

DEPARTAMENTUL DE AUTOMATICĂ

---

# Sisteme cu Evenimente Discrete

## Îndrumător de laborator

---

Dr.Ing. Camelia Claudia AVRAM

# CUPRINS

<b>Listă de figuri</b>	<b>v</b>
<b>Listă de tabele</b>	<b>ix</b>
<b>1 Descrierea logică a funcționării Sistemelor cu Evenimente Discrete</b>	<b>1</b>
1.1 Obiective . . . . .	1
1.2 Considerații teoretice . . . . .	1
1.3 Desfășurarea lucrării . . . . .	3
1.3.1 Probleme propuse . . . . .	3
<b>Bibliografie</b>	<b>7</b>
<b>2 Abordarea SED prin metoda Grafctet</b>	<b>8</b>
2.1 Obiective . . . . .	8
2.2 Considerații teoretice . . . . .	8
2.3 Desfășurarea lucrării . . . . .	9
2.3.1 Probleme propuse . . . . .	9
2.3.2 Probleme propuse . . . . .	12
<b>Bibliografie</b>	<b>17</b>
<b>3 Dezvoltarea aplicațiilor folosind ladder logic</b>	<b>18</b>
3.1 Obiective . . . . .	18
3.2 Considerații teoretice . . . . .	18
3.2.1 Ladder Logic . . . . .	19
3.2.2 CX - Programmer . . . . .	21
3.3 Desfășurarea lucrării . . . . .	28
3.3.1 Probleme rezolvate . . . . .	28
3.3.2 Probleme propuse . . . . .	33
<b>Bibliografie</b>	<b>35</b>
<b>4 Dezvoltarea aplicațiilor folosind SFC (Sequential Function Chart)</b>	<b>36</b>
4.1 Obiective . . . . .	36
4.2 Considerații teoretice . . . . .	36
4.2.1 CX - Programmer . . . . .	36
4.3 Desfășurarea lucrării . . . . .	47
4.3.1 Probleme rezolvate . . . . .	47
4.3.2 Probleme propuse . . . . .	49

<b>Bibliografie</b>	<b>55</b>
<b>5 Modelarea SED folosind rețele Petri - Evoluția marcajului rețelelor Petri</b>	<b>56</b>
5.1 Obiective . . . . .	56
5.2 Considerații teoretice . . . . .	56
5.2.1 Admisibilitatea tranzițiilor . . . . .	57
5.2.2 Comportamentul dinamic . . . . .	58
5.3 Desfășurarea lucrării . . . . .	59
5.3.1 Probleme rezolvate . . . . .	59
5.3.2 Probleme propuse . . . . .	64
<b>Bibliografie</b>	<b>70</b>
<b>6 Modelarea SED folosind rețele Petri - Proprietăți ale rețelelor Petri</b>	<b>71</b>
6.1 Obiective . . . . .	71
6.2 Considerații teoretice . . . . .	71
6.3 Desfășurarea lucrării . . . . .	73
<b>Bibliografie</b>	<b>79</b>
<b>7 Modelarea SED folosind rețele Petri - Analiza calitativă a modelelor descrise prin rețele Petri. Graful de realizare. Arborele de acoperire</b>	<b>80</b>
7.1 Obiective . . . . .	80
7.2 Considerații teoretice . . . . .	80
7.2.1 Analiza prin enumerare . . . . .	81
7.3 Desfășurarea lucrării . . . . .	82
7.3.1 Probleme propuse . . . . .	82
<b>Bibliografie</b>	<b>88</b>
<b>8 Modelarea SED folosind rețele Petri - Metode algebrice de analiză. Utilizarea invariantilor</b>	<b>89</b>
8.1 Obiective . . . . .	89
8.2 Considerații teoretice . . . . .	89
8.2.1 Invariantul de tip P . . . . .	89
8.2.2 Invariantul de tip T . . . . .	90
8.3 Desfășurarea lucrării . . . . .	91
<b>Bibliografie</b>	<b>94</b>
<b>9 Sub clase de rețele Petri - Sifoane și capcane</b>	<b>95</b>
9.1 Obiective . . . . .	95
9.2 Considerații teoretice . . . . .	95

9.3 Desfășurarea lucrării . . . . .	96
<b>Bibliografie</b>	<b>100</b>
<b>10 Modelarea SED folosind rețele Petri</b>	<b>101</b>
10.1 Obiective . . . . .	101
10.2 Desfășurarea lucrării . . . . .	101
<b>Bibliografie</b>	<b>104</b>
<b>11 Modelarea SED folosind rețele Petri - Rețele Petri stochastice</b>	<b>106</b>
11.1 Obiective . . . . .	106
11.2 Considerații teoretice . . . . .	106
11.2.1 Ordinea de execuție a tranzitțiilor într-o rețea Petri stochastică . . . . .	107
11.2.2 Analiza rețelelor Petri stochastice . . . . .	107
11.3 Desfășurarea lucrării . . . . .	109
<b>Bibliografie</b>	<b>112</b>

# LISTĂ DE FIGURI

1.1 Procesul 1 . . . . .	4
1.2 Procesul 2 . . . . .	4
1.3 Procesul 3 . . . . .	5
2.1 Etape . . . . .	8
2.2 Tranzitii . . . . .	9
2.3 Procesul 1 . . . . .	10
2.4 Procesul 1 - Modelul GRAFCET . . . . .	11
2.5 Procesul 2 . . . . .	11
2.6 Procesul 2 - Modelul GRAFCET . . . . .	13
2.7 Procesul 3 . . . . .	14
2.8 Procesul 4 . . . . .	15
3.1 Schema bloc a unui AP . . . . .	20
3.2 Deschiderea aplicatiei (Selecteaza OMRON) - pas 1 . . . . .	22
3.3 Deschiderea aplicatiei (Selecteaza CX-programmer) - pas 2 . . . . .	22
3.4 Proiect nou . . . . .	23
3.5 Configurarea noului proiect . . . . .	24
3.6 Proiect nou - modelul PLC-ului . . . . .	25
3.7 Proiect nou - protocolul de comunicatie . . . . .	25
3.8 Project Workspace . . . . .	26
3.9 Definirea simbolurilor - exemplu . . . . .	26
3.10 Foaia de lucru - exemplu . . . . .	27
3.11 Linie de cod Ladder Logic - exemplu . . . . .	28
3.12 Simboluri locale . . . . .	29
3.13 Introducerea unui nou simbol . . . . .	29
3.14 Lista simbolurilor necesare . . . . .	30
3.15 Secvențe de cod Ladder Logic . . . . .	30
3.16 Configurare Clock . . . . .	31
3.17 Stare initială . . . . .	31
3.18 Programul Ladder Logic . . . . .	32
3.19 Sfârșit program . . . . .	32
3.20 Simulare - <i>Work online Simulator</i> - opțiune din meniul Simulare . . . . .	33
3.21 Simulare - Transfer cu succes programului . . . . .	34
3.22 Simulare - secvență de funcționare . . . . .	34
4.1 OMRON . . . . .	37

4.2 Cx-programmer . . . . .	37
4.3 Creearea unui proiect nou . . . . .	38
4.4 Configurarea unui nou proiect, pas 1 . . . . .	39
4.5 Configurarea unui nou proiect, pas 2 . . . . .	39
4.6 Conexiunea cu PLC-ul . . . . .	40
4.7 Conexiunea cu Simulatorul . . . . .	40
4.8 Workspace-ul cx-programmer . . . . .	41
4.9 Ștergerea programului Ladder Logic, (i) . . . . .	42
4.10 Ștergerea programului Ladder Logic, (ii) . . . . .	42
4.11 Program nou de tip SFC . . . . .	43
4.12 Redenumirea unui program SFC . . . . .	43
4.13 Definirea limbajului default de programare . . . . .	44
4.14 Opțiuni disponibile pentru etape . . . . .	44
4.15 Adăugarea unei etape noi . . . . .	44
4.16 Redenumirea unei etape . . . . .	45
4.17 Redenumirea etapei inițiale . . . . .	45
4.18 Opțiuni disponibile pentru tranziții . . . . .	45
4.19 Adăugarea unei tranziții noi . . . . .	46
4.20 Redenumirea unei tranziții . . . . .	46
4.21 Înregistrarea unei tranziții noi (i) . . . . .	47
4.22 Înregistrarea unei tranziții noi (ii) . . . . .	47
4.23 Adăugarea legăturilor între noduri - arce . . . . .	48
4.24 Elemente grafice disponibile . . . . .	48
4.25 Programul SFC . . . . .	49
4.26 Lista de simboluri . . . . .	50
4.27 Adăugarea acțiunilor (i) . . . . .	50
4.28 Adăugarea acțiunilor (ii) . . . . .	51
4.29 Adăugarea acțiunilor (iii) . . . . .	51
4.30 Adăugarea tranzițiilor (i) . . . . .	52
4.31 Adăugarea tranzițiilor (ii) . . . . .	53
4.32 Adăugarea tranzițiilor (iii) . . . . .	53
4.33 Adăugarea tranzițiilor (iv) . . . . .	54
4.34 Proces . . . . .	54
5.1 Rețeaua Petri 1 . . . . .	59
5.2 Evoluția marcajului pentru RP 1 . . . . .	59
5.3 Rețeaua Petri 2 . . . . .	65
5.4 Rețeaua Petri 3 . . . . .	65
5.5 Rețeaua Petri 4 . . . . .	66
5.6 Rețeaua Petri 5 . . . . .	67
5.7 Rețeaua Petri 6 . . . . .	67

5.8 Rețeaua Petri 7 . . . . .	68
5.9 Rețeaua Petri 8 . . . . .	68
5.10 Rețeaua Petri 9 . . . . .	69
5.11 Rețeaua Petri 10 . . . . .	69
6.1 Rețeaua Petri 1 . . . . .	74
6.2 Rețeaua Petri 2 . . . . .	74
6.3 Rețeaua Petri 3 . . . . .	75
6.4 Rețeaua Petri 4 . . . . .	75
6.5 Rețeaua Petri 5 . . . . .	76
6.6 Rețeaua Petri 6 . . . . .	76
6.7 Rețeaua Petri 7 . . . . .	77
6.8 Rețeaua Petri 8 . . . . .	77
6.9 R.P. 9 . . . . .	78
6.10 R.P. 10 . . . . .	78
7.1 R.P. 1 . . . . .	83
7.2 R.P. 2 . . . . .	83
7.3 R.P. 3 . . . . .	84
7.4 R.P. 4 . . . . .	84
7.5 R.P. 5 . . . . .	85
7.6 R.P. 6 . . . . .	85
7.7 R.P. 7 . . . . .	86
7.8 R.P. 8 . . . . .	86
7.9 R.P. 9 . . . . .	87
7.10 R.P. 10 . . . . .	87
8.1 R.P. 1 . . . . .	91
8.2 R.P. 2 . . . . .	91
8.3 R.P. 3 . . . . .	92
8.4 R.P. 4 . . . . .	92
8.5 R.P. 5 . . . . .	93
8.6 R.P. 6 . . . . .	93
8.7 R.P. 7 . . . . .	93
9.1 R.P. 9.1 . . . . .	97
9.2 R.P. 9.2 . . . . .	97
9.3 R.P. 9.3 . . . . .	98
9.4 R.P. 9.4 . . . . .	98
9.5 R.P. 9.5 . . . . .	99
9.6 R.P. 9.6 . . . . .	99

---

10.1 Procesul 1 . . . . .	102
10.2 Procesul 2 . . . . .	102
10.3 Procesul 3 . . . . .	103
11.1 Rețeaua Petri stochastică 1 . . . . .	107
11.2 Proces 2 - RPS . . . . .	110
11.3 Proces 3 - RPS . . . . .	110
11.4 Proces 4 . . . . .	111
11.5 Proces 5 - RPS . . . . .	111

## LISTĂ DE TABELE

# 1 DESCRIEREA LOGICĂ A FUNCȚIONĂRII SISTEMELOR CU EVENIMENTE DISCRETE

---

## 1.1 OBIECTIVE



- Principii fundamentale SED;
- Utilizarea limbajelor naturale la modelarea SED.

## 1.2 CONSIDERAȚII TEORETICE

Sistemele cu evenimente discrete (SED) se deosebesc în mod esențial de celelalte categorii de sisteme prin două caracteristici majore:

- Comportarea lor dinamică este determinată de producerea (realizarea) unor evenimente (engl.: event - driven) și nu antrenată de timp (engl.: time driven), ca în cadrul altor categorii. În cazul sistemelor continue, acțiunile de control sunt permanente, continue în timp, iar în cazul sistemelor cu timp discret, au loc la momente bine precizate de timp. Evenimente pot fi considerate apăsarea unui buton, extragerea unei piese dintr-un container, defectarea imprevizibilă a unui calculator, sau depășirea unei anumite valori a mărimii urmărite. Un eveniment nu are durată.
- Cel puțin o parte a variabilelor care descriu sistemul, sunt cuantificate; un exemplu de astfel de variabile îl constituie descriptorii de stare ai unei resurse (activ, inactiv, sus, jos, etc.).
- Din punct de vedere formal, sistemele cu evenimente discrete sunt considerate, de obicei, ca sisteme dinamice (sisteme dinamice cu evenimente discrete sau SDED) caracterizate prin existența unui spațiu al stărilor și a unui mecanism de tranziție a stărilor. Evoluția SED fiind determinată de evenimente, modelarea și analiza lor necesită un cadru matematic nou, diferit de cele anterioare, bazate pe ecuații diferențiale sau ecuații cu diferențe finite.

Abordările SED pot fi grupate luând în considerare principiile fundamentale și implicit direcțiile de cercetare următe. În acest sens sunt menționate:

- limbaje naturale;
- limbaje formale și automate;
- limbaje de simulare a evenimentelor discrete;
- automate stochastice temporizate;
- mașini cu stări finite;
- rețele Petri (diferite tipuri) și grafuri de evenimente;
- procese recursive finite;
- algebra proceselor;
- lanțuri Markov;
- rețele de cozi (de așteptare);
- abordările orientate pe modele din știința calculatoarelor;
- diagrame ale fluxurilor de date etc.

*Intrările* într-un proces cu evenimente discrete sunt condiții și evenimente. Starea procesului se poate schimba dacă este adevărată o condiție, sau când se produce un eveniment.

*Starea* (mediului, sau în general a SED) este informația necesară pentru determinarea evoluției sistemului (de exemplu: numărul de piese dintr-un buffer, nivelul lichidului într-un rezervor). Starea unui SED poate fi reprezentată deseori prin variabile logice.

O propoziție logică formată cu variabile logice și operatori logici poartă numele de *predicat*. Un predicat poate lua valorile: *adevărat* și *fals*. Pentru descriere, nu se folosesc întotdeauna expresii logice. Predicatul este utilizat pentru descrierea sau condiționarea comportamentelor sistemelor cu evenimente discrete.

Un eveniment poate fi definit ca trecerea de la o valoare la alta a unei variabile logice. Evenimentul nu are durată, sau durează zero unități de timp.

Ieșirile pot fi sub formă de:

- *nivel* (modelate prin variabile logice care iau valoarea 1 sau 0 pentru intervale de timp finite sau infinite), corespunzând unor componente ale stării;
- *impuls* (nu au durată), corespunzând unor evenimente.

Descrierea logică a sistemelor cu evenimente discrete implică o succesiune de acțiuni. Acțiunile care reprezintă (sau pot fi formulate ca niște) condiții se numesc *acțiuni condiționale*. În urma altor acțiuni se atribuie anumite valori logice variabilelor logice care descriu starea unui element. Ele se numesc acțiuni asupra nivelului (acțiuni de tip nivel). Exemple de acțiuni asupra nivelului sunt: umplerea unui rezervor, deplasarea brațului unui robot spre dreapta etc. Acțiunile asupra nivelului sunt definite atunci când starea sistemului este stabilă. O altă categorie de acțiuni o constituie *acțiunile impulsionale* (*acțiuni de tip impuls*). Ele sunt foarte scurte, teoretic nu au durată și sunt executate atunci când starea locală din care sunt posibile trebuie să devină activă după ce a fost inactivă. Exemple de acțiuni impulsionale sunt comenzi: închide vana, pornește motorul, sau modifică valoarea unei variabile. Acțiunile asupra nivelului și acțiunile condiționale sunt reprezentate prin valori ale variabilelor logice, în timp ce comenzi și acțiunilor impulsionale li se asociază schimbări de stare ale variabilelor logice. Evenimentelor li se asociază de asemenea schimbări ale valorilor variabilelor logice.

Se consideră că b este o variabilă logică.  $\uparrow b$  simbolizează un evenimentul determinat de schimbarea valorii lui b.  $\uparrow a * b$  este un eveniment care se produce în același timp cu  $\uparrow a$ , dacă  $b = 1$  în momentul respectiv.

## 1.3 DESFĂȘURAREA LUCRĂRII

### 1.3.1 PROBLEME PROPUSE

Realizați descrierea logică a proceselor care urmează:

1. Se consideră un rezervor principal conectat la două rezervoare secundare, conform figurii 1.1.

Se consideră că unul din rezervoarele secundare este gol, dacă nivelul lichidului din el este situat sub un nivel  $b_i$  ( $i = 1, 2$ ) precizat și este plin dacă nivelul lichidului depășește nivelul  $h_i$  ( $i = 1, 2$ ). În starea inițială rezervoarele secundare sunt goale. În momentul în care se apasă butonul m, se deschid vanele  $v_1$  și  $v_2$  și cele două rezervoare încep să se umple. Când unul din rezervoare s-a umplut, se oprește acțiunea de umplere prin închiderea vanei  $v_i$  și conținutul rezervorului începe să fie utilizat prin deschiderea vanei  $w_i$ . Vana  $w_i$  se închide când rezervorul s-a golit. Procesul de umplere a rezervoarelor secundare poate începe din nou în momentul în care ambele rezervoare s-au golit.

- Precizați succesiunea de acțiuni corespunzătoare descrierii de mai sus.
- Stabiliți care sunt acțiuni asupra nivelului (de tip nivel) și care sunt acțiuni impulsionale.
- Asociați acțiunilor variabile logice și realizați descrierea logică aferentă.

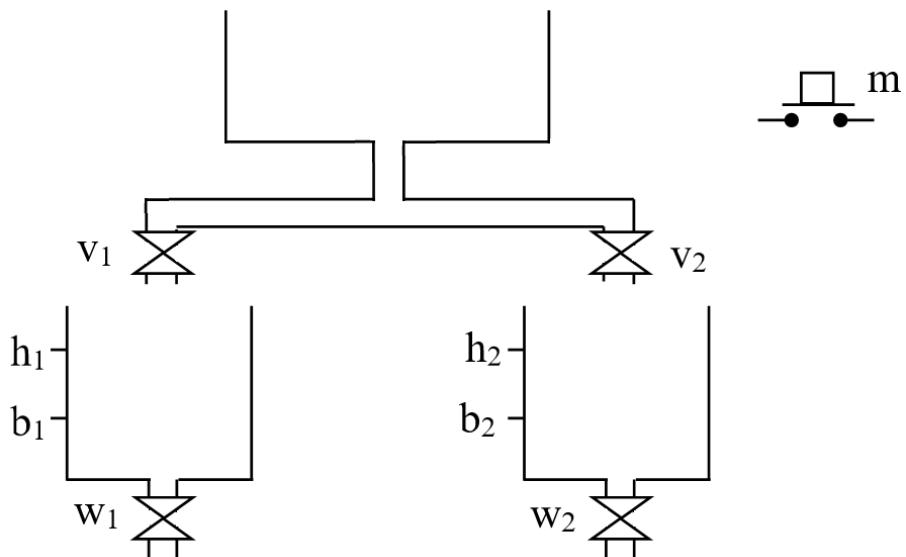


Figure 1.1: Procesul 1

2. Se consideră un cărucior aflat în repaus în punctul A (figura 1.2). Când se apasă un buton de comandă, căruciorul se deplasează la dreapta, apoi ajunge în punctul B și imediat se deplasează la stânga, revenind în punctul A. Se consideră că deplasarea căruciorului este asigurată de un motor și că starea inițială corespunde staționării căruciorului în punctul A.

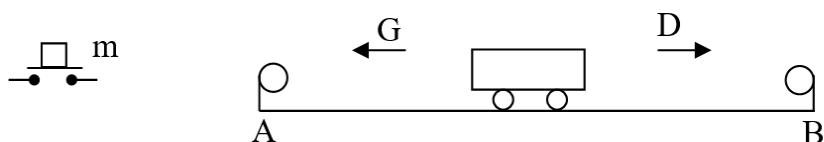


Figure 1.2: Procesul 2

- a) Precizați succesiunea de acțiuni corespunzătoare descrierii de mai sus.
  - b) Stabiliți care sunt acțiuni asupra nivelului (de tip nivel) și care sunt acțiuni impulsionale.
  - c) Asociați acțiunilor variabile logice și realizați descrierea logică aferentă.
3. Se consideră procesul din figura 1.3. Un cărucior se deplasează pe o shină, oprirea acestuia se poate realiza folosind semnalele de la contactele C1, C2 și C3. Un motor

acționează scripetele pentru o deplasare sus jos a cuvei. Stările cuvei sunt: sus (la nivelul h), jos la nivelul (b) și se deplasează (sus sau jos).

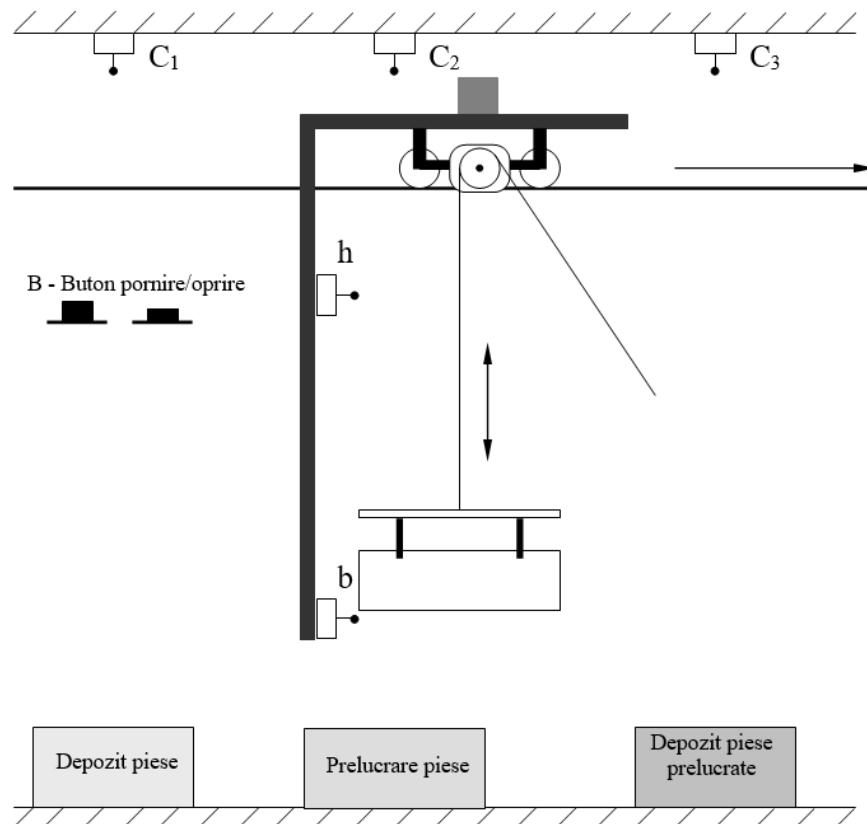


Figure 1.3: Procesul 3

În primul depozit („Depozit piese”) este o resursă infinită de piese care trebuie prelucrată. Montajul de manipulare preia piese din primul depozit și le transportă la secțiunea de prelucrare („Prelucrare piese”). Când piesele sunt prelucrate sunt preluate și transportate la depozit („Depozitul piese prelucrate”) de către montajul de manipulare. Procesul se repetă până se comandă oprirea sistemului. La pornirea sistemului trebuie poziționat montajul de manipulare pe poziția C1. Oprirea sistemului se realizează apăsând butonul B, dacă manipulatorul este între două depozite și este încărcat atunci continuă deplasarea până la poziția următoare.

- Precizați succesiunea de acțiuni corespunzătoare descrierii de mai sus.
- Stabiliti care sunt acțiuni asupra nivelului (de tip nivel) și care sunt acțiuni impulsionale.

- c) Asociați acțiunilor variabile logice și realizați descrierea logică aferentă.
4. Se consideră un divizor care realizează o divizare cu 2. Se notează cu  $i$  semnalul de intrare și cu  $I$  semnalul de ieșire. Ieșirea  $I$  își schimbă starea de fiecare dată când  $i$  trece de la valoarea logică 0 la valoarea logică 1.
- Reprezentați variația semnalului de ieșire în funcție de variația semnalului de intrare.
  - Precizați succesiunea de acțiuni corespunzătoare descrierii de mai sus.
  - Stabiliți care sunt acțiuni asupra nivelului (de tip nivel) și care sunt acțiuni impulsionale (de tip impuls).
  - Asociați acțiunilor variabile logice și realizați descrierea logică aferentă.

# BIBLIOGRAFIE

- [1] Adina Aştilean. *Sisteme cu evenimente discrete, notițe curs.*
- [2] Christos G. Cassandras, Stéphane Lafortune. *Introduction to Discrete Event Systems, Second Edition*, ISBN-13: 978-0-387-33332-8, e-ISBN-13: 978-0-387-68612-7, 2008 Springer Science+Business Media, LLC.

## 2 ABORDAREA SED PRIN METODA GRAFCET

---

### 2.1 OBIECTIVE



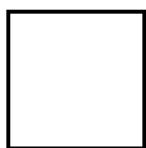
- Familiarizarea cu metoda GRAFCET de modelarea a SED;
- Realizarea de modele bazate pe metoda GRAFCET.

### 2.2 CONSIDERAȚII TEORETICE

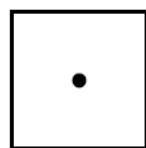
Un grafcet este un graf în care există două tipuri de noduri: etape sau pași și tranziții. Etapele și tranzițiile sunt conectate cu arce orientate.

- *Etape*

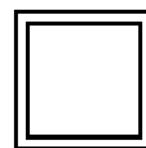
Etapele sunt reprezentate prin pătrate, figura 2.1 a. O etapă poate fi *activă* sau *inactivă*. Etapa activă este reprezentată prin desenarea unui punct în interiorul pătratului, figura 2.1 b. Etapele care sunt active la startarea sistemului se numesc *etape inițiale* și sunt reprezentate prin dublarea conturului etapei, figura 2.1 c.



(a)



(b)



(c)

Figure 2.1: Etape

Totalitatea etapelor active la un moment dat definește *situația* în acel moment. O situație corespunde unei stări a sistemului. Fiecare etapă activă este o componentă

a stării, purtând o informație și atribuindu-se o semnificație. *Semnificația simplă* corespunde situației în care etapei i se atribuie o singură semnificație. *Semnificația multiplă* corespunde situației în care etapei i se atribuie mai multe semnificații. Etapelor le pot fi asociate acțiuni și variabile logice. Acțiunile sunt menționate într-un dreptunghi situat la dreapta etapei. O variabilă logică  $X_i$ , asociată unei etape i, are valoarea 1 atunci când etapa respectivă este activă.

- *Tranziții*

O tranziție este reprezentată printr-o linie, figura 2.2a, dacă există o singură ramură de intrare și o singură ramură de ieșire din tranziție. Dacă există două sau mai multe ramuri de ieșire din tranziție, linia simplă este urmată de două linii duble, figura 2.2b. Dacă există mai multe ramuri de intrare în tranziție linia simplă este precedată de două linii duble, figura 2.2c.

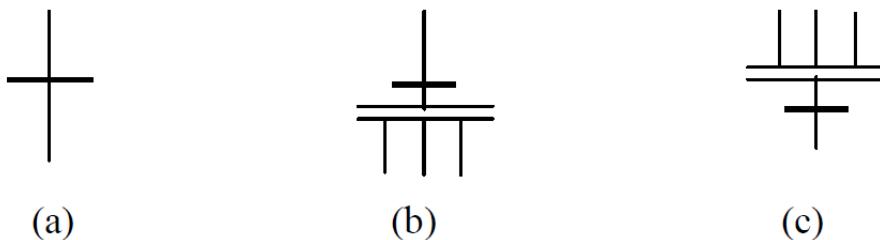


Figure 2.2: Tranzitii

Fiecărei tranziții i se asociază o receptivitate  $R_i$ . Receptivitatea poate fi exprimată ca o condiție logică, sau un eveniment extern.

- *Arce*

Arcele stabilesc legătura dintre tranziții și etape. În reprezentarea normalizată numai arcele orientate de jos în sus au sensul de parcurs indicat printr-o săgeată.

## 2.3 DESFĂȘURAREA LUCRĂRII

### 2.3.1 PROBLEME PROPUSE

Analiza unor procese și modelarea lor folosind metoda GRAFCET:

1. Se consideră procesul din figura 2.3. Se va realiza modelul GRAFCET pentru controlul unei bariere, considerând următoarele:
  - Bariera stă ridicată 20 secunde;

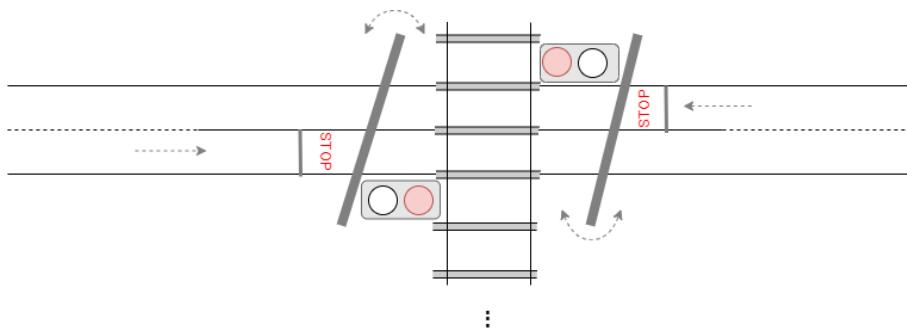


Figure 2.3: Procesul 1

- Bariera stă coborâtă 5 secunde;
  - Cât timp bariera este ridicată este aprins un bec alb;
  - Cât timp bariera este coborâtă este aprins un bec roşu;
  - Etapa 1 - bariera este coborâtă;
  - Etapa 2 - bariera este ridicată;
  - R1 - Bariera se ridică după 5 secunde;
  - R2 - Bariera se coboară după 20 secunde;
  - Actiune 1 - se aprinde un bec roşu;
  - Actiune 2 - se aprinde un bec alb.

Modelul GRAFCET pentru procesul din figura 2.3 respectând cerințele de mai sus este prezentat în figura 2.4.

Modelul poate fi extins adăugând următoarele funcționalități:

- Se pot considera ambele bariere;
  - Se pot introduce senzori de prezență auto / tren pentru avertizare și control;
  - Se pot preciza comenziile motoarelor de acționarea barierelor.
  - ...

2. Se consideră procesul din figura 2.5.

Se va realiza modelul GRAFCET pentru procesul din figura 2.5, considerănd următoarele:

- Se extrag două tipuri de produse ( $P_1$  și  $P_2$ );

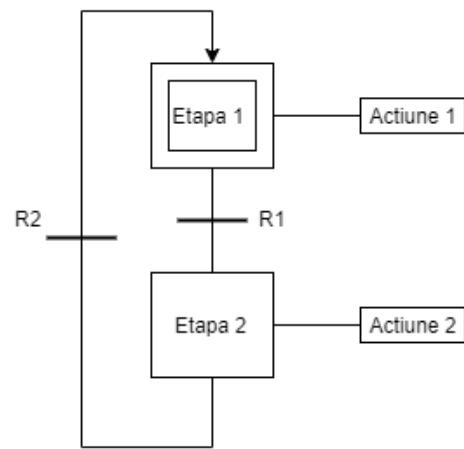


Figure 2.4: Procesul 1 - Modelul GRAFCET

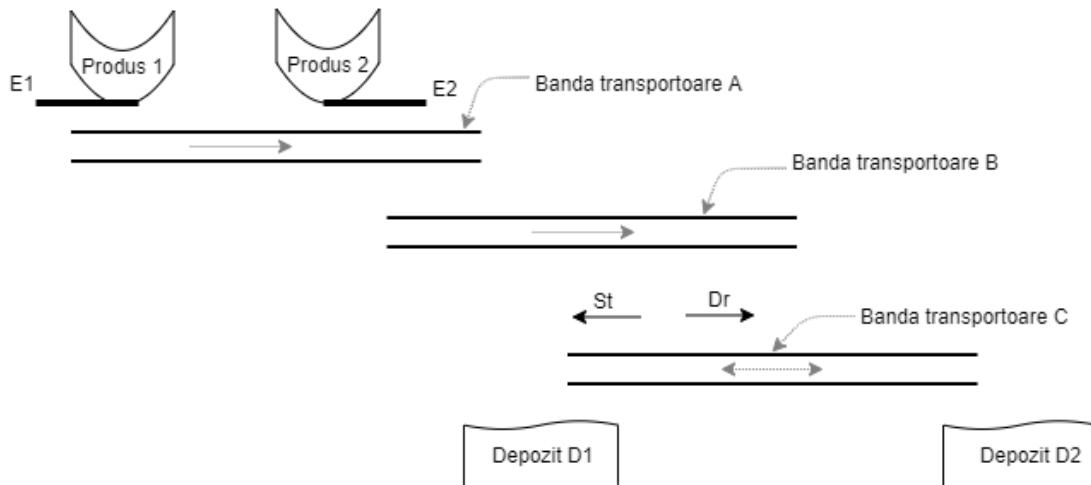


Figure 2.5: Procesul 2

- Produsele sunt extrase cu ajutorul unor extractoare ( $E_1$  și  $E_2$ ) pe banda transportoare A;
- Sensul de deplasare a benzilor transportoare A, B și C este reprezentat în figură;
- Produsele extrase vor fi introduse în depozite ( $D_1$  și  $D_2$ ) folosind benzile transportoare;
- În funcție de depozitul disponibil se acționează corespunzător benzile transportoare;
- Pentru a introduce piese în depozitul  $D_1$  se acționează C înspre stânga iar pentru a introduce piese în depozitul  $D_2$  se acționează C înspre dreapta;
- Banda transportoare B este acționată pentru 10 secunde;
- Banda transportoare A este acționată pentru 5 secunde;
- Extractoarele sunt acționate pentru 5 secunde (pentru produsele  $P_1$  și  $P_2$ );
- Oprirea procesului se face astfel:
  - Se opresc extractoarele ( $E_1$  și  $E_2$ );
  - După 10 secunde se oprește banda transportoare A (suficient pentru ca produsele să ajungă pe banda transportoare B);
  - După 5 secunde se oprește banda transportoare B;
  - După 5 secunde se oprește banda transportoare C;

Modelul GRAFCET pentru procesul din figura 2.5 respectând cerințele de mai sus este prezentat în figura 2.6.

### 2.3.2 PROBLEME PROPUSE

1. Descrieți, utilizând metoda Grafcet, procesele 1, 2 și 3 din lucrarea precedentă.
  - Precizați semnificațiile atribuite fiecărei etape.
  - În cazul grafului asociat procesului 1, precizați care etape pot avea o semnificație simplă și care pot avea o semnificație multiplă.
  - Transformați reprezentarea în care există etape cu semnificație multiplă, într-o reprezentare în care toate etapele au semnificații simple.
2. Se ia în discuție un proces chimic care combină două substanțe, prezentat în figura 2.7. Să se realizeze modelul GRAFCET al procesului.

Se introduce substanța 1, prin deschiderea robinetului  $V_1$ , până la nivelul  $L_1$  apoi se introduce substanța 2 prin deschiderea robinetului  $V_2$  în timp ce se amestecă soluția. La atingerea nivelului  $L_2$  se deschide robinetul  $V_3$  până la golirea rezervorului. Secvența de operații este următoarea:

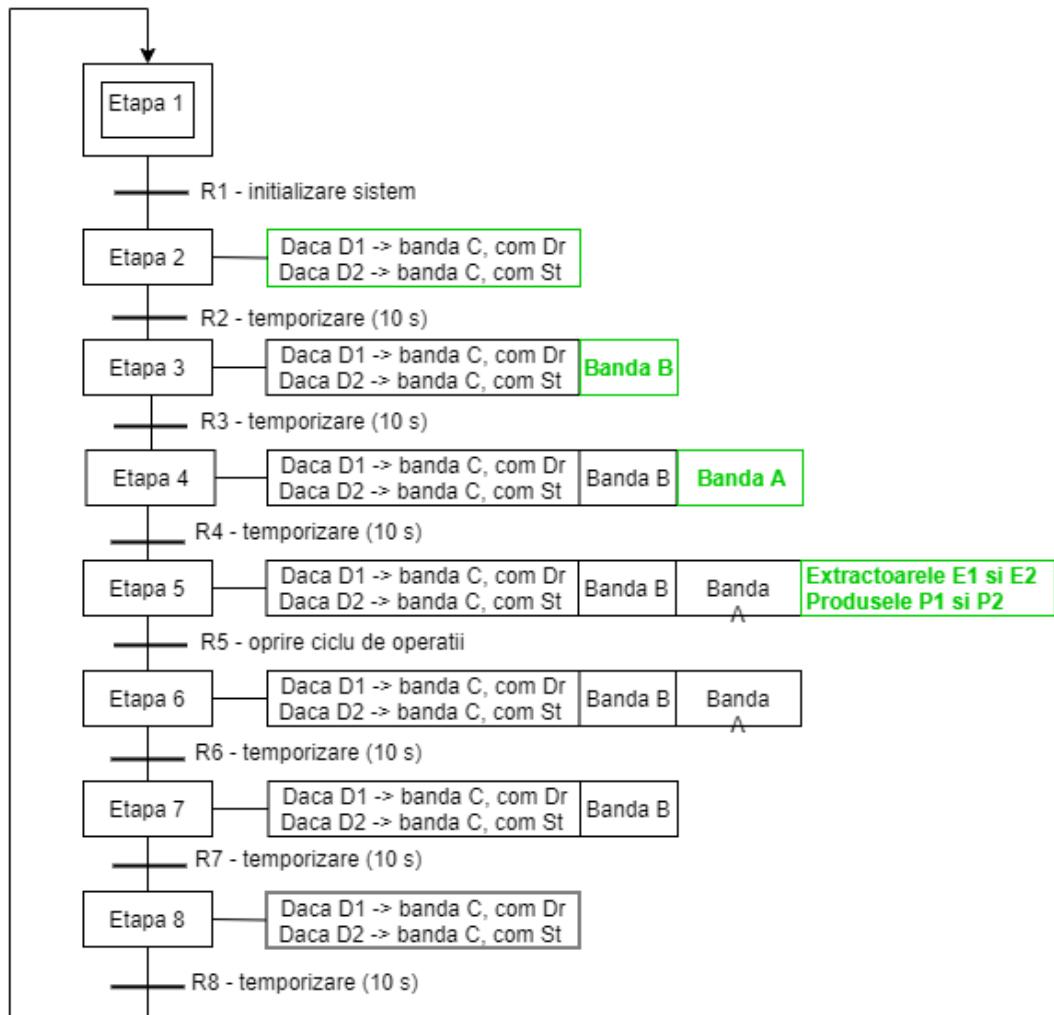


Figure 2.6: Procesul 2 - Modelul GRAFCET

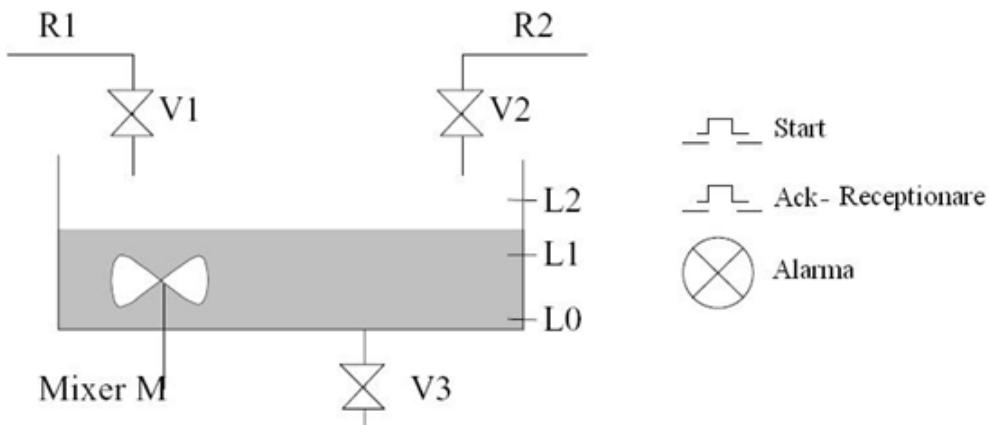


Figure 2.7: Procesul 3

- Sistemul este pornit la apăsarea butonului „START”, robinetul  $V_1$  este deschis până la atingerea nivelului  $L_1$ ;
- Când nivelul  $L_1$  este atins se pornește mixerul simultan cu deschiderea robinetului  $V_2$  și închiderea robinetului  $V_1$ ;
- Când nivelul  $L_2$  este atins mixerul se oprește, robinetul  $V_2$  se închide iar robinetul  $V_3$  se deschide. Robinetul  $V_3$  rămâne deschis până când nivelul substanței coboară sub  $L_0$ ;
- După 10 minute dacă nivelul substanței nu este sub  $L_0$  se pornește o alarmă;
- Butonul Ack oprește alarmă și permite repornirea procesului de control.

### 3. Descrieți procesul din figura 2.8 folosind metoda GRAFCET.

Un vagonet se deplasează de la dreapta (D) la stânga (G). Preia piese PP de pe banda A (limitatorul y) sau B (limitatorul y) și le depozitează pe platforma x. Se deplasează până la limita  $V-$ , aici este împins de către o paletă pe transportorul secundar C până la  $P+$ . Paleta se retrage până la  $P-$  iar platforma revine la nivelul  $V+$ . Transportorul C se presupune că este tot timpul în mișcare. Transportoarele A și B sunt acționate de către sistemul de comandă.

Realizați modelul GRAFCET pentru cazul în care sistemul primește piese doar pe transportorul A.

Modificați modelul anterior realizat pentru situația în care piesele sosesc pe ambele transportoare dar există butoane de comandă pentru a specifica pe ce bandă să vină piesa.

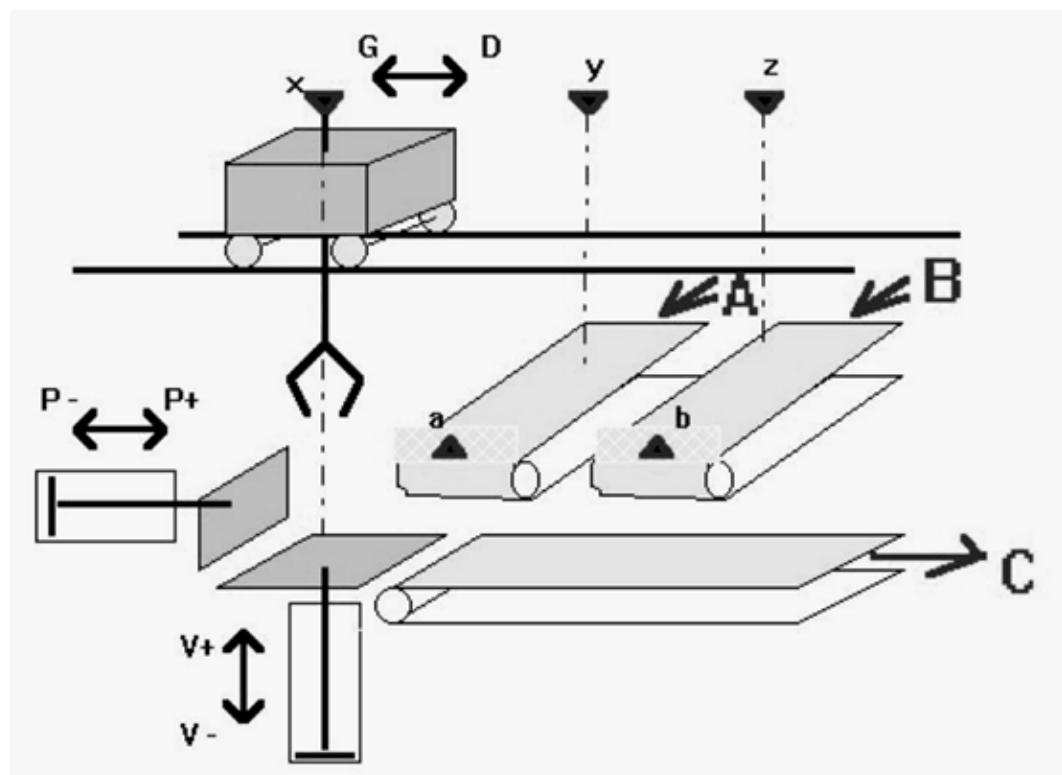


Figure 2.8: Procesul 4

Realizați modelul GRAFCET pentru situația în care piesele pot să vină pe ambele transportoare (A și B) indiferent de moment respectând următoarele condiții:

- Dacă este piesă și pe a și pe b se ia piesa de pe a.
- În caz de conflict se preia piesa de pe banda transportoare care nu a fost servită anterior.  
Atenție: dacă sunt piese doar pe o bandă sistemul nu se blochează.
- Modifică prioritățile de la un pas la altul astfel încât piesele să fie preluate de pe ambele benzi.
- **Atenție: O bandă transportoare nu poate sta mai mult de doi pași.**

4. Realizați modelul GRAFCET pentru următorul proces:

Într-o încăpere intră și ies pe rând persoane. La un moment dat în camera pot fi maxim două persoane. În momentul în care sunt 2 persoane se aprinde un bec.

## BIBLIOGRAFIE

- [1] Adina Aştilean. *Sisteme cu evenimente discrete, notițe curs.*
- [2] Christos G. Cassandras, Stéphane Lafortune. *Introduction to Discrete Event Systems, Second Edition*, ISBN-13: 978-0-387-33332-8, e-ISBN-13: 978-0-387-68612-7, 2008 Springer Science+Business Media, LLC.

## 3 DEZVOLTAREA APLICAȚIILOR FOLOSIND LADDER LOGIC

---

### 3.1 OBIECTIVE



- Familiarizarea cu mediul de dezvoltare OMRON.
- Dezvoltarea de programe folosind limbajul ladder logic.

### 3.2 CONSIDERAȚII TEORETICE

Ce este un automat programabil? Un automat programabil (AP) este format din:

- **Central Processing Unit (CPU)** - aici este încărcat un program;
- **Interfață Intrare / Ieșire** – această interfață este conectată direct în proces.

*Programul* - controlează AP iar la un semnal de la un dispozitiv conectat la intrare reacționează (în general determină o modificare a semnalului unei ieșiri).

*CPU* – este un microprocesor care coordonează activitățile AP, execută un program, procesează semnalele de intrare / ieșire și comunică cu dispozitivele externe.

*Memoria* – există mai multe tipuri de memorie. În această zonă este stocat sistemul de operare, utilizatorul are o zonă liberă pentru aplicații. Sistemul de operare este sistemul care coordonează AP. Ladder program și timer-ul sunt stocate în zona de memorie pentru utilizator.

Există mai multe tipuri de memorie:

- Read Only Memory (ROM) – este o memorie nevolatilă care poate fi programată doar o singură dată.
- Random Access Memory (RAM) – este memoria cea mai utilizată pentru zona de memorie alocată utilizatorului. Datele sunt volatile adică dacă se întârzie alimentarea informațiile sunt pierdute, această problemă este eliminată folosind o baterie.

- Erasable Programmable Read Only Memory (EPROM) – stochează permanent datele și nu are nevoie de baterie. Informațiile pot fi șterse cu ajutorul unei raze ultraviolete.
- Electrically Erasable Programmable Read Only Memory (EEPROM) – combină flexibilitatea memoriei RAM și nevolabilitatea memoriei EPROM. Această memorie poate fi reprogramată electric, (are un număr limitat de rescrieri).

*Scan Time* – procesul de citire a intrărilor este cunoscut și sub numele de scanare. Acest proces (este în general continuu și secvențial) și realizează citarea stării intrărilor, evaluatează controlul logic și recalculează ieșirile. Acest proces specifică cât de repede reacționează controlerul la modificările mărimilor de intrare.

Scan time este influențat de mai mulți factori, cum ar fi:

- Timpul necesar pentru realizarea unei scanări variază de la 0.1 ms până la zeci de ms (depinde de viteza CPU și de programul care rulează).
- Citirea intrărilor și a ieșirilor și transmiterea noilor valori controlerului.
- Monitorizarea programului de control.

Automatele Programabile au câștigat teren în controlul proceselor datorită avantajelor care le aduc, cum ar fi:

- + Costuri reduse pentru controlul unor procese complexe;
- + Flexibile, pot fi reutilizate la controlul altor procese repede și ușor;
- + Posibilitatea de încărcare în memoria proprie a unor programe permite un control mai sofisticat;
- + Componete rezistente care permit utilizarea pe un interval de timp mare (mai mulți ani).

### 3.2.1 LADDER LOGIC

Este principalul program folosit pentru Automate Programabile (AP / PLC). A fost dezvoltat să imite comportamentul releelor utilizate înainte de apariția AP-urilor. Un releu este un dispozitiv simplu care folosește câmpul magnetic pentru închidere sau deschidere. Un AP este format din modulul de intrare, modulul de ieșire și logica internă. În figura 3.1 este reprezentat schematic un AP. În programul scris folosind ladder logic se face legătura între intrări, ieșiri și variabilele interne definite pentru rezolvarea aplicației.

Pentru analiza unei noi aplicații trebuie să respectăm următorii pași:

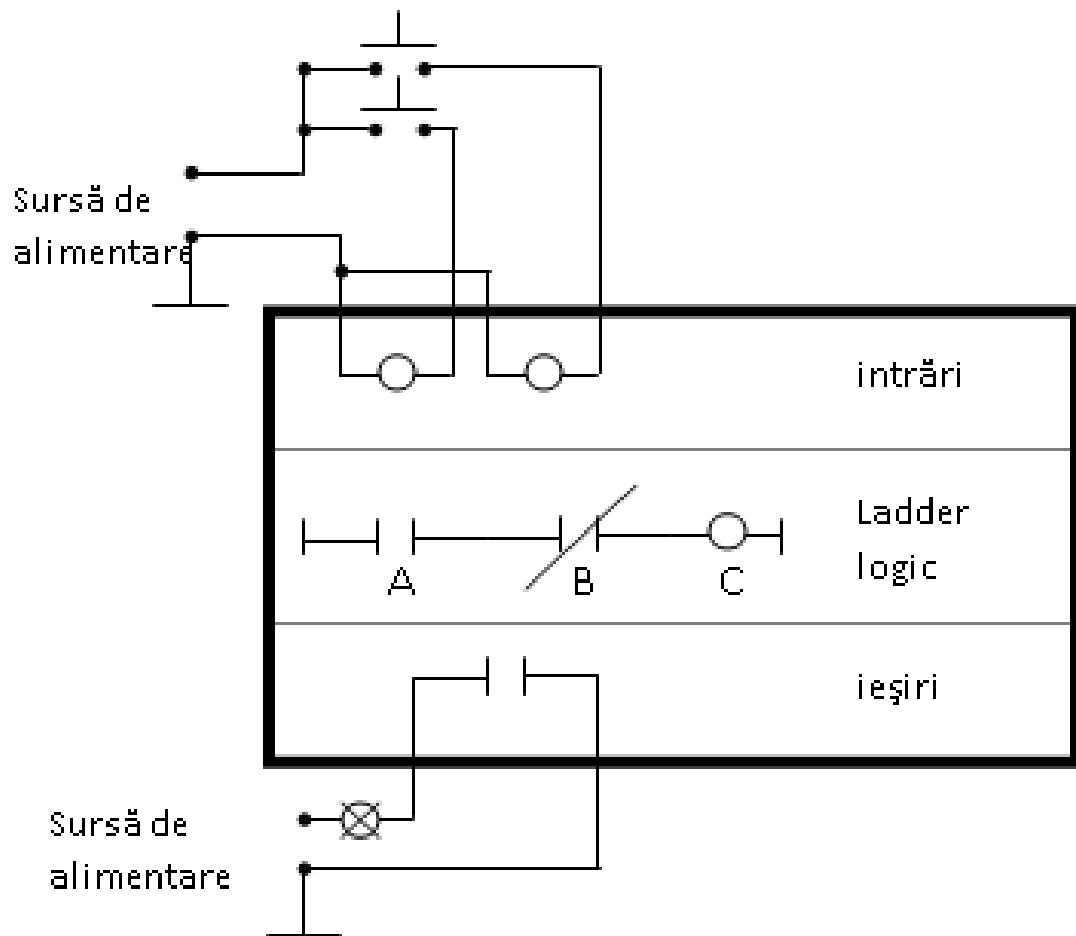


Figure 3.1: Schema bloc a unui AP

- *Determinarea secvenței de operații* – se determină echipamentele necesare controlului și comunicării cu sistemul. La acest pas se desenează o diagramă a operațiilor realizate de sistem.
- *Alocare intrărilor și a ieșirilor* – identificarea I/O (pot fi senzori, relee, motoare, valve, etc). Alocarea adreselor pentru I/O se va face înainte de a scrie programul.
- *Scrierea programului* cu ajutorul “ladder diagram program”.
- *Transferarea programului în memoria AP*.
- *Rularea aplicației folosind AP*. Înainte de a porni aplicația trebuie să ne asigurăm că sunt făcute toate legăturile corect.

Există o multitudine de aplicații unde AP sunt folosite, cum ar fi:

- Manipularea de materiale; mașini de împachetare;
- Sisteme de transport; cabine de lift; parcuri de distracție;
- Pompare de substanțe (controlul echipamentelor de pompare);
- Piscine; tratarea apei;
- Industria alimentară; etc.

Alocarea adreselor pentru AP (pentru AP / PLC OMRON).

**Intrări** – 00.00 – .. (limitat la numărul porturilor de intrare a PLC-ului).

**Ieșiri** – 10.00 – .. (limitat la numărul porturilor de ieșire a PLC-ului).

**Stări** - 200.00 - .. (de câte e nevoie).

### 3.2.2 CX - PROGRAMMER

1. Deschiderea aplicației, figura 3.2 și figura 3.3.
2. Crearea unui nou proiect presupune următorii pași (figura 3.4, figura 3.5, figura 3.6 și figura 3.7):
  - a) Verificare - înainte de a porni noul proiect se cunosc parametrii AP-ului folosit:
    - modelul automatului programabil;
    - tipul procesorului;
    - tipul interfeței de comunicare;
    - asignarea de simboluri pentru variabile.

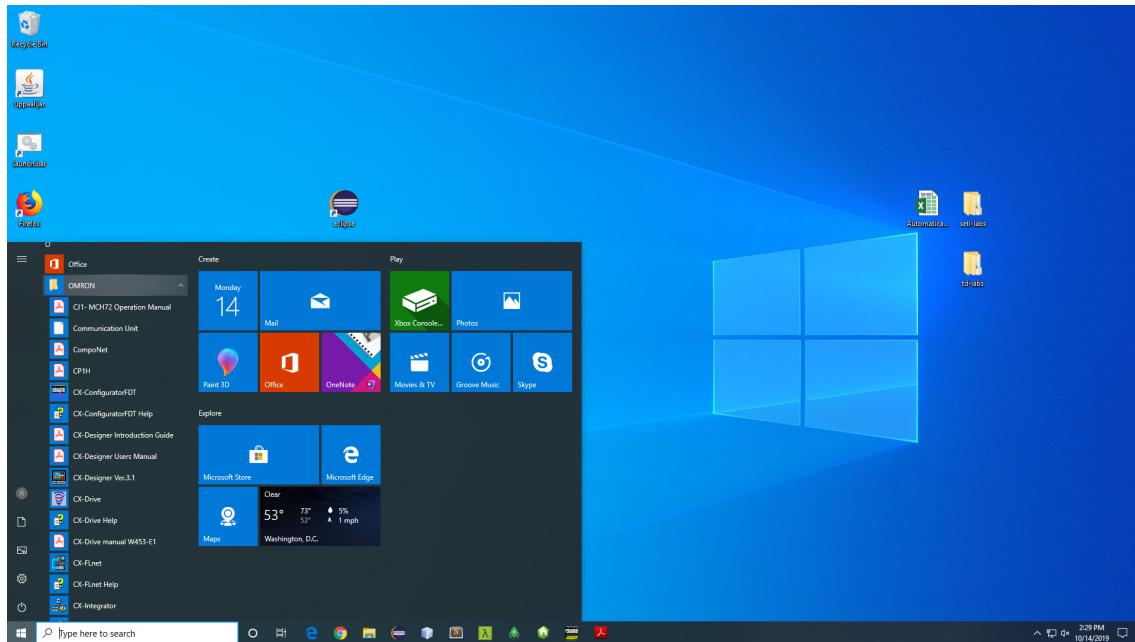


Figure 3.2: Deschiderea aplicației (Selectează OMRON) - pas 1

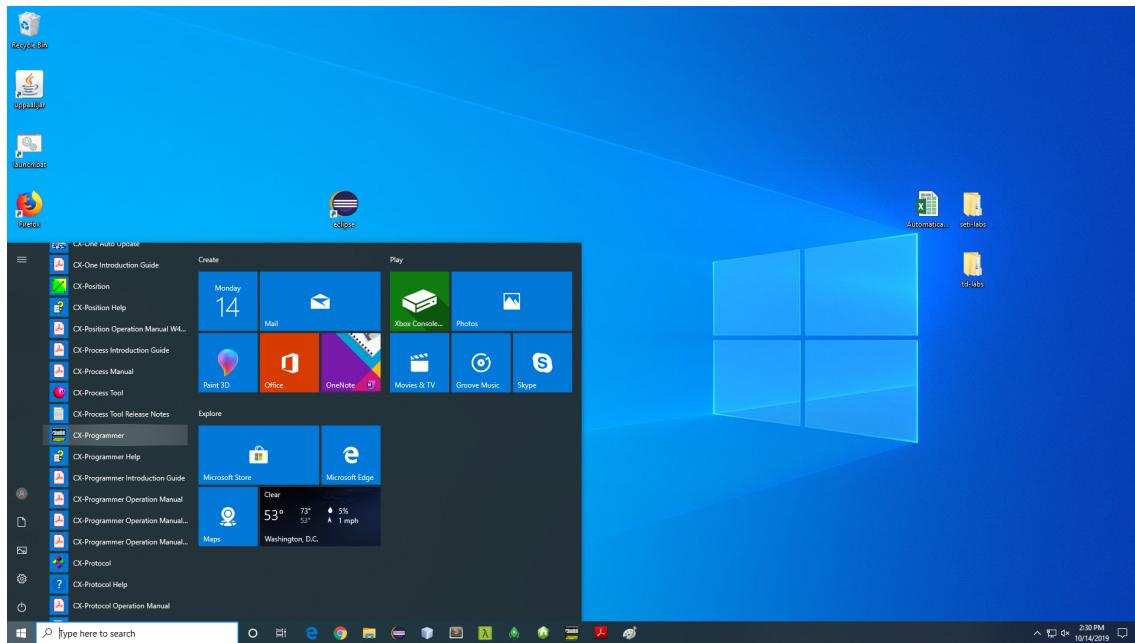


Figure 3.3: Deschiderea aplicației (Selectează CX-programmer) - pas 2

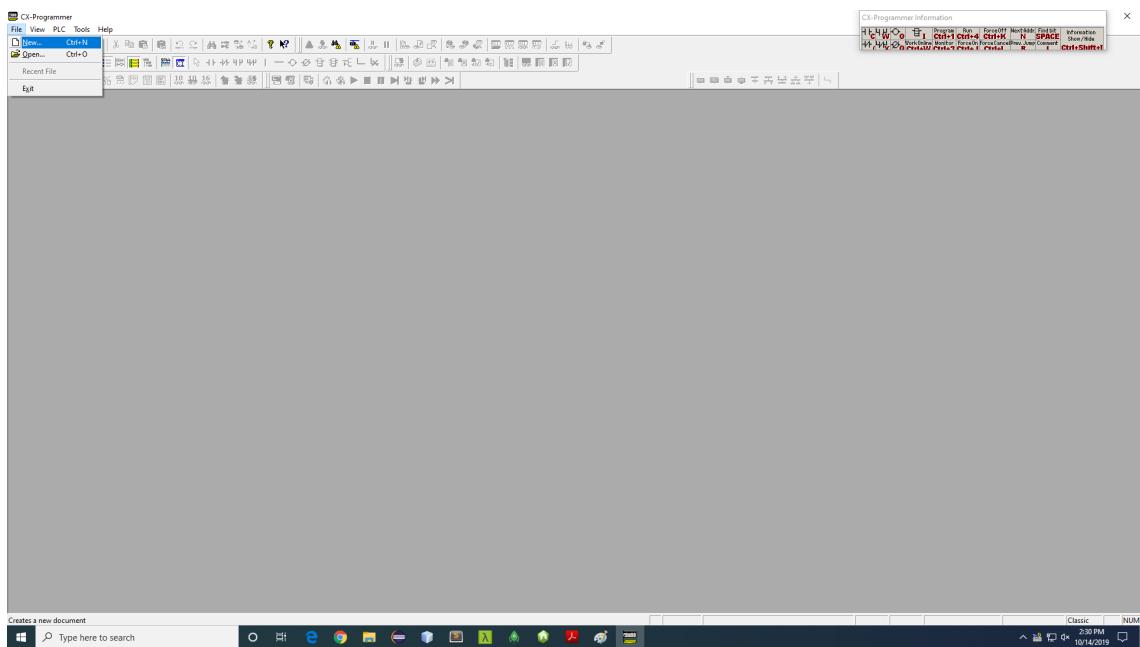


Figure 3.4: Proiect nou

3. Project Workspace, figura 3.8. Obiectele sunt reprezentate într-o structură arborescentă. unde:
  - myPrj[CJ1M] Offline (1) - numele proiectului
  - Symbols (2) – aici se pot declara variabile globale;
  - I/O Table (3) – conține un table cu toate unitățile atașate automatului;
  - Settings (4) – setările automatului programabil;
  - Memory (5) – memoria automatului programabil;
  - Programs (6) - PLC-uri / Programe adăugate în proiectul curent;
  - NewProgram1 (00) (7) - PLC-ul / Programul 1;
  - Symbols (8)– aici se pot declara variabile locale;
4. Definirea de simboluri globale și locale (figura 3.8 punctele 2 și 8), figura 3.9.  
Pasul 1 - Click dreapta (simbol local sau global), pasul 2 - Insert Symbol, pasul 3 (în fereastra nou deschisă) se introduc: Numele simbolului (pc. 3 - *Name*), tipul de date (pc. 4 - *Data type*), adresa (pc. 5 - *Address or value*) și comentarii (pc. 6 - *Comment*); confirmarea datelor introduse se face la apăsarea butonului *ok* de la pc. 7.
5. Foaia de lucru, figura 3.10.

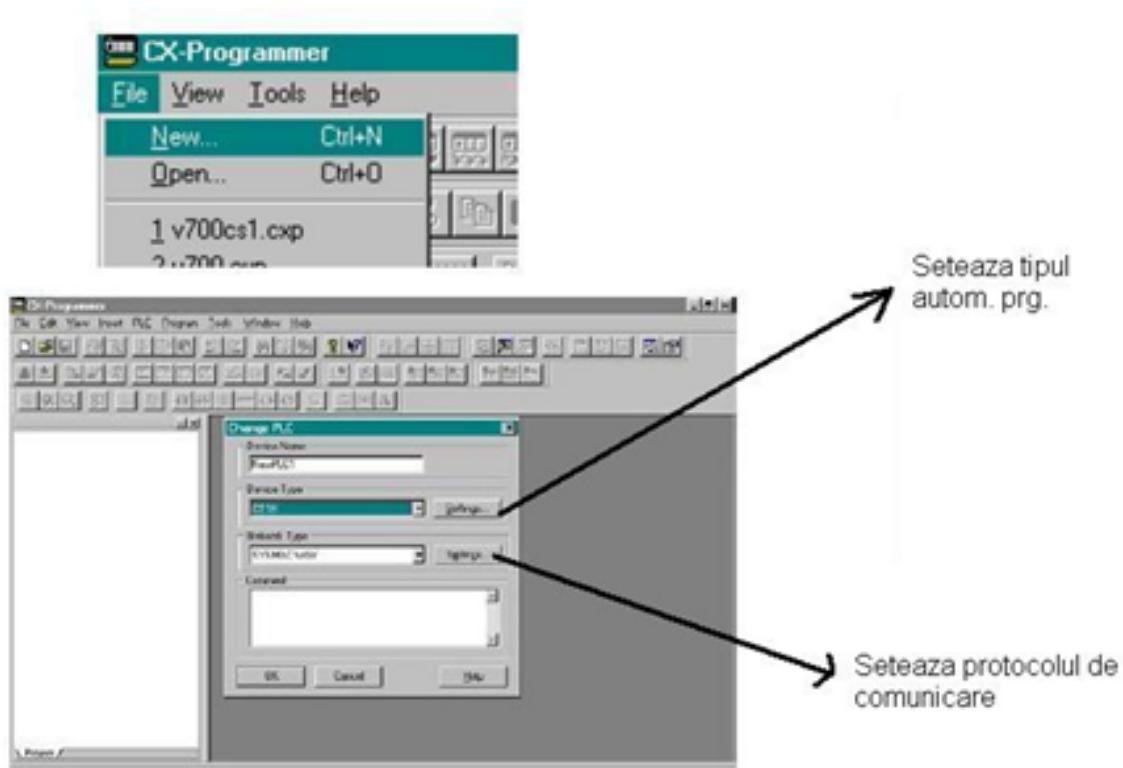


Figure 3.5: Configurarea noului proiect

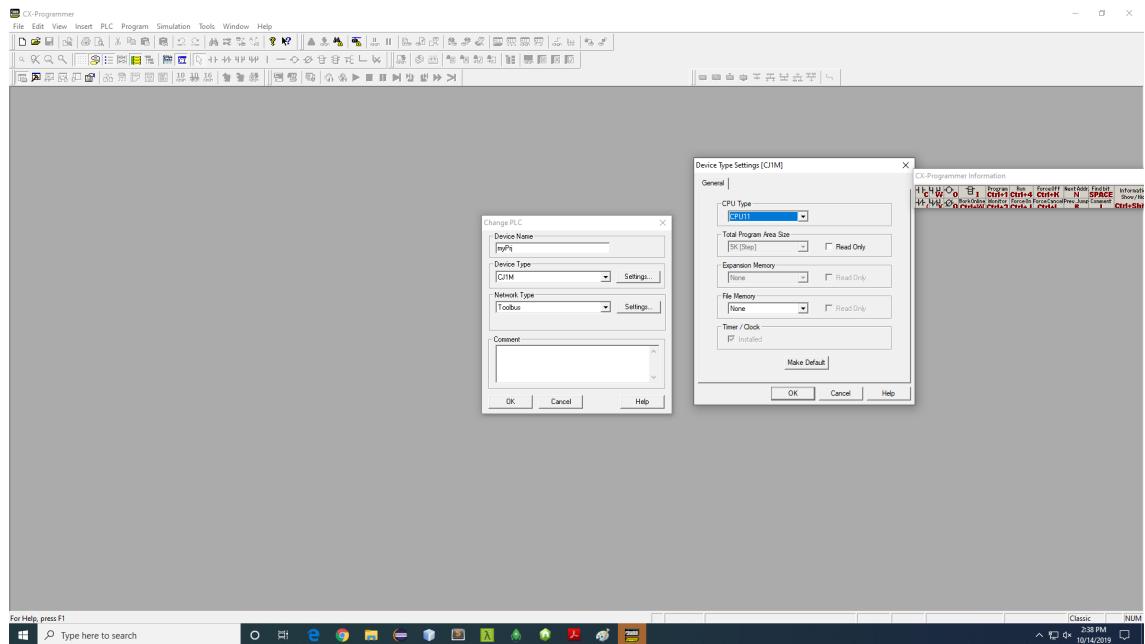


Figure 3.6: Proiect nou - modelul PLC-ului

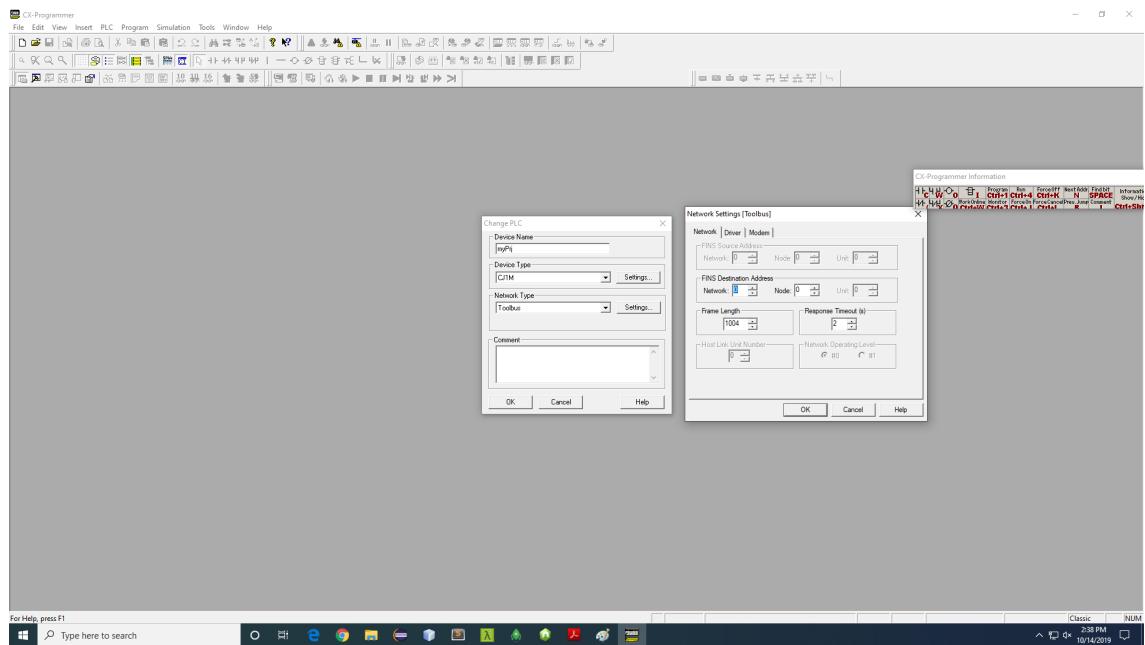


Figure 3.7: Proiect nou - protocolul de comunicație

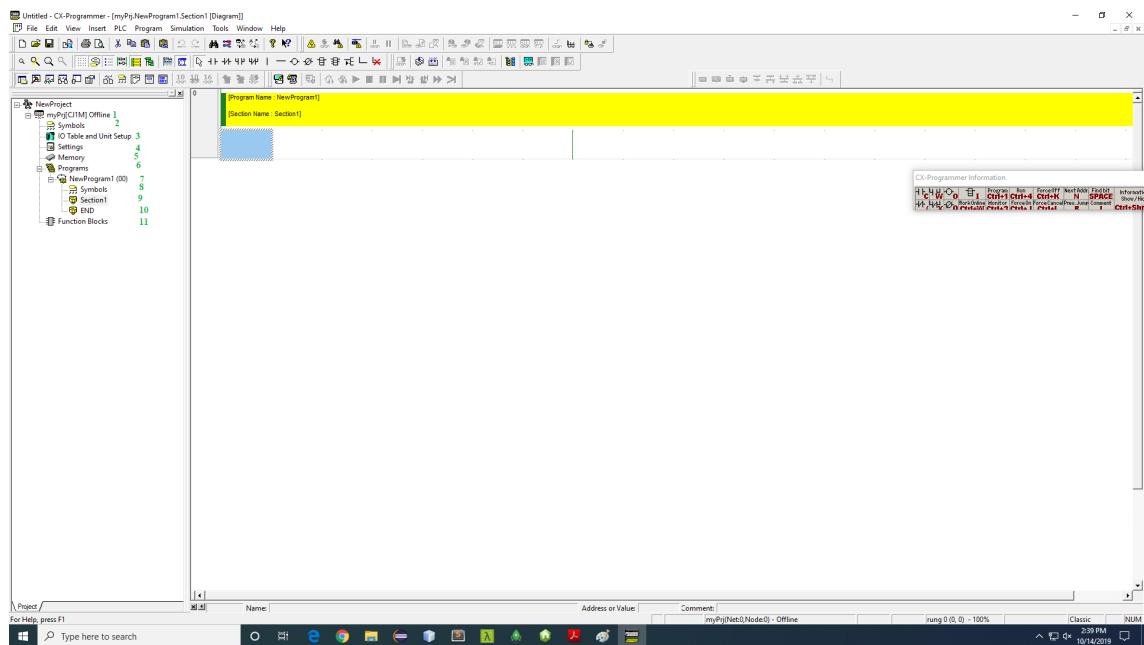


Figure 3.8: Project Workspace

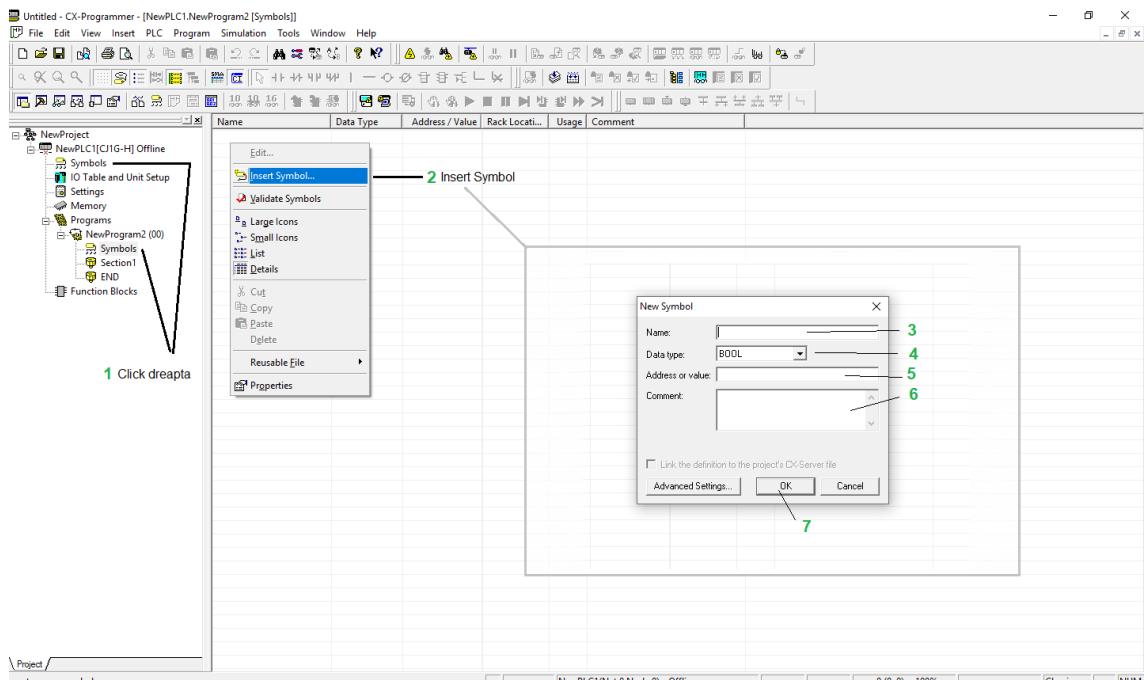


Figure 3.9: Definirea simbolurilor - exemplu

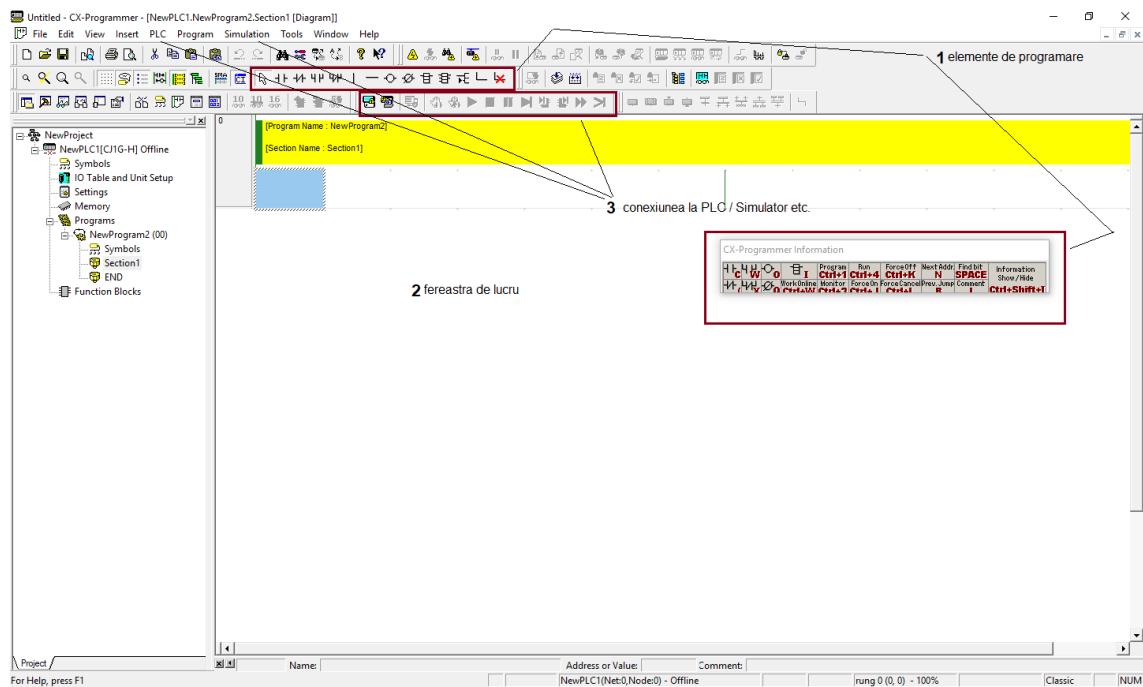


Figure 3.10: Foaia de lucru - exemplu

## 6. Linie de cod folosind Ladder Logic, figura 3.11.

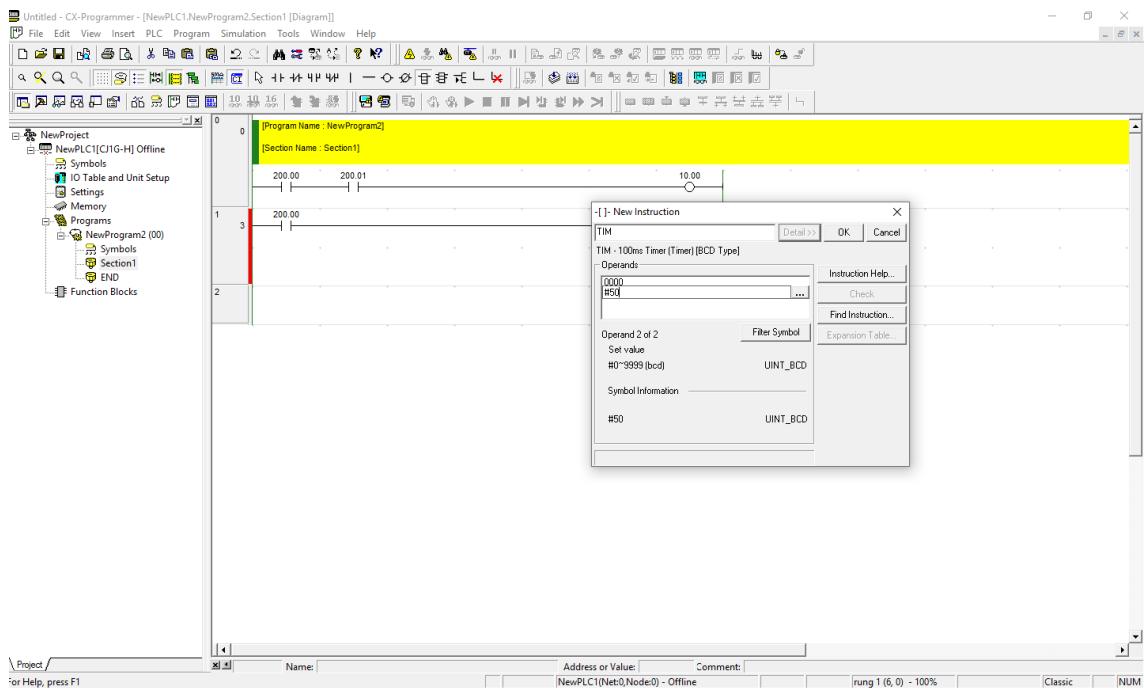


Figure 3.11: Linie de cod Ladder Logic - exemplu

## 3.3 DESFĂȘURAREA LUCRĂRII

### 3.3.1 PROBLEME REZOLVATE

1. Avem o barieră care stă ridicată 10 s și coborâtă 5 secunde. Se va realiza programul în Ladder Logic pentru automatul programabil OMRON CJ1M.

Pentru început se vor defini simbolurile necesare, figura 3.12 (în acest program sunt folosite simboluri locale). Aceste simboluri sunt recunoscute doar în interiorul acestui proiect.

Pentru a introduce un nou simbol, click dreapta și selectați "Insert new symbol". (figura 3.13, figura 3.14). Pentru un nou simbol sunt necesare câmpurile: nume, adresă și type.

Editarea programului Ladder Logic, figura 3.15

Configurarea unui clock (tranzitie temporizată), figura 3.16.

Definire stare inițială, figura 3.17.

Programul în ladder logic, figura 3.18.

Finalul programului Ladder Logic, figura 3.19.

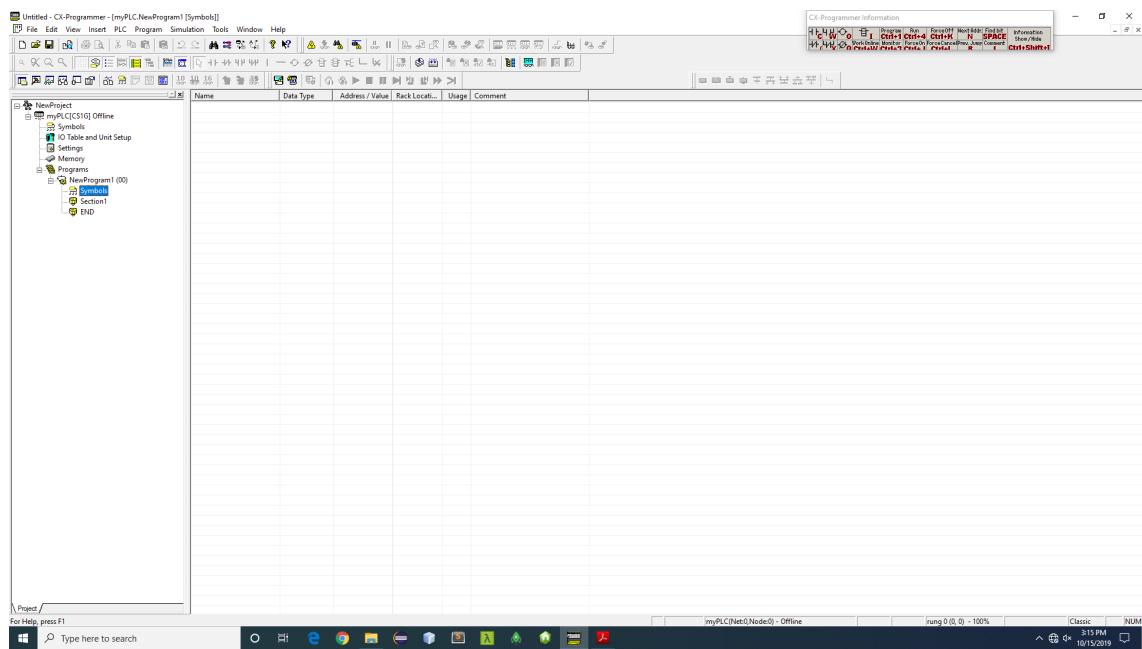


Figure 3.12: Simboluri locale

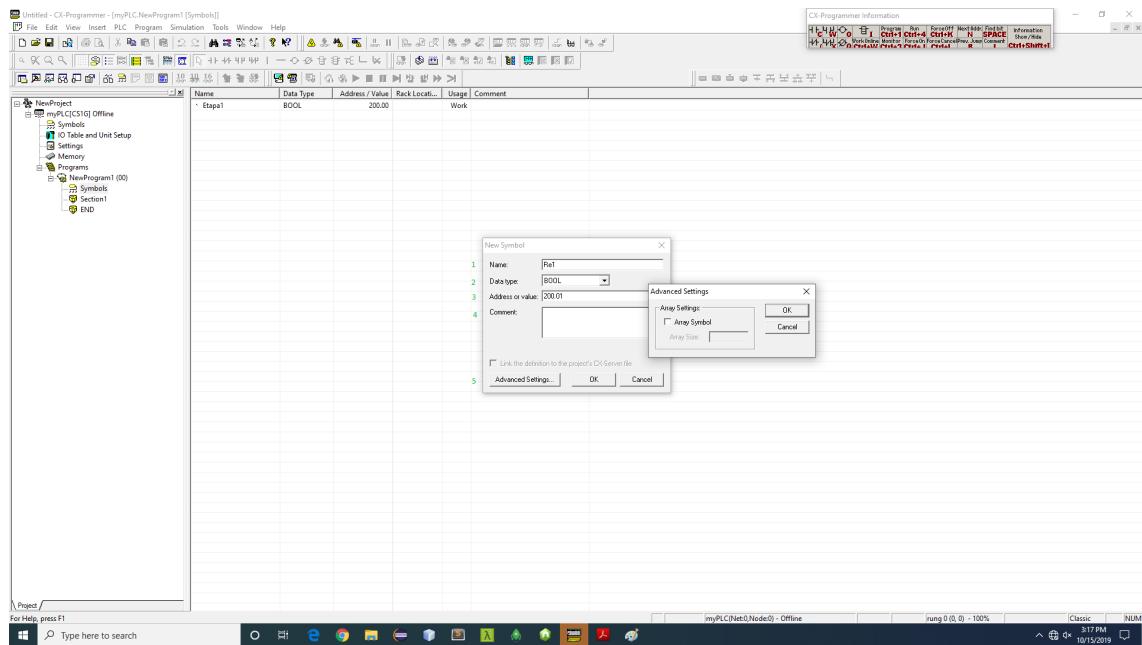


Figure 3.13: Introducerea unui nou simbol

## SED - Îndrumător de laborator

# Dezvoltarea aplicațiilor folosind ladder logic

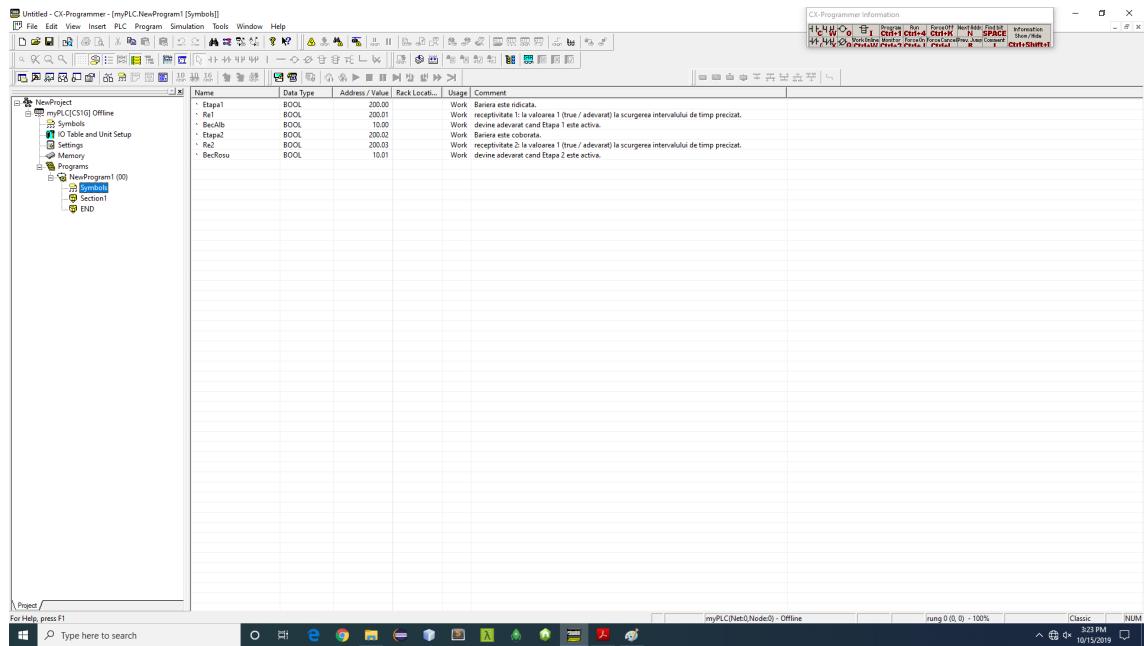


Figure 3.14: Lista simbolurilor necesare

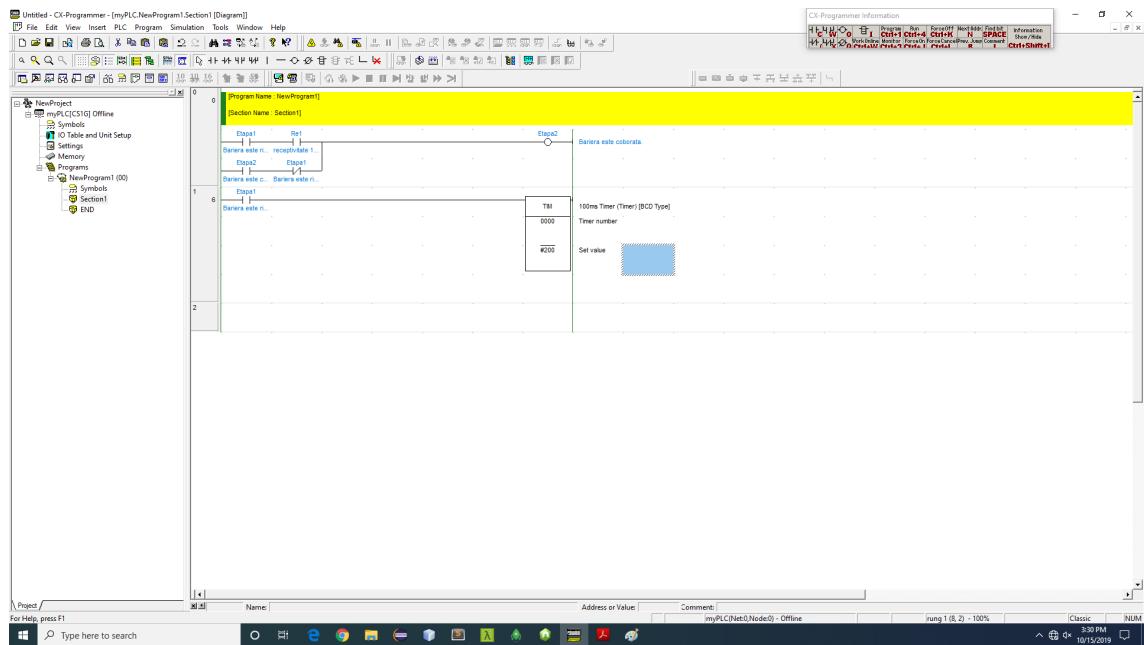


Figure 3.15: Secvențe de cod Ladder Logic

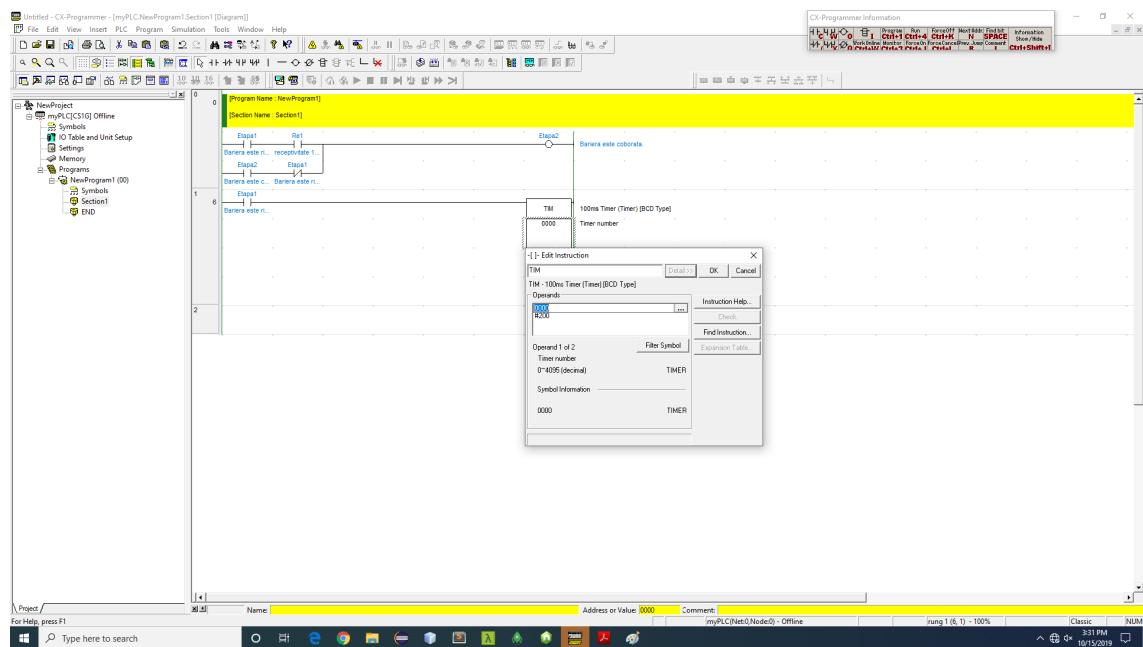


Figure 3.16: Configurare Clock

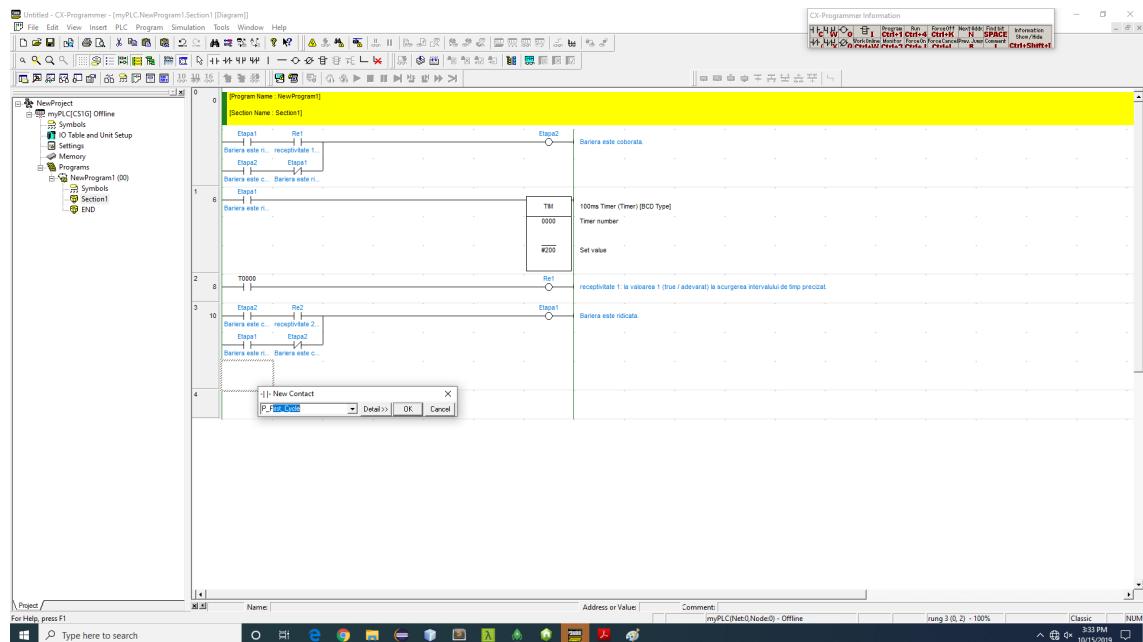


Figure 3.17: Stare inițială

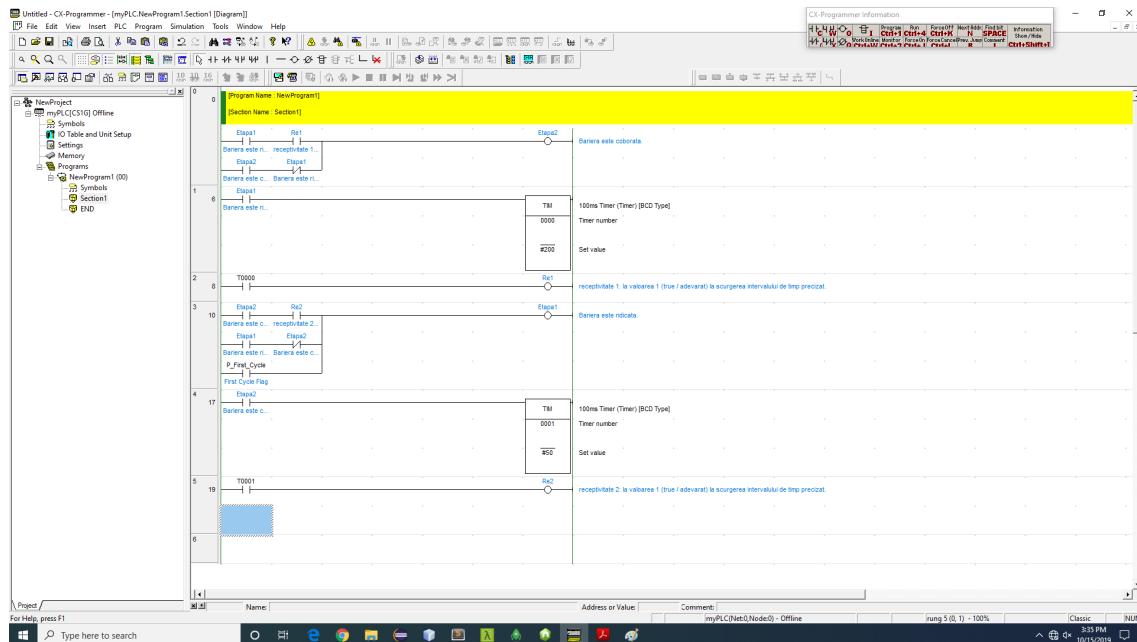


Figure 3.18: Programul Ladder Logic

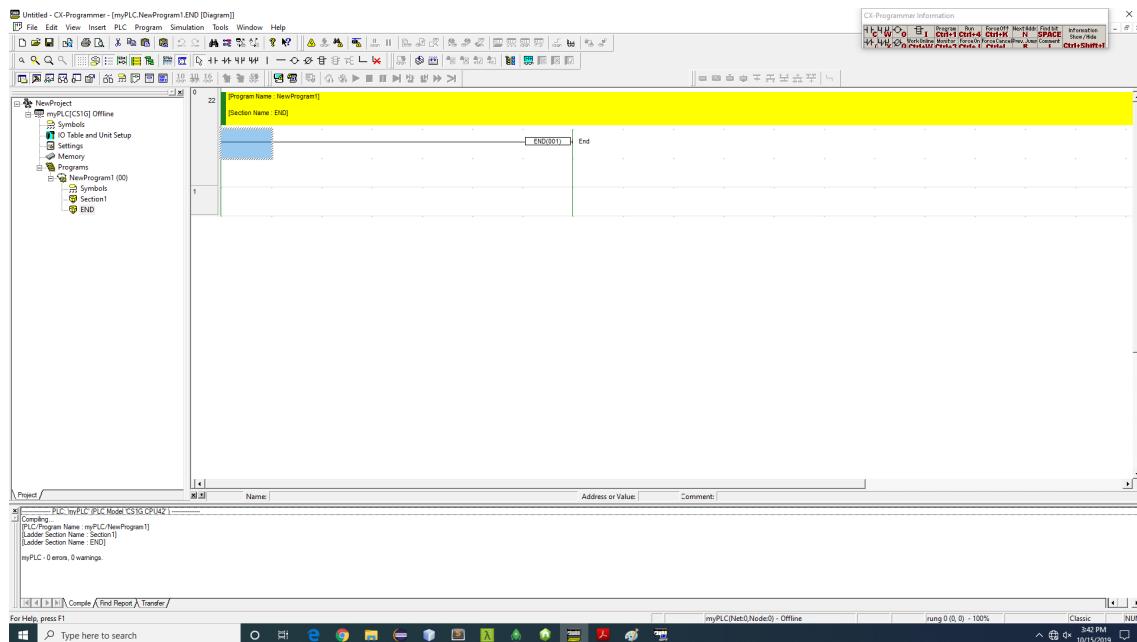


Figure 3.19: Sfârșit program

Programul scris se poate testa folosind simulatorul, figura 3.20, figura 3.21 și figura 3.22.

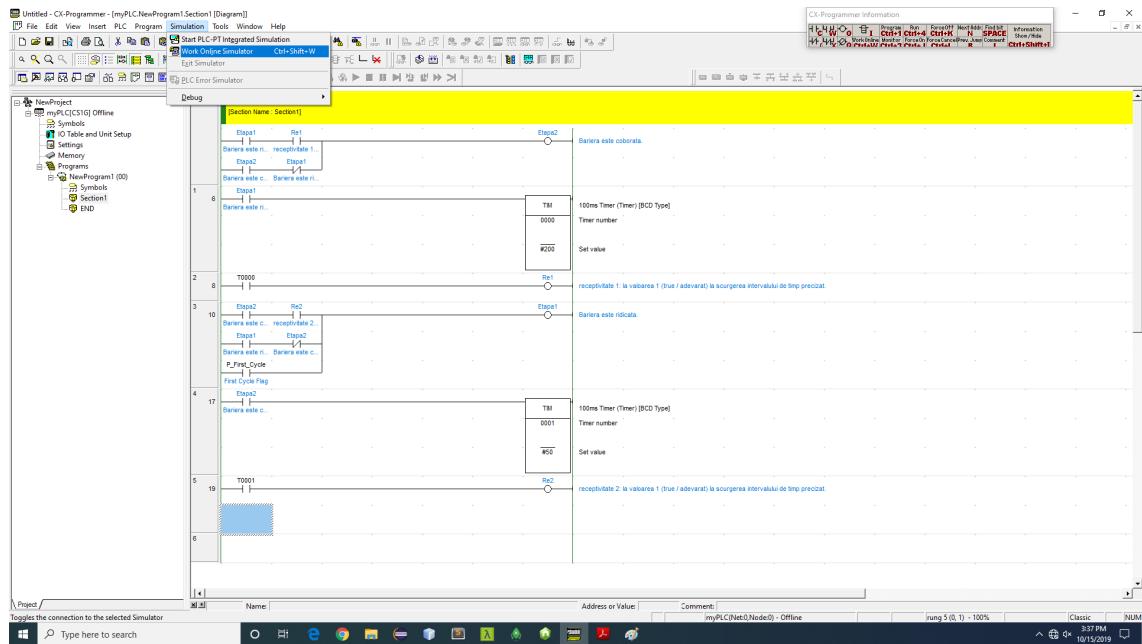


Figure 3.20: Simulare - Work online Simulator - opțiune din meniu Simulare

### 3.3.2 PROBLEME PROPUSE

1. Se consideră o intersecție de patru străzi, există două semafoare pe fiecare sens de deplasare și câte un semafor pentru pietoni. Construiți algoritmul pentru controlul semafoarelor din intersecție. Să se realizeze programul în Ladder Logic.
2. Să se controleze un lift care poate să transporte până la 10 persoane într-o clădire cu 4 etaje.

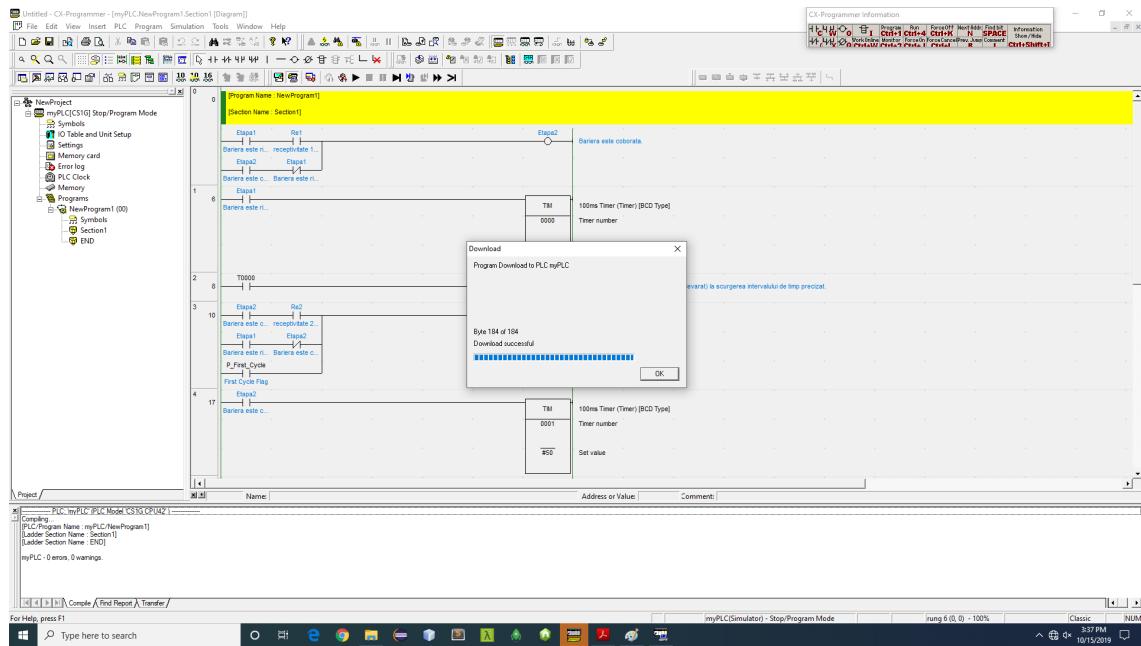


Figure 3.21: Simulare - Transfer cu succes programului

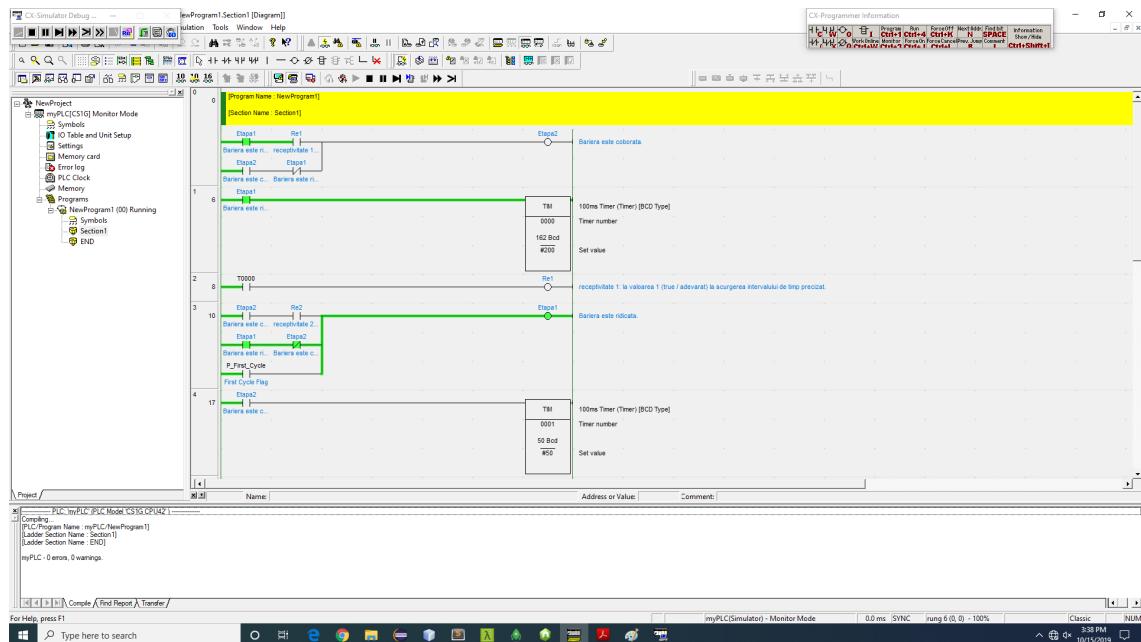


Figure 3.22: Simulare - secvență de funcționare

# BIBLIOGRAFIE

- [1] OMRON. *Cx-Programmer. Operation manual* [https://assets.omron.eu/downloads/manual/en/v1/r132\\_cx-programmer\\_fb\\_library\\_getting-started\\_guide\\_en.pdf](https://assets.omron.eu/downloads/manual/en/v1/r132_cx-programmer_fb_library_getting-started_guide_en.pdf)

# 4 DEZVOLTAREA APLICAȚIILOR FOLOSIND SFC (SEQUENTIAL FUNCTION CHART)

---

## 4.1 OBIECTIVE



- Dezvoltarea de programe folosind SFC.

## 4.2 CONSIDERAȚII TEORETICE

Sequential Function Chart (SFC) este un limbaj grafic de programare care permite controlul secvențelor de proces prin descrierea tranzițiilor și acțiunilor pentru fiecare etapă. În continuare se va folosi aplicația de la OMRON Cx-programmer din familia Cx-One.

### 4.2.1 CX - PROGRAMMER

Scrierea primului program folosind Cx-programmer în SFC.

- Deschiderea aplicației, selectați OMRON: figura 4.1 și apoi, Cx-Programmer: figura 4.2.
- Creearea unui nou proiect (*File, submeniu New*), figura 4.3.
- Introducerea parametrilor noului proiect, figura 4.4 și figura 4.5.
  1. Verificare - înainte de a porni noul proiect trebuie să știți următoarele:
    - parametrii automatului programabil folosit;
    - modelul automatului programabil;
    - tipul procesorului;

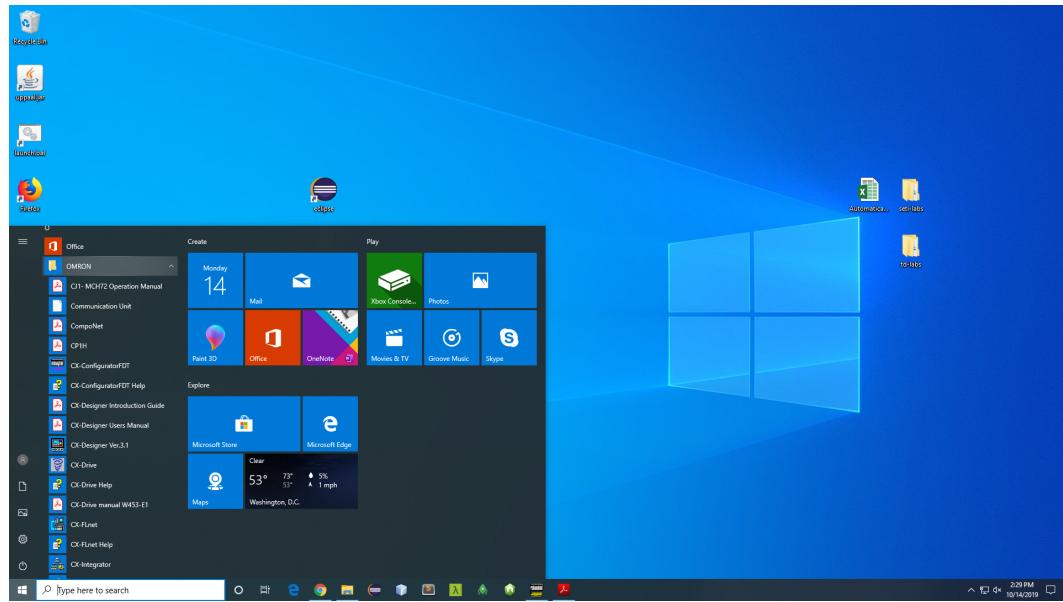


Figure 4.1: OMRON

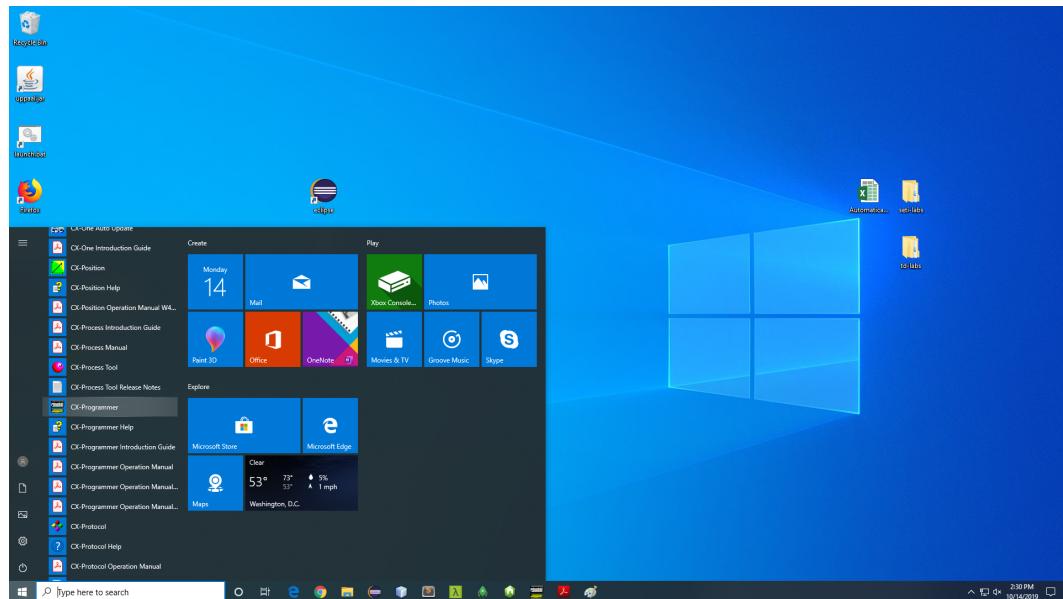


Figure 4.2: Cx-programmer

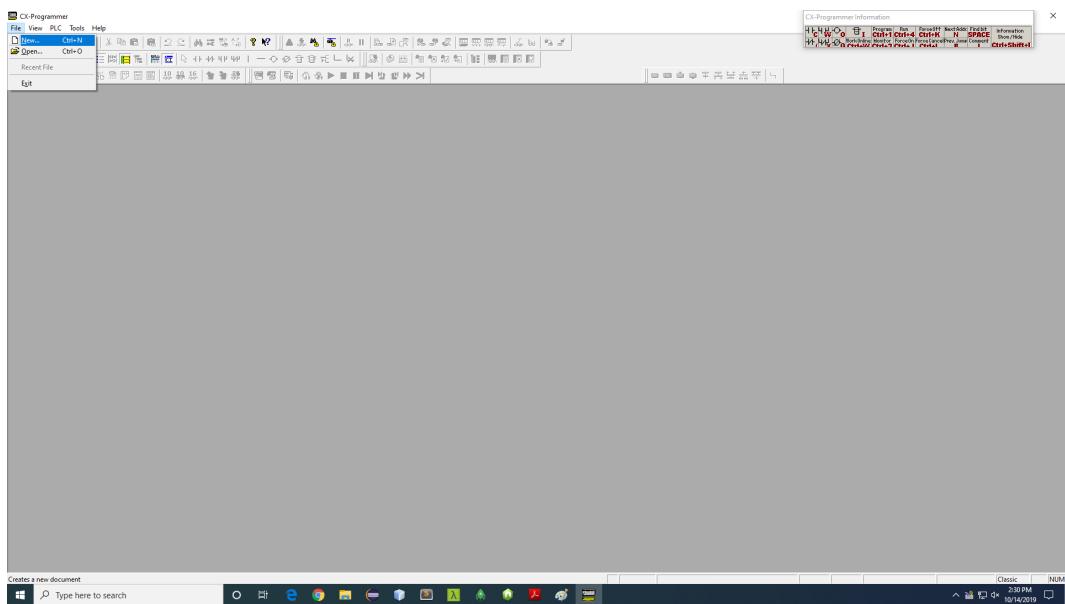


Figure 4.3: Creearea unui proiect nou

- tipul interfetei de comunicare;
- asignarea de simboluri pentru variabile.

Mai departe vom folosi:

- PLC type: CJ1G-H
- CPU type: CPU42-H
- Network type: Toolbus
- Conexiunea cu PLC-ul (PLC-ul trebuie conectat fizic la PC), figura 4.6.
- Conexiunea cu Simulatorul (dacă PLC-ul nu este conectat fizic la PC se poate folosi simulatorul), figura 4.7. Simulatorul (cx-simulator /Cx-one) are mai multe familii de PLC-uri accesibile. Conexiunea între cx-programmer și cx-simulator se face automat de către cx-server (toate aplicații software sub umbrela cx-One).
- Project Workspace, figura 4.8. Obiectele sunt reprezentate într-o structură arborescentă. Unde:
  - myPrj[CJ1M] Offline (1) - numele proiectului
  - Symbols (2) – aici se pot declara variabile globale;
  - I/O Table (3) – conține un tabel cu toate unitățile atașate automatului;

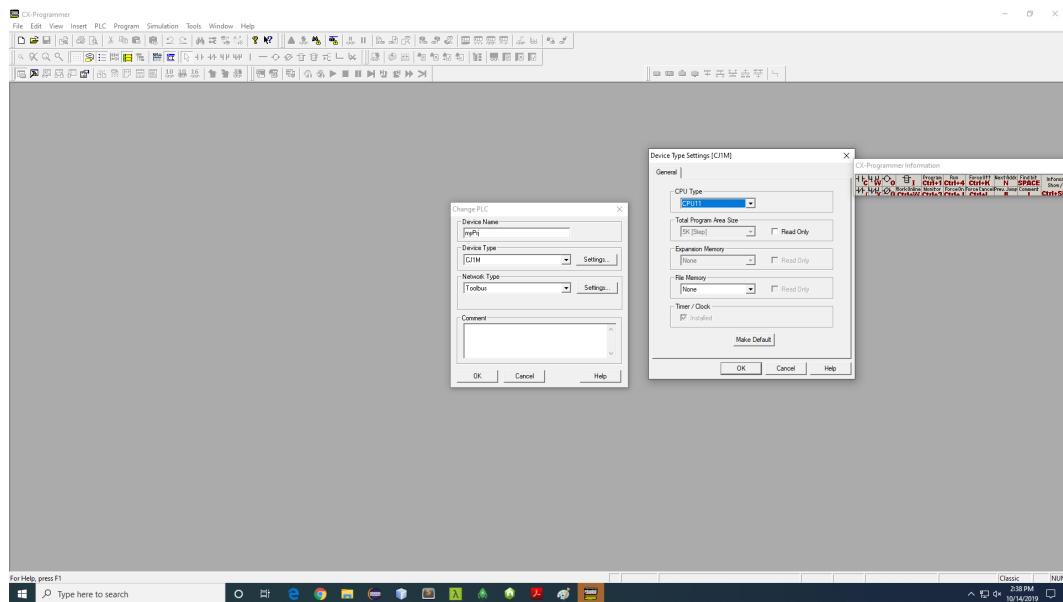


Figure 4.4: Configurarea unui nou proiect, pas 1

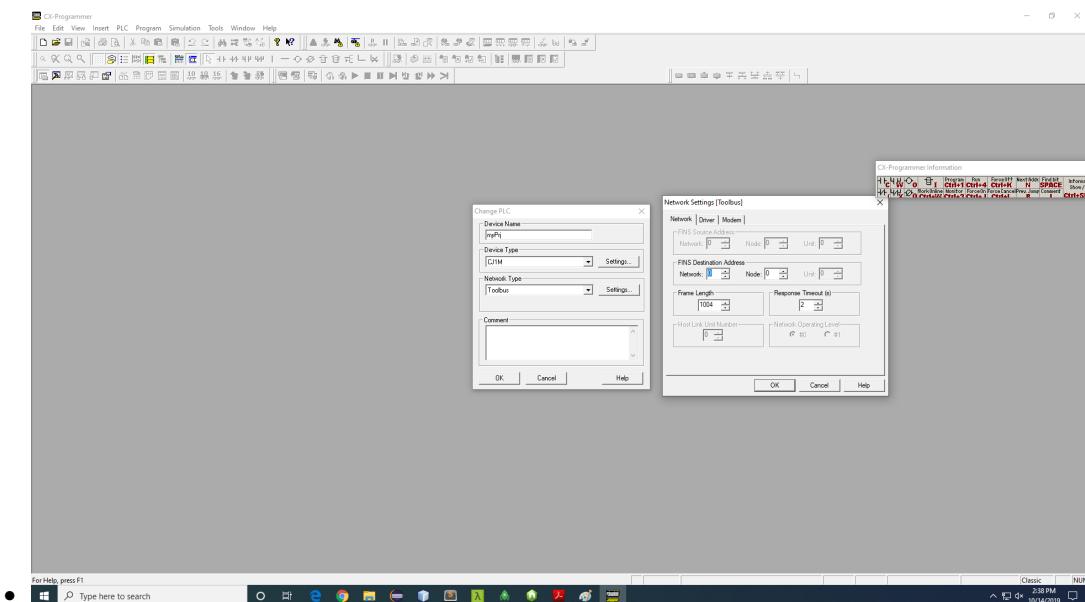


Figure 4.5: Configurarea unui nou proiect, pas 2

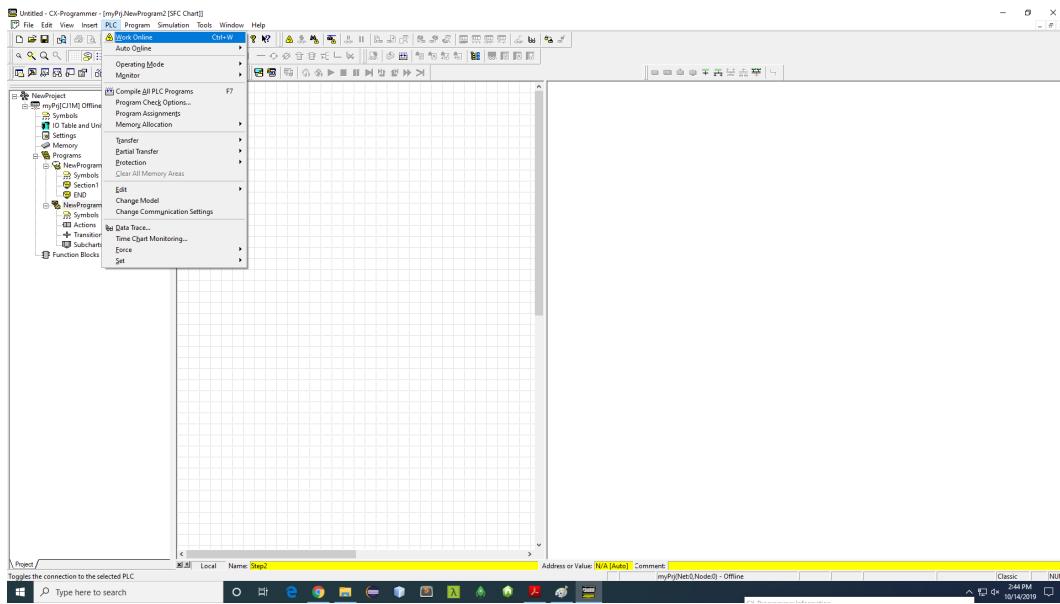


Figure 4.6: Conexiunea cu PLC-ul

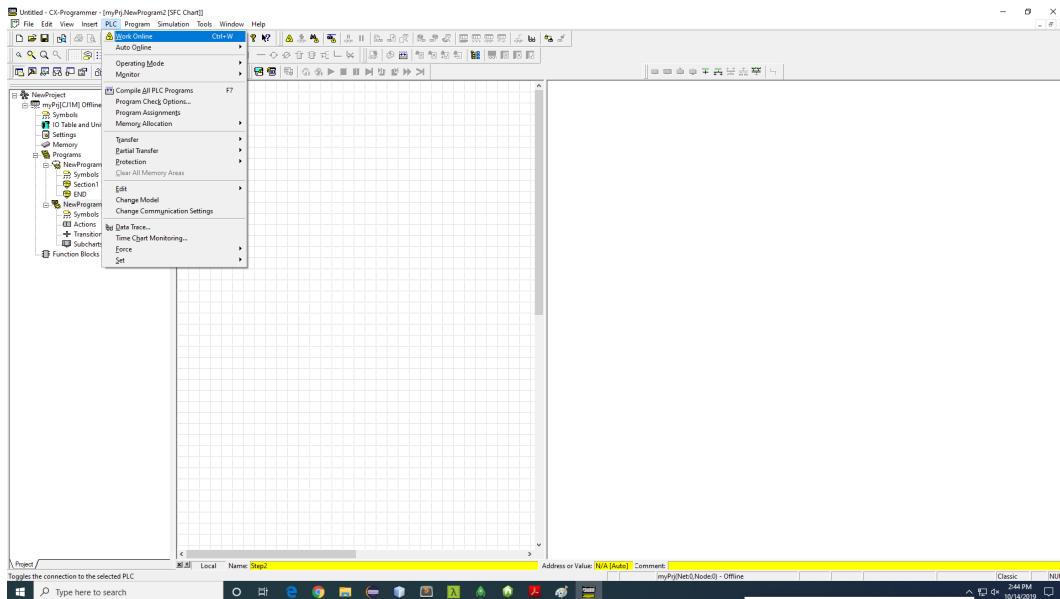


Figure 4.7: Conexiunea cu Simulatorul

- Settings (4) – setările automatului programabil;
- Memory (5) – memoria automatului programabil;
- Programs (6) - PLC-uri / Programe adăugate în proiectul curent;
- NewProgram1 (00) (7) - PLC-ul / Programul 1;
- Symbols (8) – aici se pot declara variabile locale;

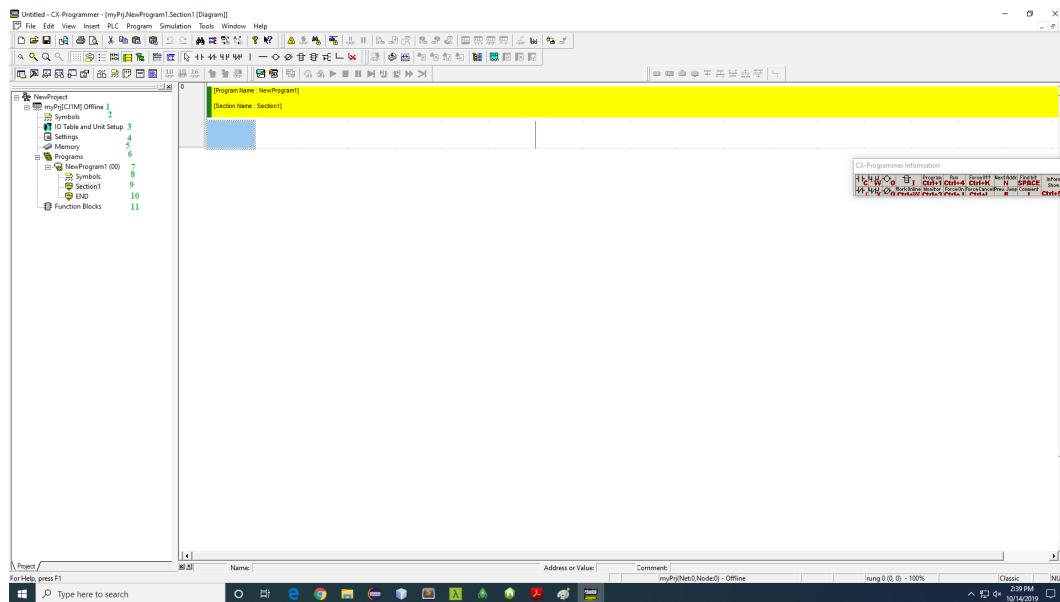


Figure 4.8: Workspace-ul cx-programmer

- Programul LadderLogic nu este folosit în acest exemplu. Stergerea programului ladder logic se face astfel, *click dreapta "New Program 1 (00)" și apoi se sectează "Delete" din meniul pop-up*, figura 4.9 și figura 4.10.
- Introducerea unui program nou de tip SFC, figura 4.11.
- Redenumirea programului nou se face astfel, *click dreapta pe "New Program 2 (00)"*, figura 4.12.
- Definirea limbajului default se face astfel, figura 4.13 (Meniul Tools, Submeniu Options, tabul PLC).
- Adăugarea unei etape, modificarea numelui unei etape, redenumirea etapei inițiale: figura 4.14, figura 4.15, figura 4.16, figura 4.17.

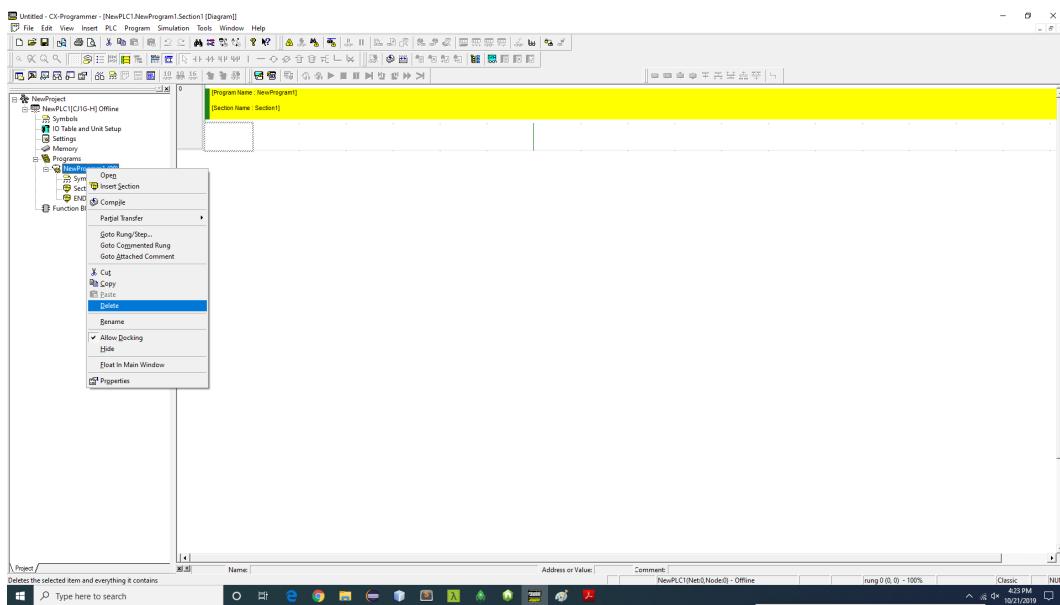


Figure 4.9: Ștergerea programului Ladder Logic, (i)

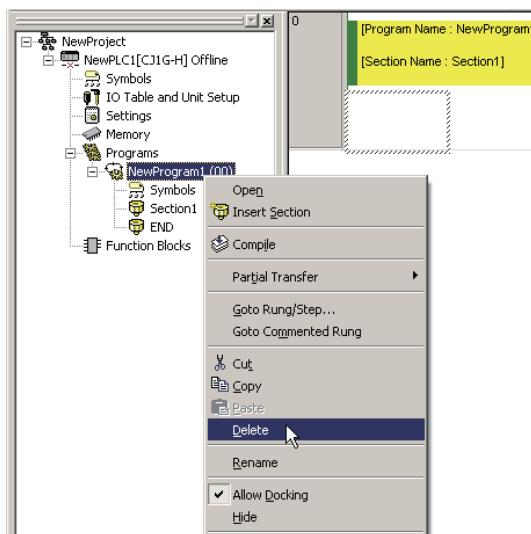


Figure 4.10: Ștergerea programului Ladder Logic, (ii)

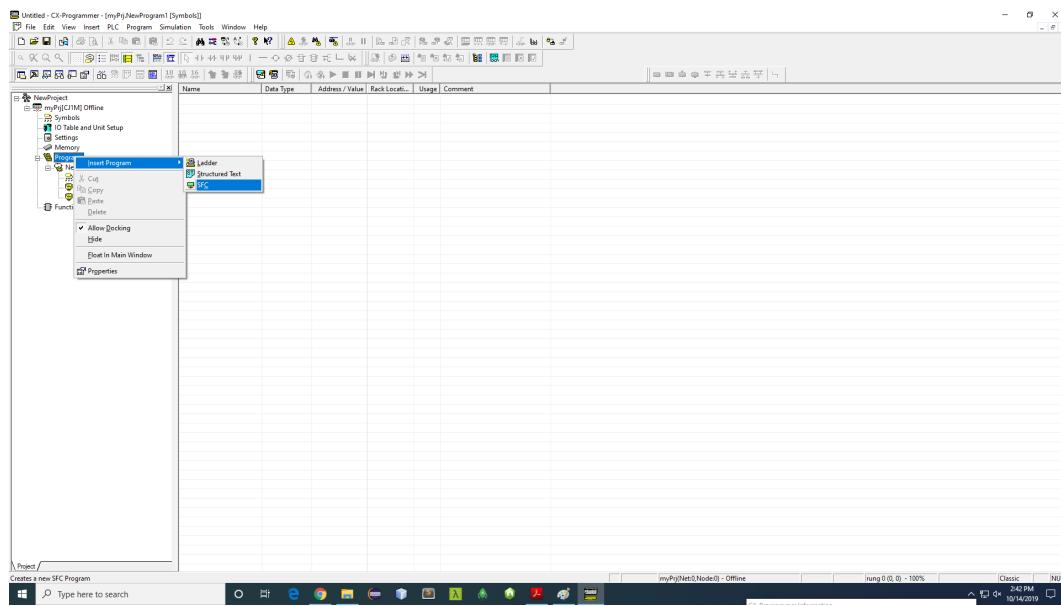


Figure 4.11: Program nou de tip SFC

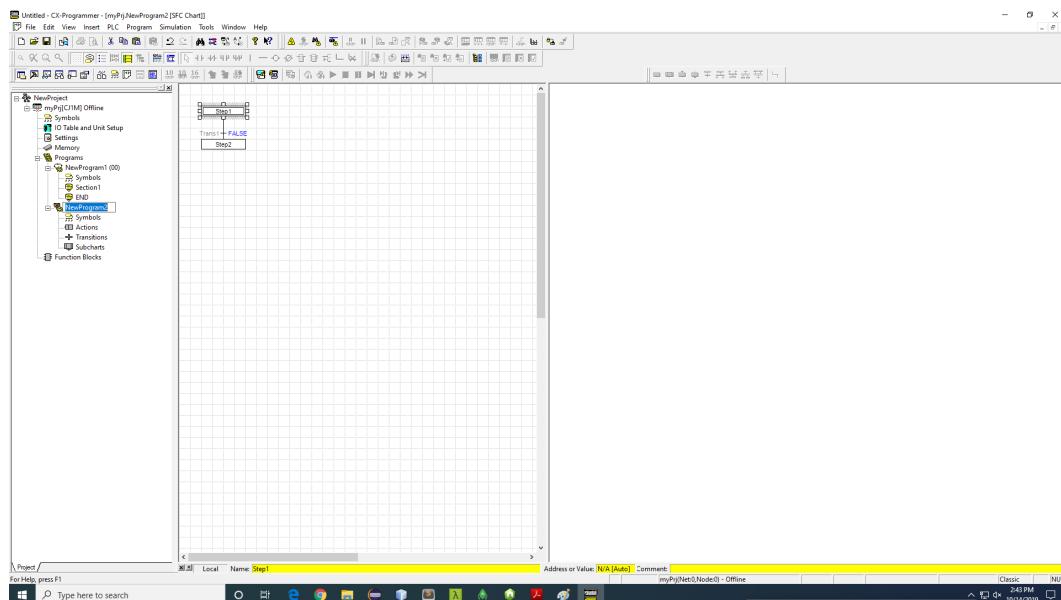


Figure 4.12: Redenumirea unui program SFC

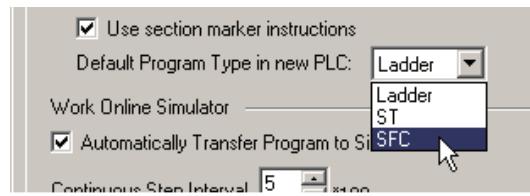


Figure 4.13: Definirea limbajului default de programare

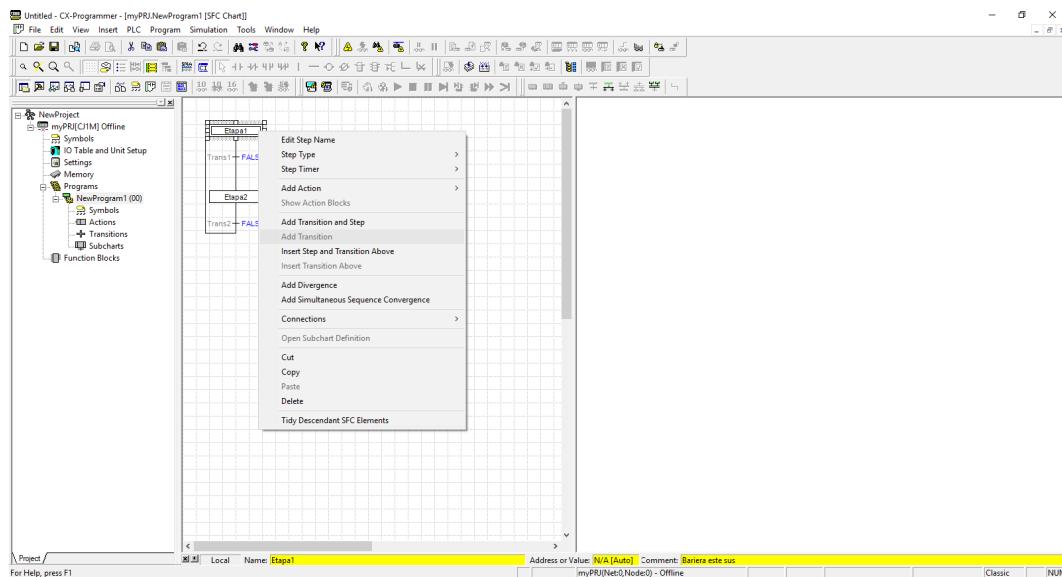


Figure 4.14: Opțiuni disponibile pentru etape

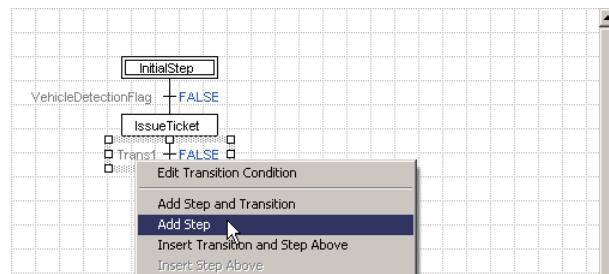


Figure 4.15: Adăugarea unei etape noi

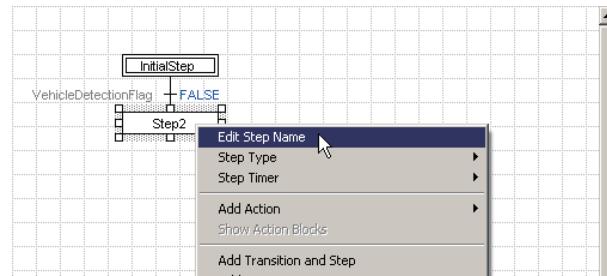


Figure 4.16: Redenumirea unei etape

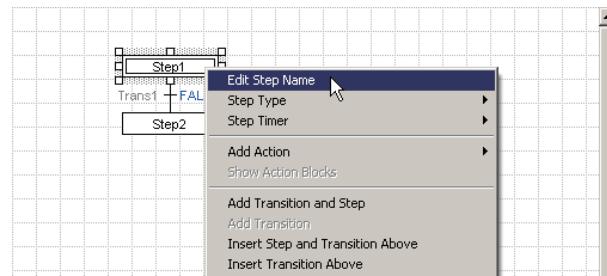


Figure 4.17: Redenumirea etapei inițiale

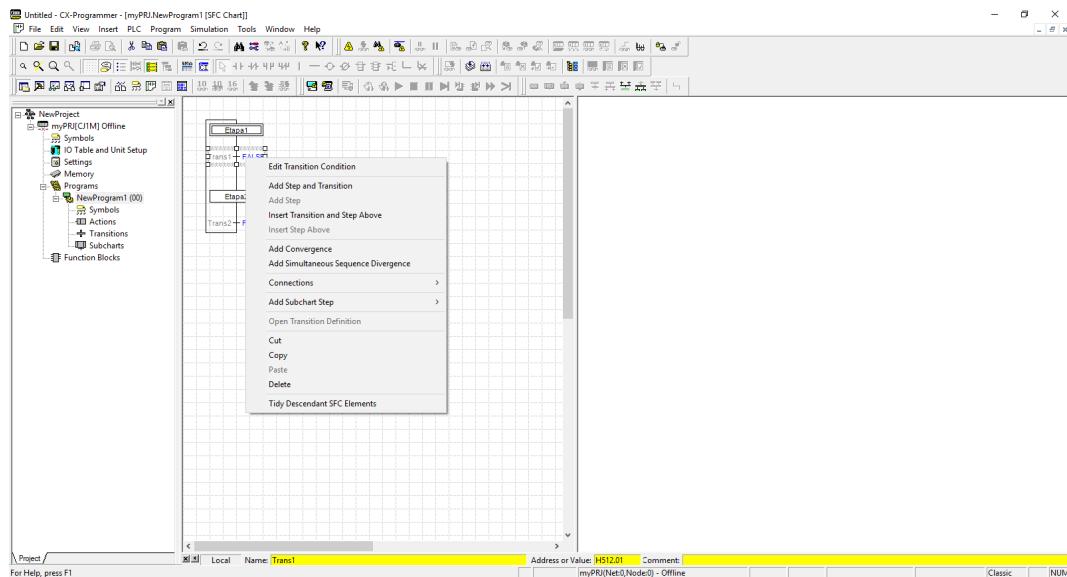


Figure 4.18: Opțiuni disponibile pentru tranzitii

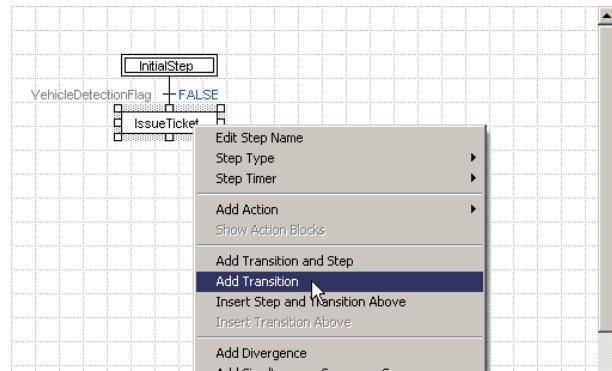


Figure 4.19: Adăugarea unei tranziții noi

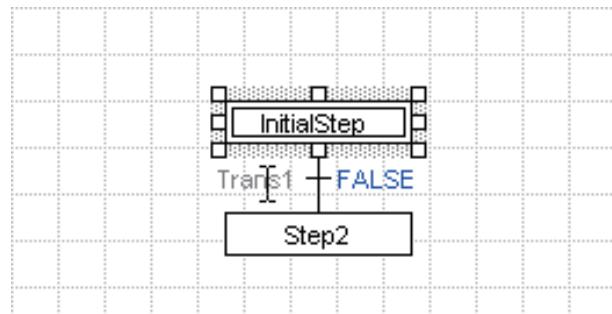


Figure 4.20: Redenumirea unei tranziții

- Adăugarea unei tranziții, modificarea numelui unei tranziții: figura 4.18, figura 4.19, figura 4.20.
- Înregistrarea unei tranziții noi, figura 4.21, figura 4.22. 4.21.



Figure 4.21: Înregistrarea unei tranziții noi (i)

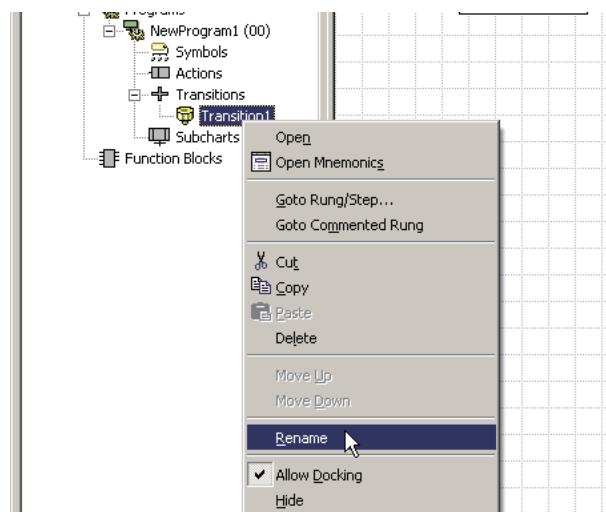


Figure 4.22: Înregistrarea unei tranziții noi (ii)

- Adăugarea legăturilor între noduri, figura 4.23.
- Elemente grafice disponibile (etape, tranziții, arce), figura 4.24.

## 4.3 DESFĂȘURAREA LUCRĂRII

### 4.3.1 PROBLEME REZOLVATE

Exemplu de implementare a unui proiect SFC:

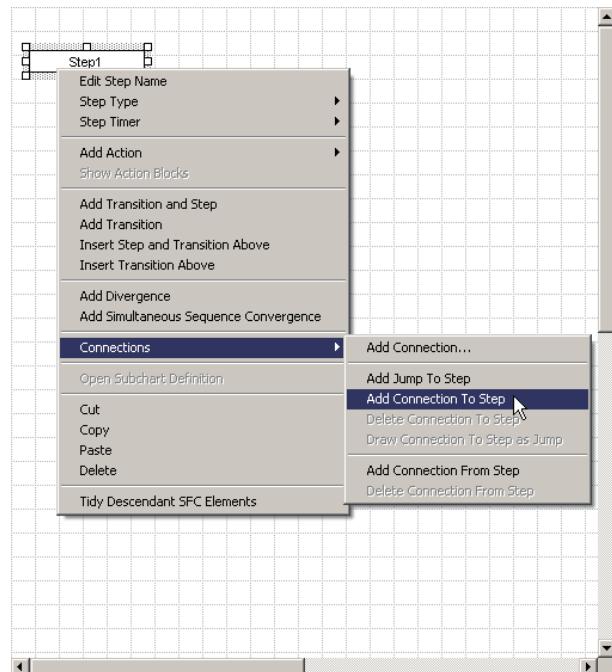


Figure 4.23: Adăugarea legăturilor între noduri - arce

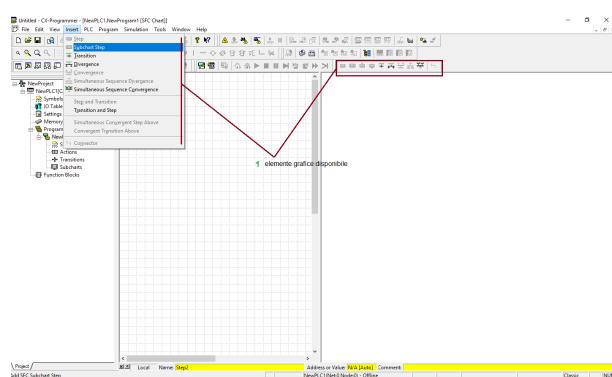


Figure 4.24: Elemente grafice disponibile

Se consideră o barieră care este acționată cu ajutorul unor senzori.

Se va realiza programul SFC în mediul de dezvoltare OMRON / Cx-programmer. Configurarea PLC-ului folosit:

- PLC type: CJ1G-H
- CPU type: CPU42-H
- Network type: Toolbus

Modelul proiectului, figura 4.25.

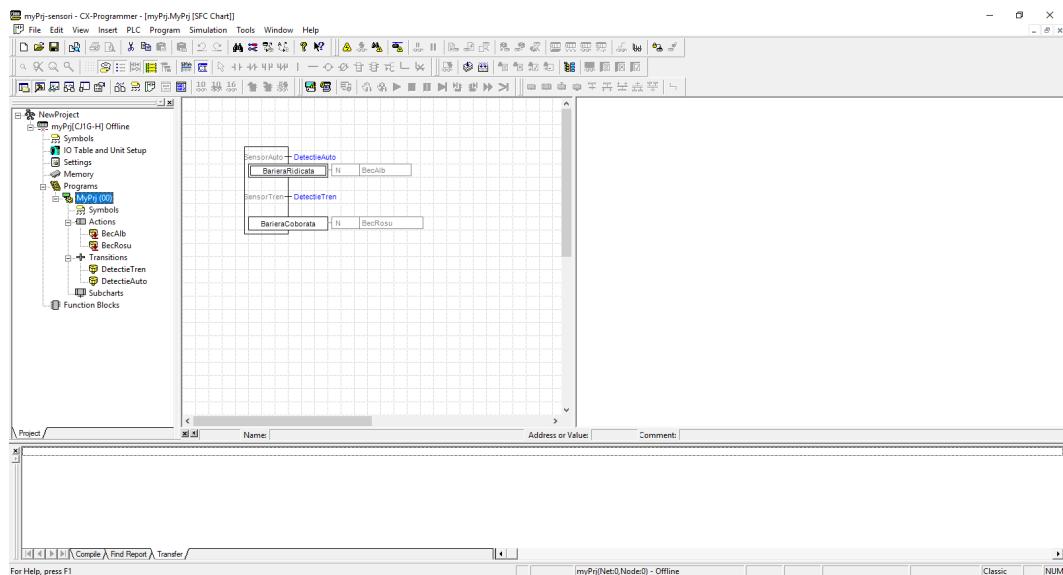


Figure 4.25: Programul SFC

Lista de simboluri, figura 4.26.

Adăugarea acțiunilor (click dreapta pe etapa la care se adaugă acțiunea și se alege *Add Action* apoi *New Ladder Action*), figura 4.27, figura 4.28 și figura 4.29.

Adăugarea tranzițiilor (Click dreapta *Transitions*, selectează *Insert Transition* apoi *Ladder*): figura 4.30, figura 4.31, figura 4.32 și figura 4.33.

#### 4.3.2 PROBLEME PROPUSE

1. Se ia în discuție un proces chimic care combină două substanțe, prezentat în figura 4.34. Să se realizeze modelul GRAFCET și programul SFC al procesului.

Se introduce substanța 1, prin deschiderea robinetului  $V_1$ , până la nivelul  $L_1$  apoi se introduce substanța 2 prin deschiderea robinetului  $V_2$  în timp ce se amestecă

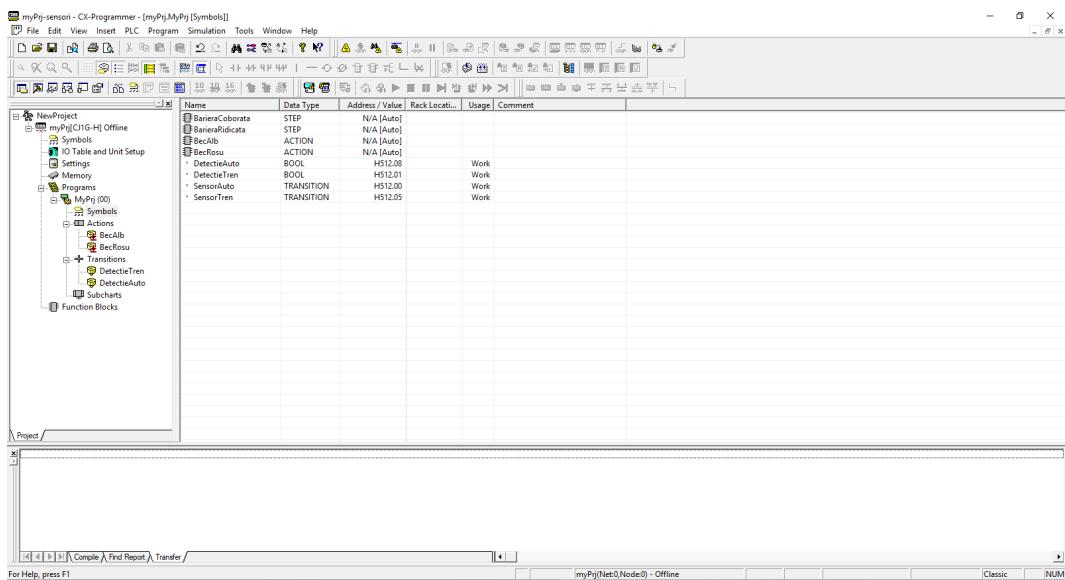


Figure 4.26: Lista de simboluri

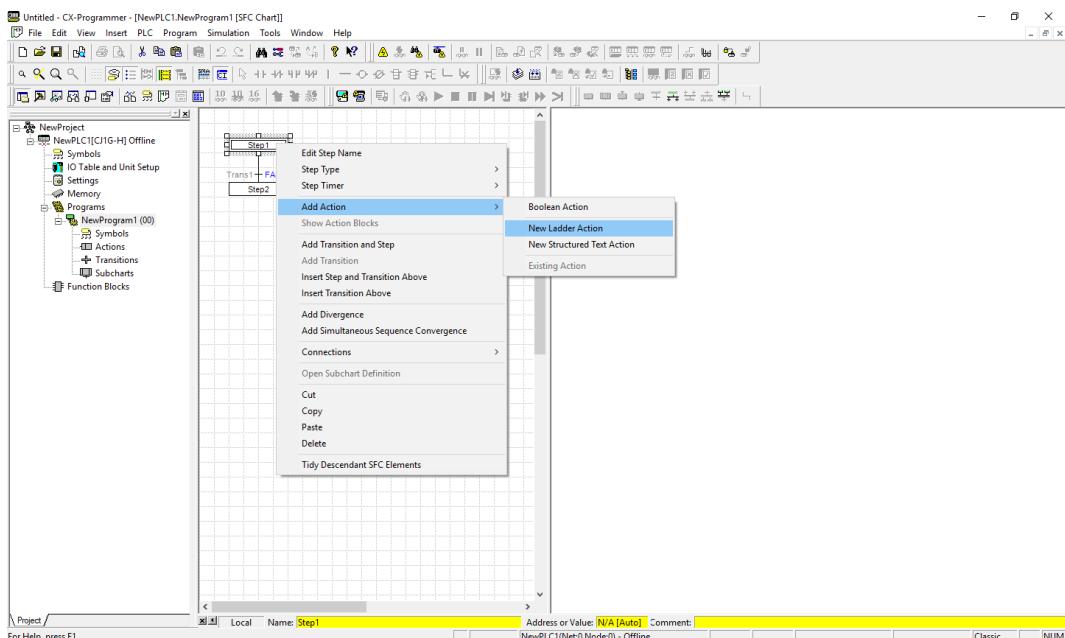


Figure 4.27: Adăugarea acțiunilor (i)

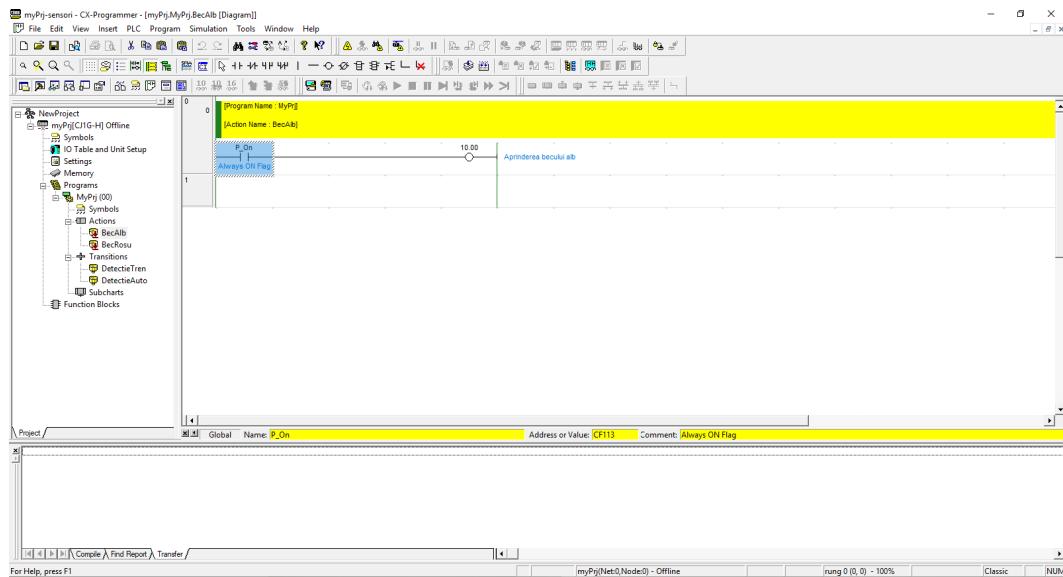


Figure 4.28: Adăugarea acțiunilor (ii)

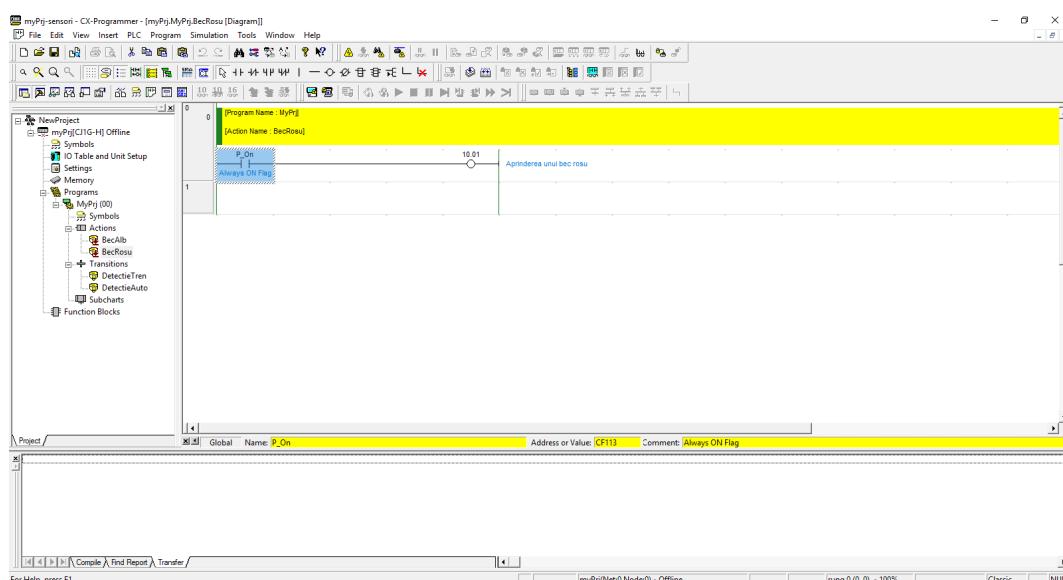


Figure 4.29: Adăugarea acțiunilor (iii)

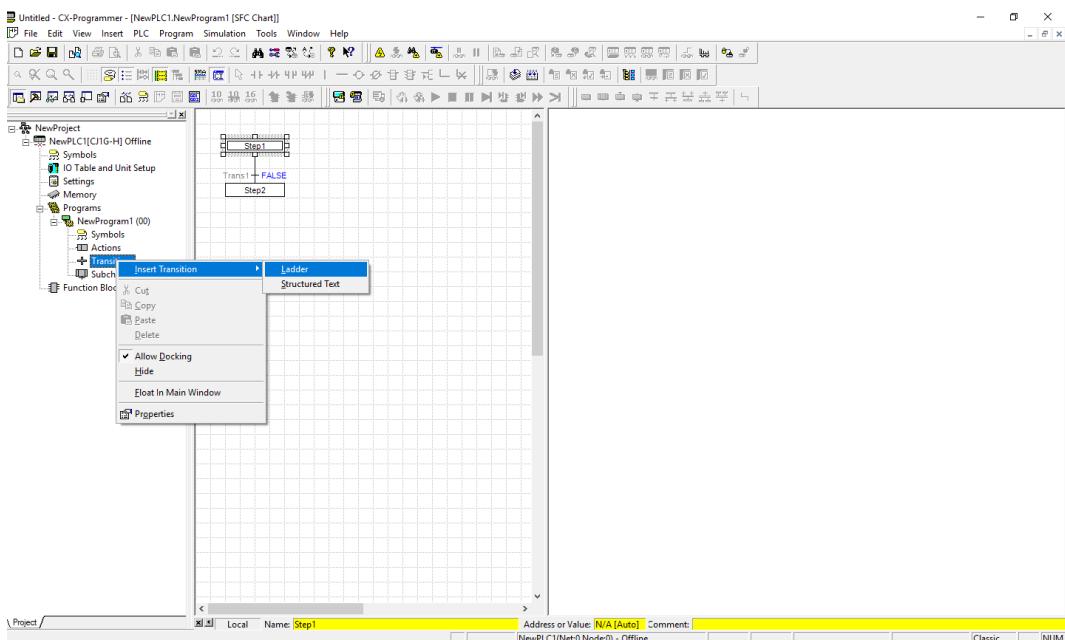


Figure 4.30: Adăugarea tranzițiilor (i)

soluția. La atingerea nivelului  $L_2$  se deschide robinetul  $V_3$  până la golirea rezervorului. Secvența de operații este următoarea:

- Sistemul este pornit la apăsarea butonului „START”, robinetul  $V_1$  este deschis până la atingerea nivelului  $L_1$ ;
- Când nivelul  $L_1$  este atins se pornește mixerul simultan cu deschiderea robinetului  $V_2$  și închiderea robinetului  $V_1$ ;
- Cand nivelul  $L_2$  este atins mixerul se oprește, robinetul  $V_2$  se închide iar robinetul  $V_3$  se deschide. Robinetul  $V_3$  rămâne deschis până când nivelul substanței coboară sub  $L_0$ ;
- După 10 minute dacă nivelul substanței nu este sub  $L_0$  se pornește o alarmă;
- Butonul Ack oprește alarma și permite repornirea procesului de control.

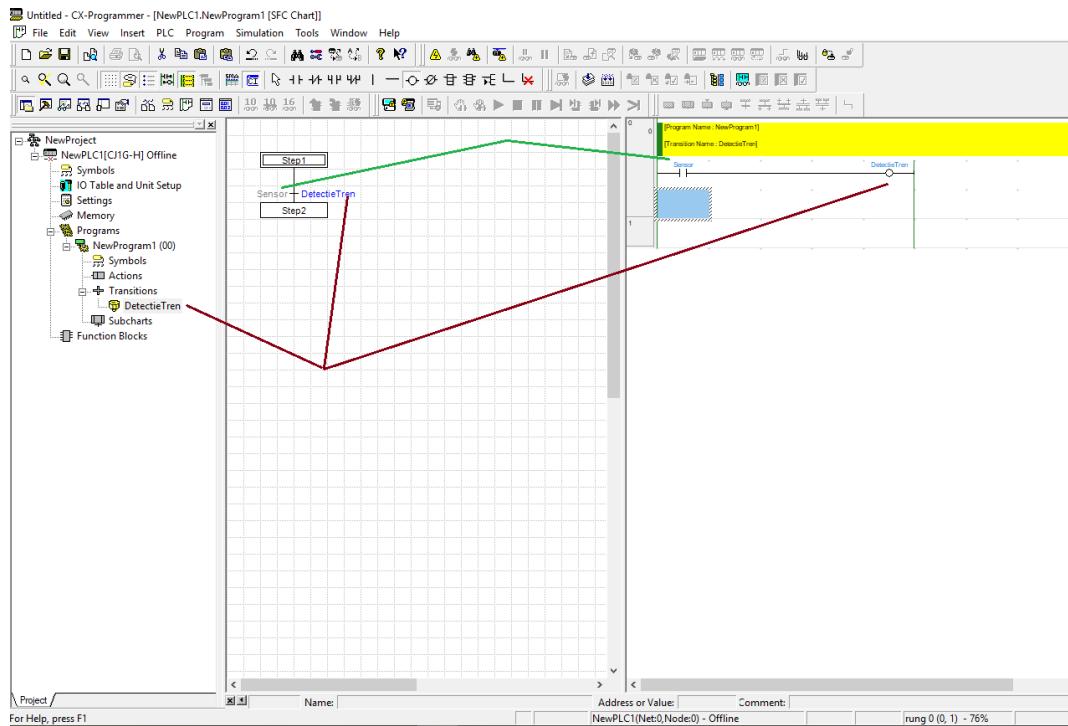


Figure 4.31: Adăugarea tranzițiilor (ii)

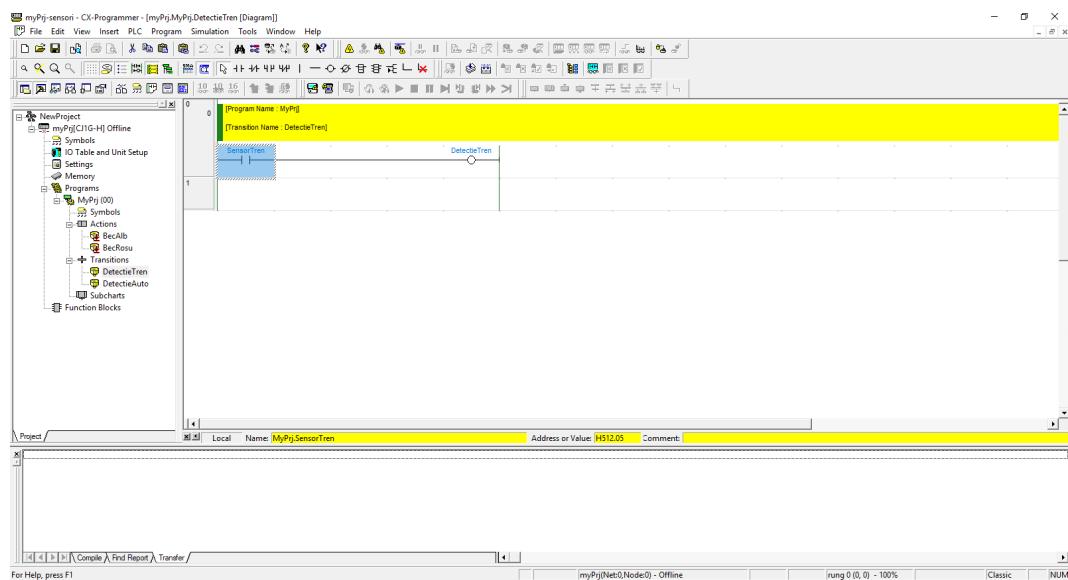


Figure 4.32: Adăugarea tranzițiilor (iii)

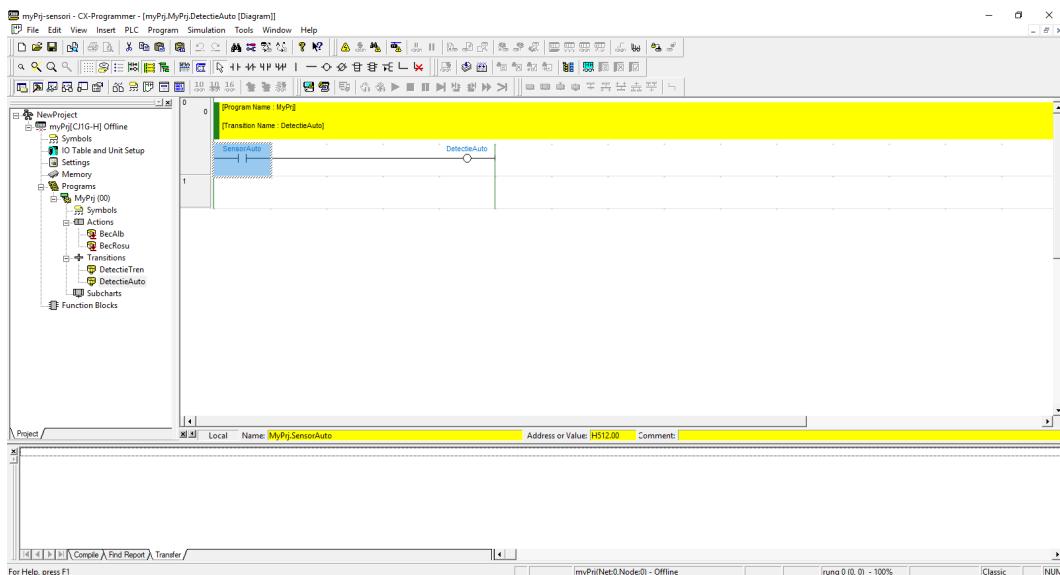


Figure 4.33: Adăugarea tranzițiilor (iv)

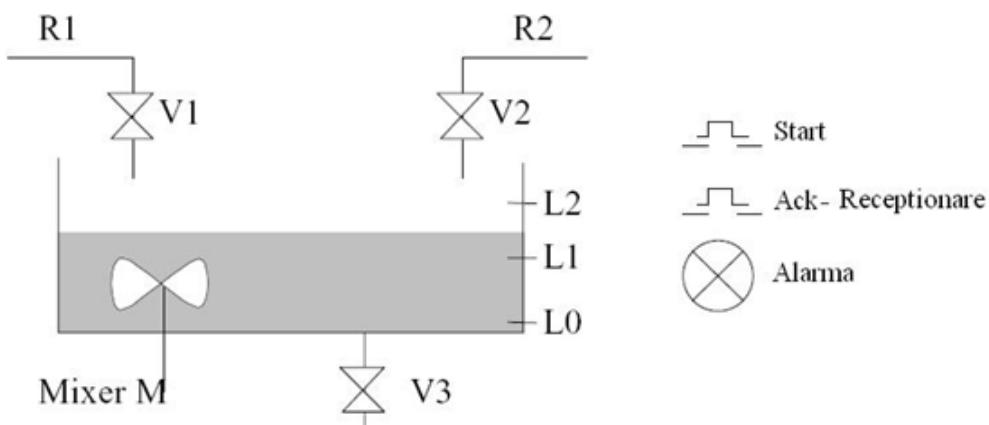


Figure 4.34: Proces

# BIBLIOGRAFIE

- [1] OMRON. *SFC getting started guide* [https://assets.omron.eu/downloads/manual/en/v1/r149\\_sfc\\_getting\\_started\\_guide\\_en.pdf](https://assets.omron.eu/downloads/manual/en/v1/r149_sfc_getting_started_guide_en.pdf)

# 5 MODELAREA SED FOLOSIND REȚELE PETRI - EVOLUȚIA MARCAJULUI REȚELELOR PETRI

---

## 5.1 OBIECTIVE



- Studiul rețelelor Petri;
- Determinarea evoluției marcajului rețelelor Petri;
- Determinarea condițiilor de executabilitate a tranzițiilor;
- Scrierea ecuațiilor de stare.

## 5.2 CONSIDERAȚII TEORETICE

Rețelele Petri modeleză sisteme dinamice cu evenimente discrete. Modelul descrie stările și evenimentele care duc la evoluția acestora. Starea este descrisă cu ajutorul unui set de variabile de stare, reprezentând condițiile. Rețea constă din două părți:

- o structură de rețea care reprezintă partea statică a sistemului;

**Definiția 1**

O rețea Petri este un tuplu de forma:

$$N = (P, T, \text{pre}, \text{post}) \quad (5.1)$$

unde:

$P = \{p_1, p_2, \dots, p_m\}$  - mulțimea finită, nevidă, a locațiilor;

$T = \{t_1, t_2, \dots, t_n\}$  - mulțimea finită, nevidă, a tranzițiilor;

**pre** este o funcție de incidență "înainte" sau de intrare;

$$\text{pre} : P \times T \rightarrow f; \quad (5.2)$$

$$pre(p, t) = \begin{cases} = 0, & \text{dacă } \nexists \text{ un arc de la } p \text{ la } t; \\ \neq 0, & \text{dacă } \exists \text{ un arc de la } p \text{ la } t. \end{cases} \quad (5.3)$$

**post** este o funcție de incidență "după" sau de ieșire;

$$post : TxP \rightarrow f; \quad (5.4)$$

$$post(t, p) = \begin{cases} = 0, & \text{dacă } \nexists \text{ un arc de la } t \text{ la } p; \\ \neq 0 & \text{dacă } \exists \text{ un arc de la } t \text{ la } p. \end{cases} \quad (5.5)$$

- un *marcăj* care corespunde stării globale a sistemului.

### Definiția 2

O rețea Petri este un tuplu de forma:

$$RP = (N, M_0) \quad (5.6)$$

Starea inițială este indicată de marcajul inițial. Marcajul este starea sistemului (rețelei) modelat. Cu marcajele celor m locații se poate construi marcajul rețelei Petri la un moment dat prin:

$$M = [M(p_1), M(p_2), \dots, M(p_m)]^T \quad (5.7)$$

Cu ajutorul funcțiilor de incidență *pre* și *post* se poate construi matricea de incidență,  $C$ , ale cărei elemente,  $c_{ij}$ , sunt definite de relația:

$$c_{ij} = post(t_j, p_i) - pre(p_i, t_j), \quad i = 1, \dots, m; j = 1, \dots, n \quad (5.8)$$

Matricea de incidență descrie structura rețelei Petri și poate fi utilizată la determinarea evoluției sistemului descris de aceasta.

$$C = Post - Pre \quad (5.9)$$

#### 5.2.1 ADMISIBILITATEA TRANZIȚIILOR

O tranziție  $t$  este admisibilă sau executabilă pentru un anumit marcaraj al rețelei,  $M = [M(p_1), \dots, M(p_m)]^T$ , dacă și numai dacă, oricare locație  $p$  din  $P$  verifică relațiile:

1.  $M(p) \geq pre(p, t);$
2.  $K(p) \geq M(p) - pre(p, t) + post(t, p).$

În unele aplicații se consideră că locațiile au o capacitate infinită și prin urmare condiția a două nu mai are rost, în acest caz se spune că se aplică regula nestrictă de admisibilitate. Dacă se iau în considerare și capacitațile locațiilor (deci și regula a două) se spune că se utilizează regula strictă de admisibilitate.

### 5.2.2 COMPORTAMENTUL DINAMIC

Fie  $M$  marcajul rețelei înainte de execuția (engl.: fire) tranziției executabile  $t$ , iar  $M'$  marcajul după execuția ei. Se notează cu  $M[t]M'$  faptul că  $M'$  este obținut prin execuția tranziției  $t$  din  $M$ . Transformarea marcajului este dată de relațiile:

$$M(p)' = M(p) - pre(p, t) + post(t, p), \forall p \in P \quad (5.10)$$

sau

$$M'^T = M^T + col_t(C) \quad (5.11)$$

S-a notat cu  $col_t(C)$  coloana corespunzătoare tranziției  $t$  a matricei  $C$ . Presupunând că rețeaua Petri este pură, rezultă:

$$M[t]M' \iff M'^T = M^T + C * e_t \geq 0 \quad (5.12)$$

unde  $e_t$  este vectorul caracteristic al tranziției  $t$ :

$$e_t(x) = \begin{cases} = 1, & \text{dacă } x = t; \\ = 0, & \text{altfel.} \end{cases} \quad (5.13)$$

Ecuția de mai sus este o *ecuație de stare*, în care  $M$  reprezintă starea actuală,  $M'$ , starea următoare, iar  $e_t$  este *vectorul de intrare*. Starea sau marcajul rețelei Petri se modifică după următoarele reguli:

1. O tranziție  $t$  este executabilă, dacă fiecare locație de intrare  $p$  a lui  $t$  are cel puțin  $pre(p, t)$  jetoane;
2. Nu este necesar ca o tranziție admisibilă să fie executată imediat. Tranzițiile se execută în momentul în care au loc evenimentele reale, sau cele cărora le corespund;
3. Execuția unei tranziții executabile deplasează  $pre(p, t)$  jetoane din fiecare locație de intrare  $p$  a lui  $t$  și adaugă  $post(t, p)$  jetoane în fiecare locație de ieșire  $p$  a lui  $t$ .

Se spune că  $\sigma$  este o secvență posibilă de tranziții (execuții) din marcajul  $M_0$ , sau că marcajul  $M_k$  este accesibil din  $M_0$  prin execuția secvenței  $\sigma : M_0 \rightarrow M_k$ , dacă și numai dacă există marcajele  $M_1, M_2, \dots, M_k$ , astfel încât:

$$M_0 \xrightarrow{t_1} M_1 \xrightarrow{t_2} M_2 \rightarrow \dots \xrightarrow{t_k} M_k$$

Ecuția de stare din marcajul  $M_0$  până la marcajul  $M_k$ , pentru secvența de tranziții  $\sigma = t_1 t_2, \dots$ , este:

$$M_0[\sigma]M_k \Rightarrow M_k = M_0 + C * \sigma \geq 0; \sigma \geq 0 \quad (5.14)$$

unde  $\sigma$  este vectorul de numărare a execuțiilor din  $\sigma$ .  $\sigma(t)$  reprezintă numărul execuției tranziției  $t$  în  $\sigma$ . Ecuția de mai sus se numește *ecuația fundamentală* sau *ecuația de stare a sistemului rețea*.

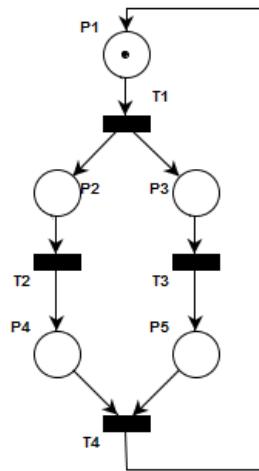


Figure 5.1: Rețeaua Petri 1

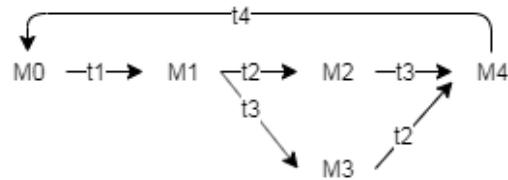


Figure 5.2: Evoluția marcajului pentru RP 1

**Observație:**

Marcajul nu poate fi negativ. Dacă în urma calculelor rezultă valori negative în vectorul  $M$ , atunci tranziția respectivă nu se poate executa din acel maraj.

## 5.3 DESFĂȘURAREA LUCRĂRII

### 5.3.1 PROBLEME REZOLVATE

1. Se consideră rețeaua Petri din figura 5.1.
  - Calculați matricea de incidență.
  - Precizați toate seturile de tranziții executabile.
  - Precizați evoluția marcajelor după fiecare tranziție.

Conform definiției 1, o R.P. este un tuplu de forma:  $RP = \{P, T, PRE, POST\} = N$ .  
Unde:

- $P = \{p_1, p_2, p_3, p_4, p_5\}$  - mulțimea locațiilor

- $T = \{t_1, t_2, t_3, t_4\}$  - mulțimea tranzițiilor
- PRE - matricea costurilor arcelor de la locații la tranziții;
- POST - matricea costurilor arcelor de la tranziții la locații.

$$POST = \begin{vmatrix} post(t_1, p_1) & post(t_2, p_1) & post(t_3, p_1) & post(t_4, p_1) \\ post(t_1, p_2) & post(t_2, p_2) & post(t_3, p_2) & post(t_4, p_2) \\ post(t_1, p_3) & post(t_2, p_3) & post(t_3, p_3) & post(t_4, p_3) \\ post(t_1, p_4) & post(t_2, p_4) & post(t_3, p_4) & post(t_4, p_4) \\ post(t_1, p_5) & post(t_2, p_5) & post(t_3, p_5) & post(t_4, p_5) \end{vmatrix} = \begin{vmatrix} 0 & 0 & 0 & 1 \\ 1 & 0 & 0 & 0 \\ 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \end{vmatrix}$$

$$PRE = \begin{vmatrix} pre(p_1, t_1) & pre(p_1, t_2) & pre(p_1, t_3) & pre(p_1, t_4) \\ pre(p_2, t_1) & pre(p_2, t_2) & pre(p_2, t_3) & pre(p_2, t_4) \\ pre(p_3, t_1) & pre(p_3, t_2) & pre(p_3, t_3) & pre(p_3, t_4) \\ pre(p_4, t_1) & pre(p_4, t_2) & pre(p_4, t_3) & pre(p_4, t_4) \\ pre(p_5, t_1) & pre(p_5, t_2) & pre(p_5, t_3) & pre(p_5, t_4) \end{vmatrix} = \begin{vmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \\ 0 & 0 & 0 & 1 \end{vmatrix}$$

Matricea de incidentă:

$$C = POST - PRE = \begin{vmatrix} -1 & 0 & 0 & 1 \\ 1 & -1 & 0 & 0 \\ 1 & 0 & -1 & 0 \\ 0 & 1 & 0 & -1 \\ 0 & 0 & 1 & -1 \end{vmatrix}$$

Conform definiției 2, o R.P. este un tuplu de forma:  $RP = \{N, M_0\}$ .  
unde:

- $N$  este rețeaua descrisă mai sus
- $M_0$  este marcajul inițial.  
 $M_0 = [M_0(p_1) M_0(p_2) M_0(p_3) M_0(p_4) M_0(p_5)] = [1 0 0 0 0]$ .

Condițiile de admisibilitatea tranzițiilor ( $t_1, t_2, t_3$  și  $t_4$ )

- pentru tranziția  $t_1$ :
  1.  $M(p_1) \geq pre(p_1, t_1)$ ;
  2.  $K(p_2) \geq M(p_2) - pre(p_2, t_1) + post(t_1, p_2)$ ;
  3.  $K(p_3) \geq M(p_3) - pre(p_3, t_1) + post(t_1, p_3)$ .
- pentru tranziția  $t_2$ :

1.  $M(p_2) \geq \text{pre}(p_2, t_2);$
  2.  $K(p_4) \geq M(p_4) - \text{pre}(p_4, t_2) + \text{post}(t_2, p_4).$
- pentru tranziția  $t_3$ :
    1.  $M(p_3) \geq \text{pre}(p_3, t_3);$
    2.  $K(p_5) \geq M(p_5) - \text{pre}(p_5, t_3) + \text{post}(t_3, p_5).$
  - pentru tranziția  $t_4$ :
    1.  $M(p_4) \geq \text{pre}(p_4, t_4);$
    2.  $M(p_5) \geq \text{pre}(p_5, t_4);$
    3.  $K(p_1) \geq M(p_1) - \text{pre}(p_1, t_4) + \text{post}(t_4, p_1).$

Pornind din  $M_0$  se poate urmări evoluția marcajului în figura 5.2.

Determinarea marcajelor se face folosind ecuațiile 5.10 sau 5.11.

Din marcajul  $M_0$  se execută tranziția  $t_1$ .

$$M_1 = [M_1(p_1) M_1(p_2) M_1(p_3) M_1(p_4) M_1(p_5)];$$

$$M_1(p_1) = M_0(p_1) - \text{pre}(p_1, t_1) + \text{post}(t_1, p_1) = 1 - 1 + 0 = 0;$$

$$M_1(p_2) = M_0(p_2) - \text{pre}(p_2, t_1) + \text{post}(t_1, p_2) = 0 - 0 + 1 = 1;$$

$$M_1(p_3) = M_0(p_3) - \text{pre}(p_3, t_1) + \text{post}(t_1, p_3) = 0 - 0 + 1 = 1;$$

$$M_1(p_4) = M_0(p_4) - \text{pre}(p_4, t_1) + \text{post}(t_1, p_4) = 0 - 0 + 0 = 0;$$

$$M_1(p_5) = M_0(p_5) - \text{pre}(p_5, t_1) + \text{post}(t_1, p_5) = 0 - 0 + 0 = 0;$$

$$M_1 = [01100];$$

Din marcajul  $M_1$  se execută tranziția  $t_2$ .

$$M_2 = [M_2(p_1) M_2(p_2) M_2(p_3) M_2(p_4) M_2(p_5)];$$

$$M_2(p_1) = M_1(p_1) - \text{pre}(p_1, t_2) + \text{post}(t_2, p_1) = 0 - 0 + 0 = 0;$$

$$M_2(p_2) = M_1(p_2) - \text{pre}(p_2, t_2) + \text{post}(t_2, p_2) = 1 - 1 + 0 = 0;$$

$$M_2(p_3) = M_1(p_3) - \text{pre}(p_3, t_2) + \text{post}(t_2, p_3) = 1 - 0 + 0 = 1;$$

$$M_2(p_4) = M_1(p_4) - \text{pre}(p_4, t_2) + \text{post}(t_2, p_4) = 0 - 0 + 1 = 1;$$

$$M_2(p_5) = M_1(p_5) - \text{pre}(p_5, t_2) + \text{post}(t_2, p_5) = 0 - 0 + 0 = 0;$$

$$M_2 = [00110];$$

Din marcajul  $M_1$  se execută tranziția  $t_3$ .

$$M_3 = [M_3(p_1) M_3(p_2) M_3(p_3) M_3(p_4) M_3(p_5)];$$

$$M_3(p_1) = M_1(p_1) - \text{pre}(p_1, t_3) + \text{post}(t_3, p_1) = 0 - 0 + 0 = 0;$$

$$M_3(p_2) = M_1(p_2) - \text{pre}(p_2, t_3) + \text{post}(t_3, p_2) = 1 - 0 + 0 = 1;$$

$$M_3(p_3) = M_1(p_3) - \text{pre}(p_3, t_3) + \text{post}(t_3, p_3) = 1 - 1 + 0 = 0;$$

$$M_3(p_4) = M_1(p_4) - \text{pre}(p_4, t_3) + \text{post}(t_3, p_4) = 0 - 0 + 0 = 0;$$

$$M_3(p_5) = M_1(p_5) - \text{pre}(p_5, t_3) + \text{post}(t_3, p_5) = 0 - 0 + 1 = 1;$$

$$M_3 = [01001];$$

Din marcajul  $M_2$  se execută tranziția  $t_3$ .

$$M_4 = [M_4(p_1) M_4(p_2) M_4(p_3) M_4(p_4) M_4(p_5)];$$

$$M_4(p_1) = M_2(p_1) - \text{pre}(p_1, t_3) + \text{post}(t_3, p_1) = 0 - 0 + 0 = 0;$$

$$M_4(p_2) = M_2(p_2) - \text{pre}(p_2, t_3) + \text{post}(t_3, p_2) = 0 - 0 + 0 = 0;$$

$$M_4(p_3) = M_2(p_3) - \text{pre}(p_3, t_3) + \text{post}(t_3, p_3) = 1 - 1 + 0 = 0;$$

$$M_4(p_4) = M_2(p_4) - \text{pre}(p_4, t_3) + \text{post}(t_3, p_4) = 1 - 0 + 0 = 1;$$

$$M_4(p_5) = M_2(p_5) - \text{pre}(p_5, t_3) + \text{post}(t_3, p_5) = 0 - 0 + 1 = 1;$$

$$M_4 = [00011];$$

Din marcajul  $M_3$  se execută tranziția  $t_2$ .

$$M_4 = [M_4(p_1) M_4(p_2) M_4(p_3) M_4(p_4) M_4(p_5)];$$

$$M_4(p_1) = M_3(p_1) - \text{pre}(p_1, t_2) + \text{post}(t_2, p_1) = 0 - 0 + 0 = 0;$$

$$M_4(p_2) = M_3(p_2) - \text{pre}(p_2, t_2) + \text{post}(t_2, p_2) = 1 - 1 + 0 = 0;$$

$$M_4(p_3) = M_3(p_3) - \text{pre}(p_3, t_2) + \text{post}(t_2, p_3) = 0 - 0 + 0 = 0;$$

$$M_4(p_4) = M_3(p_4) - \text{pre}(p_4, t_2) + \text{post}(t_2, p_4) = 0 - 0 + 1 = 1;$$

$$M_4(p_5) = M_3(p_5) - \text{pre}(p_5, t_2) + \text{post}(t_2, p_5) = 1 - 0 + 0 = 1;$$

$$M_4 = [00011];$$

Din marcajul  $M_4$  se execută tranziția  $t_4$ .

$$M_0 = [M_0(p_1) M_0(p_2) M_0(p_3) M_0(p_4) M_0(p_5)];$$

$$M_0(p_1) = M_4(p_1) - \text{pre}(p_1, t_4) + \text{post}(t_4, p_1) = 0 - 0 + 1 = 1;$$

$$M_0(p_2) = M_4(p_2) - \text{pre}(p_2, t_4) + \text{post}(t_4, p_2) = 0 - 0 + 0 = 0;$$

$$M_0(p_3) = M_4(p_3) - \text{pre}(p_3, t_4) + \text{post}(t_4, p_3) = 0 - 0 + 0 = 0;$$

$$M_0(p_4) = M_4(p_4) - \text{pre}(p_4, t_4) + \text{post}(t_4, p_4) = 1 - 1 + 0 = 0;$$

$$M_0(p_5) = M_4(p_5) - \text{pre}(p_5, t_4) + \text{post}(t_4, p_5) = 1 - 1 + 0 = 0;$$

$$M_0 = [10000];$$

sau

$$M_1^T = M_0^T + C * e_{t1}(x);$$

$$e_{t1}(x)^T = \begin{vmatrix} 1 \\ 0 \\ 0 \\ 0 \end{vmatrix};$$

$$M_1^T = \begin{vmatrix} 1 \\ 0 \\ 0 \\ 0 \\ 0 \end{vmatrix} + \begin{vmatrix} -1 & 0 & 0 & 1 \\ 1 & -1 & 0 & 0 \\ 1 & 0 & -1 & 0 \\ 0 & 1 & 0 & -1 \\ 0 & 0 & 1 & -1 \end{vmatrix} * \begin{vmatrix} 1 \\ 0 \\ 0 \\ 0 \\ 0 \end{vmatrix} = \begin{vmatrix} 1 \\ 0 \\ 0 \\ 0 \\ 0 \end{vmatrix} + \begin{vmatrix} -1 \\ 1 \\ 1 \\ 0 \\ 0 \end{vmatrix} = \begin{vmatrix} 1 \\ 1 \\ 1 \\ 0 \\ 0 \end{vmatrix}$$

$$M_2^T = \begin{vmatrix} 0 \\ 1 \\ 1 \\ 0 \\ 0 \end{vmatrix} + \begin{vmatrix} 0 \\ -1 \\ 0 \\ 1 \\ 0 \end{vmatrix} = \begin{vmatrix} 0 \\ 0 \\ 1 \\ 1 \\ 0 \end{vmatrix}$$

$$M_3^T = \begin{vmatrix} 0 \\ 1 \\ 1 \\ 0 \\ 0 \end{vmatrix} + \begin{vmatrix} 0 \\ 0 \\ -1 \\ 0 \\ 1 \end{vmatrix} = \begin{vmatrix} 0 \\ 1 \\ 0 \\ 0 \\ 1 \end{vmatrix}$$

$$M_4^T = \begin{vmatrix} 0 \\ 0 \\ 1 \\ 0 \\ 0 \end{vmatrix} + \begin{vmatrix} 0 \\ 0 \\ 0 \\ 1 \\ 1 \end{vmatrix} = \begin{vmatrix} 0 \\ 0 \\ 0 \\ 1 \\ 1 \end{vmatrix}$$

$$M_4^T = \begin{vmatrix} 0 & 0 & 0 \\ 1 & -1 & 0 \\ 0 & 0 & 0 \\ 0 & 1 & 1 \\ 1 & 0 & 1 \end{vmatrix} = \begin{vmatrix} 0 & 0 & 0 \\ 0 & 0 & 0 \\ 0 & 0 & 0 \\ 0 & 0 & 0 \\ 0 & 0 & 0 \end{vmatrix}$$

$$M_0^T = \begin{vmatrix} 0 & 1 & 1 \\ 0 & 0 & 0 \\ 0 & 0 & 0 \\ 1 & -1 & 0 \\ 1 & -1 & 0 \end{vmatrix} = \begin{vmatrix} 0 & 1 & 1 \\ 0 & 0 & 0 \\ 0 & 0 & 0 \\ 0 & 0 & 0 \\ 0 & 0 & 0 \end{vmatrix}$$

### 5.3.2 PROBLEME PROPUSE

1. Se consideră rețeaua Petri din figura 5.3.
  - Calculați matricea de incidență;
  - Precizați toate seturile de tranziții executabile și evoluția marcajelor după fiecare tranziție;
  - Precizați evoluția marcajelor pentru întreaga rețea Petri.
2. Se consideră rețeaua Petri din figura 5.4.
  - Calculați matricea de incidență;
  - Precizați toate seturile de tranziții executabile și evoluția marcajelor după fiecare tranziție;
  - Precizați evoluția marcajelor pentru întreaga rețea Petri.
3. Se consideră rețeaua Petri din figura 5.5.
  - Calculați matricea de incidență;
  - Precizați toate seturile de tranziții executabile și evoluția marcajelor după fiecare tranziție;
  - Precizați evoluția marcajelor pentru întreaga rețea Petri.
4. Pentru una dintre RP (locațiile având capacitate inifinite) din figurile 5.6, 5.7 și 5.8 să se determine evoluția marcajului.
5. Pentru RP avand structura și marcajul inițial indicat în figurile 5.6, 5.7 și 5.8 să se precizeze care dintre tranziții sunt validate și marcajul ce rezultă după executarea fiecărei din aceste tranziții în cazurile următoare:
  - a) Figura 5.6:  $k(p_1) = k(p_3) = 2; k(p_2) = 1$ .

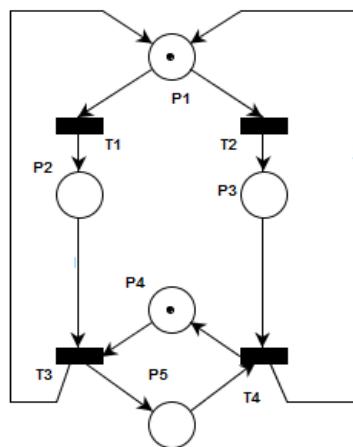


Figure 5.3: Rețeaua Petri 2

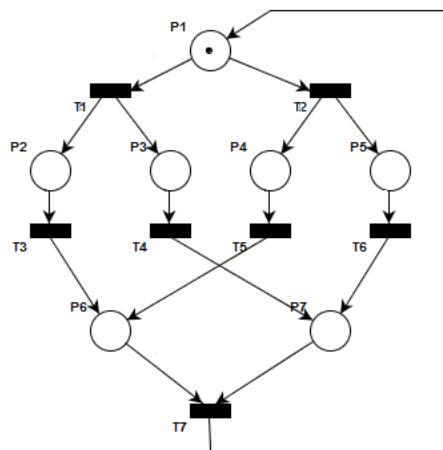


Figure 5.4: Rețeaua Petri 3

- b) Figura 5.7:  $k(p_1) = k(p_2) = k(p_3) = 1$ .
  - c) Figura 5.8:  $k(p_1) = 2; k(p_2) = 3; k(p_3) = k(p_4) = 1$ .
  - d) Să se scrie ecuația de stare pentru un marcaj la alegere.
6. Pentru RP din figura 5.9, să se determine matricea de incidență și să se scrie o secvență de cel puțin 5 tranziții executabile. Să se scrie condițiile de admisibilitate pentru fiecare tranziție.
7. Pentru RP având structura și marcajul inițial indicat în figura 5.10 să se precizeze care tranziții sunt validate și marcajul ce rezultă după executarea fiecăreia din aceste tranziții în cazurile următoare:

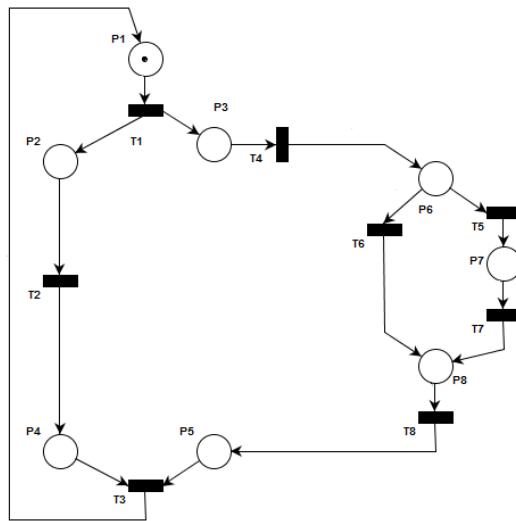


Figure 5.5: Rețeaua Petri 4

- $k(p_1) = k(p_2) = 2; k(p_3) = k(p_4) = k(p_5) = 3.$
  - Capacități infinite.
  - Care traiție este executabilă din marcajul inițial și ce maraj rezultă.
  - Să se scrie ecuația de stare pentru un maraj la alegere.
8. Pentru RP din figura 5.11, să se determine matricea de incidență și să se scrie o secvență de cel puțin 4 tranzitii executabile. Să se scrie condițiile de admisibilitate pentru fiecare traiție.

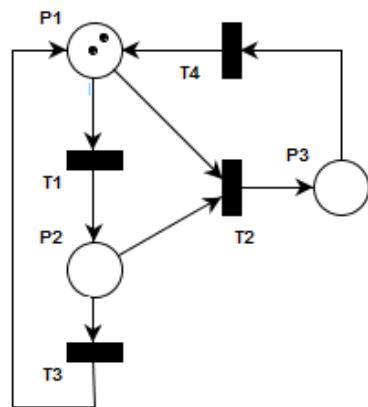


Figure 5.6: Rețeaua Petri 5

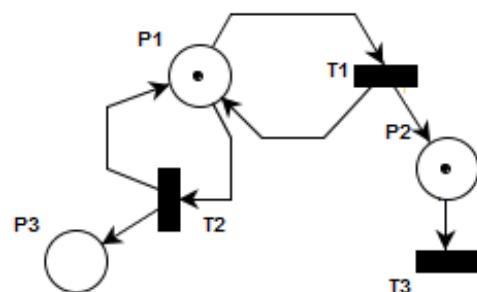


Figure 5.7: Rețeaua Petri 6

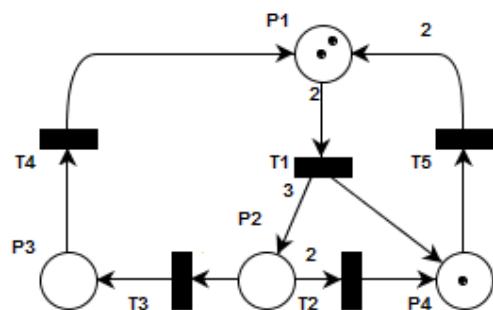


Figure 5.8: Rețeaua Petri 7

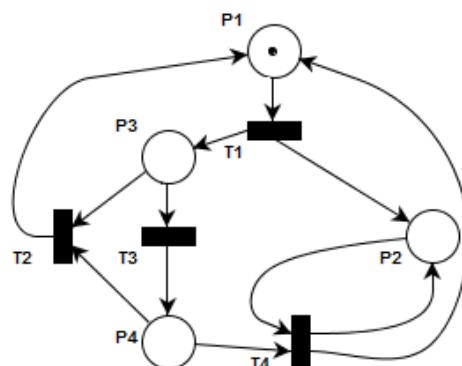


Figure 5.9: Rețeaua Petri 8

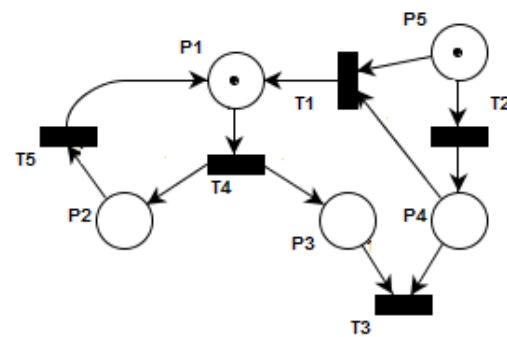


Figure 5.10: Rețeaua Petri 9

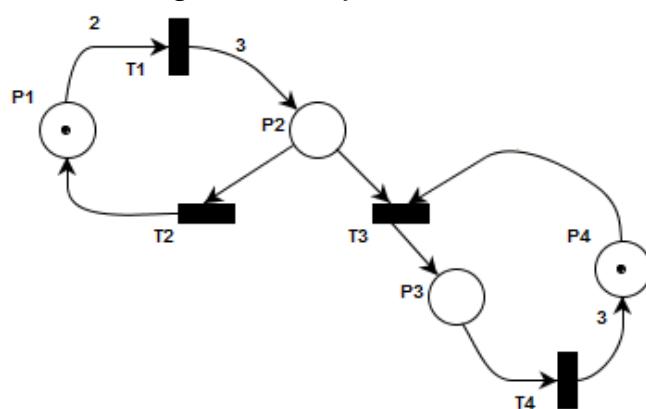


Figure 5.11: Rețeaua Petri 10

## BIBLIOGRAFIE

- [1] Adina Aştilean. *Sisteme cu evenimente discrete, notițe curs.*
- [2] Christos G. Cassandras, Stéphane Lafortune. *Introduction to Discrete Event Systems, Second Edition*, ISBN-13: 978-0-387-33332-8, e-ISBN-13: 978-0-387-68612-7, 2008 Springer Science+Business Media, LLC.

# **6 MODELAREA SED FOLOSIND REȚELE PETRI - PROPRIETĂȚI ALE REȚELELOR PETRI**

---

## **6.1 OBIECTIVE**



- Studiul proprietăților structurale ale rețelelor Petri:
  - viabilitate;
  - mărginire;
  - reversibilitate.

## **6.2 CONSIDERAȚII TEORETICE**

Viabilitatea, mărginirea și reversibilitatea sunt proprietăți importante în studiul rețelelor Petri.

### **• Mărginirea**

O rețea Petri  $PN = (N, M_0)$  se numește mărginită (sau k-mărginită) dacă numărul jetoanelor din fiecare locație nu depășește niciodată un anumit număr natural k, indiferent de marcajul realizat, adică: pentru oricare p din P și oricare M din  $R(M_0)$  rezultă  $M(p) \leq k$ .

O rețea Petri se numește sigură (engl.: safe) dacă este 1-mărginită. Dacă o rețea Petri este mărginită din oricare marcaj inițial finit, ea se numește structural mărginită.

### **• Viabilitatea**

Acest concept este legat de absența interblocajelor în sistemele în care se pune problema utilizării în comun a mai multor resurse.

O rețea Petri  $PN = (N, M_0)$  se numește viabilă (sau echivalent,  $M_0$  este un marcaj viabil pentru rețeaua N) dacă indiferent care este marcajul realizabil din  $M_0$ , se poate

execută în continuare oricare tranziție din rețea, eventual executând în prealabil anumite secvențe. Ea se numește structural viabilă dacă există un maraj inițial viabil pentru rețea.

Dacă o rețea este viabilă, atunci se garantează lipsa interblocajelor. O tranziție  $t \in T$  este viabilă dintr-un maraj  $M_0$  (inițial) dacă și numai dacă oricare ar fi marajul  $M$  accesibil din  $M_0$  există o secvență  $\sigma$  de execuții care o conține pe  $t$ . (O tranziție viabilă rămâne în permanență potențial executabilă, activitatea pe care o reprezintă fiind întotdeauna posibilă).

O tranziție  $t$  din  $PN = (N, M_0)$  se numește:

- $L_0$  - **moartă**

dacă  $t$  nu poate fi executată în nici o secvență din  $L(M_0)$ ;

- $L_1$  - **viabilă** (potențial executabilă)

dacă  $t$  poate fi executată cel puțin o dată în unele secvențe executabile din  $L(M_0)$ ;

- $L_2$  - **viabilă**

dacă, dându-se un număr natural  $k$ ,  $t$  poate fi executată de cel puțin  $k$  ori în unele secvențe din  $L(M_0)$ ;

- $L_3$  - **viabilă**

dacă  $t$  apare de o infinitate de ori în unele secvențe din  $L(M_0)$ ;

- $L_4$  - **neviabilă**

dacă  $t$  este viabilă din oricare maraj aparținând setului  $L(M_0)$ .

Rețeaua Petri este  $L_k$  - viabilă dacă fiecare tranziție a sa,  $t$ , este  $L_k$  - viabilă ( $k = 1, 2, 3, 4$ ).

$L_4$ -viabilitatea este cea mai puternică și corespunde definiției anterioare a viabilității. Dacă o rețea este  $L_k$  - viabilă, atunci este și  $L_{k-1}$  - viabilă.

O tranziție este strict  $L_k$  - viabilă dacă este  $L_k$  - viabilă, dar nu este  $L_{k+1}$  - viabilă.

#### • Reversibilitatea

O rețea Petri  $PN = (N, M_0)$  este reversibilă dacă oricare ar fi marajul  $M$  din setul  $R(M_0)$ ,  $M_0$  este realizabil din  $M$ . Prin urmare, rețeaua este reversibilă dacă totdeauna se poate ajunge din nou în starea inițială.

Uneori, în loc de starea inițială, se cere să se determine dacă rețeaua poate ajunge într-o anumită stare de bază dată.  $M'$  este o stare de bază dacă oricare ar fi marajul  $M$  din setul  $R(M_0)$ ,  $M'$  este realizabil din  $M$ .

Viabilitatea, mărginirea și reversibilitatea nu sunt dependente una relativ la celalaltă.

### 6.3 DESFĂȘURAREA LUCRĂRII

1. Pentru rețelele Petri din figurile [6.1](#), [6.2](#) și [6.3](#) stabiliți care este gradul de viabilitate al fiecărei tranziții. Precizați de asemenea gradul de viabilitate al rețelelor respective.  
Justificați răspunsul.
2. Pentru rețelele Petri din figurile [6.1](#) și [6.2](#) stabiliți dacă sunt mărginite.  
Justificați răspunsul.
3. Pentru rețelele Petri din figurile [6.1](#) și [6.3](#) stabiliți dacă sunt reversibile.  
Justificați răspunsul.
4. Se consideră rețelele Petri din figurile [6.4](#), [6.5](#), [6.6](#), [6.7](#), [6.8](#), [6.9](#) și [6.10](#). Stabiliți dacă fiecare dintre rețelele respective este mărginită, reversibilă și viabilă. Stabiliciți gradul de viabilitate. Justificați răspunsul.

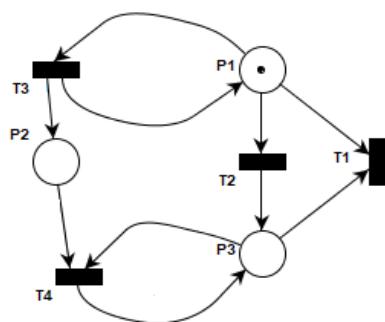


Figure 6.1: Rețeaua Petri 1

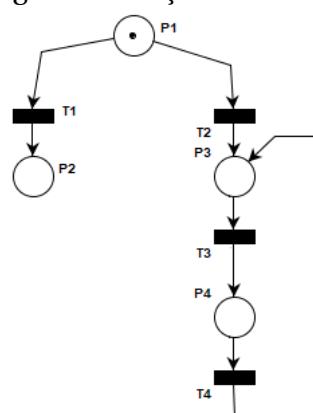


Figure 6.2: Rețeaua Petri 2

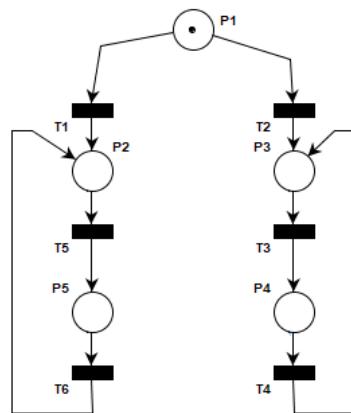


Figure 6.3: Rețeaua Petri 3

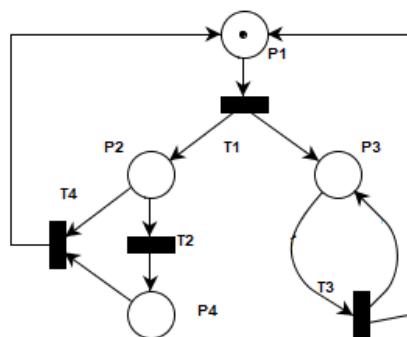


Figure 6.4: Rețeaua Petri 4

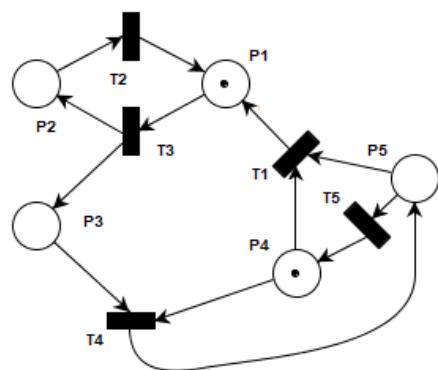


Figure 6.5: Rețeaua Petri 5

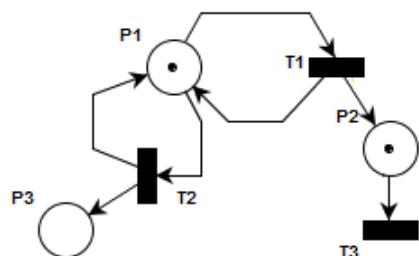


Figure 6.6: Rețeaua Petri 6

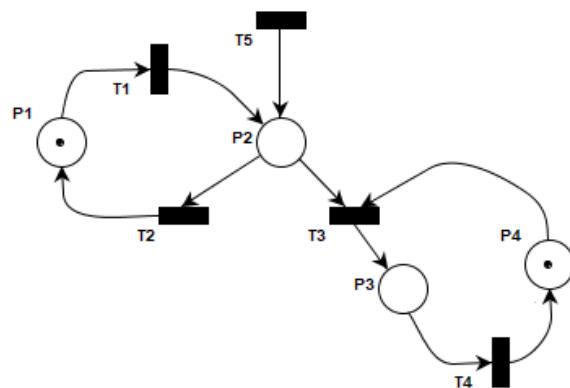


Figure 6.7: Rețeaua Petri 7

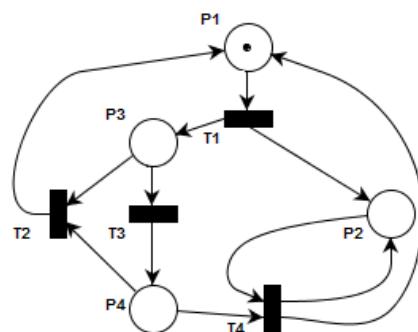


Figure 6.8: Rețeaua Petri 8

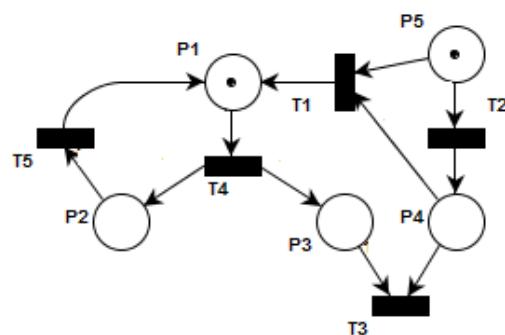


Figure 6.9: R.P. 9

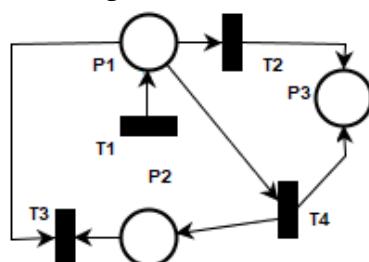


Figure 6.10: R.P. 10

## BIBLIOGRAFIE

- [1] Adina Aştilean. *Sisteme cu evenimente discrete, notițe curs.*
- [2] Christos G. Cassandras, Stéphane Lafortune. *Introduction to Discrete Event Systems, Second Edition*, ISBN-13: 978-0-387-33332-8, e-ISBN-13: 978-0-387-68612-7, 2008 Springer Science+Business Media, LLC.

# **7 MODELAREA SED FOLOSIND REȚELE PETRI - ANALIZA CALITATIVĂ A MODELELOR DESCRISE PRIN REȚELE PETRI. GRAFUL DE REALIZARE. ARBORELE DE ACOPERIRE**

---

## **7.1 OBIECTIVE**



- Studiul metodelor de analiză a rețelelor Petri;
- Familiarizarea cu metodele de analiză calitativă:
  - Graful de realizare;
  - Arborele de acoperire.

## **7.2 CONSIDERAȚII TEORETICE**

Metodele de analiză ale rețelelor Petri pot fi divizate în următoarele grupe:

- analiza prin enumerare;
- analiza prin transformare;
- analiza structurală;
- analiza prin simulare.

Primele trei metode sunt statice, iar cea de a treia este dinamică.

Analiza prin enumerare se bazează pe constituirea *grafului de realizare* sau a *arborelui de acoperire*, care descriu marcajele și tranzițiile executabile.

### 7.2.1 ANALIZA PRIN ENUMERARE

Metodele de analiză prin enumerare se bazează pe construcția grafului de realizare, care reprezintă, individual, marcajele rețelei și execuțiile tranzițiilor. Dacă sistemul nu este mărginit, graful de realizare este infinit, deci imposibil de construit, în acest caz se construiește *arborele de acoperire*.

#### 1. Metoda grafului de realizare

Se spune că un marcaj este realizabil într-o rețea  $(N, M_0)$  dacă există o secvență  $\sigma$ ,  $M_0[\sigma]M$ , astfel încât  $M \in R(N, M_0)$ . Sistemele cu stări (marcaje) infinite nu au reprezentări finite. Pentru sisteme mărginite, abordarea analizei prin metoda grafului de realizare se bazează pe simularea secvențială exhaustivă.

 **Definiție:** Graful de realizare, asociat cu o rețea  $(N, M_0)$ , este un graf  $GR(N, M_0)$  în care fiecare nod reprezintă un marcaj realizabil din  $M_0$  și fiecare arc reprezintă execuția unei tranziții. Există un arc etichetat  $t_k$ , care pornește din nodul reprezentând marcajul  $M_i$  spre nodul reprezentând marcajul  $M_j$ , dacă marcajul  $M_j$  poate fi realizat din marcajul  $M_i$  prin execuția tranziției  $t_k$ ,  $M_i[t_k]M_j$ .

2. **Metoda arborelui de acoperire** Acoperirea: Un marcaj  $M$  din  $PN = (N, M_0)$  se numește acoperibil dacă există un marcaj  $M'$  din setul  $R(M_0)$  astfel încât  $M'(p) \geq M(p)$  pentru oricare  $p$  din  $P$ .

Dându-se o rețea Petri  $PN = (N, M_0)$ , se poate construi un arbore pornind din marcajul inițial  $M_0$  și construind câte un arc pentru fiecare tranziție executabilă. La capetele arcului se notează marcajul rețelei obișnuit ca urmare a execuției tranziției, iar arcul se marchează cu tranziția care determină această transformare. Operația poate continua în același fel până la epuizarea tuturor tranzițiilor executabile și realizarea tuturor marcajelor, dacă rețeaua Petri este mărginită. Aplicarea acestei metode la rețelele Petri nemărginite este imposibilă din cauză că dezvoltarea nu are sfârșit. Pentru a menține arborele finit se propune introducerea unui simbol special  $\omega$ , care poate fi interpretat ca un infinit. El verifică, pentru orice număr natural, relațiile:  $\omega > n$ ;  $\omega \pm n = \omega$  și  $\omega \geq \omega$ .

Folosind simbolul  $\omega$ , arborele de acoperire poate fi construit utilizând următorul algoritm:

**Pas 1:** Se etichetează marcajul inițial  $M_0$  ca rădăcină și nou.

**Pas 2:** Cât timp marcajele existente sunt noi, execută:

**Pas 2.1:** Selectează un marcaj  $M$  etichetat ca nou;

**Pas 2.2:** Dacă  $M$  este identic cu un maraj din calea de la rădăcină până la  $M$ , atunci se etichetează  $M$  ca vechi și se merge la un alt maraj;

**Pas 2.3:** Dacă nu există tranziții admisibile din  $M$ , atunci se marchează  $M$  cu sfârșit;

**Pas 2.4:** Pentru fiecare tranziție  $t$ , admisibilă din  $M$ , execută:

**Pas 2.4.1:** Fie  $M'$  marajul care rezultă după execuția tranziției  $t$  din  $M$ .

**Pas 2.4.2:** Dacă există pe calea de la rădăcină până la  $M$  un maraj  $M''$ , astfel încât  $M'(p) \geq M''(p)$  pentru oricare locație  $p$  și  $M' \neq M''$ , adică  $M''$  este acoperit, atunci se înlocuiește  $M'(p)$  cu  $\omega$ , pentru fiecare locație  $p$  care verifică  $M'(p) > M''(p)$ .

**Pas 2.4.3:** Se introduce  $M'$  ca un nod nou și se desenează un arc cu eticheta  $t$  din  $M$  la  $M'$ .

Cu arborele de acoperire se poate construi graful de acoperire care este un graf orientat. El are ca noduri, nodurile etichetate distinct în arborele de acoperire, iar ca arce tranzițiile corespunzătoare.

## 7.3 DESFĂȘURAREA LUCRĂRII

### 7.3.1 PROBLEME PROPUSE

Să se realizeze graful de acoperire sau arborele de realizare după caz, pentru figurile [7.1](#), [7.2](#), [7.3](#), [7.4](#), [7.5](#), [7.6](#), [7.7](#), [7.8](#), [7.9](#) și [7.10](#).

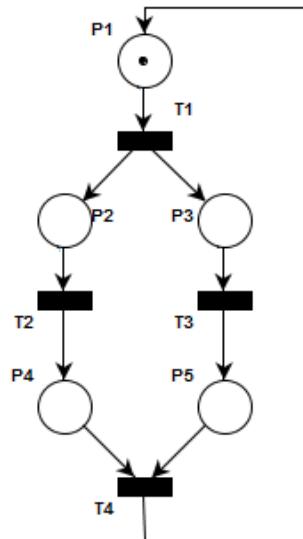


Figure 7.1: R.P. 1

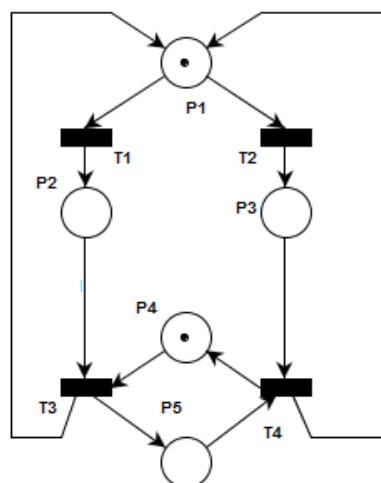


Figure 7.2: R.P. 2

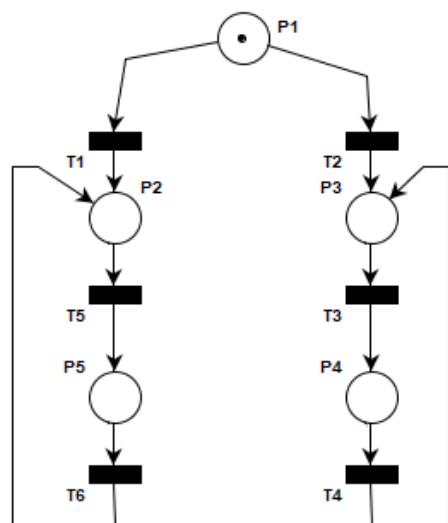


Figure 7.3: R.P. 3

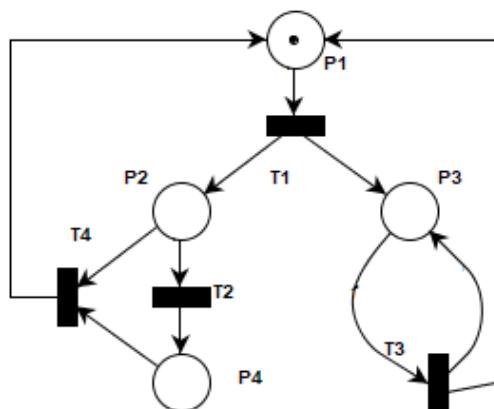


Figure 7.4: R.P. 4

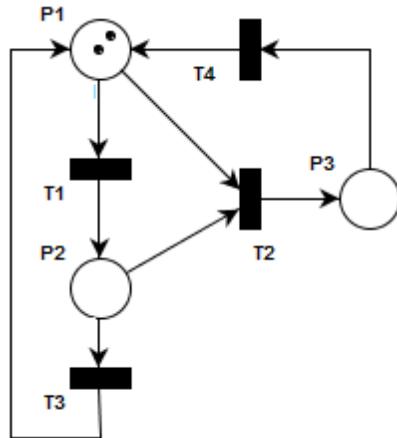


Figure 7.5: R.P. 5

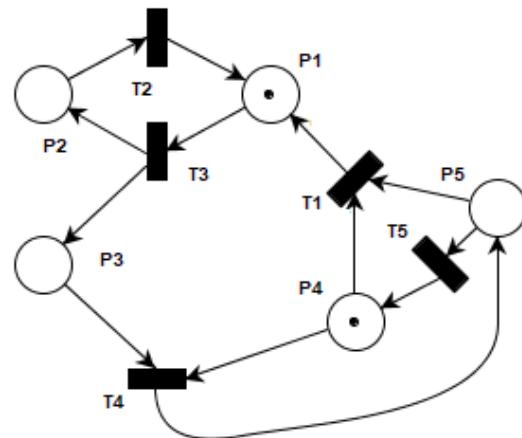


Figure 7.6: R.P. 6

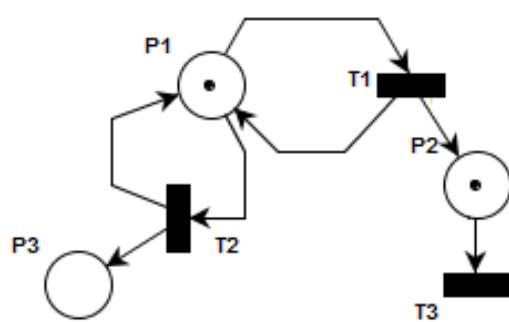


Figure 7.7: R.P. 7

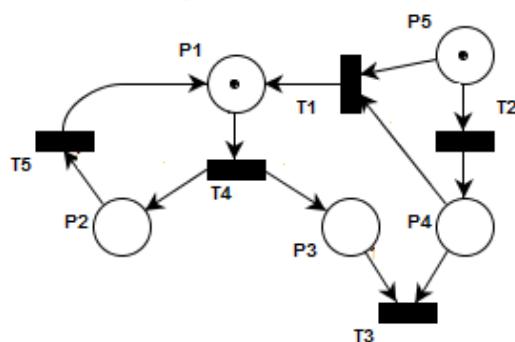


Figure 7.8: R.P. 8

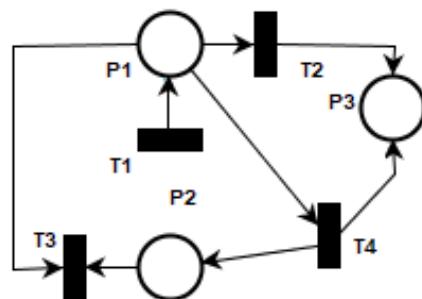


Figure 7.9: R.P. 9

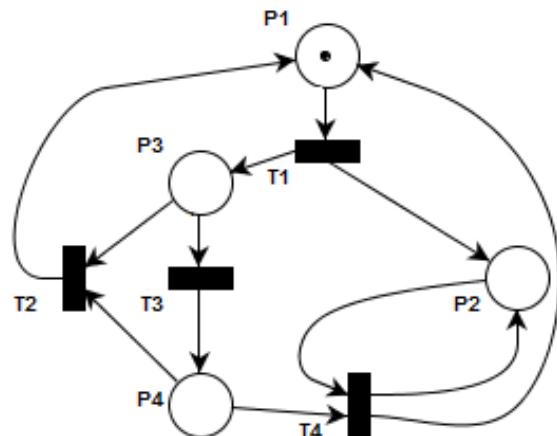


Figure 7.10: R.P. 10

## BIBLIOGRAFIE

- [1] Adina Aştilean. *Sisteme cu evenimente discrete, notițe curs.*
- [2] Christos G. Cassandras, Stéphane Lafortune. *Introduction to Discrete Event Systems, Second Edition*, ISBN-13: 978-0-387-33332-8, e-ISBN-13: 978-0-387-68612-7, 2008 Springer Science+Business Media, LLC.

# **8 MODELAREA SED FOLOSIND REȚELE PETRI - METODE ALGEBRICE DE ANALIZĂ. UTILIZAREA INVARIANȚILOR**

---

## **8.1 OBIECTIVE**



- Studiul metodelor de analiză a rețelelor Petri;
- Familiarizarea cu metode de analiză algebrice:
  - Invariantul de tip P;
  - Invariantul de tip T.

## **8.2 CONSIDERAȚII TEORETICE**

Invarianții sunt utilizati în scopul studierii comportării sistemului și al verificării proprietăților logice ale acestuia. Au fost definite două tipuri de invarianți, P și T.

### **8.2.1 INVARIANTUL DE TIP P**

Invariantul de tip P este un vector care are proprietatea că înmulțirea sa cu vectorul oricărui marcaj accesibil dă același rezultat:

$$Y^T * M = Y^T * M_0;$$

M este marcajul care se obține din marcajul  $M_0$  după execuția tranzitiei t, iar Y reprezintă invariantul.

Determinarea unui invariant se face conform relațiilor:

$$y_1 * c_{11} + y_2 * c_{21} + \dots + y_n * c_{n1} = 0$$

$$y_1 * c_{12} + y_2 * c_{22} + \dots + y_n * c_{n2} = 0$$

...

$$y_1 * c_{1m} + y_2 * c_{2m} + \dots + y_n * c_{nm} = 0$$

 Sunt  $n$  necunoscute  $y_1, y_2, \dots, y_n$  și  $m$  ecuații. Dacă  $n$  este mai mare decât  $m$ , atunci unele elemente ale lui  $Y$  trebuie să ia valori arbitrate. În consecință, pentru o rețea Petri pot exista mai mulți invarianți.

Dacă  $M$  și  $M'$  sunt marcaje accesibile, atunci:

$$\begin{aligned} y_1 * M(p_1) + y_2 * M(p_2) + \dots + y_n * M(p_n) = \\ y_1 * M'(p_1) + y_2 * M'(p_2) + \dots + y_n * M'(p_n) = ct \end{aligned}$$

### 8.2.2 INVARIANTUL DE TIP T

Invarianți de tip T sunt vectori notați cu  $X$ , care verifică relațiile:

$$C * X = 0, X \geq 0 \Rightarrow \exists M_0$$

astfel încât

$$M_0[\sigma > M_0]$$

și

$$\sigma = X$$

(comportarea ciclică a marcajului).

Dacă se poate găsi un astfel de invariant  $X$ , atunci există un marcat  $M_0$  și o secvență de ecuații care aduce marcatul înapoi în starea inițială. Secvența de execuții este vectorul de numărare a execuțiilor și este egal cu  $X$ . Multimea locațiilor rețelei N care corespunde elementelor nenule ale unui invariant P, se numesc suportul invariantului Y. Multimea tranzițiilor rețelei N care corespund elementelor nenule ale unui invariant T se numește suportul invariantului X.

Un suport de invariant (adică o multime de locații, respectiv tranziții) se numește minimal dacă nici o submultime strictă a suportului nu reprezintă tot un suport (evident pentru un alt invariant).

Un invariant P, notat Y, se numește invariant P minimal, dacă nu există nici un alt invariant P, notat  $Y'$ , astfel încât  $Y \geq Y'$ . În mod similar se definește și invariantul T minimal.

Nu orice invariant minimal are suportul minimal (adică pot exista invarianți (P sau T) minimali având suporturi neminimale). Nu orice invariant (P sau T) cu suport minimal este minimal (adică pot exista invarianți cu suport minimal care nu sunt invarianți minimali).

Un invariant minimal cu suport minimal se numește invariant de bază, sau fundamental. Dacă matricea de incidență a rețelei are rangul r, atunci rețeaua are  $m - r$  invarianți P de bază și  $n - r$  invarianți T de bază.

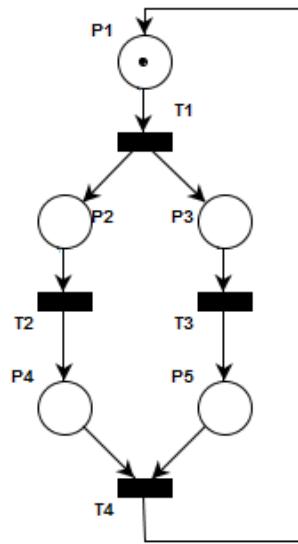


Figure 8.1: R.P. 1

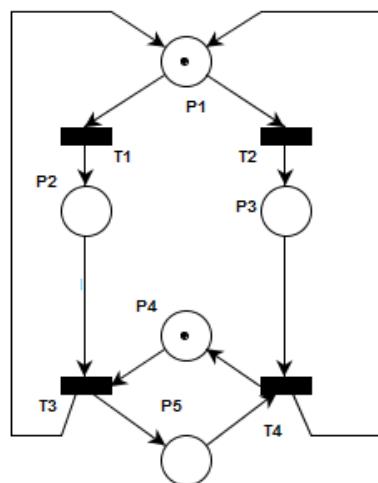


Figure 8.2: R.P. 2

### 8.3 DESFĂŞURAREA LUCRĂRII

1. Petru rețelele Petri reprezentate în figurile 8.1, 8.2, 8.3, 8.4, 8.5, 8.6 și 8.7 să se determine invarianții de bază de tip P și T. Analizați rezultatele obținute.

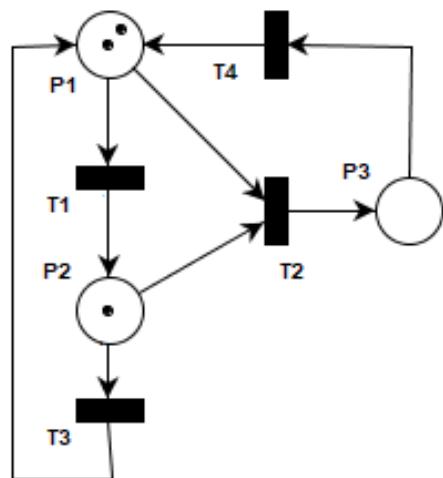


Figure 8.3: R.P. 3

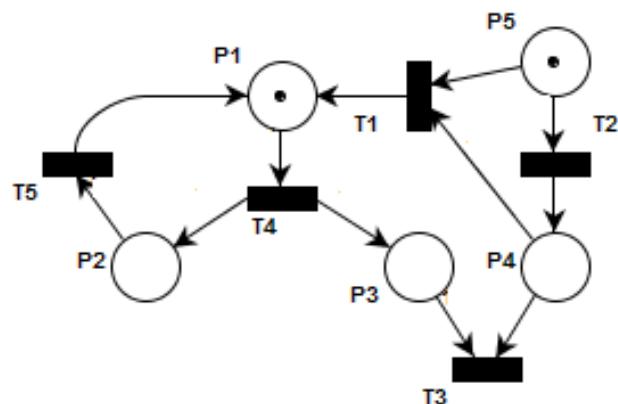


Figure 8.4: R.P. 4

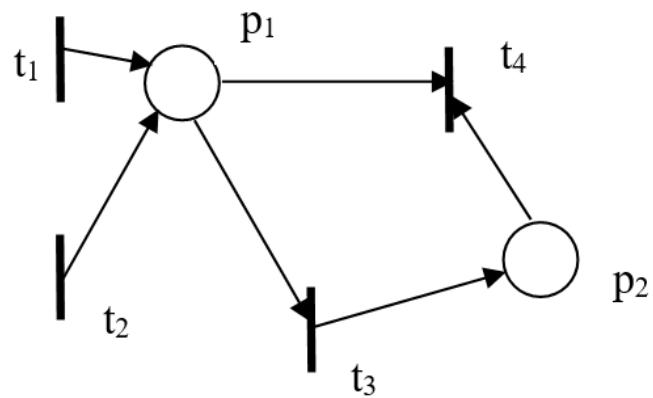


Figure 8.5: R.P. 5

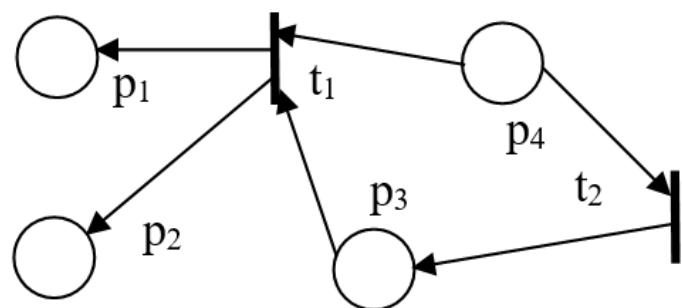


Figure 8.6: R.P. 6

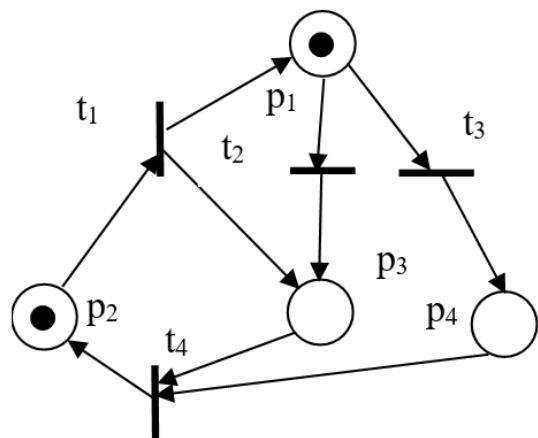


Figure 8.7: R.P. 7

## BIBLIOGRAFIE

- [1] Adina Aştilean. *Sisteme cu evenimente discrete, notițe curs.*
- [2] Christos G. Cassandras, Stéphane Lafortune. *Introduction to Discrete Event Systems, Second Edition*, ISBN-13: 978-0-387-33332-8, e-ISBN-13: 978-0-387-68612-7, 2008 Springer Science+Business Media, LLC.

# 9 SUB CLASE DE REȚELE PETRI - SIFOANE ȘI CAPCANE

---

## 9.1 OBIECTIVE



- Analiza structurală a rețelelor și subrețelelor Petri.
  - Sifoane;
  - Capcane.

## 9.2 CONSIDERAȚII TEORETICE

Sifoanele sunt blocaje structurale. Capcanele sunt subseturi de locații posibil de a fi recunoscute, generează subrețele specifice.



### Definiții:

1. Un sifon este un subset de locații pentru care setul tranzițiilor sale de intrare este conținut în setul tranzițiilor sale de ieșire este conținut în setul tranzițiilor sale de ieșire:  
 $S \subseteq P$  este un sifon  $\iff S^o \subseteq S^o$ .
2. O capcană este un subset de locații astfel încât setul tranzițiilor sale de ieșire este conținut în setul tranzițiilor sale de intrare:  
 $P \subseteq Q$  este o capcană  $\iff Q^o \subseteq S^o$ .

Sifonul și capcana sunt concepte inverse.

În general, blocajul este implicat de faptul că locațiile componente ale setului  $S$  nu pot fi încărcate cu jetoane, deoarece tranzițiile care ar putea face aceasta nu pot deveni executabile din cauză că au ca intrare tranziți din  $S$ .

Capcana se explică prin faptul că jetoanele aparținând locațiilor din setul  $Q$  nu mai pot fi extrase de către tranzițiile lor de ieșire, deoarece, execuția acestora implică încărcarea locațiilor care fac parte din setul  $Q$  (care sunt deja încărcate). Se poate spune despre

capcană că este opusă blocajului. Spre deosebire de blocaj în care nu se pot introduce jetoane (adică mulțimea Q nu își poate reduce numărul de jetoane).

Un sifon se numește sifon de bază, dacă nu poate fi reprezentat drept o reuniune de alte sifoane. Similar, o capcană se numește capcană de bază, dacă nu poate fi reprezentată drept reuniune de alte capcane.

Un sifon (capcană) se numește minimal (ă), dacă nu conține nici un alt sifon (nici o altă capcană). Sifoanele (capcanele) minime sunt și sifoane (capcane) de bază, dar nu toate sifoanele (capcanele) de bază sunt minime.

### 9.3 DESFĂȘURAREA LUCRĂRII

1. Identificați sifoanele și capcanele din subseturile rețelelor Petri reprezentate în figurile [9.1](#) și [9.2](#).
2. Pentru rețelele Petri din figurile [9.3](#), [9.4](#), [9.5](#) și [9.6](#), identificați sifoanele și capcanele, examinând, pe rând, submulțimile de locații cu trei și respectiv patru elemente. Indicați sifoanele (capcanele) minime. Analizați rezultatele obținute.

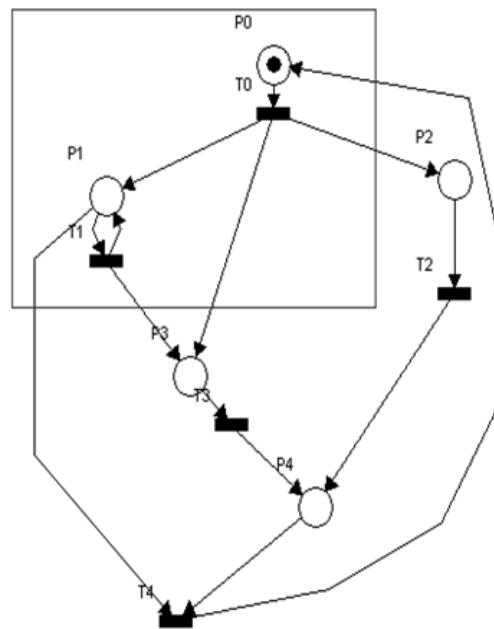


Figure 9.1: R.P. 9.1

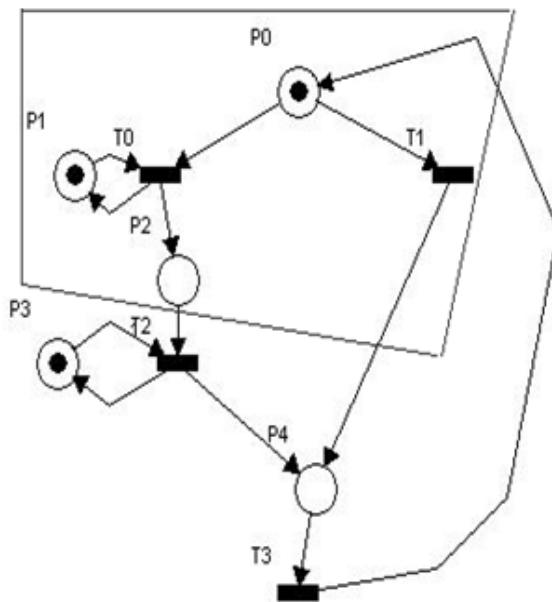


Figure 9.2: R.P. 9.2

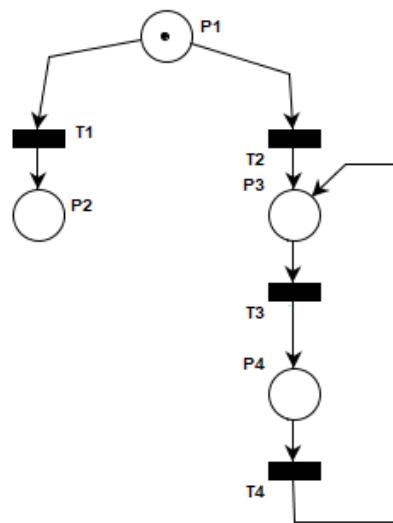


Figure 9.3: R.P. 9.3

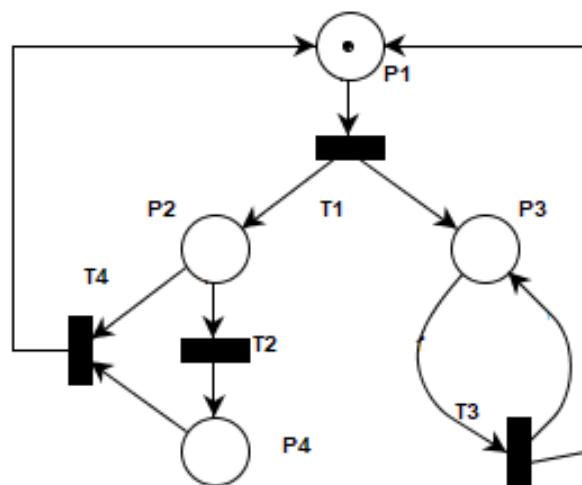


Figure 9.4: R.P. 9.4

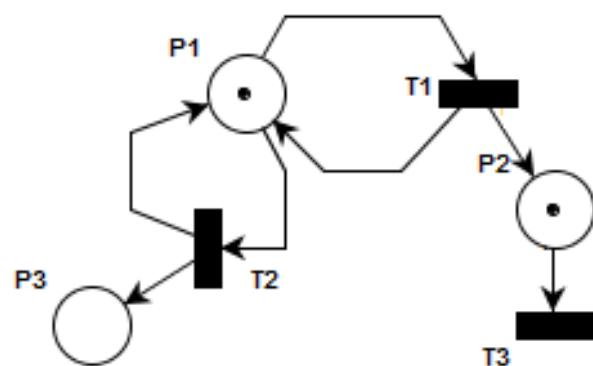


Figure 9.5: R.P. 9.5

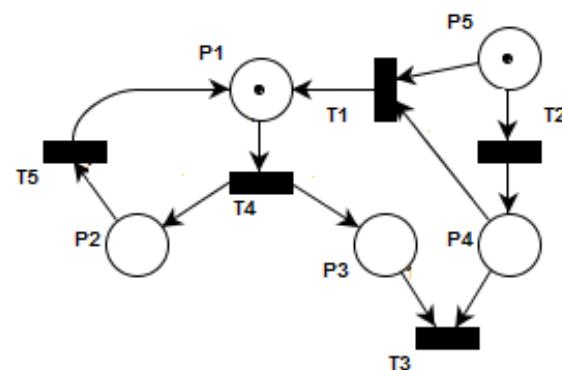


Figure 9.6: R.P. 9.6

## BIBLIOGRAFIE

- [1] Adina Aştilean. *Sisteme cu evenimente discrete, notițe curs.*
- [2] Christos G. Cassandras, Stéphane Lafortune. *Introduction to Discrete Event Systems, Second Edition*, ISBN-13: 978-0-387-33332-8, e-ISBN-13: 978-0-387-68612-7, 2008 Springer Science+Business Media, LLC.

# 10 MODELAREA SED FOLOSIND REȚELE PETRI

---

## 10.1 OBIECTIVE



- Construirea unor modele bazate pe rețele Petri;
- Transformarea unor rețele Petri cu capacitate finite în rețele cu capacitate infinite.

## 10.2 DESFĂȘURAREA LUCRĂRII

1. Se consideră procesul reprezentat în figura 10.1.

Procesul este format din:

- 4 depozite -  $D_1, D_2, D_3$  (capacitate infinită) și  $D_4$  de capacitate finită ( $= 10$ );
  - o mașină M care asamblează produse luate din depozitele  $D_1, D_2$  și  $D_3$  (câte o piesă din fiecare depozit odată);
  - un robot (resursă partajată)
    - încarcă piesele pe mașina M;
    - descarcă piesele de pe mașina M;
    - extrage câte 2 piese din depozitul  $D_4$  odată.
- a) Să se construiască modelul bazat pe rețele Petri pentru acest proces;
  - b) Transformați rețeaua Petri cu capacitate finite într-o rețea Petri cu capacitate infinite;
  - c) Verificați proprietățile de mărginire, reversibilitate și viabilitate pentru rețelele Petri construite.

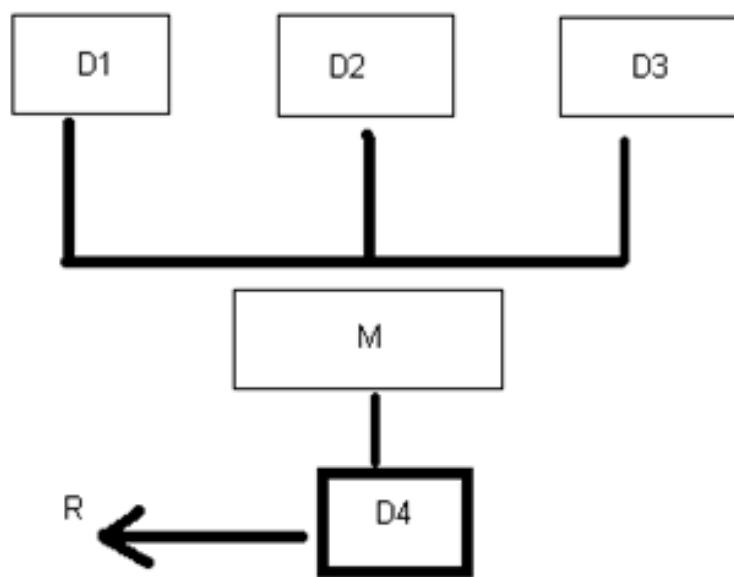


Figure 10.1: Procesul 1

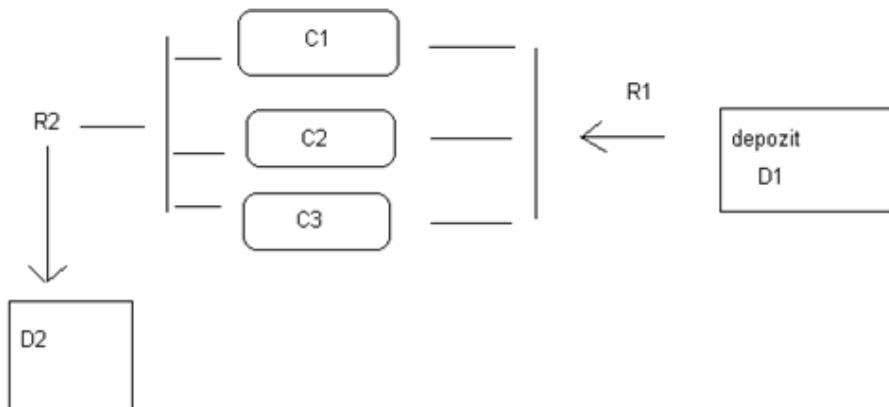


Figure 10.2: Procesul 2

2. Se consideră procesul reprezentat în figura 10.2.

Procesul este format din:

- două depozite:  $D_1$  (capacitate infinită) și  $D_2$  (capacitate finită - 10 de piese);
- 2 roboți:  $R_1$  (preia piese din  $D_1$  și introduce piese în stivele  $C_1, C_2, C_3$ ) și  $R_2$  (preia piese din  $C_1, C_2, C_3$  după ce sunt pline și le depune în  $D_2$ );

- 3 stive:  $C_1$ ,  $C_2$  și  $C_3$  - de capacitate = 2;
  - Robotul  $R_1$  preia bucată cu bucată piesele din depozitul  $D_1$ .
- Să se construiască modelul bazat pe rețele Petri pentru acest proces;
  - Transformați rețeaua Petri cu capacitate finite într-o rețea Petri cu capacitate infinite;
  - Verificați proprietățile de mărginire, reversibilitate și viabilitate pentru rețelele Petri construite.
3. Se consideră procesul reprezentat în figura 10.3.

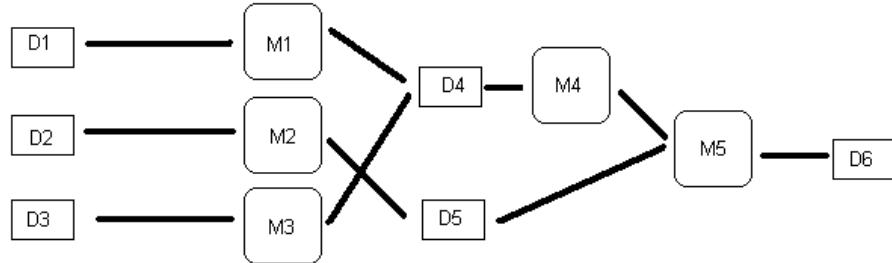


Figure 10.3: Procesul 3

Procesul are următoarea structură:

- $D_1$  -  $D_6$  depozite;
  - $D_1$  -  $D_3$  și  $D_6$  capacitate infinite;
  - $D_4$  - capacitate de 5 piese;
  - $D_5$  - capacitate de 3 piese;
  - $M_1$  -  $M_5$  mașini;
  - Mașinile  $M_1$  -  $M_3$  se alimentează cu piese din depozitele  $D_1$  -  $D_3$ .  $M_1$  și  $M_3$  depun piese prelucrate în depozitul  $D_4$ . Mașina  $M_2$  depune în  $D_5$ . Mașina  $M_4$  preia piesele din depozitul  $D_4$ .
  - $M_5$  prelucrează piese doar când depozitele anterioare sunt pline ( $D_4$ ,  $D_5$ ), depune piesele asamblate în depozitul  $D_6$ .
- Să se construiască modelul bazat pe rețele Petri pentru acest proces;
  - Transformați rețeaua Petri cu capacitate finite într-o rețea Petri cu capacitate infinite;
  - Verificați proprietățile de mărginire, reversibilitate și viabilitate pentru rețelele Petri construite.

## BIBLIOGRAFIE

- [1] Adina Aştilean. *Sisteme cu evenimente discrete, notițe curs.*
- [2] Christos G. Cassandras, Stéphane Lafortune. *Introduction to Discrete Event Systems, Second Edition*, ISBN-13: 978-0-387-33332-8, e-ISBN-13: 978-0-387-68612-7, 2008 Springer Science+Business Media, LLC.



# 11 MODELAREA SED FOLOSIND REȚELE PETRI - REȚELE PETRI STOCHASTICE

---

## 11.1 OBIECTIVE



- Definirea rețelelor Petri stochastice;
- Analiza rețelelor Petri stochastice;
- Construirea lanțului Markov.

## 11.2 CONSIDERAȚII TEORETICE

În cazul R.P. stochastice se presupune că întârzierea asociată tranziției  $t_i$ , este o variabilă nenegativă X, cu o funcție de distribuție dată. Se utilizează două feluri de distribuții: *exponențiale*, pentru sistemele în care timpul are un caracter continuu și *geometrice*, pentru sistemele în care timpul este discret. Aceste distribuții sunt considerate fără memorie, adică la un moment dat mai multe tranziții sunt admisibile dintr-un maraj oarecare, execuția uneia dintre ele nu va modifica distribuțiile (șansele) de execuție ale celorlalte.

O R.P. stochastică este o R.P. în care fiecare tranziție are asociată o variabilă aleatoare de distribuție exponențială, care exprimă întârzierea de la admisibilitate până la execuția tranziției. Se poate arăta că datorită lipsei proprietății de memorie a distribuției exponențiale a întârzierilor execuțiilor, graful de realizabilitate al unei R.P. stochastice mărginită este izomorf cu un lanț Markov finit. Lanțul Markov al unei rețele de tip R.P.S. se poate obține din graful de realizabilitate al rețelei R.P.S. =  $(N, M_0)$  luând setul marajelor realizabile,  $R(M_0)$ , ca spațiu al stărilor. Rata de tranziție de la starea  $M_i$  la starea  $M_j$  este dată de:

$$c_{ij} = \lambda'_i$$

unde  $\lambda'_i$  este rata de execuție (posibil dependentă de maraj) a tranziției  $t_i$ , transformând marajul  $M_i$  în  $M_j$ . Dacă există două sau mai multe tranziții  $t_{i1}, t_{i2}, \dots$  care determină schimbarea marajului din  $M_i$  în  $M_j$ , rata de tranziție este dată de relația:

$$c_{ij} = \lambda'_{i1} + \lambda'_{i2} + \dots$$

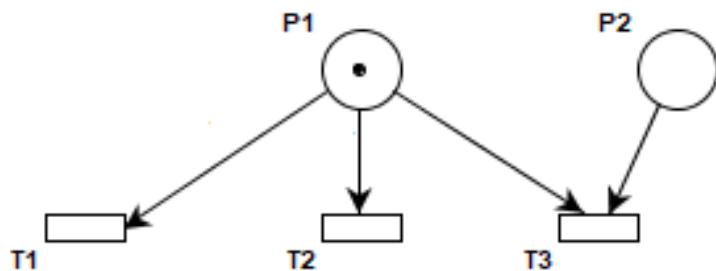


Figure 11.1: Rețeaua Petri stochastică 1

$$c_{ij} = 0$$

, dacă nu există nici o tranziție care să transforme marcajul  $M_i$  în  $M_j$ .

Matricea pătrată  $C$  având elementele  $c_{ij}$ , de dimensiune  $s \times s$ , unde  $s$  este cardinalul setului marcajelor realizabile  $R(M_0)$ , se numește matricea ratei de tranziție.

### 11.2.1 ORDINEA DE EXECUȚIE A TRANZIȚIILOR ÎNTR-O REȚEA PETRI STOCHASTICĂ

Se consideră rețeaua Petri din 11.1. La momentul  $t_1$  tranzițiile  $t_1$  și  $t_2$  sunt validate, în timp ce tranziția  $t_3$  nu este executabilă. Variabilele aleatoare  $D_1$  și  $D_2$  asociate tranzițiilor  $t_1$  și  $t_2$  la momentul  $t_1$  valorile  $D_1 = 4$  și  $D_2 = 6,4$ . Se calculează  $\min(D_1, D_2)$ , tranziția cu timpul minim, adică  $t_1$ , urmând să fie executată prima, după 4 unități de timp (U.T.). Jetonul din locația  $p_1$  nu va fi însă rezervat pentru execuția tranziției  $t_1$ . La momentul  $t_2 = t_1 + 1$ , un jeton va sosi în locația  $p_2$ . Se extrag din nou valorile pentru variabilele  $D_1$ ,  $D_2$  și  $D_3$ . Valorile extrase considerate sunt:  $D'_1 = 4,2$  U.T.;  $d_2 = 3$ ;  $d_3 = 1,8$ . Se calculează  $\min(D_1, D_2, D_3) = 1,8$  U.T. În consecință, tranziția care se va executa prima va fi  $t_3$ . Execuția acesteia va avea loc la momentul  $t_2 = D_3 = t_1 + 1 = D_3 = t_1 + 2,8$  U.T.



**Observație:** Jetoanele nu sunt rezervate pentru execuția tranzițiilor în rețelele Petri stochastice.

### 11.2.2 ANALIZA REȚELELOR PETRI STOCHASTICE

Se definesc urmatoarele sintagme:

- **stare (S<sub>i</sub>)** - situația în care se află un sistem determinată de structura sa, de condițiile exterioare etc. și definită prin anumite mărimi sau parametri. Pentru fiecare sistem pot exista  $S_i$ ,  $i = 1, \dots, n$  stări în care poate trece acesta pe timpul procesului, din care "starea inițială" ( $S_0$ ) existentă la începutul procesului și "stări reversibile", respectiv

stările în care trece sistemul ulterior. Dacă sistemul trece într-o stare din care nu mai poate trece în alta, atunci aceasta este denumită "stare absorbantă";

- **probabilitate de stare (Pi)** - probabilitatea ce reflectă posibilitatea sistemului de a se menține într-o anumită stare "i" ( $P_i$ ,  $i = 1, \dots, n$ );
- **probabilitatea de trecere (Pij)** - probabilitatea ce reflectă posibilitatea sistemului de a trece dintr-o stare în alta pe timpul procesului, respectiv din starea "i" în starea "j". Această probabilitate nu se determină în cadrul proceselor cu evoluție continuă, deoarece pentru un anumit moment determinat acestea sunt nule (la fel ca și probabilitățile oricărei valori individuale ale funcției continuue a unei mărimi aleatoare);
- **densitatea probabilităților de trecere ( $\lambda_{ij}$ )** - este caracteristică proceselor continue și reprezintă limita raportului dintre probabilitățile de trecere  $P_{ij}$  ale sistemului în intervalul de timp  $\Delta t$  din starea  $S_i$  în starea  $S_j$  raportate la acest interval, adică:

$$\lambda_{ij} = \lim_{\Delta t \rightarrow 0} \frac{P_{ij}(\Delta t)}{\Delta t}$$

unde:

- $P_{ij}(\Delta t)$  reprezintă probabilitatea ca sistemul aflat în momentul  $t$  în starea  $S_i$ , să treacă din aceasta în starea  $S_j$ , în intervalul de timp  $\Delta t$ .

Densitățile probabilităților de trecere se determină numai pentru stările la care  $j \neq i$ . Pentru un interval foarte mic putem considera  $P_{ij}(\Delta t) \approx \lambda_{ij}(\Delta t)$ . Dacă densitatea probabilității de trecere ( $\lambda_{ij}$ ) nu depinde de timp (adică în ce moment începe intervalul  $\Delta t$ ), atunci procesul Markovian se numește omogen și neomogen în caz contrar.

Spațiul stărilor unei rețele Petri stochastice este determinat de setul stărilor realizabile. Execuția unei tranziții se realizează în concordanță cu timpul necesar pentru ca sistemul să treacă dintr-o stare în alta. Datorită faptului că funcția de distribuție exponentială este fără memorie, se poate demonstra că graful de realizare al unei rețele Petri stochastice mărginită este izomorf cu un lanț Markov finit. Lanțul Markov al unei rețele Petri stochastice poate fi obținut utilizând graful de realizare al unei rețele Petri, după cum urmează: spațiul stărilor lanțului Markov este format din setul stărilor realizabile  $R(M_0)$  și rata tranziției de la starea  $M_i$  la starea  $M_j$  este dată de rata de execuție a tranziției care transformă marcajul  $M_i$  în marcajul  $M_j$ . Dacă există două sau mai multe tranziții care transformă marcajul  $M_i$  în marcajul  $M_j$ , notează cu  $t_1, t_2, \dots$ , atunci rata de execuție se va calcula cu formula:

$$\lambda = \lambda_1 + \lambda_2 + \dots$$

Elementul de generare al procesului Markov asociat cu o rețea Petri stochastică este o matrice pătrată A, de dimensiune L X L, unde L este egal cu numărul de stări ale lanțului

Markov. Matricea A este denumită *matricea de tranziție a stării*. Elementele acestei matrici sunt împărțite în două categorii: cele care aparțin și cele care nu aparțin diagonalei principale. Elementele care nu aparțin diagonalei principale, notate cu  $\lambda_{rs}$ , se determină însumând ponderea arcelor care conectează nodurile (r și s) care reprezintă stările. Elementele diagonalei principale,  $\lambda_{rr}$  se determină calculând suma complementelor față de zero al celorlalte elemente de pe linia r, adică:

$$\lambda_{rr} = -\sum \lambda_{rk}.$$

Se notează cu  $P_r^*$  un vector linie de dimensiune L. Elementele acestui vector  $P_r^*(M_k)$  sau  $P_r^*(k)$  reprezintă probabilitatea ca sistemul să fie în starea corespunzătoare marcajului  $M_k$ . Matricea ratei de tranziție este utilizată pentru a calcula probabilitatea ca sistemul să se afle intr-o stare  $M_k$ . În acest scop se calculează soluția sistemului:

$$P_r^* * A = 0; \sum_{k=1}^L P_r^*(k) = 1.$$

Frecvențele medii de execuție ale unei tranziții  $t_j \in T$  se calculează cu relația:

$$f_j^* = \sum \mu_j(k) P_r^*(k);$$

considerând toate marcajele din care tranziția este executabilă. Valorile medii ale marcajelor în locațiile  $P_i \in P$ , pot fi calculate conform relației:

$$M^*(P_i) = \sum_{k=1}^L M_k(P_i) * P_r^*(k);$$

### 11.3 DESFĂȘURAREA LUCRĂRII

1. Pentru rețeaua Petri din figura 11.2 să se construiască lanțul Markov.
2. Pentru rețeaua Petri din figura 11.3 să se construiască lanțul Markov.
3. Fie un server care deservește două tipuri de clienți în paralel, modelat prin rețele Petri stochastice. Servirea primului tip de clienți este modelată de o subrețea formată din nodurile  $p_1, t_1, p_2, t_2$ , iar servirea celui de-al doilea tip de clienți de o subrețea formată din nodurile  $p_1, t_3, p_3, t_4$ . Durata de timp asociată tranziției  $t_i$ ,  $i = 1, \dots, 4$ , are distribuție exponențială cu rata  $\lambda_i$  [U.T. -1]. Se consideră următoarele valori numerice:  $\lambda_1 = \lambda_4 = 1$ ,  $\lambda_2 = \lambda_3 = 10$ .

Să se construiască rețeaua Petri stochastică și lanțul Markov.

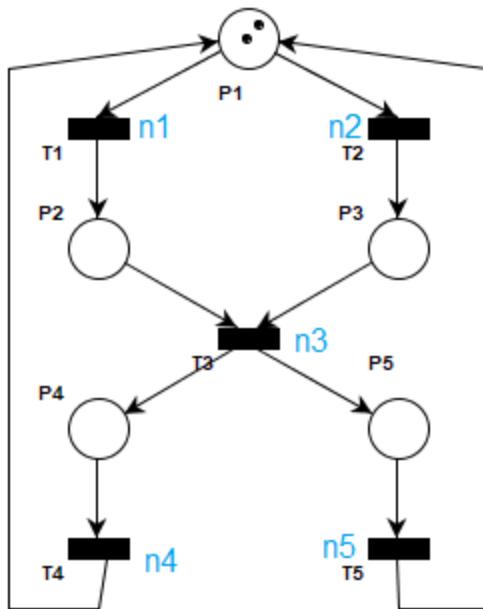


Figure 11.2: Proces 2 - RPS

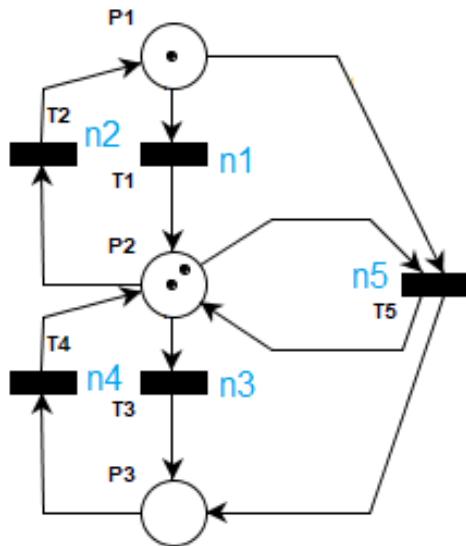


Figure 11.3: Proces 3 - RPS

4. Fie un sistem de fabricație compus dintr-o mașină M și un depozit D ca în figura 11.4, capacitatea sistemului (mașină + depozit) fiind limitată la două piese.

Dacă există loc liber în depozit, piesele intră în depozitul D după o distribuție expo-

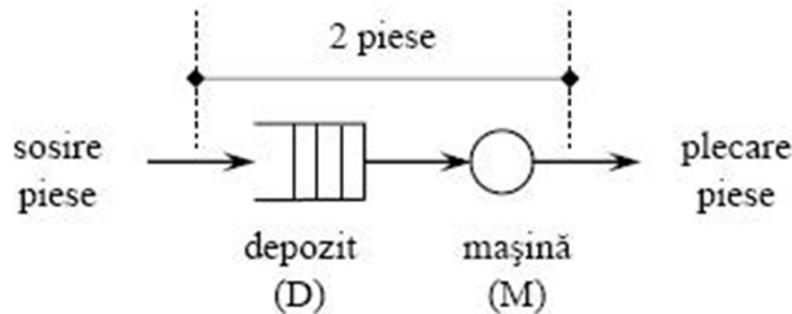


Figure 11.4: Proces 4

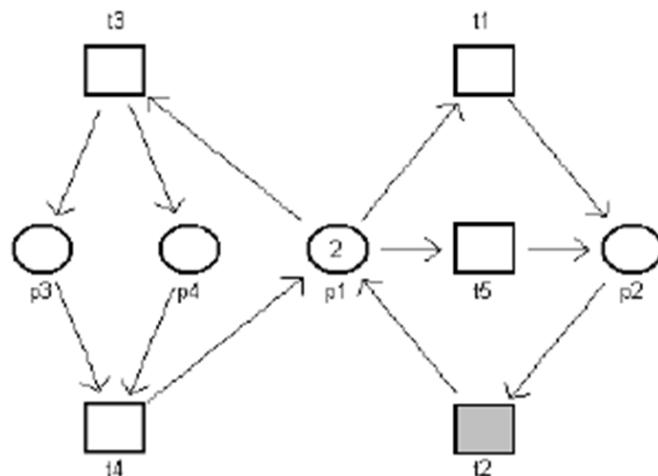


Figure 11.5: Proces 5 - RPS

nențială cu rata  $\lambda$ . Ori de câte ori mașina M este liberă și există o piesă în depozit, aceasta este preluată imediat pe mașină. Mașina M prelucrează piesele după o distribuție exponențială cu rata  $\mu$ .

Să se construiască lanțul Markov.

- Se consideră RPS din figura 11.5, unde: tranziția  $t_2$  se execută cu o rată dependentă de marcajul poziției predecesor, adică  $m_2\lambda_2$ , unde  $m_2$  reprezintă marcajul curent al poziției  $p_2$ . Ratele de execuție corespunzătoare tranzițiilor  $t_1$ ,  $t_2$ ,  $t_3$ ,  $t_4$  și  $t_5$ , notate  $\lambda_1$ ,  $\lambda_2$ ,  $\lambda_3$ ,  $\lambda_4$  și  $\lambda_5$ .  $\lambda$ , sunt independente de marcajul pozițiilor predecesor având următoarele valori numerice:  $\lambda_1 = \lambda_5 = 0.5$  [s-1] și  $\lambda_2 = \lambda_3 = \lambda_4 = 1$  [s-1].

Să se construiască lanțul Markov.

## BIBLIOGRAFIE

- [1] Adina Aştilean. *Sisteme cu evenimente discrete, notițe curs.*
- [2] Christos G. Cassandras, Stéphane Lafortune. *Introduction to Discrete Event Systems, Second Edition*, ISBN-13: 978-0-387-33332-8, e-ISBN-13: 978-0-387-68612-7, 2008 Springer Science+Business Media, LLC.