PETRI NETS

Pedro U. Lima

Instituto Superior Técnico (IST)
Instituto de Sistemas e Robótica (ISR)
Av.Rovisco Pais, 1
1049-001 Lisboa
PORTUGAL

November 2002

Petri Nets

Basic Notions
Comparison with Automata
Analysis Problems and Techniques
Control of Petri Nets

DEFINITION OF PETRI NET

Def.: A Petri net (PN) graph or structure is a weighted bipartite graph (P,T,A,w), where:

$P=\{p_1, p_2, \dots p_n\}$ is the finite set of *places*

$T=\{t_1, t_2, \dots t_m\}$ is the finite set of *transitions*

$A \subseteq (P \times T) \cup (T \times P)$ is the set of arcs from places to transitions (p_i,t_j) and transitions to places (t_j,p_i)

$w: A \rightarrow \{1,2,3,\dots\}$ is the *weight function* on the arcs

Also useful:

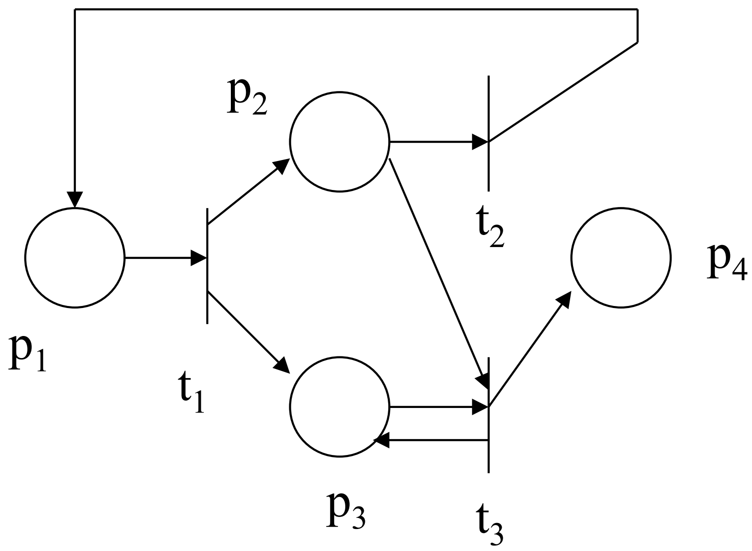
Set of input places to $t_j \in T$

$$I(t_j) = \{p_i \in P : (p_i, t_j) \in A\}$$

Set of output places from $t_j \in T$

$$O(t_j) = \{p_i \in P : (t_j, p_i) \in A\}$$

EXAMPLE OF PETRI NET



$$P = \{p_1, p_2, p_3, p_4\}$$
$$T = \{t_1, t_2, t_3\}$$
$$A = \{(p_1,t_1),(p_2,t_2),(p_2,t_3),(p_3,t_3),(t_1,p_2),(t_1,p_3),(t_2,p_1),(t_3,p_3),(t_3,p_4)\}$$
All weights are = 1

MARKING, DYNAMICS AND STATE SPACE

Def.: A *marked Petri net* is a five-tuple (*P,T,A,w,***x**), where (*P,T,A,w*) is a Petri net graph and **x** is a marking of the set of places *P*; $\mathbf{x} = [x(p_1), x(p_2), \ldots, x(p_n)] \in \mathrm{N}^n$ is the row vector associated with *x*.

Def. (PN dynamics): The *state transition function*, $f : \mathrm{N}^n \times T \to \mathrm{N}^n$ of Petri net (*P,T,A,w,***x**), is defined for transition $t_j \in T$

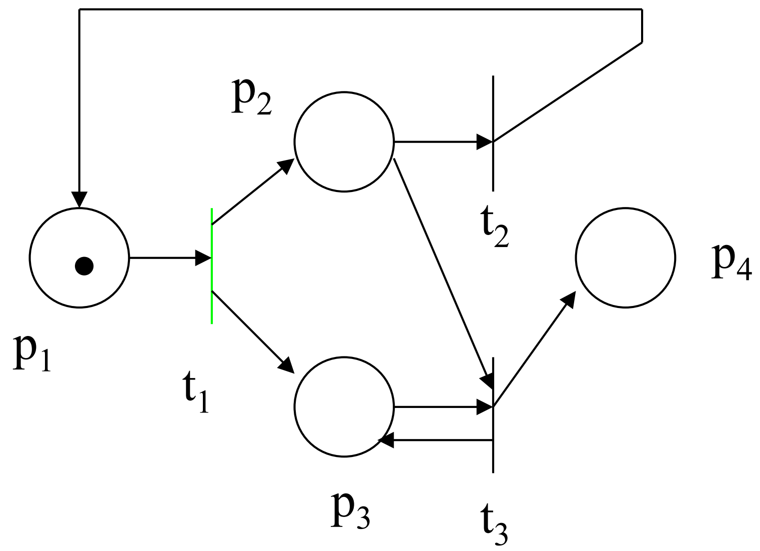
iff

$$x(p_i) \geq w(p_i, t_j), \forall p_i \in I(t_j).$$ ← ***Enabled*** $\boldsymbol{t_j}$

If $f(\mathbf{x}, t_j)$ is defined, the new state is $\mathbf{x}' = f(\mathbf{x}, t_j)$ where

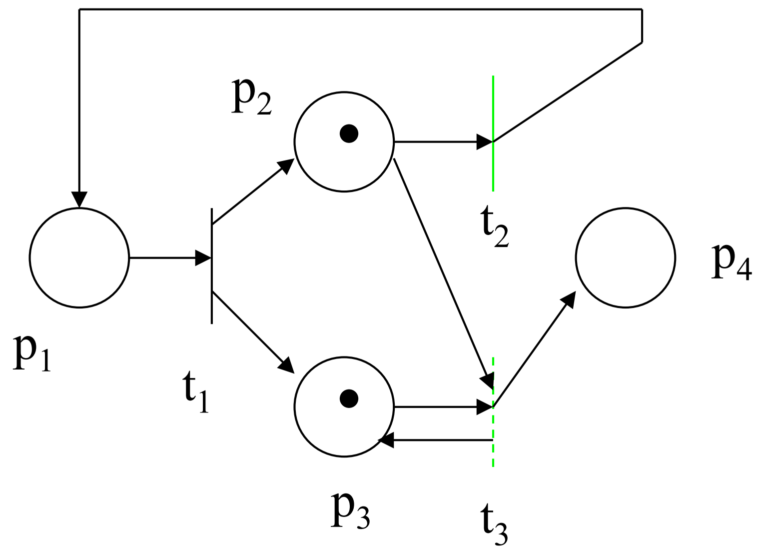
$$x'(p_i) = x(p_i) - w(p_i, t_j) + w(t_j, p_i), \ i = 1, \ldots, n.$$

MARKING, DYNAMICS AND STATE SPACE



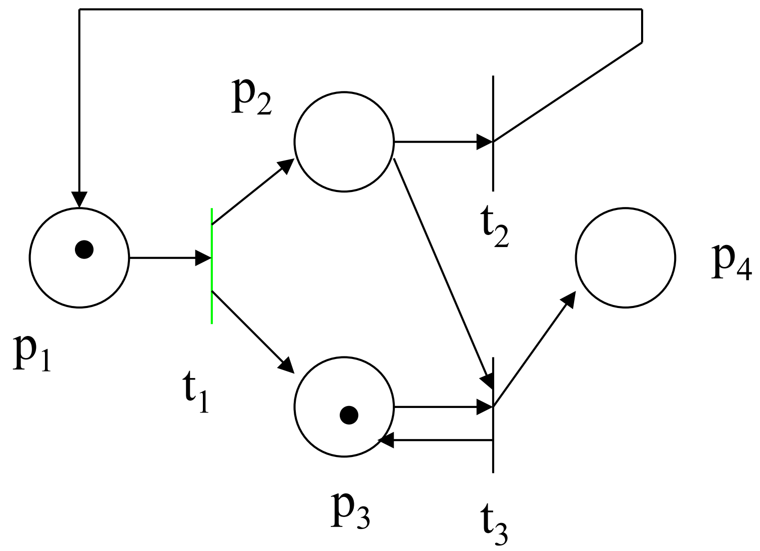
$$\mathbf{x} = \mathbf{x}_0 = \begin{bmatrix} 1 & 0 & 0 & 0 \end{bmatrix}$$

MARKING, DYNAMICS AND STATE SPACE



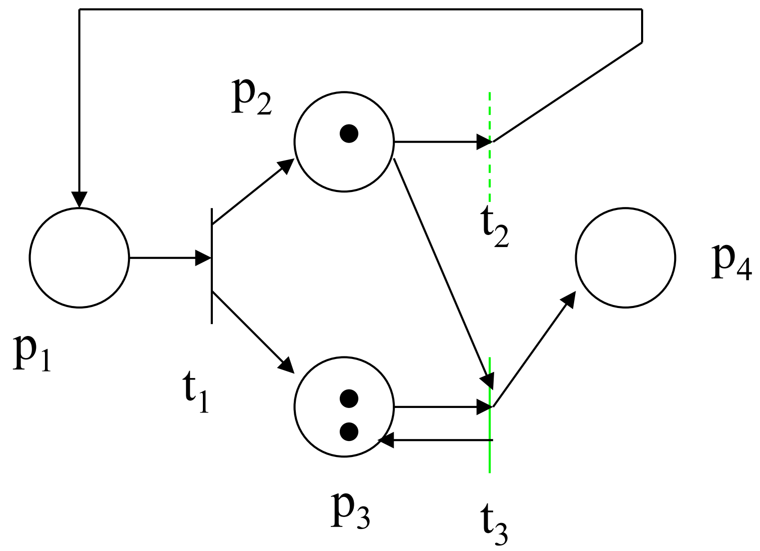
$$\mathbf{x} = \begin{bmatrix} 0 & 1 & 1 & 0 \end{bmatrix}$$

MARKING, DYNAMICS AND STATE SPACE



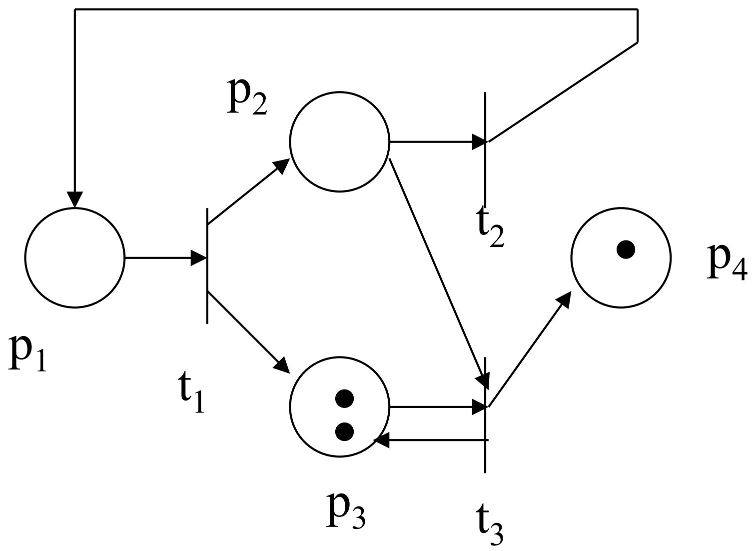
$$\mathbf{x} = \begin{bmatrix} 1 & 0 & 1 & 0 \end{bmatrix}$$

MARKING, DYNAMICS AND STATE SPACE



$$\mathbf{x} = \begin{bmatrix} 0 & 1 & 2 & 0 \end{bmatrix}$$

MARKING, DYNAMICS AND STATE SPACE



$$\mathbf{x} = \begin{bmatrix} 0 & 0 & 2 & 1 \end{bmatrix}$$

REACHABLE STATES AND STATE EQUATIONS

Def. (Extended State Transition Function):

$$f : \mathrm{N}^n \times T^* \rightarrow \mathrm{N}^n$$

$$f(\mathbf{x}, \varepsilon) := \mathbf{x}$$

$$f(\mathbf{x}, st) := f(f(\mathbf{x}, s), t), \; t \in T, s \in T^*$$

Def. (Reachable states): the set of *reachable states* of PN $(P, T, A, w, \mathbf{x})$ is $R[(P, T, A, w, \mathbf{x})] := \{\mathbf{y} \in \mathrm{N}^n : \exists s \in T^* (f(\mathbf{x}, s) = \mathbf{y})\}$

State Equation

$$\mathbf{x}' = \mathbf{x} + \mathbf{u}\mathbf{A}$$

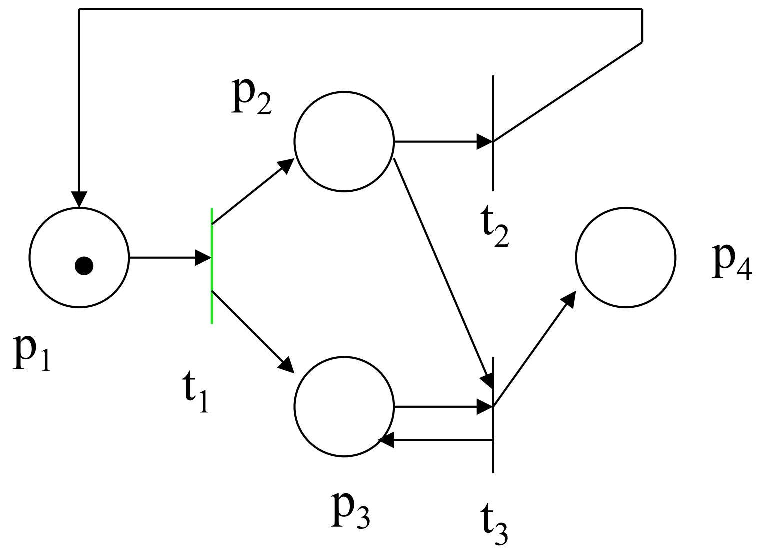
mxn **matrix whose (*j,i*) entry is**

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

***m*-dimensional *firing vector* [0 ... 0 1 0 ... 0]**

***j**th* **position**

INCIDENCE MATRIX **A**



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

$$[1 \quad 0 \quad 0 \quad 0] \xrightarrow{t_1} [0 \quad 1 \quad 1 \quad 0]$$

$$[0 \quad 1 \quad 1 \quad 0] = [1 \quad 0 \quad 0 \quad 0] + [1 \quad 0 \quad 0]\mathbf{A}$$

PETRI NET LANGUAGES

Def. (Labeled Petri net): A *labeled Petri net N* is an eight-tuple

$$N = (P, T, A, w, E, l, \mathbf{x}_0, \mathbf{X}_m)$$

where

(P, T, A, w) is a PN graph

E is the event set for transition labeling

$l : T \rightarrow E$ is the transition labeling function

$\mathbf{x}_0 \in \mathbf{N}^n$ is the initial state

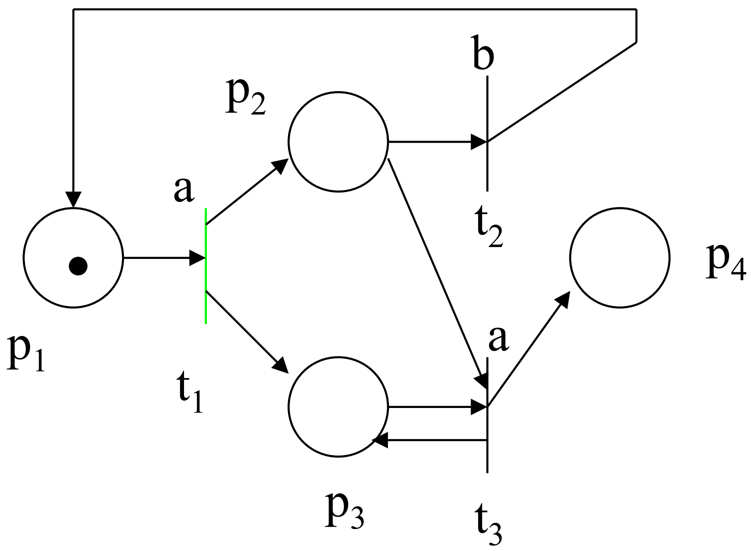
$\mathbf{X}_m \subseteq \mathbf{N}^n$ is the set of *marked states*

Def. (Languages generated and marked):

$$L(N) := \{l(s) \in E^* : s \in T^* \text{and } f(\mathbf{x}_0, s) \text{ is defined}\}$$

$$L_m(N) := \{l(s) \in L(N) : s \in T^* \text{and } f(\mathbf{x}_0, s) \in \mathbf{X}_m\}$$

PETRI NET LANGUAGES



$E = \{a, b\}$

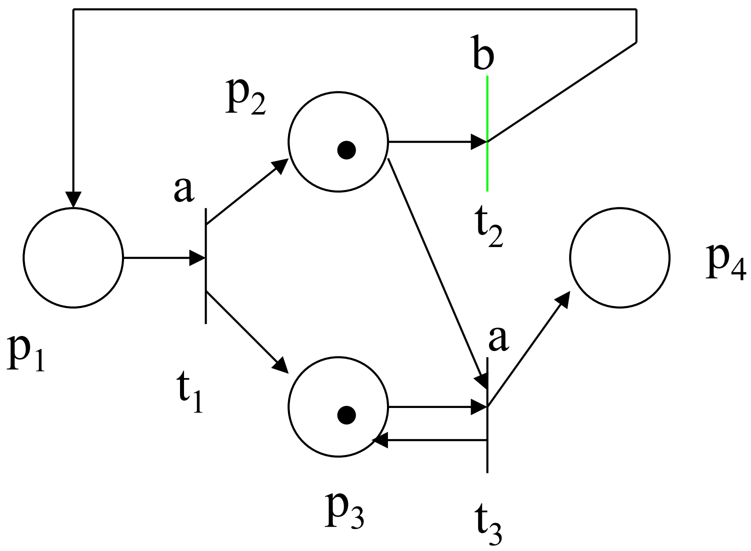
$l(t_1) = a, l(t_2) = b, l(t_3) = a$

$\mathbf{x}_0 = [1 \quad 0 \quad 0 \quad 0]$

$\mathbf{X}_m = \{[0 \quad 0 \quad k \quad 1], k > 0\}$

Generated string: ε in L

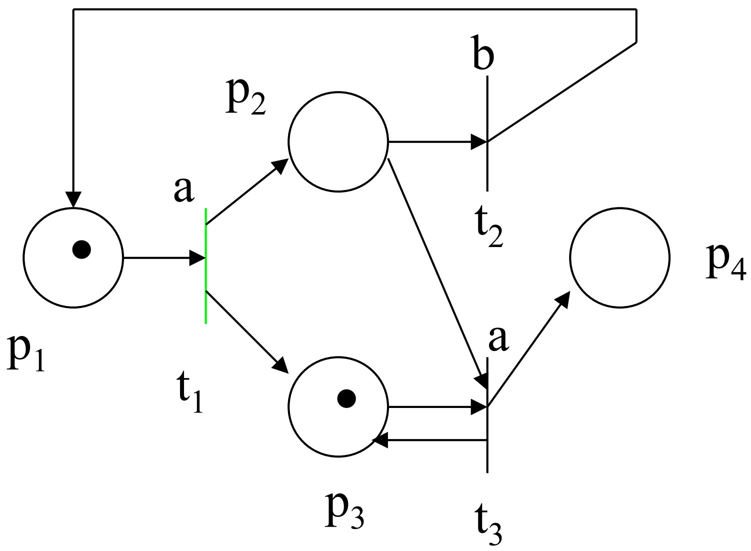
PETRI NET LANGUAGES



$$E = \{a, b\}$$
$$l(t_1) = a, l(t_2) = b, l(t_3) = a$$
$$\mathbf{x}_0 = \begin{bmatrix} 1 & 0 & 0 & 0 \end{bmatrix}$$
$$\mathbf{X}_m = \{\begin{bmatrix} 0 & 0 & k & 1 \end{bmatrix}, k > 0\}$$

Generated string: a in L

PETRI NET LANGUAGES



$E = \{a, b\}$

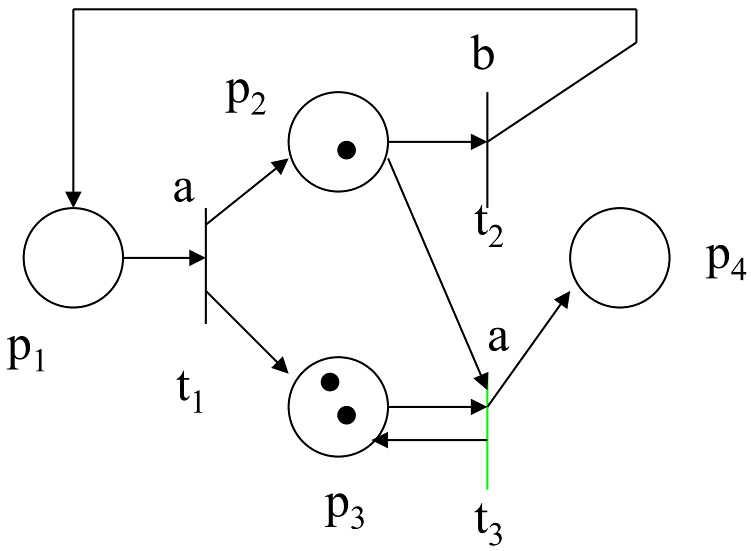
$l(t_1) = a, l(t_2) = b, l(t_3) = a$

$\mathbf{x}_0 = \begin{bmatrix} 1 & 0 & 0 & 0 \end{bmatrix}$

$\mathbf{X}_m = \{\begin{bmatrix} 0 & 0 & k & 1 \end{bmatrix}, k > 0\}$

Generated string: ab in L

PETRI NET LANGUAGES



$E = \{a, b\}$

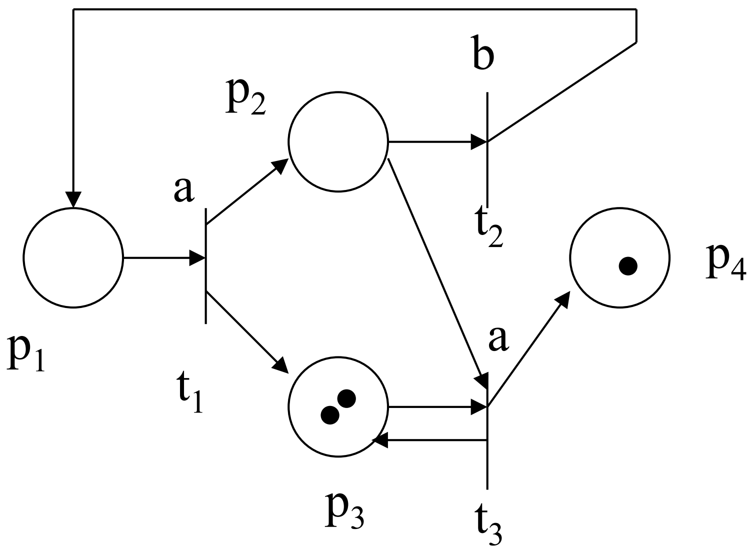
$l(t_1) = a, l(t_2) = b, l(t_3) = a$

$\mathbf{x}_0 = [1 \quad 0 \quad 0 \quad 0]$

$\mathbf{X}_m = \{[0 \quad 0 \quad k \quad 1], k > 0\}$

Generated string: aba in L

PETRI NET LANGUAGES



$E = \{a, b\}$

$l(t_1) = a, l(t_2) = b, l(t_3) = a$

$\mathbf{x}_0 = [1 \quad 0 \quad 0 \quad 0]$

$\mathbf{X}_m = \{[0 \quad 0 \quad k \quad 1], k > 0\}$

Generated string: abaa in L and L_m

$L_m(N)=\{(ab)^n a^2, n \geq 0\}$

$L(N)=\{a, aa, ab, (ab)^n, (ab)^n a, (ab)^n a^2, n \geq 0\}$

COMPARISON WITH AUTOMATA

- In PNs, the state information is distributed among a set of places which capture key conditions governing the system
- An automaton can always be represented as a PN, but not all PNs can be represented as *finite-state* automata (if the reachability set is finite, the PN can be represented as a FSA) – therefore the language expressive power is greater for PNs than for automata
- PNs have increased modularity for model-building
- All questions such as “is state *x* reachable?” are *decidable* for automata, but many of them are not for PNs

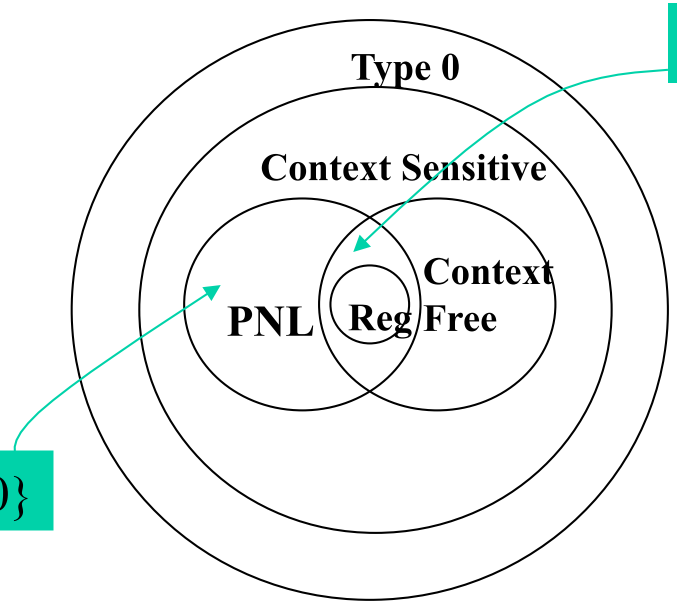
COMPARISON WITH AUTOMATA

$$PNL = \left\{ K \subseteq E^* : \exists N = (P, T, A, w, E, l, \mathbf{x_0}, \mathbf{X}_m)[L_m(N) = K] \right\}$$

Petri Net Languages (PNL) include Regular Languages (Reg)

Therefore the expressive power of PNs to describe DEDS behaviors is greater than that of FSA.

$L_m(N)=\{a^n b^n, n \geq 0\}$



$L_m(N)=\{a^n b^n \mathbf{c}^n, n \geq 0\}$

ANALYSIS PROBLEMS

Def. (Boundedness): Place $p_i \in P$ in PN N with initial state $\mathbf{x_0}$ is said to be *k-bounded*, or *k-safe*, if $x(p_i) \leq k$ for all states $\mathbf{x} \in R(N)$, i.e., for all reachable states.

Def. (State Coverability): Given a PN N with initial state $\mathbf{x_0}$, state $\mathbf{y}$ is said to be *coverable*, if there exists $\mathbf{x} \in R(N)$ such that $x(p_i) \geq y(p_i)$ for all $i=1,\ldots,n$.

Def. (Conservation): A PN N with initial state $\mathbf{x_0}$ is said to be *conservative with respect to* $\gamma = [\gamma_1, \gamma_2, \ldots, \gamma_n]$ $\gamma \in N^n$ if

$$\sum_{i=1}^{n} \gamma_i x(p_i) = \text{constant}$$

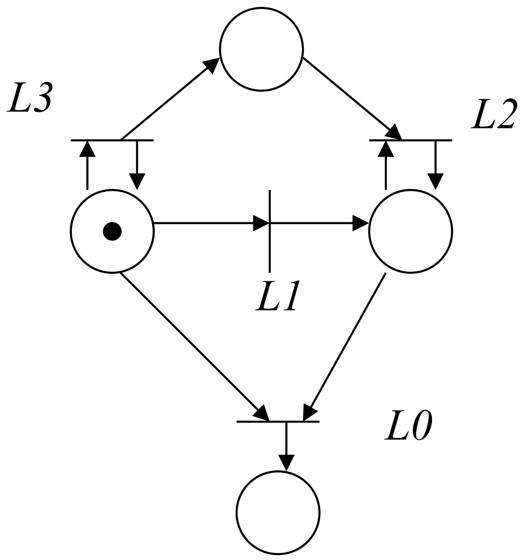
for all reachable states.

ANALYSIS PROBLEMS

Def. (Liveness): A PN *N* with initial state $\mathbf{x_0}$ is said to be *live* if there always exists some sample path such that any transition can eventually fire from any state reached from $\mathbf{x_0}$.

Liveness levels - a transition in a PN may be:

- *Dead or L0-live*, if the transition can never fire from this state
- *L1-live*, if there is some firing sequence from $\mathbf{x}_0$ such that the transition can fire at least once
- *L2-live*, if the transition can fire at least *k* times for some given positive integer *k*
- *L3-live*, if there exists some infinite firing sequence in which the transition appears infinitely often
- *L4-live*, if the transition is L1-live for every possible state reached from $\mathbf{x}_0$



ANALYSIS PROBLEMS

If a Petri Net is bounded, state safety and blocking properties can determined algorithmically – we just have to build an equivalent FSA.

It is possible to identify dead transitions by checking for coverability.

There are many other analysis problems (e.g., finding T-invariants, P-invariants, persistence)

Boundedness has to do with *stability* (the number of required resources does not explode.)

ANALYSIS PROBLEMS
The Coverability Tree

Root node: first node of the tree, corresponding to the initial state of a given marked PN.
Terminal node: any node from which no transition can fire.
Duplicate node: node identical to a node already in the tree.
Node dominance: let **x** and **y** be two states, i.e., nodes in the coverability tree. We say that "**x** dominates **y**", denoted by $\mathbf{x} >_d \mathbf{y}$, if the following two conditions hold:

(a) $x(p_i) \geq y(p_i)$, for all i=1,...,n

(b) $x(p_i) > y(p_i)$, for at least some i=1,...,n

Symbol ω: may be thought of as "infinity" in representing the *marking* of an unbounded place. It is used when a node dominance relationship is identified in the coverability tree. In particular, if $\mathbf{x} >_d \mathbf{y}$, then for all *i* such that $x(p_i) > y(p_i)$, the value of $x(p_i)$ is replaced by ω. Note that $\omega + k = \omega$.

Ex.: $[1\ 0\ 1\ 0] >_d [1\ 0\ 0\ 0] \Rightarrow [1\ 0\ 1\ 0]$ is replaced by $[1\ 0\ \omega\ 0]$.

ANALYSIS PROBLEMS
The Coverability Tree Algorithm

Step 1: Initialize $\mathbf{x} = \mathbf{x}_0$ (initial state)

Step 2: For each new node $\mathbf{x}$ evaluate the transition function $f(\mathbf{x},t_j)$ for all $t_j \in T$:

Step 2.1: If $f(\mathbf{x},t_j)$ is undefined for all $t_j \in T$ (i.e., no transition is enabled at state $\mathbf{x}$), then $\mathbf{x}$ is a *terminal node*.

Step 2.2: If $f(\mathbf{x},t_j)$ is defined for some $t_j \in T$, create a new node $\mathbf{x'} = f(\mathbf{x},t_j)$. Then:

Step 2.2.1: If $x(p_i) = \omega$ for some p_i, set $x'(p_i) = \omega$.

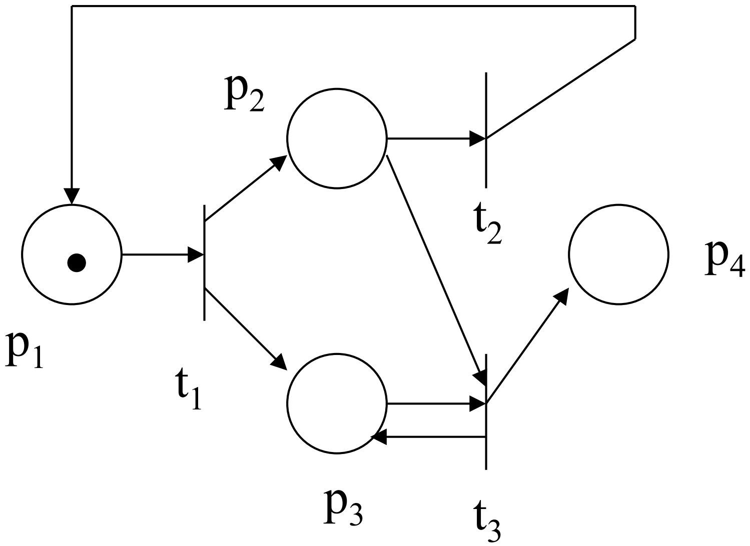
Step 2.2.2: If there exists a node $\mathbf{y}$ in the path from the root node $\mathbf{x}_0$ (included) to $\mathbf{x'}$ such that $\mathbf{x'} >_d \mathbf{y}$, set $x'(p_i) = \omega$ for all p_i such that $x'(p_i) > y(p_i)$.

Step 2.2.3: Otherwise, set $\mathbf{x'}(p_i) = f(\mathbf{x},t_j)$.

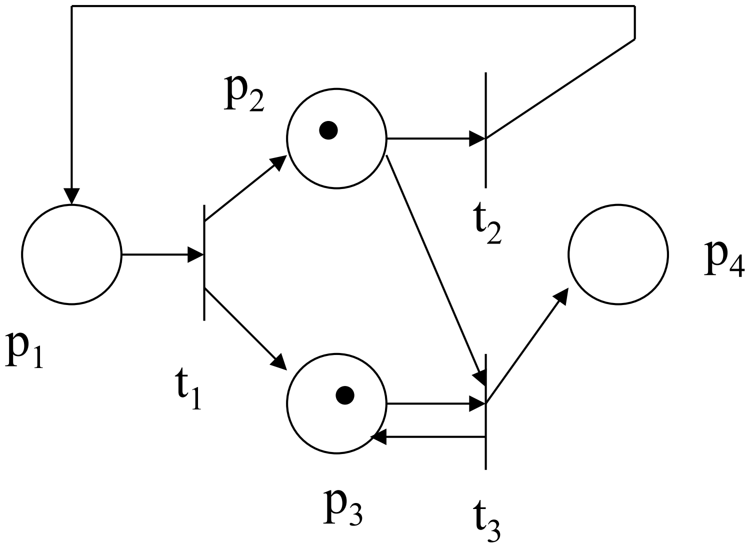
Step 3: If all new nodes are either *terminal* or *duplicate nodes*, stop. Otherwise go back to Step 2.

ANALYSIS PROBLEMS
The Coverability Tree

$$\begin{bmatrix} 1 & 0 & 0 & 0 \end{bmatrix}$$



ANALYSIS PROBLEMS
The Coverability Tree

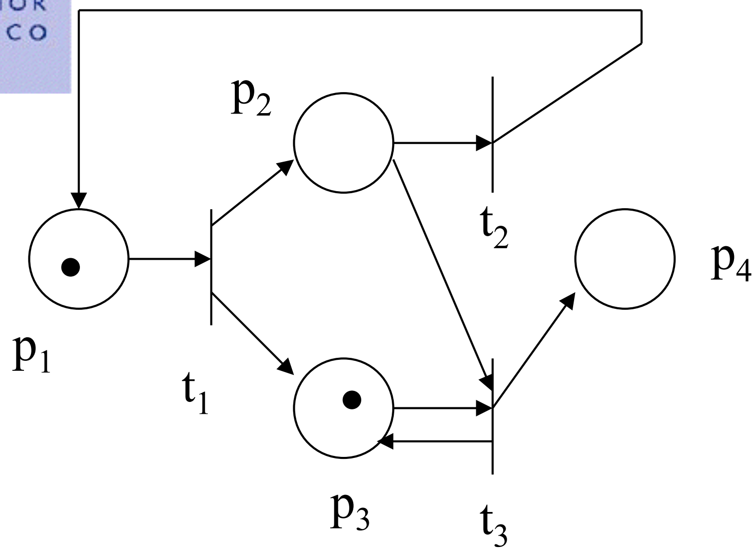


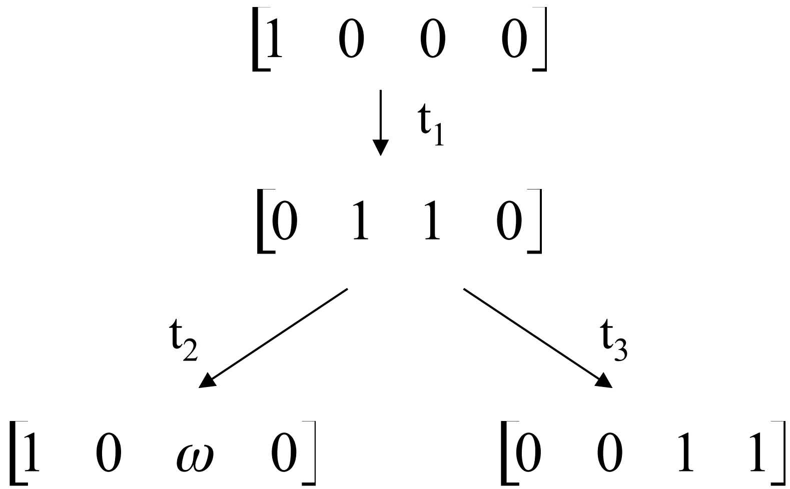
$$\begin{bmatrix}1 & 0 & 0 & 0\end{bmatrix}$$

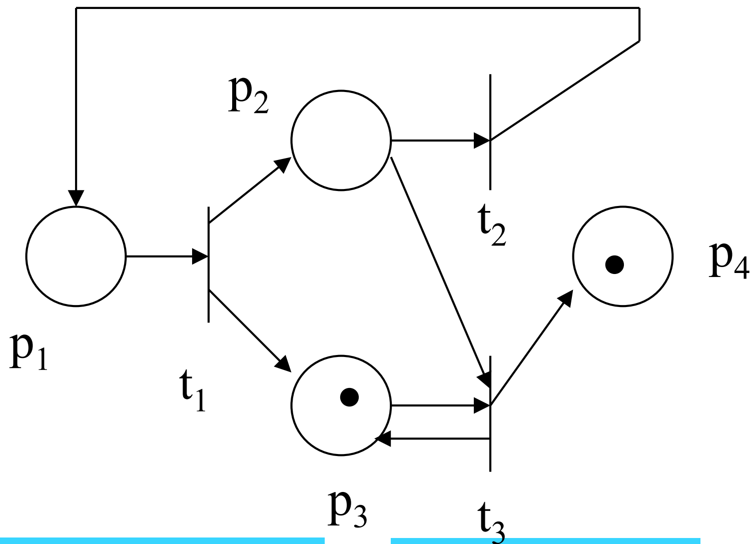
$$\downarrow t_1$$

$$\begin{bmatrix}0 & 1 & 1 & 0\end{bmatrix}$$

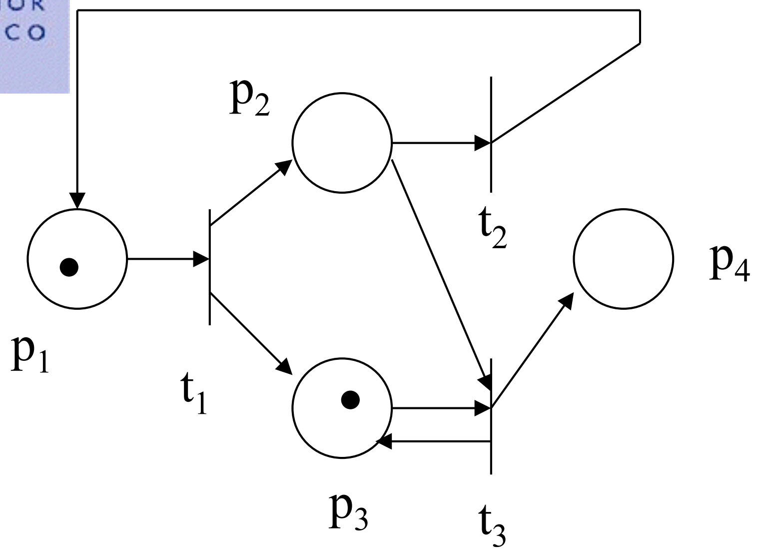
ANALYSIS PROBLEMS
The Coverability Tree

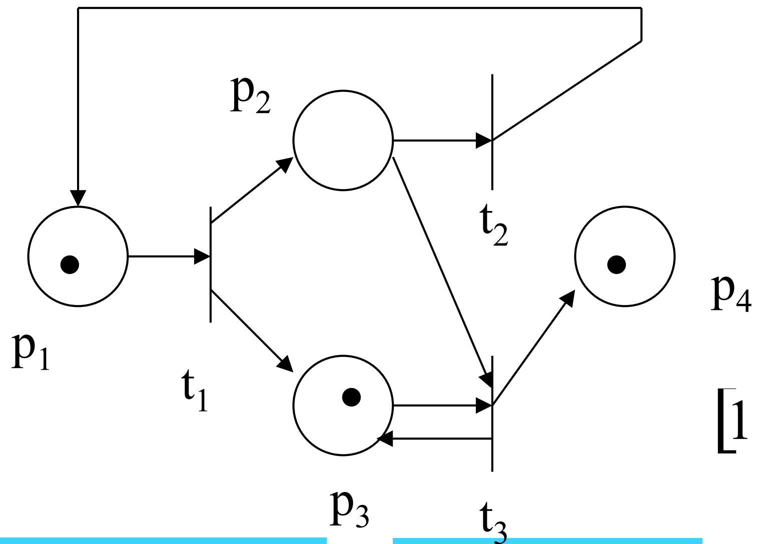


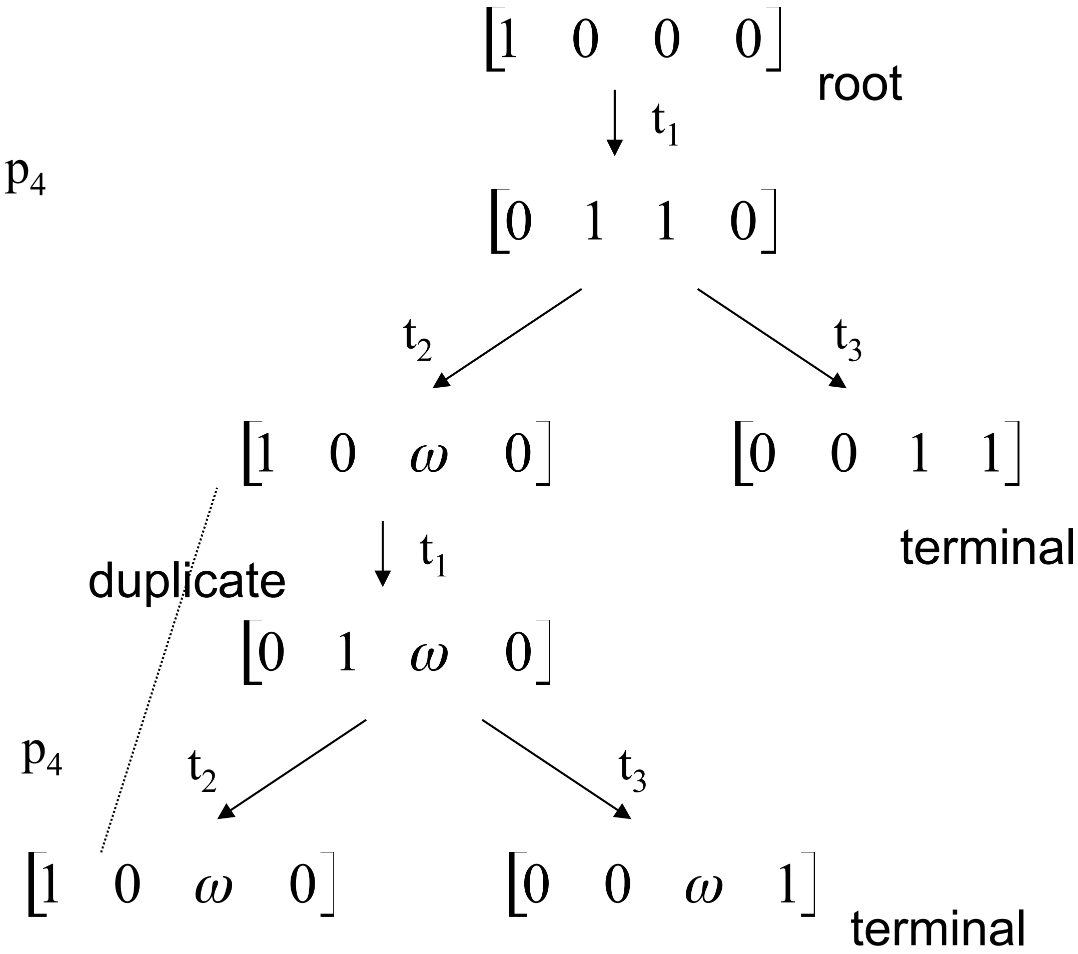




ANALYSIS PROBLEMS
The Coverability Tree







ANALYSIS PROBLEMS

Applications and Limitations of the the Coverability Tree

- The coverability tree is always finite.
- A PN is **bounded** iff the symbol ω never appears in the coverability tree.

If ω does not appear, the state space of the PN is finite.

- **Coverability** can be determined using the coverability tree.
- **Conservation** is checked by solving *r* equations of the form

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

with $n+1$ unknowns (the *n* weights plus *C*), where *r* is the number of nodes of the coverability tree. If ω appears in the coverability tree for some place, the corresponding γ must be zero.

- **Reachability** of a specific state can not be checked, when the ω symbol appears in the coverability tree, because it may represent different integer values.
- **Liveness** of transitions can not, in the general case, be determined by this technique as well

State Equation

For the extended version of the transition function

$$\mathbf{x}' = \mathbf{x} + \mathbf{v}\mathbf{A}$$

mxn **matrix whose (*j,i*) entry is**

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

m*-dimensional *firing counting vector

$$\begin{pmatrix} n_{t_1} & n_{t_2} & \ldots & n_{t_m} \end{pmatrix}$$

Number of times transition t2 fires in the sequence

ANALYSIS PROBLEMS
Linear-Algebraic Techniques

A *necessary* condition for state $\mathbf{x}$ to be reachable from initial state $\mathbf{x_0}$ is for the equation

firing count vector ← $\mathbf{v}$

$$\mathbf{vA} = \mathbf{x} - \mathbf{x}_0$$

to have a solution $\mathbf{v}$ where all the entries of $\mathbf{v}$ are non-negative integers.

The existence of a non-negative integer solution $\mathbf{v}$ does **not** guarantee that the entries in $\mathbf{v}$ can be mapped to an actual feasible ordering of transition firings.

ANALYSIS PROBLEMS
Linear-Algebraic Techniques

$$\mathbf{vA} = \mathbf{x} - \mathbf{x}_0 \quad (*)$$

Particular cases:

- if (*) has no solution OR
  has a solution with negative **v** elements OR
  has a solution with non-integer **v** elements

Then **x** is *not* reachable from $\mathbf{x}_0$

- if (*) has a solution **v** with non-negative elements

Then *there may exist* a transition sequence leading from $\mathbf{x}_0$ to **x**, *but that is not guaranteed*.

Multiple solutions of (*) are possible

Whenever there may exist a solution, at least the number of alternatives to be checked is significantly reduced.

Conservation can be checked by solving

$$\mathbf{A}\gamma^{T} = 0 \quad (**)$$

NOTE: compare with the coverability tree technique. **There x_0 matters**! Here, all possible x_0 are implicitly checked. If there exists a single one that violates conservation, there is no solution for (**).

- More powerful results can be obtained based on this technique for sub-classes of PNs, such as **marked graphs**.

PETRI NET Sub-Classes

Petri net sub-classes, with smaller modeling capability, but larger decision power:

- **State Machines**: Each transition must have exactly *one* input place and *one* output place

$$(\forall_{t_j \in T}, |I(t_j)| = 1 \wedge |O(t_j)| = 1).$$

- **Marked Graphs**: Each place is the input of exactly *one* transition and the output of exactly *one* transition

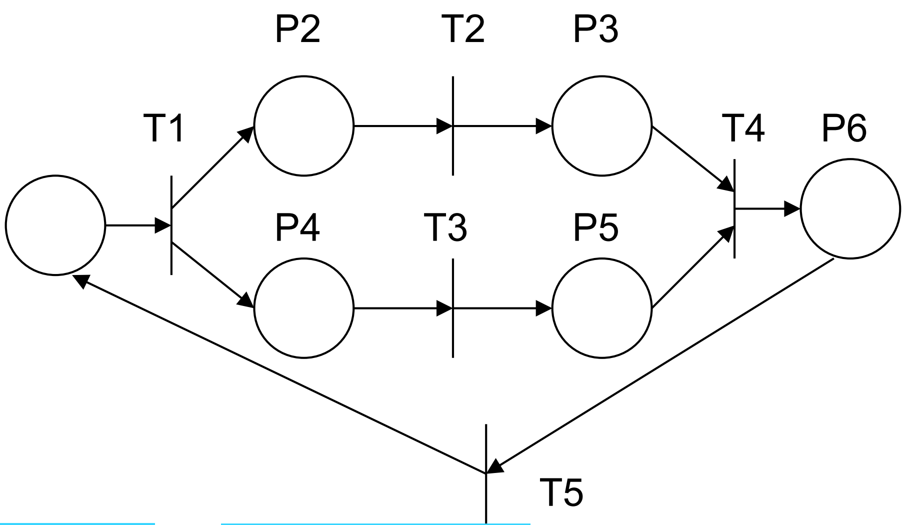
$$(\forall_{p_i \in P}, |I(p_i)| = 1 \wedge |O(p_i)| = 1).$$

- **Free-Choice PNs**: If a place is input of *more than one* transition (potential conflict), then it is the *single* input place of each of those transitions

$$\forall_{t_j \in T} \forall_{p_i \in I(t_j)}, \text{ either } I(t_j) = \{p_i\} \text{ or } O(p_i) = \{t_j\} .$$

Place Invariants correspond to sets of places whose weighted token count remains constant for all possible markings, i.e., every integer vector which satisfies

$$\mathbf{A}\gamma^{\mathrm{T}} = \mathbf{0}$$



Example

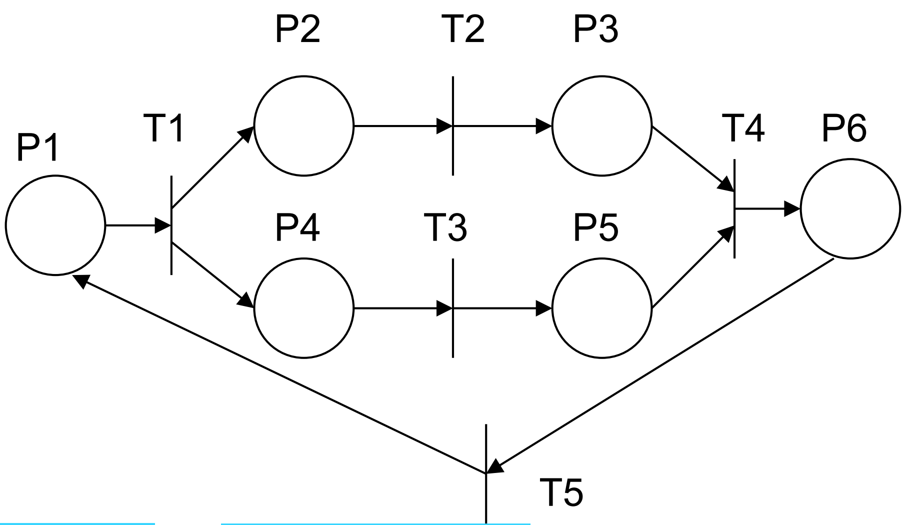
$\gamma = [2 \quad 1 \quad 1 \quad 1 \quad 1 \quad 2] \rightarrow P_1, P_2, P_3, P_4, P_5, P_6$

is aP-invariant fo this net

ANALYSIS PROBLEMS
Linear-Algebraic Techniques

Transition Invariants correspond to sets of *transition firings* that cause the marking of a net to cycle, i.e., $\mathbf{x} = \mathbf{x_0}$ after some *N* firings

$$v\mathbf{A} = \mathbf{0}$$

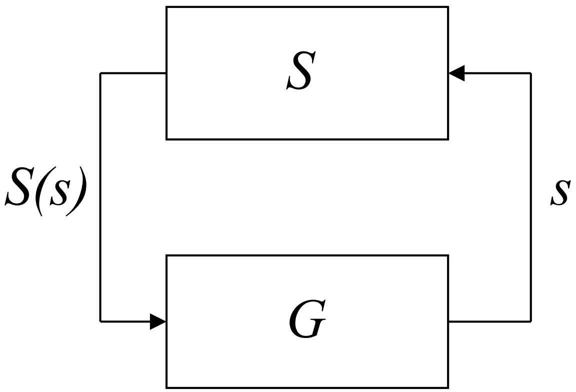


Example

$$v = \begin{bmatrix} 1 & 1 & 1 & 1 & 1 \end{bmatrix} \rightarrow t_1, t_2, t_3, t_4, t_5$$

is a T-invariant fo this net

CONTROL OF PETRI NETS



- $L_r \subseteq L(S/G) \subseteq L_a$
- If we wish to restrict the behavior of G, but not more than necessary:

$$L(S/G) = L_a^{\uparrow C}$$

largest sublanguage of L_a which is controllable

- If we wish to restrict the behavior of G as much as possible:

$$L(S/G) = L_r^{\downarrow C}$$

smallest superlanguage of L_r which is controllable

- Regular languages are closed under most supervisor synthesis operations, i.e., if L_a and L_r are regular, so will be $L_a^{\uparrow C}$ and $L_r^{\downarrow C}$.

CONTROL OF PETRI NETS

- If $L_m(G)$ is not regular (e.g., G is an unbounded PN) but L_{am} is *regular* and *controllable*, S can be realized by a finite-state deterministic automaton (FSA).
- If $L_m(G)$ is regular but L_{am} is *controllable* but *not regular*, S can not be realized by an FSA, but one may be able to realize it by a PN – *see next example*

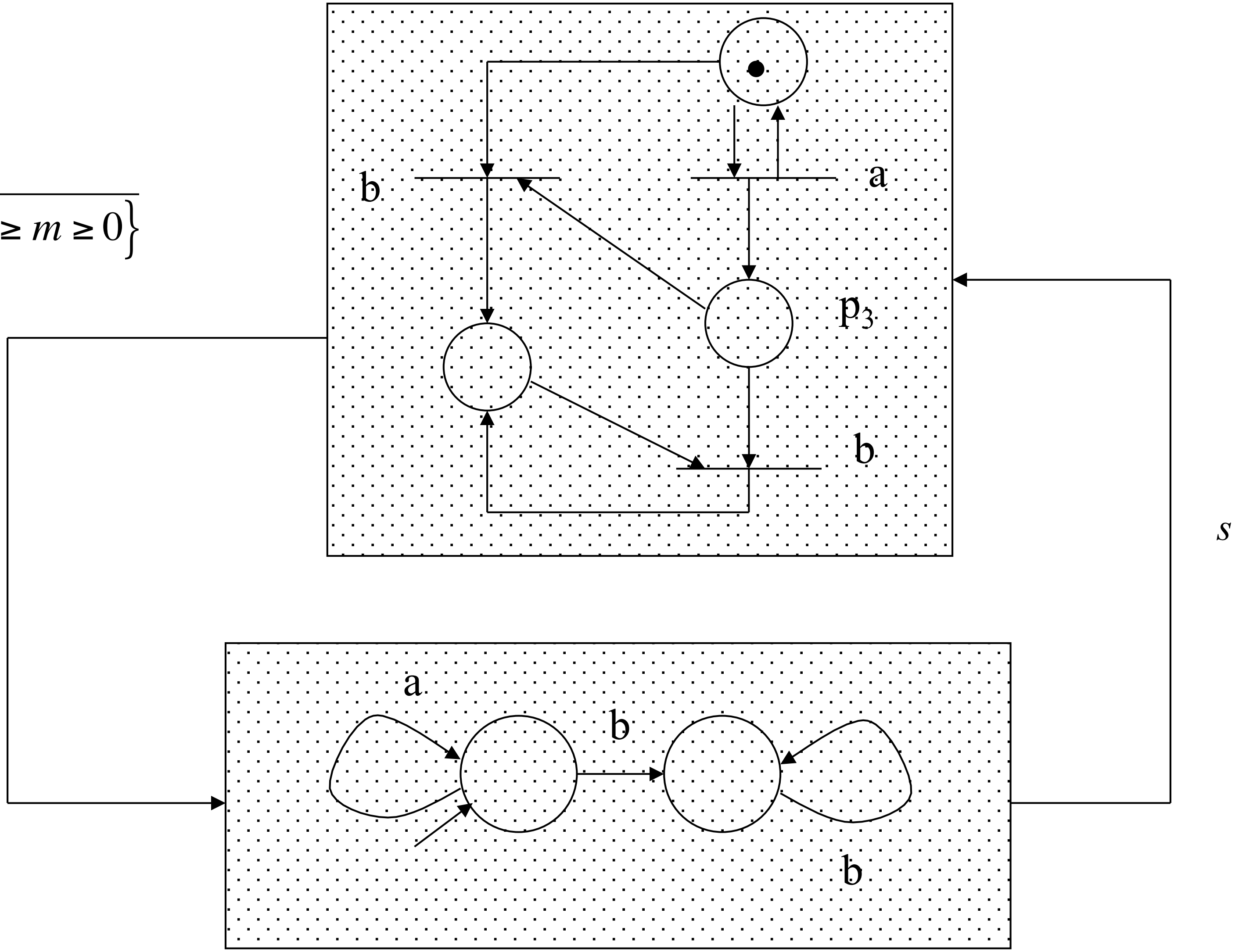
An Example

Example:

$$L(G) = a^* b^*$$

$$L_a = \overline{\{a^n b^m : n \geq m \geq 0\}}$$

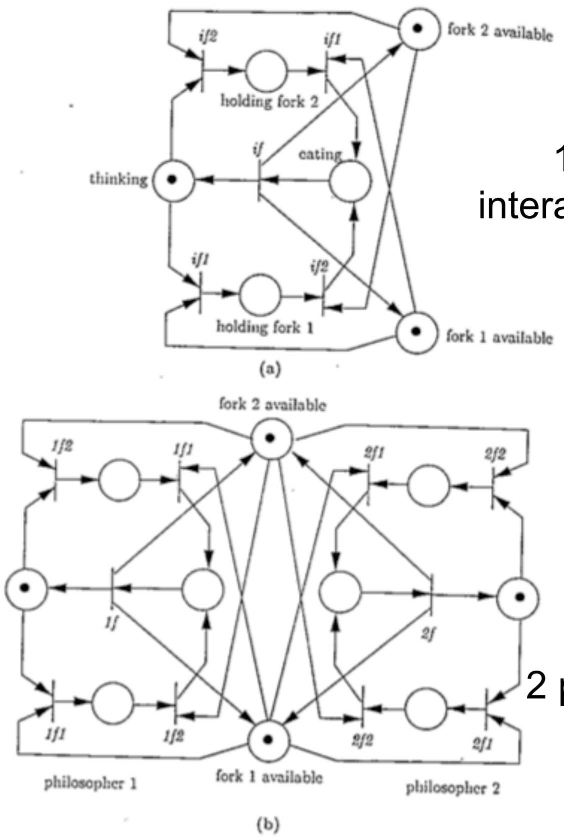
$$S(s) = \begin{cases} \{a, b\} & \text{if } x(p_3) > 0 \\ \{a\} & \text{if } x(p_3) = 0 \end{cases}$$



CONTROL OF PETRI NETS
An Example

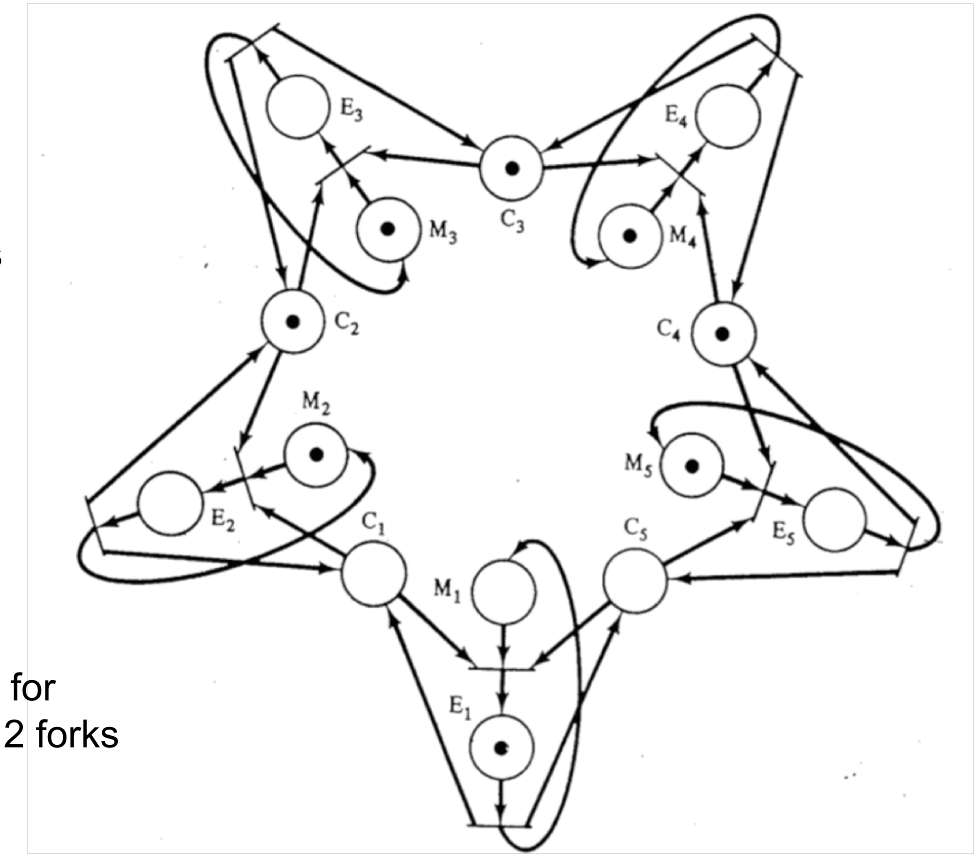
INSTITUTO SUPERIOR TÉCNICO

Dining Philosophers revisited



1 philosopher
interacting with 2 forks

complete model for
2 philosophers and 2 forks



2 philosophers / possible deadlock

5 philosophers / no deadlock

CONTROL OF PETRI NETS

- If both $L_m(G)$ and L_{am} are *not regular*, but we wish them to be PN languages, several problems may occur, including non-closure under the ↑C and ↓C operations – solvable using *inhibitor* arcs (at the expense of reduced analysis capabilities).

- If both $L_m(G)$ and L_{am} are *regular*, using PN models may lead to more compact representations than FSA.

- If specifications are made in terms of *forbidden states*, rather than *admissible languages*, several results exist for supervisor synthesis, mostly based on *linear-algebraic techniques*.

STATE-BASED PN SUPERVISION

Building PN Supervisor for a PN Model
from Linear Predicates on the State Vectors

(Moody, Antsaklis, 1998)

Specification: restrict the reachable states $\mathbf{x}_p$ of a PN model, such that

$$\mathbf{L}\mathbf{x}_p \leq \mathbf{b}, \quad \mathbf{L}_{[n_c \times n]}, \mathbf{b}_{[n_c]}$$

n_c is the number of constraints

The inequality can be seen as the logical conjunction of n_c separate inequalities

The plant (e.g., robotic task, communication system) is modeled by a PN with incidence matrix $\mathbf{D}_p = \mathbf{A}_p$

The *controller net* is a PN with incidence matrix $\mathbf{D}_c$ made up of the plant transitions and a separate set of places

STATE-BASED PN SUPERVISION

(Moody, Antsaklis, 1998)

Introducing n_c slack variables to turn the inequality into an equality

$$\mathbf{Lx}_p + \mathbf{x}_c = \mathbf{b}, \quad \mathbf{x}_{c\,[n_c \times 1]}$$

The slack variables represent n_c new places that hold the extra tokens required to meet the equality, and are part of the separate *controller net*.

$$\mathbf{x}_{1\times n}\gamma_{1\times n}^{T} = \mathbf{x}_{0_{1\times n}}\gamma_{1\times n}^{T} \rightarrow \gamma_{n\times 1}^{T}\mathbf{x}_{n\times 1} = \gamma_{n\times 1}^{T}\mathbf{x}_{0_{n\times 1}}$$

$$\mathbf{A}_{m\times n}\gamma_{1\times n}^{T} = \mathbf{0} \rightarrow \gamma_{n\times 1}^{T}\mathbf{A}_{m\times n}^{T} = \gamma_{n\times 1}^{T}\mathbf{D}_{m\times n} = \mathbf{0}$$

n_c place invariants

$\mathbf{Lx}_p + \mathbf{x}_c = \mathbf{b}$ is in the form

$\Gamma^{T}\mathbf{X} = \Gamma^{T}\mathbf{X}_0, \quad \Gamma^{T} = [\mathbf{L}\ \mathbf{I}]$

provided that $\mathbf{Lx}_{p_0} + \mathbf{x}_{c_0} = \mathbf{b}$

$\therefore\ \Gamma^{T}\mathbf{D} = 0$

$$\Gamma^{T}\mathbf{D} = \begin{bmatrix}\mathbf{L} & \mathbf{I}\end{bmatrix}\begin{bmatrix}\mathbf{D_p} \\ \mathbf{D_c}\end{bmatrix} = \mathbf{0}$$

Controlled (closed loop) PN with incidence matrix $\mathbf{D}$ and state $\mathbf{x} = \begin{bmatrix}\mathbf{x}_p \\ \mathbf{x}_c\end{bmatrix}, \mathbf{x}_0 = \begin{bmatrix}\mathbf{x}_{p_0} \\ \mathbf{x}_{c_0}\end{bmatrix}$

if $\mathbf{b} - \mathbf{Lx}_{p_0} \geq 0$

then a PN controller with incidence matrix $\mathbf{D}_c$ and initial state $\mathbf{x}_{c_0}$

$$\mathbf{D}_c = -\mathbf{LD}_p$$

$$\mathbf{x}_{c_0} = \mathbf{b} - \mathbf{Lx}_{p_0}$$

enforces the constraint $\mathbf{Lx}_p \leq \mathbf{b}$ when included in the closed loop system

STATE-BASED PN SUPERVISION

(Moody, Antsaklis, 1998)

THEOREM – Invariant-Based Controller Synthesis

if $\mathbf{b} - \mathbf{L}\mathbf{x}_{p_0} \geq 0$

then a PN controller with incidence matrix $\mathbf{D}_c = A_c^T$ and initial state $\mathbf{x}_{c_0}$

$$\mathbf{D}_c = -\mathbf{L}\mathbf{D}_p$$

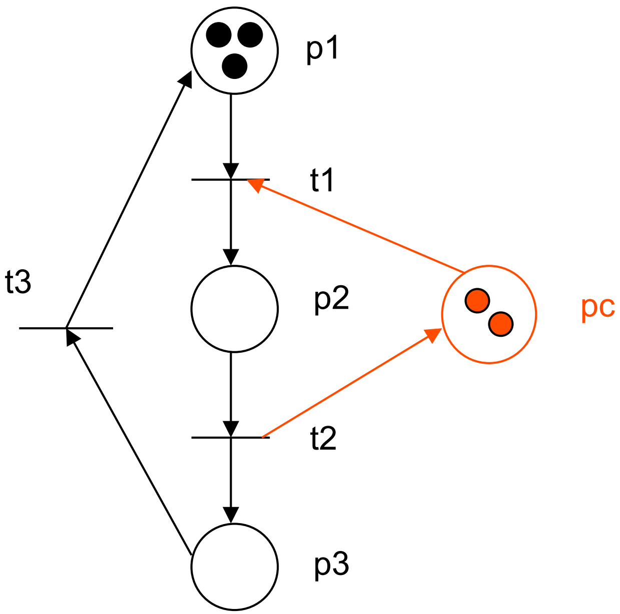
$$\mathbf{x}_{c_0} = \mathbf{b} - \mathbf{L}\mathbf{x}_{p_0}$$

enforces the constraint $\mathbf{L}\mathbf{x}_p \leq \mathbf{b}$ when included in the closed loop system with incidence matrix **D** = $\mathbf{A}^T$ and state $\mathbf{x} = \begin{bmatrix} \mathbf{x}_p \\ \mathbf{x}_c \end{bmatrix}, \mathbf{x}_0 = \begin{bmatrix} \mathbf{x}_{p_0} \\ \mathbf{x}_{c_0} \end{bmatrix}$

assuming that the transitions with input arcs from $\mathbf{D}_c$ are controllable. If the inequality $\mathbf{b} - \mathbf{L}\mathbf{x}_{p_0} \geq 0$ is not true, the constraints can not be enforced, since the initial conditions of the plant lie outside the range defined by the constraints.

STATE-BASED PN SUPERVISION

Example



$$\text{spec.}: x_2 \le 2 \qquad \mathbf{L} = \begin{bmatrix} 0 & 1 & 0 \end{bmatrix} \; \mathbf{b} = \mathbf{2}$$

$$\mathbf{D}_p = \begin{bmatrix} -1 & 0 & 1 \\ 1 & -1 & 0 \\ 0 & 1 & -1 \end{bmatrix}, \quad \mathbf{x}_{p_0} = \begin{bmatrix} 3 & 0 & 0 \end{bmatrix}^T$$

$$\mathbf{D}_c = -\mathbf{L}\mathbf{D}_p = \begin{bmatrix} -1 & 1 & 0 \end{bmatrix}$$

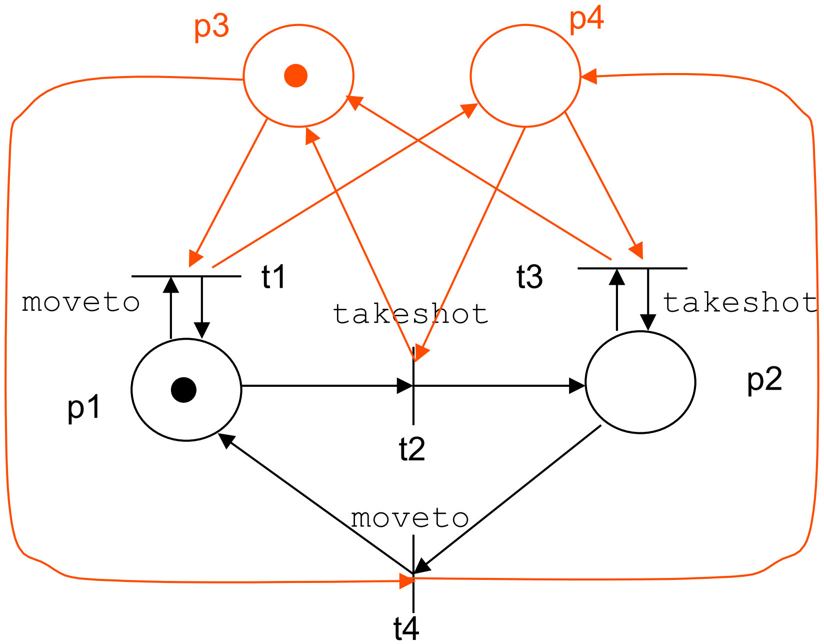
$$\mathbf{x}_{c_0} = \mathbf{b} - \mathbf{L}\mathbf{x}_{p_0} = 2$$

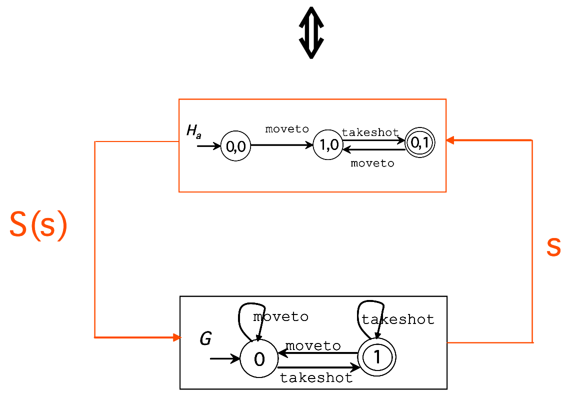
STATE-BASED PN SUPERVISION

Limitations

Specification: alternance of events `moveto` and `takeshot` (with *a* being the first event to occur) required

This controlled PN satisfies the specification, but the PN controller could not be synthesized using the Invariant-Based Controller method (*the controller places form their own invariant, separate from the plant PN places – controller is not maximally permissive*)



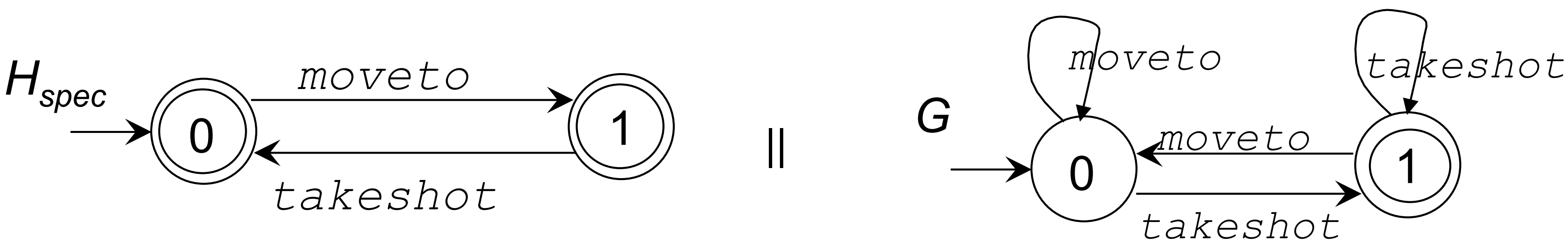


EVENT-BASED FSA SUPERVISION

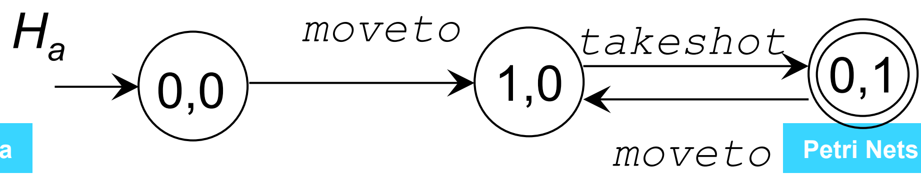
Building an FSA Supervisor for a FSA Model from Language Specifications

(Ramadge, Wonham, 1989)

Specification: alternance of events `moveto` and `takeshot` (with `moveto` being the first event to occur) required

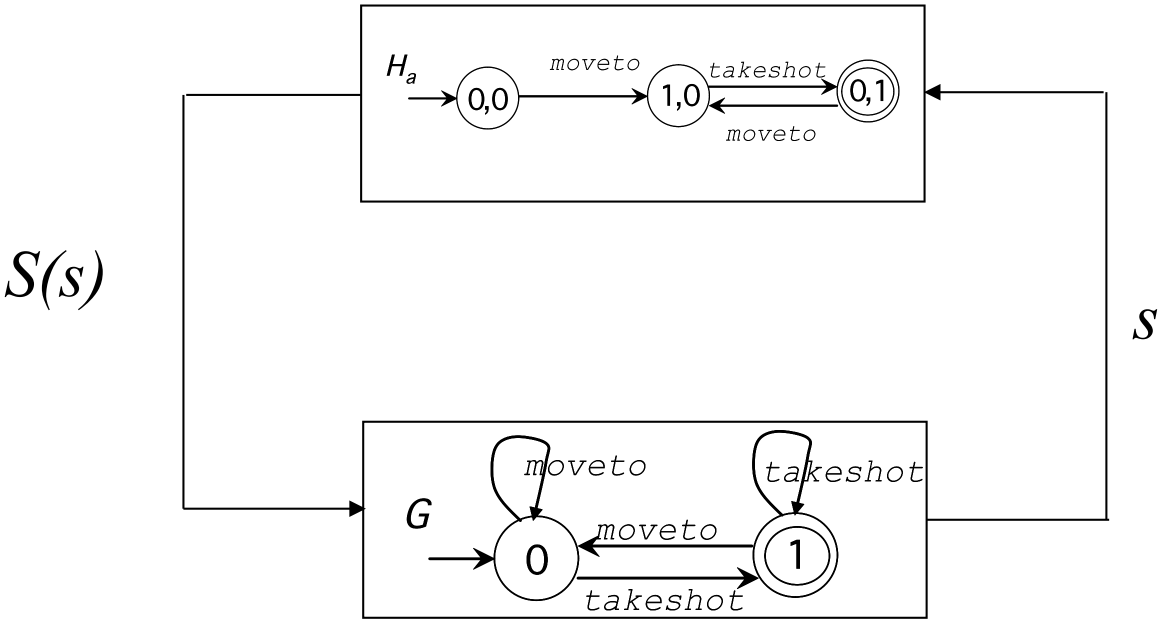


$$H_a = H_{spec} \,||\, G$$



EVENT-BASED FSA SUPERVISION

Building an FSA Supervisor for a FSA Model from Language Specifications



Ex.: *mt mt mt mt ts ts mt mt ts* ∈ L(G)
mt ts mt ts mt ts mt ts ∈ L(S/G)

FURTHER READINGS

- J. Peterson, *Petri Net Theory and the Modeling of Systems*, Prentice-Hall, 1981 - *more on modeling, languages, decidability and complexity issues*
- N. Viswanadham, Y. Narahari, *Performance Modeling of Automated Manufacturing Systems*,Prentice-Hall, 1992 - *more on modeling of manufacturing systems*
- J. O. Moody, P. J. Antsaklis, *Supervisory Control of Discrete Event Systems Using Petri Nets*, Kluwer Academic Publ., 1998 - *more on control of PNs*