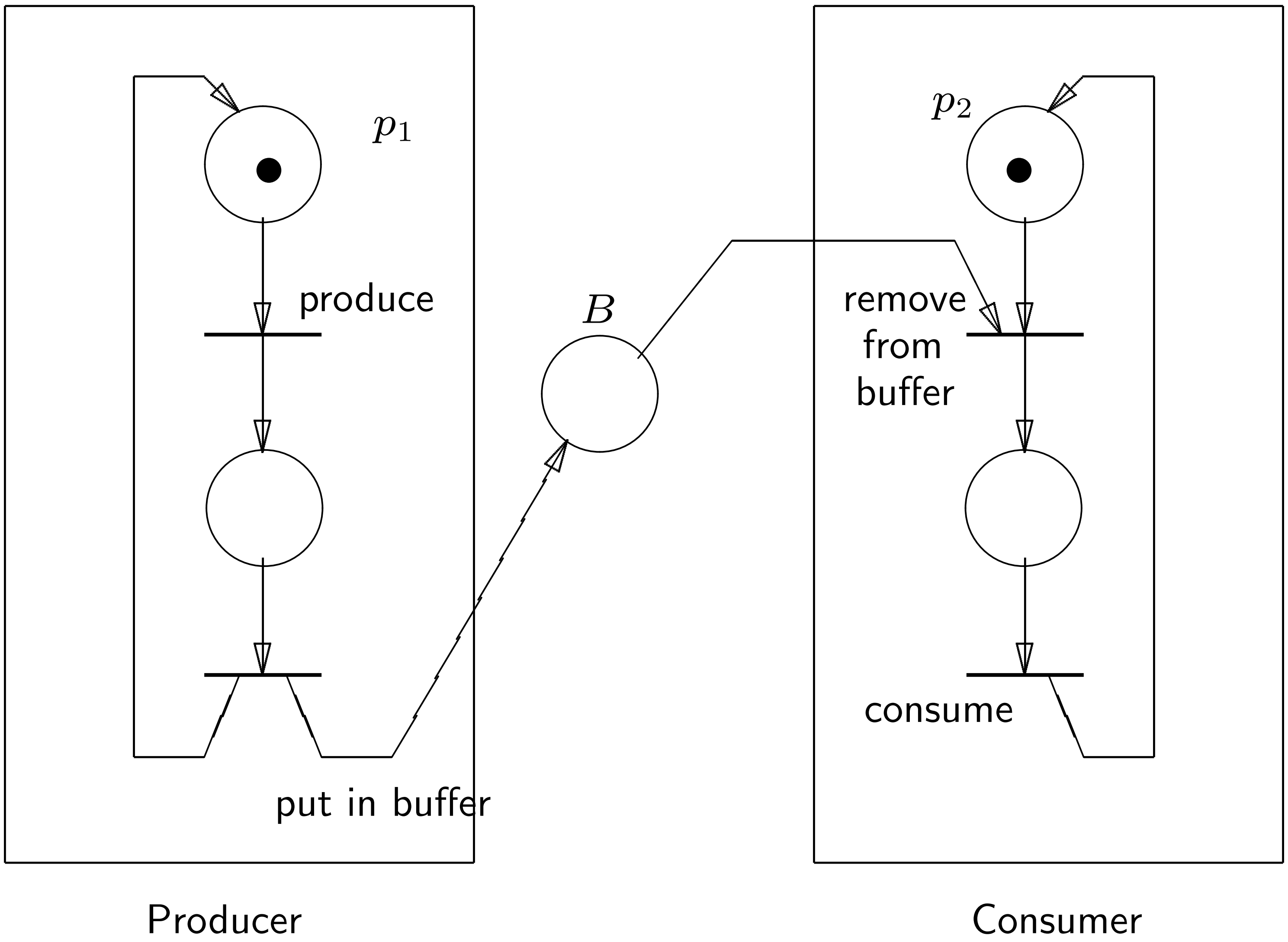
```
while (not (TestSet(S==0,S<-1));
read(x);
set x <- x + 1;
write(x);
TestSet(True,S<-0);
```

Process B

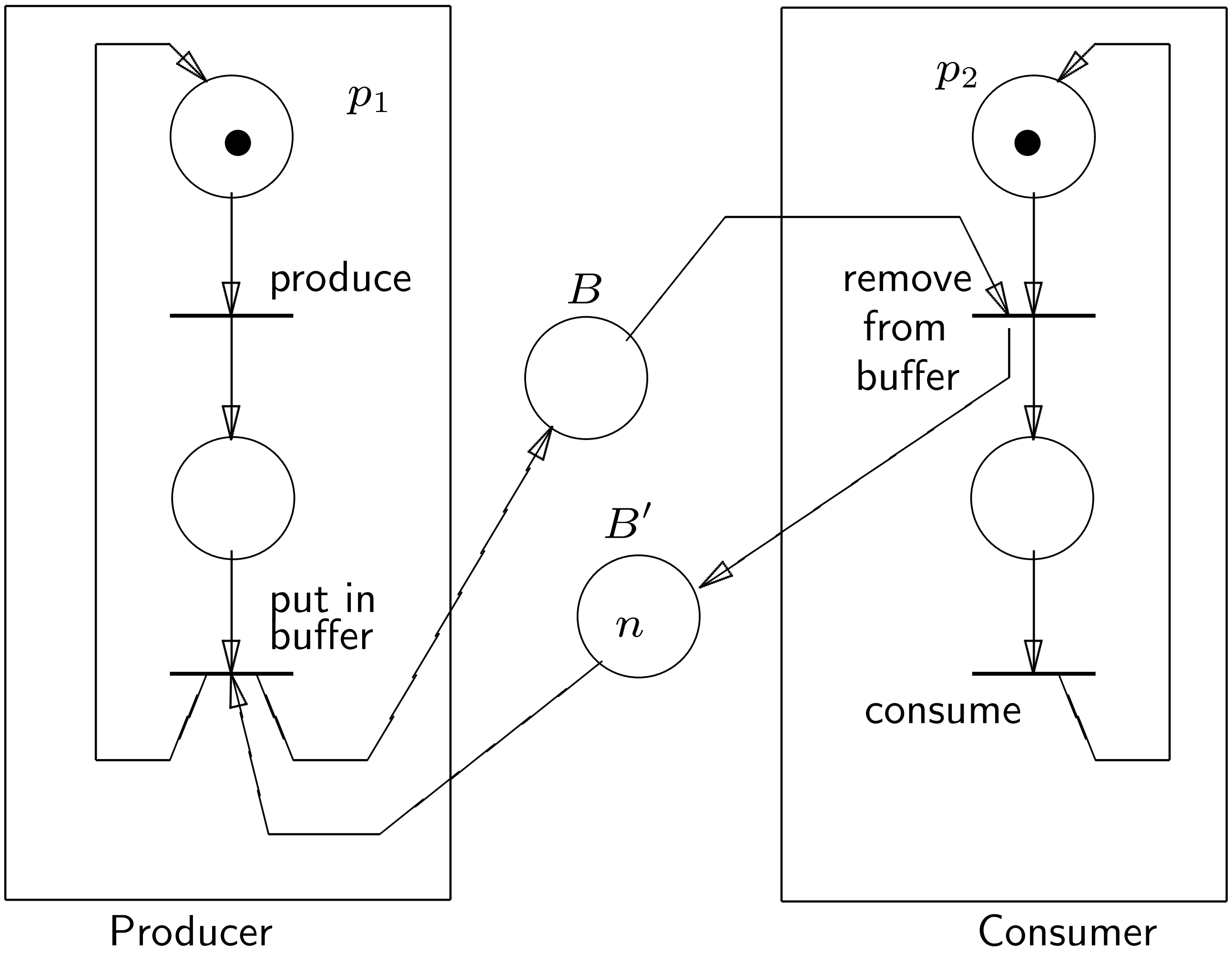
```
A.S<-1;
x <- 0;
A.read(x);
B.TestSet(S==0,S<-1);
A.set x <- x + 1;
A.write(x);
A.S<-0;
```

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

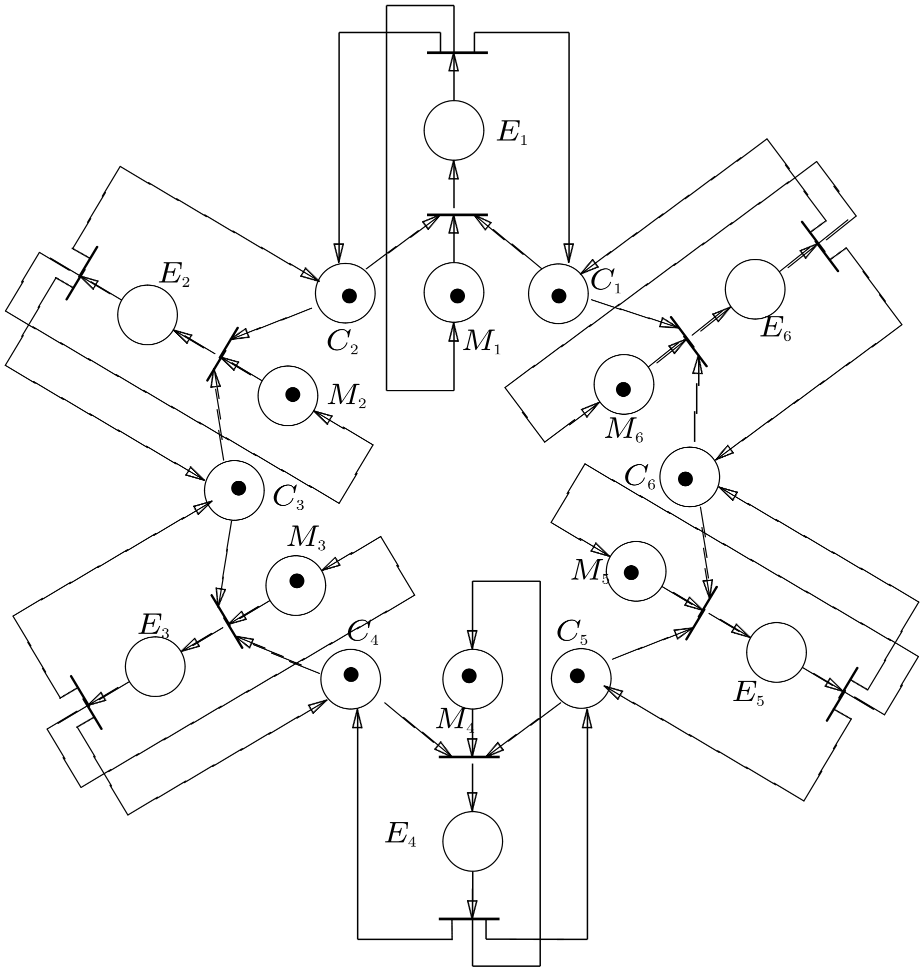
Producer/Consumer Relation



Producer/Consumer with Fixed Buffer Size



Dining Philosophers



C_i : Chop sticks
M_i : Philosopher meditating
E_i : Philosopher eating

Analysis Methods for Petri Nets

- Boundedness
- Conservation
- Liveness
- Coverability
- Persistence
- Coverability Tree

Boundedness

Definition: A place $p \in P$ in a Petri net $N = (P, T, A, w, \vec{x}_0)$ is k-bounded or k-safe if for all

$$\vec{y} \in R(\vec{x}_0) : y(p) \leq k.$$

The Petri net is called k-bounded or k-safe if all places $p \in P$ are k-bounded.

Conservation

Definition: A Petri net $N = (P, T, A, w, \vec{x}_0)$ is strictly conservative if for all $\vec{y} \in R(\vec{x}_0)$,

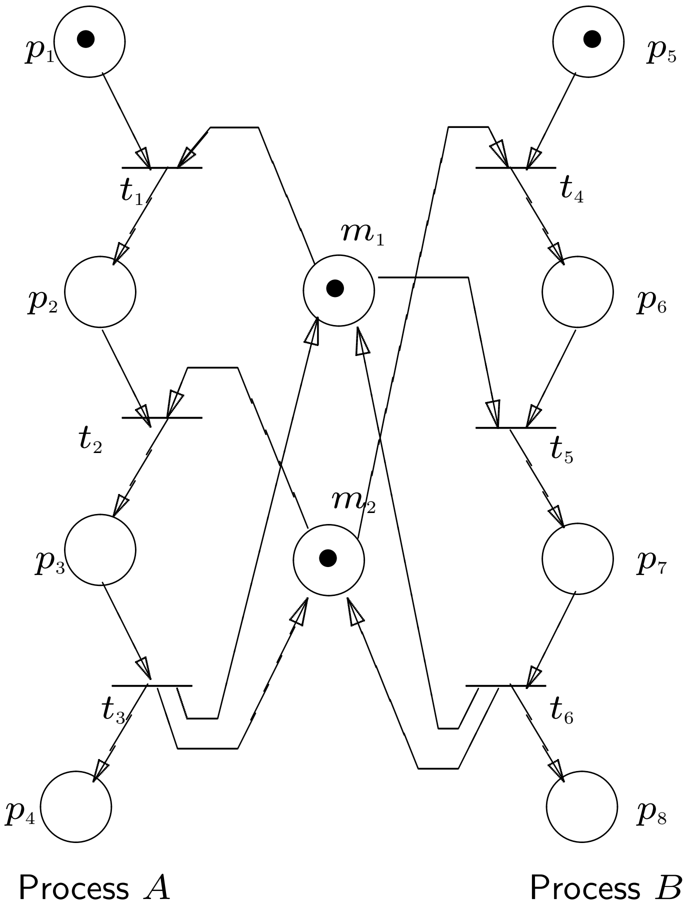
$$\sum_{p \in P} y(p) = \sum_{p \in P} x_0(p).$$

A Petri net $N = (P, T, A, w, \vec{x}_0)$ with n places is conservative with respect to a weighting vector $\vec{\gamma} = [\gamma_1, \gamma_2, ..., \gamma_n], \gamma_i \in \mathbb{N}$, if

$$\sum_{i=1}^{n} \gamma_i x(p) = \text{constant for all } p \in P \text{ and } \vec{x} \in R(\vec{x}_0).$$

The Petri net is conservative if it is conservative with respect to a weighting vector which has a positive non zero weight for all places.

Deadlock



Liveness

Definition: Let $N = (P, T, A, w, \vec{x}_0)$ be a Petri net and $\vec{x}$ a state reachable from $\vec{x}_0$.

L0-live: A transition t is live at level 0 in state $\vec{x}$ if it cannot fire in any state reachable from $\vec{x}$, i.e. it is deadlocked. .

L1-live: A transition t is live at level 1 in state $\vec{x}$ if it is potentially fire-able, i.e. if there exists a $\vec{y} \in R(\vec{x})$ such that t is enabled in $\vec{y}$.

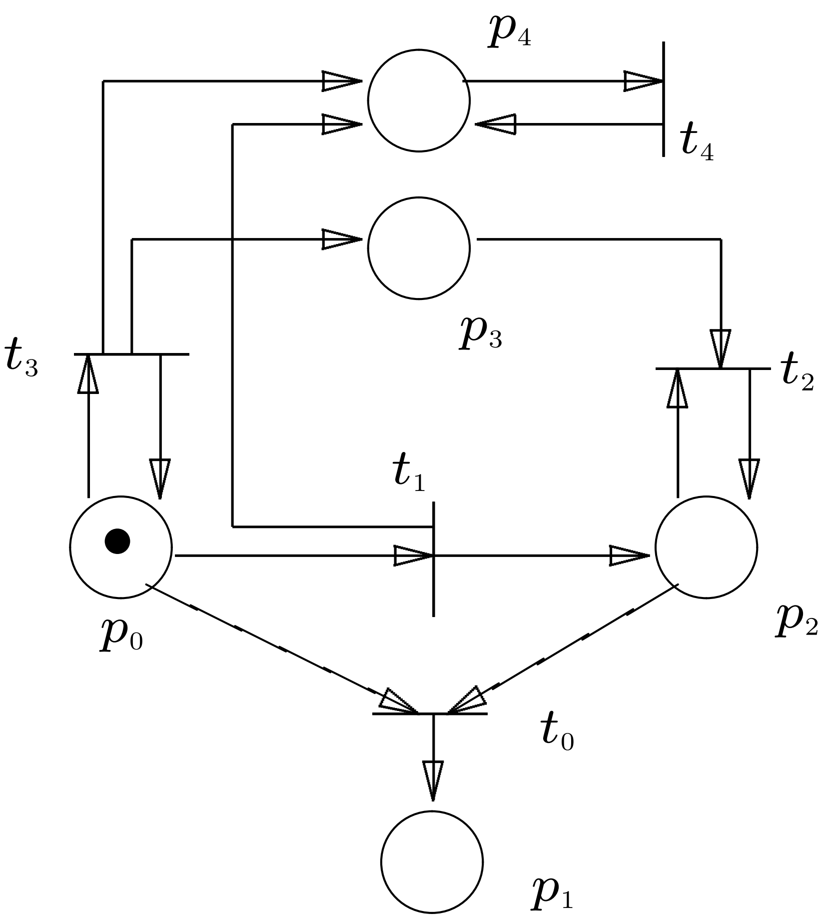
L2-live: A transition t is live at level 2 in state $\vec{x}$ if for every integer n there exists a firing sequence in which t occurs at least n times.

L3-live: A transition t is live at level 3 in state $\vec{x}$ if there is an infinite firing sequence in which t occurs infinitely often.

L4-live: A transition t is live at level 4 in state $\vec{x}$ if it is L1-live in every $\vec{y} \in R(\vec{x})$.

A Petri net is live at level i if every transition is live at level i.

Liveness Example



- t_0 is dead;
- t_1 is L1-live;
- t_2 is L2-live;
- t_3 is L3-live;
- t_4 is L4 live.

Persistence

Definition: Two transitions are persistent with respect to each other if, when both are enabled the firing of one does not disable the other.

A Petri net is persistent if any two transitions are persistent with respect to each other.

Coverability

Definition: $N = (P, T, A, w, \vec{x}_0)$ is a Petri net; $\vec{x}$ and $\vec{y}$ are arbitrary states;

State $\vec{x}$ covers state $\vec{y}$ if in $\vec{x}$ at least all transitions are enabled which are enabled in $\vec{y}$:

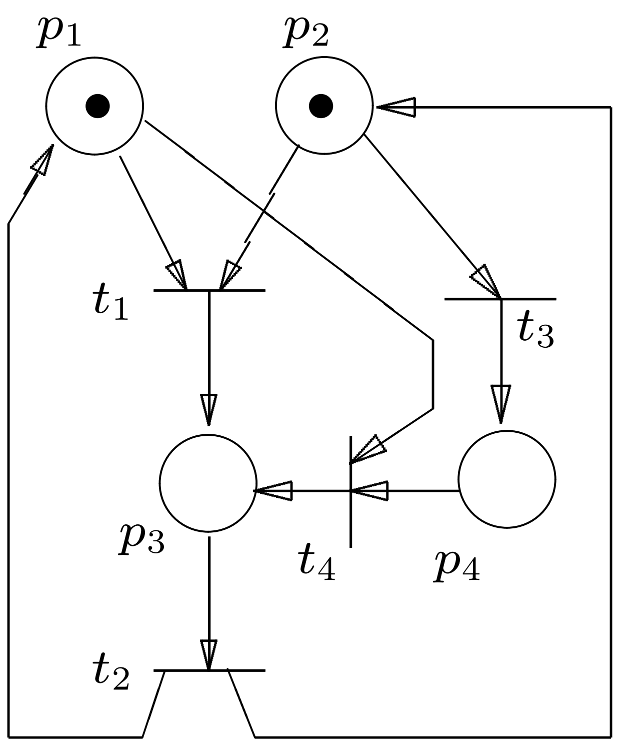
$$x(p) \geq y(p) \forall p \in P.$$

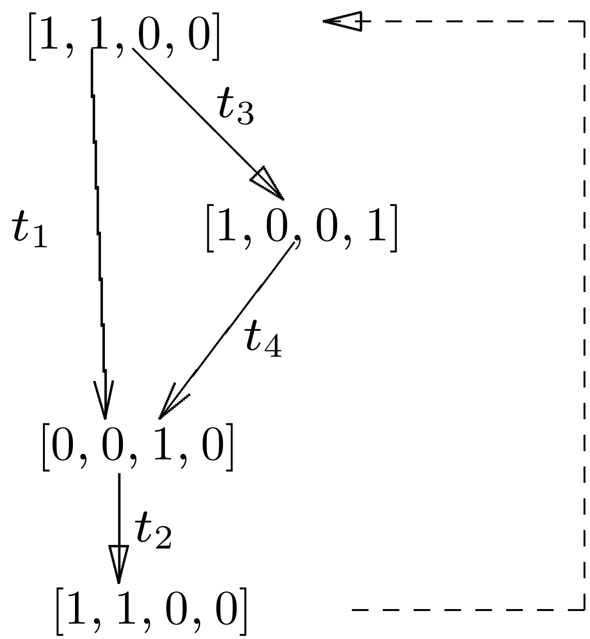
State $\vec{x}$ strictly covers state $\vec{y}$ if $\vec{x}$ covers $\vec{y}$ and, in addition,

$$\exists p \in P : x(p) > y(p).$$

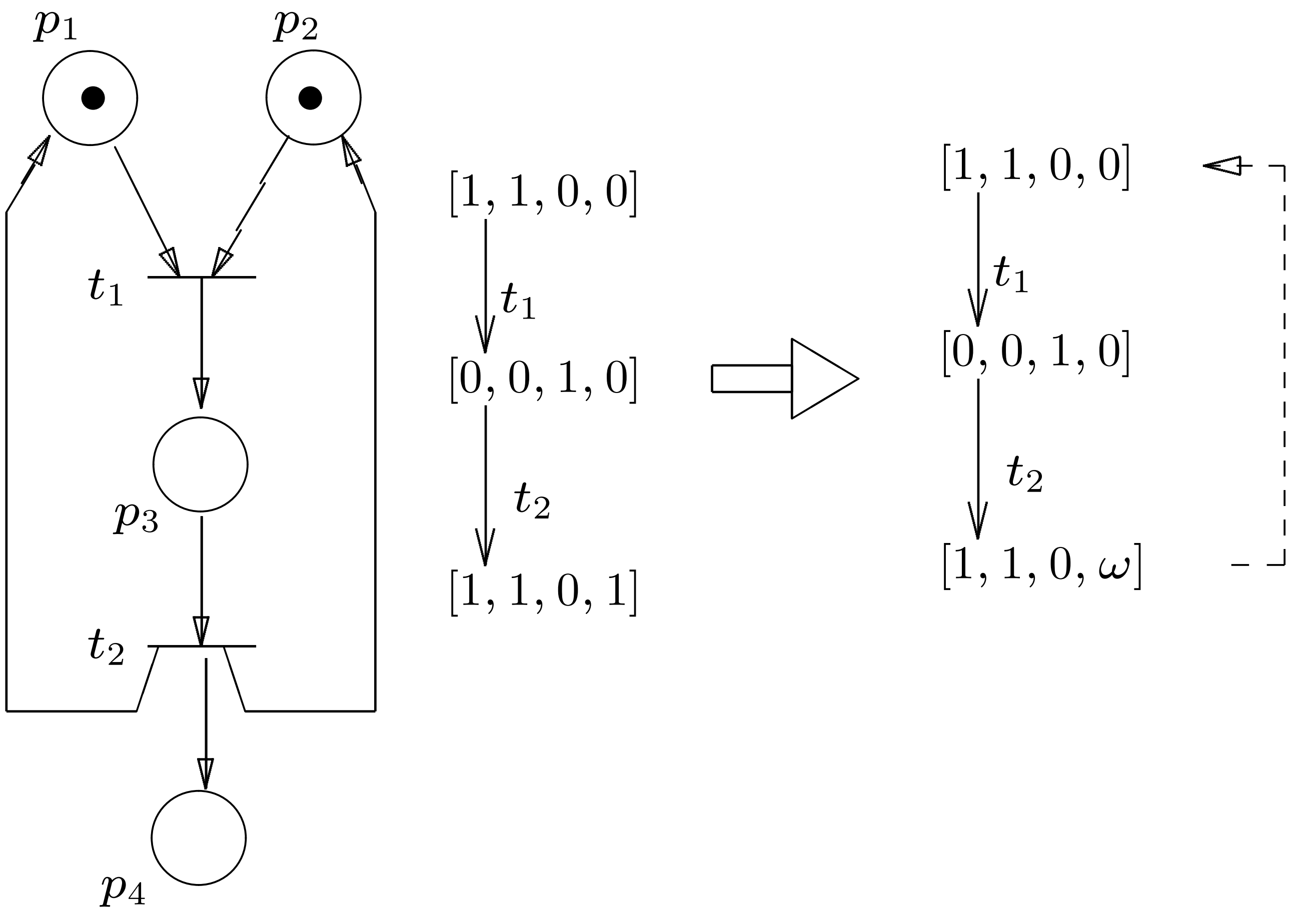
Let $\vec{x} \in R(\vec{x}_0)$. A state $\vec{y}$ is coverable by $\vec{x}$ iff there exists a state $\vec{x}' \in R(\vec{x})$ such that $x'(p) \geq y(p)$ for all $p \in P$.

Coverability Tree for a Finite State Space





Coverability Tree for an Infinite State Space



Coverability Tree Definition

Definition: Let $N = (P, T, A, w, \vec{x}_0)$ be a Petri.

A coverability tree is a tree where the arcs denote transitions $t \in T$ and the nodes represent ω-enhanced states of the Petri net.

The root node of the tree is $\vec{x}_0$.

A terminal node is an ω-enhanced state in which no transition is enabled.

A duplicate node is an ω-enhanced state which already exists somewhere else in the coverability tree.

An arc t connects two nodes $\vec{x}$ and $\vec{y}$ in the tree, iff firing of t in state $\vec{x}$ leads to state $\vec{y}$.

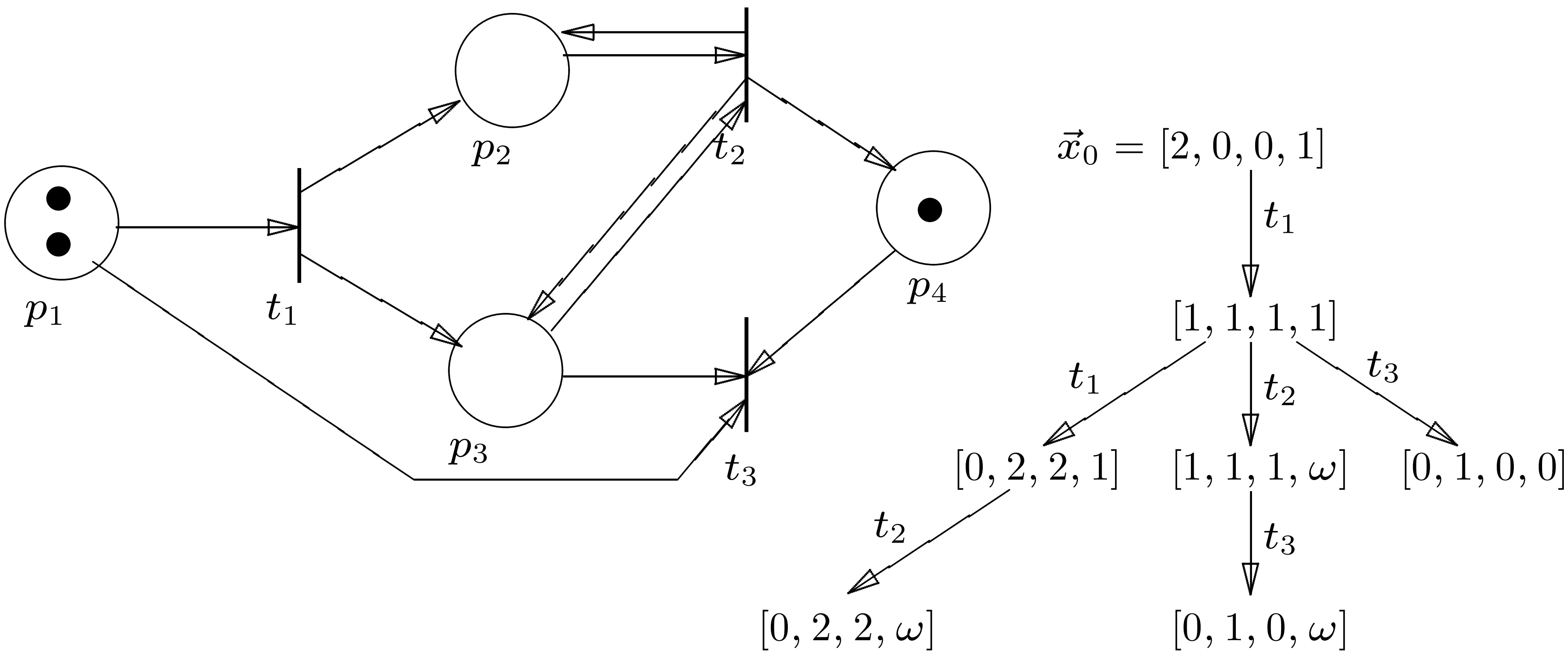
Coverability Tree Algorithm

Given is the Petri net $N = (P, T, A, w, \vec{x}_0)$.

Algorithm:

Step 1. Set L, the list of open nodes, to $L := \{\vec{x}_0\}$.
Step 2. Take one node from L, named $\vec{x}$, and remove it from L;
Step 2.1. `if` $G(\vec{x}, t) = \vec{x} \;\; \forall t \in T$
`then` $\vec{x}$ is a terminal node `goto` Step 3;
Step 2.2. `for all` $\vec{x}' \in G(\vec{x}, t), t \in T, \vec{x} \neq \vec{x}'$
Step 2.2.1. `do if` $x(p) = \omega$ `then` set $x'(p) := \omega$;
Step 2.2.2. `if` there is a node $\vec{y}$ already in the tree, such that $\vec{x}$' covers $\vec{y}$
and there is a path from $\vec{y}$ to $\vec{x}$',
`then` set $x'(p) := \omega$ for all p for which $x'(p) > y(p)$;
Step 2.2.3. `if` $\vec{x}$' is not a duplicate node `then` $L := L \cup \{\vec{x}'\}$;
Step 3. `if` L is not empty `then goto` Step 2.

Coverability Tree Example

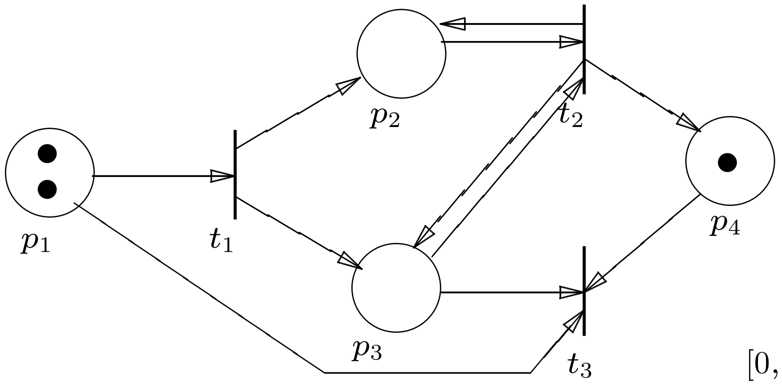


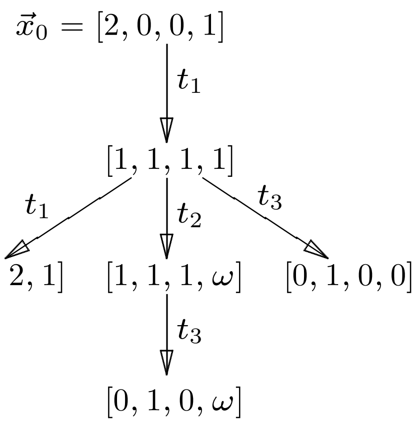
Coverability Tree: Safeness and Boundedness

- A Petri net can be k-bounded if the ω symbol never appears in its coverability tree.
- If the coverability tree contains an ω, a transition cycle to exceed a given k-bound can be identified.
- The coverability tree does not inform about the number of cycles required.

Coverability Tree: Conservation

- Recall: $\sum_{i=1}^{n} \gamma_i x(p) =$ constant for all $p \in P$ and $\vec{x} \in R(\vec{x}_0)$.
- If there is an ω the corresponding γ_i must be 0.
- We evaluate the the weighted sum for every node in the coverability tree. The net is conservative iff the result is the same for all nodes.





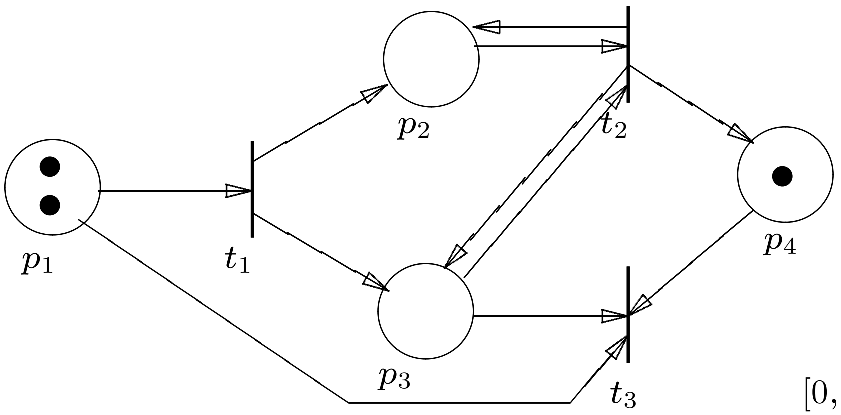
conservative for $\vec{\gamma} = [2, 3, 1, 0]$?

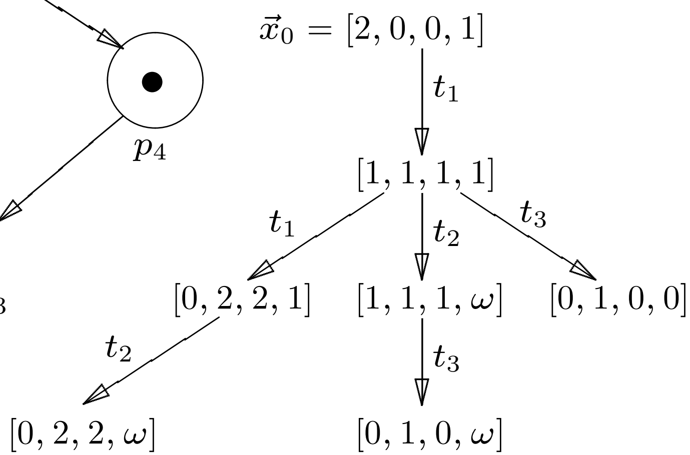
Computing the Conservation Vector

- Set $\gamma_i = 0$ for every unbounded place p_i.
- For b bounded places and r nodes in the coverability tree we set up r equations with $b + 1$ unknown variables

$$\sum_{i=1}^{r} \gamma_i x(p_i) = C.$$

Computing the Conservation Vector - Example





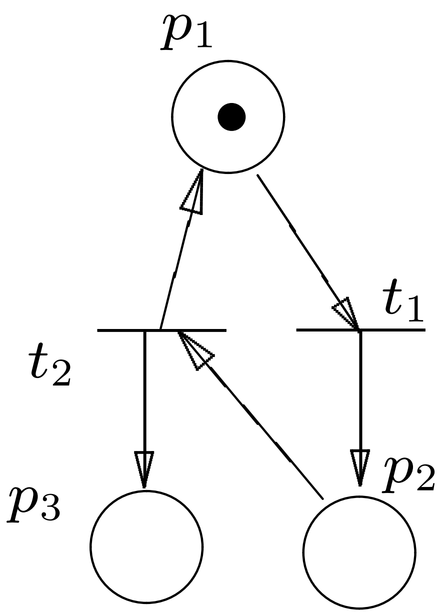
$$
\begin{aligned}
\gamma_4 &= 0 \\
2\gamma_1 + 0\gamma_2 + 0\gamma_3 &= C \\
1\gamma_1 + 1\gamma_2 + 1\gamma_3 &= C \\
0\gamma_1 + 2\gamma_2 + 2\gamma_3 &= C \\
0\gamma_1 + 1\gamma_2 + 0\gamma_3 &= C
\end{aligned}
$$

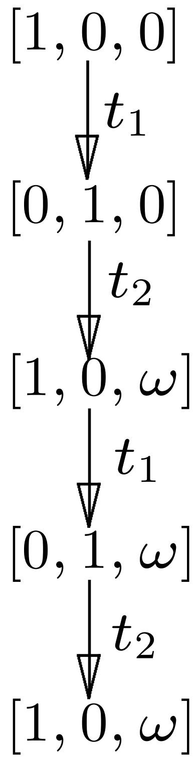
Only non-negative solution is $\vec{\gamma} = [0, 0, 0, 0]$ and $C = 0$.

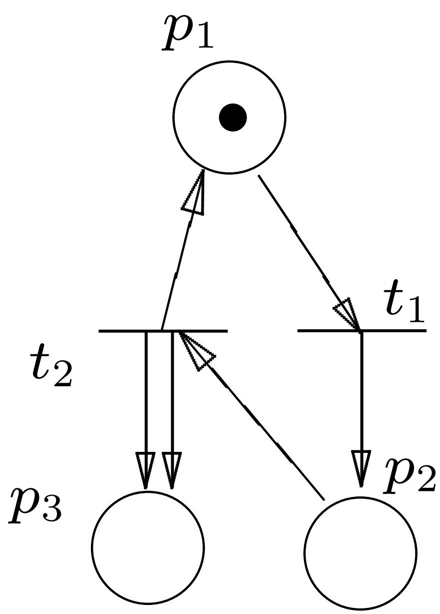
Coverability Tree: Coverability and Reachability

- The coverability problem can be solved by inspection of the coverability tree.
- The shortest transition sequence leading to a covering state can be found efficiently.
- The reachability problem cannot be solved in general.

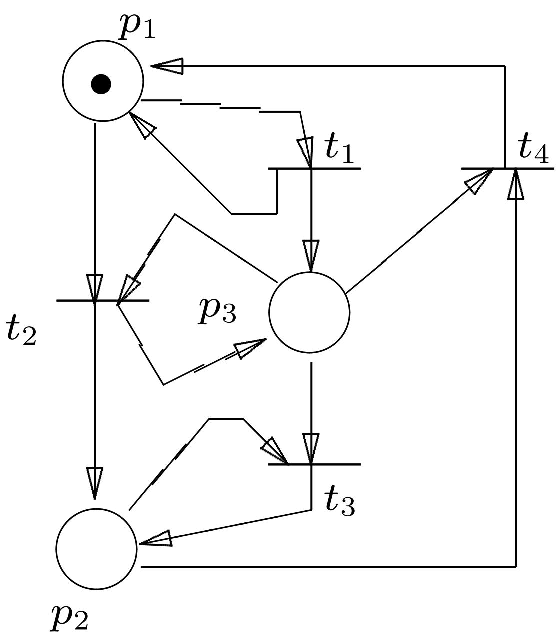
Distinct Petri Nets with Identical Coverability Tree - 1

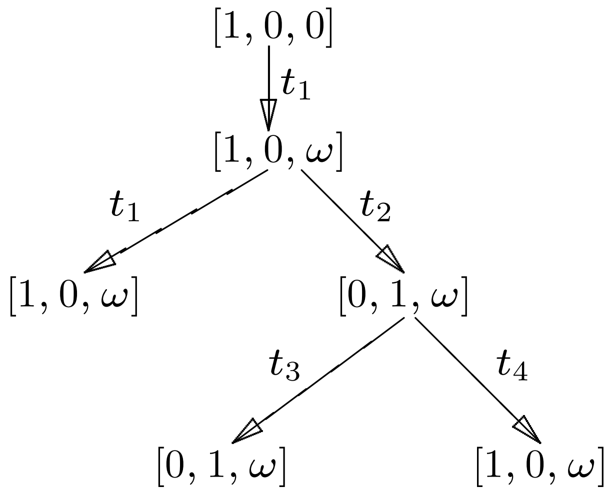


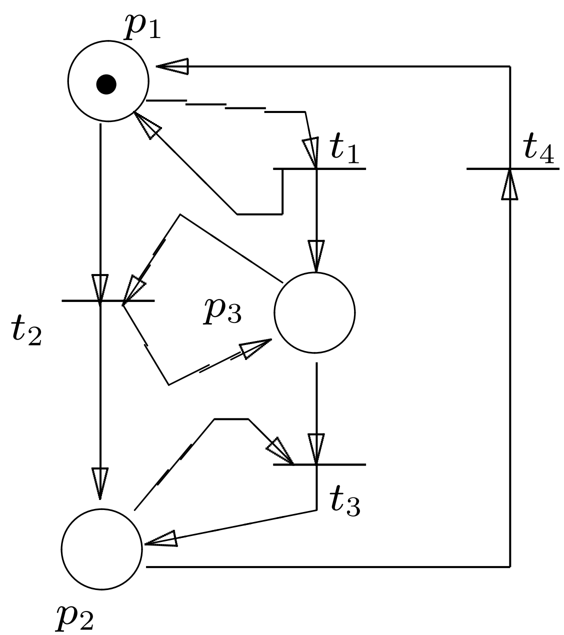




Distinct Petri Nets with Identical Coverability Tree - 2







Can deadlock after t_1, t_2, t_3.

Cannot deadlock.

Summary

- Petri Net Dynamics
- Reachability Set
- Model Patterns
  - ⋆ Server
  - ⋆ Composition
  - ⋆ Fork-Join
  - ⋆ Conflict
  - ⋆ Mutual Exclusion
  - ⋆ Consumer-Producer
- Analysis Problems
  - ⋆ Boundedness
  - ⋆ Conservation
  - ⋆ Liveness
  - ⋆ Coverability
  - ⋆ Persistence
  - ⋆ Dead-lock
- Coverability Tree