CONTRIBUŢII LA MODELAREA ŞI IMPLEMENTAREA SISTEMELOR CU EVENIMENTE DISCRETE CU APLICAŢII DIRECTE ASUPRA SISTEMELOR DE TRANSPORT CU ZONE DE ACUMULARE

Teză destinată obţinerii
titlului ştiinţific de doctor inginer
la
Universitatea "Politehnica" din Timişoara
în domeniul AUTOMATICA
de către

Ing. Dan Ungureanu-Anghel

Conducător ştiinţific:	prof.univ.dr.ing. Octavian Proştean
Referenţi ştiinţifici:	prof.univ.dr.ing. Mircea Ivănescu
	prof.univ.dr.ing. Octavian Păstrăvanu
	prof.univ.dr.ing. Nicolae Robu

Ziua susţinerii tezei: 18.07.2007

Seriile Teze de doctorat ale UPT sunt:

1. Automatică	7. Inginerie Electronică şi Telecomunicaţii
2. Chimie	8. Inginerie Industrială
3. Energetică	9. Inginerie Mecanică
4. Ingineria Chimică	10. Ştiinţa Calculatoarelor
5. Inginerie Civilă	11. Ştiinţa şi Ingineria Materialelor
6. Inginerie Electrică	

Universitatea „Politehnica" din Timişoara a iniţiat seriile de mai sus în scopul diseminării expertizei, cunoştinţelor şi rezultatelor cercetărilor întreprinse în cadrul şcolii doctorale a universităţii. Seriile conţin, potrivit H.B.Ex.S Nr. 14 / 14.07.2006, tezele de doctorat susţinute în universitate începând cu 1 octombrie 2006.

România, 300159 Timişoara, Bd. Republicii 9,
tel. 0256 403823, fax. 0256 403221
e-mail: editura@edipol.upt.ro

Cuvânt înainte

Teza de doctorat a fost elaborata pe parcusul activitatii mele in cadrul Departamentului de Automatica si Informatica Aplicata al Facultatii de Automatica si Calculatoare din Timisoara.

Sistemele sunt o realitate a vietii noastre cotidiene, formele lor de materializare si manifestare fiind diferite.

Din diversitatea de sisteme existente, lucrarea de fata abordeaza domeniul sistemelor cu evenimente discrete (SED), sisteme alcatuite dintr-o multime de secvente de evenimente care descriu comportarea lor.

Dezvoltarea si fundamentarea teoriei moderne a modelarii, precum si impactul creat de progresele spectaculoase ale tehnologiei sistemelor de calcul, au creat baza teoretica si practica fara de care constructia SED este de neconceput.

Utilizarea tehnicilor SED in modelarea si conducerea proceselor tehnologice, fac obiectul a numeroase cercetări ştiinţifice fundamentale şi aplicative.

În această teză, SED au fost abordate prin prisma stabilirii unor metodologii de modelare si implementare cu aplicatii directe asupra sistemelor de transport cu zone de acumulare.

Pentru realizarea actualei lucrări doresc să aduc alese mulţumiri conducătorului ştiinţific, domnului prof. univ. dr. Octavian Prostean pentru sprijinul şi competenta îndrumare acordată pe întreaga perioadă a elaborării tezei.

Îmi exprim întreaga consideraţie faţă membrii comisiei de doctorat, domnul preşedinte al comisiei prof. univ. dr. ing. Mircea STRATULAT prodecanul Facultăţii de Automatica si Calculatoare din Timişoara şi domnii prof.univ.dr. Mircea IVANESCU de la Universitatea din Craiova, prof. univ. dr. ing. Octavian PASTRAVANU de la Universitatea Tehnica „Gh. Asachi" din Iasi şi prof. univ. dr. ing. Nicolae ROBU de la Facultatea de Automatica si Calculatoare din Timişoara, care au răspuns solicitării de a face parte din comisia de analiză a tezei, pentru observaţiile făcute şi pentru timpul acordat lucrării.

Timişoara, iulie 2007 Dan Ungureanu-Anghel

Familiei mele

Ungureanu-Anghel, Dan

Contribuţii la modelarea şi implementarea sistemelor cu evenimente discrete cu aplicaţii directe asupra sistemelor de transport cu zone de acumulare

Teze de doctorat ale UPT, Seria 1, Nr. 4 , Editura Politehnica, 2007,
342 pagini, 236 figuri, 23 tabele.

ISSN: 1842-5208

ISBN: 978-973-625-482-6

Cuvinte cheie:

sisteme cu evenimente discrete, automate, retele Petri, sisteme de transport cu zone de acumulare, noduri, supervizoare, implementare

Rezumat:

Datorita cresterii complexitatii proceselor industriale, si dezvoltarii explozive a echipamentelor de calcul, s-au cautat noi solutii de conducere care sa poata opera astfel incat sa fie eliminate dificultatile legate de stabilirea modelului matematic. O alternativa, care se impune din ce in ce mai mult, sunt tehnicile de modelare a sistemelor de conducere industriale ca sisteme cu evenimente discrete (SED). Modul de abordare a conducerii SED, complet diferit de metodele clasice, asigura gestionarea fara probleme a tuturor resurselor unui sistem industrial, elimina complet utilizarea in modelare a ecuatilor diferentiale sau a ecuatilor discrete. Elementul timp, esential in cazul metodelor clasice, nu mai este relevant, elementele de baza in cazul SED sunt evenimentele, controlul acestora nefiind influentat de momentul aparitiei (timpul) ci numai de starea logica a acestora.

Cercetarile efectuate in cadrul tezei au urmarit: stabilirea unor metodologii pentru modelarea si implementarea SED cu aplicatii directe asupra sistemelor de transport cu zone de acumulare.

In cadrul lucrarii, modelarea si implementarea SED s-a realizat utilizand doua tehnici fundamentale: prima bazata pe utilizarea automatelor si a doua bazata pe utilizarea retelelor Petri.

Cuprins

Capitolul 1. Introducere

Capitolul 1. Introducere 11
1.1. Oportunitatea si obiectivele tezei 11
1.2. Prezentarea continutului tezei 12

Capitolul 2. Modelarea Sistemelor cu Evenimente Discrete utilizand automate si retele Petri 15
2.1. Preliminarii 15
2.2. Limbaje si automate. Notiuni teoretice 16
2.2.1. Limbaje Formale 16
2.2.2. Automate 17
2.2.3. Limbajele ca reprezentante a automatelor 18
2.2.4. Blocaje 19
2.2.5. Automate nedeterministe 20
2.2.6. Operatii cu automate 20
2.2.6.1. Operatii unare 20
2.2.6.2. Operatii de compunere 22
2.2.7. Transformarea unui automat nedeterminist in automat determinist 25
2.3. Retele Petri 26
2.3.1. Reţele Petri netemporizate 26
2.3.2. Analiza proprietăţilor comportamentale 29
2.3.3. Tehnici generale de analiză a proprietăţilor comportamentale 30
2.3.4. Tehnici generale de analiză structurala 32
2.3.4.1. Aspecte generale ale analizei structurale 32
2.3.4.2. Analiza structurala pe baza invariantilor 33
2.3.5. Retele Petri temporizate 34
2.3.5.1. Retele Petri temporizate cu timpi constanti 36
2.3.5.2. Retele Petri temporizate cu intervale de timpi 37
2.3.6. Retele Petri etichetate 37
2.3.7. Utilizarea submodelelor in modelarea SED cu ajutorul retelelor Petri 38
2.4. Conexiuni intre modelele de tip automat si modelele de tip retea Petri 44
2.5. Modelarea cu ajutorul automatelor respectiv a retelelor Petri a sistemelor de transport cu zone de acumulare 50
2.5.1. Principile de realizare si functionare ale sistemelor de transport cu zone de acumulare 50
2.5.1.1. Nodul de tip 1 – o intrare o ieşire 54
2.5.1.2. Nodul de tip 2 – două intrări o ieşire 54
2.5.1.3. Nodul de tip 3 – trei intrări o ieşire 55
2.5.1.4. Nodul de tip 4 – o intrare două ieşiri 56
2.5.2. Modelarea elementelor componente ale sistemelor de transport cu zone de acumulare considerate ca SED 56
2.5.2.1. Modelarea nodului de tip 1 – o intrare o iesire 56
2.5.2.1.1. Modelarea nodului de tip 1 cu ajutorul automatelor 56
2.5.2.1.2. Modelarea nodului de tip 1 cu ajutorul retelelor Petri 62
2.5.2.1.2.1. Cazul retea Petri netemporizata 62
2.5.2.1.2.2. Cazul retea Petri temporizata P 68
2.5.2.2. Modelarea nodului de tip 2 – doua intrari o iesire 69
2.5.2.2.1. Modelarea nodului de tip 2 cu ajutorul automatelor 69
2.5.2.2.2. Modelarea nodului de tip 2 cu ajutorul retelelor Petri 75
2.5.2.2.2.1. Cazul retea Petri netemporizata 75

Indexul principalelor abrevieri, notaţii şi simboluri

Abrevieri

AS – automat secvential

PN – reţea Petri

SED – sisteme cu evenimente discrete

STZA – sistem de transport cu zone de acumulare

SFFF – sistem flexibil de fabricatie pentru o filatura

IPC – comunicare între procese prin mesaje

SHM – memorie partajata

FIFO – primul intrat, primul iesit (first in, first out)

TEF – transelastic conveior

SOTR – sistem de operare în timp real

PLC – automat programabil

Notaţii

N – mulţimea numerelor naturale

Z – mulţimea numerelor întregi

L – limbaj generat

L_m – limbaj marcat

G – automat determininst

$G = (X, \Sigma, \delta, \Gamma, x_0, X_m)$ - definirea automatului

G_{nd} – automat nedeterminist

X - setul starilor;

Σ - setul evenimentelor

δ - functia de tranzitie

Γ - functia eveniment activ

x_0 - starea initiala

X_m - setul starilor marcate.

p – poziţie

t – tranziţie

PN = (P, T, F, w, M_0) – graful reţelei Petri

P = $\{p_1, p_2, \ldots, p_n\}$ – mulţimea poziţiilor

$T = \{ t_1, t_2, \ldots, t_m \}$ – mulţimea tranziţiilor
F – mulţimea arcelor
$w : A \rightarrow \{1, 2, \ldots\}$ – funcţia de pondere
$x : P \rightarrow N$ – vector marcaj
M_0 – marcaj iniţial
K(p) – capacitatea poziţiei p
x(p) – numărul jetoanelor din poziţia p
$\varnothing$ – mulţimea vidă
I – mulţimea poziţiilor/tranziţiilor de intrare
O – mulţimea poziţiilor/tranziţiilor de ieşire
σ – secvenţă de executări de tranziţii
R() – mulţimea marcajelor care pot fi atinse
L() – mulţimea secvenţelor de executări posibile
X – spaţiul stărilor
V – mulţimea nodurilor
E – mulţimea arcelor din graful de acoperire
G(V, E) – graf de acoperire
d() – distanţă sincronă între două tranziţii
max() – valoarea maximă
$A \in Z^{n \times m}$ – matrice de incidenţă
A^- – matrice de incidenţă de intrare
A^+ – matrice de incidenţă de ieşire
u_k – vector de control
$*^T$ – transpusa matricii $*$
rang $*$ – rangul matricii $*$

Simboluri

$=$ – egalitate
$\cup$ – reuniune
$\times$ – produs cartezian
$\cap$ – intersecţie
$\neq$ – distincţie
$\in$ – apartenenţă
$>$ – mai mare strict
$\geq$ – mai mare sau egal

$\subseteq$ – incluziune cu posibilitate de egalitate

$<$ – mai mic strict

$\leq$ – mai mic sau egal

$\Rightarrow$ – implică

ω – simbol utilizat în construcţia arborilor
sau grafurilor de acoperire pentru
reţelele nemărginite

$\forall$ – oricare ar fi

$\pm$ – adunare sau scădere

$\neg$ - negare

Termeni:

Deadlock - blocaj

Nota:

1. Definitiile originale ale autorului sunt marcate cu „*"
2. Bibliografia este notata:
    a. [XYZAA] - in cazul mai multor autori XYZ reprezentand initialele autorilor iar AA anul publicarii;
    b. [XyzAA] – in cazul unui auto runic Xyz reprezentand primele 3 litere ale numelui iar AA anul aparitiei

Capitolul 1

Introducere

1.1. Cadrul si obiectivele tezei

Sistemele sunt o realitate a vietii noastre cotidiene, formele lor de materializare si manifestare fiind diferite.

Din diversitatea de sisteme existente, lucrarea de fata abordeaza domeniul sistemelor cu evenimente discrete (SED), sisteme alcatuite dintr-o multime de secvente de evenimente care descriu comportarea lor.

Dezvoltarea si fundamentarea teoriei moderne a modelarii, precum si impactul creat de progresele spectaculoase ale tehnologiei sistemelor de calcul, au creat baza teoretica si practica fara de care constructia SED este de neconceput.

Prin SED se intelege fie un sistem real, fie un model matematic (ce descrie funcţionarea unui sistem real), a cărui evoluţie este raportată la apariţia unor evenimente. Astfel, producerea evenimentelor joacă rolul de cauză pentru dinamica sistemului şi are drept efect modificarea stărilor sistemului. Mai mult, în cazul unui SED se poate vorbi despre o funcţie de tranziţie a stărilor, care formalizează faptul că sistemul trece dintr-o stare în alta numai ca urmare a producerii unui eveniment şi că sistemul păstrează starea în care se află până la producerea unui nou eveniment; altfel spus, SED reprezintă sisteme dinamice în care mulţimea stărilor este una discretă, iar mecanismul de tranziţie al stărilor este pilotat de evenimente cu apariţie asincronă.

Existenţa unei mari varietăţi de aplicaţii tehnice (procese de fabricaţie, sisteme de comunicaţie şi procesare a datelor, sisteme şi reţele de transport, etc.) a impulsionat dezvoltarea acestui domeniu, necesitând noi strategii de conducere ce diferă principial de schemele utilizate în teoria şi practica reglării automate. Cu toate că SED au fost abordate prin prisma mai multor procedee de modelare, ce se întind de la teoria automatelor şi limbajelor formale, până la procesele Markov şi teoria cozilor de aşteptare, s-a resimţit absenţa unui cadru unificator.

Exista multe posibilitati de abordare a SED, si desi le-au fost dedicate in ultimul timp un important numar de lucrari de sinteza, [CL01] [Pastr97] [PMM02] [Whon02] [Rob97] [Aalst96a] [Aalst96b] [AB03] etc, totusi, se poate afirma ca nici pana in prezent nu exista o teorie unitara in acest domeniu, el ramanand in continuare o problema deschisa.

In cadrul SED s-au dezvoltat o gama larga de metode de modelare obtinute prin utilizarea unor mecanisme total diferite de cele „clasice" (ecuatile diferentiale sau ecuatile cu diferente), cele mai importante implicand utilizarea:

- limbajelor formale,
- automatelor,
- retelelor Petri,
- lanturilor Markov.

(lista putand continua cu alte abordari).

Lucrarea de fata isi propune sa asigure un cadru procedural si metodologic unificator, pentru conjuctia dintre modelele de tip SED si implementarea lor, cu aplicatii directe referitoare la sistemele de transport cu zone de acumulare (STZA).

Obiectivele tezei pot fi sintetizate astfel:

- elaborarea unui material unitar privind aspectele teoretice ale modelarii SED pe baza modelelor de tip automat, a modelelor de tip retele Petri netemporizate, respectiv retele Petri temporizate P;
- modelarea STZA „vazute" ca SED pornind de la cateva structuri de baza si incheind cu modele de ordin general;
- definirea teoretica a STZA;
- demonstarea prin exemple si studii de caz a metodologiei de modelare a STZA vazute ca SED;
- elaborarea unui material unitar legat de aspectele teoretice ale conducerii supervizate ale SED pe baza modelelor de tip automat, respectiv retele Petri;
- demonstrarea prin exemple si studii de caz a metodologiei de sinteza a supervizoarelor cu aplicatii directe in sistemele flexibile de fabricatie, respectiv STZA;
- elaborarea unor metodologi de implementare a modelelor de tip automat, respectiv retele Petri, corespunzatoare STZA, avand ca suport hardware calculatoare de proces respectiv automate programabile;
- elaborarea unui produs progam util simularii retelelor Petri.

1.2. Prezentarea continutului tezei

Obiectivele propuse au condus la structurarea lucrarii pe 6 capitole, al caror continut este prezentat in continuare.

Capitolul 2 este dedicat problemelor legate de modelarea SED utilizand automate, respectiv retele Petri, abordandu-se aspecte teoretice si aplicative.

In paragraful 2.1 se prezinta in mod sintetic problemele teoretice legate de utilizarea limbajelor si a automatelor ca metode de modelare a SED, fiind abordate probleme legate de: definirea limbajelor formale si a automatelor, reprezentarea automatelor cu ajutorul limbajelor formale, blocajele, definirea automatelor nedeterministe, operatii cu automate, modul de transformare a unui automat nedeterminist intr-un automat determinist..

In paragraful 2.2 sunt tratate problemele teoretice legate de utilizarea retelelor Petri ca metode de modelare a SED, fiind abordate probleme legate de: retelele Petri netemporizate, analiza proprietatilor comportamentale si structurale, retele Petri temporizate, retele Petri etichetate, utilizarea submodelelor in modelarea SED cu ajutorul retelelor Petri.

Datorita faptului ca intre modelele de tip automat, respectiv modelele de tip retele Petri exista o serie de conexiuni, aceasta problematica a fost detaliata in paragraful 2.3 prezentandu-se o serie de metode de conversie a modelelor de tip automat in modele de tip retele Petri.

In paragraful 2.4, in totalitate original, este abordata problema modelarii cu ajutorul automatelor, respectiv a retelelor Petri a STZA. Astfel, in prima parte a paragrafului se prezinta principiile de realizare si functionare ale STZA, considerandu-se patru structuri de baza (noduri) stabilite pentru modelarea in ansamblu a STZA. Pornind de la structurile de baza stabilite pentru fiecare nod in parte, modelarea s-a realizat prin trei metode distincte: automate, retele Petri netemporizate respectiv retele Petri temporizate P. Pornind de la aceste modele, simulate si validate cu ajutorul mediului Matlab, s-au elaborat modele de ordin general pentru noduri cu *„n"* intrari *„o"* iesire, *„o"* intare *„m"* iesiri si *„n"* intrari *„m"* iesiri.

In paragraful 2.5, pornind de la rezultatele obtinute in paragraful precedent s-a urmarit modelarea in ansamblu a STZA. In prima parte se prezinta simbolistica utilizata in reprezentarea nodurilor. In partea a doua a acestui paragraf este prezentata definitia STZA, precum si probleme teoretice legate de modelarea acestora. In partea a treia se prezinta doua metodologii legate de modelarea STZA cu ajutorul reteleor Petri.

In ***Capitolul 3*** sunt abordate aspecte legate de conducerea supervizata a SED.

In paragraful 3.1 sunt considerate aspecte teoretice legate de structura de baza utilizata in conducerea supervizata a SED, abordata ca si in cazul modelarii SED (capitolul 2) pe baza modelelor de tip automat, respectiv modele de tip retele Petri.

In paragraful 3.2 este aborbata problematica legata de conducerea supervizata a SED bazata pe utilizarea modelelor de tip automat. Astfel, sunt considerate o serie de aspecte teoretice legate de: supervizarea sistemelor cu reactie dupa stare, algoritmi de implementare a supervizoarelor bazate pe modele de tip automat. Metodologia de realizare a unui supervizor bazat pe modele de tip automat este exemplificata printr-un studiu de caz legat de conducerea supervizata a unui sistem flexibil de fabricatie pentru o filatura.

In paragraful 3.3 este aborbata problematica legata de conducerea supervizata a SED bazata pe utilizarea modelelor de tip retele Petri. Sunt prezentate o serie de aspecte teoretice legate de: conducerea supervizata prin utilizarea limbajelor retelelor Petri, conducerea supervizata utilizand modele de tip retele Petri bazata pe invarianti de tip P , conducerea supervizata utilizand modele de tip retele Petri bazata pe compunerea paralela a subsistemelor. Metodologia de realizare a unui supervizor bazat pe modele de tip retele Petri este exemplificata printr-un studiu de caz legat de conducerea supervizata a unui STZA.

In ***Capitolul 4*** sunt abordate aspecte legate de implementarea modelelor SED in sisteme de conducere in timp real.

In paragraful 4.1 sunt prezentate principiile generale ale proiectarii programelor de conducere in timp real.

Problematica legata de implementare este canalizata pe doua directii distincte. In paragraful 4.2 este abordata implementarea modelelor de tip automat avand ca suport hardware un calculator de proces pe care este instalat sistemul de operare in timp real QNX. Pentru o implementare eficienta se exploateaza capabilitatile SOTR QNX legate de: mecanismele de comunicatie interprocese, comunicatia intre procese prin utilizarea memoriei partajate, dispecerizarea proceselor, starile unui proces. In subparagraful 4.2.2 este prezentata o metoda de implementare originala in SOTR QNX a unui STZA pe baza modelelor de tip automat. Sunt prezentate modurile de implementare a nodurilor precum si modul de realizare a comunicatiei interprocese, atat prin mesaje cat si prin memoria partajata. Limbajul de programare utilizat pentru implementare a fost WATCOM C. In final a rezultat un produs program original care a fost testat si validat in totalitate.

In paragraful 4.3 se considera implementarea SED bazata pe modele de tip retele Petri avand ca suport hardware un automat programabil de tip Simatic S7-414. Se prezinta modul de implementare a modelului de tip retea Petri corespunzator nodului cu *„o"* intrare si *„o"* iesire, dupa care, pe baza functiei elaborate se prezinta modul de implementare a unui program de conducere pentru un STZA. Limbajul de programare utilizat pentru implementare a fost STEP7. In final a rezultat un produs program original care a fost testat si validat in totalitate.

In ***Capitolul 5*** , in totalitate original, este prezentata o aplicatie scrisa pentru simularea modelelor de tip retele Petri, denumita ***PetriTim***. Aplicatia permite generarea de fisiere care contin explicit operatiile realizate pe baza utilizarii ecuatiei de stare, si care in modul de rulare pas cu pas sa dea posibilitate utilizatorului sa aleaga tranzitiile care se executa. Aplicatia a fost dezvoltata in Windows avand ca suport limbajul VISUAL C++.

In ***Capitolul 6*** sunt sintetizate concluziile rezultate in urma realizarii tezei, evidentiindu-se contributiile originale ale autorului aduse in modelarea si implementarea SED, cu aplicatii directe asupra STZA, contributii care au fost partajate in doua categorii: teoretice si aplicative.

Capitolul 2

Modelarea Sistemelor cu Evenimente Discrete utilizand automate si retele Petri

2.1. Preliminarii

SED sunt, dupa cum s-a mai precizat, sisteme descrise pe baza evenimentelor si a starilor in care se afla sistemul, timpul in acest caz fiind un element nerelevant. In cadrul SED, schimbarea starii se face complet asincron, functie de evenimentele aparute si nu pe baza unui semnal de tact.

Modelarea SED nu se bazeaza pe utilizarea ecuatiilor diferentiale sau ecuatiilor cu diferente, ci pe utilizarea unor metode complet diferite, cum ar fi: limbajele, automatele, retelele Petri etc.

Problemele teoretice ale modelarii SED cu ajutorul modelelor de tip automat respectiv retele Petri este abordata in ultima perioada in numeroase lucrari [Cass02] [CL01] [CKLY95] [CR04a] [FG98] [GM05b] [HKG97] [JK96] [Letia98] [Pastr97] [RW89] [Schm05] [UP06a] [UP06b] [UP06c] [UP06d]. Astfel, in [Cass02][CL01] sunt prezentate in detaliu metode de modelare a SED utilizand limbajele formale, automatele, retelele Petri, lanturile Markov etc. Daca utilizarea modelelor de tip automat este o tema relativ „veche”, numerosi autori s-au concentrat in dezvoltarea teoriei de modelare avand ca suport retelele Petri [Mur89] [BGM00] [CRL06] [DT06] [HF01] [Kemp04] [Letia98] [Pastr97] [PN2000].

O analiza a posibilitatii de utilizare a modelelor de tip retele Petri in conducerea proceselor industriale poate fi regasita in [Aalst94] [Aalst96b] [Aalst98] [DHPSV93] [TTV05], iar ilustrarea modului de utilizare a acestora in [AB03] [AO95][Ciuf02] [CP04] [CR04b] [Daro05b] [EL02] [Kout01] [KW00] [LKWDS06] [FG98] [Aalst96a] [BD03] [ERRW03] [FL00a][GB05][HL97][Hsieh06][Kind04] [KT05][PWX97][Rob97].

Extinderea domeniului de aplicabilitate a utilizarii modelelor de tip retele Petri a condus la aparitia de noi tipuri de retele Petri derivate din modelul initial. Astfel, au fost dezvoltate teorii legate de retelele Petri temporizate [BC04] [BSV03] [FGMS02] [PMM02] [Pomm03] [JJRS04] [JTMZ01], de retelele Petri colorate [Jens96], de retelele Petri etichetate [CL01][GCS05]HKG97][GS05].

Cu toata abundenta de lucrari de specialitate in domeniu, se poate afirma ca problematica legata de utilizarea modelelor de tip retele Petri ca tehnica de modelare a SED este doar la inceput.

In cadrul acestui capitol se urmareste sintetizarea notiunilor fundamentale legate de modelarea SED cu ajutorul automatelor, respectiv a retelelor Petri. Aspectele teoretice sunt imbinate cu aplicatii directe asupra sistemelor de transport cu zone de acumulare (STZA) pornind de la modele simple pana la modele de ordin general.

2.2. Limbaje si automate

2.2.1. Limbaje Formale [CL01][Schm05][Wonh02][DM05]

In [DM05][CL01][Schm05][Wonh02] limbajele formale sunt tratate in detaliu atat din punct de vedere al formalismului matematic cat si din punct de vedere aplicativ. In cadrul acestui paragraf nu se va insista decat asupra unei prezentari sintetice si unitare a definitiilor si operatiilor legate de limbajele formale.
Se defineste alfabetul $\Sigma = \{e_1, e_2, ..., e_n\}, n \in N$ ca fiind un set finit de simboluri e_i denumite *evenimente*.

Definitia 2.1 Limbaj
*Un **limbaj** definit relativ la un set de evenimente* Σ *este un set de siruri de lungimi finite format din evenimentele din* Σ.

Sirurile s_i ,care formeaza limbajul, sunt definite ca fiind obtinute prin concatenarea arbitrara a elementelor e_i : $s_i = e_{i_1}e_{i_2}.....e_{i_k}$, $e_{i_k} \in \Sigma$ cu $i_1, i_2, ..., i_k \in \{1, ..., n\}$.

Setul tuturor sirurilor definite pe baza elementelor din Σ se noteaza Σ^+.

Elementul nul – sir nul - se defineste ca fiind ε , unde $\varepsilon \notin \Sigma$.

Se defineste ca fiind *inchizator Kleene* a lui Σ : $\Sigma^* := \Sigma^+ \cup \{\varepsilon\}$

Fie sirul $tuv = s$ cu $t, u, v \in \Sigma^*$, atunci:

- t este denumit prefixul sirului s;
- u este denumit subsirul lui s;
- v este denumit sufixul sirului s.

Setul uzual de operatii, cum sunt reuniunea, intersectia, diferenta si complementarea cu referire la Σ^*, sunt aplicabile limbajelor. In plus in cadrul limbajelor se mai folosesc urmatoarele operatii [CL01][SchmC05]:

- *concatenarea*: fie $L_a, L_b \subseteq \Sigma^*$, atunci

  $$L_aL_b := \{s \in \Sigma^* : (s = s_as_b) \text{ si } (s_a \in L_a) \text{ si } (s_b \in L_b)\} \tag{2.1}$$

  altfel spus, un sir este in L_aL_b daca poate fi scris ca si rezultatul concadenarii unui sir din L_a cu un sir din L_b.

- *prefix-inchizator*: Fie $L \subseteq \Sigma^*$, atunci

  $$\bar{L} := \{s \in \Sigma^* : exista\, t \in \Sigma^* \text{ a.i. } (st \in L)\} \tag{2.2}$$

  altfel spus, prefixul-inchizator a lui L este limbajul notat $\bar{L}$ si este constituit din toate prefixele tuturor sirurilor din L. In general $L \subseteq \bar{L}$.

Se spune ca L este *prefix-inchizator* daca $L = \overline{L}$. Acest limbaj L este *prefix-closed* daca orice prefix a oricarui sir din L este de asemenea element din $\overline{L}$.

- *Inchiderea* (Kleene-closure): fie $L \subseteq \Sigma^*$, atunci

$$L^* := \{\varepsilon\} \cup L \cup LL \cup LLL \cup ...... \tag{2.3}$$

2.2.2. Automate [CL01][Schm05][Wonh02]

Automatele reprezinta o metoda deosebit de uzitata in modelarea SED.

Definitia 2.2 Automatul determinist
Un automat determinist, notat prin G, este un sextuplu

$$G = (X, \Sigma, \delta, \Gamma, x_0, X_m) \qquad (2\ .4)$$

unde:

- *X este setul starilor;*
- *Σ este un set finit de evenimente asociate cu tranzitiile din G;*
- *$\delta : X \times \Sigma \to X$ este functia de tranzitie: $\delta(x, e) = y$, inseamna aparitia unei tranzitii etichetata prin evenimentul e in starea x, ceea ce are ca efect tranzitia in starea y; in general, δ este o functie partiala pe domeniul sau de definire.*
- *$\Gamma : X \to 2^E$ este functia eveniment activ; $\Gamma(x)$ este setul tuturor evenimentelor e pentru care $\delta(x, e)$ este definita si este apelata de evenimentul activ din G corespunzator starii active x.*
- *x_0 este starea initiala a sistemului;*
- *$X_m \subseteq X$ este setul starilor marcate.*

Modul de operare a unui automat G este: sistemul porneste din starea initiala x_0 si dupa aparitia evenimentului $e \in \Gamma(x_0) \subseteq \Sigma$ va avea ca efect executia unei tranzitii intr-o stare corespunzatoare rezultatului functiei $\delta(x_0, e) \in X$. Acest proces continua avand la baza definirea tranzitiilor prin functia δ.

Functia δ este intotdeauna o extensie din domeniul $X \times \Sigma$ la domeniul $X \times \Sigma^*$ in urmatorul mod recursiv:

$$\begin{aligned} &\delta(x, \varepsilon) := x; \\ &\delta(x, se) := \delta(\delta(x, s), e) \quad pentru \ s \in \Sigma^* \ si \ e \in \Sigma. \end{aligned} \tag{2.5}$$

Exemplul 2.1. Un automat simplu
In figura 2.1 se prezinta un model de tip automat.

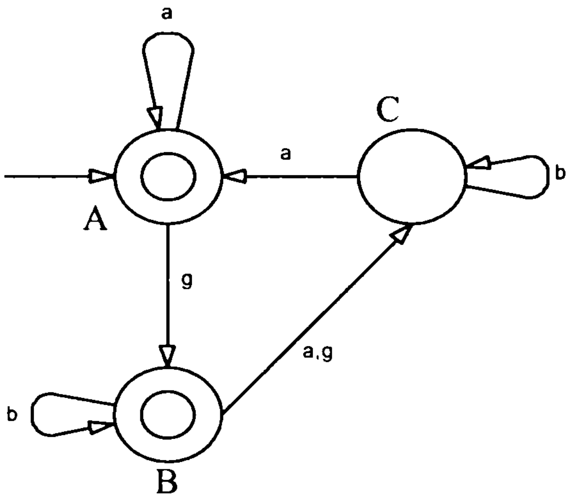


Figura 2.1: Diagrama de tranzitie a starilor pentru modelul de tip automat considerat

Automatul fiind definit prin $G = (X,\Sigma,\delta,\Gamma,x_0,X_m)$, unde

- $X = \{A,B,C\}$
- $\Sigma = \{a,b,g\}$
- $\delta(A,a) = A$
  $\delta(A,g) = B$
  $\delta(B,a) = \delta(B,g) = C$
  $\delta(B,b) = B$
  $\delta(C,b) = C$
  $\delta(C,a) = A$
- $x_0 = A$
- $X_m = \{A,B\}$

2.2.3. Limbajele ca reprezentante a automatelor [CL01] [Schm05] [Wohn02]

Conexiunea dintre limbaje si automate este usor de realizat prin inspectarea diagramelor de tranzitie a starilor unui automat. Se considera ca toate caile directe pot fi urmarite in diagrama de tranzitie a starilor pornind din starea initiala; se considera de asemenea ca toate aceste cai au finalitate intr-o stare marcata. Toate aceste considerente conduc la introducerea notiunii de ***limbaj generat*** si ***marcat*** de un automat.

Definitia 2.3 (Limbaje generate si marcate)
Limbajul generat de automatul $G = (X,\Sigma,\delta,\Gamma,x_0,X_m)$ este

$$L(G) := \{s \in \Sigma^* : \delta(x_0,s) \text{ este definit pentru un } x_0 \in X_0\} \tag{2.6}$$

Un *limbaj marcat* de *G* este

$$L_m(G) := \{s \in \Sigma^* : \delta(x_0, s) \in X_m \text{ pentru un } x_0 \in X_0\} \tag{2.7}$$

unde $\delta : X \times \Sigma^* \to X$.

Limbajul *L(G)* reprezinta toate caile directe care pot fi urmate in cadrul diagramei de stari, pornind din starea initiala; sirul corespunzator unei cai va fi constituit prin concadenarea etichetelor evenimentului corespunzatoare caii considerate. Altfel spus, un sir *s* exista in *L(G)* daca si numai daca el corespunde unei cai accesibile in diagrama de tranzitie a starilor, ceea ce este echivalent cu, daca sau numai daca δ este definita la (x_0,s). *L(G)* este prin definitie prefix-inchizator, ceea ce denota faptul ca o cale de tranzitie este validata numai daca toate prefixele sale sunt valide. Daca δ este o functie totala pe domeniul ei atunci este necesar ca $L(G) = \Sigma^*$.

Al doilea limbaj reprezentat prin *G*, L_m *(G)* este un subset a lui *L(G)* continand numai siruri *s* pentru care $\delta(x_0, s) \in X_m$, aceste siruri corespunzand cailor care au ca finalitate stari marcate din diagrama de tranzitie a starilor. Deoarece in cadrul spatiului starilor *X* nu toate starile trebuie sa fie marcate, limbajul L_m *(G)* generat prin *G* nu este neaparat un limbaj prefix-inchizator. Un limbaj marcat este de asemenea recunoscut de automat si invers, un automat este recunoscut de un limbaj dat.

2.2.4. Blocaje [CL01][Wohn02]

Pe baza definitilor enuntate pentru *G*, *L(G)* si L_m *(G)* rezulta:

$$L_m(G) \subseteq \overline{L_m(G)} \subseteq L(G) \tag{2.8}$$

Fie automatul *G* si o stare activa *x* ($x \in X$) pentru care $\Gamma(x) = 0$ si $x \notin X_m$. Aceasta situatie se numeste *blocaj* deoarece nici un eveniment aparut ulterior nu mai are efect asupra automatului (nu se mai executa tranzitii). In aceasta situatie se spune ca automatul este *blocat.* Daca se intampla un *blocaj*, atunci $\overline{L_m(G)}$ este un subset a lui *L(G)*, deoarece in *L(G)* toate sirurile care au la sfarsit starea *x* nu pot fi prefixe a sirurilor din L_m *(G)*.

O alta situatie care poate aparea este generata de faptul ca in *G* pot exista stari nemarcate (aceste stari pot fi atise din alte stari) dar nu mai exista tranzitii de iesire din acestea. Daca sistemul ajunge intr-o asemenea stare se spune ca este in bucla infinita. Astfel daca sistemul ajunge intr-o astfel de situatie, indiferent de evenimentul care se produce nu exista o tranzie satisfacuta, sarcina sistemului nefiind finalizata. Daca o situatie activare a unei bucle infinite se produce, atunci $\overline{L_m(G)}$ este un subset a lui *L(G)*. Orice sir din *L(G)* care duce la activarea starilor nemarcate nu pot fi prefixe a sirurilor in L_m*(G)*. Un sistem ajuns livelock se considera de asemenea ca este blocat.

Definitia 2.4 Blocajul

Automatul G este blocant daca

$$\overline{L_m(G)} \subset L(G) \tag{2.9}$$

unde setul de incluziune este complet, si este neblocant cand

$$\overline{L_m(G)} = L(G) \tag{2.10}$$

2.2.5. Automate nedeterministe [CL01][Wohn02]

In abordarile de pana acum s-a considerat ca in cadrul tranzitiilor starilor aparitia unui eveniment *e* are ca efect tranzitia dintr-o stare *x* intr-o stare noua *y* unic determinata. In multe cazuri, insa, se poate intampla ca dintr-o stare *x*, in urma aparitiei unui eveniment e, tranzitia sa nu se efecueze intr-o alta stare ci sa existe posibilitatea de tranzitie in mai multe stari. Pornind de la aceasta observatie se poate spune ca functia $\delta(x,e)$ nu va reprezenta o noua stare $y \in X$, ci va avea ca rezultat un set de posibile noi stari. De asemenea se doreste includerea evenimentului ε (tranzitia ε) in diagrama de tranzitie a starilor automatului considerat. Aceste tranzitii pot reprezenta evenimente care au ca efect schimbarea starii interne a SED, dar care nu sunt „observabile" de un observator extern. Astfel, un observator extern nu poate atasa un eveniment la o tranzitie efectuata, dar poate recunoaste tranzitia prin utilizarea lui ε.

Aceste doua schimbari, adica, setul de evenimente considerat nu este numai Σ ci $\Sigma \cup \{\varepsilon\}$, iar functia de transfer are domeniul si codomeniul diferite, domeniul fiind $X \times \Sigma \cup \{\varepsilon\}$ si co-domeniul 2^X, fapt care conduce la utilizarea notiunii de *automat nedeterminist*.

Definitia 2.5 Automatul nedeterminist

Un automat nedeterminist notat prin G este un secstuplu

$$G_{nd} = (X, \Sigma \cup \{\varepsilon\}, \delta_{nd}, \Gamma, x_0, X_m) \tag{2.11}$$

unde elementele componente au aceeasi interpretare ca in cazul definirii automatului determinist (definitia 2.2), cu urmatoarele diferente:

1. δ_{nd} *este o functie* $\delta_{nd} : X \times \Sigma \cup \{\varepsilon\} \to 2^X$, *astfel incat* , $\delta_{nd} \subseteq X$ *intotdeauna cand ea este definita;*
2. *starea initiala poate fi ea insasi un set de stari, astfel* $x_0 \subseteq X$.

Similar unui automat determinist, un automat nedeterminist genereaza limbajele:

$$\begin{aligned} L(G_{nd}) &= \{s \in \Sigma^* : \exists x \in x_0\,(\delta_{nd}(x,s) \text{ este definita})\} \\ L_m(G_{nd}) &= \{s \in L(G_{nd} : \exists x \in x_0\,(\delta_{nd}(x,s) \cap X_m \neq \emptyset)\} \end{aligned} \tag{2.12}$$

2.2.6. Operatii cu automate [CL01][Wohn02]

2.2.6.1. Operatii unare

Accesibilitatea

Din definitiile lui *L(G)* si $L_m(G)$, se poate observa ca din *G* pot fi sterse toate starile care nu pot fi atinse sau nu sunt accesibile din starea initiala x_0 prin utilizarea unor siruri din *L(G)*, fara ca limbajele generate si marcate de *G* sa fie afectate. Cand se „sterge” o stare, se sterg implicit si tranzitile atasate starii considerate. Aceasta operatie se noteaza prin *Ac(G)*. Formal,

$$Ac(G) := \{X_{ac}, \Sigma, \delta_{ac}, x_0, X_{ac,m}\} \tag{2.13}$$

unde,

$$X_{ac} = \{x \in X : \exists(s \in \Sigma^*)\ \delta(x_0, s) = x\}$$

$$X_{ac,m} = X_m \cap X_{ac}$$

$$\delta_{ac} = \delta|_{X_{ac} \times \Sigma \to X_{ac}}$$

Coaccesibilitatea

O stare *x* din *G* se spune ca este *coaccesebila* la X_m, sau simplu *coaccesibila*, daca exista un sir *s* in $L_m(G)$ care asigura trecerea prin *x*; aceasta inseamna ca exista o cale in diagrama de tranzitie a starilor a lui *G* care sa asigure trecerea din starea *x* intr-o stare marcata. Se indica astfel stergerea tuturor starilor lui *G* care nu sunt coaccesibile, rezultand *CoAc(G)*, unde *CoAc* contine numai partile coaccesibile. Alegerea partilor coaccesibile a unui automat inseamna construirea:

$$CoAc(G) := \{X_{coac}, \Sigma, \delta_{coac}, x_{0,coac}, X_m\} \tag{2.14}$$

unde,

$$X_{coac} = \{x \in X : \exists(s \in \Sigma^*\ \delta(x_0, s) \in X_m\}$$

$$x_{0,coac} = \begin{cases} x_0 & daca\ x_0 \in X_{coac} \\ nedefinit & daca\ x_0 \notin X_{coac} \end{cases}$$

$$\delta_{coac} = \delta|_{X_{coac} \times \Sigma \to X_{coac}}$$

Operatia trim

Un automat care este *accesibil* si *coaccesibil* se spune ca este un automat *trim*. Operatia Trim se defineste ca fiind:

$$Trim(G) := CoAc[Ac(G)] = Ac[CoAc(G)]\,. \tag{2.15}$$

Complementul

Fie automatul trim $G = (X, \Sigma, \delta, \Gamma, x_0, X_m)$ care marcheaza limbajul $L \subseteq \Sigma^*$. Astfel *G* genereaza limbajul $\overline{L}$. Se poate construi un alt automat, notat G^{comp}, care va fi marcat prin limbajul $\Sigma^* \backslash L$.

G^{comp} este construit in doi pasi dupa cum urmeaza:

1. Se completeaza functia de tranzitie δ a lui *G* astfel incat sa fie o functie completa. Noua functie de tranzitie este δ_{tot}. Aceasta se realizeaza prin adaugarea unei noi stari x_d in *X*, stare denumita *„stare*

moarta". Toate rezultatele nedefinite a lui $\delta(x,e)$ din G sunt asignate la starea x_d. Formal,

$$\delta_{tot}(x,e)=\begin{cases}\delta(x,e) & daca\ e\in\Gamma(x)\\ x_d & daca\ e\notin\Gamma(x)\end{cases} \tag{2.16}$$

In plus, se considera $\delta_{tot}(x_d,e)=x_d$ pentru toate evenimentele $e\in\Sigma$.

Cu noua stare x_d (stare nemarcata) rezulta un automat nou:

$$G_{tot}=(X\cup\{x_d\},\Sigma,\delta_{tot},x_0,X_m) \tag{2.17}$$

care genereaza $L(G_{tot})=\Sigma^*$ si $L_m(G_{tot})=L$.

2. Se schimba starile marcajelor a tuturor starilor din G_{tot} prin marcarea tuturor starilor nemarcate (inclusiv x_d) si se demarcheaza toate starile marcate. Astfel, se defineste

$$G^{comp}=(X\cup\{x_d\},\Sigma,\delta_{tot},x_0,(X\cup\{x_d\})\setminus X_m) \tag{2.18}$$

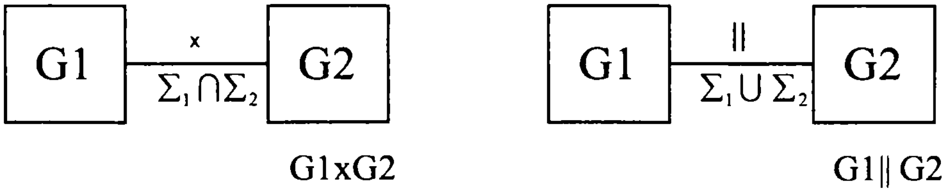
cu, $L(G^{comp})=\Sigma$ si $L_m(G^{comp})=\Sigma^*\setminus L_m(G)$.

2.2.6.2. Operatii de compunere [CL01][Wohn02]

Se definesc doua operatii de compunere care pot fi aplicate automatelor, si anume: *produsul*, notat prin „x", si *compunerea paralela*, notata prin „‖". Compunerea paralela mai este denumita si *compunerea sincrona* iar produsul mai este denumit si *operatie de compunere total sincrona*. Aceste doua operatii modeleaza in doua forme comportarea comuna a unui set de automate. Pentru exemplificarea operatiilor se considera doua automate G_1 si G_2 definite prin:

$$G_1=(X_1,\Sigma_1,\delta_1,\Gamma_1,x_{01},X_{m1})\quad \text{si}\quad G_2=(X_2,\Sigma_2,\delta_2,\Gamma_2,x_{02},X_{m2}) \tag{2.19}$$

Ambele automate sunt considerate accesibile, si nu este obligatoriu sa fie coaccesibile. Aceste doua automate sunt conectate ca in figura 2.2.



a) b)

Figura 2.2: Modul de interconectare a automatelor $G1$ si $G2$. a) operatia „x"; b) operatia „‖"

Produsul

Produsul automatelor G_1 si G_2 este un automat

$$G_1 \times G_2 := Ac(X_1 \times X_2, \Sigma_1 \cap \Sigma_2, \delta, \Gamma_{1\times 2}, (x_{01}, x_{02}), X_{m1} \times X_{m2}) \qquad (2.20)$$

unde

$$\delta((x_1, x_2), e) := \begin{cases} (\delta_1(x_1, e), \delta_2(x_2, e)) & daca\ e \in \Gamma_1(x_1) \cap \Gamma_2(x_2) \\ nedefinita & daca\ e \notin \Gamma_1(x_1) \cap \Gamma_2(x_2) \end{cases}$$
$$(2.21)$$

si $\Gamma_{1\times 2}(x_1, x_2) = \Gamma_1(x_1) \cap \Gamma_2(x_2)$.

In cadrul unui automat rezultat in urma produsului a doua automate, tranzitiile celor doua automate primare trebuie sa fie intotdeauna sincronizate pe un eveniment comun, un eveniment care este continut in $\sum_1 \bigcap \sum_2$. Astfel $G_1 \times G_2$ reprezinta „lock-step" conectarii lui G_1 si G_2, unde un aparitia unui eveniment este validat daca si numai daca el apare in ambele automate. Starile automatului $G_1 \times G_2$ sunt notate prin perechi, unde primul element este starea curenta a lui G_1 iar al doilea element este starea curenta a lui G_2. Aceasta afirmatie este usor de verificat deoarece limbajele lui $G_1 \times G_2$ sunt:

$$L(G_1 \times G_2) = L(G_1) \cap L(G_2) \qquad (2.22)$$
$$L_m(G_1 \times G_2) = L_m(G_1) \cap L_m(G_2). \qquad (2.23)$$

Aceasta prezinta faptul ca intersectia a doua limbaje poate fi implementata prin realizarea produsului corespunzator al automatelor pe care limbajele le reprezinta. Daca $\sum_1 \bigcap \sum_2 = 0$, atunci $L(G_1 \times G_2) = \{\varepsilon\}$; $L_m(G_1 \times G_2)$ va fi de asemenea 0 sau $\{\varepsilon\}$, functie de marcajul starii initiale (x_{01}, x_{02}).

Compunerea paralela

Compunerea paralela a automatelor G_1 si G_2

$$G_1 \| G_2 := Ac(X_1 \times X_2, \Sigma_1 \cup \Sigma_2, \delta, \Gamma_{1\|2}, (x_{01}, x_{02}), X_{m1} \times X_{m2}) \qquad (2.24)$$

unde

$$\delta((x_1, x_2), e) := \begin{cases} (\delta_1(x_1, e), \delta_2(x_2, e)) & daca\ e \in \Gamma_1(x_1) \cap \Gamma_2(x_2) \\ (\delta_1(x_1, e), x_2) & daca\ e \in \Gamma_1(x_1) \setminus \Sigma_2 \\ (x_1, \delta_2(x_2, e)) & daca\ e \in \Gamma_2(x_2) \setminus \Sigma_1 \\ nedefinita & altfel \end{cases}$$
$$(2.25)$$

si $\Gamma_{1\|2}(x_1,x_2)=[\Gamma_1(x_1)\cap\Gamma_2(x_2)]\cup[\Gamma_1(x_1)\setminus\Sigma_2]\cup[\Gamma_2(x_2)\setminus\Sigma_1]$.

In cadrul compunerii paralele, un eveniment comun, care este continut in $\Sigma_1\cap\Sigma_2$, poate fi executat numai daca ambele automate il executa simultan. Astfel, cele doua automate sunt sincronizate pe evenimente comune. In cazul unor evenimente „private" , care sunt continute in $(\Sigma_2\setminus\Sigma_1)\cup(\Sigma_1\setminus\Sigma_2)$, asupra lor nu exista constrangeri in a fi executate. In acest tip de conexiune, o componenta poate executa un eveniment privat fara participarea altor componente; intotdeauna un eveniment comun este executabil daca ambele componente pot sa-l execute.

Daca $\Sigma_1=\Sigma_2$, operatia de compunere paralela se reduce la operatia de produs, asftel incat toate tranzitiile sunt fortate sa se efectueze sincronizat. Daca $\Sigma_1\cap\Sigma_2=0$, nu exista tranzitii sincronizate si $G_1\| G_2$ are un comportament concurent a lui G_1 si G_2.

Operatia de compunere paralela este:

1.comutativa: $G_1\| G_2=G_2\| G_1$

2.asociativa: $G_1\|(G_2\|G_3)=(G_1\| G_2)\|G_3$.

Pentru a putea face referiri la limbajele generate si marcate de catre $G_1\| G_2$, se defineste mai intai *proiectia* ca fiind:

$$P_i:(\Sigma_1\cup\Sigma_2)^*\to\Sigma_i^* \quad pentru \quad i=1,2 \tag{2.26}$$

dupa cum urmeaza:

$$P_i(\varepsilon):=\varepsilon$$

$$P_i(e):=\begin{cases} e & daca \quad e\in\Sigma_i \\ \varepsilon & daca \quad e\notin\Sigma_i \end{cases} \tag{2.27}$$

$$P_i(se):=P_i(s)P_i(e) \quad pentru \quad s\in(\Sigma_1\cup\Sigma_2)^*, e\in(\Sigma_1\cup\Sigma_2).$$

Fie doua seturi de evenimente unde unul este un subset al celuilalt, adica $\Sigma_1\cup\Sigma_2$ si Σ_i in acest caz, acest mod de proiectie sterge evenimentele dintr-un sir format din setul larg de evenimente $(\Sigma_1\cup\Sigma_2)$ care nu apartin setului mic de evenimente (oricare din Σ_1 sau Σ_2). Acest tip de proiectie se mai numeste si *proiectie naturala*.

In multe situatii se mai utilizeaza si *proiectia inversa*:

$$P_i^{-1}:\Sigma_i^*\to 2^{(\Sigma_1\cup\Sigma_2)^*} \tag{2.28}$$

definita dupa cum urmeaza:

$$P_i^{-1}(t):=\{s\in(\Sigma_1\cup\Sigma_2)^*:P_i(s)=t\}. \tag{2.29}$$

Fie un sir care face parte din setul mic de evenimente (Σ_i), proiectia inversa returneaza un set al tuturor sirurilor din setul larg de evenimente $(\Sigma_1\cup\Sigma_2)$ care proiecteaza cu P_i la sirurile date.

Proiectia P_i si inversa sa P_i^{-1} sunt extensii la limbaje prin simpla aplicare a lor la toate sirurile din limbaj. Pentru $L\subseteq(\Sigma_1\cup\Sigma_2)^*$,

$$P_i(L):=\{t\in\Sigma_i^* : \exists s\in L \ (P_i(s)=t)\} \tag{2.30}$$

si pentru $L_i\subseteq\Sigma_i^*$,

$$P_i^{-1}(L_i) := \{s\in(\Sigma_1\cup\Sigma_2) : \exists t\in L_i \ (P_i(s)=t)\}. \tag{2.31}$$

Limbajele rezultate din operatia de compunere paralela sunt:

1. Limbajul generat: $L(G1\|G2)=P_1^{-1}[L(G_1)]\cap P_2^{-1}[L(G_2)]$

(2.32)

2. Limbajul marcat: $L_m(G_1\|G_2)=P_1^{-1}[L_m(G_1)]\cap P_2^{-1}[L_m(G_2)]$.

(2.33)

Caracterizarea comportamentelor automatelor prin intermediul operatiei de compunere paralela utilizand proiectia corespunzatoare se reduce in fapt la definirea compunerii paralele a limbajelor corespunzatoare [Cas01].

Fie un limbaj $L_i\subseteq\Sigma_i^*$ si P_i , definit dupa cum a fost prezentat mai sus, rezulta:

$$L_1\|L_2 := P_1^{-1}(L_1)\cap P_2^{-1}(L_2) \tag{2.34}$$

2.2.7. Transformarea unui automat nedeterminist in automat determinist [CL01]

In [CL01] se prezinta un algoritm pentru transformarea unui automat nedeterminist intr-un automat determinist prin intermediul unui limbaj „salvator". Rezultatul unei astfel de transformari este un automat determinist denumit *observator* corespondent la automatul nedeterminist; *observatorul* automatului nedeterminist G_{nd} se noteaza ca fiind G_{obs}.

Observatorul are rolul de a urmarii evolutia automatului nedeterminist estimand traiectorile tranzitiilor functie de evenimentele aparute.

Procedura de construire a unui observator G_{obs} pentru un automat nedeterminist G_{nd} este descrisa in continuare.

Fie automatul nedeterminist $G_{nd}=(X,\Sigma\cup\{\varepsilon\},\delta_{nd},x_0,X_m)$. Observatorul $G_{obs}=(X_{obs},\Sigma,\delta_{obs},x_{0,obs},X_{m,obs})$ se construieste in astfel:

Pasul 1. Porneste cu $X_{obs}=2^X\backslash 0$ (2.35)

Pasul 2. Pentru fiecare stare $x\in X$ se defineste $UR(x):=\delta_{nd}(x,\varepsilon)$ (2.36)

UR reprezinta „raza de actiune neobservabila" de la evenimentul ε, tranziliile nefiind observabile, in aceasta situatie lucrandu-se cu extensia functiei δ_{nd}. Pentru un set B, se defineste

$$UR(B)=\bigcup_{x\in B}UR(x) \qquad (2.37)$$

Pasul 3. Se defineste $x_{0,obs}=UR(x_0)$ (2.38)

Pasul 4. Pentru fiecare $S\subseteq X$ si $e\in\Sigma$, se defineste

$$\begin{aligned}\delta_{obs}(S,e)&=UR(\{x\in X:\exists x_e\in S\,[x\in\delta_{nd}(x_e,e)]\})\\&=\{x\in X:\exists x_e\in S\,[x\in\delta_{nd}(x_e,e)]\}\end{aligned} \qquad (2.39)$$

din definitia extensiei lui δ_{nd} la siruri.

Pasul 5. $X_{m,obs}=\{S\subseteq X:S\cap X_m\neq 0\}$ (2.40)

Pasul 6. In practica, in prima faza sunt realizate numai partile accesibile ale lui G_{obs}. Setul starilor astfel rezultat X_{obs} fiind un subset a lui 2^X.

Proprietatiile cele mai importante ale lui G_{obs} sunt:

1. G_{obs} este un automat determinist
2. $L(G_{obs})=L(G_{nd})$
3. $L_m(G_{obs})=L_m(G_{nd})$.

2.3. Retele Petri

O alternativa la automatele netemporizate ca modele ale sistemele cu evenimente discrete (SED) este data de retelele Petri. Acest concept a fost dezvoltat de C.A. Petri in anii '60 [Petri62]. Retele Petri sunt inrudite cu automatele in sensul ca prezinta explicit functia de tranzitie a unui SED. Analog automatelor, o retea Petri este un instrument care gestioneaza evenimente in functie de anumite reguli impuse.

Retelele Petri sunt descrise grafic, rezultand graful retelei Petri, graf care este intuitiv si contine toate informatile legate de sistemul analizat.

2.3.1. Reţele Petri netemporizate [Mur98][Pastr97][CL01]

O reţea Petri este alcătuită dintr-un *graf orientat* notat *N* şi o *stare iniţială* x_0, denumită *marcaj iniţial*. Graful *N* al reţelei Petri este orientat, ponderat si bipartit (arcele nu pot conecta direct două noduri de acelaşi tip), şi constă în două tipuri de noduri, denumite *poziţii* (locaţii) şi respectiv *tranziţii*. Legătura dintre poziţii şi tranziţii se face prin intermediul *arcelor*. Acestea pornesc fie de la o poziţie la o tranziţie, fie de la o tranziţie la o poziţie. Este important de reţinut că nu există arce care să conecteze direct două poziţii între ele, sau două tranziţii între ele. Pentru simbolizarea grafică, se utilizează cercuri în cazul poziţiilor şi bare (dreptunghiuri) în cazul tranziţiilor.

Fiecărui arc îi corespunde o *etichetă* care semnifică ponderea lui (numere întregi şi pozitive). Un arc cu ponderea *k* poate fi privit ca o mulţime de *k* arce paralele cu o pondere unitară. Etichetele a căror valoare este unitară se omit în reprezentarea grafică uzuală, lipsa ponderii ducând la considerarea acesteia ca fiind implicit egală cu unu.

Fiecărei poziţii *i* se atribuie un număr întreg şi pozitiv prin intermediul unui marcaj sau a unei stări. Se spune despre o poziţie *p* că este marcată cu *k* jetoane dacă acesteia îi este atribuit marcajul $k \geq 0$. În modul grafic cercului corespunzător poziţiei *p* îi sunt atribuite *k* jetoane. Marcajul *x* este un vector *n*-dimensional al poziţiilor *p* ale grafului. Dimensiunea *n* a marcajului *x* reprezintă numărul total al poziţiilor. Fiecare componentă a vectorului *x* corespunzătoare unei poziţii *p* redă numărul de jetoane din această poziţie (notaţia folosită este *x(p)*).

În problemele de modelare ce utilizează conceptele de condiţii şi evenimente, poziţiile reprezintă condiţii şi tranziţiile reprezintă evenimente. Fiecărei tranziţii i se atribuie un număr întreg şi pozitiv de poziţii de intrare şi de ieşire. Poziţiile de intrare reprezintă pre-condiţiile şi poziţiile de ieşire reprezintă post-condiţiile evenimentului respectiv. În cazul în care unei poziţii îi corespunde un jeton, atunci valoarea logică a condiţiei asociate poziţiei respective este considerată ca fiind „adevărat".

Definiţia 2.6. Retea Petri netemporizata

Un graf (sau o structură) de reţea Petri este un graf bipartit ponderat $PN = (P,T,F,W,M_0)$ *(topologia reţelei), unde:*

- *P reprezintă mulţimea finită de poziţii, unde* $P = \{p_1, p_2, p_3,..., p_n\}$;
- *T reprezintă mulţimea finită de tranziţii, unde* $T = \{ t_1, t_2, t_3,..., t_m\}$;
- $F \subseteq (P \times T) \cup (T \times P)$ *reprezintă mulţimea arcelor de la poziţii la tranziţii şi de la tranziţii la poziţii, fiecare arc fiind reprezentat prin* (p_i, t_j), *respectiv* (t_j, p_i), *unde* $i,j \in N$;
- $W : A \to \{1,2,3...\}$ *reprezintă funcţia de ponderare a arcelor.*
- $M_0 : P \to \{0,1,2,,...\}$ *reprezinta functia de marcaj initial.*

Exemplul 2.2. Un model de retea Petri

In figura 2.3 se prezinta structura unei retele Petri.

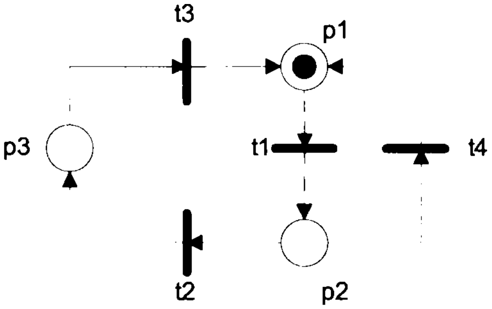


Figura 2.3: Model de retea Petri

Reteaua Petri considerata este exprimata prin formalism matematic dupa cum urmeaza:

$$PN = (P,T,F,W,M_0)$$

unde:

- $P = \{p1,p2,p3\}$;
- $T = \{t1,t2,t3,t4\}$;
- $F = \{(p1,t1),(p2,t2),(p2,t4),(p3,t3)\} \cup \{(t1,p2),(t3,p1),(t4,p1)\}$

- $W = \{w(p1,t1) = 1, w(p2,t2) = 1, w(p2,t4) = 1, w(p3,t3) = 1\},$
  $w(t1,p2) = 1, w(t3,p1) = 1, w(t4,p1) = 1\}$
- $M_0 = \{1,0,0\}$

Mulţimile P şi T trebuie să fie disjuncte $P \cap T = \varnothing$ şi în plus trebuie să satisfacă condiţia $P \cup T \neq \varnothing$.

În descrierea unei reţele Petri, se foloseşte $I(t_j)$ pentru a reprezenta mulţimea poziţiilor de intrare pentru tranziţia t_j şi $O(t_j)$ pentru a reprezenta mulţimea poziţiilor de ieşire pentru tranziţia t_j:

$$\begin{aligned} &I(t_j) = \{p_i \in P : (p_i, t_j) \in F; i, j \in N\} \text{ respectiv} \\ &O(t_j) = \{p_i \in P : \{t_j, p_i) \in F; i, j \in N\} \end{aligned} \tag{2.41}$$

Pentru a descrie tranziţiile de intrare pentru o poziţie p_i se foloseşte $I(p_i)$, iar pentru a descrie tranziţiile de ieşire se foloseşte $O(p_i)$.

Unui SED i se poate asocia un graf de reţea Petri, poziţiile lui descriind condiţiile în care evenimentele sistemului pot avea loc. Condiţiilor descrise de o poziţie le sunt asociate *jetoane*. Un *jeton* se reprezintă grafic printr-o bulină în interiorul poziţiei şi semnifică faptul că este îndeplinită condiţia descrisă de poziţie. Îndeplinirea unei condiţii caracteristice unei poziţii poate fi semnalată printr-unul sau mai multe jetone.

Asocierea de jetoane poziţiilor unui graf determină *marcarea* acestuia. Deci fiecărui graf asociat unei reţele Petri *(P, T, F, W)* îi corespunde un *marcaj x*.

Orice *marcaj x* este o funcţie de forma $x : P \to N$ ce defineşte un vector linie $x = [x(p_1)x(p_2).....x(p_n)]$, n reprezentând numărul de poziţii din graf. Numărul de jetoane corespunzător unei poziţii p_i îl regăsim în poziţia i a vectorului x. În urma marcării, o reţea Petri va deveni o reţea marcată şi va fi simbolizată prin cvintuplul *(P, T, F, W, x)*.

În urma marcării poziţiilor unei reţele Petri putem vorbi de starea reţelei. *Starea reţelei Petri* se defineşte ca fiind vectorul marcajelor. Deoarece numărul jetoanelor asociate unei poziţii nu este limitat putem avea un număr de stări infinit. Spaţiul stărilor, X, al unei reţele Petri cu n poziţii este n-dimensional, iar marcajul iniţial al reţelei va fi notat cu x_0.

O tranziţie este validată dacă în fiecare poziţie de intrare a tranziţiei respective se găseşte cel puţin un jeton.

Definiţia 2.7. Validarea unei tranzitii

O tranziţie $t_j \in T$ *a unei reţele Petri se numeşte validată dacă* $x(p_i) \geq w(p_i, t_j)$ *oricare ar fi* $p_i \in I(t_j)$.

Definiţia 2.8. Dinamica retelelor Petri

Funcţia tranziţiilor de stare, $f : N^n \times T \to N^n$ *a unei reţele Petri* $PN = (P, T, F, W, x, M_0)$ *marcate, este definită pentru tranziţia* $t_j \in T$ *dacă şi numai dacă* $x(p_i) \geq w(p_i, t_j)$ *oricare ar fi* $p_i \in I(t_j)$.

Dacă funcţia tranziţiilor de stare, $f(x, t_j)$*, este definită, atunci se noteaza* $x' = f(x, t_j)$*, unde* $x'(p_i) = x(p_i) - w(p_i, t_j) + w(t_j, p_i)$, $i = 1, ..., n$.

Definitia 2.9. Stari accesibile

Setul starilor accesibile a unei retele Petri $PN = (P,T,F,W,x,M_0)$ *este:*

$$R[(P,T,F,W,x,M_0)] := \{y \in N^n : \exists s \in T^*(f(x,s) = y)\}$$

2.3.2. Analiza proprietăţilor comportamentale [Mur89][Pastr97][CL01]

Topologia unei reţele Petri, precum şi marcajul său iniţial determină proprietăţile comportamentale ale acestei reţele. Proprietăţile comportamentale se vor analiza cu ajutorul grafului de acoperire,arborelui de acoperire şi a ecuaţiei de stare.

Accesibilitate. Un marcaj x_k este numit *marcaj accesibil* din marcajul iniţial x_0 dacă există o secvenţă de executări de tranziţii care transformă marcajul iniţial x_0 în marcajul x_k. Secvenţa de executări de tranziţii se notează uzual prin: $\sigma = x_0 t_{j1} x_1 t_{j2} \ldots . t_{jk} x_k$ sau prin $\sigma = t_{j1} t_{j2} \ldots . t_{jk}$. Cea de-a doua variantă este utilizata in cazul in care nu interesează succesiunea de marcaje.

Mărginire. Orice reţea Petri se spune că este *k-mărginită* (sau mărginită) dacă numărul de jetoane corespunzător fiecărei poziţii este maxim *k* pentru orice marcaj accesibil din starea iniţială x_0: $x(p_i) \le k$, oricare ar fi poziţia p_i considerată şi $x \in R(x_0)$.

Viabilitate. O reţea Petri $PN = (P,T,F,W,x,M_0)$ este *viabilă* dacă este posibil să fie executată orice tranziţie *t* a reţelei, indiferent de marcajul care a fost atins pornind din x_0. Pentru a se executa tranziţia *t* se pot executa un număr finit de alte tranziţii intermediare. De asemenea, se spune că marcajul x_0 este un *marcaj viabil* pentru $PN = (P,T,F,W,x,M_0)$. În cazul în care exista un marcaj pentru care nici o tranziţie a reţelei nu mai poate fi executată, această situaţie se numeşte *blocaj*.

Reversibilitate. Dacă pentru o reţea Petri $PN = (P,T,F,W,x,M_0)$ marcajul x_0 este accesibil pornind din orice marcaj $x \in R(x_0)$ atunci reţeaua se spune că este *reversibilă*.

Acoperire. Pentru o reţea Petri $PN = (P,T,F,W,x,M_0)$, un marcaj $x \in R(x_0)$ se spune că este *acoperibil* dacă există un marcaj $x' \in R(x_0)$ astfel încât $x'(p) \ge x(p)$ pentru fiecare poziţie *p* a reţelei.

Persistenţă. O reţea Petri $PN = (P,T,F,W,x,M_0)$ că este *persistentă* dacă pentru oricare două tranziţii validate, executarea uneia dintre ele nu o invalidează pe cealaltă.

Distanţă sincronă. Considerand pentru o reţea Petri $PN = (P,T,F,W,x,M_0)$ două tranziţii t_1 şi t_2 , d*istanţa sincronă* dintre aceste două tranziţii este definita ca:

$$d(t_1,t_2) = \max_{\sigma}\left|\bar{\sigma}(t1) - \bar{\sigma}(t_2)\right| \tag{2.42}$$

unde σ notează o secvenţă de executări pornind din orice marcaj $x \in R(x_0)$, iar $\bar{\sigma}(t_i)$, $i = 1, 2$ notează numărul de executări ale tranziţiei t_i, $i = 1, 2$ în secvenţa σ.

Imparţialitate

a) Imparţialitate de tip mărginire. Considerand o reţea Petri $PN = (P,T,F,W,x,M_0)$ cu tranziţiile t_1 şi t_2. Cele două tranziţii se află într-o relaţie de *imparţialitate de tip mărginire*, dacă numărul de executări pe care le poate avea una din tranziţii este finit, atât timp cât cealaltă tranziţie nu se execută niciodată. La rândul ei, o reţea Petri $PN = (P,T,F,W,x,M_0)$ este *imparţială de tip mărginire* dacă oricare două tranziţii ale reţelei se află şi ele în relaţia de imparţialitate de tip mărginire.

b) Imparţialitate de tip global. O secvenţă de executări de tranziţii σ este *imparţială de tip global* dacă ea este finită. O reţea Petri $PN = (P,T,F,W,x,M_0)$ este *imparţială de tip global* dacă orice secvenţă σ este imparţială pornind de la un marcaj oarecare $x \in R(x_0)$. Orice reţea imparţială de tip mărginire este şi imparţială de tip global, insa o reţea imparţială de tip global nu este însă neapărat imparţială de tip mărginire.

2.3.3. Tehnici generale de analiză a proprietăţilor comportamentale [Mur89][Pastr97][CL01]

Arborele de acoperire. Pentru orice reţea Petri $PN = (P,T,F,W,x,M_0)$ modificarea marcajelor ca urmare a executării tranziţiilor poate fi reprezentată sub forma unui arbore denumit *arbore de acoperire*. Pentru acest arbore rădăcina este x_0, iar marcajele generate reprezintă nodurile. Arcele arborelui corespund executării unei tranziţii care duce la transformarea marcajului asociat nodului de plecare. Noul marcaj se va regăsi la celălalt capăt al arcului.

In [Pastr97][Cas01] se prezinta in detaliu modul de construire al arborelui de acoperire si de interpretare a acestuia.

Graful de acoperire. Pe baza arborelui de acoperire se pot realiza grafurile de acoperire şi grafurile de accesibilitate. *Graful de acoperire* al unei reţele Petri $PN = (P,T,F,W,x,M_0)$ este un graf orientat $G = (V, E)$. V reprezintă mulţimea nodurilor şi este dată de mulţimea tuturor marcajelor distincte din arborele de acoperire. E este mulţimea arcelor orientate şi este folosită pentru a face legătura între două marcaje x_i, x_j din mulţimea V dacă există o tranziţie t_k în urma căreia se obţin aceste marcaje. Arcele din mulţimea E corespund arcelor din arborele de acoperire.

In [Pastr97][CL01] se prezinta in detaliu modul de construire al grafului de acoperire si de interpretare a acestuia.

Ecuaţia de stare. Se consideră o reţea Petri $PN = (P,T,F,W,x,M_0)$ cu n poziţii şi m tranziţii $(n,m \in N)$. Se numeşte *matrice de incidenţă* a reţelei Petri o matrice $A = \lfloor a_{ij} \rfloor$ de dimensiune $m \times n$ ale cărei elemente $a_{ij} \in Z$ cu $a_{ij} = a_{ij}^{+} - a_{ij}^{-}$, $i = 1,...,m$, $j = 1,....,n$, unde:

- $a_{ij}^{+} = w(t_i, p_j)$- este ponderea arcului de la tranziţia t_i către poziţia sa de ieşire p_j;
- $a_{ij}^{-} = w(p_j, t_i)$- este ponderea arcului către tranziţia t_i de la poziţia sa de intrare p_j.

Matricea de incidenţă $A^{+} = \left[a_{ij}^{+}\right]$ de dimensiune $m \times n$ se numeşte *matrice de incidenţă de ieşire*, iar matricea $A^{-} = \left[a_{ij}^{-}\right]$ de aceeaşi dimensiune se numeşte *matrice de incidenţă de intrare*. Matricea de incidenţă A se poate construi pe baza matricilor A^{+} şi A^{-} astfel:

$$A = A^{+} - A^{-}, \tag{2.43}$$

a_{ij}^{-} reprezintă numărul de jetoane scoase din poziţia p_j în urma executării tranziţiei t_i. a_{ij}^{+} reprezintă numărul de jetoane adăugate în poziţia p_j în urma executării tranziţiei t_i. Tranziţia t_i este validată de un marcaj x dacă şi numai dacă:

$a_{ij}^{-} \leq x(p_j)$ cu $j = 1, ..., n$.

Dacă se consideră o secvenţă de executii de tranziţii şi se presupune că cea de-a k-a executare din această secvenţă are loc în tranziţia t_i, atunci tranziţia desemnată prin t_i se află pe locul k în secvenţa de executări:

$$\sigma = \underbrace{t_a}_{locul\ 1} \underbrace{t_b}_{locul\ 2} \;.....\; \underbrace{t_i}_{locul\ k} \;..... \tag{2.44}$$

Fiecare executare de secvenţă k se poate reţine într-un vector coloană u_k de dimensiune $1 \times m$ prin plasarea valorii 1 pe poziţia i corespunzătoare tranziţiei t_i. Se observă că vectorul coloană rezultat din produsul $A^T u_k$ reprezintă chiar cea de-a i-a coloană a matricii A^T sau cea de-a i-a linie a matricii A. Rezultă că vectorul coloană $A^T u_k$ conţine toate schimbările de marcaj rezultate la a k-a executare de secvenţă considerată în tranziţia t_i. Pe de altă parte, schimbarea de marcaj în urma celei de-a k-a executări poate fi scrisă drept $x_k - x_{k-1}$, unde x_{k-1}, x_k notează marcajul după cea de-a $(k-1)$-a şi respectiv a k-a executare din secvenţa considerată. Ţinând cont că asupra lui k şi asupra lui i nu au fost impuse nici un fel de condiţii, putem considera în general că schimbarea de marcaj după cea de-a k-a executare este de forma:

$$x_k - x_{k-1} = A^T u_k, \quad k = 1,2,...... \tag{2.45}$$

Forma echivalentă uzual folosită a acestei relaţii reprezintă *ecuaţia de stare* a reţelei Petri şi este următoarea:

$$x_k = x_{k-1} + A^T u_k, \quad k = 1,2,...... \tag{2.46}$$

Vectorul u_k din această relaţie se numeşte *vector de executare* sau *vector de control*.

Pe baza ecuaţiei de stare *(2.46)*, se poate aborda problema de accesibilitate. Presupunem că un marcaj de destinaţie x_d este accesibil din x_0 prin secvenţa de executări $u_1, u_2, u_3, \ldots, u_d$.
Din ecuaţia de stare pentru $k = 1, 2, \ldots, d$ şi prin însumare se obţine:

$$x_d = x_0 + A^T \cdot \sum_{k=1}^{d} u_k \text{ echivalentă cu } A^T s = \Delta x\,, \tag{2.47}$$

unde: $s = \sum_{k=1}^{d} u_k$, $\Delta x = x_d - x_0$.

Vectorul coloană s, de dimensiune $m \times 1$, are toate elementele întregi, pozitive şi se numeşte *vectorul numărului de executări posibile.* Cel de-al i-lea element al vectorului s $(i=1,2,\ldots,m)$ conţine numărul de executări ale tranziţiei t_i $(i=1,2,\ldots,m)$ în secvenţa ce transformă x_0 în x_d.

Teorema 2.1 Conditia de tip necesar pentru accesibilitate
Dacă marcajul x_d este accesibil din x_0, atunci are loc egalitatea:

$$rang A^T = rang\left[A^T \vdots \Delta x\right] \tag{2.48}$$

unde A reprezintă matricea de incidenţă, iar Δx este diferenţa de marcaj definită prin $\Delta x = x_d - x_0$ Δx.

Forma contrară a reciprocei acestei teoreme oferă condiţii suficiente pentru proprietatea de neaccesibilitate. Această teoremă nu garantează atingerea marcajului x_d pornind din x_0.

Teorema 2.2. Conditia suficienta pentru neaccesibilitate
Un marcaj x_d nu este accesibil din x_0 dacă egalitatea $rang A^T = rang\left[A^T \vdots \Delta x\right]$ nu este satisfăcută.

Demonstratiile acestor teoreme se gasesc in [Pastr97].

2.3.4. Tehnici generale de analiză structurala

2.3.4.1. Aspecte generale ale analizei structurale [Mur89] [Pastr97][CL01][Kemp04]

Proprietatile utilizate in analiza structurala ale retelelor Petri sunt:

Controlabilitatea. O retea Petri $PN = (P,T,F,W,x,M_0)$ este complet *controlabila* daca orice marcaj este accesibil pornind din oricare alt marcaj. Conditia necesara:

$$rang\, A = m \ (m - numarul\ pozitilor) \tag{2.49}$$

Structural Viabilitatea. O retea Petri $PN = (P,T,F,W,x,M_0)$ este *structural viabila* daca exista un marcaj initial pentru care reteaua este viabila;

Structural Marginimea. O retea Petri $PN = (P,T,F,W,x,M_0)$ este *structural marginita* daca este marginita pentru orice marcaje initiale finite. Conditia necesara si suficienta:

$$\exists y > 0 : y^T A \leq 0$$

$$M = M_0 + Au \ (u \geq 0) \Rightarrow y^T M = y^T M_0 + y^T Au \Rightarrow y^T M \leq y^T M_0) \qquad (2.50)$$

Conservativitatea. O retea Petri $PN = (P,T,F,W,x,M_0)$ este *conservativa* daca exista un vector pozitiv y astfel incat $M^T y = M_0^T y$ pentru toate marcajele M.

Conditia necesara si suficienta: $\exists x > 0 : x^T A = 0$

Consistenta. O retea Petri $PN = (P,T,F,W,x,M_0)$ este *consistenta* daca exista un marcaj M si un vector de stare u astfel incat u porneste si se termina in M si fiecare tranzitie efectuata apare o singura data in u. Conditia necesara si suficienta este: $\exists y > 0 : Ay = 0$.

Repetitivitatea. O retea Petri $PN = (P,T,F,W,x,M_0)$ este repetitiva daca exista un marcaj initial M_0, astfel incat orice tranzitie apare infinit de des in σ.

2.3.4.2. Analiza structurala pe baza invariantilor [Mur89] [Pastr97][CL01] [Kemp04]

Se considera o retea Petri $PN = (P,T,F,W,x,M_0)$ descrisa prin matricea de incidenta A de dimensiune nxm. Un vector de dimensiune m $y>0$, cu elementele numere intregi se numeste *invariant P* al retelei *PN* daca $A \cdot y = 0$. Un vector de dimensiune n $x>0$, cu elementele numere intregi, se numeste *invariant T* al retelei *PN* daca $A^T \cdot x = 0$.

Pentru o retea Petri pot exista mai multi invarianti P si/sau mai multi invarianti T.

Teorema 2.3 Determinarea numarului de invarianti.
Daca matricea de incidenta A (de dimensiune nxm) a retelei Petri:

$$PN = (P,T,F,W,x,M_0)$$

are rangul r, atunci:

- *reteaua poseda m-r invarianti P de baza, iar fiecare invariant P al retelei PN poate fi scris drept o combinatie liniara a acestora;*
- *reteaua poseda n-r invarianti T de baza, iar fiecare invariant T al retelei PN poate fi scris drept o combinatie liniara a acestora.*

Teorema 2.4. Combinatia liniara a invariantilor
Fie a si b doi invarianti de acelasi tip (P sau T) ai unei retele Petri

$$PN = (P,T,F,W,x,M_0).$$

Daca pentru α si β din R_+, vectorul $\alpha a+\beta b$ are toate elementele intregi, nenegative, atunci:

- *$\alpha a + \beta b$ este un invariant (P respectiv T) a lui PN;*
- $< \alpha a + \beta b > = < a > \cup < b >$

Teorema 2.5. Invariantul P

Un vector $y>0$, de dimensiune m, cu toate elementele intregi, este un invariant P, daca si numai daca, pentru un marcaj initial M_0 arbitrar si pentru orice marcaj M din $R(M_0)$, are loc egalitatea:

$$M^T y = M_0^T y$$

Teorema 2.6. Invariantul T

Un vector $xy>0$, de dimensiune n, cu toate elementele intregi, este un invariant T, daca si numai daca, exista un marcaj initial M_0 si o secventa de executari σ, ce porneste din M_0 si ajunge inapoi la M_0, cu vectorul numarului de executari $\overline{\sigma} = x$

Demonstratiile acestor teoreme se gasesc in [Pastr97].

Proprietatile structurale ale retelei Petri acoperite de invarianti P (RP poseda invarianti P cu toate elementele nenule) rezulta din urmatoarea teorema.

Teorema 2.7. Conservativitate si structural marginire

Daca $PN = (P,T,F,W,x,M_0)$ este o retea Petri acoperita de invarianti P, atunci:

- *reteaua PN este conservativa;*
- *reteaua PN este structural marginita.*

Proprietatile structurale ale retelei Petri acoperite de invarianti T (PN poseda invarianti T cu toate elementele nenule) rezulta din urmatoarea teorema.

Teorema 2.8. Consistenta si repetivitate

Daca $PN = (P,T,F,W,x,M_0)$ este o retea Petri acoperita de invarianti T, atunci:

- *reteaua PN este consistenta;*
- *reteaua PN este repetitiva.*

2.3.5. Retele Petri temporizate [CL01][Aalst96a][Kemp04]

Modelul original al retelelor Petri nu include timpul ca parametru de analiza. Timpul nu a fost considerat explicit din urmatoarele motive:

- masurarea timpului in cadrul sistemelor distribuite implica sincronizarea prin intermediul unui semnal de tact general;
- in sistemele distribuite de dimensiuni mari comunicatia interprocese este modelata numai prin intermediul unor semnale de viteza limitata, ceea ce implica:
    - existenta unui semnal de tact necesar sincronizarii transferului de informatii de la un proces la altul;
    - timpul considerat este relativ, procesul de comunicatie nefiind un timp absolut.
- dependentele cauzale in retelele Petri sunt descrise ca dependente locale, deci cu egaliate in timp;
- descrierile independente a formelor de paralelism se face independent de timp;
- fara utilizarea timpului capacitatile de modelare ale retelelor Petri sunt mult mai mari.

Prin introducerea timpului, se urmareste in fapt apropierea de situatile normale de lucru, retelele Petri devenind un instrument foarte util in analiza si modelarea sistemelor.

Problemele care apar o data cu introducerea timpului pot fi sintetizate astfel:

- cat timp va fi alocat pentru procesarea unei secvente?
- care este puterea de calcul necesara pentru rezolvarea problemelor?
- care este procentajul de executie a taskurilor pana la o noua comutare?
- care este timpul de asteptare al clientilor?
- ce implicatii apar asupra fluxurilor tehnologice?

Desigur ca retelele Petri nu pot oferi solutii (raspunsuri) la toate aceste probleme comune din practica.

Introducerea timpului in retelele Petri s-a facut prin asocierea timpului cu elementele retelei considerate. Astfel, exista mai multe posibilitati de utilizare a timpului:

- Timpul este asociat jetoanelor:
  - un jeton dintr-o pozitie poate fi valid sau invalid;
  - sosirea jetoanelor nu este valida;
  - dupa o perioada de timp un jeton invalid devine valid;
  - o tranzitie poate elimina numai jetoane valide;
  - o tranzitie poate fi executata numai dupa aparitia unui numar suficient de jetoane valide;
  - tranzitia este instantanee.
- Timpul este asociat cu tranzitiile:
  - executia tranzitilor are ca efect stergerea jetoanelor din toate pozitile din setul *„pre”*;
  - jetoanele „raman” pentru o perioada de timp in tranzitie;
  - dupa un timp jetoanele sunt transferate in pozitile *„post”*.
- Timpul este asociat pozitiilor:
  - inaite de validarea unei tranzitii „se asteapta” o perioada de timp in care jetoanele sunt pastrate in pozitii;
  - dupa expirarea timpului, tranzitia se executa instantaneu.
- Timpul este asociat arcurilor:
  - fiecarui arc ii este asociata o durata de timp;
  - pentru o tranzitie valida arcul de intrare determina momentul de timp in care tranzitia este executata;
  - timpul asociat arcurilor de iesire dintr-o tranzitie determina momentul aparitiei jetoanelor in pozitiile *„post”* dupa executia tranzitiei.

Analiza si modelarea retelelor Petri temporizate depinde de modelul considerat pentru timp. In literatura de specialitate sunt prezentate trei moduri de utilizare a timpului:

- *timpi constanti* - o tranzitie este executata intotdeauna exact in aceeasi perioada de timp, indiferent de modul de utilizare a timpului;
- *intervale de timp* – o tranzitie este executata intr-un interval de timp indiferent de modul de utilizare a timpului;
- *timpi stohastici* – o tranzitie este executata dupa un timp aleator indiferent de modul de utilizare a timpului.

In cazul utilizarii timpilor constanti se considera urmatoarele ipoteze:

- duratele de timp asociate tranzitiilor sunt numere rationale pozitive;

- duratele sunt independente de marcaj;
- tranzitiile pot fi validate o singura data (numai in cazul in care in pozitia *„pre"* se afla un numar suficient de jetoane care sa valideze tranzitia)
- in fiecare moment de timp sunt validate concurent un numar maxim de tranzitii care pot fi executate.

Definita 2.10 Structura timpului

Structura timpului asociata unui set de tranzitii temporizate $T_D \subseteq T$ *a unei retele Petri marcate (P, T, F, W,M₀) este un set:*

$$V = \{v_j : t_j \in T_D\}$$

al secventei de timp:

$$v_j = \{v_{j,1}, v_{j,2}, ...\},\ t_j \in T_D,\ v_{j,k} \in R^+,\ \ k = 1,2,.....$$

2.3.5.1. Retele Petri temporizate cu timpi constanti [CL01] [Kemp04] [Aalst96a]

Definitia 2.11. Retea Petri temporizata cu timpi constanti

O retea Petri temporizata cu timpi constanti este o pereche (N,V), unde N este o retea Petri marcata (P, T, F,W,M_0), iar $V = \{v_j : t_j \in T_D\}$ *este tactul structurii, timpul fiind reprezentat de numere rationale pozitive de valoare fixa.*

Executia unei tranzitii este data de starea lui *U(t)*:

- $U(t) = 0$ – nu se executa tranzitie;
- $U(t) > 0$ - daca tranzitia este activa.

Dinamica retelelor Petri temporizate cu timpi constanti este descrisa astfel:

- Fie *[M,U]* starea retelei la momentul τ (M – marcajul retelei, U – momentul de timp);
- Daca un pas V este in curs de executie, atunci *[M',U']* la momentul $\tau + 1$ este dat de:

$$M'(p) = M(p) - \sum_{t \in V} w(p,t) + \sum_{t \in V, D(t)=1} w(t,p) + \sum_{\substack{t \in T, D(t)>1 \\ U(t)=D(t)-1}} w(t,p) \qquad (2.51)$$

pentru toate pozitile $p \in P$

$$U'(t) = \begin{cases} 1 & daca\ t \in V \wedge D(t) > 1 \\ U(t)+1 & daca\ t \notin V \wedge U(t) < D(t)-1 \\ 0 & altfel \end{cases} \qquad (2.52)$$

pentru orice $t \in T$. D(t) reprezinta normalizarea scarii timpului (cel mai mic multiplu comun).

Definitia 2.12. Retea Petri temporizata P cu timpi constanti

O retea Petri temporizata P cu timpi constanti este un sixtuplu (P,T,F,W,D,M_0), unde P, T, F, W, M_0 au aceeleasi semnificatii ca in cazul retelei Petri netemporizate, iar D reprezinta multimea timpilor alocati executiei unei pozitii, fiind reprezentat de numere rationale pozitive de valoare fixa.

2.3.5.2. Retele Petri temporizate cu intervale de timpi [CL01] [Aalst96a] [Kemp04]

In acest caz timpul este introdus prin definire unui interval $[\alpha,\beta]$ unde α,β sunt numere rationale pozitive $0 \le \alpha \le \beta < \infty$:

- α reprezinta timpul de inceput *eft*
- β reprezinta timpul de sfarsit *lft*.

Definitia 2.13. Retea Petri temporizata cu intervale de timpi
O retea Petri temporizata cu intervale de timp este o tripleta (N,eft,lft), unde N este o retea Petri marcata (P, T, F, W,M_0) si eft,lft marcheaza intervalul de timp pentru T astfel incat pentru orice $t \in T : eft(t) \le lft(t)$.

Definitia 2.14. Retea Petri temporizata P cu intervale de timp
O retea Petri temporizata P cu intervale de timp este un sixtuplu (P,T,F,W,I,M_0), unde P, T, F, W, M_0 au aceeleasi semnificatii ca in cazul retelei Petri netemporizate, iar I reprezinta multimea intervalelor de timpi alocate executiei unei pozitii, fiind reprezentat de numere rationale pozitive de valoare fixa.

O tranzitie t care este validata la momnetul τ nu va putea fi executata inainte de $\tau + eft(t)$ si va fi incheiata inainte de $\tau + lft(t)$.
Starea retelei Petri temporizate cu intervale de timp este descrisa de *[M,U]*, unde:

- M este marcajul retele;
- U este tactul setat astfel incat: $U : T \to N \cup \{*\}$, unde
    - U(t)=* $\Leftrightarrow$ t nu este validata de M;
    - U(t)≠* $\Rightarrow$ $0 \le U(t) \le lft(t)$

Starea initiala a retelei este o pereche [M_0,U_0] unde:

$$U_0(t) = \begin{cases} 0 & daca\ t\ este\ validata\ de\ M_0 \\ * & altfel \end{cases}$$

$U(t) = *$, tactul tranzitiei t este dezactivat si t este invalidat

$U(t) \neq *$, t este validata si timpul a expirat ca si cum tranzitia a fost executata in acel interval *U(t)≤lft.*

Executia unui pas este descrisa astfel: daca tranzitia $t' \in T$ este validata in [M,U] starea retelei este schimbata in [M',U'] dupa cum urmeaza

$$M'(p) = M(p) + W'(t',p) - W(p,t')\ pentru\ \forall t \in T \tag{2.53}$$

$$U'(t) = \begin{cases} 0 & daca\ W(\bullet t) \le M' \wedge [t = t' \vee \neg(W(\bullet t) \le M] \vee (W(\bullet t) \le M \wedge \bullet t \cap \bullet t' \neq 0)] \\ U(t) & dacaW(\bullet t) \le M' \wedge W(\bullet t) \le M \wedge \bullet t \cap \bullet t' = 0 \wedge t \neq t' \\ * & altfel \end{cases} \tag{2.54}$$

2.3.6. Retele Petri etichetate [CL01][HKG97]

Pana in acest moment toate referirile legate de retelele Petri nu au inclus existenta unor conditii suplimentare (etichete) in validarea tranzitilor. In cele prezentate anterior validarea, executia, unei tranzitii era strict legata de numarul de jetoane dintr-o pozitie si ponderea arcului catre tranzitia corespunzatoare.

In realitate insa, in mod similar cu cele prezentate pentru automate, validarea tranzitilor trebuie sa tina seama si de situatii reale. Astfel, pentru validarea tranzitilor trebuie sa se tina cont de evenimente concrete, evenimente care pot fi grupate intr-un alfabet, rezultand astfel un limbaj corespunzator retelei Petri analizate. Modul in care se executa o tranzitie este de fapt similar cu cel descris anterior, cu observatia ca pentru validarea tranzitiei respective, pe langa conditile amintite mai intervine si conditia impusa de validarea evenimentului asociat, eveniment descris prin intermediul etichetei asociate.

Definitia 2.15 Retea Petri etichetata

O retea Petri etichetata este un octocuplu $PN = (P,T,F,W,\Sigma,l,M_0,X_m)$ *(topologia reţelei), unde:*

- *P reprezintă mulţimea finită de poziţii, cu* $P = \{p_1, p_2, p_3,..., p_n\}$;
- *T reprezintă mulţimea finită de tranziţii, cu* $T = \{ t_1, t_2, t_3,..., t_m\}$;
- $F \subseteq (P \times T) \cup (T \times P)$ *reprezintă mulţimea arcelor de la poziţii la tranziţii şi de la tranziţii la poziţii, fiecare arc fiind reprezentat prin* (p_i, t_j)*, respectiv* (t_j, p_i)*, unde* $i,j \in N$;
- $W : A \to \{1,2,3...\}$ *reprezintă funcţia de ponderare a arcelor;*
- Σ *reprezinta setul evenimentelor pentru etichetele tranzitiilor;*
- $l : T \to \Sigma$ *reprezinta functia de tranzitie etichetata;*
- $M_0 \in N^n$ *reprezinta marcajul initial a retelei (numarul initial al jetoanelor din fiecare stare);*
- $X_m \subseteq N^n$ *multimea starilor marcate ale sistemului.*

In [HKG97] se prezinta modul de definire a limbajelor generate, respectiv marcate ale unei retele Petri etichetata. In mod normal, unei retele Petri etichetate ii sunt asociate doua tipuri de limbaje: *limbajul generat P* - comportament inchis, respectiv *limbajul marcat L - comportament marcat*. Comportamentul inchis reprezinta toate evolutiile posibile a retelei Petri etichetate. In cazul in care comportamentul marcat reprezinta in fapt un comportament terminal, inseamna ca toate evolutiile retelei ating o stare terminala. In acest caz este posibil a fi definit comportamentul moale (slabit) al retelei prin intermediul *limbajului slab G* .

Definitia 2.16.Limbajele unei retele Petri etichetate

Fie reteaua Petri etichetata $PN = (P,T,F,W,\Sigma,l,x_0,X_m)$.

Limbajul generat al retelei considerate este:

$$L(PN) := \{l(s) \in \Sigma^* : s \in T^* \text{ si } m_0[s > \text{este definit}\} \tag{2.55}$$

Limbajul marcat al retelei este:

$$L_m(PN) := \{l(s) \in \Sigma^* : s \in T^* \text{ si } m_0[s > m \text{ unde } m \in X_m\} \tag{2.56}$$

2.3.7. Utilizarea submodelelor in modelarea SED cu ajutorul retelelor Petri [PMM02]

Referirile la utilizarea submodelelor in modelarea SED cu ajutorul retelelor Petri se fac in principal prin intermediul utilizarii *retelelor Petri Colorate*, ca metoda de modelare aleasa. In continuare abordarea utilizarii submodelelor se face pornind de la definitia clasica a retelelor Petri.

Metoda de utilizare a submodelelor propusa in continuare porneste de la observatia ca in cazul modelarii sistemelor complexe cu un numar mare de pozitii si tranzitii matricile de incidenta ale modelului devin de dimensiuni foarte mari facand dificila analiza retelei considerate prin utilizarea tehnicilor descrise anterior. Prin utilizarea submodelelor se urmareste reducerea dimensiunilor matricilor de incidenta si implicit al arborelui de acoperire, rezultand o retea Petri echivalenta usor de analizat.

Definitia 2.17 Subreteaua (submodelul) unei retele Petri*

Fie reteaua Petri $PN = (P, T, F, W, M_0)$ *unde:*

- P *reprezintă mulţimea finită de poziţii, cu* $P = \{p_1, p_2, p_3, \dots, p_n\}$*;*
- T *reprezintă mulţimea finită de tranziţii, cu* $T = \{ t_1, t_2, t_3, \dots, t_m\}$*;*
- $F \subseteq (P \times T) \cup (T \times P)$ *reprezintă mulţimea arcelor de la poziţii la tranziţii şi de la tranziţii la poziţii, fiecare arc fiind reprezentat prin* (p_i, t_j)*, respectiv* (t_j, p_i)*, unde* $i,j \in N$*;*
- $W : A \to \{1,2,3...\}$ *reprezintă funcţia de ponderare a arcelor.*
- $M_0 : P \to \{0,1,2,,...\}$ *reprezinta functia de marcaj initial*

Se definestete subreteaua Petri: $PN_{SUB} = \{P_{SUB}, T_{SUB}, F_{SUB}, W_{SUB}, M_{SUB0}\}$ *ca fiind un subgraf al grafului PN unde:*

- P_{SUB} *reprezinta multimea finita a pozitiilor corespunzatoare subretelei* $P_{SUB} \subseteq P$*;*
- T_{SUB} *reprezinta multimea finita a tranzitilor corespunzatoare subretelei* $T_{SUB} \subseteq T$*;*
- F_{SUB} *reprezinta multimea finita a arcelor corespunzatoare subretelei* $F_{SUB} \subseteq F$*;*
- W_{SUB} *reprezinta multimea finita a ponderilor arcelor corespunzatoare subretelei* $W_{SUB} \subseteq W$*;*
- M_{SUB0} *reprezinta multimea finita a marcajelor initiale corespunzatoare subretelei* $M_{SUB0} \subseteq M_0$.

In urma unei expandari a unor tranzitii sau pozitii, acestea sunt inlocuite cu subretele ce poseda un anumit tip comportamental. Aceasta substituire avand la baza submodele (subretele) se face utilizand asa numitele *blocuri T* respectiv *blocuri P*.

Un *bloc T* (tip tranzitie) este definit ca fiind o retea Petri, notata N_T, a carei topologie contine:

- *cel putin o tranzitie initiala* (de tip sursa), notata t_{in};
- *cel putin o tranzitie finala* (de tip receptor), notata t_{fi}.

Un *bloc P* (de tip pozitie) este definit ca un submodel ce poate fi inlocuit prin intermediul unei pozitii, pastrand proprietatile de marginire (siguranta) si viabilitate.

Teorema 2.9.

Fie o reteaua Petri PN. Fie PN_{SUB} *reteaua Petri rezultata prin substituirea unei tranzitii (respectiv pozitii) a lui PN cu un modul standard reprezentat sub forma de bloc T respectiv bloc P.*

a) daca PN este marginita (sigura) atunci, PN_{SUB} este marginita (sigura);
b) daca PN este viabila, atunci PN_{SUB} este viabila;
c) daca PN este reversibila, atunci PN_{SUB} este reversibila.

Modulele standard utilizate sunt prezentate in figura 2.4.

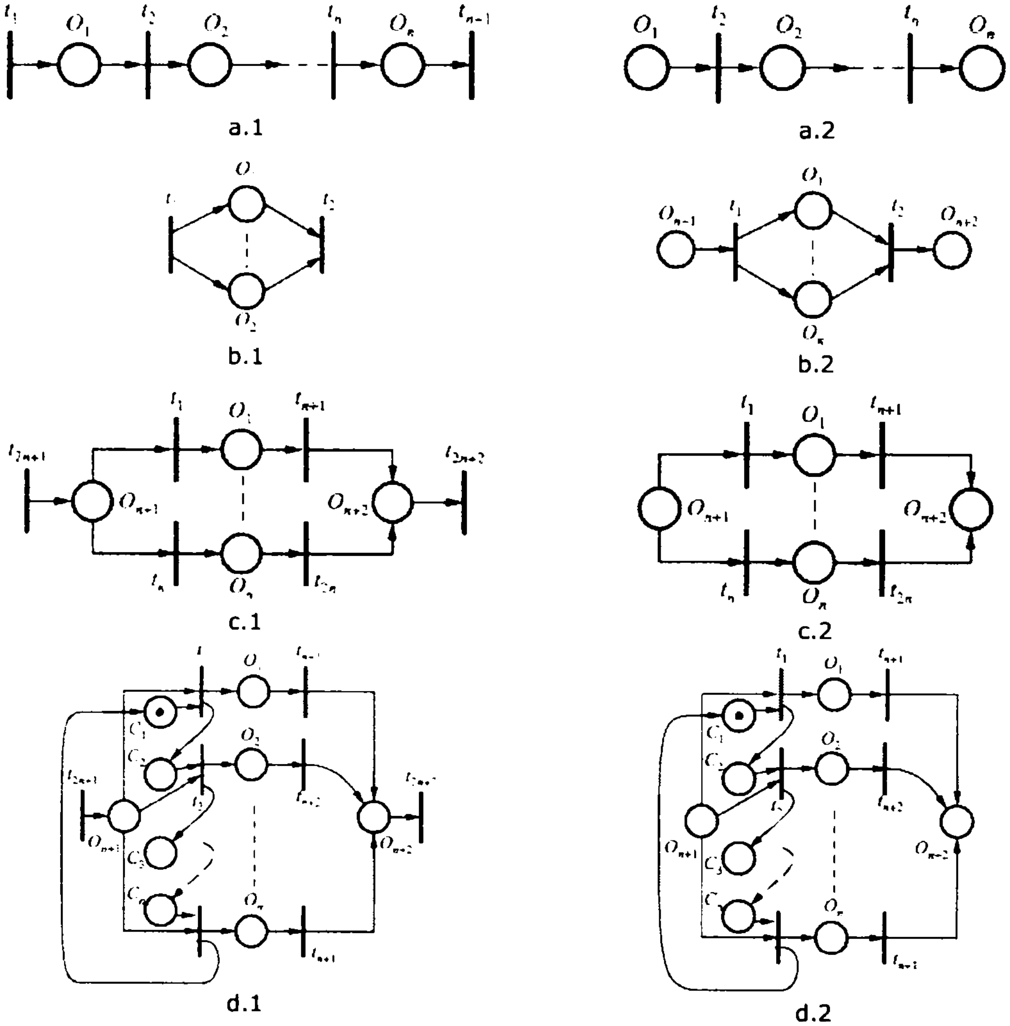


Figura 2.4: Modulele standard utilizate in cazul subretelelor. a)Operatii secventiale (a.1-*bloc T*, a.2-*bloc P*) b) Operatii paralele (b.1- *bloc T*, b.2 -*bloc P*) c) Operatii la alegere, conflictuale (c.1-*bloc T*, c.2-*bloc P*) d) Operatii la alegere, neconflictuale (d.1-*bloc T*, d.2-*bloc P*)

Structura de principiu al unui model de tip retea Petri construit pe baza unor submodele este prezentata in figura 2.5.

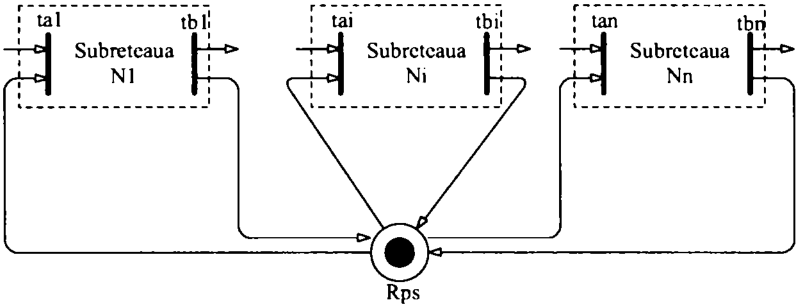


Figura 2.5: Model de tip retea Petri implementat cu submodele

Rps reprezinta pozitia de sincronizare si partajare secventiala. Fiecare subretea (submodel) are o *tranzitie de tip receptor* (intrare - t_{ai}) si o *tranzitie de tip generator* (iesire t_{bi}).

Utilizarea submodelelor in modelarea cu ajutorul retelelor Petri trebuie sa respecte urmatoarele conditii:

a) Orice drum elementar dintre t_{ai} si o pozitie asociata unei resurse generale trebuie sa contina si t_{bi};
b) Orice circuit elementar ce contine t_{ai} si R_{ps} trebuie sa contina si t_{bi};
c) Fiecare tranzitie de pe orice drum elementar dintre t_{ai} si t_{bi} trebuie sa apartina si unui drum elementar de operatii unind t_{ai} cu t_{bi};
d) Oricare ar fi marcajul *M(N)* care valideaza t_{ai}, daca t_{ai} se executa, atunci exista o secventa de executari de tranzitii din N_i care conduce la executarea lui t_{bi}.

Structura echivalenta a retelei Petri construita cu ajutorul submodelelor din figura 2.5 este prezentata in figura 2.6.

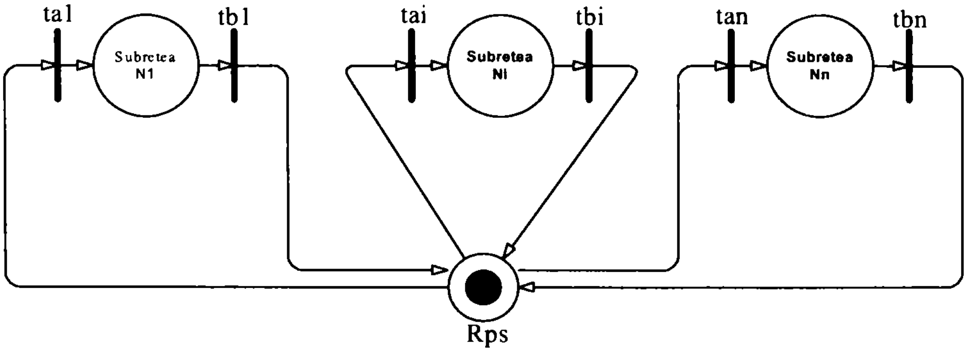


Figura 2.6: Structrua echivalenta a retelei Petri din fig 2.5

In cazul sistemelor complexe exista posibilitatea ca sincronizarea submodelelor sa se efectueze prin intermediul mai multor stari de sincronizare. In aceasta situatie tranzitiile t_{ai} , respectiv t_{bi} , pot avea mai mult decat o conexiune. In figura 2.7 se prezinta un caz general in care exista mai multe tipuri de submodele sincronizate prin mai multe stari de sincronizare.

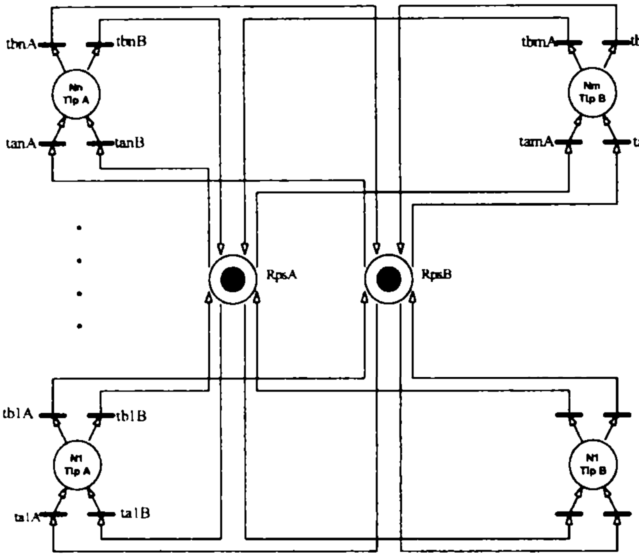


Figura 2.7: Structura generala a unei retele Petri construita cu submodele

Utilizarea submodelelor in modelarea SED complexe are ca efect obtinerea unor modele simplificate cu implicatii directe asupra matricilor de incidenta si a arborelui de acoperire. Trebuie insa mentionat faptul ca submodelul ales trebuie sa respecte in mod strict conditiile prezentate mai sus. In cazul in care submodelul nu este ales in mod corespunzator (posibile aparitii de situatii blocante) intreg modelul ales pentru SED va fi compromis.

In cazul retelelor Petri temporizate (P sau T) utilizarea submodelelor are ca efect obtinerea unui model de retea Petri cu intervale de timp.

Definitia 2.18 Executia unei retele Petri*

Se defineste executia unei retele Petri ca fiind totalul tranzitilor efectuate pornind dintr-o starea data (posibil initiala) pana la ajungerea inapoi in aceasi stare.

Dupa cum s-a precizat in paragraful 2.2.1 dinamica unei retele Petri poate fi descrisa cu ajutorul ecuatiei de stare (2.46). Pornind de la marcajul initial „traseul" urmat de reteaua Petri (pozitile prin care s-a trecut) sunt cuprinse in vectorul de control

$$u = \sum_{k=1}^{d} u_k \, . \tag{2.57}$$

In cazul retelelor Petri cu o topologie complexa, este posibil ca atingerea unui marcaj destinatie sa poata fi efectuata pe mai multe trasee (prin atingerea unor pozitii diferite). In aceasta situatie vectorul de control nu este unic, ci pentru aceeasi destinatie functie de numarul de trasee posibile exista mai multi vecori de control u_{di} unde $di=1,...,j$ reprezinta numarul de trasee posibile.

Formalizand matematic, rezulta:

$$u_{tot} = \bigcup_{di=1}^{j} u_{di} = \bigcup_{di=1}^{j} \left(\sum_{k=1}^{d} u_k \right)_{di} \tag{2.58}$$

unde u_{tot} reprezinta multimea tuturor posibilitatilor de atingere a unui marcaj.

Teorema 2.9 Modele cu intervale de timp*
Fie reteaua Petri temporizata P cu timpi constanti $PN=(P,T,F,W,D,M_0)$. O retea Petri PN' construita cu ajutorul retelei Petri PN , considerata ca submodel al retelei PN', este o retea Petri cu intervale de timp.

Demonstratie.
Fie $d(p_i)$ timpul asociat pozitiei p_i, unde $i=1,...,n$. Evolutia retelei Petri este descrisa cu ajutorul vectorului de control $u = \sum_{k=1}^{d} u_k$. Pentru un traseu, caracterizat prin vectorul u_i timpul total de executie este $d_l = \sum_{i=1}^{l} d(p_i)$ unde l reprezinta numarul de pozitii atinse. In cazul unui vector de control $u_j \neq u_i$ rezulta $d_p \neq d_l$, unde p reprezinta numarul de pozitii atinse in cazul vectorului de control u_j. Rezulta deci ca functie de timpii alocati pozitiilor atinse timpul total de executie al retelei este diferit functie de traseul parcurs. Astfel, rezulta un timp minim, respectiv un timp maxim de executie, reteau Petri astfel considerata este o retea Petri cu intervale de timpi constanti.

Lema 2.1*
Timpul minim al executitiei unei retele Petri este dat de valoarea cea mai mica a timpului total de executie, nu de traseul cel mai scurt.

Demonstratie:
Fie u_i si u_j ($u_j \neq u_i$) doi vectori de control corespunzatori executiei unei retele Petri, si fie $d_p < d_l$ timpii de executie corespunzatori vectorilor u_i respectiv u_j. Se considera ca traseul corespunzator vectorului u_i contine k pozitii iar traseul corespunzator vectorului u_j contine m pozitii unde $k > m$. Deoarece timpul de executie corespunzator vectorului u_i este mai mic, dar traseul mai lung rezulta ca timpul minim nu este influentat de lungimea traseului (numarul de pozitii atinse) ci de durata de executie a unei pozitilor atinse.

Lema 2.2*
Timpul maxim de executie al unei retele Petri este dat de valoarea cea mai mare a timpului total de executie si nu de traseul cel mai lung.

Demonstratie:
Fie u_i si u_j ($u_j \neq u_i$) doi vectori de control corespunzatori executiei unei retele Petri, si fie $d_p < d_l$ timpii de executie corespunzatori vectorilor u_i respectiv u_j. Se considera ca traseul corespunzator vectorului u_i contine k pozitii iar traseul corespunzator vectorului u_j contine m pozitii, unde $k > m$. Deoarece timpul de executie corespunzator vectorului u_j este mai mare dar traseul mai scurt, rezulta ca timpul maxim nu este influentat de lungimea traseului (numarul de pozitii atinse) ci de durata de executie a unei pozitilor atinse.

In [PMM02] se prezinta modul de comportare periodica a retelelor Petri acoperite de invarianti de tip P temporizate P. Pornind de la marcajul initial M0 si daca σ reprezinta o secventa de executari de tranzitii care are ca finalitate tot marcajul M0, reteaua prezinta un comportament ciclic caracterizat prin durata unui ciclu τ, care reprezinta intervalul de timp necesar executarii secventei σ. Cunoscand timpul asociat fiecarei pozitii $d(p_j) \geq 0,\ j = 1,\dots,m$ se construieste matricea diagonala $D \in R^{m \times n}$ care are pe diagonala valorile $d(p_j)$:

$$(D)_{ij} = \begin{cases} d(p_j), & i = j \\ 0 & i \neq j \end{cases} \quad i,j = 1,\dots,m \tag{2.59}$$

Fie $y_k,\ k=1,\dots,p$ invariantii P fundamentali ai retelei considerate. Durata unui ciclu este caracterizata prin urmatoarea teorema.

Teorema 2.10 Durata unui ciclu

Durata unui ciclu satisface inegalitatea:

$$\tau \geq \tau_{min} = \max_{k=1,p} \left\{ \frac{y_k^T D(A^+)^T x}{y_k^T M_0} \right\}$$

unde, $x = \overline{\sigma}$ *noteaza acel invariant T care corespunde secventei de executari* σ.

Determinarea timpului minim si a timpului maxim de executie corespunzatoare unei retele Petri cu intervale fixe de timp se poate face in doua moduri:

- pe baza arborelui de acoperire - considerand la executia fiecarei tranzitii timpul asociat pozitiei sursa, prin insumare rezulta valoarea de timp cautata;
- prin utilizarea ecuatiei de stare - pornind de la marcajul initial se genereaza pas cu pas vectori de control corespunzatori. La executia fiecarei tranzitii se memoreaza pozitia din care se pleaca si prin insumarea timpilor corespunzatori se obtin rezultatele cautate.

In ambele cazuri, daca mai multe pozitii sunt implicate simultan in executia unei tranzitii, pentru determinarea timpilor minim si maxim, se considera timpul asociat acelei pozitii care are valoarea cea mai mare.

Aceste intervale de timp se obtin fara dificultate, prin urmarire directa, utilizand aplicatia PetriTim care este prezenta in capitolul 5.

2.4. Conexiuni intre modelele de tip automat si modelele de tip retea Petri [Pastr97][CL01][CR04a][Kemp04][CKLY95] [CKLY98] [BHR06] [LM04] [Micz01]

Ambele metode de modelare a SED, automate respectiv retele Petri, au la baza utilizarea starilor si tranzitilor pentru descrierea unui sistem, rezulta ca intre ele exista asemanari si deosebiri.

Modelarea SED cu automate respectiv retele Petri, precum si alegerea tipului de model este lasat la latitudinea modelatorului.

Astfel, daca se doreste modelarea unui SED numai pe baza evenimentelor externe ale acestuia, fara a interesa in mod explicit activitatile ascunse, atunci un model de tip automat este satisfacator. Daca in schimb, se doreste o rafinare a operatilor interne atunci un model de tip retea Petri este mai avantajos. Un exemplu clasic, care ilustreaza diferenta intre cele doua metode de modelare, este dat de modelarea functionarii unui sistem de asteptare.

Exemplul 2.3. Modelarea unui sistem de asteptare
Se presupune ca se dispune de un singur server iar capacitatea firului de asteptare este nelimitat. Clientii sosesc si cer acces la server. Daca serverul este ocupat, atunci clientul asteapta in coada. Cand un client a fost servit complet el paraseste sistemul iar clientul urmator din coada intra imediat in servire. Evenimentele pe baza carora este condus un astfel de sistem sunt:

s: *Clientul soseste* (intra in coada de asteptare)
p: *Clientul paraseste sistemul*

Aceste semnale sunt „vazute" din exteriorul sistemului, iar in cazul modelarii cu ajutorul automatelor, existenta acestor doua semnale este suficienta.
In cazul modelului de tip retea Petri, pentru rafinare, pe langa cele doua semnale mentionate se mai defineste:

e: *Clientul intra in executie*

Structura de principiu, precum si modelele de tip automat, respectiv retea Petri al unui astfel de sistem sunt prezentate in figura 2.8.

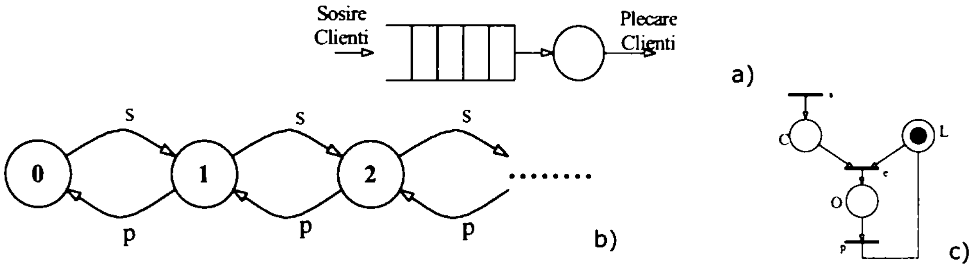


Figura 2.8: Modelarea functionarii unui sistem de asteptare: a) Structura de baza; b) modelul de tip automat; c) modelul de tip retea Petri

Dupa cum rezulta din figura 2.8.b, modelul de tip automat este un model cu stari infinite. In schimb modelul de tip retea Petri (figura 2.8.c) (asa cum s-a prezentat anterior) nu rezulta prin extinderea topologiei ci prin existenta marcajelor. Pozitiile din cadrul modelului de tip retea Petri au urmatoarele semnificatii: *C* - coada de asteptare si are capacitate infinita, *O* - server ocupat de capacitate finita si *L* - server liber de capacitate finita.

In cazul in care modelul opereaza cu un numar foarte mare de stari, un model de tip automat este dezavantajos deoarece modelarea se face prin extinderea topologiei modelului, pe cand in cazul retelelor Petri extinderea se face prin marcaje nu prin topologie.

In situatia in care modelarea se bazeaza pe folosirea submodelelor, utilizarea automatelor este mai dificila deoarece modelarea unui sistem trebuie privita in ansamblul ei, modularizarea fiind anevoiasa si dificila. In cazul retelelor Petri, utilizarea submodelelor este permisa deoarece modelul nu trebuie sa fie privit in ansamblul sau, avand posibilitatea ca prin marcaje si sincronizari suplimentare sa se asigure asamblarea intregului model.

Dupa cum s-a precizat anterior, automatele si retelele Petri opereaza cu stari si tranzitii. Astfel, sunt posibile transformari ale modelelor de tip automat in modele de tip retea Petri si invers. Subiectul sintezei modelelor dintr-un mod de reprezentare in altul este tratat pe larg in literatura [Pastr97][CR04a][CKLY95][CKLY98].

In continuare se va avea in vedere doar metodele de sinteza considerate.

Algoritmul 2.1. *Transformarea unui model de tip automat in model de tip retea Petri*

Fie automatul $G = (X, \Sigma, \delta, \Gamma, x_0, X_m)$ care se doreste a fi convertit in reteau Petri $PN = (P, T, F, W, x, M_0)$. Fiecarei perechi de stari din X desemnata prin (x, x'), cu $x' = \delta(x, e)$, $e \in \Gamma(x)$, i se asociaza o tranzitie $t \in T$ in reteaua Petri, atasand si arcele $(x,t),(t,x') \in F$, ambele de pondere 1. Aceasta revine la a defini multimile T si F prin:

$$T = \{(x, x') : x, x' \in X, x' = \delta(x, e), e \in \Gamma(x)\}$$

$$F = \{(x, t) : x \in X, t \in T\} \cup \{(t, x') : x' \in X, t \in T\}'$$

Aplicatia W ia numai valoarea 1, adica $W : F \to \{1\}$. Marcajul initial M_0 se asigneaza corespunzator starii initiale x_0. Prin aceasta metoda, unui eveniment $e \in \Sigma^*$ i se pot asocia mai multe tranzitii in multimea T.

Exemplul 2.4 Obtinerea unui model de tip retea Petri dintr-un model de tip automat pe baza algoritmului 2.1

Se considera automatul din figura 2.9.

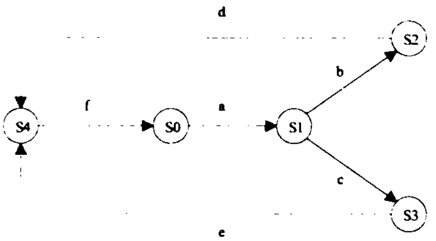


Figura 2.9: Structura automatului considerat pentru exemplul 2.4

Automatul considerat este formalizat matematic prin: $AS = (X, \Sigma, \delta, \Gamma, x_0)$ unde:

- Multimea starilor: $X = \{S0, S1, S2, S3, S4\}$
- Multimea evenimentelor: $\Sigma = \{a, b, c, d, e, f\}$
- Multimile evenimentelor posibile si functiile de tranzitie a starilor:

$\Gamma(S0) = \{a\}$	$\delta(S0, a) = S1$
$\Gamma(S1) = \{b, c\}$	$\delta(S1, b) = S2$
	$\delta(S1, c) = S3$
$\Gamma(S2) = \{d\}$	$\delta(S2, d) = S4$
$\Gamma(S3) = \{e\}$	$\delta(S3, e) = S4$
$\Gamma(S4) = \{f\}$	$\delta(S4, f) = S0$

- Starea initiala: $x_0 = S0$

Conform algoritmului 2.1, de conversie al unui model de tip automat in model de tip retea Petri, rezulta o retea Petri de forma $PN\_AS = (P,T,F,W,M_0)$ unde:

- Multimea pozitilor $P = (p1,p2,p3,p4,p5)$ unde $p1 = \{S0\}$, $p2 = \{S1\}$, $p3 = \{S2\}$,
  $p4 = \{S3\}$ si $p5 = \{S4\}$
- Multimea tranzitilor

Pentru $p1$:	$(p1,p2)$	$p2 = \delta(p1,t1)$	$t1 = \{a\}$
Pentru $p2$:	$(p2,p3)$	$p3 = \delta(p2,t2)$	$t2 = \{b\}$
	$(p2,p4)$	$p4 = \delta(p2,t3)$	$t3 = \{c\}$
Pentru $p3$:	$(p3,p5)$	$p5 = \delta(p3,t4)$	$t4 = \{d\}$
Pentru $p4$:	$(p4,p5)$	$p5 = \delta(p4,t5)$	$t5 = \{e\}$
Pentru $p5$:	$(p5,p1)$	$p1 = \delta(p5,t6)$	$t6 = \{f\}$

- Multimea arcelor:

$$F = \{(p1,t1),(p2,t2),(p2,t3),(p3,t4),(p4,t5),(p5,t6)\} \cup$$
$$\{(t1,p2),(t2,p3),(t3,p4),(t4,p5),(t5,p5),(t6,p1)\}$$

- Ponderile arcelor:

$$W(p1,t1) = 1, W(p2,t2) = 1, W(p2,t3) = 1, W(p3,t4) = 1, W(p4,t5) = 1,$$
$$W(p5,t6) = 1, W(t1,p2) = 1, W(t2,p3) = 1, W(t3,p4) = 1, W(t4,p5) = 1,$$
$$W(t5,p5) = 1, W(t6,p1) = 1$$

- Marcajul initial: $M_0 = [1,0,0,0,0]^T$

In figura 2.10 se prezinta reteaua Petri rezultata in urma conversiei.

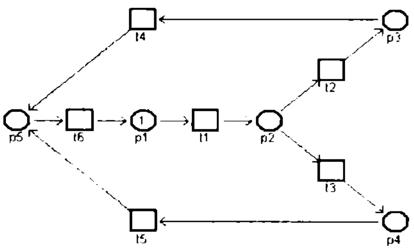


Figura 2.10: Structura retelei Petri rezultate in urma conversiei

Algoritmul 2.2*. Sinteza unei retele Petri dintr-un model de sisteme cu tranzitii finite**

Algoritmul, descris in [CKLY95][CKLY98], are la baza utilizarea sistemelor cu tranzitii finite. In continuare acest algoritm se prezinta sub forma propusa de autor pentru obtinerea modelului de tip retea Petri dintr-un model de tip automat.

Fie automatul $G = (X,\Sigma,\delta,\Gamma,x_0,X_m)$, daca in cazul algoritmului 2.1 nu s-au impus nici un fel de restrictii cu privire la modelul de tip automat, in cazul acestui algoritm sunt impuse urmatoarele restrictii:

- in model nu exista auto-bucle: $\forall x \in X \; x' = \delta(x,e)$, unde $e \in \Gamma(x) : x \neq x'$;

- orice eveniment are un efect: $\forall e \in \Sigma : \exists \delta(x,e)$, *pentru care* $e \in \Gamma(x)$;
- orice stare este accesibila din starea initiala: $\forall x \in X : x_0 \xrightarrow{*} x$;
- nu exista arce multiple intre stari perechi:
  $\forall x, x_1, x_2 \in X$ *si* $e_1, e_2 \in \Sigma$ *si* $x_1 = \delta(x,e_1), x_2 = \delta(x,e_2)$: *daca* $x_1 = x_2$ *atunci* $e_1 = e_2$

Obtinerea modelului de tip retea Petri pe baza acestui algoritm porneste de la definirea unor regiuni in cadrul modelului de tip automat.

Definitia 2.19. Conexiuni

Fie automatul $G = (X, \Sigma, \delta, \Gamma, x_0, X_m)$, *pentru care se defineste X′ ca fiind un subset a multimii X,* $X' \subseteq X$ *si un eveniment* $e \in \Sigma$.*Urmatoarele conditii sunt definite pentru X′ si e:*

1) *intern:* $in(e,S') \equiv \exists x, x' \in X$ *si* $e \in \Sigma\ x' = \delta(x,e) : x, x' \in S'$
2) *extern:* $out(e,S') \equiv \exists x, x' \in X$ *si* $e \in \Sigma\ x' = \delta(x,e) : x, x' \notin S'$
3) *intrare:* $enter(e,S') \equiv \exists x, x' \in X$ *si* $e \in \Sigma\ x' = \delta(x,e) : x \notin S' \wedge x' \in S'$
4) *iesire:* $exit(e,S') \equiv \exists x, x' \in X$ *si* $e \in \Sigma\ x' = \delta(x,e) : x \in S' \wedge x' \notin S'$

Definitia 2.20 Regiunea

Fie automatul $G = (X, \Sigma, \delta, \Gamma, x_0, X_m)$. *Un set de stari* $r \subseteq X$ *se spune ca este* ***o regiune*** *daca urmatoarele doua conditii sunt satisfacute pentru orice eveniment* $e \in \Sigma$:

1) $enter(e,r) \Rightarrow \neg in(e,r) \wedge \neg out(e,r) \wedge \neg exit(e,r)$
2) $exit(e,r) \Rightarrow \neg in(e,r) \wedge \neg out(e,r) \wedge \neg enter(e,r)$

O *regiune* reprezinta un subset al starilor pentru care toate tranzitiile sunt etichetate cu acelasi eveniment *e* avand aceleasi relatii de „intrari/iesiri" (entry/exit). Aceste relatii vor devenii predesor/succesor in reteaua Petri care va rezulta.

O regiune *r* este *pre-regiune* a unui eveniment *e* daca exista o tranzitie etichetata cu *e* care este iesire pentru *r*.

O regiune *r* este *post-regiune* a evenimentului *e* daca exista o tranzitie etichetata cu *e* care este intrare pentru *r*.

Fie automatul $G = (X, \Sigma, \delta, \Gamma, x_0, X_m)$ algoritmul 2.2, pentru conversia acestuia in reteaua Petri $PN = (P, T, F, W, x, M_0)$ devine:

Pasul 1. Pentru fiecare eveniment $e \in \Sigma$, se genereaza o tranzitie etichetata cu *e* in reteaua Petri;

Pasul 2. Pentru fiecare regiune $r_i \in R_G$, unde R_G este multimea tuturor regiunilor definite pentru *G*, se genereaza o pozitie r_i;

Pasul 3. Pozitia r_i contine un jeton in marcajul initial daca regiunea r_i contine starea initiala x_0 a automatului *G*;

Pasul 4. Relatia rezultanta este: $e \in \bullet r_i$, daca r_i este o *pre-regiune* a lui *e* si $e \in r_i \bullet$ daca r_i este o *post-regiune* a lui *e*:

$$F = \{(r,e) \mid r \in R_G \wedge e \in \Sigma \wedge r \in {}^{\circ}e\} \cup \{(e,r) \mid r \in R_G \wedge e \in \Sigma \wedge r \in e^{\circ}\}$$

Exemplul 2.5 Sinteza unui model de tip retea Petri utilizand metoda regiunilor

Se considera automatul din figura 2.11.

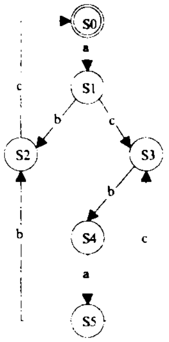


Figura 2.11: Structura automatului corespunzator exemplului 2.5

Datorita faptului ca tranzitiile dintre stari se realizeaza pe baza acelorasi evenimente in figura 2.12 se prezinta varianta echivalenta a automatului din figura 2.11.

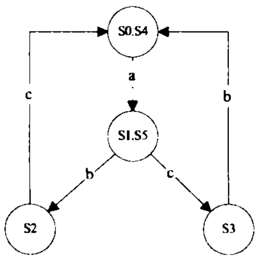


Figura 2.12: Structura echivalenta a automatului din fig. 2.11

Aplicand algoritmul 2.2, rezulta urmatoarele regiuni minimale:

$r1 = \{S0, S2, S4\}$, $r2 = \{S0, S3, S4\}$, $r3 = \{S1, S3, S5\}$, $r4 = \{S1, S2, S5\}$

Regiunile continand stari sunt:

$R_{S0} = R_{S4} = \{r1, r2\}$, $R_{S1} = R_{S5} = \{r3, r4\}$, $R_{S2} = \{r1, r4\}$, $R_{S3} = \{r2, r3\}$

Preregiunile si post regiunile sunt:

$\bullet a = \{R_{S0}, R_{S1}\}$ $\quad a\bullet = \{R_{S2}, R_{S3}\}$
$\bullet b = \{R_{S2}\}$ $\quad b\bullet = \{R_{S0}\}$
$\bullet c = \{R_{S3}\}$ $\quad a\bullet = \{R_{S1}\}$

Reteaua Petri astfel obtinuta este prezentata in figura 2.13.

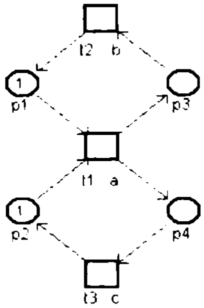


Figura 2.13: Structura retelei Petri obtinuta in urma aplicarii algoritmului 2.2

Unde: $p1 = R_{S0}$, $p2 = R_{S1}$, $p3 = R_{S1}$, $p4 = R_{S3}$.

2.5. Modelarea cu ajutorul automatelor, respectiv a retelelor Petri a sistemelor de transport cu zone de acumulare.

Dezvoltarea sistemelor flexibile de fabricatie (SFFF) a condus implicit la reevaluarea locului si importantei sistemelor de transport. Un SFFF nu poate fi considerat performant daca sistemul de transport aferent nu este la randul sau performant. In acest context, rolul sistemelor de transport a crescut, trecandu-se de la faza clasica, in care acesta avea rol preponderent de transfer a pieselor de la o masina la alta, la o faza in care sarcinile sistemului de transport au fost extinse (sortare automata, alegerea automata a destinatiei unei piese, calcularea si alegerea traseului optim etc.) vorbindu-se acum de sisteme de transport inteligente.

Sistemele de transport cu zone de acumulare (STZA) se incadreaza in clasa sistemelor de transport inteligente, fiind capabile ca pe baza unor algoritmi sau a unor specificatii de functionare (statice - fixate in momentul proiectarii, sau dinamice - care se pot modifica in timpul operarii functie de cerintele operatorului) sa execute transferuri de piese sau alte elemente de transport (carucioare) conform cerintelor. STZA isi gasesc aplicabilitate in cele mai diverse locuri: magazii automatizate, linii flexibile de fabricatie, sisteme de sortat etc.

2.5.1. Principiile de realizare si functionare ale sistemelor de transport cu zone de acumulare

Sistemele de transport industriale sunt de mai multe tipuri, funcţie de modul efectiv în care se realizeaza transportul. Astfel, există sisteme de transport orizontale (vezi figura 2.14) sau sisteme de transport suspendate (vezi figura 2.15). Fiecare tip de sistem are caracteristicile sale constructive şi funcţionale, ambele incadrandu-se in domeniul STZA.

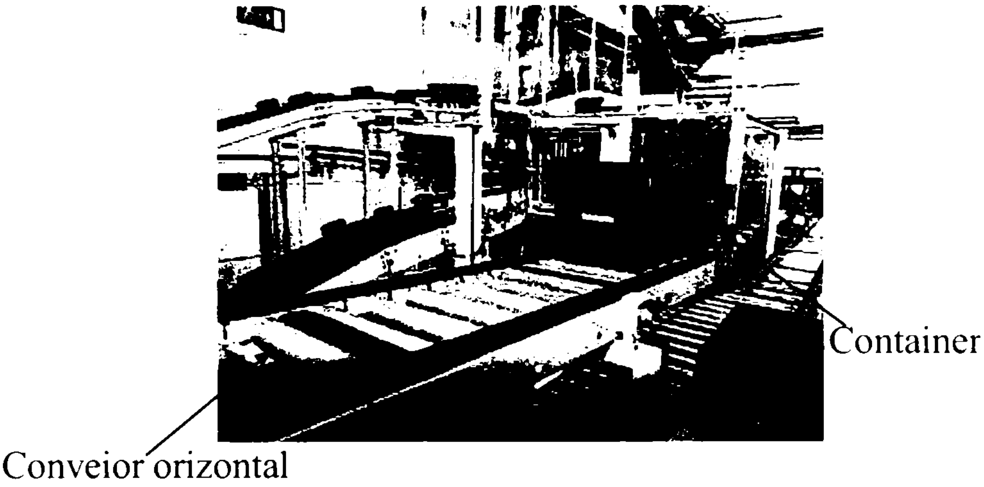


Figura 2.14. Sistem de transport orizontal

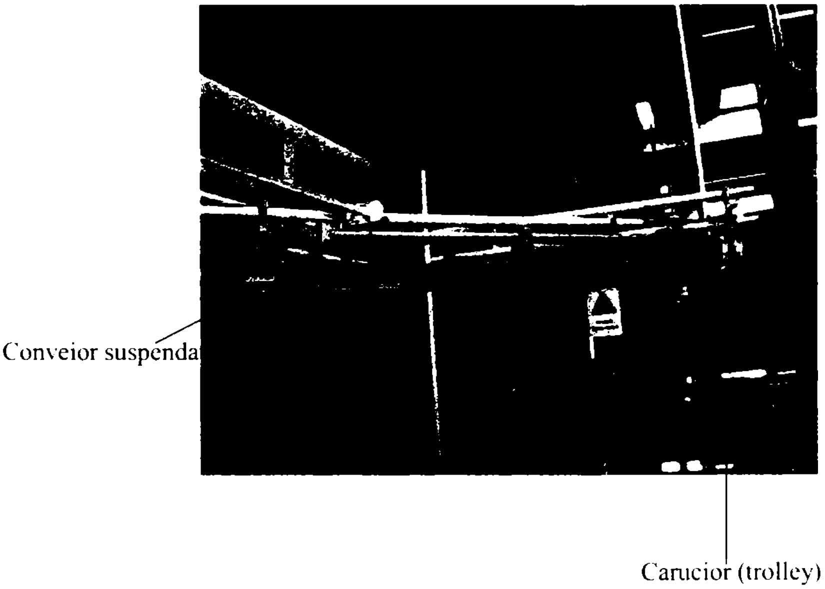


Figura 2.15. Sistem de transport suspendat

Observatie: *fotografiile 2.15, 2.16, 2.17, 2.18, 2.19 sunt realizate de catre autor in magazia complet automatizata a firmei Wacker din Germania.*

În cadrul lucrarii se vor lua in considerare numai sistemele de transport suspendate. Constructiv, acestea se bazează pe utilizarea conveioarelor cu perii (transelastic conveior - TEF) ca suport de transport şi au ca principala proprietate asigurarea de zone de acumulare (jam-uri) fără a fi necesară oprirea motoarelor de antrenare a conveioarelor[Ung01][Ung02]. In figura 2.16 se prezintă un astfel de conveior cu perii.

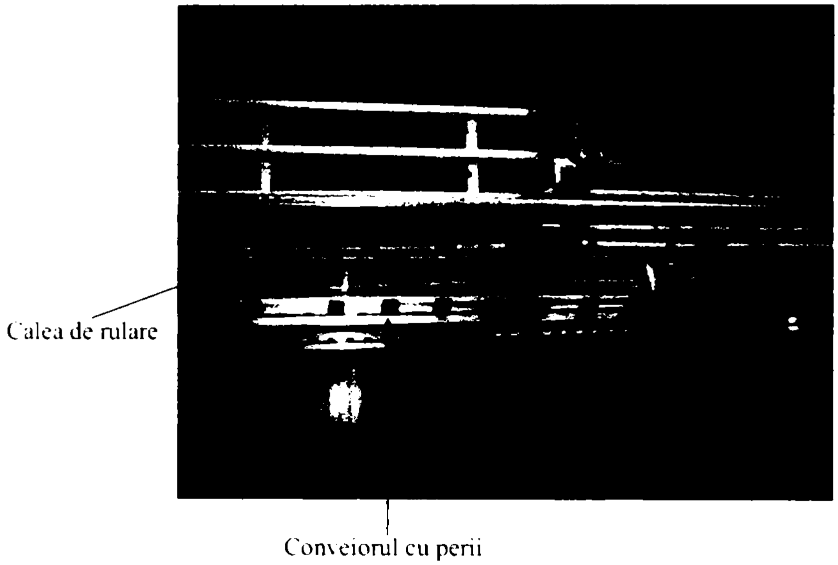


Figura 2.16. Conveior cu perii utilizat în cazul sistemelor de transport suspendate

Conveiorul este antrenat de un motor electric cuplat prin intermediul unui reductor mecanic, aflat la unul din capetele conveiorului. Acest motor asigură antrenarea conveiorului, asigurând o viteză constantă.

Elementul de transport utilizat în cazul acestor sisteme de transport este căruciorul (trolley), a cărui structură şi mod de utilizare (la nivelul conveiorului) este prezentat în figura 2.17.

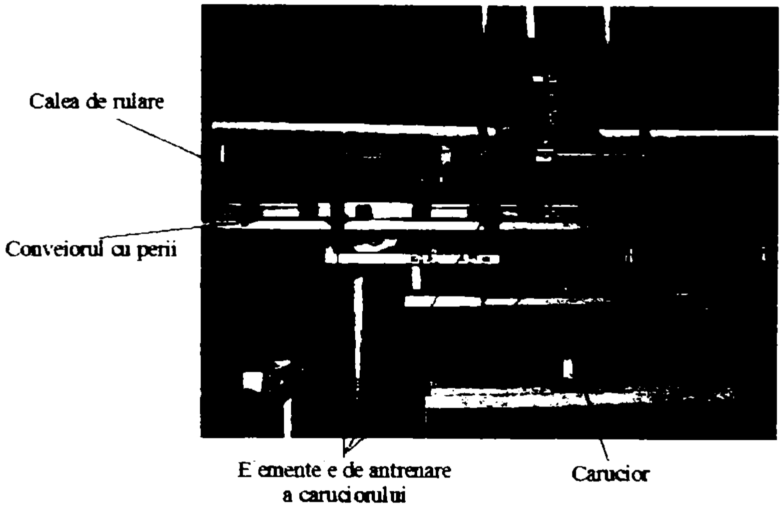


Figura 2.17. Carucior şi modul de transport

În figura 2.18 se prezintă o imagine de ansamblu a modului în care se efectuează transportul din cadrul unui depozit de piese de schimb complet automatizat.

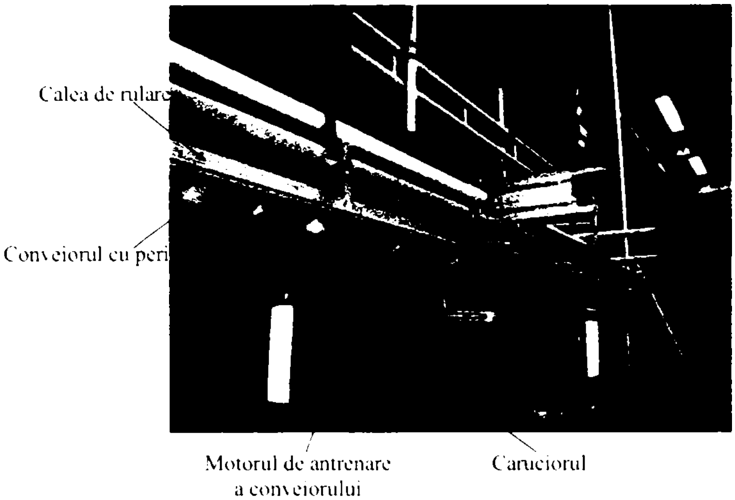


Figura 2.18. Imagine de ansamblu a sistemului de transport

Principalul avantaj al unui astfel de sistem de transport îl constituie existenta zonelor de acumulare (jam-uri) în care cărucioarele pot fi oprite, fără a necesita oprirea conveioarelor. În acest mod se asigura o flexibilitate sporita, avand posibilitatea de a opera simultan în toate zonele sistemului.

Elementul principal care asigură oprirea unui cărucior funcţie de algoritmul de conducere utilizat este stoperul. In figura 2.19 se prezintă un stoper utilizat în cazul acestor sisteme.

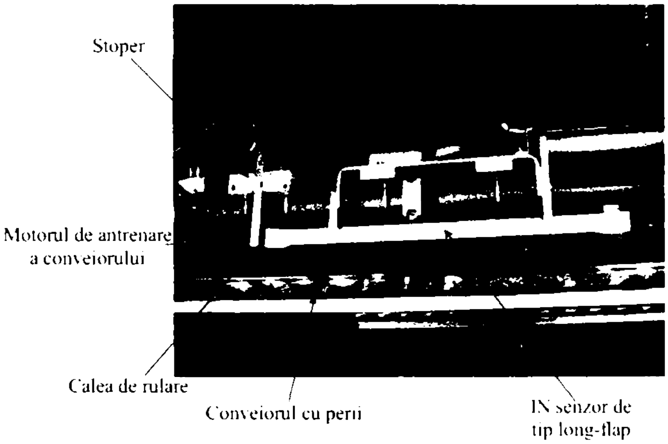


Figura 2.19. Stoper şi senzor de tip long-flap

În figura 2.19 se observa şi senzorul care semnalizează faptul că în faţa stoperului se află un cărucior.

Ideea de baza in conducerea unor astfel de sisteme este aceea de a asigura modularizarea elementelor care compun sistemul [UngAS06][UngPN06].

Au fost stabilite patru structuri de baza, cu ajutorul carora sa se poata modela orice modul, indiferent de complexitatea lui (*nodul de tip 1 o intrare o iesire, nodul de tip 2 doua intrari o iesire, nodul de tip 3 trei intrari o iesire si nodul de tip 4 o intrare doua iesiri*) [UP06a][UP06b].

2.5.1.1. Nodul de tip 1 – o intrare o ieşire [UP06a][UP06b] [Ung06b][Ung06c]

Structura de bază a unui astfel de nod este prezentată în figura 2.20.

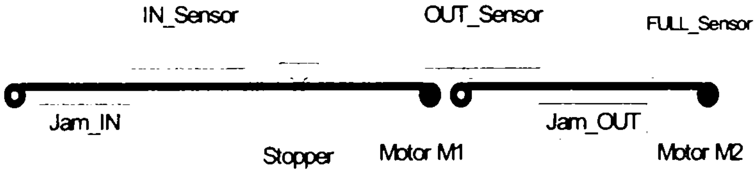


Figura 2.20. Nodul de tip 1. O intrare o ieşire

După cum se observă in figura 2.20, un astfel de nod conţine un singur stoper, având o intrare şi o singură ieşire. Senzorii utilizaţi au următoarele funcţii:

IN_Sensor	Senzorul din stoper. Cu ajutorul acestui senzor se semnalizează faptul că un cărucior este în stoper.
OUT_Sensor	Senzorul de ieşire. Cu ajutorul acestui senzor se urmăreşte momentul în care un cărucior a părăsit zona stoperului (a nodului).
FULL_Sensor	Senzor de plin. Cu ajutorul acestui senzor se verifică spaţiul disponibil în jam-ul următor.

Functionarea nodului de tip 1 este descrisa prin intermediul cronogramei din figura 2.21.

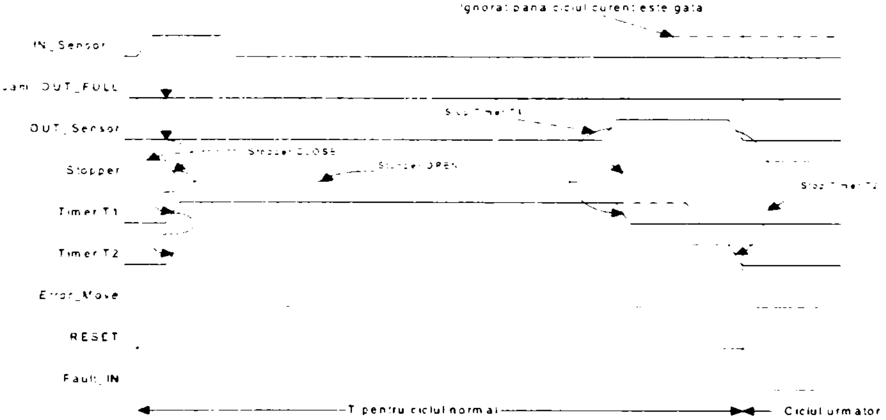


Figura 2.21. Cronogramele de funcţionare a nodului de tip 1

2.5.1.2. Nodul de tip 2 – două intrări o ieşire [UP06a][UP06b] [Ung06c] [Ung06d]

Structura de bază a unui astfel de nod este prezentată în figura 2.22.

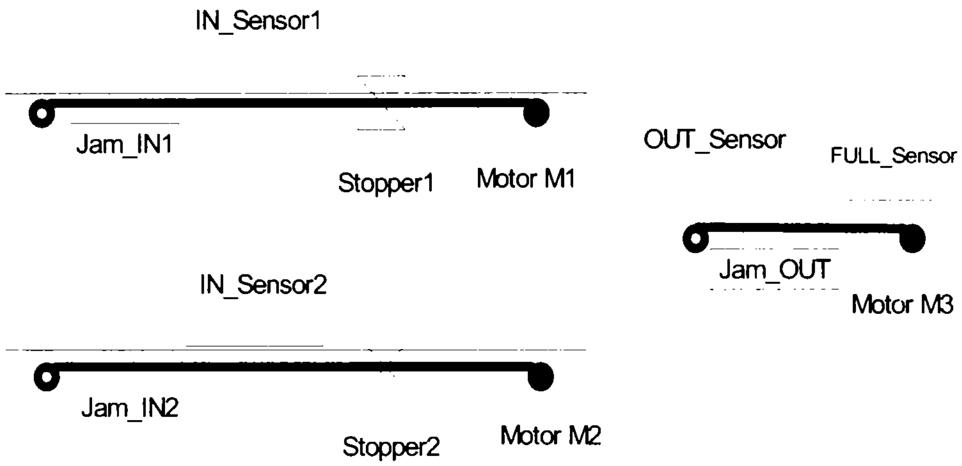


Figura 2.22. Nodul de tip 2. Două intrări o ieşire

Se observă că în plus faţă de nodul de tip 1, acest nod are două stopere şi un macaz. Semnificaţia senzorilor este similară cu cea pentru nodul 1. După cum se observă, dacă un cărucior este eliberat din stoperul 1 sau 2 (eliberarea simultană este exclusă), după poziţionarea macazului, totul se reduce la un nod de tip 1.

2.5.1.3. Nodul de tip 3 - trei intrări o ieşire [UP06a][UP06b] [Ung06b] [Ung06c]

Structura de bază a unui astfel de nod este prezentată în figura 2.23.

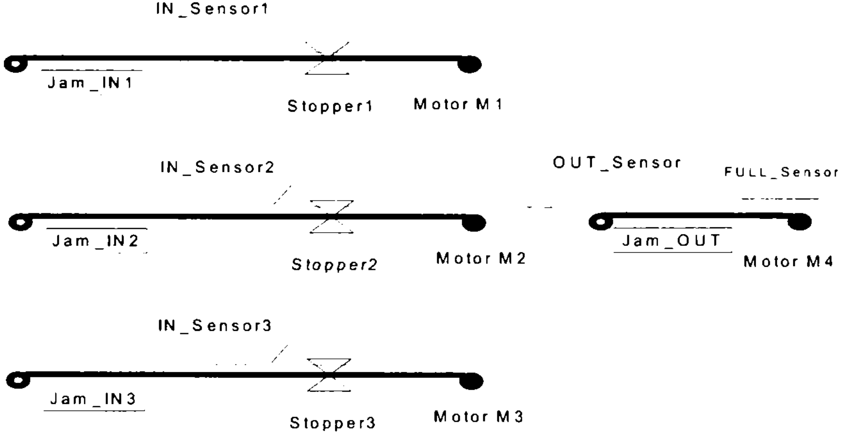


Figura 2.23. Nodul de tip 3. Trei intrări o ieşire

Se observă că în plus faţă de nodul de tip 2, acest nod are trei stopere şi două macazuri. Semnificaţia senzorilor este similară cu cea pentru nodul 1. După cum se observă, dacă un cărucior este eliberat din stoperul 1, 2 sau 3 (eliberarea simultană este exclusă), după poziţionarea macazelor, totul se reduce la un nod de tip 1.

2.5.1.4. Nodul de tip 4 – o intrare două ieşiri [UP06a][UP06b] [Ung06b][Ung06c]

Structura de bază a unui astfel de nod este prezentată în figura 2.24.

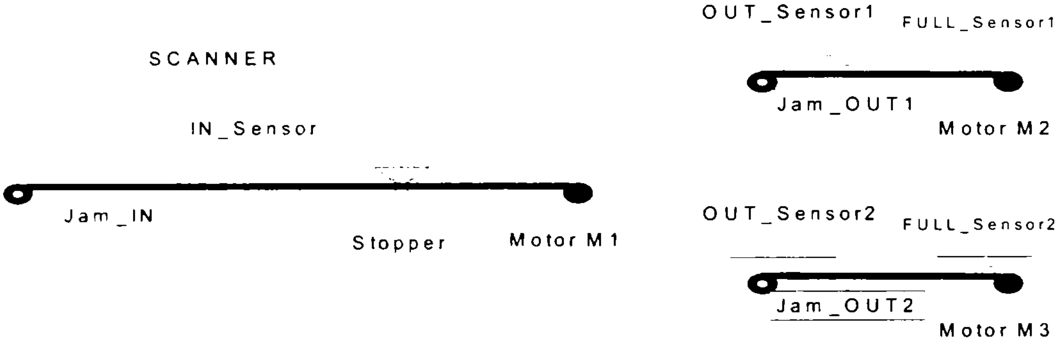


Figura 2.24. Nodul de tip 4. O intrare două ieşiri

Se observă că în plus faţă de nodul de tip 1, acest nod are în faţa stoperului un element de identificare (scanner sau staţie de identificare) şi un macaz. După identificarea căruciorului şi calcularea direcţiei de deplasare se poziţionează macazul, totul reducându-se în acest moment la un nod de tip 1.

După cum rezultă din cele prezentate, se pot construi o multitudine de configuraţii. Pentru modelare, in cadrul lucrarii se consideră, ca elemente de baza, numai cele patru tipuri de noduri prezentate, pe baza acestora putandu-se construi orice structura.

2.5.2. Modelarea elementelor componente ale sistemelor de transport cu zone de acumulare considerate ca SED

Modelarea sistemelor de transport cu zone de acumulare cu ajutorul automatelor secventiale, respectiv a reţelelor Petri, se reduce la modelarea şi simularea nodurilor de tip 1, 2, 3 şi 4 prezentate anterior. Ca mediu de modelare şi simulare s-a folosit MATLAB 6.5 şi un tool specific PN-Toolbox (PNT), elaborat la Facultatea de Automatică şi Calculatoare Iaşi si validat de firma MathWorks[MMP05].

Pentru modelare şi simulare s-au considerat (pentru fiecare tip de nod):
- automatul secvential corespunzator;
- reţea Petri netemporizata;
- retea Petri temporizata P.

2.5.2.1. Modelarea nodului de tip 1 – o intrare o iesire

2.5.2.1.1. Modelarea nodului de tip 1 cu ajutorul automatelor [UP06a] [Ung06b]

Structura automatului secvential considerat pentru modelarea nodului de tip 1 este prezentata în figura 2.25.

Semnificaţiile stărilor automatului secvential *AS_N1* sunt prezentate in tabelul 2.1.

Observatie:*In cadrul tuturor modelelor de tip automat toate starile sunt considerate ca fiind stari marcate si din acest motiv multimea starilor marcate* X_m *nu mai este prezentata.*

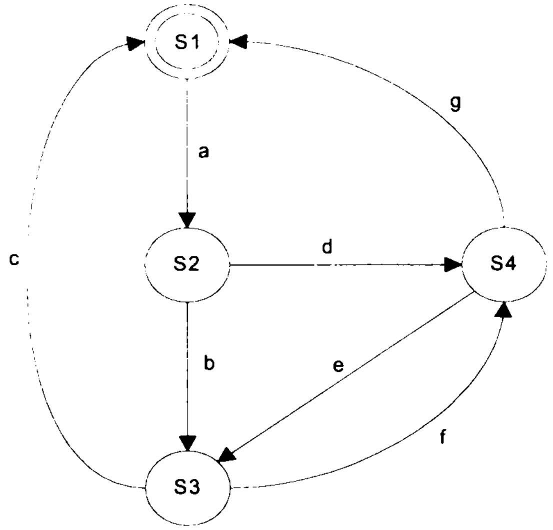


Figura 2.25. Structura automatului secvential corespunzător nodului de tip 1

Tabelul 2.1. *Semnificatiile starilor automatului AS_N1*

Stare	Comentariu
S1	WAIT - starea iniţială a sistemului. În această stare se aşteaptă ca un cărucior să ajungă în stoper (IN_Sensor = 1). Tot în această stare se asigură transportul unui cărucior din senzorul de intrare în nod până în IN_Sensor.
S2	OPEN - starea în care stoperul este deschis, asigurându-se astfel transferul căruciorului în jam-ul următor.
S3	CLOSE - stare corespunzătoare pentru stoper închis, dar căruciorul se află încă în zona nodului, nefiind încă trecut complet în jam-ul următor.
S4	EROARE - stare de eroare care se instalează dacă în timpul mişcării căruciorului a apărut o situaţie critică.

Evenimentele sub care au loc tranzitiile dintre stari sunt prezentate in tabelul 2.2.

Tabelul 2.2. *Semnificatile evenimentelor corespunzatoare automatului AS_N1*

Eveniment	Comentariu
a	Eveniment care asigură trecerea din starea S1 în starea S2. Validat (activ) pe baza conditiei: IN_Sensor * !OUT_Sensor * !FULL.
b	Eveniment care asigura trecerea din starea S2 în starea S3. Validat (activ) pe baza conditiei: FP_OUT_Sensor (front pozitiv la OUT_Sensor).
c	Eveniment care asigură trecerea din starea S3 în starea S1. Validat (activ) pe baza conditiei: FN_OUT_Sensor (front negativ la

	OUT_Sensor).
d	Eveniment care asigură trecerea din starea S2 în starea S4. Validat (activ) pe baza conditiei: FN_Timer_T2 (front negativ la timerul T2 - expirare timer).
e	Eveniment care asigură trecerea din starea S4 în starea S3. Validat (activ) pe baza conditiei: FP_OUT_Sensor.
f	Eveniment care asigură trecerea din starea S3 în starea S4. Validat (activ) pe baza conditiei: FN_Timer_T3 (timerul T3 expirat).
g	Eveniment care asigură trecerea din starea S4 în starea S1. Validat (activ) pe baza conditiei: FN_OUT_Sensor + RESET.

Automatul corespunzator nodului de tip 1 este $AS\_N1 = (X, \Sigma, \delta, \Gamma, x_0)$, unde:

- multimea starilor: $X = \{S1, S2, S3, S4\}$
- multimea evenimentelor: $\Sigma = \{a, b, c, d, e, f, g\}$
- multimile evenimentelor posibile si functiile de tranzitie a starilor:

$$\begin{array}{lll}
\Gamma(S1) = \{a\} & \delta(S1,a) = S2 & \\
\Gamma(S2) = \{b,d\} & \delta(S2,b) = S3, & \delta(S2,d) = S4 \\
\Gamma(S3) = \{c,f\} & \delta(S3,c) = S1, & \delta(S3,f) = S4 \\
\Gamma(S4) = \{e,g\} & \delta(S4,e) = S3 & \delta(S4,g) = S1
\end{array}$$

- starea initiala: $x_0 = S1$

Analiza modelului de tip automat stabilit pentru nodul de tip 1 s-a realizat utilizand mediul de simulare Petri Nets Toolbox sub Matlab 6.5. Pentru ca analiza automatului sa fie validata, conversia din modelul de tip automat in model de tip retea Petri trebuie facuta astfel incat topologia retelei rezultate sa fie tot de tip automat.

Pentru conversia din model de tip automat in model de tip retea Petri s-a utilizat algoritmul 2.2 de conversie descris in paragraful 2.3 .

Aplicand algoritmul selectat rezulta:

- multimea pozitiilor: $P = \{p1, p2, p3, p4\}$,

unde $p1 = \{S1\}, p2 = \{S2\}, p3 = \{S3\}$ si $p4 = \{S4\}$;

- multimea trranzitiilor:

• Pentru $p1$:	$(p1,p2)$	$p2 = \delta(p1,t1)$	$t1 = \{a\}$
• Pentru $p2$:	$(p2,p3)$	$p3 = \delta(p2,t2)$	$t2 = \{b\}$
	$(p2,p4)$	$p4 = \delta(p2,t4)$	$t4 = \{d\}$
• Pentru $p3$:	$(p3,p1)$	$p1 = \delta(p3,t3)$	$t3 = \{c\}$
	$(p3,p4)$	$p4 = \delta(p3,t6)$	$t6 = \{f\}$
• Pentru $p4$:	$(p4,p1)$	$p1 = \delta(p4,t7)$	$t7 = \{g\}$
	$(p4,p3)$	$p3 = \delta(p4,t5)$	$t5 = \{e\}$

In figura 2.26 se prezinta structura retelei Petri asociata automatului secvential considerat in cazul nodului de tip 1.

Semnificaţile pozitiilor P sunt prezentate in tabelul 2.3.

Tabelul 2.3. *Semnificatiile pozitilor P*

Pozitie	Comentariu
P1	WAIT - pozitia iniţială a sistemului. În această stare se aşteaptă ca un cărucior să ajungă în stoper (IN_Sensor = 1). Tot în această stare se asigură transportul unui cărucior din senzorul de intrare în nod până în IN_Sensor.
P2	OPEN - pozitia în care stoperul este deschis, asigurându-se astfel transferul căruciorului în jam-ul următor.
P3	CLOSE - pozitie corespunzătoare pentru stoper închis, dar căruciorul se află încă în zona nodului, nefiind încă trecut complet în jam-ul următor.
P4	EROARE - pozitia de eroare care se instalează dacă în timpul mişcării căruciorului a apărut o situaţie critică.

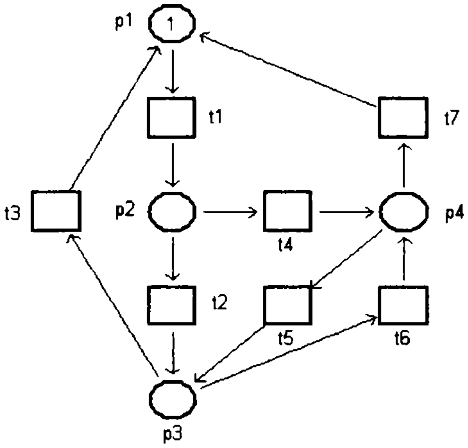


Figura 2.26. Structura retelei Petri asociata automatului secvential corespunzător nodului de tip 1

Semnificatiile tranziţilor *t* sunt prezentate in tabelul 2.4.

Tabelul 2.4. *Semnificatiile tranzitilor t*

Tranziţie	Comentariu
t1	Tranziţie care asigură trecerea din starea P1 în starea P2. Această tranziţie este activată dacă este îndeplinită condiţia: IN_Sensor * !OUT_Sensor * !FULL.
t2	Tranziţie care asigura trecerea din starea P2 în starea P3. Această tranziţie este activată dacă este îndeplinită condiţia: FP_OUT_Sensor.
t3	Tranziţie care asigură trecerea din starea P3 în starea P1. Această tranziţie este activată dacă este îndeplinită condiţia: FN_OUT_Sensor.
t4	Tranziţie care asigură trecerea din starea P2 în starea P4. Această tranziţie este activată dacă este îndeplinită condiţia: FN_Timer_T2.
t5	Tranziţie care asigură trecerea din starea P4 în starea P3. Această tranziţie este activată dacă este îndeplinită condiţia: FP_OUT_Sensor.
t6	Tranziţie care asigură trecerea din starea P3 în starea P4. Această tranziţie este activată dacă este îndeplinită condiţia: FN_Timer_T3.
t7	Tranziţie care asigură trecerea din starea P4 în starea P1. Această tranziţie este activată dacă este îndeplinită condiţia: FN_OUT_Sensor + RESET.

Topologia retelei (rezultat obtinut cu ajutorul lui PNT) este prezentata in figura 2.27.

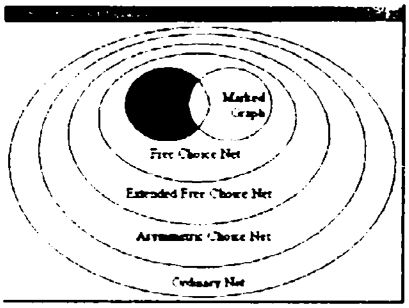


Figura 2.27. Topologia retelei Petri validata de PNT

Dupa cum se poate observa, in urma transformarii modelului de tip automat in model retea Petri, topologia retelei obtinute este tot de tip automat („state machine"), ceea ce permite analiza modelului de tip automat utilizand tehnicile din cadrul retelelor Petri.

Reteaua Petri rezultata este formalizata matematic prin cvintuplul:

$$PN\_AS\_N1 = (P, T, F, W, M_0)$$

unde:

- $P = \{p_1, p_2, p_3, p_4\}$
- $T = \{t_1, t_2, t_3, t_4, t_5, t_6, t_7\}$
- $F = \{(p_1,t_1),(p_2,t_2),(p_2,t_4),(p_3,t_3),(p_3,t_6),(p_4,t_5),(p_4,t_7)\} \cup$ $\{(t_1,p_2),(t_2,p_3),(t_3,p_1),(t_4,p_4),(t_5,p_3),(t_6,p_4),(t_7,p_1)\}$
- $W(p_1,t_1) = 1,\ W(p_2,t_2) = 1,\ W(p_2,t_4) = 1,$
  $W(p_3,t_3) = 1,\ W(p_3,t_6) = 1,\ W(p_4,t_5) = 1,$
  $W(p_4,t_7) = 1,\ W(t_1,p_2) = 1,\ W(t_2,p_3) = 1,$
  $W(t_3,p_1) = 1,\ W(t_4,p_4) = 1,\ W(t_5,p_3) = 1,$
  $W(t_6,p_4) = 1,\ W(t_7,p_1) = 1$
- $M_0 = [1,0,0,0]^T$

Matricile de incidenţă corespunzătoare sunt:

Matricea de incidenta de intrare:

$$A_I = \begin{pmatrix} 1&0&0&0 \\ 0&1&0&0 \\ 0&0&1&0 \\ 0&1&0&0 \\ 0&0&0&1 \\ 0&0&1&0 \\ 0&0&0&1 \end{pmatrix}$$

Matricea de incidenta de iesire:

$$A_O = \begin{pmatrix} 0&1&0&0 \\ 0&0&1&0 \\ 1&0&0&0 \\ 0&0&0&1 \\ 0&0&1&0 \\ 0&0&0&1 \\ 1&0&0&0 \end{pmatrix}$$

Matricea de incidenta: $A = A_O - A_I = \begin{pmatrix} -1 & 1 & 0 & 0 \\ 0 & -1 & 1 & 0 \\ 1 & 0 & -1 & 0 \\ 0 & -1 & 0 & 1 \\ 0 & 0 & 1 & -1 \\ 0 & 0 & -1 & 1 \\ 1 & 0 & 0 & -1 \end{pmatrix}$

Arborele de acoperire corespunzator este prezentat in figura 2.28.

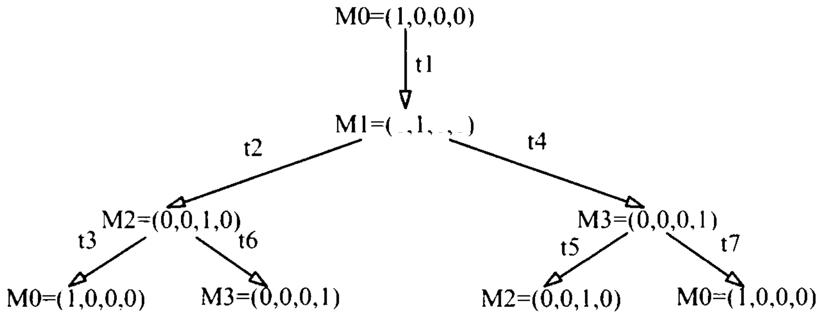


Figura 2.28 Arborele de acoperire al retelei Petri *PN_AS_N1*

Studiind proprietatile comportamentale ale retelei Petri *PN_AS_N1*, pe baza arborelui de acoperire rezulta:

- reteaua este *marginita* (simbolul ω nu apare in arborele de acoperire);
- reteaua este *sigura* (marcajele din toate nodurile arborelui de acoperire contin numai „0" si „1");
- reteaua *nu este blocanta* (toate tranzitiile au asociate arce in arborele de acoperire);
- reteaua este *accesibila* (orice marcaj poate fi atins pornind din M_0).

Graful de acoperire corespunzator este prezentat in *figura* 2.29.

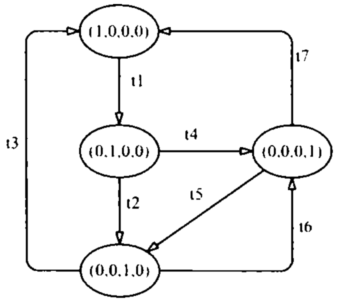


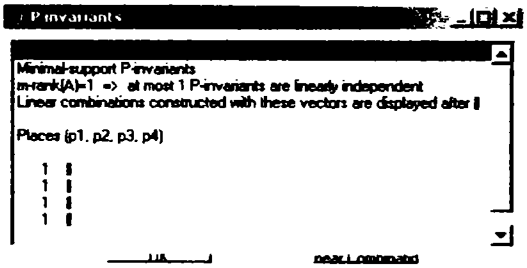
Figura 2.29. Graful de acoperire al retelei Petri *PN_AS_N1*

Structura rezultata pentru graful de acoperire confirma faptul ca modelul de retea Petri derivata din modelul de tip automat corespunzator nodului de tip 1 este *accesibil*.

Concluzionand, se poate afirma ca modelul ales este *viabil* din punct de vedere comportamental.

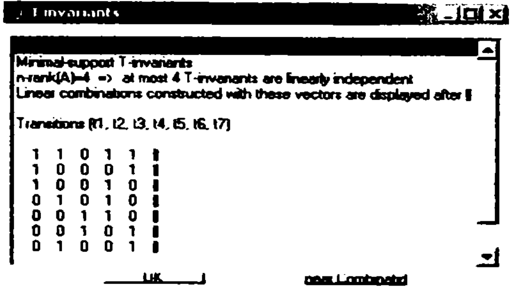
Analiza retelei Petri *PN_AS_N1* din punct de vedere structural se face pe baza analizei existentei invariantilor de tip *P* respectiv *T*. Pentru determinarea numarului de invarianti *P*, respectiv *T*, s-a aplicat teorema 2.3 (Determinarea numarului de invarianti), rangul matricei A este: $rang\,A = 3$.

Rezulta ca reteaua Petri *PN_AS_N1* este acoperita de 1 invariant de tip *P*. In mod similar, in cazul invariantiantilor de tip *T* rezulta un numar de 4 astfel de invarianti. In figurile 2.30 si 2.31 se prezinta invariantii de tip P respectiv T obtinuti in urma simularii retelei Petri *PN_AS_N1* cu ajutorul PNT-ului.



```
Minimal-support P-invariants
m-rank(A)=1  =>  at most 1 P-invariants are linearly independent
Linear combinations constructed with these vectors are displayed after |

Places (p1, p2, p3, p4)

   1  |
   1  |
   1  |
   1  |
```

Figura 2.30 Invariantii de tip P obtinuti cu ajutorul PNT



```
Minimal-support T-invariants
n-rank(A)=4  =>  at most 4 T-invariants are linearly independent
Linear combinations constructed with these vectors are displayed after |

Transitions (t1, t2, t3, t4, t5, t6, t7)

  1  1  0  1  1  |
  1  0  0  0  1  |
  1  0  0  1  0  |
  0  1  0  1  0  |
  0  0  1  1  0  |
  0  0  1  0  1  |
  0  1  0  0  1  |
```

Figura 2.31 Invariantii de tip T obtinuti cu ajutorul PNT

Conform teoremelor 2.6, respectiv 2.7, *reteaua Petri PN_AS_N1 este conservativa si structural marginita, respectiv este consistenta si repetitiva.*

În final se poate *concluziona* că structura aleasă pentru un automat secvential clasic corespunzător nodului de tip 1 este viabilă, ea putând fi implementată, fiind siguri că nu există condiţii de apariţie a blocajelor sau a apariţiei de situaţii necontrolabile.

2.5.2.1.2. Modelarea nodului de tip 1 cu ajutorul retelelor Petri

2.5.2.1.2.1. Cazul: retea Petri netemporizata [UP06b] [Ung06c]

Structura retelei Petri considerate pentru modelarea nodului de tip 1 este prezentata în figura 2.32.

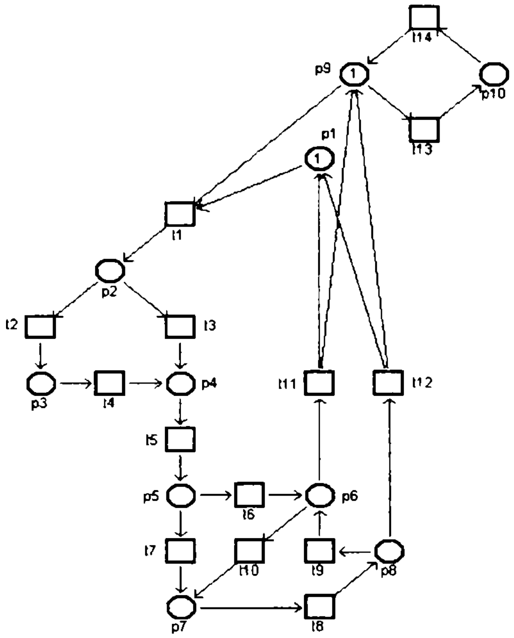


Figura 2.32. Structura retelei Petri corespunzătoare nodului de tip 1

Semnificaţiile pozitiilor *P* sunt prezentate in tabelul 2.5.

Tabelul 2.5. *Semnificatiile pozitiilor corespunzatoare retelei Petri din figura 2.32*

Pozitie	Comentariu
P1	Pozitia WAIT in care nu exista carucior
P2	Pozitia READY care corespunde situatiei in care exista un carucior si este loc in jam_out
P3	Pozitia OUT_BLOC corespunzatoare faptului ca in zona de iesire se afla un carucior care nu a parasit complet nodul
P4	Pozitia STOPER_OPEN corespunde faptului ca stoperul este deschis caruciorul deplasandu-se spre jam_out
P5	Pozitia MOVE_OUT corespunde miscarii caruciorului care se afla inca in zona stoperului
P6	Pozitia EROARE corespunzatoare aparitiei unei situatii anormale de functionare
P7	Pozitia STOPER_CLOSE corespunde situatiei in care caruciorul a trecut complet de stoper care a fost inchis
P8	Pozitia MOVE_END corespunzatoare situatiei in care se asteapta parasirea senzorului OUT de catre carurior.
P9	Pozitia Nu_FULL care corespunde situatiei in care in zona de iesire este loc pentru a muta un carucior
P10	Pozitia FULL care corespunde situatiei in care nu exista spatiu in zona de iesire (jam_out) pentru mutarea unui carucior

Semnificatiile tranziţiilor *t* corespunzatoare retelei Petri din figura 2.32 sunt prezentate in tabelul 2.6.

Tabelul 2.6. *Semnificatiile tranzitilor t corespunzatoare retelei Petri din figura 2.32.*

Tranziţie	Comentariu
t1	Activata in cazul in care in jam_out este spatiu si un carucior a sosit in stoper
t2	Activata daca iesirea nodului este blocata
t3	Activata daca iesirea nodului este libera
t4	Activata daca iesirea nodului s-a eliberat
t5	Activata daca caruciorul a inceput sa se miste spre sensorul OUT
t6	Activata daca in timpul deplasarii caruciorului spre iesire a aparut o anomalie in functionare
t7	Activata daca in urma deplasarii caruciorului s-a atins senzorul OUT
t8	Activata dupa inchiderea stoperului
t9	Activata daca in timpul deplasarii caruciorului spre jam_out a aparut o anomalie in functionare
t10	Activata daca eroarea aparuta a fost eliminata automat
t11	Activata in urma unei operatii de reset
t12	Activata la parasirea de catre carucior a senzorului OUT (sfarsit normal)
t13	Activata in cazul in care jam_out este plin
t14	Activata in cazul in care in jam_out s-a facut loc pentru unul sau mai multe carucioare

Topologia retelei Petri validata de PNT este prezentata in figura 2.33.

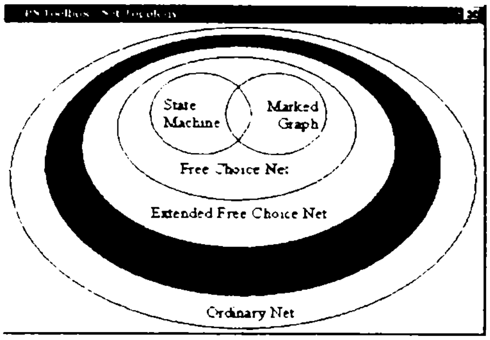


Figura 2.33. Topologia retelei Petri validata de PNT

Matematic reteaua Petri considerata este formalizata prin cvintuplul:

$$PN\_PT\_N1 = (P,T,F,W,M_0)$$

unde:

- $P = \{p_1, p_2, p_3, p_4, p_5, p_6, p_7, p_8, p_9, p_{10}\}$
- $T = \{t_1, t_2, t_3, t_4, t_5, t_6, t_7, t_8, t_9, t_{10}, t_{11}, t_{12}, t_{13}, t_{14}\}$
- $F = \{(p_1,t_1),(p_2,t_2),(p_2,t_3),(p_3,t_4),(p_4,t_5),(p_5,t_6),(p_5,t_7),(p_6,t_{10}),(p_6,t_{11}),$ $(p_7,t_8),(p_8,t_9),(p_8,t_{12}),(p_9,t_1),(p_9,t_{13}),(p_{10},t_{14}) \cup$ $\{(t_1,p_2),(t_2,p_3),(t_3,p_4),(t_4,p_4),(t_5,p_5),(t_6,p_6),(t_7,p_7),(t_8,p_8),(t_9,p_6),(t_{10},p_7),$ $(t_{11},p_1),(t_{11},p_9),(t_{12},p_1),(t_{12},p_9),(t_{13},p_{10}),(t_{14},p_9)\}$

- $W(p_1,t_1)=1, W(p_2,t_2)=1, W(p_2,t_3)=1, W(p_3,t_4)=1, W(p_4,t_5)=1, W(p_5,t_6)=1,$ $W(p_5,t_7)=1, W(p_6,t_{10})=1, W(p_6,t_{11})=1, W(p_7,t_8)=1, W(p_8,t_9)=1, W(p_8,t_{12})=1,$ $W(p_9,t_1)=1, W(p_9,t_{13})=1, W(p_{10},t_{14})=1, W(t_1,p_2)=1, W(t_2,p_3)=1, W(t_3,p_4)=1,$ $W(t_4,p_4)=1, W(t_5,p_5)=1, W(t_6,p_6)=1, W(t_7,p_7)=1, W(t_8,p_8)=1, W(t_9,p_6)=1,$ $W(t_{10},p_7)=1, W(t_{11},p_1)=1, W(t_{11},p_9)=1, W(t_{12},p_1)=1, W(t_{12},p_9)=1, W(t_{13},p_{10})=1,$ $W(t_{14},p_9)=1$
- $M_0=[1,\ 0,0,0,0,0,0,0,1,0]^T$

Matricile de incidenţă corespunzătoare sunt:

Matricea de incidenta de intrare:

$$A_{Ii}=\begin{pmatrix}
1&0&0&0&0&0&0&0&1&0\\
0&1&0&0&0&0&0&0&0&0\\
0&1&0&0&0&0&0&0&0&0\\
0&0&1&0&0&0&0&0&0&0\\
0&0&0&1&0&0&0&0&0&0\\
0&0&0&0&1&0&0&0&0&0\\
0&0&0&0&1&0&0&0&0&0\\
0&0&0&0&0&0&1&0&0&0\\
0&0&0&0&0&0&0&1&0&0\\
0&0&0&0&0&1&0&0&0&0\\
0&0&0&0&0&1&0&0&0&0\\
0&0&0&0&0&0&0&1&0&0\\
0&0&0&0&0&0&0&0&1&0\\
0&0&0&0&0&0&0&0&0&1
\end{pmatrix}$$

Matricea de incidenta de iesire:

$$A_{Oi}=\begin{pmatrix}
0&1&0&0&0&0&0&0&0&0\\
0&0&1&0&0&0&0&0&0&0\\
0&0&0&1&0&0&0&0&0&0\\
0&0&0&1&0&0&0&0&0&0\\
0&0&0&0&1&0&0&0&0&0\\
0&0&0&0&0&1&0&0&0&0\\
0&0&0&0&0&0&1&0&0&0\\
0&0&0&0&0&0&0&1&0&0\\
0&0&0&0&0&1&0&0&0&0\\
0&0&0&0&0&0&1&0&0&0\\
1&0&0&0&0&0&0&0&1&0\\
1&0&0&0&0&0&0&0&1&0\\
0&0&0&0&0&0&0&0&0&1\\
0&0&0&0&0&0&0&0&1&0
\end{pmatrix}$$

Matricea de incidenta:

$$A_i=A_{Oi}-A_{Ii}=\begin{pmatrix}
-1&1&0&0&0&0&0&0&-1&0\\
0&-1&1&0&0&0&0&0&0&0\\
0&-1&0&1&0&0&0&0&0&0\\
0&0&-1&1&0&0&0&0&0&0\\
0&0&0&-1&1&0&0&0&0&0\\
0&0&0&0&-1&1&0&0&0&0\\
0&0&0&0&-1&0&1&0&0&0\\
0&0&0&0&0&0&-1&1&0&0\\
0&0&0&0&0&1&0&-1&0&0\\
0&0&0&0&0&-1&1&0&0&0\\
1&0&0&0&0&-1&0&0&1&0\\
1&0&0&0&0&0&0&-1&1&0\\
0&0&0&0&0&0&0&0&-1&1\\
0&0&0&0&0&0&0&0&1&-1
\end{pmatrix}$$

Arborele de acoperire corespunzator retelei Petri *PN_PT_N1* este prezentat in figura 2.34.

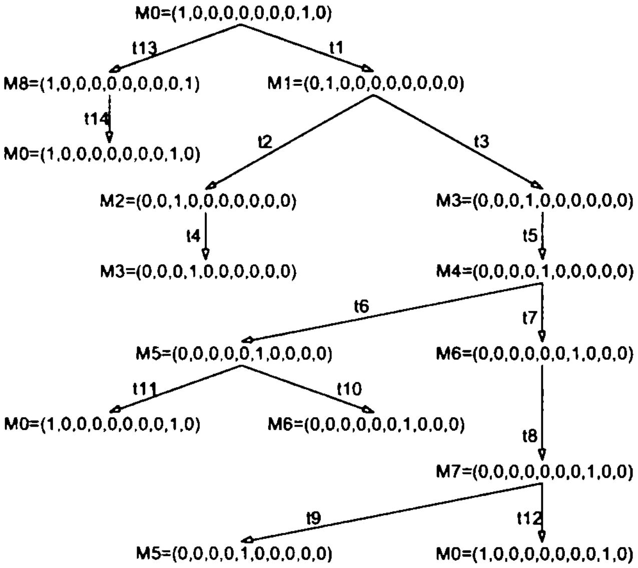


Figura 2.34: Arborele de acoperire corespunzator *PN_PT_N1*

Studiind proprietatile comportamentale ale retelei Petri *PN_PT_N1*, pe baza arborelui de acoperire rezulta:

- reteaua este *marginita* (simbolul ω nu apare in arborele de acoperire);
- reteaua este *sigura* (marcajele din toate nodurile arborelui de acoperire contin numai „0” si „1”);
- reteaua *nu este blocanta* (toate tranzitile au asociate arce in arborele de acoperire);
- reteaua este *accesibila* (orice marcaj poate fi atins pornind din M_0).

Graful de acoperire corespunzator este prezentat in figura 2.35

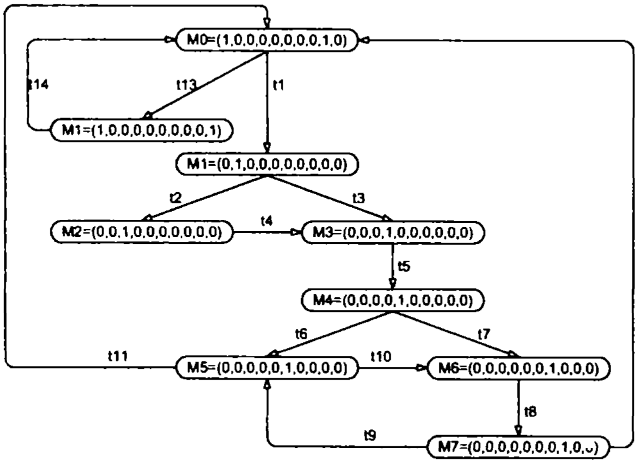


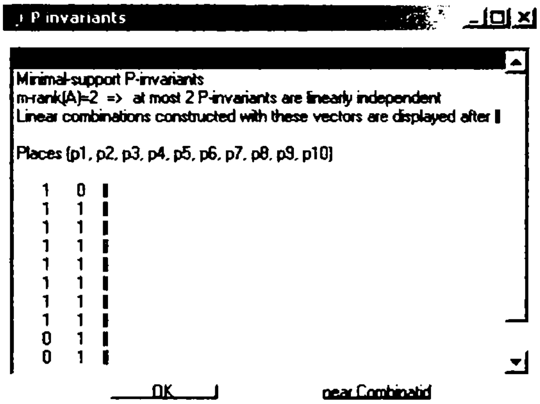
Figura 2.35. Graful de acoperire al retelei Petri *PN_PT_N1*

Structura rezultata pentru graful de acoperire confirma ca modelul de retea Petri corespunzator nodului de tip 1 este *accesibil* si *viabil* din punct de vedere comportamental.

Analiza retelei Petri *PN_PT_N1* din punct de vedere structural se face pe baza analizei existentei invariantilor de tip *P* respectiv *T*. Pentru determinarea numarului de invarianti *P* respectiv *T* s-a aplicat teorema2.3 (Determinarea numarului de invarianti [Pastr97]), rangul matricei A este: $rang\ A = 8$.

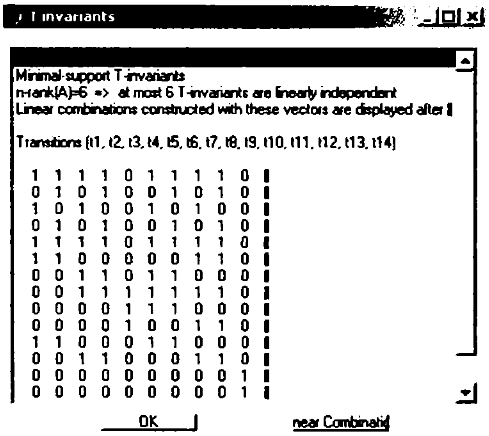
Rezulta ca reteaua Petri *PN_PT_N1* este acoperita de 2 invarianti de tip *P*, respectiv 6 invarianti de tip *T*.

In figurile 2.36 si 2.37 se prezinta invariantii de tip *P* respectiv *T* obtinuti in urma simularii retelei Petri *PN_PT_N1* cu ajutorul PNT-ului.



```
Minimal-support P-invariants
m-rank(A)=2  =>  at most 2 P-invariants are linearly independent
Linear combinations constructed with these vectors are displayed after |

Places (p1, p2, p3, p4, p5, p6, p7, p8, p9, p10)

  1  0  |
  1  1  |
  1  1  |
  1  1  |
  1  1  |
  1  1  |
  1  1  |
  1  1  |
  0  1  |
  0  1  |
```

Figura 2.36 Invariantii de tip P obtinuti cu ajutorul PNT



```
Minimal-support T-invariants
n-rank(A)=6  =>  at most 6 T-invariants are linearly independent
Linear combinations constructed with these vectors are displayed after |

Transitions (t1, t2, t3, t4, t5, t6, t7, t8, t9, t10, t11, t12, t13, t14)

  1  1  1  1  0  1  1  1  1  0  |
  0  1  0  1  0  0  1  0  1  0  |
  1  0  1  0  0  1  0  1  0  0  |
  0  1  0  1  0  0  1  0  1  0  |
  1  1  1  1  0  1  1  1  1  0  |
  1  1  0  0  0  0  0  1  1  0  |
  0  0  1  1  0  1  1  0  0  0  |
  0  0  1  1  1  1  1  1  1  0  |
  0  0  0  0  1  1  1  0  0  0  |
  0  0  0  0  1  0  0  1  1  0  |
  1  1  0  0  0  1  1  0  0  0  |
  0  0  1  1  0  0  0  1  1  0  |
  0  0  0  0  0  0  0  0  0  1  |
  0  0  0  0  0  0  0  0  0  1  |
```

Figura 2.37 Invariantii de tip T obtinuti cu ajutorul PNT

Conform teroemelor 2.6 respectiv 2.7 *reteaua Petri PN_PT_N1 este conservativa si structural marginita, respectiv este consistenta si repetitiva.*

În final se poate *concluziona* că structura aleasă pentru reteaua Petri corespunzătoare nodului de tip 1 este viabilă, ea putând fi implementată, fiind siguri că nu există condiţii de apariţie a blocajelor sau a apariţiei de situaţii necontrolabile.

2.5.2.1.2.2. Cazul: retea Petri temporizata P

Structura retelei Petri temporizata P corespunzatoare Nodului de tip 1 este aceeasi ca cea prezentata in figura 2.32.

Datorita faptului ca retelele sunt identice din punct de vedere structural si comportamental rezultatele analizelor nu se mai detaliaza, ele fiind in fapt identice cu cele obtinute in cazul nodului de tip 1 netemporizat.

Fata de situatia prezentata anterior fiecarei pozitii P i-a fost alocata o unitate de timp corespunzatoare. In acest mod situatia devine reala din punct de vedere al functionarii.

Timpii corespunzatori pozitiilor *P* sunt prezentati in tabelul 2.7.

Tabelul 2.7. *Timpii corespunzatori pozitiilor P*

Pozitie	Unitati de Timp
P1	1
P2	5
P3	4
P4	4
P5	7
P6	6
P7	3
P8	5
P9	1
P10	3

Matematic reteaua Petri temporizata P este:

$$PN\_PTtime\_N1 = (P,T,F,W,D,M_0)$$

unde:

- $P = \{p_1,p_2,p_3,p_4,p_5,p_6,p_7,p_8,p_9,p_{10}\}$
- $T = \{t_1,t_2,t_3,t_4,t_5,t_6,t_7,t_8,t_9,t_{10},t_{11},t_{12},t_{13},t_{14}\}$
- $F = \{(p_1,t_1),(p_2,t_2),(p_2,t_3),(p_3,t_4),(p_4,t_5),(p_5,t_6),(p_5,t_7),(p_6,t_{10}),(p_6,t_{11}),$
  $(p_7,t_8),(p_8,t_9),(p_8,t_{12}),(p_9,t_1),(p_9,t_{13}),(p_{10},t_{14})\cup$
  $\{(t_1,p_2),(t_2,p_3),(t_3,p_4),(t_4,p_4),(t_5,p_5),(t_6,p_6),(t_7,p_7),(t_8,p_8),(t_9,p_6),(t_{10},p_7),$
  $(t_{11},p_1),(t_{11},p_9),(t_{12},p_1),(t_{12},p_9),(t_{13},p_{10}),(t_{14},p_9)\}$
- $W(p_1,t_1) = 1, W(p_2,t_2) = 1, W(p_2,t_3) = 1, W(p_3,t_4) = 1, W(p_4,t_5) = 1,$
  $W(p_5,t_6) = 1, W(p_5,t_7) = 1, W(p_6,t_{10}) = 1, W(p_6,t_{11}) = 1, W(p_7,t_8) = 1,$
  $W(p_8,t_9) = 1, W(p_8,t_{12}) = 1, W(p_9,t_1) = 1, W(p_9,t_{13}) = 1, W(p_{10},t_{14}) = 1,$
  $W(t_1,p_2) = 1, W(t_2,p_3) = 1, W(t_3,p_4) = 1, W(t_4,p_4) = 1, W(t_5,p_5) = 1,$
  $W(t_6,p_6) = 1, W(t_7,p_7) = 1, W(t_8,p_8) = 1, W(t_9,p_6) = 1, W(t_{10},p_7) = 1,$
  $W(t_{11},p_1) = 1, W(t_{11},p_9) = 1, W(t_{12},p_1) = 1, W(t_{12},p_9) = 1, W(t_{13},p_{10}) = 1,$
  $W(t_{14},p_9) = 1$
- $D = \{d(p_1) = 1, d(p_2) = 5, d(p_3) = 4, d(p_4) = 4, d(p_5) = 7, d(p_6) = 6,$
  $d(p_7) = 3, d(p_8) = 5, d(p_9) = 1, d(p_{10}) = 3\}$
- $M_0 = [1,\ 0,0,0,0,0,0,0,\ 1,0]^T$

În final se poate *concluziona* că structura aleasă pentru reteaua Petri temporizata P corespunzătoare nodului de tip 1 este viabilă, ea putând fi implementată, fiind siguri că nu există condiţii de apariţie a blocajelor sau a apariţiei de situaţii necontrolabile.

2.5.2.2. Modelarea nodului de tip 2 - doua intrari o iesire

După cum s-a prezentat in capitolul 1, nodul de tip 2 (două intrări o ieşire) este construit cu ajutorul nodului de tip 1, având în plus un macaz care este comandat direct funcţie de tranziţia care este executată.

2.5.2.2.1. Modelarea nodului de tip 2 cu ajutorul automatelor [UP06a][Ung06b]

Structura automatului secvential considerat pentru modelarea nodului de tip 2 este prezentata în figura 2.38.

Dupa cum se observa automatul secvential al nodului 2 este realizat prin sincronizarea a doua automate secventiale *AS_N1*, cate unul pentru fiecare linie de intrare (stoper). Astfel, semnificatiile starilor este identica cu cele prezentate la *AS_N1,* cu observatia ca S1, S2, S3 si S4 corespund stoperului 1, iar S5, S6, S7 si S8 corespund stoperului 2. Starea S9 este starea suplimentara de sincronizare a automatelor secventiale corespunzatoare stoperelor 1 respectiv 2.

Evenimentele a1, b1, c1, d1, e1, f1, g1 se refera la stoperul 1 (senzorii corespunzatori stoperului 1 conform relatiilor prezentate la *AS_N1*), iar a2, b2, c2, d2, e2, f2, g2 se refera la stoperul 2 cu aceeasi observatie. Evenimentele h1 respectiv h2 reprezinta conditiile de tranzitie din starea S9 in S1 sau S5 (h1 activ - un carucior a activat senzorul de intrare din stoperul 1, respectiv h2 activ - un carucior a activat senzorul de intrare din stoperul 2).

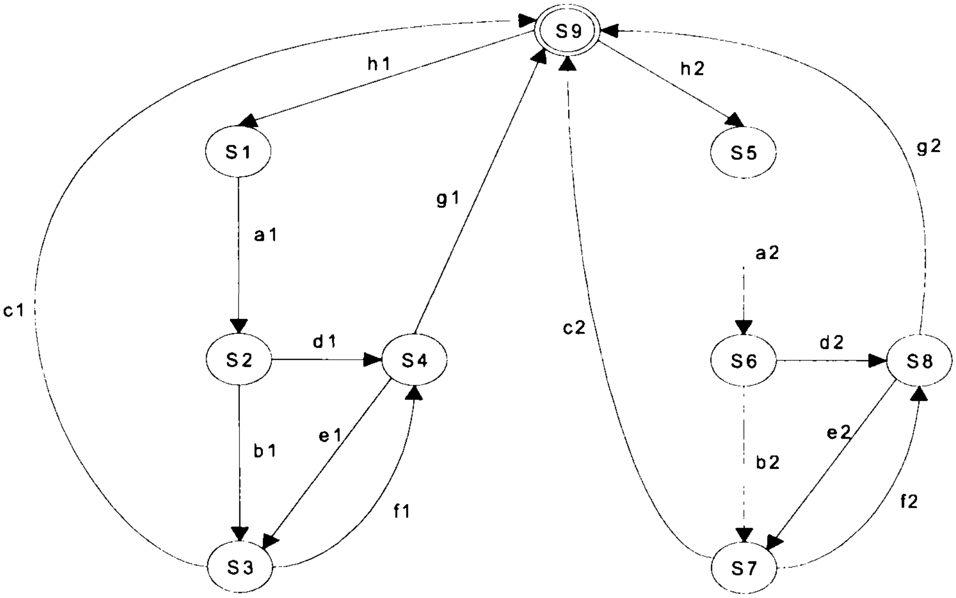


Figura 2.38. Structura automatului secvential corespunzător nodului de tip 2

Automatul *AS_N2* care descrie automatul secvential corespunzator nodului de tip 2 este definit astfel:

- multimea starilor: $X = \{S1, S2, S3, S4, S5, S6, S7, S8, S9\}$
- multimea evenimentelor:
  $\Sigma = \{a1, b1, c1, d1, e1, f1, g1, h1, a2, b2, c2, d2, e2, f2, g2, h2\}$
- multimile evenimentelor posibile si functiile de tranzitie a starilor:
  $\Gamma(S1) = \{a1\}$ $\delta(S1, a1) = S2$
  $\Gamma(S2) = \{b1, d1\}$ $\delta(S2, b1) = S3,$ $\delta(S2, d1) = S4$
  $\Gamma(S3) = \{c1, f1\}$ $\delta(S3, c1) = S9,$ $\delta(S3, f1) = S4$
  $\Gamma(S4) = \{e1, g1\}$ $\delta(S4, e1) = S3$ $\delta(S4, g1) = S9$
  $\Gamma(S5) = \{a2\}$ $\delta(S5, a2) = S6$
  $\Gamma(S6) = \{b2, d2\}$ $\delta(S6, b2) = S7,$ $\delta(S6, d2) = S8$
  $\Gamma(S7) = \{c2, f2\}$ $\delta(S7, c2) = S9,$ $\delta(S7, f2) = S8$
  $\Gamma(S8) = \{e2, g2\}$ $\delta(S8, e2) = S7$ $\delta(S8, g2) = S9$
  $\Gamma(S9) = \{h1, h2\}$ $\delta(S9, h1) = S1,$ $\delta(S9, h2) = S5$
- starea initiala: $x_0 = S9$

Considerentele legate de transformarea modelului de tip automat in model de tip retea Petri, prezentate la nodul de tip 1 sunt valabile si in cazul nodului de tip 2.

Pentru conversia din model de tip automat in model de tip retea Petri s-a utilizat aceasi metoda ca si in cazul nodului de tip 1 . Aplicand algoritmul selectat rezulta:

- multimea pozitiilor: $P = \{p1, p2, p3, p4, p5, p6, p7, p8, p9\}$, unde
  $p1 = \{S1\}, p2 = \{S2\}, p3 = \{S3\}, p4 = \{S4\}, p5 = \{S5\}, p6 = \{S6\}, p7 = \{S7\},$
  $p8 = \{S8\}$ *si* $p9 = \{S9\};$

- multimea tranzitiilor:

• Pentru *p1*	*(p1,p2)*	$p2=\delta(p1,t1)$	$t1=\{a1\}$
• Pentru *p2*	*(p2,p3)*	$p3=\delta(p2,t2)$	$t2=\{b1\}$
	(p2,p4)	$p4=\delta(p2,t4)$	$t4=\{d1\}$
• Pentru *p3*	*(p3,p9)*	$p9=\delta(p3,t3)$	$t3=\{c1\}$
	(p3,p4)	$p4=\delta(p3,t6)$	$t6=\{f1\}$
• Pentru *p4*	*(p4,p9)*	$p9=\delta(p4,t7)$	$t7=\{g1\}$
	(p4,p3)	$p3=\delta(p4,t5)$	$t5=\{e1\}$
• Pentru *p5*	*(p5,p6)*	$p6=\delta(p5,t8)$	$t8=\{a2\}$
• Pentru *p6*	*(p6,p7)*	$p7=\delta(p6,t9)$	$t9=\{b2\}$
	(p6,p8)	$p8=\delta(p6,t11)$	$t11=\{d2\}$
• Pentru *p7*	*(p7,p9)*	$p9=\delta(p7,t10)$	$t10=\{c2\}$
	(p7,p8)	$p8=\delta(p7,t13)$	$t13=\{f2\}$
• Pentru *p8*	*(p8,p9)*	$p9=\delta(p8,t14)$	$t14=\{g2\}$
	(p8,p7)	$p7=\delta(p8,t12)$	$t12=\{e2\}$
• Pentru *p9*	*(p9,p1)*	$p1=\delta(p9,t15)$	$t15=\{h1\}$
	(p9,p5)	$p5=\delta(p9,t16)$	$t16=\{h2\}$

In figura 2.39 se prezinta structura retelei Petri asociata automatului secvential considerat in cazul nodului de tip 2.

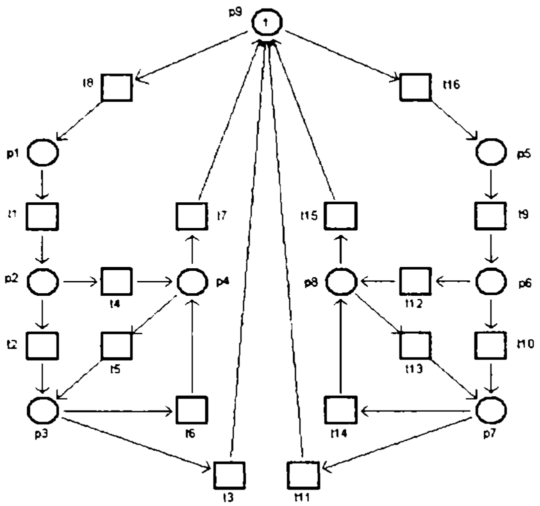


Figura 2.39. Structura retelei Petri asociata automatului secvential corespunzătoare nodului de tip 2

Semnificaţiile pozitiilor sunt similare cu prezentarea făcută pentru nodul de tip 1. Astfel, P1 şi P5 sunt similare cu P1 de la nodul de tip 1, P2 si P6 sunt similare cu P2 de la nodul de tip 1, P3 şi P7 sunt similare cu P3 de la nodul de tip 1, iar P4 şi P8 sunt similare cu P4 de la nodul de tip 1. În plus, pentru asigurarea excluderii mutuale, s-a introdus o pozitie suplimentară P9.

Semnificaţiile tranziţiilor sunt similare cu cele prezentate la nodul de tip 1. Astfel, t1 şi t8 sunt similare cu t1 de la nodul de tip 1, t2 şi t10 sunt similare cu t2 de la nodul de tip 1, t3 şi t10 sunt similare cu t3 de la nodul de tip 1, cu observaţia că apare o condiţie în plus dată de transferul jetonului în P9, t4 şi t11 sunt similare cu t4 de la nodul de tip 1, t5 şi t12 sunt similare cu t5 de la nodul de tip 1, t6 şi t13 sunt similare cu t6 de la nodul de tip 1, t7 şi t14 sunt similare cu t7 de la nodul de tip 1, cu observaţia că apare o condiţie în plus dată de comutarea in starea P9 nu in P1. Tranzitiile t15 si t16 sunt noi, ele asigurand comutarea fie in starea P1, fie in starea P5, functie de pozitia in nod.

Topologia retelei validata de PNT este prezentata in figura 2.40.

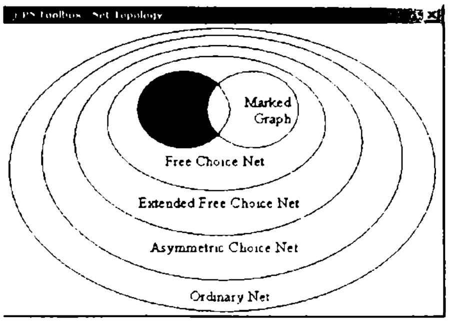


Figura 2.40 Topologia retelei Petri validata de TPN

Rezultatele obtinute confirma, ca si in cazul nodului de tip 1, ca dupa efectuarea conversiei topologia retelei Petri obtinuta este tot de tip automat („State machine").

Reteaua Petri considerata este formalizata matematic prin cvintuplul:

$$PN\_AS\_N2 = (P,T,F,W,M_0)$$

unde:

- $P = \{p_1, p_2, p_3, p_4, p_5, p_6, p_7, p_8, p_9\}$
- $T = \{t_1, t_2, t_3, t_4, t_5, t_6, t_7, t_8, t_9, t_{10}, t_{11}, t_{12}, t_{13}, t_{14}, t_{15}, t_{16}\}$
- $F = \{(p_1,t_1),(p_2,t_2),(p_2,t_4),(p_3,t_6),(p_3,t_3),(p_4,t_5),(p_4,t_7),(p_5,t_8),$
  $(p_6,t_9),(p_6,t_{11}),(p_7,t_{10}),(p_7,t_{13}),(p_8,t_{12}),(p_8,t_{14}),(p_9,t_{15}),$
  $(p_9,t_{16})\} \cup$
  $\{(t_1,p_2),(t_2,p_3),(t_3,p_9),(t_4,p_4),(t_5,p_3),(t_6,p_4),(t_7,p_9),(t_8,p_6),$
  $(t_9,p_7),(t_{10},p_9),(t_{11},p_8),(t_{12},p_7),(t_{13},p_8),(t_{14},p_9),$
  $(t_{15},p_1),(t_{16},p_5)\}$
- $W(p_1,t_1) = 1, W(p_2,t_2) = 1, W(p_2,t_4) = 1, W(p_3,t_6) = 1,$
  $W(p_3,t_3) = 1, W(p_4,t_5) = 1, W(p_4,t_7) = 1, W(p_5,t_8) = 1, W(p_6,t_9) = 1,$
  $W(p_6,t_{11}) = 1, W(p_7,t_{10}) = 1, W(p_7,t_{13}) = 1, W(p_8,t_{12}) = 1, W(p_8,t_{14}) = 1,$
  $W(p_9,t_{15}) = 1, W(p_9,t_{16}) = 1, W(t_1,p_2) = 1, W(t_2,p_3) = 1, W(t_3,p_9) = 1,$
  $W(t_4,p_4) = 1, W(t_5,p_3) = 1, W(t_6,p_4) = 1, W(t_7,p_9) = 1, W(t_8,p_6) = 1,$
  $W(t_9,p_7) = 1, W(t_{10},p_9) = 1, W(t_{11},p_8) = 1, W(t_{12},p_7) = 1,$
  $W(t_{13},p_8) = 1, W(t_{14},p_9) = 1, W(t_{15},p_1) = 1, W(t_{16},p_5) = 1$
- $M_0 = [0,0,0,0,0,0,0,0,1]^T$

Matricile de incidenta corespunzatoare sunt:

Matricea de incidenta de intrare:

$$A_{II} = \begin{pmatrix} 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 \\ 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 \end{pmatrix}$$

Matricea de incidenta de iesire:

$$A_{Oi} = \begin{pmatrix} 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 \\ 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 \\ 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 \\ 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 \end{pmatrix}$$

Matricea de incidenta:

$$A_i = A_{Oi} - A_{Ii} = \begin{pmatrix} -1 & 1 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & -1 & 1 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & -1 & 0 & 0 & 0 & 0 & 0 \\ 0 & -1 & 0 & 1 & 0 & 0 & 0 & 0 \\ 0 & 0 & 1 & -1 & 0 & 0 & 0 & 0 \\ 0 & 0 & -1 & 1 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & -1 & 0 & 0 & 0 & 0 \\ 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & -1 & 1 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & -1 & 1 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & -1 & 0 \\ 0 & 0 & 0 & 0 & 0 & -1 & 0 & 1 \\ 0 & 0 & 0 & 0 & 0 & 0 & 1 & -1 \\ 0 & 0 & 0 & 0 & 0 & 0 & -1 & 1 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & -1 \\ 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 \end{pmatrix}$$

Arborele de acoperire corespunzator este prezentat in figura 2.41

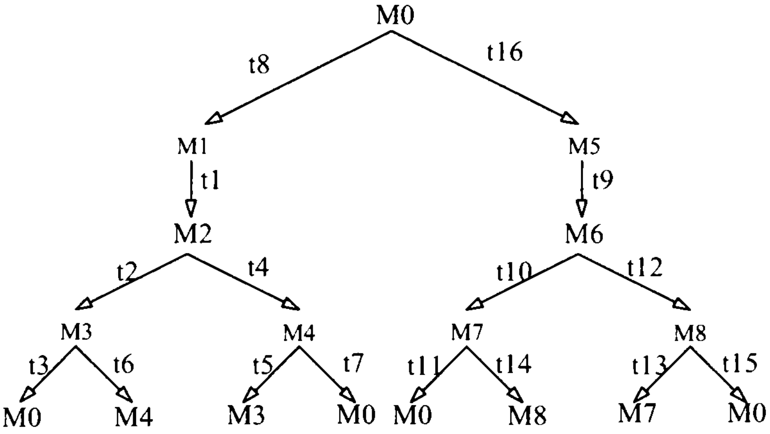


Figura 2.41 Arborele de acoperire al retelei Petri *PN_AS_N2*

Unde:
M0=(0,0,0,0,0,0,0,0,1); M1=(1,0,0,0,0,0,0,0,0); M2=(0,1,0,0,0,0,0,0,0); M3=(0,0,1,0,0,0,0,0,0); M4=(0,0,0,1,0,0,0,0,0); M5=(0,0,0,0,1,0,0,0,0); M6=(0,0,0,0,0,1,0,0,0); M7=(0,0,0,0,0,0,1,0,0); M8=(0,0,0,0,0,0,0,1,0)

Studiind *proprietatile* comportamentale ale retelei Petri *PN_AS_N2*, pe baza arborelui de acoperire rezulta:

- reteaua este *marginita* (simbolul ω nu apare in arborele de acoperire);
- reteaua este *sigura* (marcajele din toate nodurile arborelui de acoperire contin numai „0" si „1");
- reteaua *nu este blocanta* (toate tranzitile au asociate arce in arborele de acoperire);
- reteaua este *accesibila* (orice marcaj poate fi atins pornind din M_0).

Graful de acoperire corespunzator este prezentat in figura 2.42.

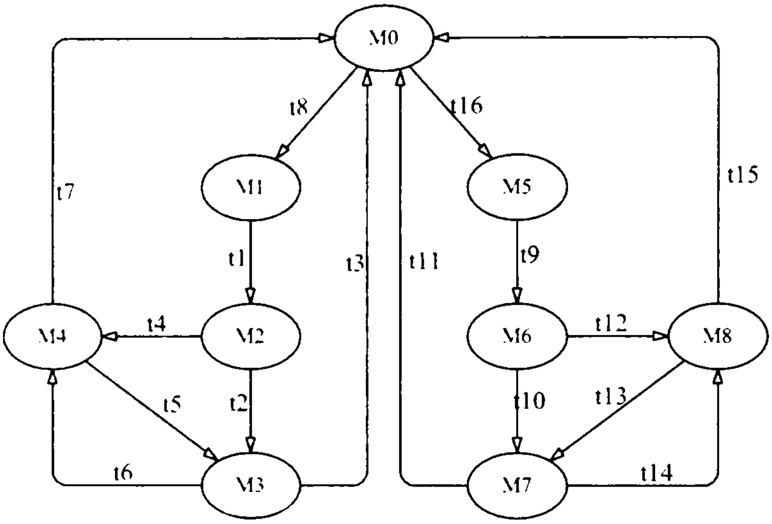


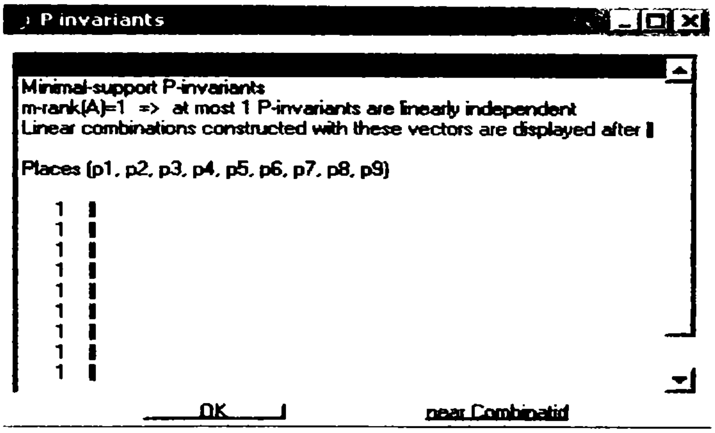
Figura 2.42. Graful de acoperire al retelei Petri *PN_AS_N2*

Structura rezultata pentru graful de acoperire confirma ca modelul de retea Petri derivata din modelul de tip automat corespunzator nodului de tip 2 este *accesibila*, respectiv *viabila* din punct de vedere comportamental.

Analiza retelei Petri *PN_AS_N2* din punct de vedere structural se face pe baza analizei existentei invariantilor de tip *P* respectiv *T*.

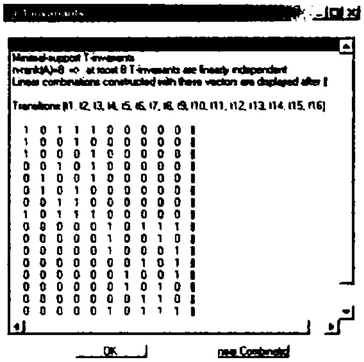
Aplicand teorema 2.3 (determinarea numarului de invarianti), rezulta ca reteaua Petri *PN_AS_N2,* este acoperita de 1 invariant de tip *P,* respectiv 8 invarianti de tip *T*, rangul matricii A fiind: $rang\, A = 8$.

In figurile 2.43 si 2.44 se prezinta invariantii de tip P respectiv T obtinuti in urma simularii retelei Petri *PN_AS_N2* cu ajutorul PNT-ului.



```
Minimal-support P-invariants
m-rank(A)=1  =>  at most 1 P-invariants are linearly independent
Linear combinations constructed with these vectors are displayed after |

Places (p1, p2, p3, p4, p5, p6, p7, p8, p9)

    1  |
    1  |
    1  |
    1  |
    1  |
    1  |
    1  |
    1  |
    1  |
```

Figura 2.43 Invariantii de tip P obtinuti cu ajutorul PNT



```
Minimal-support T-invariants
n-rank(A)=8 => at most 8 T-invariants are linearly independent
Linear combinations constructed with these vectors are displayed after !
Transitions [t1, t2, t3, t4, t5, t6, t7, t8, t9, t10, t11, t12, t13, t14, t15, t16]

1 0 1 1 1 0 0 0 0 0 |
1 0 0 1 0 0 0 0 0 0 |
1 0 0 0 1 0 0 0 0 0 |
0 0 1 0 1 0 0 0 0 0 |
0 1 0 0 1 0 0 0 0 0 |
0 1 0 1 0 0 0 0 0 0 |
0 0 1 1 0 0 0 0 0 0 |
1 0 1 1 1 0 0 0 0 0 |
0 0 0 0 0 1 0 1 1 1 |
0 0 0 0 0 1 0 0 1 0 |
0 0 0 0 0 1 0 0 0 1 |
0 0 0 0 0 0 0 1 0 1 |
0 0 0 0 0 0 1 0 0 1 |
0 0 0 0 0 0 1 0 1 0 |
0 0 0 0 0 0 0 1 1 0 |
0 0 0 0 0 1 0 1 1 1 |
```

Figura 2.44 Invariantii de tip T obtinuti cu ajutorul PNT

Conform teroemelor 2.6 respectiv 2.7, reteaua Petri *PN_AS_N2* este *conservativa si structural marginita, respectiv este consistenta si repetitiva.*

În final se poate *concluziona* că structura aleasă pentru un automat secvential clasic corespunzător nodului de tip 2 este viabilă, ea putând fi implementată, fiind siguri că nu există condiţii de apariţie a blocajelor sau a apariţiei de situaţii necontrolabile.

2.5.2.2.2. Modelarea nodului de tip 2 cu ajutorul retelelor Petri

2.5.2.2.2.1. Cazul: retea Petri netemporizata [UP06b][Ung06c]

Structura retelei Petri considerate pentru modelarea nodului de tip 2 este prezentata în figura 2.45.

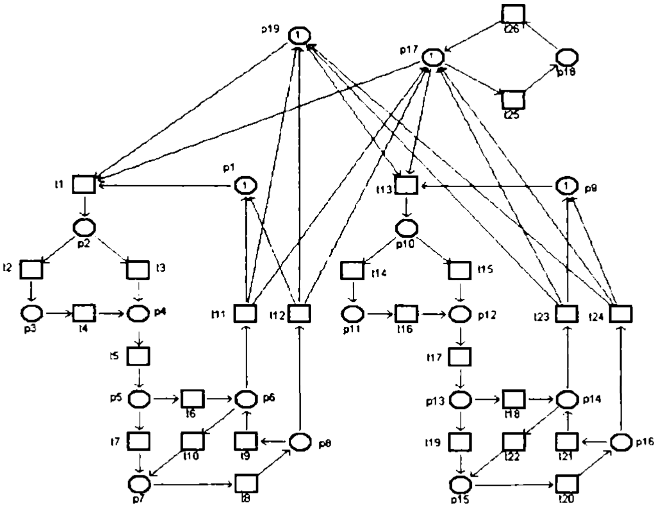


Figura 2.45. Structura retelei Petri corespunzătoare nodului de tip 2

Semnificaţiile pozitiilor *P* sunt identice cu cele prezentate pentru nodul de tip 1. Astfel P1 este identica cu P9, P2 cu P10, P3 cu P11, P4 cu P12, P5 cu P13, P6 cu

P14, P7 cu P15 si P8 cu P16. Trebuie facuta observatia ca P1, P2, P3, P4, P5, P6, P7 si P8 corespund stoperului 1, iar P9, P10, P11, P12, P13, P14, P15 si P16 corespund stoperului 2. Starile P17 si P18 corespund starilor P9 si P10 de la nodul de tip 1, iar starea P19 este stare de sinconizare nou introdusa fata de nodul de tip 1.

Tranziţiile *t* sunt similare cu cele prezentate la nodul 1. Astfel t1 este similar cu t13, t2 cu t14, t3 cu t15, t4 cu t16, t5 cu t17, t6 cu t18, t7 cu t19, t8 cu t20, t9 cu t21, t10 cu t22, t11 cu t23, t12 cu t24. Similar cu observatia facuta in cazul pozitiilor, tranzitiile t1, t2, t3, t4, t5, t6, t7, t8, t9, t10, t11 si t12 corespund stoperului 1, iar t13, t14, t15, t16, t17, t18, t19, t20, t21, t22, t23 si t24 corespund stoperului 2. Tranzitile t25 si t26 corespund tranzitilor t9 si t10 de la nodul de tip 1.

Topologia retelei Petri validata de PNT este prezentata in figura 2.46.

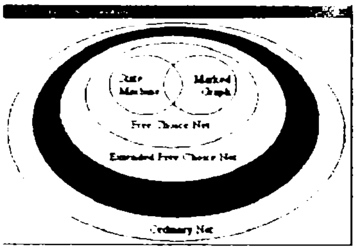


Figura 2.46. Topologia retelei Petri validata de TPN

Reteaua Petri considerata este formalizata matematic prin cvintuplul:

$$PN\_PT\_N2 = (P,T,F,W,M_0)$$

unde:

- $P = \{p_1, p_2, p_3, p_4, p_5, p_6, p_7, p_8, p_9, p_{10}, p_{11}, p_{12}, p_{13}, p_{14}, p_{15}, p_{16}, p_{17}, p_{18}, p_{19}\}$
- $T = \{t_1, t_2, t_3, t_4, t_5, t_6, t_7, t_8, t_9, t_{10}, t_{11}, t_{12}, t_{13}, t_{14}, t_{15}, t_{16}, t_{17}, t_{18}, t_{19}, t_{20}, t_{21}, t_{22}, t_{23}, t_{24}, t_{25}, t_{26}\}$
- $F = \{(p_1,t_1),(p_2,t_2),(p_2,t_3),(p_3,t_4),(p_4,t_5),(p_5,t_6),(p_5,t_7),(p_6,t_{10}), (p_6,t_{11}),(p_7,t_8),(p_8,t_9),(p_8,t_{12}),(p_9,t_{13}),(p_{10},t_{14}),(p_{10},t_{15}), (p_{11},t_{16}),(p_{12},t_{17}),(p_{13},t_{18}),(p_{13},t_{19}),(p_{14},t_{22}),(p_{14},t_{23}),(p_{15},t_{20}), (p_{16},t_{21}),(p_{16},t_{24}),(p_{17},t_1),(p_{17},t_{13}),(p_{17},t_{25}),(p_{18},t_{26}), (p_{19},t_1),(p_{19},t_{13})\} \cup \{(t_1,p_2),(t_2,p_3),(t_3,p_4),(t_4,p_4),(t_5,p_5),(t_6,p_6),(t_7,p_7),(t_8,p_8), (t_9,p_6),(t_{10},p_7),(t_{11},p_1),(t_{11},p_{17}),(t_{11},p_{19}),(t_{12},p_1),(t_{12},p_{17}), (t_{12},p_{19}),(t_{13},p_{10}),(t_{14},p_{11}),(t_{15},p_{12}),(t_{16},p_{12}),(t_{17},p_{13}),(t_{18},p_{14}), (t_{19},p_{15}),(t_{20},p_{16}),(t_{21},p_{14}),(t_{22},p_{15}),(t_{23},p_9),(t_{23},p_{17}), (t_{23},p_{19}),(t_{24},p_9),(t_{24},p_{17}),(t_{24},p_{19}),(t_{25},p_{18}),(t_{26},p_{17})\}$

- $W(p_1,t_1)=1, W(p_2,t_2)=1, W(p_2,t_3)=1, W(p_3,t_4)=1, W(p_4,t_5)=1, W(p_5,t_6)=1,$
  $W(p_5,t_7)=1, W(p_6,t_{10})=1, W(p_6,t_{11})=1, W(p_7,t_8)=1, W(p_8,t_9)=1, W(p_8,t_{12})=1,$
  $W(p_9,t_{13})=1, W(p_{10},t_{14})=1, W(p_{10},t_{15})=1, W(p_{11},t_{16})=1, W(p_{12},t_{17})=1, W(p_{13},t_{18})=1,$
  $W(p_{13},t_{19})=1, W(p_{14},t_{22})=1, W(p_{14},t_{23})=1, W(p_{15},t_{20})=1, W(p_{16},t_{21})=1, W(p_{16},t_{24})=1,$
  $W(p_{17},t_1)=1, W(p_{17},t_{13})=1, W(p_{17},t_{25})=1, W(p_{18},t_{26})=1, W(p_{19},t_1)=1, W(p_{19},t_{13})=1,$
  $W(t_1,p_2)=1, W(t_2,p_3)=1, W(t_3,p_4)=1, W(t_4,p_4)=1, W(t_5,p_5)=1, W(t_6,p_6)=1,$
  $W(t_7,p_7)=1, W(t_8,p_8)=1, W(t_9,p_6)=1, W(t_{10},p_7)=1, W(t_{11},p_1)=1, W(t_{11},p_{17})=1,$
  $W(t_{11},p_{19})=1, W(t_{12},p_1)=1, W(t_{12},p_{17})=1, W(t_{12},p_{19})=1, W(t_{13},p_{10})=1, W(t_{14},p_{11})=1,$
  $W(t_{15},p_{12})=1, W(t_{16},p_{12})=1, W(t_{17},p_{13})=1, W(t_{18},p_{14})=1, W(t_{19},p_{15})=1, W(t_{20},p_{16})=1,$
  $W(t_{21},p_{14})=1, W(t_{22},p_{15})=1, W(t_{23},p_9)=1, W(t_{23},p_{17})=1, W(t_{23},p_{19})=1, W(t_{24},p_9)=1,$
  $W(t_{24},p_{17})=1, W(t_{24},p_{19})=1, W(t_{25},p_{18})=1, W(t_{26},p_{17})=1$
- $M_0 = [1,0,0,0,0,0,0,0,1,0,0,0,0,0,0,0,1,0,1]^T$

Matricile de incidenta corespunzatoare retelei Petri PN_PT_N2 sunt:

Matricea de incidenta de intrare:

$$A_{Ii} = \begin{pmatrix}
1000000000000000\ 101 \\
0100000000000000000 \\
0100000000000000000 \\
0010000000000000000 \\
0001000000000000000 \\
0000100000000000000 \\
0000100000000000000 \\
0000001000000000000 \\
0000000100000000000 \\
0000010000000000000 \\
0000010000000000000 \\
0000000100000000000 \\
0000000010000000\ 101 \\
0000000001000000000 \\
0000000001000000000 \\
0000000000100000000 \\
0000000000010000000 \\
0000000000001000000 \\
0000000000001000000 \\
0000000000000010000 \\
0000000000000001000 \\
0000000000000100000 \\
0000000000000100000 \\
0000000000000001000 \\
0000000000000000100 \\
0000000000000000010
\end{pmatrix}$$

Matricea de incidenta de iesire:

$$A_{OI} = \begin{pmatrix}
0100000000000000000 \\
0010000000000000000 \\
0001000000000000000 \\
0001000000000000000 \\
0000100000000000000 \\
0000010000000000000 \\
0000001000000000000 \\
0000000100000000000 \\
0000010000000000000 \\
0000001000000000000 \\
1000000000000000\ 101 \\
1000000000000000\ 101 \\
0000000001000000000 \\
0000000000100000000 \\
0000000000010000000 \\
0000000000010000000 \\
0000000000001000000 \\
0000000000000100000 \\
0000000000000010000 \\
0000000000000001000 \\
0000000000000100000 \\
0000000000000010000 \\
0000000010000000\ 101 \\
0000000010000000\ 101 \\
0000000000000000010 \\
0000000000000000100
\end{pmatrix}$$

Matricea de incidenta:

$$A_i = A_{Oi} - A_{Ii} = \begin{pmatrix}
-1 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & -1 & 0 & -1 \\
0 & -1 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & -1 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & -1 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & -1 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & -1 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & -1 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & -1 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 1 & 0 & -1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & -1 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
1 & 0 & 0 & 0 & 0 & -1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 1 \\
1 & 0 & 0 & 0 & 0 & 0 & 0 & -1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 1 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & -1 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & -1 & 0 & -1 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & -1 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & -1 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & -1 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & -1 & 1 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & -1 & 1 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & -1 & 0 & 1 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & -1 & 1 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & -1 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & -1 & 1 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & -1 & 0 & 0 & 1 & 0 & 1 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & -1 & 1 & 0 & 1 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & -1 & 1 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & -1 & 0 & 0
\end{pmatrix}$$

Arborele de acoperire corespunzator retelei Petri *PN_PT_N2* este prezentat in figura 2.47.

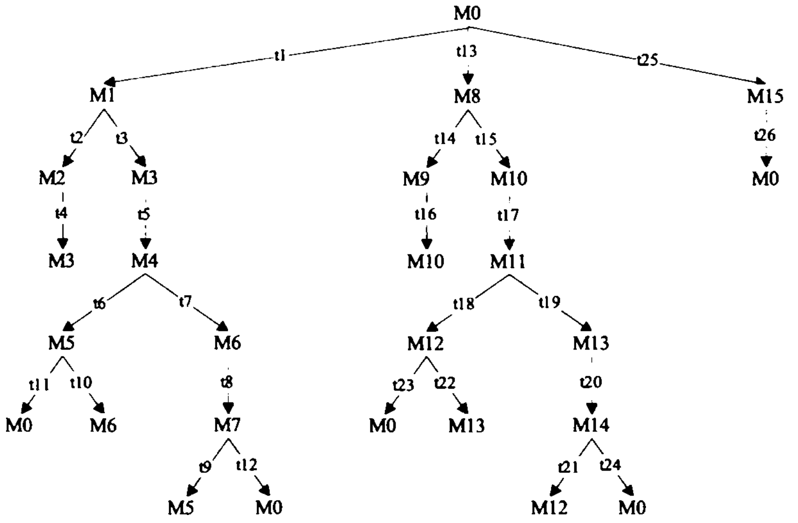


Figura 2.47. Arborele de acoperire corespunzator *PN_PT_N2*

Unde:
M0=(1,0,0,0,0,0,0,0,1,0,0,0,0,0,0,0,1,0,1);
M1=(0,1,0,0,0,0,0,0,1,0,0,0,0,0,0,0,0,0,0);

M2=(0,0,1,0,0,0,0,0,1,0,0,0,0,0,0,0,0,0,0);
M3=(0,0,0,1,0,0,0,0,1,0,0,0,0,0,0,0,0,0,0);
M4=(0,0,0,0,1,0,0,0,1,0,0,0,0,0,0,0,0,0,0);
M5=(0,0,0,0,0,1,0,0,1,0,0,0,0,0,0,0,0,0,0);
M6=(0,0,0,0,0,0,1,0,1,0,0,0,0,0,0,0,0,0,0);
M7=(0,0,0,0,0,0,0,1,1,0,0,0,0,0,0,0,0,0,0);
M8=(1,0,0,0,0,0,0,0,0,1,0,0,0,0,0,0,0,0,0);
M9=(1,0,0,0,0,0,0,0,0,0,1,0,0,0,0,0,0,0,0);
M10=(1,0,0,0,0,0,0,0,0,0,0,1,0,0,0,0,0,0,0);
M11=(1,0,0,0,0,0,0,0,0,0,0,0,1,0,0,0,0,0,0);
M12=(1,0,0,0,0,0,0,0,0,0,0,0,0,1,0,0,0,0,0);
M13=(1,0,0,0,0,0,0,0,0,0,0,0,0,0,1,0,0,0,0);
M14=(1,0,0,0,0,0,0,0,0,0,0,0,0,0,0,1,0,0,0);
M15=(1,0,0,0,0,0,0,0,1,0,0,0,0,0,0,0,0,1,1)

Studiind proprietatile comportamentale ale retelei Petri *PN_PT_N2*, pe baza arborelui de acoperire rezulta:

- reteaua este *marginita* (simbolul ω nu apare in arborele de acoperire);
- reteaua este *sigura* (marcajele din toate nodurile arborelui de acoperire contin numai „0" si „1");
- reteaua *nu este blocanta* (toate tranzitile au asociate arce in arborele de acoperire);
- reteaua este *accesibila* (orice marcaj poate fi atins pornind din M_0).

Graful de acoperire corespunzator este prezentat in figura 2.48.

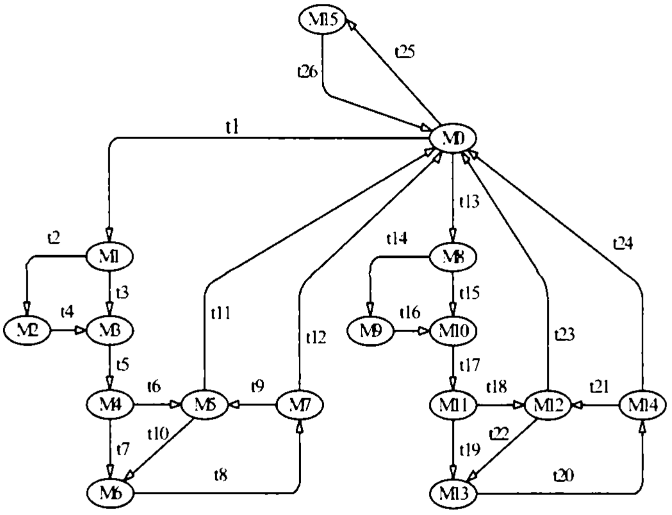


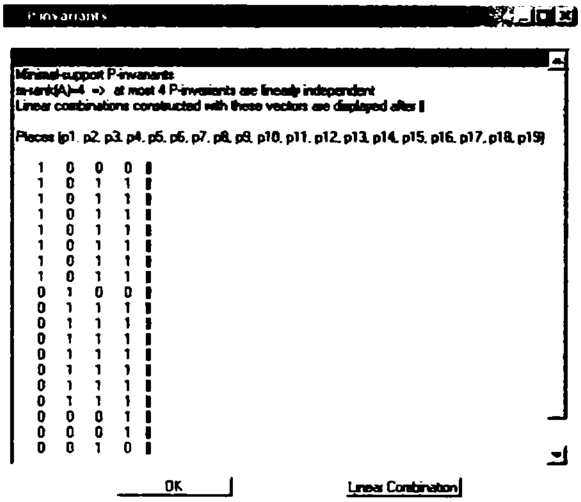
Figura 2.48. Graful de acoperire al retelei Petri *PN_PT_N2*

Structura rezultata pentru graful de acoperire confirma ca modelul de retea Petri corespunzator nodului de tip 1 este *accesibil* , modelul fiind *viabil* din punct de vedere comportamental.

Analiza retelei Petri *PN_PT_N2* din punct de vedere structural se face pe baza analizei existentei invariantilor de tip *P* respectiv *T*.

Conform teoremei 2.3 (determinarea numarului de invarianti), reteaua Petri *PN_PT_N2* este acoperita de 4 invarianti de tip, respectiv 11 invarianti de tip *T*, rangul matricei A fiind: $rang\,A = 15$.

In figurile 2.49 si 2.50 se prezinta invariantii de tip *P* respectiv *T* obtinuti in urma simularii retelei Petri *PN_PT_N2* cu ajutorul PNT-ului.



```
Minimal-support P-invariants
n-rank(A)=4  =>  at most 4 P-invariants are linearly independent
Linear combinations constructed with these vectors are displayed after |

Places [p1, p2, p3, p4, p5, p6, p7, p8, p9, p10, p11, p12, p13, p14, p15, p16, p17, p18, p19]

   1   0   0   0  |
   1   0   1   1  |
   1   0   1   1  |
   1   0   1   1  |
   1   0   1   1  |
   1   0   1   1  |
   1   0   1   1  |
   1   0   1   1  |
   0   1   0   0  |
   0   1   1   1  |
   0   1   1   1  |
   0   1   1   1  |
   0   1   1   1  |
   0   1   1   1  |
   0   1   1   1  |
   0   1   1   1  |
   0   0   0   1  |
   0   0   0   1  |
   0   0   1   0  |
```

Figura 2.49. Invariantii de tip P obtinuti cu ajutorul PNT

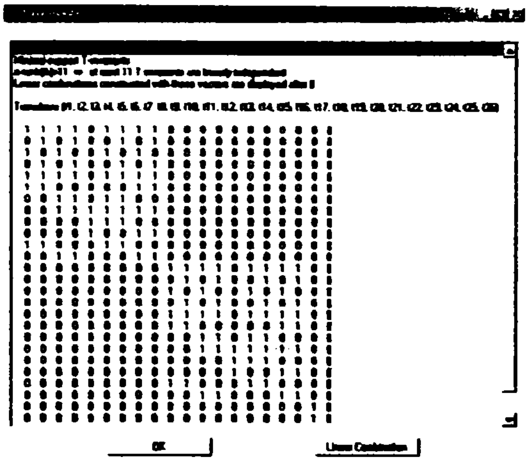


Figura 2.50 Invariantii de tip T obtinuti cu ajutorul PNT

Conform teroemelor 2.6 respectiv 2.7 reteaua Petri *PN_PT_N2* este conservativa si structural marginita, respectiv este consistenta si repetitiva.

În final se poate *concluziona* că structura aleasă pentru reteaua Petri corespunzătoare nodului de tip 2 este viabilă, ea putând fi implementată, fiind siguri că nu există condiţii de apariţie a blocajelor sau a apariţiei de situaţii necontrolabile.

2.5.2.2.2.2. Cazul: retea Petri temporizata P

Structura retelei Petri temporizata P corespunzatoare Nodului de tip 2 este aceeasi ca cea prezentata in figura 2.45.

Datorita faptului ca retelele sunt identice din punct de vedere structural si comportamental, rezultatele analizelor nu se mai detaliaza ele fiind in fapt identice cu cele obtinute in cazul nodului de tip 2 netemporizat.

Fata de situatia prezentata anterior fiecarei pozitii P i-a fost alocata o unitate de timp corespunzatoare. In acest mod situatia devine reala din punct de vedere al functionarii.

Timpii corespunzatori pozitiilor P sunt prezentati in tabelul 2.8.

Tabelul 2.8. *Timpii corespunzatori pozitilor P*

Pozitie	Unitati de Timp
P1, P9	1
P2, P10	5
P3, P11	4
P4, P12	4
P5, P13	7
P6, P14	6
P7, P15	3
P8, P16	5
P17	1
P18	3
P19	1

Reteaua Petri temporizata P pentru Nodul de tip 2 este in consecinta:

$$PN\_PTtime\_N2 = (P, T, F, W, D, M_0)$$

unde:

- $P = \{p_1, p_2, p_3, p_4, p_5, p_6, p_7, p_8, p_9, p_{10}, p_{11}, p_{12}, p_{13}, p_{14}, p_{15}, p_{16}, p_{17}, p_{18}, p_{19}\}$

- $T = \{t_1, t_2, t_3, t_4, t_5, t_6, t_7, t_8, t_9, t_{10}, t_{11}, t_{12}, t_{13}, t_{14}, t_{15}, t_{16}, t_{17}, t_{18}, t_{19}, t_{20}, t_{21}, t_{22}, t_{23}, t_{24}, t_{25}, t_{26}\}$

- $$\begin{aligned} F = \{&(p_1,t_1),(p_2,t_2),(p_2,t_3),(p_3,t_4),(p_4,t_5),(p_5,t_6),(p_5,t_7),(p_6,t_{10}),\\ &(p_6,t_{11}),(p_7,t_8),(p_8,t_9),(p_8,t_{12}),(p_9,t_{13}),(p_{10},t_{14}),(p_{10},t_{15}),\\ &(p_{11},t_{16}),(p_{12},t_{17}),(p_{13},t_{18}),(p_{13},t_{19}),(p_{14},t_{22}),(p_{14},t_{23}),\\ &(p_{15},t_{20}),(p_{16},t_{21}),(p_{16},t_{24}),(p_{17},t_1),(p_{17},t_{13}),(p_{17},t_{25}),\\ &(p_{18},t_{26}),(p_{19},t_1),(p_{19},t_{13})\} \cup\\ \{&(t_1,p_2),(t_2,p_3),(t_3,p_4),(t_4,p_4),(t_5,p_5),(t_6,p_6),(t_7,p_7),(t_8,p_8),\\ &(t_9,p_6),(t_{10},p_7),(t_{11},p_1),(t_{11},p_{17}),(t_{11},p_{19}),(t_{12},p_1),(t_{12},p_{17}),\\ &(t_{12},p_{19}),(t_{13},p_{10}),(t_{14},p_{11}),(t_{15},p_{12}),(t_{16},p_{12}),(t_{17},p_{13}),(t_{18},p_{14}),\\ &(t_{19},p_{15}),(t_{20},p_{16}),(t_{21},p_{14}),(t_{22},p_{15}),(t_{23},p_9),(t_{23},p_{17}),\\ &(t_{23},p_{19}),(t_{24},p_9),(t_{24},p_{17}),(t_{24},p_{19}),(t_{25},p_{18}),(t_{26},p_{17})\} \end{aligned}$$

-

$$
\begin{aligned}
&W(p_1,t_1)=1, W(p_2,t_2)=1, W(p_2,t_3)=1, W(p_3,t_4)=1, W(p_4,t_5)=1, W(p_5,t_6)=1,\\
&W(p_5,t_7)=1, W(p_6,t_{10})=1, W(p_6,t_{11})=1, W(p_7,t_8)=1, W(p_8,t_9)=1, W(p_8,t_{12})=1,\\
&W(p_9,t_{13})=1, W(p_{10},t_{14})=1, W(p_{10},t_{15})=1, W(p_{11},t_{16})=1, W(p_{12},t_{17})=1,\\
&W(p_{13},t_{18})=1, W(p_{13},t_{19})=1, W(p_{14},t_{22})=1, W(p_{14},t_{23})=1, W(p_{15},t_{20})=1,\\
&W(p_{16},t_{21})=1, W(p_{16},t_{24})=1, W(p_{17},t_1)=1, W(p_{17},t_{13})=1, W(p_{17},t_{25})=1,\\
&W(p_{18},t_{26})=1, W(p_{19},t_1)=1, W(p_{19},t_{13})=1, W(t_1,p_2)=1, W(t_2,p_3)=1,\\
&W(t_3,p_4)=1, W(t_4,p_4)=1, W(t_5,p_5)=1, W(t_6,p_6)=1, W(t_7,p_7)=1,\\
&W(t_8,p_8)=1, W(t_9,p_6)=1, W(t_{10},p_7)=1, W(t_{11},p_1)=1, W(t_{11},p_{17})=1,\\
&W(t_{11},p_{19})=1, W(t_{12},p_1)=1, W(t_{12},p_{17})=1, W(t_{12},p_{19})=1, W(t_{13},p_{10})=1,\\
&W(t_{14},p_{11})=1, W(t_{15},p_{12})=1, W(t_{16},p_{12})=1, W(t_{17},p_{13})=1, W(t_{18},p_{14})=1,\\
&W(t_{19},p_{15})=1, W(t_{20},p_{16})=1, W(t_{21},p_{14})=1, W(t_{22},p_{15})=1, W(t_{23},p_9)=1,\\
&W(t_{23},p_{17})=1, W(t_{23},p_{19})=1, W(t_{24},p_9)=1,\\
&W(t_{24},p_{17})=1, W(t_{24},p_{19})=1, W(t_{25},p_{18})=1, W(t_{26},p_{17})=1
\end{aligned}
$$

- $D=\{d(p_1)=1, d(p_2)=5, d(p_3)=4, d(p_4)=4, d(p_5)=7, d(p_6)=6, d(p_7)=3,$ $d(p_8)=5, d(p_9)=1, d(p_{10})=5, d(p_{11})=4, d(p_{12})=4, d(p_{13})=7,$ $d(p_{14})=6, d(p_{15})=3, d(p_{16})=5, d(p_{17})=1, d(p_{18})=3, d(p_{19})=1\}$
- $M_0=[1,0,0,0,0,0,0,0,0,0,1,0,0,0,0,0,1,0,1]^T$

În final se poate *concluziona* că structura aleasă pentru reteaua Petri temporizata P corespunzătoare nodului de tip 2 este viabilă, ea putând fi implementată, fiind siguri că nu există condiţii de apariţie a blocajelor sau a apariţiei de situaţii necontrolabile.

2.5.2.3. Modelarea nodului de tip 3 – trei intrari o iesire [UP06a][UP06b] [Ung06b][Ung06c]

După cum s-a aratat in capitolul 1, nodul de tip 3 (trei intrări o ieşire) este construit cu ajutorul nodului de tip 1, având în plus un macaz care este comandat direct în funcţie de tranziţia care este executată.

2.5.2.3.1. Modelarea nodului de tip 3 cu ajutorul automatelor [UP06a][Ung06b]

Structura automatului secvential considerat pentru modelarea nodului de tip 3 este prezentata în figura 2.51.

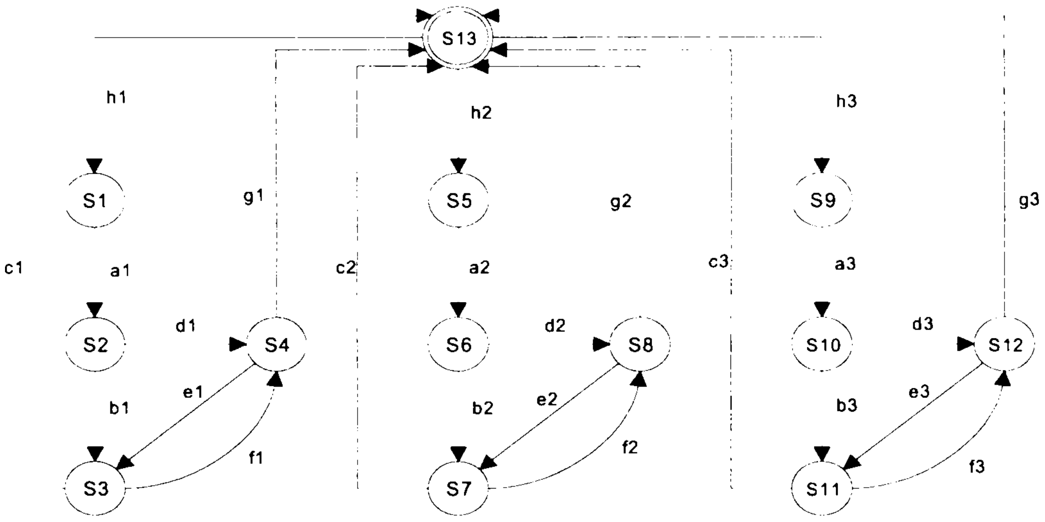


Figura 2.51. Structura automatului secvential corespunzător nodului de tip 3

Dupa cum se observa automatul secvential corespunzator nodului 3 este realizat prin sincronizarea a trei automate secventiale *AS_N1*, cate unul pentru fiecare linie de intrare (stoper). Semnificatiile starilor sunt identice cu cele prezentate la *AS_N1*, cu observatia ca S1, S2, S3 si S4 corespund stoperului 1, S5, S6, S7 si S8 corespund stoperului 2 si S9, S10, S11, S12 corespund stoperului 3. Starea S13 este starea suplimentara de sincronizare a automatelor secventiale corespunzatoare stoperelor 1, 2 si 3.

Evenimentele a1, b1, c1, d1, e1, f1, g1 se refera la stoperul 1 (senzorii corespunzatori stoperului 1 conform relatilor prezentate la *AS_N1*), a2, b2, c2, d2, e2, f2, g2 se refera la stoperul 2, iar a3, b3, c3, d3, e3, f3, g3 se refera la stoperul 3 cu aceeasi observatie ca cea facuta la stoperul 1. Evenimentele h1, h2 si h3 reprezinta conditiile de tranzitie din starea S13 in S1 sau S5 sau S9 (h1 activ - un carucior a activat senzorul de intrare din stoperul 1, h2 activ - un carucior a activat senzorul de intrare din stoperul 2 respectiv h3 activ - un carucior a activat senzorul de intrare din stoperul 3).

Automatul *AS_N3*, corespunzator nodului de tip 3, este definit in felul urmator:

- multimea starilor:
  $X = \{S1, S2, S3, S4, S5, S6, S7, S8, S9, S10, S11, S12, S13\}$
- multimea evenimentelor:
  $\Sigma = \{a1, b1, c1, d1, e1, f1, g1, h1, a2, b2, c2, d2, e2, f2, g2, h2, a3, b3, c3, d3, e3, f3, g3, h3\}$
- multimile evenimentelor posibile si functiile de tranzitie a starilor:

$\Gamma(S1) = \{a1\}$ $\delta(S1, a1) = S2$

$\Gamma(S2) = \{b1, d1\}$ $\delta(S2, b1) = S3,$ $\delta(S2, d1) = S4$

$\Gamma(S3) = \{c1, f1\}$ $\delta(S3, c1) = S13,$ $\delta(S3, f1) = S4$

$\Gamma(S4) = \{e1, g1\}$ $\delta(S4, e1) = S3$ $\delta(S4, g1) = S13$

$\Gamma(S5)=\{a2\}$ $\delta(S5,a2)=S6$

$\Gamma(S6)=\{b2,d2\}$ $\delta(S6,b2)=S7,$ $\delta(S6,d2)=S8$

$\Gamma(S7)=\{c2,f2\}$ $\delta(S7,c2)=S13,$ $\delta(S7,f2)=S8$

$\Gamma(S8)=\{e2,g2\}$ $\delta(S8,e2)=S7$ $\delta(S8,g2)=S13$

$\Gamma(S9)=\{a3\}$ $\delta(S9,a3)=S10$

$\Gamma(S10)=\{b3,d3\}$ $\delta(S10,b3)=S11,$ $\delta(S10,d3)=S12$

$\Gamma(S11)=\{c3,f3\}$ $\delta(S11,c3)=S13,$ $\delta(S11,f3)=S12$

$\Gamma(S12)=\{e3,g3\}$ $\delta(S12,e3)=S11$ $\delta(S12,g3)=S13$

$\Gamma(S13)=\{h1,h2,h3\}$ $\delta(S13,h1)=S1,$ $\delta(S13,h2)=S5,$ $\delta(S13,h3)=S9$

- starea initiala: $x_0=S13$.

Considerentele legate de transformarea modelului de tip automat in model de tip retea Petri, prezentate la nodul de tip 1 sunt valabile si in cazul nodului de tip 3.

Pentru conversia din model de tip automat in model de tip retea Petri s-a utilizat acceasi metoda ca si in cazul nodului de tip 1 .

Aplicand algoritmul 2.1 rezulta:

- multimea pozitiilor: $P=\{p1,p2,p3,p4,p5,p6,p7,p8,p9,p10,p11,p12,p13\}$, cu $p1=\{S1\},p2=\{S2\},p3=\{S3\},p4=\{S4\},p5=\{S5\},p6=\{S6\},p7=\{S7\},$ $p8=\{S8\},p9=\{S9\},p10=\{S10\},p11=\{S11\},p12=\{S12\}$ si $p13=\{S13\}$;
- multimea trranzitiilor:

• Pentru *p1*	$(p1,p2)$	$p2=\delta(p1,t1)$	$t1=\{a1\}$
• Pentru *p2*	$(p2,p3)$	$p3=\delta(p2,t2)$	$t2=\{b1\}$
	$(p2,p4)$	$p4=\delta(p2,t4)$	$t4=\{d1\}$
• Pentru *p3*	$(p3,p13)$	$p13=\delta(p3,t3)$	$t3=\{c1\}$
	$(p3,p4)$	$p4=\delta(p3,t6)$	$t6=\{f1\}$
• Pentru *p4*	$(p4,p13)$	$p13=\delta(p4,t7)$	$t7=\{g1\}$
	$(p4,p3)$	$p3=\delta(p4,t5)$	$t5=\{e1\}$
• Pentru *p5*	$(p5,p6)$	$p6=\delta(p5,t8)$	$t8=\{a2\}$
• Pentru *p6*	$(p6,p7)$	$p7=\delta(p6,t9)$	$t9=\{b2\}$
	$(p6,p8)$	$p8=\delta(p6,t11)$	$t11=\{d2\}$
• Pentru *p7*	$(p7,p13)$	$p13=\delta(p7,t10)$	$t10=\{c2\}$
	$(p7,p8)$	$p8=\delta(p7,t13)$	$t13=\{f2\}$
• Pentru *p8*	$(p8,p13)$	$p13=\delta(p8,t14)$	$t14=\{g2\}$
	$(p8,p7)$	$p7=\delta(p8,t12)$	$t12=\{e2\}$
• Pentru *p9*	$(p9,p10)$	$p10=\delta(p9,t17)$	$t17=\{a3\}$
• Pentru *p10*	$(p10,p11)$	$p11=\delta(p10,t18)$	$t18=\{b3\}$
	$(p10,p12)$	$p12=\delta(p10,t20)$	$t20=\{d3\}$
• Pentru *p11*	$(p11,p13)$	$p13=\delta(p11,t19)$	$t19=\{c3\}$
	$(p11,p12)$	$p12=\delta(p11,t22)$	$t22=\{f3\}$
• Pentru *p12*	$(p12,p13)$	$p13=\delta(p12,t23)$	$t23=\{g3\}$
	$(p12,p11)$	$p11=\delta(p12,t21)$	$t21=\{e3\}$
• Pentru *p13*	$(p13,p1)$	$p1=\delta(p13,t15)$	$t15=\{h1\}$
	$(p13,p5)$	$p5=\delta(p13,t16)$	$t16=\{h2\}$
	$(p13,p9)$	$p9=\delta(p13,t24)$	$t24=\{h3\}$

In figura 2.52 se prezinta structura retelei Petri asociata automatului secvential considerat in cazul nodului de tip 3.

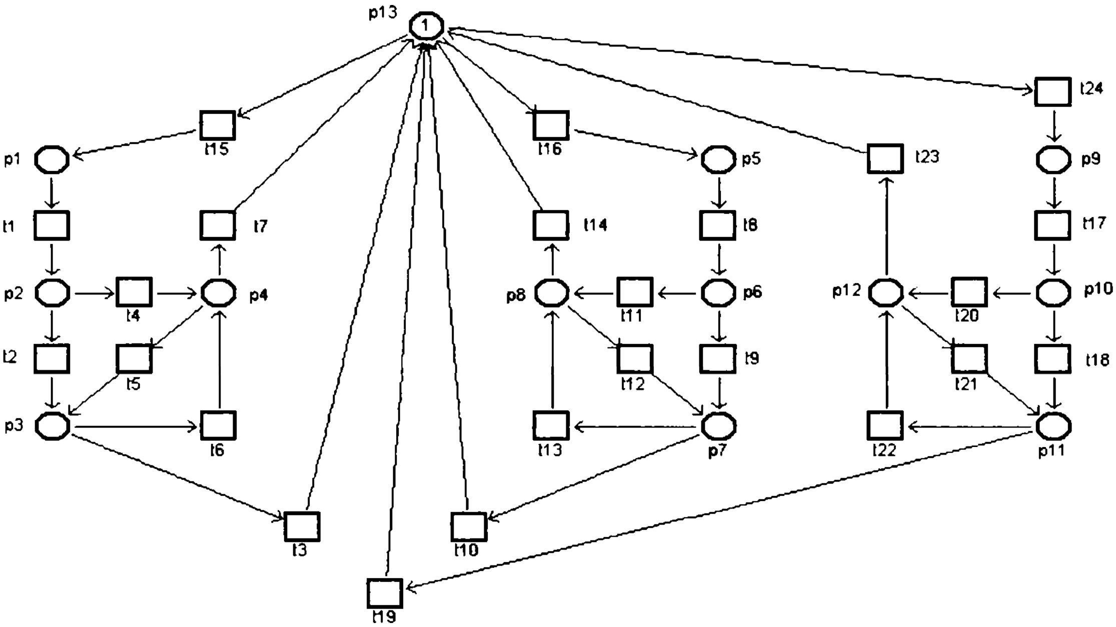


Figura 2.52. Structura reţelei Petri asociata automatului secvential corespunzător nodului de tip 3

Semnificaţiile pozitiilor este similară cu prezentarea făcută pentru nodul de tip 1. Astfel, P1,P5 si P9 sunt similare cu P1 de la nodul de tip 1, P2, P6 si P10 sunt similare cu P2 de la nodul de tip 1, P3, P7 si P11 sunt similare cu P3 de la nodul de tip 1, iar P4, P8 si P12 sunt similare cu P4 de la nodul de tip 1. În plus, pentru asigurarea excluderii mutuale, s-a introdus o stare suplimentară P13.

Semnificaţiile tranziţiilor sunt similare cu cele prezentate la nodul de tip 1. Astfel, t1, t8 si 17 sunt similare cu t1 de la nodul de tip 1, t2, t9 si t18 sunt similare cu t2 de la nodul de tip 1, t3, t10 si t19 sunt similare cu t3 de la nodul de tip 1, cu observaţia că apare o condiţie în plus dată de transferul jetonului în P13, t4, t11 si t20 sunt similare cu t4 de la nodul de tip 1, t5, t12 si t21 sunt similare cu t5 de la nodul de tip 1, t6, t13 si t22 sunt similare cu t6 de la nodul de tip 1, t7, t14 si t23 sunt similare cu t7 de la nodul de tip 1, cu observaţia că apare o condiţie în plus dată de comutarea in starea P13 nu in P1. Tranzitiile t15, t16 si t24 sunt noi, ele asigurand comutarea fie in starea P1, fie in starea P5, fie in starea P9, functie de pozitia in nod.

Topologia retelei validata de PNT este prezentata in figura 2.53.

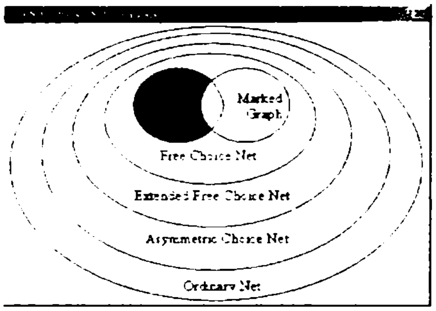


Figura 2.53. Topologia retelei Petri echivalenta validata de PNT

Rezultatele obtinute confirma, ca si in cazurile nodurilor de tip 1 si 2, ca dupa efectuarea conversiei topologia retelei Petri obtinuta este tot de tip automat („State machine").

Reteaua Petri considerata este formalizata matematic prin cvintuplul:

$$PN\_AS\_N3 = (P,T,F,W,M_0)$$

unde:

- $P = \{p_1, p_2, p_3, p_4, p_5, p_6, p_7, p_8, p_9, p_{10}, p_{11}, p_{12}, p_{13}\}$
- $T = \{t_1,t_2,t_3,t_4,t_5,t_6,t_7,t_8,t_9,t_{10},t_{11},t_{12},t_{13},t_{14},t_{15},t_{16},t_{17},$
  $t_{18},t_{19},t_{20},t_{21},t_{22},t_{23},t_{24}\}$
- $F = \{(p_1,t_1),(p_2,t_2),(p_2,t_4),(p_3,t_6),(p_3,t_3),(p_4,t_5),(p_4,t_7),$
  $(p_5,t_8),(p_6,t_9),(p_6,t_{11}),(p_7,t_{10}),(p_7,t_{13}),(p_8,t_{12}),(p_8,t_{14}),$
  $(p_9,t_{17}),(p_{10},t_{18}),(p_{10},t_{20}),(p_{11},t_{19}),(p_{11},t_{22})$
  $(p_{12},t_{21}),(p_{12},t_{23}),(p_{13},t_{15}),(p_{13},t_{16}),(p_{13},t_{24})\} \cup$
  $\{(t_1,p_2),(t_2,p_3),(t_3,p_{13}),(t_4,p_4),(t_5,p_3),(t_6,p_4),(t_7,p_{13}),$
  $(t_8,p_6),(t_9,p_7),(t_{10},p_{13}),(t_{11},p_8),(t_{12},p_7),(t_{13},p_8),$
  $(t_{14},p_{13}),(t_{15},p_1),(t_{16},p_5),(t_{17},p_{10}),(t_{18},p_{11}),(t_{19},p_{13})$
  $(t_{20},p_{12}),(t_{21},p_{11}),(t_{22},p_{12}),(t_{23},p_{13})(t_{24},p_9)\}$
- $W(p_1,t_1) = 1, W(p_2,t_2) = 1, W(p_2,t_4) = 1, W(p_3,t_6) = 1, W(p_3,t_3) = 1,$
  $W(p_4,t_5) = 1, W(p_4,t_7) = 1, W(p_5,t_8) = 1, W(p_6,t_9) = 1, W(p_6,t_{11}) = 1,$
  $W(p_7,t_{10}) = 1, W(p_7,t_{13}) = 1, W(p_8,t_{12}) = 1, W(p_8,t_{14}) = 1, W(p_9,t_{17}) = 1,$
  $W(p_{10},t_{18}) = 1, W(p_{10},t_{20}) = 1, W(p_{11},t_{19}) = 1, W(p_{11},t_{22}) = 1,$
  $W(p_{12},t_{21}) = 1, W(p_{12},t_{23}) = 1, W(p_{13},t_{15}) = 1, W(p_{13},t_{16}) = 1,$
  $W(p_{13},t_{24}) = 1, W(t_1,p_2) = 1, W(t_2,p_3) = 1, W(t_3,p_{13}) = 1,$
  $W(t_4,p_4) = 1, W(t_5,p_3) = 1, W(t_6,p_4) = 1, W(t_7,p_{13}) = 1,$
  $W(t_8,p_6) = 1, W(t_9,p_7) = 1, W(t_{10},p_{13}) = 1, W(t_{11},p_8) = 1,$
  $W(t_{12},p_7) = 1, W(t_{13},p_8) = 1, W(t_{14},p_{13}) = 1, W(t_{15},p_1) = 1,$
  $W(t_{16},p_5) = 1, W(t_{17},p_{10}) = 1, W(t_{18},p_{11}) = 1,$
  $W(t_{19},p_{13}) = 1, W(t_{20},p_{12}) = 1, W(t_{21},p_{11}) = 1,$
  $W(t_{22},p_{12}) = 1, W(t_{23},p_{13}) = 1, W(t_{24},p_9) = 1$
- $M_0 = [0,0,0,0,0,0,0,0,0,0,0,0,1]^T$

Matricile de incidenţă corespunzătoare sunt.

Matricea de incidenta de intrare:

$$A_{Ii} = \begin{pmatrix} 1\ 000000000000 \\ 0\ 100000000000 \\ 00\ 10000000000 \\ 0\ 100000000000 \\ 000\ 1000000000 \\ 00\ 10000000000 \\ 000\ 1000000000 \\ 000000000000\ 1 \\ 00001\ 00000000 \\ 00000\ 10000000 \\ 000000\ 1000000 \\ 00000\ 10000000 \\ 0000000\ 100000 \\ 000000\ 1000000 \\ 0000000\ 100000 \\ 000000000000\ 1 \\ 000000001\ 0000 \\ 000000000\ 1000 \\ 0000000000\ 100 \\ 000000000\ 1000 \\ 00000000000\ 10 \\ 0000000000\ 100 \\ 00000000000\ 10 \\ 000000000000\ 1 \end{pmatrix}$$

Matricea de incidenta de iesire:

$$A_{Oi} = \begin{pmatrix} 0100000000000 \\ 0010000000000 \\ 0000000000001 \\ 0001000000000 \\ 0010000000000 \\ 0001000000000 \\ 0000000000001 \\ 1000000000000 \\ 0000010000000 \\ 0000001000000 \\ 0000000000001 \\ 0000000100000 \\ 0000001000000 \\ 0000000100000 \\ 0000000000001 \\ 0000\ 100000000 \\ 0000000001000 \\ 0000000000100 \\ 0000000000001 \\ 0000000000010 \\ 0000000000100 \\ 0000000000010 \\ 0000000000001 \\ 00000000\ 10000 \end{pmatrix}$$

Matricea de incidenta:

$$A_i = A_{Oi} - A_{Ii} = \begin{pmatrix} -1 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & -1 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & -1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 \\ 0 & -1 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 1 & -1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & -1 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & -1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 \\ 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & -1 \\ 0 & 0 & 0 & 0 & -1 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & -1 & 1 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & -1 & 0 & 0 & 0 & 0 & 0 & 1 \\ 0 & 0 & 0 & 0 & 0 & -1 & 0 & 1 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 1 & -1 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & -1 & 1 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & -1 & 0 & 0 & 0 & 0 & 1 \\ 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & -1 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & -1 & 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & -1 & 1 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & -1 & 0 & 1 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & -1 & 0 & 1 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & -1 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & -1 & 1 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & -1 & 1 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & -1 \end{pmatrix}$$

Arborele de acoperire corespunzator este prezentat in figura 2.54.

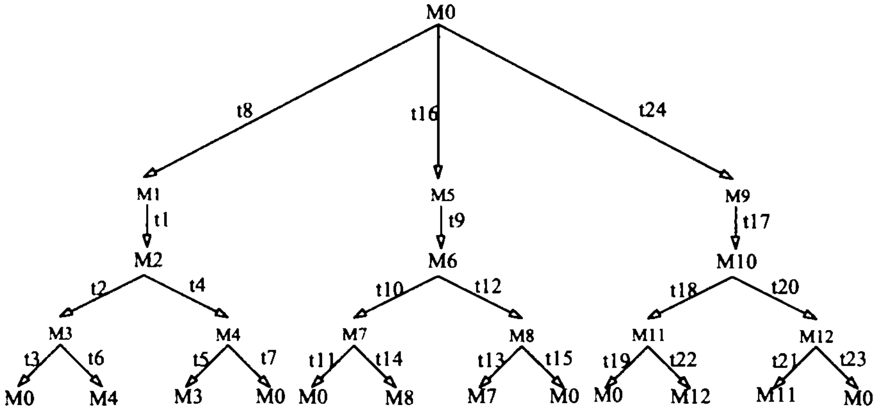


Figura 2.54 Arborele de acoperire al retelei Petri *PN_AS_N3*

Unde:
M0=(0,0,0,0,0,0,0,0,0,0,0,0,1); M1=(1,0,0,0,0,0,0,0,0,0,0,0,0);
M2=(0,1,0,0,0,0,0,0,0,0,0,0,0); M3=(0,0,1,0,0,0,0,0,0,0,0,0,0);
M4=(0,0,0,1,0,0,0,0,0,0,0,0,0); M5=(0,0,0,0,1,0,0,0,0,0,0,0,0);
M6=(0,0,0,0,0,1,0,0,0,0,0,0,0); M7=(0,0,0,0,0,0,1,0,0,0,0,0,0);
M8=(0,0,0,0,0,0,0,1,0,0,0,0,0); M9=(0,0,0,0,0,0,0,0,1,0,0,0,0);
M10=(0,0,0,0,0,0,0,0,0,1,0,0,0); M11=(0,0,0,0,0,0,0,0,0,0,1,0,0);
M12=(0,0,0,0,0,0,0,0,0,0,0,1,0)

Studiind proprietatile comportamentale ale retelei Petri *PN_AS_N3*, pe baza arborelui de acoperire rezulta:

- reteaua este *marginita* (simbolul ω nu apare in arborele de acoperire);
- reteaua este *sigura* (marcajele din toate nodurile arborelui de acoperire contin numai „0" si „1");
- reteaua *nu este blocanta* (toate tranzitile au asociate arce in arborele de acoperire);
- reteaua este *accesibila* (orice marcaj poate fi atins pornind din M_0).

Graful de acoperire corespunzator este prezentat in figura 2.55.

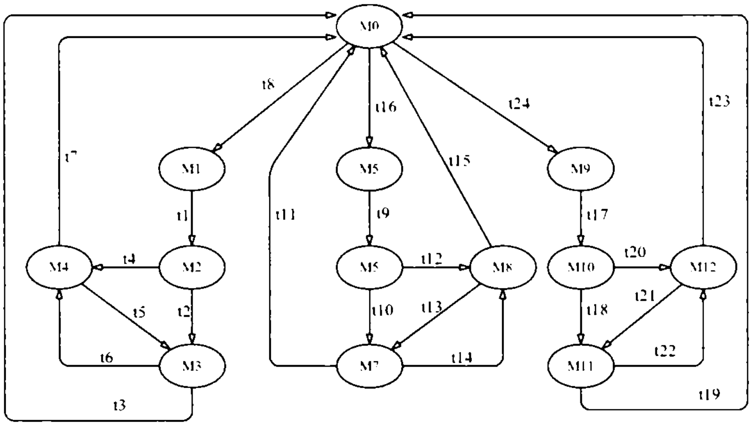


Figura 2.55. Graful de acoperire al retelei Petri *PN_AS_N3*

Structura rezultata pentru graful de acoperire confirma ca modelul de retea Petri derivata din modelul de tip automat corespunzator nodului de tip 3 este *acesibila*.

Concluzionand, se poate afirma ca modelul ales este *viabil* din punct de vedere comportamental.

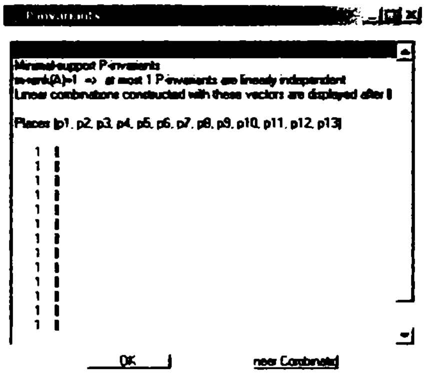
Analiza retelei Petri *PN_AS_N3* din punct de vedere structural se face pe baza analizei existentei invariantilor de tip *P* respectiv *T*.

Pentru determinarea numarului de invarianti *P* respectiv *T* s-a aplicat teorema 2.3 (Determinarea numarului de invarianti).

Rangul matricei A este: $rang A = 12$.

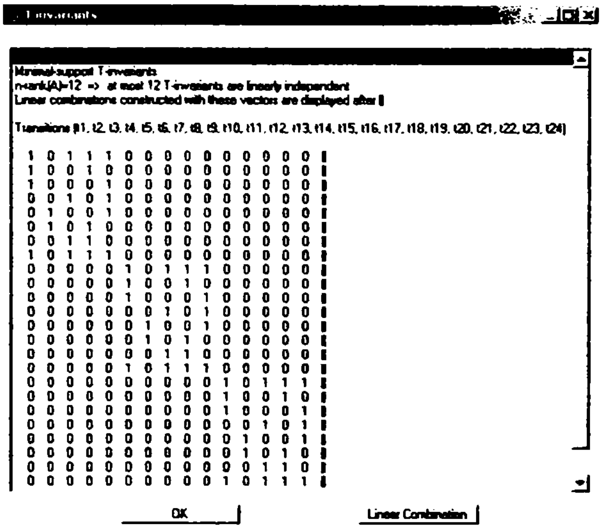
Rezulta ca reteaua Petri *PN_AS_N3* este acoperita de 1 invariant de tip *P,* respectiv 12 invarianti de tip *T.*

In figurile 2.56 si 2.57 se prezinta invariantii de tip P respectiv T obtinuti in urma simularii retelei Petri *PN_AS_N3* cu ajutorul PNT-ului.



```
Minimal-support P-invariants
m-rank(A)=1  =>  at most 1 P-invariants are linearly independent
Linear combinations constructed with these vectors are displayed after |

Places [p1, p2, p3, p4, p5, p6, p7, p8, p9, p10, p11, p12, p13]

    1  |
    1  |
    1  |
    1  |
    1  |
    1  |
    1  |
    1  |
    1  |
    1  |
    1  |
    1  |
    1  |
```

Figura 2.56 Invariantii de tip P obtinuti cu ajutorul PNT



```
Minimal-support T-invariants
n-rank(A)=12  =>  at most 12 T-invariants are linearly independent
Linear combinations constructed with these vectors are displayed after |

Transitions [t1, t2, t3, t4, t5, t6, t7, t8, t9, t10, t11, t12, t13, t14, t15, t16, t17, t18, t19, t20, t21, t22, t23, t24]

  1  0  1  1  1  0  0  0  0  0  0  0  0  0  0  |
  1  0  0  1  0  0  0  0  0  0  0  0  0  0  0  |
  1  0  0  0  1  0  0  0  0  0  0  0  0  0  0  |
  0  0  1  0  1  0  0  0  0  0  0  0  0  0  0  |
  0  1  0  0  1  0  0  0  0  0  0  0  0  0  0  |
  0  1  0  1  0  0  0  0  0  0  0  0  0  0  0  |
  0  0  1  1  0  0  0  0  0  0  0  0  0  0  0  |
  1  0  1  1  1  0  0  0  0  0  0  0  0  0  0  |
  0  0  0  0  0  1  0  1  1  1  0  0  0  0  0  |
  0  0  0  0  0  1  0  0  1  0  0  0  0  0  0  |
  0  0  0  0  0  1  0  0  0  1  0  0  0  0  0  |
  0  0  0  0  0  0  0  1  0  1  0  0  0  0  0  |
  0  0  0  0  0  0  1  0  0  1  0  0  0  0  0  |
  0  0  0  0  0  0  1  0  1  0  0  0  0  0  0  |
  0  0  0  0  0  0  0  1  1  0  0  0  0  0  0  |
  0  0  0  0  0  1  0  1  1  1  0  0  0  0  0  |
  0  0  0  0  0  0  0  0  0  0  1  0  1  1  1  |
  0  0  0  0  0  0  0  0  0  0  1  0  0  1  0  |
  0  0  0  0  0  0  0  0  0  0  1  0  0  0  1  |
  0  0  0  0  0  0  0  0  0  0  0  0  1  0  1  |
  0  0  0  0  0  0  0  0  0  0  0  1  0  0  1  |
  0  0  0  0  0  0  0  0  0  0  0  1  0  1  0  |
  0  0  0  0  0  0  0  0  0  0  0  0  1  1  0  |
  0  0  0  0  0  0  0  0  0  0  1  0  1  1  1  |
```

Figura 2.57 Invariantii de tip T obtinuti cu ajutorul PNT

Conform teroemelor 2.6 respectiv 2.7 reteaua Petri PN_AS_N3 este *conservativa si structural marginita, respectiv este consistenta si repetitiva.*

În final se poate *concluziona* că structura aleasă pentru un automat secvential clasic corespunzător nodului de tip 3 este viabilă, ea putând fi implementată, fiind siguri că nu există condiţii de apariţie a blocajelor sau a apariţiei de situaţii necontrolabile.

2.5.2.3.2.1. Modelarea nodului de tip 3 cu ajutorul retelelor Petri

2.5.2.3.2.1.1. Cazul: retea Petri netemporizata [Ung06b] [Ung06c]

Structura retelei Petri considerate pentru modelarea nodului de tip 3 este prezentata în figura 2.58.

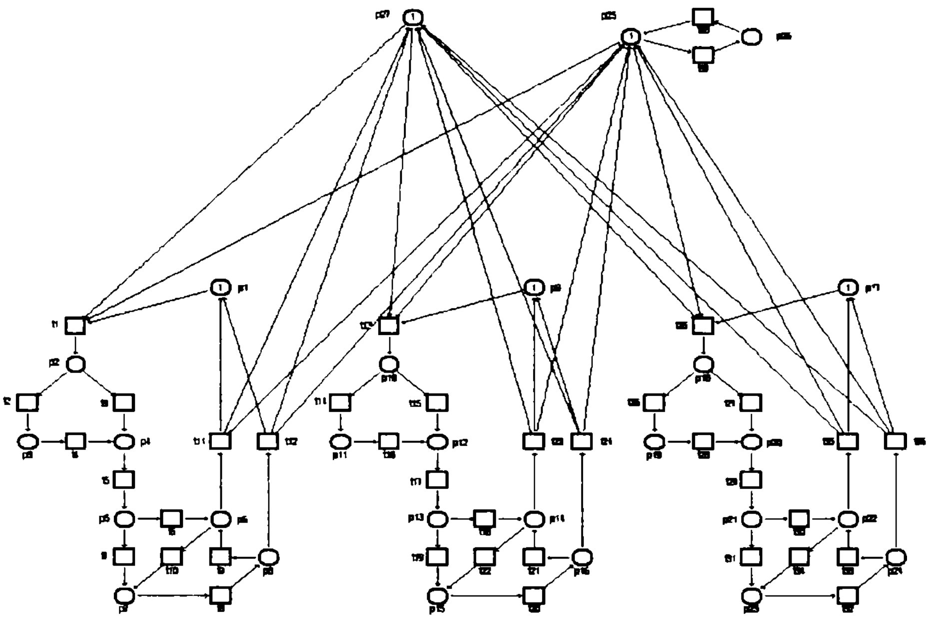


Figura 2.58. Structura retelei Petri corespunzătoare nodului de tip 3

Semnificaţiile pozitiilor *P* sunt identice cu cele prezentate pentru nodul de tip 2. Astfel P1 este identica cu P9 si P17, P2 cu P10 si P18, P3 cu P11 si P19, P4 cu P12 si P20, P5 cu P13 si P21, P6 cu P14 si P22, P7 cu P15 si P23, si P8 cu P16 si P24. Trebuie facuta observatia ca P1, P2, P3, P4, P5, P6, P7 si P8 corespund stoperului 1, P9, P10, P11, P12, P13, P14, P15 si P16 corespund stoperului 2 si P17, P18, P19, P20, P21, P22, P23 si P24 corespund stoperului 3. Starile P25 si P26 corespund starilor P9 si P10 de la nodul de tip 1 iar starea P27 este stare de sinconizare nou introdusa fata de nodul de tip 1.

Tranziţiile *t* sunt similare cu cele prezentate la nodul 1. Astfel t1 este similar cu t13 si t25, t2 cu t14 si t26, t3 cu t15 si t27, t4 cu t16 si t28, t5 cu t17 si t29, t6 cu t18 si t30, t7 cu t19 si t31, t8 cu t20 si t32, t9 cu t21 si t33, t10 cu t22 si t34, t11 cu t23 si t35, t12 cu t24 si t36. Similar cu observatia facuta in cazul pozitiilor, tranzitiile t1, t2, t3, t4, t5, t6, t7, t8, t9, t10, t11 si t12 corespund stoperului 1, t13, t14, t15, t16, t17, t18, t19, t20, t21, t22, t23 si t24 corespund stoperului 2 si t25, t26, t27, t28, t29, t30, t31, t32, t33, t34, t35 si t36 corespund stoperului 3. Tranzitile t37 si t38 corespund tranzitilor t9 si t10 de la nodul de tip 1.

Topologia retelei Petri validata de PNT este prezentata in figura 2.59.

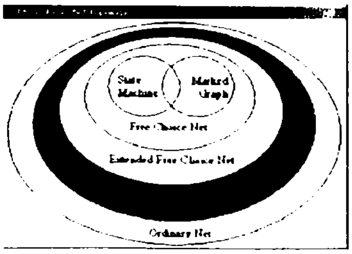


Figura 2.59. Topologia retelei Petri validata de PNT

Reteaua Petri considerata este formalizata prin cvintuplul:

$$PN\_PT\_N3 = (P,T,F,W,M_0)$$

unde:

- $P = \{p_1,p_2,p_3,p_4,p_5,p_6,p_7,p_8,p_9,p_{10},p_{11},p_{12},p_{13},p_{14},p_{15},p_{16},p_{17},p_{18},$
  $p_{19},p_{20},p_{21},p_{22},p_{23},p_{24},p_{25},p_{26},p_{27}\}$

- $T = \{t_1,t_2,t_3,t_4,t_5,t_6,t_7,t_8,t_9,t_{10},t_{11},t_{12},t_{13},t_{14},t_{15},t_{16},t_{17},t_{18},t_{19},t_{20},t_{21},$
  $t_{22},t_{23},t_{24},t_{25},t_{26},t_{27},t_{28},t_{29},t_{30},t_{31},t_{32},t_{33},t_{34},t_{35},t_{36},t_{37},t_{38}\}$

- $F = \{(p_1,t_1),(p_2,t_2),(p_2,t_3),(p_3,t_4),(p_4,t_5),(p_5,t_6),(p_5,t_7),$
  $(p_6,t_{10}),(p_6,t_{11}),(p_7,t_8),(p_8,t_9),(p_8,t_{12}),(p_9,t_{13}),$
  $(p_{10},t_{14}),(p_{10},t_{15}),(p_{11},t_{16}),(p_{12},t_{17}),(p_{13},t_{18}),(p_{13},t_{19}),$
  $(p_{14},t_{22}),(p_{14},t_{23}),(p_{15},t_{20}),(p_{16},t_{21}),(p_{16},t_{24}),(p_{17},t_{25}),$
  $(p_{18},t_{26}),(p_{18},t_{27}),(p_{19},t_{28}),(p_{20},t_{29}),(p_{21},t_{30}),(p_{21},t_{31}),$
  $(p_{22},t_{34}),(p_{22},t_{35}),(p_{23},t_{32}),(p_{24},t_{33}),(p_{24},t_{36}),$
  $(p_{25},t_1),(p_{25},t_{13}),(p_{25},t_{25}),(p_{25},t_{37}),(p_{26},t_{38}),(p_{27},t_1),$
  $(p_{27},t_{13}),(p_{27},t_{25})\} \cup$
  $\{(t_1,p_2),(t_2,p_3),(t_3,p_4),(t_4,p_4),(t_5,p_5),(t_6,p_6),(t_7,p_7),$
  $(t_8,p_8),(t_9,p_6),(t_{10},p_7),(t_{11},p_1),(t_{11},p_{17}),(t_{11},p_{19}),$
  $(t_{12},p_1),(t_{12},p_{17}),(t_{12},p_{19}),(t_{13},p_{10}),(t_{14},p_{11}),(t_{15},p_{12}),$
  $(t_{16},p_{12}),(t_{17},p_{13}),(t_{18},p_{14}),(t_{19},p_{15}),(t_{20},p_{16}),(t_{21},p_{14}),$
  $(t_{22},p_{15}),(t_{23},p_9),(t_{23},p_{25}),(t_{23},p_{27}),(t_{24},p_9),(t_{24},p_{25}),(t_{24},p_{27}),$
  $(t_{25},p_{18}),(t_{26},p_{19}),(t_{27},p_{20}),(t_{28},p_{20}),(t_{29},p_{21}),(t_{30},p_{22}),$
  $(t_{31},p_{23}),(t_{32},p_{24}),(t_{33},p_{22}),(t_{34},p_{23}),$
  $(t_{35},p_{17}),(t_{35},p_{25}),(t_{35},p_{27}),(t_{36},p_{17}),(t_{36},p_{25}),$
  $(t_{36},p_{27}),(t_{37},p_{26}),(t_{38},p_{25})\}$

$W(p_1,t_3) = 1, W(p_2,t_1) = 1, W(p_2,t_3) = 1, W(p_2,t_{15}) = 1,$

$W(p_2,t_{27}) = 1, W(p_3,t_2) = 1, W(p_4,t_4) = 1, W(p_4,t_5) = 1,$

$W(p_5,t_6) = 1, W(p_6,t_7) = 1, W(p_7,t_8) = 1, W(p_7,t_9) = 1,$

$W(p_8,t_{12}) = 1, W(p_8,t_{13}) = 1, W(p_9,t_{10}) = 1, W(p_{10},t_{11}) = 1,$

$W(p_{10},t_{14}) = 1, W(p_{11},t_{15}) = 1, W(p_{12},t_{16}) = 1, W(p_{12},t_{17}) = 1,$

$W(p_{13},t_{18}) = 1, W(p_{14},t_{19}) = 1, W(p_{15},t_{20}) = 1, W(p_{15},t_{21}) = 1,$

$W(p_{16},t_{24}) = 1, W(p_{16},t_{25}) = 1, W(p_{17},t_{22}) = 1, W(p_{18},t_{23}) = 1,$

$W(p_{18},t_{26}) = 1, W(p_{19},t_3) = 1, W(p_9,t_5) = 1, W(p_{19},t_{27}) = 1,$

$W(p_{20},t_{27}) = 1, W(p_{21},t_{28}) = 1, W(p_{21},t_{29}) = 1, W(p_{22},t_{30}) = 1,$

$W(p_{23},t_{31}) = 1, W(p_{24},t_{32}) = 1, W(p_{24},t_{33}) = 1, W(p_{25},t_{36}) = 1,$

$W(p_{25},t_{37}) = 1, W(p_{26},t_{34}) = 1, W(p_{27},t_{35}) = 1, W(p_{27},t_{38}) = 1,$

$W(t_1,p_3) = 1, W(t_2,p_2) = 1, W(t_3,p_4) = 1, W(t_4,p_5) = 1,$

$W(t_5,p_6) = 1, W(t_6,p_6) = 1, W(t_7,p_7) = 1, W(t_8,p_8) = 1, W(t_9,p_9) = 1,$

$W(t_{10},p_{10}) = 1, W(t_{11},p_8) = 1, W(t_{12},p_9) = 1, W(t_{13},p_1) = 1, W(t_{13},p_2) = 1,$

$W(t_{13},p_{19}) = 1, W(t_{14},p_1) = 1, W(t_{14},p_2) = 1, W(t_{14},p_{19}) = 1, W(t_{15},p_{12}) = 1,$

$W(t_{16},p_{13}) = 1, W(t_{17},p_{14}) = 1, W(t_{18},p_{14}) = 1,$

$W(t_{19},p_{15}) = 1, W(t_{20},p_{16}) = 1, W(t_{21},p_{17}) = 1, W(t_{22},p_{18}) = 1, W(t_{23},p_{16}) = 1,$

$W(t_{24},p_{17}) = 1, W(t_{25},p_{11}) = 1, W(t_{25},p_2) = 1, W(t_{25},p_{19}) = 1, W(t_{26},p_{11}) = 1,$

$W(t_{26},p_2) = 1, W(t_{26},p_{19}) = 1, W(t_{27},p_{21}) = 1, W(t_{28},p_{22}) = 1, W(t_{29},p_{23}) = 1,$

- $W(t_{30},p_{23}) = 1, W(t_{31},p_{24}) = 1, W(t_{32},p_{25}) = 1, W(t_{33},p_{26}) = 1, W(t_{34},p_{27}) = 1,$

$W(t_{35},p_{25}) = 1, W(t_{36},p_{26}) = 1, W(t_{37},p_{20}) = 1, W(t_{37},p_2) = 1,$

$W(t_{37},p_{19}) = 1, W(t_{38},p_{20}) = 1, W(t_{38},p_2) = 1, W(t_{38},p_{19}) = 1$

- $M_0 = [1,0,0,0,0,0,0,0,1,0,0,0,0,0,0,0,1,0,0,0,0,0,0,0, 1,0, 1]^T$

Matricile de incidenţă obtinute cu ajutorul PNT sunt prezentate mai jos.

Matricea de incidenta de intrare:

$$A_{II} - \begin{pmatrix}
100000000000000000000000 \; 101 \\
010000000000000000000000000 \\
010000000000000000000000000 \\
001000000000000000000000000 \\
000100000000000000000000000 \\
000010000000000000000000000 \\
000010000000000000000000000 \\
000000100000000000000000000 \\
000000010000000000000000000 \\
000001000000000000000000000 \\
000001000000000000000000000 \\
000000010000000000000000000 \\
000000001000000000000000 \; 101 \\
000000000100000000000000000 \\
000000000100000000000000000 \\
000000000010000000000000000 \\
000000000001000000000000000 \\
000000000000100000000000000 \\
000000000000100000000000000 \\
000000000000001000000000000 \\
000000000000000100000000000 \\
000000000000010000000000000 \\
000000000000010000000000000 \\
000000000000000100000000000 \\
000000001000000000000000 \; 101 \\
000000000000000001000000000 \\
000000000000000001000000000 \\
000000000000000000100000000 \\
000000000000000000010000000 \\
000000000000000000001000000 \\
000000000000000000001000000 \\
000000000000000000000010000 \\
000000000000000000000001000 \\
000000000000000000000100000 \\
000000000000000000000100000 \\
000000000000000000000001000 \\
000000000000000000000000100 \\
000000000000000000000000010
\end{pmatrix}$$

Matricea de incidenta de iesire:

$$A_{OI} - \begin{pmatrix}
010000000000000000000000000 \\
001000000000000000000000000 \\
000100000000000000000000000 \\
000100000000000000000000000 \\
000010000000000000000000000 \\
000001000000000000000000000 \\
000000100000000000000000000 \\
000000010000000000000000000 \\
000001000000000000000000000 \\
000000100000000000000000000 \\
100000000000000000000000 \; 101 \\
100000000000000000000000 \; 101 \\
000000000100000000000000000 \\
000000000010000000000000000 \\
000000000001000000000000000 \\
000000000001000000000000000 \\
000000000000100000000000000 \\
000000000000010000000000000 \\
000000000000001000000000000 \\
000000000000000100000000000 \\
000000000000010000000000000 \\
000000000000001000000000000 \\
000000001000000000000000 \; 101 \\
000000001000000000000000 \; 101 \\
000000000000000001000000000 \\
000000000000000000100000000 \\
000000000000000000010000000 \\
000000000000000000010000000 \\
000000000000000000001000000 \\
000000000000000000000100000 \\
000000000000000000000010000 \\
000000000000000000000001000 \\
000000000000000000000100000 \\
000000000000000000000010000 \\
000000000000000010000000 \; 101 \\
000000000000000010000000 \; 101 \\
000000000000000000000000010 \\
000000000000000000000000100
\end{pmatrix}$$

Matricea de incidenta:

$$A_I - A_{OI} - A_{II} = \begin{pmatrix}
-1 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & -1 & 0 & -1 \\
0 & -1 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & -1 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & -1 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & -1 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & -1 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & -1 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & -1 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 1 & 0 & -1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & -1 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
1 & 0 & 0 & 0 & 0 & -1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 1 \\
1 & 0 & 0 & 0 & 0 & 0 & 0 & -1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 1 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & -1 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & -1 & 0 & -1 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & -1 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & -1 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & -1 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & -1 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & -1 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & -1 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & -1 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & -1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & -1 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & -1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 1 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & -1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 1 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & -1 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & -1 & 0 & -1 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & -1 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & -1 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & -1 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & -1 & 1 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & -1 & 1 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & -1 & 0 & 1 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & -1 & 1 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & -1 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & -1 & 1 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & -1 & 0 & 0 & 1 & 0 & 1 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & -1 & 1 & 0 & 1 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & -1 & 1 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & -1 & 0
\end{pmatrix}$$

Arborele de acoperire corespunzator retelei Petri *PN_PT_N3* este prezentat in figura 2.60.

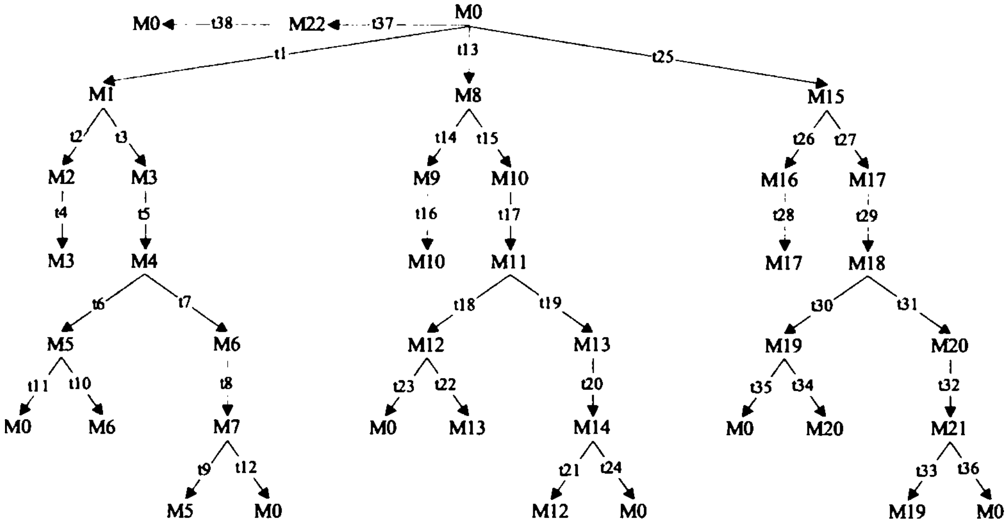


Figura 2.60. Arborele de acoperire corespunzator *PN_PT_N3*

Unde:

M0=(1,0,0,0,0,0,0,0,1,0,0,0,0,0,0,0,1,0,0,0,0,0,0,0,1,0,1);
M1=(0,1,0,0,0,0,0,0,1,0,0,0,0,0,0,0,1,0,0,0,0,0,0,0,0,0,0);
M2=(0,0,1,0,0,0,0,0,1,0,0,0,0,0,0,0,1,0,0,0,0,0,0,0,0,0,0);

M3=(0,0,0,1,0,0,0,0,1,0,0,0,0,0,0,0,1,0,0,0,0,0,0,0,0,0,0);
M4=(0,0,0,0,1,0,0,0,1,0,0,0,0,0,0,0,1,0,0,0,0,0,0,0,0,0,0);
M5=(0,0,0,0,0,1,0,0,1,0,0,0,0,0,0,0,1,0,0,0,0,0,0,0,0,0,0);
M6=(0,0,0,0,0,0,1,0,1,0,0,0,0,0,0,0,1,0,0,0,0,0,0,0,0,0,0);
M7=(0,0,0,0,0,0,0,1,1,0,0,0,0,0,0,0,1,0,0,0,0,0,0,0,0,0,0);
M8=(1,0,0,0,0,0,0,0,0,1,0,0,0,0,0,0,1,0,0,0,0,0,0,0,0,0,0);
M9=(1,0,0,0,0,0,0,0,0,0,1,0,0,0,0,0,1,0,0,0,0,0,0,0,0,0,0);
M10=(1,0,0,0,0,0,0,0,0,0,0,1,0,0,0,0,1,0,0,0,0,0,0,0,0,0,0);
M11=(1,0,0,0,0,0,0,0,0,0,0,0,1,0,0,0,1,0,0,0,0,0,0,0,0,0,0);
M12=(1,0,0,0,0,0,0,0,0,0,0,0,0,1,0,0,1,0,0,0,0,0,0,0,0,0,0);
M13=(1,0,0,0,0,0,0,0,0,0,0,0,0,0,1,0,1,0,0,0,0,0,0,0,0,0,0);
M14=(1,0,0,0,0,0,0,0,0,0,0,0,0,0,0,1,1,0,0,0,0,0,0,0,0,0,0);
M15=(1,0,0,0,0,0,0,0,1,0,0,0,0,0,0,0,0,1,0,0,0,0,0,0,0,0,0);
M16=(1,0,0,0,0,0,0,0,1,0,0,0,0,0,0,0,0,0,1,0,0,0,0,0,0,0,0);
M17=(1,0,0,0,0,0,0,0,1,0,0,0,0,0,0,0,0,0,0,1,0,0,0,0,0,0,0);
M18=(1,0,0,0,0,0,0,0,1,0,0,0,0,0,0,0,0,0,0,0,1,0,0,0,0,0,0);
M19=(1,0,0,0,0,0,0,0,1,0,0,0,0,0,0,0,0,0,0,0,0,1,0,0,0,0,0);
M20=(1,0,0,0,0,0,0,0,1,0,0,0,0,0,0,0,0,0,0,0,0,0,1,0,0,0,0);
M21=(1,0,0,0,0,0,0,0,1,0,0,0,0,0,0,0,0,0,0,0,0,0,0,1,0,0,0);
M22=(1,0,0,0,0,0,0,0,1,0,0,0,0,0,0,0,1,0,0,0,0,0,0,0,0,1,1)

Studiind proprietatile comportamentale ale retelei Petri *PN_PT_N3*, pe baza arborelui de acoperire rezulta:

- reteaua este *marginita* (simbolul ω nu apare in arborele de acoperire);
- reteaua este *sigura* (marcajele din toate nodurile arborelui de acoperire contin numai „0" si „1");
- reteaua *nu este blocanta* (toate tranzitile au asociate arce in arborele de acoperire);
- reteaua este *accesibila* (orice marcaj poate fi atins pornind din M_0).

Graful de acoperire corespunzator este prezentat in figura 2.61.

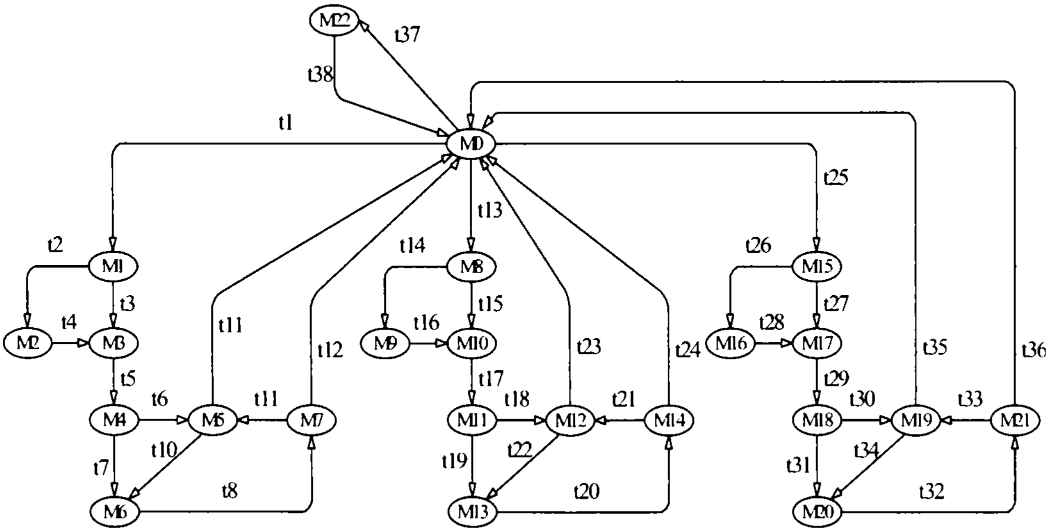


Figura 2.61. Graful de acoperire al retelei Petri *PN_PT_N3*

Structura rezultata pentru graful de acoperire confirma ca modelul de retea Petri corespunzator nodului de tip 3 este *accesibil*.

Concluzionand, se poate afirma ca modelul ales este *viabil* din punct de vedere comportamental.

Analiza retelei Petri *PN_PT_N3* din punct de vedere structural se face pe baza analizei existentei invariantilor de tip *P* respectiv *T*, utilizand teorema 2.3 (Determinarea numarului de invarianti [Pastr97]), rangul matricei A fiind: $rang\, A = 22$.

Rezulta ca reteaua Petri *PN_PT_N3* este acoperita de 5 invarianti de tip *P*, respectiv 16 invarianti de tip *T*.

In figura 2.62 se prezinta invariantii de tip *P* obtinuti in urma simularii retelei Petri *PN_PT_N3* cu ajutorul PNT-ului.

Conform teroemelor 2.6 respectiv 2.7 reteaua Petri *PN_PT_N3* este *conservativa si structural marginita, respectiv este consistenta si repetitiva.*

În final se poate *concluziona* că structura aleasă pentru reteaua Petri corespunzătoare nodului de tip 3 este viabilă, ea putând fi implementată, fiind siguri că nu există condiţii de apariţie a blocajelor sau a apariţiei de situaţii necontrolabile.

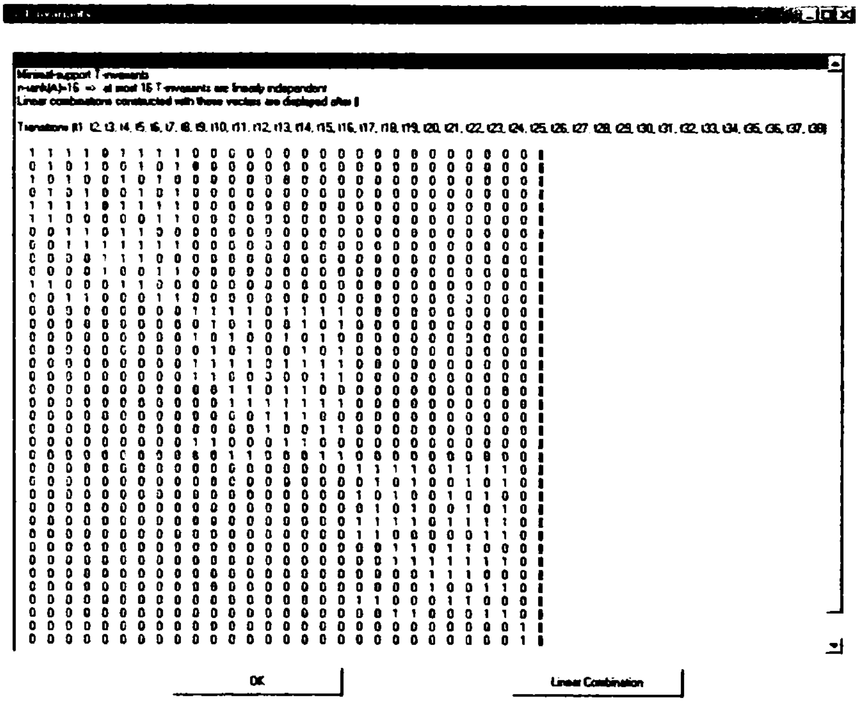


Figura 2.62 Invariantii de tip P obtinuti cu ajutorul PNT

2.5.2.3.2.2. Cazul: retea Petri temporizata P

Structura retelei Petri temporizata P corespunzatoare Nodului de tip 3 este aceeasi ca cea prezentata in figura 2.59.

Datorita faptului ca retelele sunt identice din punct de vedere structural si comportamental, rezultatele analizelor nu se mai detaliaza ele fiind in fapt identice cu cele obtinute in cazul nodului de tip 3 netemporizat.

Fata de situatia prezentata anterior fiecarei pozitii P i-a fost alocata o unitate de timp corespunzatoare. In acest mod situatia devine reala din punct de vedere al functionarii.

Timpii corespunzatori pozitiilor P sunt prezentati in tabelul 2.9.

Tabelul 2.9. *Timpii corespunzatori pozitilor P*

Pozitie	Unitati de Timp
P1, P9,P17	1
P2, P10,P18	5
P3, P11,P19	4
P4, P12,P20	4
P5, P13,P21	7
P6, P14,P22	6
P7, P15,P23	3
P8, P16,P24	5
P25	1
P26	3
P27	1

Reteaua Petri considerata poate fi formalizata prin sixtuplul:

$$PN\_PTtime\_N3 = (P,T,F,W,D,M_0)$$

unde:

- $P = \{p_1, p_2, p_3, p_4, p_5, p_6, p_7, p_8, p_9, p_{10}, p_{11}, p_{12}, p_{13}, p_{14}, p_{15}, p_{16}, p_{17}, p_{18}, p_{19}, p_{20}, p_{21}, p_{22}, p_{23}, p_{24}, p_{25}, p_{26}, p_{27}\}$
- $T = \{t_1, t_2, t_3, t_4, t_5, t_6, t_7, t_8, t_9, t_{10}, t_{11}, t_{12}, t_{13}, t_{14}, t_{15}, t_{16}, t_{17}, t_{18}, t_{19}, t_{20}, t_{21}, t_{22}, t_{23}, t_{24}, t_{25}, t_{26}, t_{27}, t_{28}, t_{29}, t_{30}, t_{31}, t_{32}, t_{33}, t_{34}, t_{35}, t_{36}, t_{37}, t_{38}\}$
- $F = \{(p_1,t_1),(p_2,t_2),(p_2,t_3),(p_3,t_4),(p_4,t_5),(p_5,t_6),(p_5,t_7),(p_6,t_{10}), (p_6,t_{11}),(p_7,t_8),(p_8,t_9),(p_8,t_{12}),(p_9,t_{13}),(p_{10},t_{14}),(p_{10},t_{15}),(p_{11},t_{16}), (p_{12},t_{17}),(p_{13},t_{18}),(p_{13},t_{19}),(p_{14},t_{22}),(p_{14},t_{23}),(p_{15},t_{20}),(p_{16},t_{21}), (p_{16},t_{24}),(p_{17},t_{25}),(p_{18},t_{26}),(p_{18},t_{27}),(p_{19},t_{28}),(p_{20},t_{29}),(p_{21},t_{30}), (p_{21},t_{31}),(p_{22},t_{34}),(p_{22},t_{35}),(p_{23},t_{32}),(p_{24},t_{33}),(p_{24},t_{36}), (p_{25},t_1),(p_{25},t_{13}),(p_{25},t_{25}),(p_{25},t_{37}),(p_{26},t_{38}),(p_{27},t_1),(p_{27},t_{13}), (p_{27},t_{25})\} \cup \{(t_1,p_2),(t_2,p_3),(t_3,p_4),(t_4,p_4),(t_5,p_5),(t_6,p_6),(t_7,p_7),(t_8,p_8), (t_9,p_6),(t_{10},p_7),(t_{11},p_1),(t_{11},p_{17}),(t_{11},p_{19}),(t_{12},p_1),(t_{12},p_{17}),(t_{12},p_{19}), (t_{13},p_{10}),(t_{14},p_{11}),(t_{15},p_{12}),(t_{16},p_{12}),(t_{17},p_{13}),(t_{18},p_{14}),(t_{19},p_{15}), (t_{20},p_{16}),(t_{21},p_{14}),(t_{22},p_{15}),(t_{23},p_9),(t_{23},p_{25}),(t_{23},p_{27}),(t_{24},p_9), (t_{24},p_{25}),(t_{24},p_{27}),(t_{25},p_{18}),(t_{26},p_{19}),(t_{27},p_{20}),(t_{28},p_{20}),(t_{29},p_{21}), (t_{30},p_{22}),(t_{31},p_{23}),(t_{32},p_{24}),(t_{33},p_{22}),(t_{34},p_{23}),(t_{35},p_{17}),(t_{35},p_{25}), (t_{35},p_{27}),(t_{36},p_{17}),(t_{36},p_{25}),(t_{36},p_{27}),(t_{37},p_{26}),(t_{38},p_{25})\}$

$W(p_1,t_3)=1, W(p_2,t_1)=1, W(p_2,t_3)=1, W(p_2,t_{15})=1, W(p_2,t_{27})=1,$
$W(p_3,t_2)=1, W(p_4,t_4)=1, W(p_4,t_5)=1, W(p_5,t_6)=1, W(p_6,t_7)=1,$
$W(p_7,t_8)=1, W(p_7,t_9)=1, W(p_8,t_{12})=1, W(p_8,t_{13})=1, W(p_9,t_{10})=1,$
$W(p_{10},t_{11})=1, W(p_{10},t_{14})=1, W(p_{11},t_{15})=1, W(p_{12},t_{16})=1,$
$W(p_{12},t_{17})=1, W(p_{13},t_{18})=1, W(p_{14},t_{19})=1, W(p_{15},t_{20})=1, W(p_{15},t_{21})=1,$
$W(p_{16},t_{24})=1, W(p_{16},t_{25})=1, W(p_{17},t_{22})=1, W(p_{18},t_{23})=1, W(p_{18},t_{26})=1,$
$W(p_{19},t_3)=1, W(p_9,t_5)=1, W(p_{19},t_{27})=1, W(p_{20},t_{27})=1, W(p_{21},t_{28})=1,$
$W(p_{21},t_{29})=1, W(p_{22},t_{30})=1, W(p_{23},t_{31})=1, W(p_{24},t_{32})=1, W(p_{24},t_{33})=1,$
$W(p_{25},t_{36})=1, W(p_{25},t_{37})=1, W(p_{26},t_{34})=1, W(p_{27},t_{35})=1, W(p_{27},t_{38})=1,$
$W(t_1,p_3)=1, W(t_2,p_2)=1, W(t_3,p_4)=1, W(t_4,p_5)=1, W(t_5,p_6)=1, W(t_6,p_6)=1,$
$W(t_7,p_7)=1, W(t_8,p_8)=1, W(t_9,p_9)=1, W(t_{10},p_{10})=1, W(t_{11},p_8)=1,$
$W(t_{12},p_9)=1, W(t_{13},p_1)=1, W(t_{13},p_2)=1, W(t_{13},p_{19})=1, W(t_{14},p_1)=1,$
$W(t_{14},p_2)=1, W(t_{14},p_{19})=1, W(t_{15},p_{12})=1, W(t_{16},p_{13})=1, W(t_{17},p_{14})=1,$
$W(t_{18},p_{14})=1, W(t_{19},p_{15})=1, W(t_{20},p_{16})=1, W(t_{21},p_{17})=1, W(t_{22},p_{18})=1,$
$W(t_{23},p_{16})=1, W(t_{24},p_{17})=1, W(t_{25},p_{11})=1, W(t_{25},p_2)=1, W(t_{25},p_{19})=1,$
$W(t_{26},p_{11})=1, W(t_{26},p_2)=1, W(t_{26},p_{19})=1, W(t_{27},p_{21})=1, W(t_{28},p_{22})=1,$
$W(t_{29},p_{23})=1, W(t_{30},p_{23})=1, W(t_{31},p_{24})=1, W(t_{32},p_{25})=1,$
$W(t_{33},p_{26})=1, W(t_{34},p_{27})=1, W(t_{35},p_{25})=1, W(t_{36},p_{26})=1, W(t_{37},p_{20})=1,$
• $W(t_{37},p_2)=1, W(t_{37},p_{19})=1, W(t_{38},p_{20})=1, W(t_{38},p_2)=1, W(t_{38},p_{19})=1$

$D=\{d(p_1)=1, d(p_2)=5, d(p_3)=4, d(p_4)=4, d(p_5)=7, d(p_6)=6,$
$d(p_7)=3, d(p_8)=5, d(p_9)=1, d(p_{10})=5, d(p_{11})=4, d(p_{12})=4,$
• $d(p_{13})=7, d(p_{14})=6, d(p_{15})=3, d(p_{16})=5, d(p_{17})=1, d(p_{18})=5,$ •
$d(p_{19})=4, d(p_{20})=4, d(p_{21})=7, d(p_{22})=6, d(p_{23})=3, d(p_{24})=5,$
$d(p_{25})=1, d(p_{26})=3, d(p_{27})=1\}$

$$M_0=[1,0,0,0,0,0,0,0,1,0,0,0,0,0,0,0,1,0,0,0,0,0,0,0,1,0,1]^T$$

În final se poate *concluziona* că structura aleasă pentru o retea Petri temporizata P corespunzătoare nodului de tip 3 este viabilă, ea putând fi implementată, fiind siguri că nu există condiţii de apariţie a blocajelor sau a apariţiei de situaţii necontrolabile.

2.5.2.4. Modelarea nodului de tip 4 – o intrare doua iesiri [UP06a] [UP06b] [Ung06b][Ung06c]

După cum s-a prezentat in capitolul 4, nodul de tip 4 (o intrare două ieşiri) este construit cu ajutorul nodului de tip 1, având în plus un macaz care este comandat direct în funcţie de tranziţia care este executată şi un element de identificare (scanner) a căruciorului. Prin citirea identificatorului, macazul se va poziţiona corespunzător şi din acest moment totul se rezuma la un nod de tip 1.

2.5.2.4.1. Modelarea nodului de tip 4 cu ajutorul automatelor [UP06a] [Ung06b]

Structura automatului secvential considerat pentru modelarea nodului de tip 4 este prezentata în figura 2.63.

Dupa cum se observa, automatul secvential al nodului 4 este realizat prin sincronizarea a doua automate secventiale *AS_N1*, cate unul pentru fiecare linie de iesire. Semnificatiile starilor sunt identice cu cele prezentate la *AS_N1,* cu observatia ca S1, S2, S3 si S4 corespund deplasarii la dreapta, iar S5, S6, S7 si S8 corespund deplasarii la stanga. Starea S9 este starea de identificare (scanare) in care se ia decizia de miscare la dreapta (identificatorul citit corespunde miscarii la dreapta), respectiv la stanga (identificatorul nu este „recunoscut", de la scaner nu s-a primit rezultat in timp util sau codul citit nu este corect).

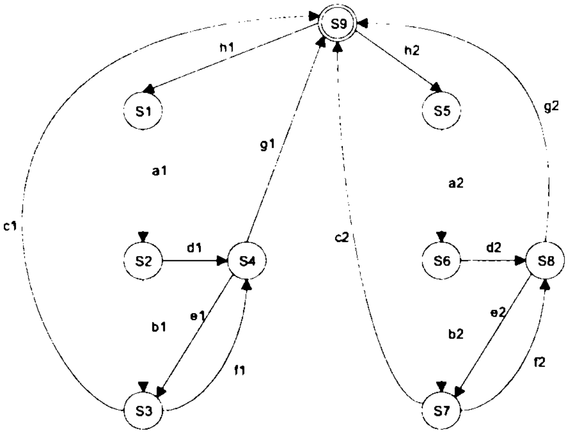


Figura 2.63. Structura automatului secvential corespunzător nodului de tip 4

Evenimentele a1, b1, c1, d1, e1, f1, g1 se refera la miscarea spre dreapta (senzorii corespunzatori liniei din dreapta, conform relatiilor prezentate la *AS_N1*), iar a2, b2, c2, d2, e2, f2, g2 se refera la miscarea spre stanga cu aceeasi observatie ca cea anterioara. Evenimentele h1 respectiv h2 reprezinta conditiile de tranzitie din starea S9 in S1 sau S5 (h1 activ - un carucior identificat se misca spre dreapta, respectiv h2 activ - un carucior neidentificat, sau eroare la citire, se misca spre stanga).

Automatul *AS_N4* care descrie automatul secvential corespunzator nodului de tip 4 este definit astfel:

- multimea starilor: $X = \{S1, S2, S3, S4, S5, S6, S7, S8, S9\}$
- multimea evenimentelor:

  $\Sigma = \{a1, b1, c1, d1, e1, f1, g1, h1, a2, b2, c2, d2, e2, f2, g2, h2\}$
- multimile evenimentelor posibile si functiile de tranzitie a starilor:

  $\Gamma(S1) = \{a1\}$ $\delta(S1, a1) = S2$
  $\Gamma(S2) = \{b1, d1\}$ $\delta(S2, b1) = S3,$ $\delta(S2, d1) = S4$
  $\Gamma(S3) = \{c1, f1\}$ $\delta(S3, c1) = S9,$ $\delta(S3, f1) = S4$
  $\Gamma(S4) = \{e1, g1\}$ $\delta(S4, e1) = S3$ $\delta(S4, g1) = S9$
  $\Gamma(S5) = \{a2\}$ $\delta(S5, a2) = S6$
  $\Gamma(S6) = \{b2, d2\}$ $\delta(S6, b2) = S7,$ $\delta(S6, d2) = S8$
  $\Gamma(S7) = \{c2, f2\}$ $\delta(S7, c2) = S9,$ $\delta(S7, f2) = S8$
  $\Gamma(S8) = \{e2, g2\}$ $\delta(S8, e2) = S7$ $\delta(S8, g2) = S9$
  $\Gamma(S9) = \{h1, h2\}$ $\delta(S9, h1) = S1,$ $\delta(S9, h2) = S5$
- starea initiala: $x_0 = S9$

Considerentele legate de transformarea modelului de tip automat in model de tip retea Petri, prezentate la nodul de tip 1, sunt valabile si in cazul nodului de tip 4.

Pentru conversia din model de tip automat in model de tip retea Petri s-a utilizat acceasi metoda ca si in cazul nodului de tip 1 .

Aplicand algoritmul 2.1 rezulta:

- multimea pozitiilor: $P=\{p1,p2,p3,p4,p5,p6,p7,p8,p9\}$, cu
  $p1=\{S1\},p2=\{S2\},p3=\{S3\},p4=\{S4\},p5=\{S5\},p6=\{S6\}$,
  $p7=\{S7\},p8=\{S8\}$ *si* $p9=\{S9\}$.
- multimea tranzitiilor:

• Pentru *p1*	*(p1,p2)*	$p2=\delta(p1,t1)$	$t1=\{a1\}$
• Pentru *p2*	*(p2,p3)*	$p3=\delta(p2,t2)$	$t2=\{b1\}$
	(p2,p4)	$p4=\delta(p2,t4)$	$t4=\{d1\}$
• Pentru *p3*	*(p3,p9)*	$p9=\delta(p3,t3)$	$t3=\{c1\}$
	(p3,p4)	$p4=\delta(p3,t6)$	$t6=\{f1\}$
• Pentru *p4*	*(p4,p9)*	$p9=\delta(p4,t7)$	$t7=\{g1\}$
	(p4,p3)	$p3=\delta(p4,t5)$	$t5=\{e1\}$
• Pentru *p5*	*(p5,p6)*	$p6=\delta(p5,t8)$	$t8=\{a2\}$
• Pentru *p6*	*(p6,p7)*	$p7=\delta(p6,t9)$	$t9=\{b2\}$
	(p6,p8)	$p8=\delta(p6,t11)$	$t11=\{d2\}$
• Pentru *p7*	*(p7,p9)*	$p9=\delta(p7,t10)$	$t10=\{c2\}$
	(p7,p8)	$p8=\delta(p7,t13)$	$t13=\{f2\}$
• Pentru *p8*	*(p8,p9)*	$p9=\delta(p8,t14)$	$t14=\{g2\}$
	(p8,p7)	$p7=\delta(p8,t12)$	$t12=\{e2\}$
• Pentru *p9*	*(p9,p1)*	$p1=\delta(p9,t15)$	$t15=\{h1\}$
	(p9,p5)	$p5=\delta(p9,t16)$	$t16=\{h2\}$

In figura 2.64 se prezinta structura retelei Petri asociata automatului secvential considerat in cazul nodului de tip 4.

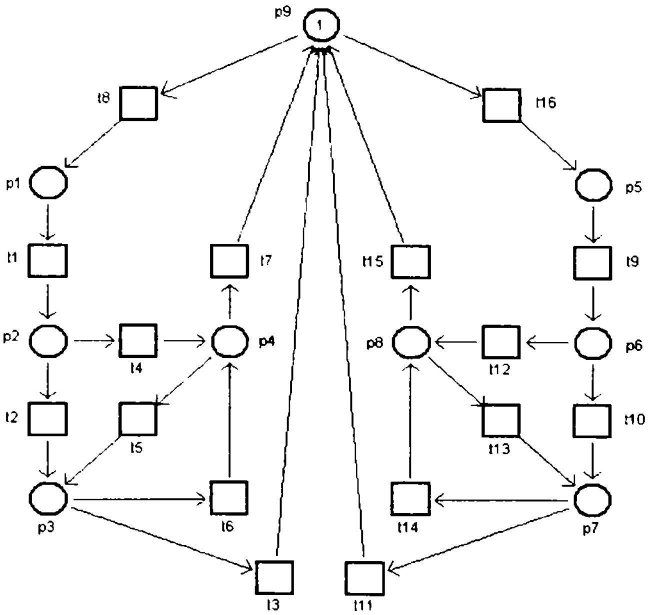


Figura 2.64. Structura retelei Petri asociata automatului secvential corespunzătoare nodului de tip 4

Semnificaţiile pozitiilor este similară cu prezentarea făcută pentru nodul de tip 1. Astfel, P1 şi P5 sunt similare cu P1 de la nodul de tip 1, P2 si P6 sunt similare cu P2 de la nodul de tip 1, P3 şi P7 sunt similare cu P3 de la nodul de tip 1, iar P4 şi P8 sunt similare cu P4 de la nodul de tip 1. În plus, s-a introdus o pozitie noua P9 corespunzatoare pozitiei de scanare.

Semnificaţiile tranziţiilor sunt similare cu cele prezentate la nodul de tip 1. Astfel, t1 şi t8 sunt similare cu t1 de la nodul de tip 1, t2 şi t10 sunt similare cu t2 de la nodul de tip 1, t3 şi t10 sunt similare cu t3 de la nodul de tip 1, cu observaţia că apare o condiţie în plus dată de transferul jetonului în P9, t4 şi t11 sunt similare cu t4 de la nodul de tip 1, t5 şi t12 sunt similare cu t5 de la nodul de tip 1, t6 şi t13 sunt similare cu t6 de la nodul de tip 1, t7 şi t14 sunt similare cu t7 de la nodul de tip 1, cu observaţia că apare o condiţie în plus dată de comutarea in starea P9, si nu in P1. Tranzitile t15 si t16 sunt noi, ele asigurand comutarea fie in starea P1 fie in starea P5, functie de rezultatul scanarii. t5 este activata daca identificatorul citit este corespunzator miscarii spre dreapta, respectiv t16 este activata daca nu s-a recunoscut identificatorul sau a aparut o eroare in functionarea scannerului.

Topologia retelei validata de PNT este prezentata in figura 2.65.

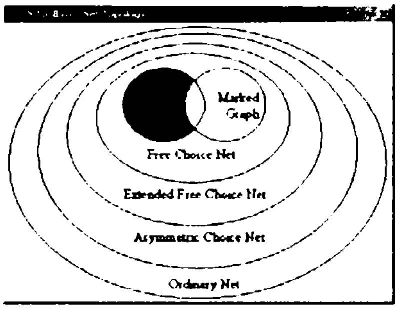


Figura 2.65. Topologia retelei Petri echivalenta validata de PNT

Reteaua Petri considerata este formalizata prin cvintuplul:

$$PN\_AS\_N4 = (P,T,F,W,M_0)$$

unde:

- $P = \{p_1,p_2,p_3,p_4,p_5,p_6,p_7,p_8,p_9\}$
- $T = \{t_1,t_2,t_3,t_4,t_5,t_6,t_7,t_8,t_9,t_{10},t_{11},t_{12},t_{13},t_{14},t_{15},t_{16}\}$
- $$\begin{aligned} F = \{&(p_1,t_1),(p_2,t_2),(p_2,t_4),(p_3,t_6),(p_3,t_3),(p_4,t_5),(p_4,t_7),\\ &(p_5,t_8),(p_6,t_9),(p_6,t_{11}),(p_7,t_{10}),(p_7,t_{13}),(p_8,t_{12}),(p_8,t_{14}),\\ &(p_9,t_{15}),(p_9,t_{16})\} \cup\\ \{&(t_1,p_2),(t_2,p_3),(t_3,p_9),(t_4,p_4),(t_5,p_3),(t_6,p_4),(t_7,p_9),\\ &(t_8,p_6),(t_9,p_7),(t_{10},p_9),(t_{11},p_8),(t_{12},p_7),(t_{13},p_8),(t_{14},p_9),\\ &(t_{15},p_1),(t_{16},p_5)\}\end{aligned}$$

- $W(p_1,t_1) = 1, W(p_2,t_2) = 1, W(p_2,t_4) = 1, W(p_3,t_6) = 1,$
  $W(p_3,t_3) = 1, W(p_4,t_5) = 1, W(p_4,t_7) = 1, W(p_5,t_8) = 1,$
  $W(p_6,t_9) = 1, W(p_6,t_{11}) = 1, W(p_7,t_{10}) = 1, W(p_7,t_{13}) = 1,$
  $W(p_8,t_{12}) = 1, W(p_8,t_{14}) = 1, W(p_9,t_{15}) = 1, W(p_9,t_{16}) = 1,$
  $W(t_1,p_2) = 1, W(t_2,p_3) = 1, W(t_3,p_9) = 1, W(t_4,p_4) = 1,$
  $W(t_5,p_3) = 1, W(t_6,p_4) = 1, W(t_7,p_9) = 1, W(t_8,p_6) = 1,$
  $W(t_9,p_7) = 1, W(t_{10},p_9) = 1, W(t_{11},p_8) = 1, W(t_{12},p_7) = 1,$
  $W(t_{13},p_8) = 1, W(t_{14},p_9) = 1, W(t_{15},p_1) = 1, W(t_{16},p_5) = 1$
- $M_0 = [0,0,0,0,0,0,0,0,1]^T$

Matricile de incidenta corespunzatoare sunt:

Matricea de incidenta de intrare:

$$A_{Ii} = \begin{matrix} 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 \\ 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 \end{matrix}$$

Matricea de incidenta de iesire:

$$A_{Oi} = \begin{matrix} 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 \\ 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 \\ 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 \\ 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 \end{matrix}$$

Matricea de incidenta:

$$A_i = A_{Oi} - A_{Ii} = \begin{pmatrix} -1 & 1 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & -1 & 1 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & -1 & 0 & 0 & 0 & 0 & 0 \\ 0 & -1 & 0 & 1 & 0 & 0 & 0 & 0 \\ 0 & 0 & 1 & -1 & 0 & 0 & 0 & 0 \\ 0 & 0 & -1 & 1 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & -1 & 0 & 0 & 0 & 0 \\ 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & -1 & 1 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & -1 & 1 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & -1 & 0 \\ 0 & 0 & 0 & 0 & 0 & -1 & 0 & 1 \\ 0 & 0 & 0 & 0 & 0 & 0 & 1 & -1 \\ 0 & 0 & 0 & 0 & 0 & 0 & -1 & 1 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & -1 \\ 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 \end{pmatrix}$$

Arborele de acoperire corespunzator este prezentat in figura 2.66.

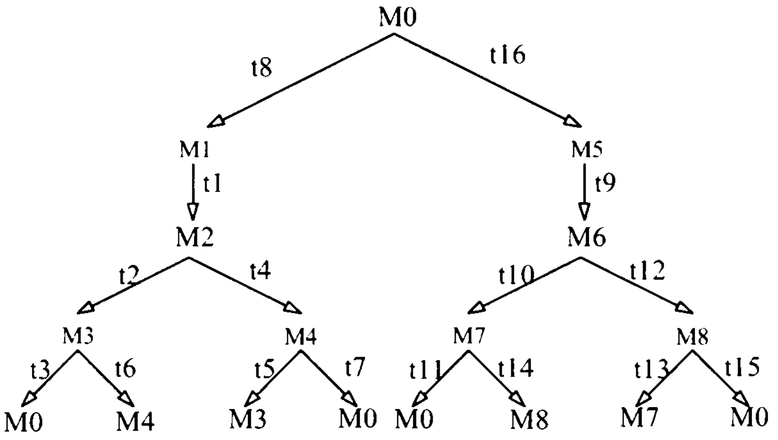


Figura 2.66. Arborele de acoperire al retelei Petri *PN_AS_N4*

Unde:
M0=(0,0,0,0,0,0,0,0,1); M1=(1,0,0,0,0,0,0,0,0); M2=(0,1,0,0,0,0,0,0,0);
M3=(0,0,1,0,0,0,0,0,0); M4=(0,0,0,1,0,0,0,0,0); M5=(0,0,0,0,1,0,0,0,0);
M6=(0,0,0,0,0,1,0,0,0); M7=(0,0,0,0,0,0,1,0,0); M8=(0,0,0,0,0,0,0,1,0)

Studiind proprietatile comportamentale ale retelei Petri *PN_AS_N4*, pe baza arborelui de acoperire rezulta:

- reteaua este *marginita* (simbolul ω nu apare in arborele de acoperire);
- reteaua este *sigura* (marcajele din toate nodurile arborelui de acoperire contin numai „0" si „1");
- reteaua *nu este blocanta* (toate tranzitile au asociate arce in arborele de acoperire);
- reteaua este *accesibila* (orice marcaj poate fi atins pornind din M_0).

Graful de acoperire corespunzator este prezentat in figura 2.67.

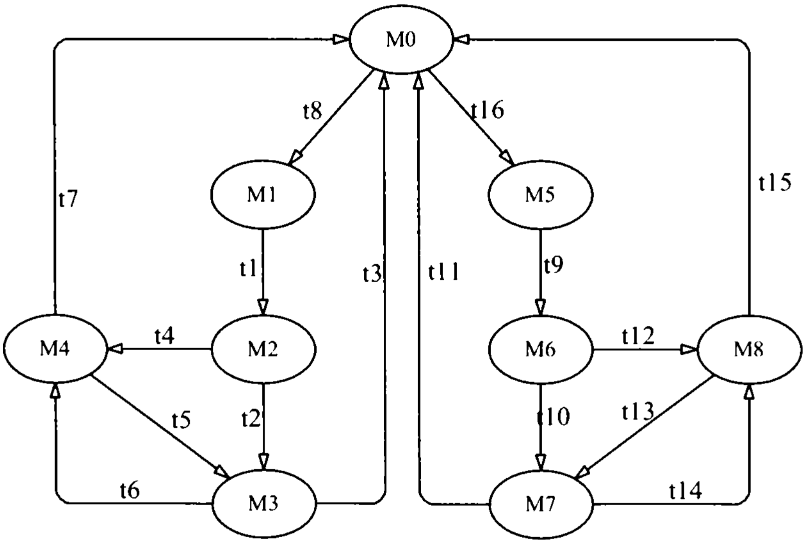


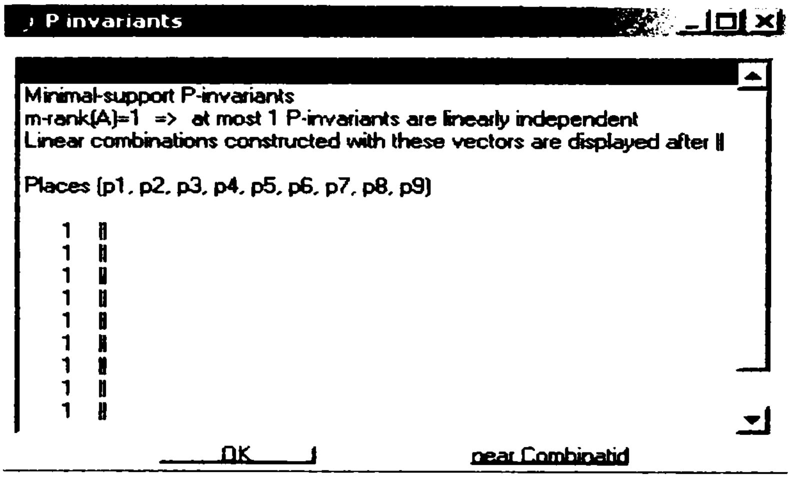
Figura 2.67. Graful de acoperire al retelei Petri *PN_AS_N4*

Structura rezultata pentru graful de acoperire confirma ca modelul de retea Petri derivata din modelul de tip automat corespunzator nodului de tip 4 este *acesibila*.

Concluzionand, se poate afirma ca modelul ales este *viabil* din punct de vedere comportamental.

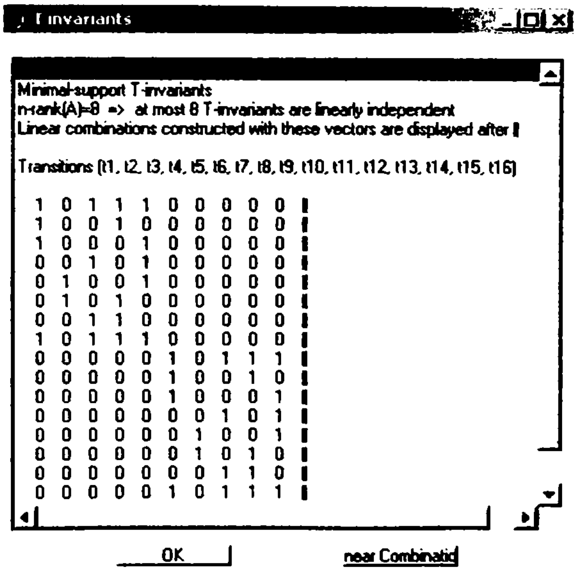
Analiza retelei Petri *PN_AS_N4* din punct de vedere structural se face pe baza analizei existentei invariantilor de tip *P* respectiv *T*.

Pentru determinarea numarului de invarianti *P* respectiv *T* s-a aplicat teorema 2.3, rangul matricei A este: $rang\,A = 8$, rezultand ca reteaua Petri *PN_AS_N4* este acoperita de 1 invariant de tip *P*, respectiv 8 invarianti de tip *T*.
In figurile 2.68 si 2.69 se prezinta invariantii de tip P respectiv T obtinuti in urma simularii retelei Petri *PN_AS_N4* cu ajutorul PNT-ului.



```
Minimal-support P-invariants
m-rank(A)=1  =>  at most 1 P-invariants are linearly independent
Linear combinations constructed with these vectors are displayed after |

Places (p1, p2, p3, p4, p5, p6, p7, p8, p9)

  1  |
  1  |
  1  |
  1  |
  1  |
  1  |
  1  |
  1  |
  1  |
```

Figura 2.68. Invariantii de tip P obtinuti cu ajutorul PNT



```
Minimal-support T-invariants
m-rank(A)=8  =>  at most 8 T-invariants are linearly independent
Linear combinations constructed with these vectors are displayed after |

Transitions (t1, t2, t3, t4, t5, t6, t7, t8, t9, t10, t11, t12, t13, t14, t15, t16)

  1  0  1  1  1  0  0  0  0  0  |
  1  0  0  1  0  0  0  0  0  0  |
  1  0  0  0  1  0  0  0  0  0  |
  0  0  1  0  1  0  0  0  0  0  |
  0  1  0  0  1  0  0  0  0  0  |
  0  1  0  1  0  0  0  0  0  0  |
  0  0  1  1  0  0  0  0  0  0  |
  1  0  1  1  1  0  0  0  0  0  |
  0  0  0  0  0  1  0  1  1  1  |
  0  0  0  0  0  1  0  0  1  0  |
  0  0  0  0  0  1  0  0  0  1  |
  0  0  0  0  0  0  0  1  0  1  |
  0  0  0  0  0  0  1  0  0  1  |
  0  0  0  0  0  0  1  0  1  0  |
  0  0  0  0  0  0  0  1  1  0  |
  0  0  0  0  0  1  0  1  1  1  |
```

Figura 2.69. Invariantii de tip T obtinuti cu ajutorul PNT

Conform teroemelor 2.6 respectiv 2.7 reteaua Petri *PN_AS_N4,* este *conservativa si structural marginita, respectiv este consistenta si repetitiva.*

În final se poate *concluziona* că structura aleasă pentru un automat secvential clasic corespunzător nodului de tip 4 este *viabilă*, ea putând fi implementată, fiind siguri că nu există condiţii de apariţie a blocajelor sau a apariţiei de situaţii necontrolabile.

2.5.2.4.2. Modelarea nodului de tip 4 cu ajutorul retelelor Petri [UP06b] [Ung06c]

2.5.2.4.2.1. Cazul: retea Petri netemporizata

Structura retelei Petri considerate pentru modelarea nodului de tip 4 este prezentata în figura 2.70.

Dupa cum se poate observa, reteaua considerata este construita din doua retele Petri corespunzatoare Nodului de tip 1, sincronizate intre ele printr-o subretea care modeleaza operatia de identificare (scanner-ul).

Semnificaţiile pozitiilor *P*1, P2, P3, P4, P5, P6, P7, P8, P9, P10, P11, P12, P13, P14, P15, P16, P17, P18, P19 si P20 sunt similare cu cele prezentate la nodul de tip 1. P21 este pozitia de asteptare a scannerului, P22 este pozitia de trigger (spotul este aprins), P23 este pozitia de asteptare a rezultatului.

Tranziţiile t1, t2, t3, t4, t5, t6, t7, t8, t9, t10, t11, t12, t13, t14, t15, t16, t17, t18, t19, t20, t21, t22, t23, t24, t25, t26, t27 si t28 sunt similare cu cele prezentate la nodul de tip 1. Tranzitia t29 corespunde trecerii din P21 in P22 ca urmare a declansarii unei operatii de scanare (identificare) , t30 corespunde situatiei in care operatia de triggerare s-a incheiat urmand sa se astepte rezultatul.

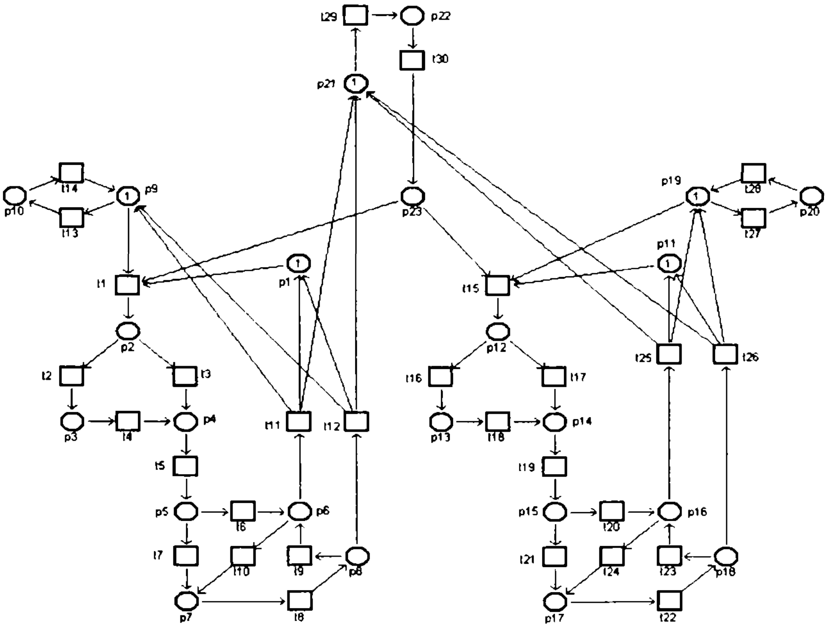


Figura 2.70. Structura retelei Petri corespunzătoare nodului de tip 4

Topologia retelei Petri validata de PNT este prezentata in figura 2.71.

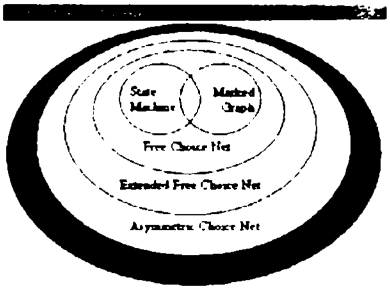


Figura 2.71. Topologia retelei Petri validata de TPN

Reteaua Petri considerata este formalizata prin cvintuplul:

$$PN\_PT\_N4 = (P, T, F, W, M_0)$$

unde:

- $P = \{p_1, p_2, p_3, p_4, p_5, p_6, p_7, p_8, p_9, p_{10}, p_{11}, p_{12}, p_{13}, p_{14}, p_{15}, p_{16},$
  $p_{17}, p_{18}, p_{19}, p_{20}, p_{21}, p_{22}, p_{23}\}$
- $T = \{t_1, t_2, t_3, t_4, t_5, t_6, t_7, t_8, t_9, t_{10}, t_{11}, t_{12}, t_{13}, t_{14}, t_{15}, t_{16}, t_{17}, t_{18}, t_{19}, t_{20},$
  $t_{21}, t_{22}, t_{23}, t_{24}, t_{25}, t_{26}, t_{27}, t_{28}, t_{29}, t_{30}\}$
- $F = \{(p_1, t_1), (p_2, t_2), (p_2, t_3), (p_3, t_4), (p_4, t_5), (p_5, t_6), (p_5, t_7), (p_6, t_{10}),$
  $(p_6, t_{11}), (p_7, t_8), (p_8, t_9), (p_8, t_{12}), (p_9, t_{13}), (p_{10}, t_{14}), (p_{11}, t_{15}),$
  $(p_{12}, t_{16}), (p_{12}, t_{17}), (p_{13}, t_{18}), (p_{14}, t_{19}), (p_{15}, t_{20}), (p_{15}, t_{21}),$
  $(p_{16}, t_{24}), (p_{16}, t_{25}), (p_{17}, t_{22}), (p_{18}, t_{23}), (p_{18}, t_{26}), (p_{19}, t_{27}),$
  $(p_{20}, t_{28}), (p_{21}, t_{29}), (p_{22}, t_{30}), (p_{23}, t_1), (p_{23}, t_{15})\} \cup$
  $\{(t_1, p_2), (t_2, p_3), (t_3, p_4), (t_4, p_4), (t_5, p_5), (t_6, p_6), (t_7, p_7), (t_8, p_8),$
  $(t_9, p_6), (t_{10}, p_7), (t_{11}, p_1), (t_{11}, p_9), (t_{11}, p_{21}), (t_{12}, p_1), (t_{12}, p_9),$
  $(t_{12}, p_{21}), (t_{13}, p_{10}), (t_{14}, p_9), (t_{15}, p_{12}), (t_{16}, p_{13}), (t_{17}, p_{14}),$
  $(t_{18}, p_{14}), (t_{19}, p_{15}), (t_{20}, p_{16}), (t_{21}, p_{17}), (t_{22}, p_{18}), (t_{23}, p_{16}), (t_{24}, p_{17}),$
  $(t_{25}, p_{11}), (t_{25}, p_{19}), (t_{25}, p_{21}), (t_{26}, p_{11}), (t_{26}, p_{19}), (t_{26}, p_{21}), (t_{27}, p_{20}),$
  $(t_{28}, p_{19}), (t_{29}, p_{22}), (t_{30}, p_{23})\}$

$W(p_1,t_1) = 1, W(p_2,t_2) = 1, W(p_2,t_3) = 1, W(p_3,t_4) = 1, W(p_4,t_5) = 1,$
$W(p_5,t_6) = 1, W(p_5,t_7) = 1, W(p_6,t_{10}) = 1, W(p_6,t_{11}) = 1, W(p_7,t_8) = 1,$
$W(p_8,t_9) = 1, W(p_8,t_{12}) = 1, W(p_9,t_{13}) = 1, W(p_{10},t_{14}) = 1,$
$W(p_{11},t_{15}) = 1, W(p_{12},t_{16}) = 1, W(p_{12},t_{17}) = 1, W(p_{13},t_{18}) = 1,$
$W(p_{14},t_{19}) = 1, W(p_{15},t_{20}) = 1, W(p_{15},t_{21}) = 1, W(p_{16},t_{24}) = 1,$
- $W(p_{16},t_{25}) = 1, W(p_{17},t_{22}) = 1, W(p_{18},t_{23}) = 1, W(p_{18},t_{26}) = 1,$
$W(p_{19},t_{27}) = 1, W(p_{20},t_{28}) = 1, W(p_{21},t_{29}) = 1, W(p_{22},t_{30}) = 1,$
$W(p_{23},t_1) = 1, W(p_{23},t_{15}) = 1, W(t_1,p_2) = 1, W(t_2,p_3) = 1, W(t_3,p_4) = 1,$
$W(t_4,p_4) = 1, W(t_5,p_5) = 1, W(t_6,p_6) = 1, W(t_7,p_7) = 1, W(t_8,p_8) = 1, W(t_9,p_6) = 1,$
$W(t_{10},p_7) = 1, W(t_{11},p_1) = 1, W(t_{11},p_9) = 1, W(t_{11},p_{21}) = 1, W(t_{12},p_1) = 1,$
$W(t_{12},p_9) = 1, W(t_{12},p_{21}) = 1, W(t_{13},p_{10}) = 1, W(t_{14},p_9) = 1, W(t_{15},p_{12}) = 1,$
$W(t_{16},p_{13}) = 1, W(t_{17},p_{14}) = 1, W(t_{18},p_{14}) = 1, W(t_{19},p_{15}) = 1, W(t_{20},p_{16}) = 1,$
$W(t_{21},p_{17}) = 1, W(t_{22},p_{18}) = 1, W(t_{23},p_{16}) = 1, W(t_{24},p_{17}) = 1, W(t_{25},p_{11}) = 1,$
$W(t_{25},p_{19}) = 1, W(t_{25},p_{21}) = 1, W(t_{26},p_{11}) = 1, W(t_{26},p_{19}) = 1, W(t_{26},p_{21}) = 1,$
$W(t_{27},p_{20}) = 1, W(t_{28},p_{19}) = 1, W(t_{29},p_{22}) = 1, W(t_{30},p_{23}) = 1\}$

- $M_0 = [1,\ 1,0,0,0,0,0,0,0,0,1,1,0,0,0,0,0,0,0,0,1,0,0,0,0]^T$

Matricile de incidenţă corespunzătoare generate cu ajutorul lui PNT sunt:

Matricea de incidenta de intrare:

$$A_{Ii} = \begin{pmatrix}
1000000010000000000000\;1 \\
0\;1000000000000000000000 \\
0\;1000000000000000000000 \\
00\;100000000000000000000 \\
000\;10000000000000000000 \\
0000\;1000000000000000000 \\
0000\;1000000000000000000 \\
000000\;10000000000000000 \\
0000000\;1000000000000000 \\
00000\;100000000000000000 \\
00000\;100000000000000000 \\
0000000\;1000000000000000 \\
00000000\;100000000000000 \\
000000000\;10000000000000 \\
0000000000\;10000000\;1000\;1 \\
00000000000\;100000000000 \\
00000000000\;100000000000 \\
000000000000\;10000000000 \\
0000000000000\;1000000000 \\
00000000000000\;100000000 \\
00000000000000\;100000000 \\
0000000000000000\;1000000 \\
00000000000000000\;100000 \\
000000000000000\;10000000 \\
000000000000000\;10000000 \\
00000000000000000\;100000 \\
000000000000000000\;10000 \\
0000000000000000000\;1000 \\
00000000000000000000\;100 \\
000000000000000000000\;10
\end{pmatrix}$$

Matricea de incidenta de iesire:

$$A_{Oi} = \begin{pmatrix}
01000000000000000000000 \\
00100000000000000000000 \\
00010000000000000000000 \\
00010000000000000000000 \\
00001000000000000000000 \\
00000100000000000000000 \\
00000010000000000000000 \\
00000001000000000000000 \\
00000100000000000000000 \\
00000010000000000000000 \\
10000000100000000000\;100 \\
10000000100000000000\;100 \\
00000000010000000000000 \\
00000000100000000000000 \\
00000000000100000000000 \\
00000000000010000000000 \\
00000000000001000000000 \\
00000000000001000000000 \\
00000000000000100000000 \\
00000000000000010000000 \\
00000000000000001000000 \\
00000000000000000100000 \\
00000000000000010000000 \\
00000000000000001000000 \\
00000000001000000010\;100 \\
00000000001000000010\;100 \\
00000000000000000001000 \\
00000000000000000010000 \\
00000000000000000000010 \\
00000000000000000000001
\end{pmatrix}$$

Matricea de incidenta:

$$A_i = A_{0i} - A_{Ii} = \begin{pmatrix}
-1 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & -1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & -1 \\
0 & -1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & -1 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & -1 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & -1 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & -1 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & -1 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & -1 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 1 & 0 & -1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & -1 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
1 & 0 & 0 & 0 & 0 & -1 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 \\
1 & 0 & 0 & 0 & 0 & 0 & 0 & -1 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & -1 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & -1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & -1 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & -1 & 0 & 0 & 0 & -1 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & -1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & -1 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & -1 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & -1 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & -1 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & -1 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & -1 & 1 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & -1 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & -1 & 1 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & -1 & 0 & 0 & 1 & 0 & 1 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & -1 & 1 & 0 & 1 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & -1 & 1 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & -1 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & -1 & 1 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & -1 & 1
\end{pmatrix}$$

Arborele de acoperire corespunzator retelei Petri *PN_PT_N4* este prezentat in figura 2.72.

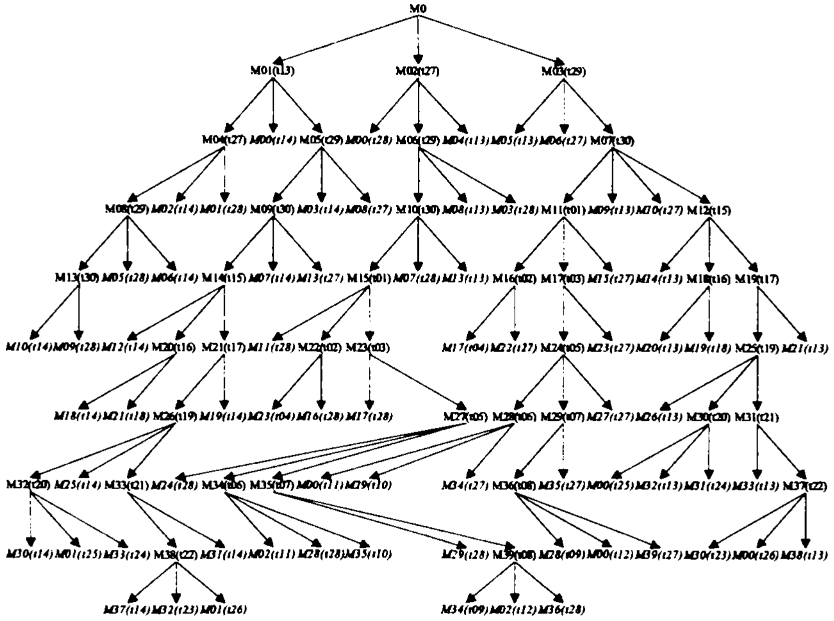


Figura 2.72. Arborele de acoperire corespunzator *PN_PT_N4*

Unde:
M0 = [1,0,0,0,0,0,0,0,1,0,1,0,0,0,0,0,0,0,1,0,1,0,0],
M1 = [1,0,0,0,0,0,0,0,0,1,1,0,0,0,0,0,0,0,1,0,1,0,0],
M2 = [1,0,0,0,0,0,0,0,1,0,1,0,0,0,0,0,0,0,0,1,1,0,0],
M3 = [1,0,0,0,0,0,0,0,1,0,1,0,0,0,0,0,0,0,1,0,0,1,0],
M4 = [1,0,0,0,0,0,0,0,0,1,1,0,0,0,0,0,0,0,0,1,1,0,0],
M5 = [1,0,0,0,0,0,0,0,0,1,1,0,0,0,0,0,0,0,1,0,0,1,0],
M6 = [1,0,0,0,0,0,0,0,1,0,1,0,0,0,0,0,0,0,0,1,0,1,0],
M7 = [1,0,0,0,0,0,0,0,1,0,1,0,0,0,0,0,0,0,1,0,0,0,1],
M8 = [1,0,0,0,0,0,0,0,0,1,1,0,0,0,0,0,0,0,0,1,0,1,0],
M9 = [1,0,0,0,0,0,0,0,0,1,1,0,0,0,0,0,0,0,1,0,0,0,1],
M10= [1,0,0,0,0,0,0,0,1,0,1,0,0,0,0,0,0,0,0,1,0,0,1],
M11 = [0,1,0,0,0,0,0,0,0,0,1,0,0,0,0,0,0,0,1,0,0,0,0],
M12 = [1,0,0,0,0,0,0,0,1,0,0,1,0,0,0,0,0,0,0,0,0,0,0],
M13 = [1,0,0,0,0,0,0,0,0,1,1,0,0,0,0,0,0,0,0,1,0,0,1],
M14 = [1,0,0,0,0,0,0,0,0,1,0,1,0,0,0,0,0,0,0,0,0,0,0],
M15 = [0,1,0,0,0,0,0,0,0,0,1,0,0,0,0,0,0,0,0,1,0,0,0],
M16 = [0,0,1,0,0,0,0,0,0,0,1,0,0,0,0,0,0,0,1,0,0,0,0],
M17 = [0,0,0,1,0,0,0,0,0,0,1,0,0,0,0,0,0,0,1,0,0,0,0],
M18 = [1,0,0,0,0,0,0,0,1,0,0,0,1,0,0,0,0,0,0,0,0,0,0],
M19 = [1,0,0,0,0,0,0,0,1,0,0,0,0,1,0,0,0,0,0,0,0,0,0],
M20 = [1,0,0,0,0,0,0,0,0,1,0,0,1,0,0,0,0,0,0,0,0,0,0],
M21 = [1,0,0,0,0,0,0,0,0,1,0,0,0,1,0,0,0,0,0,0,0,0,0],
M22 = [0,0,1,0,0,0,0,0,0,0,1,0,0,0,0,0,0,0,0,1,0,0,0],
M23 = [0,0,0,1,0,0,0,0,0,0,1,0,0,0,0,0,0,0,0,1,0,0,0],
M24 = [0,0,0,0,1,0,0,0,0,0,1,0,0,0,0,0,0,0,1,0,0,0,0],
M25 = [1,0,0,0,0,0,0,0,1,0,0,0,0,0,1,0,0,0,0,0,0,0,0],
M26 = [1,0,0,0,0,0,0,0,0,1,0,0,0,0,1,0,0,0,0,0,0,0,0],
M27 = [0,0,0,0,1,0,0,0,0,0,1,0,0,0,0,0,0,0,0,1,0,0,0],
M28 = [0,0,0,0,0,1,0,0,0,0,1,0,0,0,0,0,0,0,1,0,0,0,0],
M29 = [0,0,0,0,0,0,1,0,0,0,1,0,0,0,0,0,0,0,1,0,0,0,0],
M30 = [1,0,0,0,0,0,0,0,1,0,0,0,0,0,0,1,0,0,0,0,0,0,0],
M31 = [1,0,0,0,0,0,0,0,1,0,0,0,0,0,0,0,1,0,0,0,0,0,0],
M32 = [1,0,0,0,0,0,0,0,0,1,0,0,0,0,0,1,0,0,0,0,0,0,0],
M33 = [1,0,0,0,0,0,0,0,0,1,0,0,0,0,0,0,1,0,0,0,0,0,0],
M34 = [0,0,0,0,0,1,0,0,0,0,1,0,0,0,0,0,0,0,0,1,0,0,0],
M35 = [0,0,0,0,0,0,1,0,0,0,1,0,0,0,0,0,0,0,0,1,0,0,0]
M36 = [0,0,0,0,0,0,0,1,0,0,1,0,0,0,0,0,0,0,1,0,0,0,0],
M37 = [1,0,0,0,0,0,0,0,1,0,0,0,0,0,0,0,0,1,0,0,0,0,0]
M38 = [1,0,0,0,0,0,0,0,0,1,0,0,0,0,0,0,0,1,0,0,0,0,0],
M39 = [0,0,0,0,0,0,0,1,0,0,1,0,0,0,0,0,0,0,0,1,0,0,0]

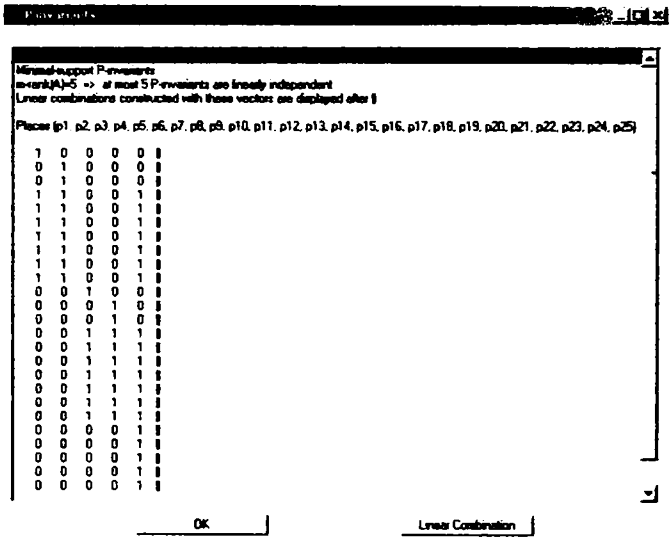
Studiind proprietatile comportamentale ale retelei Petri *PN_PT_N4*, pe baza arborelui de acoperire rezulta:

- reteaua este *marginita* (simbolul ω nu apare in arborele de acoperire);
- reteaua este *sigura* (marcajele din toate nodurile arborelui de acoperire contin numai „0” si „1”);
- reteaua *nu este blocanta* (toate tranzitile au asociate arce in arborele de acoperire);
- reteaua este *accesibila* (orice marcaj poate fi atins pornind din M_0).

Concluzionand, se poate afirma ca modelul ales este *viabil* din punct de vedere comportamental.

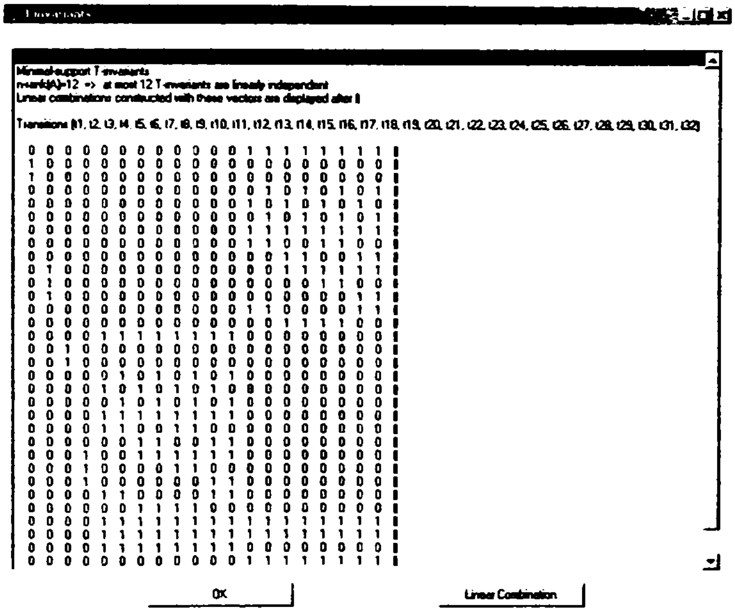
Analiza retelei Petri *PN_PT_N4* din punct de vedere structural se face pe baza analizei existentei invariantilor de tip *P* respectiv *T*.

Pentru determinarea numarului de invarianti *P* respectiv *T* s-a aplicat si acest caz teorema 2.3, rangul matricei A fiind: $rang\,A = 20$, rezulta ca reteaua Petri *PN_PT_N4* este acoperita de 5 invarianti de tip *P*, respectiv 16 invarianti de tip *T*.
In figurile 2.73 si 2.74 se prezinta invariantii de tip *P* respectiv *T* obtinuti in urma simularii retelei Petri *PN_PT_N4* cu ajutorul PNT-ului.



```
Minimal-support P-invariants
m-rank(A)=5 => at most 5 P-invariants are linearly independent
Linear combinations constructed with these vectors are displayed after |

Places (p1, p2, p3, p4, p5, p6, p7, p8, p9, p10, p11, p12, p13, p14, p15, p16, p17, p18, p19, p20, p21, p22, p23, p24, p25)

  1  0  0  0  0 |
  0  1  0  0  0 |
  0  1  0  0  0 |
  1  1  0  0  1 |
  1  1  0  0  1 |
  1  1  0  0  1 |
  1  1  0  0  1 |
  1  1  0  0  1 |
  1  1  0  0  1 |
  1  1  0  0  1 |
  0  0  1  0  0 |
  0  0  0  1  0 |
  0  0  0  1  0 |
  0  0  1  1  1 |
  0  0  1  1  1 |
  0  0  1  1  1 |
  0  0  1  1  1 |
  0  0  1  1  1 |
  0  0  1  1  1 |
  0  0  1  1  1 |
  0  0  0  0  1 |
  0  0  0  0  1 |
  0  0  0  0  1 |
  0  0  0  0  1 |
  0  0  0  0  1 |
```

Figura 2.73. Invariantii de tip P obtinuti cu ajutorul PNT



```
Minimal-support T-invariants
n-rank(A)=12 => at most 12 T-invariants are linearly independent
Linear combinations constructed with these vectors are displayed after |

Transitions (t1, t2, t3, t4, t5, t6, t7, t8, t9, t10, t11, t12, t13, t14, t15, t16, t17, t18, t19, t20, t21, t22, t23, t24, t25, t26, t27, t28, t29, t30, t31, t32)

  0  0  0  0  0  0  0  0  0  0  0  0  1  1  1  1  1  1  1  1 |
  1  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0 |
  1  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0 |
  0  0  0  0  0  0  0  0  0  0  0  0  0  1  0  1  0  1  0  1 |
  0  0  0  0  0  0  0  0  0  0  0  0  1  0  1  0  1  0  1  0 |
  0  0  0  0  0  0  0  0  0  0  0  0  0  1  0  1  0  1  0  1 |
  0  0  0  0  0  0  0  0  0  0  0  0  1  1  1  1  1  1  1  1 |
  0  0  0  0  0  0  0  0  0  0  0  0  1  1  0  0  1  1  0  0 |
  0  0  0  0  0  0  0  0  0  0  0  0  0  0  1  1  0  0  1  1 |
  0  1  0  0  0  0  0  0  0  0  0  0  0  0  1  1  1  1  1  1 |
  0  1  0  0  0  0  0  0  0  0  0  0  0  0  0  0  1  1  0  0 |
  0  1  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  1  1 |
  0  0  0  0  0  0  0  0  0  0  0  0  1  1  0  0  0  0  1  1 |
  0  0  0  0  0  0  0  0  0  0  0  0  0  0  1  1  1  1  0  0 |
  0  0  0  0  1  1  1  1  1  1  1  1  0  0  0  0  0  0  0  0 |
  0  0  1  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0 |
  0  0  1  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0 |
  0  0  0  0  0  1  0  1  0  1  0  1  0  0  0  0  0  0  0  0 |
  0  0  0  0  1  0  1  0  1  0  1  0  0  0  0  0  0  0  0  0 |
  0  0  0  0  0  1  0  1  0  1  0  1  0  0  0  0  0  0  0  0 |
  0  0  0  0  1  1  1  1  1  1  1  1  0  0  0  0  0  0  0  0 |
  0  0  0  0  1  1  0  0  1  1  0  0  0  0  0  0  0  0  0  0 |
  0  0  0  0  0  0  1  1  0  0  1  1  0  0  0  0  0  0  0  0 |
  0  0  0  1  0  0  1  1  1  1  1  1  0  0  0  0  0  0  0  0 |
  0  0  0  1  0  0  0  0  1  1  0  0  0  0  0  0  0  0  0  0 |
  0  0  0  1  0  0  0  0  0  0  1  1  0  0  0  0  0  0  0  0 |
  0  0  0  0  1  1  0  0  0  0  1  1  0  0  0  0  0  0  0  0 |
  0  0  0  0  0  0  1  1  1  1  0  0  0  0  0  0  0  0  0  0 |
  0  0  0  0  1  1  1  1  1  1  1  1  1  1  1  1  1  1  1  1 |
  0  0  0  0  1  1  1  1  1  1  1  1  1  1  1  1  1  1  1  1 |
  0  0  0  0  1  1  1  1  1  1  1  1  0  0  0  0  0  0  0  0 |
  0  0  0  0  0  0  0  0  0  0  0  0  1  1  1  1  1  1  1  1 |
```

Figura 2.74. Invariantii de tip T obtinuti cu ajutorul PNT

Conform teroemelor 2.6 respectiv 2.7, reteaua Petri *PN_PT_N4* este *conservativa si structural marginita, respectiv este consistenta si repetitiva.*

În final se poate *concluziona* că structura aleasă pentru reteaua Petri corespunzătoare nodului de tip 4 este viabilă, ea putând fi implementată, fiind siguri că nu există condiţii de apariţie a blocajelor sau a apariţiei de situaţii necontrolabile.

2.5.2.4.2.2. Cazul: retea Petri temporizata P

Structura retelei Petri temporizata P corespunzatoare Nodului de tip 4 este aceeasi ca cea prezentata in figura 2.70.

Datorita faptului ca retelele sunt identice din punct de vedere structural si comportamental rezultatele analizelor nu se mai detaliaza, ele fiind in fapt identice cu cele obtinute in cazul nodului de tip 4 netemporizat.

Fata de situatia prezentata anterior fiecarei pozitii P i-a fost alocata o unitate de timp corespunzatoare. In acest mod situatia devine reala din punct de vedere al functionarii.

Timpii corespunzatori pozitiilor P sunt prezentati in tabelul 2.10.

Tabelul 2.10. *Timpii corespunzatori pozitilor P*

Pozitie	Unitati de Timp
P1, P11	1
P2,P12	1
P3,P13	3
P4, P14	5
P5, P15	4
P6, P16	4
P7, P17	7
P8, P18	6
P9, P19	3
P10, P20	5
P21	1
P22	3
P23	4

Reteaua Petri considerata este formalizata prin sixtuplul:

$$PN\_PTTime\_N4 = (P,T,F,W,D,M_0)$$

unde:

- $P = \{p_1, p_2, p_3, p_4, p_5, p_6, p_7, p_8, p_9, p_{10}, p_{11}, p_{12}, p_{13}, p_{14}, p_{15}, p_{16}, p_{17}, p_{18}, p_{19}, p_{20}, p_{21}, p_{22}, p_{23}\}$
- $T = \{t_1, t_2, t_3, t_4, t_5, t_6, t_7, t_8, t_9, t_{10}, t_{11}, t_{12}, t_{13}, t_{14}, t_{15}, t_{16}, t_{17}, t_{18}, t_{19}, t_{20}, t_{21}, t_{22}, t_{23}, t_{24}, t_{25}, t_{26}, t_{27}, t_{28}, t_{29}, t_{30}\}$

- $$\begin{aligned}F = \{&(p_1,t_1),(p_2,t_2),(p_2,t_3),(p_3,t_4),(p_4,t_5),(p_5,t_6),(p_5,t_7),(p_6,t_{10}),\\ &(p_6,t_{11}),(p_7,t_8),(p_8,t_9),(p_8,t_{12}),(p_9,t_{13}),(p_{10},t_{14}),(p_{11},t_{15}),\\ &(p_{12},t_{16}),(p_{12},t_{17}),(p_{13},t_{18}),(p_{14},t_{19}),(p_{15},t_{20}),(p_{15},t_{21}),\\ &(p_{16},t_{24}),(p_{16},t_{25}),(p_{17},t_{22}),(p_{18},t_{23}),(p_{18},t_{26}),(p_{19},t_{27}),\\ &(p_{20},t_{28}),(p_{21},t_{29}),(p_{22},t_{30}),(p_{23},t_1),(p_{23},t_{15})\} \cup\\ &\{(t_1,p_2),(t_2,p_3),(t_3,p_4),(t_4,p_4),(t_5,p_5),(t_6,p_6),(t_7,p_7),(t_8,p_8),\\ &(t_9,p_6),(t_{10},p_7),(t_{11},p_1),(t_{11},p_9),(t_{11},p_{21}),(t_{12},p_1),(t_{12},p_9),\\ &(t_{12},p_{21}),(t_{13},p_{10}),(t_{14},p_9),(t_{15},p_{12}),(t_{16},p_{13}),(t_{17},p_{14}),\\ &(t_{18},p_{14}),(t_{19},p_{15}),(t_{20},p_{16}),(t_{21},p_{17}),(t_{22},p_{18}),(t_{23},p_{16}),(t_{24},p_{17}),\\ &(t_{25},p_{11}),(t_{25},p_{19}),(t_{25},p_{21}),(t_{26},p_{11}),(t_{26},p_{19}),(t_{26},p_{21}),(t_{27},p_{20}),\\ &(t_{28},p_{19}),(t_{29},p_{22}),(t_{30},p_{23})\}\end{aligned}$$

- $$\begin{aligned}&W(p_1,t_1)=1,W(p_2,t_2)=1,W(p_2,t_3)=1,W(p_3,t_4)=1,W(p_4,t_5)=1,\\ &W(p_5,t_6)=1,W(p_5,t_7)=1,W(p_6,t_{10})=1,W(p_6,t_{11})=1,W(p_7,t_8)=1,\\ &W(p_8,t_9)=1,W(p_8,t_{12})=1,W(p_9,t_{13})=1,W(p_{10},t_{14})=1,\\ &W(p_{11},t_{15})=1,W(p_{12},t_{16})=1,W(p_{12},t_{17})=1,W(p_{13},t_{18})=1,\\ &W(p_{14},t_{19})=1,W(p_{15},t_{20})=1,W(p_{15},t_{21})=1,W(p_{16},t_{24})=1,\\ &W(p_{16},t_{25})=1,W(p_{17},t_{22})=1,W(p_{18},t_{23})=1,W(p_{18},t_{26})=1,\\ &W(p_{19},t_{27})=1,W(p_{20},t_{28})=1,W(p_{21},t_{29})=1,W(p_{22},t_{30})=1,\\ &W(p_{23},t_1)=1,W(p_{23},t_{15})=1,W(t_1,p_2)=1,W(t_2,p_3)=1,W(t_3,p_4)=1,\\ &W(t_4,p_4)=1,W(t_5,p_5)=1,W(t_6,p_6)=1,W(t_7,p_7)=1,W(t_8,p_8)=1,W(t_9,p_6)=1,\\ &W(t_{10},p_7)=1,W(t_{11},p_1)=1,W(t_{11},p_9)=1,W(t_{11},p_{21})=1,W(t_{12},p_1)=1,\\ &W(t_{12},p_9)=1,W(t_{12},p_{21})=1,W(t_{13},p_{10})=1,W(t_{14},p_9)=1,W(t_{15},p_{12})=1,\\ &W(t_{16},p_{13})=1,W(t_{17},p_{14})=1,W(t_{18},p_{14})=1,W(t_{19},p_{15})=1,W(t_{20},p_{16})=1,\\ &W(t_{21},p_{17})=1,W(t_{22},p_{18})=1,W(t_{23},p_{16})=1,W(t_{24},p_{17})=1,W(t_{25},p_{11})=1,\\ &W(t_{25},p_{19})=1,W(t_{25},p_{21})=1,W(t_{26},p_{11})=1,W(t_{26},p_{19})=1,W(t_{26},p_{21})=1,\\ &W(t_{27},p_{20})=1,W(t_{28},p_{19})=1,W(t_{29},p_{22})=1,W(t_{30},p_{23})=1\}\end{aligned}$$

- $$\begin{aligned}D = \{&d(p_1)=1,d(p_2)=1,d(p_3)=3,d(p_4)=5,d(p_5)=4,\\ &d(p_6)=4,d(p_7)=7,d(p_8)=6,d(p_9)=3,d(p_{10})=5,\\ &d(p_{11})=1,d(p_{12})=1,d(p_{13})=3,d(p_{14})=5,d(p_{15})=4,\\ &d(p_{16})=4,d(p_{17})=7,d(p_{18})=6,d(p_{19})=3,d(p_{20})=5,\\ &d(p_{21})=1,d(p_{22})=3,d(p_{23})=4,d(p_{24})=3,d(p_{25})=3\}\end{aligned}$$

- $M_0 = [1,\ 1,0,0,0,0,0,0,0,0,1,1,0,0,0,0,0,0,0,0,1,0,0,0,0]^T$

În final se poate *concluziona* că structura aleasă pentru o retea Petri temporizata P corespunzătoare nodului de tip 4 este viabilă, ea putând fi implementată, fiind siguri că nu există condiţii de apariţie a blocajelor sau a apariţiei de situaţii necontrolabile.

2.5.2.5. Modelarea nodurilor cu doua intrari si doua iesiri

Nodurile complexe cu doua intrari si *doua* iesiri sunt construite cu ajutorul nodurilor definite anterior. Un astfel de nod are structura de principiu prezentata in figura 2.75.

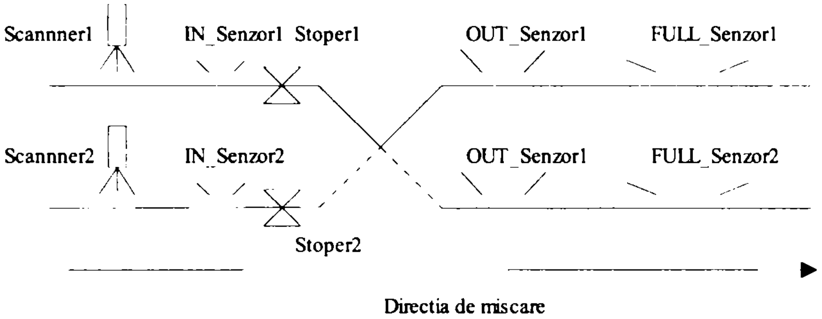


Figura 2.75. Structura de principiu a unui nod cu „doua" intrari si „doua" iesiri

2.5.2.5.1. Modelarea nodului cu doua intrari si doua iesiri cu ajutorul automatelor

Structura automatului secvential considerat pentru modelarea unui astfel de nod este prezentata în figura 2.76.

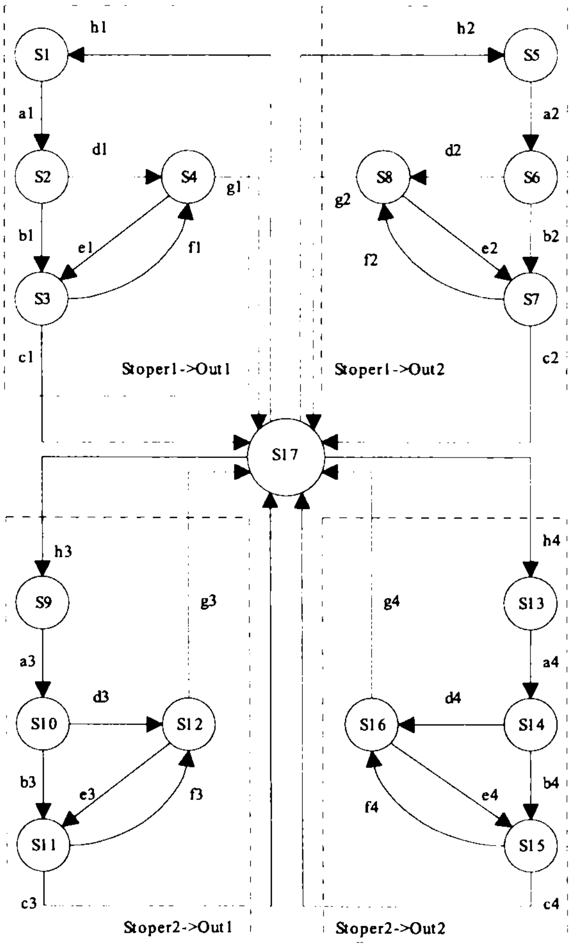


Figura 2.76. Structura automatului secvential corespunzător nodului cu doua intrari si doua iesiri

Dupa cum se poate observa, automatul secvential al nodului este realizat prin sincronizarea a patru automate secventiale *AS_N1*, cate doua pentru fiecare linie de intrare (stoper).

Automatul *AS_N2-2,* corespunzator nodului cu doua intrari si doua iesiri este descris astfel:

- multimea starilor:
  $X = \{S1, S2, S3, S4, S5, S6, S7, S8, S9, S10, S11, S12, S13, S14, S15, S16, S17\}$

- multimea evenimentelor:
  $\Sigma = \{a1, b1, c1, d1, e1, f1, g1, h1, a2, b2, c2, d2, e2, f2, g2, h2,$
  $a3, b3, c3, d3, e3, f3, g3, h3, a4, b4, c4, d4, e4, f4, g4, h4\}$
- multimile evenimentelor posibile si functiile de tranzitie a starilor:
  $\Gamma(S1) = \{a1\}$ $\delta(S1, a1) = S2$
  $\Gamma(S2) = \{b1, d1\}$ $\delta(S2, b1) = S3,$ $\delta(S2, d1) = S4$
  $\Gamma(S3) = \{c1, f1\}$ $\delta(S3, c1) = S17,$ $\delta(S3, f1) = S4$
  $\Gamma(S4) = \{e1, g1\}$ $\delta(S4, e1) = S3$ $\delta(S4, g1) = S17$
  $\Gamma(S5) = \{a2\}$ $\delta(S5, a2) = S6$
  $\Gamma(S6) = \{b2, d2\}$ $\delta(S6, b2) = S7,$ $\delta(S6, d2) = S8$
  $\Gamma(S7) = \{c2, f2\}$ $\delta(S7, c2) = S17,$ $\delta(S7, f2) = S8$
  $\Gamma(S8) = \{e2, g2\}$ $\delta(S8, e2) = S7$ $\delta(S8, g2) = S17$
  $\Gamma(S9) = \{a3\}$ $\delta(S9, a3) = S10$
  $\Gamma(S10) = \{b3, d3\}$ $\delta(S10, b3) = S11,$ $\delta(S10, d3) = S12$
  $\Gamma(S11) = \{c3, f3\}$ $\delta(S11, c3) = S17,$ $\delta(S11, f3) = S12$
  $\Gamma(S12) = \{e3, g3\}$ $\delta(S12, e3) = S11$ $\delta(S12, g3) = S17$
  $\Gamma(S13) = \{a4\}$ $\delta(S13, a4) = S14$
  $\Gamma(S14) = \{b4, d4\}$ $\delta(S14, b4) = S15,$ $\delta(S14, d4) = S16$
  $\Gamma(S15) = \{c4, f4\}$ $\delta(S15, c4) = S17,$ $\delta(S15, f4) = S16$
  $\Gamma(S16) = \{e4, g4\}$ $\delta(S16, e4) = S15$ $\delta(S16, g4) = S17$
  $\Gamma(S17) = \{h1, h2, h3, h4\}$ $\delta(S17, h1) = S1$ $\delta(S17, h2) = S5$
  $\delta(S17, h3) = S9$ $\delta(S17, h4) = S13$
- starea initiala: $x_0 = S17$.

Considerentele legate de transformarea modelului de tip automat in model de tip retea Petri, prezentate la nodul de tip 1 sunt valabile si in cazul nodului cu *n* intrari si o iesire.

Pentru conversia din model de tip automat in model de tip retea Petri s-a utilizat acceasi metoda ca si in cazul nodului de tip 1 .

Aplicand algoritmul 2.1 rezulta:

- multimea pozitiilor:
  $P = \{p1, p2, p3, p4, p5, p6, p7, p8, p9, p10, p11, p12, p13, p14, p15, p16, p17\}$,
  unde
  $p1 = \{S1\}, p2 = \{S2\}, p3 = \{S3\}, p4 = \{S4\}, p5 = \{S5\}, p6 = \{S6\}, p7 = \{S7\},$
  $p8 = \{S8\}, p9 = \{S9\}, p10 = \{S10\}, p11 = \{S11\}, p12 = \{S12\}, p13 = \{S13\},$
  $p14 = \{S14\}, p15 = \{S15\}, p16 = \{S16\}$ *si* $p17 = \{S17\}$

- multimea trranzitiilor:

• Pentru *p1*	*(p1,p2)*	*p2=δ(p1,t1)*	*t1={a1}*
• Pentru *p2*	*(p2,p3)*	*p3=δ(p2,t2)*	*t2={b1}*
	(p2,p4)	*p4=δ(p2,t4)*	*t4={d1}*
• Pentru *p3*	*(p3,p17)*	*p17=δ(p3,t3)*	*t3={c1}*
	(p3,p4)	*p4=δ(p3,t6)*	*t6={f1}*
• Pentru *p4*	*(p4,p17)*	*p17=δ(p4,t7)*	*t7={g1}*
	(p4,p3)	*p3=δ(p4,t5)*	*t5={e1}*
• Pentru *p5*	*(p5,p6)*	*p6=δ(p5,t9)*	*t9={a2}*
• Pentru *p6*	*(p6,p7)*	*p7=δ(p6,t10)*	*t10={b2}*
	(p6,p8)	*p8=δ(p6,t12)*	*t12={d2}*
• Pentru *p7*	*(p7,p17)*	*p17=δ(p7,t11)*	*t11={c2}*
	(p7,p8)	*p8=δ(p7,t14)*	*t14={f2}*
• Pentru *p8*	*(p8,p17)*	*p17=δ(p8,t15)*	*t15={g2}*
	(p8,p7)	*p7=δ(p8,t13)*	*t13={e2}*
• Pentru *p9*	*(p9,p10)*	*p10=δ(p9,t17)*	*t17={a2}*
• Pentru *p10*	*(p10,p11)*	*p11=δ(p10,t18)*	*t18={b2}*
	(p10,p12)	*p12=δ(p10,t20)*	*t20={d2}*
• Pentru *p11*	*(p11,p17)*	*p17=δ(p11,t19)*	*t19={c2}*
	(p11,p12)	*p12=δ(p11,t22)*	*t22={f2}*
• Pentru *p12*	*(p12,p17)*	*p17=δ(p12,t23)*	*t23={g2}*
	(p12,p11)	*p11=δ(p12,t21)*	*t21={e2}*
• Pentru *p13*	*(p13,p14)*	*p14=δ(p13,t25)*	*t25={a2}*
• Pentru *p14*	*(p14,p15)*	*p15=δ(p14,t26)*	*t26={b2}*
	(p14,p16)	*p16=δ(p14,t28)*	*t28={d2}*
• Pentru *p15*	*(p15,p17)*	*p17=δ(p15,t27)*	*t27={c2}*
	(p15,p16)	*p16=δ(p15,t30)*	*t30={f2}*
• Pentru *p16*	*(p16,p17)*	*p17=δ(p16,t31)*	*t31={g2}*
	(p16,p15)	*p15=δ(p16,t29)*	*t29={e2}*
• Pentru *p17*	*(p17,p1)*	*p1=δ(p17,t8)*	*t8={h1}*
	(p17,p5)	*p5=δ(p17,t16)*	*t16={h2}*
	(p17,S9)	*p9=δ(p17,t24)*	*t24={h3}*
	(p17,S13)	*p13=δ(p17,t32)*	*t32={h4}*

Modelarea automatului secvential cu ajutorul retelelor Petri se efectueaza pe utilizand submodele.

In figura 2.77 se prezinta structura submodelului retelei Petri asociata unei directii de miscare (Stoper1->Out1, Stoper1->Out2 Stoper2->Out1 respectiv Stoper2->Out2)

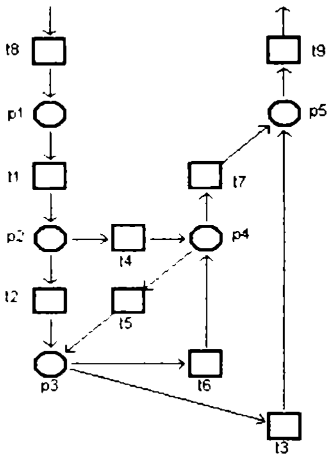


Figura 2.77. Structura reţelei Petri asociata submodelului corespunzator unei directii

Dupa cum se poate observa, submodelul ales este similar cu cel prezentat in cazul modelarii celorlalte tipuri de noduri, cu observatia ca apare in plus pozitia de unificare a iesirilor tranzitilor t_3 si t_7, *p_sinc,* deoarece ambele tranzitii au ca destinatie aceeasi pozitie p_{17}.

Analiza structurala si comportamentala a submodelului ales se face introducand pozitia care contine marcajul initial p_{17}. Structura retelei Petri astfel obtinuta este prezentata in figura 2.78.

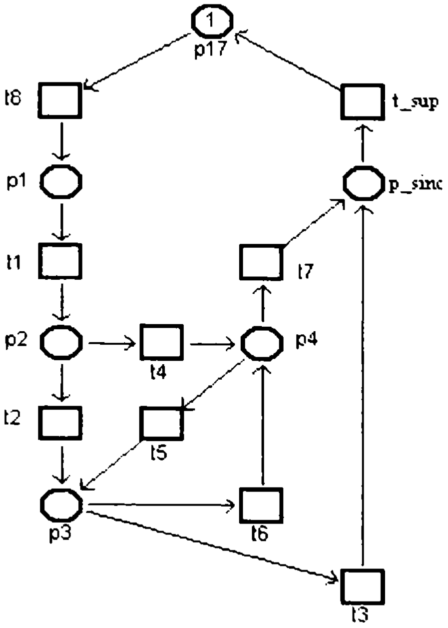


Figura 2.78. Structura reţelei Petri asociata analizei submodelului

Topologia retelei validata de PNT este prezentata in figura 2.79.

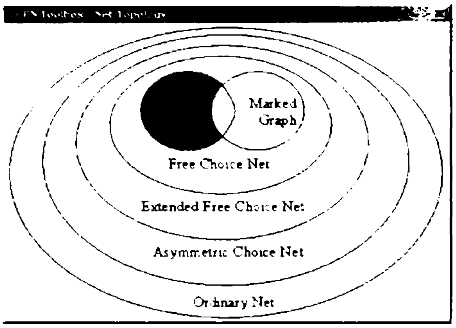


Figura 2.79. Topologia retelei Petri echivalenta validata de PNT

Reteaua Petri considerata pentru analiza submodelului este formalizata prin cvintuplul:

$$PN_{SUB}\_AS\_N2-2=(P,T,F,W,M_0)$$

unde:

- $P=\{p_1,p_2,p_3,p_4,p_{17}\}\cup\{p\_sinc\}$
- $T=\{t_1,t_2,t_3,t_4,t_5,t_6,t_7,t_8\}\cup\{t\_sup)$
- $F=\{\{(p_1,t_1),(p_2,t_2),(p_2,t_4),(p_3,t_6),(p_3,t_3),(p_4,t_5),(p_4,t_7),(p_{17},t_8),$ $(p\_sinc,t\_sup)\}\cup\{(t_1,p_2),(t_2,p_3),(t_3,p\_sinc),(t_4,p_4),(t_5,p_3),$ $(t_6,p_4),(t_7,p\_sinc),(t_8,p_1),(t\_sup,p_{17})\}\}$
- $W(p_1,t_1)=1,W(p_2,t_2)=1,W(p_2,t_4)=1,W(p_3,t_6)=1,W(p_3,t_3)=1,$ $W(p_4,t_5)=1,W(p_4,t_7)=1,W(p_{17},t_8)=1,W(p\_sinc,t\_sup)=1,$ $W(t_1,p_2)=1,W(t_2,p_3)=1,W(t_3,p\_sinc)=1,W(t_4,p_4)=1,$ $W(t_5,p_3)=1,W(t_6,p_4)=1,W(t_7,p\_sinc)=1,$ $W(t_8,p_1)=1,W(t\_sup,p_{17})=1$
- $M_0=[0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,1]^T$

Matricile de incidenţă sunt.

Matricea de incidenta de intrare:

$$A_{Ii}=\begin{pmatrix}1&0&0&0&0&0\\0&1&0&0&0&0\\0&0&1&0&0&0\\0&1&0&0&0&0\\0&0&0&1&0&0\\0&0&1&0&0&0\\0&0&0&1&0&0\\0&0&0&0&0&1\\0&0&0&0&1&0\end{pmatrix}$$

Matricea de incidenta de iesire:

$$A_{Oi}=\begin{pmatrix}0&1&0&0&0&0\\0&0&1&0&0&0\\0&0&0&0&1&0\\0&0&0&1&0&0\\0&0&1&0&0&0\\0&0&0&1&0&0\\0&0&0&0&1&0\\1&0&0&0&0&0\\0&0&0&0&0&1\end{pmatrix}$$

Matricea de incidenta:

$$A_i = A_{Oi} - A_{Ii} = \begin{pmatrix} -1 & 1 & 0 & 0 & 0 & 0 \\ 0 & -1 & 1 & 0 & 0 & 0 \\ 0 & 0 & -1 & 0 & 1 & 0 \\ 0 & -1 & 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & -1 & 0 & 0 \\ 0 & 0 & -1 & 1 & 0 & 0 \\ 0 & 0 & 0 & -1 & 1 & 0 \\ 1 & 0 & 0 & 0 & 0 & -1 \\ 0 & 0 & 0 & 0 & -1 & 1 \end{pmatrix}$$

Arborele de acoperire corespunzator este prezentat in figura 2.80.

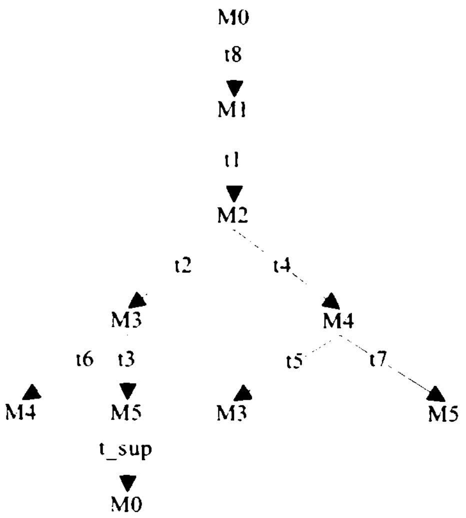


Figura 2.80. Arborele de acoperire al retelei Petri PN_{SUB}_AS_N2-2

Unde:
M0=(0,0,0,0,0,1); M1=(1,0,0,0,0,0); M2=(0,1,0,0,0,0); M3=(0,0,1,0,0,0);
M4=(0,0,0,1,0,0); M5=(0,0,0,0,1,0).

Studiind proprietatile comportamentale ale retelei Petri PN_{SUB}_AS_N2-2, pe baza arborelui de acoperire rezulta:

- reteaua este *marginita* (simbolul ω nu apare in arborele de acoperire);
- reteaua este *sigura* (marcajele din toate nodurile arborelui de acoperire contin numai „0" si „1");
- reteaua *nu este blocanta* (toate tranzitile au asociate arce in arborele de acoperire);
- reteaua este *accesibila* (orice marcaj poate fi atins pornind din M_0).

Graful de acoperire corespunzator este prezentat in figura 2.81.

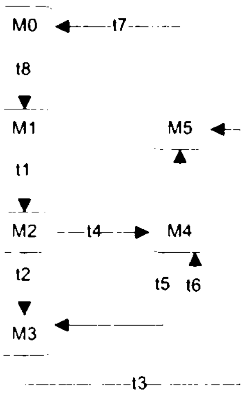


Figura 2.81. Graful de acoperire al retelei Petri PN_{SUB}_AS_N2-2

Structura rezultata pentru graful de acoperire confirma ca submodelul de retea Petri derivata din modelul de tip automat corespunzator nodului cun *doua* intrari si *doua* iesiri este acesibila.

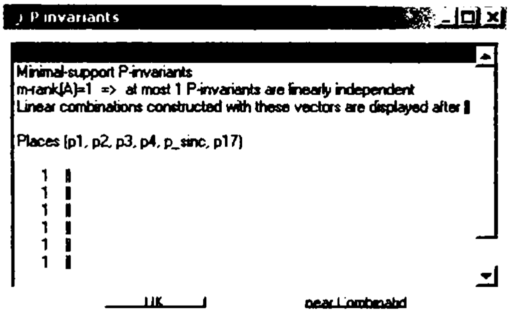
Concluzionand, se poate afirma ca submodelul ales este viabil din punct de vedere comportamental.

Analiza retelei Petri PN_{SUB}_AS_N2-2 din punct de vedere structural se face pe baza analizei existentei invariantilor de tip *P* respectiv *T*.

Pentru determinarea numarului de invarianti *P* respectiv *T* s-a aplicat teorema 2.3 (Determinarea numarului de invarianti).Rangul matricei A este: $rang\,A = 5$

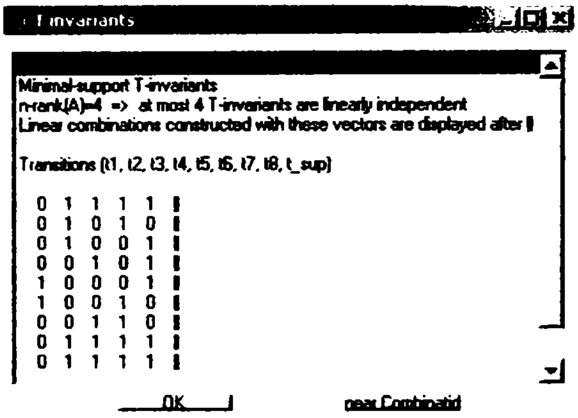
Rezulta ca reteaua Petri PN_{SUB}_AS_N2-2 este acoperita de 1 invariant de tip *P*, respectiv 4 invarianti de tip *T*.

In figurile 2.82 si 2.83 se prezinta invariantii de tip P respectiv T obtinuti in urma simularii retelei Petri PN_{SUB}_AS_N2-2 cu ajutorul PNT-ului.



```
Minimal-support P-invariants
m-rank(A)=1  =>  at most 1 P-invariants are linearly independent
Linear combinations constructed with these vectors are displayed after

Places (p1, p2, p3, p4, p_sinc, p17)

    1
    1
    1
    1
    1
    1
```

Figura 2.82. Invariantii de tip P obtinuti cu ajutorul PNT



```
Minimal-support T-invariants
n-rank(A)=4  =>  at most 4 T-invariants are linearly independent
Linear combinations constructed with these vectors are displayed after ||

Transitions (t1, t2, t3, t4, t5, t6, t7, t8, t_sup)

  0  1  1  1  1  |
  0  1  0  1  0  |
  0  1  0  0  1  |
  0  0  1  0  1  |
  1  0  0  0  1  |
  1  0  0  1  0  |
  0  0  1  1  0  |
  0  1  1  1  1  |
  0  1  1  1  1  |
```

Figura 2.83. Invariantii de tip T obtinuti cu ajutorul PNT

Conform teoremelor 2.6 respectiv 2.7, reteaua Petri PN_{SUB}_AS_N2-2 este *conservativa si structural marginita, respectiv este consistenta si repetitiva*.

In acest moment, se poate *concluziona* că structura aleasă pentru submodelul ales corespunzator nodului cu doua intrari si doua iesiri este *viabilă*, ea putând fi utilizata in modelarea intregului nod, fiind siguri că nu există condiţii de apariţie a blocajelor sau a apariţiei de situaţii necontrolabile.

Reteaua Petri rezultata prin utilizarea submodelului definit anterior este prezentata in figura 2.84.

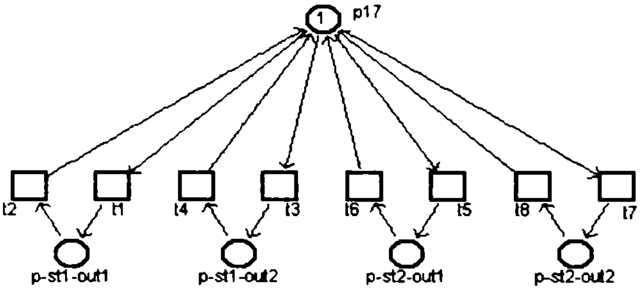


Figura 2.84. Reteaua Petri corespunzatoare modelarii nodului doua intrari doua iesiri derivata din automatul secvential *AS_N2-2*

Topologia retelei validata de PNT este prezentata in figura 2.85.

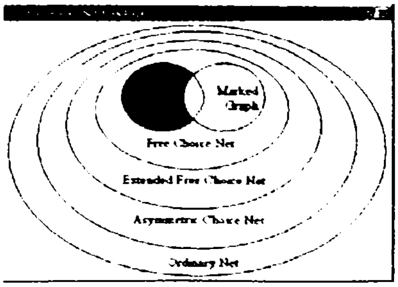


Figura 2.85. Topologia retelei Petri echivalenta validata de PNT

Reteaua Petri considerata pentru analiza submodelului este formalizata prin cvintuplul:

$$PN\_AS\_N2-2=(P,T,F,W,M_0)$$

unde:

- $P=\{p-st1-out1, p-st2-out2, p-st2-out1, p-st2-out2, p_{17}\}$
- $T=\{t_1,t_2,t_3,t_4,t_5,t_6,t_7,t_8\}$
- $$\begin{aligned}F=\{&\{(p-st1-out1,t_2),(p-st1-out2,t_4),(p-st2-out1,t_6),\\&(p-st2-out2,t_8),(p_{17},t_1),(p_{17},t_3),(p_{17},t_5),(p_{17},t_7)\}\cup\\&\{(t_1,p-st1-out1),(t_2,p_{17}),(t_3,p-st1-out2),(t_4,p_{17}),\\&(t_5,p-st2\_out1),(t_6,p_{17}),(t_7,p-st2-out2),(t_8,p_{17})\}\}\end{aligned}$$
- $$\begin{aligned}&W(p-st1-out1,t_2)=1, W(p-st1-out2,t_4)=1,\\&W(p-st2-out1,t_6)=1, W(p-st2-out2,t_8)=1,\\&W(p_{17},t_1)=1, W(p_{17},t_3)=1, W(p_{17},t_5)=1, W(p_{17},t_7)=1,\\&W(t_1,p-st1-out1)=1, W(t_2,p_{17})=1, W(t_3,p-st1-out2)=1,\\&W(t_4,p_{17})=1, W(t_5,p-st2\_out1)=1, W(t_6,p_{17})=1,\\&W(t_7,p-st2-out2)=1, W(t_8,p_{17})=1\end{aligned}$$
- $M_0=[0,0,0,0,1]^T$

Matricile de incidenţă sunt.

Matricea de incidenta de intrare:

$$A_{Ii}=\begin{pmatrix}0&0&0&0&1\\1&0&0&0&0\\0&0&0&0&1\\0&1&0&0&0\\0&0&0&0&1\\0&0&1&0&0\\0&0&0&0&1\\0&0&0&1&0\end{pmatrix}$$

Matricea de incidenta de iesire:

$$A_{Oi}=\begin{pmatrix}1&0&0&0&0\\0&0&0&0&1\\0&1&0&0&0\\0&0&0&0&1\\0&0&1&0&0\\0&0&0&0&1\\0&0&0&1&0\\0&0&0&0&1\end{pmatrix}$$

Matricea de incidenta:

$$A_i=A_{Oi}-A_{Ii}=\begin{pmatrix}1&0&0&0&-1\\-1&0&0&0&1\\0&1&0&0&-1\\0&-1&0&0&1\\0&0&1&0&-1\\0&0&-1&0&1\\0&0&0&1&-1\\0&0&0&-1&1\end{pmatrix}$$

Arborele de acoperire corespunzator este prezentat in figura 2.86.

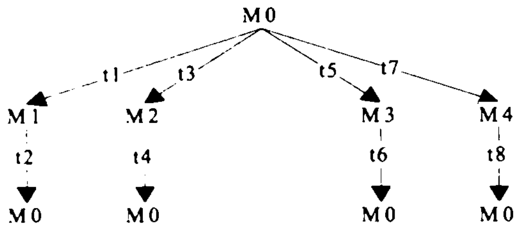


Figura 2.86. Arborele de acoperire al retelei Petri *PN$_{SUB}$_AS_N2-2*

Unde:
M0=(0,0,0,0,1); M1=(1,0,0,0,0); M2=(0,1,0,0,0); M3=(0,0,1,0,0); M4=(0,0,0,1,0).

Studiind proprietatile comportamentale ale retelei Petri *PN_AS_N2-2*, pe baza arborelui de acoperire rezulta:

- reteaua este *marginita* (simbolul ω nu apare in arborele de acoperire);
- reteaua este *sigura* (marcajele din toate nodurile arborelui de acoperire contin numai „0” si „1”);
- reteaua *nu este blocanta* (toate tranzitile au asociate arce in arborele de acoperire);
- reteaua este *accesibila* (orice marcaj poate fi atins pornind din M_0).

Graful de acoperire corespunzator este prezentat in figura 2.87.

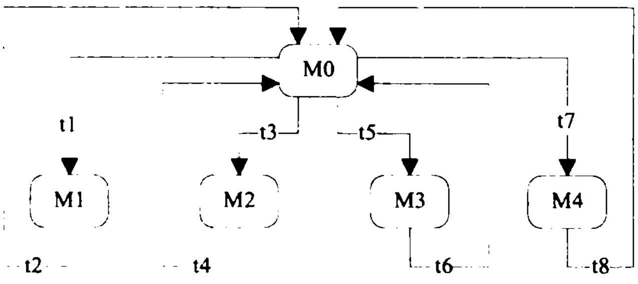


Figura 2.87. Graful de acoperire al retelei Petri *PN_AS_N2-2*

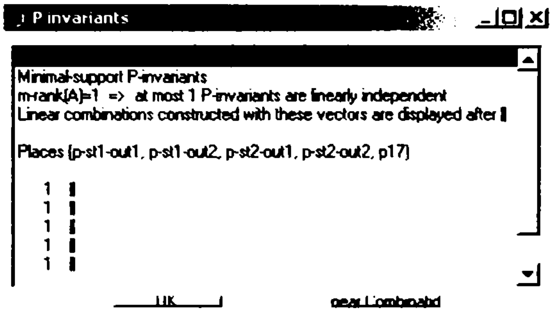
Structura rezultata pentru graful de acoperire confirma ca reteaua Petri obtinuta cu ajutorul submodelelor corespunzatoare automatului secvential al nodului cun *doua* intrari si *doua* iesiri este *acesibila*.

Concluzionand, se poate afirma ca modelul ales este *viabil* din punct de vedere comportamental.

Analiza retelei Petri *PN_AS_N2-2* din punct de vedere structural se face pe baza analizei existentei invariantilor de tip *P* respectiv *T*.

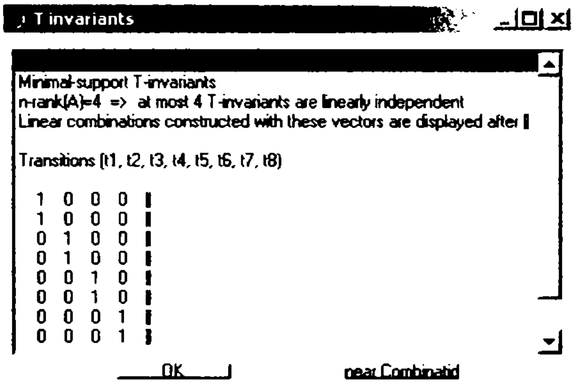
Pentru determinarea numarului de invarianti *P* respectiv *T* , aplicand aceasi procedura ca si in cazul anterior a rezultat $rang\,A = 5$, reteaua Petri *PN$_{SUB}$_AS_N2-2* fiind acoperita de 1 invariant de tip *P*, respectiv de 4 invarianti de tip *T*.

In figurile 2.88 si 2.89 se prezinta invariantii de tip P respectiv T obtinuti in urma simularii retelei Petri *PN_AS_N2-2* cu ajutorul PNT-ului.



```
Minimal-support P-invariants
m-rank(A)=1  =>  at most 1 P-invariants are linearly independent
Linear combinations constructed with these vectors are displayed after |

Places (p-st1-out1, p-st1-out2, p-st2-out1, p-st2-out2, p17)

  1  |
  1  |
  1  |
  1  |
  1  |
```

Figura 2.88. Invariantii de tip P obtinuti cu ajutorul PNT



```
Minimal-support T-invariants
n-rank(A)=4  =>  at most 4 T-invariants are linearly independent
Linear combinations constructed with these vectors are displayed after |

Transitions (t1, t2, t3, t4, t5, t6, t7, t8)

  1  0  0  0  |
  1  0  0  0  |
  0  1  0  0  |
  0  1  0  0  |
  0  0  1  0  |
  0  0  1  0  |
  0  0  0  1  |
  0  0  0  1  |
```

Figura 2.89. Invariantii de tip T obtinuti cu ajutorul PNT

Conform teroemelor 2.6 respectiv 2.7 reteaua Petri *PN_AS_N2-2* este *conservativa si structural marginita, respectiv este consistenta si repetitiva.*

În final se poate *concluziona* că structura aleasă pentru un automat secvential clasic corespunzător nodului cu doua intrari si doua iesiri este viabilă, ea putând fi implementată, fiind siguri că *nu există condiţii de apariţie a blocajelor sau* a apariţiei de situaţii necontrolabile.

2.5.2.5.2. Modelarea nodului cu doua intrari si doua iesiri cu ajutorul retelelor Petri [UP06b][Ung06c]

2.5.2.5.2.1. Cazul: retea Petri netemporizata

Structura retelei Petri considerate pentru modelarea unui atfel de nod prezentata in figura 2.90.

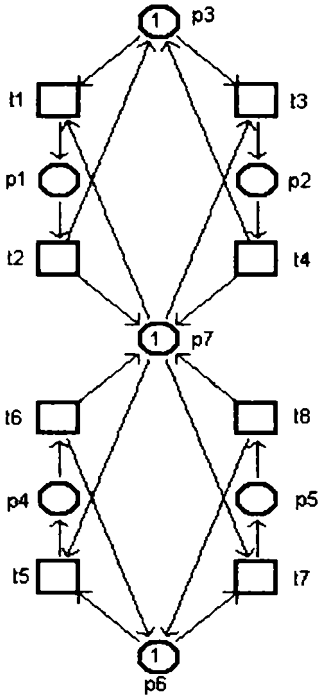


Figura 2.90. Structura retelei Petri corespunzatoare nodului cu doua intrari si doua iesiri

In cazul retelei rezultate pozitiile p_1, p_2, p_4 si p_5 reprezinta submodelul ales pentru controlul unei directii dintr-un stoper. Pozitiile p_3 si p_6 reprezinta subretelele alese pentru modelarea scanerelor iar pozitia p_7 este pozitia de sincronizare. Structurile si analizele corespunzatoare subretelelor (a submodelelor) vor fi prezentate dupa analiza retelei din figura 2.79 .

Astfel, topologia retelei Petri corespunzatoare modelarii nodului cu doua intrari si doua iesiri validata de PNT este prezentata in figura 2.91.

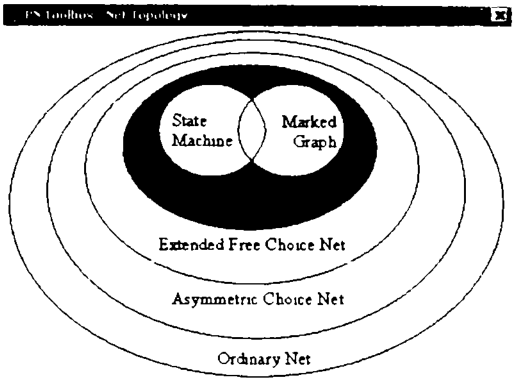


Figura 2.91. Topologia retelei Petri validata de TPN

Reteaua Petri considerata este formalizata prin cvintuplul:

$$PN\_PT\_N2-2=(P,T,F,W,M_0)$$

unde:

- $P = \{p_1, p_2, p_3, p_4, p_5, p_6, p_7\}$
- $T = \{t_1, t_2, t_3, t_4, t_5, t_6, t_7, t_8\}$
- $F = \{(p_1,t_2),(p_2,t_4),(p_3,t_1),(p_3,t_3),(p_4,t_6),(p_5,t_8),(p_6,t_5),(p_6,t_7),$ $(p_7,t_1),(p_7,t_3),(p_7,t_5),(p_7,t_7)\} \cup$ $\{(t_1,p_1),(t_2,p_3),(t_2,p_7),(t_3,p_2),(t_4,p_3),(t_4,p_7),(t_5,p_4),(t_6,p_6),$ $(t_6,p_7),(t_7,p_5),(t_8,p_6),(t_8,p_7)\}$
- $W(p_1,t_2) = 1, W(p_2,t_4) = 1, W(p_3,t_1) = 1, W(p_3,t_3) = 1, W(p_4,t_6) = 1,$ $W(p_5,t_8) = 1, W(p_6,t_5) = 1, W(p_6,t_7) = 1, W(p_7,t_1) = 1, W(p_7,t_3) = 1,$ $W(p_7,t_5) = 1, W(p_7,t_7) = 1, W(t_1,p_1) = 1, W(t_2,p_3) = 1, W(t_2,p_7) = 1,$ $W(t_3,p_2) = 1, W(t_4,p_3) = 1, W(t_4,p_7) = 1, W(t_5,p_4) = 1, W(t_6,p_6) = 1,$ $W(t_6,p_7) = 1, W(t_7,p_5) = 1, W(t_8,p_6) = 1, W(t_8,p_7) = 1$
- $M_0 = [\,0,0,1,0,0,1,1]^T$

Matricile de incidenta sunt:

Matricea de incidenta de intrare:

$$A_{Ii} = \begin{pmatrix} 0 & 0 & 1 & 0 & 0 & 0 & 1 \\ 1 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 & 0 & 1 \\ 0 & 1 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 1 & 1 \\ 0 & 0 & 0 & 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 1 & 1 \\ 0 & 0 & 0 & 0 & 1 & 0 & 0 \end{pmatrix}$$

Matricea de incidenta de iesire:

$$A_{Oi} = \begin{pmatrix} 1 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 & 0 & 1 \\ 0 & 1 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 & 0 & 1 \\ 0 & 0 & 0 & 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 1 & 1 \\ 0 & 0 & 0 & 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 1 & 1 \end{pmatrix}$$

Matricea de incidenta:

$$A_i = A_{Oi} - A_{Ii} = \begin{pmatrix} 1 & 0 & -1 & 0 & 0 & 0 & -1 \\ -1 & 0 & 1 & 0 & 0 & 0 & 1 \\ 0 & 1 & -1 & 0 & 0 & 0 & -1 \\ 0 & -1 & 1 & 0 & 0 & 0 & 1 \\ 0 & 0 & 0 & 1 & 0 & -1 & -1 \\ 0 & 0 & 0 & -1 & 0 & 1 & 1 \\ 0 & 0 & 0 & 0 & 1 & -1 & -1 \\ 0 & 0 & 0 & 0 & -1 & 1 & 1 \end{pmatrix}$$

Arborele de acoperire corespunzator retelei Petri *PN_PT_N2-2* este prezentat in figura 2.92.

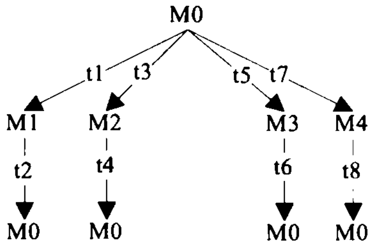


Figura 2.92. Arborele de acoperire corespunzator *PN_PT_N2-2*

Unde:
M0=(0,0,1,0,0,1,1); M1=(1,0,0,0,0,1,0); M2=(0,1,0,0,0,1,0);
M3=(1,0,0,1,0,0,0);
M4=(1,0,0,0,1,0,0).

Studiind proprietatile comportamentale ale retelei Petri *PN_PT_N2-2*, pe baza arborelui de acoperire rezulta:

- reteaua este *marginita* (simbolul ω nu apare in arborele de acoperire);
- reteaua este *sigura* (marcajele din toate nodurile arborelui de acoperire contin numai „0" si „1");
- reteaua *nu este blocanta* (toate tranzitile au asociate arce in arborele de acoperire);
- reteaua este *accesibila* (orice marcaj poate fi atins pornind din M_0).

Graful de acoperire corespunzator este prezentat in figura 2.93.

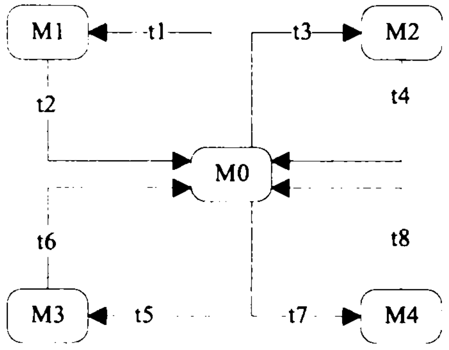


Figura 2.93. Graful de acoperire al retelei Petri *PN_PT_N2-2*

Structura rezultata pentru graful de acoperire confirma ca modelul de retea Petri corespunzator nodului cu doua intrari si doua iesiri este *acesibil*.

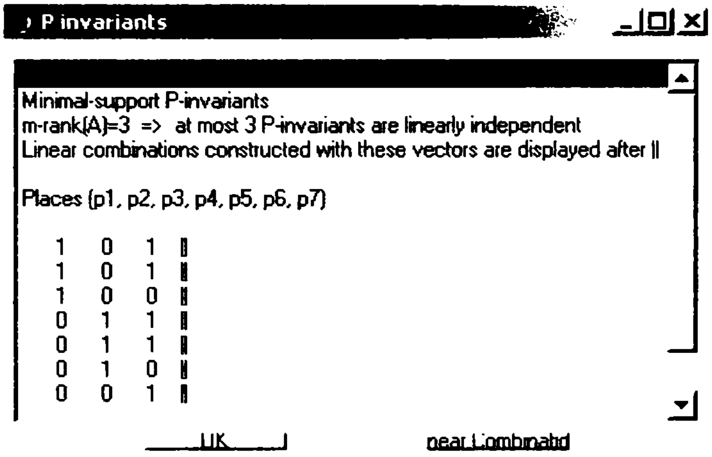
Concluzionand, se poate afirma ca modelul ales este *viabil* din punct de vedere comportamental.

Analiza retelei Petri *PN_PT_N2-2* din punct de vedere structural se face pe baza analizei existentei invariantilor de tip *P* respectiv *T*.

Pentru determinarea numarului de invarianti *P* respectiv *T* s-a aplicat teorema 2.3, rangul matricei A fiind $rang\,A = 4$.

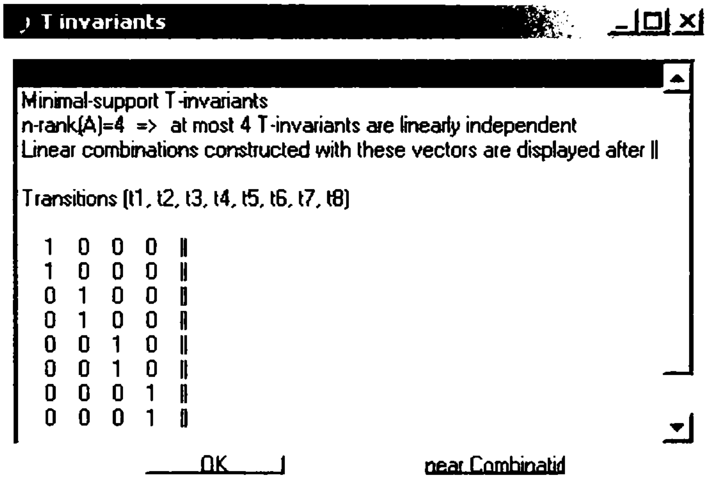
Rezulta ca reteaua Petri *PN_PT_N2-2* este acoperita de 3 invarianti de tip *P*, respectiv 4 invarianti de tip *T*.

In figurile 2.94 si 2.95 se prezinta invariantii de tip *P* respectiv *T* obtinuti in urma simularii retelei Petri *PN_PT_N2-2* cu ajutorul PNT-ului.



```
Minimal-support P-invariants
m-rank(A)=3 => at most 3 P-invariants are linearly independent
Linear combinations constructed with these vectors are displayed after ||

Places (p1, p2, p3, p4, p5, p6, p7)

  1  0  1 ||
  1  0  1 ||
  1  0  0 ||
  0  1  1 ||
  0  1  1 ||
  0  1  0 ||
  0  0  1 ||
```

Figura 2.94. Invariantii de tip P obtinuti cu ajutorul PNT



```
Minimal-support T-invariants
n-rank(A)=4 => at most 4 T-invariants are linearly independent
Linear combinations constructed with these vectors are displayed after ||

Transitions (t1, t2, t3, t4, t5, t6, t7, t8)

  1  0  0  0 ||
  1  0  0  0 ||
  0  1  0  0 ||
  0  1  0  0 ||
  0  0  1  0 ||
  0  0  1  0 ||
  0  0  0  1 ||
  0  0  0  1 ||
```

Figura 2.95. Invariantii de tip T obtinuti cu ajutorul PNT

Conform teroemelor 2.6 respectiv 2.7 reteaua Petri *PN_PT_N2-2* este *conservativa si structural marginita, respectiv este consistenta si repetitiva.*

Pentru validarea in totalitate a rezultatelor obtinute in continuare se analizeaza submodelele utilizate.

Conform cu cele prezentate mai sus, au fost stabilite doua tipuri de submodele: unul pentru modelarea unei directii de miscare si altul pentru modelarea operatiei de identificare (scanare).

Submodelul unei directii de miscare

Submodelul ales pentru modelarea unei directii de miscare este prezentat in figura 2.96.

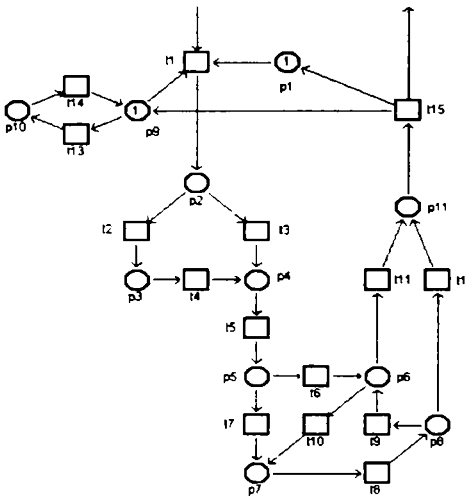


Figura 2.96. Submodelul de tip retea Petri pentru o directie de miscare

Dupa cum se poate observa, submodelul reprezinta varianta „clasica" de modelare a unui nod de tip 1 (a unei directii de miscare), avand in plus o pozitie de unificare a tranzitilor t_{11} si t_{12} , p_{11} si o tranzitie finala t_{15}, fapt care nu schimba cu nimic functionarea si topologia retelei.

Analiza submodelului propus se face cu ajutorul retelei din figura 2.97.

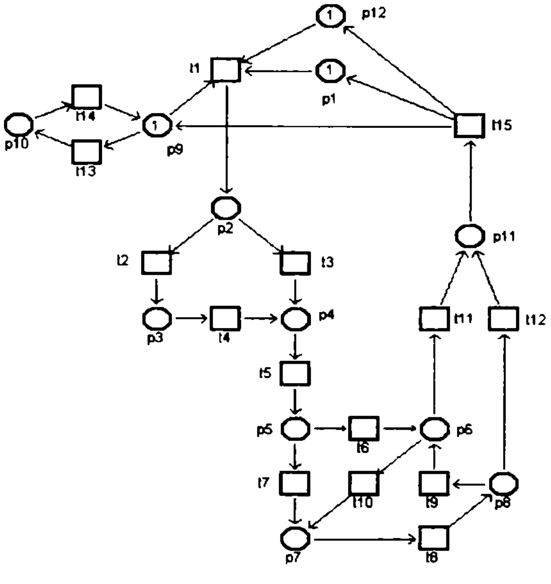


Figura 2.97. Reteaua Petri utilizata in analiza submodelului corespunzator unei directii

Se observa ca in plus fata de submodelul ales (figura 2.96) apar pozitia p_{12}, care este conectata exact ca si pozitia p1. Rolul acestei pozitii suplimentare este acela de a simula „exteriorul" submodelului.

Topologia retelei Petri corespunzatoare submodelului stabilit pentru o directie de miscare validata de PNT este prezentata in figura 2.98.

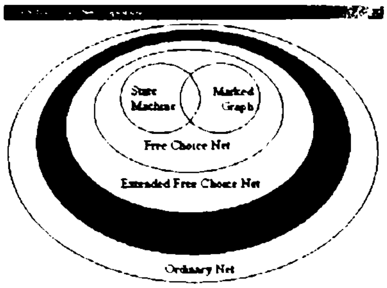


Figura 2.98. Topologia retelei Petri validata de TPN

Reteaua Petri considerata este formalizata prin cvintuplul:

$$PN_{SUB}\_PT_{DIR}\_N2-2=(P,T,F,W,M_0)$$

unde:

- $P=\{p_1,p_2,p_3,p_4,p_5,p_6,p_7,p_8,p_9,p_{10},p_{11},p_{12}\}$
- $T=\{t_1,t_2,t_3,t_4,t_5,t_6,t_7,t_8,t_9,t_{10},t_{11},t_{12},t_{13},t_{14},t_{15}\}$
- $F=\{(p_1,t_1),(p_2,t_2),(p_2,t_3),(p_3,t_4),(p_4,t_5),(p_5,t_6),(p_5,t_7),$ $(p_6,t_{10}),(p_6,t_{11}),(p_7,t_8),(p_8,t_9),(p_8,t_{12}),(p_9,t_{13}),$ $(p_{10},t_{14}),(p_{11},t_{15}),(p_{12},t_1)\}\cup$ $\{(t_1,p_2),(t_2,p_3),(t_3,p_4),(t_4,p_4),(t_5,p_5),(t_6,p_6),(t_7,p_7),$ $(t_8,p_8),(t_9,p_6),(t_{10},p_7),(t_{11},p_{11}),(t_{12},p_{11}),(t_{13},p_{10}),$ $(t_{14},p_9),(t_{15},p_1),(t_{15},p_9),(t_{15},p_{12})\}$
- $W(p_1,t_1)=1,W(p_2,t_2)=1,W(p_2,t_3)=1,W(p_3,t_4)=1,W(p_4,t_5)=1,$ $W(p_5,t_6)=1,W(p_5,t_7)=1,W(p_6,t_{10})=1,W(p_6,t_{11})=1,W(p_7,t_8)=1,$ $W(p_8,t_9)=1,W(p_8,t_{12})=1,W(p_9,t_{13})=1,W(p_{10},t_{14})=1,$ $W(p_{11},t_{15})=1,W(p_{12},t_1)=1W(t_1,p_2)=1,W(t_2,p_3)=1,W(t_3,p_4)=1,$ $W(t_4,p_4)=1,W(t_5,p_5)=1,W(t_6,p_6)=1,W(t_7,p_7)=1,W(t_8,p_8)=1,$ $W(t_9,p_6)=1,W(t_{10},p_7)=1,W(t_{11},p_{11})=1,W(t_{12},p_{11})=1,$ $W(t_{13},p_{10})=1,W(t_{14},p_9)=1,W(t_{15},p_1)=1,W(t_{15},p_9)=1,W(t_{15},p_{12})=1$
- $M_0=[\,1,0,0,0,0,0,0,0,1,0,0,1\,]^T$

Matricile de incidenta sunt:

Matricea de incidenta de intrare:

$$A_{II}-\begin{bmatrix}1&0&0&0&0&0&0&0&1&0&0&1\\0&1&0&0&0&0&0&0&0&0&0&0\\0&1&0&0&0&0&0&0&0&0&0&0\\0&0&1&0&0&0&0&0&0&0&0&0\\0&0&0&1&0&0&0&0&0&0&0&0\\0&0&0&0&1&0&0&0&0&0&0&0\\0&0&0&0&1&0&0&0&0&0&0&0\\0&0&0&0&0&0&1&0&0&0&0&0\\0&0&0&0&0&0&0&1&0&0&0&0\\0&0&0&0&0&1&0&0&0&0&0&0\\0&0&0&0&0&1&0&0&0&0&0&0\\0&0&0&0&0&0&0&1&0&0&0&0\\0&0&0&0&0&0&0&0&1&0&0&0\\0&0&0&0&0&0&0&0&0&1&0&0\\0&0&0&0&0&0&0&0&0&0&1&0\end{bmatrix}$$

Matricea de incidenta de iesire:

$$A_{OI}=\begin{bmatrix}0&1&0&0&0&0&0&0&0&0&0&0\\0&0&1&0&0&0&0&0&0&0&0&0\\0&0&0&1&0&0&0&0&0&0&0&0\\0&0&0&1&0&0&0&0&0&0&0&0\\0&0&0&0&1&0&0&0&0&0&0&0\\0&0&0&0&0&1&0&0&0&0&0&0\\0&0&0&0&0&0&1&0&0&0&0&0\\0&0&0&0&0&0&0&1&0&0&0&0\\0&0&0&0&0&1&0&0&0&0&0&0\\0&0&0&0&0&0&1&0&0&0&0&0\\0&0&0&0&0&0&0&0&0&0&1&0\\0&0&0&0&0&0&0&0&0&0&1&0\\0&0&0&0&0&0&0&0&0&1&0&0\\0&0&0&0&0&0&0&0&1&0&0&0\\1&0&0&0&0&0&0&0&1&0&0&1\end{bmatrix}$$

Matricea de incidenta:

$$A_i = A_{Oi} - A_{Ii} = \begin{pmatrix} -1 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & -1 & 0 & 0 & -1 \\ 0 & -1 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & -1 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & -1 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & -1 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & -1 & 1 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & -1 & 0 & 1 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & -1 & 1 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 1 & 0 & -1 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & -1 & 1 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & -1 & 0 & 0 & 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & -1 & 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & -1 & 1 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & -1 & 0 & 0 \\ 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & -1 & 1 \end{pmatrix}$$

Arborele de acoperire corespunzator retelei Petri *PN_{SUB}_PT_{DIR}_N2-2* este prezentat in figura 2.99.

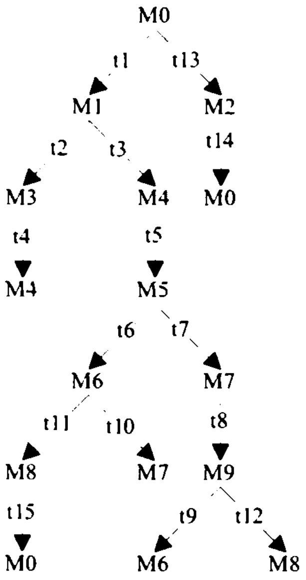


Figura 2.99. Arborele de acoperire corespunzator *PN_{SUB}_PT_{DIR}_N2-2*

Unde:
M0=(1,0,0,0,0,0,0,0,1,0,0,1);
M1=(0,1,0,0,0,0,0,0,0,0,0,0);
M2=(0,0,0,0,0,0,0,0,0,1,0,0);
M3=(0,0,1,0,0,0,0,0,0,0,0,0);
M4=(0,0,0,1,0,0,0,0,0,0,0,0);
M5=(0,0,0,0,1,0,0,0,0,0,0,0);

M6=(0,0,0,0,0,1,0,0,0,0,0,0);
M7=(0,0,0,0,0,0,1,0,0,0,0,0);
M8=(0,0,0,0,0,0,0,0,0,0,1,0);
M9=(0,0,0,0,0,0,0,1,0,0,0,0)

Studiind proprietatile comportamentale ale retelei Petri $PN_{SUB}\_PT_{DIR}\_N2-2$, pe baza arborelui de acoperire rezulta:

- reteaua este *marginita* (simbolul ω nu apare in arborele de acoperire);
- reteaua este *sigura* (marcajele din toate nodurile arborelui de acoperire contin numai „0” si „1”);
- reteaua *nu este blocanta* (toate tranzitile au asociate arce in arborele de acoperire);
- reteaua este *accesibila* (orice marcaj poate fi atins pornind din M_0).

Graful de acoperire corespunzator este prezentat in figura 2.100.

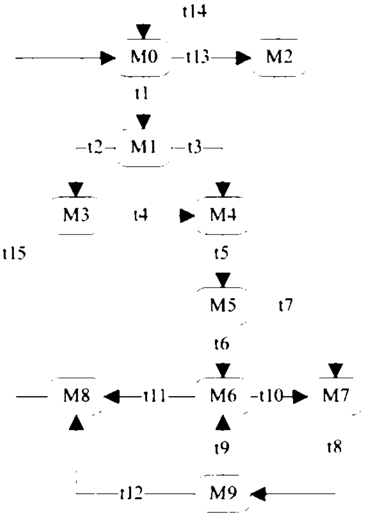


Figura 2.100. Graful de acoperire al retelei Petri $PN_{SUB}\_PT_{DIR}\_N2-2$

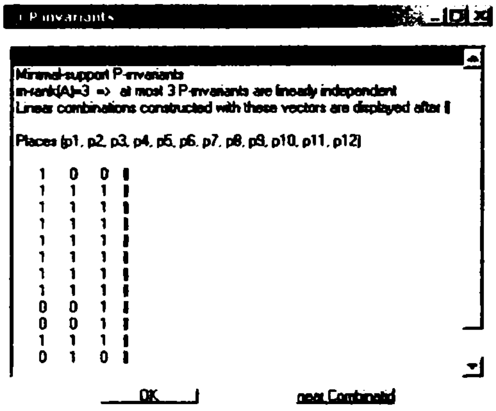
Structura rezultata pentru graful de acoperire confirma ca modelul de retea Petri corespunzator nodului cu doua intrari si doua iesiri este *acesibil*.

Concluzionand, se poate afirma ca modelul ales este *viabil* din punct de vedere comportamental.

Analiza retelei Petri $PN_{SUB}\_PT_{DIR}\_N2-2$ din punct de vedere structural se face pe baza analizei existentei invariantilor de tip *P* respectiv *T*.

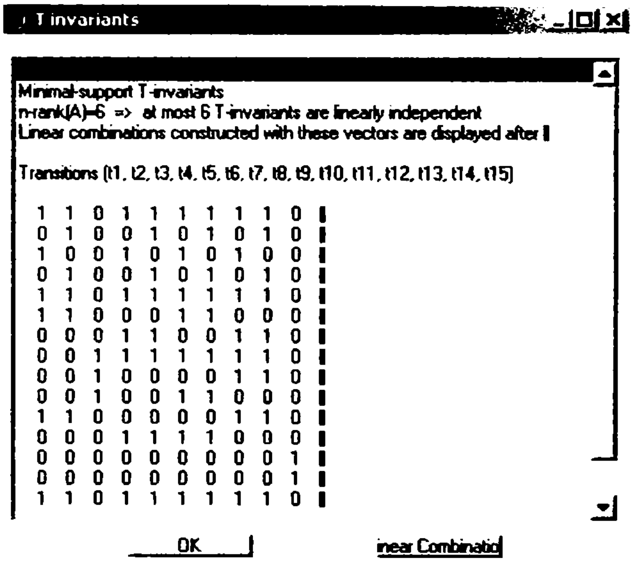
Aplicand teorema 2.3 pentru determinarea numarului de invarianti *P* respectiv *T*, cu rangul matricii A $rang\,A = 9$, rezulta ca reteaua Petri $PN_{SUB}\_PT_{DIR}\_N2-2$ este acoperita de 3 invarianti de tip *P*, respectiv 6 invarianti de tip *T*.

In figurile 2.101 si 2.102 se prezinta invariantii de tip *P* respectiv *T* obtinuti in urma simularii retelei Petri $PN_{SUB}\_PT_{DIR}\_N2-2$ cu ajutorul PNT-ului.



```
Minimal-support P-invariants
m-rank(A)=3 => at most 3 P-invariants are linearly independent
Linear combinations constructed with these vectors are displayed after |

Places (p1, p2, p3, p4, p5, p6, p7, p8, p9, p10, p11, p12)

  1   0   0  |
  1   1   1  |
  1   1   1  |
  1   1   1  |
  1   1   1  |
  1   1   1  |
  1   1   1  |
  1   1   1  |
  0   0   1  |
  0   0   1  |
  1   1   1  |
  0   1   0  |
```

Figura 2.101. Invariantii de tip P obtinuti cu ajutorul PNT



```
Minimal-support T-invariants
n-rank(A)=6 => at most 6 T-invariants are linearly independent
Linear combinations constructed with these vectors are displayed after |

Transitions (t1, t2, t3, t4, t5, t6, t7, t8, t9, t10, t11, t12, t13, t14, t15)

 1  1  0  1  1  1  1  1  1  0  |
 0  1  0  0  1  0  1  0  1  0  |
 1  0  0  1  0  1  0  1  0  0  |
 0  1  0  0  1  0  1  0  1  0  |
 1  1  0  1  1  1  1  1  1  0  |
 1  1  0  0  0  1  1  0  0  0  |
 0  0  0  1  1  0  0  1  1  0  |
 0  0  1  1  1  1  1  1  1  0  |
 0  0  1  0  0  0  0  1  1  0  |
 0  0  1  0  0  1  1  0  0  0  |
 1  1  0  0  0  0  0  1  1  0  |
 0  0  0  1  1  1  1  0  0  0  |
 0  0  0  0  0  0  0  0  0  1  |
 0  0  0  0  0  0  0  0  0  1  |
 1  1  0  1  1  1  1  1  1  0  |
```

Figura 2.102. Invariantii de tip T obtinuti cu ajutorul PNT

Conform teroemelor 2.6 respectiv 2.7, reteaua Petri $PN_{SUB}\_PT_{DIR}\_N2\text{-}2$ este *conservativa si structural marginita, respectiv este consistenta si repetitiva.*

În final se poate *concluziona* că structura aleasă pentru submodelul de tip retea Petri corespunzătoare unei directii de miscare a nodului cu doua intrari si doua iesiri este *viabil*, submodelul putand fi utilizat cu succes la modelarea completa a nodului considerat, fiind siguri că nu există condiţii de apariţie a blocajelor sau a apariţiei de situaţii necontrolabile.

Submodelul operatiei de scanare (identificare)

Submodelul ales pentru modelarea operatiei de scanare este prezentat in figura 2.103.

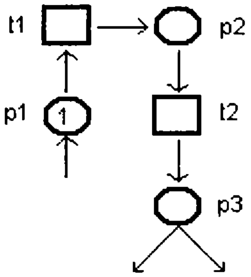


Figura 2.103. Submodelul de tip retea Petri pentru operatia de scanare

Se poate observa ca submodelul reprezinta varianta „clasica” pentru modelarea operatiei de scanare (vezi structura nodului de tip 4).

Analiza submodelului propus se face cu ajutorul retelei din figura 2.104.

In plus fata de submodelul ales (figura 2.103) apare pozitia *p4* care are rol de simulare a submodelului corespunzator directiei de deplasare. Rolul acestei pozitii suplimentare este acela de a simula „exteriorul” submodelului.

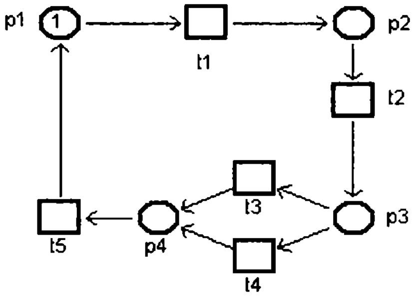


Figura 2.104. Reteaua Petri utilizata in analiza submodelului corespunzator unei operatii de scanare

Topologia retelei Petri corespunzatoare submodelului stabilit pentru o directie de miscare validata de PNT este prezentata in figura 2.105.

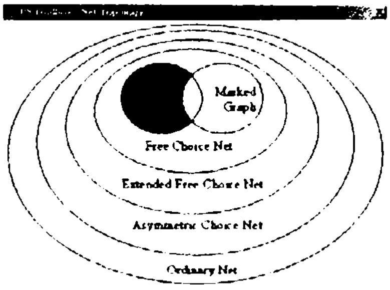


Figura 2.105. Topologia retelei Petri validata de TPN

Reteaua Petri considerata este formalizata prin cvintuplul:

$$PN_{SUB}\_PT_{SCAN}\_N2-2=(P,T,F,W,M_0)$$

unde:

- $P=\{p_1,p_2,p_3,p_4\}$

- $T = \{t_1, t_2, t_3, t_4, t_5\}$
- $F = \{(p_1,t_1),(p_2,t_2),(p_3,t_3),(p_3,t_4),(p_4,t_5)\} \cup$
  $\{(t_1,p_2),(t_2,p_3),(t_3,p_4),(t_4,p_4),(t_5,p_1)\}$
- $W(p_1,t_1) = 1, W(p_2,t_2) = 1, W(p_3,t_3) = 1, W(p_3,t_4) = 1, W(p_4,t_5) = 1,$
  $W(t_1,p_2) = 1, W(t_2,p_3) = 1,$
  $W(t_3,p_4) = 1, W(t_4,p_4) = 1, W(t_5,p_1) = 1$
- $M_0 = [\ 1,0,0,0\ ]^T$

Matricile de incidenta sunt:

Matricea de incidenta de intrare:

$$A_{Ii} = \begin{pmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{pmatrix}$$

Matricea de incidenta de iesire:

$$A_{Oi} = \begin{pmatrix} 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \\ 0 & 0 & 0 & 1 \\ 1 & 0 & 0 & 0 \end{pmatrix}$$

Matricea de incidenta:

$$A_i = A_{Oi} - A_{Ii} = \begin{pmatrix} -1 & 1 & 0 & 0 \\ 0 & -1 & 1 & 0 \\ 0 & 0 & -1 & 1 \\ 0 & 0 & -1 & 1 \\ 1 & 0 & 0 & -1 \end{pmatrix}$$

Arborele de acoperire corespunzator retelei Petri *PN*

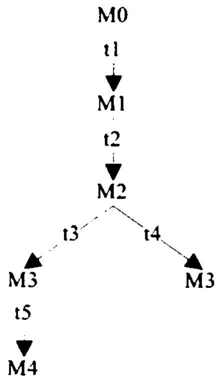


Figura 2.106. Arborele de acoperire corespunzator $PN_{SUB}\_PT_{SCAN}\_N2\text{-}2$

Unde:
M0=(1,0,0,0); M1=(0,1,0,0); M2=(0,0,1,0); M3=(0,0,0,1).

Studiind proprietatile comportamentale ale retelei Petri $PN_{SUB}\_PT_{SCAN}\_N2\text{-}2$, pe baza arborelui de acoperire rezulta:

- reteaua este *marginita* (simbolul ω nu apare in arborele de acoperire);
- reteaua este *sigura* (marcajele din toate nodurile arborelui de acoperire contin numai „0" si „1");
- reteaua *nu este blocanta* (toate tranzitile au asociate arce in arborele de acoperire);
- reteaua este *accesibila* (orice marcaj poate fi atins pornind din M_0).

Graful de acoperire corespunzator este prezentat in figura 2.107.

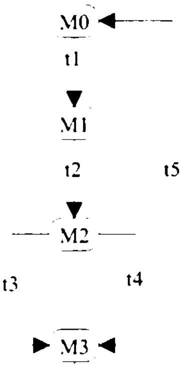


Figura 2.107. Graful de acoperire al retelei Petri $PN_{SUB}\_PT_{SCAN}\_N2\text{-}2$

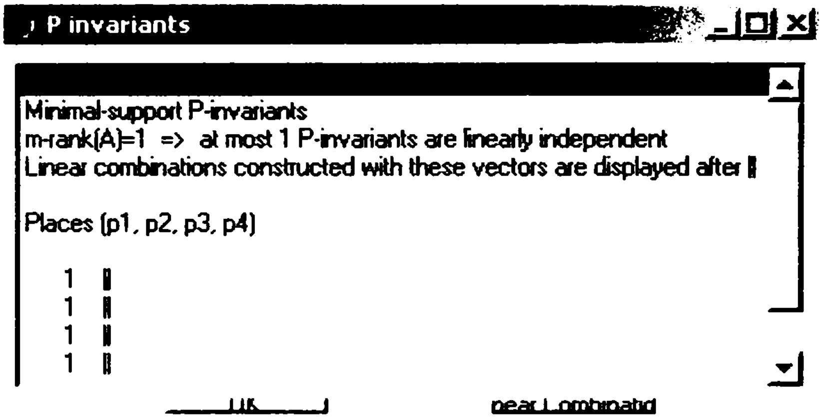
Structura rezultata pentru graful de acoperire confirma ca modelul de retea Petri corespunzator nodului cu doua intrari si doua iesiri este *acesibil*.

Concluzionand, se poate afirma ca modelul ales este *viabil* din punct de vedere comportamental.

Analiza retelei Petri $PN_{SUB}\_PT_{SCAN}\_N2\text{-}2$ din punct de vedere structural se face pe baza analizei existentei invariantilor de tip *P* respectiv *T*, aplicand teorema 2.3, cu $rang\,A = 3$.

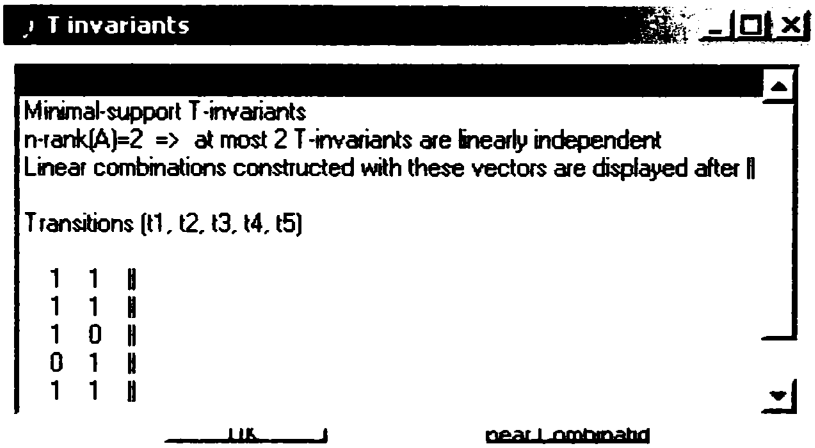
Rezulta ca reteaua Petri $PN_{SUB}\_PT_{SCAN}\_N2\text{-}2$ este acoperita de 1 invariant de tip *P*, respectiv 3 invarianti de tip *T*.

In figurile 2.108 si 2.109 se prezinta invariantii de tip *P* respectiv *T* obtinuti in urma simularii retelei Petri $PN_{SUB}\_PT_{SCAN}\_N2\text{-}2$ cu ajutorul PNT-ului.



```
Minimal-support P-invariants
m-rank(A)=1 => at most 1 P-invariants are linearly independent
Linear combinations constructed with these vectors are displayed after |

Places (p1, p2, p3, p4)

   1 |
   1 |
   1 |
   1 |
```

Figura 2.108. Invariantii de tip P obtinuti cu ajutorul PNT



```
Minimal-support T-invariants
n-rank(A)=2 => at most 2 T-invariants are linearly independent
Linear combinations constructed with these vectors are displayed after |

Transitions (t1, t2, t3, t4, t5)

  1  1 |
  1  1 |
  1  0 |
  0  1 |
  1  1 |
```

Figura 2.109. Invariantii de tip T obtinuti cu ajutorul PNT

Conform teroemelor 2.6 respectiv 2.7, reteaua Petri $PN_{SUB}\_PT_{SCAN}\_N2\text{-}2$ este *conservativa si structural marginita, respectiv este consistenta si repetitiva*.

În final se poate *concluziona* că structura aleasă pentru submodelul de tip retea Petri corespunzătoare operatiei de scanare a nodului cu doua intrari si doua iesiri este *viabil*, submodelul putand fi utilizat cu succes la modelarea completa a nodului considerat, neexistand condiţii de apariţie a blocajelor sau a apariţiei de situaţii necontrolabile.

Rezultatele obtinute in urma stabilirii submodelelor componente ale modelului de tip retea Petri corespunzator nodului cu doua intrari si doua iesiri, asigura premizele ca rezultatele obtinute anterior (in cazul modelului) sa fie validate in totalitate.

2.5.2.5.2.2. Cazul: retea Petri temporizata P

In cazul modelarii utilizand retele Petri temporizate P , structura retelei si modul de utilizare a submodeleor este aceeasi ca si in cazul modelarii cu retele Petri netemporizate. Fata de modul de abordare prezentat in cazul celorlalte tipuri de noduri, datorita utilizarii submodelelor si conform teoremei 2.9, reteaua Petri care va modela ingreg nodul este o retea Petri temporizata P cu intervale de timp.

Datorita faptului ca retelele sunt identice din punct de vedere structural si comportamental rezultatele analizelor nu se mai detaliaza ele fiind identice cu cele obtinute in cazul nodului cu doua intrari si doua iesiri netemporizat.

Determinarea intervalelor de timp corespunzatoare submodelelor se face pornind de la timpii alocati fiecarei pozitii ale submodelului.

Pentru submodelul corespunzator unei directii de miscare timpii alocati pozitiilor sunt prezentati in tabelul 2.11.

Tabelul 2.11. *Timpii corespunzatori pozitiilor P ai unei directii de miscare*

Pozitie	Unitati de Timp
P1	1
P2	5
P3	4
P4	4
P5	7
P6	6
P7	3
P8	5
P9	1
P10	3
P11	0
P12	1

Reteaua Petri temporizata P pentru o directie de miscare este:

$$PN_{SUB}\_PTtime_{DIR}\_N2-2=(P,T,F,W,D,M_0)$$

unde:

- $P=\{p_1,p_2,p_3,p_4,p_5,p_6,p_7,p_8,p_9,p_{10},p_{11},p_{12}\}$
- $T=\{t_1,t_2,t_3,t_4,t_5,t_6,t_7,t_8,t_9,t_{10},t_{11},t_{12},t_{13},t_{14},t_{15}\}$
- $F=\{(p_1,t_1),(p_2,t_2),(p_2,t_3),(p_3,t_4),(p_4,t_5),(p_5,t_6),(p_5,t_7),(p_6,t_{10}),(p_6,t_{11}),(p_7,t_8),(p_8,t_9),(p_8,t_{12}),(p_9,t_{13}),(p_{10},t_{14}),(p_{11},t_{15}),(p_{12},t_1)\}\cup\{(t_1,p_2),(t_2,p_3),(t_3,p_4),(t_4,p_4),(t_5,p_5),(t_6,p_6),(t_7,p_7),(t_8,p_8),(t_9,p_6),(t_{10},p_7),(t_{11},p_{11}),(t_{12},p_{11}),(t_{13},p_{10}),(t_{14},p_9),(t_{15},p_1),(t_{15},p_9),(t_{15},p_{12})\}$
- $W(p_1,t_1)=1,W(p_2,t_2)=1,W(p_2,t_3)=1,W(p_3,t_4)=1,W(p_4,t_5)=1,W(p_5,t_6)=1,W(p_5,t_7)=1,W(p_6,t_{10})=1,W(p_6,t_{11})=1,W(p_7,t_8)=1,W(p_8,t_9)=1,W(p_8,t_{12})=1,W(p_9,t_{13})=1,W(p_{10},t_{14})=1,W(p_{11},t_{15})=1,W(p_{12},t_1)=1W(t_1,p_2)=1,W(t_2,p_3)=1,W(t_3,p_4)=1,W(t_4,p_4)=1,W(t_5,p_5)=1,W(t_6,p_6)=1,W(t_7,p_7)=1,W(t_8,p_8)=1,W(t_9,p_6)=1,W(t_{10},p_7)=1,W(t_{11},p_{11})=1,W(t_{12},p_{11})=1,W(t_{13},p_{10})=1,W(t_{14},p_9)=1,W(t_{15},p_1)=1,W(t_{15},p_9)=1,W(t_{15},p_{12})=1$
- $D=\{d(p_1)=1,d(p_2)=5,d(p_3)=4,d(p_4)=4,d(p_5)=7,d(p_6)=6,d(p_7)=3,d(p_8)=5,d(p_9)=1,d(p_{10})=3,d(p_{11})=0,d(p_{12})=1\}$

- $M_0 = [\ 1,0,0,0,0,0,0,0,1,0,0,1\ ]^T$

Analizand reteaua, si utilizand aplicatia *PetriTim*, rezulta ca timpul minim de executie al retelei Petri considerate este de 4 unitati de timp iar timpul maxim este de 37 unitati de timp.

Pentru submodelul corespunzator unei operatii de scanare timpii alocati pozitiilor sunt prezentati in tabelul 2.12.

Tabelul 2.12. *Timpii corespunzatori pozitiilor P ai operatiei de scanare*

Pozitie	Unitati de Timp
P1	1
P2	3
P3	4
P4	0

Reteaua Petri temporizata P pentru o directie de miscare este:

$$PN_{SUB}\_PTtime_{SCAN}\_N2-2 = (P,T,F,W,D,M_0)$$

unde:

- $P = \{p_1,p_2,p_3,p_4\}$
- $T = \{t_1,t_2,t_3,t_4,t_5\}$
- $F = \{(p_1,t_1),(p_2,t_2),(p_3,t_3),(p_3,t_4),(p_4,t_5)\} \cup$
  $\{(t_1,p_2),(t_2,p_3),(t_3,p_4),(t_4,p_4),(t_5,p_1)\}$
- $W(p_1,t_1) = 1, W(p_2,t_2) = 1, W(p_3,t_3) = 1, W(p_3,t_4) = 1, W(p_4,t_5) = 1,$
  $W(t_1,p_2) = 1, W(t_2,p_3) = 1,$
  $W(t_3,p_4) = 1, W(t_4,p_4) = 1, W(t_5,p_1) = 1$
- $D = \{d(p_1) = 1, d(p_2) = 5, d(p_3) = 4, d(p_4) = 0\}$
- $M_0 = [\ 1,0,0,0\ ]^T$

Analizand reteaua, si utilizand aplicatia *PetriTim,* rezulta ca timpul minim de executie al retelei Petri considerate este egal cu timpul maxim avand valoarea de 8 unitati de timp.

Avand intervalele de timp corespunzatoare submodelelor, reteaua Petri finala va vea intervalele de timpi prezentate in tabelul 2.13.

Tabelul 2.13.*Intervalele de timpi corespunzatoare pozitiilor P*

Pozitie	Unitati de Timp	
	Min	Max
P1,P2,P4,P5	4	37
P3,P6	8	8
P7	0	0

În final se poate *concluziona* că structura aleasă pentru o retea Petri temporizata P corespunzătoare nodului cu doua intrari si doua iesiri este *viabilă*, ea putând fi implementată, neexistand condiţii de apariţie a blocajelor sau a apariţiei de situaţii necontrolabile.

2.5.2.6. Modelarea unui nod cu „n" intrari si o iesire [UP06a] [Ung06b]

Nodul cu *n* intrari si *o* iesire este construit cu ajutorul nodului de tip 1.

2.5.2.6.1. Modelarea nodului „n" intrari si o iesire cu ajutorul automatelor

Structura automatului secvential considerat pentru modelarea unui astfel de nod este prezentata în figura 2.110.

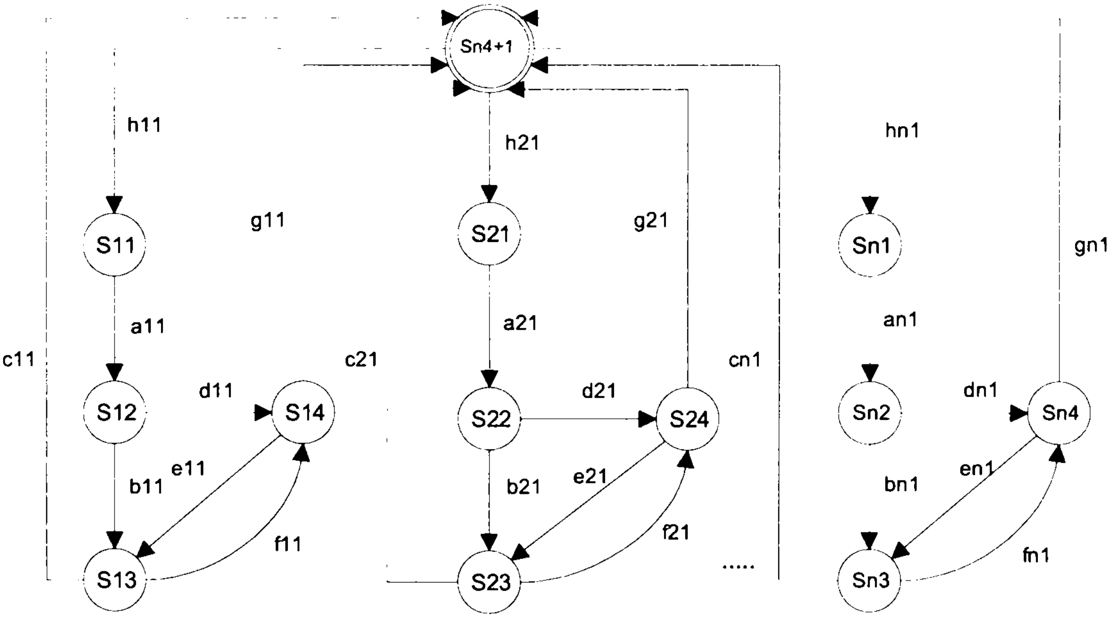


Figura 2.110. Structura automatului secvential corespunzător nodului cu „n" intrari si o iesire

Se observa ca automatul secvential al nodului este realizat prin sincronizarea a *n* automate secventiale AS_N1, cate unul pentru fiecare linie de intrare (stoper).

Automatul *AS_Nn* corespunzator nodului cu *n* intrari si *o* iesire este descris in modul urmator:

- multimea starilor: $X = \bigcup_{i=1}^{n} \{Si1, Si2, Si3, Si4\} \cup \{S(4 \cdot n + 1)\}$
- multimea evenimentelor: $\Sigma = \bigcup_{i=1}^{n} \{ai1, bi1, ci1, di1, ei1, fi1, gi1, hi1\}$
- multimile evenimentelor posibile si functiile de tranzitie a starilor:

$\Gamma(S) = \bigcup_{i=1}^{n} \Gamma(Si)$, unde:

$$\begin{array}{lll}
\Gamma(Si1) = \{ai\} & \delta(Si1, ai) = Si2 & \\
\Gamma(Si2) = \{bi, di\} & \delta(Si2, bi) = Si3, & \delta(Si2, di) = Si4 \\
\Gamma(Si3) = \{ci, fi\} & \delta(Si3, ci) = S(4 \cdot n + 1), & \delta(Si3, fi) = Si4 \\
\Gamma(Si4) = \{ei, gi\} & \delta(Si4, ei) = Si3 & \delta(Si4, gi) = S(4 \cdot n + 1)
\end{array}$$

$$\Gamma(S(4 \cdot n + 1) = \bigcup_{i=1}^{n} \{hi\} \qquad \delta(S(4 \cdot n + 1), hi) = Si1$$

- starea initiala: $x_0 = S(4 \cdot n + 1)$.

Considerentele legate de transformarea modelului de tip automat in model de tip retea Petri, prezentate la nodul de tip 1 sunt valabile si in cazul nodului cu *n* intrari si o iesire.

Pentru conversia din model de tip automat in model de tip retea Petri s-a utilizat aceeasi metoda ca si in cazul nodului de tip 1 .

Aplicand algoritmul 2.1 rezulta:

- multimea pozitiilor: $P = \bigcup_{i=1}^{n} \{pi1, pi2, pi3, pi4\} \cup \{p(4n + 1)\}$, unde
  $pi1 = \{Si1\}, pi2 = \{Si2\}, pi3 = \{Si3\}, pi4 = \{Si4\}$ si
  $p(4 \cdot n + 1) = \{S(4 \cdot n + 1\}$;
- multimea trranzitiilor:

• Pentru *pi1*	*(pi1,pi2)*	*pi2=δ(pi1,ti1)*	*ti1={ai}*
• Pentru *pi2*	*(pi2,pi3)*	*pi3=δ(pi2,ti2)*	*ti2={bi}*
	(pi2,pi4)	*pi4=δ(pi2,ti4)*	*ti4={di}*
• Pentru *pi3*	*(pi3,p(4n+1))*	*p(4n+1)=δ(pi3,ti3)*	*ti3={ci}*
	(pi3,pi4)	*pi4=δ(pi3,ti6)*	*ti6={fi}*
• Pentru *pi4*	*(pi4,p(4n+1))*	*p(4n+1)=δ(pi4,ti7)*	*ti7={gi}*
	(pi4,pi3)	*pi3=δ(pi4,ti5)*	*ti5={ei}*
• Pentru *p(4n+1)*	*(p(4n+1),pi1)*	*pi1=δ(p4n+1),ti8)*	*ti8={hi}*

In figura 2.111 se prezinta structura retelei Petri asociata automatului secvential considerat in cazul nodului cu *n* intrari si o iesire.

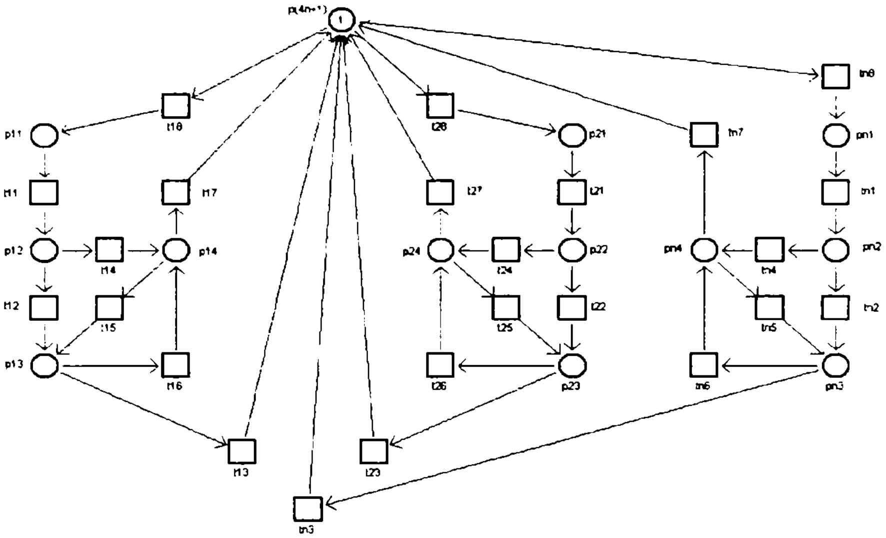


Figura 2.111. Structura reţelei Petri asociata automatului secvential corespunzător nodului cu *n* intrari si o iesire

Topologia retelei validata de PNT este prezentata in figura 2.112.

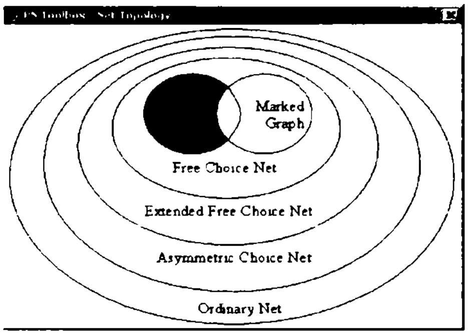


Figura 2.112. Topologia retelei Petri echivalenta validata de PNT

Rezultatele obtinute confirma ca dupa efectuarea conversiei din modelul de tip automat in model de tip retea Petri, topologia retelei Petri obtinuta este tot de tip automat („State machine").

Reteaua Petri considerata este formalizata prin cvintuplul:

$$PN\_AS\_Nn = (P, T, F, W, M_0)$$

unde:

- $P = \bigcup_{i=1}^{n} \{p_{i1}, p_{i2}, p_{i3}, p_{i4}\} \cup \{p_{4n+1}\}$
- $T = \bigcup_{i=1}^{n} \{t_{i1}, t_{i2}, t_{i3}, t_{i4}, t_{i5}, t_{i6}, t_{i7}, t_{i8}\}$

- $$F = \bigcup_{i=1}^{n} \{\{(p_{i1},t_{i1}),(p_{i2},t_{i2}),(p_{i2},t_{i4}),(p_{i3},t_{i6}),(p_{i3},t_{i3}),(p_{i4},t_{i5}), (p_{i4},t_{i7}),(p_{4n+1},t_{i8})\} \cup \{(t_{i1},p_{i2}),(t_{i2},p_{i3}),(t_{i3},p_{4n+1}),(t_{i4},p_{i4}),(t_{i5},p_{i3}), (t_{i6},p_{i4}),(t_{i7},p_{4n+1}),(t_{i8},p_{i1})\}\}$$
- $$W(p_{i1},t_{i1}) = 1, W(p_{i2},t_{i2}) = 1, W(p_{i2},t_{i4}) = 1, W(p_{i3},t_{i6}) = 1, W(p_{i3},t_{i3}) = 1, W(p_{i4},t_{i5}) = 1, W(p_{i4},t_{i7}) = 1, W(p_{4n+1},t_{i8}) = 1, W(t_{i1},p_{i2}) = 1, W(t_{i2},p_{i3}) = 1, W(t_{i3},p_{4n+1}) = 1, W(t_{i4},p_{i4}) = 1, W(t_{i5},p_{i3}) = 1, W(t_{i6},p_{i4}) = 1, W(t_{i7},p_{4n+1}) = 1, W(t_{i8},p_{i1}) = 1,$$
- $M_0 = [0,0,0,0,0,0,0,0,0,0,0,0,\ldots,1]^T$

Matricile de incidenţă corespunzătoare stoperului (intrarii) i , unde coloanele sunt p_{i1}, p_{i2}, p_{i3} si p_{i4} iar linile: t_{i1}, t_{i2}, t_{i3}, t_{i4}, t_{i5}, t_{i6}, t_{i7} si t_{i8} sunt.

Matricea de incidenta de intrare:

$$A_{Ii} = \begin{pmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 1 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \\ 0 & 0 & 0 & 0 \end{pmatrix}$$

Matricea de incidenta de iesire:

$$A_{Oi} = \begin{pmatrix} 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \\ 0 & 0 & 0 & 0 \\ 1 & 0 & 0 & 0 \end{pmatrix}$$

Matricea de incidenta:

$$A_i = A_{Oi} - A_{Ii} = \begin{pmatrix} -1 & 1 & 0 & 0 \\ 0 & -1 & 1 & 0 \\ 0 & 0 & -1 & 0 \\ 0 & -1 & 0 & 1 \\ 0 & 0 & 1 & -1 \\ 0 & 0 & -1 & 1 \\ 0 & 0 & 0 & -1 \\ 1 & 0 & 0 & 0 \end{pmatrix}$$

Avand structurile matricilor pentru un stoper, matricile corespunzatoare nodului cu n intrari (stopere) si o iesire sunt.

Matricea de intrare:

$$A_I=\left(\begin{array}{c|c|c|c|c|l}
p11\ p12\ p13\ p14 & p21\ p22\ p23\ p24 & \cdots & pn1\ pn2\ pn3\ pn4 & p4n+1 & \\
 & & & & 0 & t11 \\
 & & & & 0 & t12 \\
 & & & & 0 & t13 \\
A_{1I} & 0 & & 0 & 0 & t14 \\
 & & & & 0 & t15 \\
 & & & & 0 & t16 \\
 & & & & 0 & t17 \\
 & & & & 1 & t18 \\
\hline
 & & & & 0 & t21 \\
 & & & & 0 & t22 \\
 & & & & 0 & t23 \\
0 & A_{2I} & & 0 & 0 & t24 \\
 & & & & 0 & t25 \\
 & & & & 0 & t26 \\
 & & & & 0 & t27 \\
 & & & & 1 & t28 \\
\hline
 & & & & & . \\
 & & & & & . \\
\hline
 & & & & 0 & tn1 \\
 & & & & 0 & tn2 \\
 & & & & 0 & tn3 \\
0 & 0 & & A_{nI} & 0 & tn4 \\
 & & & & 0 & tn5 \\
 & & & & 0 & tn6 \\
 & & & & 0 & tn7 \\
 & & & & 1 & tn8
\end{array}\right)$$

Matricea de iesire:

$$A_O=\left(\begin{array}{c|c|c|c|c}
p11\ p12\ p13\ p14 & p21\ p22\ p23\ p24 & \cdots\cdots & pn1\ pn2\ pn3\ pn4 & p4n+1 \\
 & & & & 0 \\
 & & & & 0 \\
 & & & & 1 \\
A_{1O} & 0 & & 0 & 0 \\
 & & & & 0 \\
 & & & & 0 \\
 & & & & 1 \\
 & & & & 0 \\
\hline
 & & & & 0 \\
 & & & & 0 \\
 & & & & 1 \\
0 & A_{2O} & & 0 & 0 \\
 & & & & 0 \\
 & & & & 0 \\
 & & & & 1 \\
 & & & & 0 \\
\hline
 & & & & \vdots \\
\hline
 & & & & 0 \\
 & & & & 0 \\
 & & & & 1 \\
0 & 0 & & A_{nO} & 0 \\
 & & & & 0 \\
 & & & & 0 \\
 & & & & 1 \\
 & & & & 0
\end{array}\right)
\begin{array}{l}
\\ t11 \\ t12 \\ t13 \\ t14 \\ t15 \\ t16 \\ t17 \\ t18 \\ t21 \\ t22 \\ t23 \\ t24 \\ t25 \\ t26 \\ t27 \\ t28 \\ \vdots \\ tn1 \\ tn2 \\ tn3 \\ tn4 \\ tn5 \\ tn6 \\ tn7 \\ tn8
\end{array}$$

Matricea de incidenta:

$$A=A_O-A_I=\left(\begin{array}{c|c|c|c|c}
p11\ p12\ p13\ p14 & p21\ p22\ p23\ p24 & \cdots\cdots & pn1\ pn2\ pn3\ pn4 & p4n+1 \\
 & & & & 0 \\
 & & & & 0 \\
 & & & & 1 \\
A_1 & 0 & & 0 & 0 \\
 & & & & 0 \\
 & & & & 0 \\
 & & & & 1 \\
 & & & & -1 \\
\hline
 & & & & 0 \\
 & & & & 0 \\
 & & & & 1 \\
0 & A_2 & & 0 & 0 \\
 & & & & 0 \\
 & & & & 0 \\
 & & & & 1 \\
 & & & & -1 \\
\hline
 & & & & \vdots \\
\hline
 & & & & 0 \\
 & & & & 0 \\
 & & & & 1 \\
0 & 0 & & A_n & 0 \\
 & & & & 0 \\
 & & & & 0 \\
 & & & & 1 \\
 & & & & -1
\end{array}\right)
\begin{array}{l}
\\ t11 \\ t12 \\ t13 \\ t14 \\ t15 \\ t16 \\ t17 \\ t18 \\ t21 \\ t22 \\ t23 \\ t24 \\ t25 \\ t26 \\ t27 \\ t28 \\ \vdots \\ tn1 \\ tn2 \\ tn3 \\ tn4 \\ tn5 \\ tn6 \\ tn7 \\ tn8
\end{array}$$

Arborele de acoperire corespunzator este prezentat in figura 2.113.

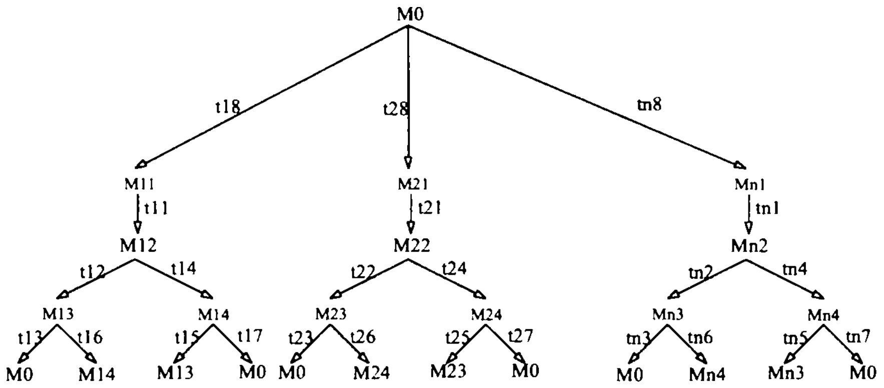


Figura 2.113. Arborele de acoperire al retelei Petri *PN_AS_Nn*

Unde:

M0=(0,0,0,0,0,0,0,0,...,0,0,0,0,1); M11=(1,0,0,0,0,0,0,0,...,0,0,0,0,0);
M12=(0,1,0,0,0,0,0,0,...,0,0,0,0,0); M13=(0,0,1,0,0,0,0,0,...,0,0,0,0,0);
M14=(0,0,0,1,0,0,0,0,...,0,0,0,0,0); M21=(0,0,0,0,1,0,0,0,...,0,0,0,0,0);
M22=(0,0,0,0,0,1,0,0,...,0,0,0,0,0); M23=(0,0,0,0,0,0,1,0,...,0,0,0,0,0);
M24=(0,0,0,0,0,0,0,1,...,0,0,0,0,0); Mn1=(0,0,0,0,0,0,0,0,...,1,0,0,0,0);
Mn2=(0,0,0,0,0,0,0,0,...,0,1,0,0,0); Mn3=(0,0,0,0,0,0,0,0,...,0,0,1,0,0);
Mn4=(0,0,0,0,0,0,0,0,...,0,0,0,1,0)

Studiind proprietatile comportamentale ale retelei Petri *PN_AS_Nn*, pe baza arborelui de acoperire rezulta:

- reteaua este *marginita* (simbolul ω nu apare in arborele de acoperire);
- reteaua este *sigura* (marcajele din toate nodurile arborelui de acoperire contin numai „0" si „1");
- reteaua *nu este blocanta* (toate tranzitile au asociate arce in arborele de acoperire);
- reteaua este *accesibila* (orice marcaj poate fi atins pornind din M_0).

Graful de acoperire corespunzator este prezentat in figura 2.114.

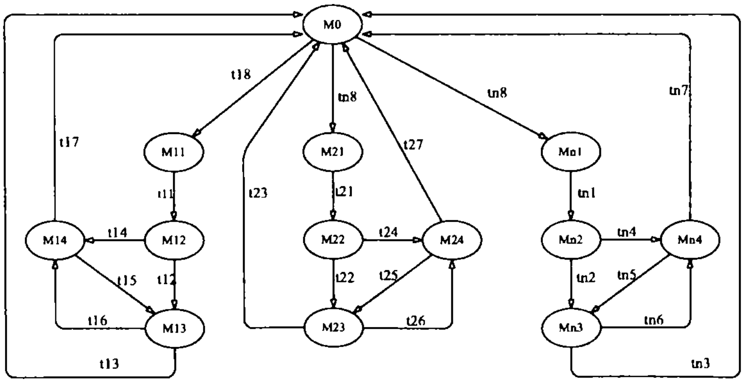


Figura 2.114. Graful de acoperire al retelei Petri *PN_AS_Nn*

Structura rezultata pentru graful de acoperire confirma faptul ca modelul de retea Petri derivat din modelul de tip automat, corespunzator nodului cun *n* intrari si o iesire este *acesibila*.

Concluzionand, se poate afirma ca modelul ales este *viabil* din punct de vedere comportamental.

Analiza retelei Petri *PN_AS_Nn* din punct de vedere structural se face pe baza analizei existentei invariantilor de tip *P* respectiv *T* utilizand teorema 2.3, rangul matricii A fiind:

$$rang\,A = nr\_nod \cdot rangN_1$$

unde:

- *nr_nod* – reprezinta numarul de intrari ale nodului;
- *rangN₁* – reprezinta rangul maticii de incidenta corespunzatoare nodului de tip 1 ($rangN_1=4$).

Numarul de invarianti *P* corespunzatori este:

$$Nr\_Inv\_P = m - rang\,A = m - nr\_nod \cdot rangN_1$$

Numarul de invarianti *T* corespunzatori este:

$$Nr\_Inv\_T = n - rang\,A = n - nr\_nod \cdot rangN_1$$

2.5.2.6.2. Modelarea nodului cu *n* intrari si *o* iesire cu ajutorul retelelor Petri

2.5.2.6.2.1. Cazul: retea Petri netemporizata

Structura retelei Petri considerate pentru modelarea nodului cu *n* intrari si o iesire este prezentata în figura 2.115.

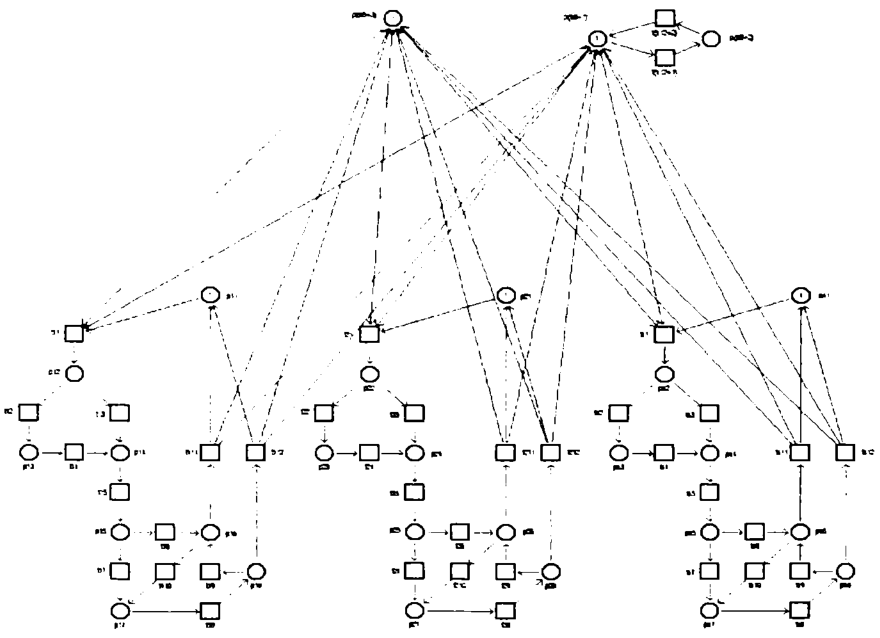


Figura 2.115. Structura retelei Petri corespunzătoare nodului cu *n* intrari si o iesire

Reteaua Petri a nodului este realizat prin sincronizarea a *n* retele Petri *PN_PT_N1*, cate una pentru fiecare linie de intrare (stoper).

Topologia retelei Petri validata de PNT este prezentata in figura 2.116.

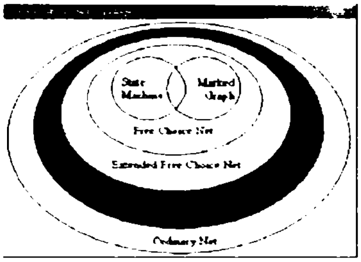


Figura 2.116. Topologia retelei Petri validata de PNT

Reteaua Petri considerata este formalizata prin cvintuplul:

$$PN\_PT\_Nn = (P,T,F,W,M_0)$$

unde:

- $$P = \bigcup_{i=1}^{n} \{p_{i1}, p_{i2}, p_{i3}, p_{i4}, p_{i5}, p_{i6}, p_{i7}, p_{i8}\} \cup \{p_{(n8+1)}, p_{(n8+2)}, p_{(n8+3)}\}$$
- $$T = \bigcup_{i=1}^{n} \{t_{i1}, t_{i2}, t_{i3}, t_{i4}, t_{i5}, t_{i6}, t_{i7}, t_{i8}, t_{i9}, t_{i10}, t_{i11}, t_{i12}\} \cup \{t_{(n12+1)}, t_{(n12+2)}\}$$
- $$\begin{aligned} F = \bigcup_{i=1}^{n} \{\{&(p_{i1},t_{i1}),(p_{i2},t_{i2}),(p_{i2},t_{i3}),(p_{i3},t_{i4}),(p_{i4},t_{i5}),(p_{i5},t_{i6}),\\ &(p_{i5},t_{i7}),(p_{i6},t_{i10}),(p_{i6},t_{i11}),(p_{i7},t_{i8}),(p_{i8},t_{i9}),(p_{i8},t_{i12}),\\ &(p_{n8+1},t_{i1}),(p_{n8+3},t_{i1})\} \cup\\ &\{(t_{i1},p_{i2}),(t_{i2},p_{i3}),(t_{i3},p_{i4}),(t_{i4},p_{i4}),(t_{i5},p_{i5}),(t_{i6},p_{i6}),\\ &(t_{i7},p_{i7}),(t_{i8},p_{i8}),(t_{i9},p_{i6}),(t_{i10},p_{i7}),(t_{i11},p_{i1}),(t_{i11},p_{n8+1}),\\ &(t_{i11},p_{n8+3}),(t_{i12},p_{i1}),(t_{i12},p_{n8+1}),(t_{i12},p_{n8+3})\}\} \cup\\ &\{\{(p_{n8+1},t_{n12+1}),(p_{n8+2},t_{n12+2})\} \cup \{(t_{n12+1},p_{n8+2}),\\ &(t_{n12+2},p_{n8+1})\}\} \end{aligned}$$
- $$\begin{aligned} &W(p_{i1},t_{i1}) = 1, W(p_{i2},t_{i2}) = 1, W(p_{i2},t_{i3}) = 1, W(p_{i3},t_{i4}) = 1,\\ &W(p_{i4},t_{i5}) = 1, W(p_{i5},t_{i6}) = 1, W(p_{i5},t_{i7}) = 1, W(p_{i6},t_{i10}) = 1,\\ &W(p_{i6},t_{i11}) = 1, W(p_{i7},t_{i8}) = 1, W(p_{i8},t_{i9}) = 1, W(p_{i8},t_{i12}) = 1,\\ &W(p_{n8+1},t_{i1}) = 1, W(p_{n8+3},t_{i1}) = 1, W(p_{n8+1},t_{n12+1}) = 1,\\ &W(p_{n8+2},t_{n12+2}) = 1, W(t_{i1},p_{i2}) = 1, W(t_{i2},p_{i3}) = 1, W(t_{i3},p_{i4}) = 1,\\ &W(t_{i4},p_{i4}) = 1, W(t_{i5},p_{i5}) = 1, W(t_{i6},p_{i6}) = 1, W(t_{i7},p_{i7}) = 1,\\ &W(t_{i8},p_{i8}) = 1, W(t_{i9},p_{i6}) = 1, W(t_{i10},p_{i7}) = 1, W(t_{i11},p_{i1}) = 1,\\ &W(t_{i11},p_{n8+1}) = 1, W(t_{i11},p_{n8+3}) = 1, W(t_{i12},p_{i1}) = 1, W(t_{i12},p_{n8+1}) = 1,\\ &W(t_{i12},p_{n8+3}) = 1, W(t_{n12+1},p_{n8+2}) = 1, W(t_{n12+2},p_{n8+1}) = 1 \end{aligned}$$
- $$M_0 = [1, 0,0,0,0,0,0,0,1,0,0,0,0,0,0,0,......,1,0,0,0,0,0,0,0,1,0,1]^T$$

Matricile de incidenţă corespunzătoare stoperului (intrarii) i , unde coloanele sunt p_{i1}, p_{i2}, p_{i3} , p_{i4} , p_{i5}, p_{i6}, p_{i7} si p_{i8} iar linile: t_{i1}, t_{i2}, t_{i3}, t_{i4}, t_{i5}, t_{i6}, t_{i7}, t_{i8}, t_{i9}, t_{i10}, t_{i11}, si t_{i12} sunt.

Matricea de incidenta de intrare:

$$A_{Ii} = \begin{matrix} 10000000 \\ 0\ 1000000 \\ 0\ 1000000 \\ 00\ 100000 \\ 000\ 10000 \\ 0000\ 1000 \\ 0000\ 1000 \\ 000000\ 10 \\ 0000000\ 1 \\ 00000\ 100 \\ 00000\ 100 \\ 0000000\ 1 \end{matrix}$$

Matricea de incidenta de iesire:

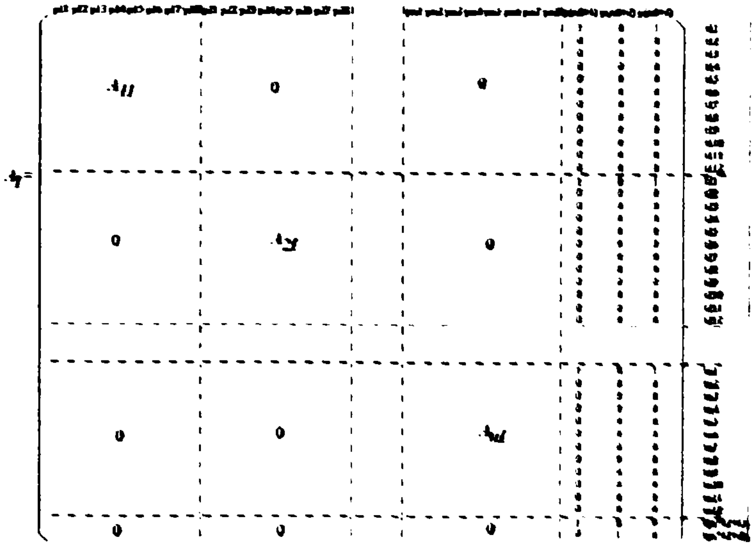
$$A_{Oi} = \begin{matrix} 0\ 1000000 \\ 00\ 100000 \\ 000\ 10000 \\ 000\ 10000 \\ 0000\ 1000 \\ 00000\ 100 \\ 000000\ 10 \\ 0000000\ 1 \\ 00000\ 100 \\ 000000\ 10 \\ 10000000 \\ 10000000 \end{matrix}$$

Matricea de incidenta:

$$A_i = A_{Oi} - A_{Ii} = \begin{pmatrix} -1 & 1 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & -1 & 1 & 0 & 0 & 0 & 0 & 0 \\ 0 & -1 & 0 & 1 & 0 & 0 & 0 & 0 \\ 0 & 0 & -1 & 1 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & -1 & 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & -1 & 1 & 0 & 0 \\ 0 & 0 & 0 & 0 & -1 & 0 & 1 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & -1 & 1 \\ 0 & 0 & 0 & 0 & 0 & 1 & 0 & -1 \\ 0 & 0 & 0 & 0 & 0 & -1 & 1 & 0 \\ 1 & 0 & 0 & 0 & 0 & -1 & 0 & 0 \\ 1 & 0 & 0 & 0 & 0 & 0 & 0 & -1 \end{pmatrix}$$

Avand structurile matricilor pentru un stoper, matricile corespunzatoare nodului cu n intrari (stopere) si o iesire sunt.

Matricea de intrare:



Matricea de iesire:

$$A_O=\left(\begin{array}{c|c|c|c|ccc|l}
\text{p11 p12 p13 p14 p15 p16 p17 p18} & \text{p21 p22 p23 p24 p25 p26 p27 p28} & & \text{pn1 pn2 pn3 pn4 pn5 pn6 pn7 pn8} & p(n8+1) & p(n8+2) & p(n8+3) & \\
 & & & & 0 & 0 & 0 & t11 \\
 & & & & 0 & 0 & 0 & t12 \\
 & & & & 0 & 0 & 0 & t13 \\
 & & & & 0 & 0 & 0 & t14 \\
 & & & & 0 & 0 & 0 & t15 \\
A_{1O} & 0 & & 0 & 0 & 0 & 0 & t16 \\
 & & & & 0 & 0 & 0 & t17 \\
 & & & & 0 & 0 & 0 & t18 \\
 & & & & 0 & 0 & 0 & t19 \\
 & & & & 0 & 0 & 0 & t110 \\
 & & & & 1 & 0 & 1 & t111 \\
 & & & & 1 & 0 & 1 & t112 \\
\hline
 & & & & 0 & 0 & 0 & t21 \\
 & & & & 0 & 0 & 0 & t22 \\
 & & & & 0 & 0 & 0 & t23 \\
 & & & & 0 & 0 & 0 & t24 \\
 & & & & 0 & 0 & 0 & t25 \\
0 & A_{2O} & & 0 & 0 & 0 & 0 & t26 \\
 & & & & 0 & 0 & 0 & t27 \\
 & & & & 0 & 0 & 0 & t28 \\
 & & & & 0 & 0 & 0 & t29 \\
 & & & & 0 & 0 & 0 & t210 \\
 & & & & 1 & 0 & 1 & t211 \\
 & & & & 1 & 0 & 1 & t212 \\
\hline
 & & & & & & & \\
\hline
 & & & & 0 & 0 & 0 & tn1 \\
 & & & & 0 & 0 & 0 & tn2 \\
 & & & & 0 & 0 & 0 & tn3 \\
 & & & & 0 & 0 & 0 & tn4 \\
 & & & & 0 & 0 & 0 & tn5 \\
0 & 0 & & A_{nO} & 0 & 0 & 0 & tn6 \\
 & & & & 0 & 0 & 0 & tn7 \\
 & & & & 0 & 0 & 0 & tn8 \\
 & & & & 0 & 0 & 0 & tn9 \\
 & & & & 0 & 0 & 0 & tn10 \\
 & & & & 1 & 0 & 1 & tn11 \\
 & & & & 1 & 0 & 1 & tn12 \\
 & & & & 0 & 1 & 0 & t(n*12+1) \\
0 & 0 & & 0 & 1 & 0 & 0 & t(n*12+2)
\end{array}\right)$$

Matricea de incidenta:

$$A=A_O-A_I=\left(\begin{array}{c|c|c|c|ccc|l}
\text{p11 p12 p13 p14 p15 p16 p17 p18} & \text{p21 p22 p23 p24 p25 p26 p27 p28} & & \text{pn1 pn2 pn3 pn4 pn5 pn6 pn7 pn8} & p(n8+1) & p(n8+2) & p(n8+3) & \\
 & & & & -1 & 0 & -1 & t11 \\
 & & & & 0 & 0 & 0 & t12 \\
 & & & & 0 & 0 & 0 & t13 \\
 & & & & 0 & 0 & 0 & t14 \\
 & & & & 0 & 0 & 0 & t15 \\
A_1 & 0 & & 0 & 0 & 0 & 0 & t16 \\
 & & & & 0 & 0 & 0 & t17 \\
 & & & & 0 & 0 & 0 & t18 \\
 & & & & 0 & 0 & 0 & t19 \\
 & & & & 0 & 0 & 0 & t110 \\
 & & & & 1 & 0 & 1 & t111 \\
 & & & & 1 & 0 & 1 & t112 \\
\hline
 & & & & -1 & 0 & -1 & t21 \\
 & & & & 0 & 0 & 0 & t22 \\
 & & & & 0 & 0 & 0 & t23 \\
 & & & & 0 & 0 & 0 & t24 \\
 & & & & 0 & 0 & 0 & t25 \\
0 & A_2 & & 0 & 0 & 0 & 0 & t26 \\
 & & & & 0 & 0 & 0 & t27 \\
 & & & & 0 & 0 & 0 & t28 \\
 & & & & 0 & 0 & 0 & t29 \\
 & & & & 0 & 0 & 0 & t210 \\
 & & & & 1 & 0 & 1 & t211 \\
 & & & & 1 & 0 & 1 & t212 \\
\hline
 & & & & & & & \\
\hline
 & & & & -1 & 0 & -1 & tn1 \\
 & & & & 0 & 0 & 0 & tn2 \\
 & & & & 0 & 0 & 0 & tn3 \\
 & & & & 0 & 0 & 0 & tn4 \\
 & & & & 0 & 0 & 0 & tn5 \\
0 & 0 & & A_n & 0 & 0 & 0 & tn6 \\
 & & & & 0 & 0 & 0 & tn7 \\
 & & & & 0 & 0 & 0 & tn8 \\
 & & & & 0 & 0 & 0 & tn9 \\
 & & & & 0 & 0 & 0 & tn10 \\
 & & & & 1 & 0 & 1 & tn11 \\
 & & & & 1 & 0 & 1 & tn12 \\
\hline
 & & & & -1 & 1 & 0 & t(n*12+1) \\
0 & 0 & & 0 & 1 & -1 & 0 & t(n*12+2)
\end{array}\right)$$

Arborele de acoperire corespunzator este prezentat in figura 2.117.

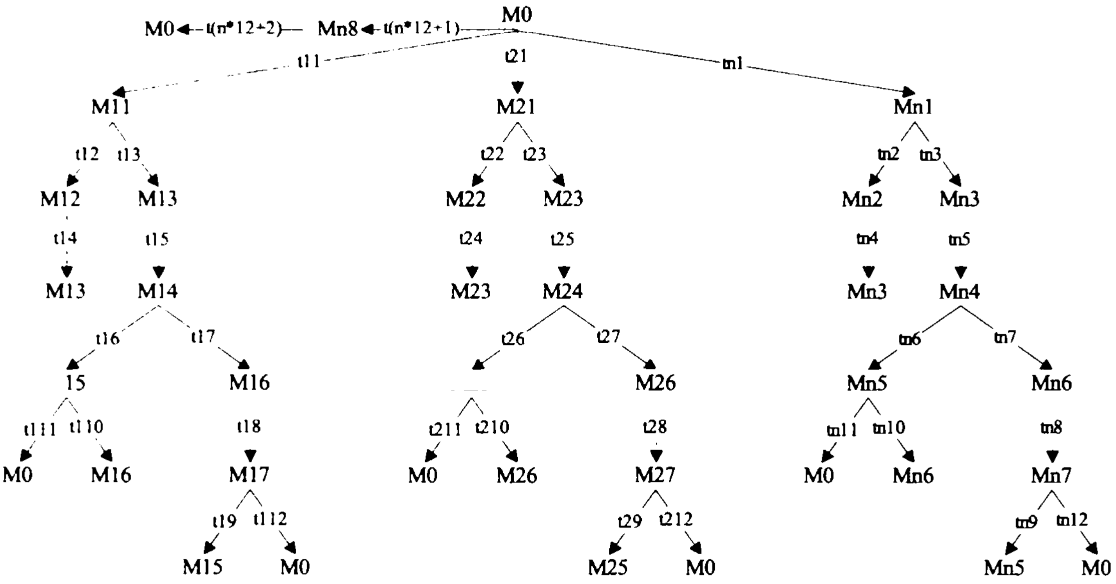


Figura 2.117. Arborele de acoperire al retelei Petri *PN_PT_Nn*

Unde:

M0=(1,0,0,0,0,0,0,0,1,0,0,0,0,0,0,0,.....,1,0,0,0,0,0,0,1,0,1);
M11=(0,1,0,0,0,0,0,0,1,0,0,0,0,0,0,0,.....,1,0,0,0,0,0,0,0,0,0);
M12=(0,0,1,0,0,0,0,0,1,0,0,0,0,0,0,0,.....,1,0,0,0,0,0,0,0,0,0);
M13=(0,0,0,1,0,0,0,0,1,0,0,0,0,0,0,0,.....,1,0,0,0,0,0,0,0,0,0);
M14=(0,0,0,0,1,0,0,0,1,0,0,0,0,0,0,0,.....,1,0,0,0,0,0,0,0,0,0);
M15=(0,0,0,0,0,1,0,0,1,0,0,0,0,0,0,0,.....,1,0,0,0,0,0,0,0,0,0);
M16=(0,0,0,0,0,0,1,0,1,0,0,0,0,0,0,0,.....,1,0,0,0,0,0,0,0,0,0);
M17=(0,0,0,0,0,0,0,1,1,0,0,0,0,0,0,0,.....,1,0,0,0,0,0,0,0,0,0);
M21=(1,0,0,0,0,0,0,0,0,1,0,0,0,0,0,0,.....,1,0,0,0,0,0,0,0,0,0);
M22=(1,0,0,0,0,0,0,0,0,0,1,0,0,0,0,0,.....,1,0,0,0,0,0,0,0,0,0);
M23=(1,0,0,0,0,0,0,0,0,0,0,1,0,0,0,0,.....,1,0,0,0,0,0,0,0,0,0);
M24=(1,0,0,0,0,0,0,0,0,0,0,0,1,0,0,0,.....,1,0,0,0,0,0,0,0,0,0);
M25=(1,0,0,0,0,0,0,0,0,0,0,0,0,1,0,0,.....,1,0,0,0,0,0,0,0,0,0);
M26=(1,0,0,0,0,0,0,0,0,0,0,0,0,0,1,0,.....,1,0,0,0,0,0,0,0,0,0);
M27=(1,0,0,0,0,0,0,0,0,0,0,0,0,0,0,1,.....,1,0,0,0,0,0,0,0,0,0);
Mn1=(1,0,0,0,0,0,0,0,1,0,0,0,0,0,0,0,.....,0,1,0,0,0,0,0,0,0,0);
Mn2=(1,0,0,0,0,0,0,0,1,0,0,0,0,0,0,0,.....,0,0,1,0,0,0,0,0,0,0);
Mn3=(1,0,0,0,0,0,0,0,1,0,0,0,0,0,0,0,.....,0,0,0,1,0,0,0,0,0,0);
Mn4=(1,0,0,0,0,0,0,0,1,0,0,0,0,0,0,0,.....,0,0,0,0,1,0,0,0,0,0);
Mn5=(1,0,0,0,0,0,0,0,1,0,0,0,0,0,0,0,.....,0,0,0,0,0,1,0,0,0,0);
Mn6=(1,0,0,0,0,0,0,0,1,0,0,0,0,0,0,0,.....,0,0,0,0,0,0,1,0,0,0);
Mn7=(1,0,0,0,0,0,0,0,1,0,0,0,0,0,0,0,.....,0,0,0,0,0,0,0,1,0,0);
Mn8=(1,0,0,0,0,0,0,0,1,0,0,0,0,0,0,0,.....,1,0,0,0,0,0,0,0,1,1)

Studiind proprietatile comportamentale ale retelei Petri *PN_AS_Nn*, pe baza arborelui de acoperire rezulta:

- reteaua este *marginita* (simbolul ω nu apare in arborele de acoperire);
- reteaua este *sigura* (marcajele din toate nodurile arborelui de acoperire contin numai „0” si „1”);

- reteaua *nu este blocanta* (toate tranzitile au asociate arce in arborele de acoperire);
- reteaua este *accesibila* (orice marcaj poate fi atins pornind din M_0).

Graful de acoperire corespunzator este prezentat in figura 2.118

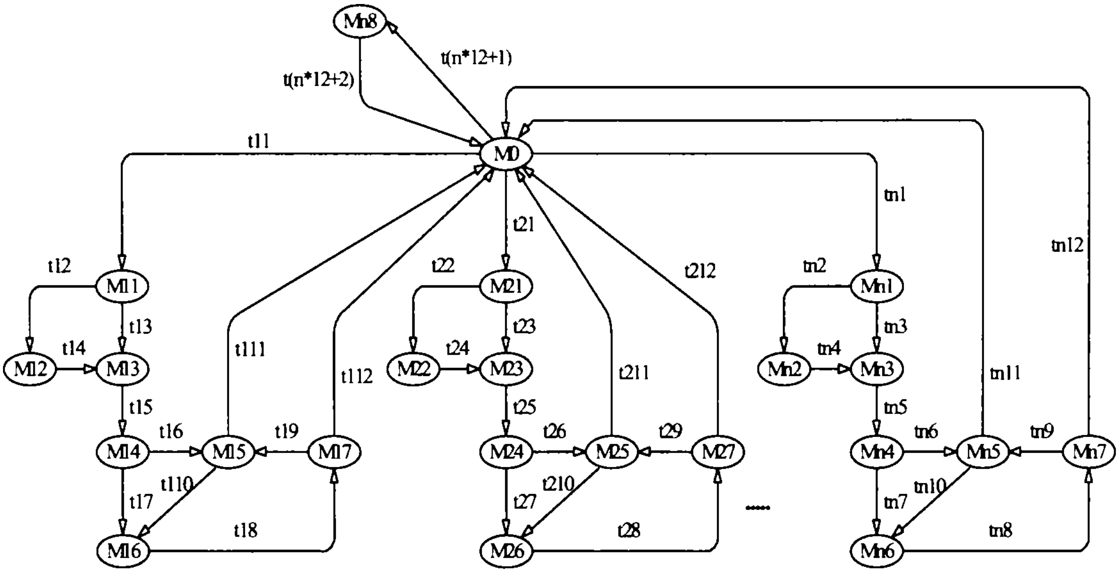


Figura 2.118. Graful de acoperire al retelei Petri *PN_AS_Nn*

Structura rezultata pentru graful de acoperire confirma ca modelul de retea Petri derivata din modelul de tip automat corespunzator nodului cun *n* intrari si o iesire este *acesibila*.

Concluzionand, se poate afirma ca modelul ales este *viabil* din punct de vedere comportamental.

Din punct de vedere structural, reteaua Petri *PN_PT_Nn,* este analizata pe baza analizei existentei invariantilor de tip *P* respectiv *T*.

Pentru determinarea numarului de invarianti *P* respectiv *T* s-a aplicat teorema 2.3, rangul matricii A este in acest caz:

$$rang\,A = nr\_nod \cdot rangN_1$$

unde:

- *nr_nod* – reprezinta numarul intrarilor;
- $rangN_1$ – reprezinta matricii de incidenta a nodului de tip 1 ($rangN_1=7$).

Numarul de invarianti *P* corespunzatori se poate calcula cu relatia:

$$Nr\_Inv\_P = m - rang\,A = m - nr\_nod \cdot rangN_1$$

Numarul de invarianti *T* corespunzatori este:

$$Nr\_Inv\_T = n - rang\,A = n - nr\_nod \cdot rangN_1$$

2.5.2.6.2.1. Cazul: retea Petri temporizata P

Structura retelei Petri temporizata P corespunzatoare nodului cu *n* intrari si *o* iesire este aceeasi ca cea prezentata in figura 2.115.

Datorita faptului ca retelele sunt identice din punct de vedere structural si comportamental rezultatele analizelor nu se mai detaliaza, ele fiind identice cu cele obtinute in cazul nodului netemporizat.

Fata de situatia prezentata anterior fiecarei pozitii P i-a fost alocata o unitate de timp corespunzatoare. In acest mod situatia devine reala din punct de vedere al functionarii.

Timpii alocati pozitiilor sunt prezentati in tabelul 2.14.

Tabelul 2.14. *Timpii corespunzatori pozitiilor P*

Pozitie	Unitati de Timp
Pi1	1
Pi2	5
Pi3	4
Pi4	4
Pi5	7
Pi6	6
Pi7	3
Pi8	5
P(n*8+1)	1
P(n*8+2)	3
P(n*8+3)	1

unde i=1,...,n.

Reteaua Petri considerata este formalizata prin sixtuplul:

$$PN\_PTtime\_Nn = (P,T,F,W,D,M_0)$$

unde:

- $P = \bigcup_{i=1}^{n} \{p_{i1}, p_{i2}, p_{i3}, p_{i4}, p_{i5}, p_{i6}, p_{i7}, p_{i8}\} \cup \{p_{(n8+1)}, p_{(n8+2)}, p_{(n8+3)}\}$
- $T = \bigcup_{i=1}^{n} \{t_{i1}, t_{i2}, t_{i3}, t_{i4}, t_{i5}, t_{i6}, t_{i7}, t_{i8}, t_{i9}, t_{i10}, t_{i11}, t_{i12}\} \cup \{t_{(n12+1)}, t_{(n12+2)}\}$

- $$\begin{aligned}F = \bigcup_{i=1}^{n} \{ & \{(p_{i1},t_{i1}),(p_{i2},t_{i2}),(p_{i2},t_{i3}),(p_{i3},t_{i4}),(p_{i4},t_{i5}),(p_{i5},t_{i6}),\\ & (p_{i5},t_{i7}),(p_{i6},t_{i10}),(p_{i6},t_{i11}),(p_{i7},t_{i8}),(p_{i8},t_{i9}),(p_{i8},t_{i12}),\\ & (p_{n8+1},t_{i1}),(p_{n8+3},t_{i1})\} \cup\\ & \{(t_{i1},p_{i2}),(t_{i2},p_{i3}),(t_{i3},p_{i4}),(t_{i4},p_{i4}),(t_{i5},p_{i5}),(t_{i6},p_{i6}),\\ & (t_{i7},p_{i7}),(t_{i8},p_{i8}),(t_{i9},p_{i6}),(t_{i10},p_{i7}),(t_{i11},p_{i1}),\\ & (t_{i11},p_{n8+1}),(t_{i11},p_{n8+3}),(t_{i12},p_{i1}),(t_{i12},p_{n8+1}),\\ & (t_{i12},p_{n8+3})\}\} \cup \{\{(p_{n8+1},t_{n12+1}),(p_{n8+2},t_{n12+2})\}\\ & \cup \{(t_{n12+1},p_{n8+2}),(t_{n12+2},p_{n8+1})\}\}\end{aligned}$$
- $$\begin{aligned}& W(p_{i1},t_{i1}) = 1, W(p_{i2},t_{i2}) = 1, W(p_{i2},t_{i3}) = 1, W(p_{i3},t_{i4}) = 1,\\ & W(p_{i4},t_{i5}) = 1, W(p_{i5},t_{i6}) = 1, W(p_{i5},t_{i7}) = 1, W(p_{i6},t_{i10}) = 1,\\ & W(p_{i6},t_{i11}) = 1, W(p_{i7},t_{i8}) = 1, W(p_{i8},t_{i9}) = 1, W(p_{i8},t_{i12}) = 1,\\ & W(p_{n8+1},t_{i1}) = 1, W(p_{n8+3},t_{i1}) = 1, W(p_{n8+1},t_{n12+1}) = 1,\\ & W(p_{n8+2},t_{n12+2}) = 1, W(t_{i1},p_{i2}) = 1, W(t_{i2},p_{i3}) = 1, W(t_{i3},p_{i4}) = 1,\\ & W(t_{i4},p_{i4}) = 1, W(t_{i5},p_{i5}) = 1, W(t_{i6},p_{i6}) = 1,\\ & W(t_{i7},p_{i7}) = 1, W(t_{i8},p_{i8}) = 1, W(t_{i9},p_{i6}) = 1, W(t_{i10},p_{i7}) = 1,\\ & W(t_{i11},p_{i1}) = 1, W(t_{i11},p_{n8+1}) = 1, W(t_{i11},p_{n8+3}) = 1, W(t_{i12},p_{i1}) = 1,\\ & W(t_{i12},p_{n8+1}) = 1, W(t_{i12},p_{n8+3}) = 1, W(t_{n12+1},p_{n8+2}) = 1,\\ & W(t_{n12+2},p_{n8+1}) = 1\end{aligned}$$
- $$\begin{aligned}D = \{ & d(p_{i1}) = 1, d(p_{i2}) = 5, d(p_{i3}) = 4, d(p_{i4}) = 4, d(p_{i5}) = 7,\\ & d(p_{i6}) = 6, d(p_{i7}) = 3, d(p_{i8}) = 5, d(p_{n\cdot 8+1}) = 1,\\ & d(p_{n\cdot 8+2}) = 3, d(p_{n\cdot 8+3}) = 1\}\end{aligned}$$
- $$M_0 = [1, 0,0,0,0,0,0,0,1,0,0,0,0,0,0,0,......,1,0,0,0,0,0,0,0,1,0,1]^T$$

2.5.2.7. Modelarea unui nod cu *o* intrare si *m* iesiri

2.5.2.7.1. Modelarea nodului „o" intrare si „m" iesiri cu ajutorul automatelor

Structura automatului secvential considerat pentru modelarea unui astfel de nod este prezentata în figura 2.119.

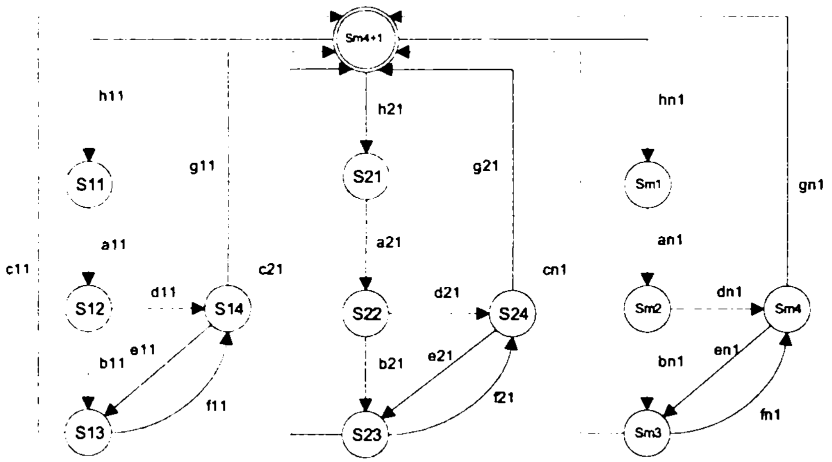


Figura 2.119. Structura automatului secvential corespunzător nodului cu „*o*" intrare si „*m*" iesiri

Automatul secvential al nodului este realizat prin sincronizarea a *m* automate secventiale AS_N4, cate unul pentru fiecare linie de intrare (stoper).

Matematic, automatul *AS_Nm* corespunzator nodului cu o intrare si m iesiri este descris astfel:

- multimea starilor: $X = \bigcup_{i=1}^{m} \{Si1, Si2, Si3, Si4\} \cup \{S(4 \cdot m + 1)\}$
- multimea evenimentelor: $\Sigma = \bigcup_{i=1}^{m} \{ai1, bi1, ci1, di1, ei1, fi1, gi1, hi1\}$
- multimile evenimentelor posibile si functiile de tranzitie a starilor:

$\Gamma(S) = \bigcup_{i=1}^{m} \Gamma(Si)$ unde:

$\Gamma(Si1) = \{ai\}$ $\delta(Si1, ai) = Si2$

$\Gamma(Si2) = \{bi, di\}$ $\delta(Si2, bi) = Si3,$ $\delta(Si2, di) = Si4$

$\Gamma(Si3) = \{ci, fi\}$ $\delta(Si3, ci) = S(4 \cdot m + 1),$ $\delta(Si3, fi) = Si4$

$\Gamma(Si4) = \{ei, gi\}$ $\delta(Si4, ei) = Si3$ $\delta(Si4, gi) = S(4 \cdot m + 1)$

$\Gamma(S(4 \cdot n + 1) = \bigcup_{i=1}^{m} \{hi\}$ $\delta(S(4 \cdot m + 1), hi) = Si1$

- starea initiala: $x_0 = S(4 \cdot m + 1)$.

Considerentele legate de transformarea modelului de tip automat in model de tip retea Petri, prezentate la nodul de tip 4, sunt valabile si in cazul nodului cu *o* intrare si m iesiri.

Pentru conversia din model de tip automat in model de tip retea Petri s-a utilizat acceasi metoda ca si in cazul nodului de tip 1 .

Aplicand algoritmul 2.1 rezulta:

- multimea pozitiilor: $P = \bigcup_{i=1}^{m} \{pi1, pi2, pi3, pi4\} \cup \{p(4m+1)\}$, unde

  $pi1 = \{Si1\}, pi2 = \{Si2\}, pi3 = \{Si3\}, pi4 = \{Si4\}$ si

  $p(4 \cdot m + 1) = \{S(4 \cdot m + 1\}$;
- multimea trranzitiilor:

• Pentru *pi1*	*(pi1,pi2)*	$pi2=\delta(pi1,ti1)$	$ti1=\{ai\}$
• Pentru *pi2*	*(pi2,pi3)*	$pi3=\delta(pi2,ti2)$	$ti2=\{bi\}$
	(pi2,pi4)	$pi4=\delta(pi2,ti4)$	$ti4=\{di\}$
• Pentru *pi3*	*(pi3,p(4m+1))*	$p(4m+1)=\delta(pi3,ti3)$	$ti3=\{ci\}$
	(pi3,pi4)	$pi4=\delta(pi3,ti6)$	$ti6=\{fi\}$
• Pentru *pi4*	*(pi4,p(4m+1))*	$p(4m+1)=\delta(pi4,ti7)$	$ti7=\{gi\}$
	(pi4,pi3)	$pi3=\delta(pi4,ti5)$	$ti5=\{ei\}$
• Pentru *p(4m+1)*	*(p(4m+1),pi1)*	$pi1=\delta(p4m+1),ti8)$	$ti8=\{hi\}$

In figura 2.120 se prezinta structura retelei Petri asociata automatului secvential considerat in cazul nodului cu *o* intrare si *m* iesiri.

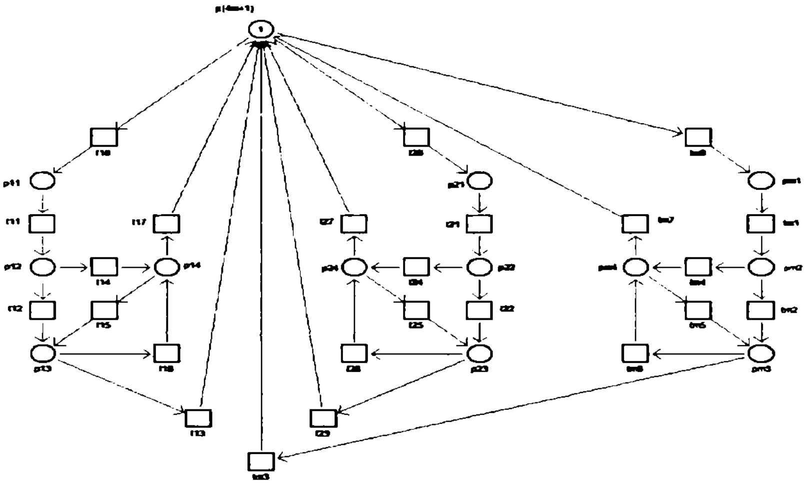


Figura 2.120. Structura reţelei Petri asociata automatului secvential corespunzător nodului cu *o* intrare si *m* iesiri

Topologia retelei validata de PNT este prezentata in figura 2.121.

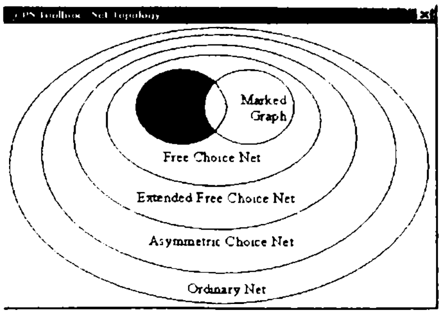


Figura 2.121. Topologia retelei Petri echivalenta validata de PNT

Rezultatele obtinute confirma ca dupa efectuarea conversiei din modelul de tip automat in model de tip retea Petri, topologia retelei Petri obtinuta este tot de tip automat („State machine").

Reteaua Petri considerata este formalizata prin cvintuplul:

$$PN\_AS\_Nm = (P, T, F, W, M_0)$$

unde:

- $P = \bigcup_{i=1}^{m} \{p_{i1}, p_{i2}, p_{i3}, p_{i4}\} \cup \{p_{4m+1}\}$
- $T = \bigcup_{i=1}^{m} \{t_{i1}, t_{i2}, t_{i3}, t_{i4}, t_{i5}, t_{i6}, t_{i7}, t_{i8}\}$
- $F = \bigcup_{i=1}^{m} \{\{(p_{i1}, t_{i1}), (p_{i2}, t_{i2}), (p_{i2}, t_{i4}), (p_{i3}, t_{i6}), (p_{i3}, t_{i3}), (p_{i4}, t_{i5}),$ $(p_{i4}, t_{i7}), (p_{4m+1}, t_{i8})\} \cup$ $\{(t_{i1}, p_{i2}), (t_{i2}, p_{i3}), (t_{i3}, p_{4m+1}), (t_{i4}, p_{i4}), (t_{i5}, p_{i3}),$ $(t_{i6}, p_{i4}), (t_{i7}, p_{4n+1}), (t_{i8}, p_{i1})\}\}$
- $W(p_{i1}, t_{i1}) = 1, W(p_{i2}, t_{i2}) = 1, W(p_{i2}, t_{i4}) = 1, W(p_{i3}, t_{i6}) = 1,$ $W(p_{i3}, t_{i3}) = 1, W(p_{i4}, t_{i5}) = 1, W(p_{i4}, t_{i7}) = 1, W(p_{4m+1}, t_{i8}) = 1,$ $W(t_{i1}, p_{i2}) = 1, W(t_{i2}, p_{i3}) = 1, W(t_{i3}, p_{4m+1}) = 1, W(t_{i4}, p_{i4}) = 1,$ $W(t_{i5}, p_{i3}) = 1, W(t_{i6}, p_{i4}) = 1, W(t_{i7}, p_{4m+1}) = 1, W(t_{i8}, p_{i1}) = 1,$
- $M_0 = [0,0,0,0,0,0,0,0,0,0,0,0,...,1]^T$

Matricile de incidenţă corespunzătoare stoperului (intrarii) i , unde coloanele sunt p_{i1}, p_{i2}, p_{i3} si p_{i4} iar linile: t_{i1}, t_{i2}, t_{i3}, t_{i4}, t_{i5}, t_{i6}, t_{i7} si t_{i8} sunt.

Matricea de incidenta de intrare:

$$A_{Ii} = \begin{pmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 1 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \\ 0 & 0 & 0 & 0 \end{pmatrix}$$

Matricea de incidenta de iesire:

$$A_{Oi} = \begin{pmatrix} 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \\ 0 & 0 & 0 & 0 \\ 1 & 0 & 0 & 0 \end{pmatrix}$$

Matricea de incidenta:

$$A_i = A_{Oi} - A_{Ii} = \begin{pmatrix} -1 & 1 & 0 & 0 \\ 0 & -1 & 1 & 0 \\ 0 & 0 & -1 & 0 \\ 0 & -1 & 0 & 1 \\ 0 & 0 & 1 & -1 \\ 0 & 0 & -1 & 1 \\ 0 & 0 & 0 & -1 \\ 1 & 0 & 0 & 0 \end{pmatrix}$$

Avand structurile matricilor pentru un stoper, matricile corespunzatoare nodului cu *o* intrare (un stoper) si *m* iesiri sunt.

Matricea de intrare:

$$A_I=\left(\begin{array}{c|c|c|c|c}
p11\ p12\ p13\ p14 & p21\ p22\ p23\ p24 & & pm1\ pm2\ pm3\ pm4 & p4m+1 \\
A_{1I} & 0 & & 0 & \begin{matrix}0\\0\\0\\0\\0\\0\\0\\1\end{matrix} \\ \hline
0 & A_{2I} & & 0 & \begin{matrix}0\\0\\0\\0\\0\\0\\0\\1\end{matrix} \\ \hline
 & & & & \\ \hline
0 & 0 & & A_{mI} & \begin{matrix}0\\0\\0\\0\\0\\0\\0\\1\end{matrix}
\end{array}\right)
\begin{array}{l}
\\
\begin{matrix}t11\\t12\\t13\\t14\\t15\\t16\\t17\\t18\end{matrix}\\
\begin{matrix}t21\\t22\\t23\\t24\\t25\\t26\\t27\\t28\end{matrix}\\
\\
\begin{matrix}tm1\\tm2\\tm3\\tm4\\tm5\\tm6\\tm7\\tm8\end{matrix}
\end{array}$$

Matricea de iesire:

$$A_O=\left(\begin{array}{c|c|c|c|c}
p11\ p12\ p13\ p14 & p21\ p22\ p23\ p24 & & pm1\ pm2\ pm3\ pm4 & p4m+1 \\
A_{1O} & 0 & & 0 & \begin{matrix}0\\0\\0\\0\\0\\0\\0\\1\end{matrix} \\ \hline
0 & A_{2O} & & 0 & \begin{matrix}0\\0\\0\\0\\0\\0\\0\\1\end{matrix} \\ \hline
 & & & & \\ \hline
0 & 0 & & A_{mO} & \begin{matrix}0\\0\\0\\0\\0\\0\\0\\1\end{matrix}
\end{array}\right)
\begin{array}{l}
\\
\begin{matrix}t11\\t12\\t13\\t14\\t15\\t16\\t17\\t18\end{matrix}\\
\begin{matrix}t21\\t22\\t23\\t24\\t25\\t26\\t27\\t28\end{matrix}\\
\\
\begin{matrix}tm1\\tm2\\tm3\\tm4\\tm5\\tm6\\tm7\\tm8\end{matrix}
\end{array}$$

Matricea de incidenta:

$$A=A_O-A_I=\left(\begin{array}{c|c|c|c|c}
p_{11}\;p_{12}\;p_{13}\;p_{14} & p_{21}\;p_{22}\;p_{23}\;p_{24} & & p_{m1}\;p_{m2}\;p_{m3}\;p_{m4} & p_{4m+1}\\
 & & & & 0\\
 & & & & 0\\
 & & & & 0\\
A_1 & 0 & & 0 & 0\\
 & & & & 0\\
 & & & & 0\\
 & & & & 0\\
 & & & & 1\\
\hline
 & & & & 0\\
 & & & & 0\\
 & & & & 0\\
0 & A_2 & & 0 & 0\\
 & & & & 0\\
 & & & & 0\\
 & & & & 0\\
 & & & & 1\\
\hline
 & & & & \\
\hline
 & & & & 0\\
 & & & & 0\\
 & & & & 0\\
0 & 0 & & A_m & 0\\
 & & & & 0\\
 & & & & 0\\
 & & & & 0\\
 & & & & 1
\end{array}\right)\begin{array}{l}
\\ t_{11}\\ t_{12}\\ t_{13}\\ t_{14}\\ t_{15}\\ t_{16}\\ t_{17}\\ t_{18}\\ t_{21}\\ t_{22}\\ t_{23}\\ t_{24}\\ t_{25}\\ t_{26}\\ t_{27}\\ t_{28}\\ \\ t_{m1}\\ t_{m2}\\ t_{m3}\\ t_{m4}\\ t_{m5}\\ t_{m6}\\ t_{m7}\\ t_{m8}
\end{array}$$

Arborele de acoperire corespunzator este prezentat in figura 2.122.

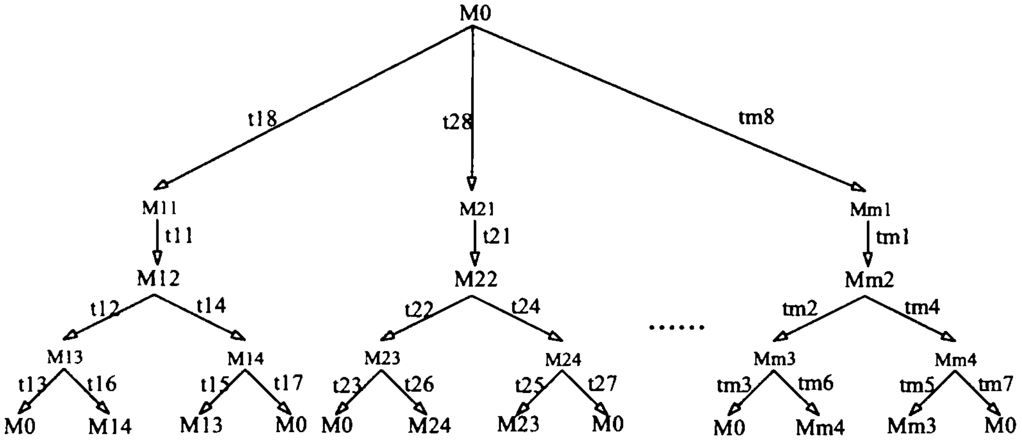


Figura 2.122. Arborele de acoperire al retelei Petri *PN_AS_Nn*

Unde:
M0=(0,0,0,0,0,0,0,0,...,0,0,0,0,1); M11=(1,0,0,0,0,0,0,0,...,0,0,0,0,0);
M12=(0,1,0,0,0,0,0,0,...,0,0,0,0,0); M13=(0,0,1,0,0,0,0,0,...,0,0,0,0,0);
M14=(0,0,0,1,0,0,0,0,...,0,0,0,0,0); M21=(0,0,0,0,1,0,0,0,...,0,0,0,0,0);
M22=(0,0,0,0,0,1,0,0,...,0,0,0,0,0); M23=(0,0,0,0,0,0,1,0,...,0,0,0,0,0);
M24=(0,0,0,0,0,0,0,1,...,0,0,0,0,0); Mm1=(0,0,0,0,0,0,0,0,...,1,0,0,0,0);
Mm2=(0,0,0,0,0,0,0,0,...,0,1,0,0,0); Mm3=(0,0,0,0,0,0,0,0,...,0,0,1,0,0);
Mm4=(0,0,0,0,0,0,0,0,...,0,0,0,1,0)

Studiind proprietatile comportamentale ale retelei Petri *PN_AS_Nm*, pe baza arborelui de acoperire rezulta:

- reteaua este *marginita* (simbolul ω nu apare in arborele de acoperire);

- reteaua este *sigura* (marcajele din toate nodurile arborelui de acoperire contin numai „0” si „1”);
- reteaua *nu este blocanta* (toate tranzitile au asociate arce in arborele de acoperire);
- reteaua este *accesibila* (orice marcaj poate fi atins pornind din M_0).

Graful de acoperire corespunzator este prezentat in figura 2.123.

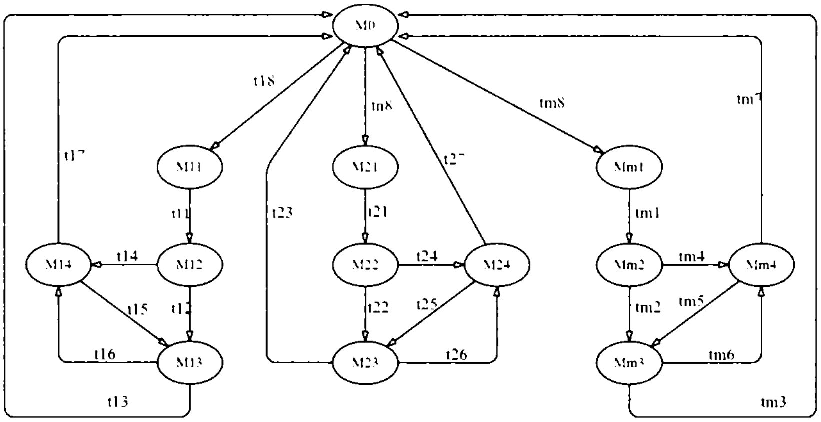


Figura 2.123. Graful de acoperire al retelei Petri *PN_AS_Nm*

Structura rezultata pentru graful de acoperire confirma ca modelul de retea Petri derivata din modelul de tip automat corespunzator nodului cun *n* intrari si o iesire este *acesibila*.

Concluzionand, se poate afirma ca modelul ales este *viabil* din punct de vedere comportamental.

Analiza retelei Petri *PN_AS_Nm* din punct de vedere structural se face pe baza analizei existentei invariantilor de tip *P* respectiv *T*.

Pentru determinarea numarului de invarianti *P* respectiv *T* s-a aplicat teorema 2.3.

Rangul matricii A este in acest caz:

$$rang\,A = nr\_ies\_nod \cdot rangN_1$$

unde:

- *nr_ies_nod* – reprezinta numarul de iesiri ale nodului;
- *rangN₁* – reprezinta rangul maticii de incidenta corespunzatoare nodului de tip 1 ($rangN_1=4$).

Numarul de invarianti *P* corespunzatori se poate calcula cu:

$$Nr\_Inv\_P = m - rang\,A = m - nr\_ies\_nod \cdot rangN_1$$

iar numarul de invarianti *T* cu:

$$Nr\_Inv\_T = n - rang\,A = n - nr\_ies\_nod \cdot rangN_1$$

2.5.2.7.2. Modelarea nodului cu *o* intrare si *m* iesiri cu ajutorul retelelor Petri

2.5.2.7.2.1. Cazul: retea Petri netemporizata

Structura retelei Petri considerate pentru modelarea nodului cu *o* intrare si *m* iesiri este prezentata în figura 2.124.

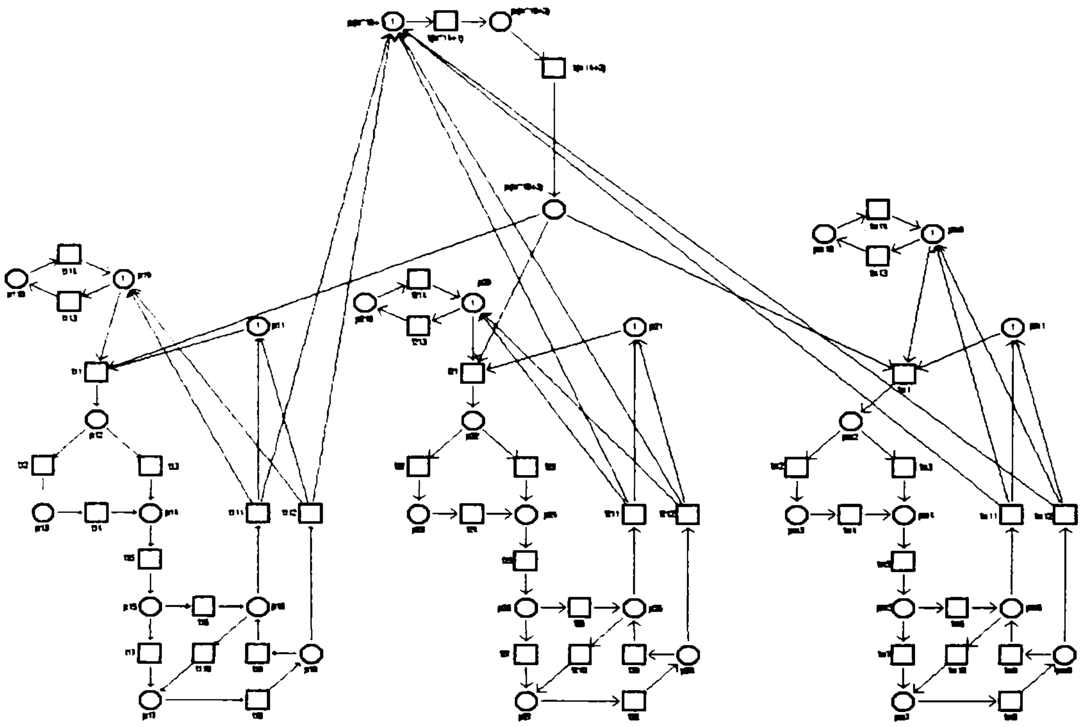


Figura 2.124. Structura retelei Petri corespunzătoare nodului cu *o* intrare si *m* iesiri

Dupa cum se observa automatul secvential al nodului este realizat prin sincronizarea a *m* retele Petri *PN_PT_N1*, in mod similar cu cele prezentate la nodul de tip 4.

Topologia retelei Petri validata de PNT este prezentata in figura 2.125.

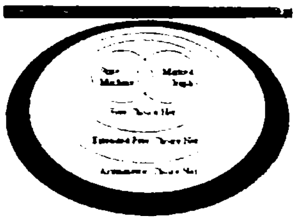


Figura 2.125. Topologia retelei Petri validata de PNT

Reteaua Petri considerata este formalizata prin cvintuplul:

$$PN\_PT\_Nm = (P,T,F,W,M_0)$$

unde:

- $$P = \bigcup_{i=1}^{m} \{p_{i1}, p_{i2}, p_{i3}, p_{i4}, p_{i5}, p_{i6}, p_{i7}, p_{i8}, p_{i9}, p_{i10}\} \cup \{p_{(m10+1)}, p_{(m10+2)}, p_{(m8+3)}\}$$
- $$T = \bigcup_{i=1}^{m} \{t_{i1}, t_{i2}, t_{i3}, t_{i4}, t_{i5}, t_{i6}, t_{i7}, t_{i8}, t_{i9}, t_{i10}, t_{i11}, t_{i12}, t_{i13}, t_{i14}\} \cup \{t_{(m14+1)}, t_{(m14+2)}\}$$
- $$\begin{aligned} F = \bigcup_{i=1}^{m} \{\{&(p_{i1},t_{i1}),(p_{i2},t_{i2}),(p_{i2},t_{i3}),(p_{i3},t_{i4}),(p_{i4},t_{i5}),(p_{i5},t_{i6}),\\ &(p_{i5},t_{i7}),(p_{i6},t_{i10}),(p_{i6},t_{i11}),(p_{i7},t_{i8}),(p_{i8},t_{i9}),(p_{i8},t_{i12}),\\ &(p_{n9},t_{i13}),(p_{n10},t_{i14}),(p_{m10+3},t_{i1})\} \cup\\ &\{(t_{i1},p_{i2}),(t_{i2},p_{i3}),(t_{i3},p_{i4}),(t_{i4},p_{i4}),(t_{i5},p_{i5}),(t_{i6},p_{i6}),\\ &(t_{i7},p_{i7}),(t_{i8},p_{i8}),(t_{i9},p_{i6}),(t_{i10},p_{i7}),(t_{i11},p_{i1}),(t_{i11},p_{n9}),\\ &(t_{i11},p_{(m10+1)}),(t_{i12},p_{i1}),(t_{i12},p_{(m10+1)}),(t_{i13},p_{i10}),\\ &(t_{i14},p_{i9})\}\} \cup \{\{(p_{m10+1},t_{m14+1}),(p_{m10+2},t_{m14+2})\} \cup\\ &\{(t_{m14+1},p_{m10+2}),(t_{m14+2},p_{m10+3})\}\} \end{aligned}$$
- $$\begin{aligned} &W(p_{i1},t_{i1}) = 1, W(p_{i2},t_{i2}) = 1, W(p_{i2},t_{i3}) = 1, W(p_{i3},t_{i4}) = 1,\\ &W(p_{i4},t_{i5}) = 1, W(p_{i5},t_{i6}) = 1, W(p_{i5},t_{i7}) = 1,\\ &W(p_{i6},t_{i10}) = 1, W(p_{i6},t_{i11}) = 1, W(p_{i7},t_{i8}) = 1,\\ &W(p_{i8},t_{i9}) = 1, W(p_{i8},t_{i12}) = 1, W(p_{n9},t_{i13}) = 1,\\ &W(p_{n10},t_{i14}) = 1, W(p_{m10+3},t_{i1}) = 1, W(t_{i1},p_{i2}) = 1,\\ &W(t_{i2},p_{i3}) = 1, W(t_{i3},p_{i4}) = 1, W(t_{i4},p_{i4}) = 1,\\ &W(t_{i5},p_{i5}) = 1, W(t_{i6},p_{i6}) = 1, W(t_{i7},p_{i7}) = 1, W(t_{i8},p_{i8}) = 1,\\ &W(t_{i9},p_{i6}) = 1, W(t_{i10},p_{i7}) = 1, W(t_{i11},p_{i1}) = 1, W(t_{i11},p_{n9}) = 1,\\ &W(t_{i11},p_{(m10+1)}) = 1, W(t_{i12},p_{i1}) = 1, W(t_{i12},p_{(m10+1)}) = 1,\\ &W(t_{i13},p_{i10}) = 1, W(t_{i14},p_{i9}) = 1, W(p_{m10+1},t_{m14+1}) = 1,\\ &W(p_{m10+2},t_{m14+2}) = 1, W(t_{m14+1},p_{m10+2}) = 1, W(t_{m14+2},p_{m10+3}) = 1 \end{aligned}$$
- $$M_0 = [1, 0,0,0,0,0,0,0,1,0,1,0,0,0,0,0,0,0,1,0,......,1,0,0,0,0,0,0,0,1,0,1,0,0]^T$$

Matricile de incidenţă corespunzătoare stoperului (intrarii) i , unde coloanele sunt p_{i1}, p_{i2}, p_{i3} , p_{i4} , p_{i5}, p_{i6}, p_{i7}, p_{i8} , p_{i9} si p_{i10} iar linile: t_{i1}, t_{i2}, t_{i3}, t_{i4}, t_{i5}, t_{i6}, t_{i7}, t_{i8}, t_{i9}, t_{i10}, t_{i11}, t_{i12} t_{i13} si t_{i14} sunt.

Matricea de incidenta de intrare:

$$A_{Ii} = \begin{pmatrix} 1000000010 \\ 0100000000 \\ 0100000000 \\ 0010000000 \\ 0001000000 \\ 0000100000 \\ 0000100000 \\ 0000001000 \\ 0000000100 \\ 0000010000 \\ 0000010000 \\ 0000000100 \\ 0000000010 \\ 0000000001 \end{pmatrix}$$

Matricea de incidenta de iesire:

$$A_{Oi} = \begin{pmatrix} 0100000000 \\ 0010000000 \\ 0001000000 \\ 0001000000 \\ 0000100000 \\ 0000010000 \\ 0000001000 \\ 0000000100 \\ 0000010000 \\ 0000001000 \\ 1000000010 \\ 1000000010 \\ 0000000001 \\ 0000000010 \end{pmatrix}$$

Matricea de incidenta:

$$A_i = A_{Oi} - A_{Ii} = \begin{pmatrix} -1 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & -1 & 0 \\ 0 & -1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & -1 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & -1 & 1 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & -1 & 1 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & -1 & 1 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & -1 & 0 & 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & -1 & 1 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 1 & 0 & -1 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & -1 & 1 & 0 & 0 & 0 \\ 1 & 0 & 0 & 0 & 0 & -1 & 0 & 0 & 1 & 0 \\ 1 & 0 & 0 & 0 & 0 & 0 & 0 & -1 & 1 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & -1 & 1 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & -1 \end{pmatrix}$$

Avand structurile matricilor pentru un stoper, matricile corespunzatoare nodului cu *o* intrare (un stoper) si *m* iesiri sunt.

Matricea de intrare:

$$A_I=\begin{array}{c|c|c|c|ccc|l}
\text{p11 p12 p13 p14 p15 p16 p17 p18 p19 p10} & \text{p21 p22 p23 p24 p25 p26 p27 p28 p29 p210} & \cdots & \text{pm1 pm2 pm3 pm4 pm5 pm6 pm7 pm8 pm9 pm10} & \text{p(m10+1)} & \text{p(m10+2)} & \text{p(m10+3)} & \\
\hline
 & & & & 0 & 0 & 1 & \text{t11} \\
 & & & & 0 & 0 & 0 & \text{t12} \\
 & & & & 0 & 0 & 0 & \text{t13} \\
 & & & & 0 & 0 & 0 & \text{t14} \\
 & & & & 0 & 0 & 0 & \text{t15} \\
 & & & & 0 & 0 & 0 & \text{t16} \\
A_{1I} & 0 & & 0 & 0 & 0 & 0 & \text{t17} \\
 & & & & 0 & 0 & 0 & \text{t18} \\
 & & & & 0 & 0 & 0 & \text{t19} \\
 & & & & 0 & 0 & 0 & \text{t110} \\
 & & & & 0 & 0 & 0 & \text{t111} \\
 & & & & 0 & 0 & 0 & \text{t112} \\
 & & & & 0 & 0 & 0 & \text{t113} \\
 & & & & 0 & 0 & 0 & \text{t114} \\
\hline
 & & & & 0 & 0 & 1 & \text{t21} \\
 & & & & 0 & 0 & 0 & \text{t22} \\
 & & & & 0 & 0 & 0 & \text{t23} \\
 & & & & 0 & 0 & 0 & \text{t24} \\
 & & & & 0 & 0 & 0 & \text{t25} \\
 & & & & 0 & 0 & 0 & \text{t26} \\
0 & A_{2I} & & 0 & 0 & 0 & 0 & \text{t27} \\
 & & & & 0 & 0 & 0 & \text{t28} \\
 & & & & 0 & 0 & 0 & \text{t29} \\
 & & & & 0 & 0 & 0 & \text{t210} \\
 & & & & 0 & 0 & 0 & \text{t211} \\
 & & & & 0 & 0 & 0 & \text{t212} \\
 & & & & 0 & 0 & 0 & \text{t213} \\
 & & & & 0 & 0 & 0 & \text{t214} \\
\hline
 & & & & & & & \\
\hline
 & & & & 0 & 0 & 1 & \text{tm1} \\
 & & & & 0 & 0 & 0 & \text{tm2} \\
 & & & & 0 & 0 & 0 & \text{tm3} \\
 & & & & 0 & 0 & 0 & \text{tm4} \\
 & & & & 0 & 0 & 0 & \text{tm5} \\
 & & & & 0 & 0 & 0 & \text{tm6} \\
0 & 0 & & A_{mI} & 0 & 0 & 0 & \text{tm7} \\
 & & & & 0 & 0 & 0 & \text{tm8} \\
 & & & & 0 & 0 & 0 & \text{tm9} \\
 & & & & 0 & 0 & 0 & \text{tm10} \\
 & & & & 0 & 0 & 0 & \text{tm11} \\
 & & & & 0 & 0 & 0 & \text{tm12} \\
 & & & & 0 & 0 & 0 & \text{tm13} \\
 & & & & 0 & 0 & 0 & \text{tm14} \\
\hline
0 & 0 & & 0 & 1 & 0 & 0 & \text{t(m14+1)} \\
 & & & & 0 & 1 & 0 & \text{t(m14+2)}
\end{array}$$

Matricea de iesire:

$$A_O=\begin{array}{c|c|c|c|ccc|l}
\text{p11 p12 p13 p14 p15 p16 p17 p18 p19 p10} & \text{p21 p22 p23 p24 p25 p26 p27 p28 p29 p210} & \cdots & \text{pm1 pm2 pm3 pm4 pm5 pm6 pm7 pm8 pm9 pm10} & \text{p(m10+1)} & \text{p(m10+2)} & \text{p(m10+3)} & \\
\hline
 & & & & 0 & 0 & 0 & \text{t11} \\
 & & & & 0 & 0 & 0 & \text{t12} \\
 & & & & 0 & 0 & 0 & \text{t13} \\
 & & & & 0 & 0 & 0 & \text{t14} \\
 & & & & 0 & 0 & 0 & \text{t15} \\
 & & & & 0 & 0 & 0 & \text{t16} \\
A_{1O} & 0 & & 0 & 0 & 0 & 0 & \text{t17} \\
 & & & & 0 & 0 & 0 & \text{t18} \\
 & & & & 0 & 0 & 0 & \text{t19} \\
 & & & & 0 & 0 & 0 & \text{t110} \\
 & & & & 1 & 0 & 0 & \text{t111} \\
 & & & & 1 & 0 & 0 & \text{t112} \\
 & & & & 0 & 0 & 0 & \text{t113} \\
 & & & & 0 & 0 & 0 & \text{t114} \\
\hline
 & & & & 0 & 0 & 0 & \text{t21} \\
 & & & & 0 & 0 & 0 & \text{t22} \\
 & & & & 0 & 0 & 0 & \text{t23} \\
 & & & & 0 & 0 & 0 & \text{t24} \\
 & & & & 0 & 0 & 0 & \text{t25} \\
 & & & & 0 & 0 & 0 & \text{t26} \\
0 & A_{2O} & & 0 & 0 & 0 & 0 & \text{t27} \\
 & & & & 0 & 0 & 0 & \text{t28} \\
 & & & & 0 & 0 & 0 & \text{t29} \\
 & & & & 0 & 0 & 0 & \text{t210} \\
 & & & & 1 & 0 & 0 & \text{t211} \\
 & & & & 1 & 0 & 0 & \text{t212} \\
 & & & & 0 & 0 & 0 & \text{t213} \\
 & & & & 0 & 0 & 0 & \text{t214} \\
\hline
 & & & & & & & \\
\hline
 & & & & 0 & 0 & 0 & \text{tm1} \\
 & & & & 0 & 0 & 0 & \text{tm2} \\
 & & & & 0 & 0 & 0 & \text{tm3} \\
 & & & & 0 & 0 & 0 & \text{tm4} \\
 & & & & 0 & 0 & 0 & \text{tm5} \\
 & & & & 0 & 0 & 0 & \text{tm6} \\
0 & 0 & & A_{mO} & 0 & 0 & 0 & \text{tm7} \\
 & & & & 0 & 0 & 0 & \text{tm8} \\
 & & & & 0 & 0 & 0 & \text{tm9} \\
 & & & & 0 & 0 & 0 & \text{tm10} \\
 & & & & 1 & 0 & 0 & \text{tm11} \\
 & & & & 1 & 0 & 0 & \text{tm12} \\
 & & & & 0 & 0 & 0 & \text{tm13} \\
 & & & & 0 & 0 & 0 & \text{tm14} \\
\hline
0 & 0 & & 0 & 0 & 1 & 0 & \text{t(m14+1)} \\
 & & & & 0 & 0 & 1 & \text{t(m14+2)}
\end{array}$$

Matricea de incidenta:

$$
A=A_O-A_I=\left(\begin{array}{c:c:c:ccc}
p11\ p12\ p13\ p14\ p15\ p16\ p17\ p18\ p19\ p110 & p21\ p22\ p23\ p24\ p25\ p26\ p27\ p28\ p29\ p210 & pm1\ pm2\ pm3\ pm4\ pm5\ pm6\ pm7\ pm8\ pm9\ pm10 & p(m10+1) & p(m10+2) & p(m10+3) \\
 & & & 0 & 0 & -1 \\
 & & & 0 & 0 & 0 \\
 & & & 0 & 0 & 0 \\
 & & & 0 & 0 & 0 \\
 & & & 0 & 0 & 0 \\
 & & & 0 & 0 & 0 \\
A_1 & 0 & 0 & 0 & 0 & 0 \\
 & & & 0 & 0 & 0 \\
 & & & 0 & 0 & 0 \\
 & & & 0 & 0 & 0 \\
 & & & 1 & 0 & 0 \\
 & & & 1 & 0 & 0 \\
 & & & 0 & 0 & 0 \\
 & & & 0 & 0 & 0 \\
\hdashline
 & & & 0 & 0 & -1 \\
 & & & 0 & 0 & 0 \\
 & & & 0 & 0 & 0 \\
 & & & 0 & 0 & 0 \\
 & & & 0 & 0 & 0 \\
 & & & 0 & 0 & 0 \\
0 & A_2 & 0 & 0 & 0 & 0 \\
 & & & 0 & 0 & 0 \\
 & & & 0 & 0 & 0 \\
 & & & 0 & 0 & 0 \\
 & & & 1 & 0 & 0 \\
 & & & 1 & 0 & 0 \\
 & & & 0 & 0 & 0 \\
 & & & 0 & 0 & 0 \\
\hdashline
 & & & & & \\
\hdashline
 & & & 0 & 0 & -1 \\
 & & & 0 & 0 & 0 \\
 & & & 0 & 0 & 0 \\
 & & & 0 & 0 & 0 \\
 & & & 0 & 0 & 0 \\
 & & & 0 & 0 & 0 \\
0 & 0 & A_m & 0 & 0 & 0 \\
 & & & 0 & 0 & 0 \\
 & & & 0 & 0 & 0 \\
 & & & 0 & 0 & 0 \\
 & & & 1 & 0 & 0 \\
 & & & 1 & 0 & 0 \\
 & & & 0 & 0 & 0 \\
 & & & 0 & 0 & 0 \\
\hdashline
0 & 0 & 0 & -1 & 1 & 0 \\
 & & & 0 & -1 & 1
\end{array}\right)
\begin{array}{l}
 \\
t11\\ t12\\ t13\\ t14\\ t15\\ t16\\ t17\\ t18\\ t19\\ t110\\ t111\\ t112\\ t113\\ t114\\
t21\\ t22\\ t23\\ t24\\ t25\\ t26\\ t27\\ t28\\ t29\\ t210\\ t211\\ t212\\ t213\\ t214\\
 \\
tm1\\ tm2\\ tm3\\ tm4\\ tm5\\ tm6\\ tm7\\ tm8\\ tm9\\ tm10\\ tm11\\ tm12\\ tm13\\ tm14\\
t(m14+1)\\ t(m14+2)
\end{array}
$$

Datorita complexitatii acestui model, arborele de acoperire corespunzator este prezentat in tabelul 2.15 sub forma text.

Tabelul 2.15 *Arborele de acoperire corespunzator nodului cu m intrari si o iesire*

M0	t113	M1	**M0**	**t213**	**M2**	M0	tm13	M3
M0	**t(m*14+1)**	**M4**	M1	t114	M0	**M1**	**t213**	**M5**
M1	tm13	M6	**M1**	**t(m*14+1)**	**M7**	M2	t113	M5
M2	**t214**	**M0**	M2	tm13	M8	**M2**	**t(m*14+1)**	**M9**
M3	t113	M6	**M3**	**t213**	**M8**	M3	tm14	M0
M3	**t(m*14+1)**	**M10**	M4	t113	M7	**M4**	**t213**	**M9**
M4	tm13	M10	**M4**	**t(m14+2)**	**M11**	M5	t114	M2
M5	**t214**	**M1**	M5	tm13	M12	**M5**	**t(m*14+1)**	**M13**
M6	t114	M3	**M6**	**t213**	**M12**	M6	tm14	M1
M6	**t(m*14+1)**	**M14**	M7	t114	M4	**M7**	**t213**	**M13**
M7	tm13	M14	**M7**	**t(m14+2)**	**M15**	M8	t113	M12
M8	**t214**	**M3**	M8	tm14	M2	**M8**	**t(m*14+1**	**M16**

							)	
M9	t113	M13	**M9**	**t214**	**M4**	M9	tm13	M16
M9	**t(m14+2)**	**M17**	M10	t113	M14	**M10**	**t213**	**M16**
M10	tm14	M4	**M10**	**t(m14+2)**	**M18**	M11	t11	M19
M11	**t113**	**M15**	M11	t21	M20	**M11**	**t213**	**M17**
M11	tm1	M21	**M11**	**tm13**	**M18**	M12	t114	M8
M12	**t214**	**M6**	M12	tm14	M5	**M12**	**t(m*14+1)**	**M22**
M13	t114	M9	**M13**	**t214**	**M7**	M13	tm13	M22
M13	**t(m14+2)**	**M23**	M14	t114	M10	**M14**	**t213**	**M22**
M14	tm14	M7	**M14**	**t(m14+2)**	**M24**	M15	t114	M11
M15	**t21**	**M25**	M15	t213	M23	**M15**	**tm1**	**M26**
M15	tm13	M24	**M16**	**t113**	**M22**	M16	t214	M10
M16	**tm14**	**M9**	M16	t(m14+2)	M27	**M17**	**t11**	**M28**
M17	t113	M23	**M17**	**t214**	**M11**	M17	tm1	M29
M17	**tm13**	**M27**	M18	t11	M30	**M18**	**t113**	**M24**
M18	t21	M31	**M18**	**t213**	**M27**	M18	tm14	M11
M19	**t12**	**M32**	M19	t13	M33	**M19**	**t213**	**M28**
M19	tm13	M30	**M20**	**t113**	**M25**	M20	t22	M34
M20	**t23**	**M35**	M20	tm13	M31	**M21**	**t113**	**M26**
M21	t213	M29	**M21**	**tm2**	**M36**	M21	tm3	M37
M22	**t114**	**M16**	M22	t214	M14	**M22**	**tm14**	**M13**
M22	t(m14+2)	M38	**M23**	**t114**	**M17**	M23	t214	M15
M23	**tm1**	**M39**	M23	tm13	M38	**M24**	**t114**	**M18**
M24	t21	M40	**M24**	**t213**	**M38**	M24	tm14	M15
M25	**t114**	**M20**	M25	t22	M41	**M25**	**t23**	**M42**
M25	tm13	M40	**M26**	**t114**	**M21**	M26	t213	M39
M26	**tm2**	**M43**	M26	tm3	M44	**M27**	**t11**	**M45**
M27	t113	M38	**M27**	**t214**	**M18**	M27	tm14	M17
M28	**t12**	**M46**	M28	t13	M47	**M28**	**t214**	**M19**
M28	tm13	M45	**M29**	**t113**	**M39**	M29	t214	M21
M29	**tm2**	**M48**	M29	tm3	M49	**M30**	**t12**	**M50**
M30	t13	M51	**M30**	**t213**	**M45**	M30	tm14	M19
M31	**t113**	**M40**	M31	t22	M52	**M31**	**t23**	**M53**
M31	tm14	M20	**M32**	**t14**	**M33**	M32	t213	M46
M32	**tm13**	**M50**	M33	t15	M54	**M33**	**t213**	**M47**

M33	tm13	M51	**M34**	**t113**	**M41**	M34	t24	M35
M34	**tm13**	**M52**	M35	t113	M42	**M35**	**t25**	**M55**
M35	tm13	M53	**M36**	**t113**	**M43**	M36	t213	M48
M36	**tm4**	**M37**	M37	t113	M44	**M37**	**t213**	**M49**
M37	tm5	M56	**M38**	**t114**	**M27**	M38	t214	M24
M38	**tm14**	**M23**	M39	t114	M29	**M39**	**t214**	**M26**
M39	tm2	M57	**M39**	**tm3**	**M58**	M40	t114	M31
M40	**t22**	**M59**	M40	t23	M60	**M40**	**tm14**	**M25**
M41	t114	M34	**M41**	**t24**	**M42**	M41	tm13	M59
M42	**t114**	**M35**	M42	t25	M61	**M42**	**tm13**	**M60**
M43	t114	M36	**M43**	**t213**	**M57**	M43	tm4	M44
M44	**t114**	**M37**	M44	t213	M58	**M44**	**tm5**	**M62**
M45	t12	M63	**M45**	**t13**	**M64**	M45	t214	M30
M45	**tm14**	**M28**	M46	t14	M47	**M46**	**t214**	**M32**
M46	tm13	M63	**M47**	**t15**	**M65**	M47	t214	M33
M47	**tm13**	**M64**	M48	t113	M57	**M48**	**t214**	**M36**
M48	tm4	M49	**M49**	**t113**	**M58**	M49	t214	M37
M49	**tm5**	**M66**	M50	t14	M51	**M50**	**t213**	**M63**
M50	tm14	M32	**M51**	**t15**	**M67**	M51	t213	M64
M51	**tm14**	**M33**	M52	t113	M59	**M52**	**t24**	**M53**
M52	tm14	M34	**M53**	**t113**	**M60**	M53	t25	M68
M53	**tm14**	**M35**	M54	t16	M69	**M54**	**t17**	**M70**
M54	t213	M65	**M54**	**tm13**	**M67**	M55	t113	M61
M55	**t26**	**M71**	M55	t27	M72	**M55**	**tm13**	**M68**
M56	t113	M62	**M56**	**t213**	**M66**	M56	tm6	M73
M56	**tm7**	**M74**	M57	t114	M48	**M57**	**t214**	**M43**
M57	tm4	M58	**M58**	**t114**	**M49**	M58	t214	M44
M58	**tm5**	**M75**	M59	t114	M52	**M59**	**t24**	**M60**
M59	tm14	M41	**M60**	**t114**	**M53**	M60	t25	M76
M60	**tm14**	**M42**	M61	t114	M55	**M61**	**t26**	**M77**
M61	t27	M78	**M61**	**tm13**	**M76**	M62	t114	M56
M62	**t213**	**M75**	M62	tm6	M79	**M62**	**tm7**	**M80**
M63	t14	M64	**M63**	**t214**	**M50**	M63	tm14	M46
M64	**t15**	**M81**	M64	t214	M51	**M64**	**tm14**	**M47**
M65	t16	M82	**M65**	**t17**	**M83**	M65	t214	M54

M65	**tm13**	**M81**	M66	t113	M75	**M66**	**t214**	**M56**
M66	tm6	M84	**M66**	**tm7**	**M85**	M67	t16	M86
M67	**t17**	**M87**	M67	t213	M81	**M67**	**tm14**	**M54**
M68	t113	M76	**M68**	**t26**	**M88**	M68	t27	M89
M68	**tm14**	**M55**	M69	t110	M70	**M69**	**t111**	**M0**
M69	t213	M82	**M69**	**tm13**	**M86**	M70	t18	M90
M70	**t213**	**M83**	M70	tm13	M87	**M71**	**t113**	**M77**
M71	t210	M72	**M71**	**t211**	**M0**	M71	tm13	M88
M72	**t113**	**M78**	M72	t28	M91	**M72**	**tm13**	**M89**
M73	t113	M79	**M73**	**t213**	**M84**	M73	tm10	M74
M73	**tm11**	**M0**	M74	t113	M80	**M74**	**t213**	**M85**
M74	tm8	M92	**M75**	**t114**	**M66**	M75	t214	M62
M75	**tm6**	**M93**	M75	tm7	M94	**M76**	**t114**	**M68**
M76	t26	M95	**M76**	**t27**	**M96**	M76	tm14	M61
M77	**t114**	**M71**	M77	t210	M78	**M77**	**t211**	**M1**
M77	tm13	M95	**M78**	**t114**	**M72**	M78	t28	M97
M78	**tm13**	**M96**	M79	t114	M73	**M79**	**t213**	**M93**
M79	tm10	M80	**M79**	**tm11**	**M1**	M80	t114	M74
M80	**t213**	**M94**	M80	tm8	M98	**M81**	**t16**	**M99**
M81	t17	M100	**M81**	**t214**	**M67**	M81	tm14	M65
M82	**t110**	**M83**	M82	t111	M2	**M82**	**t214**	**M69**
M82	tm13	M99	**M83**	**t18**	**M101**	M83	t214	M70
M83	**tm13**	**M100**	M84	t113	M93	**M84**	**t214**	**M73**
M84	tm10	M85	**M84**	**tm11**	**M2**	M85	t113	M94
M85	**t214**	**M74**	M85	tm8	M102	**M86**	**t110**	**M87**
M86	t111	M3	**M86**	**t213**	**M99**	M86	tm14	M69
M87	**t18**	**M103**	M87	t213	M100	**M87**	**tm14**	**M70**
M88	t113	M95	**M88**	**t210**	**M89**	M88	t211	M3
M88	**tm14**	**M71**	M89	t113	M96	**M89**	**t28**	**M104**
M89	tm14	M72	**M90**	**t19**	**M69**	M90	t112	M0
M90	**t213**	**M101**	M90	tm13	M103	**M91**	**t113**	**M97**
M91	t29	M71	**M91**	**t212**	**M0**	M91	tm13	M104
M92	**t113**	**M98**	M92	t213	M102	**M92**	**tm9**	**M73**
M92	tm12	M0	**M93**	**t114**	**M84**	M93	t214	M79
M93	**tm10**	**M94**	M93	tm11	M5	**M94**	**t114**	**M85**

M94	t214	M80	**M94**	**tm8**	**M105**	M95	t114	M88
M95	**t210**	**M96**	M95	t211	**M6**	**M95**	**tm14**	**M77**
M96	t114	M89	**M96**	**t28**	**M106**	M96	tm14	M78
M97	**t114**	**M91**	M97	t29	**M77**	**M97**	**t212**	**M1**
M97	tm13	M106	**M98**	**t114**	**M92**	M98	t213	M105
M98	**tm9**	**M79**	M98	tm12	**M1**	**M99**	**t110**	**M100**
M99	t111	M8	**M99**	**t214**	**M86**	M99	tm14	M82
M100	**t18**	**M107**	M100	t214	**M87**	**M100**	**tm14**	**M83**
M101	t19	M82	**M101**	**t112**	**M2**	M101	t214	M90
M101	**tm13**	**M107**	M102	t113	**M105**	**M102**	**t214**	**M92**
M102	tm9	M84	**M102**	**tm12**	**M2**	M103	t19	M86
M103	**t112**	**M3**	M103	t213	**M107**	**M103**	**tm14**	**M90**
M104	t113	M106	**M104**	**t29**	**M88**	M104	t212	M3
M104	**tm14**	**M91**	M105	t114	**M102**	**M105**	**t214**	**M98**
M105	tm9	M93	**M105**	**tm12**	**M5**	M106	t114	M104
M106	**t29**	**M95**	M106	t212	**M6**	**M106**	**tm14**	**M97**
M107	t19	M99	**M107**	**t112**	**M8**	M107	t214	M103
M107	**tm14**	**M101**						

unde:
M[p11,p12,p13,p14,p15,p16,p17,p18,p19,p110,p21,p22,p23,p24,p25,p26,p27,p28,
p29,p210,pm1,pm2,pm3,pm4,pm5,pm6,pm7,pm8,pm9,pm10,p(m*10+1),p(m*10+
2),
p(m*10+3)]

M0 = [1,0,0,0,0,0,0,0,1,0,1,0,0,0,0,0,0,0,1,0,1,0,0,0,0,0,0,0,1,0,1,0,0]

M1 = [1,0,0,0,0,0,0,0,0,1,1,0,0,0,0,0,0,0,1,0,1,0,0,0,0,0,0,0,1,0,1,0,0]

M2 = [1,0,0,0,0,0,0,0,1,0,1,0,0,0,0,0,0,0,0,1,1,0,0,0,0,0,0,0,1,0,1,0,0]

M3 = [1,0,0,0,0,0,0,0,1,0,1,0,0,0,0,0,0,0,1,0,1,0,0,0,0,0,0,0,0,1,1,0,0]

M4 = [1,0,0,0,0,0,0,0,1,0,1,0,0,0,0,0,0,0,1,0,1,0,0,0,0,0,0,0,1,0,0,1,0]

M5 = [1,0,0,0,0,0,0,0,0,1,1,0,0,0,0,0,0,0,0,1,1,0,0,0,0,0,0,0,1,0,1,0,0]

M6 = [1,0,0,0,0,0,0,0,0,1,1,0,0,0,0,0,0,0,1,0,1,0,0,0,0,0,0,0,0,1,1,0,0]

M7 = [1,0,0,0,0,0,0,0,0,1,1,0,0,0,0,0,0,0,1,0,1,0,0,0,0,0,0,0,1,0,0,1,0]

M8 = [1,0,0,0,0,0,0,0,1,0,1,0,0,0,0,0,0,0,0,1,1,0,0,0,0,0,0,0,0,1,1,0,0]

M9 = [1,0,0,0,0,0,0,0,1,0,1,0,0,0,0,0,0,0,0,1,1,0,0,0,0,0,0,0,1,0,0,1,0]

M10 =
[1,0,0,0,0,0,0,0,1,0,1,0,0,0,0,0,0,0,1,0,1,0,0,0,0,0,0,0,0,1,0,1,0]

M11 =

[1,0,0,0,0,0,0,0,1,0,1,0,0,0,0,0,0,0,1,0,1,0,0,0,0,0,0,0,1,0,0,0,1]

M12 =
[1,0,0,0,0,0,0,0,0,1,1,0,0,0,0,0,0,0,0,1,1,0,0,0,0,0,0,0,0,1,1,0,0]

M13 =
[1,0,0,0,0,0,0,0,0,1,1,0,0,0,0,0,0,0,0,1,1,0,0,0,0,0,0,0,1,0,0,1,0]

M14 =
[1,0,0,0,0,0,0,0,0,1,1,0,0,0,0,0,0,0,1,0,1,0,0,0,0,0,0,0,0,1,0,1,0]

M15 =
[1,0,0,0,0,0,0,0,0,1,1,0,0,0,0,0,0,0,1,0,1,0,0,0,0,0,0,0,1,0,0,0,1]

M16 =
[1,0,0,0,0,0,0,0,1,0,1,0,0,0,0,0,0,0,0,1,1,0,0,0,0,0,0,0,0,1,0,1,0]

M17 =
[1,0,0,0,0,0,0,0,1,0,1,0,0,0,0,0,0,0,0,1,1,0,0,0,0,0,0,0,1,0,0,0,1]

M18 =
[1,0,0,0,0,0,0,0,1,0,1,0,0,0,0,0,0,0,1,0,1,0,0,0,0,0,0,0,0,1,0,0,1]

M19 =
[0,1,0,0,0,0,0,0,0,0,1,0,0,0,0,0,0,0,1,0,1,0,0,0,0,0,0,0,1,0,0,0,0]

M20 =
[1,0,0,0,0,0,0,0,1,0,0,1,0,0,0,0,0,0,0,0,1,0,0,0,0,0,0,0,1,0,0,0,0]

M21 =
[1,0,0,0,0,0,0,0,1,0,1,0,0,0,0,0,0,0,1,0,0,1,0,0,0,0,0,0,0,0,0,0,0]

M22 =
[1,0,0,0,0,0,0,0,0,1,1,0,0,0,0,0,0,0,0,1,1,0,0,0,0,0,0,0,0,1,0,1,0]

M23 =
[1,0,0,0,0,0,0,0,0,1,1,0,0,0,0,0,0,0,0,1,1,0,0,0,0,0,0,0,1,0,0,0,1]

M24 =
[1,0,0,0,0,0,0,0,0,1,1,0,0,0,0,0,0,0,1,0,1,0,0,0,0,0,0,0,0,1,0,0,1]

M25 =
[1,0,0,0,0,0,0,0,0,1,0,1,0,0,0,0,0,0,0,0,1,0,0,0,0,0,0,0,1,0,0,0,0]

M26 =
[1,0,0,0,0,0,0,0,0,1,1,0,0,0,0,0,0,0,1,0,0,1,0,0,0,0,0,0,0,0,0,0,0]

M27 =
[1,0,0,0,0,0,0,0,1,0,1,0,0,0,0,0,0,0,0,1,1,0,0,0,0,0,0,0,0,1,0,0,1]

M28 =
[0,1,0,0,0,0,0,0,0,0,1,0,0,0,0,0,0,0,0,1,1,0,0,0,0,0,0,0,1,0,0,0,0]

M29 =
[1,0,0,0,0,0,0,0,1,0,1,0,0,0,0,0,0,0,0,1,0,1,0,0,0,0,0,0,0,0,0,0,0]

M30 =
[0,1,0,0,0,0,0,0,0,0,1,0,0,0,0,0,0,0,1,0,1,0,0,0,0,0,0,0,0,1,0,0,0]

M31 =
[1,0,0,0,0,0,0,0,1,0,0,1,0,0,0,0,0,0,0,0,1,0,0,0,0,0,0,0,0,1,0,0,0]

M32 =

[0,0,1,0,0,0,0,0,0,1,0,0,0,0,0,0,1,0,1,0,0,0,0,0,0,1,0,0,0,0]

M33 =

[0,0,0,1,0,0,0,0,0,1,0,0,0,0,0,0,1,0,1,0,0,0,0,0,0,1,0,0,0,0]

M34 =

[1,0,0,0,0,0,0,1,0,0,0,1,0,0,0,0,0,0,1,0,0,0,0,0,0,1,0,0,0,0]

M35 =

[1,0,0,0,0,0,0,1,0,0,0,0,1,0,0,0,0,0,1,0,0,0,0,0,0,1,0,0,0,0]

M36 =

[1,0,0,0,0,0,0,1,0,1,0,0,0,0,0,0,1,0,0,1,0,0,0,0,0,0,0,0,0,0]

M37 =

[1,0,0,0,0,0,0,1,0,1,0,0,0,0,0,0,1,0,0,0,1,0,0,0,0,0,0,0,0,0]

M38 =

[1,0,0,0,0,0,0,0,1,1,0,0,0,0,0,0,0,1,1,0,0,0,0,0,0,0,1,0,0,1]

M39 =

[1,0,0,0,0,0,0,0,1,1,0,0,0,0,0,0,0,1,0,1,0,0,0,0,0,0,0,0,0,0]

M40 =

[1,0,0,0,0,0,0,0,1,0,1,0,0,0,0,0,0,0,1,0,0,0,0,0,0,0,1,0,0,0]

M41 =

[1,0,0,0,0,0,0,0,1,0,0,1,0,0,0,0,0,0,1,0,0,0,0,0,0,1,0,0,0,0]

M42 =

[1,0,0,0,0,0,0,0,1,0,0,0,1,0,0,0,0,0,1,0,0,0,0,0,0,1,0,0,0,0]

M43 =

[1,0,0,0,0,0,0,0,1,1,0,0,0,0,0,0,1,0,0,1,0,0,0,0,0,0,0,0,0,0]

M44 =

[1,0,0,0,0,0,0,0,1,1,0,0,0,0,0,0,1,0,0,0,1,0,0,0,0,0,0,0,0,0]

M45 =

[0,1,0,0,0,0,0,0,0,1,0,0,0,0,0,0,0,1,1,0,0,0,0,0,0,0,1,0,0,0]

M46 =

[0,0,1,0,0,0,0,0,0,1,0,0,0,0,0,0,0,1,1,0,0,0,0,0,0,1,0,0,0,0]

M47 =

[0,0,0,1,0,0,0,0,0,1,0,0,0,0,0,0,0,1,1,0,0,0,0,0,0,1,0,0,0,0]

M48 =

[1,0,0,0,0,0,0,1,0,1,0,0,0,0,0,0,0,1,0,0,1,0,0,0,0,0,0,0,0,0]

M49 =

[1,0,0,0,0,0,0,1,0,1,0,0,0,0,0,0,0,1,0,0,0,1,0,0,0,0,0,0,0,0]

M50 =

[0,0,1,0,0,0,0,0,0,1,0,0,0,0,0,0,1,0,1,0,0,0,0,0,0,0,1,0,0,0]

M51 =

[0,0,0,1,0,0,0,0,0,1,0,0,0,0,0,0,1,0,1,0,0,0,0,0,0,0,1,0,0,0]

M52 =

[1,0,0,0,0,0,0,1,0,0,0,1,0,0,0,0,0,0,1,0,0,0,0,0,0,0,1,0,0,0]

M53 =

[1,0,0,0,0,0,0,0,1,0,0,0,0,1,0,0,0,0,0,0,1,0,0,0,0,0,0,0,0,1,0,0,0]

M54 =
[0,0,0,0,1,0,0,0,0,0,1,0,0,0,0,0,0,0,1,0,1,0,0,0,0,0,0,0,1,0,0,0,0]

M55 =
[1,0,0,0,0,0,0,0,1,0,0,0,0,0,1,0,0,0,0,0,1,0,0,0,0,0,0,0,1,0,0,0,0]

M56 =
[1,0,0,0,0,0,0,0,1,0,1,0,0,0,0,0,0,0,1,0,0,0,0,0,1,0,0,0,0,0,0,0,0]

M57 =
[1,0,0,0,0,0,0,0,0,1,1,0,0,0,0,0,0,0,0,1,0,0,1,0,0,0,0,0,0,0,0,0,0]

M58 =
[1,0,0,0,0,0,0,0,0,1,1,0,0,0,0,0,0,0,0,1,0,0,0,1,0,0,0,0,0,0,0,0,0]

M59 =
[1,0,0,0,0,0,0,0,0,1,0,0,1,0,0,0,0,0,0,0,1,0,0,0,0,0,0,0,0,1,0,0,0]

M60 =
[1,0,0,0,0,0,0,0,0,1,0,0,0,1,0,0,0,0,0,0,1,0,0,0,0,0,0,0,0,1,0,0,0]

M61 =
[1,0,0,0,0,0,0,0,0,1,0,0,0,0,1,0,0,0,0,0,1,0,0,0,0,0,0,0,1,0,0,0,0]

M62 =
[1,0,0,0,0,0,0,0,0,1,1,0,0,0,0,0,0,0,1,0,0,0,0,0,1,0,0,0,0,0,0,0,0]

M63 =
[0,0,1,0,0,0,0,0,0,0,1,0,0,0,0,0,0,0,0,1,1,0,0,0,0,0,0,0,0,1,0,0,0]

M64 =
[0,0,0,1,0,0,0,0,0,0,1,0,0,0,0,0,0,0,0,1,1,0,0,0,0,0,0,0,0,1,0,0,0]

M65 =
[0,0,0,0,1,0,0,0,0,0,1,0,0,0,0,0,0,0,0,1,1,0,0,0,0,0,0,0,1,0,0,0,0]

M66 =
[1,0,0,0,0,0,0,0,1,0,1,0,0,0,0,0,0,0,0,1,0,0,0,0,1,0,0,0,0,0,0,0,0]

M67 =
[0,0,0,0,1,0,0,0,0,0,1,0,0,0,0,0,0,0,1,0,1,0,0,0,0,0,0,0,0,1,0,0,0]

M68 =
[1,0,0,0,0,0,0,0,1,0,0,0,0,0,1,0,0,0,0,0,1,0,0,0,0,0,0,0,0,1,0,0,0]

M69 =
[0,0,0,0,0,1,0,0,0,0,1,0,0,0,0,0,0,0,1,0,1,0,0,0,0,0,0,0,1,0,0,0,0]

M70 =
[0,0,0,0,0,0,1,0,0,0,1,0,0,0,0,0,0,0,1,0,1,0,0,0,0,0,0,0,1,0,0,0,0]

M71 =
[1,0,0,0,0,0,0,0,1,0,0,0,0,0,0,1,0,0,0,0,1,0,0,0,0,0,0,0,1,0,0,0,0]

M72 =
[1,0,0,0,0,0,0,0,1,0,0,0,0,0,0,0,1,0,0,0,1,0,0,0,0,0,0,0,1,0,0,0,0]

M73 =
[1,0,0,0,0,0,0,0,1,0,1,0,0,0,0,0,0,0,1,0,0,0,0,0,0,1,0,0,0,0,0,0,0]

M74 =

[1,0,0,0,0,0,0,0,1,0,1,0,0,0,0,0,0,0,1,0,0,0,0,0,0,0,1,0,0,0,0,0,0]

M75 =
[1,0,0,0,0,0,0,0,0,1,1,0,0,0,0,0,0,0,0,1,0,0,0,0,1,0,0,0,0,0,0,0,0]

M76 =
[1,0,0,0,0,0,0,0,0,1,0,0,0,0,1,0,0,0,0,0,1,0,0,0,0,0,0,0,0,1,0,0,0]

M77 =
[1,0,0,0,0,0,0,0,0,1,0,0,0,0,0,1,0,0,0,0,1,0,0,0,0,0,0,0,1,0,0,0,0]

M78 =
[1,0,0,0,0,0,0,0,0,1,0,0,0,0,0,0,1,0,0,0,1,0,0,0,0,0,0,0,1,0,0,0,0]

M79 =
[1,0,0,0,0,0,0,0,0,1,1,0,0,0,0,0,0,0,1,0,0,0,0,0,0,1,0,0,0,0,0,0,0]

M80 =
[1,0,0,0,0,0,0,0,0,1,1,0,0,0,0,0,0,0,1,0,0,0,0,0,0,1,0,0,0,0,0,0,0]

M81 =
[0,0,0,0,1,0,0,0,0,0,1,0,0,0,0,0,0,0,0,1,1,0,0,0,0,0,0,0,0,1,0,0,0]

M82 =
[0,0,0,0,0,1,0,0,0,0,1,0,0,0,0,0,0,0,0,1,1,0,0,0,0,0,0,0,1,0,0,0,0]

M83 =
[0,0,0,0,0,0,1,0,0,0,1,0,0,0,0,0,0,0,0,1,1,0,0,0,0,0,0,0,1,0,0,0,0]

M84 =
[1,0,0,0,0,0,0,0,1,0,1,0,0,0,0,0,0,0,0,1,0,0,0,0,0,1,0,0,0,0,0,0,0]

M85 =
[1,0,0,0,0,0,0,0,1,0,1,0,0,0,0,0,0,0,0,1,0,0,0,0,0,1,0,0,0,0,0,0,0]

M86 =
[0,0,0,0,0,1,0,0,0,0,1,0,0,0,0,0,0,0,1,0,1,0,0,0,0,0,0,0,0,1,0,0,0]

M87 =
[0,0,0,0,0,0,1,0,0,0,1,0,0,0,0,0,0,0,1,0,1,0,0,0,0,0,0,0,0,1,0,0,0]

M88 =
[1,0,0,0,0,0,0,0,1,0,0,0,0,0,0,1,0,0,0,0,1,0,0,0,0,0,0,0,0,1,0,0,0]

M89 =
[1,0,0,0,0,0,0,0,1,0,0,0,0,0,0,0,1,0,0,0,1,0,0,0,0,0,0,0,0,1,0,0,0]

M90 =
[0,0,0,0,0,0,0,1,0,0,1,0,0,0,0,0,0,0,1,0,1,0,0,0,0,0,0,0,1,0,0,0,0]

M91 =
[1,0,0,0,0,0,0,0,1,0,0,0,0,0,0,0,0,1,0,0,1,0,0,0,0,0,0,0,1,0,0,0,0]

M92 =
[1,0,0,0,0,0,0,0,1,0,1,0,0,0,0,0,0,0,1,0,0,0,0,0,0,0,1,0,0,0,0,0,0]

M93 =
[1,0,0,0,0,0,0,0,0,1,1,0,0,0,0,0,0,0,0,1,0,0,0,0,0,1,0,0,0,0,0,0,0]

M94 =
[1,0,0,0,0,0,0,0,0,1,1,0,0,0,0,0,0,0,0,1,0,0,0,0,0,1,0,0,0,0,0,0,0]

M95 =

[1,0,0,0,0,0,0,0,0,1,0,0,0,0,0,0,0,1,0,0,0,0,1,0,0,0,0,0,0,0,0,0,0,1,0,0,0]

M96 =
[1,0,0,0,0,0,0,0,0,1,0,0,0,0,0,0,0,0,1,0,0,0,1,0,0,0,0,0,0,0,0,0,0,1,0,0,0]

M97 =
[1,0,0,0,0,0,0,0,0,1,0,0,0,0,0,0,0,0,0,1,0,0,1,0,0,0,0,0,0,0,0,0,1,0,0,0,0]

M98 =
[1,0,0,0,0,0,0,0,0,1,1,0,0,0,0,0,0,0,0,0,1,0,0,0,0,0,0,0,0,0,0,1,0,0,0,0,0]

M99 =
[0,0,0,0,0,1,0,0,0,0,1,0,0,0,0,0,0,0,0,0,0,1,1,0,0,0,0,0,0,0,0,0,0,1,0,0,0]

M100 =
[0,0,0,0,0,0,1,0,0,0,1,0,0,0,0,0,0,0,0,0,0,1,1,0,0,0,0,0,0,0,0,0,0,1,0,0,0]

M101 =
[0,0,0,0,0,0,0,1,0,0,1,0,0,0,0,0,0,0,0,0,0,1,1,0,0,0,0,0,0,0,0,0,1,0,0,0,0]

M102 =
[1,0,0,0,0,0,0,0,1,0,1,0,0,0,0,0,0,0,0,0,0,1,0,0,0,0,0,0,0,0,0,1,0,0,0,0,0]

M103 =
[0,0,0,0,0,0,0,1,0,0,1,0,0,0,0,0,0,0,0,0,1,0,1,0,0,0,0,0,0,0,0,0,0,1,0,0,0]

M104 =
[1,0,0,0,0,0,0,0,1,0,0,0,0,0,0,0,0,0,0,1,0,0,1,0,0,0,0,0,0,0,0,0,0,1,0,0,0]

M105 =
[1,0,0,0,0,0,0,0,0,1,1,0,0,0,0,0,0,0,0,0,0,1,0,0,0,0,0,0,0,0,0,1,0,0,0,0,0]

M106 =
[1,0,0,0,0,0,0,0,0,1,0,0,0,0,0,0,0,0,0,1,0,0,1,0,0,0,0,0,0,0,0,0,0,1,0,0,0]

M107 =
[0,0,0,0,0,0,0,1,0,0,1,0,0,0,0,0,0,0,0,0,0,1,1,0,0,0,0,0,0,0,0,0,0,1,0,0,0]

Studiind proprietatile comportamentale ale retelei Petri *PN_AS_Nn*, pe baza arborelui de acoperire rezulta:

- reteaua este *marginita* (simbolul ω nu apare in arborele de acoperire);
- reteaua este *sigura* (marcajele din toate nodurile arborelui de acoperire contin numai „0” si „1”);
- reteaua *nu este blocanta* (toate tranzitile au asociate arce in arborele de acoperire);
- reteaua este *accesibila* (orice marcaj poate fi atins pornind din M_0).

Concluzionand, se poate afirma ca modelul ales este *viabil* din punct de vedere comportamental.

Analiza retelei Petri *PN_PT_Nm* din punct de vedere structural se face pe baza analizei existentei invariantilor de tip *P* respectiv *T*.

Pentru determinarea numarului de invarianti *P* respectiv *T* s-a aplicat teorema 2.3.

Rangul matricii A in acest caz este:

$$rang\,A = nr\_nod \cdot rangN_1$$

unde:

- *nr_nod* – reprezinta numarul intrarilor;

- *rangN*$_1$ – reprezinta matricii de incidenta a nodului de tip 1 ($rangN_1=7$).

Numarul de invarianti *P* corespunzatori se poate calcula cu:

$$Nr\_Inv\_P = m - rang\,A = m - nr\_nod \cdot rangN_1$$

iar numarul de invarianti *T* corespunzatori cu:

$$Nr\_Inv\_T = n - rang\,A = n - nr\_nod \cdot rangN_1$$

2.5.2.7.2.2. Cazul: retea Petri temporizata P

Structura retelei Petri temporizata P, corespunzatoare nodului cu *o* intrare si *m* iesiri, este aceeasi ca cea prezentata in figura 2.124.

Datorita faptului ca retelele sunt identice din punct de vedere structural si comportamental, rezultatele analizelor nu se mai detaliaza, ele fiind similare celor obtinute in cazul nodului netemporizat.

Fata de situatia prezentata anterior, fiecarei pozitii P i-a fost alocata o unitate de timp corespunzatoare. In acest mod situatia devine reala din punct de vedere al functionarii.

Timpii alocati pozitiilor sunt prezentati in tabelul 2.16.

Tabelul 2.16. *Timpii corespunzatori pozitiilor P*

Pozitie	Unitati de Timp
Pi1	1
Pi2	5
Pi3	4
Pi4	4
Pi5	7
Pi6	6
Pi7	3
Pi8	5
Pi9	1
Pi10	3
P(m*10+1)	1
P(m*10+2)	3
P(m*10+3)	4

unde i=1,...,m.

Reteaua Petri considerata este formalizata prin sixtuplul:

$$PN\_PTtime\_Nn = (P, T, F, W, D, M_0)$$

unde:

- $$P = \bigcup_{i=1}^{m} \{P_{i1}, P_{i2}, P_{i3}, P_{i4}, P_{i5}, P_{i6}, P_{i7}, P_{i8}, P_{i9}, P_{i10}\} \cup \{P_{(m10+1)}, P_{(m10+2)}, P_{(m8+3)}\}$$

- $$\begin{aligned} T = \bigcup_{i=1}^{m} & \{t_{i1},t_{i2},t_{i3},t_{i4},t_{i5},t_{i6},t_{i7},t_{i8},t_{i9},t_{i10},t_{i11},t_{i12},t_{i13},t_{i14}\} \cup \\ & \{t_{(m14+1)},t_{(m14+2)}\} \end{aligned}$$
- $$\begin{aligned} F = \bigcup_{i=1}^{m} & \{\{(p_{i1},t_{i1}),(p_{i2},t_{i2}),(p_{i2},t_{i3}),(p_{i3},t_{i4}),(p_{i4},t_{i5}),(p_{i5},t_{i6}), \\ & (p_{i5},t_{i7}),(p_{i6},t_{i10}),(p_{i6},t_{i11}),(p_{i7},t_{i8}),(p_{i8},t_{i9}),(p_{i8},t_{i12}), \\ & (p_{n9},t_{i13}),(p_{n10},t_{i14}),(p_{m10+3},t_{i1})\} \cup \\ & \{(t_{i1},p_{i2}),(t_{i2},p_{i3}),(t_{i3},p_{i4}),(t_{i4},p_{i4}),(t_{i5},p_{i5}),(t_{i6},p_{i6}), \\ & (t_{i7},p_{i7}),(t_{i8},p_{i8}),(t_{i9},p_{i6}),(t_{i10},p_{i7}),(t_{i11},p_{i1}),(t_{i11},p_{n9}), \\ & (t_{i11},p_{(m10+1)}),(t_{i12},p_{i1}),(t_{i12},p_{(m10+1)}),(t_{i13},p_{i10}), \\ & (t_{i14},p_{i9})\}\} \cup \{\{(p_{m10+1},t_{m14+1}),(p_{m10+2},t_{m14+2})\} \cup \\ & \{(t_{m14+1},p_{m10+2}),(t_{m14+2},p_{m10+3})\}\} \end{aligned}$$
- $$\begin{aligned} & W(p_{i1},t_{i1}) = 1, W(p_{i2},t_{i2}) = 1, W(p_{i2},t_{i3}) = 1, W(p_{i3},t_{i4}) = 1, \\ & W(p_{i4},t_{i5}) = 1, W(p_{i5},t_{i6}) = 1, W(p_{i5},t_{i7}) = 1, \\ & W(p_{i6},t_{i10}) = 1, W(p_{i6},t_{i11}) = 1, W(p_{i7},t_{i8}) = 1, W(p_{i8},t_{i9}) = 1, \\ & W(p_{i8},t_{i12}) = 1, W(p_{n9},t_{i13}) = 1, W(p_{n10},t_{i14}) = 1, \\ & W(p_{m10+3},t_{i1}) = 1, W(t_{i1},p_{i2}) = 1, W(t_{i2},p_{i3}) = 1, W(t_{i3},p_{i4}) = 1, \\ & W(t_{i4},p_{i4}) = 1, W(t_{i5},p_{i5}) = 1, W(t_{i6},p_{i6}) = 1, \\ & W(t_{i7},p_{i7}) = 1, W(t_{i8},p_{i8}) = 1, W(t_{i9},p_{i6}) = 1, W(t_{i10},p_{i7}) = 1, \\ & W(t_{i11},p_{i1}) = 1, W(t_{i11},p_{n9}) = 1, W(t_{i11},p_{(m10+1)}) = 1, \\ & W(t_{i12},p_{i1}) = 1, W(t_{i12},p_{(m10+1)}) = 1, W(t_{i13},p_{i10}) = 1, \\ & W(t_{i14},p_{i9}) = 1, W(p_{m10+1},t_{m14+1}) = 1, W(p_{m10+2},t_{m14+2}) = 1, \\ & W(t_{m14+1},p_{m10+2}) = 1, W(t_{m14+2},p_{m10+3}) = 1 \end{aligned}$$
- $$\begin{aligned} D = \{ & d(p_{i1}) = 1, d(p_{i2}) = 5, d(p_{i3}) = 4, d(p_{i4}) = 4, d(p_{i5}) = 7, \\ & d(p_{i6}) = 6, d(p_{i7}) = 3, d(p_{i8}) = 5, d(p_{i9}) = 1, d(p_{i10}) = 3, \\ & d(p_{m*10+1}) = 1, d(p_{m*10+2}) = 3, d(p_{m*10+3}) = 4\} \end{aligned}$$
- $$M_0 = [1,\ 0,0,0,0,0,0,0,1,0,0,0,0,0,0,0,\ldots\ldots,1,0,0,0,0,0,0,0,1,0,1]^T$$

2.5.2.8. Modelarea nodurilor cu „n” intrari si „m” iesiri [Ung07a]

Nodurile complexe cu *n* intrari si *m* iesiri sunt construite cu ajutorul nodurilor definite anterior. Un astfel de nod are structura de principiu prezentata in figura 2.126.

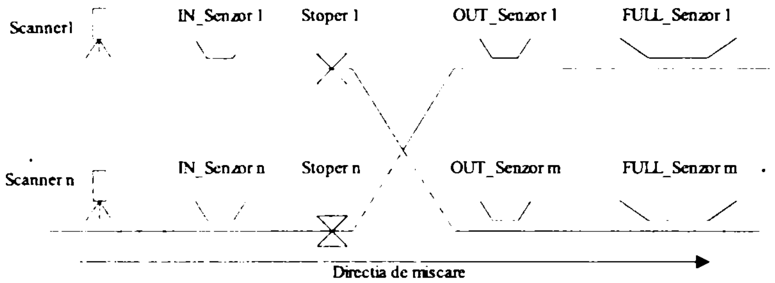


Figura 2.126 Structura de principiu a unui nod cu „doua" intrari si „doua" iesiri

2.5.2.8.1. Modelarea nodului cu *n* intrari si *m* iesiri cu ajutorul automatelor

Structura automatului secvential considerat pentru modelarea unui astfel de nod este prezentata în figura 2.127.

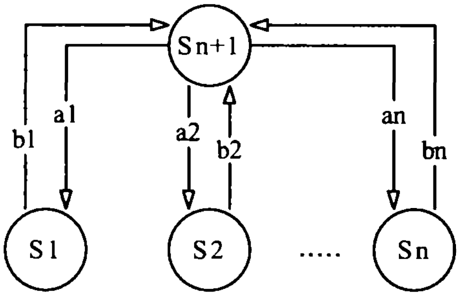


Figura 2.127. Structura automatului secvential corespunzător nodului cu *n* intrari si *m* iesiri

In figura de mai sus S1, S2 si Sn reprezinta automatele care modeleaza un nod cu *o* intrare si *m* iesiri (sunt modele de tip *AS_Nm*). Sincronizarea automatelor este realizata prin intermediul starii suplimentare Sn+1.

Automatul *AS_Nn-m* corespunzator nodului cu *n* intrari si *m* iesiri este descris in modul urmator:

- multimea starilor:

$$X = \bigcup_{i=1}^{n} \{S_i\} \cup \{S_{n+1}\}\}$$

- multimea evenimentelor:

$$\Sigma = \bigcup_{i=1}^{n} \{a_i, b_i\}$$

- multimile evenimentelor posibile si functiile de tranzitie a starilor

$$\Gamma(S) = \bigcup_{i=1}^{m} \Gamma(Si) \text{ unde:}$$

$$\Gamma(Si) = \{bi\} \qquad \delta(Si, bi) = S(n+1)$$

$$\Gamma(S(n+1)) = \{a1, a2, \dots, an\} \qquad \delta(S(n+1), ai) = Si$$

- starea initiala: $x_0 = S(n+1)$.

Submodelele utilizate pentru modelarea starilor S_i, $i=1,...,n$, sunt definite in mod identic cu cele prezentate in cazul modelului *AS_Nm*.

Aplicand algoritmul 2.1 de transformare a automatelor in retele Petri rezulta:

- multimea pozitiilor: $P = \bigcup_{i=1}^{m} \{pi\} \cup \{p(n+1)\}$, unde

  $pi = \{Si\}$ si $p(n+1) = \{S(n+1)\}$;
- multimea trranzitiilor:

• Pentru *pi*	$(pi,p(n+1))$	$p(n+1)=\delta(pi,ti1)$	$ti1=\{bi\}$
• Pentru *p(n+1)*	$(p(n+1),pi)$	$pi=\delta(p(n+1),ti2)$	$ti2=\{ai\}$

Modelarea automatului secvential cu ajutorul retelelor Petri se realizeaza prin utilizarea submodelelor.

In figura 2.128 se prezinta structura submodelului retelei Petri asociata unei intrari (pozitie p_i)

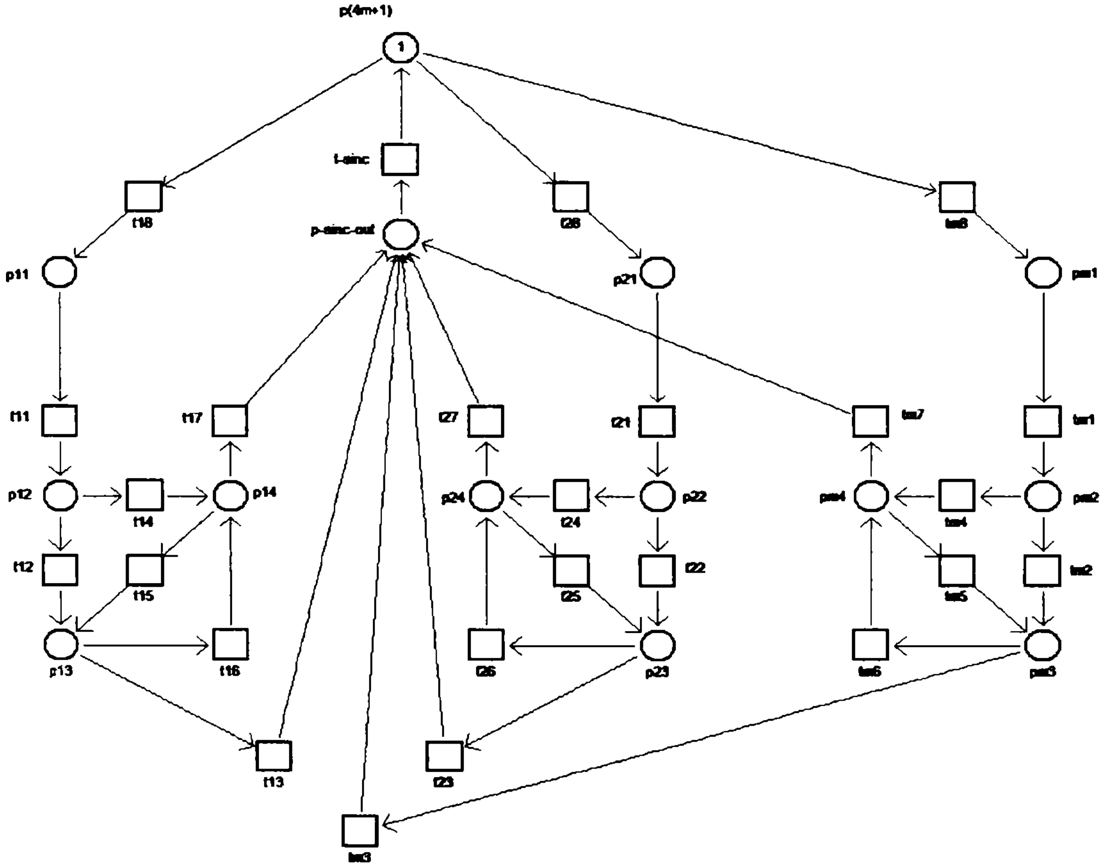


Figura 2.128. Structura reţelei Petri asociata submodelului corespunzator unei intrari (pozitii p_i)

Dupa cum se poate observa, submodelul ales este similar cu modelul validat in cazul modelarii unui nod cu *o* intrare si *m* iesiri (*PN_AS_Nm*), singura diferenta fiind data de introducerea starii suplimentare *p-sinc-out* si a tranzitiei de sincronizare *t-sinc*, care au rolul de a asigura conexiunea spre exterior numai prin intrmediul unei singure tranzitii. Introducerea acestei pozitii nu modifica proprietatile structurale si comportamentale ale retelei, fiind modificata numai structura matricilor de incidenta. In acest fel toate rezultatele validate in cazul PN_ASm sunt valabile si pentru submodelul considerat.

Pe baza celor prezentate, se poate *concluziona* că structura stabilita pentru submodelul ales corespunzator nodului cu *n* intrari si *m* iesiri este *viabilă*, ea putând fi utilizata in modelarea intregului nod, fiind siguri că nu există condiţii de apariţie a blocajelor sau a apariţiei de situaţii necontrolabile.

Reteaua Petri rezultata prin utilizarea submodelului definit anterior este prezentata in figura 2.129.

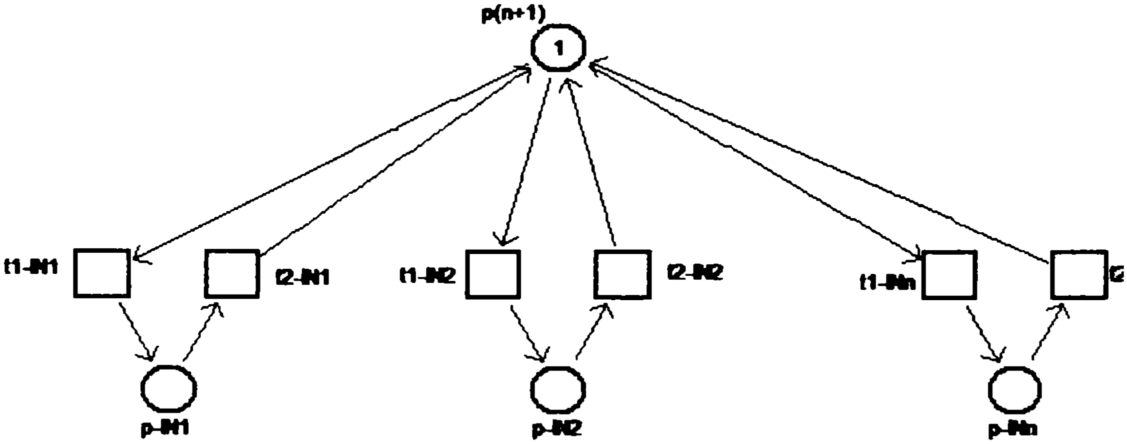


Figura 2.129. Reteaua Petri corespunzatoare modelarii nodului cu *n* intrari si *m* iesiri derivata din automatul secvential *AS_Nm*

Topologia retelei validata de PNT este prezentata in figura 2.130.

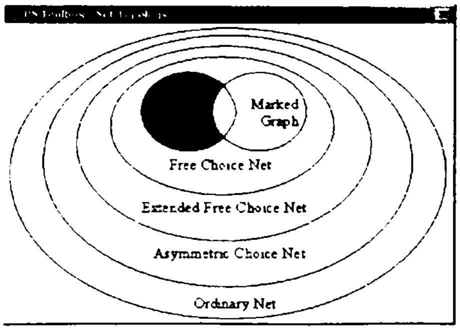


Figura 2.130. Topologia retelei Petri echivalenta validata de PNT

Reteaua Petri considerata pentru analiza submodelului este formalizata prin cvintuplul:

$$PN\_AS\_Nn-m=(P,T,F,W,M_0)$$

unde:

- $P=\bigcup_{i=1}^{n}\{p-IN_i\}\cup\{p_{n+1}\}$
- $T=\bigcup_{i=1}^{n}\{t1-IN_i,t2-IN_i\}$
- $F=\left\{\bigcup_{i=1}^{n}\{\{(p-IN_i,t2-IN_i)\}\cup\{(t1-IN_i,p-IN_i\}\}\right\}\cup$ $\left\{\bigcup_{i=1}^{n}\{\{(p_{n+1},t1-IN_i)\}\cup\{(t2-IN_i,p_{n+1})\}\}\right\}$
- $W(p-IN_i,t2-IN_i)=1, W(t1-IN_i,p-IN_i)=1,$ $W(p_{n+1},t1-IN_i)=1, W(t2-IN_i,p_{n+1})=1$
- $M_0=[0,0,.....,1]^T$

Matricile de incidenţă sunt.

Matricea de incidenta de intrare:

$$A_I=\begin{pmatrix}0 & 0.....0 & 1\\ 1 & 0.....0 & 0\\ 0 & 0.....0 & 1\\ 0 & 1.....0 & 0\\ \multicolumn{3}{c}{...............}\\ 0 & 0.....0 & 1\\ 0 & 0.....1 & 0\end{pmatrix}$$

Matricea de incidenta de iesire:

$$A_O=\begin{pmatrix}1 & 0.....0 & 0\\ 0 & 0.....0 & 1\\ 0 & 1.....0 & 0\\ 0 & 0.....0 & 1\\ \multicolumn{3}{c}{...............}\\ 0 & 0.....1 & 0\\ 0 & 0.....0 & 1\end{pmatrix}$$

Matricea de incidenta:

$$A=A_O-A_I=\begin{pmatrix}1 & 0..... & 0 & -1\\ -1 & 0..... & 0 & 1\\ 0 & 1..... & 0 & -1\\ 0 & -1..... & 0 & 1\\ \multicolumn{4}{c}{..................}\\ 0 & 0..... & 1 & -1\\ 0 & 0..... & -1 & 1\end{pmatrix}$$

Arborele de acoperire corespunzator este prezentat in figura 2.131.

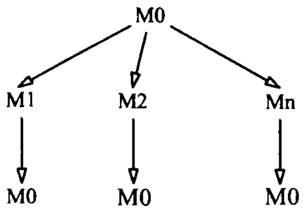


Figura 2.131. Arborele de acoperire al retelei Petri PN_{SUB}_*AS_Nn-m*

Unde:

M0=(0,0,....,0,1); M1=(1,0,....,0,0); M2=(0,1,....,0,0); Mn=(0,0,....,1,0);

Studiind proprietatile comportamentale ale retelei Petri *PN_AS_Nn-m*, pe baza arborelui de acoperire rezulta:

- reteaua este *marginita* (simbolul ω nu apare in arborele de acoperire);
- reteaua este *sigura* (marcajele din toate nodurile arborelui de acoperire contin numai „0" si „1");
- reteaua *nu este blocanta* (toate tranzitiile au asociate arce in arborele de acoperire);
- reteaua este *accesibila* (orice marcaj poate fi atins pornind din M_0).

Graful de acoperire corespunzator este prezentat in figura 2.132.

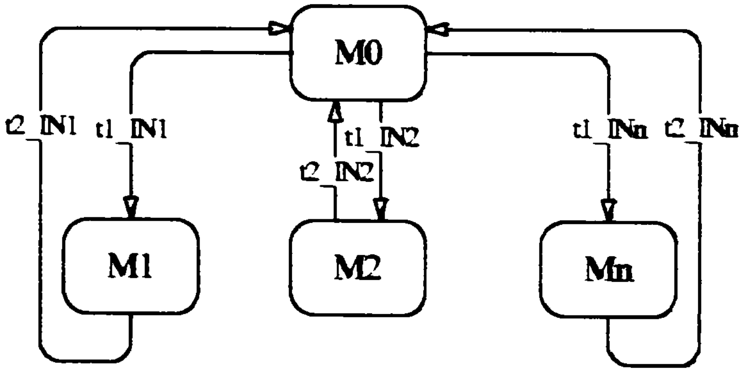


Figura 2.132. Graful de acoperire al retelei Petri *PN_AS_Nn-m*

Structura rezultata pentru graful de acoperire confirma ca reteaua Petri obtinuta cu ajutorul submodelelor corespunzatoare automatului secvential al nodului cu *n* intrari si *m* iesiri este *acesibila*.

Concluzionand, se poate afirma ca modelul ales este *viabil* din punct de vedere comportamental.

Analiza retelei Petri *PN_AS_Nn-m* din punct de vedere structural se face pe baza analizei existentei invariantilor de tip *P* respectiv *T* utilizand teorema 2.3.

Rangul matricei A este:

$$rang\,A = n$$

unde:

- *n* – reprezinta numarul de intrari ale nodului;

Numarul de invarianti *P* corespunzatori este dat de relatia:

$$Nr\_Inv\_P = nr\_pozitii \quad rang\,A - n + 1 - n = 1$$

iar numarul de invarianti *T* corespunzatori este:

$$Nr\_Inv\_T = nr\_tranzitii \quad rang\,A - n * 2 \quad n = n$$

2.5.2.8.2. Modelarea nodului cu *n* intrari si *m* iesiri cu ajutorul retelelor Petri

2.5.2.8.2.1. Cazul: retea Petri netemporizata

Structura retelei Petri considerate pentru modelarea unui atfel de nod este prezentata in figura 2.133.

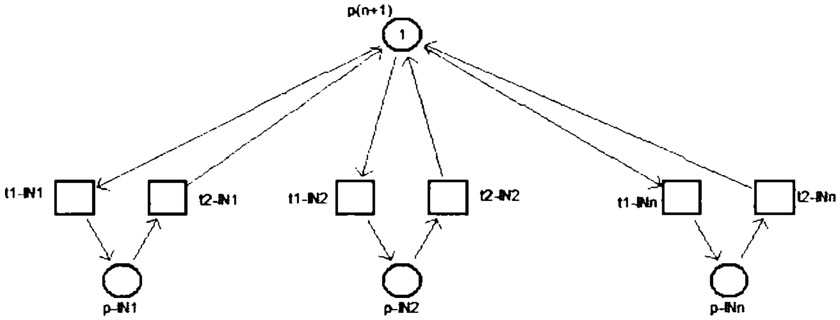


Figura 2.133. Structura retelei Petri corespunzatoare nodului cu doua intrari si doua iesiri

Structura este identica cu cea analizata si validata in cazul retelei *PN_A-Sn-m,* toate rezultatele obtinute in urma analizei acestei retele fiind validate implicit.

Diferenta fata de situatia anterioara este data de submodelul utilizat, de structura retelei Petri determinata in cazul nodului cu *o* intrare si *m* iesiri, introducerea pozitiilor si tranzitiilor de sincronizare nemodificand proprietatile retelei.

Structura submodelului este prezentata in figura 2.134.

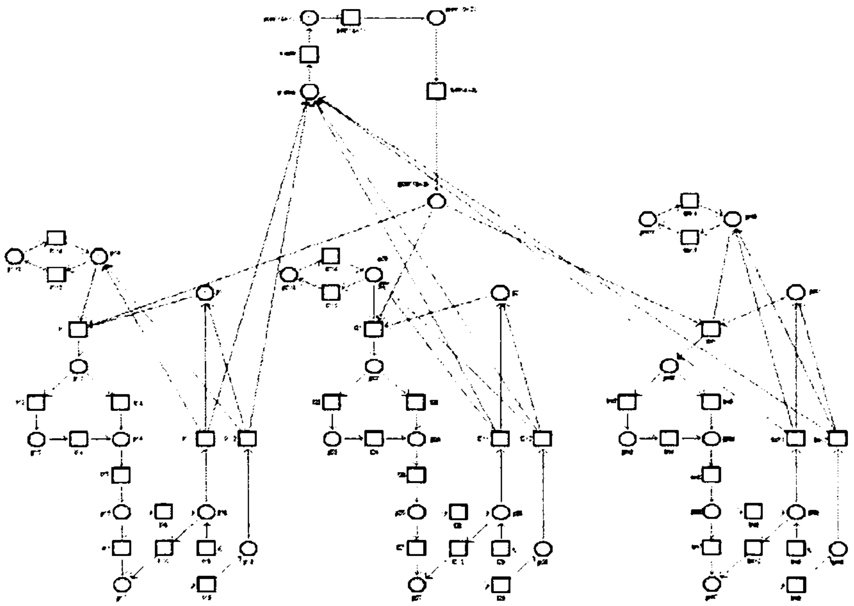


Figura 2.134. Submodelul de tip retea Petri pentru *o* intrare si *m* iesiri

Datorita faptului ca, in structura retelei *PN_PT_Nm* s-a intervenit numai prin adaugarea pozitiei *p-sinc* (cu rol de pozitie sincronizatoare), si tranzitiei *t-sinc* (tranzitie sincronizatoare), necesare conectarii ca submodel, rezulta ca toate rezultatele obtinute in cazul retelei din care a rezultat submodelul sunt valide si in acest caz.

2.5.2.8.2.2. Cazul: retea Petri temporizata P

Structura retelei Petri temporizata P corespunzatoare nodului cu *o* intrare si *m* iesiri este aceeasi ca cea prezentata in figura 2.133.

Datorita faptului ca retelele sunt identice din punct de vedere structural si comportamental, rezultatele analizelor nu se mai detaliaza, ele fiind in fapt identice cu cele obtinute in cazul nodului netemporizat.

Fata de situatia prezentata anterior, fiecarei pozitii P i-a fost alocata o unitate de timp corespunzatoare.

Intervalele de timp corespunzatoare pozitiilor P ale modelului sunt prezentate in tabelul 2.17

Tabelul 2.17.*Intervalele de timpi corespunzatori pozitiilor P*

Pozitie	Unitati de Timp	
	Min	Max
Pi	4	37
P(n+1)	0	0

unde i=1,...,n.

2.6. Modelarea sistemelor de transport cu zone de acumulare

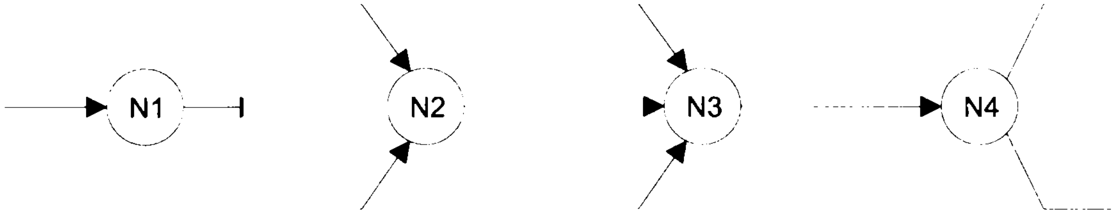
Pana in aceasta etapa, modelarea s-a concentrat asupra elementelor de baza ale unui sistem de transport cu zone de acumulare. Avand stabilite modelele de baza pentru elementele componente (nodurile definite anterior), cunoscand modul de functionare a acestor componente, se poate aborda modelarea in ansamblu a STZA.

Datorita complexitatii si particularitatilor fiecarui STZA in parte, este destul de dificil, daca nu imposibil, de stabilit un model cu putere de generalizare, care sa descrie comportarea oricarui STZA. Abordarea modelarii unui astfel de sistem va demara cu analiza modului de integrare a nodurilor prezentate anterior, intr-un sistem complex. In acest context, structura interna a unui nod nu mai este edificatoare, important fiind modul de conexiune al nodurilor in cadrul sistemului. In cele ce urmeaza se prezinta modul de reprezentare grafica (simbolurile) propus si utilizat in cadrul lucrarii, pentru nodurile de baza din cadrul unui STZA complex.

In aceasta etapa nu intereseaza in fapt algoritmul de functionare al sistemului ci numai analiza din punctul de vedere al eliminarii blocajelor din sistem. Adaugarea algoritmului de functionare constituie o alta problema distincta, dezvoltata in capitolul 3, care se ocupa in detaliu de conducerea supervizata a STZA.

2.6.1. Reprezentarea nodurilor de baza si a nodurilor complexe

Reprezentarea simbolica a nodurilor este prezentata in figura 2.135 [UPS06a].



a) b) c) d)

Figura 2.135. Reprezentarea simbolica a nodurilor.a) nod de tip 1; b) nod de tip 2; c)nod de tip 3; d)nod de tip 4

Avand ca referinte rezultatele obtinute in urma modelarii nodurilor de baza definite anterior, se pot analiza fara probleme configuratii diverse, de complexitati diferite.

Datorita complexitatii matricilor de incidenta, precum si a dimensiunilor mari ale arborelui de acoperire, pentru nodurile cu mai mult de 3 intrari, respectiv 2 iesiri, se utilizeaza scheme echivalente, si nu modelele de tip general stabilite anterior, datorita faptului ca analiza este mult mai facila in acest caz. Modelele de tip general stabilite (n intrari - o iesire, o intrare - m iesiri, n intrari - m iesiri) sunt utilizabile in validarea primara a unui anumit tip de model urmand ca apoi pentru implementare sa fie utilizate modele simple.

- ***Modelarea unui nod cu 5 intrari si o iesire[UP06a]***

Structura de baza este prezentata in figura 2.136.

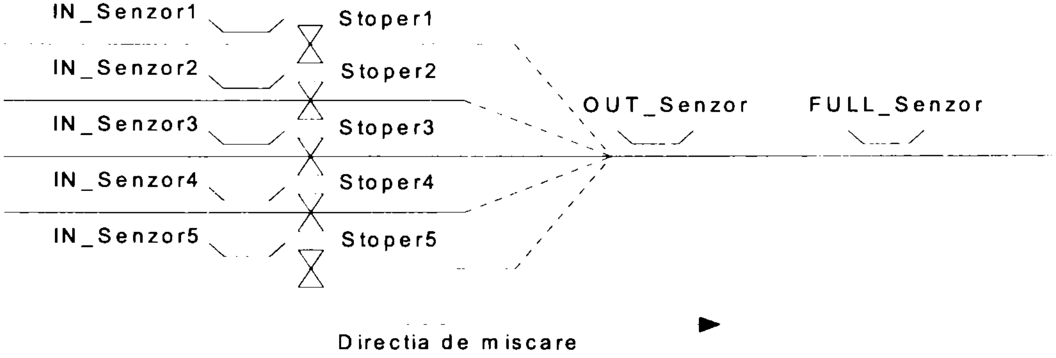


Figura 2.136. Structura de baza a unui nod cu 5 intrari si o iesire

Modelarea acestui tip de nod se poate realiza pe doua cai:

a) modelarea directa, prin sincronizarea a 5 noduri de tip 1 (in mod similar cu cele prezentate la nodul de tip 3), ceea ce ar avea insa ca efect obtinerea unei retele Petri de mari dimensiuni, cu implicatii directe asupra timpului de simulare si analiza.

b) modelarea prin descompunerea in noduri de baza, fara a mai detalia structura interna a fiecarui nod, ceea ce conduce la micsorarea timpului de simulare si analiza.

Metoda b este evident cea mai avantajoasa, si se va fi cea utilizata in anliza acetui tip de nod.. Structura echivalenta este prezentata in figura 2.137.

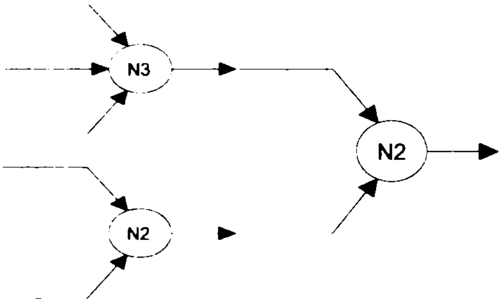


Figura 2.137. Structura echivalenta a nodului cu 5 intrari si o iesire

In acest mod s-a realizat conversia unui nod complex in noduri de baza, urmand ca in cadrul ansamblului sa se asigure numai sincronizarea corecta a nodurilor utilizate.

- ***Modelarea unui nod cu o intrare si 3 iesiri[UP06a]***

Structura de baza a acestui tip de nod este prezentata in figura 2.138.

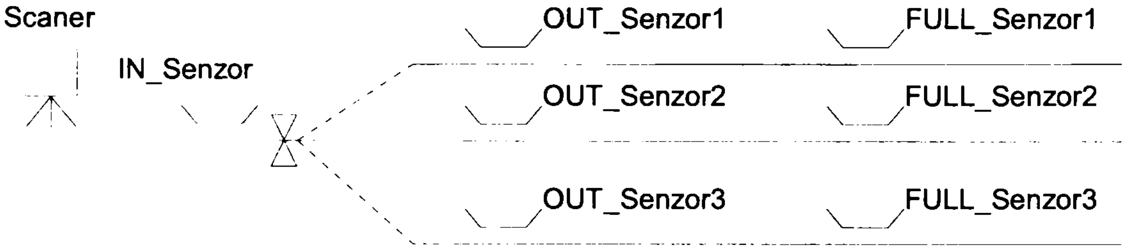


Figura 2.138. Structura de baza a unui nod cu o intrare si 3 iesiri

Structura echivalenta este prezentata in figura 1.139.

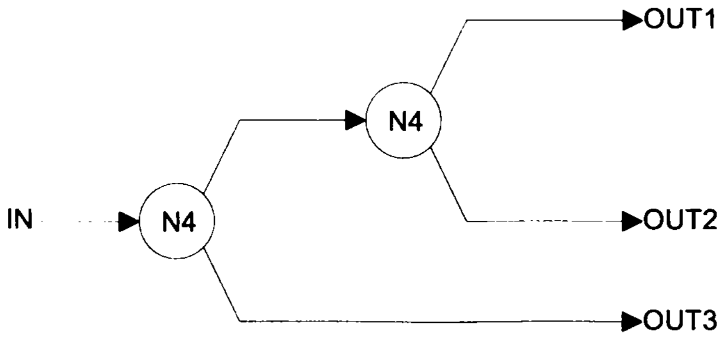


Figura 2.139. Structura echivalenta a nodului cu o intrare si 3 iesiri

- ***Modelarea unui nod cu 5 intrari si 5 iesiri[UP06a]***

Structura de baza a acestui tip de nod este prezentata in figura 2.140.

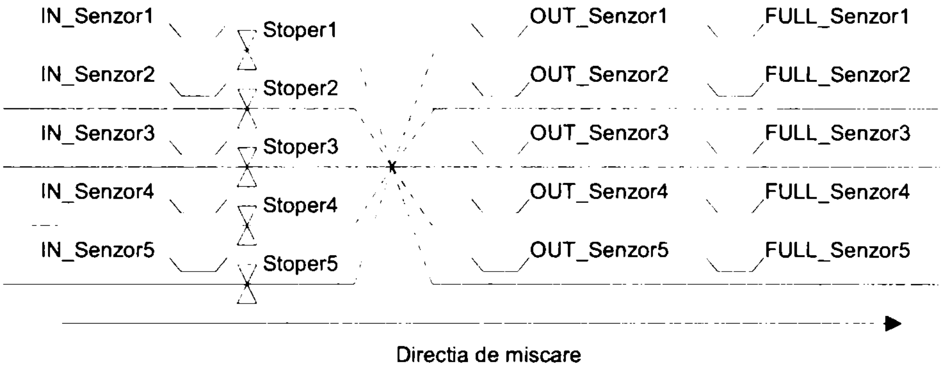


Figura 2.140. Structura de baza a unui nod cu 5 intrari si 5 iesiri

Structura echivalenta realizata cu noduri standard este prezentata in figura 2.141.

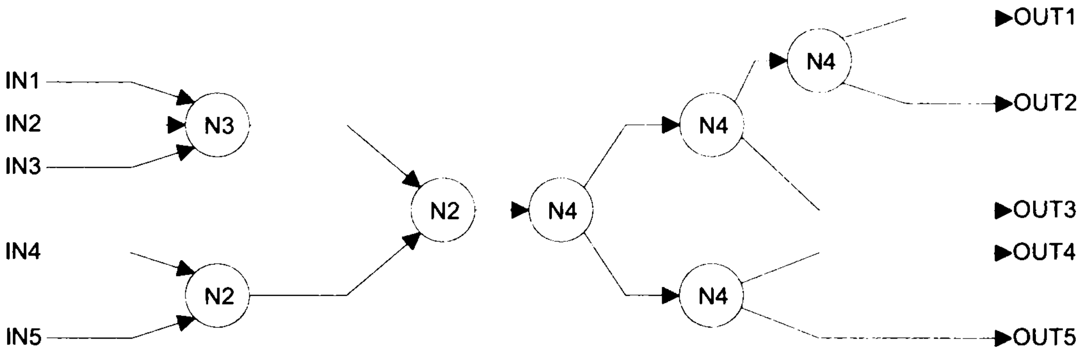


Figura 2.141. Structura echivalenta a unui nod cu 5 intrari si 5 iesiri

2.6.2. Definirea sistemului de transport cu zone de acumulare (STZA)

Definitia 2.21 Sistem de transport cu zone de acumulare*
Un sistem de transport cu zone de acumulare cu n noduri si m jam-uri este definit formal prin cvatruplul de forma:

$$STZA = \{ND, J, Conect, Cap_{max}Jam, I\}$$

unde:

- $ND = \{N_1, N_2, ..., N_n\}$ - *multimea nodurilor;*
- $J = \{J_1, J_2, ..., J_m\}$ - *multimea jam-urilor;*
- $Conect = \{(Jam_1, N_1), (Jam_2, N_2).....\} \cup \{(N_1, Jam_2), (N_2, Jam_3)....\}$ - *multimea prin care se descriu conexiunile jam-uri - noduri si noduri - jam-uri;*

- $Cap_{max}Jam = \{Cap_{max}Jam_1, Cap_{max}Jam_2,.....,Cap_{max}Jam_m\}$ *- multimea capacitatilor maxime ale jam-urilor:*
- $IJam = \{NICarJam_1, NICarJam_2,.....,NICarJam_m\}$ *- mutlimea corespunzatoare starii initiale a jam-urilor (reprezinta gradul de umplere a fiecarui jam in parte - numarul de carucioare din jam-ul respectiv).*

Functionarea STZA este influentata de un algoritm flexibil de functionare atasat corespunzator.

Un astfel de sistem are capabilitatea de a opera astfel incat logica de functionare sa se modifice in functie de cerintele sistemului care este deservit de STZA.

Exemplul 2.6. STZA reprezentat prin intermediul nodurilor definite
Pentru exemplificarea modului de utlizare a nodurilor intr-o reprezentare a unui sistem de transport se considera structura din figura 2.142.

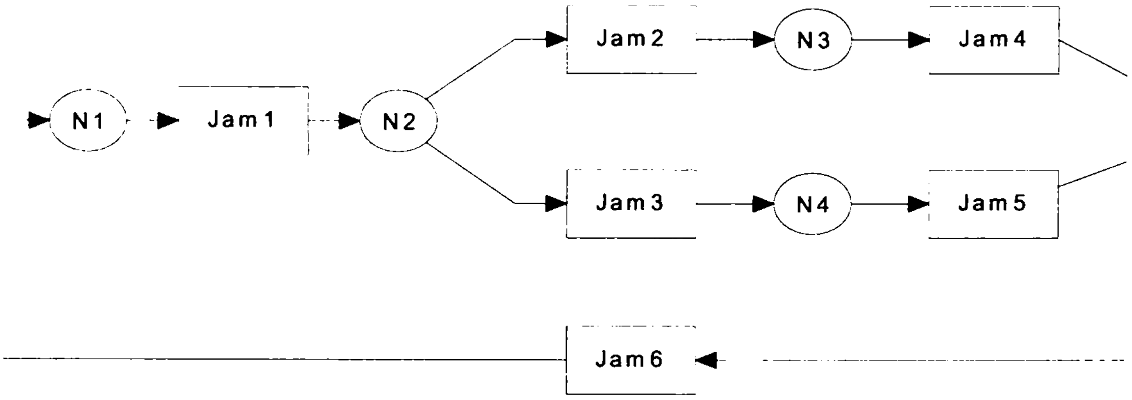


Figura 2.142. Schema de principiu a unui STZA

Sistemul este constituit din 5 noduri conectate intre ele, nodurile N1, N3 si N4 sunt noduri de tip 1, nodul N2 este nod de tip 4 si nodul N5 este nod de tip 2. Conexiunea intre noduri, pentru ca sistemul sa poata functiona, este asigurata prin intermediul jam-urilor de capacitate finita si dimensiune minima unu, numarul maxim al carucioarelor trebuind sa fie mai mic decat capacitatea maxima a tututor jam-urilor.

Conform definitiei 2.21, STZA-ul considerat este descris prin:

$$STS = \{ND, J, Conect, Cap_{max}Jam, I\},$$

unde:

- $ND = \{N_1, N_2, N_3, N_4, N_5\}$
- $J = \{Jam_1, Jam_2, Jam_3, Jam_4, Jam_5, Jam_6\}$
- $Conect = \{(Jam_1,N_2),(Jam_2,N_3),(Jam_3,N_4),(Jam_4,N_5),(Jam_5,N_5),(Jam_6,N_1)\} \cup \{(N_1,Jam_1),(N_2,Jam_2),(N_2,Jam_3),(N_3,Jam_4),(N_4,Jam_5),(N_5,Jam_6)\}$
- $Cap_{max}Jam = \{Cap_{max}Jam_1 = 3, Cap_{max}Jam_2 = 5, Cap_{max}Jam_3 = 5, Cap_{max}Jam_4 = 3, Cap_{max}Jam_5 = 3, Cap_{max}Jam_6 = 6\}$

- $I = \{NICarJam_1 = 1, NICarJam_2 = 3, NICarJam_3 = 3,$
  $NICarJam_4 = 2, NICarJam_5 = 2, NICarJam_6 = 4\}$

Pe baza celor prezentate rezulta:

- Capacitatea maxima a sistemului:

$$Cap_{max} = \sum_{i=1}^{6} Cap_{max}Jam_i = 3 + 5 + 5 + 3 + 3 + 6 = 25$$

- Numarul de carucioare existente in sistem:

$$NCar = \sum_{i=1}^{6} NICarJam_i = 1 + 3 + 3 + 2 + 2 + 4 = 15$$

Definitia 2.22. Operabilitatea unui STZA*

Un STZA indeplineste conditia de operabilitate daca:

$$\sum_{i=1}^{n} Cap_{max}Jam_i > \sum_{i=1}^{n} NICarJam_i$$

Definitia 2.23. Validarea functionarii unui nod*

Un nod este functional daca continutul jam-ului sursa este diferit de zero si in jam-ul destinatie exista cel putin o locatie libera.

Altfel spus, fie Jam_i jam-ul sursa si Jam_j jam-ul destinatie, se spune ca nodul *i* este functional daca :

$$NCarJam_i \neq 0 \quad si \quad NCarJam_j < Cap_{max}Jam_j$$

Definitia 2.24. Pasul unui STZA*

Pasul unui STZA reprezinta capacitatea STZA de a efectua o miscare a unui element (carucior) dintr-un jam sursa, intr-un jam destinatie, peste (prin intermediul unui) singur nod.

Fie nodul N_i cu jam-urile sursa Jam_i si respectiv destinatie Jam_j , executia unui *pas* in cadrul nodului N_i poate fi descrisa astfel:

```
if ($pas_i$ = TRUE){
    THEN{
        $NCarJam_i = NCarJam_i - 1;$
        $NCarJam_j = NCarJam_j + 1;$
    }
}
```

unde: $NCarJam_i$ - reprezinta numarul de carucioare din jam-ul *i* la momentul de timp considerat.

Pe baza definitiilor de mai sus se pot enunta urmatoarele propozitii:

Propozitia 2.1. Conditia de operabilitate a unui STZA*
Un STZA este operabil daca si numai daca in conditiile initiale exista cel putin o pozitie libera intr-un jam.

Demonstratie:
Fie $NCar$ numarul de carucioare din sistem. Capacitatea maxima a sistemului este data de relatia: $Cap_{max}Jam = \sum_{i=1}^{n} Cap_{max}Jam_i$.

Se presupune ca $NCar \geq Cap_{\max}Jam$, ceea ce inseamna ca toate jam-urile din sistem suntpline. Conform definitiei 2.23, pentru ca un nod sa fie validat (functional), in jam-ul destinatie trebuie sa fie cel putin o locatie libera. Dar din presupunerea facuta anterior, in sistem nu exista nici un nod care sa fie validat functional. Rezulta ca presupunerea $NCar \geq Cap_{max}Jam$ este falsa. Astfel, conditia de operabilitate a unui STZA devine:

$$Cap_{max}Jam - NCar \geq 1 .$$

Propozitia 2.2. Blocajul STZA*
Un STZA este blocat din punct de vedere al operabililitatii daca si numai daca:

$$\sum_{i=1}^{n} Cap_{max}Jam_i \leq \sum_{i=1}^{n} NICarJam_i$$

Demonstratie:
Demonstratia este imediata pornind de la observatile facute in demonstratia propozitiei 2.1. STZA este blocat operational deoarece nici un nod, conform definitiei 2.23, nu este validat.

Propozitia 2.3. Blocarea executiei unui pas*
Un pas corespunzator unui nod N_i este blocat (pas=FALSE) daca si numai daca in jam-ul destinatie nu exista locatii libere.

Demonstratie:
Fie Jam_i jam-ul sursa si Jam_j jam-ul destinatie.
Se considera ca $NCarJam_j = Cap_{max}Jam_j$. Presupunand ca *pas_i=TRUE,* in urma executiei acestui pas s-ar obtine:

$$NCarJam_i = NCarJam_i - 1 \quad ,respectiv$$

$$NCarJam_j = NCarJam_j + 1$$

dar intr-un jam pot fi maxim atatea carucioare cat este dimensiunea maxima a jam-ului corespunzator ($NCar_{max}Jam_j = Cap_{max}Jam_j$), deci presupunerea *pas_i=TRUE* nu este posibila, ceea ce implica *pas_i=FALSE*.

2.6.3. Modelarea STZA cu ajutorul retelelor Petri

Modelarea STZA, se realizeaza prin utilizarea modelelor *sistemelor de asteptare* utilizate in analiza si modelarea sistemelor de calcul [CL01][PMM02] (vezi exemplul 2.3 paragraful 2.3), modele modificate si adaptate de catre autor pentru analiza si modelarea STZA.

Structura modificata a unui sistem de asteptare este prezentata in figura 2..143.

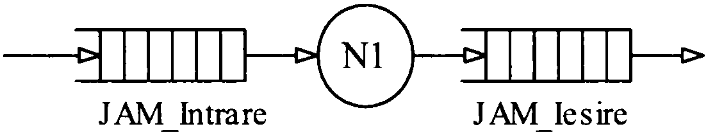


Figura 2.143. Structura valabila pentru un sistem de asteptare corespunzatoare STZA

In plus, fata de sistemul clasic, iesirea nodului N1 este conectata la jam-ul de iesire, un carucior fiind transferat, prin intermediul nodului N1, din jam-ul de intrare in jam-ul de iesire. Modelul de tip retea Petri propus pentru aceast sistem este prezentat in figura 2.144.

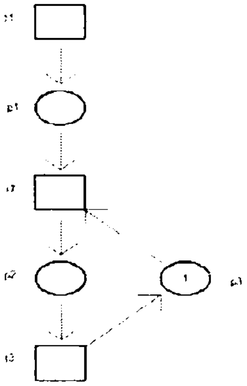


Figura 2.144. Modelul de tip retea Petri corespunzator nodului de tip 1 incadrat in STZA

Matricile de incidenta corespunzatoare sunt :

Matricea de incidenta de intrare :

$$A_I = \begin{pmatrix} 0 & 0 & 0 \\ 1 & 0 & 1 \\ 0 & 1 & 0 \end{pmatrix}$$

Matricea de incidenta de iesire :

$$A_O = \begin{pmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{pmatrix}$$

Matricea de incidenta:

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

Dupa cum se observa, structura de baza este mentinuta (pozitiile p1, p2, p3 tranzitiile t1, t2,t3 avand aceleasi semnificatii), diferenta fata de structura clasica este data de faptul ca *jam-ul* de iesire al nodului curent este *jam* de intrare din nodul urmator.

In cazul nodului de tip 2 (doua intrari o iesire) structura de la care se porneste, precum si modelul de retea Petri considerat, sunt prezentate in figurile 2.145 si 2.146.

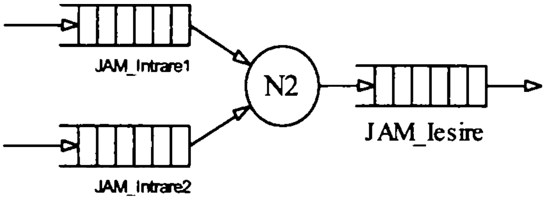


Figura 2.145. Structura nodului de tip 2 in cadrul unui STZA

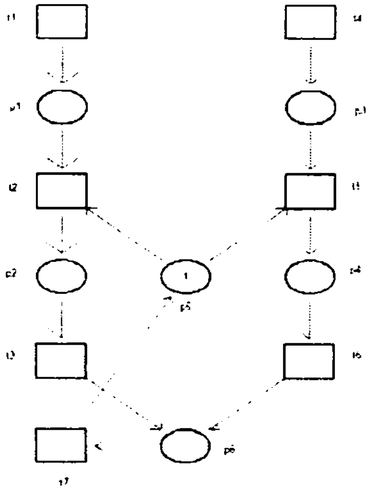


Figura 2.146. Modelul de tip retea Petri corespunzator nodului de tip 2 in cadrul unui STZA

Matricile de incidenta corespunzatoare sunt :

Matricea de incidenta de intrare :

$$A_I = \begin{pmatrix} 0 & 0 & 0 & 0 & 0 & 0 \\ 1 & 0 & 0 & 0 & 1 & 0 \\ 0 & 1 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 1 \end{pmatrix}$$

Matricea de incidenta de iesire :

$$A_O = \begin{pmatrix} 1 & 0 & 0 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 1 \\ 0 & 0 & 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 1 \\ 0 & 0 & 0 & 0 & 1 & 0 \end{pmatrix}$$

Matricea de incidenta:

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

In mod similar cazului modelarii nodului de tip 1, modelarea nodului de tip 2 din cadrul STZA se bazeaza pe sincronizarea a doua noduri de tip 1. In rest semnificatiile pozitiilor si tranzitiilor sunt aceleasi.

In cazul nodului de tip 3 (trei intrari o iesire), structura de la care se porneste precum si modelul de retea Petri considerat sunt prezentate in figurile 2.147 si 2.148.

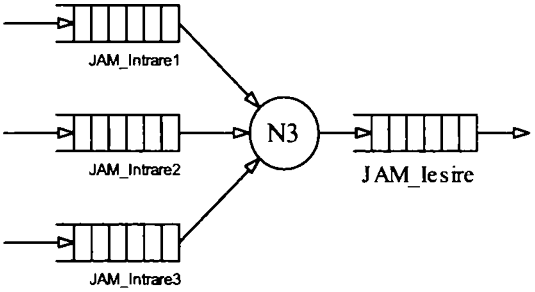


Figura 2.147. Structura nodului de tip 3 in cadrul unui STZA

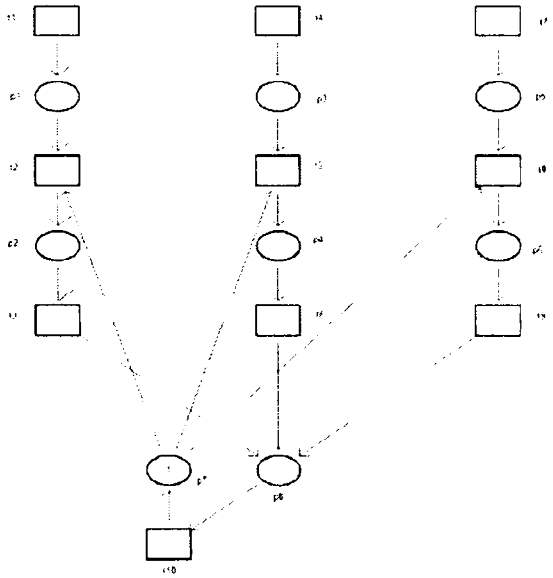


Figura 2.148. Modelul de tip retea Petri corespunzator nodului de tip 3 in cadrul unui STZA

Matricile de incidenta corespunzatoare sunt :

Matricea de incidenta de intrare :

$$A_I = \begin{pmatrix} 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 1 & 0 & 0 & 0 & 0 & 0 & 1 & 0 \\ 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 1 & 0 & 1 & 0 \\ 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 \end{pmatrix}$$

Matricea de incidenta de iesire :

$$A_O = \begin{pmatrix} 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 \\ 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 \\ 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 \\ 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 \end{pmatrix}$$

Matricea de incidenta:

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

In mod similar cazului modelarii nodului de tip 1, modelarea nodului de tip 3 din cadrul STZA se bazeaza pe sincronizarea a trei noduri de tip 1, semnificatiile pozitiilor si tranzitiilor ramanand aceleasi.

In cazul nodului de tip 4 (o intrare doua iesiri), structura de la care se porneste, precum si modelul de retea Petri considerat sunt prezentate in figurile 2.149 si 2.150.

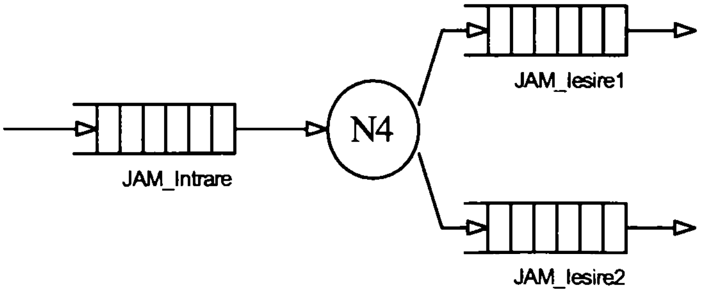


Figura 2.149. Structura nodului de tip 3 in cadrul unui STZA

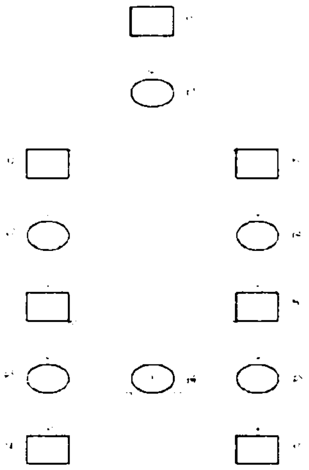


Figura 2.150. Modelul de tip retea Petri corespunzator nodului de tip 4 in cadrul unui STZA

Matricile de incidenta corespunzatoare sunt :

Matricea de incidenta de intrare :

$$A_I = \begin{pmatrix} 0 & 0 & 0 & 0 & 0 & 0 \\ 1 & 0 & 0 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 & 0 & 1 \\ 0 & 0 & 1 & 0 & 0 & 0 \\ 1 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 & 0 & 1 \\ 0 & 0 & 0 & 0 & 1 & 0 \end{pmatrix}$$

Matricea de incidenta de iesire :

$$A_O = \begin{pmatrix} 1 & 0 & 0 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 1 \\ 0 & 0 & 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 0 & 0 & 1 \end{pmatrix}$$

Matricea de incidenta:

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

In mod similar ca si in cazul modelarii nodului de tip 1 modelarea nodului de tip 4 din cadrul STZA se bazeaza pe sincronizarea a doua noduri de tip 1. In rest semnificatiile pozitilor si tranzitilor sunt aceleasi.

Datorita complexitatii modelarii unor astfel de sisteme sunt prezentate doua metode dezvoltate de autor pentru sinteza STZA:

- Prima metoda se bazeaza pe conectarea directa a modelelor corespunzatoare nodurilor utilizate;

- A doua metoda se bazeaza pe construirea matricii de incidenta a sistemului utilizand, conexiunea intre noduri fiind realizata prin introducerea unor pozitii de sincronizare.

Modelarea STZA prin conectarea directa a nodurilor

Aceasta metoda are la baza utilizarea urmatorului algoritm:

Algoritmul 2.3. *Modelarea cu ajutorul retelelor Petri a STZA prin conectare directa*

- *Pasul 1* - Stabilirea structurilor mecanice de baza corespunzatoare sistemului analizat (structura nodurilor);
- *Pasul 2* - Modelarea si analiza structurilor mecanice cu ajutorul retelelor Petri
    - *Pasul 2.1.* Elaborarea modelelor de tip retea Petri si analizarea lor acestora din punct de vedere al proprietatilor structurale si comportamentale, corespunzatoare structurilor stabilite la pasul 1;
    - *Pasul 2.2.* Analiza modelelor validate la punctul 2.1.prin asocierea timpului (retea Petri temporizata P) si stabilirea intervalelor de timp corespunzatoare pe baza algoritmului 2.1
- *Pasul 3* - Modelarea si analiza in ansamblu a intregului sistem utilizand rezulatele de la punctul 2. Modelarea si analiza intregului sistem utilizand submodelele obtinute la pasul 2.
- *Pasul 4* - Determinarea cu ajutorul algoritmului 2.1 a timpilor de executie ai retelei elaborate si validate la pasul 3
- *Pasul 5* - Validarea tuturor modelelor elaborate.

Se considera STZA din figura 2.151.

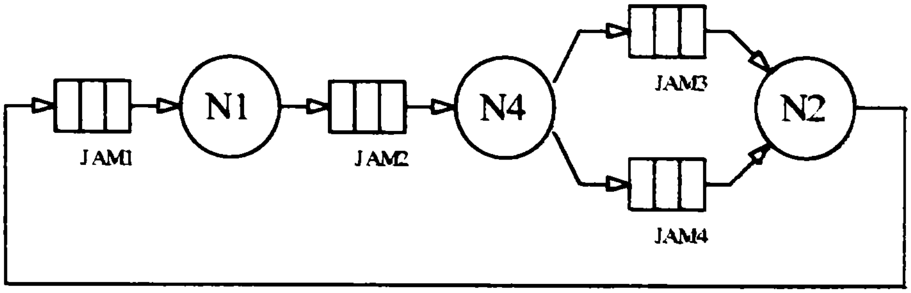


Figura 2.151. Structura STZA asociata schemei de principiu din fig. 142

Sistemul contine un nod de tip 1, un nod de tip 2 si un nod de tip 4, omitandu-se nodul de tip 3, deoarece validarea nodurilor de tip 1 si 2 implica automat validarea nodului de tip 3 (practic acesta este construit pe baza nodurilor 1 si 2).

Modelul de tip retea Petri asociata acestui sistem (*PN_Sistem*) este prezentata in figura 2.152.

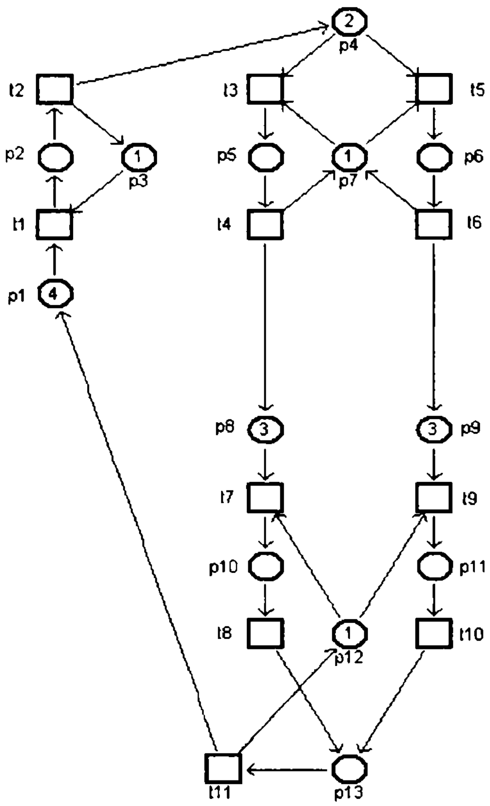


Figura 2.152. Modelul de tip retea Petri asociat sistemului din figura 2.151

Multimea pozitiilor corespunzatoare acestei retele este:

$P=\{p1,p2,p3,p4,p5,p6,p7,p8,p9,p10,p11,p12,p13\}$

iar multimea tranzitiilor:

$T=\{t1,t2,t3,t4,t5,t6,t7,t8,t9,t10,t11\}$

In figura 2.151 s-au delimitat simbolic nodurile corespunzatoare ale sistemului propus in figura 2.142. Se poate remarca modul de conexiune al nodurilor prin intermediul pozitiilor care reprezinta jam-urile (p1 - pentru JAM1, p4 - pentru JAM2, p8 - pentru JAM3 si p9-pentru JAM4.

Pentru fiecare pozitie care reprezinta un JAM, s-au fixat numarul maxim de elemente care pot sa-l contina (capacitatea maxima a jam-ului respectiv), precum si numarul de carucioare aflate in jam-ul respectiv in stare initiala (numarul de jetoane).Aceste alocari sunt prezentate in tabelul 2.18.

Tabelul 2.18 *Capacitatile maxime si continutul jam-urilor*

Jam ID	Cap. Maxima	Continut curent
Jam 1	5	4
Jam 2	3	2
Jam 3	4	3
Jam 4	4	3

Topologia retelei Petri validata de PNT este prezentata in figura 2.153.

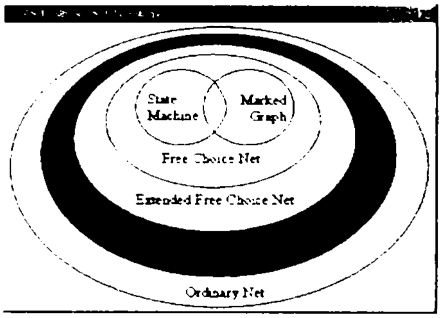


Figura 2.153. Topologia retelei Petri validata de PNT

Matricile de incidenta ale retelei Petri sunt urmatoarele :

Matricea de incidenta de intrare

$$A_I = \begin{pmatrix} 1&0&1&0&0&0&0&0&0&0&0&0&0 \\ 0&1&0&0&0&0&0&0&0&0&0&0&0 \\ 0&0&0&1&0&0&1&0&0&0&0&0&0 \\ 0&0&0&0&1&0&0&0&0&0&0&0&0 \\ 0&0&0&1&0&0&1&0&0&0&0&0&0 \\ 0&0&0&0&0&1&0&0&0&0&0&0&0 \\ 0&0&0&0&0&0&0&1&0&0&0&1&0 \\ 0&0&0&0&0&0&0&0&0&1&0&0&0 \\ 0&0&0&0&0&0&0&0&1&0&0&1&0 \\ 0&0&0&0&0&0&0&0&0&0&1&0&0 \\ 0&0&0&0&0&0&0&0&0&0&0&0&1 \end{pmatrix}$$

Matricea de incidenta de iesire

$$A_O = \begin{pmatrix} 0&1&0&0&0&0&0&0&0&0&0&0&0 \\ 0&0&1&1&0&0&0&0&0&0&0&0&0 \\ 0&0&0&0&1&0&0&0&0&0&0&0&0 \\ 0&0&0&0&0&0&1&1&0&0&0&0&0 \\ 0&0&0&0&0&1&0&0&0&0&0&0&0 \\ 0&0&0&0&0&0&1&0&1&0&0&0&0 \\ 0&0&0&0&0&0&0&0&0&1&0&0&0 \\ 0&0&0&0&0&0&0&0&0&0&0&0&1 \\ 0&0&0&0&0&0&0&0&0&0&1&0&0 \\ 0&0&0&0&0&0&0&0&0&0&0&0&1 \\ 1&0&0&0&0&0&0&0&0&0&0&1&0 \end{pmatrix}$$

Matricea de incidenta:

$$A = A_O - A_I = \begin{pmatrix} -1&1&-1&0&0&0&0&0&0&0&0&0&0 \\ 0&-1&1&1&0&0&0&0&0&0&0&0&0 \\ 0&0&0&-1&1&0&-1&0&0&0&0&0&0 \\ 0&0&0&0&-1&0&1&1&0&0&0&0&0 \\ 0&0&0&-1&0&1&-1&0&0&0&0&0&0 \\ 0&0&0&0&0&-1&1&0&1&0&0&0&0 \\ 0&0&0&0&0&0&0&-1&0&1&0&-1&0 \\ 0&0&0&0&0&0&0&0&0&-1&0&0&1 \\ 0&0&0&0&0&0&0&0&-1&0&1&-1&0 \\ 0&0&0&0&0&0&0&0&0&0&-1&0&1 \\ 1&0&0&0&0&0&0&0&0&0&0&1&-1 \end{pmatrix}$$

Pentru obtinerea arborelui de acoperire se considera situatia in care in sistem exista afla un singur carucior, situatie reprezentata prin introducerea unui jeton in pozitia p1. Aceasta conventie este facuta deoarece in cazul in care in sistem sunt mai multe carucioare (mai multe jetoane) in marcajul initial structura arborelui devine prea complicata. Arborele de acoperire corespunzator existentei unui singur carucior (jeton) este prezentat in figura 2.154.

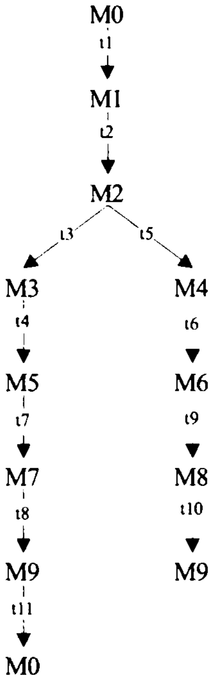


Figura 2.154. Arborele de acoperire al retelei Petri *PN_Sistem*

Unde:
M0=(1,0,1,0,0,0,1,0,0,0,0,1,0); M1=(0,1,0,0,0,0,1,0,0,0,0,1,0);
M2=(0,0,1,1,0,0,1,0,0,0,0,1,0); M3=(0,0,1,0,1,0,0,0,0,0,0,1,0);
M4=(0,0,1,0,0,1,0,0,0,0,0,1,0); M5=(0,0,1,0,0,0,1,1,0,0,0,1,0);
M6=(0,0,1,0,0,0,1,0,1,0,0,1,0); M7=(0,0,1,0,0,0,1,0,0,1,0,0,0);
M8=(0,0,1,0,0,0,1,0,0,0,1,0,0); M9=(0,0,1,0,0,0,1,0,0,0,0,0,1).

Studiind proprietatile comportamentale ale retelei Petri pe baza arborelui de acoperire rezulta:

- reteaua este *marginita* (simbolul ω nu apare in arborele de acoperire);
- reteaua este *sigura* (marcajele din toate nodurile arborelui de acoperire contin numai „0” si „1”);
- reteaua *nu este blocanta* (toate tranzitile au asociate arce in arborele de acoperire);
- reteaua este *accesibila* (orice marcaj poate fi atins pornind din M_0).

Graful de acoperire corespunzator este prezentat in figura 2.155.

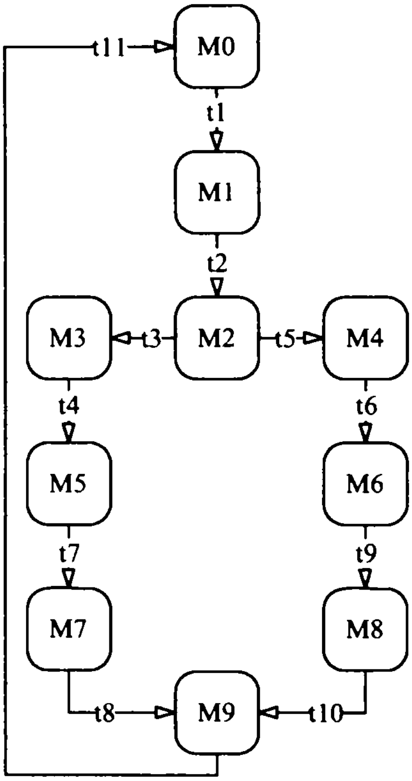


Figura 2.155. Graful de acoperire al retelei Petri *PN_Sistem*

Structura rezultata pentru graful de acoperire confirma ca modelul de retea Petri considerat pentru sistemul ales este acesibil.

Concluzionand, se poate afirma ca modelul ales este viabil din punct de vedere comportamental.

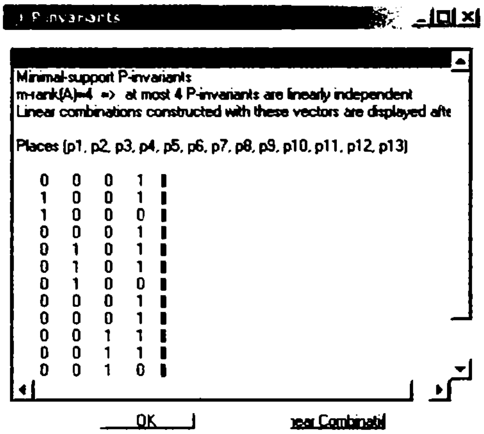
Analiza retelei Petri *PN_Sistem* din punct de vedere structural se efectueaza similar metodologiei aplicate anterior, pe baza analizei existentei invariantilor de tip *P* respectiv *T*.

Rangul matricei A, in acest caz este:

$$rang\,A = 9\,.$$

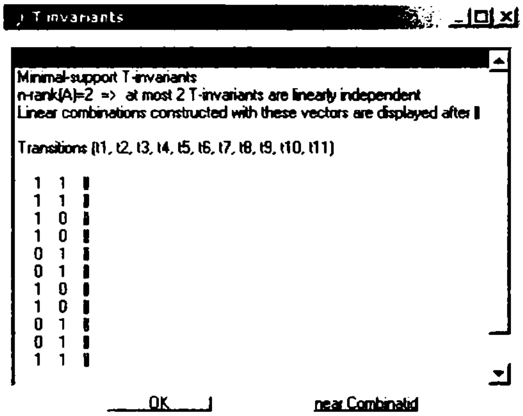
Rezulta ca reteaua Petri *PN_Sistem* este acoperita de 4 invarianti de tip P, si 2 invarianti de tip T.

In figurile 2.156 si 2.157 se prezinta invariantii de tip P respectiv T obtinuti in urma simularii retelei Petri *PN_Sistem* cu ajutorul PNT-ului.



```
Minimal-support P-invariants
m-rank(A)=4  =>  at most 4 P-invariants are linearly independent
Linear combinations constructed with these vectors are displayed afte

Places (p1, p2, p3, p4, p5, p6, p7, p8, p9, p10, p11, p12, p13)

  0  0  0  1
  1  0  0  1
  1  0  0  0
  0  0  0  1
  0  1  0  1
  0  1  0  1
  0  1  0  0
  0  0  0  1
  0  0  0  1
  0  0  1  1
  0  0  1  1
  0  0  1  0
```

Figura 2.156. Invariantii de tip P obtinuti cu ajutorul PNT



```
Minimal-support T-invariants
n-rank(A)=2  =>  at most 2 T-invariants are linearly independent
Linear combinations constructed with these vectors are displayed after

Transitions (t1, t2, t3, t4, t5, t6, t7, t8, t9, t10, t11)

  1  1
  1  1
  1  0
  1  0
  0  1
  0  1
  1  0
  1  0
  0  1
  0  1
  1  1
```

Figura 2.157. Invariantii de tip T obtinuti cu ajutorul PNT

Conform teoremelor 2.6 respectiv 2.7 reteaua Petri *PN_Sistem* este *conservativa si structural marginita, respectiv este consistenta si repetitiva.*

Reteaua propusa fiind validata din punct de vedere structural si comportamental, se poate trece la analiza retelei prin asocierea fiecarei pozitii a unui interval de timp corespunzator. Intervalele de timp alocate sunt prezentate in tabelul 2.19.

Tabelul 2.19. *Intervalele de timp corespunzatoare pozitiilor P*

Pozitie	Unitati de Timp	
	Min	Max
P1,P4,P8,P9	0	0
P2,P5,P6,P10,P11	4	37
P3,P7,P12	0	0
P13	0	0

Se observa ca numai pozitiile corespunzatoare nodurilor (miscare respectiv jam) au timpi alocati diferiti de zero, celelalte pozitii considerandu-se ca sunt activate instantaneu.

Pe baza rezultatelor obtinute, se poate afirma solutia propusa pentru modelarea in ansamblu al STZA, prin conectarea directa a nodurilor, cu ajutorul modelelor de tip retele Petri este viabila ea putand fi utilizata cu succes in analiza sistemelor complexe.

Modelarea STZA prin determinarea matricii de incidenta a STZA[Ung07b]

Detrminarea matricii de incidenta a unui STZA avand ca suport matricile nodurilor componente se face pe baza urmatorului algoritm elaborat de autor.

Algoritmul 2.4. *Modelarea cu ajutorul retelelor Petri a STZA prin determinarea matricii de incidenta*

- *Pasul 1* - Se stabilesc nodurile de baza necesare modelarii STZA;
- *Pasul 2* - Se construieste matricea intermediara A_{INT} de dimensiune (n_t,n_p) unde numarul de linii n_t este dat de numarul total de tranzitii din system, iar numarul de coloane n_p este dat de numarul total de pozitii din system;
- *Pasul 3* - Se atribuie elementelor matricii A_{INT} valorile corespunzatoare din matricile de incidenta ale nodurilor: $a_{INT_{ij}} = a_{NOD_{ij}}$, unde *i* reprezinta tranzitia iar *j* pozitia;
- *Pasul 4* - Se adauga coloane corespunzatoare pozitiilor de sincronizare, rezultand astfel matricea de incidenta cautata A_{STZA} , efectuandu-se in acelasi timp si modificarea in mod corespunzator a elementelor matricii A_{STZA} pentru conectarea acestor noi pozitii in system;
- *Pasul 5* - Pe baza matricii de incidenta A_{STZA} se poate construii graful corespunzator retelei Petrii ale STZA considerat.

Exemplificarea modului de utilizare a algoritmului 2.4 se face avand ca suport structura pentru STZA prezentata in figura 2.151.

Pasul 1. Nodurile utilizate sunt: pentru N1 – nod de tip 1, pentru N2 – nod de tip 4 iar pentru N3 – nod de tip 2.

Pasii 2si 3. Matricea A_{INT} are urmatoarea structura:

$$A_{INT} = \begin{array}{c|c|c|l}
p1\ p2\ p3 & p4\ p5\ p6\ p7\ p8\ p9 & p10\,p11\,p12\,p13\,p14\,p15p & \\
 & & & t1 \\
A_{N1} & 0 & 0 & t2 \\
 & & & t3 \\
\hline
 & & & t4 \\
 & & & t5 \\
 & & & t6 \\
0 & A_{N2} & 0 & t7 \\
 & & & t8 \\
 & & & t9 \\
 & & & t10 \\
\hline
 & & & t11 \\
 & & & t12 \\
 & & & t13 \\
0 & 0 & A_{N4} & t14 \\
 & & & t15 \\
 & & & t16 \\
 & & & 7
\end{array}$$

Pasul 4. Matricea A_{STZA} are urmatoarea structura:

$$A_{STZA} = \begin{array}{c|c|c|cccc|l}
p1\ p2\ p3 & p4\ p5\ p6\ p7\ p8\ p9 & p10\,p11\,p12\,p13\,p14\,p15 & p16 & p17 & p18 & p19 & \\
 & & & 0 & 0 & 0 & -1 & t1 \\
A_{N1} & 0 & 0 & 0 & 0 & 0 & 0 & t2 \\
 & & & 1 & 0 & 0 & 0 & t3 \\
\hline
 & & & -1 & 0 & 0 & 0 & t4 \\
 & & & 0 & 0 & 0 & 0 & t5 \\
 & & & 0 & 0 & 0 & 0 & t6 \\
0 & A_{N2} & 0 & 0 & 1 & 0 & 0 & t7 \\
 & & & 0 & 0 & 0 & 0 & t8 \\
 & & & 0 & 0 & 0 & 0 & t9 \\
 & & & 0 & 0 & 1 & 0 & t10 \\
\hline
 & & & 0 & -1 & 0 & 0 & t11 \\
 & & & 0 & 0 & 0 & 0 & t12 \\
 & & & 0 & 0 & 0 & 0 & t13 \\
0 & 0 & A_{N4} & 0 & 0 & -1 & 0 & t14 \\
 & & & 0 & 0 & 0 & 0 & t15 \\
 & & & 0 & 0 & 0 & 0 & t16 \\
 & & & 0 & 0 & 0 & 1 & t17
\end{array}$$

Pasul 5. Graful corespunzator este prezentat in figura 2.158.

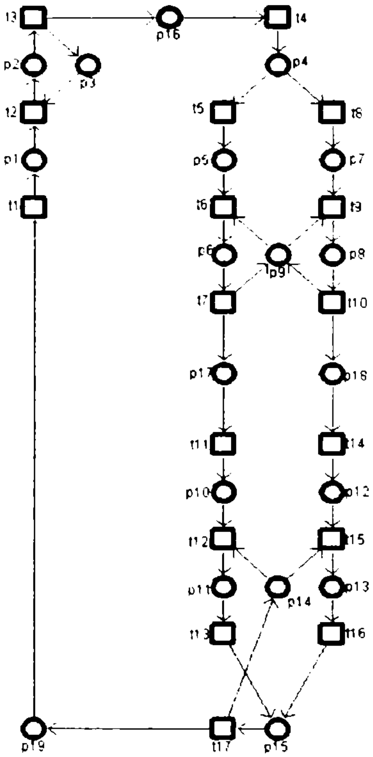


Figura 2.158. Graful corespunzator retelei Petri obtinute prin aplicarea algoritmului 2.4

Sincronizarea nodurilor este realizata prin intermediul pozitiilor p16, p17, p18 si p19, aceste pozitii avand asociati timpi egali cu zero. In acest mod ele nu influenteaza timpii de executie a retelei.

Prima metoda propusa este greu de utilizat, in cazul sistemelor complexe, deoarece modelarea se face pornind de la construirea grafului, structura matricii de incidenta fiind influentata puternic de structura sistemului. In acest caz este imposibila utilizarea metodei pentru un caz general.

Metoda a doua este mai avantajoasa, ea putand fi considerata ca metoda generala de modelare avand ca suport obtinerea matricii de incidenta a sistemului asigurandu-se astfel modularizarea in ansamblu.

2.7. Concluzii

Modelarea reprezinta o componenta importanta a proiectarii ingineresti fiind utilizata la analiza comportamentului, evaluarea performantelor si optimizarea sistemelor. SED constituie o clasa aparte de sisteme dinamice neliniare, care necesita pentru investigare instrumente proprii de modelare si analiza, complet diferite de cele utilizate la sistemele clasice (continuale/discrete) bazate pe ecuatii diferentiale sau ecuatii cu diferente finite. Scopul analizei SED este dezvoltarea de modele corespunzatoare, care descriu in mod adecvat comportamentul acestor sisteme.

Sintetizarea principalelor concepte din cadrul domeniului sistemelor cu evenimente discrete a demonstrat importanta acestui domeniu ca suport de modelare ca SED a sistemelor de transport cu zone de acumulare.

In cadrul acestui capitol s-a urmarit realizarea unei sinteze legate de aspectul teoretic al modelarii SED. Temele abordate, din acest punct de vedere, au fost directionate, pe de o parte, pe aspectele legate de utilizarea limbajelor si automatelor ca metode de modelare a SED, iar pe de alta parte, pe sintetizarea aspectelor legate de modelarea SED cu ajutorul retelelor Petri.

In cazul utilizarii retelelor Petri ca tehnica de modelare, au fost abordate atat metodele clasice (retele Petri ordinare, retele Petri temporizate), cat si metodele de data recenta de abordare a domeniului (retele Petri etichetate, utilizarea submodelelor de tip retele Petri).

De asemenea a fost abordata, din punct de vedere teoretic, problematica legata de conexiunile dintre modelele de tip automat si modelele de tip retele Petri.

Contributiile legate de aspectele teoretice ale modelarii SED cu ajutorul automatelor, respectiv al reteleor Petri sunt prezentate in capitolul 6.

Aspectele teoretice legate de modelarea SED au fost utilizate pentru modelarea STZA. Astfel, a fost prezentat conceptul de modelare a STZA ca SED, elaborandu-se patru structuri de baza denumite generic noduri.

Pornind de la aceste structuri de baza au fost elaborate trei tipuri de modele pentru fiecare tip de nod: modelul de tip automat, modelul de tip retea Petri netemporizata si modelul de tip retea Petri temporizata. Fiecare model in parte a fost analizat si validat utilizand mediul Matlab.

Dupa validarea acestor modele particulare au fost elaborate, analizate si validate modele corespunzatoare de ordin general.

Un paragraf important al acestui capitol este legat de prezentarea STZA din punct de vedere teoretic, fiind formulate o serie de definitii, teoreme si leme corespunzatoare sintetizarii notiunilor teoretice legate de STZA „vazute” ca SED.

Au fost prezentate metode de modelare a nodurilor complexe, si nu in ultimul rand a fost elaborate doua metodologii bazata pe cozile de asteptare pentru obtinerea modelului unui STZA. Un model particular bazat pe construirea grafului corespunzator unui STZA, respectiv un model determinat pe construirea matricii de incidenta utilizand pozitii suplimentare de sincronizare.

Contributiile legate de aspectele modelarii STZA ca SED sunt prezentate in detaliu in capitolul final al lucrarii (capitolul 6).

Capitolul 3

Conducerea supervizata a sistemelor cu evenimente discrete

Notiunea de supervizor a fost introdusa de P.J. Ramadge si W.M. Wonham in contextul teoriei supervizarii, dezvoltata in anul 1987 si prezentata in [RW87][RW89]. Problematica legata de conducerea supervizata a SED a fost abordata ulterior in foarte multe lucrari [WR88] [CL01] [GM04] [GM05a][Khou04][Vahi04][XTPS02][KN06]. In [CL01] se prezinta modul de conducere supervizata a unui SED pe baza modelelor de tip automat. Pe langa conducerea supervizata centralizata, neumeroase lucrari abordeaza subiecte legate de conducerea supervizata prin utilizarea metodelor de conducere supervizata ierarhizata [Daro05a] [GM05b] [LBW00] [LBWL01] [Led02] [LLD06] [LLW01] [LWL01].

In cadrul acestui capitol se abordeaza problematica legata de elaborarea unor algoritmi de implementare a supervizoarelor corespunzatori sistemelor centralizate, modelate cu ajutorul automatelor, respectiv a retelelor Petri.

3.1. Structura de baza utilizata in conducerea supervizata a SED [CL01][RW87]

Conducerea sistemelor automate complexe (de ex. sisteme flexibile de fabricatie, sisteme de transport etc.) necesita o structura de conducere ierarhizata (multinivel) pornind de la nivelul ierarhic cel mai de jos (reprezentat prin senzori si traductoare, elemente de executie etc.), pana la programe software prin care se implementeaza algoritmi de conducere complecsi necesari unei conduceri supervizate.

Pentru monitorizarea/conducerea proceselor, considerate ca fiind SED, este necesara utilizarea unui controler supervizor (supervizor), cu ajutorul caruia se urmareste atat starea procesului cat si informatii provenite de la alte componente, necesare generarii comenzilor.

Comportamentul controlerului supervizor depinde de doua tipuri de informatii: conditii si evenimente, care constituie intrarile controlerului.

Starea unui controler supervizor se poate modifica:

- daca o conditie este adevarata - aceatsa poate fi exprimata prin variabile logice care corespund unor marimi interne sau externe. In cazul conditiilor externe acestea se refera la starea sistemului, putand fi carcterizata prin predicate, care sunt propozitii adevarate sau false si care pot fi modificate prin schimbarea unor variabile logice.
- cand se realizeaza un eveniment - acesta se refera la schimbarea starii sistemului (schimbare numita eveniment). Un proces tehnic cu functionare pilotata de evenimente este constituit dintr-o multime de resurse care sunt

utilizate pentru a efectua o succesiune de operatii. Realizarea fiecarei operatii necesita alocarea uneia sau mai multor resurse care sunt eliberate dupa incheierea operatiei respective.

Problema conducerii supervizate a unui de proces consta in a asigura indeplinirea urmatoarelor conditii de functionare:

- corectitudinea succesiunii operatiilori;
- corectitudinea alocarii si eliberarii resurselor necesitate de fiecare operatie in parte;
- prestarea unui anumit tip de serviciu de indata ce resursele necesare pentru operatia respectiva sunt disponibile;
- repetabilitatea prestarii servicilor, fara blocaje circulare datorate utilizarii partajate a unora dintre resurse.

In figura 3.1 se prezinta o structura de conducere care asigura satisfacerea conditiilor de functionare.

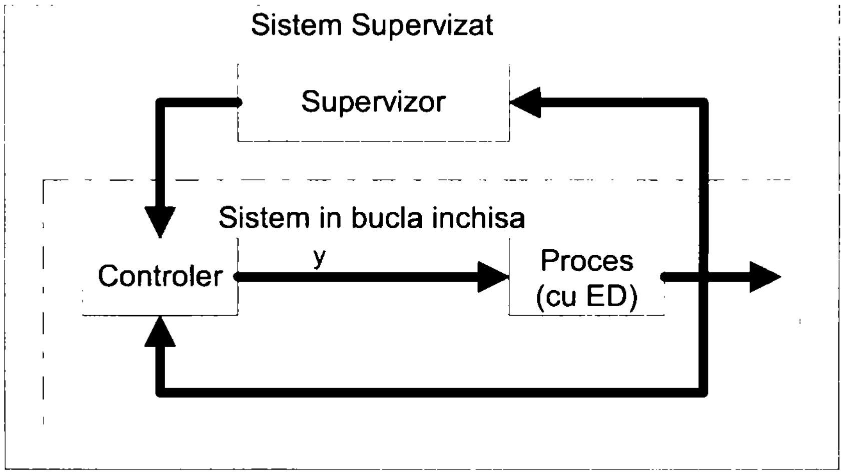


Figura 3.1. Structura de conducere

*Un **supervizor** este o unitate care supravegheaza si ghideaza comportamentul unui subsitem cu evenimente discrete condus.*

Pentru structura de conducere reprezentata in figura 3.2, din multitudinea de traiectorii de stare posibile ale sistemului, supervizorul le va selecta pe cele care indeplinesc anumite conditii impuse. Supervizorul nu creeaza noi evolutii posibile, el alegand din cele existente pe cele dorite.

Supervizorul unui SED are rolul de baza, de a preveni realizarea unor evenimente in sistemul supervizat, prin introducerea unor specificatii. Specificatiile pot fi de doua categorii:

- probleme de evitare a starilor, unde obiectivul consta in evitarea unor anumite de stari;
- probleme de evitare a secventelor, care au ca obiectiv evitarea unor anumite secvente de evenimente nedorite.

Evitarea unor stari, sau a unor secvente de evenimente, este echivalenta cu introducerea unor constrangeri. In consecinta supervizorul trebuie sa fie capabil sa introduca aceste constrangeri si poate fi realizat sub forma unui controler on-line.

Functiile posibile ale supervizorului sunt:

- sa impiedice sistemul sa intre in anumite stari interzise (evitarea starilor blocante);
- sa impiedice executia unor secvente nedorite;
- sa forteze executia unor secvente dorite din anumite stari;
- sa forteze ajungerea in anumite stari dorite, din stari date;
- sa rezolve conflictele.

Pentru ca supervizorul sa ghideze subsistemul intre mai multe evolutii posibile, trebuie sa existe criterii de alegere pe care sa le poata implementa. In acest sens, exista mai multe solutii posibile.

Daca solutia aleasa se bazeaza pe un supervizor optimal, atunci trebuie sa:

- se precizeaza un criteriu de performanta;
- se minimeaza sau se maximizeaza criteriul de performanta;
- se transmit informatii supervizorului, astfel incat, acesta sa determine o evolutie dupa cea mai performanta traiectorie de stare, a sistemului in bucla inchisa.

Daca trebuie sa se implementeze anumite reguli de decizie, atunci este necesar sa existe un mod de descriere a acestora.

3.2. Conducerea supervizata pe baza modelelor de tip automat

3.2.1. Supervizarea sistemelor cu reactie dupa stare [CL01][RW87] [Wohn02] [Schm05]

Formularea problemei de conducere porneste de la urmatoarele: se considera un SED modelat prin *intermediul unei perechi de limbaje*, L si L_m, unde L este setul tuturor sirurilor pe care SED-ul considerat le poate genera si $L_m \subseteq L$ este limbajul sirurilor marcate, care sunt utilizate la reprezentarea completa a unor operatii sau taskuri; *definirea lui L_m este in fapt rezultatul modelarii*. L si L_m sunt definite pornind de la multimea evenimentelor E. L este intotdeauna prefix-inchizator, ceea ce inseamna $L = \overline{L}$ iar L_m nu este obligatoriu sa fie prefix-inchis. Fara a pierde din generalitatea abordarii se admite ca L si L_m sunt limbaje generate si marcate de automatul:

$$G = (X, E, f, \Gamma, x_0, X_m) \qquad (3.1)$$

unde X nu este obligatoriu sa fie finit, ceea ce conduce la:

$$L(G) = L \quad si \quad L_m(G) = L_m. \qquad (3.2)$$

Astfel, se poate vorbi de sistemul cu evenimente discrete G – „SED G”. In figura 3.2 se prezinta modul de conectare al unui supervizor.

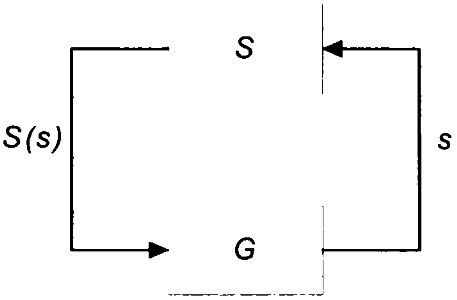


Figura 3.2. Conexiunea de tip reactie dintre supervizorul S si sistemul necotrolat reprezentat de automatul G

Definitia 3.1. Supervizorul

Se considera Σ *un alfabet relativ la un set de evenimente dat, si fie* $L_1, L_2 \subseteq \Sigma^*$ *doua limbaje. Fie sistemul* $G = (L_1, L_2)$*, pentru care sunt indeplinite conditile:*

- L_1 *si* L_2 *sunt limbaje regulare;*
- L_1 *este prefix-inchizator;*
- $L_2 \subseteq L_1$*;*
- $\Sigma = \Sigma_{uc} \cup \Sigma_c$*, unde* Σ_{uc} *este setul evenimentelor necontrolabile iar* Σ_c *este setul evenimentelor controlabile.*

Un supervizor S pentru G este o harta:

$$S : L_1 \to \Gamma, \tag{3.3}$$

Unde S(s) reprezinta setul evenimentelor valide dupa executia unui sir $s \in L_1$.

Pornind de la definitia 3.1, se doreste determinarea unui supervizor *S,* care sa interactioneze cu *G,* intr-o conexiune de tip reactie (figura 3.3).

Fie Σ format din doua multimi disjuncte Σ_c si Σ_{uc}

$$\Sigma = \Sigma_c \cup \Sigma_{uc} \tag{3.4}$$

unde:

- Σ_c este setul evenimentelor controlabile: acestea sunt evenimentele care pot fi evitate in momentul aparitiei sau pot fi inhibate de catre supervizorul *S;*
- Σ_{uc} este setul evenimentelor necontrolabile: aceste evenimente nu pot fi evitate in momentul aparitiei lor de catre supervizorul *S.*

Se presupune ca toate evenimentele din Σ executate de *G* sunt observabile de catre supervizorul *S.* In figura 3.2 *s* este sirul tuturor evenimentelor posibile de executat de catre *G* si sunt in intregime supravegheate de *S.*

In general, modul de operare al sistemului poate fi descris: functia de tranzitiei a lui *G* poate fi controlata de *S,* in sensul ca evenimentele controlabile a lui *G* pot fi validate sau invalidate in mod dinamic prin intermediul lui *S.* Formal, un supervizor *S* este o functie definita pe limbajul generat de *G* si cu valori in setul puterilor lui Σ :

$$S : L(G) \to 2^{\Sigma}. \tag{3.5}$$

Pentru fiecare $s \in L(G)$ generat din *G* (sub controlul lui *S*),

$$S(s) \cap \Gamma(\delta(x_0,s)) \qquad (3.6)$$

este un set de evenimente valide, pe care *G* le executa pornind din starea curenta cu ajutorul functiei $\delta(x_0,s)$. Cu alte cuvinte, *G* nu poate executa un eveniment chiar daca acesta exista in setul curent de evenimente $\Gamma(\delta(x_0,s))$, daca acest eveniment nu este continut in *S(s)*. Setul de evenimente Σ este in fapt o reuniune intre multimea evenimentelor controlabile si multimea evenimentelor necontrolabile.

Se spune despre un supervizor *S* ca este admisibil daca pentru toate $s \in L(G)$:

$$\Sigma_{uc} \cap \Gamma(\delta(x_0,s)) \subseteq S(s) \qquad (3.7)$$

ceea ce inseamna ca *S* nu este tolerant totdeauna la posibilele evenimentele necontrolabile dezactivate.

S(s) este functie de conducere relativa la *s*. Functia de conducere poate fi schimbata intr-un subset de supraveghere a starilor. Admitand un automat *G* si un supervizor admisibil *S*, rezultatul conducerii sistemului in bucla inchisa este dat de *S/G* („*S conduce G*"). Sistemul condus *S/G* este un SED, care poate fi caracterizat prin limbajul generat si limbajul marcat. Aceste doua limbaje sunt simple subseturi a lui *L(G)* si respectiv $L_m(G)$ continand sirurile ramase fezabile in prezenta lui *S*.

Definitia 3.2 Limbaje generate si marcate de *S/G*

***Limbajul generat** de sistemul in bucla inchisa S/G este definit recursiv astfel:*

1. $\varepsilon \in L(S/G)$ (3.8)
2. $[(s \in L(S/G))$ *si* $(s\sigma \in L(G))$ *si* $(\sigma \in S(s))] \Leftrightarrow [s\sigma \in L(S/G)]$. (3.9)

***Limbajul marcat** de sistemul in bucla inchisa S/G este definit ca:*

$$L_m(S/G) := L(S/G) \cap L_m(G) \qquad (3.10)$$

Definitia 3.3 Blocajul in sisteme controlate

Un SED S/G este blocant daca:

$$L(S/G) \neq \overline{L_m(S/G)} \qquad (3.11)$$

si este neblocant cand:

$$L(S/G) = \overline{L_m(S/G)} \qquad (3.12)$$

Un supervizor *S* care controleaza un SED *G* este blocant daca *S/G* este blocant si un supervizor *S* este neblocant daca *S/G* este neblocant.

Definitia 3.4 Controlabilitatea

Fie K si $M = \overline{M}$ *limbaje relative la un set de evenimente* Σ, *si fie* Σ_{uc} *proiectat ca un subset a lui* Σ. *Se spune ca K este controlabil relativ la M si* Σ_{uc} *daca:*

$$\overline{K}\Sigma_{uc} \cap M \subseteq \overline{K}. \qquad (3.13)$$

□

Teorema 3.1. Teorema Controlabilitatii

Fie SED reprezentat prin $G=(X,\Sigma,\delta,\Gamma,x_0)$, *unde* $\Sigma_{uc} \subseteq \Sigma$ *este setul evenimentelor necontrolabile, si fie* $K \subseteq L(G)$, *unde* $K \neq 0$. *Atunci exista un supervizor S astfel incat* $L(S/G) = \overline{K}$, *daca si numai daca*

$$\overline{K}E_{UC} \cap L(G) \subseteq \overline{K} . \qquad (3.14)$$

Aceasta conditie impusa lui K se numeste ***conditia de controlabilitate***.

Demonstratia acestei teoreme se gaseste in [CL01].

3.2.2. Algoritmi de implementare a supervizoarelor bazate pe modele de tip automat

Pentru sinteza si implementarea unui supervizor trebuie indeplinite urmatoarele trei elemente de baza:

1. Specificarea supervizorului;
2. Supervizorul trebuie sa evite situatiile blocante;
3. Supervizorul trebuie sa fie controlabil.

In literatura nu se prezinta o solutie directa care sa rezolve problema intr-un singur pas, toate metodele de implementare bazandu-se pe metode iterative. Situatiile cele mai critice se datoreaza faptului ca un supervizor non-blocant poate fi necontrolabil si invers, un supervizor controlabil poate fi blocant.

Pornind de la aceste observatii trebuie avute in vedere toate aspectele vizate (nonblocaj si controlabilitate).

Pentru prezentarea algoritmilor de implementare a supervizoarelor, se considera un sistem care nu satisface din punct de vedere al comportamentului necontrolabil. Dupa cum s-a precizat anterior, in acest caz este necesara utilizarea unui supervizor. Efectul supervizorului asupra sistemului va fi dat de faptul ca acesta va impune o serie de restrictii comportamentale.

Un rol deosebit de important il are modul de specificare a supervizorului, literatura de specialitate prezentand mai multe metode de specificare [CL01] [RW87] [Wohn02] [Schm05]:

- prin limbajul generat $L(S/G)$;
- prin limbajul marcat $L_m(S/G)$;
- prin automatul corespunzator.

Aceste metode de specificare sunt metode „text" care trebuie transformate in modele corespunzatoare procesului considerat. O specificare urmareste in fapt cea ce este relevant pentru un model, functie de cum se doreste a fi realizat. Trebuie mentionat faptul ca un proces limiteaza el insusi modelul.

Specificatia corespunzatoare unui sistem contine:

- starile marcate;
- starile interzise;
- primul venit - primul servit, cu referire la evenimente;
- evenimentele alternante;
- specificarea ordinii evenimentelor;
- secventele evenimentelor interzise;
- de cate ori evenimentul *„a"* este permis intre doua evenimente *„b"*.

De asemenea specificarea poate fi partiala sau totala. Astfel, o *specificare totala* este data de faptul ca toate evenimentele sunt observabile. O specificare partiala se bazeaza pe faptul ca numai o parte din evenimente (un subset) sunt observabile.

O specificare totala este data de asemenea de faptul ca specificatile automatului sunt si specificatile supervizorului:

$$S = S_P = S_P \| G \tag{3.19}$$

O specificatie partiala poate fi data si prin faptul ca evenimentele neobservabile nu sunt restrictive, sistemul putand lua decizia cum sa reactioneze in cazul aparitiei unui astfel de eveniment. Specificatiile mai pot fi *statice* (situatie in care starile marcate si starile interzise sunt definite explicit), sau *dinamice* (situatie in care starile marcate si starile interzise nu sunt definite explicit).

Algoritmul 3.1 *Sinteza supervizorului prin* ***metoda Wonham*** *[RW88]*

Algoritm a fost propus de Wonham, si se bazeaza pe operatorii definiti pentru limbaje In cazul acestui algoritm singurele restrictii impuse sunt legate de modelul care caracterizeaza comportamentul inchis si marcat al sistemului.

Pentru sinteza unui supervizor monolitic este:

- ***Pasul 1*** Se construieste prin compunere paralela $G = G_1 \|_d G_2 \|_d \ldots \|_d G_m$, G reprezentand modelul obtinut prin unificarea comportamentelor tuturor subsistemelor considerate. Alfabetul lui G este $\Sigma = \Sigma_1 \cup \Sigma_2 \cup ..... \cup \Sigma_m$

- ***Pasul 2*** Extinderea specificatiilor corespunzatoare buclei de reactie Hj prin introducerea tuturor evenimentelor etichetate din G, dar care pentru Hj nu reprezinta constrangeri. Aceste etichete se regasesc in: $\Sigma_{e_j} = \Sigma \setminus \Sigma_{H_j}$, automatele noi redenumindu-se ca fiind $H_1', ..., H_n'$

- ***Pasul 3*** Se consruieste intersectia: $H = H_1' \cap H_n'$. H reprezentand in acest caz specificatiile globale corespunzatoare comportamentului dorit a fi impus lui G.
- ***Pasul 4*** Se construieste intersectia , unde E reprezinta comportamentul global al intregului sistem considerat, impus prin specificatii
- ***Pasul 5*** Automatul obtinut nu reprezinta supervizor dorit. Pentru verificarea conditiei de supervizor trebuiesc verificate:
  - ***Pasul 5.1*** Verificarea nonblocajului corespunzator lui E pentru determinarea starilor blocante ale sistemului
  - ***Pasul 5.2*** Verificarea controlabilitatii lui E, avandu-se in vedere ca limbajul L(E) poate fi necontrolabil in raport cu L(G). Aceasta inseamna ca in situatia in care G si E ruleaza in paralel sa apara situatii de genul ca un eveniment necontrolabil validat de G sa fie invalidat de E.

Pentru evitarea obtinerii unui sistem blocant sau necontrolabil pentru E, trebuie impuse cateva restrictii minimale, rezultand astfel un supervizor S care este neblocant si controlabil.

Exemplul 3.1 Implementarea unui supervizor pentru o celula flexibila de fabricatie (CFF) pe baza algoritmului Wonham

Fie sistemele *G1, G2* si specificatiile comportamentale *H* (vezi figura 3.3).

In cadrul CFF, se considera *G1* ca fiind un brat de robot care ia o piesa din sistemul de transport (evenimentul *a*) si depune (incarca) o masina (evenimentul *b*). *G2* este o masina care poate incepe lucrul (la aparitia evenimentului *c*) si poate depune piesa prelucrata fie pe conveiorul A fie pe conveiorul B (evenimentele *d* respectiv *e*), dupa care revine in starea initiala de asteptare. Specificatiile sunt introduse prin intermediul generatorului *H*, in care se specifica faptul ca masina poate incepe lucrul numai dupa ce a fost incarcata (evenimentele *b* si *c* trebuie sa se produca alternativ). Se considera ca singurul eveniment necontrolabil este *b*. Pentru obtinerea supervizorului dorit se va aplica algoritmul prezentat mai sus.

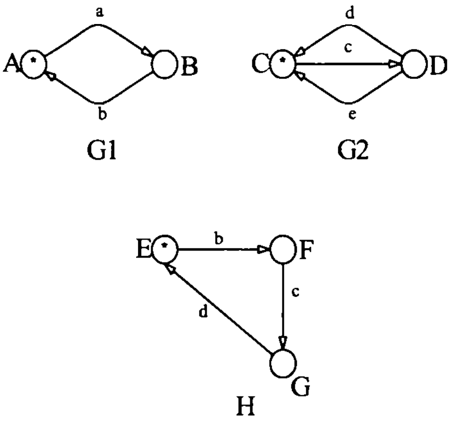


Figura 3.3. Structura sistemelor G1 si G2 precum si specificatile H corespunzatoare exemplului 3.1

Specificatiile automatului *G1* sunt:

$$G1 = (Q_1, \Sigma_1, \delta_1, i_1, M_1)$$

unde:

1. Q_1 – multimea starilor: $Q_1 = \{A, B\}$
2. Σ_1 - multimea evenimentelor: $\Sigma_1 = \{a, b\}$
3. δ_1 - functia de tranzitie:

   $\delta_1(A, a) = B$

   $\delta_1(B, b) = A$
4. i_1 – starea initiala: $i_1 = A$
5. M_1 – multimea starilor marcate: $M_1 = \{A, B\}$

Specificatile automatului *G2* sunt:

$$G2 = (Q_2, \Sigma_2, \delta_2, i_2, M_2)$$

unde:

1. Q_2 – multimea starilor: $Q_2 = \{C, D\}$
2. Σ_2 - multimea evenimentelor: $\Sigma_2 = \{c, d, e\}$
3. δ_2 - functia de tranzitie:

$\delta_2(C,c) = D$
$\delta_2(D,d) = C$
$\delta_2(D,e) = C$

4. i_2 – starea initiala: $i_2=C$
5. M_2 – multimea starilor marcate: $M_2 = \{C,D\}$

Specificatile automatului H sunt:

$$H = (Q_H, \Sigma_H, \delta_H, i_H, M_H)$$

unde:

1. Q_H – multimea starilor: $Q_H = \{E,F,G\}$
2. Σ_H - multimea evenimentelor: $\Sigma_H = \{b,c,d\}$
3. δ_H - functia de tranzitie:

$\delta_H(E,b) = F$
$\delta_H(F,c) = G$
$\delta_H(G,d) = E$

4. i_H – starea initiala: $i_H=E$
5. M_H – multimea starilor marcate: $M_H = \{E,F,G\}$

Cu aceste date algoritmul de sinteza a supervizorului monolitic dorit devine:

Pasul 1 : *construirea lui* $G = G_1 \,||\, G_2$.

Automatul rezultat in urma compunerii paralele este:

$$G = G1 \,||\, G2 = (Q_{G1||G2}, \Sigma_{G1||G2}, \delta_{G1||G2}, i_{G1||G2}, M_{G1||G2})$$

unde:

1. $Q_{G1||G2}$ – multimea starilor: $Q_{G1||G2} = \{A.C, A.D, B.C, B.D\}$
2. $\Sigma_{G1||G2}$ - multimea evenimentelor: $\Sigma_{G1||G2} = \{a,b,c,d,e\}$
3. $\delta_{G1||G2}$ - functia de tranzitie:

$\delta_{G1||G2}(A.C, a) = B.C$
$\delta_{G1||G2}(A.C, c) = A.D$
$\delta_{G1||G2}(B.C, b) = A.C$
$\delta_{G1||G2}(B.C, c) = B.D$
$\delta_{G1||G2}(A.D, a) = A.C$
$\delta_{G1||G2}(A.D, d) = A.C$
$\delta_{G1||G2}(A.D, e) = A.C$
$\delta_{G1||G2}(B.D, b) = A.D$
$\delta_{G1||G2}(B.D, d) = B.C$
$\delta_{G1||G2}(B.D, e) = B.C$

4. $i_{G1||G2}$ – starea initiala: $i_{G1||G2}=A.C$

5. $M_{G1||G2}$ – multimea starilor marcate: $M_{G1||G2} = \{A,B,C,D\}$

Automatul G1||G2 rezultat in urma compunerii paralele este prezentat in figura 3.4.

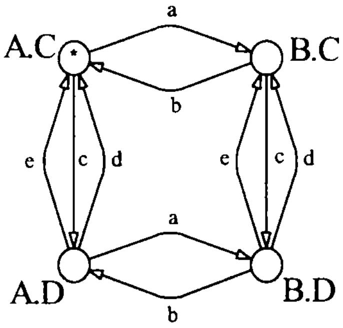


Figura 3.4. Structura automatului G1||G2

Pasul 2: *construirea lui H'* prin marirea specificatiilor.

In plus fata de structura initiala se adauga evenimentele *a* si *e* corespunzatoare fiecarei stari.

Automatul rezultat este:

$$H' = (Q_{H'}, \Sigma_{H'}, \delta_{H'}, i_{H'}, M_{H'})$$

unde:

1. $Q_{H'}$ – multimea starilor: $Q_{H'} = \{E,F,G\}$
2. $\Sigma_{H'}$ - multimea evenimentelor: $\Sigma_{H'} = \{a,b,c,d,e\}$
3. δ_H - functia de tranzitie:

$\delta_{H'}(E,a) = E$

$\delta_{H'}(E,e) = E$

$\delta_{H'}(E,b) = F$

$\delta_{H'}(F,a) = F$

$\delta_{H'}(F,e) = F$

$\delta_{H'}(F,c) = G$

$\delta_{H'}(G,a) = G$

$\delta_{H'}(G,e) = G$

$\delta_{H'}(G,d) = E$

4. $i_{H'}$ – starea initiala: $i_{H'}=E$
5. $M_{H'}$ – multimea starilor marcate: $M_{H'} = \{E,F,G\}$

In figura 3.5 se prezinta automatul *H'* obtinut prin adaugarea in automatul *H* a evenimentelor *a* si *e*.

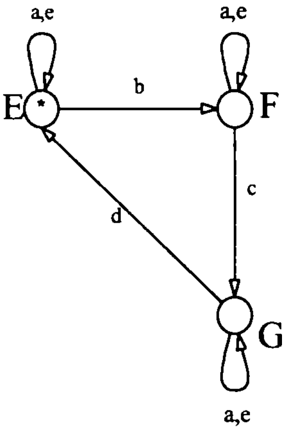


Figura 3.5. Structura automatului H'

Pasul 3. Deoarece exista un singur automat H nu mai este necesara executia acestui pas.

Pasul 4. *Se construieste* $E = G \cap H'$

Automatul rezultat este:

$$E = (Q_E, \Sigma_E, \delta_E, i_E, M_E)$$

unde:

1. $Q_{E'}$ – multimea starilor:

$$Q_E = \{S0, S1, S2, S3, S4, S5, S6, S7\}$$

unde:

S0 = A.C.E; S1=B.C.E; S2=A.C.F; S3=B.C.F; S4=A.D.G; S5=B.D.G; S6=A.C.G;
S7=B.C.G

2. Σ_E - multimea evenimentelor: $\Sigma_E = \{a,b,c,d,e\}$
3. δ_E - functia de tranzitie:

$$\delta_E(S0,a) = S1$$
$$\delta_E(S1,b) = S2$$
$$\delta_E(S2,a) = S3$$
$$\delta_E(S2,c) = S4$$
$$\delta_E(S3,c) = S5$$
$$\delta_E(S4,a) = S5$$
$$\delta_E(S4,e) = S6$$
$$\delta_E(S4,d) = S0$$
$$\delta_E(S5,d) = S1$$
$$\delta_E(S5,e) = S7$$
$$\delta_E(S6,a) = S7$$

4. $i_{E'}$ – starea initiala: $i_{E'}=S0$

5. $M_{E'}$ - multimea starilor marcate: $M_E = \{S0,S1,S2,S3,S4,S5,S6,S7\}$

Structura lui *E* este prezentata in figura 3.6.

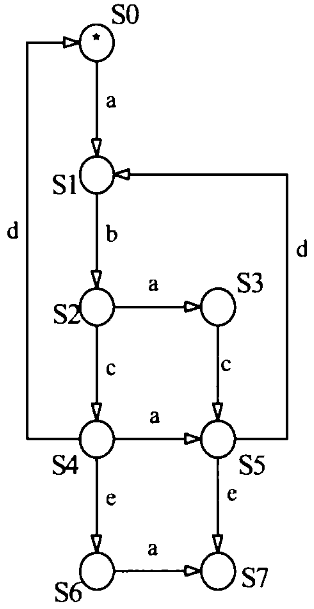


Figura 3.6. Structura automatului $E = G \cap H'$

Sistemul *E* rezultat nu este un supervizor deoarece contine o stare blocanta (S7), precum si o stare necontrolabile (S5). Deoarece evenimentul *b* a fost declarat ca necontrolabil rezulta ca limbajul generat de *E* este necontrolabil. Solutia pentru ca sistemul *E* sa devina supervizor este aceea de a sterge toate starile blocante si necontrolabile, obtinand astfel un automat S de urmatoarea structura.

$$S = (Q_S, \Sigma_S, \delta_S, i_S, M_S)$$

unde:

1. $Q_{S'}$ - multimea starilor: $Q_S = \{S0,S1,S2,S4\}$
2. Σ_S - multimea evenimentelor: $\Sigma_S = \{a,b,c,d\}$
3. δ_H - functia de tranzitie:

$$\delta_S(S0,a) = S1$$
$$\delta_S(S1,b) = S2$$
$$\delta_S(S2,c) = S4$$
$$\delta_S(S4,d) = S0$$

4. $i_{S'}$ - starea initiala: $i_S = S0$
5. M_S - multimea starilor marcate: $M_S = \{S0,S1,S2,S4\}$

In figura 3.7 este prezentata structura supervizorului S.

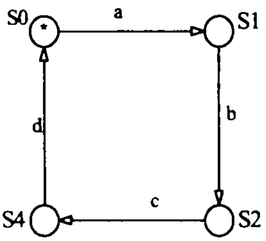


Figura 3.7. Structura supervizorului S

Algoritmul 3.2 *Implementarea supervizoarelor prin* ***metoda Cassandras***[CL01]
Se bazeaza pe modul de specificare prezentat anterior. Algoritmul propus este urmatorul:

- ***Pasul 1*** $S_0 = Ac(G \| S_P)$ i = 0
- ***Pasul 2*** $S_{i+1} = NonBlock(S_i)$;cautare pentru stari blocante si ;cele gasite ca fiind blocante se ;marcheaza ca interzise
- ***Pasul 3*** $S_{i+2} = Contrl(S_{i+1})$;cautare pentru stari ;necontrolabile si cela gasite ca ;necontrolabile se marcheaza ;ca interzise
- ***Pasul 4*** Daca $S_{i+1} \neq S_{i+2}$ atunci
  i=i+2 si trece la pasul 2
- ***Pasul 5*** $S = S_{i+1}$

Rezultatul obtinut (S) este nonblocant, controlabil, corespunzator specificatiilor date prin S_P si este restrictiv minimal. S obtinut poate fi un automat gol, rezultand in acest caz fie ca este prea restrictiv, fie modelul adoptat pentru instalatie este incorect, fie specificarea modelului s-a facut incorect.

Algoritmul propus pentru determinarea starilor nonblocante ***NonBlock(S_i)*** porneste de la urmatoarele premize:

Q_A = setul starilor in A
M_A = setul starilor marcate
X_A = setul starilor interzise
$\delta_A(q,\sigma)$ = functia de tranzitie, dependenta de starea activa q si evenimentul σ

Se urmareste obtinerea:
Q_{CO} = setul starilor co-accesibile
Q_X = setul starilor blocante sau previzibil interzise

Algoritmul de obtinere al lui Q_{CO} si Q_X se bazeaza pe cautarea pentru toate starile o traiectorie care are ca finalitate o stare marcata, stari care sunt inregistrate ca fiind co-accesibile. Restul starilor fiind considerate blocante.

Algoritmul 3.3. Determinarea starilor blocante ale unui automat NonBlock(Si)[CL01]*

```
• Pasul 1   $Q_{CO} = M_A; Q_X = X_A; Q = 0$
• Pasul 2   $(\forall q \in Q_A \setminus (Q_{CO} \cup Q_X))$   if $\exists \sigma \in \Sigma_A : \delta(q,\sigma) \in Q_{CO}$ then $Q = Q \cup \{q\}$
                else (if $\forall \sigma \in \Sigma_A : \delta(q,\sigma) \in Q_X$ or $\delta(q,\sigma)$ nede
                    then $Q_X = Q_X \cup \{q\}$)
• Pasul 3   if $Q \neq 0$ then $Q_{CO} = Q_{CO} \cup Q$; $Q = 0$; trece la pasul 2
• Pasul 4   $X_A = Q_A \setminus Q_{CO}$
```

Exemplul 3.2 Utilizarea algoritmului *NonBlock*

Se considera automatul cu structura din figura 3.8. Se cere determinarea starilor blocante prin aplicarea algoritmului NonBlock.

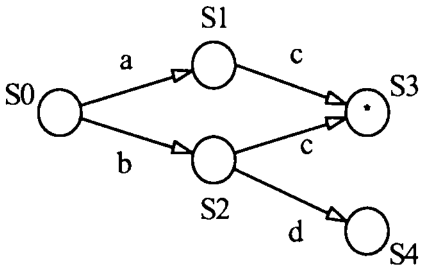


Figura 3.8. Structura automatului considerat la exemplul 3.2

Starea initiala este S0 iar starea S3 este stare marcata.
Pasii corespunzatori aplicarii algoritmului *NonBlock* sunt:

- ***Pasul 1*** $Q_{CO} = \{S3\},\ Q_X = 0$ *si* $Q = 0$
- ***Pasul 2.1*** $Q_A \setminus (Q_{CO} \cup Q_X) = \{S0, S1, S2, S4\}$

Verificare conditie if

$$\left.\begin{array}{l} \delta(S0,a) = S1 \\ \delta(S0,b) = S2 \\ \delta(S1,c) = S3 \\ \delta(S2,c) = S3 \\ \delta(S2,d) = S4 \end{array}\right\} \Rightarrow$$ *S1* si *S2* indeplinesc conditia *if* deci: $Q = \{S1, S2\}$

Verificare conditie else if

$$\left.\begin{array}{l}\delta(S0,a)=S1\\ \delta(S0,b)=S2\\ \delta(S1,c)=S3\\ \delta(S2,c)=S3\\ \delta(S2,d)=S4\\ \delta(S4,*)=nedefinit\end{array}\right\}\Rightarrow S4 \text{ indepliseste conditile } \textit{else if} \text{ deci: } Q_x=\{S4\}$$

- ***Pasul 3.1*** $Q_{CO} = Q_{CO} \cup Q = \{S1, S2, S3\}$ si $Q = 0$ si trece la pasul 2.2
- ***Pasul 2.2*** $Q_A \setminus (Q_{CO} \cup Q_X) = \{S0\}$

  Verificare conditie if

  $$\left.\begin{array}{l}\delta(S0,a)=S1\\ \delta(S0,b)=S2\end{array}\right\}\Rightarrow S0 \text{ indeplineste conditia } \textbf{if} \text{ deci: } Q=\{S0\}$$

  Verificarea conditiei else if nu mai este necesara
- ***Pasul 3.2*** $Q_{CO} = Q_{CO} \cup Q = \{S0, S1, S2, S3\}$ si $Q = 0$ si trece la pasul 2.3
- ***Pasul 2.3*** $Q_A \setminus (Q_{CO} \cup Q_X) = 0$ trece la pasul 3.3
- ***Pasul 3.3*** Trece la pasul 4
- ***Pasul 4*** $X_A = Q_X = \{S4\}$

Algoritmul propus pentru determinarea situatilor necontrolabile ***Contrl*** porneste de la urmatoarele premize:

Q_A = setul starilor in A

M_A = setul starilor marcate

X_A = setul starilor interzise

$\delta_A(q,\sigma)$ = functia de tranzitie, dependenta de starea activa q si evenimentul σ

Σ_{uc} = setul evenimentelor necontrolabile

Se urmareste obtinerea:

Q_X = setul starilor blocante sau previzibil interzise

Algoritmul se bazeaza pe determinarea tuturor tranzitilor necontrolabile la starile interzise.

Algoritmul 3.4. Determinarea tranzitilor necontrolabile Contrl[CL01]*

- **Pasul 1** $Q_X = X_A; Q = 0$
- **Pasul 2** $(\forall q \in Q_A \setminus Q_x)$ *if* $\exists \sigma_{uc} \in \Sigma_{uc} : \delta(q,\sigma_{uc}) \in Q_X$ *then* Q
- **Pasul 3** *if* $Q \neq 0$ *then* $Q_X = Q_X \cup Q; Q = 0;$ trece la pasul 2
- **Pasul 4** $X_A = Q_X$

Exemplul 3.3 Utilizarea algoritmului *Contrl*

Se considera automatul cu structura din figura 3.9. Se cere determinarea starilor necontrolabile prin aplicarea algoritmului *Contrl*.

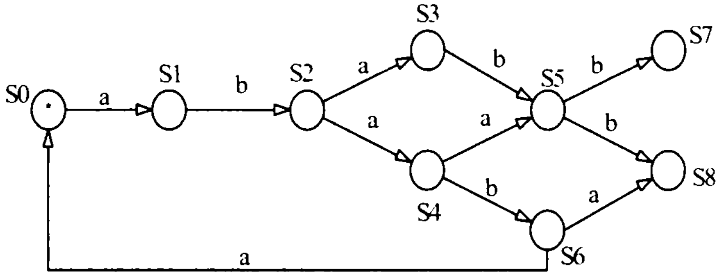


Figura 3.9. Structura automatului considerat la exemplul 3.3

Starea S0 este considerata stare initiala si marcata, iar starea S7 este considerata stare necontrolabila. Evenimentul *b* este considerat necontrolabil.
Pasii corespunzatori aplicarii algoritmului *Contrl* sunt:

- **Pasul 1** $Q_X = \{S7\}$ *si* $Q = 0$
- **Pasul 2.1** $Q_A \setminus Q_X = \{S0, S1, S2, S3, S4, S5, S6, S8\}$

Verificare conditie semnal necontrolabil:

$$\left.\begin{aligned} \delta(S0,b) &= nedefinit \\ \delta(S1,b) &= S2 \\ \delta(S2,b) &= nedefinit \\ \delta(S3,b) &= S5 \\ \delta(S4,b) &= S6 \\ \delta(S5,b) &= S7 \\ \delta(S6,b) &= nedefinit \\ \delta(S7,b) &= nedefinit \\ \delta(S8,b) &= nedefinit \end{aligned}\right\} \Rightarrow S5 \text{ indeplineste conditia } \textbf{if} \text{ deci: } Q = \{S5\}$$

- **Pasul 3.1** $Q_X = \{S5, S7\}$ si $Q = 0$ si trece la pasul 2.2
- **Pasul 2.2** $Q_A \setminus Q_X = \{S0, S1, S2, S3, S4, S6, S8\}$

Verificare conditie semnal necontrolabil

$$\left.\begin{aligned} \delta(S0,b) &= nedefinit \\ \delta(S1,b) &= S2 \\ \delta(S2,b) &= nedefinit \\ \delta(S3,b) &= S5 \\ \delta(S4,b) &= S6 \\ \delta(S5,b) &= S7 \\ \delta(S6,b) &= nedefinit \\ \delta(S7,b) &= nedefinit \\ \delta(S8,b) &= nedefinit \end{aligned}\right\} \Rightarrow S3 \text{ indeplineste conditia } \textbf{if} \text{ deci: } Q = \{S3\}$$

- **Pasul 3.2** $Q_X = \{S3, S5, S7\}$ si $Q = 0$ si trece la pasul 2.3
- **Pasul 2.3** $Q_A \setminus Q_X = \{S0, S1, S2, S4, S6, S8\}$

Verificare conditie semnal necontrolabil

$$\left.\begin{array}{l}\delta(S0,b) = nedefinit\\ \delta(S1,b) = S2\\ \delta(S2,b) = nedefinit\\ \delta(S3,b) = S5\\ \delta(S4,b) = S6\\ \delta(S5,b) = S7\\ \delta(S6,b) = nedefinit\\ \delta(S7,b) = nedefinit\\ \delta(S8,b) = nedefinit\end{array}\right\} \Rightarrow \text{ nu este indeplinita conditia } \textbf{if} \text{ deci: } Q = 0$$

- ***Pasul 3.3*** Q = 0 trece la pasul 4
- ***Pasul 4*** $X_A = Q_X = \{S3, S5, S7\}$

3.2.3. Studiu de caz. Conducerea supervizata pe baza modelelor de tip automat a unui sistem flexibil de fabricatie pentru o filatura [UM06] [Ung04a][Ung04b][Ung05]

3.2.3.1. Prezentare generală a unui sistem flexibil de fabricatie pentru o filatura (SFFF)

SFFF-ul considerat este constituit din mai multe tipuri de masini conectate intre ele printr-un sistem de transport. Structura de principiu a unui astfel de sistem este prezentata in figura 3.10.

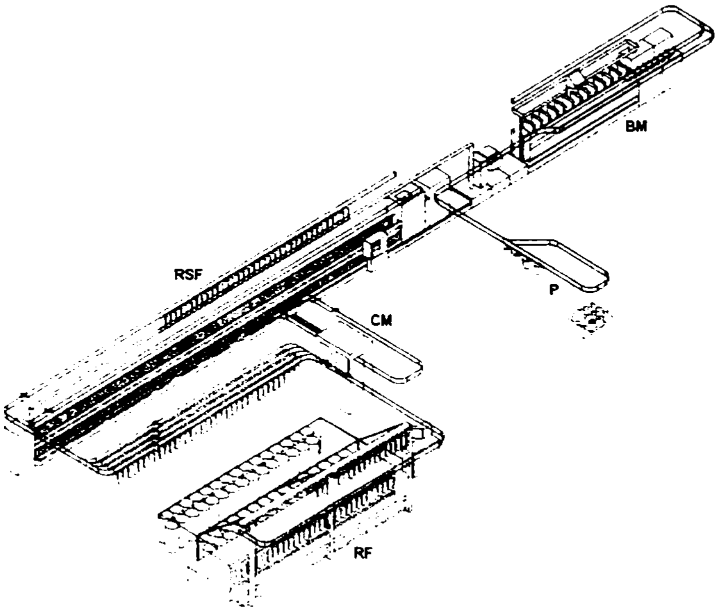


Figura 3.10. Structura de principiu al unui sistem flexibil de fabricatie dintr-o filatura. RF – Roving Frame – grup de masini pentru formarea firului necesar filarii finale, RSF – Ring Spinning Frame – grup de masini pentru filarea finala a firului textil, BM – Bobbin Macker - masina pentru asigurarea continuitatii si duritatii firului textil obtinut la RSF, CM – masina de curatat bobine, P – Packing - zona de impachetare

In figura 3.10 se observa elementele componente ale sistemului:RF (Roving Frame) - nivelul la care firul este pregatit pentru subtierea finala (vezi figura 3.11).

***Observatie**: fotografiile 3.12, 3.13, 3.14, 3.15, 3.16, 3.17 sunt realizate de catre autor in fabricile Albarekeh si Oulabitex din Siria.*

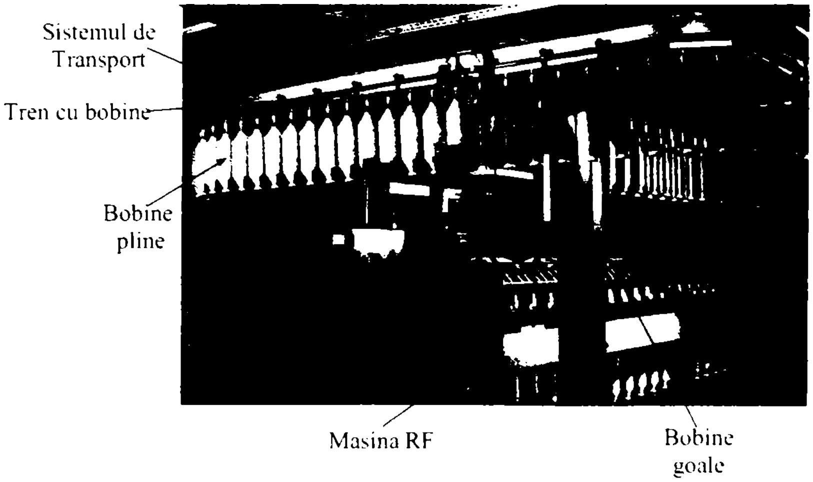


Figura 3.11. Roving Frame (RF)

RSF (Ring Spinning Frame) - nivelul la care firul textil este adus la dimensiunea (grosimea) dorita (vezi figurile 3.12 si 3.13),

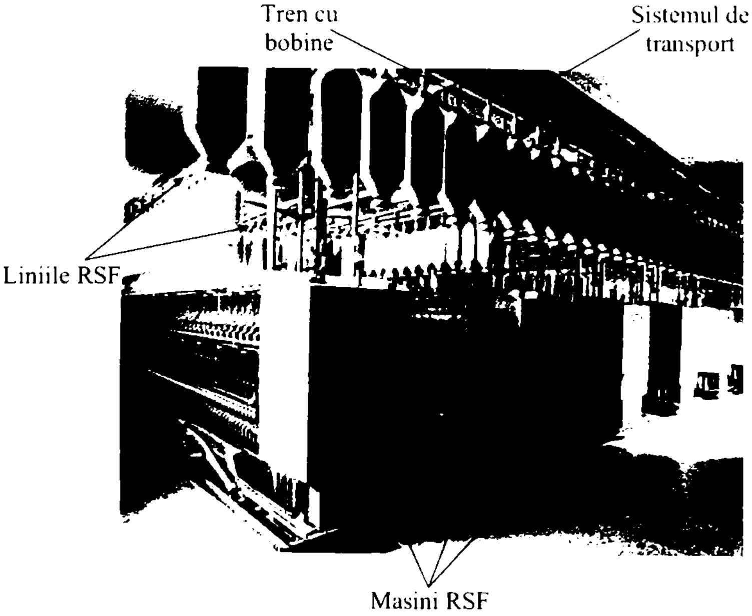


Figura 3.12. Imagine cu Ring Spinning Frame

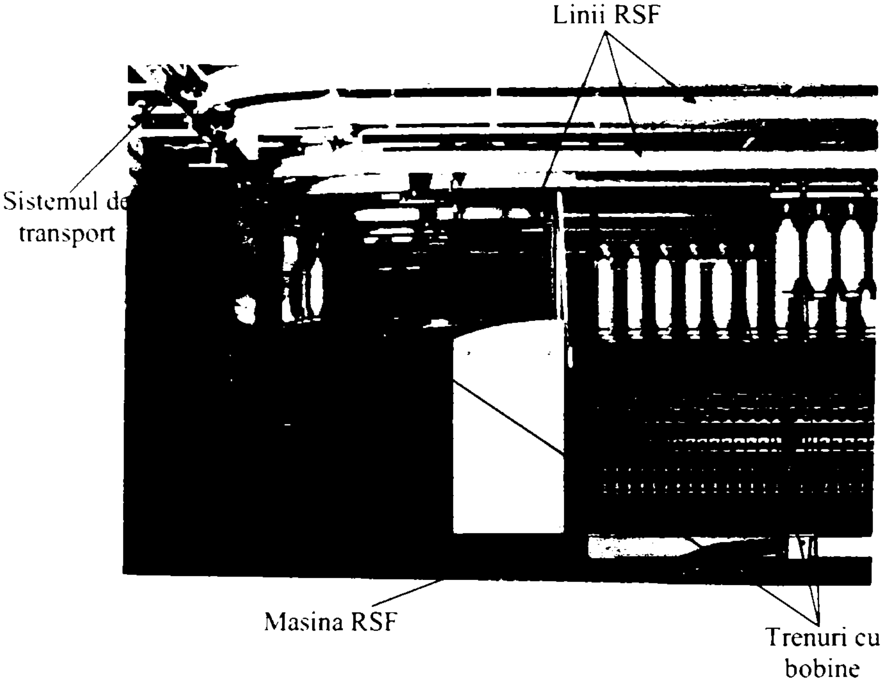


Figura 3.13. Imagine cu Ring Spinning Frame

In CM (Cleaning Machine) bobinele „murdare" provenite de la RSF sunt curatate si pregatite pentru RF. In figura 3.14 se prezinta o imagine a unei CM.

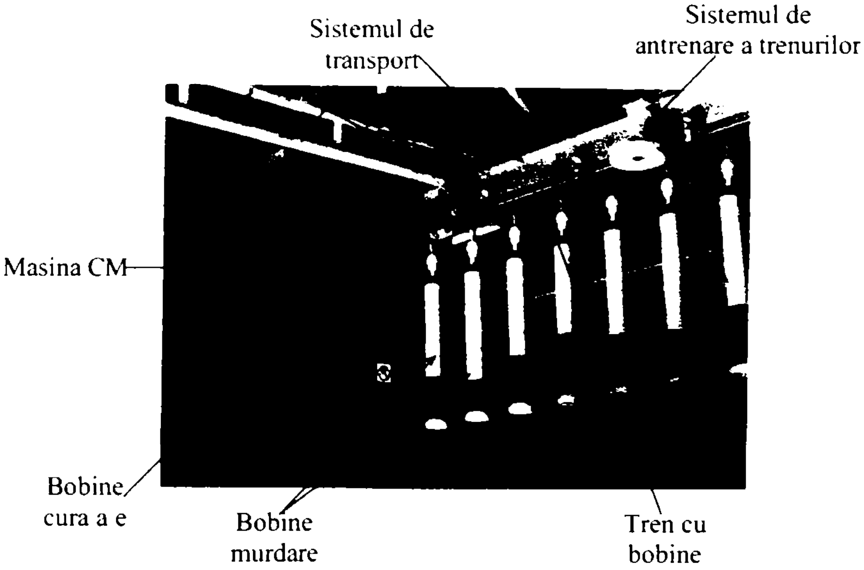


Figura 3.14. Imagine cu Cleaning Machine CM (masina de curatat)

In figura 3.10 mai sunt reprezentate subsistmele BM (Bobbin Macker) prin intermediul carora se asigura continuitatea si duritatea firului si P (Packing) zona de impachetare si expediere a produsului.

Intre aceste masini (subsisteme) exista un sistem de transport care asigura transferul materialului intre subsisteme. In figurile 3.15 si 3.16 se prezinta doua imagini cu sistemul de transport corespunzator.

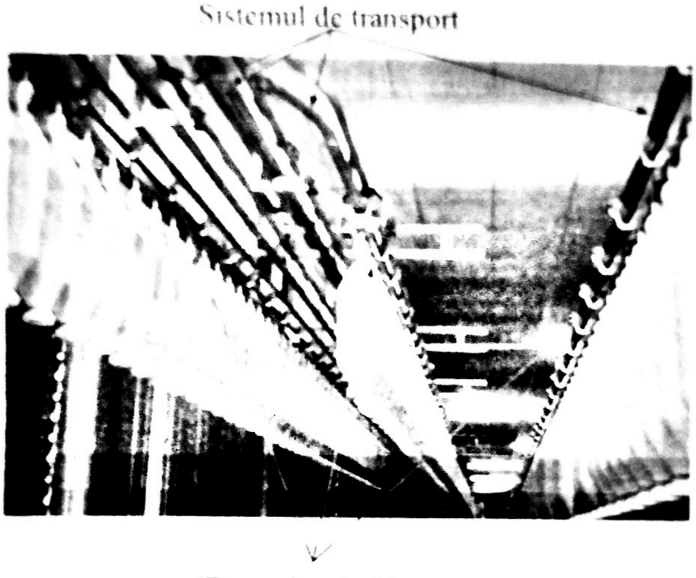


Figura 3.15. Imagine cu Sistemul de Transport

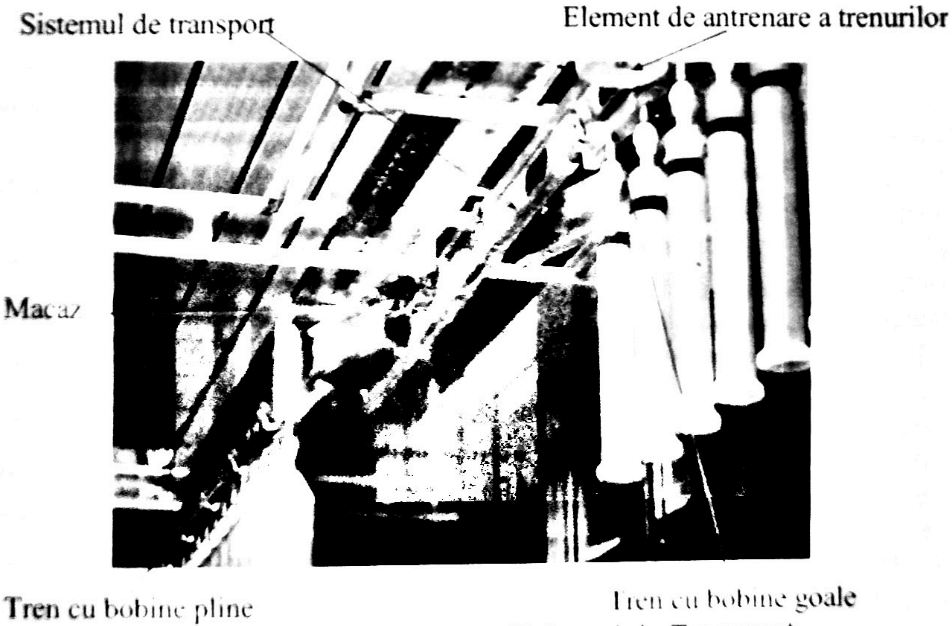


Figura 3.16. Imagine cu Sistemul de Transport

Elementul transportat este trenul (vezi figurile de mai sus)care contine un numar fix de bobine (140). Lungimea acestuia fiind fixa (24 m).

3.2.3.2. Schema bloc a SFFF-ului. Definirea subsistemelor SFFF-ului [UM06]

In figura 3.17 se prezinta schema bloc a SFFF-ului considerat.

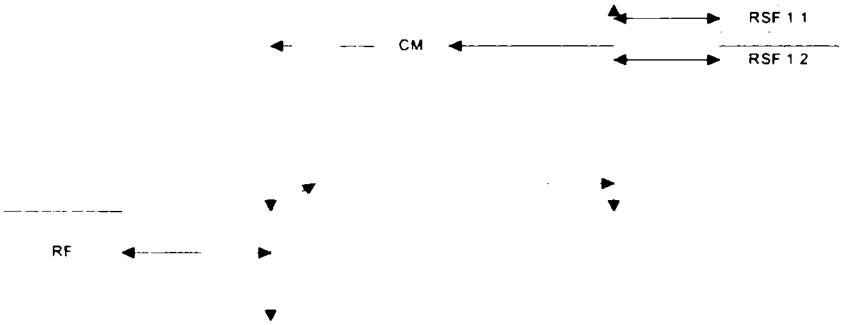


Figura 3.17. Schema bloc a SFFF-ului considerat

Sistemul are in componenta sa:

- o masina RF;
- o masina CM;
- o masina RSF;
- un sistem de transport care asigura transferul intre subsisteme.

Functionarea normala se bazeaza pe producerea de bobine cu fir textil gros in subsistemul RF, transferul acestor bobine catre RSF (prin intermediul sistemului de transport), unde prin prelucrare rezulta firul de dimensiune dorita. Dupa prelucrare in RSF rezulta bobine „murdare" care sunt transportate catre CM, unde sunt curatate si transferate catre RF, ciclul incheindu-se.

Functionarea fiecarui subsistem (RSF,CM si RF) este independenta, dar conectata direct cu sistemul de transport. In cazul in care sistemul de transport este nefunctional, intreg SFFF-ul este blocat, nefiind posibila alimentarea RSF cu material, ceea ce duce la blocarea completa a sistemului.

Problemele care trebuie solutionate pentru SFFF-ul considerat sunt:

1. Asigurarea incarcarii unui tren cu bobine pline produse de RF;
2. Transferul trenului incarcat in RF catre un RSF destinatie;
3. Preluarea trenului de catre RSF-ul corespunzator;
4. Transferul trenurilor „murdare" dinspre RSF catre CM;
5. Curatirea trenurilor „murdare" de catre CM;
6. Transferul trenurilor „curate" catre RF-ul corespunzator.

In cadrul acestei aplicatii nu intereseaza modul de functionare efectiva a masinilor considerate ci numai modul de interfatare cu sistemul de transport.

Restrictiile care sunt impuse SFFF-ului considerat sunt:

1. Fiecare masina RSF are doua linii de lucru care pot prelucra aceeasi calitate a materialului;
2. In sistem pot exista una sau mai multe masini RSF care lucreaza cu aceeasi calitate;
3. O masina RF poate produce la un moment dat o singura calitate;
4. In sistem pot exista una sau mai multe masini RF care produc aceeasi calitate.
5. O masina CM poate curata mai multe trenuri cu calitati diferite, dar un singur tren la un moment dat.

6. Sistemul de transport poate transporta unul sau mai multe trenuri simultan.

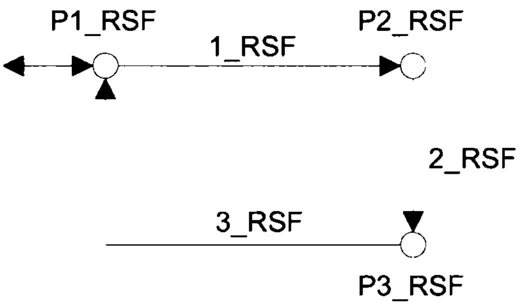
Pentru sistemul considerat, modul de sinteza a supervizorului nu tine cont de calitate.

Pornind de cele prezentate supervizorul care se doreste a fi elaborat trebuie sa asigure gestionarea traficului, corespunzator sistemului de transport, pentru evitarea coliziunilor dintre trenuri.

Pentru implementare sa utilizat algoritmul 3.2 (algoritmul Cassandras) de sinteza al supervizoarelor.

- Automatul unei linii RSF

In figura 3.18 se prezinta automatul unei linii corespunzatoare unei masinii RSF.



P1_RSF		starea de asteptare a unui tren
P2_RSF		tren sosit si aflat in procesare
P3_RSF		tren procesat si transportat la iesire
1_RSF	:	trenul venit a intrat complet in linia RSF
2_RSF		tren procesat gata de miscare spre iesire
3_RSF		tren iesit din linia RSF

Figura 3.18. Automatul corespunzator functionarii unei linii RSF

Automatul unei linii RSF se defineste ca fiind:

$$RSF = (Q_{RSF}, \Sigma_{RSF}, \delta_{RSF}, i_{RSF}, M_{RSF})$$

unde:

1. Q_{RSF} - *multimea Starilor:* $Q_{RSF} = \{P1\_RSF, P2\_RSF, P3\_RSF\}$
2. Σ_{RSF} - *multimea evenimentelor:* $\Sigma_{RSF} = \{1\_RSF, 2\_RSF, 3\_RSF\}$
3. δ_{RSF} - *functia de tranzitie:*

$$\delta_{RSF}(P1\_RSF, 1\_RSF) = P2\_RSF$$

$$\delta_{RSF}(P2\_RSF, P2\_RSF) = P3\_RSF$$

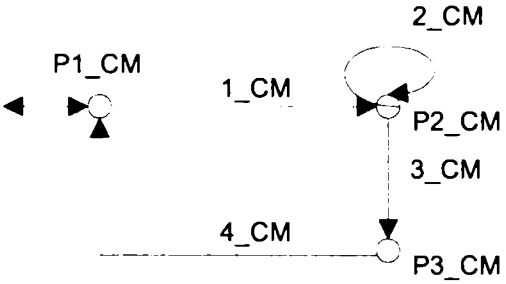
$$\delta_{RSF}(P3\_RSF, 3\_RSF) = P1\_RSF$$

4.i_{RSF} - *starea initiala:* $i_{RSF} = P1\_RSF$

5. M_{RSF} - *multimea starilor marcate:* $M_{RSF} = \{P1\_RSF, P2\_RSF, P3\_RSF\}$

- Automatul masinii CM

In figura 3.19 se prezinta automatul corespunzator unei masini CM.



P1_CM :	Starea de asteptare a unui tren
P2_CM	tren sosit si aflat in procesare (curatire)
P3_CM :	tren procesat (curat) si transportat la iesire
1_CM :	trenul venit a intrat complet in masina de curatat
2_CM	tren aflat in procesul de curatare
3_CM	tren curatat
4_CM	tren iesit complet din CM

Figura 3.19. Automatul corespunzator functionarii unei masini CM

Automatul masinii CM se defineste ca fiind:

$$CM = (Q_{CM}, \Sigma_{CM}, \delta_{CM}, i_{CM}, M_{CM})$$

unde:

1. Q_{CM} - *multimea Starilor:* $Q_{CM} = \{P1\_CM, P2\_CM, P3\_CM\}$
2. Σ_{CM} - *multimea evenimentelor:* $\Sigma_{CM} = \{1\_CM, 2\_CM, 3\_CM, 4\_CM\}$
3. δ_{CM} - *functia de tranzitie:*

$$\delta_{CM}(P1\_CM, 1\_CM) = P2\_CM$$

$$\delta_{CM}(P2\_CM, 2\_CM) = P2\_CM$$

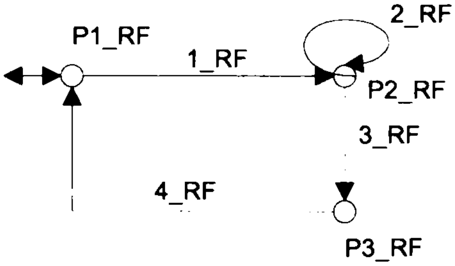
$$\delta_{CM}(P2\_CM, 3\_CM) = P3\_CM$$

$$\delta_{CM}(P3\_CM, 4\_CM) = P1\_CM$$

4. i_{CM} - *starea initiala:* $i_{CM} = P1\_CM$
5. M_{CM} - *multimea starilor marcate:* $M_{CM} = \{P1\_CM, P2\_CM, P3\_CM\}$

- Automatul masinii RF

In figura 3.20 se prezinta automatul corespunzator unei masini RF.



P1_RF :	Starea de asteptare a unui tren
P2_RF :	tren sosit si aflat in procesare (incarcare cu bobine pline)
P3_RF :	tren procesat (incarcat) si transportat la iesire
1_RF :	trenul venit a intrat complet in linia RF
2_RF :	tren in procesare (incarcare)
3_RF :	tren procesat (incarcat) gata de iesire
4_RF	tren iesit complet din RF

Figura 3.20. Automatul corespunzator functionarii unei masini RF

Automatul masinii RF este:

$$RF = (Q_{RF}, \Sigma_{RF}, \delta_{RF}, i_{RF}, M_{RF})$$

unde:

1.Q_{RF} - *multimea Starilor:* $Q_{RF} = \{P1\_RF, P2\_RF, P3\_RF\}$

2. Σ_{RF} - *multimea evenimentelor:* $\Sigma_{RF} = \{1\_RF, 2\_RF, 3\_RF, 4\_RF\}$

3. δ_{RF} - *functia de tranzitie*

$$\delta_{RF}(P1\_RF, 1\_RF) = P2\_RF$$

$$\delta_{RF}(P2\_RF, 2\_RF) = P2\_RF$$

$$\delta_{RF}(P2\_RF, 3\_RF) = P3\_RF$$

$$\delta_{RF}(P3\_RF, 4\_RF) = P1\_RF$$

4.i_{RF} - *starea initiala:* $i_{RF} = P1\_RF$

5.M_{RF} - *multimea starilor marcate:* $M_{RF} = \{P1\_RF, P2\_RF, P3\_RF\}$

- Automatul sistemului de transport TR

In figura 3.21 se prezinta automatul corespunzator sistemului de transport .

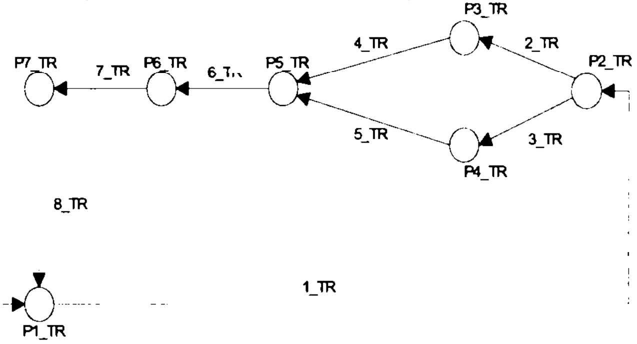


Figura 3.21. Automatul corespunzator functionarii sistemului de transport

Unde:

P1_TR	stare de asteptare a unui tren din RF (trenul este in RF)
P2_TR	tren in transport catre RSF1 sau RSF2
P3_TR	starea de asteptare a unui tren din prima linie RSF (tren in RSF1)
P4_TR	starea de asteptare a unui tren din a doua linie RSF (tren in RSF1)
P5_TR	tren in transport catre CM. Tren venit din RSF1 sau RSF2
P6_TR	starea de asteptare a unui tren din CM (trenul este in CM)
P7_TR	tren in transport din CM catre RF
1_TR	operatia de incrcare in RF gata
2_TR	tren ajuns in linia RSF1
3_TR	tren ajuns in linia RSF2
4_TR	trenul plecat din RSF1 a ajuns in CM
5_TR	trenul plecat dun RSF2 a ajuns in CM

6_TR : tren ajuns in CM
7_TR : operatia de curatare in CM gata
8_TR : tren ajuns in RF

Automatul sistemului de transport este:

$$TR = (Q_{TR}, \Sigma_{TR}, \delta_{TR}, i_{TR}, M_{TR})$$

unde:

1.Q_{TR} *- multimea Starilor:*

$$Q_{TR} = \{P1\_TR, P2\_TR, P3\_TR, P3\_TR, P5\_TR, P6\_TR, P7\_TR\}$$

2. Σ_{TR} *- multimea evenimentelor:*

$$\Sigma_{TR} = \{1\_TR, 2\_TR, 3\_TR, 4\_TR, 5\_TR, 6\_TR, 7\_TR, 8\_TR\}$$

3. δ_{TR} *- functia de tranzitie*

$$\delta_{RF}(P1\_TR, 1\_TR) = P2\_TR$$
$$\delta_{RF}(P2\_TR, 2\_TR) = P3\_TR$$
$$\delta_{RF}(P2\_TR, 3\_TR) = P4\_TR$$
$$\delta_{RF}(P3\_TR, 4\_TR) = P5\_TR$$
$$\delta_{RF}(P4\_TR, 5\_TR) = P5\_TR$$
$$\delta_{RF}(P5\_TR, 6\_TR) = P6\_TR$$
$$\delta_{RF}(P6\_TR, 7\_TR) = P7\_TR$$
$$\delta_{RF}(P7\_TR, 8\_TR) = P1\_TR$$

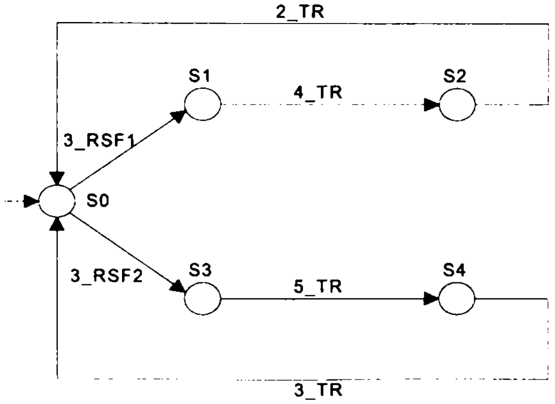
4.i_{TR} *- starea initiala:* $i_{TR} = P1\_TR$

5.M_{TR} *- multimea starilor marcate:*

$$M_{TR} = \{P1\_TR, P2\_TR, P3\_TR, P4\_TR, T5\_TR, P6\_TR, P7\_TR\}$$

- Specificatile SFFF

In figura 3.22 se prezinta automatul care sintetizeaza specificatile impuse SFFF-ului.



S0 : stare de asteptare a unui tren din RSF1 sau RSF2
S1 : tren in transport din RSF1. RSF2 nu poate scoate tren
S2 : starea de asteptare a unui tren din RF pentru RSF1
S3 : tren in transport din RSF2. RSF1 nu poate scoate tren
S4 : starea de asteptare a unui tren din RF pentru RSF1

Figura 3.22. Automatul corespunzator functionarii sistemului de transport

Automatul sistemului de transport este:

$$S = (Q_S, \Sigma_S, \delta_S, i_S, M_S)$$

unde:

1.Q_S - *multimea Starilor:* $Q_S = \{S0, S1, S2, S3, S4\}$

2. Σ_S - *multimea evenimentelor:* $\Sigma_S = \{3\_RSF1, 4\_TR, 2\_TR, 3\_RSF2, 5\_TR, 3\_TR\}$

3. δ_S - *functia de tranzitie*

$$\delta_S(S0, 3\_RSF1) = S1$$
$$\delta_S(S1, 4\_TR) = S2$$
$$\delta_S(S2, 2\_TR) = S0$$
$$\delta_S(S0, 3\_RSF2) = S3$$
$$\delta_S(S3, 5\_TR) = S4$$
$$\delta_S(S3, 3\_TR) = S0$$

4.i_S - *starea initiala:* $i_S = S0$

5.M_S - *multimea starilor marcate:* $M_S = \{S0, S1, S2, S3, S4\}$

3.2.3.3. Sinteza Supervizorului

Sinteza supervizorului dorit in cazul SFFF-ului considerat se bazeaza la primul pas pe compunerea paralela a tuturor automatelor definite mai sus. Astfel:

$$G_{SFFF} = RSF1 \| RSF2 \| CM \| RF \| TR \,||\, S$$

dupa care supervizorul se obtine combinand G_{SFFF} cu specificatile impuse sistemului si utilizand algoritmul 1 de verificare a supervizorului (vezi capitolul 4) se valideaza solutia obtinuta.

Pentru compunerea paralela se utilizeaza proprietatea de asociativitate a operatiei (vezi paragraful 2.1.6.2.2). Astfel sinteza automatelor se face in mai multi pasi.

Pasul 1 *Compunerea paralela RSF1 || RSF2*

In cazul compunerii paralele intre RSF1 si RSF2 rezulta automatul:

$$RSF1||RSF2 = RSF12 =< Q_{RSF12}, \Sigma_{RSF12}, \delta_{RSF12}, i_{RSF12}, M_{RSF12} >$$

Pasul 2 *Compunerea paralela RSF1 || RSF2 || CM*

Compunerea paralela *RSF1||RSF2||CM* se reduce la compunerea automatelor *RSF12 cu CM.*

$$RSF12\,||CM = RSF\_CM =< Q_{RSF\_CM}, \Sigma_{RSF\_CM}, \delta_{RSF\_CM}, i_{RSF\_CM}, M_{RSF\_CM} >$$

Pasul 3 *Compunerea paralela RSF1 || RSF2 || CM || RF*

Compunerea paralela *RSF1||RSF2||CM||RF* se reduce la compunerea automatelor *RSF_CM cu RF.*

$$RSF\_CM\,||RF = RSF\_CM\_RF =< Q_{RSF\_CM\_RF}, \Sigma_{RSF\_CM\_RF}, \delta_{RSF\_CM\_RF}, i_{RSF\_CM\_RF}, M_{RSF\_CM\_RF} >$$

Pasul 4 *Compunerea paralela RSF1 || RSF2 || CM || RF || TR*

Compunerea paralela *RSF1||RSF2||CM||RF||TR* se reduce la compunerea automatelor *RSF_CM_RF* cu *TR*.

$$RSF\_CM\_RF \parallel TR = RSF\_CM\_RF\_TR =$$

$$(Q_{RSF\_CM\_RF\_TR}, \Sigma_{RSF\_CM\_RF\_TR}, \delta_{RSF\_CM\_RF\_TR},$$
$$i_{RSF\_CM\_RF\_TR}, M_{RSF\_CM\_RF\_TR})$$

Pasul 5 *Compunerea paralela RSF1 || RSF2 || CM || RF || TR || S*

Compunerea paralela *RSF1||RSF2||CM||RF||TR||S* se reduce la compunerea automatelor *RSF_CM_RF_ TR cu S.*

In cazul compunerii paralele intre RSF_CM_RF_TR si S rezulta automatul:

$$RSF\_CM\_RF\_TR \parallel S = G_{SFFF} =$$

$$(Q_{RSF\_CM\_RF\_TR||S}, \Sigma_{RSF\_CM\_RF\_TR||S}, \delta_{RSF\_CM\_RF\_TR||S},$$
$$i_{RSF\_CM\_RF\_TR}.i_S, M_{RSF\_CM\_RF\_TR||S})$$

unde:

1. $Q_{RSF\_CM\_RF\_TR||S}$ - *multimea Starilor*

$$Q_{RSF\_CM\_RF\_TR||S} = \cup_{k=1,637}\{Pk_{RSF\_CM\_RF||TR}.S0, Pk_{RSF\_CM\_RF||TR}.S1,$$
$$Pk_{RSF\_CM\_RF||TR}.S2, Pk_{RSF\_CM\_RF||TR}.S3,$$
$$Pk_{RSF\_CM\_RF||TR}.S4\}$$

2. $\Sigma_{RSF\_CM\_RF\_TR||S}$ - *multimea evenimentelor*

$$\Sigma_{RSF\_CM\_RF\_TR||S} = \{1\_RSF1, 2\_RSF1, 3\_RSF1, 1\_RSF2, 2\_RSF, 3\_RSF3,$$
$$1\_CM, 2\_CM, 3\_CM, 4\_CM, 1\_RF, 2\_RF, 3\_RF, 4\_RF,$$
$$1\_TR, 2\_TR, 3\_TR, 4\_TR, 5\_TR, 6\_TR, 7\_TR\}$$

3. $\delta_{RSF\_CM\_RF\_TR||S}$ - *functia de tranzitie*

Se considera: $s \in \Sigma_{RSF\_CM\_RF\_TR||S}$ pentru care se defineste functia de tranzitie corespunzatoare starilor din RSF_CM_RF_TR astfel:

$$\forall s \in \Sigma_{RSF\_CM\_RF\_TR} \quad si \ i = 0,4 \quad avem:$$

$$\delta_{RSF\_CM\_RF\_TR||S}(P_k\_RSF\_CM\_RF\_TR.Si\_TR, s) =$$
$$= \delta_{RSF\_CM\_RF\_TR}(P_k\_RSF\_CM\_RF, s).\delta_S(Si, s)$$

4. $i_{RSF\_CM\_RF\_TR||S}$ - *starea initiala:*

$$i_{RSF\_CM\_RF\_TR}.i_S = P1\_RSF\_CM\_RF\_TR.S0$$

5. $M_{RSF\_CM\_RF\_TR||S}$ - *multimea starilor marcate*

$$M_{RSF\_CM\_RF\_TR||S} = \cup_{k=1,637}\{Pk_{RSF\_CM\_RF||TR}.S0, Pk_{RSF\_CM\_RF||TR}.S1,$$
$$Pk_{RSF\_CM\_RF||TR}.S2, Pk_{RSF\_CM\_RF||TR}.S3,$$
$$Pk_{RSF\_CM\_RF||TR}.S4\}$$

Pentru a obtine supervizorul dorit trebuie eliminate starile blocante sau necontrolabile din sistem. Deoarece in acest caz nu exista semnale necontrolabile in sistem nu exista posibilitatea de a avea stari necontrolabile.

In privinta blocajului insa, acesta trebuie verificat cu ajutorul algoritmului NonBlock, rezultand astfel starile care trebuie eliminate din cadrul lui S.

Dupa verificare si eliminarea starilor si tranzitiilor corespunzatoare, sistemul S va reprezenta supervizorul cautat.

Datorita faptului ca toate starile sunt marcate, limbajele generate si marcate ale supervizorului sunt identice, fapt care asigura nonblocajul sistemului precum si controlabilitatea acestuia.

3.3. Conducerea supervizata a SED pe baza modelelor de tip retele Petri

Datorita faptului ca implementarea supervizoarelor in cazul utilizarii modelelor de tip automat este, dupa cum s-a aratat in paragraful 3.2, extrem de laborioasa, modelul final avand dimensiuni foarte mari, ideea de a introduce conducerea supervizata si asupra modelelor de tip retea Petri a fost abordata in numeroase lucrari [Mood99][IMA99] [IMA01] [Giua92] [PC95] [CL01]. Prin utilizarea retelelor Petri, pentru modelul final se poate ajunge dimensiuni acceptabile datorita topologiei utilizate prin extinderea marcajelor si nu prin extinderea retelei (cazul automatelor). Pornind de la aceasta observatie se poate concluziona ca si pentru supervizorul obtinut pentru un sistem dat avem dimensiuni acceptabile.

Datorita generalitatii teoremei supervizorului al lui Wohnam [RW88], aceasta poate fi aplicata cu succes si in cazul retelelor Petri, cu conditia ca pentru acestea sa poata fi definite limbajele generate si marcate corespunzatoare [WR87][CL01].

Pornind de la aceasta observatie, tot ceea ce s-a prezentat in cazul modelelor de tip automat (toate referirile facute pentru limbaje) sunt valabile si in cazul modelelor de tip retele Petri.

In continuare abordarea conducerii supervizate in cazul modelelor de tip retele Petri este prezentata prin trei metode distincte:

1) prin utilizarea limbajelor
2) prin utilizarea modelelor acoperite de invarianti de tip P
3) prin utilizarea compunerii paralele a sistemelor.

3.3.1. Conducerea supervizata utilizand modele de tip retele Petri bazate pe invarianti de tip P [IA02a][IA02c] [MA97][Mood99]

Metoda de implementare a unui supervizor corespunzator unei retele Petri a fost prezntata prima data de Moody. si dezvoltata ulterior in numeroase lucrari [Mood99][IMA99] [IMA01] [Giua92] [PC95] [CL01][IA02a][IA02b][IA02c][IA02d] [IMA02] .

In cadrul metodei se urmareste determinarea structurii unei retele Petri cu reactie, bazata pe utilizarea invariantilor de tip P. Acest procedeu, pornind de la o serie de specificatii impuse functionarii sistemului, va determina numarul de pozitii suplimentare, cu marcajul initial corespunzator, si modul de conectare al acestora la reteaua considerata ininitial. Metoda a fost dezvoltata pentru retele Petri netemporizate, dar ea nu este afectata de introducerea timpului. Supervizorul rezultat va tine cont de specificatiile impuse, care pot fi descrise prin introducerea starilor nepermise, a excluderii mutuale etc.

Un sistem modelat prin intermediul retelei Petri netemporizate $PN = (P,T,F,W,X_m)$, contine n pozitii si m tranzitii si este caracterizata prin intermediul matricii de incidenta A_P . Reteaua Petri de tip controler (supervizor) este descrisa prin intermediul matricii de incidenta A_C , care are acelasi numar de linii (in fapt sunt aceleasi tranzitii) ca si matricea A si un numar diferit de coloane (pozitii). Reteaua Petri supervizata (sistem plus controler), va fi descrisa prin intermediul matricii de incidenta A. Obiectivul vizat este de a forta sistemul sa satisfaca constrangeri de forma:

$$Lm_p^T \le b \tag{3.20}$$

unde m_p este marcajul sistemului, L este o matrice de tip $n_c \times n$ cu elemente de tip intreg, b este un vector $n_c \times 1$, n_c numarul constrangerilor impuse sistemului.

Inegalitatea 3.20 generata de introducerea constrangerilor este transformata in egalitate prin introducerea unui controler de tip retea Petri extern, care contine pozitii reprezentate prin asa numitele *variabile slabe*. Astfel:

$$L \cdot m_p^T + m_c^T = b \tag{3.21}$$

unde m_c este un vector cu n_c linii si o coloana si reprezinta marcajul starii controlabile.

Matricea de incidenta Ac contine conexiunile (arcurile) dintre pozitiile introduse de controler si tranzitiile din modelul sistemului. Matricea de incidenta a sistemului in bucla inchisa este:

$$A = \lfloor A_P \;\; A_C \rfloor \tag{3.22}$$

iar vectorul corespunzator marcajului este:

$$m = \lfloor m_p \;\; m_c \rfloor \; ; \; m_0 = \lfloor m_{p_0} \;\; m_{c_0} \rfloor . \tag{3.23}$$

Tinand cont de modul de definire al invariantilor de tip P, sistemul nou definit trebuie sa indeplineasca conditia:

$$X^T A = [L \; I][A_P \;\; A_C]^T = 0 \tag{3.24}$$

unde X este o matrice specifica al n_c-ulea invariant.

In aceste conditii:

$$LA_p^T + Ac^T = 0 \tag{3.25}$$

I fiind matricea identitate de tip $n_c x n_c$ coeficintii variabilelor slabe introduse fiind 1.

Rezulta deci:

$$A_C^T = -LA_P^T \tag{3.26}$$

si in final:

$$A_C = -A_P L^T \tag{3.27}$$

Teorema 3.4. Sinteza controlerului

Daca:

$$b - Lm_{P0}^T \ge 0 \tag{3.28}$$

atunci controlerul de tip retea Petri , caracterizat prin matricea de incidenta A_C cu marcajul initial m_{C0} , este

$$A_C = -A_P L^T$$
$$m_{C_0} = b^T - m_{P_0} L^T \tag{3.29}$$

Exemplul 3.4 Sinteza unui supervizor urilizand metoda Moody
Fie un sistem de transport caracterizat prin reteaua Petri din figura 3.23.

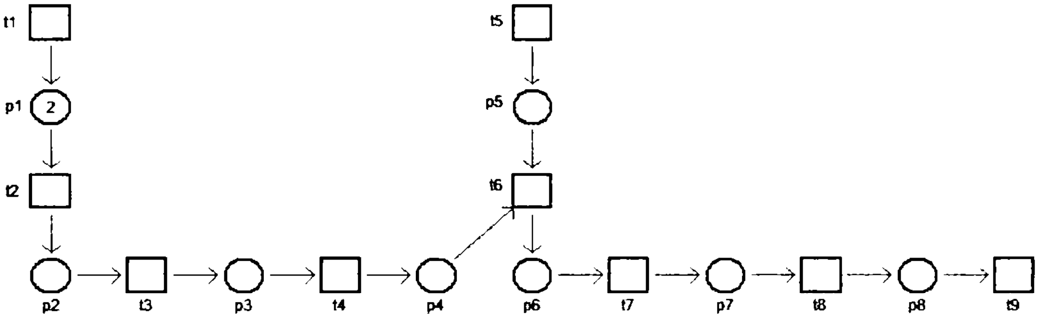


Figura 3.23. Structura retelei Petri corespunzatoare modelarii sistemului de transport considerat in exemplul 3.4

Pozitiile Pi corespund nodurilor din sistem, nodurile incluzand si jam-urile, motiv pentru care capacitatile sunt diferite de unu.
Restrictiile comportamentale ale sistemului sunt date prin urmatoarele specificatii:

$$m_2 + m_3 \le 3$$
$$m_5 + m_6 + m_7 + m_8 \le 3$$
$$m_5 + m_6 \le 2\,.$$

adica: in nodurile 2 si 3 nu pot fi cumulate mai mult de trei carucioare, in nodurile 5, 6, 7 si 8 pot fi maxim 3, iar in 5 si 6 pot fi simultan doua carucioare.

Aplicand algoritmul prezentat rezulta:

$$A_C = -\begin{bmatrix} 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ -1 & 1 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & -1 & 1 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & -1 & 1 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & -1 & -1 & 1 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & -1 & 1 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & -1 & 1 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & -1 \end{bmatrix} \cdot \begin{bmatrix} 0 & 0 & 0 \\ 1 & 0 & 0 \\ 1 & 0 & 0 \\ 0 & 0 & 0 \\ 0 & 1 & 1 \\ 0 & 1 & 1 \\ 0 & 1 & 0 \\ 0 & 1 & 0 \end{bmatrix} = \begin{bmatrix} 0 & 0 & 0 \\ -1 & 0 & 0 \\ 0 & 0 & 0 \\ 1 & 0 & 0 \\ 0 & -1 & -1 \\ 0 & 0 & 0 \\ 0 & 0 & 1 \\ 0 & 0 & 0 \\ 0 & 1 & 0 \end{bmatrix}$$

$$m_{C_0} = [3\ \ 3\ \ 2] - [2\ 0\ 0\ 0\ 0\ 0\ 0\ 0\ 0] \cdot \begin{bmatrix} 0 & 0 & 0 \\ 1 & 0 & 0 \\ 1 & 0 & 0 \\ 0 & 0 & 0 \\ 0 & 1 & 1 \\ 0 & 1 & 1 \\ 0 & 1 & 0 \\ 0 & 1 & 0 \end{bmatrix} = [3\ \ 3\ \ 2]$$

Pentru matricea de incidenta si marcajul inital se ajunge la.

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

$$m = [2\ 0\ 0\ 0\ 0\ 0\ 0\ 0\ 3\ 3\ 2]$$

Structura finala a retelei Petri corespunzatoare sistemului in bucla inchisa este prezentata in figura 3.24.

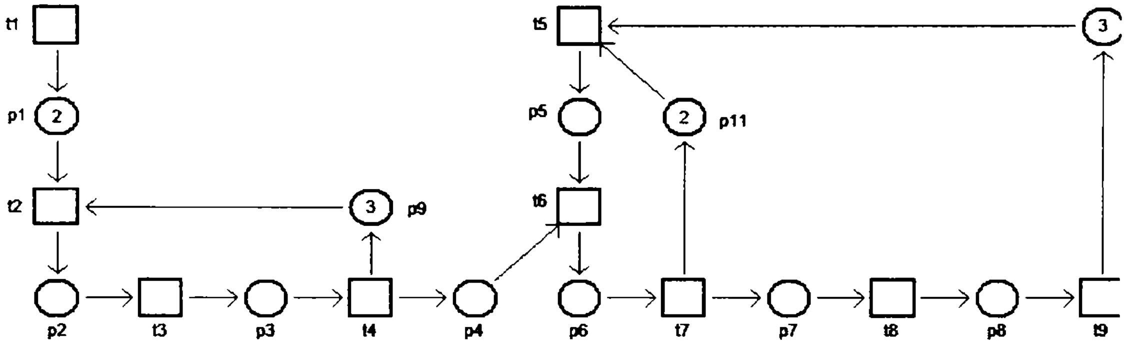


Figura 3.24. Structura retelei Petri corespunzatoare sistemului de transport cu reactie exemplul 3.4

In [Mood99][MA97] este tratat cazul in care se doreste construirea unui supervizor in situatia in care exista tranzitii necontrolabile si neobservabile.

Definitia 3.5 Tranzitie necontrolabila

O tranzitie corespunzatoare unei retele Petri $PN = (P,T,F,W,X_m)$*, se spune ca este necontrolabila daca executia acesteia nu poate fi inhibata printr-o actiune externa.*

Libertatea unei tranzitii necontrolabile de a fi executatata este limitata numai de structura si starile retelei. Pentru ca un supervizor de tip retea Petri sa poata inhiba o tranzitie, aceasta trebuie sa contina un arc de la pozitia de tip controler catre tranzitia vizata. Tranzitia va fi dezactivata numai in cazul in care numarul de jetoane din pozitia controler este mai mic decat capacitatea arcului tranzitiei (conditia de executie a unei tranzitii este nesatisfacuta in acest caz).

Pentru a putea efectua sinteza unui supervizor in situatia in care exista tranzitii necontrolabile, se defineste o matrice de incidenta A_{uc} , extrasa din matricea de incidenta A_P , compusa din liniile corespunzatoare tranzitiilor necontrolabile din sistem. A_{uc} este de tip nxn_{uc} , unde n_{uc} este numarul de tranzitii necontrolabile din sistem.

Definitia 3.6 Tranzitie neobservabila

O tranzitie corespunzatoare unei retele Petri $PN = (P,T,F,W,X_m)$ *se spune ca este neobservabila daca executia acesteia nu poate fi detectata sau masurata.*

O tranzitie neobservabila neputand fi detectata, starea controlerului nu poate fi modificata de executia acestei tranzitii. Pentru un supervizor de tip retea Petri, in cazul unei tranzitii nebservabile, ambele arcuri de intrare, respectiv de iesire, sunt folosite pentru executia tranzitiei.

Pentru a putea efectua sinteza unui supervizor in situatia in care exista tranzitii neoservabile, se defineste o matrice de incidenta A_{uo} , extrasa din matricea de incidenta A_P , compusa din liniile corespunzatoare tranzitiilor neobservabile din sistem. A_{uo} este de tip nxn_{uo} , unde n_{uo} este numarul de tranzitii neobservabile din sistem.

In mod similar cu metodologia descrisa anterior (cazul fara constrangeri), pentru sinteza unui supervizor in cazul existentei de tranzitii necontrolabile se considera ca setul de constrangeri este fortat sa satisfaca conditia:

$$A_{uc}L^T \leq 0 \qquad (3.30)$$

In cazul tranzitilor nobservabile, setul de constrangeri trebuie sa indeplineasca conditia:

$$A_{uo}L^T = 0 \qquad (3.31)$$

Aceste doua conditii indica faptul ca este posibil sa fie observata o tranzitie care sa nu poata fi inhibata, fiind interzisa o inhibare directa a unei tranzitii pe care nu o putem observa.

Teorema 3.5 Structura transformarii constrangerilor

Fie matricile R_1*, cu elemente de tip intregi si dimensiune* $n_c x n$*, care satisface*

conditia: $R_1 \cdot m_p{}^T \geq 0, \ \forall m_p$ *,* (3.32)

si R_2 *de tip diagonala, cu elemente intregi pozitive, si dimensiune* $n_c x n_c$.

Daca $L' \cdot m_p{}^T \leq b'$*, cu*

$$\begin{aligned} L' &= R_1 + R_2 L \\ b' &= R_2(b+1) - 1 \end{aligned} \qquad (3.33)$$

unde 1 este un vector nc dimensional cu toate elementele egale cu 1, atunci:

$$L \cdot m_p{}^T \leq b \qquad (3.34)$$

Teorema 3.6 Transformarea constrangerilor si sinteza supervizorului
Fie o retea Petri $PN = (P, T, F, W, X_m)$ *, cu matricea de incidenta* $A_{p.}$ *Fie setul tranzitiilor necotrolabile descrise prin intermediul matrici* A_{uc} *si setul tranzitiilor neobservabile descrise prin intermediul matricii* A_{uo}*. Retelei Petri considerate i se impune setul liniar de constrangeri* $L \cdot m_p{}^T \leq b$*. Fie* R_1 *si* R_2 *doua matrici de forma celor definite in teorema 3.5 care indeplinesc conditia* $R_1 + R_2 L \neq 0$ *si fie:*

$$[R_1 \;\; R_2] \begin{bmatrix} A_{uc}{}^T & A_{uo}{}^T & -A_{uo}{}^T & m_{p_0}{}^T \\ LA_{uc}{}^T & LA_{uo}{}^T & -LA_{uo}{}^T & Lm_{p_0}{}^T - b - 1 \end{bmatrix} \leq [0 \;\; 0 \;\; 0 \;\; -1] \qquad (3.34)$$

atunci supervizorul

$$\begin{aligned} A_c{}^T &= -(R_1 + R_2 L)A_p{}^T = -L' A_p{}^T \\ m_{c_0}{}^T &= R_2(b+1) - 1 - (R_1 + R_2 L)m_{p_0}{}^T = b' - L' m_{p_0}{}^T \end{aligned} \qquad (3.35)$$

exista si determina ca toate subsecventele marcate a buclei inchise sa satisfaca constrangerea de forma $L \cdot m_p{}^T \leq b$ *fara incercarea de inhibare a tranzitilor necontrolabile si fara detectarea tranzitiilor neobservabile.*

3.3.2. Conducerea supervizata utilizand modele de tip retele Petri bazate pe compunerea paralela a subsistemelor

Principiul de baza al metodei, porneste de la definirea tuturor subsistemelor care compun intreg ansamblul si modelarea acestora astfel incat sa rezulte modele de tip retele Petri care sa satisfaca toate cerintele din punct de vedere structural si comportamental. Dupa definirea specificatiilor de control, pe baza modelelor de tip Retea Petri, prin compunere paralela a acestora va rezulta un sistem *PN_Comp,* care insa nu reprezinta supervizorul dorit datorita faptului ca acesta poate contine pozitii blocante, respectiv tranzitii necontrolabile. Dupa obtinerea sistemului *PN_Comp*, se analizeaza existenta blocajelor sau a tranzitiilor necontrolabile, trecandu-se la eliminarea acestor situatii.

Pasii care trebuiesc urmati in cadrul acestei metode sunt prezentati in cadrul algoritmului 3.5.

Algoritmul 3.5. Sinteza supervizoarelor de tip retea Petri prin compunerea paralela*

- ***Pasul 1*** Se definesc toate modelele de tip retele Petri corespunzatoare subsistemelor precum si specificatiile de control
- ***Pasul 2*** Toate submodelele sun analizate separat, astfel incat ele sa fie viabile din punct de vedere structural si comportamental
- ***Pasul 3*** Se compun paralel toate subsistemele definite impreuna cu specificatiile

de control

- ***Pasul 4*** Se determina setul starilor accesibile, respectiv neaccesibile (generatoare de blocaje)
- ***Pasul 5*** Se analizeaza reteaua din punct de vedere al necontrolabilitatii, incercandu-se ca prin rafinarea retelei sa se elimine situatiile necontrolabile
- ***Pasul 6*** Reteaua finala obtinuta dupa eliminarea situatiilor blocante si necontrolabile reprezinta supervizorul cautat

In [Giua92] este prezentat un algoritm pentru analiza unui model de tip Retea Petri si eliminarea situatilor blocante sau necontrolabile.

Algoritmul 3.6 Eliminarea blocajelor si a situatiilor necontrolabile dintr-o retea Petri prin metoda *Giua*

Se considera o retea Petri etichetata $PN = (P,T,F,W,\Sigma,l,x_0,X_m)$, si fie *t* o tranzitie care trebuie verificata din punct de vedere al generarii de situatii blocante, si *„a”* eticheta acestei tranzitii.

1. Se determina setul marcajelor admisibile la care se ajunge prin executia tranzitiei *t,* si se divide acest set in doua seturi disjuncte $\overline{M_e}$ (setul marcajelor care trebuie atinse la executia tranzitiei *t*) si $\overline{M_d}$ (setul marcajelor pentru care tranzitia *t* trebuie sa fie invalidata, executia ei, ducand la aparitia unei situatii blocante). Daca $\overline{M_e}$ este egal cu multimea vida atunci tranzitia *t* poate fi stearsa direct si aplicarea algoritmului s-a incheiat , daca nu continua cu pasul 2;
2. Se determina o constructie de forma:

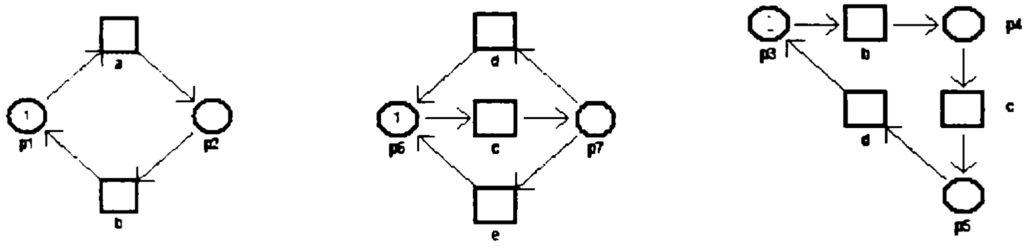
$$U(M) = [(M(p_1^1) \geq n_1^1) \wedge \ldots \wedge (M(p_{k1}^1) \geq n_{k1}^1)] \vee \ldots \vee [(M(p_1^l) \geq n_1^l) \wedge \ldots \wedge (M(p_{k1}^l) \geq n_{k1}^l)]$$

   astfel incat $U(M) = TRUE$ daca $M \in \overline{M_e}$ si $U(M) = FALSE$ daca $M \in \overline{M_d}$

3. Se inlocuieste tranzitia *t* cu *l* tranzitii $t^1,\ldots.,t^l$ etichetate toate cu *„a”*. Arcele de intrare/iesire a tranzitiei t^j, $j=1,\ldots,l$ vor fi aceleasi ca ale tranzitiei *t,* plus un numar de n_i^j arce de intrare/iesire de la pozitia p_i^j , $i=1,\ldots,k_j$

Exemplul 3.5 Implementarea unui supervizor prin metoda compunerii paralele

Fie un sistem caracterizat prin doua subsisteme modelate prin intermediul retelelor Petri *PN1* (figura 3.25.a), respectiv *PN2* (figura 3.25.b) si fie specificatiile de control date prin intermediul retelei Petri *PN_Spec* (vezi figura 3.25.c). Se doreste construirea unui supervizor corespunzator sistemului propus.



a) b) c)

Figura 3.25. Subsistemele (a si b) si specificatile (c) corespunzatoare sistemului considerat la exemplul 3.5

Compunerea paralela a acestor retele Petri va genera o retea Petri cu matricea de incidenta de dimensiune 5x7 (5 tranzitii a, b, c, d si e si 7 pozitii p1,p2 p3, p4, p5, p6 si p7).Aplicand algoritmul de compunere paralela rezulta matricea de incidenta:

$$A_{SIST} = \begin{pmatrix} -1 & 1 & 0 & 0 & 0 & 0 & 0 \\ 1 & -1 & -1 & 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & -1 & 1 & -1 & 1 \\ 0 & 0 & 1 & 0 & -1 & 1 & -1 \\ 0 & 0 & 0 & 0 & 0 & 1 & -1 \end{pmatrix}$$

Structura retelei Petri PN_Sist=PN1||PN2||PN_Spec este prezentata in figura 3.26.

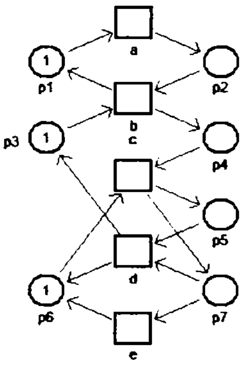


Figura 3.26. Structura retelei Petri rezultatate in urma compunerii paralele

Arborele de acoperire al retelei rezultate este prezentat in figura 3.27.

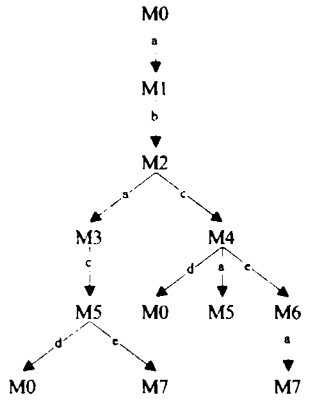


Figura 3.27. Arborele de acoperire corespunzator retelei Petri *PN_Sist*

unde:

M0=[1,0,1,0,0,1,0]^T; M1=[0,1,1,0,0,1,0]^T; M2=[1,0,0,1,0,1,0]^T;
M3=[0,1,0,1,0,1,0]^T;
M4=[1,0,0,0,1,0,1]^T; M5=[0,1,0,0,1,0,1]^T; M6=[1,0,0,0,1,1,0]^T;
M7=[1,0,0,0,1,1,0]^T.

Dupa cum se observa din structura arborelui de acoperire, prin atingerea marcajului *M7* se ajunge intr-o stare de blocaj (deadlock), datorata executiei tranzitiei *e*, tranzitie generatoare de blocaj. Trebuie facuta insa observatia ca marcajul *M7* mai poate fi atins si prin executia tranzitiei *a* (din *M6*), dar aceasta tranzitie nu este generatoare de blocaj datorita faptului ca exista un circuit care contine tranzitia *a* si care pornind din marcajul initial *M0,* are ca rezultat final tot *M0*, adica: $M0 \xrightarrow{t1} M1 \xrightarrow{t2} M2 \xrightarrow{t1} M3 \xrightarrow{t3} M5 \xrightarrow{t4} M0$. Blocajul sistemului este relevant in cazul construirii grafului de acoperire, care reprezinta in fapt automatul retelei. In figura 3.28 se prezinta graful de acoperire corespunzator retelei Petri *PN_Sist*.

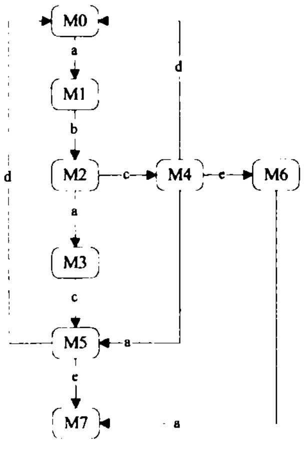


Figura 3.28 Graful de acoperire corespunzator retelei Petri *PN_Sist*

Este vizibil faptul ca prin executia tranzitiei *e* se ajunge in situatia de blocaj (deadlock). Rezolvarea acestei situatii (eliminarea blocajului) se rezolva prin suprimarea tranzitiei *e*, rezultand o retea Petri caracterizata prin urmatoarea matrice de incidenta:

$$A_{SIST} = \begin{pmatrix} -1 & 1 & 0 & 0 & 0 & 0 & 0 \\ 1 & -1 & -1 & 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & -1 & 1 & -1 & 1 \\ 0 & 0 & 1 & 0 & -1 & 1 & -1 \end{pmatrix}$$

Structura retelei Petri rezultata este prezentata in figura 3.29.

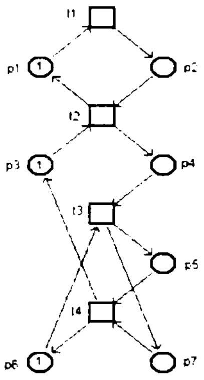


Figura 3.29. Structura retelei Petri *PN_Sist_Mod* obtinuta din *PN_Sist* dupa eliminarea tranzitiei *e*

Arborele de acoperire corespunzator retelei PN_Siste_Mod este prezentat in figura 3.30.

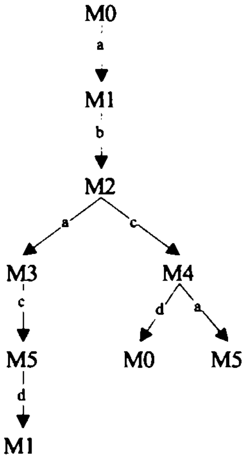


Figura 3.30 Arborele de acoperire corespunzator retelei Petri *PN_Sist_Mod*

unde:
$M0=[1,0,1,0,0,1,0]^T$; $M1=[0,1,1,0,0,1,0]^T$;
$M2=[1,0,0,1,0,1,0]^T$; $M3=[0,1,0,1,0,1,0]^T$;
$M4=[1,0,0,0,1,0,1]^T$; $M5=[0,1,0,0,1,0,1]^T$.

Din structura arborelui de acoperire rezulta deci ca reteaua nu este blocanta.

Reteaua *Petri PN_Sist_Mod* (vezi figura 3.29) reprezinta in fapt supervizorul cautat pentru aplicatia considerata.

3.3.3. Studiu de caz. Conducerea supervizata a unui sistem de transport cu zone de acumulare [Ung03][Ung06a]

Se considera un sistem de transport cu structura prezentata in figura 3.31.

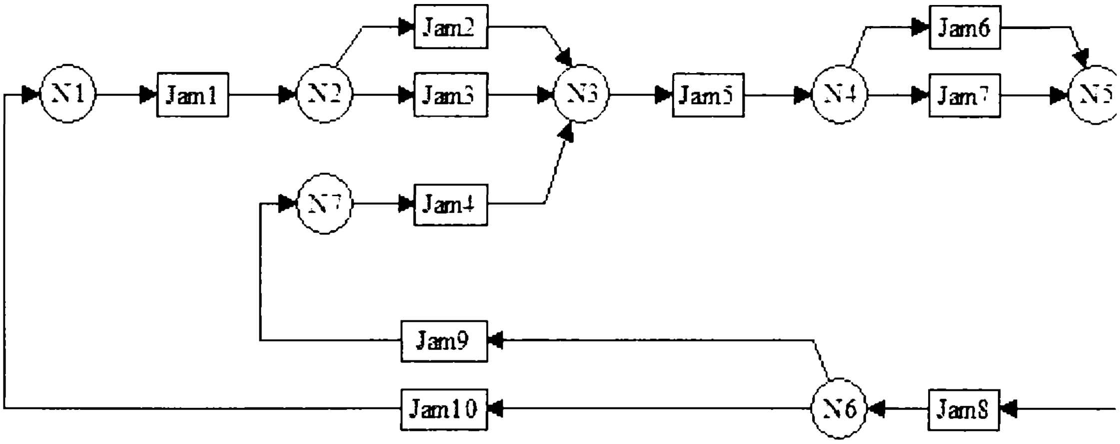


Figura 3.31. Structura sistemului de transport

Capacitatile maxime ale jam-urilor, precum si starea initiala a acestora, sunt prezentate in tabelul 3.1.

Tabelul 3.1 *Capacitatile maxime si continutul initial al jam-urilor*

Jam ID	Cap. Maxima	Cap. Initial
Jam 1	2	1
Jam 2	5	4
Jam 3	5	4
Jam 4	5	4
Jam 5	2	1
Jam 6	5	4
Jam 7	5	4
Jam 8	3	1
Jam 9	5	4
Jam10	10	8

Dupa cum rezulta din tabelul 3.1 in sistem se gasesc 35 de carucioare, iar spatiul maxim disponibil este de 47 de carucioare. Datorita acestei incarcari sistemul este functional, neexistand riscul de aparitie a blocajelor de functionare.

Analizand structura sistemului de transport considerata in figura 3.31 se observa ca acesta are in componenta s-a sase noduri, din care: doua de tip 1 (N1 si N7), trei de tip 4 (N2, N4 si N6), unul de tip 3 (N3) si unul de tip 2 (N5).

Structura retelei Petri corespunzatoare modelarii sistemului propus este prezentata in figura 3.32.

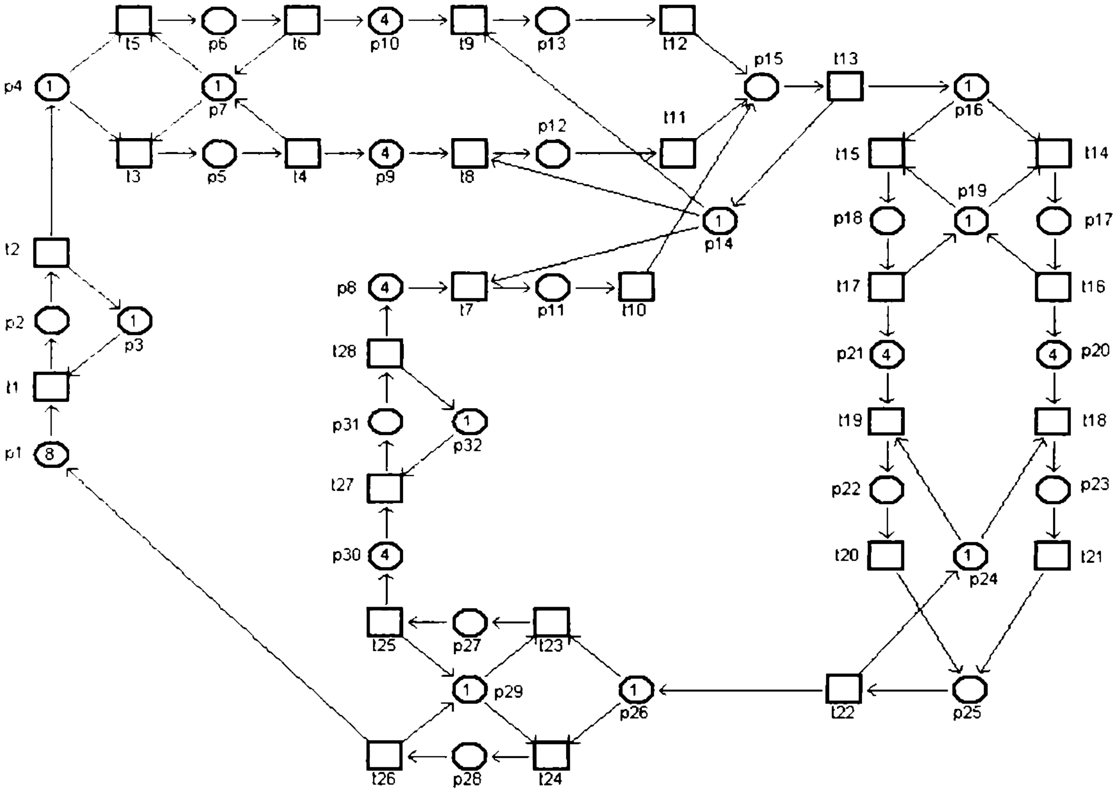


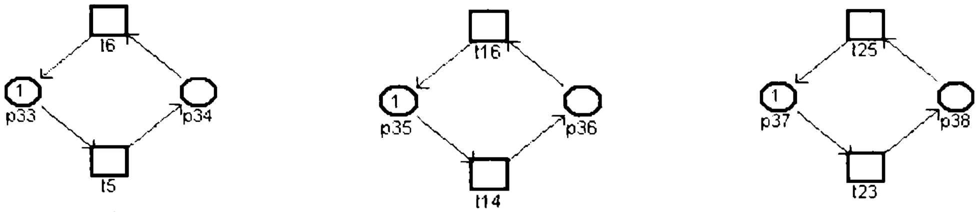
Figura 3.32. Structura retelei Petri corespunzatoare sistemului propus

Obs. Reteaua Petri care modeleaza sistemul este considerata ca fiind o retea Petri cu etichete, semnificatiile etichetelor fiind usor de intuit avand la baza exemplul 3.5.

Functionarea sistemului se bazeaza pe urmatoarele specificatii:

1) Nodul N2 (nod de tip 4) va functiona astfel incat in situatia in care rezultatul scanarii este diferit de o valoare prescrisa (de exemplu 2), toate carucioarele corespunzatoare vor fi mutate in jam-ul 3;
2) Nodul N3 (nod de tip 3) va functiona dupa principiul rotatiei – din fiecare stoper se elibereaza rand pe rand cate un carucior. Daca intr-un jam nu se afla carucioare atunci se trece la urmatorul jam.
3) Nodul N4 (nod de tip 4) va functiona astfel incat in situatia in care rezultatul scanarii este diferit de o valoare prescrisa (de exemplu 6), toate carucioarele corespunzatoare vor fi mutate in jam-ul 7;
4) Nodul N5 (nod de tip 2) functioneaza dupa principiul descris pentru nodul N3;
5) Nodul N6 (nod de tip 4) va functiona astfel incat in situatia in care rezultatul scanarii este diferit de o valoare prescrisa (de exemplu 9), toate carucioarele corespunzatoare vor fi mutate in jam-ul 10.

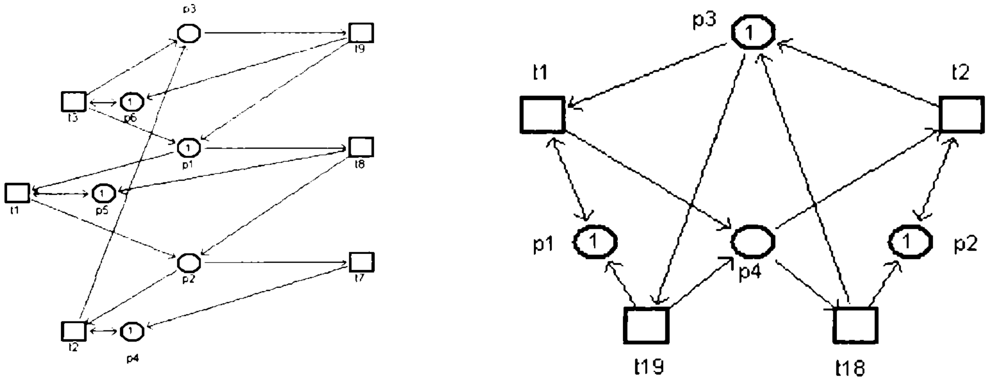
Modelele de tip retea Petri corespunzatoare specificatiilor impuse nodurilor de tip 4 (N2, N4 si N6) sunt prezentate in figura 3.33.



a) b) c)

Figura 3.33. Modelele de tip retea Petri corespunzatoare specificatiilor nodurilor N2 (a), N4 (b) si N6 (c)

Modelele de tip retea Petri corespunzatoare specificatiilor impuse nodului de tip 3 (N3), respectiv nodului de tip 2 (N5), sunt prezentate in figurile 3.34.a respectiv 3.34.b



a) b)

Figura 3.34. Modelele de tip retea Petri aferente specificatiilor nodului N3 (a), respectiv nodul N5 (b)

Pentru determinarea structurii supervizorului cautat se efectueaza compunerea paralela a tuturor modelelor propuse. Matricea de incidenta a retelei Petri rezultate (*PN_Sist*) este:

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

Structura retelei Petri *PN_Sist* obtinute este prezentata in figura 3.35.

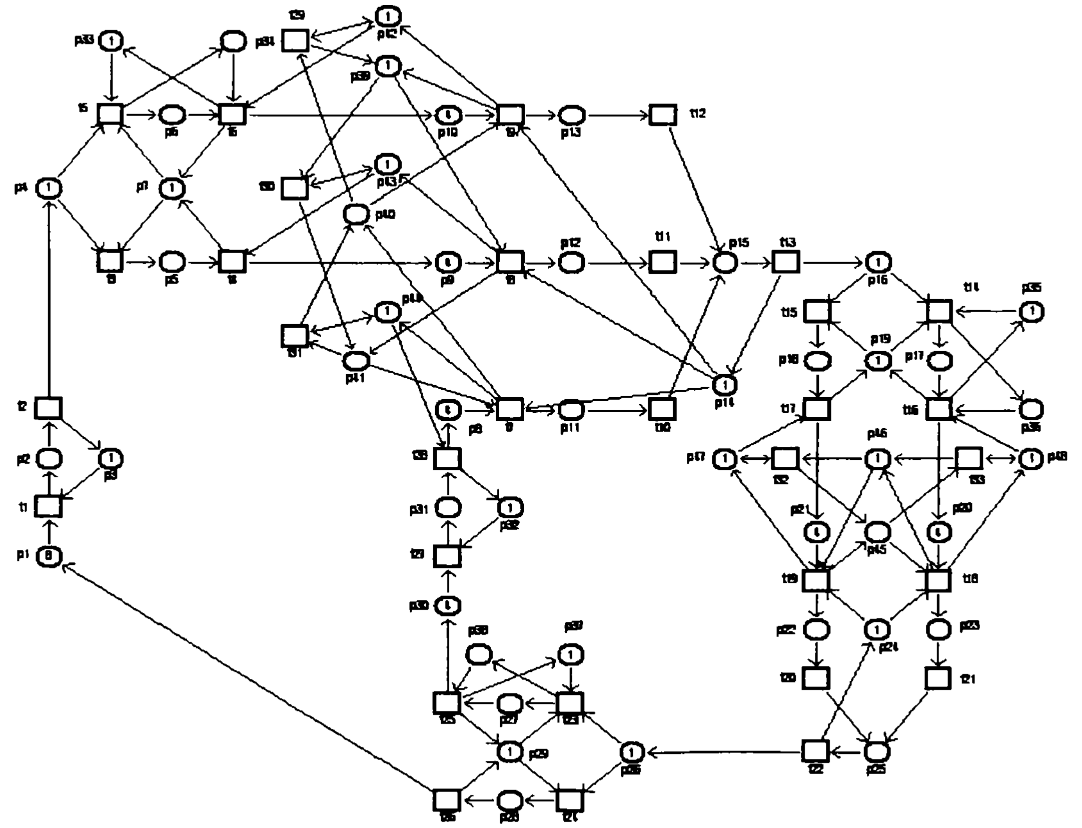


Figura 3.35. Structura retelei Petri *PN_Sist* dupa compunerea paralela

Pornind de la modul de constructie al retelei Petri *PN_Sist,* analiza din punct de vedere al blocajelor se face pe portiuni, in fapt pe baza utilizarii submodelelor. Aceasta analiza este posibila deoarece submodelele utilizate sunt conectate intre ele in asa fel incat proprietatile lor nu sunt influentate intre ele. Submodelele de baza sunt cele analizate in paragraful 2.5.3 si care au fost validate prin simularea lor in Matlab.

Prin adaugarea specificatiilor stabilite, structurile de baza pentru noduri s-au modificat si din acest motiv este necesara analiza proprietatilor structurale si comportamentale ale acestora. Structurile nodurilor N2, N4 si N6 sunt identice, astfel ca analiza s-a realizat doar pentru un singur nod (N2). Structura modificata corespunzator nodului N2 (nod de tip 4) este prezentata in figura 3.36.

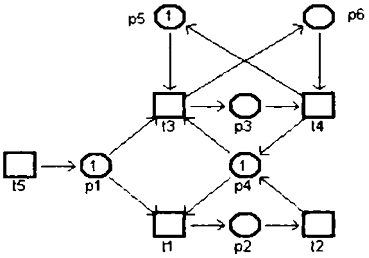


Figura 3.36. Structura modificata corespunzatoare nodului N2 - nod de tip 4

In figura 3.36, tranzitia t5 are rol de generator , prin intermediul ei asigurandu-se alimentarea cu jetoane a retelei Petri considerate.

Deoarece in cazul acestei retele intereseaza numai daca exista posibilitatea aparitiei blocajelor, in continuare se analizeaza reteaua numai pe baza arborelui de acoperire, care este prezentat in figura 3.37.

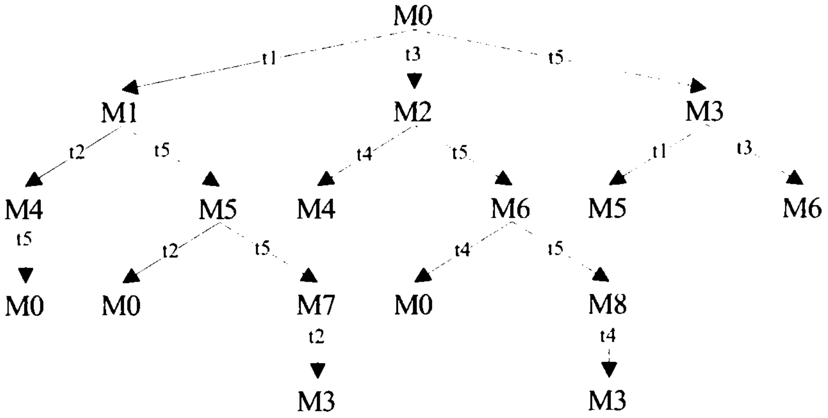


Figura 3.37. Arborele de acoperire al modelului modificat de tip retea Petri pentru nodul de tip 4

unde:
M0=[1,0,0,1,1,0]; M1=[0,1,0,0,1,0]; M2=[0,0,1,0,0,1]; M3=[2,0,0,1,1,0]; M4=[0,0,0,1,1,0]; M5=[1,1,0,0,1,0]; M6=[1,0,1,0,0,1]; M7=[2,1,0,0,1,0]; M8=[2,0,1,0,0,1].

Structura arborelui de acoperire arata faptul ca reteaua Petri considerata, subreteaua corespunzatoare nodului N2, nu este blocanta. Acest rezultat este valabil si pentru nodurile N4 si N6, deoarece si ele sunt noduri de tip 4, cu aceleasi specificatii functionale impuse in cadrul ansamblului.

In continuare se analizeaza nodul N3 (nod de tip 3) pentru verificarea existentei sau nu a blocajelor in urma introducerii specificatiilor de conducere. Structura subretelei considerata este prezentata in figura 3.38.

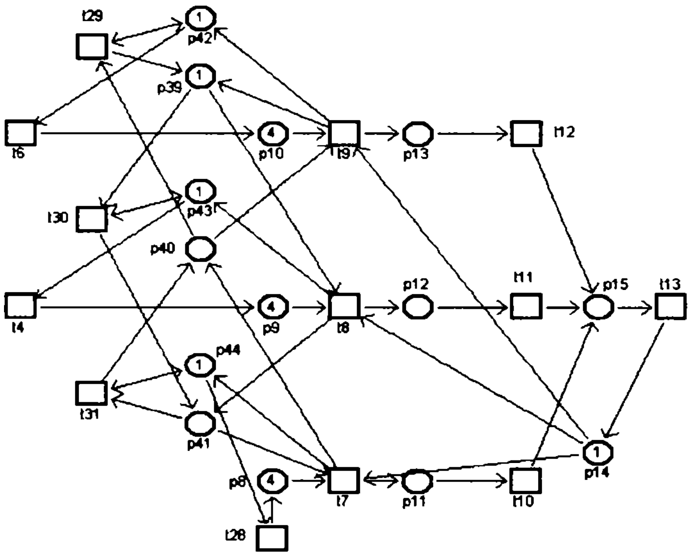


Figura 3.38. Structura modificata a nodului N3 - nod de tip 3

Tranzitile *t8, t9* si *t10* sunt tranzitii generatoare pentru alimentarea retelei cu jetoane.

Datorita complexitatii arborelui de acoperire acesta este prezentat in mod text in tabelul 3.2.

Tabelul 3.2 *Arborele de acoperire corespunzator nodului N3 modificat*

[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]
M0	t8	M1	**M0**	**t28**	**M2**	M1	t11	M3
M1	**t28**	**M4**	M2	t8	M4	**M3**	**t13**	**M5**
M3	t28	M6	**M4**	**t11**	**M6**	M5	t7	M7
M5	**t28**	**M8**	M6	t13	M8	**M7**	**t10**	**M9**
M7	t28	M10	**M8**	**t7**	**M10**	M9	t13	M11
M9	**t28**	**M12**	M10	t10	M12	**M10**	**t28**	**M13**
M11	t9	M14	**M11**	**t28**	**M15**	M12	t13	M15
M12	**t28**	**M16**	M13	t10	M16	**M14**	**t12**	**M17**
M14	t28	M18	**M15**	**t9**	**M18**	M15	t28	M19
M16	**t13**	**M19**	M17	t13	M20	**M17**	**t28**	**M21**
M18	t12	M21	**M18**	**t28**	**M22**	M19	t9	M22

M20	**t8**	**M23**	M20	t28	M24	**M21**	**t13**	**M24**
M21	t28	M25	**M22**	**t12**	**M25**	M23	t11	M26
M23	**t28**	**M27**	M24	t8	M27	**M24**	**t28**	**M28**
M25	t13	M28	**M26**	**t13**	**M29**	M26	t28	M30
M27	**t11**	**M30**	M27	t28	M31	**M28**	**t8**	**M31**
M29	t7	M32	**M29**	**t28**	**M33**	M30	t13	M33
M30	**t28**	**M34**	M31	t11	M34	**M32**	**t10**	**M35**
M32	t28	M36	**M33**	**t7**	**M36**	M33	t28	M37
M34	**t13**	**M37**	M35	t13	M38	**M35**	**t28**	**M39**
M36	t10	M39	**M36**	**t28**	**M40**	M37	t7	M40
M38	**t9**	**M41**	M38	t28	M42	**M39**	**t13**	**M42**
M39	t28	M43	**M40**	**t10**	**M43**	M40	t28	M44
M41	**t12**	**M45**	M41	t28	M46	**M42**	**t9**	**M46**
M42	t28	M47	**M43**	**t13**	**M47**	M43	t28	M48
M44	**t10**	**M48**	M45	t13	M49	**M45**	**t28**	**M50**
M46	t12	M50	**M46**	**t28**	**M51**	M47	t9	M51
M47	**t28**	**M52**	M48	t13	M52	**M49**	**t8**	**M53**
M49	t28	M54	**M50**	**t13**	**M54**	M50	t28	M55
M51	**t12**	**M55**	M51	t28	M56	**M52**	**t9**	**M56**
M53	t11	M57	**M53**	**t28**	**M58**	M54	t8	M58
M54	**t28**	**M59**	M55	t13	M59	**M55**	**t28**	**M60**
M56	t12	M60	**M57**	**t13**	**M61**	M57	t28	M62
M58	**t11**	**M62**	M58	t28	M63	**M59**	**t8**	**M63**
M59	t28	M64	**M60**	**t13**	**M64**	M61	t7	M65
M61	**t28**	**M66**	M62	t13	M66	**M62**	**t28**	**M67**
M63	t11	M67	**M63**	**t28**	**M68**	M64	t8	M68
M65	**t10**	**M69**	M65	t28	M70	**M66**	**t7**	**M70**
M66	t28	M71	**M67**	**t13**	**M71**	M67	t28	M72
M68	**t11**	**M72**	M69	t13	M73	**M69**	**t28**	**M74**
M70	t10	M74	**M70**	**t28**	**M75**	M71	t7	M75
M71	**t28**	**M76**	M72	t13	M76	**M73**	**t9**	**M77**
M73	t28	M78	**M74**	**t13**	**M78**	M74	t28	M79
M75	**t10**	**M79**	M75	t28	M80	**M76**	**t7**	**M80**
M77	t12	M81	**M77**	**t28**	**M82**	M78	t9	M82
M78	**t28**	**M83**	M79	t13	M83	**M79**	**t28**	**M84**

M80	t10	M84	**M80**	**t28**	**M85**	M81	t13	M86
M81	**t28**	**M87**	M82	t12	M87	**M82**	**t28**	**M88**
M83	t9	M88	**M83**	**t28**	**M89**	M84	t13	M89
M84	**t28**	**M90**	M85	t10	M90	**M86**	**t8**	**M91**
M86	t28	M92	**M87**	**t13**	**M92**	M87	t28	M93
M88	**t12**	**M93**	M88	t28	M94	**M89**	**t9**	**M94**
M89	t28	M95	**M90**	**t13**	**M95**	M91	t11	M96
M91	**t28**	**M97**	M92	t8	M97	**M92**	**t28**	**M98**
M93	t13	M98	**M93**	**t28**	**M99**	M94	t12	M99
M94	**t28**	**M100**	M95	t9	M100	**M96**	**t13**	**M101**
M96	t28	M102	**M97**	**t11**	**M102**	M97	t28	M103
M98	**t8**	**M103**	M98	t28	M104	**M99**	**t13**	**M104**
M99	t28	M105	**M100**	**t12**	**M105**	M101	t7	M106
M101	**t28**	**M107**	M102	t13	M107	**M102**	**t28**	**M108**
M103	t11	M108	**M103**	**t28**	**M109**	M104	t8	M109
M104	**t28**	**M110**	M105	t13	M110	**M106**	**t10**	**M111**
M106	t28	M112	**M107**	**t7**	**M112**	M107	t28	M113
M108	**t13**	**M113**	M108	t28	M114	**M109**	**t11**	**M114**
M109	t28	M115	**M110**	**t8**	**M115**	M111	t13	M116
M111	**t28**	**M117**	M112	t10	M117	**M112**	**t28**	**M118**
M113	t7	M118	**M113**	**t28**	**M119**	M114	t13	M119
M114	**t28**	**M120**	M115	t11	M120	**M116**	**t9**	**M121**
M116	t28	M122	**M117**	**t13**	**M122**	M117	t28	M123
M118	**t10**	**M123**	M118	t28	M124	**M119**	**t7**	**M124**
M119	t28	M125	**M120**	**t13**	**M125**	M121	t12	M126
M121	**t28**	**M127**	M121	t30	M128	**M122**	**t9**	**M127**
M122	t28	M129	**M123**	**t13**	**M129**	M123	t28	M130
M124	**t10**	**M130**	M124	t28	M131	**M125**	**t7**	**M131**
M126	t13	M132	**M126**	**t28**	**M133**	M126	t30	M134
M127	**t12**	**M133**	M127	t28	M135	**M127**	**t30**	**M136**
M128	t12	M134	**M128**	**t28**	**M136**	M129	t9	M135
M129	**t28**	**M137**	M130	t13	M137	**M130**	**t28**	**M138**
M131	t10	M138	**M131**	**t28**	**M139**	M132	t28	M140
M132	**t30**	**M141**	M133	t13	M140	**M133**	**t28**	**M142**
M133	t30	M143	**M134**	**t13**	**M141**	M134	t28	M143

M135	**t12**	**M142**	M135	t28	M144	**M135**	**t30**	**M145**
M136	t12	M143	**M136**	**t28**	**M145**	M137	t9	M144
M137	**t28**	**M146**	M138	t13	M146	**M138**	**t28**	**M147**
M139	t10	M147	**M140**	**t28**	**M148**	M140	t30	M149
M141	**t28**	**M149**	M142	t13	M148	**M142**	**t28**	**M150**
M142	t30	M151	**M143**	**t13**	**M149**	M143	t28	M151
M144	**t12**	**M150**	M144	t28	M152	**M144**	**t30**	**M153**
M145	t12	M151	**M145**	**t28**	**M153**	M146	t9	M152
M146	**t28**	**M154**	M147	t13	M154	**M148**	**t28**	**M155**
M148	t30	M156	**M149**	**t7**	**M157**	M149	t28	M156
M150	**t13**	**M155**	M150	t28	M158	**M150**	**t30**	**M159**
M151	t13	M156	**M151**	**t28**	**M159**	M152	t12	M158
M152	**t28**	**M160**	M152	t30	M161	**M153**	**t12**	**M159**
M153	t28	M161	**M154**	**t9**	**M160**	M155	t28	M162
M155	**t30**	**M163**	M156	t7	M164	**M156**	**t28**	**M163**
M157	t10	M165	**M157**	**t28**	**M164**	M157	t29	M166
M158	**t13**	**M162**	M158	t28	M167	**M158**	**t30**	**M168**
M159	t13	M163	**M159**	**t28**	**M168**	M160	t12	M167
M160	**t30**	**M169**	M161	t12	M168	**M161**	**t28**	**M169**
M162	t28	M170	**M162**	**t30**	**M171**	M163	t7	M172
M163	**t28**	**M171**	M164	t10	M173	**M164**	**t28**	**M172**
M164	t29	M174	**M165**	**t13**	**M175**	M165	t28	M173
M165	**t29**	**M126**	M166	t10	M126	**M166**	**t28**	**M174**
M166	t30	M176	**M167**	**t13**	**M170**	M167	t30	M177
M168	**t13**	**M171**	M168	t28	M177	**M169**	**t12**	**M177**
M170	t30	M178	**M171**	**t7**	**M179**	M171	t28	M178
M172	**t10**	**M180**	M172	t28	M179	**M172**	**t29**	**M181**
M173	t13	M182	**M173**	**t28**	**M180**	M173	t29	M133
M174	**t10**	**M133**	M174	t28	M181	**M174**	**t30**	**M183**
M175	t28	M182	**M175**	**t29**	**M132**	M176	t10	M134
M176	**t28**	**M183**	M177	t13	M178	**M178**	**t7**	**M184**
M179	t10	M185	**M179**	**t28**	**M184**	M179	t29	M186
M180	**t13**	**M187**	M180	t28	M185	**M180**	**t29**	**M142**
M181	t10	M142	**M181**	**t28**	**M186**	M181	t30	M188
M182	**t28**	**M187**	M182	t29	M140	**M183**	**t10**	**M143**

M183	t28	M188	**M184**	**t10**	**M189**	M184	t28	M190
M184	**t29**	**M191**	M185	t13	M192	**M185**	**t28**	**M189**
M185	t29	M150	**M186**	**t10**	**M150**	M186	t28	M191
M186	**t30**	**M193**	M187	t28	M192	**M187**	**t29**	**M148**
M188	t10	M151	**M188**	**t28**	**M193**	M189	t13	M194
M189	**t28**	**M195**	M189	t29	M158	**M190**	**t10**	**M195**
M190	t29	M196	**M191**	**t10**	**M158**	M191	t28	M196
M191	**t30**	**M197**	M192	t28	M194	**M192**	**t29**	**M155**
M193	t10	M159	**M193**	**t28**	**M197**	M194	t28	M198
M194	**t29**	**M162**	M195	t13	M198	**M195**	**t29**	**M167**
M196	t10	M167	**M196**	**t30**	**M199**	M197	t10	M168
M197	**t28**	**M199**	M198	t29	M170	**M199**	**t10**	**M177**

M0 = [4,4,4,0,0,0,1,0,1,0,0,1,1,1]	M1 = [4,3,4,0,1,0,0,0,0,0,1,1,2,1]
M2 = [5,4,4,0,0,0,1,0,1,0,0,1,1,0]	M3 = [4,3,4,0,0,0,0,1,0,0,1,1,2,1]
M4 = [5,3,4,0,1,0,0,0,0,0,1,1,2,0]	M5 = [4,3,4,0,0,0,1,0,0,0,1,1,2,1]
M6 = [5,3,4,0,0,0,0,1,0,0,1,1,2,0]	M7 = [3,3,4,1,0,0,0,0,0,1,0,1,2,2]
M8 = [5,3,4,0,0,0,1,0,0,0,1,1,2,0]	M9 = [3,3,4,0,0,0,0,1,0,1,0,1,2,2]
M10 = [4,3,4,1,0,0,0,0,0,1,0,1,2,1]	M11 = [3,3,4,0,0,0,1,0,0,1,0,1,2,2]
M12 = [4,3,4,0,0,0,0,1,0,1,0,1,2,1]	M13 = [5,3,4,1,0,0,0,0,0,1,0,1,2,0]
M14 = [3,3,3,0,0,1,0,0,1,0,0,2,2,2]	M15 = [4,3,4,0,0,0,1,0,0,1,0,1,2,1]
M16 = [5,3,4,0,0,0,0,1,0,1,0,1,2,0]	M17 = [3,3,3,0,0,0,0,1,1,0,0,2,2,2]
M18 = [4,3,3,0,0,1,0,0,1,0,0,2,2,1]	M19 = [5,3,4,0,0,0,1,0,0,1,0,1,2,0]
M20 = [3,3,3,0,0,0,1,0,1,0,0,2,2,2]	M21 = [4,3,3,0,0,0,0,1,1,0,0,2,2,1]
M22 = [5,3,3,0,0,1,0,0,1,0,0,2,2,0]	M23 = [3,2,3,0,1,0,0,0,0,0,1,2,3,2]
M24 = [4,3,3,0,0,0,1,0,1,0,0,2,2,1]	M25 = [5,3,3,0,0,0,0,1,1,0,0,2,2,0]
M26 = [3,2,3,0,0,0,0,1,0,0,1,2,3,2]	M27 = [4,2,3,0,1,0,0,0,0,0,1,2,3,1]
M28 = [5,3,3,0,0,0,1,0,1,0,0,2,2,0]	M29 = [3,2,3,0,0,0,1,0,0,0,1,2,3,2]
M30 = [4,2,3,0,0,0,0,1,0,0,1,2,3,1]	M31 = [5,2,3,0,1,0,0,0,0,0,1,2,3,0]
M32 = [2,2,3,1,0,0,0,0,0,1,0,2,3,3]	M33 = [4,2,3,0,0,0,1,0,0,0,1,2,3,1]
M34 = [5,2,3,0,0,0,0,1,0,0,1,2,3,0]	M35 = [2,2,3,0,0,0,0,1,0,1,0,2,3,3]
M36 = [3,2,3,1,0,0,0,0,0,1,0,2,3,2]	M37 = [5,2,3,0,0,0,1,0,0,0,1,2,3,0]
M38 = [2,2,3,0,0,0,1,0,0,1,0,2,3,3]	M39 = [3,2,3,0,0,0,0,1,0,1,0,2,3,2]
M40 = [4,2,3,1,0,0,0,0,0,1,0,2,3,1]	M41 = [2,2,2,0,0,1,0,0,1,0,0,3,3,3]
M42 = [3,2,3,0,0,0,1,0,0,1,0,2,3,2]	M43 = [4,2,3,0,0,0,0,1,0,1,0,2,3,1]

M44 = [5,2,3,1,0,0,0,0,0,1,0,2,3,0]
M45 = [2,2,2,0,0,0,0,1,1,0,0,3,3,3]
M46 = [3,2,2,0,0,1,0,0,1,0,0,3,3,2]
M47 = [4,2,3,0,0,0,1,0,0,1,0,2,3,1]
M48 = [5,2,3,0,0,0,0,1,0,1,0,2,3,0]
M49 = [2,2,2,0,0,0,1,0,1,0,0,3,3,3]
M50 = [3,2,2,0,0,0,0,1,1,0,0,3,3,2]
M51 = [4,2,2,0,0,1,0,0,1,0,0,3,3,1]
M52 = [5,2,3,0,0,0,1,0,0,1,0,2,3,0]
M53 = [2,1,2,0,1,0,0,0,0,0,1,3,4,3]
M54 = [3,2,2,0,0,0,1,0,1,0,0,3,3,2]
M55 = [4,2,2,0,0,0,0,1,1,0,0,3,3,1]
M56 = [5,2,2,0,0,1,0,0,1,0,0,3,3,0]
M57 = [2,1,2,0,0,0,0,1,0,0,1,3,4,3]
M58 = [3,1,2,0,1,0,0,0,0,0,1,3,4,2]
M59 = [4,2,2,0,0,0,1,0,1,0,0,3,3,1]
M60 = [5,2,2,0,0,0,0,1,1,0,0,3,3,0]
M61 = [2,1,2,0,0,0,1,0,0,0,1,3,4,3]
M62 = [3,1,2,0,0,0,0,1,0,0,1,3,4,2]
M63 = [4,1,2,0,1,0,0,0,0,0,1,3,4,1]
M64 = [5,2,2,0,0,0,1,0,1,0,0,3,3,0]
M65 = [1,1,2,1,0,0,0,0,0,1,0,3,4,4]
M66 = [3,1,2,0,0,0,1,0,0,0,1,3,4,2]
M67 = [4,1,2,0,0,0,0,1,0,0,1,3,4,1]
M68 = [5,1,2,0,1,0,0,0,0,0,1,3,4,0]
M69 = [1,1,2,0,0,0,0,1,0,1,0,3,4,4]
M70 = [2,1,2,1,0,0,0,0,0,1,0,3,4,3]
M71 = [4,1,2,0,0,0,1,0,0,0,1,3,4,1]
M72 = [5,1,2,0,0,0,0,1,0,0,1,3,4,0]
M73 = [1,1,2,0,0,0,1,0,0,1,0,3,4,4]
M74 = [2,1,2,0,0,0,0,1,0,1,0,3,4,3]
M75 = [3,1,2,1,0,0,0,0,0,1,0,3,4,2]
M76 = [5,1,2,0,0,0,1,0,0,0,1,3,4,0]
M77 = [1,1,1,0,0,1,0,0,1,0,0,4,4,4]
M78 = [2,1,2,0,0,0,1,0,0,1,0,3,4,3]
M79 = [3,1,2,0,0,0,0,1,0,1,0,3,4,2]
M80 = [4,1,2,1,0,0,0,0,0,1,0,3,4,1]
M81 = [1,1,1,0,0,0,0,1,1,0,0,4,4,4]
M82 = [2,1,1,0,0,1,0,0,1,0,0,4,4,3]
M83 = [3,1,2,0,0,0,1,0,0,1,0,3,4,2]
M84 = [4,1,2,0,0,0,0,1,0,1,0,3,4,1]
M85 = [5,1,2,1,0,0,0,0,0,1,0,3,4,0]
M86 = [1,1,1,0,0,0,1,0,1,0,0,4,4,4]
M87 = [2,1,1,0,0,0,0,1,1,0,0,4,4,3]
M88 = [3,1,1,0,0,1,0,0,1,0,0,4,4,2]
M89 = [4,1,2,0,0,0,1,0,0,1,0,3,4,1]
M90 = [5,1,2,0,0,0,0,1,0,1,0,3,4,0]
M91 = [1,0,1,0,1,0,0,0,0,0,1,4,5,4]
M92 = [2,1,1,0,0,0,1,0,1,0,0,4,4,3]
M93 = [3,1,1,0,0,0,0,1,1,0,0,4,4,2]
M94 = [4,1,1,0,0,1,0,0,1,0,0,4,4,1]
M95 = [5,1,2,0,0,0,1,0,0,1,0,3,4,0]
M96 = [1,0,1,0,0,0,0,1,0,0,1,4,5,4]
M97 = [2,0,1,0,1,0,0,0,0,0,1,4,5,3]
M98 = [3,1,1,0,0,0,1,0,1,0,0,4,4,2]
M99 = [4,1,1,0,0,0,0,1,1,0,0,4,4,1]
M100 = [5,1,1,0,0,1,0,0,1,0,0,4,4,0]
M101 = [1,0,1,0,0,0,1,0,0,0,1,4,5,4]
M102 = [2,0,1,0,0,0,0,1,0,0,1,4,5,3]
M103 = [3,0,1,0,1,0,0,0,0,0,1,4,5,2]
M104 = [4,1,1,0,0,0,1,0,1,0,0,4,4,1]
M105 = [5,1,1,0,0,0,0,1,1,0,0,4,4,0]
M106 = [0,0,1,1,0,0,0,0,0,1,0,4,5,5]
M107 = [2,0,1,0,0,0,1,0,0,0,1,4,5,3]
M108 = [3,0,1,0,0,0,0,1,0,0,1,4,5,2]
M109 = [4,0,1,0,1,0,0,0,0,0,1,4,5,1]
M110 = [5,1,1,0,0,0,1,0,1,0,0,4,4,0]
M111 = [0,0,1,0,0,0,0,1,0,1,0,4,5,5]
M112 = [1,0,1,1,0,0,0,0,0,1,0,4,5,4]
M113 = [3,0,1,0,0,0,1,0,0,0,1,4,5,2]

M114 = [4,0,1,0,0,0,0,1,0,0,1,4,5,1]

M115 = [5,0,1,0,1,0,0,0,0,0,1,4,5,0]

M116 = [0,0,1,0,0,0,1,0,0,1,0,4,5,5]

M117 = [1,0,1,0,0,0,0,1,0,1,0,4,5,4]

M118 = [2,0,1,1,0,0,0,0,0,1,0,4,5,3]

M119 = [4,0,1,0,0,0,1,0,0,0,1,4,5,1]

M120 = [5,0,1,0,0,0,0,1,0,0,1,4,5,0]

M121 = [0,0,0,0,0,1,0,0,1,0,0,5,5,5]

M122 = [1,0,1,0,0,0,1,0,0,1,0,4,5,4]

M123 = [2,0,1,0,0,0,0,1,0,1,0,4,5,3]

M124 = [3,0,1,1,0,0,0,0,0,1,0,4,5,2]

M125 = [5,0,1,0,0,0,1,0,0,0,1,4,5,0]

M126 = [0,0,0,0,0,0,0,1,1,0,0,5,5,5]

M127 = [1,0,0,0,0,1,0,0,1,0,0,5,5,4]

M128 = [0,0,0,0,0,1,0,0,0,0,1,5,5,5]

M129 = [2,0,1,0,0,0,1,0,0,1,0,4,5,3]

M130 = [3,0,1,0,0,0,0,1,0,1,0,4,5,2]

M131 = [4,0,1,1,0,0,0,0,0,1,0,4,5,1]

M132 = [0,0,0,0,0,0,1,0,1,0,0,5,5,5]

M133 = [1,0,0,0,0,0,0,1,1,0,0,5,5,4]

M134 = [0,0,0,0,0,0,0,1,0,0,1,5,5,5]

M135 = [2,0,0,0,0,1,0,0,1,0,0,5,5,3]

M136 = [1,0,0,0,0,1,0,0,0,0,1,5,5,4]

M137 = [3,0,1,0,0,0,1,0,0,1,0,4,5,2]

M138 = [4,0,1,0,0,0,0,1,0,1,0,4,5,1]

M139 = [5,0,1,1,0,0,0,0,0,1,0,4,5,0]

M140 = [1,0,0,0,0,0,1,0,1,0,0,5,5,4]

M141 = [0,0,0,0,0,0,1,0,0,0,1,5,5,5]

M142 = [2,0,0,0,0,0,0,1,1,0,0,5,5,3]

M143 = [1,0,0,0,0,0,0,1,0,0,1,5,5,4]

M144 = [3,0,0,0,0,1,0,0,1,0,0,5,5,2]

M145 = [2,0,0,0,0,1,0,0,0,0,1,5,5,3]

M146 = [4,0,1,0,0,0,1,0,0,1,0,4,5,1]

M147 = [5,0,1,0,0,0,0,1,0,1,0,4,5,0]

M148 = [2,0,0,0,0,0,1,0,1,0,0,5,5,3]

M149 = [1,0,0,0,0,0,1,0,0,0,1,5,5,4]

M150 = [3,0,0,0,0,0,0,1,1,0,0,5,5,2]

M151 = [2,0,0,0,0,0,0,1,0,0,1,5,5,3]

M152 = [4,0,0,0,0,1,0,0,1,0,0,5,5,1]

M153 = [3,0,0,0,0,1,0,0,0,0,1,5,5,2]

M154 = [5,0,1,0,0,0,1,0,0,1,0,4,5,0]

M155 = [3,0,0,0,0,0,1,0,1,0,0,5,5,2]

M156 = [2,0,0,0,0,0,1,0,0,0,1,5,5,3]

M157 = [0,0,0,1,0,0,0,0,0,1,0,5,5,5]

M158 = [4,0,0,0,0,0,0,1,1,0,0,5,5,1]

M159 = [3,0,0,0,0,0,0,1,0,0,1,5,5,2]

M160 = [5,0,0,0,0,1,0,0,1,0,0,5,5,0]

M161 = [4,0,0,0,0,1,0,0,0,0,1,5,5,1]

M162 = [4,0,0,0,0,0,1,0,1,0,0,5,5,1]

M163 = [3,0,0,0,0,0,1,0,0,0,1,5,5,2]

M164 = [1,0,0,1,0,0,0,0,0,1,0,5,5,4]

M165 = [0,0,0,0,0,0,0,1,0,1,0,5,5,5]

M166 = [0,0,0,1,0,0,0,0,1,0,0,5,5,5]

M167 = [5,0,0,0,0,0,0,1,1,0,0,5,5,0]

M168 = [4,0,0,0,0,0,0,1,0,0,1,5,5,1]

M169 = [5,0,0,0,0,1,0,0,0,0,1,5,5,0]

M170 = [5,0,0,0,0,0,1,0,1,0,0,5,5,0]

M171 = [4,0,0,0,0,0,1,0,0,0,1,5,5,1]

M172 = [2,0,0,1,0,0,0,0,0,1,0,5,5,3]

M173 = [1,0,0,0,0,0,0,1,0,1,0,5,5,4]

M174 = [1,0,0,1,0,0,0,0,1,0,0,5,5,4]

M175 = [0,0,0,0,0,0,1,0,0,1,0,5,5,5]

M176 = [0,0,0,1,0,0,0,0,0,0,1,5,5,5]

M177 = [5,0,0,0,0,0,0,1,0,0,1,5,5,0]

M178 = [5,0,0,0,0,0,1,0,0,0,1,5,5,0]

M179 = [3,0,0,1,0,0,0,0,0,1,0,5,5,2]

M180 = [2,0,0,0,0,0,0,1,0,1,0,5,5,3]

M181 = [2,0,0,1,0,0,0,0,1,0,0,5,5,3]

M182 = [1,0,0,0,0,0,1,0,0,1,0,5,5,4]

M183 = [1,0,0,1,0,0,0,0,0,0,1,5,5,4]

M184 = [4,0,0,1,0,0,0,0,0,1,0,5,5,1]	M185 = [3,0,0,0,0,0,0,1,0,1,0,5,5,2]
M186 = [3,0,0,1,0,0,0,0,1,0,0,5,5,2]	M187 = [2,0,0,0,0,0,1,0,0,1,0,5,5,3]
M188 = [2,0,0,1,0,0,0,0,0,0,1,5,5,3]	M189 = [4,0,0,0,0,0,0,1,0,1,0,5,5,1]
M190 = [5,0,0,1,0,0,0,0,0,1,0,5,5,0]	M191 = [4,0,0,1,0,0,0,0,1,0,0,5,5,1]
M192 = [3,0,0,0,0,0,1,0,0,1,0,5,5,2]	M193 = [3,0,0,1,0,0,0,0,0,0,1,5,5,2]
M194 = [4,0,0,0,0,0,1,0,0,1,0,5,5,1]	M195 = [5,0,0,0,0,0,0,1,0,1,0,5,5,0]
M196 = [5,0,0,1,0,0,0,0,1,0,0,5,5,0]	M197 = [4,0,0,1,0,0,0,0,0,0,1,5,5,1]
M198 = [5,0,0,0,0,0,1,0,0,1,0,5,5,0]	M199 = [5,0,0,1,0,0,0,0,0,0,1,5,5,0]

Din analiza structurii arborelui de acoperire rezulta ca modelul obtinut in urma compunerii paralele dintre modelul nodului N3 (nod de tip 3) si specificatiile impuse nu este blocant.

In continuare se analizeaza nodul N5 (nod de tip 2) pentru verificarea existentei sau nu a blocajelor in urma introducerii specificatiilor de conducere. Structura subretelei considerata este prezentata in figura 3.39.

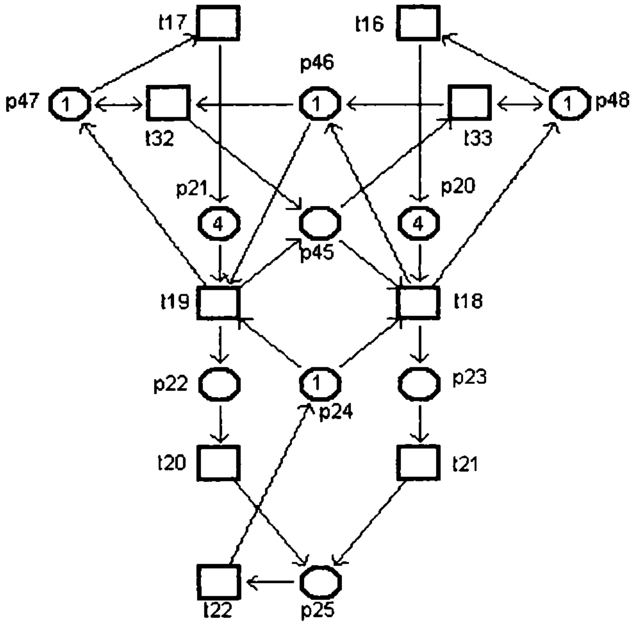


Figura 3.39. Structura modificata a nodului N5 - nod de tip 2

In tabelul 3.3 se prezinta structura arborelui de acoperire in mod text corespunzator nodului N5, modificat prin compunerea paralela intre un nod de tip 2 (nodul N5) si specificatiile de functionare impuse acestuia.

Tabelul 3.3 *Arborele de acoperire corespunzator nodului N5 modificat*

M0	t19	M1	**M1**	**t20**	**M2**	M2	t22	M3
M3	**t18**	**M4**	M4	t21	M5	**M5**	**t22**	**M6**
M6	t19	M7	**M7**	**t20**	**M8**	M8	t22	M9
M9	**t18**	**M10**	M10	t21	M11	**M11**	**t22**	**M12**
M12	t19	M13	**M13**	**t20**	**M14**	M14	t22	M15
M15	**t18**	**M16**	M16	t21	M17	**M17**	**t22**	**M18**
M18	t19	M19	**M19**	**t20**	**M20**	M20	t22	M21
M21	**t18**	**M22**	M22	t21	M23	**M22**	**t32**	**M24**
M23	t22	M25	**M23**	**t32**	**M26**	M24	t21	M26
M24	**t33**	**M22**	M25	t32	M27	**M26**	**t22**	**M27**
M26	t33	M23	**M27**	**t33**	**M25**			

M0 = [4,4,0,0,1,0,0,1,1,1]	M1 = [4,3,1,0,0,0,1,0,2,1]
M2 = [4,3,0,0,0,1,1,0,2,1]	M3 = [4,3,0,0,1,0,1,0,2,1]
M4 = [3,3,0,1,0,0,0,1,2,2]	M5 = [3,3,0,0,0,1,0,1,2,2]
M6 = [3,3,0,0,1,0,0,1,2,2]	M7 = [3,2,1,0,0,0,1,0,3,2]
M8 = [3,2,0,0,0,1,1,0,3,2]	M9 = [3,2,0,0,1,0,1,0,3,2]
M10 = [2,2,0,1,0,0,0,1,3,3]	M11 = [2,2,0,0,0,1,0,1,3,3]
M12 = [2,2,0,0,1,0,0,1,3,3]	M13 = [2,1,1,0,0,0,1,0,4,3]
M14 = [2,1,0,0,0,1,1,0,4,3]	M15 = [2,1,0,0,1,0,1,0,4,3]
M16 = [1,1,0,1,0,0,0,1,4,4]	M17 = [1,1,0,0,0,1,0,1,4,4]
M18 = [1,1,0,0,1,0,0,1,4,4]	M19 = [1,0,1,0,0,0,1,0,5,4]
M20 = [1,0,0,0,0,1,1,0,5,4]	M21 = [1,0,0,0,1,0,1,0,5,4]
M22 = [0,0,0,1,0,0,0,1,5,5]	M23 = [0,0,0,0,0,1,0,1,5,5]
M24 = [0,0,0,1,0,0,1,0,5,5]	M25 = [0,0,0,0,1,0,0,1,5,5]
M26 =	M27 =

[0,0,0,0,0,1,1,0,5,5]　　　[0,0,0,0,1,0,1,0,5,5]

Din analiza structurii arborelui de acoperire rezulta ca modelul obtinut in urma compunerii paralele dintre modelul nodului N5 (nod de tip 2) si specificatile impuse nu este blocant.

Deoarece submodele considerate nu introduc blocaje in functionarea in ansamblu a sistemului rezultat in urma compunerii paralele, se poate concluziona ca reteaua Petri *PN_Sist* (figura 3.35) reprezinta modelul de supervizor dorit.

3.4. Concluzii

In cadrul acestui capitol a fost abordat modul de realizare a supervizoarelor, atat din punct de vedere teoretic cat si aplicativ, printr-o serie de aplicatii si studii de caz,.

Tematica implementarii supervizoarelor a fost abordata pornind de la tipurile de modele utilizate. Au fost considerate aspectele teoretice referitoare la existenta si implementarea supervizoarelor, avand la baza modelarea cu ajutorul limbajelor, automatelor respectiv a retelelor Petri.

In partea intai, s-a urmarit prezentarea structurii de baza utilizata in conducerea supervizata a SED, introducandu-se notiunea de supervizor precum si locul, rolul si functiile acestuia in cadrul conducerii sistemelor automate complexe.

In partea a doua, s-a analizat problematica legata de conducerea supervizata bazata pe modele de tip automat descrise cu ajutorul limbajelor, fiind prezentate aspecte teoretice legate de supervizarea sistemelor cu reactie dupa stare, respectiv aspecte legate de sinteza a doi algoritmi de implementare a supervizoarelor bazate pe modele de tip automat. Exemplificarea modului de implementare a unui supervizor, bazat pe modele de tip automat, s-a realizat printr-un studiu de caz legat de conducerea supervizata a unui SFFF utilizat in filaturile de bumbac, utilizand algoritmului 3.2.

In partea a treia, s-a analizat problematica legata de conducerea supervizata bazata pe utilizarea modelelor de tip retele Petri, fiind prezentate doua metode de implementare, una bazata pe utilizarea invariantilor de tip P, iar cealalta bazata pe compunerea paralela a modelelor de tip retele Petri. Validarea acestor metode s-a realizat printr-o serie de exemple si un studiu de caz legat de implementarea unui supervizor pentru un STZA.

Rezultatele obtinute in cazul implementarii supervizoarelor bazate pe modele de tip automat, conduc la concluzia ca utilizarea acestei metode, este foarte laborioasa, ea devenind aproape inoperabila in cazul sistemelor complexe (compunand un sistem cu 10 componente cu un sistem cu 10 stari rezulta un sistem cu 10^{10} stari).

In cazul utilizarii retelelor Petri, datorita faptului ca un sistem complex nu este descris prin extinderea topologiei retelei ci prin extinderea marcajelor, problematica legata de implementarea supervizoarelor, bazate pe astfel de modele, se simplifica semnificativ.

Pe baza rezultatelor obtinute, metodele propuse pentru implementarea supervizoarelor (automate, respectiv retele Petri), au fost confirmate si validate.

In capitolul 6 se prezinta in detaliu contributiile legate de conducerea supervizata a SED.

Capitolul 4

Utilizarea modelelor sistemelor cu evenimente discrete in implementarea sistemelor de conducere in timp real

4.1. Consideratii generale ale proiectarii programelor de conducere in timp real [Letia00][HP97]

Sistemele de conducere in timp real (SCR) depind in mod decisiv de „viteza" de lucru a sistemelor de calcul, timpul de raspuns in acest caz fiind esential. Performantele sistemelor de conducere in timp real depind de doi factori esentiali:

- performantele echipamentelor hardware si software;
- structura modelului stabilit pentru conducerea DES.

Caracteristicile sistemelor de conducere in timp real sunt sintetizate dupa cum urmeaza:

- utilizarea explicita a timpului, prin asigurarea corealrii activitatilor de conducere cu timpul;
- stabilirea unor intervale clare de timp necesare efectuarii tuturor activitatilor care trebuie efectuate;
- depasirea intervalului de timp specificat activitatilor care trebuie executate implica lansarea in executie a altor activitati speciale numite „exceptii";
- asigurarea concurentei programelor (taskurilor) prin utilizarea mecanismelor de comutare;
- reactia SCR la stimuli externi se realizeaza prin utilizarea metodelor specifice de masura si control;
- aspectele legate de integrarea SCR sunt mult mai critice decat in cazul altor categorii de sisteme de calcul;
- pentru implementarea SCR trebuie alese limbajele de programare dedicate conducerii in timp real.

Elementele care trebuie luate in considerare in cazul implementarii unor sisteme de conducere in timp real sunt:

- *viteza de raspuns a sistemului de calcul*;
- *utilizarea sistemele multitasking;*
- *asigurarea sincronizarii componentelor cu timpul fizic si intre ele;*
- *determinismul*;
- *adaptabilitatea si flexibilitatea programelor de conducere*;
- *stabilitatea programelor de conducere*.

Fazele dezvoltarii unui program de conducere in timp real sunt:

- *specificarea cerintelor functionale si structura hardware a sistemului de calcul* – se precizeaza functiile si sarciniile pe care sistemul de calcul trebuie sa le indeplineasca astfel incat sa poata fi asigurata conducerea procesului vizat, stabilindu-se si structura hardware (memorie, module de achizitie si control etc.) corespunzatoare.

- *stabilirea algoritmilor de conducere utilizati* – stabilirea algoritmilor de conducere tinand cont de tipul aplicatiei, respectiv de cerintele functionale.
- *stabilirea cerintelor software ale sistemului de calcul* - se stabileste limbajul de programare utilizat pentru implementare tinand cont de structura hardware stabilita;
- *proiectarea logica a programelor de conducere* - analizand modelul sistemului de conducere, se stabilesc numarul si activitatiile taskurilor, definindu-se interfetelele si mecanismele de sincronizare dintre ele, tinand cont de viteza de lucru si capacitatea de memorie a sistemului de calcul.
- *codificarea si depanarea programelor de conducere* – se realizeaza utilizand un limbaj de programare adecvat, respectiv un compilatorul corespunzator. Aceasta faza se poate realiza si pe un alt sistem de calcul decat cel dedicat conducerii;
- *instalarea, testarea, evaluarea si masurarea performantelor sistemului de conducere* – in aceasta faza programele de conducere elaborate sunt instalate pe sistemul de calcul dedicat conducerii, urmarindu-se testarea in conditii reale a programelor de conducere.

In literatura de specialitate [LJ06][Glei03][HL04][YGG03][KSH05] acest subiect al implementarii programelor de conducere in timp real este tratata pe larg propunandu-se o mare varietate de solutii care depind atat de sistemul de calcul ales cat si de sistemul de operare folosit. In prezent se incearca o standardizare a acestor solutii [Stand04][Stand05] [GG06].

In cadrul acestei lucrari sunt abordate doua metode de implementare:

- bazata pe utilizarea unui sistem de calcul industrial, echipat cu interfetele aferente conducerii unui proces industrial (modul de achizitii de tip INTERBUS) pe care ruleaza sistemul de operare in timp real (SOTR) QNX, ca mediu de dezvoltare fiind utilizat limbajul WATCOM C.
- bazata pe utilizarea automatelor programabile (PLC) de tip Siemens SIMATIC S7-414, ca mediu de dezvoltare fiind utilizat limbajul STEP 7.

Deoarece in capitolele 2 si 3 au fost tratate doua metode de modelare a sistemelor cu evenimente discrete (utilizand modele de tip automat, respectiv utilizand modele de tip retele Petri) abordarea modului de implementare a programelor de conducere se face corespunzator celor doua directii. Astfel, pentru modelele de tip automat a fost utilizata pentru implementare prima metoda) iar pentru modelele de tip retele Petri a fost utilizata a doua metoda).

4.2. Implementarea modelelor de tip automat in sistemul de operare in timp real QNX

4.2.1. Capabilitatile SOTR QNX [QNX00]

Implementarea modelelor de tip automat se bazeaza pe capabilitatile oferite de catre SOTR QNX.

Pentru a putea prezenta modul de implementare a modelelor de tip automat, trebuie facute, in prealabil cateva precizari legate de modul de lucru al SOTR QNX.

4.2.1.1. Mecanismele de comunicatie interprocese

SOTR QNX pune la dispozitia utilizatorului o serie de mecanisme, extrem de puternice, prin intermediul caroara se poate asigura comunicatia intre taskuri. Aceste mecanisme sunt sintetizate in cele ce urmeaza:

- comunicatie intertaskuri prin mesaje;
- comunicatie intertaskuri prin intermediul proxies;
- comunicatie itertaskuri prin intermediul semnalelor;
- comunicatie intertaskuri prin intermediul semafoarelor.

Pe langa aceste mecanisme, SOTR QNX mai pune la dispozitie un mecanism de transfer al datelor prin intermediul memoriei partajate (shared memory).

În QNX, un mesaj este constituit dintr-un pachet de octeţi care sunt transmişi sincron de la un proces la altul. Pe parcursul comunicării, la conţinutul mesajului transmis nu se ataşează nimic, acesta având sens numai pentru emiţător şi receptor.

Pentru construirea unui canal de comunicaţie directă între procese, utilizand limbajul WATCOM C ca mediu de dezvoltare, se utilizează următoarele funcţii:

- *Send(...)* – pentru emiterea de mesaje către receptor;
- *Receive(...)* – pentru recepţionarea mesajelor emise;
- *Reply(...)* – pentru confirmarea primirii de către receptor a mesajului transmis de către emiţător.

Aceste trei funcţii au un caracter general de valabilitate, putând fi utilizate atât pentru comunicaţiile locale între procese, cât şi pentru comunicaţiile între procese care rulează pe noduri diferite (in retea). Funcţiile *Send(), Receive(), Reply()* nu sunt necesare în cazul în care nu se doreşte ca procesul să comunice direct cu un alt proces.

• *Funcţia Send()*

Această funcţie este folosită în cazul în care se doreşte transmiterea unui mesaj între procese. Prototipul funcţiei este:

Send(pid, smsg, rmsg, smsg_len, rmsg_len).

Utilizarea funcţiei implică stabilirea unor valori pentru argumente, *pid* - reprezintă ID-ul procesului căruia îi este destinat mesajul. Orice proces este cunoscut de către sistemul de operare şi de către alt proces după identificatorul *pid*. *Send()*, fiind o funcţie de transmitere a unui mesaj, are un câmp destinat mesajului.

Conţinutul mesajului se găseşte în bufferul mesajului, cunoscut prin *smsg*. În urma recepţionării unui mesaj, procesul destinaţie trimite la rândul lui un mesaj de răspuns prin funcţia *Reply()* care va fi recepţionat în bufferul desemnat de *rmsg*. În momentul transmiterii unui mesaj de către un proces, acesta va transmite pe lângă mesaj şi lungimea acestuia prin argumentul *smsg_len*. Lungimea mesajului de răspuns nu poate să depăşească lungimea maximă impusă de emitor prin argumentul *rmsg_len* al funcţiei *Send()*.

• *Funcţia Receive()*

Un proces este pregătit să recepţioneze un mesaj după executarea funcţiei *Receive()*.

Condiţiile în care un mesaj poate fi recepţionat sunt impuse prin intermediul parametrilor funcţiei. Prototipul funcţiei este prezentat în continuare:

Receive(pid, 0, msg, msg_len).

Argumentul *pid* conţine ID-ul procesului care a transmis mesajul, cunoscându-se astfel locul unde trebuie transmis răspunsul de confirmare al recepţionării mesajului. Dacă al doilea argument are valoarea 0 este precizat faptul că se acceptă mesaje de la orice proces. Argumentul *msg* conţine adresa bufferului unde mesajul transmis va fi recepţionat de către procesul receptor. Lungimea maximă a mesajului care va putea fi preluat în bufferul de recepţie al unui proces este impusă prin intermediul argumentului *msg_len*.

- ***Funcţia Reply()***

Această funcţie este folosită pentru confirmarea către un proces că mesajul transmis de el a fost recepţionat. Funcţia este executată de către procesul receptor. Prototipul funcţiei este următorul:

Reply(pid, reply, reply_len).

Primul argument, *pid*, reprezintă ID-ul procesului căruia trebuie să-i fie transmis mesajul de răspuns şi confirmare a recepţiei. Bufferul mesajului de răspuns este precizat prin intermediul argumentului *reply*. Mesajele transmise ca răspuns nu pot avea o lungime mai mare ca cea impusă cu ajutorul argumentului *reply_len*.

Între cele trei funcţii se stabileşte o regulă pe baza lungimii mesajului transmis şi recepţionat. Mesajul transmis de emitor va avea lungimea precizată prin argumentul *smsg_len* al funcţiei *Send()*, dar receptorul va prelua mesaje de lungimea precizată de argumentul *msg_len* al funcţiei *Receive()*. Dimensiunea mesajului de confirmare al recepţionării este dată prin argumentul *reply_len* al funcţiei *Reply()*. Acest mesaj este destinat procesului emitor, care a impus prin argumentul *rmsg_len* al funcţiei *Send()* care este dimensiunea mesajului aşteptat la confirmarea recepţiei.

Folosirea celor trei funcţii este cunoscută sub denumirea de mecanismul send-receive-reply. În figura 4.1. este prezentată o secvenţă simplă a evenimentelor care au loc în cazul utilizării mecanismului send-receive-reply pentru a asigura comunicaţia între două procese A şi B.

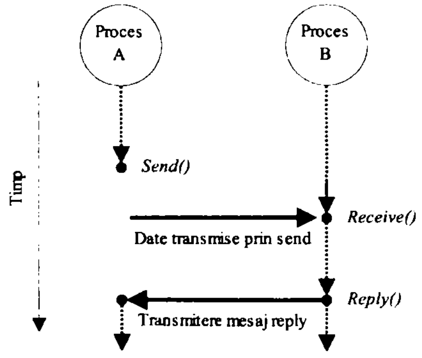


Figura 4.1. Comunicaţia simplă între două procese A şi B utilizând mecanismul send-receive-reply

Etapele prin care se trece în executarea acestui mecanism sunt:

- Procesul A considerat emitor trimite un mesaj către procesul B considerat receptor şi care va fi identificat prin argumentul *pid* al funcţiei *Send()*. Ca urmare, se emite o cerere *Send()* către microkernel. Mesajul transmis se va regăsi în bufferul desemnat prin argumentul *smsg* şi va fi destinat procesului receptor. Procesul A va fi blocat în emitere (send-blocked) până în momentul în care procesul B execută funcţia *Receive()* şi astfel recepţionează mesajul.
- După ce procesul B execută funcţia *Receive()* şi recepţionează mesajul în bufferul desemnat prin argumentul *msg* al funcţiei, procesul A va intra în starea blocat la răspuns (reply-blocked). Până în momentul recepţionării mesajului transmis de procesul A, procesul B nu este blocat. Dacă procesul B execută funcţia *Receive()* înainte de transmiterea mesajului, de către procesul A, el intră în starea blocat la recepţie (receive-blocked) până la sosirea mesajului. În acest caz procesul A va intra şi el direct în starea blocat la răspuns (reply-blocked) după transmiterea mesajului.
- În urma recepţionării mesajului transmis de procesul A, procesul B emite un răspuns prin intermediul funcţiei *Reply()*. Procesul căruia îi este destinat mesajul de răspuns este cunoscut după ID-ul precizat de argumentul *pid* al funcţiei. Mesajul de răspuns al procesului B este transmis procesului A care părăseşte starea de blocare şi devine apt de rulare. Funcţia *Reply()* nu conduce la blocarea procesului care a executat-o, astfel încât şi procesul B este apt de rulare. Procesul care va intra în rulare depinde de prioritatea relativă asociată lui.

4.2.1.2. Sincronizarea proceselor

Din ansamblul activităţilor pe care le comportă un proces, unele sunt legate de anumite evenimente realizate de alte procese. Astfel, este necesară sincronizarea acţiunilor proceselor. Deşi message-passing nu este singura metodă prin care datele pot fi transferate între procese, ea este cea mai bună metodă prin care se poate asigura sincronizarea operaţiilor executate pentru mai multe procese cooperante. Sincronizarea proceselor se realizează prin procese de blocare şi deblocare.

Dacă un proces A emite o cerere *Send()*, acesta va trece într-o stare de blocare până la recepţia mesajului de răspuns generat de procesul B către care s-a efectuat transmisia. Astfel, procesului A îi este confirmat faptul că procesul receptor al mesajului a primit mesajul său. Este evitată în acest fel efectuarea de operaţii fără acordul altor procese implicate. Pe de altă parte, un proces B poate fi blocat, prin intermediul funcţiei *Receive()*, până la recepţionarea unui mesaj de la un alt proces. Prin metoda message-passing se menţine ordinea în cadrul executării acţiunilor mai multor procese.

4.2.1.3. Stările proceselor în cazul utilizării comunicatiei interprocese prin intermediul mesajelor

Orice proces se poate afla într-una din stările blocat sau neblocat. Un proces se spune că este blocat când acesta nu este apt să-şi continue execuţia deoarece trebuie să aştepte terminarea unei secvenţe a protocolului de comunicaţie. Secvenţele protocolului de comunicaţie sunt determinate în raport cu cele trei funcţii, *Send()*, *Receive()*, *Reply()*, cunoscându-se trei stări de blocare: blocat la transmisie (send-blocked), blocat la recepţie (receive-blocked) şi blocat la răspuns (reply-blocked).

Un proces este blocat la transmisie atât timp cât în urma cererii *Send()*, a transmis un mesaj care nu a fost încă recepţionat de către procesul destinaţie. Un proces este blocat la răspuns în urma efectuării cererii *Send()*, atât timp cât procesul destinaţie a recepţionat mesajul, dar nu a răspuns încă. Un proces este blocat la recepţie dacă în urma efectuării unei cereri *Receive()* încă mai aşteaptă mesajul.

În figura 4.2 este prezentată diagrama de stări pentru tranziţiile mecanismului send-receive-reply.

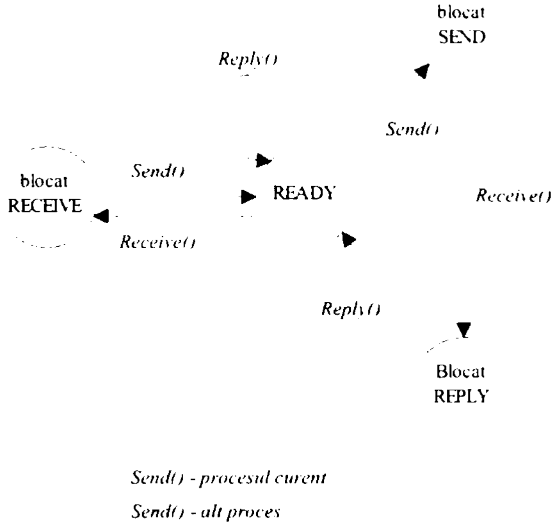


Figura 4.2. Diagrama de stări pentru tranzacţii *send-receive-reply*

4.2.1.4. Comunicatia intre procese prin intermediul memoriei partajate

Dupa cum rezulta din prezentarea mecanismelor de transfer intre taskuri (paragraful 4.2.1.1) prin utilizarea mecanismului *Send - Reply* volumul datelor vehiculate este relativ mic. Marirea volumului de date transferate, se poate rezolva, in principiu, pe doua cai:

- prin utilizarea fisierelor temporale - problemele care pot sa apara sunt legate de viteza de lucru, care depinde in cea mare masura de viteza de lucru a echipamentelor periferice (*discul fix - hard-discul*);
- prin utilizarea memoriei partajate - SHM- *shared memory* -

Solutia abordata in continuare se bazeaza pe utilizarea mecanismului de trasfer al datelor prin SHM, solutie care se considera ca fiind cea mai eficienta.

Acest mecanism se bazeaza pe definirea in memoria sistemului a unei zone speciale, care reprezinta in fapt o zona de memorie tampon, utilizata de taskuri pentru depunere (scriere) respectiv preluare (citire) de date. In figura 4.3 se prezinta structura unui sistem multitasking care utilizeaza SHM.

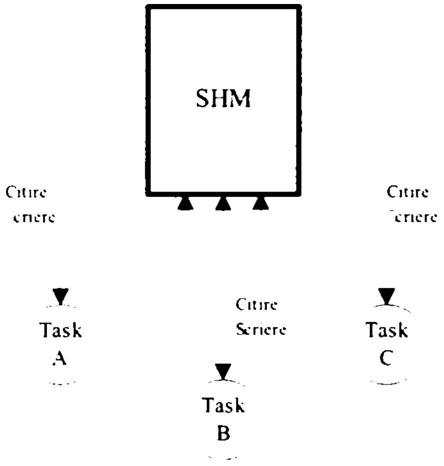


Figura 4.3. Structura SHM intr-un sistem multitasking

In cazul utilizarii SHM trebuie facuta precizarea ca structurile de date folosite (tipurile datelor) trebuie sa fie aceleasi pentru toate taskurile care au acces la SHM considerata.

Intr-o aplicatie este posibila utilizarea mai multor zone SHM diferite, numerele si dimensiunile sunt limitate numai de capacitatea memoriei sistemului de calcul. In cazul unei aplicatii care utilizeaza mai multe zone SHM, accesul taskurilor la zonele declarate nu este limitat, astfel incat un task poate avea acces la toate zonele SHM declarate. In figura 4.4 se prezinta un sistem multitasking cu mai multe zone SHM.

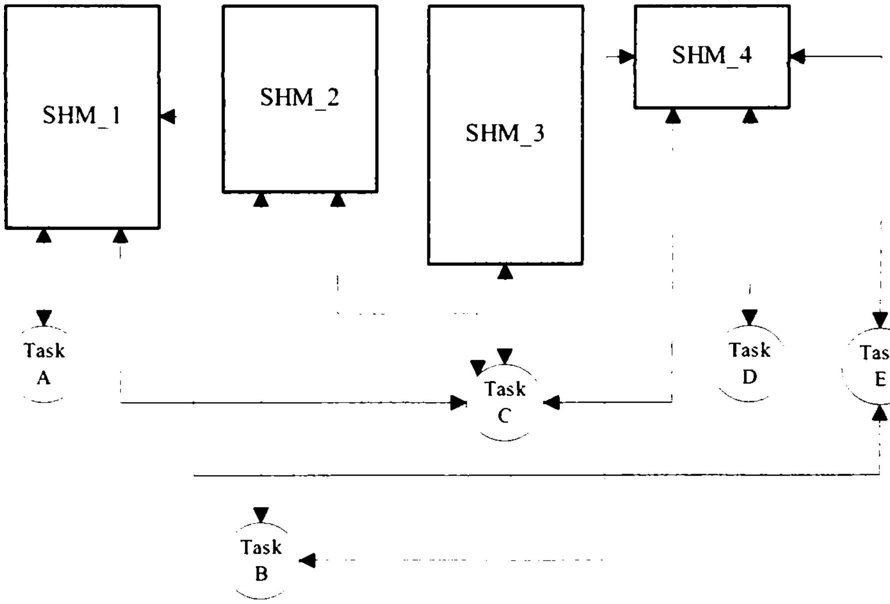


Figura 4.4. Sistem multitasking cu mai multe zone SHM

Pentru operarea cu SHM mediul de dezvoltare WATCOM C pune la dispozitie o serie de functii specifice [QNX_C05]. In tabelul 4.1 sunt prezentate functile utilizate si rolul lor in cazul operarii cu SHM.

Tabelul 4.1. *Functiile corespunzatoare operarii cu SHM*

Nume Functie	Comentariu
shm_open	Functie cu ajutorul careia se creaza o zona SHM sau se asigura conectarea unui task la aceasta. O zona SHM este creata o singura data de catre un task, celelalte taskuri avand doar posibilitatea de conectare.
ltrunc	Functie cu ajutorul careia se seteaza dimensiunea zonei SHM create cu *shm_open*. Aceasta operatie este executata o singura data in momentul creerii zonei SHM
mmap	Functie cu ajutorul careia se mapeaza zona SHM. Valoarea returnata de aceasta functie este un pointer la zona de inceput al SHM. Aceasta functie este utilizata de toate taskurile care au acces la zona SHM.

4.2.1.3. Dispecerizarea proceselor

Dispecerul kernelului SOTR QNX ia decizii in situatia in care:

- un proces devine apt pentru rulare
- perioada de timp alocată unui proces aflat în rulare expiră
- un proces aflat în rulare pierde procesorul.

În sistemul de operare QNX fiecărui proces îi este asignată o prioritate. Pentru citirea şi setarea priorităţii unui proces se utilizează funcţiile: *getprio()*, respectiv *setprio()*. Priorităţile asignate proceselor iau valori de la 0 (prioritatea cea mai scăzută) la 31 (prioritatea cea mai ridicată), unui proces nou creat atribuindu-se implicit prioritatea de valoare 10.

Pentru exemplificarea modurilor de dispecerizare, se prezintă cazul în care şase procese (A, B, C, D, E şi F) se află în starea ready, iar restul proceselor (G...Z) se află în starea blocat (vezi figura 4.5). Procesele au priorităţi diferite, procesul A aflandu-se în rulare. După finalizarea acestui proces se trece la realizarea celorlalte procese, în ordinea în care sunt găsite în coada de aşteptare. În cazul considerat urmeaza procesul B, apoi procesele C, D, E şi F. Chiar dacă procesele E şi F au o prioritate mai mică decât procesele G...Z, se vor executa înaintea lor, deoarece acestea din urmă se află în starea blocat. În figura 4.5 este prezentată dispecerizarea proceselor din cazul considerat.

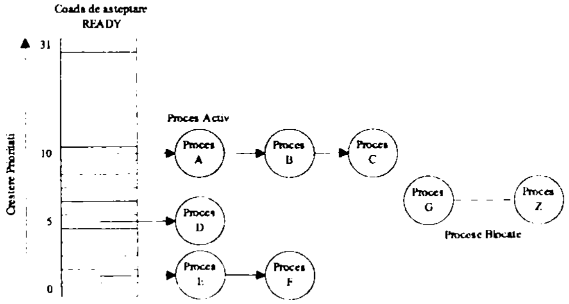


Figura 4.5. Exemplu de dispecerizarea proceselor

Există trei tipuri de baza de dispecerizare pe care kernelul le foloseşte în funcţie de nevoile aplicaţiilor care rulează:

- Dispecerizarea de tip FIFO
- Dispecerizarea prin rotaţie
- Dispecerizarea adaptivă.

Aceste metode de dispecerizare se aplică in cazul in care două sau mai multe procese care au aceeaşi prioritate se află în starea ready. Dacă un proces cu o prioritate mai mare trece în starea ready, acesta va intra în rulare, eliminând procesele cu priorităţi mai mici decât a lui.

În figura 4.6 se prezintă situaţiile unor procese de prioritate egală, aflate în starea ready.

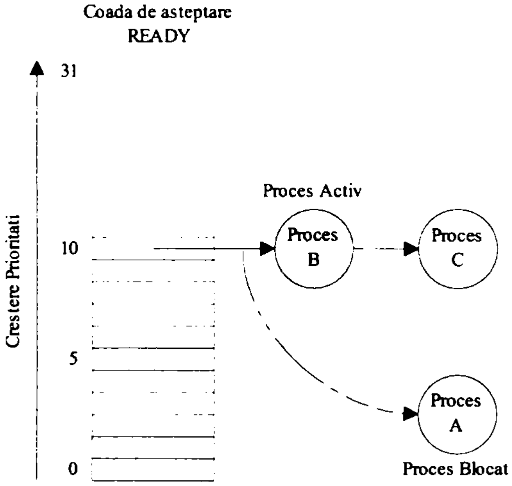


Figura 4.6. Dispecerizarea proceselor de aceeasi prioritate

Procesele A, B şi C au aceeaşi prioritate. Dacă toate se află în starea *ready*, primul proces care va fi rulat va fi procesul A. Dacă procesul A se blochează, în continuare va fi rulat procesul B, apoi procesul C şi in continuare celelalte procese de prioritate mai mică (dacă există).

Există posibilitatea de a afla tipul dispecerizării active la un moment dat prin utilizarea funcţiei *getscheduler()*, sau se poate chiar opta pentru o anumită formă de dispecerizare prin utilizarea funcţiei *setscheduler()*.

- **Dispecerizarea de tip FIFO.** Când este activă această formă de dispecerizare, un proces care a ajuns la procesor este rulat în continuare până când cedează controlul altui proces în mod voluntar, sau până când este înlocuit de un proces mai prioritar (vezi figura 4.7). Două procese care au aceeaşi prioritate pot folosi dispecerizarea de tip FIFO pentru a asigura excluderea mutuală la o resursă partajată. Nici un proces nu va fi înlocuit de un alt proces atât timp cât el se află în rulare, indiferent de priorităţile lor.De exemplu, dacă două procese împart un segment de memorie, fiecare dintre ele pot utiliza aceast segment fără a apela vre-un semafor care să asigure excluderea mutuală.

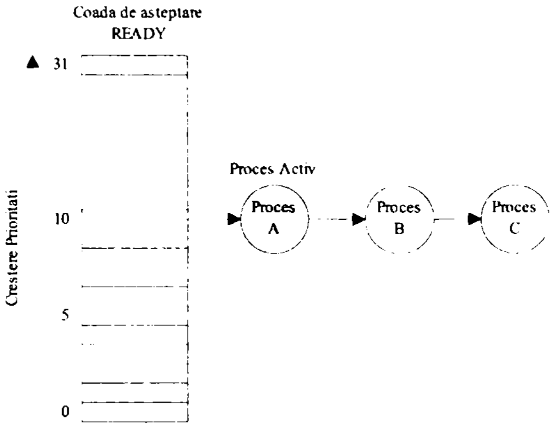


Figura 4.7. Dispecerizarea proceselor prin metoda FIFO

- **Dispecerizarea prin rotaţie.** Această formă de dispecerizare presupune existenţa unei liste unice de aşteptare la procesor, în care sunt înscrise toate procesele aflate în starea ready la un moment dat (vezi figura 4.8). Dispecerul alege întotdeauna procesul pe care-l găseşte în prima poziţie a listei, iar după procesul este trecut în coada listei.

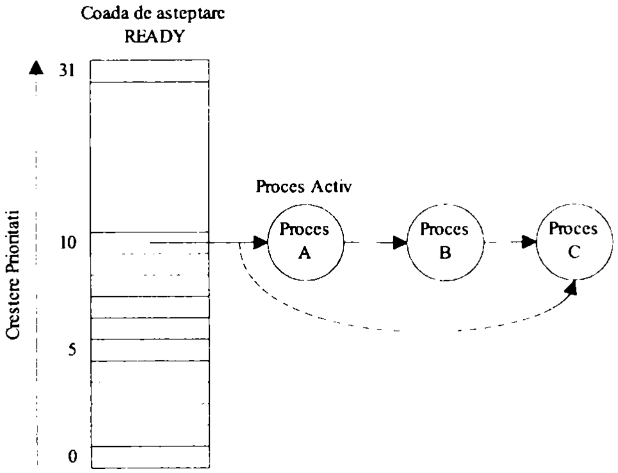


Figura 4.8. Dispecerizarea proceselor prin rotatie

- **Dispecerizarea adaptivă.** Se caracterizeaza prin urmatoarele:
    - Când un proces îşi consumă perioada de timp alocată, prioritatea lui este decrementată cu un nivel, dacă există alt proces cu aceeaşi prioritate aflat în starea ready.
    - Dacă procesul a cărui prioritate a decăzut rămâne nedispecerizat timp de o secundă, prioritatea lui este incrementată cu un nivel, dar nu se va efectua niciodată o ridicare a priorităţii peste nivelul original.
    - În cazul în care procesul se blochează, acesta este adus la prioritatea originală.

Dispecerizarea adaptivă (vezi figura 4.9) este indicată pentru medii în care procesele din background-ul sistemului împart resursele calculatorului cu procese server interactive. Această metodă de dispecerizare este activă pentru programele create prin *shell*.

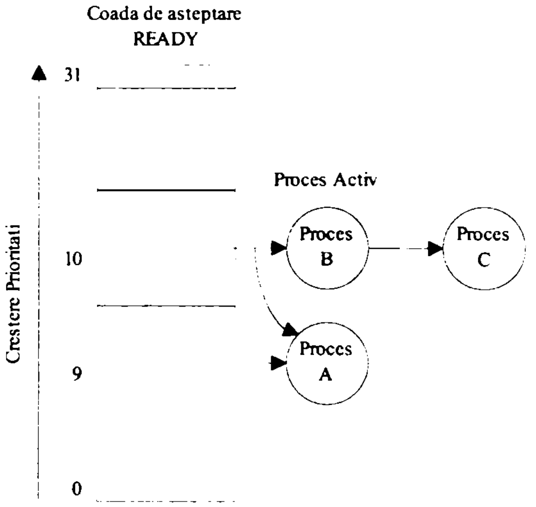


Figura 4.9. Dispecerizarea proceselor prin metoda adaptiva

4.2.1.6. Starile unui proces

Un proces se afla intotdeauna intr-una din urmatoarele stari:

- READY - procesul este capabil sa utilizeze unitatea centrala de prelucrare (UCP) - nu este in asteptarea unui eveniment ci asteapta eliberarea procesorului;
- BLOCAT - procesul se afla in una din urmatoarele stari blocante:
  - blocat de SEND;
  - blocat de RECEIVE;
  - blocat de REPLY;
  - blocat de SIGNAL
  - blocat de SEMAPHORE.
- HELD - procesul a receptionat un semnal de tip SIGSTOP. Pana cand procesul nu este scos din starea HELD el nu va putea prelua controlul asupra procesorului (nu va intra in rulare). Singura metoda de scoatere din starea HELD este aceea de receptionare a un semnal SIGCONT sau de terminare a procesului prin intermediul unui semnal specific;
- blocat de WAIT - procesul a executat o functie de tipul *wait()* sau *waitpid()* pentru a astepta primirea unor informatii despre starea unuia sau mai multor procese de tip *child*.
- DEAD - procesul este incheiat, dar nu poate trimite starea corespunzatoare procesului parinte deoarece acesta din urma nu a executat o functie *wait()* sau *waitpid()*. Un proces DEAD are o stare activa, memoria alocata lui nefiind eliberata. Un proces DEAD se mai numeste si proces *zombie*.

In figura 4.10 se prezinta starile posibile si mecanismele de comutare a starilor corespunzatoare unui proces in SOTR QNX.

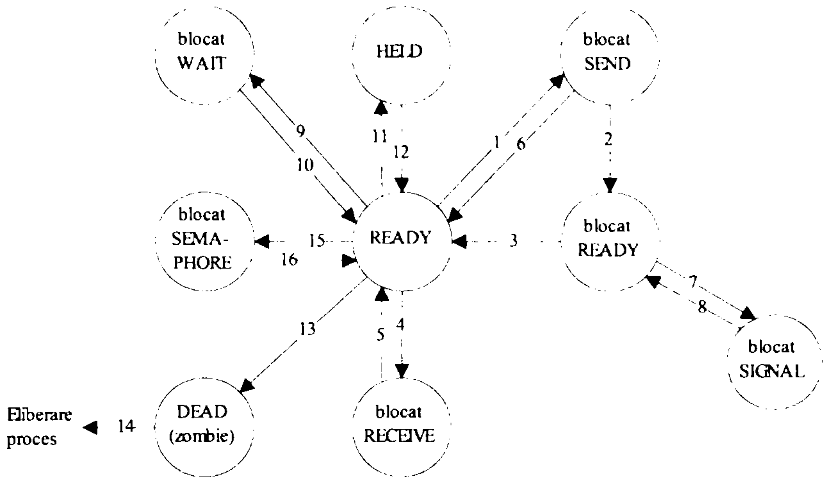


Figura 4.10 Starile posibile ale unui proces in SOTR QNX

Tranzactile din figura 4.10 sunt urmatoarele:

1. Procesul trimte un mesaj.
2. Procesul destinatar a receptionat mesajul.
3. Procesul destinatie trimte mesajul *reply* de confirmare a receptiei.
4. Procesul asteapta un mesaj.
5. Procesul a receptionat un mesaj.
6. Semnal de deblocare a procesului.
7. Semnal de incercare a deblocarii procesului; destinatarul a cerut „prinderea" unui semnal.
8. Procesul destinatie a receptionat semnalul.
9. Procesul asteapta terminarea unui proces *child*.
10. Procesul *child* terminat, sau receptionarea unui semnal de deblocare.
11. Aparitie semnal SIGSTOP.
12. Aparitie semnal SIGCONT.
13. Proces terminat
14. Procesul parinte asteapta terminarea, sau procesul in cauza s-a incheiat singur.
15. Procesul apeleaza functia *semwait()* pentru determinarea unei stari negative corespunzatoare unui semafor.
16. Un alt proces apeleaza functia *sempost(),* sau a fost receptionat un semnal nemascat.

4.2.2. Implementarea in SOTR QNX a unui STZA pe baza modelelor de tip automat [Ung01][Ung02][Ung05][Ung06a]

Se considera un STZA a carui structura este prezentata in figura 4.11.

Sistemul este constituit din 5 noduri conectate intre ele, nodurile N1, N3 si N4 sunt noduri de tip 1, nodul N2 este nod de tip 4 si nodul N5 este nod de tip 2.

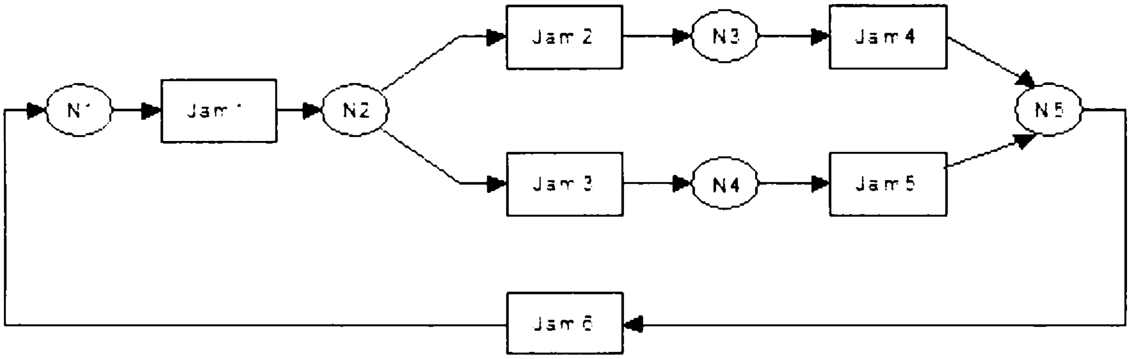


Figura 4.11. Structura STZA considerat

Fiecare nod este conectat la numarul de jam-uri corespunzatoare. Structura de date a unui jam este prezentata in tabelul 4.2.

Tabelul 4.2. *Structura de date ale unui jam*

Nume	Tip	Comentariu
JAM_ID	STRING	Identificatorul jam-ului
JAM_Cap	INT	Capacitatea maxima a jam-ului
JAM_Count	INT	Numarul curent de elemente din jam

Structura de date corespunzatoare declarata in WATCOM_C [WATC00] este:

```
//structura corespunzatoare a unui Jam
typedef struct{
        char Jam_ID[10];
        int  Jam_Val;
        int  Jam_Count;
}Jam;
```

Modul de conectare a taskurilor este prezentata in figura 4.12.

Pentru fiecare tip de nod utilizat a fost implementat un task corespunzator. Astfel, pentru nodul de tip 1 a fost dezvoltat taskul NT1, pentru nodul de tip 2 taskul NT2 iar pentru nodul de tip 4 taskul NT4. Intr-o prima etapa se urmareste modul de implementare a mecanismelor de comunicatie intertaskuri. Trebuie facuta precizarea ca in cadrul sistemului, taskul NT1 este utilizat de trei ori fara a avea trei programe distincte. In SOTR QNX vor rula trei taskuri de tip NT1, dar aceste taskuri sunt parametrizate individual prin linia de comanda de lansare in executie, dupa cum urmeaza:

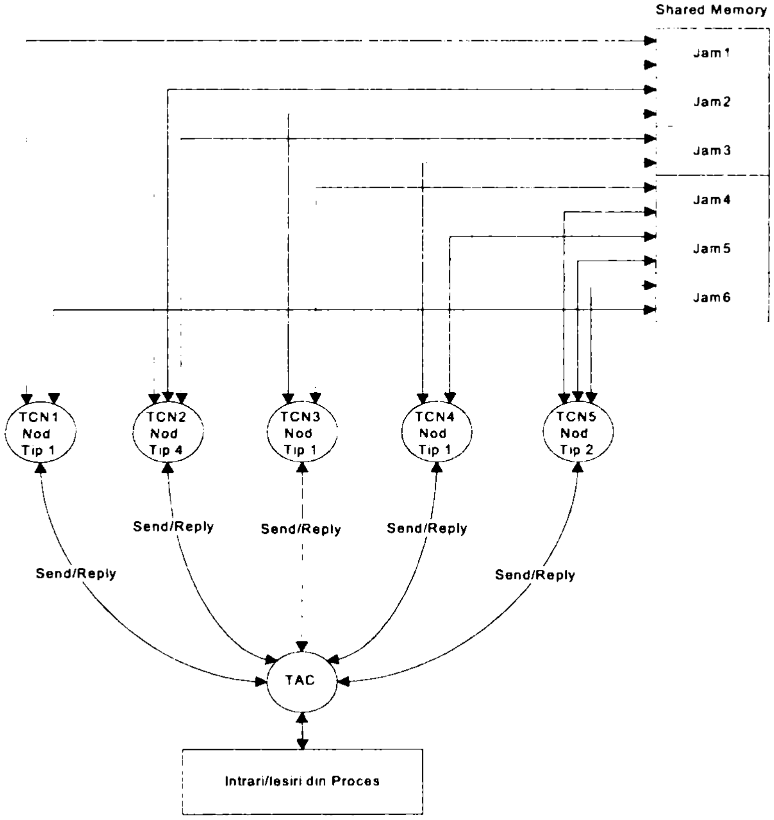
NT1 *nume_nod jam_in jam_out*

unde:

- NT1 - reprezinta taskul corespunzator nodului de tip 1;
- *nume_nod* - este numele atasat nodului corespunzator pentru identificarea acestuia in sistem. Pentru nodul 1 numele atasat este *nod1*, pentru nodul 3 *nod3* iar pentru nodul 4 *nod4*.

- *jam_in* - numele jamului de intrare. Pentru nodul 1, *Jam6*, pentru nodul 3, *Jam2* si pentru nodul 4, *Jam3*.
- *jam_out* - numele jamului de iesire. Pentru nodul 1, *Jam1*, pentru nodul 3, *Jam4* si pentru nodul 4, *Jam5*.

Existenta in sistem a mai multor taskuri de acelasi tip (de ex. NT1), nu ridica probleme din punct de vedere al rularii deoarece fiecare task este rulat separat, la lansarea in executie ele primind un *pid* corespunzator.



TAC - task achizitii de date/ transmitere de comenzi
TCN1 - task control nod N1 - Nod de Tipul 1 - Functie de tip NT1
TCN2 - task control nod N2 - Nod de Tipul 4 - Functie de tip NT4
TCN3 - task control nod N3 - Nod de Tipul 1 - Functie de tip NT1
TCN4 - task control nod N4 - Nod de Tipul 1 - Functie de tip NT1
TCN5 - task control nod N5 - Nod de Tipul 2 - Functie de tip NT2

Figura 4.12. Modul de conectare a taskurilor si transferul informatiei

In cazul nodului 2, nod de tip 4, implementarea s-a realizat in taskul NT4, parametrizarea taskului fiind facuta prin intermediul liniei de comanda:

NT4 *nume_nod jam_in jam_out1 jam_out2*

unde:

- NT4 - reprezinta taskul corespunzator nodului de tip 4;
- *nume_nod* - este numele atasat nodului corespunzator pentru identificarea acestuia in sistem. Pentru nodul 2 numele atasat este *nod2*.
- *jam_in* - numele jamului de intrare, in acest caz *Jam1* .
- *jam_out1* - numele primului jam de iesire, in acest caz *Jam2* .

- *jam_out2* - numele celui de al doilea jam de iesire. In acest caz *Jam3* .

In cazul nodului 5 nod de tip 2, implementarea s-a realizat in taskul NT2, parametrizarea taskului fiind facuta prin intermediul liniei de comanda:

NT2 *nume_nod jam_in1 jam_in2 jam_out*

unde:

- NT2 - reprezinta taskul corespunzator nodului de tip 2;
- *nume_nod* - este numele atasat nodului corespunzator pentru identificarea acestuia in sistem. Pentru nodul 5 numele atasat este *nod5*.
- *jam_in1* - numele primului jam de intrare, in acest caz *Jam4* .
- *jam_in2* - numele celui de al doilea jam de intrare. In acest caz *Jam5*.
- *jam_out* - numele jamului de iesire, in cazul considerat *Jam6* .

Pentru achizitia de date, respectiv generarea comenzilor (citirea starilor senzorilor respectiv generarea comenzilor catre elementele de executie), s-a implementat un task separat denumit TAC. Aceasta solutie de implementare (utilizarea unui task specializat in achizita de date respectiv generarea de comenzi) este viabila datorita faptului ca datele (starea senzorilor) sunt citie cu o anumita periodicitate (T=20 msec), asigurandu-se o citire simultana a tuturor senzorilor din sistem. O citire separata (fiecare task in parte) nu este utila deoarece in acest caz pot aparea desincronizari generate de timpul de executie al acestor sectiuni. Prin utilizarea taskului TAC aceste neajunsuri au fost eliminate, toate taskurile care controleaza nodurile primind informatii despre starea senzorilor in acelasi moment de timp.

Taskul TAC are rolul, de a supraveghea intrarile si de a transmite catre taskurile care controleaza nodurile valorile aferente cu o cadenta de 20 msec, caz in care se asigura citirea senzorilor si transmiterea starii lor care taskul destinatie, prin mecanismul *Send-Reply*. In cazul receptionarii unui *Reply* in corpul mesajului sunt transmise comenzile care trebuiesc transferate elementelor de executie.

Structura modului de alocare a senzorilor este citita direct de catre TAC dintr-un fisier *sistem.ini* si este memorata intr-o zona de memorie alocata dinamic.

Structura fisierului *sistem.ini* corespunzator sistemului considerat este:

```
%Tip Nod,Nume Nod, Senzori
NT1,nod1,IN=I1.0,OUT=O1.0,FULL=I1.1;
NT4,nod2,REF=I2.0,ID=I2.1,IN=I2.2,OUT1=O2.0,OUT2=O2.1,FULL1=I2.3,FULL2=I
2.4;
NT1,nod3,IN=I3.0,OUT=O3.0,FULL=I3.1;
NT1,nod4,IN=I4.0,OUT=O4.0,FULL=I4.1;
NT2,nod5,IN1=I5.0,IN2=I5.1,OUT=O5.0,FULL=I5.2;
end_nod;

%Numar jamuri sistem
Nr_Jamuri = 6;
%Declarare jamuri: Nume Jam, Capacitate Maxima, Valoare initiala
Jam1,5,0;
Jam2,5,0;
Jam3,5,0;
Jam4,5,0;
Jam5,5,0;
Jam6,5,5;
end_jam;
End;
```

unde :

- I *oct.bit* - reprezinta intrarea I, corespunzatoare octetului *oct* bitul *bit* al magistralei INTERBUS utilizate [INTERBUS_05].
- O *oct.bit* - reprezinta iesirea O, corespunzatoare octetului *oct* bitul *bit* al magistralei INTERBUS utilizate [INTERBUS_05].

Referitor la modul de transfer al informatiei intre taskuri, se face precizarea ca intre taskurile care controleaza nodurile si taskul de achizitii si generare de comenzi se foloseste mecanismul *Send-Reply,* iar intre taskurile nodurilor se foloseste memoria partajata (jam-urile fiind implementate in shared memory).

Structura mesajelor *Send-Reply* are format fix tinand cont de numarul maxim de senzori. Structura unui mesaj Send este:

Tip_Nod	Nume_Nod	REF	IN1/ID	IN2	OUT1	OUT2	FULL1	FULL2

In cazul nodurilor care au un singur senzor de intrare starea senzorului corespunzator este transmis prin intermediul campului IN1/ID, restul campurilor fiind ignorate la receptie. Acelasi lucru este valabil pentru senzorii de OUT si FULL, starea fiind transmisa prin campurile OUT1 respectiv FULL1.

Raspunsul de confirmare a taskurilor (*REPLY*) contine comenzile necesare elementelor de executie. Structura mesajului este:

Tip_Nod	Nume_Nod	CMD_Stoper1	CMD_Stoper2	CMD_Switch

In cazul mesajului *REPLY,* care contine comenzile corespunzatoare elementelor de executie, in cazul existentei unui singur stoper, comanda este transmisa prin intermediul campului CMD_Stoper1. Campul CMD_Switch reprezinta comanda corespunzatoare a macazului (pentru nodurilor de tip 2 si 4).

Dupa cum s-a precizat anterior, transferul de date intre taskurile care controleaza nodurile se face prin intermediul unei zone SHM in care sunt implementate jamurile. Operatiile pe care taskurile le executa asupra datelor din SHM sunt de incrementare/decrementare a campului *JAM_count*. Responsabil cu crearea si initializarea zonei SHM care contine jamurile este taskul TAC. Conditiile de creare a acestei zone SHM (numarul de jamuri) este transmisa taskului TAC tot prin intermediul fisierului *sistem.ini*. Modul de declarare a numarului si structurile corespunzatoare jamurilor din sistem este:

```
%Numar jamuri sistem
Nr_Jamuri = 6;
%Declarare jamuri: Nume Jam, Capacitate Maxima, Valoare initiala
Jam1,5,0;
Jam2,5,0;
Jam3,5,0;
Jam4,5,0;
Jam5,5,0;
Jam6,5,5;
end_jam;
```

Secventa de cod prin care se asigura crearea si initializarea zonei SHM este:

```
// Creare zona SHM cu numele jam_shm
          fd = shm_open("jam_shm", O_RDWR | O_CREAT, 0777);
    if (fd == -1) {
          cprintf("SHM Open failed\n");
                    exit(1);
          }
 // Setare dimensiune jam_shm
    if (ltrunc(fd, nr_jam*sizeof(Jam), SEEK_SET) == -1) {
                    cprintf("ltrunc Error\n");
                    exit(1);
          }

// Mapare jam_shm
          addr = mmap(0, nr_jam*sizeof(Jam), PROT_READ |
PROT_WRITE,MAP_SHARED, fd, 0);
          if (addr == (void *) -1) {
                    cprintf("mmap failed\n");
                    exit(1);
          }

//Initializare jam_shm
            for(i=0;i<nr_jam;i++){
              strcpy(addr[i].Jam_ID,nume_jam[i]);    //nume_jam[i] preluat din
memorie si obtinut dupa
                                                           //procesarea fisierului
sistem.ini
              addr[i].Jam_Val   = val_jam[i];         //val_jam[i] preluat din
memorie si obtinut dupa
                                                          //procesarea fisierului
sistem.ini
              addr[i].Jam_Count = count_jam[i];      //count_jam[i] preluat din
memorie si obtinut dupa
                                                          //procesarea fisierului
sistem.ini
             }
//Inchide conexiune la jam_shm
       close(fd);
```

Pentru ca intreg sistemul multitasking sa functioneze corect, fiecare task se „inregistreaza" in sistem sub un nume alocat (vezi lansarea in executie cu linie de comanda).

Primul task care se inregistreaza este TAC, el avand sarcina, de a crea zona SHM necesara. Secventa de cod corespunzatoare „inregistrarii" in sistem a taskului TAC este:

```
//atasare nume tac
     id_tac = qnx_name_attach( 0, "tac" );
     if( id_tac == -1 ) {
         printf("Attach TAC failed.\n" );
         return;
```

```
    }
//cautare procese
    contor_procese:=0;
    while(1){
            if(contor_procese == nr_jam){
                  cprintf(„\n Toate procesele sunt pornite!”);
                  cprintf(„\n Incepe operatia de sincronizare a taskurilor!”);
                  break;
            }
//cautare proces task[contor_procese]
            for(;;){

pid_n[contor_procese]=qnx_name_locate(0,task[contor_procese],255,0);
                 if(pid_n[contor_procese] != -1){
                     cprintf("\nTaskul  %s este pornit pid:
%d",task[contor_procese],pid_n[contor_procese]);
                       contor_procese++;
                     break;
                 }
                else{
                     cprintf("\n Taskul  %s oprit",task[contor_procese]);
                 }
            }
    }

//trimitere mesaj de sincronizare
     for(i=0;i<nr_jam;i++){
           sincro_noduri(pid_n[0],i+1);
     }
     cprintf("\n Sincronizare gata. Start transmisie date!");
```

Taskul TAC meoreaza intern (intr-o zona de memorie alocata dinamic), pentru fiecare nod in parte, toate informatile legate de starile senzorilor si comenzile care trebuie transmise. Structura de date corespunzatoare este:

```
typedef struct{
            char     nod[10];
            int      tip_nod;
         short    ref;
         short    id;
         short    in1;
         short    in2;
         short    out1;
         short    out2;
         short    full1;
         short    full2;
           short    cmd_st1;
           short    cmd_st2;
           short    cmd_switch;
}Nod_Struct;
```

Secventa de cod prin care s-a implementat functia de sincronizare a nodurilor in taskul TAC este:

```
//functia sincro_noduri
void sincro_noduri(pid_t pid_nod,int nr_nod ){
   char msg_send_nod[255];
   char msg_rec_nod[255];
   char tmp_string[5];

   cprintf("\nTrimitere mesaj de sincronizare TAC - N%d...",nr_nod);
   itoa(nr_nod,tmp_string,10);
   strcpy(msg_send_nod,"Cerere Sincronizare TAC -N");
   strcat(msg_send_nod,tmp_string);

Send(pid_nod,&msg_send_nod,&msg_rec_nod,sizeof(msg_send_nod),sizeof(msg
_rec_nod));
   cprintf("\n %s",msg_rec_nod);
   cprintf("\n Sincronizare gata Nod:N%d",nr_nod);
}
```

Dupa cum se observa din secventele de cod prezentate, toate taskurile corespunzatoare nodurilor trebuie „sa se afle in rulare”.

Conexiunea software dintre TAC si modulul INTERBUS se face prin intermediul unui driver specializat, transferul datelor realizandu-se prin intermediul unei zone SHM creata si gestionata de functii speciale [PhoenixWeb].

In cazul taskurilor de control, ale nodurilor, secventa prin care acestea se „inregistreaza” in sistem este identica. Secventa de cod pentru „inregistrare”, conectare la SHM si sincronizare este:

```
//verifica daca tac este pornit
     pid_tac = qnx_name_locate(0,"tac",255,0);
     while(pid_tac == -1){
         pid_tac = qnx_name_locate(0,"tac",255,0);
         cprintf("TAC nu este pornit!\n");
         cprintf("\nAsteapta pornire TAC...");
     }

//atasare nume task – primit prin linia de comanda
          id_nod=qnx_name_attach(0,argv[1]);
          if(id_nod == -1){
                 cprintf("Nume task  nealocat!");
                 exit(0);
          }

// Conectare la jam_shm creata de TAC
         fd = shm_open("jam_shm", O_RDWR , 0777);
            if (fd == -1) {
                cprintf("jam_shm open failed\n");
                exit(1);
         }
```

```
// Mapare la jam_shm
        addr = mmap(0, nr_jam*sizeof(Jam), PROT_READ |
PROT_WRITE,MAP_SHARED, fd, 0);
        if (addr == (void *) -1) {
                cprintf("mmap failed\n");
                exit(1);
        }

//Asteapta sincronizarea cu TAC
          Receive(pid_tac,&msg_rec_tac,sizeof(msg_rec_tac));
          cprintf("\n %s",msg_rec_tac);
          strcpy(msg_reply_tac,"Sincronizare TAC - NT1 gata!");
          Reply(pid_tac,&msg_reply_tac,sizeof(msg_reply_tac));
```

Modul de formare a mesajelor generate de TAC catre taskurile de control ale nodurilor, precum si preluarea comenzilor de la taskurile nodurilor este:

```
//transmisie date - in functia main
//formare mesaj
         for(;;){
              cprintf("\n Apasa X sau x pentru iesire din program...");
            comanda=getc();
            if(comanda == 'X' || comanda =='x'){
                exit(0);
             }
             else{
                     for(i=0;i<nr_nod;i++){
                     send_data_nod(pid_n[i],&Noduri[i]);
                     }
               }
         }

//functia pentru construirea si transmiterea mesajului
void send_data_nod(pid_t pid_nod,Nod_Struct *ValNod){
        char send_data[255];
        char rec_data[255];
          char tmp_string[10];

        if(ValNod->nod_tip == 1){
                strcpy(send_data,"NT1,");
                    strcpy(tmp_string,ValNod->nod;
                  strcat(send_data,tmp_string);
                  strcat(send_data,",IN=0,OUT=0,FULL=0");
                    itoa(ValNod->in1,send_data[12],10);
                    itoa(ValNod->out1,send_data[18],10);
                    itoa(ValNod->full1,send_data[25],10);
        }
        if(ValNod->nod_tip == 2){
                strcpy(send_data,"NT2,");
```

```
                    strcpy(stmp_string,ValNod->nod);
                      strcat(send_data,tmp_string);
                      strcat(send_data,",IN1=0,IN2=0,OUT=0,FULL=0");
                        itoa(ValNod->in1,send_data[13],10);
                        itoa(ValNod->in2,send_data[19],10);
                        itoa(ValNod->out1,send_data[25],10);
                        itoa(ValNod->full1,send_data[32],10);
        }
        if(ValNod->nod_tip == 4){
                    strcpy(send_data,"NT4,");
                    strcpy(stmp_string,ValNod.nod);
                      strcat(send_data,tmp_string);

strcat(send_data,",REF=0,ID=0,IN=0,OUT1=0,OUT2=0,FULL1=0,FULL2=0");
                        itoa(ValNod.ref,send_data[13],10);
                        itoa(ValNod.id,send_data[18],10);
                        itoa(ValNod.in1,send_data[23],10);
                        itoa(ValNod.out1,send_data[30],10);
                        itoa(ValNod.out2,send_data[37],10);
                        itoa(ValNod.full1,send_data[45],10);
                        itoa(ValNod.full2,send_data[53],10);
        }
        Send(pid_nod,&send_data,&rec_data,sizeof(send_data),sizeof(rec_data));
//receptie prin REPLY
          if(ValNod->nod_tip == 1){
                 ValNod->Cmd_st1=atoi(rec_data[21]);
          }

          if(ValNod->nod_tip == 2){
                 ValNod->Cmd_st1=atoi(rec_data[21]);
                 ValNod->Cmd_st2=atoi(rec_data[34]);
                 ValNod->Cmd_switch=atoi(rec_data[47]);
          }
          if(ValNod->nod_tip == 4){
                 ValNod->Cmd_st1=atoi(rec_data[21]);
                 ValNod->Cmd_switch=atoi(rec_data[34]);
          }
}
```

4.2.3. Implementarea nodurilor

4.2.3.1. Taskul NT1 - Nodul de tip 1

Implementarea Nodului de tip 1 are la baza structura automatului stabilita in paragraful 2.4.2.1.1. In fiecare stare se controleaza conditiile de executie a unei tranzitii, generandu-se comenzile corespunzatoare de deschidere, respectiv inchidere a stoperului.

Deoarece toate nodurile prezentate in capitolul 2 au la baza nodul de tip 1, prezentarea ordinogramelor corespunzatoare implementarii starilor si efectuarii tranzitiilor se prezinta numai pentru acest tip de nod . Pentru restul de noduri se va detalia numai partea care este diferita de nodul de tip 1. De asemena secventele de cod corespunzatoare vor fi prezentate numai in cadrul acestui nod.

Secventa de cod prin care s-a implementat automatul este:

```
//variabile de conducere
int     Stare_Activa = 0;
short  IN_Senzor;
short  OUT_Senzor;
short  FULL_Senzor;
short  Reset = 0;

//variabile actualizate din jam_shm
int     OutJam_Val;
int     OutJam_Count;
int     InJam_Count;

//variabile de iesire
short  Cmd_Stoper = 0;

//declaratii pentru timere
pid_t      proxy_timer1, proxy_timer2;
timer_t   id_timer1, id_timer2;
struct itimerspec   timer1, timer2;
struct sigvevent   event_timer1, event_timer2;

//variabile de sincronizare pentru noduri complexe
short   tmp_busy_OUT = 0;
short   tmp_busy_IN;

//variabila de codificare a erorilor
short   error_Code=0;

//verifica starea activa
     switch(Stare_Activa){
               case   1:
//Stare OPEN activa
                         Stare_Open();
                         break;
               case   2:
//Stare CLOSE activa
                         tmp_busy_OUT=Stare_Close();
                         break;
               case   3:
//Stare Error activa
                         Stare_Error();
                         break;
               default:
//Stare WAIT activa
                         tmp_busy_OUT=Stare_Wait(tmp_busy_IN);
                         break;
       }
```

- ***Implementarea starii S0 – WAIT***

Ordinograma dupa care a fost implementata starea WAIT este prezentata in figura 4.13.

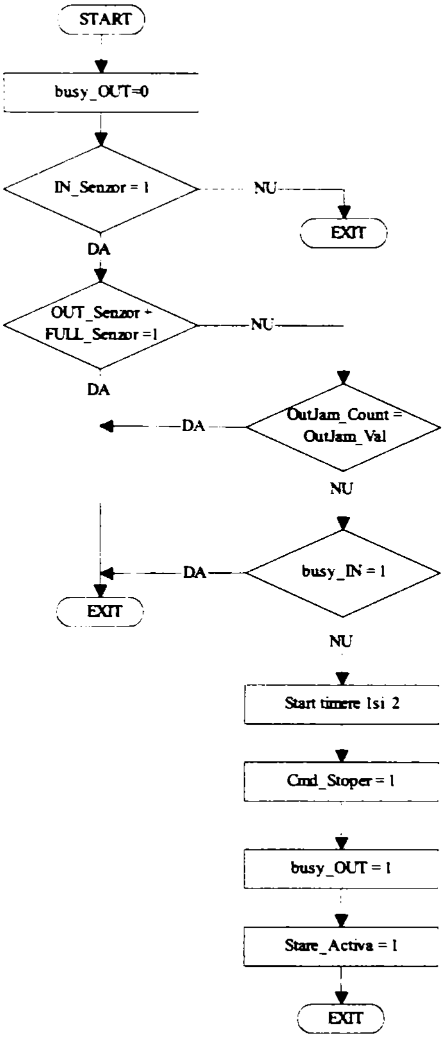


Figura 4.13. Ordinograma starii WAIT

Secventa de cod prin care s-a implementat starea WAIT este:

```
//starea Wait – functia primeste ca parametru busy_IN si genereaza busy_OUT
int  Stare_Wait (int busy_IN){
//verifica IN_Senzor
        if ( IN_Senzor == 0 )
//daca este 0 iese
            return;
//verifica daca OUT_Senzor + FULL_Senzor sunt 1
        if ( OUT_Senzor == 1 || FULL_Senzor == 1 )
//daca este 1 iese
```

```
          return;
//verifica daca este spatiu in OutJam
       if ( OutJam_Val == OutJam_Count )
//daca este egalitate iese
          return;
//verifica stare fanion busy_IN
       if ( busy_IN == 1 )
//daca este 1 iese
          return;
//toate conditile pentru efectuarea unei miscari sunt indeplinite
//creaza si porneste timerele 1 si 2 - expirarea timerelor se detecteaza prin proxy
//creare proxy pentru timer 1
        proxy_timer1 = qnx_proxy_attach(0,0,0,1);
        (if  proxy_timer1 == -1){
            cprintf („Nu s-a creat proxy pentru timer 1!”);
            return;
        }
//atasare proxy la timer 1
        event_timer1.sigev_signo = proxy_timer1;
//creare timer 1
        id_timer1=timer_create(CLOCK_ABSTIME,&event_timer1);
        if ( id_timer1 == -1){
             cprintf („Nu s-a creat timerul 1!”){
             return;
        }
//setari valori timer 1 - expira dupa 3 min
        timer1.it_value.tv_spec = time(NULL) + 180;
        timer1.it_value.tv_nsec = 0L;
        timer1.it_interval.tv_sec = 0L;
        timer1.it_interval.tv_nsec = 0L;
        timer_settime(id_timer1,TIMER_ABSTIME,&timer1);
//creare proxy pentru timer 2
        proxy_timer2 = qnx_proxy_attach(0,0,0,1);
        (if  proxy_timer2 == -1){
            cprintf („Nu s-a creat proxy pentru timer 2!”);
            return;
        }
//atasare proxy la timer 2
        event_timer2.sigev_signo = proxy_timer2;
//creare timer 2
        id_timer2=timer_create(CLOCK_ABSTIME,&event_timer2);
        if ( id_timer2 == -1){
             cprintf („Nu s-a creat timerul 2!”){
             return;
        }
//setari valori timer 2 - expira dupa 5 min
        timer2.it_value.tv_spec = time(NULL) + 300;
        timer2.it_value.tv_nsec = 0L;
        timer2.it_interval.tv_sec = 0L;
        timer2.it_interval.tv_nsec = 0L;
        timer_settime(id_timer2,TIMER_ABSTIME,&timer2);
```

```
//genereaza comanda de deschidere a stoperului
        Cmd_Stoper = 1;
//seteaza busy_OUT
        busy_OUT = 1;
//seteaza starea OPEN activa
        Stare_Activa = 1;
//iese din functie
        return(busy_OUT);
}
```

- ***Implementarea starii S1 – OPEN***

Ordinograma dupa care a fost implementata starea OPEN este prezentata in figura 4.14.

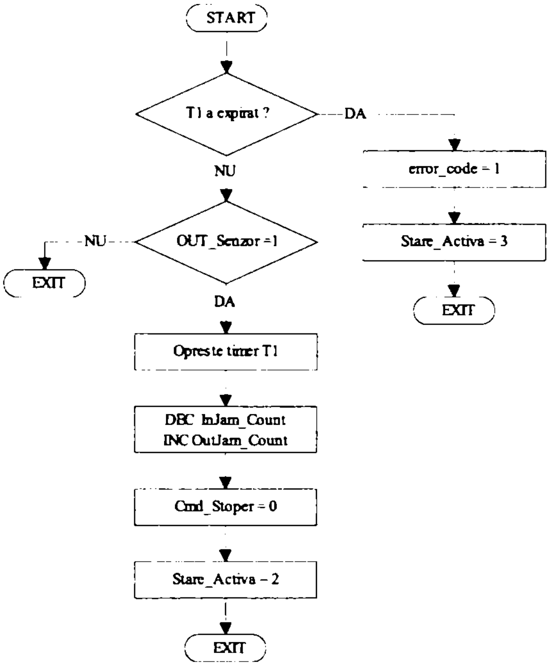


Figura 4.14. Ordinograma starii OPEN

Secventa de cod prin care s-a implementat starea OPEN este:

```
//starea OPEN
void  Stare_Open (void){
//verifica daca timerul 1 a expirat
       if ( Creceive(proxy_timer1,0,0) == proxy_timer1 )
//daca este adevarat timer 1 a expirat si trece in starea de eroare
//seteaza codul de eroare
            error_Code = 1;
//seteaza starea Error activa
            Stare_Activa = 3;
//sterge timer 1
```

```
            timer_delete (id_timer1);
//iese din starea OPEN
            return;
        }
//timerul 1 merge – verifica starea senzorului OUT
       if ( OUT_Senzor == 0 )
//daca este 0 iese
            return;
//daca este 1 – opreste timer 1
       timer_delete (id_timer1);
//decrementeaza jamul de intrare
        InJam_Count--;
//incrementeaza jamul de iesire
        OutJam_Count++;
//inchide stoperul
        Cmd_Stoper = 0;
//seteaza starea CLOSE activa
        Stare_Activa = 2;
//iese din functie
        return;
}
```

- ***Implementarea starii S2 - CLOSE***

Ordinograma dupa care a fost implementata starea CLOSE este prezentata in figura 4.15.

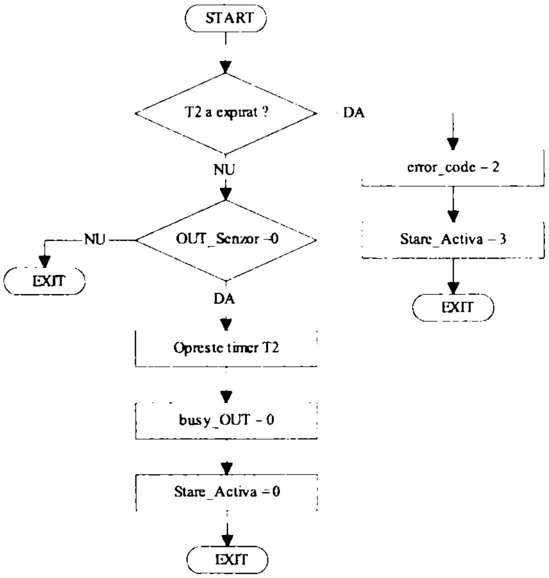


Figura 4.15. Ordinograma starii CLOSE

Secventa de cod prin care s-a implementat starea CLOSE este:

```
//starea CLOSE – transmite starea lui busy_OUT
int Stare_Close (void){
```

```
//verifica daca timerul 2 a expirat
      if ( Creceive(proxy_timer2,0,0) == proxy_timer2 )
//daca este adevarat timer 2 a expirat si trece in starea de eroare
//seteaza codul de eroare
            error_Code = 2;
//seteaza starea Error activa
            Stare_Activa = 3;
//sterge timer 2
            timer_delete (id_timer2);
//iese din starea OPEN
            return(1);
       }
//timerul 2 merge - verifica starea senzorului OUT
      if ( OUT_Senzor == 1 )
//daca este 0 iese
            return(1);
//daca este 0 - opreste timer 2
      timer_delete (id_timer2);
//seteaza starea WAIT activa
       Stare_Activa = 0;
//iese din functie
       return (0);
}
```

- ***Implementarea starii S3 – Error***

Ordinograma dupa care a fost implementata starea Error este prezentata in figura 4.16.

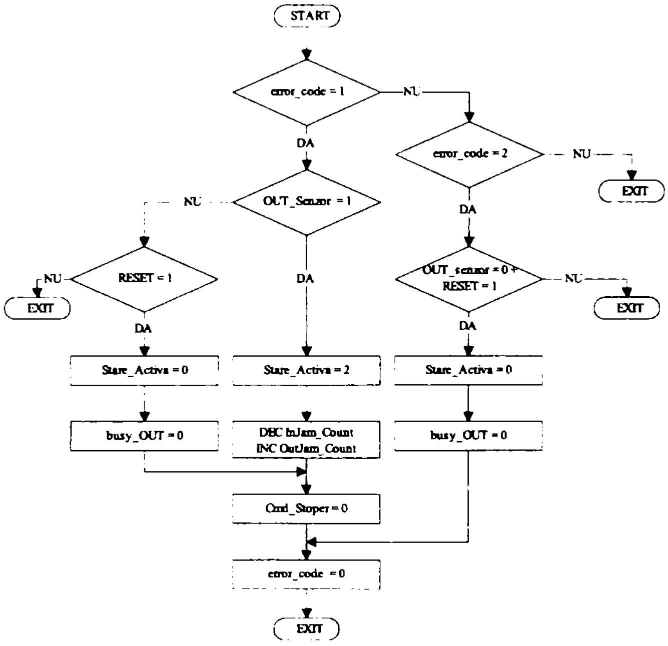


Figura 4.16. Ordinograma starii ERROR

Secventa de cod prin care s-a implementat starea Error este:

```
//starea Error – transmite stare busy_OUT
int  Stare_Error (void){
//verifica codul de eroare
     if ( error_Code == 1){
//daca este 1 verifica daca senzorul OUT s-a activat
          if (OUT_Senzor == 1){
//daca da
//decrementeza jamul de intrare
               InJam_Count--;
//incrementeaza jamul de iesire
               OutJam_Count++;
//inchide stoperul
               Cmd_Stoper = 0;
//seteaza starea CLOSE activa
               Stare_Activa = 2;
//iese din functie
               return(1);
            }
//daca este 0 verifica daca s-a dat reset
          if (Reset == 1){
//daca da
//sterge comanda de reset
               Reset = 0;
//decrementeza jamul de intrare
               InJam_Count--;
//incrementeaza jamul de iesire
               OutJam_Count++;
//inchide stoperul
               Cmd_Stoper = 0;
//seteaza starea Wait activa
               Stare_Activa = 0;
//opreste timer 2
               timer_delete(id_timer2);
//iese din functie cu 0 - busy_OUT = 0;
               return(0);
            }
//daca nu este reset iese
            else
               return(1);
//verifica codul de eroare este 2
     if ( error_Code == 2){
//verifica starea senzorului OUT si a comenzi de reset
            if ( OUT_Senzor == 1 || Reset == 1){
//daca este adevarat sterge Reset
                Reset = 0;
//sterge timer 2
                 timer_delete (id_timer2);
//seteaza starea WAIT activa
                 Stare_Activa = 0;
```

```
//iese din functie cu 0 - busy_OUT = 0
                return (0);
            }
//daca nu iese
            else
                return(0);
    }
//daca nu iese
    else
            return(0);
}
```

4.2.3.2. Taskul NT2 - Nodul de tip 2

Acest tip de nod este construit prin sincronizarea a doua noduri de tip 1 (vezi paragraful 2.4.2.2.1). Modul de implementare a starilor, fiind prezentat anterior, in continuare este prezentat numai modul in care se face sincronizarea celor doua intrari. Organigrama corespunzatoare este prezentata in figura 4.17.

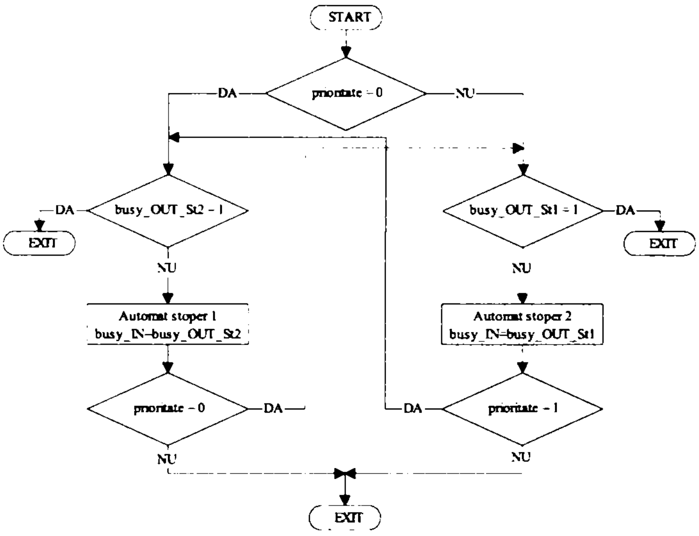


Figura 4.17. Ordinograma corespunzatoare sincronizarii intrarilor intr-un nod de tip 2

Avand in vedere ca implementarea unui nod cu 3 intrari si o iesire este similara cu cele prezentate, aceasta varianta nu a mai fost abordata la nivelul implementarii in QNX.

De asemenea codul corespunzator fiind similar cu cel prezentat in cazul nodului de tip 1, nu se mai prezinta.

4.2.3.3. Taskul NT4 - Nodul de tip 4

Diferenta fata de nodul de tipul 1 este data de starea suplimentara in care se face identificarea caruciorului pentru a se determina directia de deplasare. Organigrama starii S0 (SCANN) este prezentata in figura 4.18.

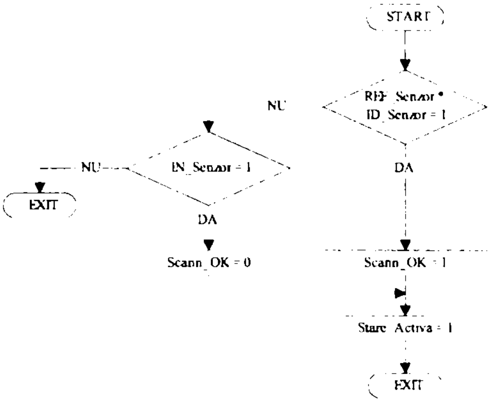


Figura 4.18. Organigrama corespunzatoare starii S0 a unui nod de tip 4

Secventa de cod prin care s-a implementat starea SCANN este:

```
//functia de scanare SCANN - returneaza valoarea lui Scann_OK
void Scann(void){
    short  tmp_Scann_OK;

//initializare tmp_Scann_OK;
     tmp_Scann_OK = 0;
//verifica identitate
     if (REF_Senzor == 1 &&  ID_Senzor == 1){
//exista egalitate seteaza tmp_Scann_OK;
          tmp_Scann_OK = 1;
//trece in starea Wait
          Stare_Activa = 1;
//iese
          return(tmp_Scann_OK);
     }
//daca nu verifica IN_Senzor
       else{
          if(IN_Senzor == 1){
//seteaza starea Wait activa
              Stare_Activa = 1;
//iese
             return(tmp_Scann_OK);
          }
//daca nu este IN_Senzor iese
           else
              return(tmp_Scann_OK);
       }
}
```

4.3. Implementarea modelelor de tip retele Petri cu ajutorul automatelor programabile (PLC) [Ung04a][Ung04b][Ung06a]

In modelarea SED utilizarea modelelor de tip automat a pierdut teren in favoarea modelarii cu ajutorul retelelor Petri, in ultimul timp au fost elaborate o serie de metode de implementare avand ca suport hardware PLC-uri. In [RF02][KWFLL02] se prezinta o metoda de implementare si verificare a modelelor de tip retele Petri utilizand diagramele secventiale. In [UJ96][UJ97] sunt prezentate o serie de metode de implementare a modelelor de tip retele Petri utilizand tehnica de programare tip scara - ladder LAD. In [FG98][Frey02][FL00a][FL00b][GBFP01][GG01][GMMP03][MF02] sunt prezentate metode de implementare a modelelor de tip retele Petri utilizand o tehnica bazata pe retele Petri interpretoare de semnale (**S**ignal **I**nterpreted **P**etri Nets - SPIN). Deoarece aceasta metoda este cea care ia in considerare in modul cel mai concret interfatarea modelelor de tip retele Petri cu modul de functionare al PLC-urilor, ea a fost aleasa pentru implementarea modelelor de tip retele Petri stabilite in capitolul 2.

Ca suport hardware pentru implementare s-a utilizat un PLC SIEMENS, de tip Simatic S7-414 [Berg00].

In continuare se prezinta modul de modelare a SED utilizand tehnica SPIN, precum si modul de implementare in limbajul STEP7, caracteristic PLC-urilor din familia SIEMENS. Exemplificarea se face numai pentru modelul de tip retea Petri corespunzator nodului de tip 1 (paragraful 2.4.2.1.2.1.), pentru celelalte tipuri de noduri procedandu-se analog.

4.3.1. Modelarea SED utilizand tehnica SPIN

Aceasta metoda se bazeaza pe faptul ca reteaua Petri utilizata are atasate corespunzator intrarile si iesirile fizice ale PLC-ului. Intrarile sunt atasate tranzitiilor, avand rol de validare suplimentara a acestora, iar iesirile (comenzile) sunt asignate pozitiilor in care trebuie generate (activate sau dezactivate). In figura 4.19 se prezinta reteaua Petri de tipul SPIN corespunzatoare nodului de tip 1.

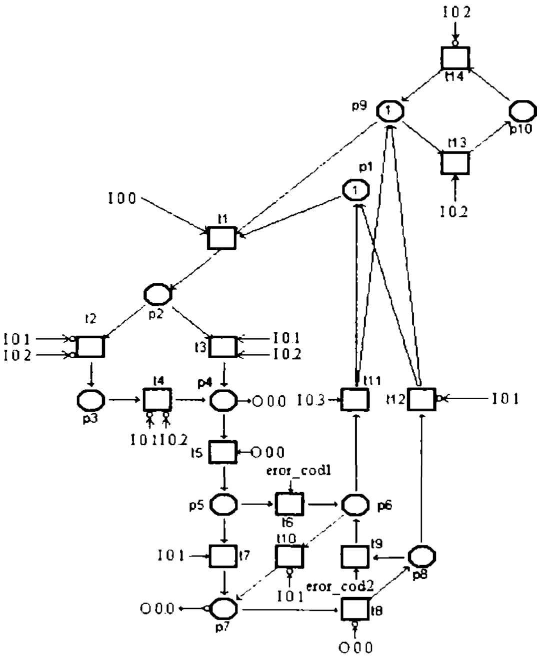


Figura 4.19. Reteaua Petri de tip SPIN corespunzatoare nodului de tip 1

Modul de alocare a intrarilor si iesirilor fizice la PLC sunt prezentate in tabelul 4.3.

Tabelul 4.3. *Alocarea intrarilor/iesirilor fizice*

Intrare/Iesire	Comentariu
I 0.0	Semnal de intrare corespunzator intrarii IN_Senzor
I 0.1	Semnal de intrare corespunzator intrarii OUT_Senzor
I 0.2	Semnal de intrare corespunzator intrarii FULL_Senzor
I 0.3	Semnal de intrare corespunzator comenzii de Reset
O 0.0	Semnal de iesire corespunzator comenzii stoperului

Observatie: in cadrul implementarii utilizand PLC-uri nu se mai considera situatia in care se verifica si capacitatile jamurilor. Aceasta conditie este verificata numai prin intermediul senzorului de FULL.

Dupa cum se observa in figura 4.19 tranzitile t2, t7,t11, t12, t13, t14 sunt validate suplimentar functie de valorile logice ale intrarilor corespunzatoare. In pozitile p4 si p7 sunt generate comenzile corespunzatoare pentru stoper, deschidere (p4), respectiv inchidere (p7).

4.3.2. Implementarea in STEP7 a modelului SPIN corespunzator nodului de tip 1

Mediul de dezvoltare STEP7 permite implementarea programelor de conducere prin mai multe metode: LAD (programare in logica ladder), STL (programare in lista de declaratii), SCL (programare prin limbaj de control structurat) etc. Abordarea implementarii programului corespunzator se va face prin metoda STL, metoda care prin asemanarea cu limbajele de asamblare asigura o mai mare flexibilitate in elaborarea codului [Berg00].

Secventa de cod prin care s-au declarat variabilele corespunzatoare modelul SPIN este:

```
FUNCTION_BLOCK "SPIN_Nod1"
TITLE =SPIN_Nod1
//Functie care implementeaza modelul de tip SPIN corespunzator nodului de tip 1
VERSION : 0.1

VAR_INPUT
  IN_Senzor     : BOOL ;        //senzorul de intrare
  OUT_Senzor    : BOOL ;        //senzorul de iesire
  FULL_Senzor : BOOL ;          //senzorul de FULL
  Reset           : BOOL ;      //comanda de reset
END_VAR
VAR_OUTPUT
  Cmd_Stoper    : BOOL ;        //Comanda stoperului
  Nod_OK        : BOOL ;        //Variabila activata la revenire in starea initiala
END_VAR
VAR
  Contor_P1     : INT  := 1;    //Contorul jetoanelor din pozitia p1
```

```
  Contor_P2     : INT ;  //Contorul jetoanelor din pozitia p2
  Contor_P3     : INT ;  //Contorul jetoanelor din pozitia p3
  Contor_P4     : INT ;  //Contorul jetoanelor din pozitia p4
  Contor_P5     : INT ;  //Contorul jetoanelor din pozitia p5
  Contor_P6     : INT ;  //Contorul jetoanelor din pozitia p6
  Contor_P7     : INT ;  //Contorul jetoanelor din pozitia p7
  Contor_P8     : INT ;  //Contorul jetoanelor din pozitia p8
  Contor_P9     : INT  := 1;      //Contorul jetoanelor din pozitia p9
  Contor_P10    : INT ;  //Contorul jetoanelor din pozitia p10
  Tranzitie_t1  : BOOL ;          //Stare tranzitie t1
  Tranzitie_t2  : BOOL ;          //Stare tranzitie t2
  Tranzitie_t3  : BOOL ;          //Stare tranzitie t3
  Tranzitie_t4  : BOOL ;          //Stare tranzitie t4
  Tranzitie_t5  : BOOL ;          //Stare tranzitie t5
  Tranzitie_t6  : BOOL ;          //Stare tranzitie t6
  Tranzitie_t7  : BOOL ;          //Stare tranzitie t7
  Tranzitie_t8  : BOOL ;          //Stare tranzitie t8
  Tranzitie_t9  : BOOL ;          //Stare tranzitie t9
  Tranzitie_t10 : BOOL ;          //Stare tranzitie t10
  Tranzitie_t11 : BOOL ;          //Stare tranzitie t11
  Tranzitie_t12 : BOOL ;          //Stare tranzitie t12
  Tranzitie_t13 : BOOL ;          //Stare tranzitie t13
  Tranzitie_t14 : BOOL ;          //Stare tranzitie t14
  eror_cod1       : BOOL ;        //cod de eroare generat de timer 1
  eror_cod2       : BOOL ;        //cod de eroare generat de timer 2
  Timer1          : S5TIME  := S5T#3M;         //constanta de timp timer 1
  Timer2          : S5TIME  := S5T#5M;         //constanta de timp timer 1
  FN_Timer1     : BOOL ;          //detectie expirare timer 1
  FN_Timer2     : BOOL ;          //detectie expirare timer 2
END_VAR
VAR_TEMP
  tmp_P1 : BOOL ;
  tmp_P2 : BOOL ;
  tmp_P3 : BOOL ;
  tmp_P4 : BOOL ;
  tmp_P5 : BOOL ;
  tmp_P6 : BOOL ;
  tmp_P7 : BOOL ;
  tmp_p8 : BOOL ;
  tmp_P9 : BOOL ;
  tmp_P10 : BOOL ;
  tmp_FN_Timer1 : BOOL ;
  tmp_FN_Timer2 : BOOL ;
END_VAR
```

Modul de operare al programului se bazeaza pe regula de validare a unei tranzitii (capitolul 2). In continuare se prezinta codul program aferent verificarii si executiei tranzitiei t1.

```
NETWORK
TITLE =Verifica conditie tranzitie t1
//Daca tranzitia este validata:
//- sterge jetoanele din pozitile P1 si P9
//- transfera un jeton in pozitia P2
//verifica daca in pozitia P1 este jeton
      L    #Contor_P1;
      L    1;
      ==I  ;
      =    #tmp_P1;
//verifica daca in pozitia P9 este jeton
      L    #Contor_P9;
      L    1;
      ==I  ;
      =    #tmp_P9;

//verifica conditia de validare tranzitie t1
      A    #tmp_P1;
      A    #tmp_P9;
      A    #IN_Senzor;
//daca da seteaza t1 activa
      =    #Tranzitie_t1;

//verificare validare tranzitie t1 si executie
      A    #Tranzitie_t1;
//daca este zero salt la verificare urmatoare tranzitie
      JCN  n001;
//tranzitia este validata
//sterge jetonul din P1
      L    #Contor_P1;
      L    -1;
      +I   ;
      T    #Contor_P1;

//sterge jetonul din P9
      L    #Contor_P9;
      L    -1;
      +I   ;
      T    #Contor_P9;

//transfera un jeton in P2
      L    #Contor_P2;
      L    1;
      +I   ;
      T    #Contor_P2;

n001: NOP  0;
```

Codul program corespunzator generarii comenzilor este:

```
NETWORK
TITLE =Formare comenzi pentru stoper
//Daca pozitia P4 este activa seteaza Cmd_Stoper
//Daca pozitia P7 este activa reseteaza Cmd_Stoper
//verifica pozitia P4
      L    #Contor_P4;
      L    1;
      ==I  ;
      S    #Cmd_Stoper;

//verifica pozitia p7
      L    #Contor_P7;
      L    1;
      ==I  ;
      R    #Cmd_Stoper;
```

Formarea codurilor de eroare *eror_cod1* respectiv *eror_cod2,* se face tinand cont ca in pozitia P4, cand stoperul a fost deschis, se pornesc doua timere de control a miscarii (*Timer1* respectiv *Timer2*) care nu trebuie sa expire: primul pana jetonul ajunge in pozitia P7 iar al doilea pana jetonul ajunge in pozitia P8. Daca aceste conditii nu sunt indeplinite la expirarea lui *Timer1* se seteaza *eror_cod1,* iar la expirarea lui *Timer2* se seteaza *eror_cod2*. Daca operatiunile decurg normal la ajungerea jetonului in pozitia P7 *Timer1,* este oprit iar in P8 *Timer2* este oprit. In cazul activarii codurilor de eroare acestea sunt sterse in momentul in care pozitia P6 devine inactiva (jetonul paraseste pozitia). Secventa de cod prin care s-a implementat setarea codurilor de eroare este:

```
NETWORK
TITLE =Formare coduri de eroare
//In pozitia P4 start timer1 si timer2.
//Daca timer1 a expirat si nu s-a atins P7 eror_cod1 = TRUE
//Daca timer2 a expirat si nu s-a atins P1 eror_cod2 = TRUE
//verifica pozitia 4 activa
      L    #Contor_P4;
      L    1;
      ==I  ;
      =     #tmp_P4;

//start timer1
      A    #tmp_P4;
      FR   T     1;
      L    #Timer1;
      SE   T     1;

//start timer2
      A    #tmp_P4;
      FR   T     2;
      L    #Timer2;
      SE   T     2;

//detectie expirare timer 1
```

```
    A   T    1;
    FN   #FN_Timer1;
    =    #tmp_FN_Timer1;

//detectie expirare timer 2
    A   T    2;
    FN   #FN_Timer2;
    =    #tmp_FN_Timer2;

//verificare eror_cod1
    AN   #OUT_Senzor;
    A    #tmp_FN_Timer1;
    S    #eror_cod1;

//verificare eror_cod2
    AN   #OUT_Senzor;
    A    #tmp_FN_Timer2;
    S    #eror_cod2;
```

In cazul nodurilor de tip 2, 3 si 4 modul de implementare este similar, tinand cont ca structurile corespunzatoare sunt construite pe baza nodului de tip 1 si din acest motiv ele nu mai sunt prezentate.

4.3.3. Implementarea in STEP7 a modelului SPIN corespunzator unui STZA

Se considera STZA-ul din figura 2.151 a carui retea Petri validata este prezentata in figura 2.152. Modelul SPIN corespunzator acestui model este prezentat in figura 4.20.

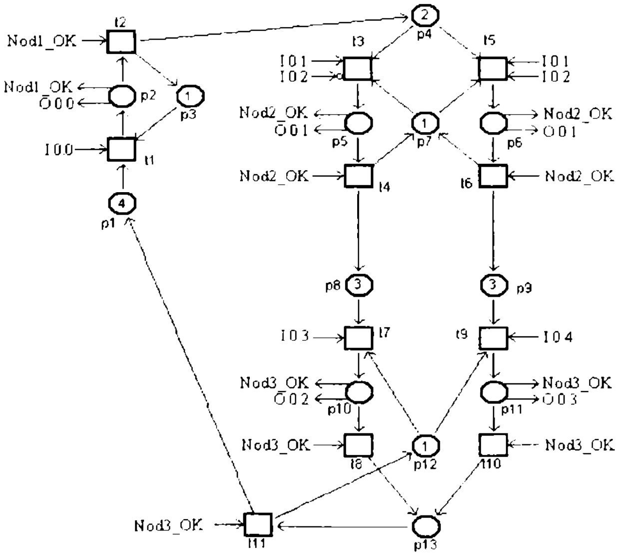


Figura 4.20. Modelul SPIN corespunzator STZA din figura 2.152

Modul de alocare a intrarilor si iesirilor fizice la PLC sunt prezentate in tabelul 4.4.

Tabelul 4.4. *Alocarea intrarilor/iesirilor fizice*

Intrare/Iesire	Comentariu
I 0.0	Semnal de intrare corespunzator lui IN_Senzor pentru nodul 1
I 0.1	Semnal de intrare corespunzator lui OUT_Senzor pentru nodul 1
I 0.2	Semnal de intrare corespunzator lui FULL_Senzor pentru nodul 1
I 0.3	Semnal de intrare corespunzator comenzii de Reset pentru nodul 1
I 0.4	Semnal de intrare corespunzator lui IN_Senzor pentru nodul 2
I 0.5	Semnal de intrare Scann_OK pentru nodul 2
I 0.6	Semnal de intrare corespunzator lui OUT1_Senzor pentru nodul 2
I 0.7	Semnal de intrare corespunzator lui OUT2_Senzor pentru nodul 2
I 1.0	Semnal de intrare corespunzator lui FULL1_Senzor pentru nodul 2
I 1.1	Semnal de intrare corespunzator lui FULL2_Senzor pentru nodul 2
I 1.2	Semnal de intrare corespunzator comenzii de reset pentru nodul 2
I 1.3	Semnal de intrare corespunzator lui IN1_Senzor pentru nodul 3
I 1.4	Semnal de intrare corespunzator lui IN2_Senzor pentru nodul 3
I 1.5	Semnal de intrare corespunzator lui OUT_Senzor pentru nodul 3
I 1.6	Semnal de intrare corespunzator lui FULL_Senzor pentru nodul 3
I 1.7	Semnal de intrare corespunzator comenzii de reset pentru nodul 3
O 0.0	Semnal de iesire corespunzator comenzii stoperului pentru nodul 1
O 0.1	Semnal de iesire corespunzator comenzii stoperului pentru nodul 2
O 0.2	Semnal de iesire corespunzator comenzii stoperului 1 pentru nodul 3
O 0.3	Semnal de iesire corespunzator comenzii stoperului 2 pentru nodul 3

In figura 4.20 nu au fost atasate toate intrarile deoarece ele opereaza strict asupra nodurilor de tip 1 utilizate pentru implementare. Variabilele *Nodx_OK* sunt generate de noduri asa cum s-a precizat in paragraful anterior.

In cazul implementarii STZA se tine cont de jamuri, structura de date corespunzatoare fiind:

```
TYPE "Jam_Struct"
VERSION : 0.1

  STRUCT
   Contor      : INT ;   //valoarea curenta
   Cap_Max   : INT ;     //capacitatea maxima
  END_STRUCT ;
END_TYPE
```

Secventa de cod prin care s-au declarat variabilele corespunzatoare modelul SPIN considerat pentru STZA este:

```
FUNCTION_BLOCK "STZA"
TITLE =Functie care implementeaza STZA
VERSION : 0.1
```

```
VAR_INPUT
  IN_Senzor_Nod1       : BOOL ;        //Senzorul de intrare pentru nodul 1
  IN_Senzor_Nod2       : BOOL ;        //Senzorul de intrare pentru nodul 2
  IN1_Senzor_Nod3      : BOOL ;        //Senzorul de intrare directie 1 nod 3
  IN2_Senzor_Nod3      : BOOL ;        //Senzorul de intrare directie 2 nod 3
  Scann_OK              : BOOL ;       //Rezultat scanare nod 2
  OUT_Senzor_Nod1      : BOOL ;        //Senzorul de iesire pentru nodul 1
  OUT1_Senzor_Nod2     : BOOL ;        //Senzorul de iesire directie 1 pentru nodul 2
  OUT2_Senzor_Nod2     : BOOL ;        //Senzorul de iesire directie 2 pentru nodul 2
  OUT_Senzor_Nod3      : BOOL ;        //Senzorul de iesire pentru nodul 3
  FULL_Senzor_Nod1    : BOOL ;//Senzorul de FULL pentru nodul 1
  FULL1_Senzor_Nod2 : BOOL ;//Senzorul de FULL pentru directia 1 nodul 2
  FULL2_Senzor_Nod2 : BOOL ;//Senzorul de FULL pentru directia 2 nodul 2
  FULL_Senzor_Nod3    : BOOL ;//Senzorul de FULL pentru nodul 3
  Reset_Nod1              : BOOL ;     //Comanda de reset pentru nodul 1
  Reset_Nod2              : BOOL ;     //Comanda de reset pentru nodul 2
  Reset_Nod3              : BOOL ;     //Comanda de reset pentru nodul 3
END_VAR
VAR_OUTPUT
  Cmd_Stoper_Nod1          : BOOL ;    //Comanda stoper nod1
  Cmd_Stoper_Nod2          : BOOL ;    //Comanda stoper nod 2
  Cmd_Stoper_Dir1_Nod3 : BOOL ;        //Comanda stoper directie 1 nod3
  Cmd_Stoper_Dir2_Nod3 : BOOL ;        //Comanda stoper directie 2 nod3
END_VAR
VAR
  Nod1 : "SPIN_Nod1";
  Nod2_Dir1 : "SPIN_Nod1";
  Nod2_Dir2 : "SPIN_Nod1";
  Nod3_Dir1 : "SPIN_Nod1";
  Nod3_Dir2 : "SPIN_Nod1";
  P1 : "Jam_Struct";
  P4 : "Jam_Struct";
  P8 : "Jam_Struct";
  P9 : "Jam_Struct";
  Contor_P2 : INT ;
  Contor_P3 : INT  := 1;
  Contor_P5 : INT ;
  Contor_P6 : INT ;
  Contor_P7 : INT  := 1;
  Contor_P10 : INT ;
  Contor_P11 : INT ;
  Contor_P12 : INT  := 1;
  Contor_P13 : INT ;
  Tranzitie_t1 : BOOL ;
  Tranzitie_t2 : BOOL ;
  Tranzitie_t3 : BOOL ;
  Tranzitie_t4 : BOOL ;
  Tranzitie_t5 : BOOL ;
  Tranzitie_t6 : BOOL ;
  Tranzitie_t7 : BOOL ;
```

```
  Tranzitie_t8 : BOOL ;
  Tranzitie_t9 : BOOL ;
  Tranzitie_t10 : BOOL ;
  Tranzitie_t11 : BOOL ;
  Nod1_OK : BOOL ;
  Nod2_OK : BOOL ;
  Nod3_OK : BOOL ;
END_VAR
VAR_TEMP
  tmp_P1 : BOOL ;
  tmp_P2 : BOOL ;
  tmp_P3 : BOOL ;
  tmp_P4 : BOOL ;
  tmp_P5 : BOOL ;
  tmp_P6 : BOOL ;
  tmp_P7 : BOOL ;
  tmp_P8 : BOOL ;
  tmp_P9 : BOOL ;
  tmp_P10 : BOOL ;
  tmp_P11 : BOOL ;
  tmp_P12 : BOOL ;
  tmp_P13 : BOOL ;
END_VAR
```

Implementarea nodurilor s-a realizat similar cu cele prezentate in cazul implementarii nodului de tip 1. In continuare se prezinta codul sursa corespunzator implementarii directiei 1 de miscare a nodului 2 (nod de tip 4).

```
NETWORK
TITLE =Verifica conditie tranzitie t3
//Daca tranzitia este validata:
//- sterge un jeton din pozitile P4 si P7
//- transfera un jeton in pozitia P5

//verifica daca in pozitia P4 este cel putin un jeton
      L     #P4.Contor;
      L     0;
      <>I   ;
      =     #tmp_P4;

//verifica daca in pozitia P5 este jeton
      L     #Contor_P5;
      L     1;
      ==I   ;
      =     #tmp_P5;

//verifica daca in pozitia P7 este jeton
      L     #Contor_P7;
      L     1;
      ==I   ;
      =     #tmp_P7;
```

```
//verifica conditia de validare tranzitie t3
     A     #tmp_P4;
     AN    #tmp_P5;
     A     #tmp_P7;
     A     #IN_Senzor_Nod2;
     AN    #Scann_OK;
//daca da seteaza t1 activa
     =     #Tranzitie_t3;

//verificare validare tranzitie t3 si executie
     A     #Tranzitie_t3;
//daca este zero salt la verificare urmatoare tranzitie
     JCN   n003;
//tranzitia este validata
//sterge jetonul din P4
     L     #P4.Contor;
     L     -1;
     +I    ;
     T     #P4.Contor;

//sterge jetonul din P7
     L     #Contor_P7;
     L     -1;
     +I    ;
     T     #Contor_P7;

//transfera un jeton in P5
     L     #Contor_P5;
     L     1;
     +I    ;
     T     #Contor_P5;

n003: NOP   0;
NETWORK
TITLE =Verifica conditie tranzitie t4
//Daca tranzitia este validata:
//- sterge jetonul din pozitia P5
//- transfera un jeton in pozitia P7 si P8
//verifica daca in pozitia P8 este cel putin un loc liber
     L     #P8.Contor;
     L     #P8.Cap_Max;
     <>I   ;
     =     #tmp_P8;

//verifica daca in pozitia P5 este jeton
     L     #Contor_P5;
     L     1;
     ==I   ;
     =     #tmp_P5;
```

```
//verifica daca in pozitia P7 este jeton
      L    #Contor_P7;
      L    1;
      ==I  ;
      =     #tmp_P7;

//verifica conditia de validare tranzitie t4
      A    #tmp_P8;
      A    #tmp_P5;
      AN   #tmp_P7;
      A    #Nod2_OK;
//daca da seteaza t4 activa
      =     #Tranzitie_t4;

//verificare validare tranzitie t4 si executie
      A    #Tranzitie_t4;
//daca este zero salt la verificare urmatoare tranzitie
      JCN  n004;
//tranzitia este validata
//sterge jetonul din P5
      L    #Contor_P5;
      L    -1;
      +I   ;
      T    #Contor_P5;
//sterge Nod2_OK
      CLR  ;
      =     #Nod2_OK;
//transfera un jeton in P7
      L    #Contor_P7;
      L    1;
      +I   ;
      T    #Contor_P7;
//transfera un jeton in P8
      L    #P8.Contor;
      L    1;
      +I   ;
      T    #P8.Contor;
n004: NOP  0;
```

Secventa de cod elaborata pentru implementarea controlului miscarilor, exemplificata pentru nodul 2 (nod de tip 4) este prezentata in continuare.

```
NETWORK
TITLE =Apelare control Nod2

//verifica jetoanele din P5 - directia 1
      L    #Contor_P5;
      L    1;
      <>I  ;
      JC   n013;
```

```
//daca e un jeton apeleaza functia de control
    CALL #Nod1 (
        IN_Senzor              := #IN_Senzor_Nod2,
        OUT_Senzor              := #OUT1_Senzor_Nod2,
        FULL_Senzor             := #FULL1_Senzor_Nod2,
        Reset                 := #Reset_Nod2,
        Cmd_Stoper              := #Cmd_Stoper_Nod2,
        Nod_OK                 := #Nod2_OK);

n013: NOP  0;

//verifica jetoanele din P6 - directia 2
    L    #Contor_P2;
    L    1;
    <>I  ;
    JC   n014;

//daca e un jeton apeleaza functia de control
    CALL #Nod1 (
        IN_Senzor              := #IN_Senzor_Nod2,
        OUT_Senzor              := #OUT2_Senzor_Nod2,
        FULL_Senzor             := #FULL2_Senzor_Nod2,
        Reset                 := #Reset_Nod2,
        Cmd_Stoper              := #Cmd_Stoper_Nod2,
        Nod_OK                 := #Nod2_OK);

n014: NOP  0;
```

Structura programului principal corespunzator conducerii unui STZA pe baza modelelor de tip retele Petri este:

```
    CALL "STZA" , "DI_STZA" (
        IN_Senzor_Nod1          := I    0.0,
        IN_Senzor_Nod2          := I    0.4,
        IN1_Senzor_Nod3          := I    1.3,
        IN2_Senzor_Nod3          := I    1.4,
        Scann_OK                  := I    0.5,
        OUT_Senzor_Nod1          := I    0.1,
        OUT1_Senzor_Nod2         := I    0.6,
        OUT2_Senzor_Nod2         := I    0.7,
        OUT_Senzor_Nod3          := I    1.5,
        FULL_Senzor_Nod1         := I    0.2,
        FULL1_Senzor_Nod2        := I    1.0,
        FULL2_Senzor_Nod2        := I    1.1,
        FULL_Senzor_Nod3          := I    1.6,
        Reset_Nod1                  := I    0.3,
        Reset_Nod2                  := I    1.2,
        Reset_Nod3                  := I    1.7,
        Cmd_Stoper_Nod1            := Q    0.0,
        Cmd_Stoper_Nod2            := Q    0.1,
        Cmd_Stoper_Dir1_Nod3   := Q    0.2,
```

```
        Cmd_Stoper_Dir2_Nod3   := Q      0.3);
```

Implementarea modelelor de tip retele Petri s-a realizat avand ca suport consideratiile legate de validarea executiei unei tranzitii. Verificarile si testele efectuate au validat in totalitate modalitatea de implementare a modelelor de tip retele Petri avand ca suport hardware PLC-uri din familia Simatic S7. Cu toate ca pentru teste s-a utilizat un PLC Simatic S7-414, programul a fost astfel conceput incat codul sa fie portabil pe toate PLC-urile din familiile S7-300 si S7-400.

4.4. Concluzii

In cadrul acestui capitol s-a urmarit elaborarea unor metode de implementare a programelor de conducere a SED avand ca suport teoretic modele de tip automat, respectiv modele de tip retele Petri. Modelele utilizate in cadrul operatiei de implementare au fost in prealabil validate prin simulare in mediul Matlab.

Metodele de implementare depind in mare masura de echipamentele hardware utilizate. In acest moment tendintele sunt caracterizate prin:

- utilizarea calculatoarelor de proces;
- utilizarea automatelor programabile.

In cadrul capitolului s-au elaborat, in consecinta, doua metodologii de implementare distincte: prima, bazata pe utilizarea unui calculator de proces ca suport hardware, pe care este instalat sistemul de operare in timp real QNX, programul de conducere fiind scris in limbajul WATCOM C, iar a doua bazata pe utilizarea unui PLC tip Simatic S7-414 programul de conducere fiind scris in limbajul STEP7.

In cazul implementarii pe calculatorul de proces, modelul considerat a fost de tip automat. In cadrul procesului de implementare s-a tinut cont de capabilitatile SOTR QNX, astfel incat programul nu este elaborat intr-un singur task, ci s-a implementat un sistem multitasking, comunicatia dintre taskuri fiind asigurata prin mecanisme specifice sistemelor de operare in timp real (mesaje, proxy, memorie partajata). Pentru fiecare tip de nod , mai putin nodul de tip 3, au fost elaborate taskuri independente parametrizabile, asigurandu-se astfel individualizarea lor in sistem. Modul de comanda al starii senzorilor si generarea comenzilor a fost implementata separat intr-un task special (TAC), asigurandu-se astfel „izolarea" taskurilor de conducere ale nodurilor de sistemul fizic. Conexiunea dintre taskurile de control ale nodurilor si sistemul fizic s-a realizat prin implementarea unui mecanism de tipul *Send-Reply* , taskul TAC fiind „conducatorul" sistemului.

Taskurile elaborate pentru controlul nodurilor au caracter general, astfel incat indiferent de structura STZA-ului care se doreste a fi condus, acestea pot fi utilizate fara a efectua modificari interne (reprogramare). Aceasta facilitate este asigurata prin parametrizarea taskurilor prin linia de comanda.

Comunicatia directa intre taskurile de control ale nodurilor, prin mesaje sau proxy, a fost eliminata complet deoarece in acesta situatie nu s-ar mai fi pastrat generalitatea taskurilor. Mecanismul utilizat pentru asigurarea comunicatiei directe intre taskurile de control ale nodurilor a fost implementat prin utilizarea memoriei partajate, fiecare task avand acces la zone bine stabilite, transmiterea acestor zone facandu-se prin lina de comanda in momentul lansarii in executie a taskului.

Rezultatele obtinute in urma testarii sistemului multitasking elaborat pentru implementarea modelelor de tip automat au validat si confirmat in totalitate solutia propusa.

In cazul implementarrii avand casuport hardware un PLC de tip Simatic S7-414, modelele considerate au fost de tip retele Petri. Pentru implementarea acestor modele s-au elaborat modele echivalente de tip SPIN. Ideea de baza in cazul acestei metode de implementare a constat in elaborarea unei functii prin intermediul careia sa se asigura conducerea unui nod de tip 1. Aceasta functie a fost utilizata apoi pentru implementarea tuturor nodurilor din sistem.

Implementarea modelelor SPIN, atat pentru nodul de tip 1 cat si pentru un STZA considerat, s-a realizat prin verificarea conditiei de executie a unei tranzitii. Datorita acestui fapt functia prin care s-a implementat STZA-ul considerat nu are un caracter general, o astfel de implementare depinzand in totalitate de structura sistemului. In schimb datorita faptului ca modelul nodului de tip 1 sta la baza modelarii tuturor nodurilor complexe, functia scrisa pentru acest nod este de tip general, prin parametrizare asigurandu-se particularitatile fiecarui nod.

Pentru implementarea modelelor de tip retele Petri s-a utilizat metoda STL, ceea ce a oferit o mai mare flexibilitate in programare decat metoda LAD. In literatura de specialitate studiata , implementarea unor modele de tip retele Petri s-a realizat exclusiv prin metoda LAD.

Rezultatele obtinute in urma testarii programului elaborat pentru implementarea modelelor de tip retele Petri au validat si confirmat in totalitate solutia propusa.

Capitolul 5

PetriTim – aplicatie software pentru analiza si simularea retelelor Petri

Aplicatia PetriTim,conceputa si dezvoltata de catre autor, este destinata analizei structurale si comportamentale a retelelor Petri, precum si pentru simularea acestora.. Aceasta aplicatie a aparut ca o necesitate a studierii în mod combinat (grafic si algebric) a modelelor de tip retele Petri.

Ceea ce deosebeste aplicatia PetriTim de restul programelor existente este faptul ca in modul de operare pas cu pas validarea executiei tranzitiilor se realizeaza prin validare de catre utilizator si nu in mod automat (aleator), putandu-se testa direct diferite circuite ale retelelor Petri. Pe langa facilitatea mai sus mentionata aplicatia asigura generarea unui fisier text in care se salveaza sub forma algebrica structura retelei Petri desenata, si de asemenea se mai genereaza un fisier care contine matricile de incidenta. Intreg procesul de simulare a retelei Petri este salvat sub forma matriciala intr-un fisier separat.

Validarea aplicatiei PetriTim s-a realizat prin simularea modelelor stabilite pentru nodurile de baza (capitolul 2), modele care in prealabil au fost validate cu ajutorul mediului Matlab. Rezultatele obtinute in urma simularilor cu programul PetriTim au fost identice cu cele obtinute in cazul utilizarii mediului Matlab.

5.1. Specificatiile aplicatiei

Aplicatia are ca principal scop reprezentarea într-un mod grafic a unei retele Petri definita prin (P,T,F,W,M_0). In acest sens, aplicatia permite construirea grafului prin utilizarea de obiecte grafice corespunzatoare elementelor fiecarei multimi din reprezentarea matematica a retelei Petri. Aceste obiecte sunt accesibile prin intermediul unui toolbar. Obiectele accesibile (necesare desenarii unei retele Petri) sunt: **Cursor**, **Place**, **Transition**, **Arc**, **Token**. Suprafata de desenare contine o retea de puncte (grid) distribuite pe linii şi coloane echidistante, elementele grafice (pozitiile şi tranzitiile) fiind centrate în cel mai apropiat punct al retelei fata de punctul în care au fost pozitionate.

Pozitiile ***P*** sunt plasate prin intermediul comenzii **Place**, ele fiind reprezentate prin cercuri, care pot avea de raze diferite, **Small** – reprezentare la scara redusa, **Normal** – reprezentare la scara normala, **Large** – reprezentare la scara extinsa, dimensiunea implicita fiind **Normal**.

Tranzitiile ***T*** sunt plasate prin intermediul comenzii **Transition**, ele fiind reprezentate prin suprafete dreptunghiulare negre, care pot avea dimensiuni diferite, **Small**, **Normal**, **Large,** dimensiunea implicita a unei tranzitii este **Normal** în pozitie orizontala.

Repozitionarea unuia dintre obiectele descrise mai sus se realizeaza prin interediul comenzii **Cursor**.

Trasarea unui arc de la o pozitie la o tranzitie sau de la o tranzitie la o pozitie se realizeaza prin intermediul comenzii **Arc**. Ponderea implicita a unui arc este 1, existand si posibilitatea de modificare a acesteia.

Adaugarea de jetoane într-o pozitie se realizeaza cu ajutorul comenzii **Token.** Un jeton este reprezentat prin intermediul unei buline, iar daca intr-o pozitie se afla mai multe jetoane ele for fi reprezentate corespunzator.

Aplicatia *PetriTim* ofera posibilitatea salvarii grafului retelei Petri desenate intr-un fisier cu extensia **.pnd**. Fisierul cuprinde multimea pozitiilor, a tranzitiilor, a arcelor, a ponderile acestora şi marcajul initial. Un alt fişier ce poate fi generat este cel care cuprinde matricile de incidenta (fişier cu extensia **.pnm**). În el se regasesc matricea de incidenta de ieşire, matricea de incidenta de intrare, matricea de incidenta (diferenta celor doua) şi matricea de incidenta transpusa. Cel de-al treilea fişier ce poate fi generat este cel referitor la ecuatia de stare (fişier cu extensia **.peq**). Acesta cuprinde marcajul initial, matricea de incidenta şi transpusa acesteia, vectorul de control şi starea urmatoare ca rezultat al ecuatiei de stare.

Pentru simularea unei retele Petri, aplicatia PetriTim pune la dispozitie comenziile:

- **Step** - utilizata pentru simularea pas cu pas, tranzitiile validate sunt prezentate prin intermediul unui dreptunghi albastru utilizatorul avand posibilitatea de a alege „traseul" dorit. Din acest mod se iese prin apasarea butonului **Stop** sau **Reset**.
- **Run** - utilizata pentru simularea in mod continuu a retelei Petri, tranzitiile fiind validate automat de aplicatie. Din acest mod se iese prin apasarea butonului **Stop** sau **Reset**.
- **Stop** - utilizata pentru oprirea simularii fara revenire in starea initiala.
- **Reset** - utilizata pentru aducerea retelei Petri simulata in starea initiala.

Pentru fiecare din obiectele grafice (pozitii, tranzitii, arce) sunt prevazute meniuri pop-up care se deschid în momentul în care un obiect este activat pentru editare. Pentru pozitii exista posibilitatea alegerii dimensiunii acestora, a numelui, a capacitatii, a numarului de jetoane, precum şi posibilitatea ştergerii acestora. Pentru tranzitii exista posibilitatea alegerii dimensiunii acestora, a numelui, a pozitiei (orizontala sau verticala), precum şi posibilitatea ştergerii acestora. Pentru arce, exista posibilitatea alegerii ponderii acestora şi a ştergerii lor.

Daca se doreşte ştergerea unui obiect, din meniul pop-up corespunzator, se selecteaza optiunea Delete. La stergerea unei pozitii sau a unei tranzitii arcele corespunzatoare sunt sterse automat.

5.2. Interfata cu utilizatorul

Interfata cu utilizatorul, a aplicatiei, este una prietenoasa, existand o bara de meniu care contine diferite optiuni, doua toolbar-uri, mai multe meniuri de context, precum şi o serie de ferestre de dialog care afişeaza sau permit setarea unor parametrii referitori la graful retelei Petri. Fereastra principala a aplicatiei este prezentata în figura 5.1.

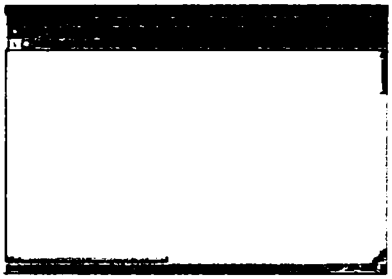


Figura 5.1. Aplicatia PetriTim – fereastra principala

5.3. Meniurile aplicatiei

Meniul **File** este prezentat în figura. 5.2 şi are urmatoarele optiuni:

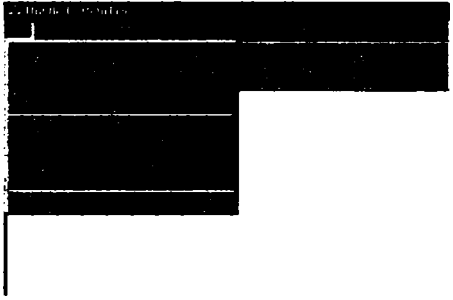


Figura 5.2. Meniul *File*

- **New** - realizeaza închiderea fişierului curent (daca este deschis vreunul) şi crearea unui nou fişier pentru desenarea unui graf nou.
- **Open** - deschide o fereastra de dialog în care utilizatorul poate alege un fişier deja creat care va fi incarcat (fişiere cu extensia **.pn**). Daca exista deja un fişier deschis, acesta va fi închis.
- **Save** - optiune care permite salvarea fişierului; daca fişierul este nou creat şi nu are un nume, se va deschide o fereastra de dialog în care se alege un nume pentru fişierul respectiv (la nume se adauga extensia **.pn**).
- **Save As** - deschide o fereastra de dialog în care utilizatorul poate alege un nume pentru fişierul ce urmeaza a fi salvat (la nume se adauga extensia **.pn**).
- **1 D:\Home...** - reprezinta ultimele fişiere deschise în cadrul aplicatiei.
- **Exit** - optiune care permite terminarea programului.

Meniul **View** este prezentat în figura 5.3 şi are urmatoarea structura:

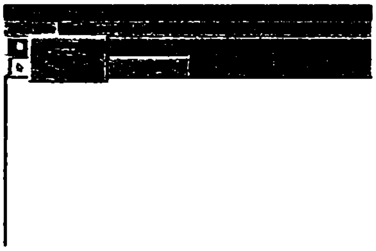


Figura 5.3. Meniul *View*

- **Toolbar** - permite afişarea/disparitia toolbar-ului principal.
- **Status Bar** - permite afişarea/disparitia barei de status de la baza ferestrei principale.
- **Paint** - optiune care permite afişarea sau disparitia toolbar-ului cu elemente grafice de retea Petri şi cu butoane pentru simularea dinamicii acestora.

Meniul **Run** este prezentat în figura 5.4 şi are urmatoarele optiuni: **Step , Run, Stop, Reset**.

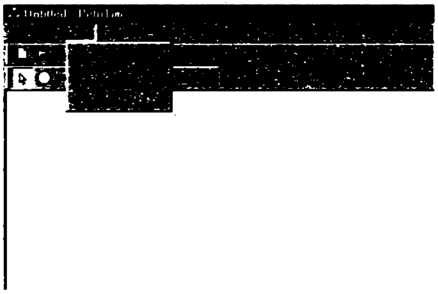


Figura 5.4. Meniul *Run*

Meniul **Generate** este prezentat în figura 5.5 şi are urmatoarea structura:

- **PN description** - are ca efect crearea unui fişier care cuprinde descrierea retelei Petri.

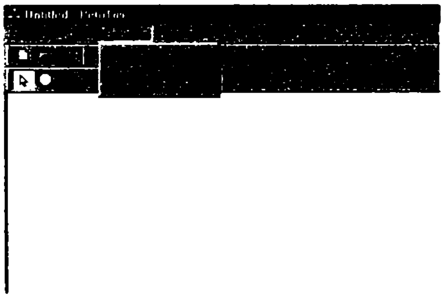


Figura 5.5. Meniul *Generate*

- **Incidence matrix** - are ca efect crearea unui fişier care cuprinde matricea de incidenta de intrare, matricea de incidenta de ieşire, matricea de incidenta (calculata ca diferenta a celor doua) şi matricea de incidenta transpusa.

- **State equation** - optiune care creaza un fişier ce cuprinde desrierea ecuatiei de stare. Fişierul va contine marcajul initial, matricea de incidenta, matricea de incidenta transpusa, vectorul de control şi starea urmatoare (marcajul urmator) a retelei Petri, stare rezultata din calculul ecuatiei de stare.

Meniul **Help** are o singura optiune şi este prezentat în figura 5.6.

- **About Petri...** - afişeaza informatii despre aplicatia PetriTim.

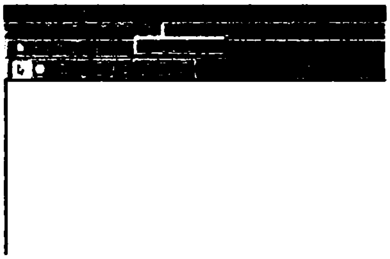


Figura 5.6. Meniul *Help*

Pe lânga meniurile de baza, mai exista o serie de meniuri pop-up, numite "meniuri de context" şi care apar în momentul în care se activeaza un obiect grafic (pozitie, tranzitie, arc).

În cazul **pozitiei**, meniul corespunzator este prezentat în figura 5.7.

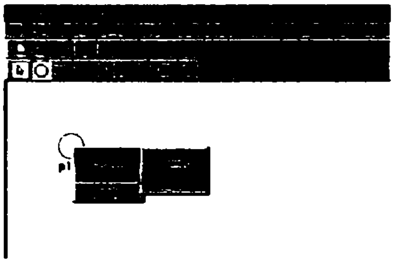


Figura 5.7. Meniul de context pentru pozitii

În cazul **tranzitiei**, meniul corespunzator este prezentat în figura 5.8.

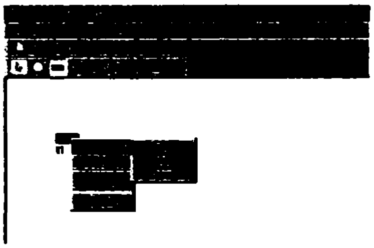


Figura 5.8. Meniul de context pentru tranzitii

Meniul corespunzator unui **arc** este prezentat în figura 5.9.

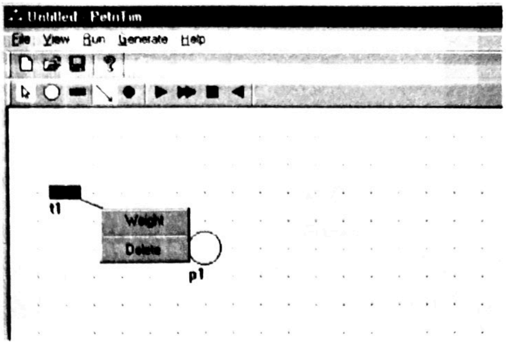


Figura 5.9. Meniul de context pentru arce

5.4. Toolbar-urile aplicatiei

In cadrul aplicatiei sunt implementate doua toolbar-uri şi un status bar care permite afişarea a diferite mesaje cu privire la butoanele şi meniurile aplicatiei. Butoanele toolbar-ului principal al aplicatiei sunt prezentate în figura 5.10.

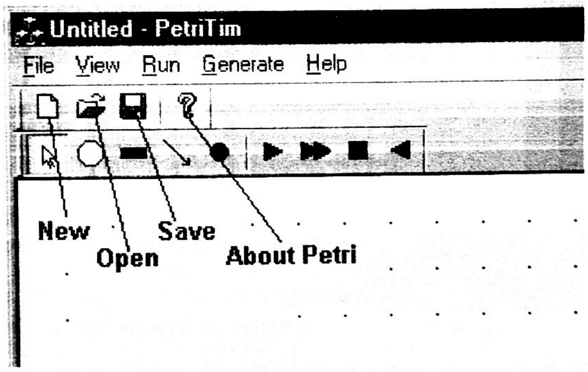


Figura 5.10. Toolbar-ul principal al aplicatiei PetriTim

Butoanele toolbar-ului numit "Paint" sunt prezentate în figura 5.11 şi sunt împartite în doua grupuri:

- grupul butoanelor de desenare – contine obiectele grafice necesare desenarii retelei Petri (*Cursor, Place, Transition, Arc, Token*).
- grupul butoanelor de simulare (*Step, Run, Stop, Reset*).

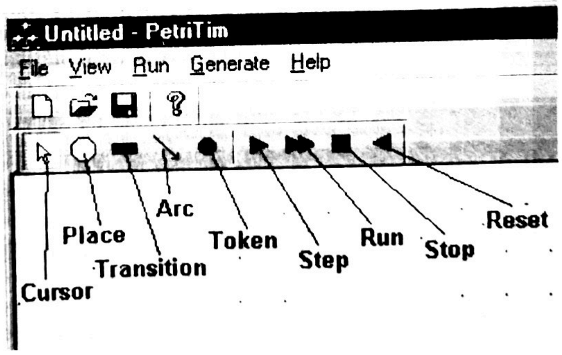


Figura 5.11. Toolbar-ul „paint" al aplicatiei PetriTim

5.5. Ferestre de dialog

Modificarea parametrilor corespunzatori unei pozitii se realizeaza cu ajutorul comenzii **Options**, fereastra de dialog corespunzatoare este prezentata în figura 5.12. Cu ajutorul acestei comenzi se pot schimba numele pozitiei, capacitatea şi numarul de jetoane corespunzatoare marcajului initial.

În cazul unei tranzitii, prin intermediul optiunii **Name**, utilizatorul poate schimba numele tranzitiei respective. Fereastra de dialog corespunzatoare este prezentata în figura 5.13.a.

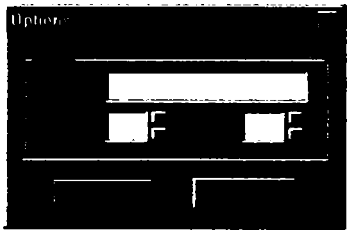


Figura 5.12. Fereatra de dialog pentru setarea optiunilor unei pozitii

a) b)

Figura 5.13. Fereatra de dialog pentru setarea numelui unei tranzitii (a) şi a ponderii unui arc (b)

În cazul arcelor, prin intermediul optiunii **Weight**, utilizatorul poate modifica ponderea arcului respectiv. In figura 5.13.b este prezentata fereastra corespunzatoare optiunii Weight..

5.6. Implementare

Programul a fost implementat utilizând mediul de dezvoltare Visual C++, si este bazat pe folosirea tehnicii orientata pe obiecte. Pentru implementare au fost definite mai multe clase: a obiectelor grafice (pozitie, tranzitie, arc), a grafului retelei Petri, a operatiilor matriciale, a fereastrelor de dialog.

*Clasa **Cplace** - corespunzatoare implementarii unei pozitii*

Structura corespunzatoare acestei clase este:

[illegible]

```
        UINT m_Tokens;
        UINT m_Capacity;
        CPlace(UINT x, UINT y, CString name);
        CString m_Name;
        UINT m_Y;
        UINT m_X;
        CPlace();
        virtual ~CPlace();
        BOOL operator==(CPlace place);
        BOOL operator!=(CPlace place);
        DECLARE_SERIAL(CPlace)
};
```

Membrii acestei clase sunt:

- **m_X** şi **m_Y** - reprezinta coordonatele centrului pozitiei corespunzatoare din graful reprezentat (membri de tipul **UINT**).
- **m_Name** - reprezinta numele pozitiei şi la creare, primeste un nume format din litera *p* şi un numar care reprezinta numarul pozitiei din graf (membru de tipul **CString**).
- **m_Capacity** - reprezinta capacitatea pozitiei respective (membru de tipul **UINT**).
- **m_Tokens** - reprezinta numarul de jetoane din pozitia respectiva (membru de tipul **UINT**).

Metodele acestei clase sunt:

- **CPlace()** - constructor fara argumente.
- **CPlace (UINT x, UINT y, CString name)** - un constructor care primeşte ca parametri coordonatele centrului pozitiei şi numele acesteia şi seteaza membrii corespunzatori cu aceste valori.
- **~CPlace()** - destructorul clasei **CPlace**.
- **operator==(CPlace place)** - reprezinta redefinirea operatorului **==**, iar ca tip returnat, **BOOL**.
- **operator!=(CPlace place)** - reprezinta redefinirea operatorului **!=**, iar ca tip returnat, **BOOL**.
- **Serialize(CArchive &ar)**, functie care se apeleaza la salvarea / încarcarea informatiilor corespunzatoare pozitiei într-un / dintr-un fişier.

*Clasa **Ctransition** – corespunzatoare implementarii unei tranzitii*

Structura acestei clase este:

```
class CTransition : public CObject
{
        DECLARE_SERIAL(CTransition);
public:
        BOOL m_Valid;
        void Serialize(CArchive& ar);
        BOOL m_Horiz;
        CTransition(UINT x, UINT y, CString name, BOOL horiz);
        CString m_Name;
        UINT m_X;
        UINT m_Y;
        CTransition();
```

```
        virtual ~CTransition();
        BOOL operator==(CTransition transition);
        BOOL operator!=(CTransition transition);

};
```

Membrii acestei clase sunt:

- **m_X** şi **m_Y** - reprezinta coordonatele centrului tranzitiei corespunzatoare din graful reprezentat (membri de tipul **UINT**).
- **m_Name** - reprezinta numele tranzitiei şi la creare i se da un nume format din litera *t* şi un numar care reprezinta numarul tranzitiei din graf graf (membru de tipul **CString**).
- **m_Horiz** - reprezinta un membru a carui valoare indica orientarea tranzitiei (orizontala sau verticala) graf (membru de tipul **BOOL**).

Metodele acestei clase sunt:

- **CTransition()** - constructor fara argumente.
- **CTransition (UINT x, UINT y, CString name, BOOL horiz)** - constructor care primeşte ca parametri coordonatele centrului tranzitiei, numele acesteia şi orientarea ei. Membrii corespunzatori vor fi setati cu aceste valori.
- **~CTransition()** - reprezinta destructorul clasei **CTransition**.
- **operator ==(CTransition transition)** - reprezinta redefinirea operatorului **==**, iar ca tip returnat, **BOOL**.
- **operator !=(CTransition transition)** - reprezinta redefinirea operatorului **!=**, iar ca tip returnat, **BOOL**.
- **Serialize(CArchive &ar)**, functie care se apeleaza la salvarea / încarcarea informatiilor corespunzatoare tranzitiei într-un / dintr-un fişier.

*Clasa **CArc** - corespunzatoare implementarii unui arc*

Structura acestei clase este:

```
class CArc : public CObject
{
        DECLARE_SERIAL(CArc);
public:
        void Serialize(CArchive& ar);
        CArc(CPlace* place, CTransition* transition, UINT sense, UINT weight);
        UINT m_Sense;
        UINT m_Weight;
        CTransition* m_pTransition;
        CPlace* m_pPlace;
        CArc();
        virtual ~CArc();

};
```

Membrii clasei sunt:

- **m_pPlace** - reprezinta un pointer la pozitia corespunzatoare arcului (un arc leaga obligatoriu o pozitie şi o tranzitie).
- **m_pTransition** - reprezinta un pointer la tranzitia corespunzatoare arcului.
- **m_Sense** - reprezinta sensul arcului şi ia urmatoarele valori:

    - **PLACE_TRANSITION** daca arcul este orientat de la pozitie la tranzitie;
    - **TRANSITION_PLACE** daca arcul e orientat invers (membru de tipul **UINT**).
- **m_Weight** - reprezinta ponderea arcului respectiv (membru de tipul **UINT**).

Metodele acestei clase sunt:

- **CArc()** - constructor fara argumente.
- **CArc (CPlace* place, CTransition* transition, UINT sense, UINT weight)** - constructor care primeşte ca parametri câte un pointer la pozitia şi tranzitia corespunzatoare arcului, sensul acestuia şi ponderea lui.
- **~CArc()** - reprezinta destructorul clasei CArc.
- **Serialize(CArchive &ar)** - functie care se apeleaza la salvarea / încarcarea informatiilor corespunzatoare arcului într-un / dintr-un fişier.

*Clasa **Cgraph** - corespunzatoare grafului retelei Petri*

Aceasta clasa are urmatoarea structura:

```
class CGraph : public CObject
{
public:
        BOOL IsValid(CTransition* pTransition);
        void Serialize(CArchive& ar);
        BOOL IsSingleArc(CPlace* place, CTransition* transition);
        CArc* GetArc(CPlace* pPlace, CTransition* pTransition, UINT sense);
        CArc* IsArc(CPoint point);
        void RemoveArc(CArc* pArc);
        void RemoveTransition(CTransition* pTransition);
        void RemovePlace(CPlace* pPlace);
        int m_IS[MAX_PLACES];
        int m_R;
        int m_H;
        int m_W;
        void GetRect(CTransition* transition, CRect* rect);
        void GetRect(CPlace* place, CRect* rect);
        CPlace* IsPlace(CPoint point);
        CTransition* IsTransition(CPoint point);
        int AddArc(CArc* pArc);
        int AddTransition(CTransition* pTransition);
        void DrawGraph(CDC* pDC);
        BOOL IsValidArc(CPlace* place, CTransition* transition, UINT sense);
        BOOL IsValidTransition(CPoint point, CRect rect, CTransition* transition);
        BOOL IsValidPlace(CPoint point, CRect rect, CPlace* place);
        int AddPlace(CPlace* pPlace);
        CArray <CTransition*, CTransition*> m_Transitions;
        CArray <CPlace*, CPlace*> m_Places;
        CArray <CArc*, CArc*> m_Arcs;
        CGraph();
        virtual ~CGraph();
```

```
private:
        void DrawArc(CDC* pDC, CPoint a, CPoint b, double beta,UINT weight);
        DECLARE_SERIAL(CGraph)
};
```

Membrii acestei clase sunt:

- **m_Places** - reprezinta un tablou de pointeri la pozitiile din graful reprezentat.
- **m_Transitions** - reprezinta un tablou de pointeri la tranzitiile din graful reprezentat.
- **m_Arcs -** reprezinta un tablou de pointeri la arcele din graful reprezentat.
- **m_R** - reprezinta raza pozitiilor din graf; toate pozitiile au la un moment dat aceeaşi dimensiune (membru de tipul **int**).
- **m_W** şi **m_H** - reprezinta jumatatea lungimii respectiv jumatatea latimii tranzitiilor (membri de tipul **int**).
- **m_IS** - reprezinta un tablou de întregi care are rol de a memora marcajul initial necesar la aducerea grafului la starea initiala ca urmare a optiunii **Reset**.

Metodele acestei clase sunt:

- **CGraph()** - constructor fara argumente, în care se fixeaza nişte valori initiale pentru **m_R** (**Normal**), **m_W** (**Normal**), **m_H** (**Normal**), iar marcajul initial va fi [0,0,...,0].
- **~CGraph()** - reprezinta destructorul clasei **CGraph**.
- **AddArc(CArc* pArc)** - metoda care adauga un arc la tabloul **m_Arcs** şi returneaza indexul lui în tablou.
- **AddPlace(CPlace* pPlace)** - metoda care adauga o pozitie la tabloul **m_Places** şi returneaza indexul ei în tablou.
- **AddTransition(CTransition* pTransition)** - metoda care adauga o tranzitie la tabloul **m_Transitions** şi returneaza indexul ei în tablou.
- **DrawArc(CDC *pDC, CPoint a, CPoint b, double beta, UINT weight)** - functie care deseneaza un arc de la punctul **a** la punctul **b**.
- **DrawGraph(CDC* pDC)** - functie care deseneaza graful retelei Petri. Un exemplu de graf desenat se poate observa în figura 5.14.
- **GetArc(CPlace *pPlace, CTransition *pTransition, UINT sense)** - aceasta functie returneaza un pointer la arcul ce uneşte pozitia şi tranzitia trimise ca parametri şi are sensul **sense**. Daca nu exista un astfel de arc, se returneaza **NULL**.

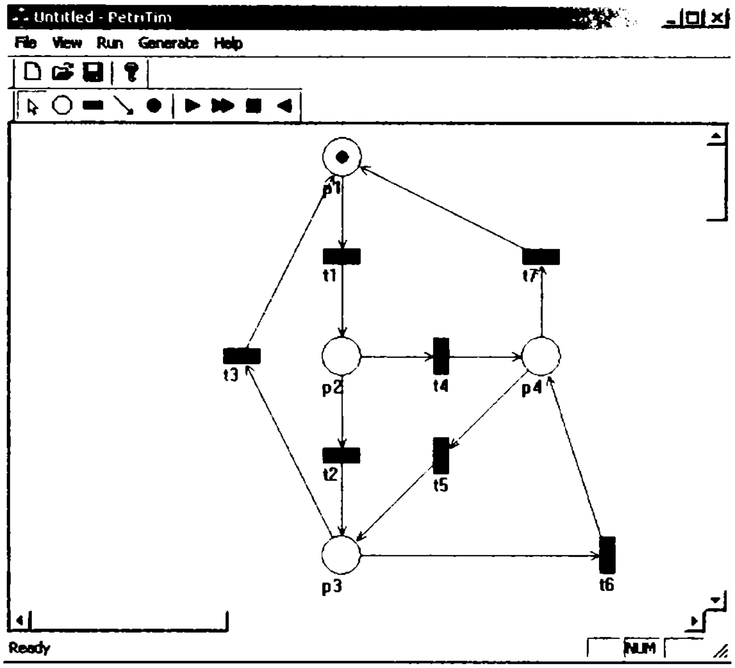


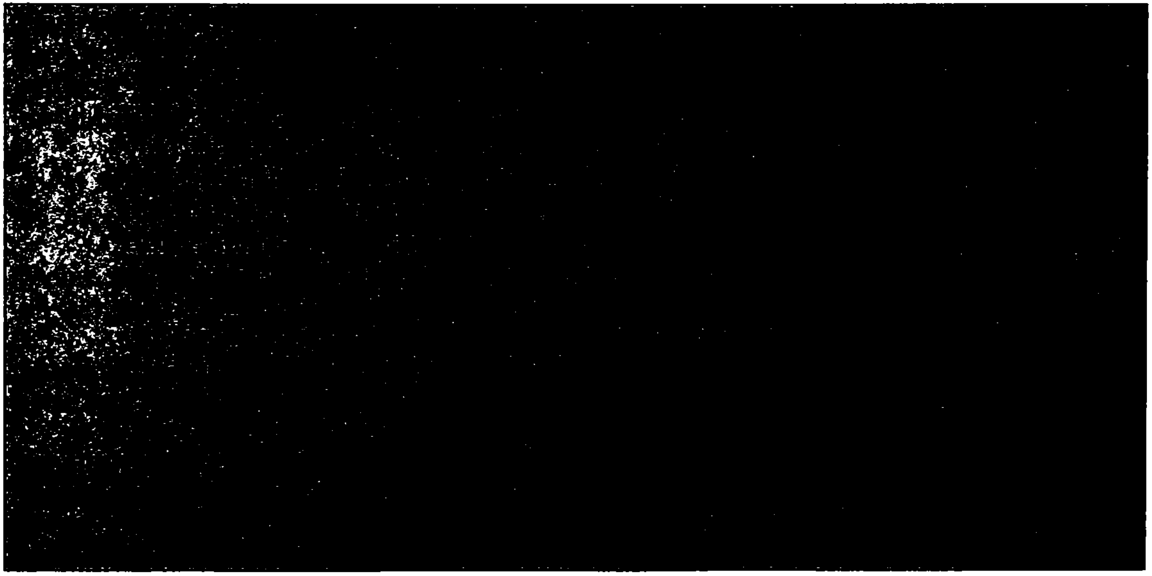
Figura 5.14. Un exemplu de graf desenat în aplicatia PetriTim

- **GetRect(CPlace* pPlace, CRect* rect)** - functie care returneaza în variabila **rect** suprafata cea mai mica care cuprinde pozitia **pPlace** şi celelate tranzitii care sunt legate printr-un arc de aceasta pozitie. Este folosita de functia **InvalidateRect(rect)** care va redesena suprafata respectiva.
- **GetRect(CTransition* pTransition, CRect* rect)** - functie care returneaza în variabila **rect** suprafata cea mai mica care cuprinde tranzitia **pTransition** şi celelate pozitii care sunt legate printr-un arc de aceasta tranzitie. Este folosita de functia **InvalidateRect(rect)** care va redesena suprafata respectiva.
- **IsArc(CPoint point)** - functie care determina daca prin punctul **point** trece vreun arc. Daca da, se va returna un pointer la acel arc, daca nu, se va returna **NULL**.
- **IsPlace(CPoint point)** - functie care verifica daca în punctul **point** exista sau nu o pozitie. Daca da, va returna un pointer la acea pozitie, daca nu, se returneaza **NULL**.
- **IsTransition(CPoint point)** - functie care determina daca în punctul **point** exista sau nu o tranzitie. Daca da, va returna un pointer la acea tranzitie, daca nu, se returneaza **NULL**.
- **IsSingleArc(CPlace* place, CTransition* transition)** - functie care determina daca între pozitia **place** şi tranzitia **transition** exista un sau mai multe arce, tipul rezultat fiind **BOOL**.
- **IsValid(CTransition* pTransition)** - functie care determina daca tranzitia **pTransition** este validata sau nu prin verificarea urmatoarelor conditii: numarul de jetoane al fiecarei pozitii de intrare trebuie sa fie mai mare sau egal cu ponderea arcului corespunzator, iar capacitatea fiecarei pozitii de ieşire trebuie sa fie mai mare sau egala cu suma jetoanelor acestora şi a ponderii arcului corespunzatoare.

- **IsValidArc(CPlace* place, CTransition* transition, UINT sense)** - functie care determina daca intre tranzitia şi pozitia transmise ca parametri nu exista deja un arc cu sensul **sense**, tipul returnat fiind **BOOL**.
- **IsValidPlace(CPoint point, CRect rect, CPlace* place)** - functie care determina daca în punctul **point** şi în suprafata **rect** poate exista o pozitie. Când se creeaza o pozitie noua, parametrul **place** va fi **NULL**. Când se schimba locatia unei pozitii, acest parametru reprezinta chiar pointerul la pozitia respectiva.
- **IsValidTransition(CPoint point, CRect rect, CTransition* transition)** - functie care determina daca în punctul **point** şi în suprafata **rect** poate exista o tranzitie. Când se creeaza o tranzitie noua, parametrul **transition** va fi **NULL**. Când se schimba locatia unei tranzitii, acest parametru reprezinta chiar pointerul la tranzitia respectiva.
- **RemoveArc(CArc* pArc)** - functie care sterge arcul **pArc** din tabloul **m_Arcs**.
- **RemovePlace(CPlace* pPlace)** - functie care sterge pozitia **pPlace** din tabloul **m_Places**. De asemenea, aceasta functie sterge şi arcele care leaga aceasta pozitie de tranzitii.
- **RemoveTransition(CTransition* pTransition)** - functie care sterge tranzitia **pTransition** din tabloul **m_Transitions**. De asemenea, aceasta functie sterge şi arcele care leaga aceasta tranzitie de pozitii.
- **Serialize(CArchive &ar)** - functie apelata la salvarea / încarcarea grafului retelei Petri într-un / dintr-un fişier.

*Clasa **Cmatrix** - corespunzatoare operatiior matriciale*

Structura acestei clase este:



Membrii acestei clase sunt:

- **m_uLines** - reprezinta numarul de linii ale matricii (membru de tipul **UINT**).
- **m_uCols** - reprezinta numarul de coloane ale matricii (membru de tipul **UINT**).

- **m_Matrix** - reprezinta un tablou bidimensional de întregi pentru memorarea valorilor elementelor matricii.

Metodele acestei clase sunt:

- **CMatrix()** - constructor fara argumente.
- **CMatrix(UINT uLines, UINT uCols)** - constructor care primeşte ca parametri numarul de linii şi de coloane al matricii respective.
- **~CMatrix()** - destructorul clasei CMatrix.
- **SetElem(UINT uLine, UINT uCol, int nVal)** - functie care atribuie elementului din linia **uLine** şi coloana **uCol** valoarea **nVal**.
- **GetElem(UINT uLine, UINT uCol)** - functie care returneaza valoarea elementului din linia **uLine** şi coloana **uCol** (tipul **int**).
- **AddMatrix(CMatrix matrix)** - functie care realizeaza adunarea a doua matrici. Daca dimensiunile celor doua matrici nu coincid, functia va returna **false**, altfel ea returneaza **true**.
- **SubMatrix(CMatrix matrix)** - functie care realizeaza scaderea a doua matrici. Daca dimensiunile celor doua matrici nu coincid, functia va returna **false**, altfel ea returneaza **true**.
- **MulMatrix(CMatrix matrix)** - functie care realizeaza înmultirea a doua matrici. Daca numarul de coloane al obiectului pentru care se face apelul de functie nu coincide cu numarul de linii al matricii matrix, se va returna **false**, altfel se returneaza **true**.
- **TranspMatrix()** - functie care calculeaza transpusa unei matrici.

*Clasa **CplaceOptDlg** – corespunzatoare unei ferestre de dialog pentru o pozitie*

Aceasta clasa are urmatoarea structura:

```
class CPlaceOptDlg : public CDialog
{
// Construction
public:
        CPlaceOptDlg(CWnd* pParent = NULL);   // standard constructor

// Dialog Data
        enum { IDD = IDD_PLACE_OPT };
        CSpinButtonCtrl        m_CapSpin;
        CSpinButtonCtrl        m_TokSpin;
        CString m_Name;
        UINT    m_Tokens;
        UINT    m_Capacity;

// Overrides
// ClassWizard generated virtual function overrides
        protected:
        virtual void DoDataExchange(CDataExchange* pDX);

// Implementation
protected:
// Generated message map functions
        virtual BOOL OnInitDialog();
        DECLARE_MESSAGE_MAP()
};
```

Membrii acestei clase sunt:

- **m_Name** - pentru câmpul "Name" şi permite schimbarea numelui pozitiei (membru de tipul **CString**).
- **m_Capacity** - pentru câmpul "Capacity" şi permite setarea capacitatii pozitiei (valoarea implicita a acestui câmp este 10, membru de tipul **UINT**).
- **m_Tokens** - pentru câmpul "Tokens" şi permite setarea numarului de jetoane în pozitia respectiva (în intervalul cuprins între 0 şi **Capacity,** membru de tipul **UINT**).
- **m_CapSpin** pentru controlul de tip spin asociat câmpului "Capacity".
- **m_TokSpin** pentru controlul de tip spin asociat câmpului "Tokens".

Metoda acestei clase este **OnInitDialog()**.

*Clasa **CtransOptDlg** – corespunzatoare unei ferestre de dialog pentru o tranzitie*

Aceasta clasa are urmatoarea structura:

```
class CTransOptDlg : public CDialog
{
// Construction
public:
        CTransOptDlg(CWnd* pParent = NULL);   // standard constructor

// Dialog Data
        enum { IDD = IDD_TRANS_NAME };
        CString m_Name;

// Overrides
        // ClassWizard generated virtual function overrides
        protected:
        virtual void DoDataExchange(CDataExchange* pDX);

// Implementation
protected:
        // Generated message map functions
        // NOTE: the ClassWizard will add member functions here
        DECLARE_MESSAGE_MAP()
};
```

Acesta clasa are un singur membru:

- **m_Name** care este o mapare pentru câmpul "Name" şi care permite schimbarea numelui tranzitiei (membru de tipul **CString**).

*Clasa **CarcOptDlg** – corespunzatoare unei ferestre de dialog pentru un arc*

Aceasta clasa are urmatoarea structura:

```
class CArcOptDlg : public CDialog
{
// Construction
public:
        CArcOptDlg(CWnd* pParent = NULL);   // standard constructor
```

```
// Dialog Data
        enum { IDD = IDD_ARC_WEIGHT };
        CSpinButtonCtrl         m_WeightSpin;
        UINT    m_Weight;

// Overrides
        // ClassWizard generated virtual function overrides
        protected:
        virtual void DoDataExchange(CDataExchange* pDX);

// Implementation
protected:
        // Generated message map functions
        virtual BOOL OnInitDialog();
        DECLARE_MESSAGE_MAP()
};
```

Membrii acestei clase sunt:

- **m_Weight -** este o mapare pentru câmpul "Weight" şi care permite setarea ponderii arcului respectiv (ponderea implicita pentru fiecare arc este 1, membru de tipul **UINT**).
- **m_WeightSpin** - reprezinta o mapare pentru controlul de tip spin asociat câmpului "Weight".

Metoda acestei clase este **OnInitDialog()**.

In plus, mai exista o serie de clase, generate în mod automat de catre Visual C++ la crearea unei aplicatii de tip SDI (Single Document Interface), asa cum este PetriTim: **CMainFrame**, **CPetriTimApp**, **CPetriTimDoc** şi **CPetriTimView**. În ultima clasa (*CpetriTimView*) se gasesc functiile care intercepteaza diferite mesaje generate de alegerea unei optiuni de meniu sau de un click de mouse. În continuare se vor descrie o parte din aceste functii.

- **OnButtonArc()** - functie care se apeleaza la apasarea butonului **Arc** din toolbar-ul **Paint** şi care atribuie membrului acestei clase, **m_PaintObj**, valoarea **ARC**. Acest buton permite desenarea de arce intre pozitii şi tranzitii.
- **OnButtonCursor()** - functie care se apeleaza la apasarea butonului **Cursor** din toolbar-ul **Paint** şi care atribuie membrului acestei clase, **m_PaintObj**, valoarea **CURSOR**. Acest buton trebuie apasat în momentul în care se doreşte mutarea unui obiect gata desenat.
- **OnButtonPlace()** - functie care se apeleaza la apasarea butonului **Place** din toolbar-ul **Paint** şi care atribuie membrului acestei clase, **m_PaintObj**, valoarea **PLACE**. Acest buton permite desenarea de pozitii.
- **OnButtonTransition()** - functie care se apeleaza la apasarea butonului **Transition** din toolbar-ul **Paint** şi care atribuie membrului acestei clase, **m_PaintObj**, valoarea **TRANSITION**. Acest buton permite desenarea de tranzitii.
- **OnButtonRun()** - functie care se apeleaza la apararea butonului **Run** şi creeaza un fir de executie **CreateThread(NULL, 0, FuncThread, this, 0, &dwId)**.
- **OnButtonStep()** - functie care se apeleaza la apasarea butonului **Step**. Acesta functie evidentiaza toate tranzitiile validate prin schimbarea culorii

tranzitiilor în albastru (un dreptunghi albastru). În acest caz, simularea se va face printr-un click pe tranzitia care se doreşte a fi executata.

- **OnButtonReset()** - functie care se apeleaza la apasarea butonului **Reset** şi opreşte firul de executie creat la apasarea butonului **Run** sau opreşte simularea pas cu pas initiata la apasarea butonului **Step**. Dupa aceste operatii, se reface marcajul initial şi se redeseneaza graful.
- **OnButtonStop()** - functie care se apeleaza la apasarea butonului **Stop** şi opreşte firul de executie creat la apasarea butonului **Run** (daca a fost creat vreunul) sau opreşte simularea pas cu pas initiata la apasarea butonului **Step.**
- **OnGeneratePnfile()** - functie care se apeleaza la alegerea din meniul **Generate** a optiunii **PN description**. Pe ecran va aparea o fereastra de dialog în care utilizatorului i se cere sa introduca un nume pentru fişierul în care se va salva structura retelei Petri. Acest fişier are urmatoarea structura:

```
%Fisierul descrie reteaua Petri
%P - multimea pozitiilor
P = {p1,p2,p3,p4}
%T - multimea tranzitiilor
T = {t1,t2,t3,t4,t5,t6,t7}
%F - multimea arcelor
F = {(p1,t1),(p2,t2),(p2,t4),(p3,t3),(p3,t6),(p4,t5),(p4,t7),(t1,p2),(t2,p3),
(t3,p1),(t4,p4),(t5,p3),(t6,p4),(t7,p1)}
%w - functia de ponderare a arcelor
w(p1,t1) = 1;w(p2,t2) = 1;w(p2,t4) = 1;w(p3,t3) = 1;w(p3,t6) = 1;
w(p4,t5) = 1;w(p4,t7) = 1;w(t1,p2) = 1;w(t2,p3) = 1;w(t3,p1) = 1;
w(t4,p4) = 1;w(t5,p3) = 1;w(t6,p4) = 1;w(t7,p1) = 1;
%M0 - marcajul initial
M0 = [1,0,0,0];
end.
```

- **OnGenerateMatrix()** - functie care se apeleaza la alegerea din meniul **Generate** a optiunii **Incidence matrix**. Pe ecran va aparea o fereastra de dialog în care utilizatorului i se cere sa introduca un nume pentru fişierul în care se vor salva matricile corespunzatoare retelei Petri. Acest fişier are urmatoarea structura:

```
%Fisierul descrie matricile de incidenta ale retelei Petri
%Matricea de incidenta de iesire (A+)
        || 0 1 0 0 ||
        || 0 0 1 0 ||
        || 1 0 0 0 ||
        || 0 0 0 1 ||
        || 0 0 1 0 ||
        || 0 0 0 1 ||
        || 1 0 0 0 ||
%Matricea de incidenta de intrare (A-)
        || 1 0 0 0 ||
        || 0 1 0 0 ||
        || 0 0 1 0 ||
```

```
        || 0 1 0 0 ||
        || 0 0 0 1 ||
        || 0 0 1 0 ||
        || 0 0 0 1 ||

%Matricea de incidenta A = (A+) - (A-)
        || -1  1  0  0 ||
        ||  0 -1  1  0 ||
        ||  1  0 -1  0 ||
        ||  0 -1  0  1 ||
        ||  0  0  1 -1 ||
        ||  0  0 -1  1 ||
        ||  1  0  0 -1 ||
%Matricea de incidenta transpusa At
        || -1  0  1  0  0  0  1 ||
        ||  1 -1  0 -1  0  0  0 ||
        ||  0  1 -1  0  1 -1  0 ||
        ||  0  0  0  1 -1  1 -1 ||
end.
```

- **OnGenerateStateeq()** - functie care se apeleaza la alegerea din meniul **Generate** a optiunii **State equation**. Fişierul generat are urmatoarea structura:

```
%Fisierul descrie
ecuatia de stare a
retelei Petri

%Marcajul initial M0
      || 1 ||
      || 0 ||
      || 0 ||
      || 0 ||

%Matricea de incidenta
A = (A+) - (A-)
      || -1  1  0  0 ||
      ||  0 -1  1  0 ||
      ||  1  0 -1  0 ||
      ||  0 -1  0  1 ||
      ||  0  0  1 -1 ||
      ||  0  0 -1  1 ||
      ||  1  0  0 -1 ||

%Matricea de incidenta
 transpusa At
|| -1  0  1  0  0  0  1 ||
||  1 -1  0 -1  0  0  0 ||
||  0  1 -1  0  1 -1  0 ||
||  0  0  0  1 -1  1 -1 ||

%Vector de control
      || 1 ||
      || 0 ||
      || 0 ||
      || 0 ||
      || 0 ||
      || 0 ||
      || 0 ||

%Starea urmatoare
      || 0 ||
      || 1 ||
      || 0 ||
      || 0 ||

(continuare 1)
%Vector de
control
      || 0 ||
      || 1 ||
      || 0 ||
      || 0 ||
      || 0 ||
      || 0 ||
      || 0 ||

%Starea
urmatoare
      || 0 ||
      || 0 ||
      || 1 ||
      || 0 ||

%Vector de
control
      || 0 ||
      || 0 ||
      || 1 ||
      || 0 ||
      || 0 ||
      || 0 ||
      || 0 ||

%Starea
urmatoare
      || 1 ||
      || 0 ||
      || 0 ||
      || 0 ||

%Vector de
control
      || 1 ||
      || 0 ||
      || 0 ||
      || 0 ||
      || 0 ||
      || 0 ||
      || 0 ||

(continuare 2)
%Starea
urmatoare
      || 0 ||
      || 1 ||
      || 0 ||
      || 0 ||

%Vector de
control
      || 0 ||
      || 0 ||
      || 0 ||
      || 1 ||
      || 0 ||
      || 0 ||
      || 0 ||

%Starea
urmatoare
      || 0 ||
      || 0 ||
      || 0 ||
      || 1 ||

%Vector de
control
      || 0 ||
      || 0 ||
      || 0 ||
      || 0 ||
      || 1 ||
      || 0 ||
      || 0 ||

%Starea
urmatoare
      || 0 ||
      || 0 ||
      || 1 ||
      || 0 ||

(continuare 3)
%Vector de
control
      || 0 ||
      || 0 ||
      || 0 ||
      || 0 ||
      || 0 ||
      || 1 ||
      || 0 ||

%Starea
urmatoare
      || 0 ||
      || 0 ||
      || 0 ||
      || 1 ||

%Vector de
control
      || 0 ||
      || 0 ||
      || 0 ||
      || 0 ||
      || 0 ||
      || 0 ||
      || 1 ||

%Starea
urmatoare
      || 1 ||
      || 0 ||
      || 0 ||
      || 0 ||
```

- **DoStep(CTransition* pTransition)** - functie care executa un pas din simularea grafului retelei Petri executând tranzitia **pTransition**, daca aceasta este validata. Dupa ce se initiaza simularea pas cu pas, se evidentiaza tranzitiile validate, iar în momentul în care se activeaza una din ele, se apeleaza functia **DoStep** pentru a o executa

5.7. Modelarea nodurilor de baza utilizand PetriTim

Validarea aplicatiei PetriTim s-a realizat prin testarea modelelor de baza stabilite pentru nodurile definite in capitolul 2.

Validarea modelului de tip retea Petri cu ajutorul programului PetriTim s-a realizat in doua moduri:

- prin vizualizarea in mod grafic a functionarii in regim pas cu pas;
- prin urmarire directa si analiza fisierului *n1.peq*.

In urma simularii functionarii unui model, pe baza fisierului *n1.peq*, interpretand marcajul si vectorul de control se poate construi arborele de acoperire.

Pentru exemplificarea modului in care s-a efectuat verificarea corectitudinii modului de operare al aplicatiei PetriTim se considera modelul de tip retea Petri corespunzator nodului de tip 1.

5.7.1. Testare model nod de tip 1

In figura 5.16 se prezinta modelul de tip retea Petri a nodului de tip 1 in PetriTim.

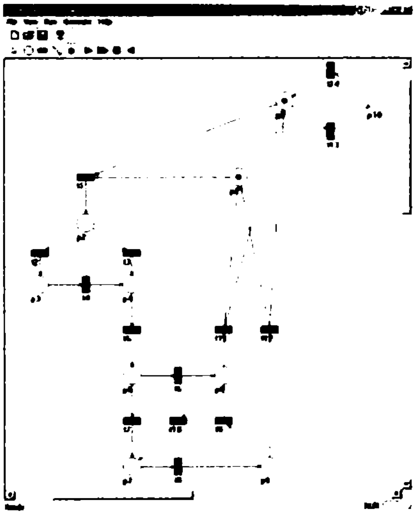


Figura. 5.16. Modelul corespunzator nodului de tip 1 in PetriTim

Fisierul corespunzator descrierii retelei Petri are urmatoarea structura:

```
%Fisierul descrie reteaua Petri
%P - multimea pozitiilor
P = {p1,p2,p3,p4,p5,p6,p7,p8,p9,p10}
%T - multimea tranzitiilor
T = {t1,t2,t3,t4,t5,t6,t7,t8,t9,t10,t11,t12,t13,t14}
%F - multimea arcelor
F={(p1,t1),(p2,t2),(p2,t3),(p3,t4),(p4,t5),(p5,t6),(p5,t7),(p6,t10),(p6,t11),(p7,t8),
(p8,t9),(p8,t12),(p9,t1),(p9,t13),(p10,t14),
(t1,p2),(t2,p3),(t3,p4),(t4,p4),(t5,p5),(t6,p6),(t7,p7),(t8,p8),(t9,p6),(t10,p7),
(t11,p1),(t11,p9),(t12,p1),(t12,p9),(t13,p10),(t14,p9)}
%w - functia de ponderare a arcelor
w(p1,t1) = 1; w(p2,t2) = 1; w(p2,t3) = 1; w(p3,t4) = 1;
w(p4,t5) = 1; w(p5,t6) = 1; w(p5,t7) = 1; w(p6,t10) = 1;w(p6,t11) = 1;
w(p7,t8) = 1;w(p8,t9) = 1;w(p8,t12) = 1;w(p9,t1) = 1;
w(p9,t13) = 1;w(p10,t14) = 1;w(t1,p2) = 1;w(t2,p3) = 1;w(t3,p4) = 1;
w(t4,p4) = 1;w(t5,p5) = 1;w(t6,p6) = 1;w(t7,p7) = 1;
w(t8,p8) = 1;w(t9,p6) = 1;w(t10,p7) = 1;w(t11,p1) = 1;w(t11,p9) = 1;
w(t12,p1) = 1;w(t12,p9) = 1;w(t13,p10) = 1;w(t14,p9) = 1;
%M0 - marcajul initial
M0 = [1,0,0,0,0,0,0,0,1,0];
end.
```

Comparand cele doua modele (cel stabilit si validat in capitolul 2 si cel generat de PetriTim) se constata ca ele sunt identice.

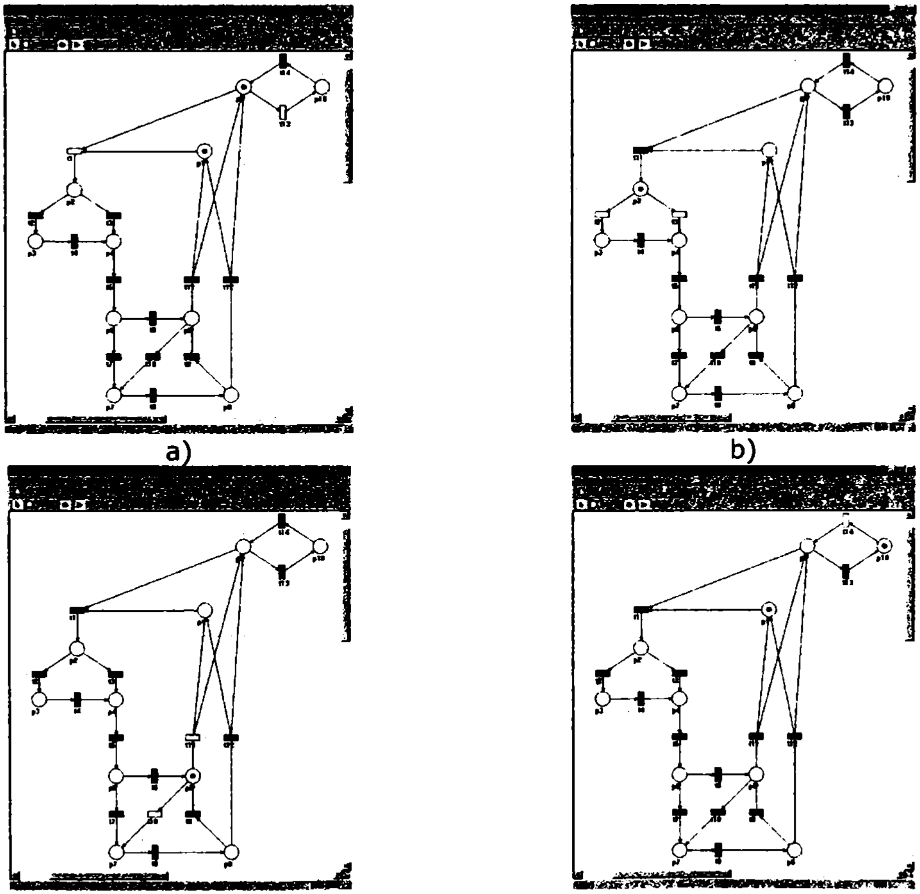
Matricile de incidenta generate de PetriTim sunt:

```
%Fisierul descrie matricile de
incidenta ale retelei Petri

%Matricea de incidenta de iesire (A+)
          || 0 1 0 0 0 0 0 0 0 0 ||
          || 0 0 1 0 0 0 0 0 0 0 ||
          || 0 0 0 1 0 0 0 0 0 0 ||
          || 0 0 0 1 0 0 0 0 0 0 ||
          || 0 0 0 0 1 0 0 0 0 0 ||
          || 0 0 0 0 0 1 0 0 0 0 ||
          || 0 0 0 0 0 0 1 0 0 0 ||
          || 0 0 0 0 0 0 0 1 0 0 ||
          || 0 0 0 0 0 1 0 0 0 0 ||
          || 0 0 0 0 0 0 1 0 0 0 ||
          || 1 0 0 0 0 0 0 0 1 0 ||
          || 1 0 0 0 0 0 0 0 1 0 ||
          || 0 0 0 0 0 0 0 0 0 1 ||
          || 0 0 0 0 0 0 0 0 1 0 ||


%Matricea de incidenta de intrare (A-)
          || 1 0 0 0 0 0 0 0 1 0 ||
          || 0 1 0 0 0 0 0 0 0 0 ||
          || 0 1 0 0 0 0 0 0 0 0 ||
          || 0 0 1 0 0 0 0 0 0 0 ||
          || 0 0 0 1 0 0 0 0 0 0 ||
```

```
%Matricea de incidenta A = (A+) - (A-)
          || -1  1  0  0  0  0  0  0 -1  0 ||
          ||  0 -1  1  0  0  0  0  0  0  0 ||
          ||  0 -1  0  1  0  0  0  0  0  0 ||
          ||  0  0 -1  1  0  0  0  0  0  0 ||
          ||  0  0  0 -1  1  0  0  0  0  0 ||
          ||  0  0  0  0 -1  1  0  0  0  0 ||
          ||  0  0  0  0 -1  0  1  0  0  0 ||
          ||  0  0  0  0  0  0 -1  1  0  0 ||
          ||  0  0  0  0  0  1  0 -1  0  0 ||
          ||  0  0  0  0  0 -1  1  0  0  0 ||
          ||  1  0  0  0  0 -1  0  0  1  0 ||
          ||  1  0  0  0  0  0  0 -1  1  0 ||
          ||  0  0  0  0  0  0  0  0 -1  1 ||
          ||  0  0  0  0  0  0  0  0  1 -1 ||


%Matricea de incidenta transpusa At
|| -1  0  0  0  0  0  0  0  0  0  1  1  0  0 ||
||  1 -1 -1  0  0  0  0  0  0  0  0  0  0  0 ||
||  0  1  0 -1  0  0  0  0  0  0  0  0  0  0 ||
||  0  0  1  1 -1  0  0  0  0  0  0  0  0  0 ||
||  0  0  0  0  1 -1 -1  0  0  0  0  0  0  0 ||
```

Comparand aceste matrici cu cele obtinute prin modelarea in Matlab (paragraful 2.4.1.1) se observa ca matricile generate de PetriTim sunt identice cu cele obtinute in urma simularii modelului in Matlab.

In figura 5.17 se prezinta executia a patru pasi in timpul efectuarii simularii retelei.



a) b) c) d)

Figura 5.17. Executia pasilor in timpul simularii cu PetriTim

In figura 5.17 se poate observa cum tranzitile care sunt validate la un moment dat sunt activate. Printr-un simplu click pe tranzitia dorita aceasta este validata si executata.

Descrierea executie retelei, pe baza ecuatiei de stare, generata de PetriTim este (fisierul *n1.eq*):

```
%Fisierul descrie
ecuatia de stare a
retelei Petri

%Marcajul    initial
M0
    || 1 ||
    || 0 ||
    || 0 ||
    || 0 ||
    || 0 ||
    || 0 ||
    || 0 ||
    || 0 ||
    || 1 ||
    || 0 ||
```

```
%Matricea de incidenta
A = (A+) - (A-)

|| -1  1  0  0  0  0  0  0 -1  0 ||
||  0 -1  1  0  0  0  0  0  0  0 ||
||  0 -1  0  1  0  0  0  0  0  0 ||
||  0  0 -1  1  0  0  0  0  0  0 ||
||  0  0  0 -1  1  0  0  0  0  0 ||
||  0  0  0  0 -1  1  0  0  0  0 ||
||  0  0  0  0 -1  0  1  0  0  0 ||
||  0  0  0  0  0  0 -1  1  0  0 ||
||  0  0  0  0  0  1  0 -1  0  0 ||
||  0  0  0  0  0 -1  1  0  0  0 ||
||  1  0  0  0  0 -1  0  0  1  0 ||
||  1  0  0  0  0  0  0 -1  1  0 ||
||  0  0  0  0  0  0  0  0 -1  1 ||
||  0  0  0  0  0  0  0  0  1 -1 ||
```

```
%Matricea de incidenta transpusa At

|| -1  0  0  0  0  0  0  0  0  0  1  1  0  0 ||
||  1 -1 -1  0  0  0  0  0  0  0  0  0  0  0 ||
||  0  1  0 -1  0  0  0  0  0  0  0  0  0  0 ||
||  0  0  1  1 -1  0  0  0  0  0  0  0  0  0 ||
||  0  0  0  0  1 -1 -1  0  0  0  0  0  0  0 ||
||  0  0  0  0  0  1  0  0  1 -1 -1  0  0  0 ||
||  0  0  0  0  0  0  1 -1  0  1  0  0  0  0 ||
||  0  0  0  0  0  0  0  1 -1  0  0 -1  0  0 ||
|| -1  0  0  0  0  0  0  0  0  0  1  1 -1  1 ||
||  0  0  0  0  0  0  0  0  0  0  0  0  1 -1 ||
```

```
%Vector de control
    ||  1 ||
    ||  0 ||
    ||  0 ||
    ||  0 ||
    ||  0 ||
    ||  0 ||
    ||  0 ||
    ||  0 ||
    ||  0 ||
    ||  0 ||
    ||  0 ||
    ||  0 ||
    ||  0 ||
    ||  0 ||

%Starea urmatoare
    ||  0 ||
    ||  1 ||
    ||  0 ||
    ||  0 ||
    ||  0 ||
    ||  0 ||
    ||  0 ||
    ||  0 ||
    ||  0 ||
    ||  0 ||
```

```
%Starea urmatoare
    ||  0 ||
    ||  0 ||
    ||  0 ||
    ||  1 ||
    ||  0 ||
    ||  0 ||
    ||  0 ||
    ||  0 ||
    ||  0 ||
    ||  0 ||

%Vector de control
    ||  0 ||
    ||  0 ||
    ||  0 ||
    ||  0 ||
    ||  1 ||
    ||  0 ||
    ||  0 ||
    ||  0 ||
    ||  0 ||
    ||  0 ||
    ||  0 ||
    ||  0 ||
    ||  0 ||
    ||  0 ||
```

```
%Vector de control
    ||  0 ||
    ||  0 ||
    ||  0 ||
    ||  0 ||
    ||  0 ||
    ||  0 ||
    ||  0 ||
    ||  1 ||
    ||  0 ||
    ||  0 ||
    ||  0 ||
    ||  0 ||
    ||  0 ||
    ||  0 ||

%Starea urmatoare
    ||  0 ||
    ||  0 ||
    ||  0 ||
    ||  0 ||
    ||  0 ||
    ||  0 ||
    ||  0 ||
    ||  1 ||
    ||  0 ||
    ||  0 ||
```

```
%Vector de control
|| 0 ||
|| 1 ||
|| 0 ||
|| 0 ||
|| 0 ||
|| 0 ||
|| 0 ||
|| 0 ||
|| 0 ||
|| 0 ||
|| 0 ||
|| 0 ||
|| 0 ||
|| 0 ||

%Starea urmatoare
|| 0 ||
|| 0 ||
|| 1 ||
|| 0 ||
|| 0 ||
|| 0 ||
|| 0 ||
|| 0 ||
|| 0 ||
|| 0 ||
%Vector de control
|| 0 ||
|| 0 ||
|| 0 ||
|| 1 ||
|| 0 ||
|| 0 ||
|| 0 ||
|| 0 ||
|| 0 ||
|| 0 ||
|| 0 ||
|| 0 ||
|| 0 ||
|| 0 ||
```

```
%Starea urmatoare
|| 0 ||
|| 0 ||
|| 0 ||
|| 0 ||
|| 1 ||
|| 0 ||
|| 0 ||
|| 0 ||
|| 0 ||
|| 0 ||

%Vector de control
|| 0 ||
|| 0 ||
|| 0 ||
|| 0 ||
|| 0 ||
|| 0 ||
|| 1 ||
|| 0 ||
|| 0 ||
|| 0 ||
|| 0 ||
|| 0 ||
|| 0 ||
|| 0 ||

%Starea urmatoare
|| 0 ||
|| 0 ||
|| 0 ||
|| 0 ||
|| 0 ||
|| 0 ||
|| 1 ||
|| 0 ||
|| 0 ||
|| 0 ||
```

```
%Vector de control
|| 0 ||
|| 0 ||
|| 0 ||
|| 0 ||
|| 0 ||
|| 0 ||
|| 0 ||
|| 0 ||
|| 1 ||
|| 0 ||
|| 0 ||
|| 0 ||
|| 0 ||
|| 0 ||

%Starea urmatoare
|| 0 ||
|| 0 ||
|| 0 ||
|| 0 ||
|| 0 ||
|| 1 ||
|| 0 ||
|| 0 ||
|| 0 ||
|| 0 ||
%Vector de control
|| 0 ||
|| 0 ||
|| 0 ||
|| 0 ||
|| 0 ||
|| 0 ||
|| 0 ||
|| 0 ||
|| 0 ||
|| 0 ||
|| 1 ||
|| 0 ||
|| 0 ||
|| 0 ||
```

```
%Starea urmatoare
|| 1 ||
|| 0 ||
|| 0 ||
|| 0 ||
|| 0 ||
|| 0 ||
|| 0 ||
|| 0 ||
|| 1 ||
|| 0 ||

%Vector de control
|| 0 ||
|| 0 ||
```

```
%Vector de control
|| 1 ||
|| 0 ||
|| 0 ||
|| 0 ||
|| 0 ||
|| 0 ||
|| 0 ||
|| 0 ||
|| 0 ||
|| 0 ||
|| 0 ||
|| 0 ||
|| 0 ||
|| 0 ||
```

```
%Starea urmatoare
|| 0 ||
|| 0 ||
|| 0 ||
|| 0 ||
|| 1 ||
|| 0 ||
|| 0 ||
|| 0 ||
|| 0 ||
|| 0 ||

%Vector de control
|| 0 ||
|| 0 ||
```

```
        || 0 ||
        || 0 ||
        || 0 ||
        || 0 ||
        || 0 ||
        || 0 ||
        || 0 ||
        || 0 ||
        || 0 ||
        || 0 ||
        || 1 ||
        || 0 ||

%Starea urmatoare
        || 1 ||
        || 0 ||
        || 0 ||
        || 0 ||
        || 0 ||
        || 0 ||
        || 0 ||
        || 0 ||
        || 0 ||
        || 1 ||

%Vector de control
        || 0 ||
        || 0 ||
        || 0 ||
        || 0 ||
        || 0 ||
        || 0 ||
        || 0 ||
        || 0 ||
        || 0 ||
        || 0 ||
        || 0 ||
        || 0 ||
        || 0 ||
        || 1 ||

%Starea urmatoare
        || 1 ||
        || 0 ||
        || 0 ||
        || 0 ||
        || 0 ||
        || 0 ||
        || 0 ||
        || 0 ||
        || 1 ||
        || 0 ||
```

```
%Starea urmatoare
        || 0 ||
        || 1 ||
        || 0 ||
        || 0 ||
        || 0 ||
        || 0 ||
        || 0 ||
        || 0 ||
        || 0 ||
        || 0 ||

%Vector de control
        || 0 ||
        || 0 ||
        || 1 ||
        || 0 ||
        || 0 ||
        || 0 ||
        || 0 ||
        || 0 ||
        || 0 ||
        || 0 ||
        || 0 ||
        || 0 ||
        || 0 ||
        || 0 ||

%Starea urmatoare
        || 0 ||
        || 0 ||
        || 0 ||
        || 1 ||
        || 0 ||
        || 0 ||
        || 0 ||
        || 0 ||
        || 0 ||
        || 0 ||
%Vector de control
        || 0 ||
        || 0 ||
        || 0 ||
        || 0 ||
        || 1 ||
        || 0 ||
        || 0 ||
        || 0 ||
        || 0 ||
        || 0 ||
        || 0 ||
        || 0 ||
        || 0 ||
        || 0 ||
```

```
        || 0 ||
        || 0 ||
        || 0 ||
        || 1 ||
        || 0 ||
        || 0 ||
        || 0 ||
        || 0 ||
        || 0 ||
        || 0 ||
        || 0 ||
        || 0 ||

%Starea urmatoare
        || 0 ||
        || 0 ||
        || 0 ||
        || 0 ||
        || 0 ||
        || 1 ||
        || 0 ||
        || 0 ||
        || 0 ||
        || 0 ||

%Vector de control
        || 0 ||
        || 0 ||
        || 0 ||
        || 0 ||
        || 0 ||
        || 0 ||
        || 0 ||
        || 0 ||
        || 0 ||
        || 0 ||
        || 1 ||
        || 0 ||
        || 0 ||
        || 0 ||

%Starea urmatoare
        || 1 ||
        || 0 ||
        || 0 ||
        || 0 ||
        || 0 ||
        || 0 ||
        || 0 ||
        || 0 ||
        || 1 ||
        || 0 ||
```

Comparand marcajele atinse prin intermediul simularii cu PetriTim si marcajele obtinute in urma simularii in Matlab se observa identitatea lor.

5.8. Concluzii

Aplicatia PetriTim, elaborata de catre autor, se prezinta ca un instrument extrem de util în studierea dinamicii retelelor Petri, prin posibilitatea vizualizarii sub o forma "animata" a functiei tranzitiilor de stare, a executarii tranzitiilor. Construirea grafului este facuta usor, prin selectarea obiectelor (pozitii, tranzitii, arce, jetoane). De asemenea, simularea tranzitiilor se face simplu, printr-o serie de comenzi sau optiuni din meniuri. Interfata utilizator este prietenoasa, furnizand mesaje şi dialoguri cu utilizatorul în toate situatiile care pot sa apara.

O trasatura importanta a acestui program consta in posibilitatea generarii elementelor necesare analizei proprietatilor comportamentale (marcaj initial, vectori de stare, matrici de incidenta, ecuatia de stare), pe baza lor putand fi abordate problemele legate de proprietatile structurale si comportamentale ale retelei Petri analizate.

Validarea acestui program s-a realizat prin compararea rezultatelor obtinute in procesul de simulare a modelelor de tip retea Petri, prin utilizarea mediului Matlab si reespectiv PetriTim. Modelele utilizate, sunt cele care se refera la nodurile de tip 1, 2, 3 si 4 iar rezultatele obtinute cu PetriTim au fost identice cu cele obtinute in cazul utilizarii Matlab-ului, ceea ce face din PetriTim o aplicatie viabila in studiul retelelor Petri.

Functionalitatea programului, multitudinea de procese tehnice ce pot fi studiate (calitativ) cu ajutorul lui PetriTim, alaturi de posibilitatile de îmbunatatire, fac din aceasta aplicatie un instrument deosebit de util in analiza si simularea retelelor Petri.

Contributile autorului din cadrul capitolului, in intregime original, se regasesc in cadrul tuturor subiectelor abordate si sunt evidentiate in detaliu in cadrul ultimului capitol al tezei.

Capitolul 6

Concluzii finale si contributii originale. Perspective

Datorita cresterii complexitatii proceselor industriale, si dezvoltarii explozive a echipamentelor de calcul, s-au cautat noi solutii de conducere care sa poata opera astfel incat sa fie eliminate dificultatile legate de stabilirea modelului matematic. O alternativa, care se impune din ce in ce mai mult, sunt tehnicile de modelare a sistemelor de conducere industriale ca SED. Modul de abordare a conducerii SED, complet diferit de metodele clasice, asigura gestionarea fara probleme a tuturor resurselor unui sistem industrial, elimina complet utilizarea in modelare a ecuatilor diferentiale sau a ecuatilor discrete. Elementul timp, esential in cazul metodelor clasice, nu mai este relevant, elementele de baza in cazul SED sunt evenimentele, controlul acestora nefiind influentat de momentul aparitiei (timpul) ci numai de starea logica a acestora.

In cadrul lucrarii, modelarea si implementarea SED s-a realizat utilizand doua tehnici fundamentale: prima bazata pe utilizarea automatelor si a doua bazata pe utilizarea retelelor Petri.

Fiecare are avantajele si dezavantajele sale, ele putand fi sintetizate astfel:

Model	*Avantaje*	*Dezavantaje*
Automat	-Modul simplu de realizare a structurii -Utilizarea exclusiva a evenimentelor externe	-Constructia modelului in cazul sistemelor complexe se bazeaza pe extinderea topologiei -imposibilitatea de a utiliza submodele -Pentru fiecare stare trebuie elaborat un algoritm de conducere individual
Retea Petri	-Modul simplu de constructie -In cazul sistemelor complexe extinderea nu se face pe baza topologiei retelei ci pe baza extinderii marcajelor -Rafinarea „adanca" a operatiilor interne ale modelului -Utilizarea evenimentelor externe fara extinderea topologiei -Posibilitatea utilizarii submodelelor	- In cazul unei retele Petri de mari dimensiuni matricile de incidenta cresc, ceea ce creaza dezavantaje in verificarea proprietatilor retelei pe baza arborelui de acoperire, respectiv pe baza ecuatiei de stare.

Un alt element care diferentiaza alegerea tipului de model este dat de programele utilitare existente pe baza carora se efectueaza simularea si analiza modelelor. Daca in cazul automatelor solutiile sunt relativ reduse, in cazul reteleor

Petri se dispune in general de software-uri specializate. Mediul de simulare utilizat pentru validarea modelelor de tip retele Petri considerate in lucrare este *Petri Nets Toolbox 2.0* (PNT) care ruleaza sub *Matlab*. A fost elaborat de asemenea un program aplicatie original *PetriTim* (capitolul 5) care asigura generarea de fisiere, care contin explicit operatiile realizate pe baza utilizarii ecuatiei de stare, si care in modul de rulare pas cu pas ofera posibilitatea utilizatorului sa aleaga tranazitiile care se executa.

Inexistenta unor tool-uri performante pentru analiza si simularea modelelor de tip automat a fost rezolvata, in cadrul lucrarii, prin transformarea acestor modele in modele de tip retele Petri, care ulterior au fost simulate si validate cu ajutorul *PNT*-ului. Aceasta abordare s-a realizat astfel incat topologia retelei Petri rezultata sa fie tot de tipul masina de stare, ceea ce a asigurat validarea rezultatelor.

Modul in care s-a efectuat modelarea SED are ca suport STZA.

Abordarea modelarii STZA s-a realizat in mai multi pasi:

- au fost stabilite patru structuri de baza (elementare) denumite *noduri*, pentru care au fost elaborate si validate atat modele de tip automat cat si modele de tip retele Petri. Pornind de la aceste modele de baza au fost determinate modele de tip general.
- s-a enuntat modul de definire a STZA, precum si proprietatile teoretice ale acestora;
- s-a elaborat doua metodologii de modelare a STZA pornind de la structurile utilizate in modelarea cozilor de asteptare.

Problematica legata de conducerea SED se bazeaza, pe stabilirea unor modele viabile si impune utilizarea supervizoarelor ca metoda de conducere. In cadrul tezei (capitolul 3), aceasta problematica, a fost abordata utilizand ambele metode de modelare, avandu-se in vedere atat aspecte teoretice cat si aplicative ale sintezei supervizoarelor.

In cazul modelelor de tip automat au fost considerati doi algoritmi de sinteza. Validarea metodelor de sinteza si a supervizoarelor concepute s-a realizat printr-un studiul de caz care abordeaza sinteza unui supervizor pentru un SFFF.

In cazul modelelor de tip retele Petri au fost considerate doua metode de sinteza a supervizoarelor ambele validate prin intermediul unor studii de caz.

Implementarea modelelor validate (capitolul 4), s-a realizat considerand doua solutii hardware distincte.

In cazul modelelor de tip automat, ca suport hardware s-a utilizat un calculator de proces pe care a fost instalat sistemul de operare in timp real QNX, dezvoltarea programului facandu-se in limbajul WATCOM C. Pornind de la modelele validate, au fost elaborate trei taskuri de ordin general: NT1 – task pentru nodul de tip 1, NT2 – task pentru nodul de tip 2 si NT4 – task pentru nodul de tip 4. Pentru interfatarea directa cu procesul s-a elaborat un task special denumit TAC. Modul de transfer al informatiilor s-a efectuat avand la baza capabilitatile SOTR QNX.

In cazul modelelor de tip retele Petri, ca suport hardware s-a utilizat un automat programabil de tip Simatic S7-414, programarea facandu-se in limbajul STEP7. Implementarea s-a realizat tinand cont ca toate modelele sunt construite pe baza nodului de tip 1. A fost elaborata o functie de ordin general pentru implementarea retelei Petri corespunzatoare nodului de tip1. Datorita faptului ca structurile STZA depind de aplicatie este imposibila realizarea unei functii generale de conducere. Cu toate acestea, prin metoda de implementare propusa (utilizarea functiei nodului de tip 1), dezvoltarea unui program de conducere indiferent de structura STZA, nu mai ridica probleme deosebite.

Contributile autorului, impartite in doua categorii: teoretice si aplicative, sunt sintetizate in cele ce urmeaza.

Contributii teoretice

Capitolul 2

- sintetizarea problematicii legate de modelarea SED cu ajutorul limbajelor si automatelor prin abordarea in mod unitar al: limbajelor formale, a automatelor, a modului reprezentare a automatelor pe baza limbajelor, a blocajelor, a automatelor nedeterministe, a operatiilor cu automate, a modului de transformare a unui automat nedeterminist in automat determinist ;
- sintetizarea problematicii legate de modelarea SED cu ajutorul retelelor Petri prin abordarea in mod unitar al: reteleor Petri netemporizate, a modului de analiza a proprietatilor comportamentale si structurale, a retelelor Petri temporizate, a retelelor Petri etichetate;
- modul de utilizare a submodelelor de tip retele Petri in modelarea SED, contributiile originale constand in:
    - definirea subretelei unei retele Petri (definitia 2.17),
    - definirea executiei unei retele Petri (definitia 2.18),
    - teorema 2.9 referitoare la modelele cu intervale de timp,
    - lema 2.1 referitoare la timpul minim de executie a unei retele Petri,
    - lema 2.2 referitoare la timpul maxim de executie a unei retele Petri;
- stabilirea structurilor de baza (elementare) corespunzatoare STZA:
    - nodul de tip 1 – *„o”* intrare *„o”* iesire,
    - nodul de tip 2 – *„doua”* intrari *„o”* iesire,
    - nodul de tip 3 – *„trei”* intrari *„o”* iesire,
    - nodul de tip 4 – *„o”* intrare *„doua”* iesiri;
- stabilirea, simularea si validarea modelelor de tip automat, respectiv retea Petri netemporizata si retea Petri temporizata P pentru nodurile de baza stabilite;
- analiza modelelor de tip automat prin transformare in model de tip retea Petri cu pastrarea topologiei retelei ca masina de stare;
- stabilirea si validarea unor modele de tip general pentru:
    - un nod cu *„n”* intrari si *„o”* iesire,
    - un nod cu *„o”* intrare si *„m”* iesiri,
    - un nod cu *„n”* intrari si *„m”* iesiri;
- elaborarea simbolisticii utilizate in reprezentarea nodurilor;
- definirea teoretica a STZA concretizata prin:
    - definitia 2.21 - sistem de transport cu zone de acumulare,
    - definitia 2.22. - operabilitate a unui STZA,
    - definitia 2.23. - validarea functionarii unui nod,
    - definitia 2.24. - pasul unui STZA,
    - propozitia 2.1. - conditia de operabilitate a unui STZA,
    - propozitia 2.2. - blocajul STZA, propozitia 2.3. - blocarea executiei unui pas;
- elaborarea a doi algoritmi (unul particular si unul general) legati de metodologia de modelare a STZA
    - algoritmul 2.3 – algoritm particular – bazat pe construirea grafului corespunzator STZA modelat, prin conectarea directa a nodurilor;

  - algoritmul 2.4 – algoritm general - bazat pe construirea matricii de incidenta a STZA utilizand pozitii de sincronizare a nodurilor.

Capitolul 3

- sintetizarea intr-o abordare unitara a problematicilor legate de conducerea supervizata a SED;
- abordarea unitara a metodelor de sinteza al supervizoarelor bazate pe utilizarea modelelor de tip automat;
- formularea si sistematizarea metodelor de sinteza a supervizoarelor bazate pe utilizarea modelelor de tip retele Petri;
- elaborarea unui algoritm (algoritmul 3.3) pentru determinarea starilor non-blocante corespunzator unui supervizor implementat pe baza modelelor de tip automat;
- elaborarea unui algoritm (algoritmul 3.4) pentru determinarea starilor controlabile corespunzatoare unui supervizor implementat pe baza modelelor de tip automat;
- elaborarea unui algoritm (algoritmul 3.5) pentru sinteza supervizoarelor bazate pe modele de tip retele Petri prin compunere paralela.

Capitolul 4

- formularea principiilor generale de proiectare a programelor de conducere in timp real;
- elaborarea unei metodologii de implementare a modelelor de tip automat avand ca suport hardware un calculator de proces pe care este instalat un sistem de operare in timp real;
- conceperea unei metodologii de transformare a modelelor de tip retele Petri netemporizate in modele de tip SPIN;
- elaborarea unei metodologii de implementare a modelelor de tip SPIN avand ca suport hardware PLC-uri.

Contributii aplicative

Capitolul 2

- modelarea unui nod cu *„doua"* intrari si *„doua"* iesiri;
- elaborarea de modele pentru diferite noduri complexe: nod cu *„5"* intrari *„o"* iesire, nod cu *„o"* intrare si *„3"* iesiri, nod cu *„5"* intrari si *„5"* iesiri (utilizand nodurile de baza stabilite);
- modificarea modelului clasic de tip retea Petri a unei cozi de asteptare, corespunzatoare nodurilor de baza definite;
- modelarea STZA cu ajutorul retelelor Petri.

Capitolul 3

- sinteza unui supervizor pentru un SFFF avand la baza modele de tip automat;
- sinteza unui supervizor pentru un STZA avand la baza modele de tip retele Petri.

Capitolul 4

- elaborarea, testarea si validarea unei metode de implementare a unui STZA avand ca suport un calculator de proces pe care se afla instalat SOTR QNX si care se bazeaza pe modele de tip automat;

- elaborarea unor metode de comunicatie intertaskuri pe baza mesajelor interprocese, respectiv a memoriei partajate;
- elaborarea unor taskuri in care sunt implementate nodurile de baza stabilite anterior (in limbajul WATCOM C);
- elaborarea si implementarea unor ordinograme corespunzatoare starilor automatelor considerate: Wait, Open, Close, Error, Scann (in WATCOM C);
- elaborarea, testarea si validarea unei metode de implementare a unui STZA avand ca suport hardware un PLC de tip Simatic S7-414 si care se bazeaza pe modele de tip retele Petri;
- elaborarea (in limbajul STEP7) a unei functii prin care s-a realizat implementarea modelului de tip retea Petri corespunzator nodului de tip 1;
- elaborarea, testarea si validarea unei functii care implementeaza un STZA, functie particulara, care tine cont de structura STZA considerat si care are la baza modelele de tip retele Petri derivate din modelul clasic al unei cozi de asteptare.

Capitolul 5

- realizarea unui produs program ***PetriTim***, destinat analizei si simularii modelelor de tip retele Petri;
- implementarea unui mecanism de validare a executiei unei tranzitii in modul de lucru pas cu pas;
- implementarea unui mecanism de generare a unui fisier, care contine toate informatiile legate de executia unei tranzitii, informatii cuprinse in ecuatiile de stare.

Perspective

Dinamica dezvoltarii industriale, a aparitiei de noi echipamente hardware de conducere a proceselor industriale, implica in mod direct cautarea de noi si noi metode de modelare si conducere. Modelarea prin intermediul SED constituie o directie care trebuie abordata si dezvoltata.

Dintre directiile posibile pe care poate evolua o cercetare viitoare se pot enumera:

- dezvoltarea unor metodologii de modelare utilizand sisteme cu evenimente distribuite;
- dezvoltarea unor metodologii (pachete software) pentru sinteza supervizoarelor;
- conducerea ierarhizata a proceselor industriale „vazute” ca SED.

Desigur aceste directii sunt numai cateva idei, lista putand fi completata cu generozitate.

Bibliografie

[Aalst94] W.M.P. van der Aalst. "Putting Petri nets to work in industry", Computers in Industry, 25(1):45-54, 1994.
http://is.tm.tue.nl/staff/wvdaalst/publications/publications.htm

[Aalst96a] W.M.P. van der Aalst. "Petri net based scheduling." OR Spectrum, 18:219-229, 1996 http://is.tm.tue.nl/staff/wvdaalst/publications/publications.htm

[Aalst96b] W.M.P. van der Aalst. "Three Good Reasons for Using a Petri-net-based Workflow Management System."In S. Navathe and T. Wakayama, editors, Proceedings of the International Working Conference on Information and Process Integration in Enterprises (IPIC'96), pages 179-201, Camebridge, Massachusetts, Nov 1996.
http://is.tm.tue.nl/staff/wvdaalst/publications/publications.htm

[Aalst97] W.M.P. van der Aalst, „Parallel Computation of Reachable Dead States in a Free-choice Petri Net", 1997, Department of Mathematics and Computing Science, Eindhoven University of Technology,
http://is.tm.tue.nl/staff/wvdaalst/publications/publications.htm

[Aalst98] W.M.P. van der Aalst. "The Application of Petri Nets to Workflow Management." The Journal of Circuits, Systems and Computers, 8(1):21-66, 1998.http://is.tm.tue.nl/staff/wvdaalst/publications/publications.htm

[AB03] Wil M. P. van der Aalst, Eike Best, „Applications and Theory of Petri Nets 2003", 24th International Conference, ICATPN 2003, Eindhoven, The Netherlands, June 23-27, 2003, Proceedings. Lecture Notes in Computer Science 2679 Springer 2003, ISBN 3-540-40334-5

[AI06] Aybar, A. Iftar, A., „Supervisory Controller Design for Timed Petri Nets", IEEE/SMC International Conference on System of Systems Engineering 2006, April 24-26, 2006 pp: 59- 64

[AO95] W.M.P. van der Aalst, M.A. Odijk. "Analysis of Railway Stations by means of Interval Timed Coloured Petri Nets." Real-Time Systems, 9(3):241-263, 1995. http://is.tm.tue.nl/staff/wvdaalst/publications/publications.htm

[BC04] Simona Bernardi, Javier Campos, „On Performance Bounds for Interval Time Petri Nets", Proceedings of The Quantitative Evaluation of Systems, First International Conference on (QEST'04)pp. 50-59

[BD03] Simona Bernardi, Susanna Donatelli, "Building Petri net scenarios for dependable automation systems", 10th IEEE International Workshop on Petri Nets and Performance Models (PNPM 2003), 02–05 September 2003 - Urbana-Champaign, IL, USA

[Berg00] Hans Berger, „Automating with STEP7 in STL and SCL", MCD Verlang, 2000.

[BGM00] Fabio Balduzzi, Alessandro Giua, Giuseppe Menga, „First-Order Hybrid Petri Nets: A Model for Optimization and Control", IEEE Transactions on Robotics and Automation, Vol. 16, No. 4, August 2000

[BHR06] P. Bouyer, S. Haddan, P.A. Reynier, „Extended Time Automata and Time Petri Nets", Research Report LSV-06-01, Ecole Normale Superieure de Cachan, 2006.

[BSV03] G. Bucci, L. Sassoli, E. Vicario, "A discrete time model for performance evaluation and correctness verification of real time systems", 10th IEEE International Workshop on Petri Nets and Performance Models (PNPM 2003), 02-05 September 2003 - Urbana-Champaign, IL, USA

[Cass02] Christos G. Cassandras, "From Discrete Event to Hybrid Systems", 6th International Workshop on Discrete Event Systems (WODES 2002), 02-04 October 2002 — Zaragoza, SPAIN

[Ciuf02] Calin Ciufudean, „Sisteme cu Evenimente Discrete pentru Modelarea Traficului Feroviar", Editura MATRIXROM, Bucuresti 2002.

[CL01] Christos G. Cassandras, Stephane Lafortune, "Introduction to Discrete Event Systems", Kluwer Academic Publishers, 2001, Boston

[CKLY95] J. Cortadella, M. Kishinevsky, L. Lavagno, and A. Yakovlev. „Synthesizing petri nets from state-based models". In Proc. International Conf. Computer-Aided Design (ICCAD), 1995.
www.lsi.upc.edu/~jordicf/gavina/BIB/CONFERENCE.html

[CKLY98] Jordi Cortadella, Michael Kishinevsky, Luciano Lavagno, Alexandre Yakovlev, "Deriving Petri Nets from Finite Transition Systems", IEEE Transactions on Computers, Vol. 47, No. 8, August 1998

[CP04] Christine Choppy, Laure Petrucci, „Towards a Methodology for Modeling with Petri Nets", Proceedings Workshop on Practical Use of Coloured Petri Nets (CPN'2004), Aarhus, Denmark, pages 39 - 56, October 2004.
http://www-lipn.univ-paris13.fr/~petrucci/PAPERS

[CR04a] Franck Cassez, Olivier Roux, „From Petri Nets to Timed Automata", Published by Elsevier Science B.V., 2004

[CR04b] Jordi Cortadella, Wolfgang Reisig, „Applications and Theory of Petri Nets 2004", 25th International Conference, ICATPN 2004, Bologna, Italy, June 21-25, 2004, Proceedings. 3099 Springer 2004, ISBN 3-540-22236-7

[CRL06] Campos-Rodriguez, R. Ramirez-Trevino, A. Lopez-Mellado, E., „Observability Analysis of Free-Choice Petri Net Models", IEEE/SMC International Conference on System of Systems Engineering 2006, April 24-26, 2006, pp: 77- 82

[Daro05a] Philippe Darondeau, „Distributed implementations of Ramadge-Wonham supervisory control with Petri nets", Proceedings of the 44th IEEE Conference on Decision and Control, and the European Control Conference 2005 Seville, Spain, December 12-15, 2005

[Daro05b] Philippe Darondeau, „Applications and Theory of Petri Nets 2005", 26th International Conference, ICATPN 2005, Miami, USA, June 20-25, 2005, Proceedings. 3536 Springer 2005, ISBN 3-540-26301-2

[DHPSV93] F. DiCesare, G. Harhalakis, J.M. Proth, M. Silva, F.B. Vernadat, „Practice of Petri Nets in Manufacturing", Chapman and Hall, 1993.

[DT06] Susanna Donatelli, P. S. Thiagarajan, „Petri Nets and Other Models of Concurrency", ICATPN 2006, 27th International Conference on Applications and Theory of Petri Nets and Other Models of Concurrency, Turku, Finland, June 26-30, 2006, Proceedings. Lecture Notes in Computer Science 4024 Springer 2006, ISBN 3-540-34699-6

[EL02] Javier Esparza, Charles Lakos, „Applications and Theory of Petri Nets 2002", 23rd International Conference, ICATPN 2002, Adelaide, Australia, June 24-30, 2002, Proceedings. Lecture Notes in Computer Science 2360 Springer 2002, ISBN 3-540-43787-8

[ERRW03] H. Ehrig, W. Reisig, G. Rozenberg, H. Weber, „Petri Net Technology for Communication-Based Systems", Lecture Notes in Computer Science, vol. 2472, Springer-Verlag, 2003, ISBN: 3-540-20538-1.

[FG98] A. Fanni, A. Giua, „ Discrete Event Representation of Qualitative Models Using Petri Nets", IEEE Transaction on systems, man and cibernetics - Part B: Cybernetics, Vol. 28, No. 8, December 1998.

[FGMS02] Angela Di Febbraro, Davide Giglio, Riccardo Minciardi, Simona Sacone, „Optimization of Manufacturing Systems Modelled by Timed Petri Nets", 6th International Workshop on Discrete Event Systems (WODES 2002), 02–04 October 2002 — Zaragoza, SPAIN

[FL00a] Georg Frey, Lothar Litz, „ Transparency analylis of Petri Net based logic controllers. A measure for software quality in automation", Proceedings of the American Control Conference ACC 2000, Chicago, 28-30 Juni 2000

[FL00b] Georg Frey, Lothar Litz,"Formal methods in PLC programming", Proceedings of IEEE Conference on System Man and Cybernetics SMC 2000, Nashville, Oct. 8-11, 2000.

[Frey02] G. Frey,"Software Quality in Logic Controller Programming", Proceedings of the IEEE SMC 2002, Hammamet (Tunisia), Paper ID MP1A4, Oct. 2002

[GB05] Gomes, L. Joao Paulo Barros, „Structuring and composability issues in Petri nets modeling", IEEE Transactions on Industrial Informatics, May 2005 Volume: 1, Issue: 2, pp: 112- 123

[GBFP01] Gaeta, R. Bobbio, A. Franceschinis, G. Portinale, L., „Dependability assessment of an industrial Programmable LogicController via Parametric Fault-Tree and High Level Petri net", Proceedings of 9th International Workshop on Petri Nets and Performance Models, 2001. 09/11/2001 - 09/14/2001, Aachen, Germany

[GCS05] A. Giua, D. Corona, C. Seatzu, „State Estimation of lambda-free Labeled Petri Nets with Contact-Free Nondeterministic Transitions," Discrete Event Dynamic Systems, Vol. 15, No. 1, pp. 85–108, March 2005.

[GG01] Gerald C. Gannod, Sunil Gupta, „An Automated Tool for Analyzing Petri Nets using SPIN", Proceedings of the 16th International Conference on Automated Software Engineering, Nov. 2001, pp.404-407, IEEE

[GG06] Garcia, R.G. Gelle, E., „Applying and adapting the IEC 61346 standard to industrial automation applications", IEEE Transactions on Industrial Informatics, Aug. 2006 Volume: 2, Issue: 3, pp: 185- 191

[Giua92] Alessandro Giua, „Petri Nets as Discrete Event Models for Supervisory Control", A Thesis Submitted to the Graduate Faculty of Rensselaer Polytechnic Institute in Partial Fulfillment of the Requirement for the Degree of Doctor of Philosophy. Major Subject: Computer and Systems Engineering. Rensselaer Polytechnic Institute (Troy, New York) July 1992

[Glei03] Thomas Gleixner, „Real-Time Systems for Industrial Use: Requirements for the Future", Proceedings of the International Parallel and Distributed Processing Symposium, 2003, IEEE.

[GM04] B. Gaudin, H. Marchand. „Modular Supervisory Control of a Class of Concurrent Discrete Event Systems". Workshop on Discrete Event Systems, 2004.

[GM05a] B. Gaudin, H. Marchand. „Efficient Computation of Supervisors for loosely synchronous Discrete Event Systems: A State-Based Approach." IFAC World Congress, 2005.

[GM05b] B. Gaudin, H. Marchand. „Safety Control of Hierarchical Synchronous Discrete Event Systems: A State-Based Approach." Meditteranean Conference on Control and Automation, 2005.

[GMMP03] Maria del Mar Gallardo, Jesus Martinez, Pedro Merino, Ernesto Pimentel, „Abstract Model Checking and Refinement of Temporal Logic in SPIN", 3rd International Conference on Application of Concurrency to System Design (ACSD 2003), 18–20 June 2003 — Guimaraes, PORTUGAL

[GS05] Alessandro Giua, Carla Seatzu, „Identification of free-labeled Petri nets via integer programming", Proceedings of the 44th IEEE Conference on Decision and Control, and the European Control Conference 2005 Seville, Spain, December 12-15, 2005

[HF01] T. Hummel, W. Fengler,"Design of Embedded Control Systems Using Hybrid Petri Nets", The International Workshop on Discrete-Event System Design, DESDes'01, June 27÷29, 2001; Przytok, Poland

[HKG97] L.E. Holloway, B.H. Krough, A. Giua, „A survey of Petri net methods for controlled discrete event systems", Discrete Event Dynamic Systems: Theory and Applications, 7, pag. 151-190, 1997, Kluwer Academic Publishers, Boston

[HL97] Jeffrey W. Herrmann, Edward Lin, "Petri Nets: Tutorial and Applications", The 32th Annual Symposium of the Washington Operations Research -Management Science Council, Washington, D. C., November 5, 1997

[HL04] Pao-Ann Hsiung, Shang-Wei Lin, „Automatic Synthesis and Verification of Real-Time Embedded Software", EUC 2004, LNCS 3207, Springer-Verlang Berlin, 2004.

[HLF02] Anders Hellgren, Bengt Lennartson, Martin Fabian, „Modelling and PLC-Based Implementation of Modular Supervisory Control", 6th International Workshop on Discrete Event Systems (WODES 2002), 02–04 October 2002 — Zaragoza, SPAIN

[HP97] W. Halang, C. Pereira, s.a. „Real-Time Computing Education: Responding to a Challenge of Next Century", IFAC Real-Time Programming WRTP'97, Pergamon Press 1997.

[Hsieh06] Fu-Shiung Hsieh, „Robustness analysis of Petri nets for assembly/disassembly processes with unreliable resources", Automatica, July, 2006, Volume 42, Issue 7

[IA02a] Marian V. Iordache and Panos J. Antsaklis, „Synthesis of Supervisors Enforcing Firing Vector Constraints in Petri Nets", Technical Report of the ISIS Group at the University of Notre Dame ISIS-2002-002 February, 2002

[IA02b] Marian V. Iordache, Panos J. Antsaklis, „Decentralized Control of Petri Nets", Technical Report of the ISIS Group at the University of Notre Dame, ISIS-2002-005, October, 2002

[IA02c] Marian V. Iordache and Panos J. Antsaklis, „Software Tools for the Supervisory Control of Petri Nets Based on Place Invariants", Technical Report of the ISIS Group at the University of Notre Dame ISIS-2002-003 April, 2002

[IA02d] Marian V. Iordache and Panos J. Antsaklis, „Decentralized Control of Petri Nets", Technical Report of the ISIS Group at the University of Notre Dame ISIS-2002-005 October, 2002

[IMA99] Marian V. Iordache John O. Moody Panos J. Antsaklis, „A Method for Deadlock Prevention in Discrete Event Systems Using Petri Nets", Technical Report of the ISIS Group at the University of Notre Dame ISIS-99-006 July, 1999

[IMA01] Marian V. Iordache John O. Moody Panos J. Antsaklis, „Automated Synthesis of Deadlock Prevention Supervisors Using Petri Nets", Technical Report of the ISIS Group at the University of Notre Dame ISIS-2000-003 May, 2000,Revised in October 2000 and November 2001

[IMA02] Marian V. Iordache John O. Moody Panos J. Antsaklis, „Automated Synthesis of Liveness Enforcing Supervisors Using Petri Nets", Technical Report of the ISIS Group at the University of Notre Dame ISIS-00-004 October, 2000 Revised in January 2001 and May 2002

[Iord03] Marian Valentin Iordache, „Methods for the supervisory control of concurrent systems based on Petri net abstractions", A Dissertation Submitted to the Graduate School of the University of Notre Dame in Partial Ful_llment of the Requirements for the Degree of Doctor of Philosophy Notre Dame, Indiana December 2003

[Jens96] Kurt Jensen,"An Introduction to the Practical Use of Coloured Petri Nets", Lecture Notes in Computer Scinces, Apringer-Verlang 1996

[JJRS04] Jorge Julvez, Emilio Jimenez, Laura Recalde, Manuel Silva, „On Observability in Timed Continuous Petri Net Systems", Proceedings of The Quantitative Evaluation of Systems, First International Conference on (QEST'04), pp. 60-69

[JK96], R. Johnson , D. Kearney, "A tutorial on timed descrete event modelling and its applications to automation systems.", School of Electrical Engineering, University of South Australia, 1996, www.cis.unisa.edu.au/~cisdak/ResearchPapers/ICCARVTimedDESNovember1996.ps

[JTMZ01] Eric Y.T. Juan, Jeffrey J.P. Tsai, Tadao Murata, Fellow, Yi Zhou, „Reduction Methods for Real-Time Systems Using Delay Time Petri Nets", IEEE Transactions on Software Engineering, Vol. 27, No. 5, May 2001

[KA98] Xenofon D. Koutsoukos, Panos J. Antsaklis, „Hybrid Control Systems Using Timed Petri Nets: Supervisory Control Design Based on Invariant Properties", National Science Foundation grant ECS95-31485.Department of Electrical Engineering University of Notre Dame Notre Dame, IN 46556 1998.

[KCJ98] Lars M. Kristensen, Soren Christensen, Kurt Jensen, „The practitioner's guide to coloured Petri Nets", Springer Verlang 1998

[Kemp04] Peter Kemper, „Petri Nets", Lecture for CL. Dipl. Informatik, http://www.iai.inf.tu-dresden.de/ms/ms.main.html

[Khou04] Ahmed Khoumsi, „Supervisory Control for the Conformance of Real-Time Discrete Event Systems", http://www.gel.usherb.ca/khoumsi/Research/Public/WODES04.ps

[Kind04] Ekkart Kindler, „Definition, Implementation and Application of a Standard Interchange Format for Petri Nets", Proceedings of the Workshop on the Definition, Implementation and Application of a Standard Interchange Format for Petri Nets Satellite event of the 25th International Conference on Application and Theory of Petri Nets 2004, Bologna, Italy, June 26, 2004

[KN06] A. Khatab, M. Nourelfath, „Analytical approach to evaluate language measure parameters for discrete-event supervisory control", International Journal of Control, 2006, Volume 79, Issue 7

[Kout01] Maciej Koutny, „Application and Theory of Petri Nets 2001", 22nd International Conference, ICATPN 2001, Newcastle upon Tyne, UK, June 25-29, 2001, Proceedings. Lecture Notes in Computer Science 2075 Springer 2001, ISBN 3-540-42252-8

[KSH05] Christoph M. Kirsch, Marco A.A. Sanvido, Thomas A. Henzinger, „A Programmable Microkernel for RealTime Systems", VEE'05, June 11-12, 2005, Chicago, Illinois, USA

[KT05] D. I. Kharitonov, G. V. Tarasov, „Towards Petri nets application in parallel program debugging", The 6th Asian Computaitional Fluid Dynamics Conference Taiwan, October 24~October 27, 2005

[KW00] Ekkart Kindler, Michael Weber, „A Universal Module Concept for Petri Nets an implementation-oriented approach", In: Informatik-Berichte 150, Humboldt-Universität zu Berlin
http://www.informatik.hu-berlin.de/top/pnml/download/modPNML_TB.ps

[KWFLL02] S. Klein, X. Weng, G. Frey, J.J. Lesage, L. Litz,"Controller design for an FMS using signal interpreted Petri nets and SFC",Proceedings of the American Control Conference 2002 (ACC2002), Anchorage, Alaska, pp.4141-4146, May 2002.

[LBW00] R.J. Leduc, B.A. Brandin, and W. Murray Wonham. „Hierarchical interface-based nonblocking veritcation". Proceedings of the Canadian Conference on Electrical and Computer Engineering, May 2000.

[LBWL01]R.J. Leduc, B.A. Brandin, W. Murray Wonham, and M. Lawford. „Hierarchical interface-based supervisory control: Serial case".Proceedings of 40th Conf. Decision Contr.,Orlando, USA, December 2001.

[Led96] Ryan Leduc. „PLC implementation of a DES supervisor for a manufacturing testbed: An implementation perspective". Master's thesis, Department of Electrical and Computer Engineering, University of Toronto, Toronto, Ont, 1996.

[Led02] R.J. Leduc. „Hierarchical Interface Based Supervisory Control." PhD thesis, Department of Electrical and Computer Engineering, University of Toronto, 2002.

[Letia98] Tiberiu S. Letia, Adina M. Astilean , „Sisteme cu Evenimente Discrete: modelare, analiza, sinteza si control", Editura microInformatica, 1998, Cluj

[Letia00] T. Letia, „Sisteme de timp-real", Editura Albastra,Cluj-Napoca, ISBN: 973-9443-49-4,2000.

[LJ06] Robert Lorenz, Gabriel Juhas. Towards Synthesis of Petri Nets from Scenarios, 26th International Conference On Application and Theory of Petri Nets and Other Models of Concurrency (Petri Nets), Turku, FINLAND, June 2006

[LKWDS06] Cong Liu, Alex Kondratyev, Yosinori Watanabe, Jorg Desel, Alberto Sangiovanni-Vincentelli. „Schedulability Analysis of Petri Nets Based on Structural Properties", 26th International Conference On Application and Theory of Petri Nets and Other Models of Concurrency (Petri Nets), Turku, FINLAND, June 2006

[LLD06] Leduc, R.J.; Lawford, M.; Pengcheng Dai, „Hierarchical interface-based supervisory control of a flexible manufacturing system", IEEE Transactions on Control Systems Technology, July 2006, Volume: 14, Issue: 4, pp: 654- 668

[LLW01]R. Leduc, M. Lawford, and W. Murray Wonham. „Hierarchical interface-based supervisory control: AIP example". Proceedings of 39th Annual Allerton Conference on Comm.,Contr., and Comp., Oct 2001.

[LM04] James J. Leifer, Robin Milner, „Transition systems, link graphs and Petri nets", Technical Report Number 598 August 2004 Computer Laboratory UCAM-CL-TR-598 ISSN 1476-2986 Cambridge United Kingdom

[LWL01] R.J. Leduc, W. Murray Wonham, and M. Lawford. „Hierarchical interface-based supervisory control: Parallel case". Proceedings of 39th Annual Allerton Conference on Comm., Contr., and Comp., Oct 2001.

[MA97] John O. Moody, Panos J. Antsaklis, „Petri Net Supervisors for DES in the Presence of Uncontrollable and Unobservable Transitions", Technical Report of the ISI Group at the University of Notre Dame, ISIS-97-012, October, 1997,

[MF02] Mark Minas, Georg Frey,"Visual PLC-Programming using Signal Interpreted Petri Nets", Proceedings of American Control Conference 2002 (ACC2002), Anchorage, Alaska, pp.5019-5024, May 2002.

[Micz01] Piotr Miczulski, „State Space Calculation Algorithm Of Hierarchical Petri Nets With Application Of Decision Diagrams", The International Workshop on Discrete-Event System Design, DESDes'01, June 27÷29, 2001; Przytok near Zielona Góra, Poland

[MMP05], Mihaela Matcovschi, Cristian Mahulea, Octavian Patravanu, „Learning about Petri Nets Toolbox for use with Matlab", Technical University „Gh. Asachi" of Iasi,2005

[Mood99] John O. Moody, „Petri Net Supervisors for DES with Uncontrollable and Unobservable Transitions", Technical Report of the ISIS Group at the University of Notre Dame ISIS-99-004, http://www.nd.edu/ isis/tech.html, February, 1999

[Mur89] Tadao Murata, "Petri Nets: Properties, Analysis and Applications", Proceedings of the IEEE Vol 77, No. 4, April 1989

[Pastr97] Octavian Pastravanu, „Sisteme cu evenimente discrete. Tehnici calitative bazate pe formalismul retelelor Petri", Editura MatrixRom, 1997, Bucuresti

[PC95] Marco A. Pena, Jordi Cortadella, „Combining Process Algebras and Petri Nets for the Specification and Synthesis of Asynchronous Circuits", UPC/DAC Report No. RR-95/48, Dec. 1995.

[Petri62] C.A. Petri, „Kommunikation mit Automaten", Institut fur Instrumentelle Mathematik Bonn, Schriften des IIM Nr. 2

[PMM02] Octavian Pastravanu,Mihaela Matcovschi, Cristian Mahulea, „Aplicatii ale Retelelor Petri in Studierea Sistemelor Cu Evenimente Discrete",Editura Gh. ASACHI Iasi,2002

[PN2000], ***, „Petri Nets 2000. Introductory Tutorial Petri Nets", 21st International Conference on Application and Theory of Petri Nets Aarhus, Denmark, June 26-30, 2000

[Pomm03] F. Pommereau, „Petri nets as Executable Specifications of High-Level Timed Parallel Systems", Technical Report TR-2003-09 Laboratory of Algorithms, Complexity and Logic University of Paris

[PWX97] Jean-Marie Proth, Liming Wang, Xiaolan Xie, „Class of Petri Nets for Manufacturing System Integration", IEEE Transactions on Robotics and Automation, Vol. 13, No. 3, June 1997
[QC02] Max H. de Queiroz, Jose E. R. Cury, „Synthesis and Implementation of Local Modular Supervisory Control for a Manufacturing Cell", 6th International Workshop on Discrete Event Systems (WODES 2002), 02–04 October 2002 — Zaragoza, SPAIN

[QNX00] ***, „QNX. System Architecture", QNX Software Systems Ltd.,Ontario,2000

[RF02] Jean-Marc Roussel, Jean-Marc Faure, „An Algebraic Approach for PLC Programs Verification" , 6th International Workshop on Discrete Event Systems (WODES 2002), 02–04 October 2002 — Zaragoza, SPAIN

[Rob97] Nicolae Robu,"Tehnici de modelare si metode de ordonantare in fabricatia integrata prin calculator", Editura Orizonturi Universitare, Timisoara 1997

[RW87b] P.J. Ramadge,W.M. Wonham. „Supervisory Control of a Class of Discrete Event Processes." SIAM Journal of Control and Optimization, 25:206–230, 1987.

[RW89] P.J. Ramadge, W.M. Wonham. „The Control of Discrete Event Systems." Proceedings IEEE, Special Issue Discrete Event Dynamic Systems, 77:81–98, 1989.

[Schm05] Klaus Schmidt, „Hierachical Control of Descentralized Discrete Event Systems. Theory and Application", PhD Thesis, Der Technishen Fakultat der Universitat Erlangen-Nurnberg, 2005.

[Stand04] ***, Software and system engineering - High-level Petri nets. Part 1: Concepts, definitions and graphical notation, International Standard ISO/IEC 15909-1 First edition 2004-12-01

[Stand05] ***, Software and Systems Engineering – High-level Petri Nets Part 2: Transfer Format, International Standard ISO/IEC 15909-2 WD Version 0.9.0, June 23, 2005

[TTV05] G. J Tsinarakis, N. C. Tsourveloudis, K. P. Valavanis, „Studying Multi-assembly Machine Production Systems with Hybrid Timed Petri Nets", Proceedings of the 2005 IEEE International Conference on Automation Science and Engineering Edmonton, Canada, August 1 & 2, 2005

[UJ96] M. Uzam, A.H. Jones, N. Ajlouni, „ Conversion of Petri Net Controllers for Manufacturing Systems into Ladder Logic Diagrams", Proceedings of the 1996 IEEE

Conference on Emerging Technologies and Factory Automation - ETFA`96, Kauai Mariot, Kauai,Hawaii,USA, November 18-21, vol. 2 pp. 649-655.

[UJ97] M. Uzam, A.H. Jones, „ Real-Time implementation of Petri net controllers using programmable logic controllers", 4th IFAC Workshop on Algorithms and Architectures for Real-Time Control - AARTC'97, Vilmoura, Portugal, April 9-11, 1997, pp.421-426.

[UM06] Dan Ungureanu-Anghel, Dan Lucian Mihaescu, „Supervisory control of a Flexible Manufacturing System used in Spinning Mills", Proceedings of 3rd Romanian-Hungarian Joint Symposium on Applied Computational Intelligence, SACII 2006, Timisoara, Romania, May 25-26, 2006, pp. 561-570.

[UP06a] Dan Ungureanu-Anghel, Octavian Prostean, „Modeling the Basic Components of Suspended Transport Systems Using Sequential Automata", Proceedings of the 7th International Conference on Technical Informatics - CONTI`06, Vol. 1 Automation and Applied Informatics, Timisoara, 8-9 June, 2006, pp.77 - 80.

[UP06b] Dan Ungureanu-Anghel, Octavian Prostean, „Modeling the Basic Components of Suspended Transport Systems with the Help of Untimed Petri Nets", Proceedings of the 7th International Conference on Technical Informatics - CONTI`06, Vol. 1 Automation and Applied Informatics, Timisoara, 8-9 June, 2006, pp.71 - 76.

[Ung01] Dan Ungureanu-Anghel, " Technical reports. Project No. 30186, Textilot-France. Automatic sorting system" Schonenberger Systeme GMBH, Landsberg am Lech, Germany, Dec. 2001

[Ung02] Dan Ungureanu-Anghel, " Technical reports. Project No. 97190, Pearl-Germany. Automatic transport system for a full automatic warehouse" Schonenberger Systeme GMBH, Landsberg am Lech, Germany, Dec. 2002

[Ung03] Dan Ungureanu-Anghel, " Technical reports. Project No. 94790, Wacker-Germany. Automatic transport system for a full automatic warehouse" Schonenberger Systeme GMBH, Landsberg am Lech, Germany, Sept. 2003

[Ung04a] Dan Ungureanu-Anghel, " Technical reports. Project No. 99280, Albarakeh-Syria. Automatic transport system for spinning mills" Schonenberger Systeme GMBH, Landsberg am Lech, Germany, Nov. 2004

[Ung04b] Dan Ungureanu-Anghel, " Technical reports. Project No. 99780, Oulabitex-Syria. Automatic transport system for spinning mills" Schonenberger Systeme GMBH, Landsberg am Lech, Germany, Nov. 2004

[Ung05] Dan Ungureanu-Anghel, " Technical reports. Project No. 94435, Blanco-Germany. Automatic transport system for a full automatic warehouse" Schonenberger Systeme GMBH, Landsberg am Lech, Germany, Dec. 2004

[Ung06a] Dan Ungureanu-Anghel, " Technical reports. Project No. 30531, Sogetiss-Morocco. Automatic transport system for spinning mills" Schonenberger Systeme GMBH, Landsberg am Lech, Germany, Jul. 2005

[Ung06b] Dan Ungureanu-Anghel, "Modelling the basic components of automatic transport systems with accumulation areas using sequential automata converted in Petri nets", SCIENTIFIC BULLETIN of "Politehnica" University of Timisoara, ROMANIA, Transactions on AUTOMATIC CONTROL and COMPUTER SCIENCE, Vol. 51 (65), No. 3, 2006

[Ung06c] Dan Ungureanu-Anghel, „Two general untimed Petri net models for the basic components of automatic transport systems with accumulation areas", SCIENTIFIC BULLETIN of "Politehnica" University of Timisoara, ROMANIA, Transactions on AUTOMATIC CONTROL and COMPUTER SCIENCE, Vol. 51 (65), No. 4, 2006

[Ung07a] Dan Ungureanu-Anghel, "A general untimed Petri net model for a node with „n" inputs and „m" outputs component of automatic transport systems with accumulation areas", SCIENTIFIC BULLETIN of "Politehnica" University of Timisoara, ROMANIA, Transactions on AUTOMATIC CONTROL and COMPUTER SCIENCE, Vol. 52 (65), No. 1, 2007

[UP07b] Dan Ungureanu-Anghel, Octavian Prostean, "An Untimed Petri Net Model for an Automatic Transport System with Accumulation Areas", Proceedings of 4th International Symposium on Applied Computational Intelligence and Informatics, SACI 2007, Timisoara, Romania, May 17-18, 2007, pp. 279-284.

[Vahi04], Arash Vahidi, „Efficient Analysis of DES. Supervisor Synthesis with Binary Decision Diagrams", PhD Thesis 2004, Department of Signals and Systems Chalmers University of Technology, Goteborg, Sweden

[WATC00] ***, „WATCOM C. User Manual", QNX Software Systems Ltd.,Ontario,2000

[Wonh02] W. Murray Wonham. „Notes on Control of Discrete-Event Systems". Department of Electrical and Computer Engineering, University of Toronto, 2002. http://odin.control.toronto.edu/DES/.

[WR87] W.M. Wonham, P.J. Ramadge. „On the Supremal Controllable Sublanguage of a Given Language." SIAM Journal of Control and Optimization, 25:637-659, 1987.

[WR88] W.M. Wonham, P.J. Ramadge. „Modular Supervisory Control of Discrete Event Systems." Mathematics of Control, Signals and Systems, 1(1):13-30, 1988.

[XTPS02] Hao Xia, Anastasios Trontis, Yan Pang, Michael P. Spathopoulos, „Supervisory Eventuality Synthesis", 6th International Workshop on Discrete Event Systems (WODES 2002), 02-04 October 2002 — Zaragoza, SPAIN

[YGG03] Haobo Yu, A. Gerstlauer, D. Gajski, „RTOS Scheduling in Transaction Level Models", CODES_ISSS `03, October 1-3, 2003, Newport Beach California, USA.