4. Specificare, verificare, proiectare

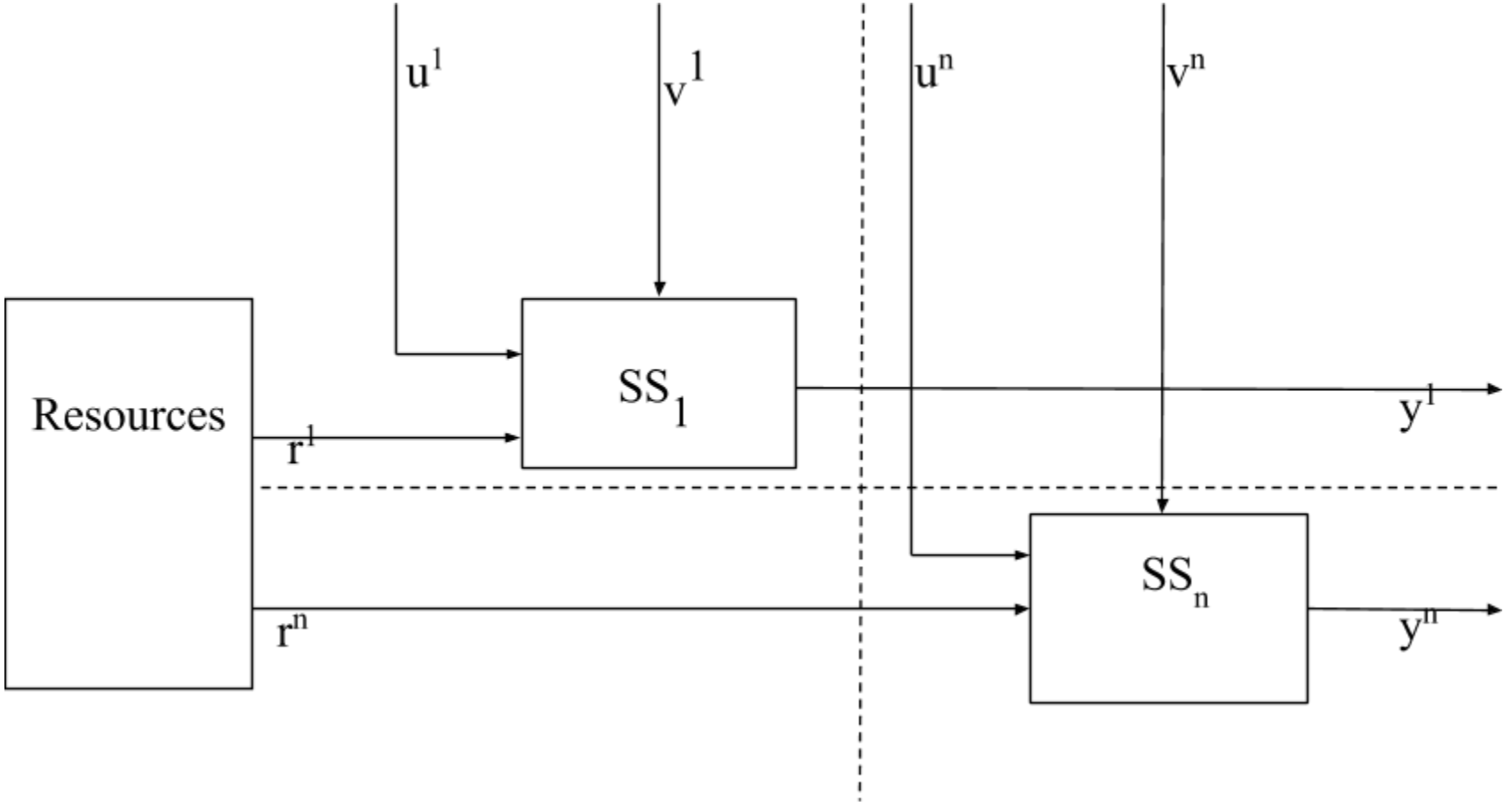
4.1. Sisteme de mari dimensiuni (Large Scale Systems)

- ***Sisteme continue*** *(continuous systems)*
- ***Sisteme cu timp discret*** *(discrete time systems)*
- ***Sisteme cu evenimente discrete*** *(discrete event systems)*
- ***Sisteme hibride - sisteme de control hibrid*** *(hybrid control systems)*
  → ***evenimente discrete + timp continuu***

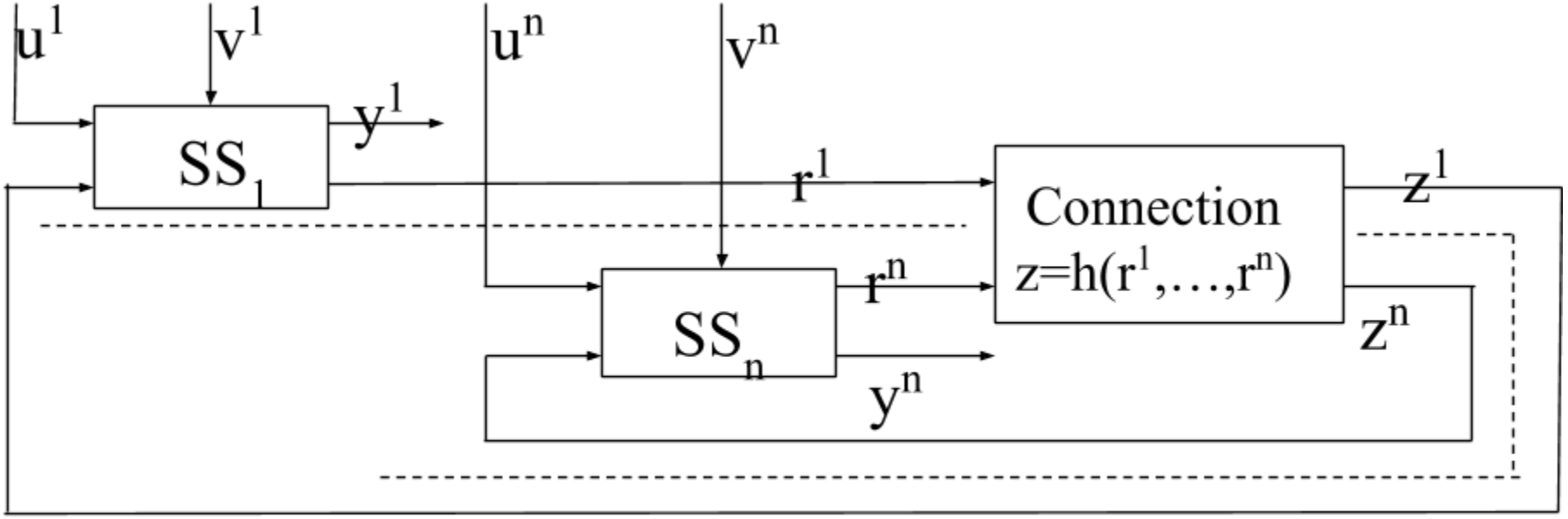
Definirea sistemelor mari:

- dimensiuni mari (stări, intrări, ieșiri)
- mai multe componente interconectate
- mai multe obiective (de multe ori vag definite sau conflictuale)
- constrângeri de structură a informației și acces la informație
- apariția evenimentelor și valorile pentru intrări/stări/ieșiri: nesigure

1) Sisteme interconectate prin resurse



2) Subsisteme interconectate rigid (sau direct)



3) Sisteme interconectate flexibile - interconectate prin buffere

$$\frac{dz}{dt} = h(z, r^1, \ldots, r^n)$$

sau

$$z(k+1) = h(z(k), r^1(k), \ldots, r^n(k)) .$$

4.2 Reţele Petri (recapitulare)

Justificarea reţelelor Petri:

- dezvoltarea sistemelor paralele și distribuite
- analiza cerințelor (requirements analysis)
- dezvoltarea specificațiilor, proiectarea, testarea
- simularea sistemelor
- analiza formală a comportamentului sistemelor critice

Definiţii

Definiţia 1.

Sunt perechi de 4 elemente:

$$N = (P, T, pre, post)$$

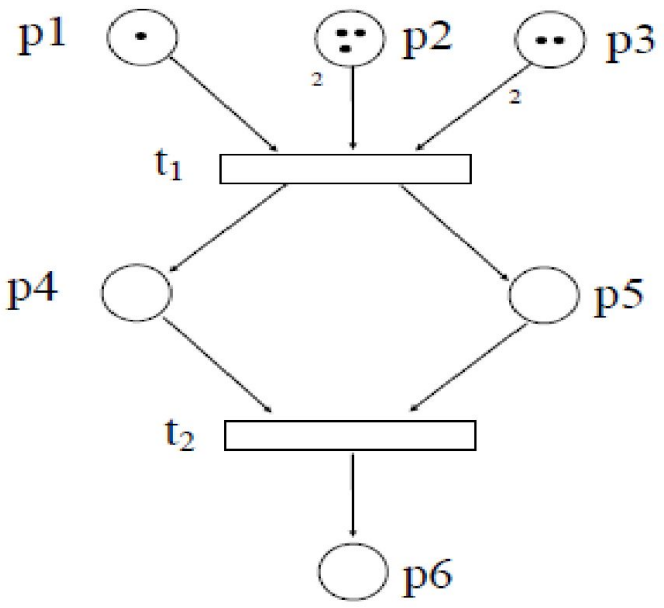
unde

$P = \{p_1, p_2, ..., p_m\}$ muţimea finită de locaţii;

$T = \{t_1, t_2, ..., t_n\}$ mulţimea finită de tranziţii

pre: P x T → **N** (mulţimea numerelor naturale) ***funcţia de incidenţă înapoi***:

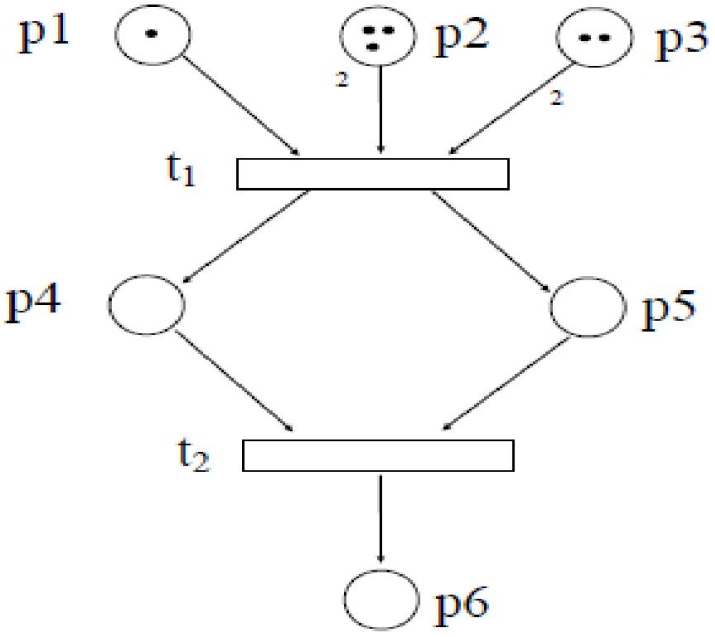
$$pre(p,t) = \begin{cases} 0, \text{ dacă nu există arc de la p la t} \\ \neq 0, \text{ dacă există arc de la p la t} \end{cases}$$



Ex.: Petri net

post: P x T → **N** (mulţimea numerelor naturale) ***funcţia de incidenţă înainte***:

$$post(t,p) = \begin{cases} 0, \text{ dacă nu există arc de la t la p} \\ \neq 0, \text{ dacă există arc de la t la p} \end{cases}$$



Ex.: Petri net

N = (P, T, *pre*, *post*) ← structura fără marcaj

PN = (N, $\mathbf{M_0}$) ← structura și marcajul

M : P → **N** (marcajul)

Marcajul descrie starea rețelei Petri

$\mathbf{M} = [\, M(p_1), M(p_2), \ldots, M(p_m)]^T$

Relaţii existente:

$$P \cap T = \Phi \quad \text{şi} \quad P \cup T \neq \Phi,$$

Constrângeri ale marcajului:

$$M(p) \leq K(p), \ \forall p \in P.$$

Definiţia 2.

N = (P, T, F, W)

P = $\{p_1, p_2, ..., p_m\}$ mulţimea finită de locaţii ;

T = $\{t_1, t_2, ..., t_n\}$ mulţimea finită de tranziţii;

F este mulţimea arcelor orientate.

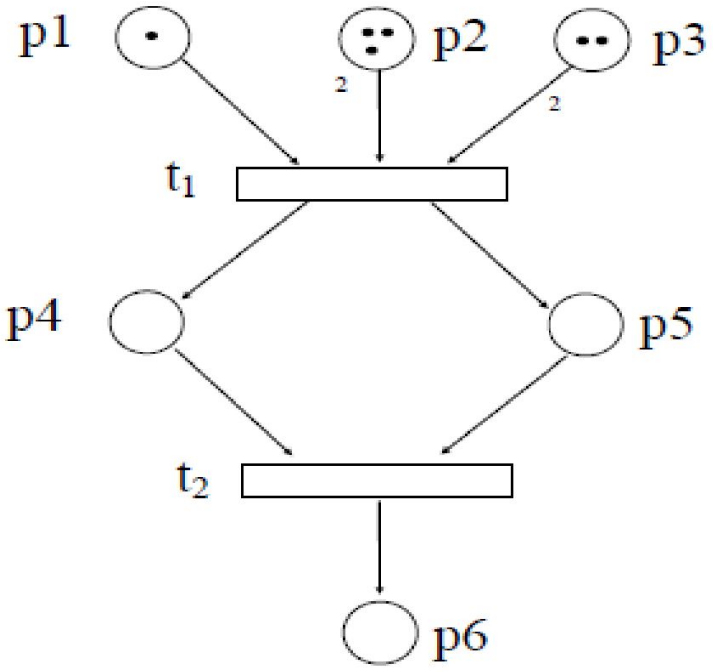
$F \subset (P \times T) \cup (T \times P)$,

W : F → **N**, (N – mulţimea numerelor naturale)

$W(p_i, t_j)$ sau $W(t_j, p_i)$ asignează ponderi la arce.

M : P → **N** este marcajul.

Marcajul descrie starea reţelei Petri.



Ex.: Petri net

Mulţimea locaţiilor de intrare pentru tranziţii:

Tranziţia t ➔ locaţii de intrare ${}^0t$

$${}^0t = \{p \mid p \in P, pre(p,t) \neq 0\}$$

or

$${}^0t = \{p \mid p \in P, \exists (p,t) \in F\}$$

Mulţimea locaţiilor de ieşire pentru tranziţii:

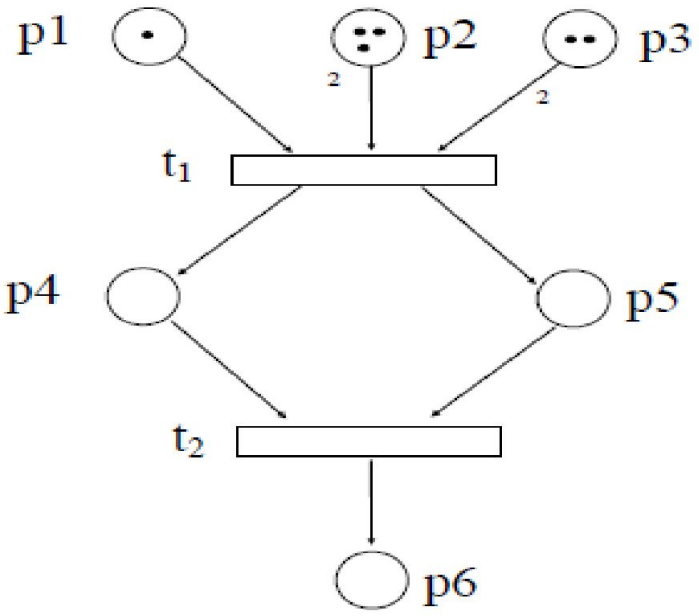
Tranziţia t ➔ locaţii de ieşire t^0:

$$t^0 = \{p \mid p \in P, post(t,p) \neq 0\}$$

sau

$$t^0 = \{p \mid p \in P, \exists (t,p) \in F\}$$

$$\mathbf{M} = [\, M(p_1), M(p_2), \ldots, M(p_m)]^T$$



Ex.: Petri net

Mulţimea tranziţiilor de intrare pentru locaţii:

Locaţia p ➔ tranziţii de intrare ${}^0p$

$${}^0p = \{ t \mid t \in T, post(t,p) \neq 0\}$$

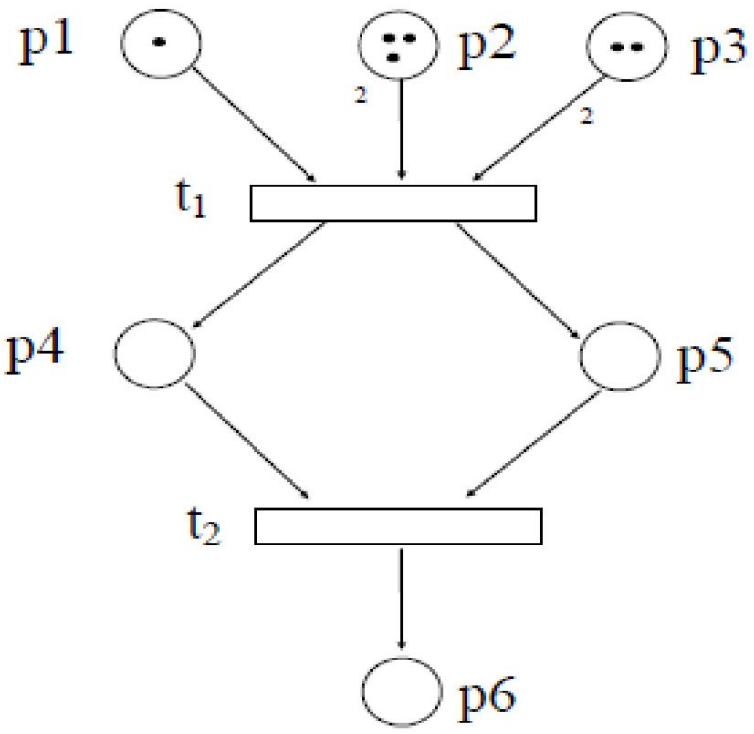
$${}^0p = \{ t \mid t \in T, \exists (t,p) \in F\}$$

Mulţimea tranziţiilor de intrare pentru locaţii

Locaţia p ➔ tranziţii de ieşire p^0:

$$p^0 = \{ t \mid t \in T, pre(p,t) \neq 0\}$$

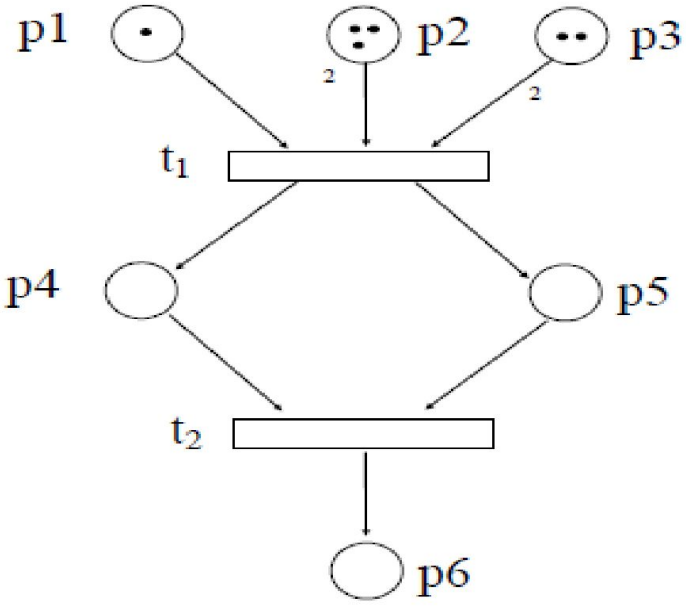
$$p^0 = \{ t \mid t \in T, \exists (p,t) \in F\}$$



Ex.: Petri net

Matricea de incidenţă C

$\mathbf{c_{ij} = post(t_j, p_i) - pre(p_i, t_j), i = 1, ..., m; j = 1, ... n}$



Ex.: Petri net

Exemplu 1

$P = \{p_1, ... , p_6\}$,

$T = \{t_1, t_2\}$,

$\mathbf{M_0} = [1, 3, 2, 0, 0, 0]^T$,

${}^0t_1 = \{p_1, p_2, p_3\}$,

$t_1{}^0 = \{p_4, p_5\}$,

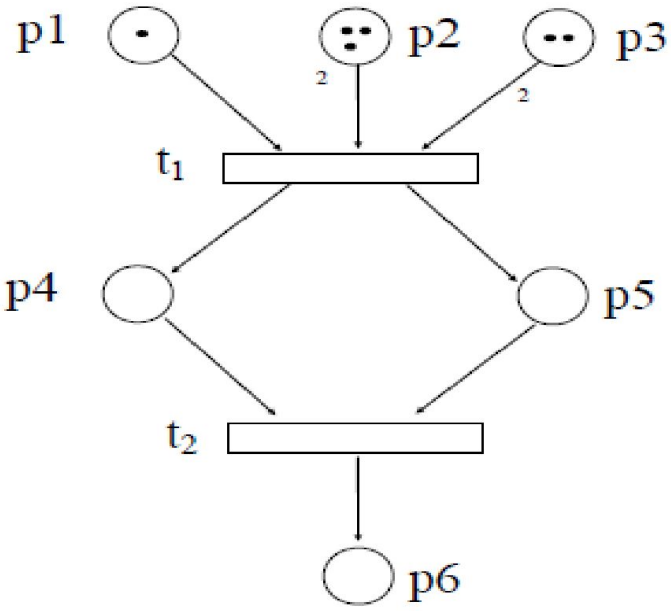
${}^0p_1 = \varnothing$, $p_1{}^0 = \{t_1\}$.

$pre(p_1, t_1) = 1, pre(p_2, t_1) = 2, ... , pre(p_6, t_2) = 0,$

$post(t_1, p_1) = 0, post(t_1, p_2) = 0, ... , post(t_2, p_6) = 1.$

Matricea de incidenţă C:

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



Ex.: Petri net

Matricile **Pre** și **Post** compun C

C = Post - Pre

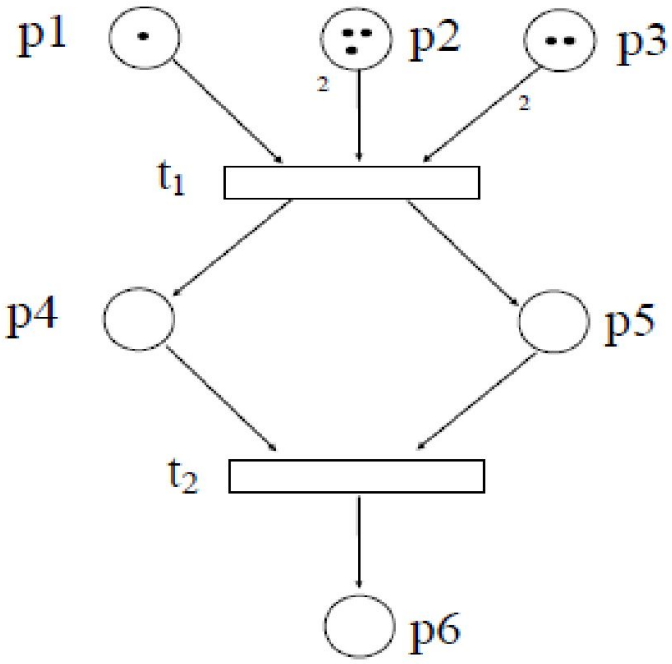
Tranziţie activă (executabilă) – Regula de activare

Tranziția t este ***activă*** pentru marcajul curent M dacă:

$\mathbf{M} = [M(p_1), ..., M(p_m)]^T$

1. $M(p) \geq pre(p,t) \quad \forall p \in {}^0t$;
2. $K(p) \geq M(p) - pre(p,t) + post(t,p) \quad \forall p \in t^0$.

O tranziție activă poate sau nu să se execute.



Ex.: Petri net

Comportament dinamic

O tranziție activă t șterge w(p,t) jetoane din fiecare locație de intrare și adaugă w(t,p) jetoane în fiecare locație de ieșire a tranziției t. În mod similar pot fi folosite matricile *pre* and *post*.

- regula de activare slabă
- regula de activare strictă

$\mathbf{M} \rightarrow_t \mathbf{M'}$

$$\mathbf{M'}(p) = \mathbf{M}(p) - \boldsymbol{pre}(p,t) + \boldsymbol{post}(p,t), \quad \forall p \in P$$

sau

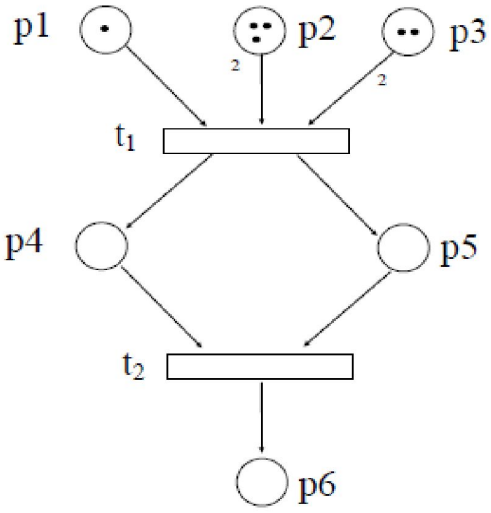
$\mathbf{M'} = \mathbf{M} + col_t(\mathbf{C})$

Notație: $col_t(\mathbf{C})$ coloana t din **C**.

$\mathbf{M}[t\rangle \mathbf{M'} \longleftrightarrow \mathbf{M'} = \mathbf{M} + \mathbf{C} \cdot \mathbf{e}_t \geq 0$

$$\mathbf{e}_t(x) = \begin{cases} 1, \text{ dacă } x = t \\ 0, \text{ altfel} \end{cases}$$

$\mathbf{e}_t$ este ***vectorul de intrare***



Ex.: Petri net

$$\mathbf{M}_0 \xrightarrow{t_1} \mathbf{M}_1 \xrightarrow{t_2} \mathbf{M}_2 \xrightarrow{} \ldots \xrightarrow{t_k} \mathbf{M}_k$$

$\sigma = t_1 * t_2 * \ldots ,$

$\mathbf{M}_0[\sigma\rangle \mathbf{M}_k \quad \Rightarrow \quad \mathbf{M}_k = \mathbf{M}_0 + \mathbf{C} \cdot \boldsymbol{\sigma} \geq 0, \boldsymbol{\sigma} \geq 0$

Implementarea reţelelor Petri P/T

P/T PN simulator:
Input: Pre, Post, M_0, P, T
Output: M, tranziţia executată
Initialization: M = M_0; execList
for ever **do**
*găsim toate tranziţiile executabile:
execList = empty list;// lista tranziţiilor executabile la momentul curent
for all t of T **do**
exec= true;
for all p of P **do**
if (M(p) < Pre((p,t)) **then** exec = false;
end for
if (exec) add *t* to *execList*
end for
* alegem o tranziţie executabilă din lista *execList*
* executăm tranziţia t:
M = M – Pre[t] + Post[t]; //coloana t a matricii corespunzătoare
write M and t;
end for

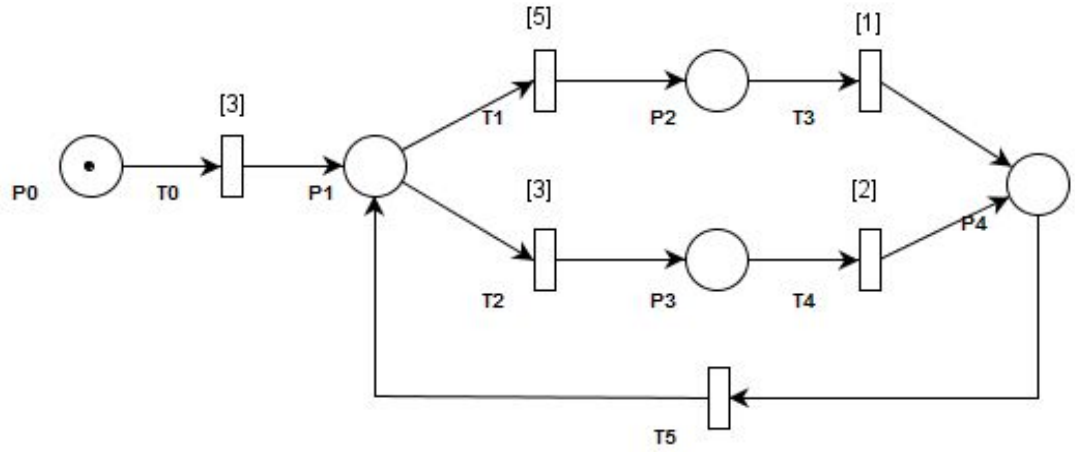
Reţele Petri de timp (Time Petri Nets - TPNs)

N = (P, T, *pre*, *post*) ← P/T PN

Marcajul descrie starea rețelei Petri.

Putem avea (TPN):

- **Locaţie temporizată**
- **Tranziţie temporizată** (folosite în continuare)



$\mathbf{M} = [\, M(p_1), M(p_2), \ldots, M(p_m)]^T$

$\Delta : P \rightarrow \mathbf{N}$ asociază întârzieri (delay) pentru extragerea jetoanelor din locațiile de intrare

$\Delta : T \rightarrow \mathbf{N}$ asociază întârzieri (delay) pentru executarea tranzițiilor

Relativ la TPN vom folosi următoarele ***semantici***:
Modelele de control TPN sunt deterministe și respectă următoarele presupuneri:
1. TPN nu are conflicte sau alegeri libere.
2. Dacă mai mult de o tranziție este executabilă (la un moment dat), atunci este aleasă tranziția cu cel mai mic delay;
3. Dacă modelul TPN are conflicte sau alegeri libere, se aplică regula: ordinea de alegere a tranziției este dată de index-ul ei;
4. Sistemul funcționează cu jetoane rezervate. O tranziției care a început execuția nu poate fi anulată de o alta cu un delay mai mic;
5. Locațiile din mulțimile P_C și P_R sunt folosite exclusiv pentru operații de intrare, respectiv ieșire.
6. Locațiile corespund la acțiuni și nu au durată sau delay.

Implementarea **TPN**

TPN simulator:

Input: Pre, Post, M_0, P, T, D;// D este vectorul întârzierilor pentru tranziţii

Initialization: M_t =M = M_0, *execList*, *pendingTransList*;// M_t – marcaj temporar

for ever **do**

*găsim toate tranziţiile executabile:

execList = empty list;// lista tranziţiilor executabile pentru marcajul curent

do

for all t of T **do**

exec= true;

for all p of P **do**

if (M(p) < Pre((p,t)) **then** exec = false;

end for

if (exec) add *t* to *execList*

end for

* alegem o tranziţie executabilă *t* din lista *execList*;

* începem execuţia tranziţiei t:

M = M – Pre[t];//coloana t din matricea corespunzătoare

$\mathbf{M_{temp}}$ = **M – Pre[t] + Post[t];**//folosit pentru execuţia strictă a tranziţiei

add t to *pendingTransList;*

Delay[t] = D[t];

while(execList **is not empty)**

***** descreştem delay-urile din *pendingTransList;*

if (Delay[t] becomes 0) **then** *se termină execuţia lui t:

M = M + Pos[t];

* ştergem t din *pendingTransList*;

write M and t;

wait(1 t.u.);

end for

Ce pot să descrie reţelele Petri ?

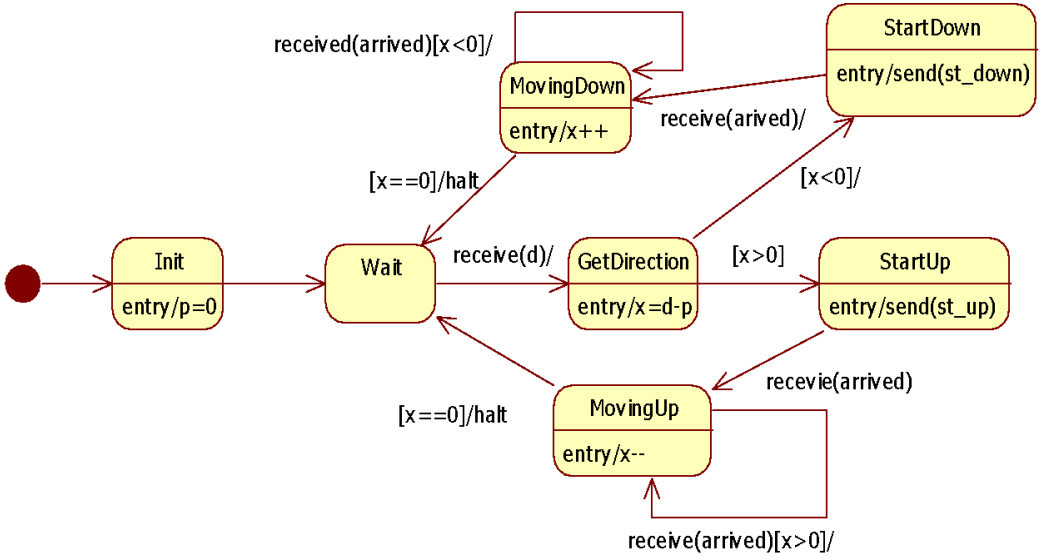
- Structură și comportament
- ȘI (AND)
- SAU (OR)
- Concurența
- Sincronizarea
- Excluderea mutuală
- Interblocajul
- Execuția infinită (dorită sau nu)
- Temporizări
- Arce inhibitoare ???
- Arce de reset ??

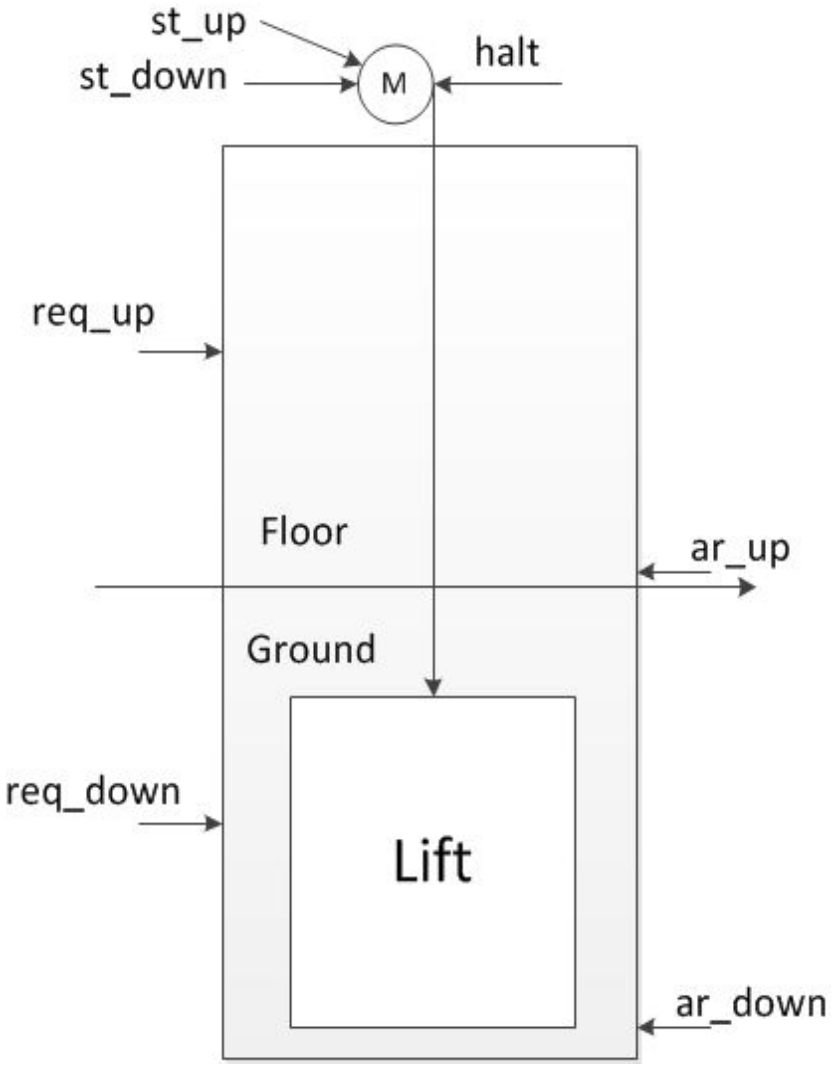
4.3 Specificarea sistemelor cu evenimente discrete

Notații: d (demand), st (start), req (request)

Temă de casă: Construiți modelul ETPN corespunzător.

Specificati interfața dintre controler și sistem !



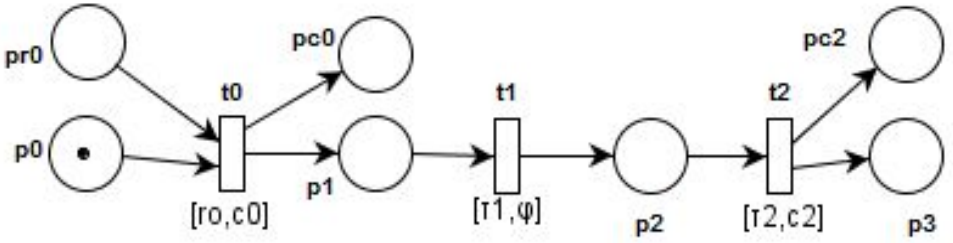


Specificațiile includ:

- structură
- comportament
- interfețe

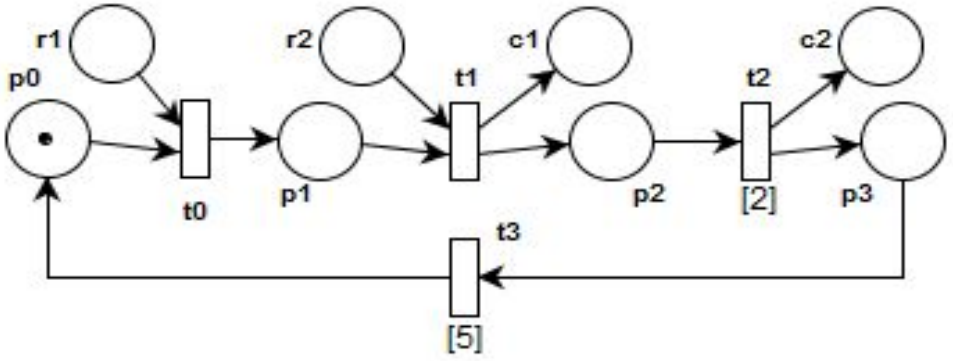
Enhanced Time Petri Nets (ETPNs)

$$\sigma = t0[r_0, c_0] * t1[\tau_1, \phi] * t2[\tau_2, c_2]$$



InputPlaceSet = {pr0}
OutputPlaceSet = {pc0, pc2}

întârzieri: τ1, τ2



ETPN

Inp = $\{r_1, r_2\}$; *Out* = $\{c_1, c_2\}$;

D = {0, 0, 2, 5}

Formal description:

$\sigma = (t_0[r_1,\varphi] * t_1[r_2,c_1] * t_2[2,c_2]) \# (t_3[5, \varphi])$

4.4 Rețele Petri de nivel înalt

HighLevel Petri Nets (HLPN)

Dezavantajul rețelelor Petri: explozia numărului de elemente.

HLPN ←➔ programare de nivel înalt

HLPN:

- rețele Petri colorate
- rețele de tranziții cu predicate

High Level Petri Net Graph

HLPN=(P, T, D, Type, Pre, Post, M_0)

D este o mulțime finită nevidă de domenii unde fiecare element este un tip (type).

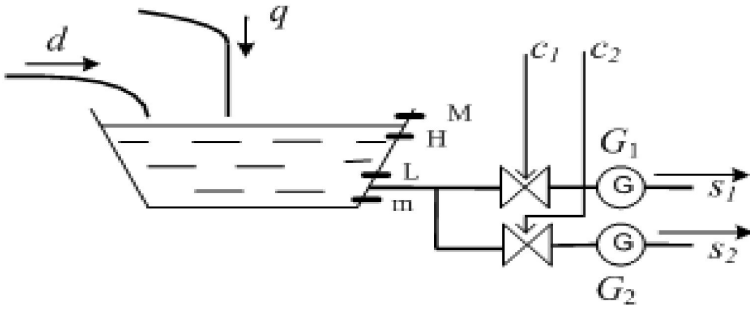
$Type : P \cup T \rightarrow D$ o funcție care asignează tipuri la locații și determină modurile tranzițiilor.

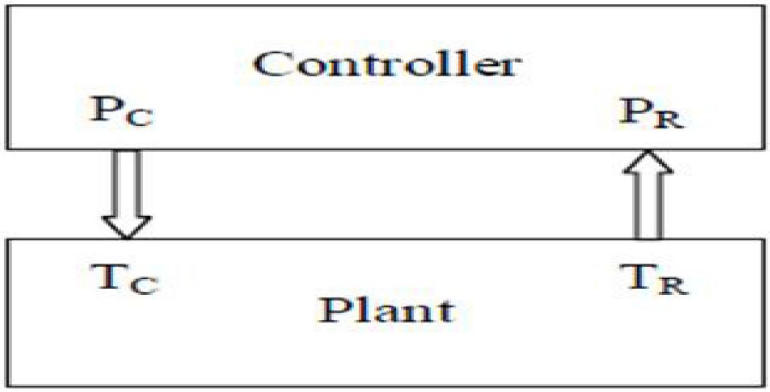
$Pre,\ Post : TRANS \rightarrow \mu PLACE$

- O structură algebrică
- Set de locații
- Set de tranziții
- Set de tipuri
- Fiecare tranziție are un tip
- Un set de moduri (pentru fiecare tranziție)
- Pre – o funcție pentru intrarea jetoanelor în tranziție (pt. fiecare mod)
- Post – o funcție pentru ieșirea jetoanelor din tranziție (pt. fiecare mod)
- Un marcaj inițial

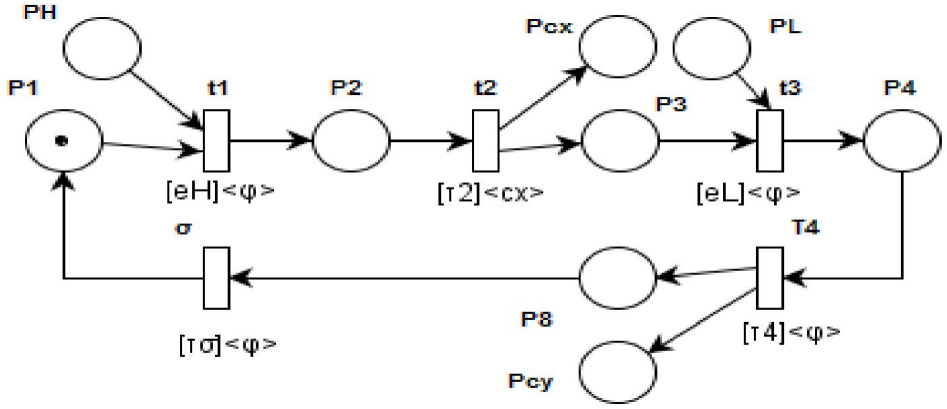
4.5 Specificarea sistemelor continue

exemplu: controlul unui sistem de putere hidroelectrică





- structură
- comportament
- interfețe



Modelul unui controler ETPN

Octavian Cuibus: Sisteme de control distribuit: Specificare, verificare, proiectare

Modele:

- ecuații diferențiale
- Transformări Laplace pentru sisteme liniare

Modelele descriu relația dintre variabile (intrări, stări, ieșiri, parametri).

logică fuzzy ?

Logică Fuzzy = logică neclară, vagă

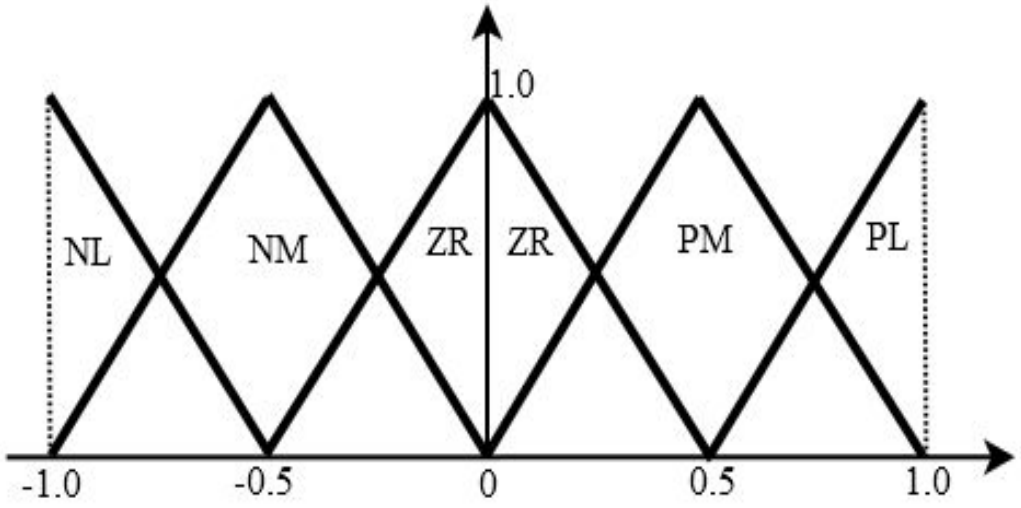
De ce logică fuzzy ?
De ce Fuzzy Logic Control (FLC) ?
Metodele convenționale sunt potrivite pentru probleme simple.
Sistemele fuzzy pot fi folosite pentru probleme complexe sau aplicații care includ modul de gândire uman (intuitiv, descriptiv: când cunoștințele despre sistem sunt într-o formă aproximativă sau necesită reguli euristice).

Matematicianul german **Georg Cantor** (1845-1918)
Mulțimi Fuzzy (mulțimi neclare, vagi)
O *mulțime fuzzy A* este un set de obiecte definite și distincte din intuiția noastră care pot fi tratate ca un întreg.

Mulțimi Fuzzy: FS=*{H- high, M-medium, L-low};*
FS={NL, NM, ZR, PM, PL}
{NL-negative large, NM-negative medium, , NS-negative small, ZE-zero, PS-positive small, PM-positive medium, PL-positive large }

Lotfi Zadeh (1965) – ***logică fuzzy***

Gradul de apartenență



Mulțimea de obiecte U
Mulțimea fuzzy A în U este:

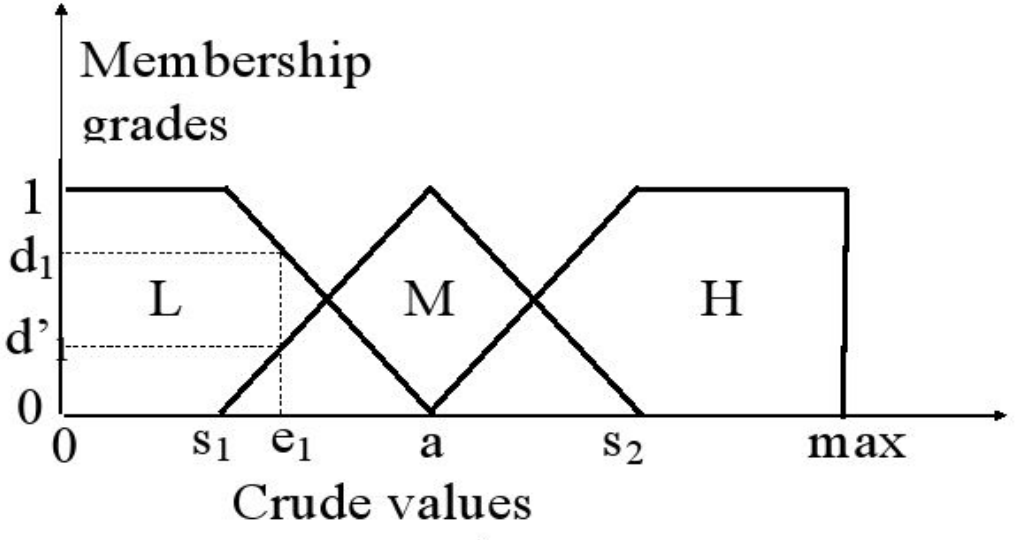
$A \equiv \{<x, \mu_A(x)> \mid x \in U\}$

$\mu_A(.)$ este ***funcția de apartenență (membership function)***

$0 \leq \mu \leq 1$

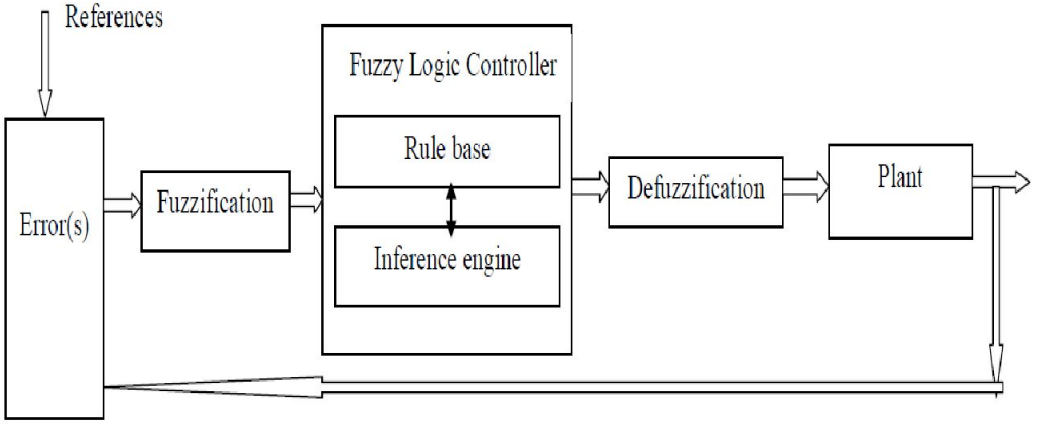
$<x, \mu_A(x)>$ este un ***fuzzy singleton***

$\mu \in \{0, 1\}$ logică clasică



FLC (fuzzy logic control):

- fuzzificator
  - măsoară intrările;
  - transformă intrările în **grade de apartenenţă** (pe baza setului de mărimi fuzzy)
- ***Control fuzzy*** (FLC) - setul de reguli determină politica de control
- defuzzificator - determină mărimile de control



e = r - y

e = [e_1, e_2, ...] ; **r** = [r_1, r_2, ...] ; **y** = [y_1, y_2, ...] ;

fuzzificare: (**e**) ➔E

FC(**E**) ➔ C

Defuzzyficare(C) ➔ **c** (mărimi de control)

Fuzzificare (fuzzyfication)

Funcţii de apartenenţă (Membership functions)

Mulțimi fuzzy:

Univers U={H- high, M-medium, L-low}; sau

U = {NL-negative large, NM-negative medium, NS-negative small, ZE-zero, PS-positive small, PM-positive medium, PL-positive large }

e_1 este L cu grad de apartenență d_1

e_1 este M cu grad de apartenență d'_1

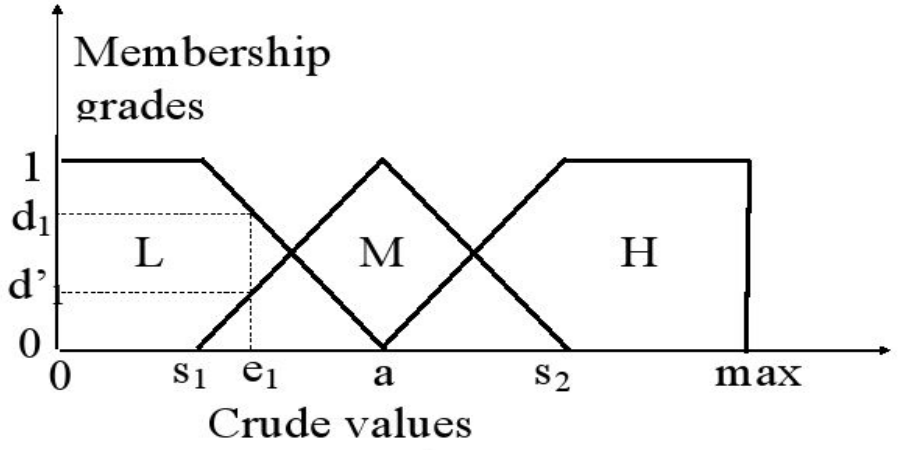
$e_1 = <L, \mu_1> = <M, \mu'_1>$

$\mu_1 + \mu'_1 = 1$

Ex.:

$y \in <M, 0.3>$

$y \in <L, 0.7>$

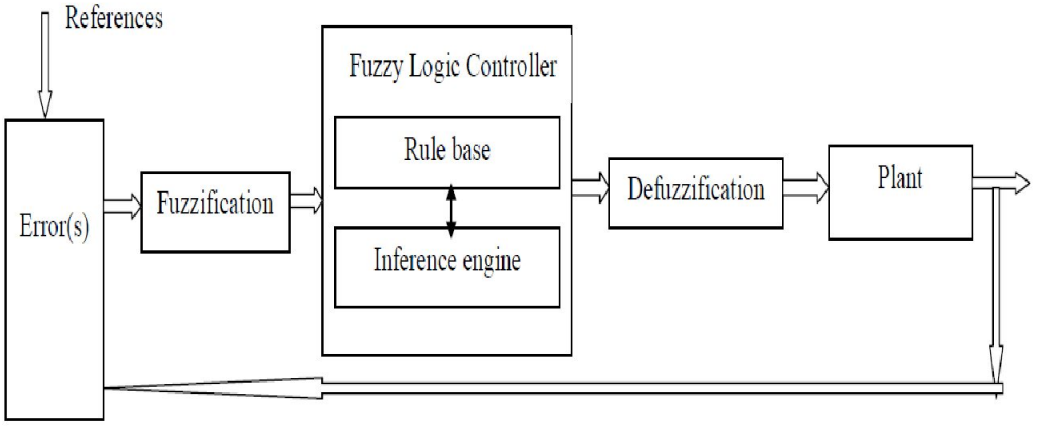


În literatură se găsesc mai multe modele de funcții de apartenență.

Control Fuzzy (FLC)
(pe baza regulilor fuzzy)

Logica fuzzy lucrează cu:

- propoziţii
- ¬ pentru ’NOT’
- ∧ pentru ’AND’
- ∨ pentru ’OR’
- ⇒ pentru ’IF-THEN’ (implicaţie)
- ⇔ pentru ’IF AND ONLY IF’ (echivalenţă)



A ={ A_1, A_2, …, A_m} ; Ex.: U={H,M,L} univers discret; $A_i \in$ {H,M,L}

IF (x_1 is A_1) AND (x_2 is A_2) AND ….AND (x_n is A_m)

THEN (z_1 is C_1); (z_2 is C_2);…; (z_n is C_n)

unde A_i, C_i aparţin mulţimii U={H, M, L}; i=1, 2, …

$\boldsymbol{C}$ ={ C_1, C_2, …, C_n} ; Ex.: $C_i \in$ {H,M,L}

	E_2			
E_1		**L**	**M**	**H**
	L		(M, L)	
	M	(L,L)	**(c_1,c_2)**	
	H		(M,L)	

Surse pentru aflarea regulilor fuzzy de control:

- experiența experților și cunoștințe de ingineria controlului
- pe baza acțiunilor operatorului uman
- învățare
- folosind algoritmi genetici

Surse pentru stabilirea funcţiilor de apartenenţă:

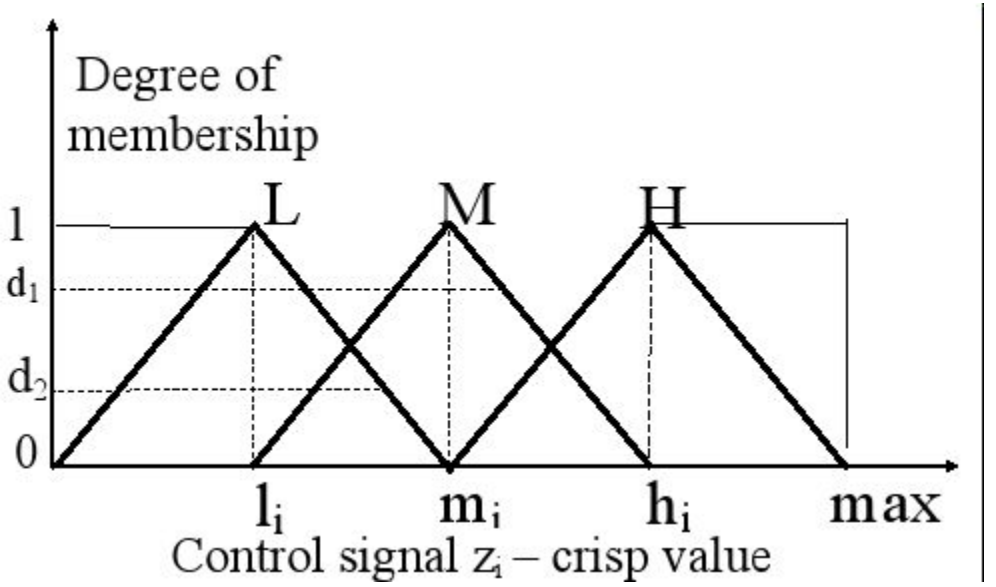
- experiența experților
- folosind algoritmi genetici
- pe bază de simulări și evaluarea performanțelor

Defuzzificare

➔ Valorile „crisp”

Defuzzificarea este transformarea rezultatelor inferențelor ale regulilor activate în ieșiri numerice (crisp)

Una sau mai multe reguli pot fi activate simultan.



Agregarea:
Operația de *agregare* este folosită pentru a calcula ***gradele de indeplinire*** sau ***puterea regulii*** s_k a condiției regulii k.

r_k: *IF (e_1 is L) AND (e_2 is M) THEN (c is M);*

$e_1 = <L, \mu_L(e_1)>$; $e_2 = <M, \mu_M(e_2)>$;
Puterea regulii *k* (de tip AND) poate fi calculată ca:
$s_k = \min\{ \mu_L(e_1) , \mu_M(e_2)\}$ sau
$s_k = \mu_L(e_1) * \mu_M(e_2)$

Activarea:
O regulă *k* poate fi scalată cu un ***factor de scalare*** $\gamma_k \in [0, 1]$ (reprezintă gradul de încredere în acea regulă)
$f_i = \gamma_k * s_k$

Acumularea

Toate concluziile activate sunt acumulate (adunate) folosind:

$\gamma_1 * s_1 + \gamma_2 * s_2 + \ldots + \gamma_n * s_n$

pentru a calcula valorile mărimilor de ieșire.

Formula de defuzzificare

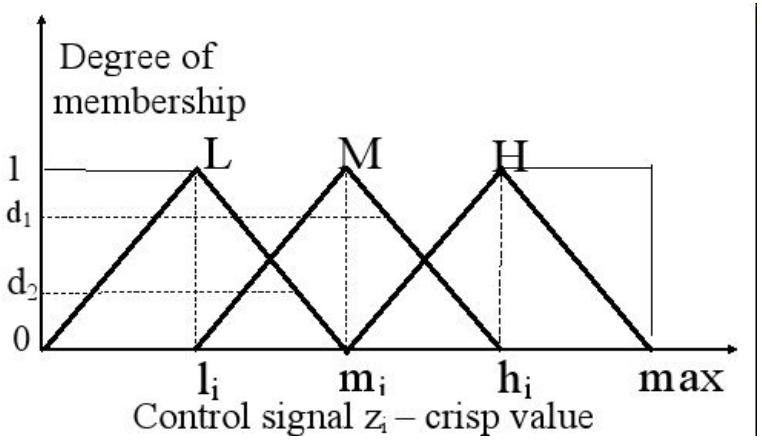
Centrul de greutate (centrul ariilor):

$$c_i = \frac{\sum_{j=1}^{m} z_j \cdot \mu(z_j)}{\sum_{j=1}^{m} \mu(z_j)}$$

Exemplu (centrul de greutate):

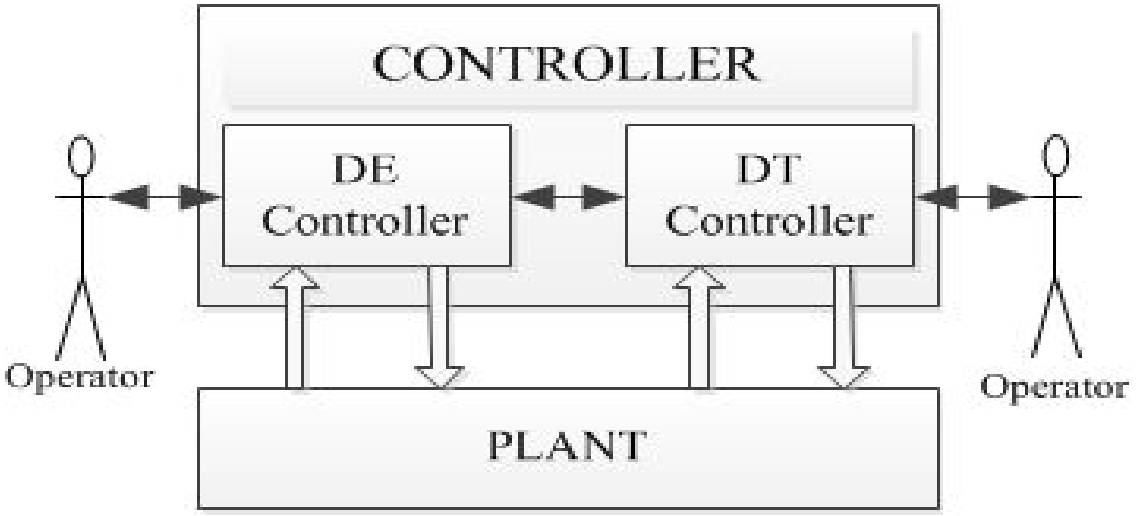
$$c_i = (s_1 \cdot z^1_i + s_2 \cdot z^2_2 + \cdots + s_k \cdot z^k_i)/(d_1 + d_2 + \cdots + d_k) =$$
$$(s_1 \cdot m_i + s_2 \cdot l_i + \cdots + s_k \cdot h_i)/ (d_1 + d_2 + \cdots + d_k)$$

- l_i, m_i, h_i este poziția lui j în universul acelei reguli
- s_j este puterea acelei reguli



4.6 Sisteme de control hibrid

DE: evenimente discrete
DT: timp discret



Scopul specificării

- pentru a concepe modele capabile să descrie ***comportamentul şi structura controlerului***
- pentru a face posibilă verificarea că modelul îndeplineşte cerinţele specificate
- ***scop ascuns*** ➔ modelele să fie folosibile pentru ***sinteza automată***

Ce vrem: modele dinamice capabile să reprezinte următoarele specificații:

- reacția la ***evenimente sincrone și asincrone*** (care au variabile continue în loc de variabile cu valoare unică)
- unele ***mărimi de control să fie în domeniul continuu***
- unele reacții necesită execuția activităților care includ perioade neneglijabile și care pot să aibă constrângeri de timp real ➔ ***comportament concurent***

Rețele Petri de nivel jos: au doar elemente grafice (locații, tranziții, arce, jetoane). Pot modela fluxul de control, dar nu pot modela funcționalitatea sistemelor complexe

Rețele Petri de nivel înalt (HLPNs) pot modela fluxul de control, dar și funcționalități complexe. Tipuri de HLPNs:

- rețele Petri colorate

- rețele Petri cu predicate pe tranziții
- rețele Petri bazate pe logică fuzzy

Fuzzy Logic Enhanced Time Petri Nets (FLETPN)

Lucrurile cu care interacţionăm includ tot mai multă capacitate de inferenţă. Devin mai inteligente !

Cum ne facem mai inteligenţi decât lucrurile din jurul nostru ?

➔ prin îmbunătăţirea gândirii şi a stilului de abordare !

Reţele Petri de nivel jos

$PN = (P, T, pre, post, \mathbf{M})$

$P = \{p_1, p_2, \ldots, p_m\}$

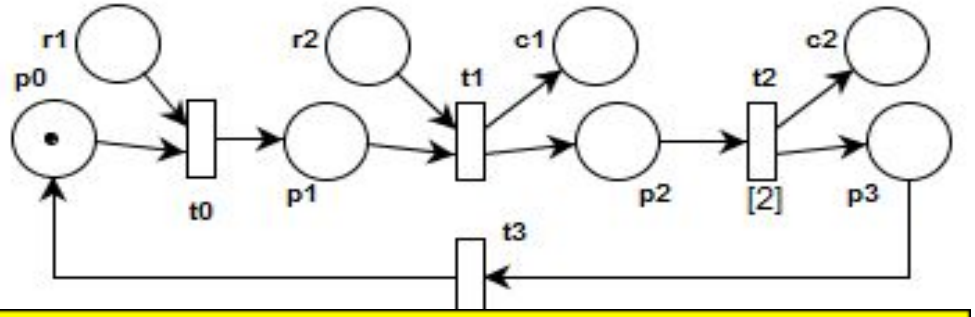
$T = \{t_1, t_2, \ldots, t_n\}$

$\mathbf{M} = [M(p_1), M(p_2), \ldots, M(p_m)]$

Reţele Petri de timp (Time Petri Nets)

$TPN = (P, T, pre, post, D, \mathbf{M})$

$D = \{d_1, d_2, \ldots, d_n\}$; //întârzieri



ETPN

$Inp = \{r_1, r_2\}$; $Out = \{c_1, c_2\}$;

$D = \{0, 0, 2, 5\}$

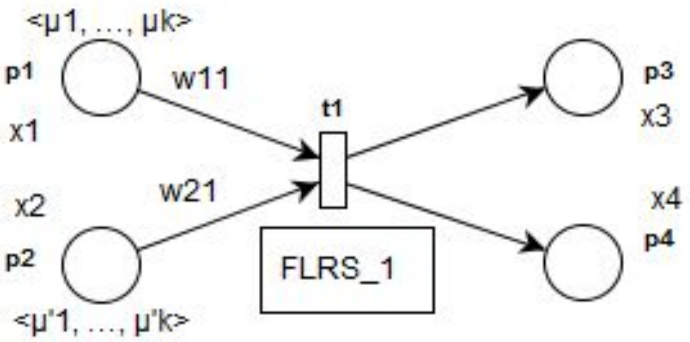
Descriere formală:

$\sigma = (t_0[r_1,\varphi]*t_1[r_2,c_1]*t_2[2,c_2]) \# (t_3[5, \varphi])$

Enhanced Time Petri Nets

Sunt rețele Petri de timp care au locații de intrare și de ieșire.

Reţele Petri de nivel înalt (jetoane distincte)



FLETPN =(P, T, pre, post, D, W, X, EFS, ***FLRS****, α, β, γ,* ***M****)*

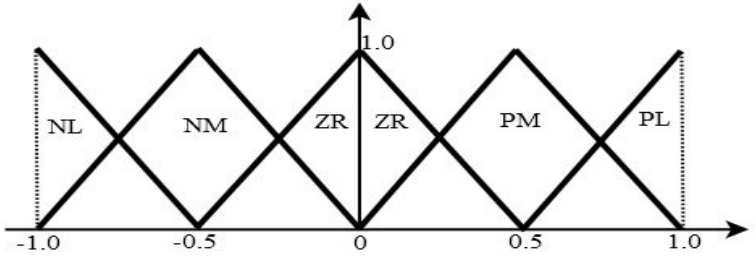
$X = \{x_1, x_2, \ldots, x_m\}; x_i \in [-1, 1]$

$W: P \times T \rightarrow \mathbf{R}; W(p_i, t_j) \in \mathbf{R}$

$\alpha: P \rightarrow X$

$\beta: T \rightarrow \mathbf{FLRS}$

$\gamma : T \rightarrow R \square X \square Inp \; ; Inp \square P$



Fuzzy Logic ←➔ Fuzzy Set

FS = {NL, NM, ZR, PM, PL}

EFS = {NL, NM, ZR, PM, PL, Φ}

EFS: setul fuzzy extins

FLRS (fuzzy logic rule set) setul de reguli fuzzy.

$\mathbf{FLRS} = \{ FLRS_1 , FLRS_2 , ..., FLRS_k \}$

$FLRS_i$ este un set de reguli fuzzy asociate tranziţiei t_i.

Regulile fuzzy pentru diferite tranziţii nu e neapărat să fie diferite

Mulţimea fuzzy

FS = {NL, NM, ZR, PM, PL}

$x_1 \hat{} x_2$	***NL***	***NM***	***ZR***	***PM***	***PL***	*Φ*
NL	*Φ, Φ*	*PL, PM*	*PL, ZR*	*PL, NM*	*PL, NL*	*Φ, Φ*
NM	*PM, PL*	*Φ, Φ*	*PM, ZR*	*PM, NM*	*PM, NL*	*Φ, Φ*
ZR	*ZR, PL*	*,*	*ZR, ZR*	*ZR, NM*	*ZR, NL*	*Φ, Φ*
PM	*NM, PL*	*NM, PM*	*NM, ZR*	*,*	*NM, NL*	*Φ, Φ*
PL	*NL, PL*	*NL, PM*	*NL, ZR*	*NL, NM*	*,*	*Φ, Φ*
Φ	*Φ, Φ*	*-, -*	*-, Φ*	*Φ, -*	*-, -*	*-, -*

EFS = {NL, NM, ZR, PM, PL, Φ}

Examples of Fuzzy Logic Rule Set:

$x_1 \check{} x_2$	**NL**	**NM**	**ZR**	**PM**	**PL**	***Φ***
NL	PL, PL	PL, PM	PL, ZR	PL, NM	PL, NL	*Φ, Φ*
NM	*,*	PM, PM	PM, ZR	*Φ* , NM	PM, NL	ZR, *Φ*
ZR	ZR, PL	*Φ, Φ*	ZR, ZR	NM,PM	ZR, NL	NL, ZR
PM	NM, PL	*Φ* , PM	NM, ZR	NM, NM	*,*	*Φ, Φ*
PL	NL, PL	*,*	NL, ZR	*Φ* , NM	NL, NL	*Φ, Φ*
Φ	*Φ, Φ*	*Φ, Φ*	NL, ZR	ZR, *Φ*	*Φ, Φ*	*Φ, Φ*

IF (x_1 ix NM) ^ (x_2 ix PM) THEN

$(x_3$ *is PM)* $\hat{}$ $(x_4$ *is NM)*

IF $(x_1$ *is ZR)* $\check{}$ $(x_2$ *is PM) THEN*

$(x_3$ *is NM)* $\hat{}$ $(x_4$ *is PM)*

Pentru reprezentare facilă, regulile $FLRS_i$ ale tranziției t_i sunt stocate în două tabele: unul pentru regulile AND și unul pentru regulile OR).

Nu este neapărat să avem definite toate regulile pentru toate premisele posibile. Când regula lipsește, este marcată în tabelul FLRS cu "-" sau loc gol.

Exemple:
IF (x_1 *is ZR*) ^ (x_2 *is NM);* // consecința lipsește ➔ nu este nimic în tabel: (,)

IF (x_1 *is ZR*) ^ (x_2 *is PM*) *THEN* (x_3 *is Φ*) ^ (x_4 *is Φ);* // avem în tabelul ^ FLRS consecința (Φ, Φ).

Ultima regulă spune "Nu sunt cunoscute valorile lui x_3 și x_4"

Jetonul: $<\mu_{NL}, \mu_{NM}, \mu_{ZR}, \mu_{PM}, \mu_{PL}>$

$\mu_{NL} + \mu_{NM} + \mu_{ZR} + \mu_{PM} + \mu_{PL} = 1$

x este Φ ←➔ Nu se cunoaște valorea lui x la momentul curent.
Φ nu produce jetoane. Jetoanele sunt construite pe baza μ_K. K ϵ FS

Setul tuturor jetoanelor este:
$S = \{<\mu_1, \mu_2, \mu_3, \mu_4, \mu_5> |$ pentru toți *i*, $\mu_i \in [0,1], \sum_i \mu_i = 1\}$;

Fuzzificare xϵ [-1, 1]

regula AND

a) $x \epsilon [0, 0.5]$ ➔ $\mu_{ZR} = 1 - x/0.5$

➔ $\mu_{PM} = x/0.5$

Test: $\mu_{ZR} + \mu_{PM} = 1$

b) $x \epsilon [0.5, 1]$ ➔ $\mu_{PM} = ?$

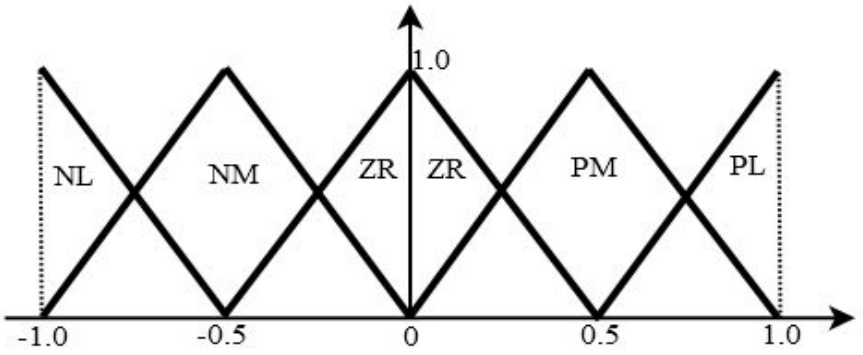
➔ $\mu_{PL} = ?$

c) $x \epsilon [-1, 0.5]$ ➔ $\mu_{NM} = ?$

➔ $\mu_{NL} = ?$

d) $x \epsilon [-0.5, 0]$ ➔ $\mu_{NM} = ?$

➔ $\mu_{ZR} = ?$



Example of inference rules

$x_1 \backslash x_2$	NL	NM	ZR	PM	PL
NL	NL, ϕ	NL, Φ	NL,PL	ZR,PL	PL,PL
NM	NM, Φ	PL, Φ	PL,PL	PL,PL	NL,PL
ZR	PM,PL	PM,PL	ZR,PL	NL,PL	PM,PL
PM	ZR,PL	PL,PL	NL,PL	ϕ, NM	ϕ, PL
PL	NL,PL	ZR,PL	ZR,PL	ϕ, PM	ϕ, PM

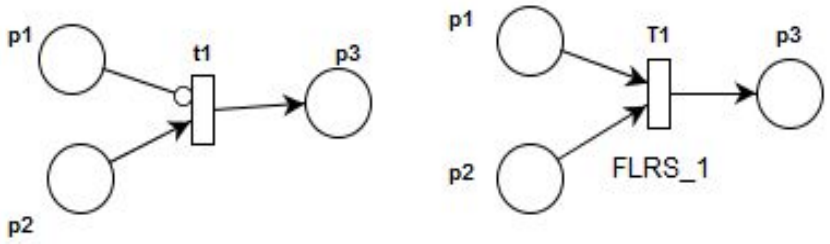
Marcaj:

$M :\rightarrow S \square \Phi$; $M(p_i) = <\mu_{NL}, \mu_{NM}, \mu_{ZR}, \mu_{PM}, \mu_{PL}>$ or Φ

O locație poate să nu conțină jeton (adică *Φ)* sau poate conține doar un singur jeton. Dacă o locație conține deja un jeton și trebuie pus un alt jeton la momentul prezent, ultimul jeton îl va înlocui pe cel existent

FLETPN cu arce inhibitoare

IF (x_1 is Φ) ^ (x_2 is PL) THEN x_3 is NL



rețele Petri cu arce inhibitoare

x_1 este Φ ➔ *p_1* nu are jetoane

dacă FLRS_1 conține regula

IF (x_1 is Φ) ^ (x_2 is PL) THEN x_3 is NL

atunci cele două rețele au comportament similar dacă nu există jeton în p_1.

Dacă p_1 nu are jeton și p_2 are un jeton care activează o regulă din FLRS atunci t_1 este executabilă.

Diferența dintre cele două rețele: în a doua rețea Petri, t_1 poate fi executată și daca există jeton în p_1 și acesta poate activa o regulă din FLRS_1.

Defuzzificare

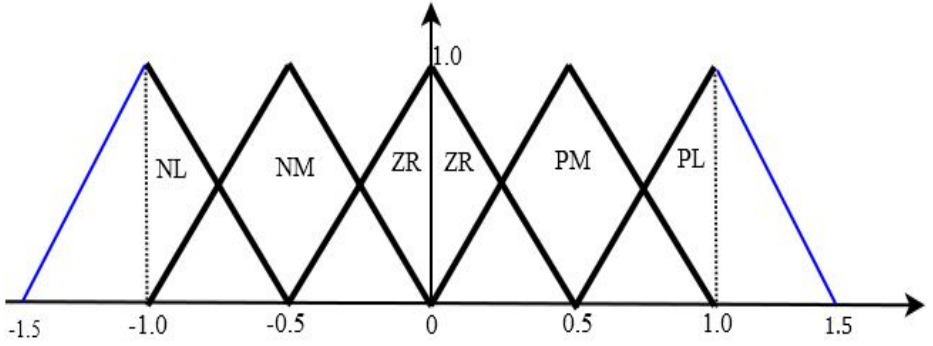
Example of inference rules

$x_1 \backslash x_2$	NL	NM	ZR	PM	PL
NL	NL, ϕ	NL, Φ	NL,PL	ZR,PL	PL,PL
NM	NM, Φ	PL, Φ	PL,PL	PL,PL	NL,PL
ZR	PM,PL	PM,PL	ZR,PL	NL,PL	PM,PL
PM	ZR,PL	PL,PL	NL,PL	ϕ, NM	ϕ, PL
PL	NL,PL	ZR,PL	ZR,PL	ϕ, PM	ϕ, PM

IF (x_1 is ?) ^ (x_2 is ?) THEN (x_3 is ?) ^ (x_4 is ?)

x' ϵ {x_3, x_4} valoarea defuzzificată

$$x' = \frac{\sum_l z_l \cdot s_l}{\sum_l s_l}$$



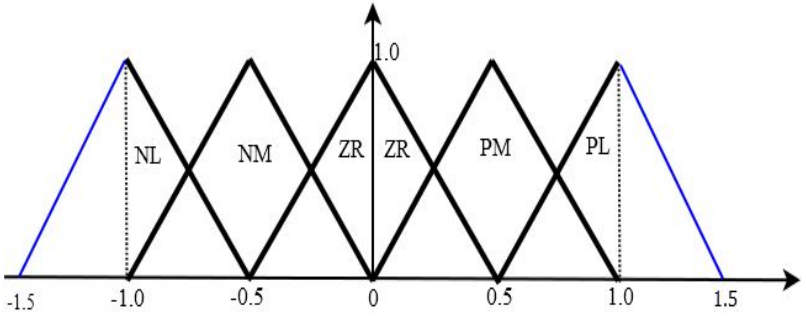
z_l ϵ {-1, -0.5, 0, 0.5, 1};
if (x' is PM) ➔ z_{PM} = 0.5; if (x' is PL) ➔ z_{PL} = 1
$s_l = \mu_i \cdot \mu_j \cdots$ puterea regulii r_l este dată de gradele de apartenență ale intrărilor;

Exemplu: x_1=0.25 ➔ μ_{ZR}=0.5; μ_{PM}=0.5; μ_{PL}= μ_{NL}= μ_{NM}=0; <0, 0, 0.5, 0.5, 0>;
x_1 is {(ZR,0.5), (PM,0.5))}

x_2= 0.5 ➔ μ_{PL}= μ_{NL}= μ_{NM}= μ_{ZR}=0; μ_{PM}=1; <0,0,0.,1,0>; *x_2 is (PM, 1)*

IF (x_1 is ZR) ^ (x_2 is PM) THEN (x_3 is NL) ^ (x_4 is PL)➔ puterea: s=0.5
x_i ϵ [-1, 1]; x_3=?; x_4=?

Defuzzificare 2



Dacă regula este o regulă AND :

IF (x_1 is μ_1) ^ (x_2 is μ_2) THEN (x_3 is ?) ^ (x_4 is ?)

$s_{l=} \mu_1 \cdot \mu_2$ puterea regulii r_l este dată de gradele de apartenență ale valorilor intrărilor;

Exemplu: x_1=0.25 ➔ μ_{ZR}=0.5; μ_{PM}=0.5; μ_{PL}= μ_{NL}= μ_{NM}=0; <0 , 0, 0 5, 0.5, 0>;

x_1 *is {(ZR, 0.5), (PM, 0.5)}*

x_2= 0.5 ➔ μ_{PL}= μ_{NL}= μ_{NM}= μ_{ZR}=0; μ_{PM}=1; <0,0,0.,1,0>; x_2 *is (PM, 1)*

IF (x_1 is ZR) ^ (x_2 is PM) THEN (x_3 is NL) ^ (x_4 is PL)➔ puterea: s=0.5

$x_i \in [-1, 1]$; x_3=?; x_4=?

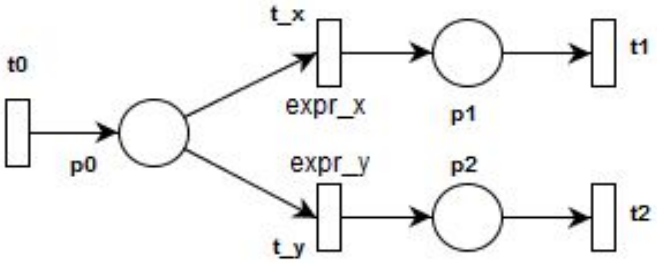
Dacă regula este o regulă OR:

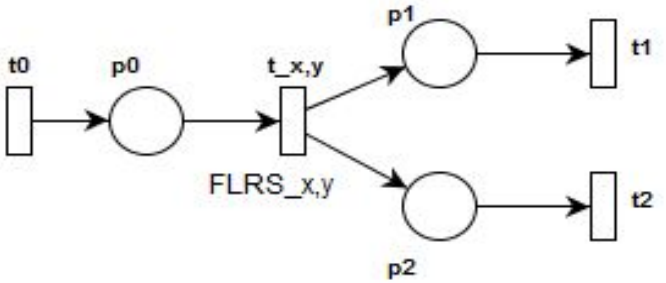
IF (x_1 is A_1) ˇ (x_2 is A_2) THEN (x_3 is A_3) ^ (x_4 is A_4)

$x_1 = (A_1, \mu_1)$, $x_2 = (A_2, \mu_2)$,

$s_i = max \{ \mu_1, \mu_2\}$; or $s_i = min \{ \mu_1, \mu_2\}$; ➔ $x_3 = (A_3, \mu_3)$

Selecţia execuţiilor





x is Φ este folosit pentru selecție

Nu se creează jeton pentru *x is Φ*!

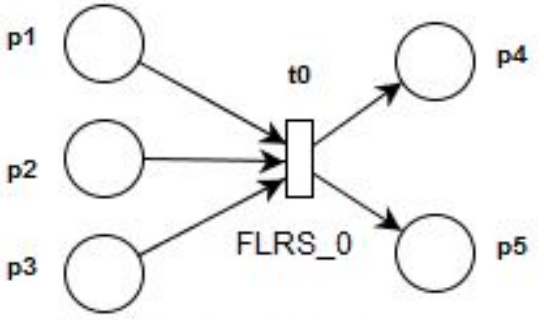
NL	NM	ZR	PM	PL
Φ,ZR	Φ,ZR	PL, PL	ZR,Φ	ZR,Φ

Păstrăm FLRS cu două intrări

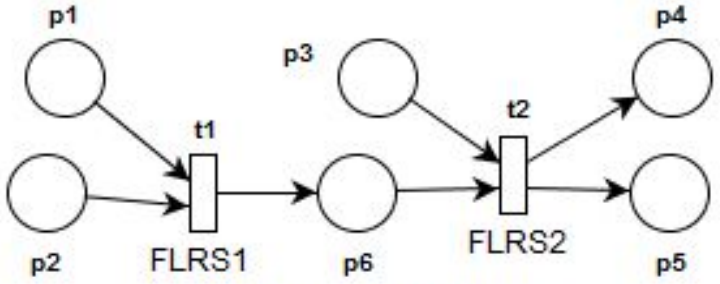
Example of inference rules

$x_1 \backslash x_2$	NL	NM	ZR	PM	PL
NL	NL, ϕ	NL, Φ	NL,PL	ZR,PL	PL,PL
NM	NM, Φ	PL, Φ	PL,PL	PL,PL	NL,PL
ZR	PM,PL	PM,PL	ZR,PL	NL,PL	PM,PL
PM	ZR,PL	PL,PL	NL,PL	ϕ, NM	ϕ, PL
PL	NL,PL	ZR,PL	ZR,PL	ϕ, PM	ϕ, PM

A



a) Initial FLETPN



b) Developed FLETPN

Tranziţia t_i este activată dacă și numai dacă există cel puțin o regulă în $FLRS_i$ care poate fi activată cu conținutul locației sale de intrare la momentul curent.

Dacă mai mult de o regulă poate fi activată la momentul curent, toate regulile sunt folosite și jetonul rezultant este suma jetoanelor. Se face normalizarea ieșirilor pentru a ne asigura că locația de ieșire conține un jeton care respectă condiția:

$$\mu_{NL} + \mu_{NM} + \mu_{ZR} + \mu_{PM} + \mu_{PL} = 1.$$

Execuţia unei tranziţii executabile t_j implică:

- extragerea jetoanelor din locațiile de intrare (notate cu ${}^{o}t_j$);
- defuzzificarea mărimilor de intrare x_i ;
- înmulțirea variabilelor cu factorii corespunzători de scalare:

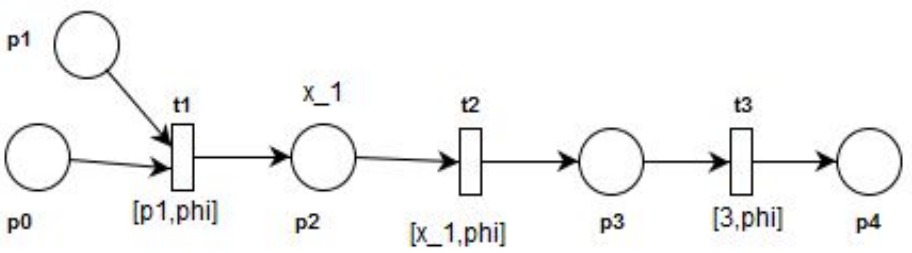
$$x_{ij} = w_{ij} \cdot x_i$$

- fuzzificarea variabilelor x_{ij} ;
- folosirea regulilor FLRS cu x_{ij} ca intrări;
- operația de normalizare care reduce consecințele de mai sus la una singură și asigură inserarea unui singur jeton în locațiile de ieșire;
- inserarea jetoanelor de ieșire în locațiile de ieșire ale tranziției (notate cu t^{o}_{j}) atunci când expiră întârzierea;
- dacă tranziția a fost startată, o altă startare poate avea loc numai după terminarea execuției în curs.

Tranziţii cu întârzieri variabile

Momentul în care o tranziţie este declanşată depinde de:

- o variabilă externă (de ex. t_1 , $t_1[p_1,\varphi]$)
- valoarea dintr-un interval (e.g. $t_2[x_1,\varphi]$)
- o durată fixă (e.g. $t_3[3, \varphi]$)

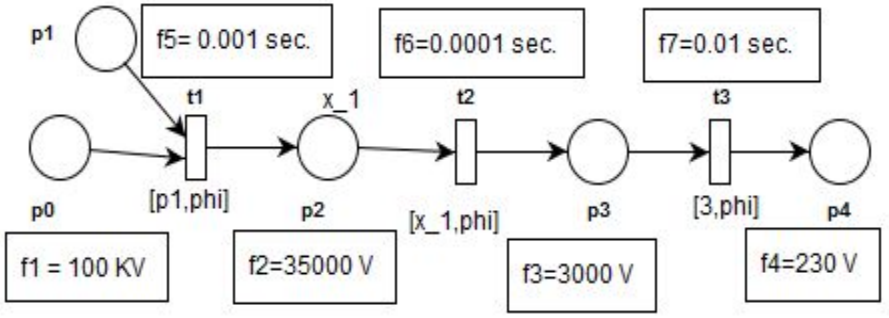


Simularea

Întotdeauna există factori de scalare între modele şi procesele reale. Le notăm cu:

$F=\{f_1, f_2, \cdots, f_m, f_{m+1}, \cdots, f_{m+n}\}$

Sunt folosiţi pentru a obţine valorile reale din cele ale simulării, sau pentru a sincroniza simularea cu timpul fizic.



Exemplu: FLRS – inversor

$(x_1, x_2) \rightarrow (x_3, x_4)$ $x_3 = - x_1; x_4 = - x_2;$ $w_1 = w_2 = 1$

x_1 ^ x_2	NL	NM	ZR	PM	PL
NL	PL, PL	PL, PM	PL, ZR	PL, NM	PL, NL
NM	PM, PL	PM, PM	PM, ZR	PM, NM	PM, NL
ZR	ZR, PL	ZR, PM	ZR, ZR	ZR, NM	ZR, NL
PM	NM, PL	NM, PM	NM, ZR	NM, NM	NM, NL
PL	NL, PL	NL, PM	NL, ZR	NL, NM	NL, NL

Exemplu:FLRS – sumator/ diferenţiator

$(x_1, x_2) \rightarrow (x_3, x_4)$ $x_3 = x_1 + x_2;$ $x_4 = x_1 - x_2;$ $w_1 = w_2 = 1$

x_1 ^ x_2	NL	NM	ZR	PM	PL
NL	NL, ZR	NL, NM	NL, NL	NM, NL	ZR, NL
NM	NL, PM	NL, ZR	NM, NM	ZR, NL	PM, NL
ZR	NL, PL	NM, PM	ZR, ZR	PM, NM	PL, NL
PM	NM, PL	ZR, PL	PM, PM	PL, ZR	PL, NM
PL	ZR, PL	PM, PL	PL, PL	PL, PM	PL, ZR

Găsirea regulilor FLRS şi a factorilor de scalare

x_1^x_2	NL	NM	ZR	PM	PL
NL	NL,NL	NL,PL	NL,PM	NL,PL	NL,PM
NM	NL,M	NM,PM	PM,PM	NM,PM	NM,PM
ZR	ZR,PM	ZR,PL	PM,PL	PM,PL	PM,PL
PM	NL,PL	NL,PM	NL,PL	NL,PM	NL,PL
PL	PM,PM	PM,PM	PM,PM	PL,PM	PL,PL

Algoritmi genetici

Funcția de performanță:

$$J = \sum_{j=0}^{K} \| r_i(j) - x_i(j) \|$$

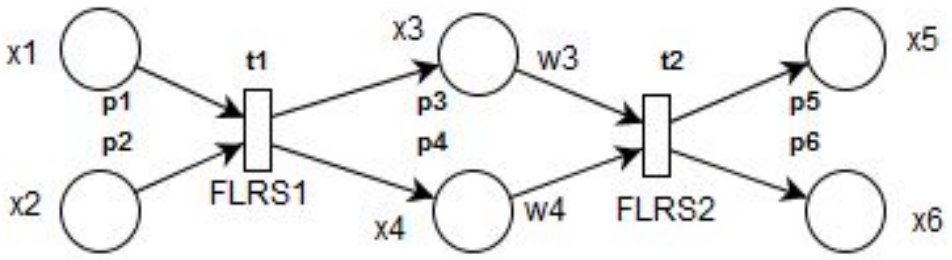
Criteriul de oprire: $J < \varepsilon_t$ ➔ {FLRS, W}

➔ o soluție ***acceptabilă***, nu o soluție ***optimă*** !

$c_{1,1}$	$c_{1,2}$	...	$c_{4,5}$	$c_{5,5}$	w_{11}	w_{21}
000010	010100	...	100100	001100	2, 35	−3, 28

Exemplu: Compararea a două valori

Folosim FLETPN pentru a împărți sau nu execuția programului în funcție de rezultatul comparării (ε este o valoare mică specificată).



Cerința: construiți modelul pentru care:

if $(x_1 - x_2) > \varepsilon$ then $x_5 = 1$ and x_6 is Φ
if $(x_2 - x_1) > \varepsilon$ then $x_6 = -1$ and x_5 is Φ
if $|x_1 - x_2| \leq \varepsilon$ then $x_5 \neq 1$ and $x_6 \neq -1$

x_i is Φ nu setează jetonul!
Concluzie: Dacă $|x_1 - x_2| > \varepsilon$ atunci după execuția tranziției numai una din locațiile p_5 sau p_6 va avea jeton.
$w_3 = w_4 \geq 1/\varepsilon$.

Octavian Cuibus: Sisteme de control distribuit: Specificare, verificare, proiectare

FLETPN executor

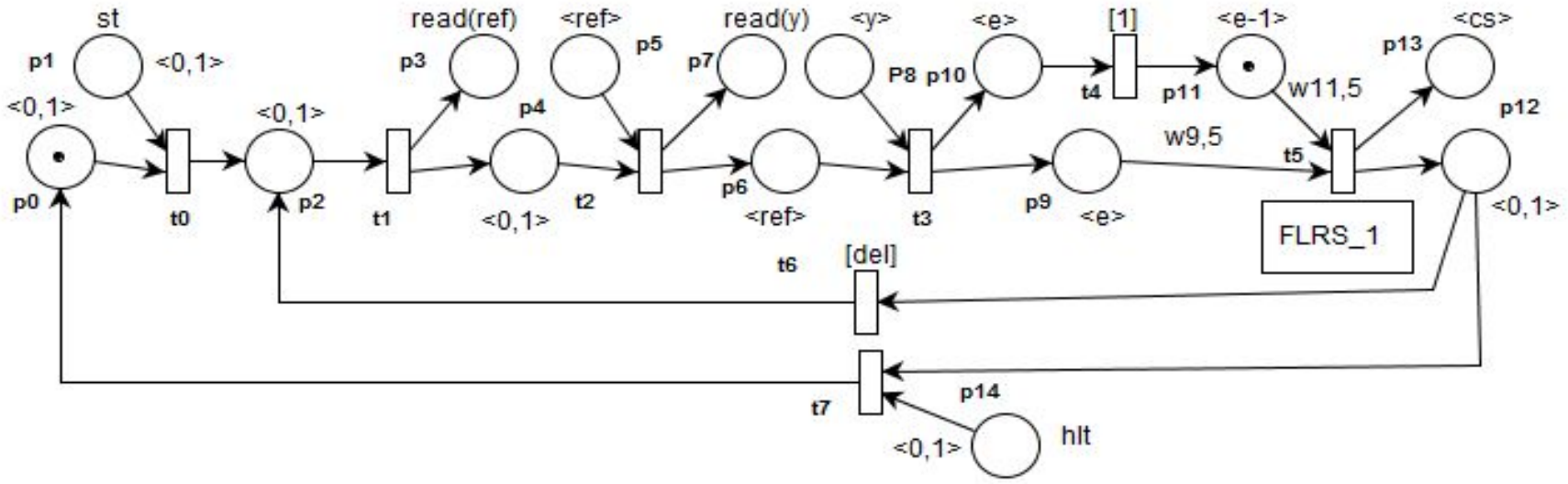
- $\mathbf{M_t}$ - vector de marcaj temporar
- *execList* - lista tranzițiilor executabile la momentul curent
- *Inp* - setul de locații de intrare
- *Out* - setul de locații de ieșire
- $FLRS_i$ - setul de reguli fuzzy pentru tranziția t_i
- ${}^{o}t_i$ - locațiile de intrare pentru t_i
- t^{o}_i - locațiile de ieșire pentru t_i
- $X_{i,out}$ valorile variabilelor care trebuie transmise la ieșire

FLETPN algoritmul de execuție:

1 *Input:* **Pre**, **Post**, X, M_0, P, T, D, FLRS, Inp, Out;
2 *Initialization:* M = M_0, *execList* = *empty*; $\mathbf{M_t = 0}$;
3 * ordonăm setul de tranziţii T după întârzierile lor;
4 **repeat**
5 *wait(event);*
6 **if** *event* is *tic* **then**
7 * descreştem *Delays* ale tranziţiilor în *execList*;
8 **else**
9 *receive(Inp, X_{inp})*
10 * *update* **M**;
11 **endif;**
12 **repeat**
13 **for** all $t_i \in T$ **do**
14 **if** $FLRS_i$ conţine cel puţin o regulă care poate fi activată cu marcajul curent
15 **then** * add t_i to *execList*;
16 * move the tokens of ${}^{o}t_i$ from **M** to $\mathbf{M_t}$;
17 *Delay[t_i] = d_i*; // actualizăm valoarea lui d_i dacă este necesar!
18 **endif;**
19 **endfor;**
20 **for** all t_i in *execList* **do**

21 **if** (*Delay[t_i]* is 0) **then**
22 * remove t_i from *execList*;
23 * calculate and set the tokens in **M** for all t^{o}_{i};
24 * remove the tokens from $\mathbf{M_t}$ for all $^{o}t_i$;
25 **endif;**
26 **if** $t_i \in$ Out **then**
27 *send(Out, $X_{i,out}$)*
28 **endif;**
29 **endfor;**
30 **until** nici o altă tranziţie nu mai poate fi executată;
31 **until** orizontul de timp;

Exemplu – controler PI cu FLETPN

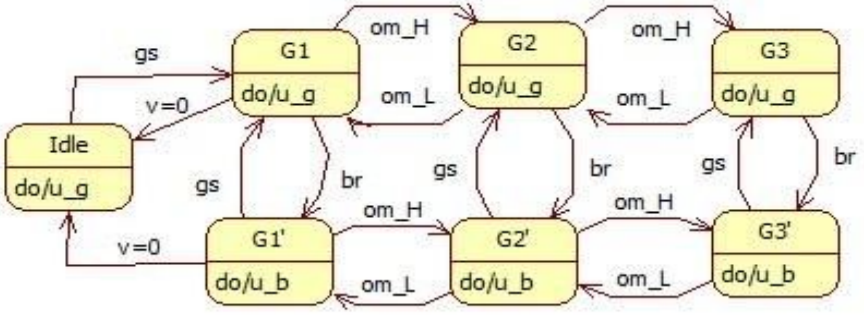


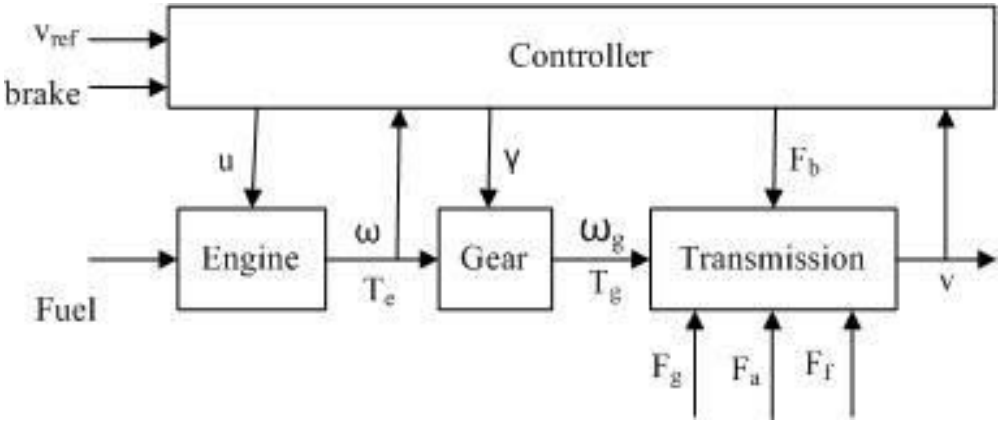
x_1^x_2	NL	NM	ZR	PM	PL
NL	NL,NL	NL,PL	NL,PM	NL,PL	NL,PM
NM	NL,M	NM,PM	PM,PM	NM,PM	NM,PM
ZR	ZR,PM	ZR,PL	PM,PL	PM,PL	PM,PL
PM	NL,PL	NL,PM	NL,PL	NL,PM	NL,PL
PL	PM,PM	PM,PM	PM,PM	PL,PM	PL,PL

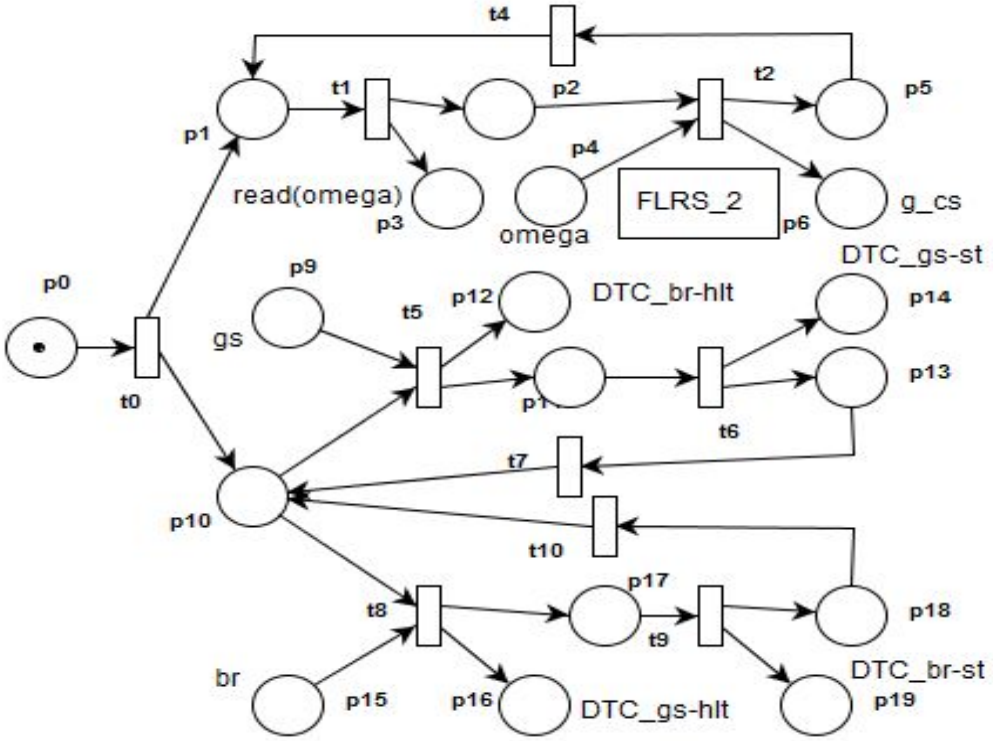
- *În același grafic: dinamica părții continue și jetoanele - evenimente discrete*
- *O tranziție implică un set de reguli fuzzy*

$$e = ref - y$$
$$e\_1[k] = e[k-1]$$
$$cs[k] = w_{9,5} \cdot e[k] + w_{11,5} \cdot e[k-1]$$

Exemplu: Controlul independent al unui vehicul







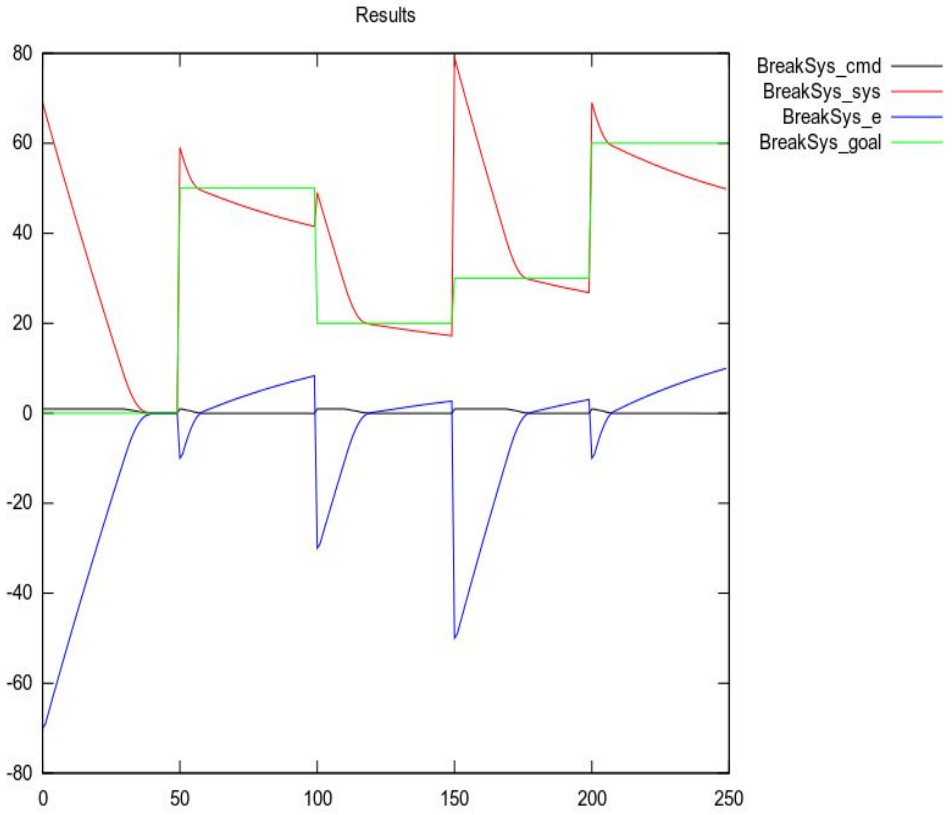
Inference rules for the gas and break cases

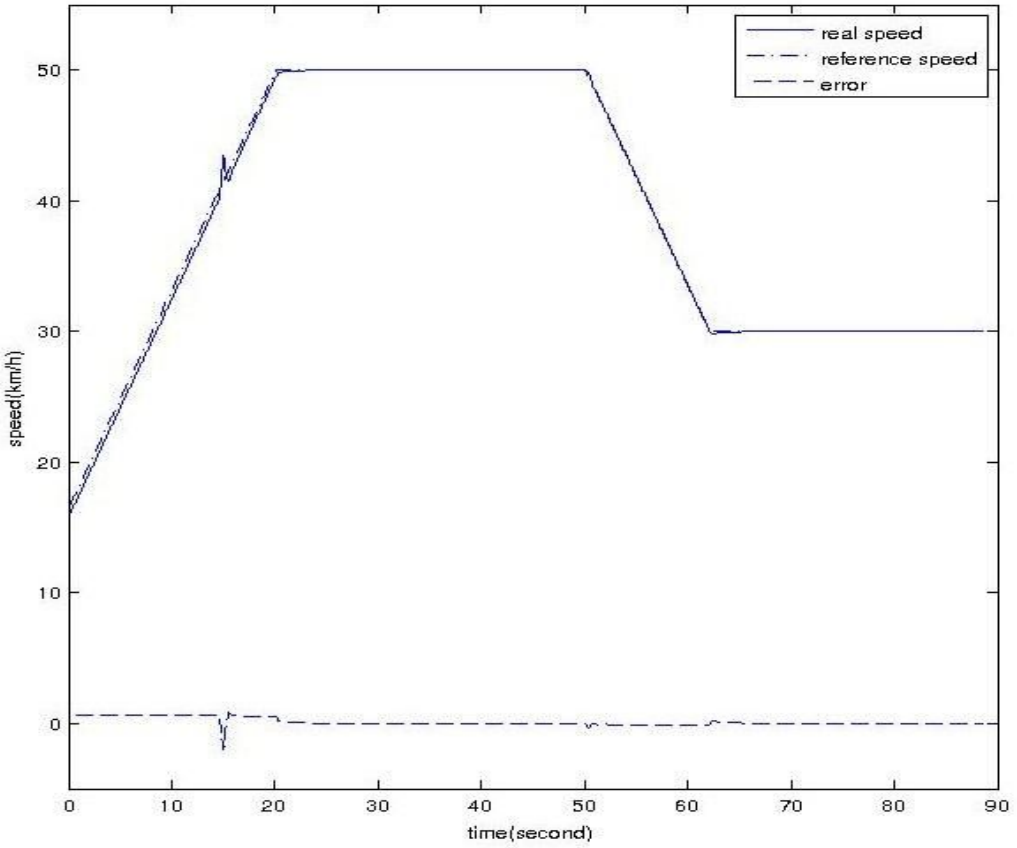
$x_1 \backslash x_2$	NL	NM	ZR	PM	PL
NL	NL,PL	NL,PL	NL,PL	ZR,PL	PL,PL
NM	NM,PL	PL,PL	PL,PL	PL,PL	NL,PL
ZR	PM,PL	PM,PL	ZR,PL	NL,PL	PM,PL
PM	ZR,PL	PL,PL	NL,PL	NL,PL	NM,PL
PM	NL,PL	ZR,PL	ZR,PL	NL,PL	ZR,PL

$x_1 \backslash x_2$	NL	NM	ZR	PM	PL
NL	ZR,PL	PM,PL	ZR,PL	NM,PL	NM,PL
NM	PM,PL	NM,PL	ZR,PL	NL,PL	PM,PL
ZR	NM,PL	PM,PL	ZR,PL	ZR,PL	PL,PL
PM	PM,PL	ZR,PL	PM,PL	ZR,PL	PM,PL
PM	PM,PL	NM,PL	ZR,PL	PL,PL	ZR,PL

Setul FLRSs și **W** pot fi obținute folosind algoritmi genetici.

Scale factor = ?





4.7 Specificarea și proiectarea cu componente

UVTC = Urban Vehicle Traffic Control (Controlul urban al traficului vehiculelor)

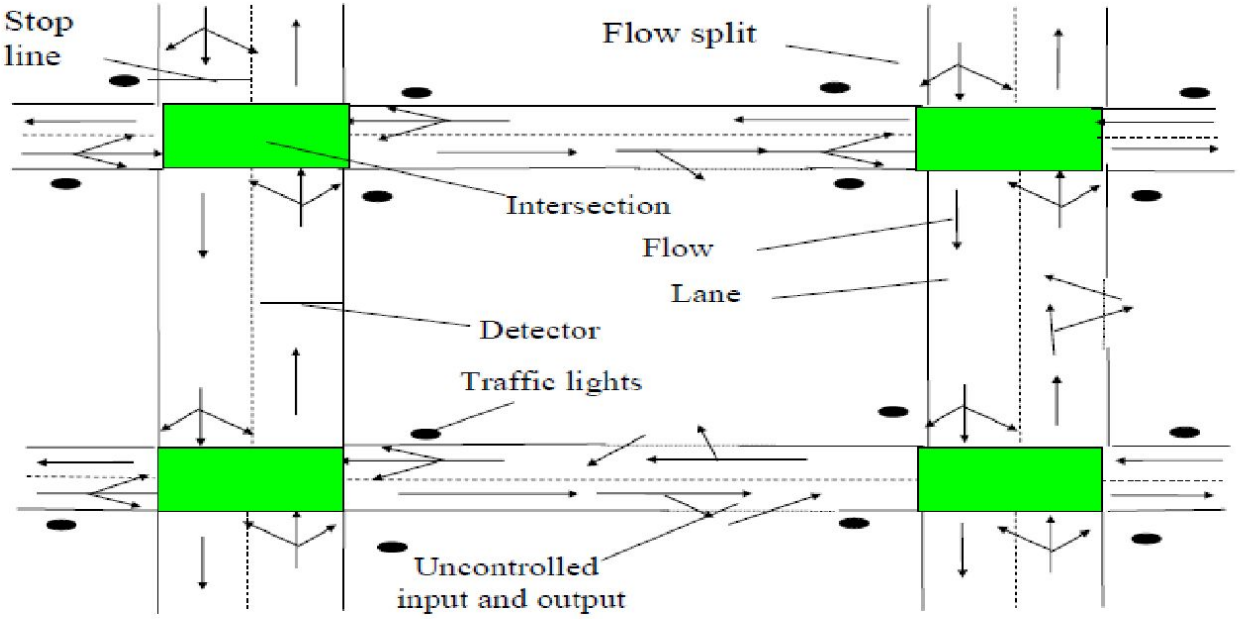
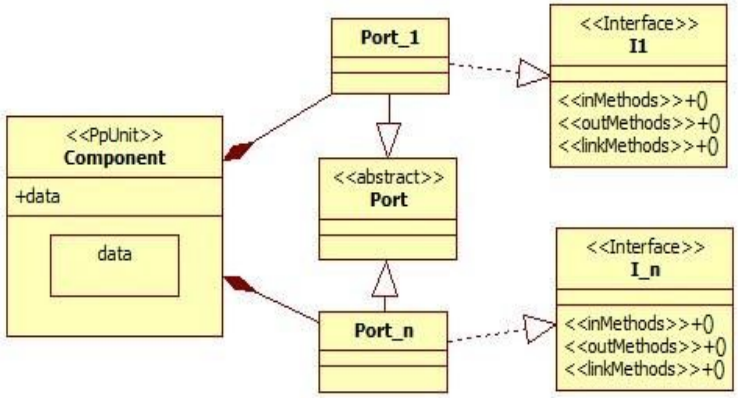
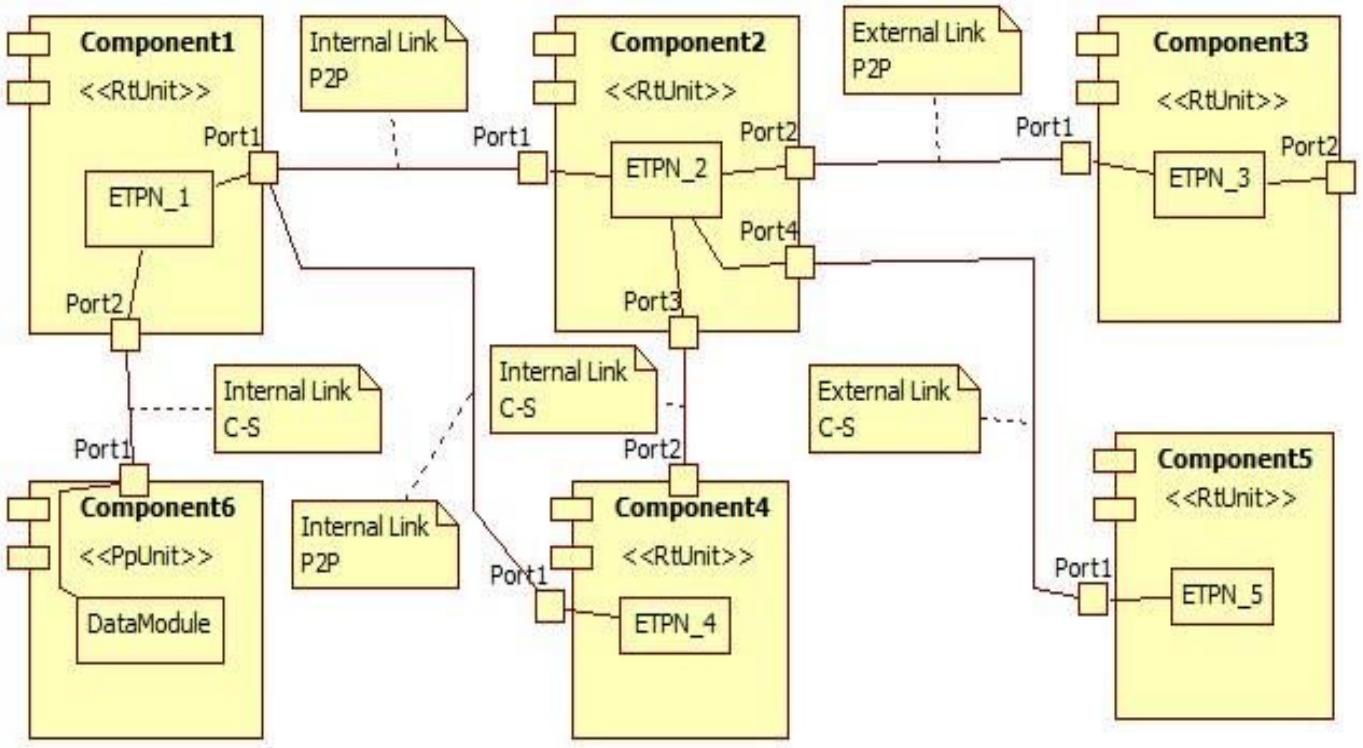


Fig. 3.14. Four crossroad system of UVT

Diagrama de componente: porturi



Componente cu ETPN



Diagrame de componente cu FLETPN

Dezvoltare bazată pe model

Proiectare bazată pe componente

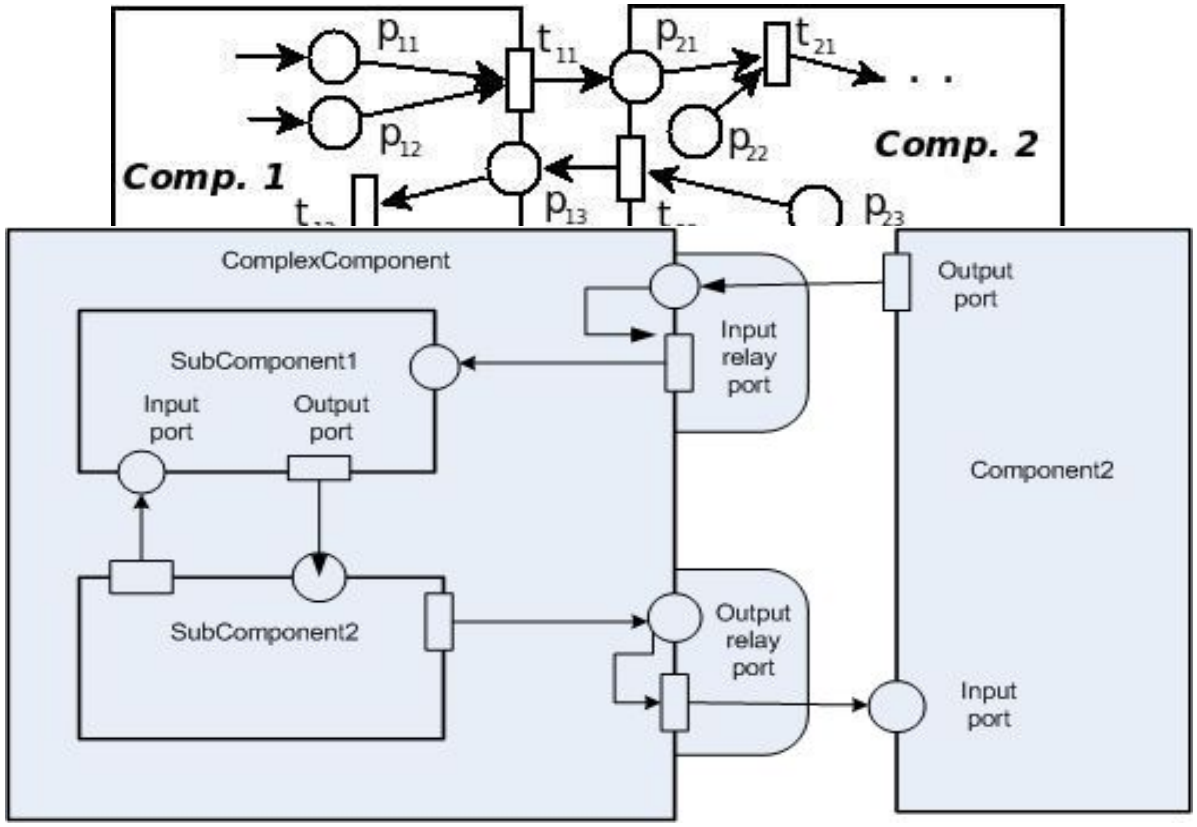
Fiecare componentă are pe graniță:

- porturi (locații) de intrare
- porturi (tranziții) de ieșire

Fiecare componentă este o FLETPN

Fiecare FLETPN are executorul său

Fiecare executor este implementat printr-un fir de execuție.

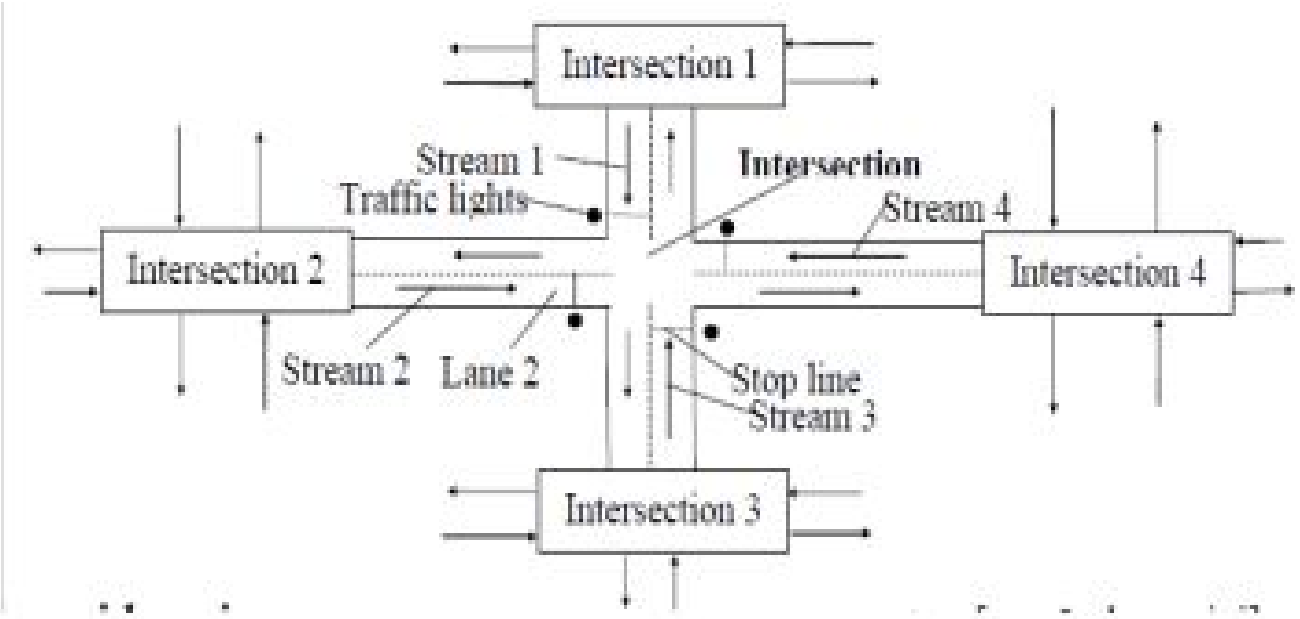


Componentele complexe includ alte componente. Ele au: ***porturi de intrare şi ieşire***.

Exemplu de aplicaţie 1
Traficul urban al vehiculelor: controlul independent al unei intersecţii

Dezvoltare:

a) Specificare
b) sitenza controlerului
c) proiectare software
d) verificare
e) implementare
f) testare
g) integrare



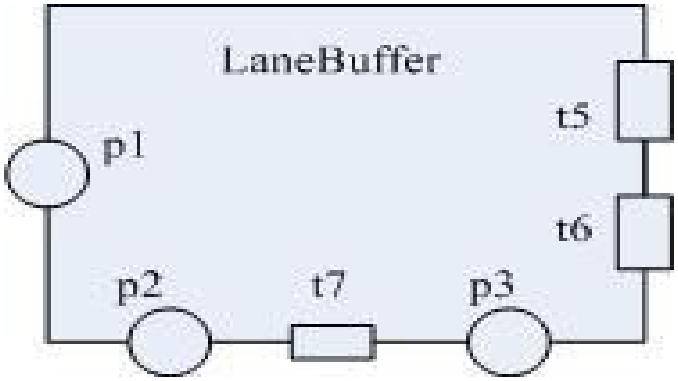
a) Specificare. Specificarea sistemului

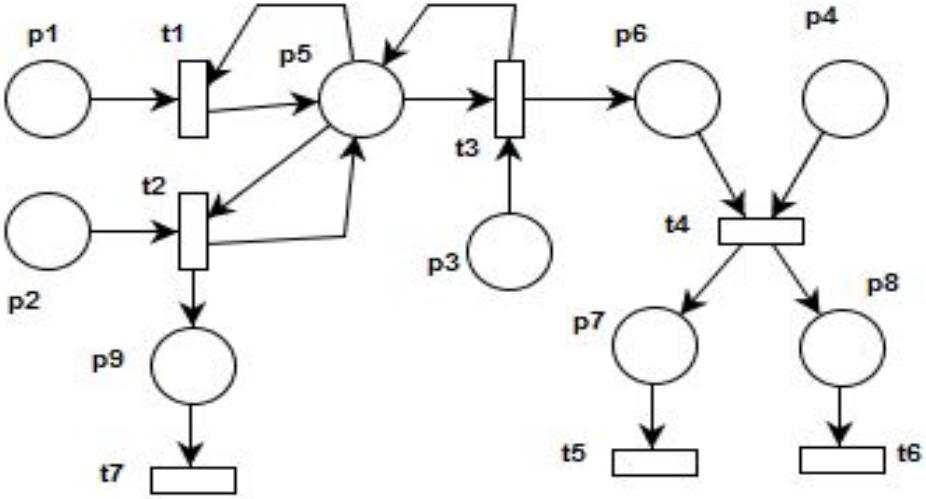
Specificare textuală sau formală. Model FLETPN de tip buffer

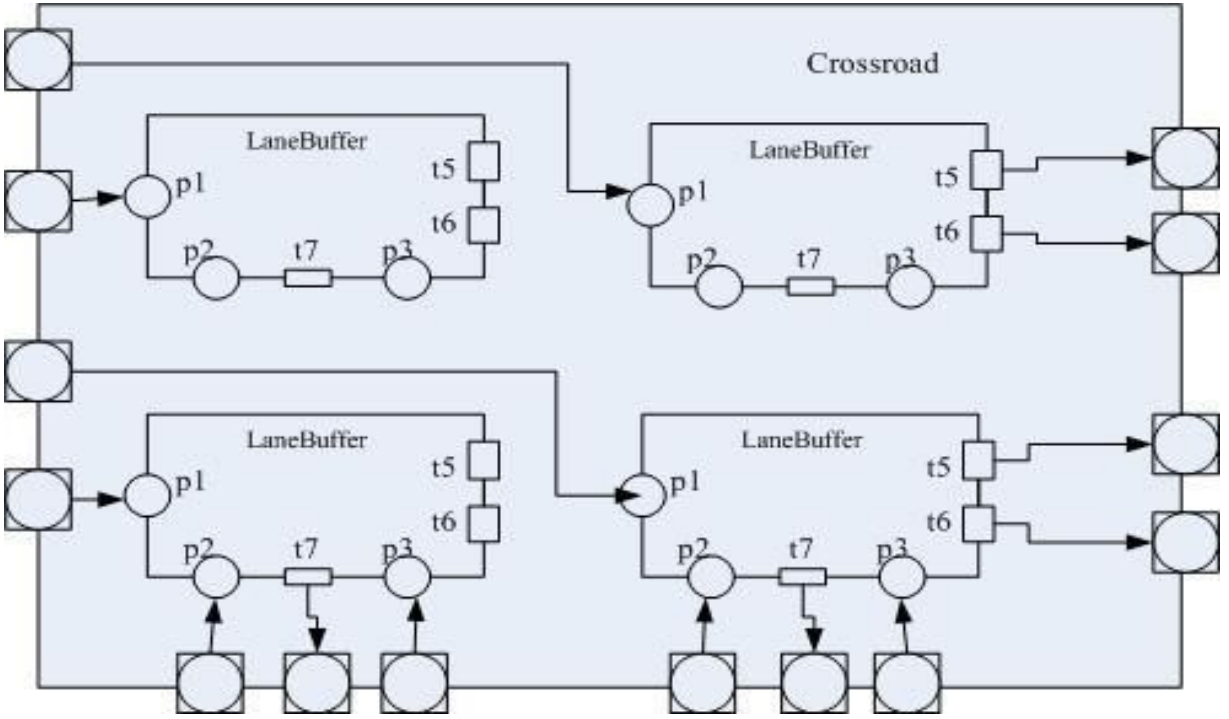
Locaţie	Ce înseamnă	Variabilă	Domeniu în model	Domeniu în realitate	Factor de scalare
p_1	lungimea cozii de intrare	x_1	[0,1]	[0,50]	50
p_2	citirea evenimentului (semnal)	x_2	Φ, 1	0,1	1
p_3	durata fazei 1	x_3	[0,1]	[0,50] sec.	50
p_4	impărţirea fluxului la ieşire (flow split) L_1/L_2	x_4	[0,1]	[0,100] %	100
p_5	lungimea cozii pe bandă	x_5	[0,1]	[0, 50] veh,	50
p_6	numărul de vehicule care ies	x_6	[0,1]	[0,50] veh.	50
p7	nr vehicule care ies pe banda 1	x_7	[0,1]	[0,50] veh.	50
p_8	nr vehicule care ies pe banda 2	x_8	[0,1]	[0,50] veh.	50
p_9	lungimea cozii	x_9	[0,1]	[0, 50] veh,	50

Tranziţia	Ce înseamnă	Întârziere	FLRS
t_1	actualizează lungimea cozii	0	-
t_2	citeşte lungimea cozii la comanda controlerului	0	-
t_3	actualizează lungimea cozii când faza este deschisă	0	-
t_4	modelează împărţirea fluxului la ieşire	0	$FLRS_4$
t_5	Trimite lungimea cozii spre ieşirea 1	0	-
t_6	Trimite lungimea cozii spre ieşirea 2	0	-
t_7	Trimite lungimea actuală a cozii	0	-

Verificarea modelului
Testarea implementării modelului



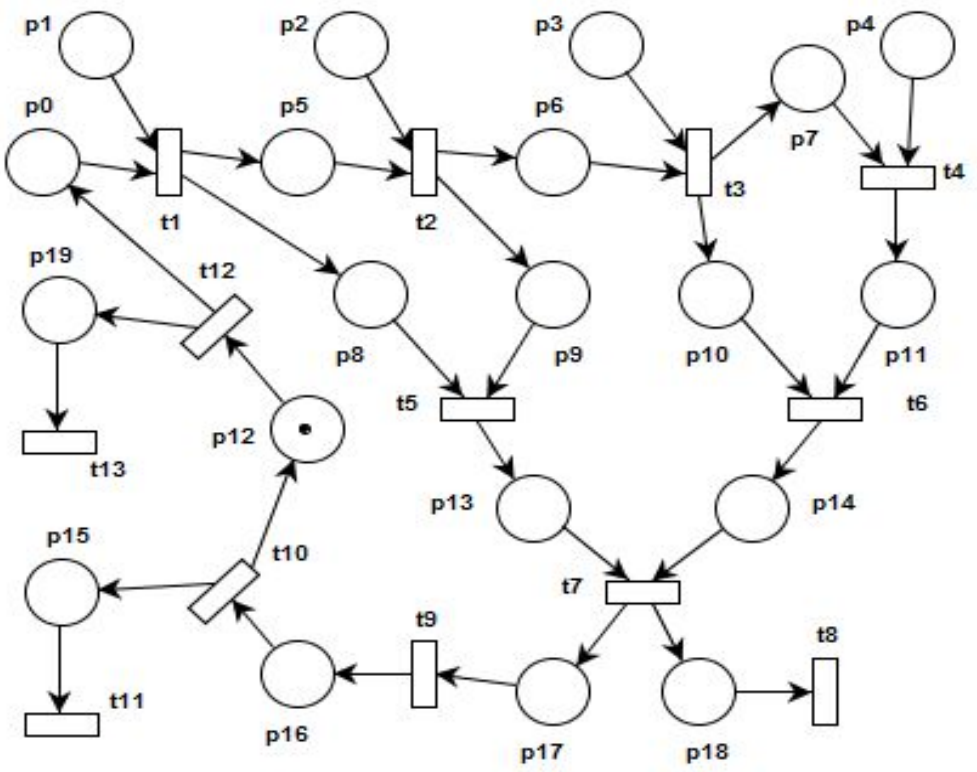




Specificarea controlerului

Citeşte lungimea cozii q_i a benzii *i* (de tip buffer). Controlerul intersecţiei calculează duratele fazeor (ph_1 and ph_2) pentru fiecare ciclu conform cu lungimea cozii. Durata ciclului este 60 sec.
Durata fazei galbene este 5 sec.
$ph_2 = 50 - ph_1;\ \ 10 \leq ph_1 \leq 40\ sec$
Notaţii:

- dem_1 este cererea (demand) pentru faza 1 (ph_1)
- dem_2 este cererea (demand) pentru faza 2 (ph_2)
- q_i este lungimea cozii *i*
- *IF* $(q_1\ is\ H)\ \hat{}\ (q_2\ is\ L)\ THEN\ (dem_1\ is\ H)$ se notează pe scurt astfel: $q_1 \epsilon H\ \hat{}\ q_2 \epsilon L \rightarrow\ dem_1 \epsilon H;$



The experts say:
$q_1 \epsilon H\ \check{}\ q_2 \epsilon H \rightarrow\ dem_1 \epsilon H;$
$q_1 \epsilon M\ \hat{}\ q_2 \epsilon M \rightarrow\ dem_1 \epsilon M;$
$q_1 \epsilon L\ \hat{}\ q_2 \epsilon L \rightarrow\ dem_1 \epsilon L;$
$q_1 \epsilon M\ \hat{}\ q_2 \epsilon L \rightarrow\ dem_1 \epsilon M;$
$q_1 \epsilon L\ \hat{}\ q_2 \epsilon M \rightarrow\ dem_1 \epsilon M;$

$dem_1 \epsilon M\ \hat{}\ dem_2 \epsilon M \rightarrow\ ph_1 \epsilon M;$
$dem_1 \epsilon L\ \hat{}\ dem_2 \epsilon L \rightarrow\ ph_1 \epsilon M;$
$dem_1 \epsilon L\ \hat{}\ dem_2 \epsilon M \rightarrow\ ph_1 \epsilon L;$

Octavian Cuibus: Sisteme de control distribuit: Specificare, verificare, proiectare

$$dem_1 \epsilon H \ \hat{} \ dem_2 \epsilon H \rightarrow \ ph_1 \epsilon M;$$

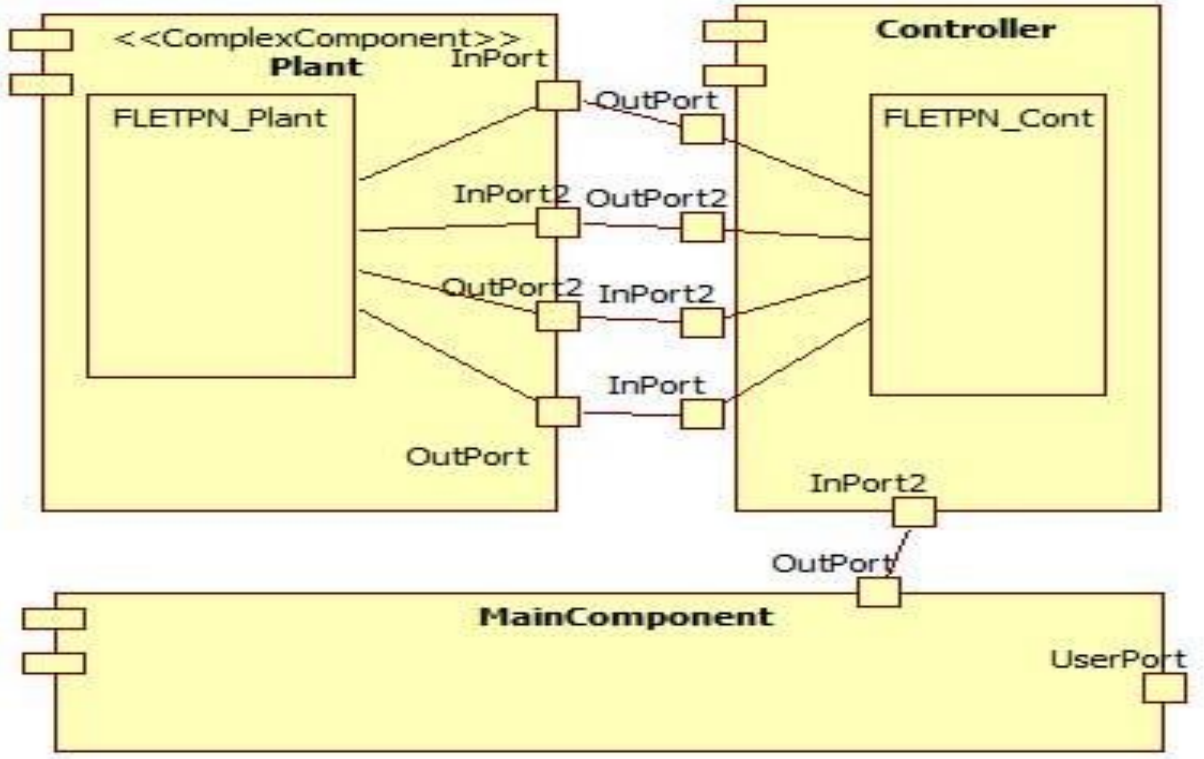
Locaţie	Ce înesamnă	Variabila	Domeniul modelului	Domeniul în realitate	Factor de scalare
p_0	începutul ciclului	x_0	1	1	1
p_1	lungimea cozii 1	x_1	[0,1]	[0,50]	50
p_2	lungimea cozii 2	x_2	[0,1]	[0,50]	50
p_3	lungimea cozii 3	x_3	[0,1]	[0,50]	50
p_4	lungimea cozii 4	x_4	[0,1]	[0,50]	50
p_5	următoarea stare	x_5	1	1	1
p_6	următoarea stare	x_6	1	1	1
p7	următoarea stare	x_7	1	1	1
p_8	stochează lungimea cozii 1	x_8	[0,1]	[0,50]	50
p_9	stochează lungimea cozii 2	x_9	[0,1]	[0,50]	50
p_{10}	stochează lungimea cozii 3	x_{10}	[0,1]	[0,50]	50
p_{11}	stochează lungimea cozii 4	x_{11}	[0,1]	[0,50]	50
p_{12}	stochează durata fazei 2	x_{12}	[0,1]	[0,50]	50
p_{13}	Cererea pentru faza 1 (dem_1)	x_{13}	[0,1]	[0,50]	50
p_{14}	Cererea pentru faza 2 (dem_2)	x_{14}	[0,1]	[0,50]	50
p_{15}	lungimea fazei 1	x_{15}	[0,1]	[0,50]	50
p_{16}	lungimea fazei 2	x_{16}	[0,1]	[0,50]	50
p_{17}	lungimea fazei 3	x_{17}	[0,1]	[0,50]	50
p_{18}	lungimea fazei 4	x_{18}	[0,1]	[0,50]	50
p_{19}	Marked at the new cycle	x_{19}	1	1	1

Tranziţia	Ce înseamnă	Întârziere	FLRS
t_{12}	Cererea de a citi lungimea cozilor	x_{12}	-
t_1	Citeşte q_1	5	-
t_2	Citeşte q_2	0	-
t_3	Citeşte q_3	0	-
t_4	Citeşte q_4	0	-
t_5	Calculează dem_1	0	$FLRS_5$
t_6	Calculează dem_2	0	$FLRS_6$
t_7	Calculează lungimile fazelor	0	$FLRS_7$
t_8	Trimite durata fazei 1	0	-
t_9	Asteaptă durata fazei 1	x_{17}	

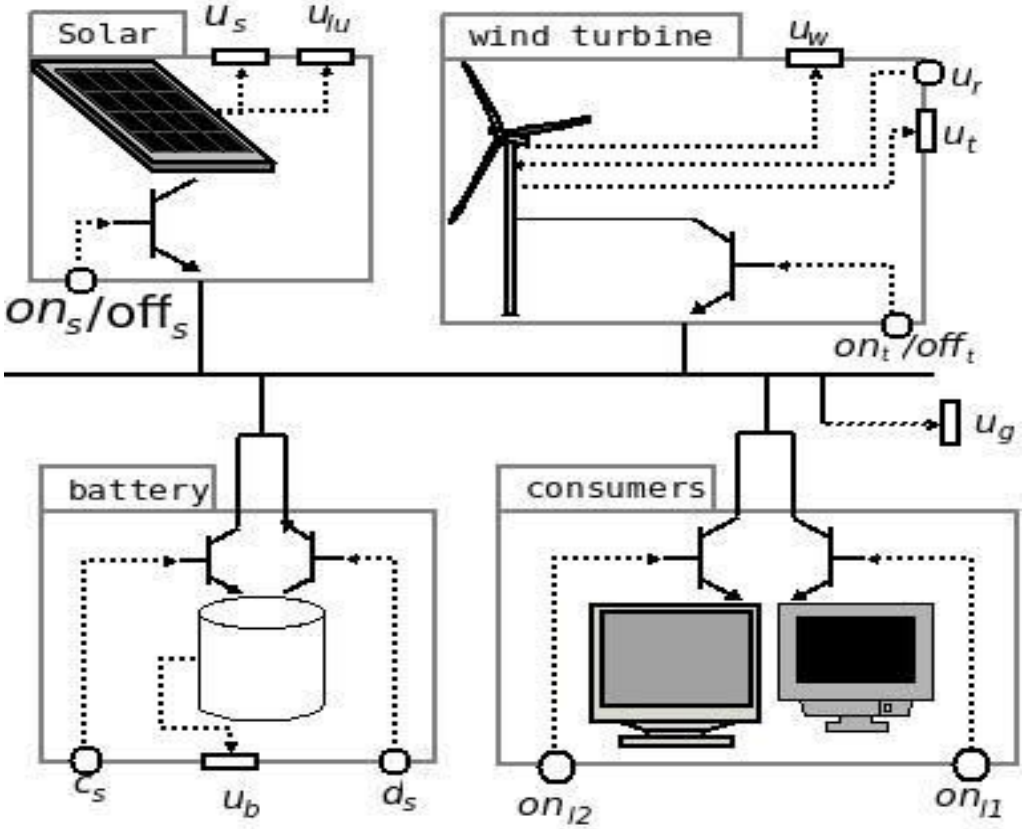
t_{10}	Calculează durata fazei 2	5	$FLRS_{10}$
t_{11}	Trimite durata fazei 2	0	-

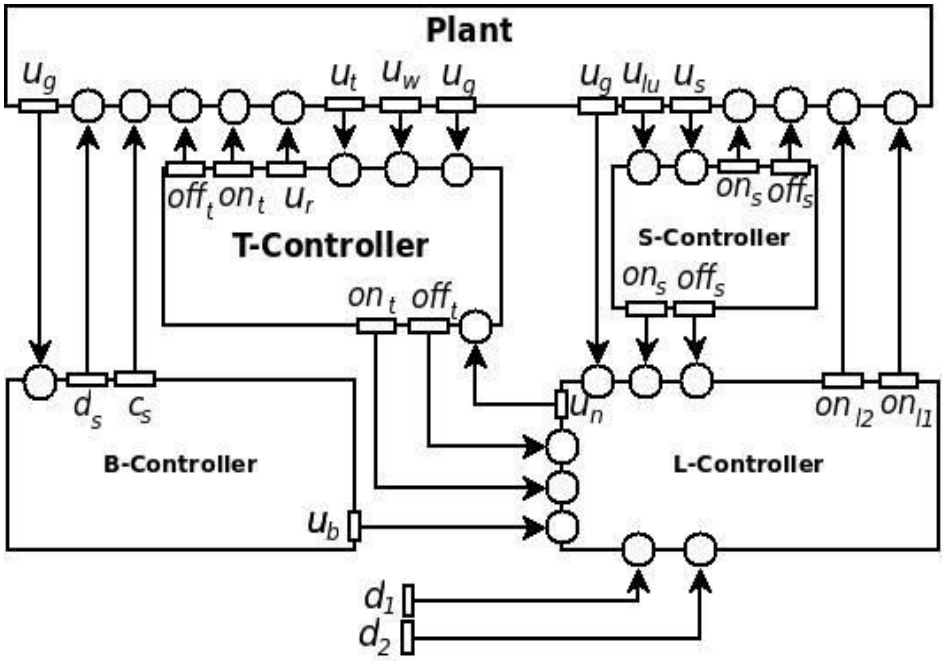
Verificarea modelului = verificarea controlului + verificarea concurenței + verificarea logicii.

Testarea implementării modelului



Exemplu de aplicaţie:
Mini-sisteme de putere electrică





T-Controler (turbină)

w_1	w_2	w_3	w_4	w_5	w_6
-0.1829	-2.5079	-7.7209	0.1332	4.7337	0.0076

Fuzzy logic rules of transition t_4 in Fig. 9

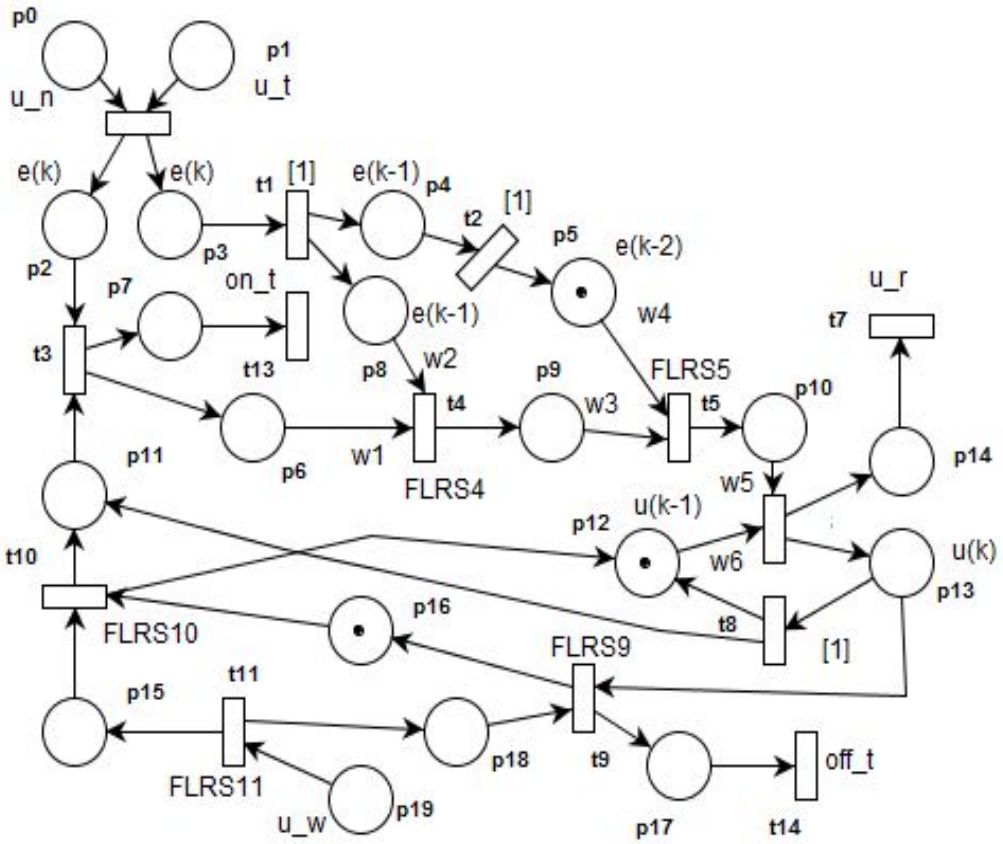
$x_6 \backslash x_8$	NL	NM	ZR	PM	PL
NL	NM,PM	PM,ZR	NL,NM	NL,ZR	NL,NM
NM	NL,ZR	NM,NL	NM,ZR	ZR,ZR	PL,PM
ZR	NL.PM	NL,NM	NL,PM	NL,NL	NL,ZR
PM	ZR,NL	PM,NM	ZR,NM	ZR,PM	PL,PM
PL	PM,PM	NM,ZR	NL,NM	PM,NL	NM,NM

Fuzzy logic rules of transition t_5 in Fig. 9

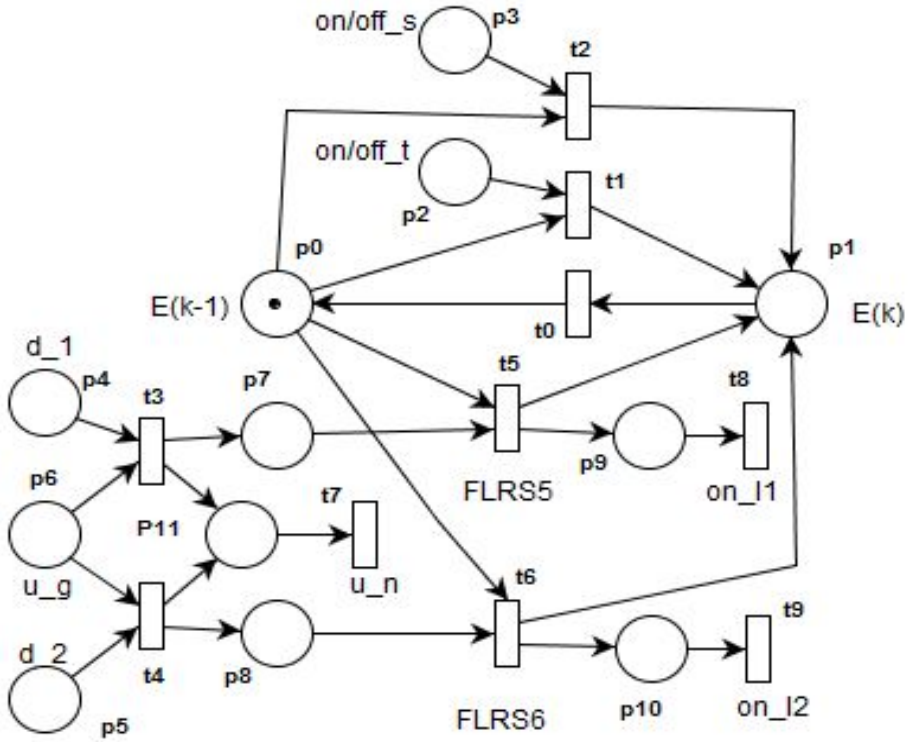
$x_9 \backslash x_5$	NL	NM	ZR	PM	PL
NL	PM,ZR	PL,ZR	ZR,PL	ZR,NM	PM,PL
NM	PL,NL	NM,ZR	NM,PM	NL,PL	ZR,PM
ZR	PL,NM	PL,NL	NM,PL	NL,NL	PM,NL
PM	PM,PM	NL,NM	PM,NM	PM,NM	PL,ZR
PL	NL,NL	NL,PM	ZR,ZR	PL,ZR	NL,ZR

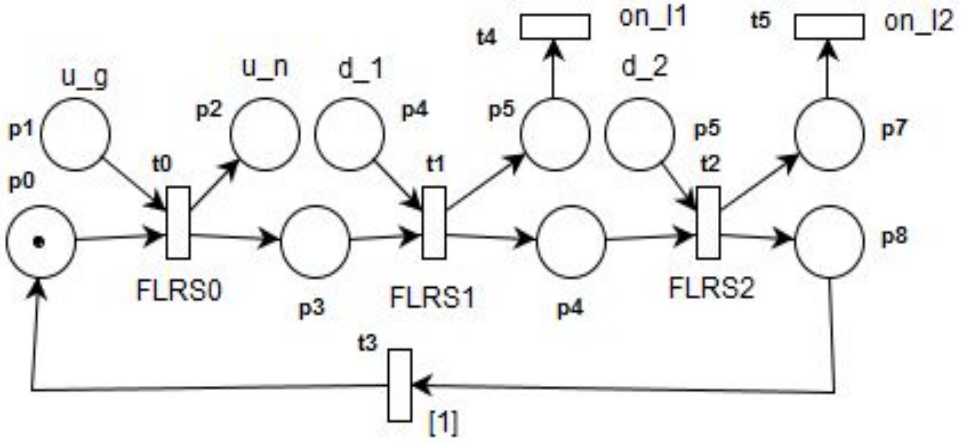
Fuzzy logic rules of transition t_{10} in Fig. 9

$x_{15} \backslash x_{16}$	NL	NM	ZR	PM	PL
NL	Φ, Φ	Φ, Φ	Φ, Φ	ZR,PM	ZR,PL
NM	Φ, Φ	Φ, Φ	Φ, Φ	ZR,PM	ZR,PL
ZR	Φ, Φ	Φ, Φ	Φ, Φ	ZR,PM	ZR,PL
PM	Φ, Φ	Φ, Φ	Φ, Φ	ZR,PM	ZR,PL
PL	Φ, Φ	Φ, Φ	Φ, Φ	ZR,PM	ZR,PL



L-Controlere (load, sarcină)





Fuzzy logic rules of transitions t_5 and t_6 in Fig. 11

$x_0 \backslash x_5, x_6$	NL	NM	ZR	PM	PL
NL	NL, Φ	NM, Φ	ZR, Φ	PM, Φ	PL, Φ
NM	NL, Φ	NM, Φ	ZR, Φ	PM, Φ	PL, Φ
ZR	NL, Φ	NM, Φ	ZR, Φ	PM, Φ	PL, Φ
PM	NL, Φ	NM, Φ	ZR, Φ	ZR, ZR	PM, ZR
PL	NL, Φ	NM, Φ	ZR, Φ	ZR, ZR	PM, ZR

Verificarea modelului vs. Testare

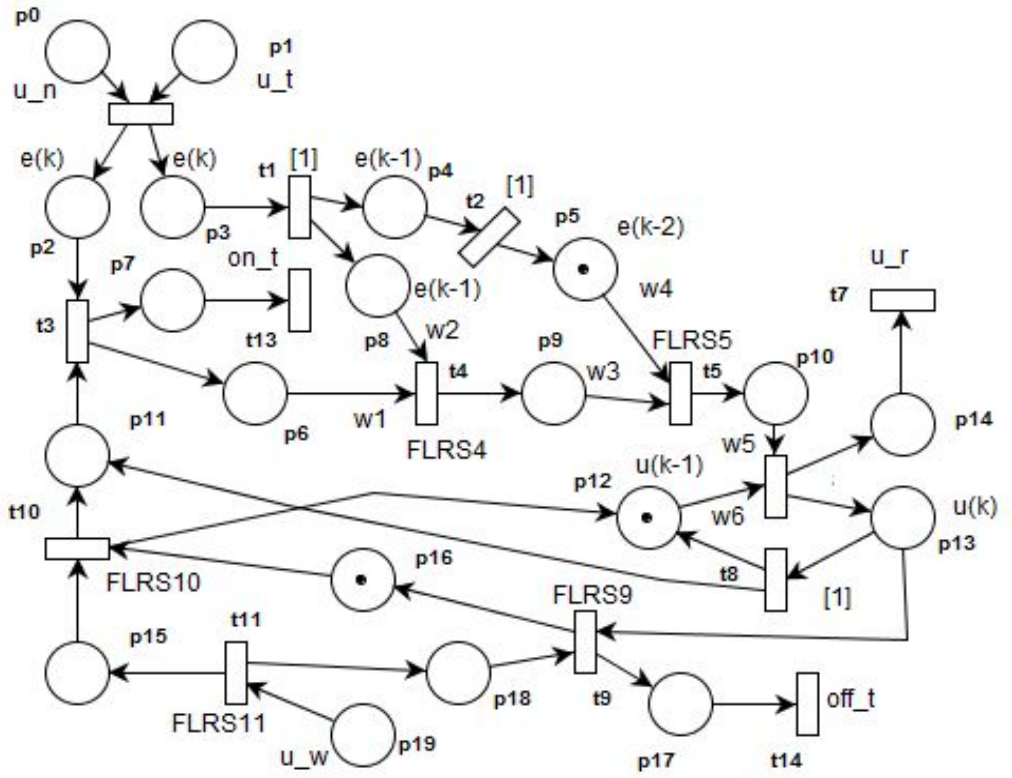
Testare:

- număr de teste NT< 1000
- număr de intrări şi ieşiri?

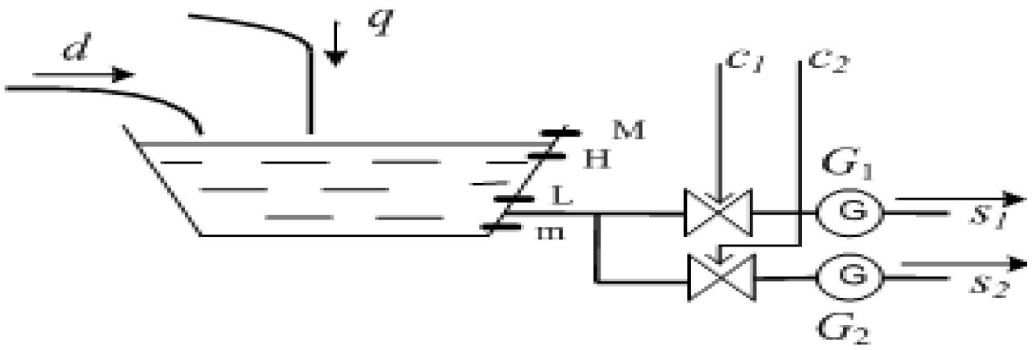
$$Test\ \mathrm{Re}liability = \frac{NT}{\infty * \infty} = 0\ \%$$

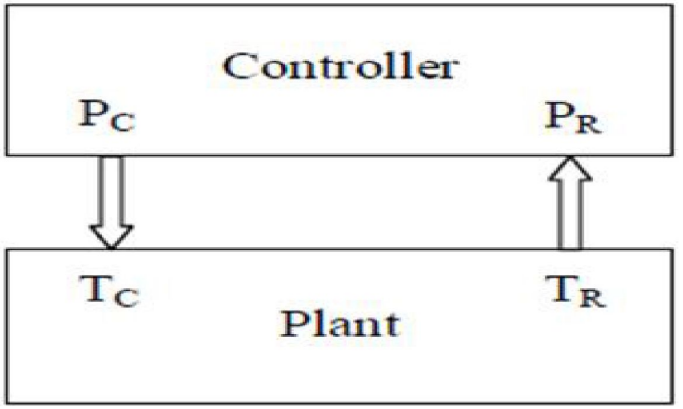
Cazuri limită (corner cases) =

situaţii critice



Specificarea controlului pentru o hidrocentrală

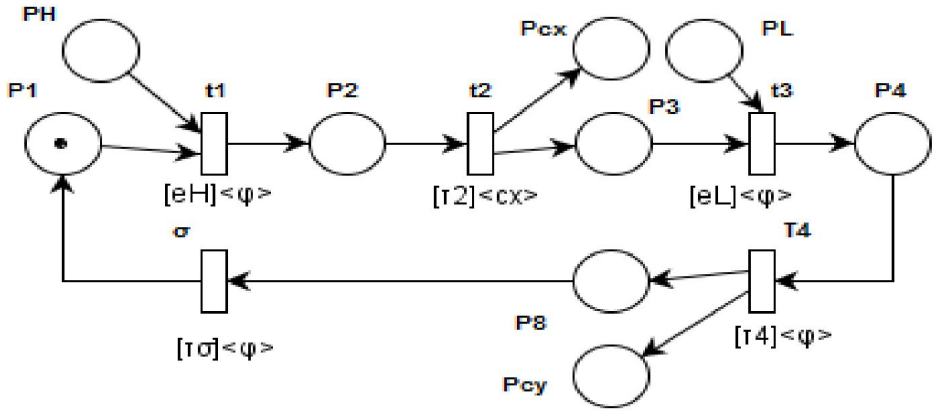




Model TPN al controlerului

Temă de casă: construiți:

- diagrama componentelor pentru 2 controlere cooperative
- adăugați rețeaua ETPN pentru controler
- adăugați funcțiile de control continuu

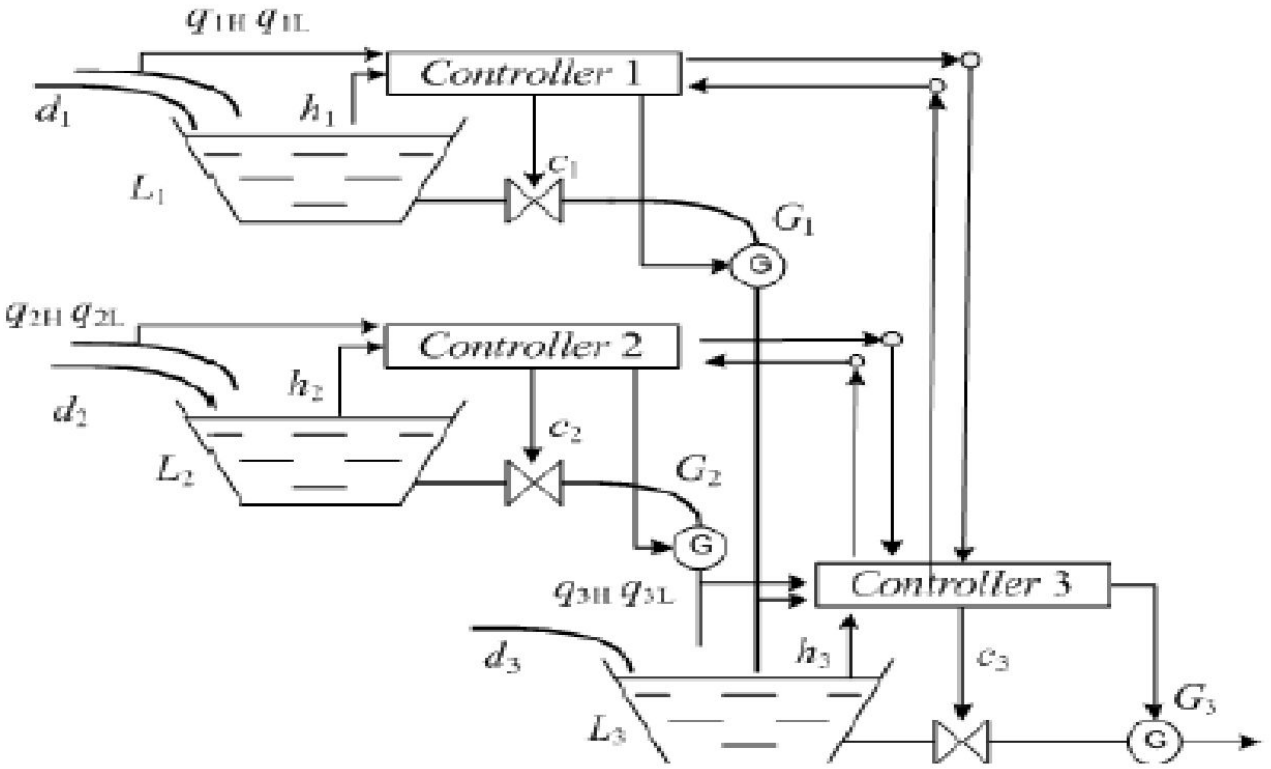


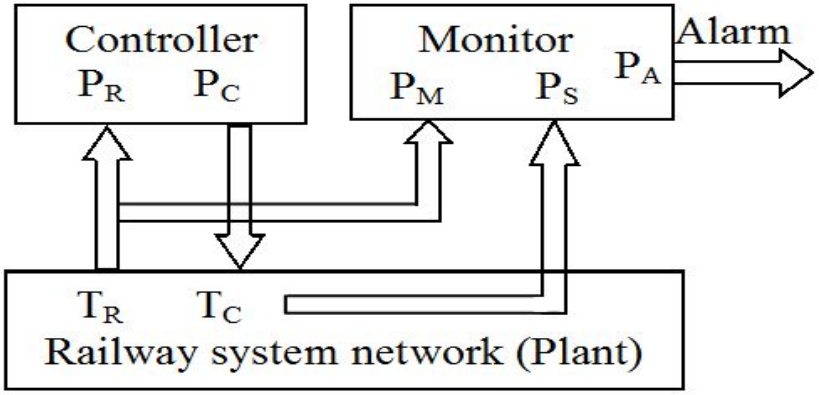
Definiți interfețele sistemului!
Interfețe sunt între:

- sistem și controler
- controler si controler

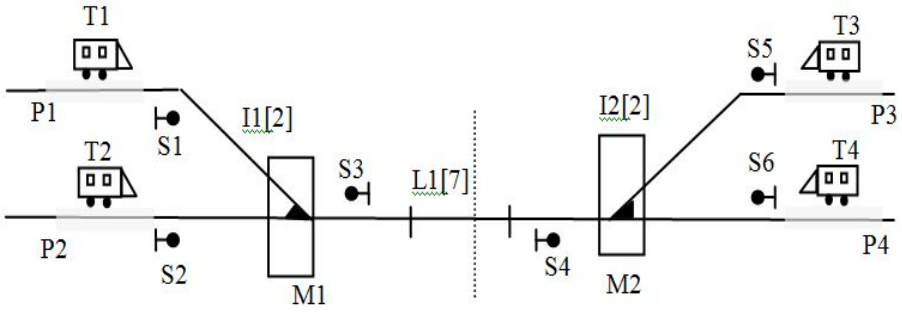
Temă de casă: Concepeți

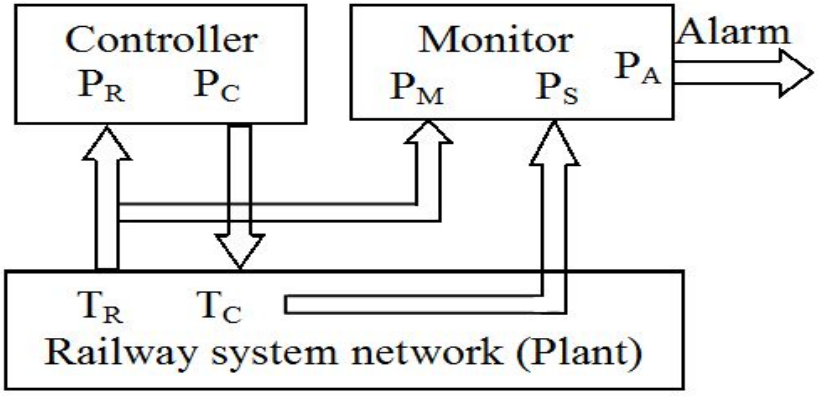
- diagrama componentelor pentru 2 controlere cooperative
- adăugați rețeaua ETPN pentru controler
- adăugați funcțiile de control continuu





Sistem de control şi monitorizare pentru traficul feroviar





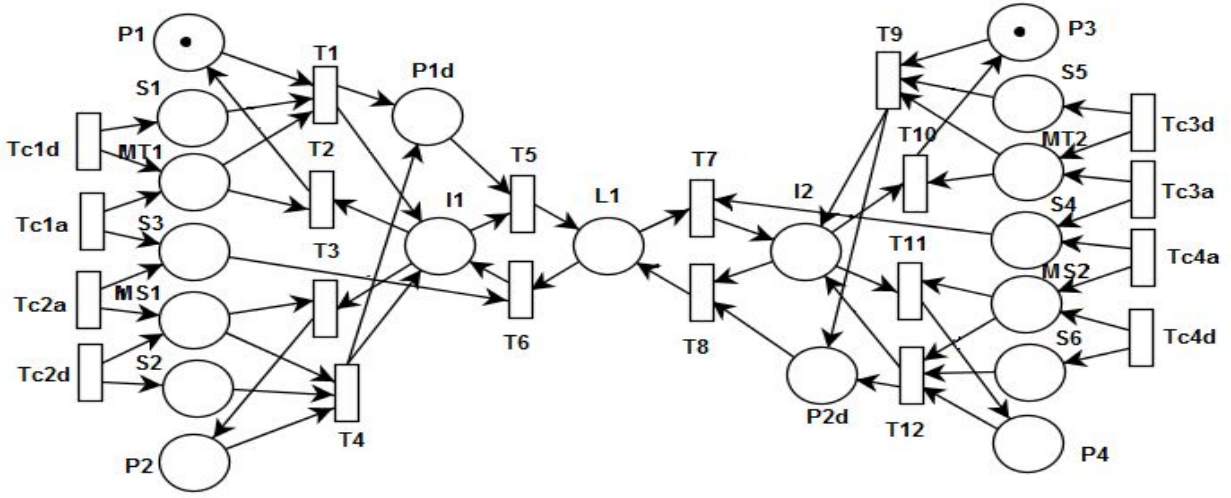
Definiți interfețele sistemului!

Avem interfețe între:

- sistem și controler
- controler și sistemul de monitorizare

Temă de casă: Concepeți

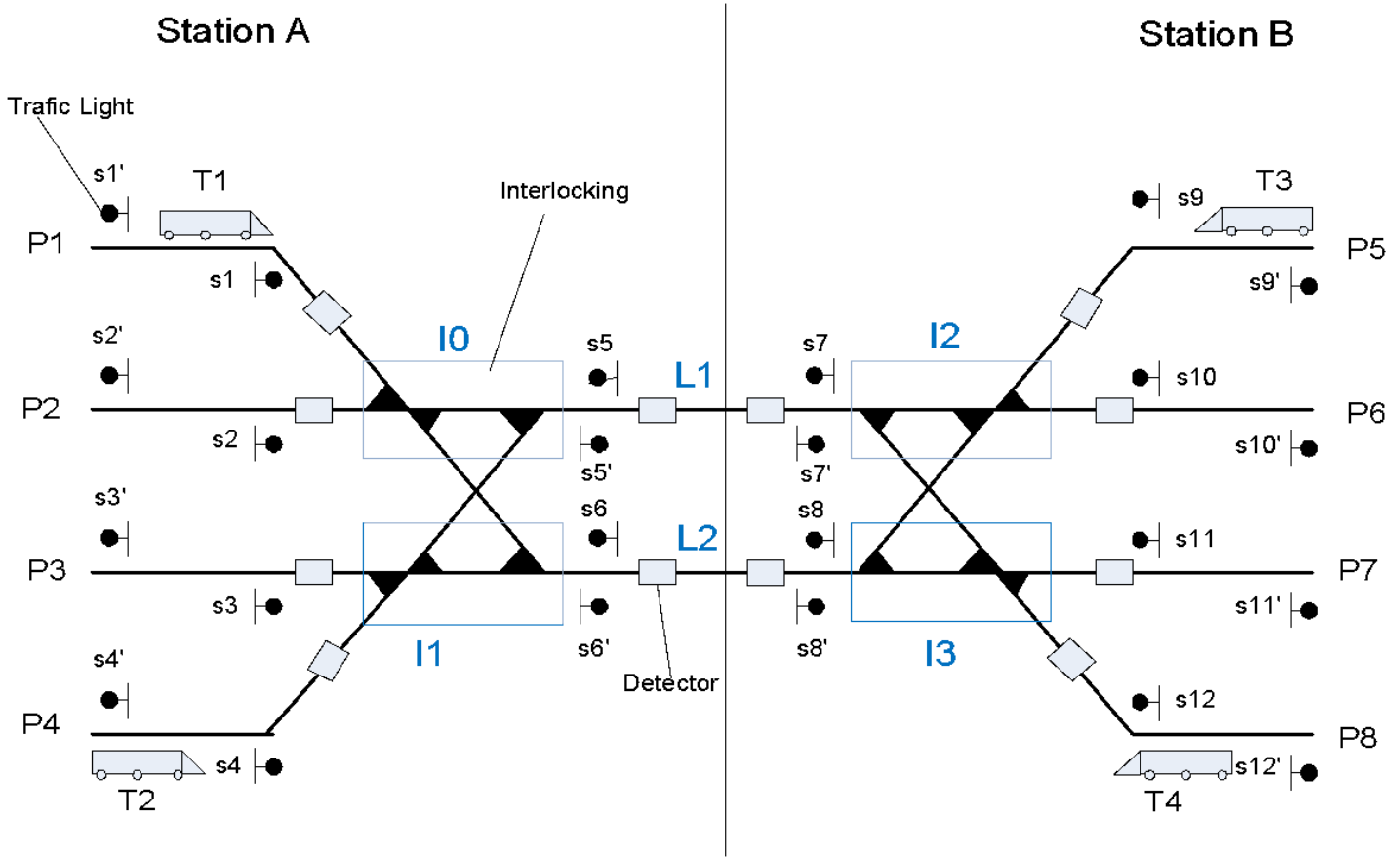
- diagrama componentelor pentru 2 controlere cooperative
- adăugați rețeaua ETPN pentru controler
- adăugați funcțiile de control continuu



Controlul traficului feroviar

Temă de casă: construiţi diagrama componentelor
Adăugaţi: reţelele ETPN şi

reţelele FLETPN



4.8 Verificarea reţelelor Petri de timp (DTPN)

Partiţionarea reţelelor Petri

- **Locaţii comune**
- **tranziţii comune**
- **ambele**

Locaţiile comune modelează starea sistemului ca date sau elemente fizice în poziţii specificare, etc.

Tranziţiile comune modelează faptul că sistemul a terminat o activitate şi schimbă starea sau semnalează un eveniment.

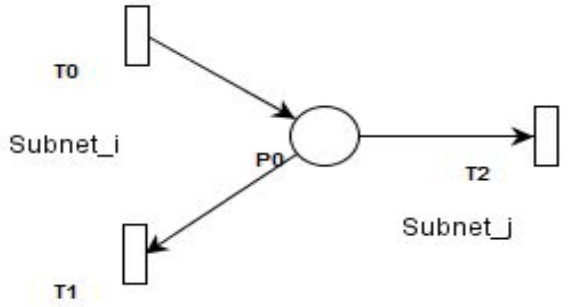
Tranzițiile reprezintă:

- semnalizarea producerii unor evenimente sau
- trecerea de la desfăşurarea unei activități la alta.

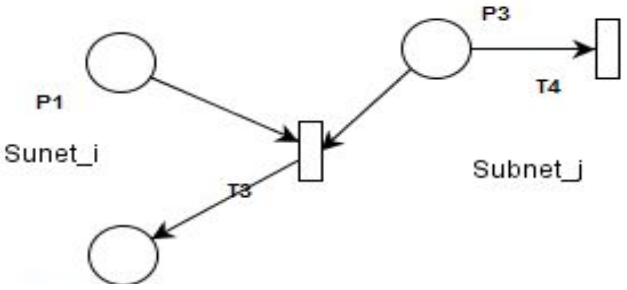
Situaţii critice la reţele Petri

Nu pot fi rezolvate uşor

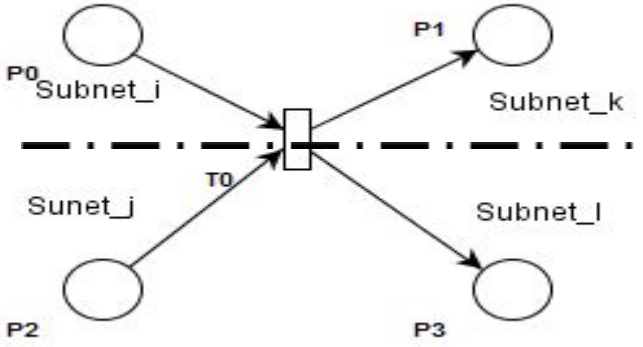
Două controlere diferite pot extrage jetoane din aceeaşi locaţie P_i. De evitat.



Controlul execuţiei tranziţiei comune T_3 trebuie să ia informaţii din partiţii diferite. De evitat.

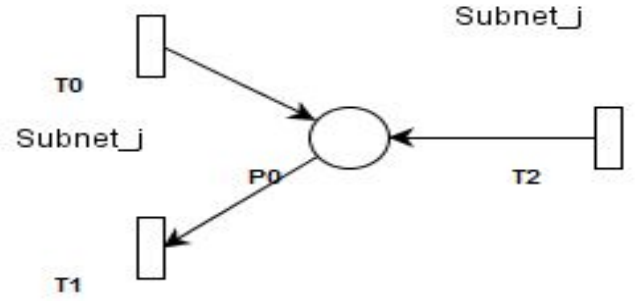


Tranziţia comună T_0 este condiţionată de diferite partiţii şi execuţia afectează locaţii din partiţii diferite. De evitat.

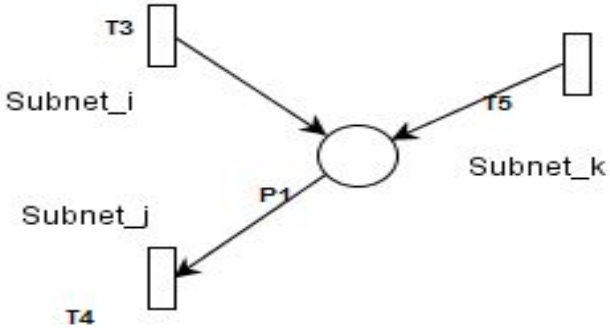


Situaţii acceptabile

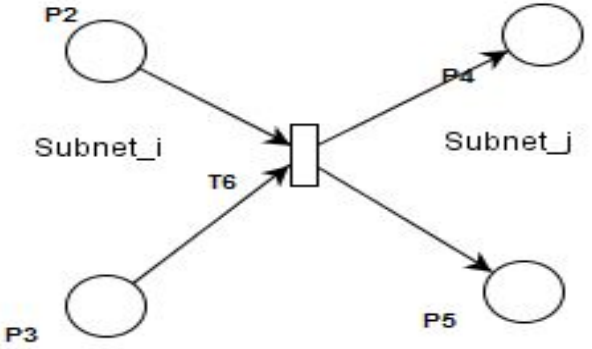
O locaţie comună încărcată din două partiţii diferite și care condiționează o tranziție din una dintre partiții.



O locaţie comună încărcată din două partiţii diferite și care condiționează o tranziția din a a treia partiție.



O tranziție comună care are condiția de execuție din aceeași partiție încarcă locații din altă partiție.



Partiţionarea cu locaţii comune

N = (P, T, pre, post)

- ***subreţele*** (subrețelele în care se descompune)

$N_i = (P_i, T_i, pre_i, post_i)$ i=1,2,...,n

with:

$$\mathbf{P_i \subseteq P, T_i \subseteq T}$$

$$pre_i = pre \cap (P_i \times T_i), post_i = post \cap (T_i \times P_i)$$

Jetoanele rămân aceleași (ca în rețeaua N)
Predicatele, acțiunile și intervalele de timp ale tranzițiilor rămân aceleași ca în rețeaua N.

Marcajul inițial al lui N_i este restricție marcajului inițial la subrețea.

$N_i = (P_i, T_i, pre_i, post_i)$ i=1,2,...,n

Interfeţe

$$\bigsqcup_{i=1}^{n} P_i = P \qquad \bigsqcup_{i=1}^{n} T_i = T$$

$$\bigsqcup_{i=1}^{n} pre_i = pre \qquad \bigsqcup_{i=1}^{n} post_i = post$$

$N_i = (P_i, T_i, pre_i, post_i)$ i=1,2,...,n

Partiţionarea lui N ➔ $\{N_i , N_j\}$ cu locaţii comune dar fără tranziţii comune:

$$\forall i, j : i \neq j \Rightarrow T_i \cap T_j = \Phi$$

Locaţii comune $S_{i,j}$ ➔ interfeţe între N_i şi N_j

Setul interfețelor lui N_i este

I_i - locațiile de intrare din subrețeaua N_i

$N_i = (P_i, T_i, pre_i, post_i)$

$N_j = (P_j, T_j, pre_j, post_j)$

$$p_S \in P_i \wedge \exists t \in T_j \mid p_s \in^o t$$

O_i - locații de ieșire din subrețeaua N_i

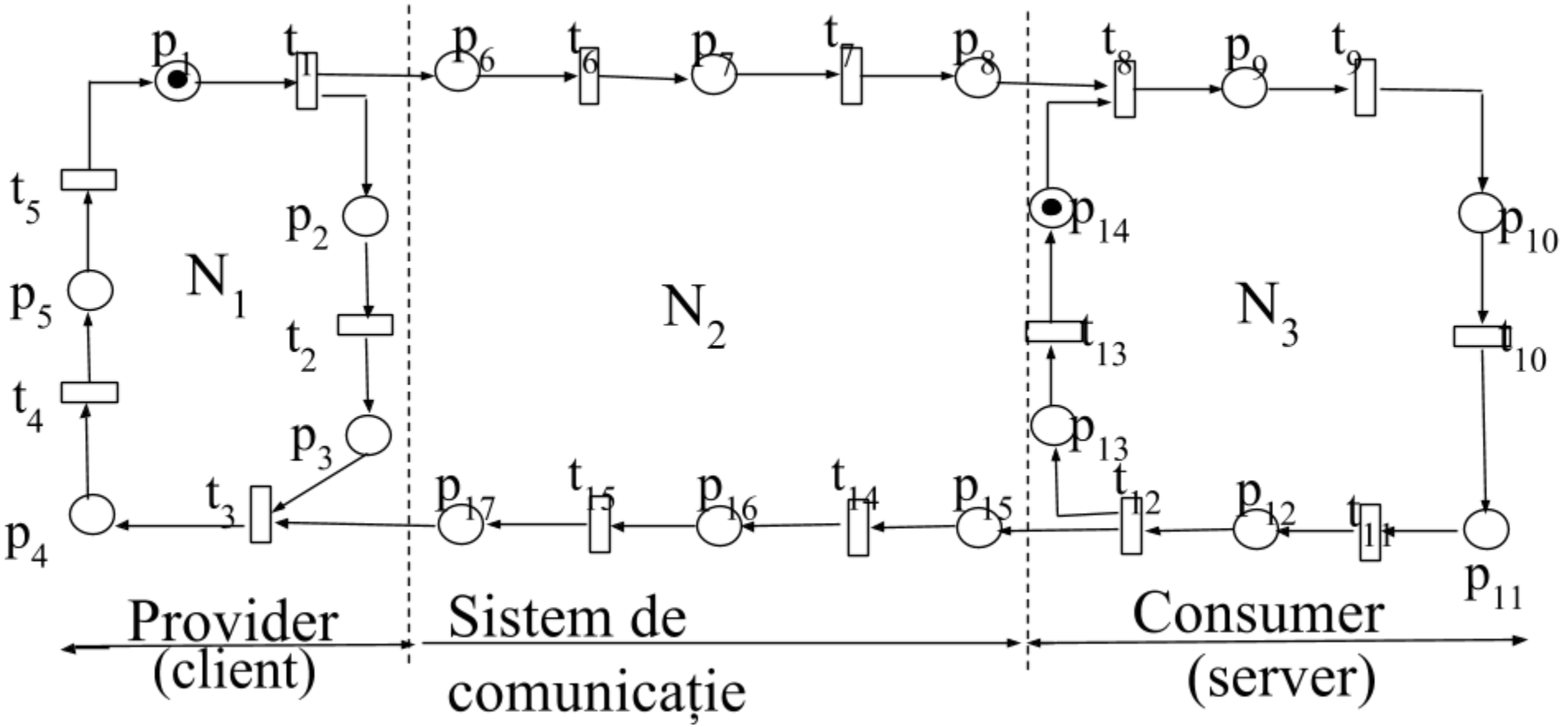
.

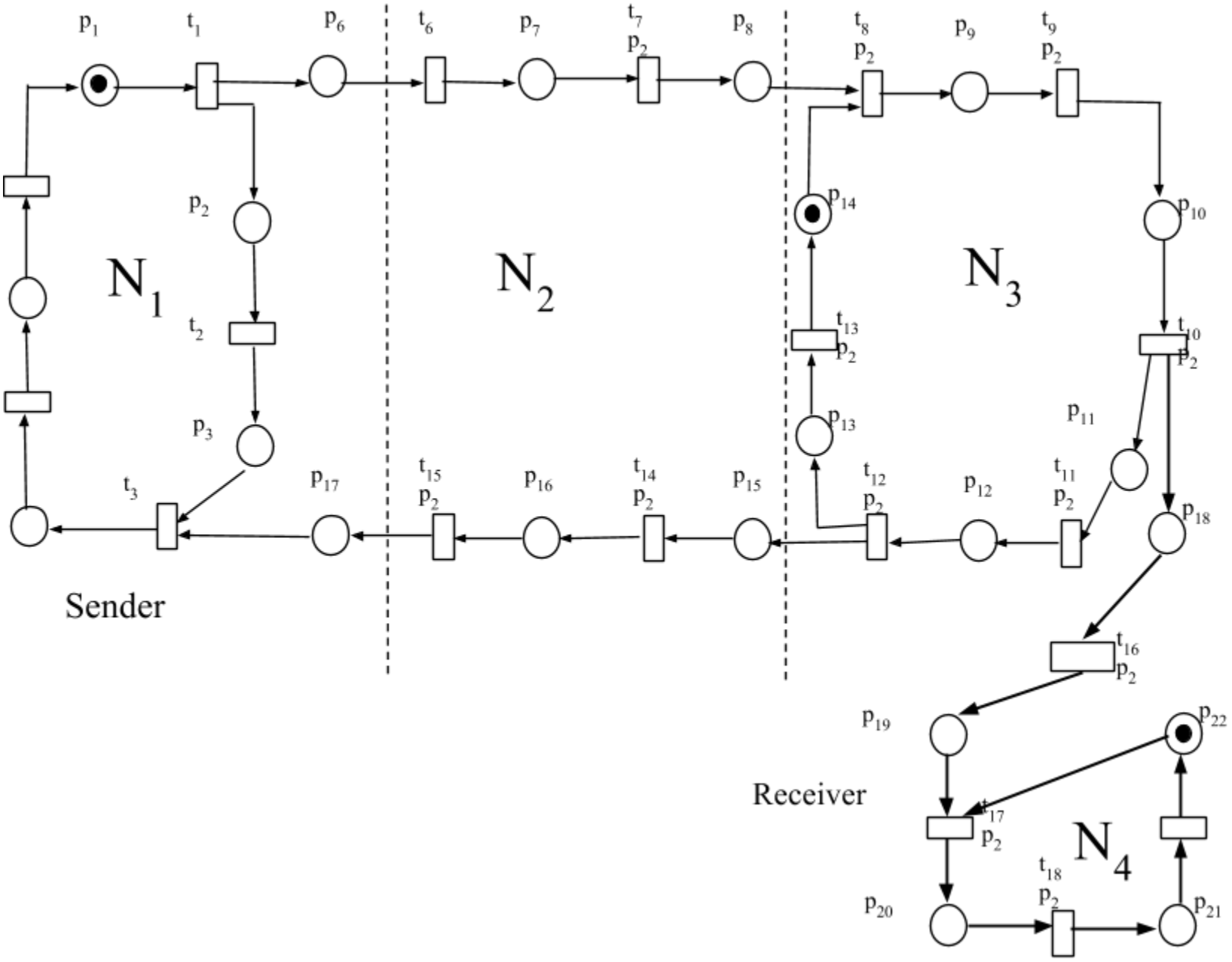
$$p_S \in P_i \wedge \exists t \in T_j \mid p_s \in t^o$$

Se notează cu O_i setul locațiilor de ieşire a lui N_i similar cu setul S_i.

Pot exista locații proprii ale lui N_i, precum şi tranziții de intrare sau de ieşire proprii.

Exemple de reţele Petri distribuite




p1
t1
p6
t6
p7
t7
p2
p8
t8
p2
p9
t9
p2
p14
p10
p2
N1
N2
N3
t13
p2
t10
p2
t2
p13
p11
p3
t3
p17
t15
p2
p16
t14
p2
p15
t12
p2
p12
t11
p2
p18
Sender
t16
p2
p19
p22
Receiver
t17
p2
N4
t18
p2
p20
p21

Exemplu: Specificarea controlului pentru traficul feroviar

Pentru cazul unei aplicații de control al traficului feroviar se poate realiza partiționarea rețelelor de cale ferată (sistemul este descompus în subrețele interconectate între ele) sau/şi sistemul de control (modelele realizate utilizând rețele Petri sunt implementate pe calculatoare diferite intreconectate într-o rețea).

Partiționarea cu locații comune

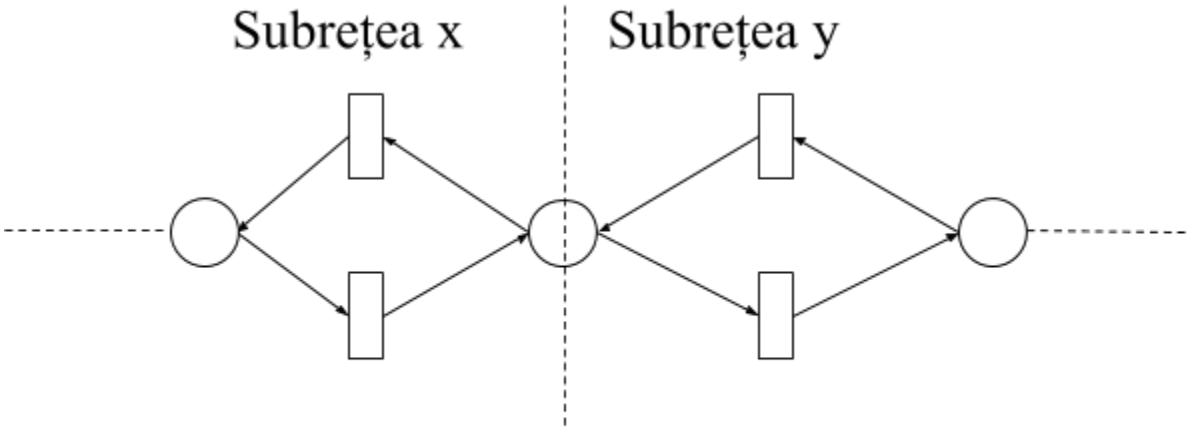


Fig. 4.10. Partitioning with shared places

Partiționarea cu tranziții comune

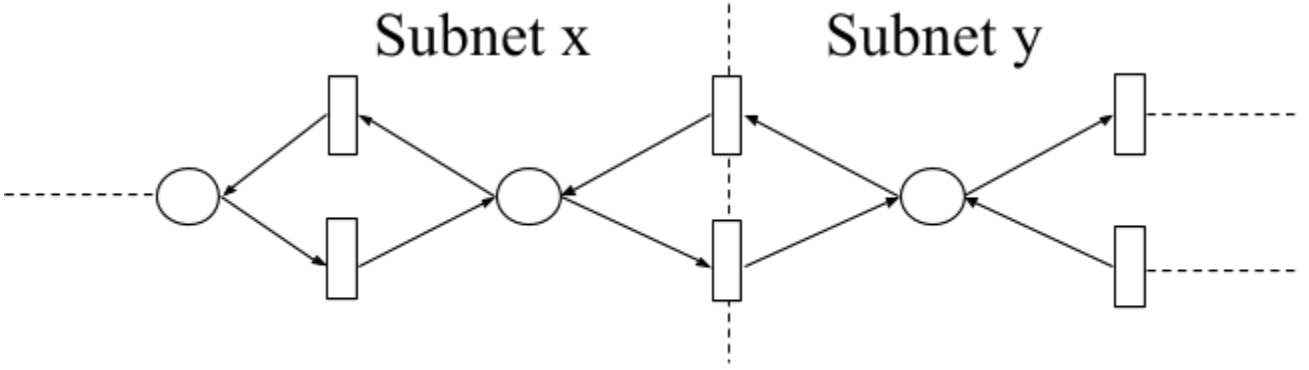


Fig. 4.11. Partitioning with shared transitions

4.9 Unified Enhanced Time Petri Nets

4.10 Object Petri Nets

*

***

****

*** Sfârşit ***

****

***

*

Octavian Cuibus: Sisteme de control distribuit: Specificare, verificare, proiectare