International Humanitarian University
http://mgu.edu.ua

Sleptsov Net Computing

Dmitry Zaitsev
http://member.acm.org/~daze

Write Programs

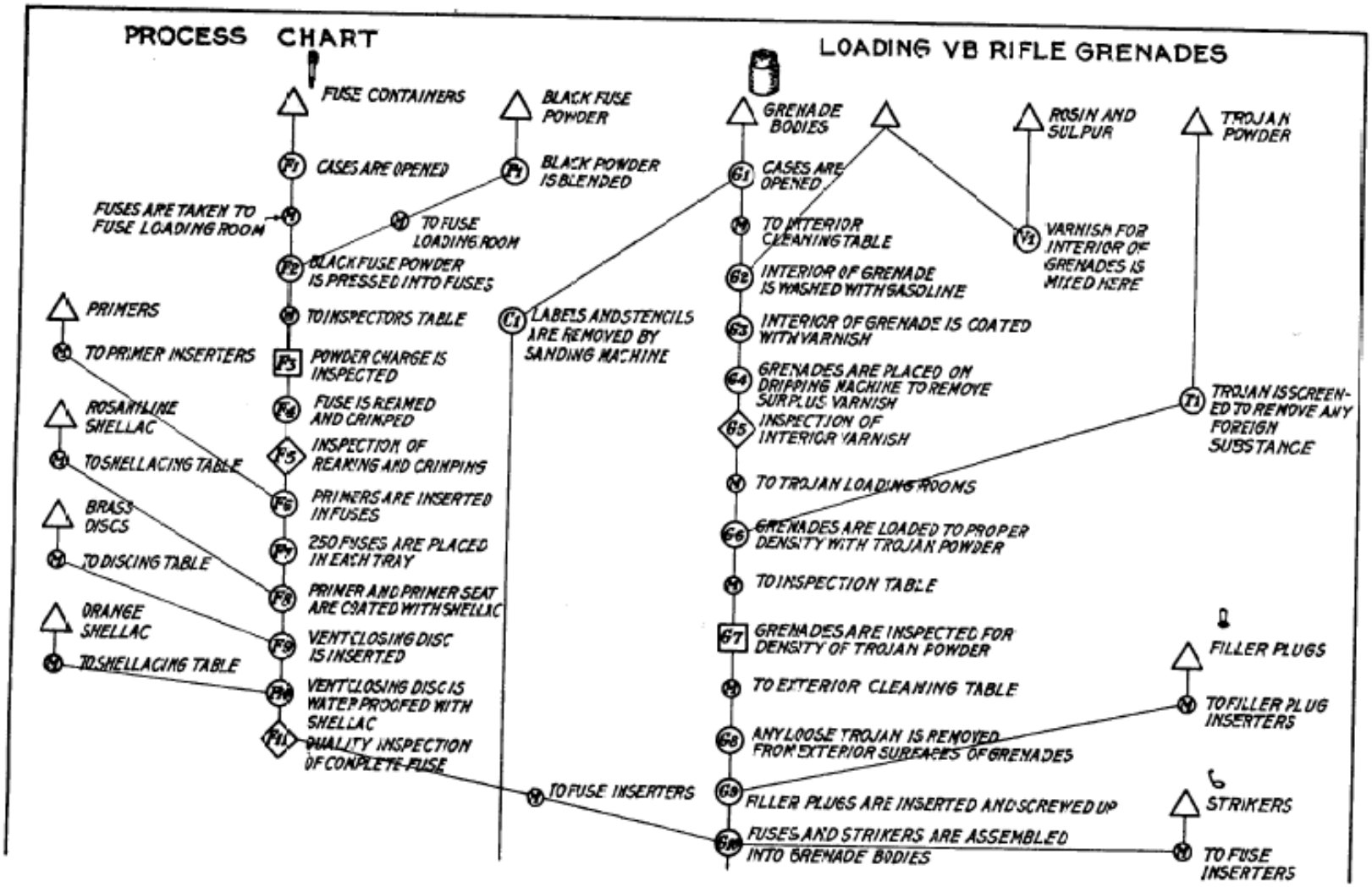
or

Draw Programs?

Flow charts

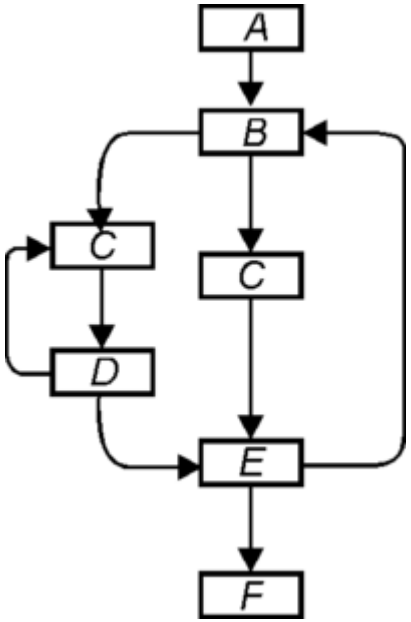
- Process Charts, Frank and Lillian Gilbreth, 1921
- ASME Standard: Operation and Flow Process Charts, 1947
- Planning and coding of problems for an electronic computing instrument, Part II, Volume 1, 1947, Herman Goldstine and John von Neumann

Frank and Lillian Gilbreth example



Program schemata

Yanov, 1958



Martiniuk, 1961

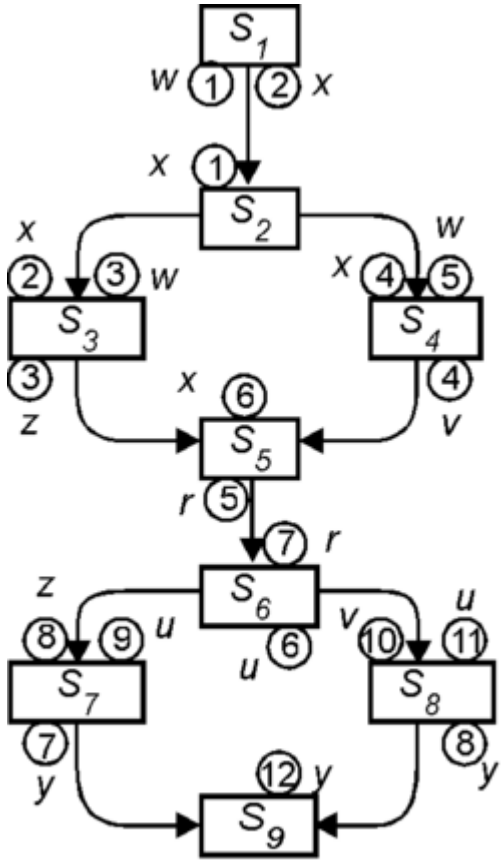


Схема Лаврова

Lavrov, 1961

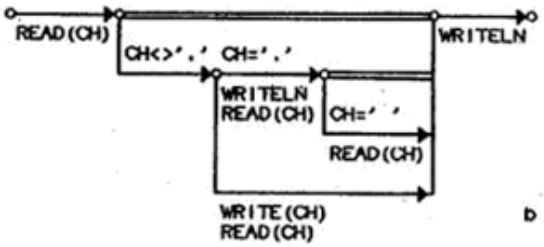
Functional equivalence and optimization, Lyapunov A.A., Ershov, 1950-1970

R-technology of programming

```
PROGRAM PRINID(INPUT,OUTPUT);
VAR CH:CHAR;
BEGIN
   READ(CH);
   WHILE CH<>'.' DO
      IF CH=','
         THEN
            BEGIN
               WRITELN;
               READ(CH)
               WHILE CH=' ' DO
                  READ(CH)
            END
         ELSE
            BEGIN
               WRITE(CH);
               READ(CH)
            END;
      WRITELN
   END.
```

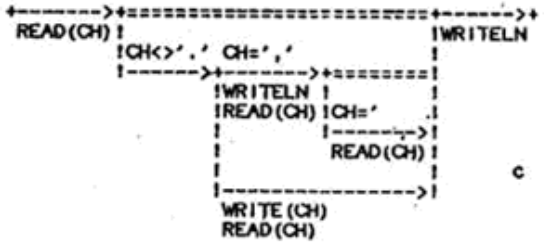
a

```
PROGRAM PRINID(INPUT,OUTPUT);
   VAR CH:CHAR;
```

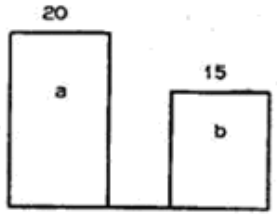


b

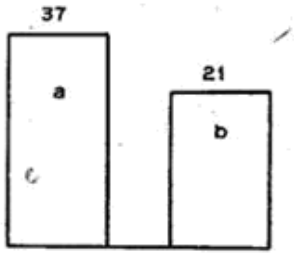
```
PROGRAM PRINID(INPUT,OUTPUT);
   VAR(CH);
```



c



Число строк на экране дисплея или на листинге



Число байтов памяти (по управляющим структурам программы) или скорость ввода

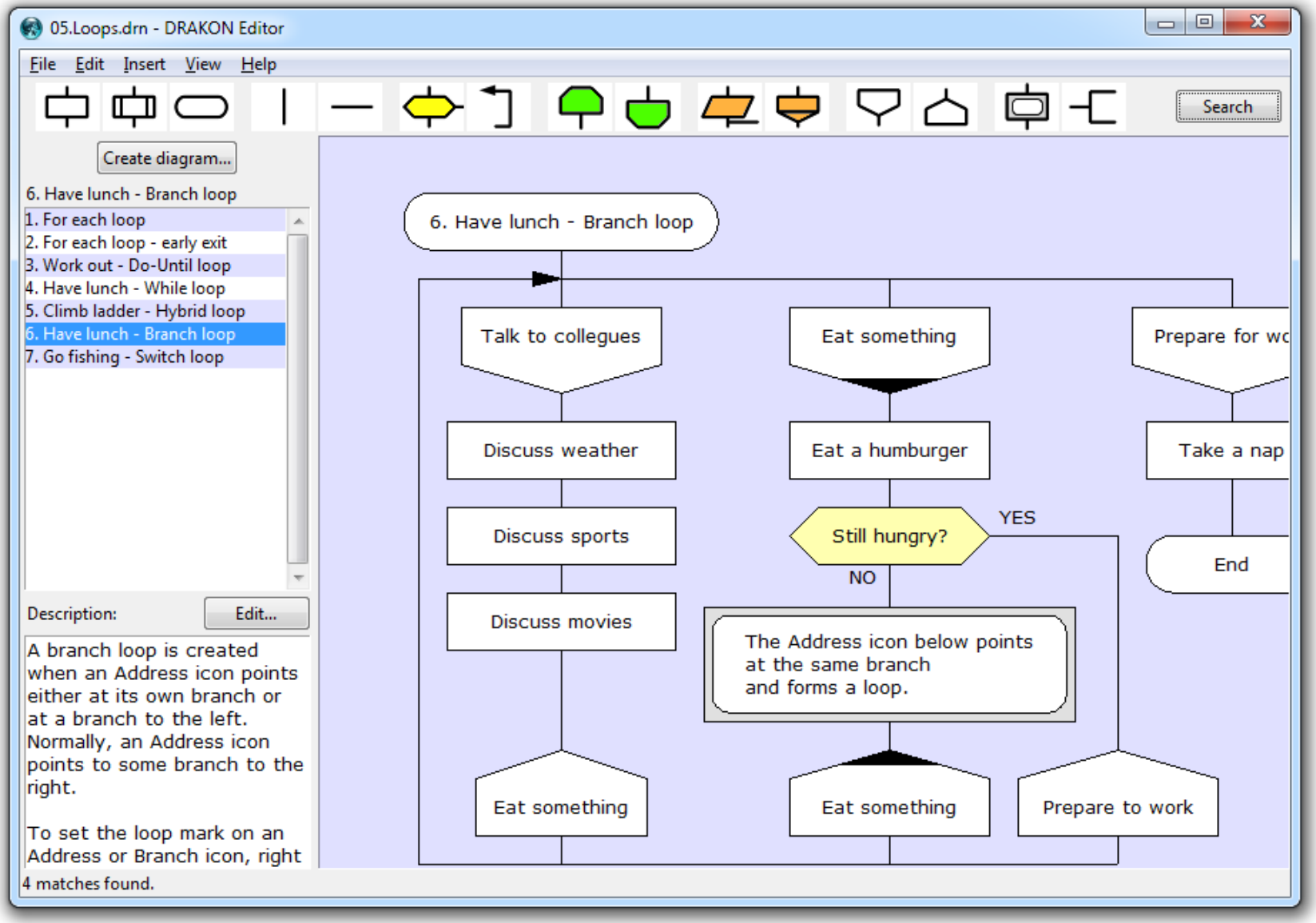
d

Ukraine,
V.M. Glushkov,
I.V. Velbitsky,
1970-1990

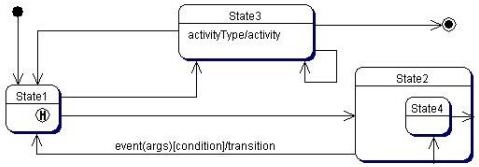
Modern Visual Programming

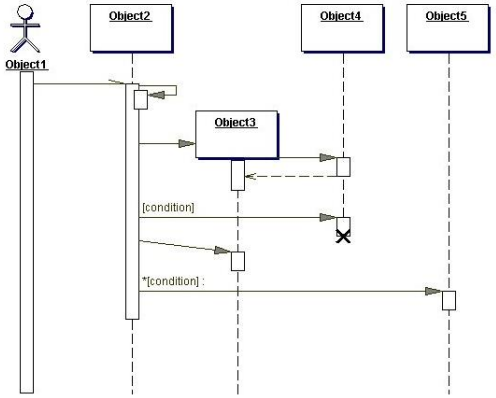
- DRAKON - the Buran space project
- Microsoft Visual Programming Language, MVPL
- Scratch for Android
- Node-RED
- Ardublock
- DGLux5
- AT&T Flow Designer
- ReactiveBlocks

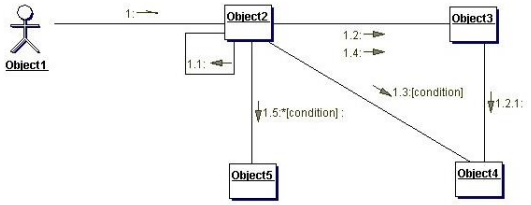
DRAKON

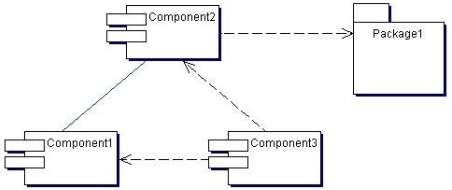

05.Loops.drn - DRAKON Editor
File Edit Insert View Help
Search
Create diagram...
6. Have lunch - Branch loop
1. For each loop
2. For each loop - early exit
3. Work out - Do-Until loop
4. Have lunch - While loop
5. Climb ladder - Hybrid loop
6. Have lunch - Branch loop
7. Go fishing - Switch loop
Description:
Edit...
A branch loop is created when an Address icon points either at its own branch or at a branch to the left. Normally, an Address icon points to some branch to the right.
To set the loop mark on an Address or Branch icon, right
4 matches found.
6. Have lunch - Branch loop
Talk to collegues
Eat something
Prepare for wo
Discuss weather
Eat a humburger
Take a nap
Discuss sports
Still hungry?
YES
End
NO
Discuss movies
The Address icon below points at the same branch and forms a loop.
Eat something
Eat something
Prepare to work

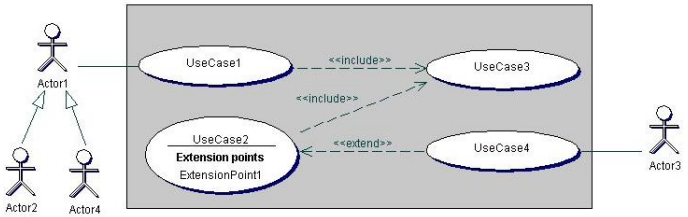
UML

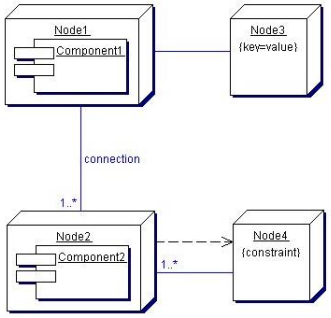

State1
State3
activityType/activity
State2
State4
event(args)[condition]/transition

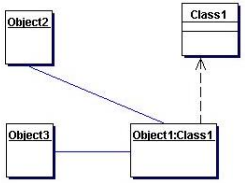

Object1
Object2
Object4
Object5
Object3
[condition]
*[condition] :

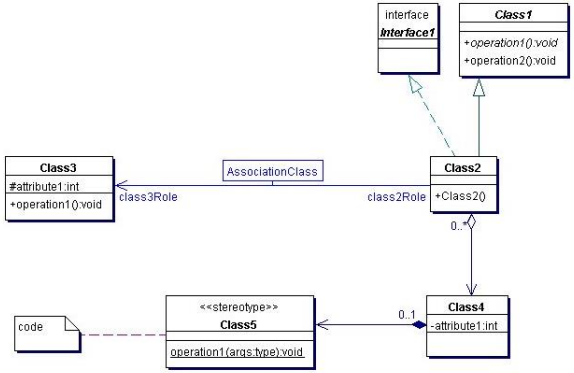

Object1
1:
Object2
Object3
1.1:
1.2:
1.4:
1.3:[condition]
1.2.1:
1.5.*[condition] :
Object5
Object4

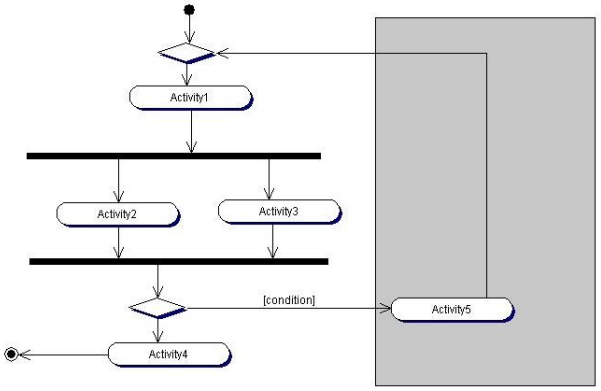

Component2
Package1
Component1
Component3


Actor1
UseCase1
<<include>>
UseCase3
<<include>>
UseCase2
Extension points
ExtensionPoint1
<<extend>>
UseCase4
Actor3
Actor2
Actor4

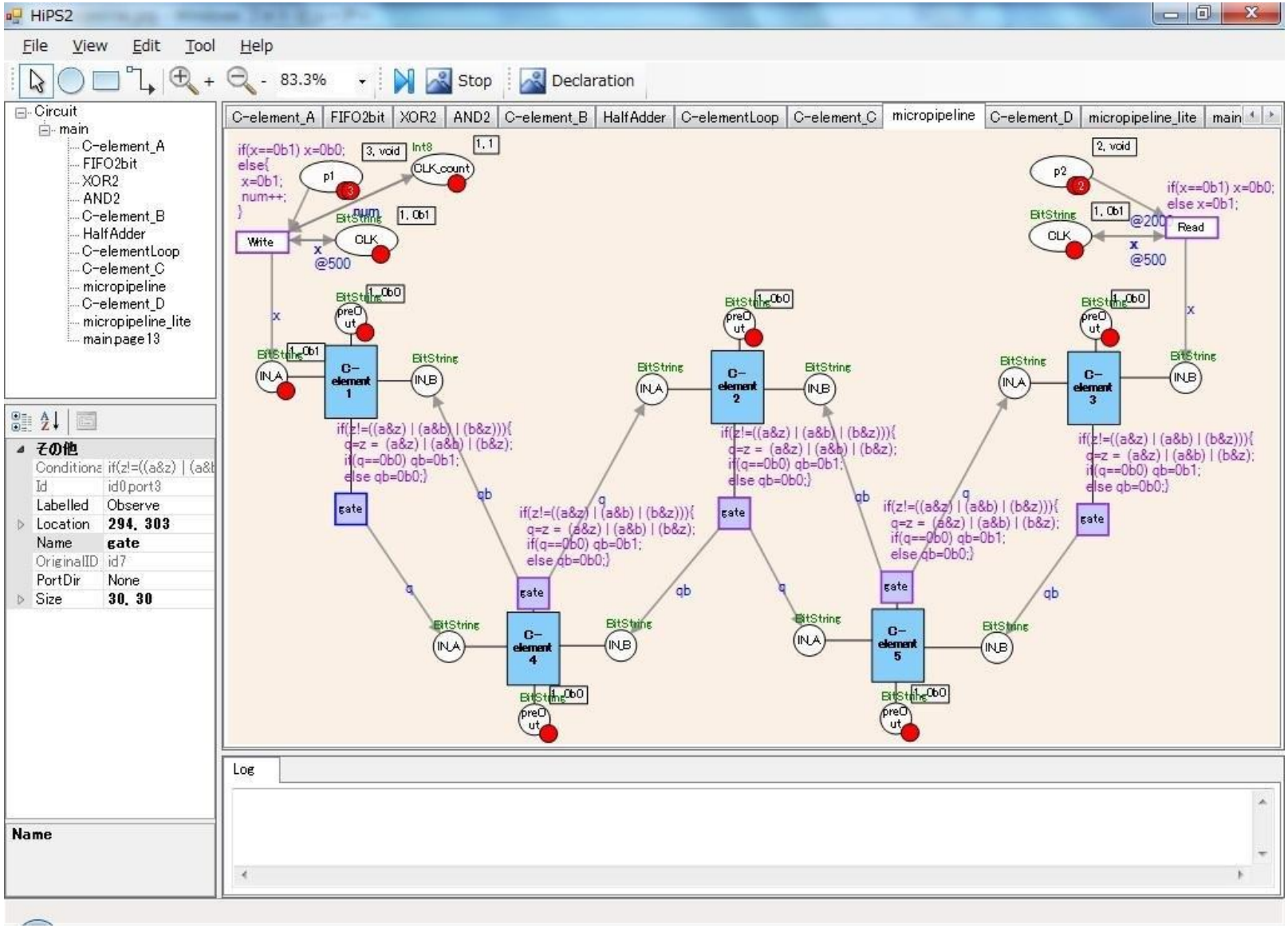

Node1
Component1
Node3
{key=value}
connection
1..*
Node2
Component2
Node4
{constraint}
1..*


Object2
Class1
Object3
Object1:Class1


interface
Interface1
Class1
+operation1():void
+operation2():void
Class3
#attribute1:int
+operation1():void
AssociationClass
class3Role
class2Role
Class2
+Class2()
0..*
<<stereotype>>
Class5
operation1(args:type):void
code
0..1
Class4
-attribute1:int


Activity1
Activity2
Activity3
[condition]
Activity5
Activity4

Programming on Petri nets



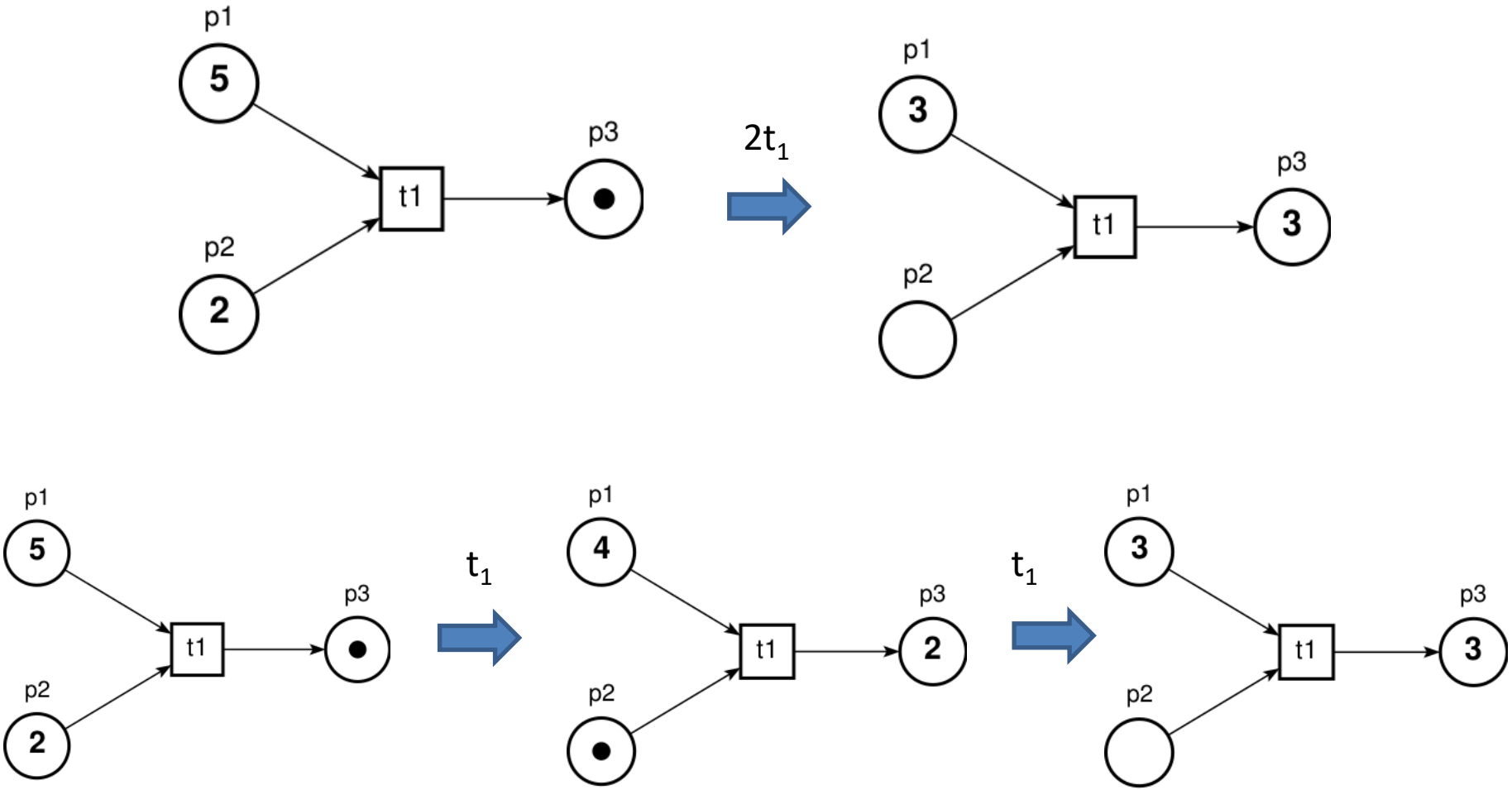
Path to a uniform concept

- Textual programming
- Graphs loaded by textual language
- **Pure graphical programming - nothing save graphs**
- Inhibitor Sleptsov net – fast universal language of concurrent programming
- Massively parallel computations
- Fine granulation
- Computing memory implementation

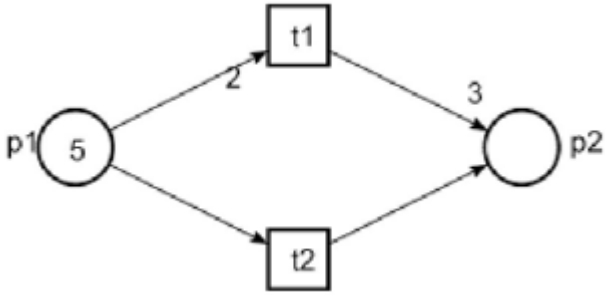
Transition firing strategy

- Petri
  - a single transition at a step
- Salwicki
  - the maximal firing strategy
- Sleptsov
  - the multiple firing strategy

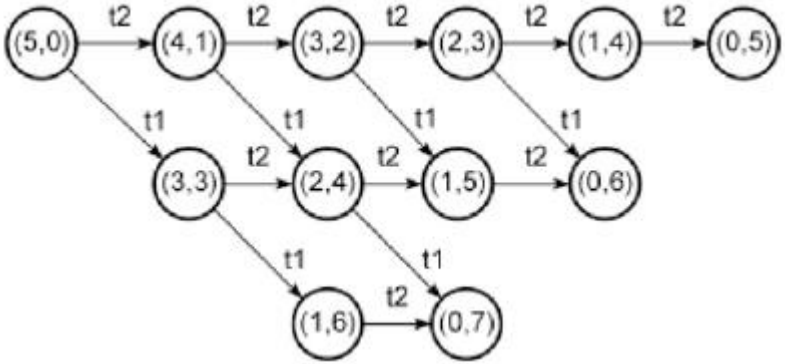
Sleptsov net vs Petri net



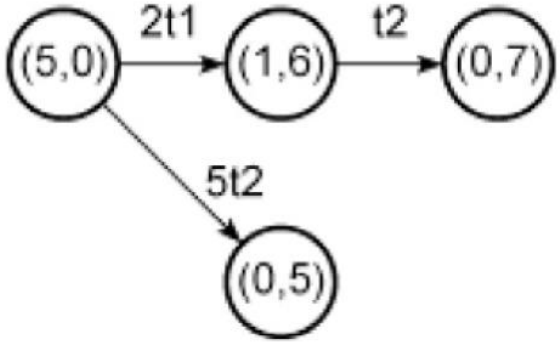
Sleptsov Net – Multiple Firing



Reachability graphs



Petri net



Sleptsov net

Sleptsov Nets Run Fast

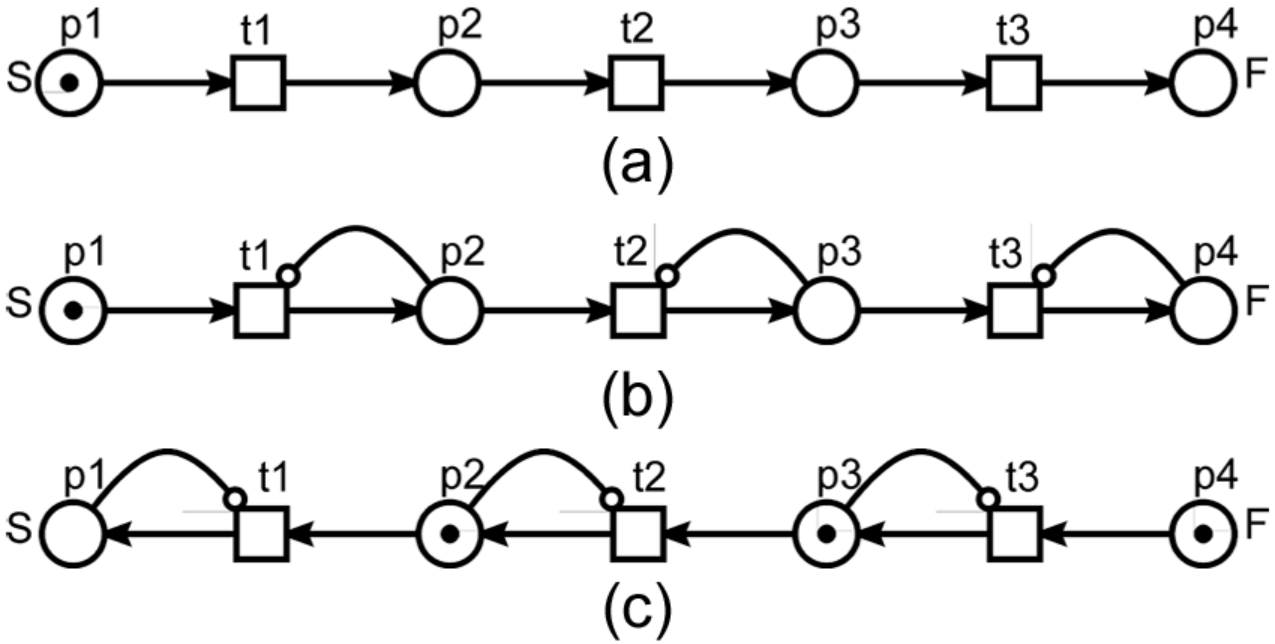
$$x \succ y = \begin{cases} x/y, & \text{if } y > 0 \\ 0, & \text{if } y = -1, x > 0 \\ \infty, & \text{if } y = -1, x = 0. \end{cases}$$

$$v_i = v(t_i) = \min_j \left(\mu_j \succ w^-_{j,i} \right), 1 \le j \le m, w^-_{j,i} \neq 0$$

COMPARING TIME COMPLEXITIES OF OPERATIONS (LINEAR SCALE - NUMBER OF STEPS)

Operation	PN	SN
CLEAN	$x+2$	2
MOVE	$x+2$	2
COPY	$2 \cdot x+3$	4
ADD	$x+y+2$	3
SUB	$\max(x, y)+3$	3
GT	$\max(x, y)+3$	4
MUL	$y \cdot (2 \cdot x+3)+x+3$	$11 \cdot \log_2 y+3$
DIV	$(x / y) \cdot (2 \cdot y+2)+(x\%y)+y+4$	$39 \cdot (\log_2 x-\log_2 y)+19$

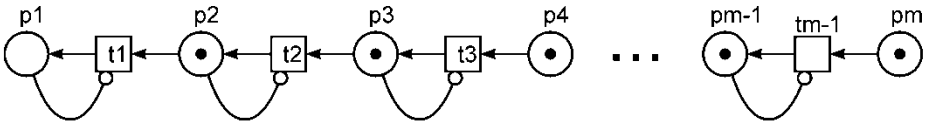
I. Peculiarities of programming in SNs



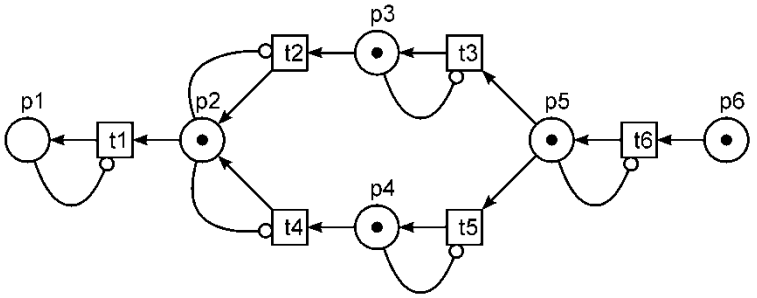
A. Reversed control flow (c)

B. Using inhibitor arcs to control a transition firing

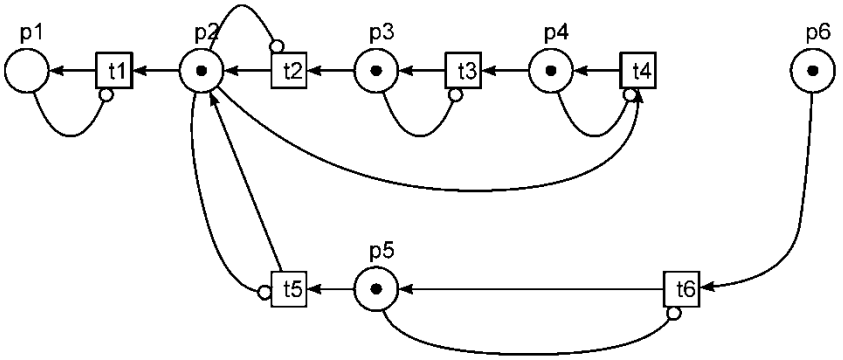
Basic operators



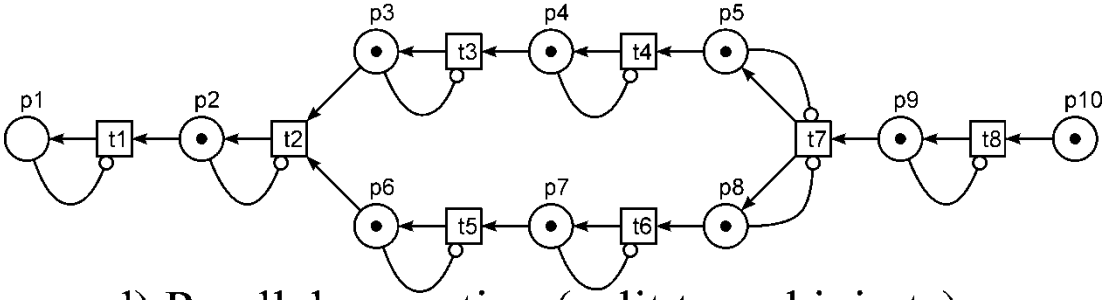
a) sequence;



b) branching;

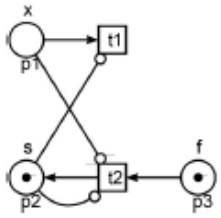


c) cycle;



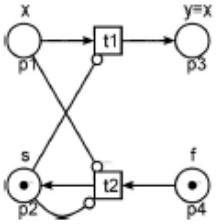
d) Parallel execution (split t_2 and join t_7)

Basic subnets (subroutines)



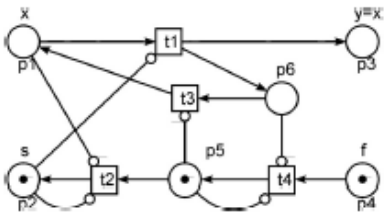
CLEAN:

$x := 0$



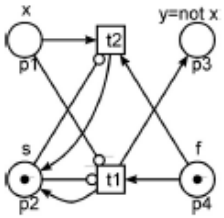
MOVE:

$y := x, x := 0$



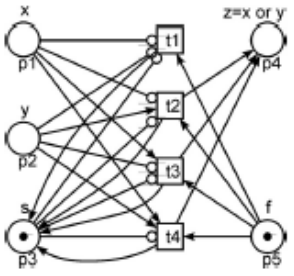
COPY:

$y := x$



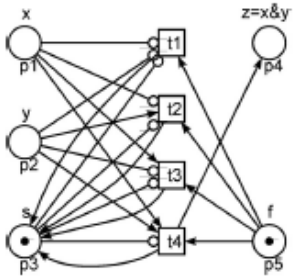
NOT(x):

$y := \neg x, x := 0$



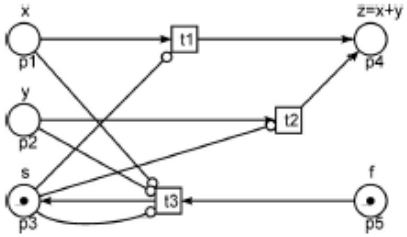
OR(x,y)

$z := x \vee y, x := 0, y := 0$



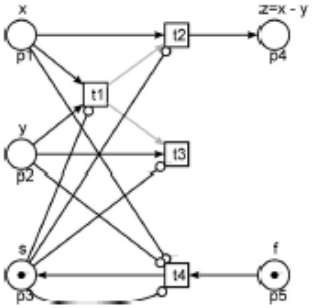
AND(x,y):

$z := x \wedge y, x := 0, y := 0$



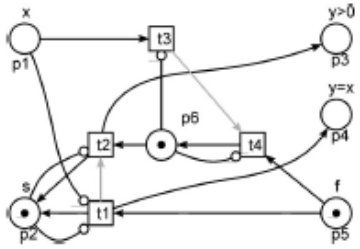
ADD(x,y):

$z := x + y, x := 0, y := 0$



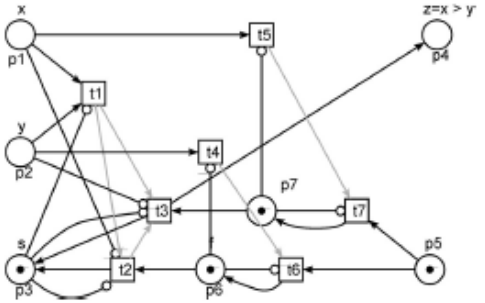
SUB(x,y):

$z := x - y, x := 0, y := 0$



GT0(x):

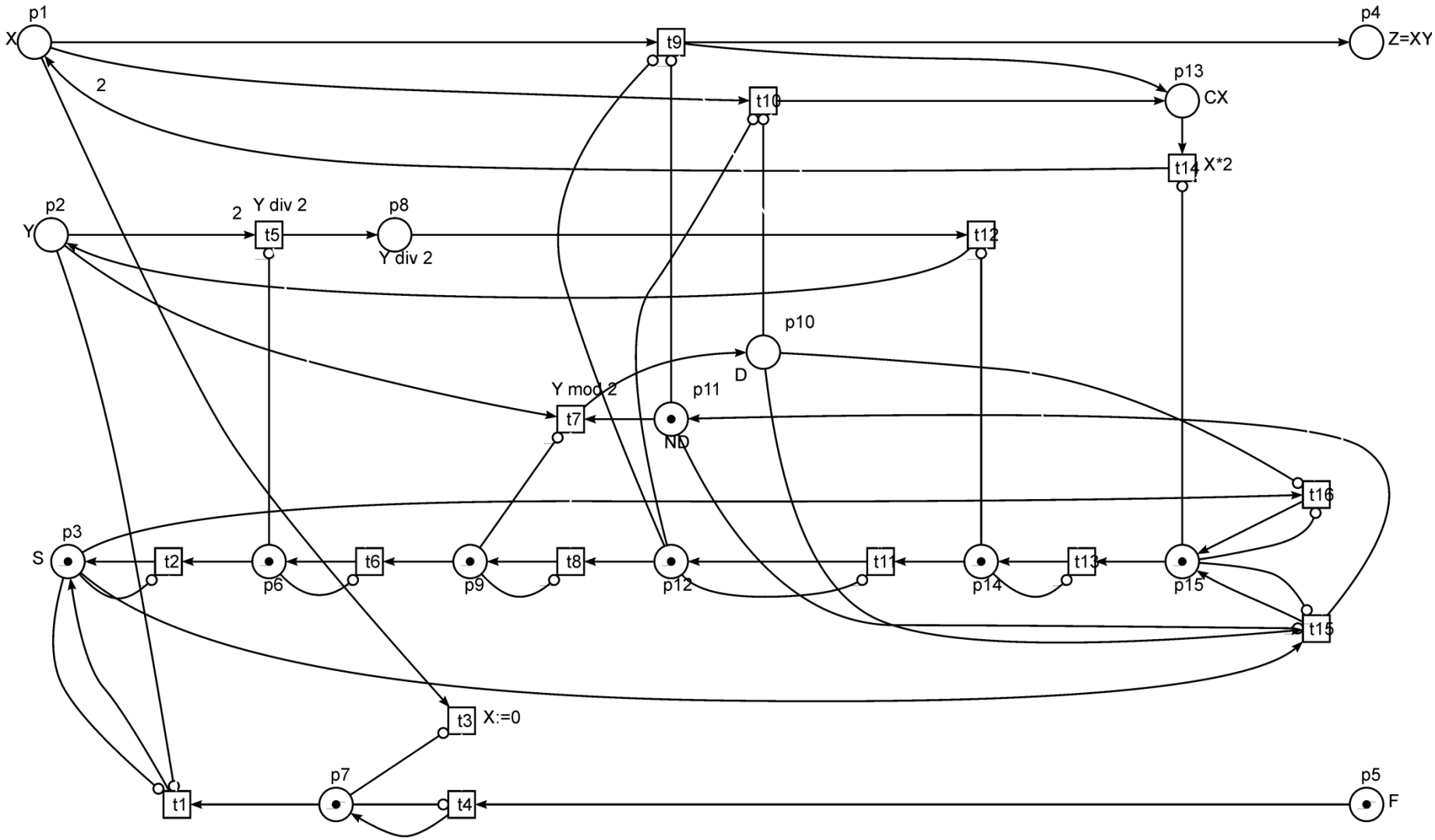
$y := (x > 0), z := (x = 0)$



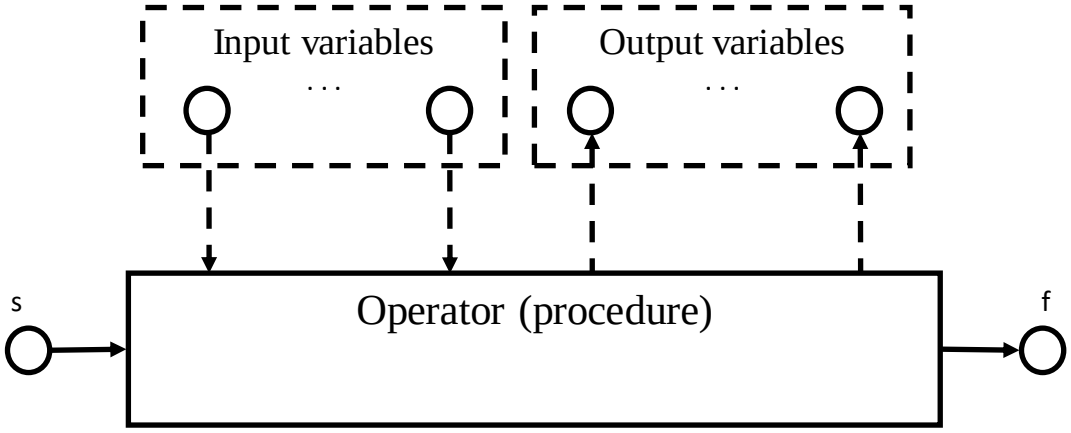
GT(x,y):

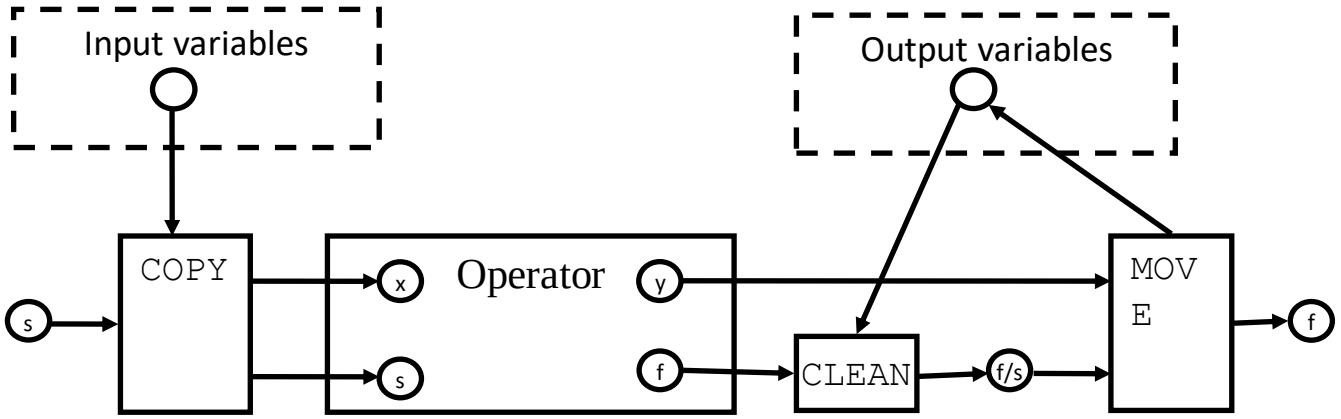
$z := (x > y), x := 0, y := 0$

Fast multiplication in Sleptsov nets

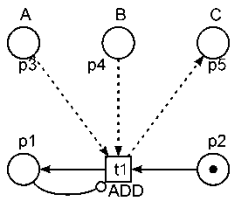


Work with variables

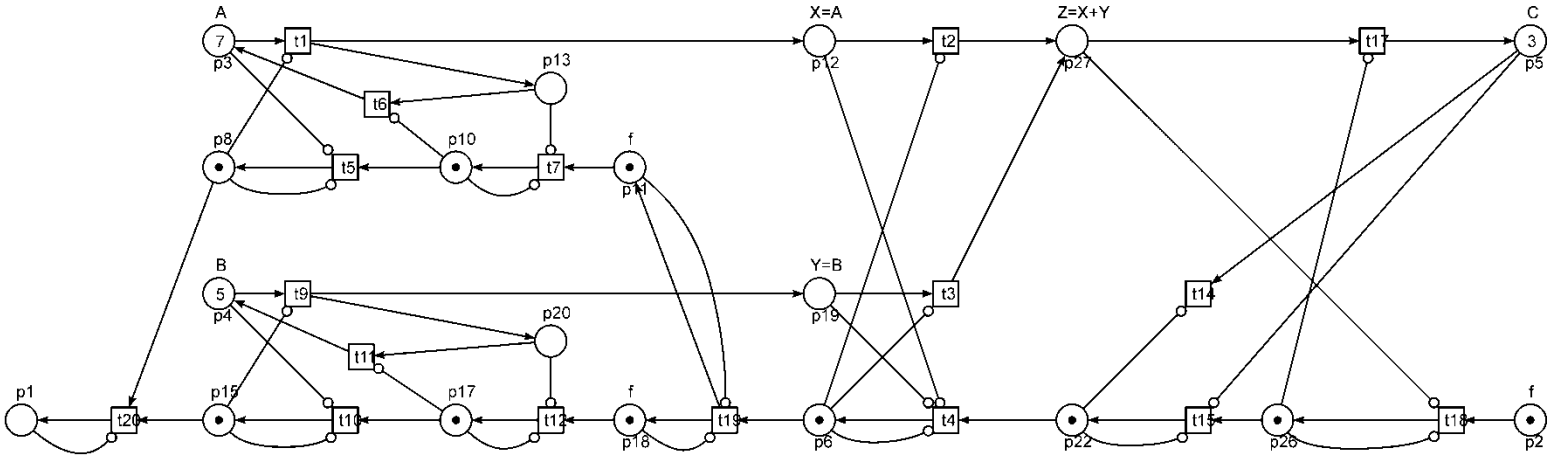




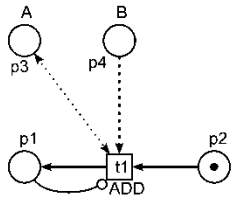
Expansion of dashed/dotted arcs



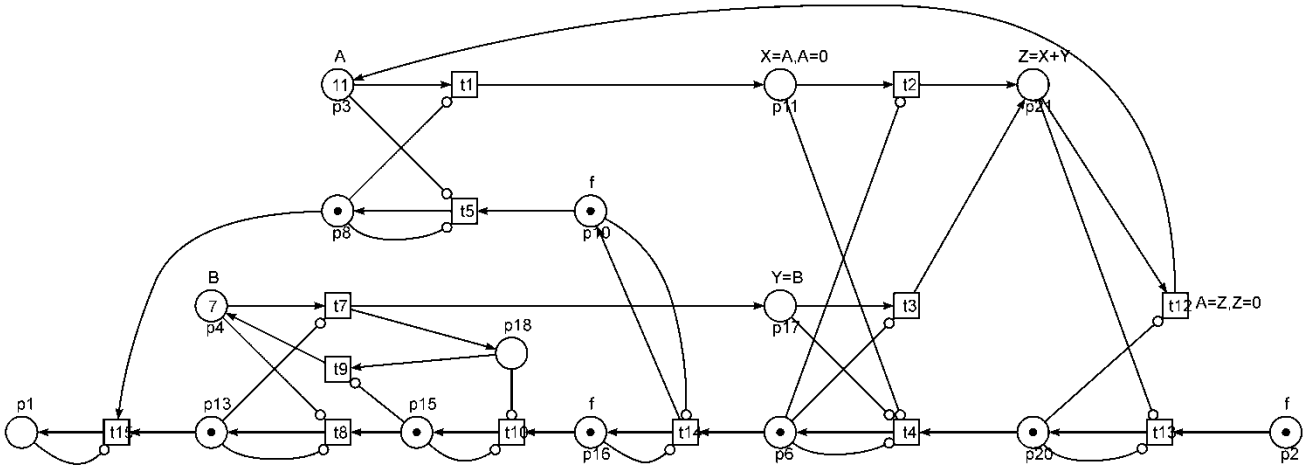
a)
c=add(a,b)



b) expansion of a)

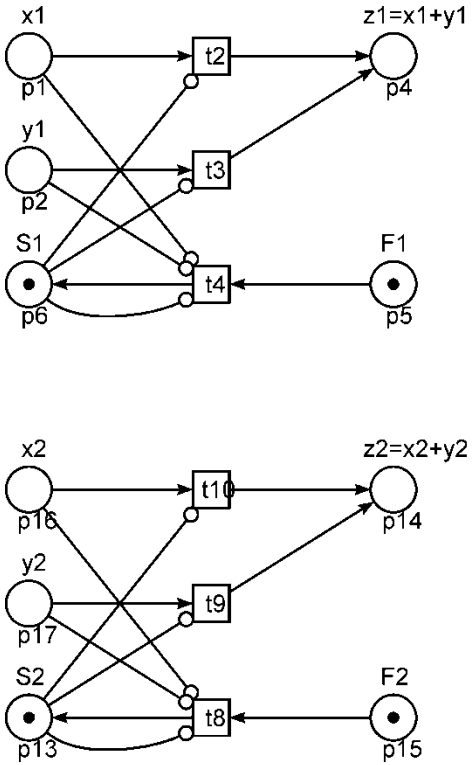


c)
a=add(a,b)

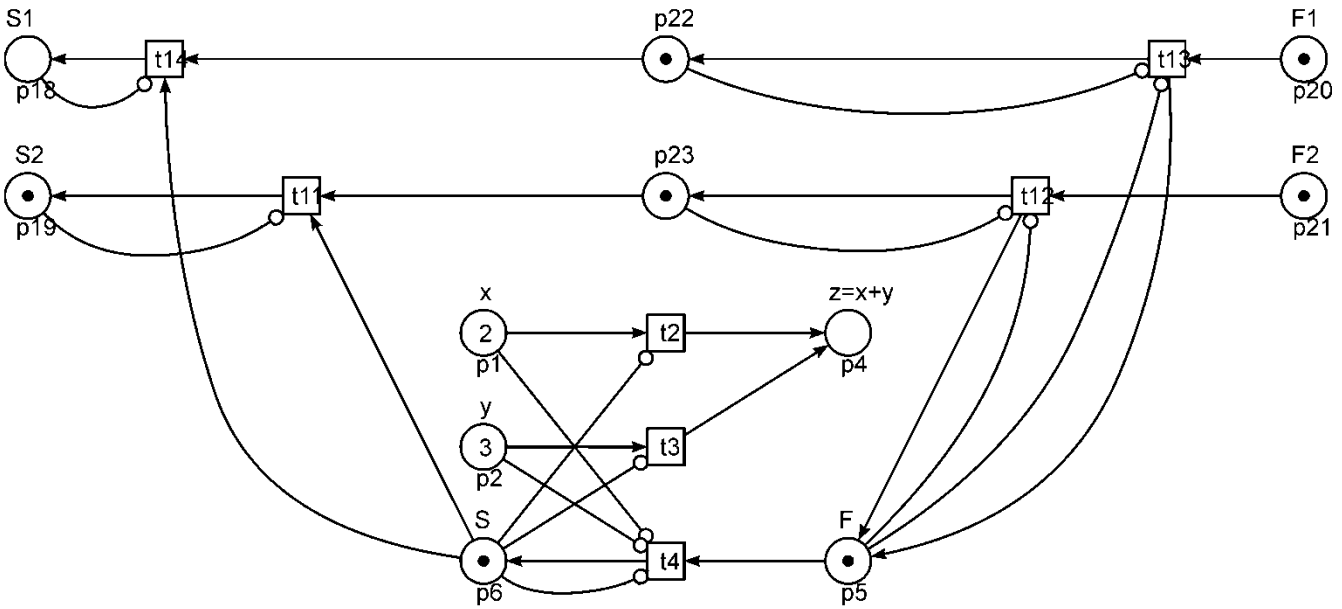


d) expansion of c)

Subnets (routines) calls

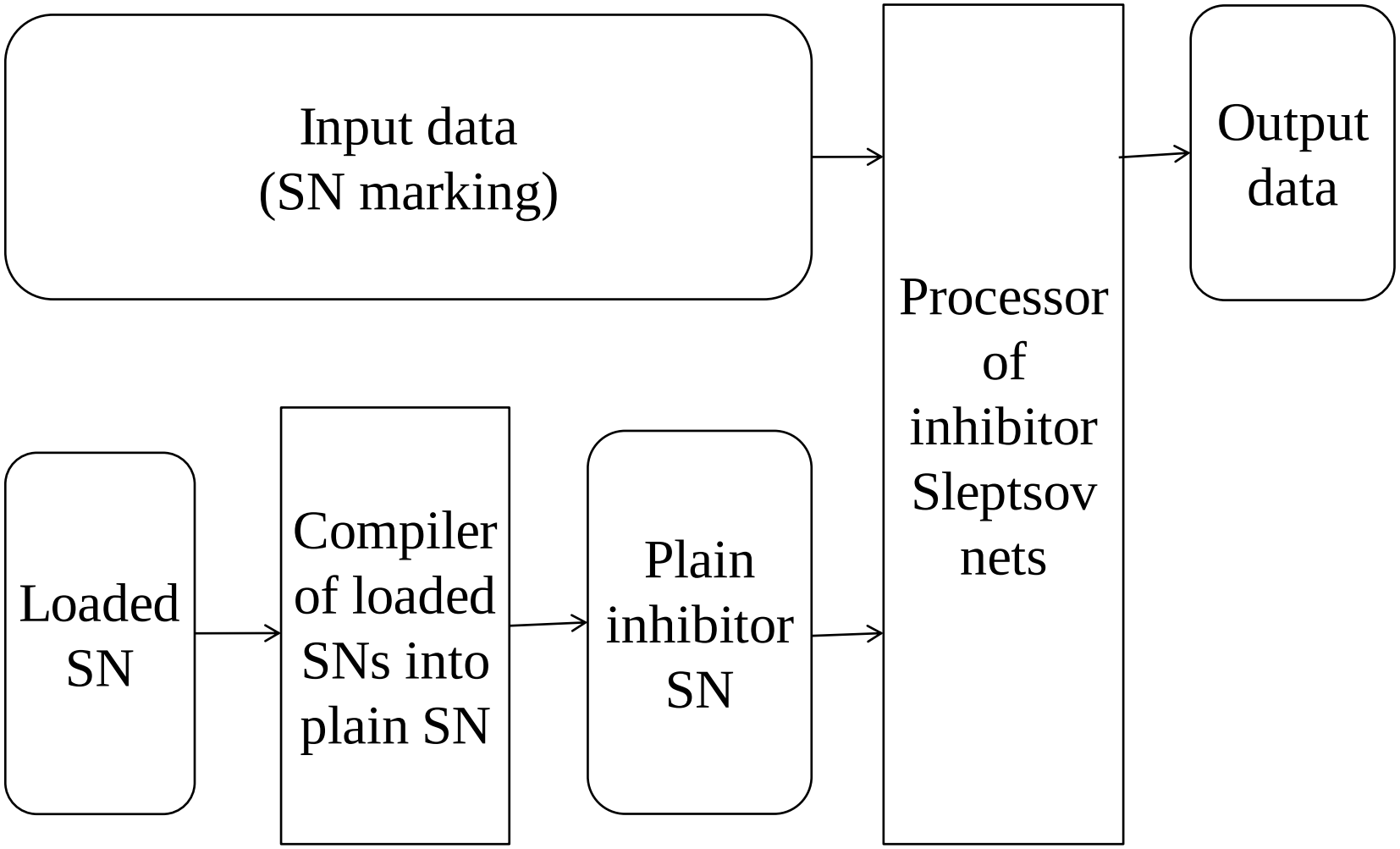


a) inline



b) call-return

Sleptsov Net Paradigm of Computing

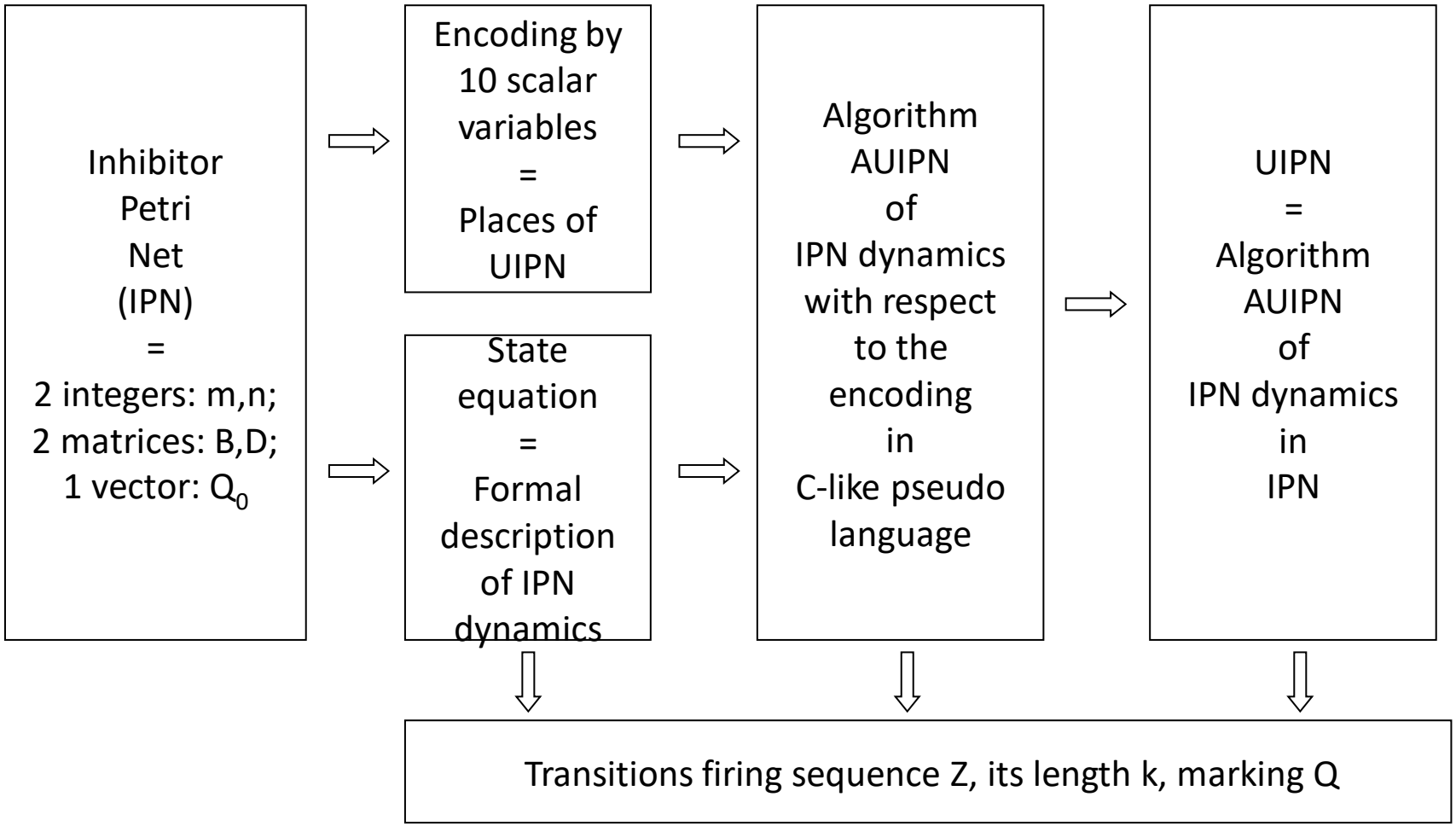


Universal Sleptsov Net Concept

Explicitly Constructed Universal Petri/Sleptsov Nets

Year	Technique	Size (nodes)	Time complexity
2010	Direct simulation of inhibitor PN by inhibitor PN	1000	polynomial
2010	Simulation of a given Turing machine by deterministic inhibitor PN (DIPN)	1000	exponential
2011	Simulation of a given Markov normal algorithm by DIPN	1000	exponential
2013	Simulation of small universal Turing machine by DIPN	56	exponential
2013	Simulation of weak small universal Turing machine by DIPN	43	exponential
2015	Simulation of cellular automaton Rule 110 by infinite PN	$21 \cdot n$	polynomial
2015	Simulation of Turing machine that simulates Rule 110 by infinite PN	$14 \cdot n$	polynomial
2017	Simulation of weak small universal Turing machine by Sleptsov net	39	polynomial

II. Direct simulation of an inhibitor PN by an inhibitor PN



Formal Representation of Inhibitor Petri Net behavior

$N = (G, Q_0)$, where G – graph of net, and Q_0 – its initial marking

$G = (P, T, B, D)$, $P = \{p_1, .., p_m\}$ – places, $T = \{t_1, .., t_n\}$ – transitions;

$B: P \times T \to \mathbb{N} \cup \{-1\}$, $D: T \times P \to \mathbb{N}$ – arcs, $Q: P \to \mathbb{N}$ – marking.

Vector (matrix) representation: $N=(m, n, B, D, Q_0)$,

$$B = \|b_{i,j}\|, b_{i,j} = B(p_j, t_i), D = \|d_{i,j}\|, d_{i,j} = D(t_i, p_j), Q = \|q_j\|, q_j = Q(p_j).$$

State equation

$$\begin{cases} q_j^k = q_j^{k-1} - x(b_{l,j}) + d_{l,j}, j = \overline{1,m} \\ u(t_i) = \bigwedge\limits_{j=1,m} ((y(b_{i,j}) \wedge (q_j^{k-1} = 0)) \vee (\overline{y}(b_{i,j}) \wedge (q_j^{k-1} \geq b_{i,j})), i = \overline{1,n} \\ u(t_l) = 1, l \in \overline{1,n} \\ k = 1, 2, \ldots \\ x(b) = \begin{cases} b, b \geq 0 \\ 0, b = -1 \end{cases}, y(b) = \begin{cases} 0, b \geq 0 \\ 1, b = -1 \end{cases}. \end{cases}$$

Encodings

Encoding of a vector (marking)

$$s = \varphi(Q) = \sum_{j=0}^{m-1} r^j \cdot q_j\,, \qquad r = \max_j q_j + 1$$

Encoding of a matrix (incidence)

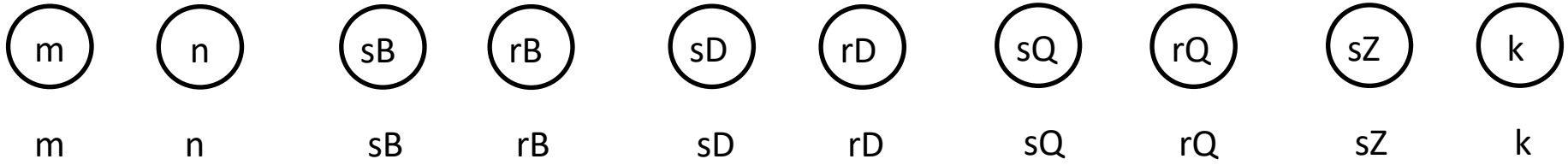
$$s = \varphi(A) = \sum_{i=0}^{n-1} \sum_{j=0}^{m-1} r^{(m \cdot i + j)} \cdot a_{i,j}\,, \qquad r = \max_{i,j} a_{i,j} + 1$$

Recursive encoding

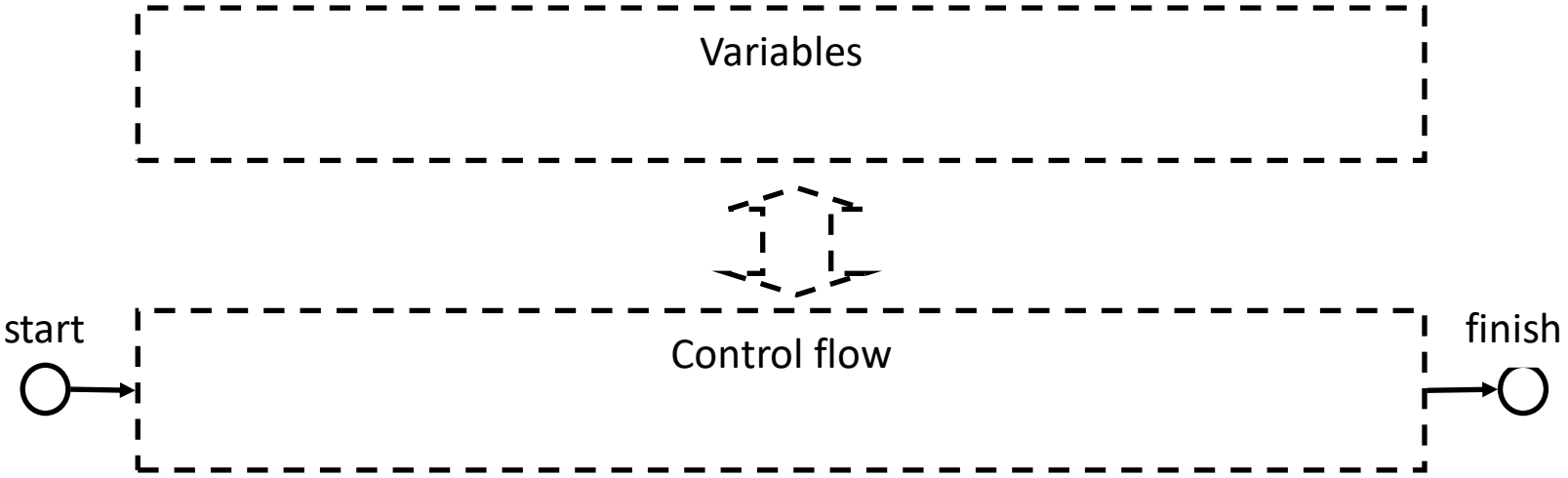
$$s_v = s_{v-1} \cdot r + a_{i,j}, s_0 = a_{n-1,m-1}$$

$$a_{i,j} = s_{n \cdot m-1-v}\ mod\ r, s_{n \cdot m-1-(v+1)} = s_{n \cdot m-1-v}\ div\ r, s_{n \cdot m-1} = s$$

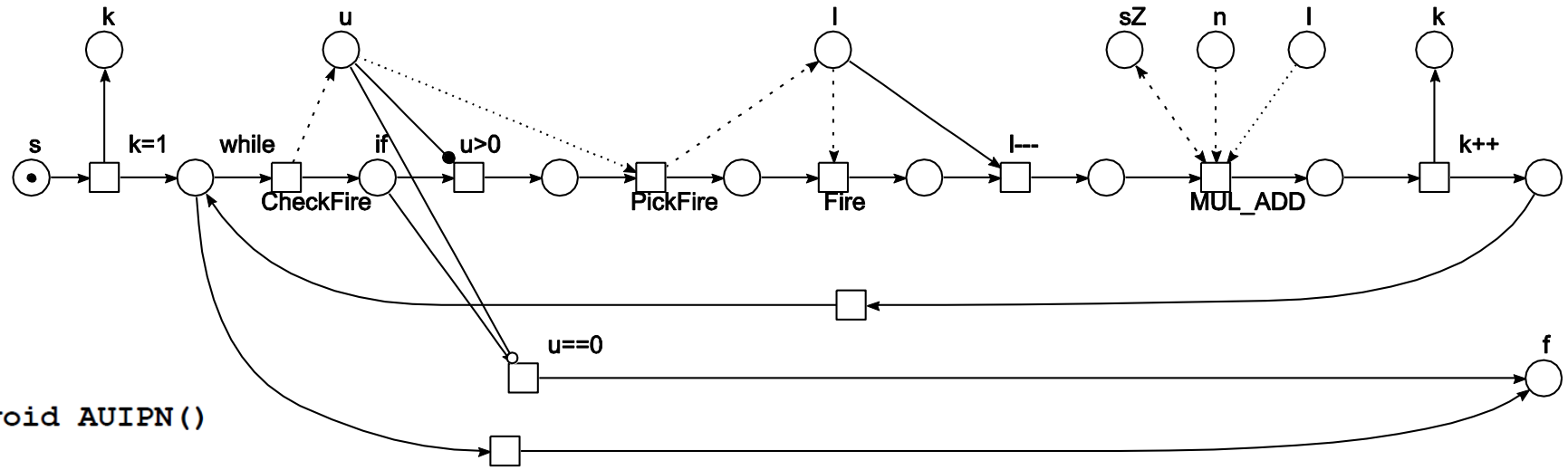
Composition of data and control flow



Code of a given Petri net

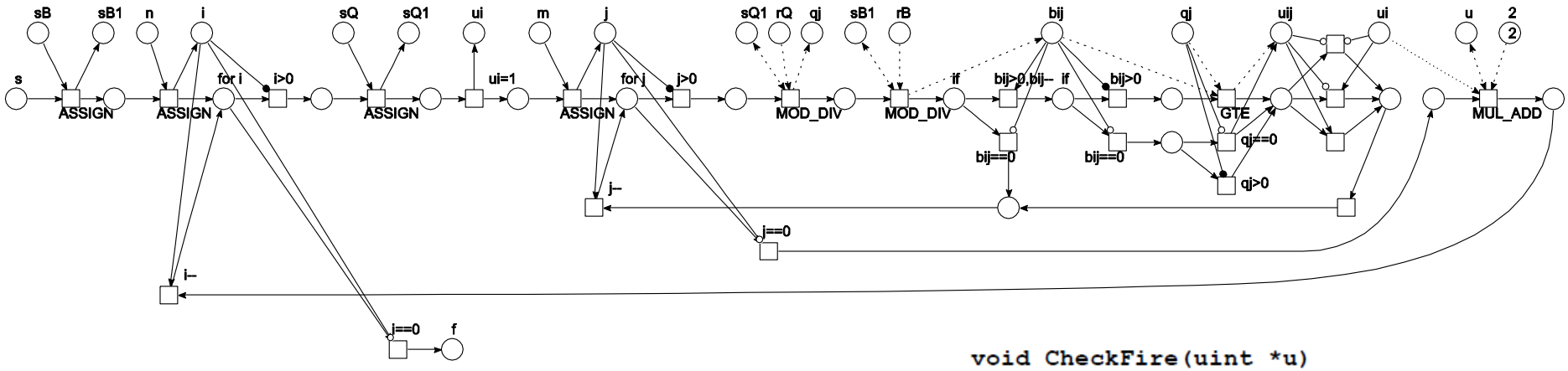


UIPN



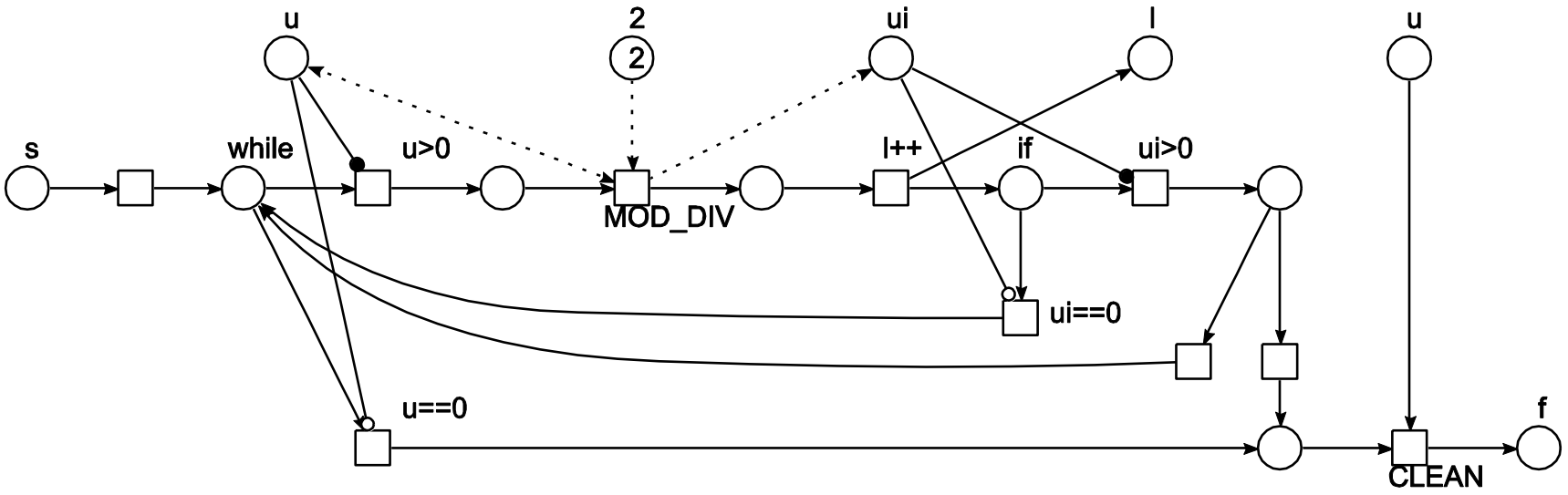

```
void AUIPN()
{
   uint u, l;

   inputXIPN();
   k=0; sZ=0;
   while(NonDeterministic())
   {
      CheckFire(&u);
      if(u==0) goto out;
      PickFire(u, &l);
      Fire(l);
      MUL_ADD(&sZ,n,l-1);
      k++;
   }
   out: outputXIPN();
}
```

CheckFire



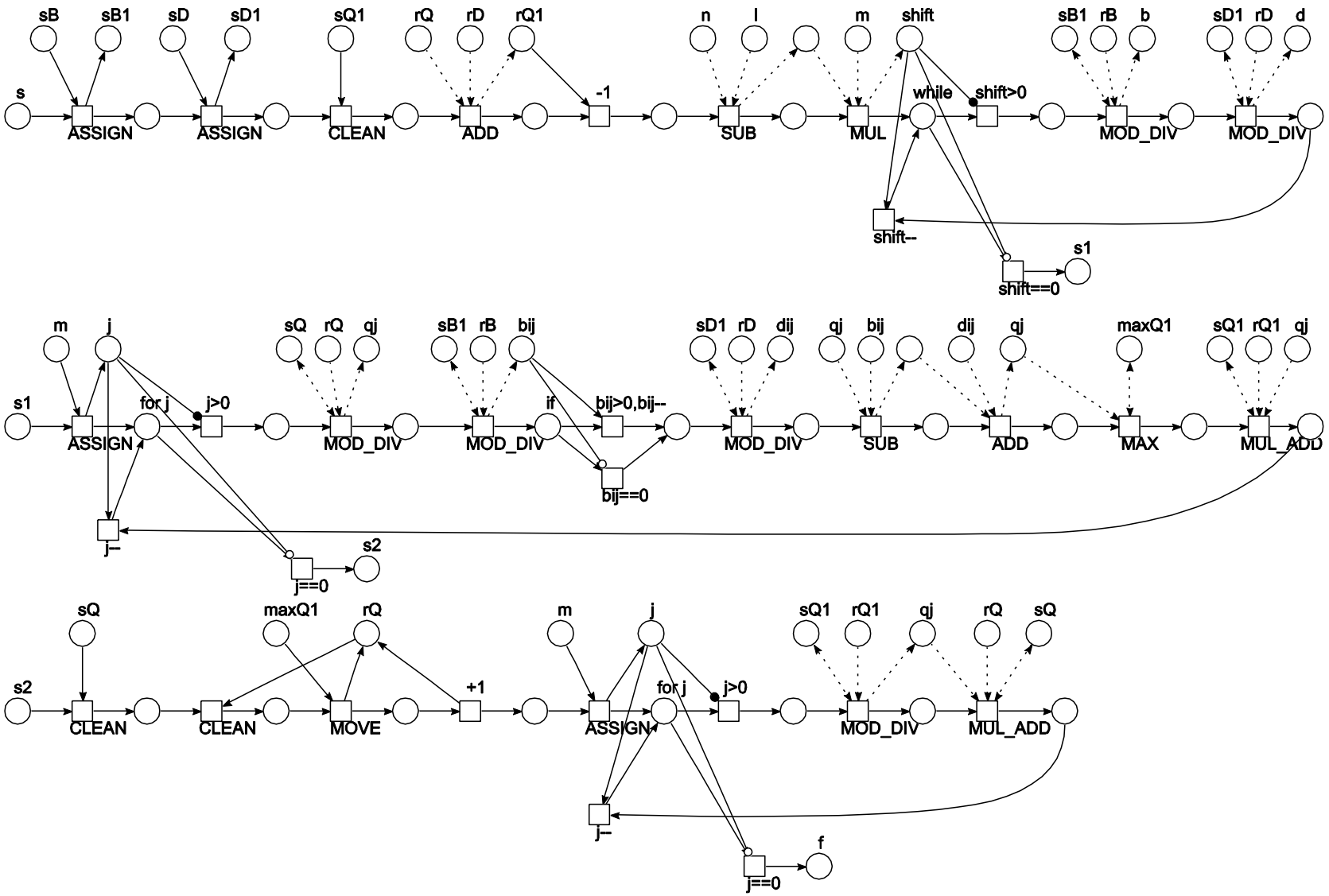

```
void CheckFire(uint *u)
{
     uint i, j, qj, bij, ui, uij;
     uint sB1, sQ1;
     sB1=sB; &u=0;
     for(i=n; i>0; i--)
     {
          sQ1=sQ;
          ui=1;
          for(j=m; j>0; j--)
          {
               MOD_DIV(&qj,&sQ1,rQ);
               MOD_DIV(&bij,&sB1,rB);
               uij=1;
               if(bij==0) continue;
               bij--;
               if(bij==0) uij=(qj==0);
               else uij=(qj>=bij);
               ui=ui && uij;
          }
          MUL_ADD(&u,2,ui);
     }
}
```

PickFire




```
void PickFire(uint u, uint *l)
{
      uint ui, i;
      i=0;
      while(u>0)
      {
            MOD_DIV(&ui,&u,2);
            i++;
            if(ui==0) continue;
            if(NonDeterministic()) goto out;
      }
      out: *l=i;
}
```

Fire


sB
sB1
sD
sD1
sQ1
rQ
rD
rQ1
n
l
m
shift
sB1
rB
b
sD1
rD
d
s
-1
while
shift>0
ASSIGN
ASSIGN
CLEAN
ADD
SUB
MUL
MOD_DIV
MOD_DIV
shift--
shift==0
s1
m
j
sQ
rQ
qj
sB1
rB
bij
sD1
rD
dij
qj
bij
dij
qj
maxQ1
sQ1
rQ1
qj
s1
for j
j>0
if
bij>0,bij--
ASSIGN
MOD_DIV
MOD_DIV
MOD_DIV
SUB
ADD
MAX
MUL_ADD
bij==0
j--
s2
j==0
sQ
maxQ1
rQ
m
j
sQ1
rQ1
qj
rQ
sQ
s2
+1
for j
j>0
CLEAN
CLEAN
MOVE
ASSIGN
MOD_DIV
MUL_ADD
j--
f
j==0

Examples of nets encoding

Petri net graph

Net	m	n	sB	rB	sD	rD
ADD	6	4	21180169496	3	282946	2
MAX	8	8	254813592433189871074065241412	3	293862152152879368	2
MUL	10	9	64654907206110145566888903466348174395265 4	3	19352259085292454555975681	2

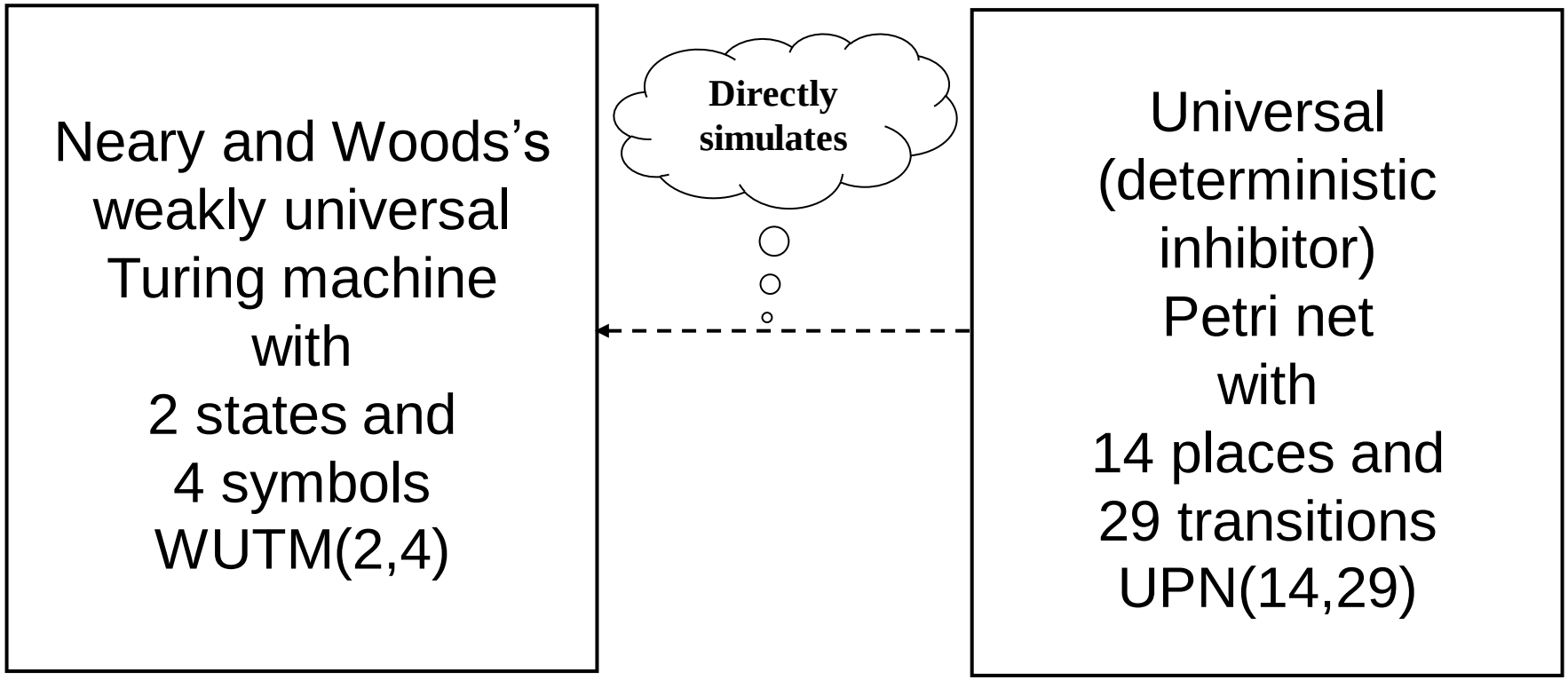
Marking

Net	Marking	*Q*	sQ	rQ
ADD	ADDQ0	(2,3,1,0,0,0)	2880	4
ADD	ADDQ	(0,0,0,5,1,0)	186	6
MAX	MAXQ0	(2,3,1,0,0,0,0,0)	46080	4
MAX	MAXQ	(0,0,0,3,1,0,0,0)	832	4
MUL	MULQ0	(2,3,1,0,0,0,0,0,0,0)	737280	4
MUL	MULQ	(0,0,0,6,1,0,0,0,0,0)	722701	7

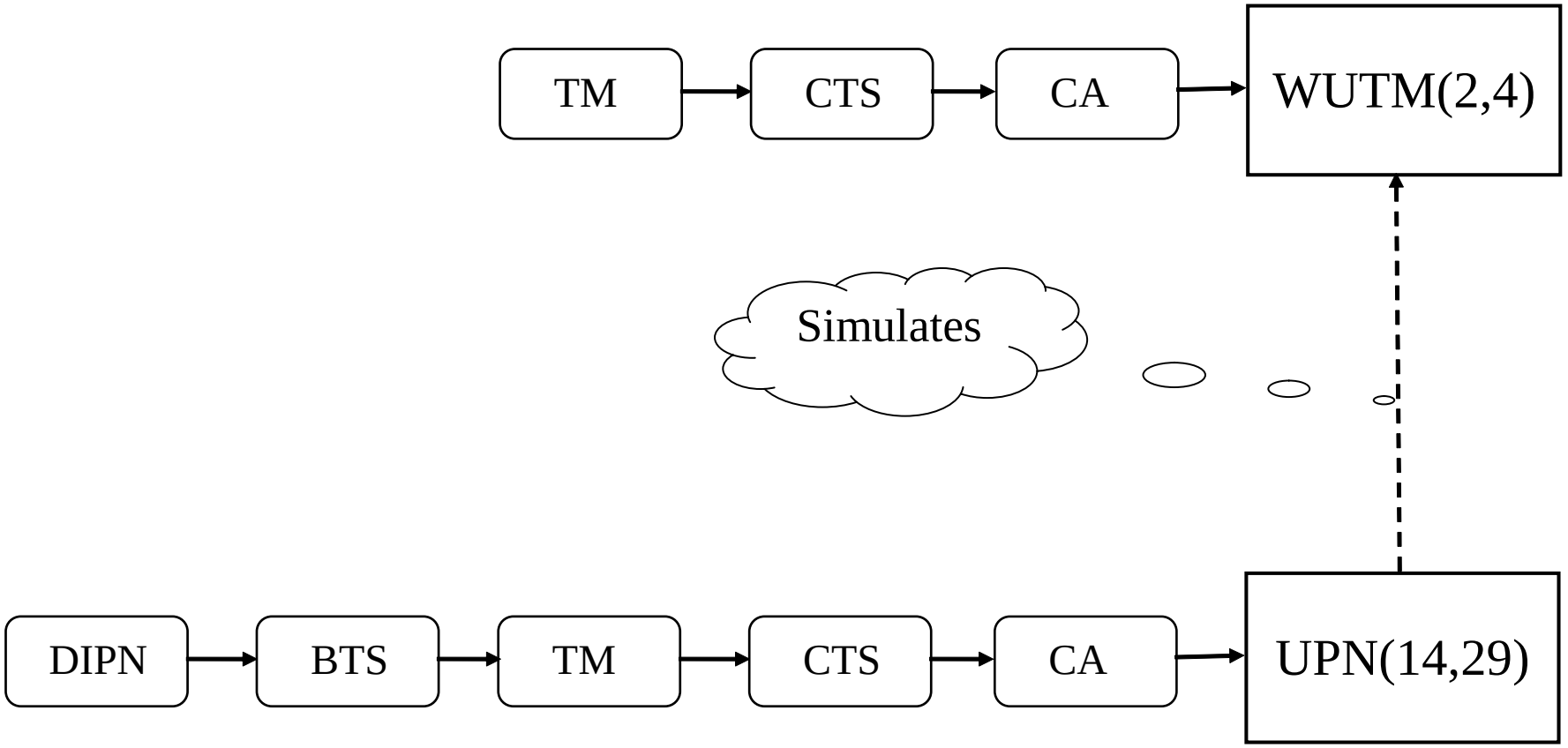
Transitions firing sequence

Net	Q0	Q	Z	sZ	k
ADD	ADDQ0	ADDQ	t1,t3,t2,t2,t3,t3,t4	2411	7
MAX	MAXQ0	MAXQ	t1,t2,t2,t6,t7,t8	4983	6
MUL	MULQ0	MULQ	t1,t2,t4,t4,t5,t6,t6,t7,t2, t4,t4,t5,t6,t6,t7,t2,t4,t4, t5,t6,t6,t7,t3,t9,t9,t8	1098157122123397237052 98	26

II. UPN(14,29)


Neary and Woods's
weakly universal
Turing machine
with
2 states and
4 symbols
WUTM(2,4)
Directly
simulates
Universal
(deterministic
inhibitor)
Petri net
with
14 places and
29 transitions
UPN(14,29)

Chains of translations

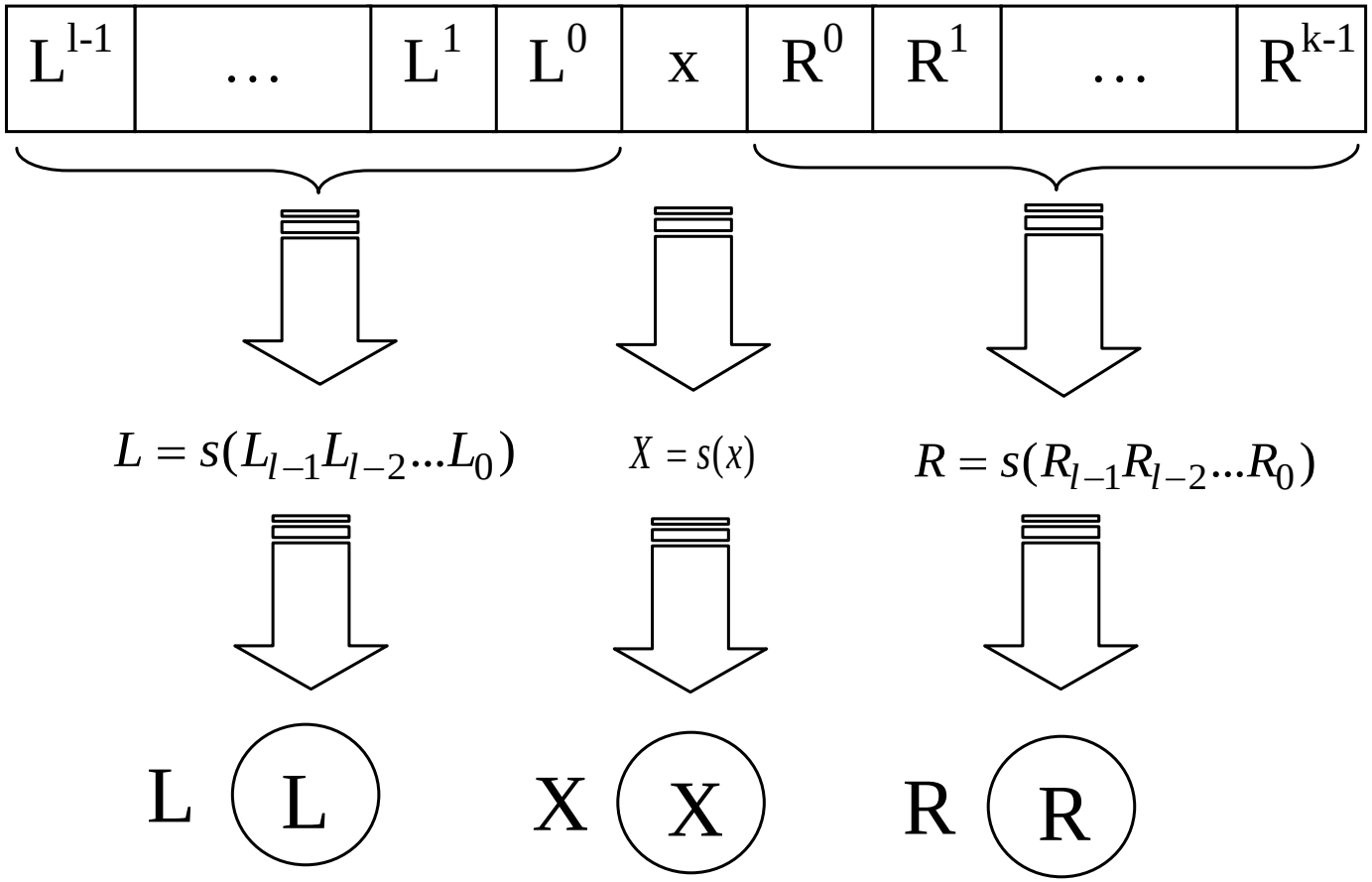


Encoding of states, symbols, and transition function

$\Sigma\backslash\Omega$		u_1	u_2
	$s(\Sigma)\backslash s(\Omega)$	0	1
0	1	$3, left, 0$	$4, right, 0$
1	2	$4, left, 1$	$3, left, 1$
$\emptyset$	3	$4, left, 0$	$1, right, 1$
$\not{1}$	4	$4, left, 0$	$2, right, 1$

Tape encoding

Encoding function: $s(x_{l-1}x_{l-2}...x_0) = \sum_{i=0}^{l-1} s(x_i) \cdot r^i$



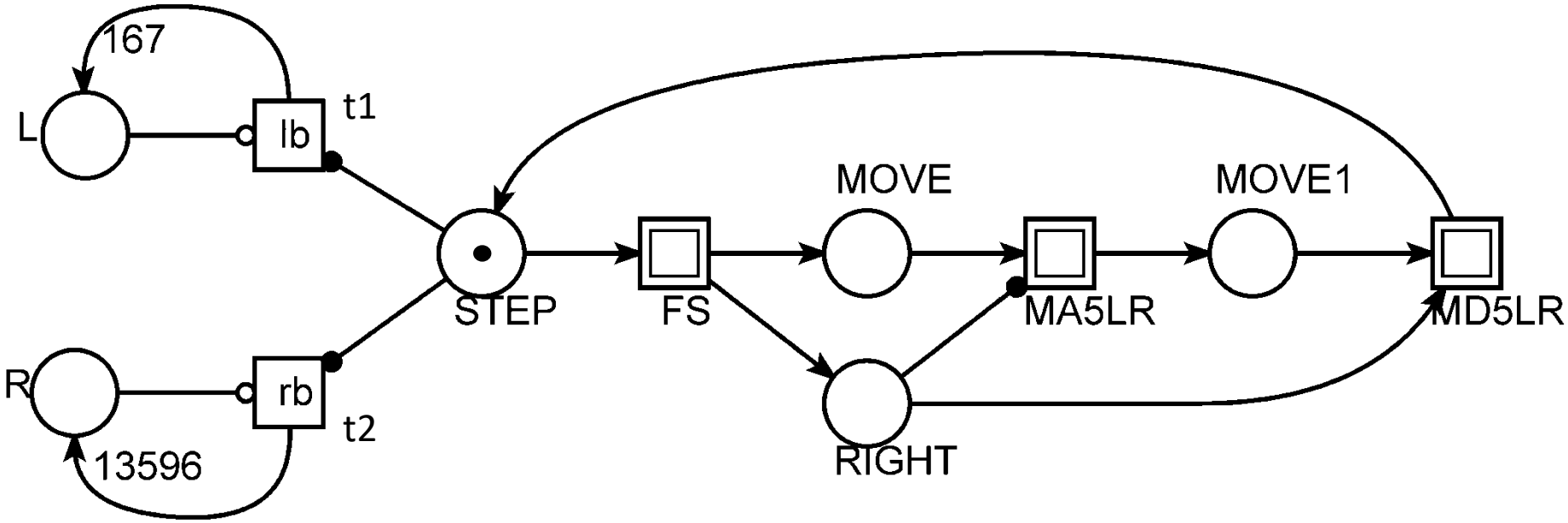
Encoding of blank words

$$w_l = 00\emptyset 1 \qquad w_r = 0\mathbb{1}\emptyset\emptyset 0\mathbb{1}$$

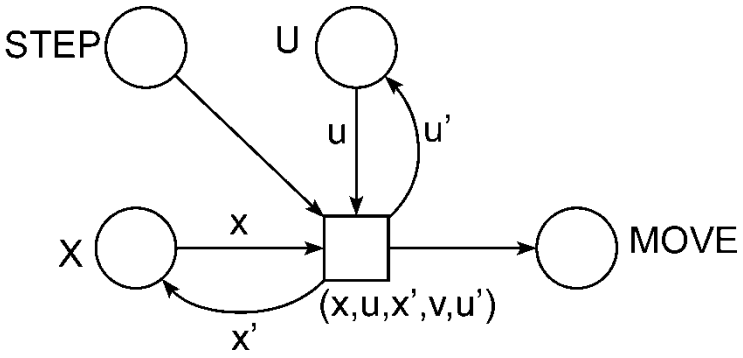
$$sw_l = s(w_l) = ((s(0)\cdot rX + s(0))\cdot rX + s(\emptyset))\cdot rX + s(1) = ((1\cdot 5+1)\cdot 5+3)\cdot 5+2 = 167$$

$$sw_r = s(w_r) = ((((s(\mathbb{1})\cdot rX + s(0))\cdot rX + s(\emptyset))\cdot rX + s(\emptyset))\cdot rX + s(\mathbb{1}))\cdot rX + s(0) =$$
$$= ((((4\cdot 5+1)\cdot 5+3)\cdot 5+3)\cdot 5+4)\cdot 5+1 = 13596.$$

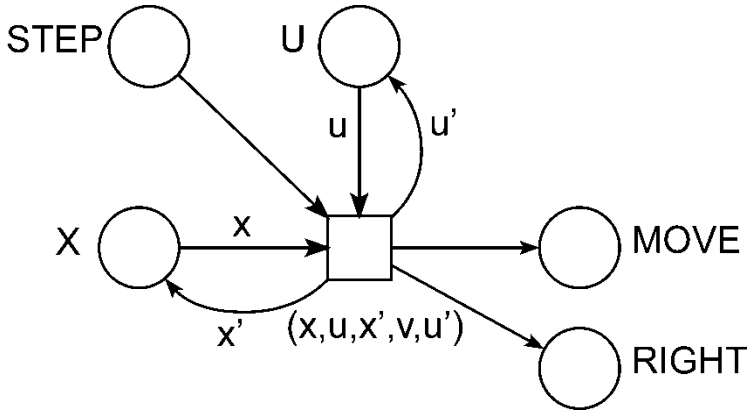
General arrangement of UPN(14,29)



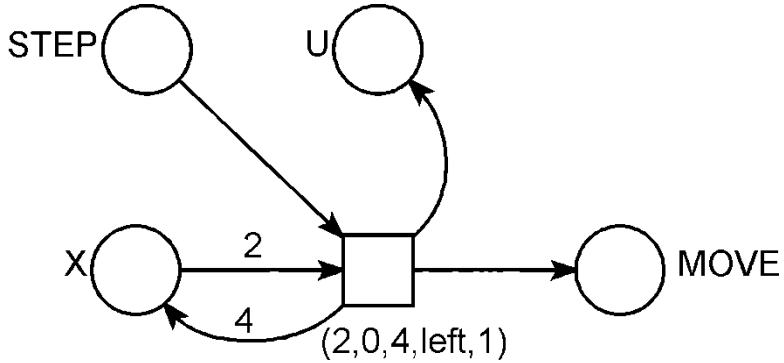
Simulating TM instruction



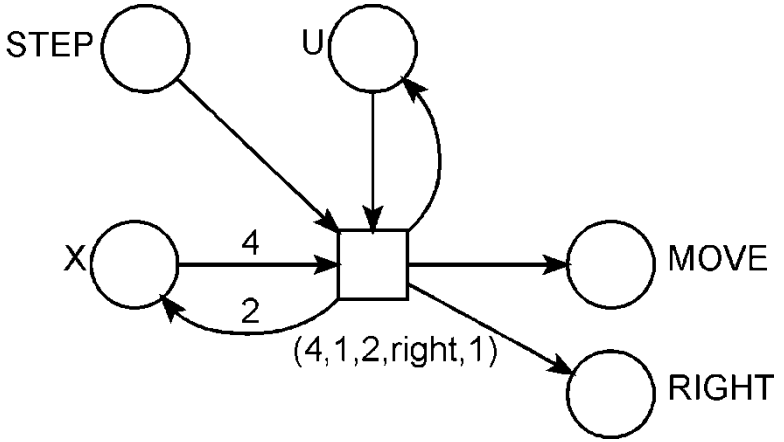
an instruction with the left move



an instruction with the right move

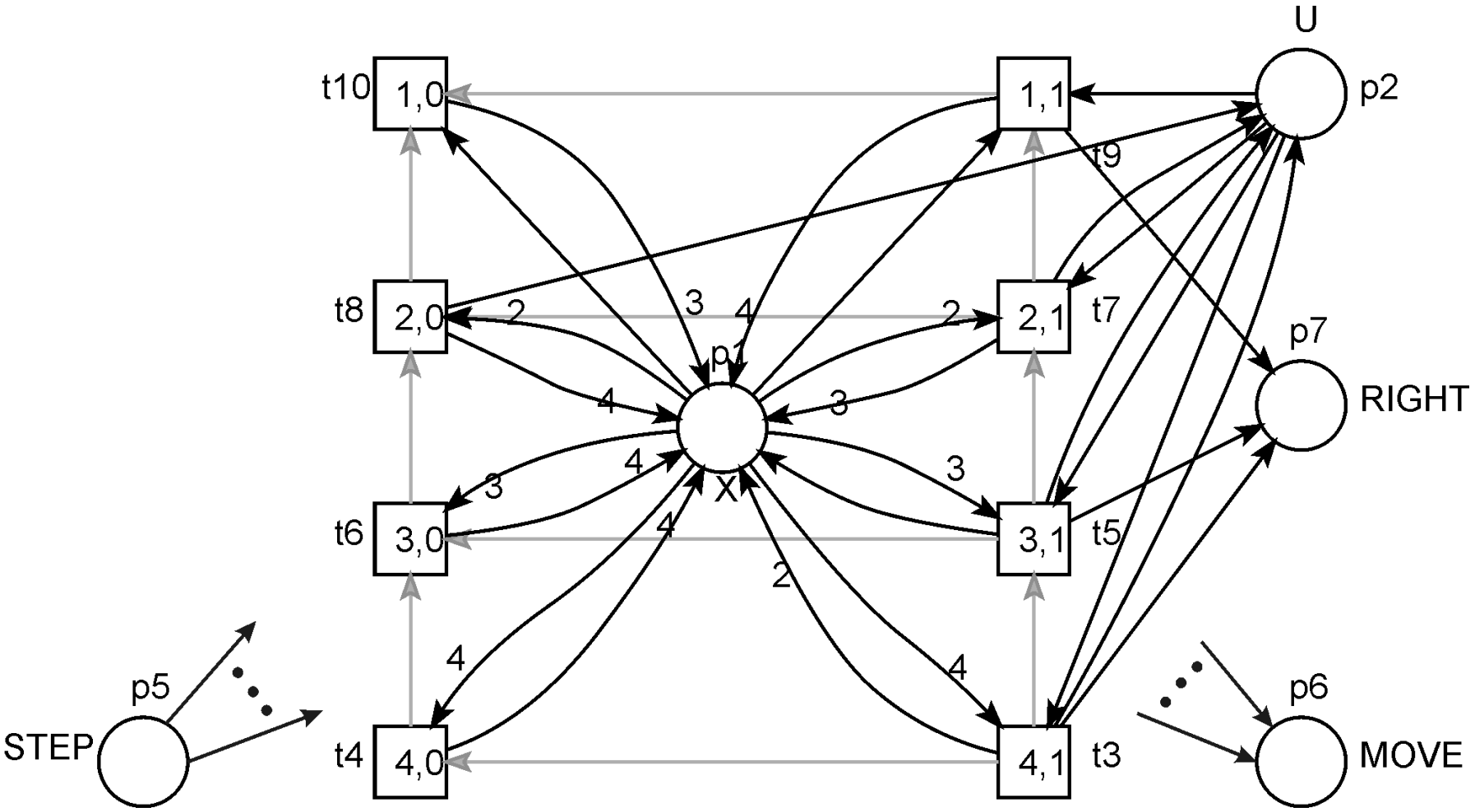


(2,0,4,*left*,1)

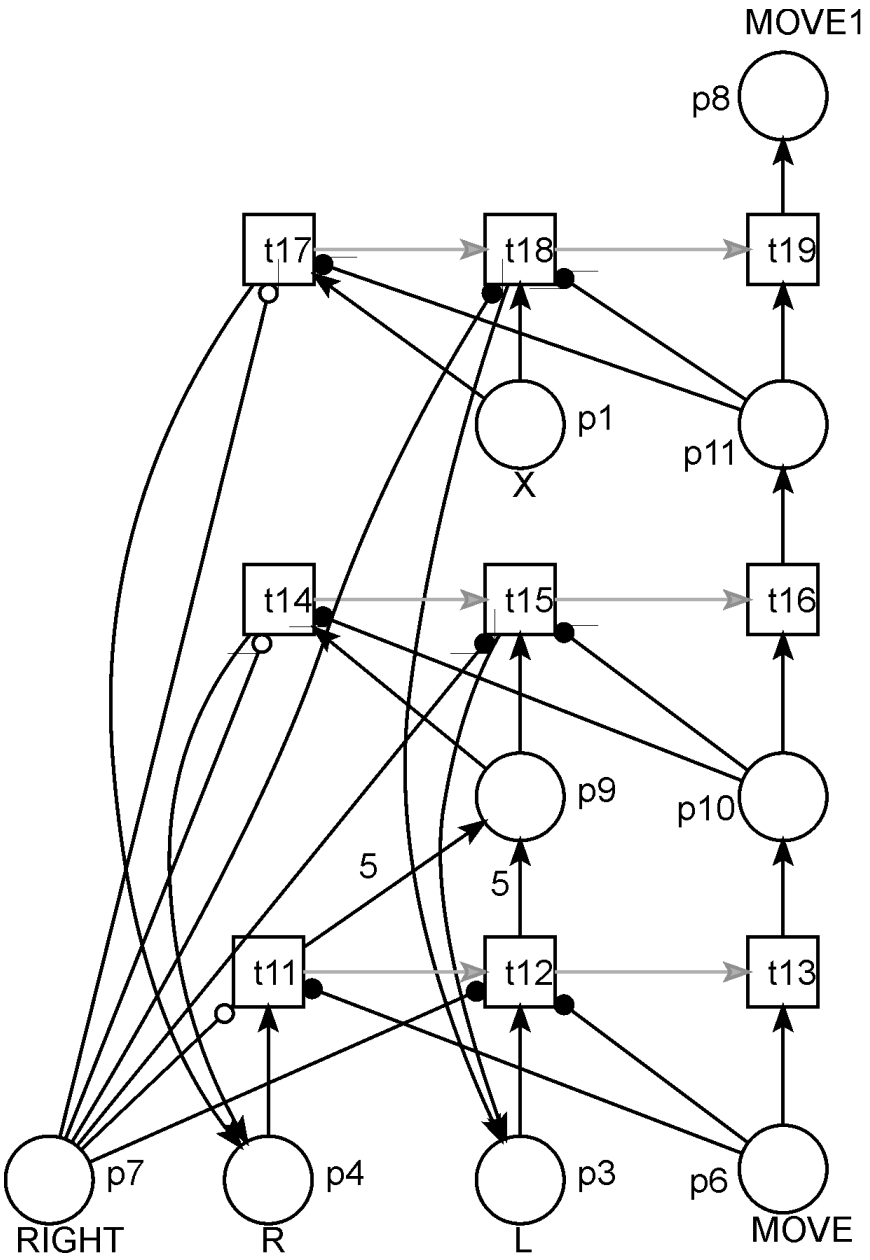


(4,1,2,*right*,1)

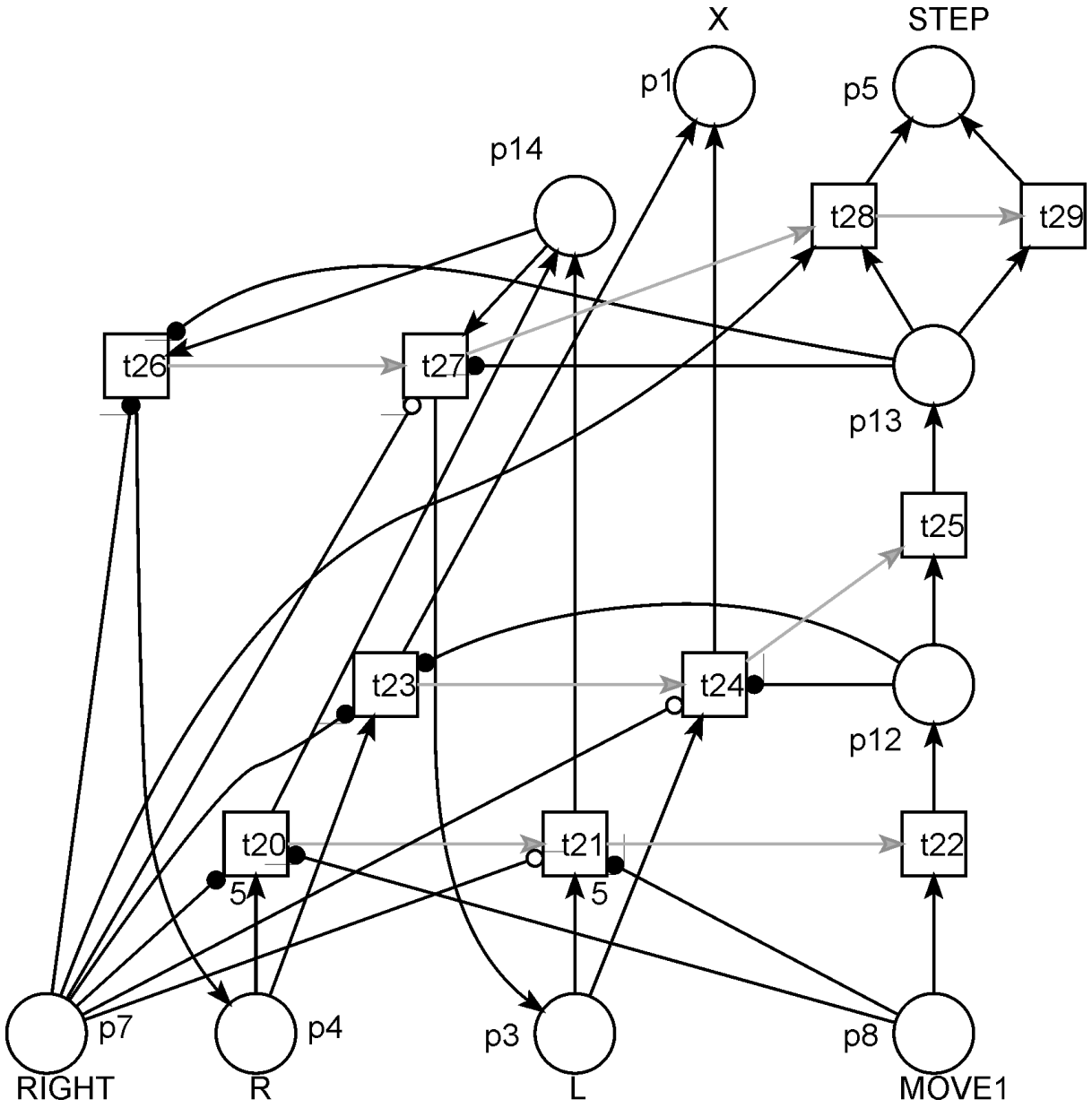
Subnet FS simulating WUTM(2,4) transition function



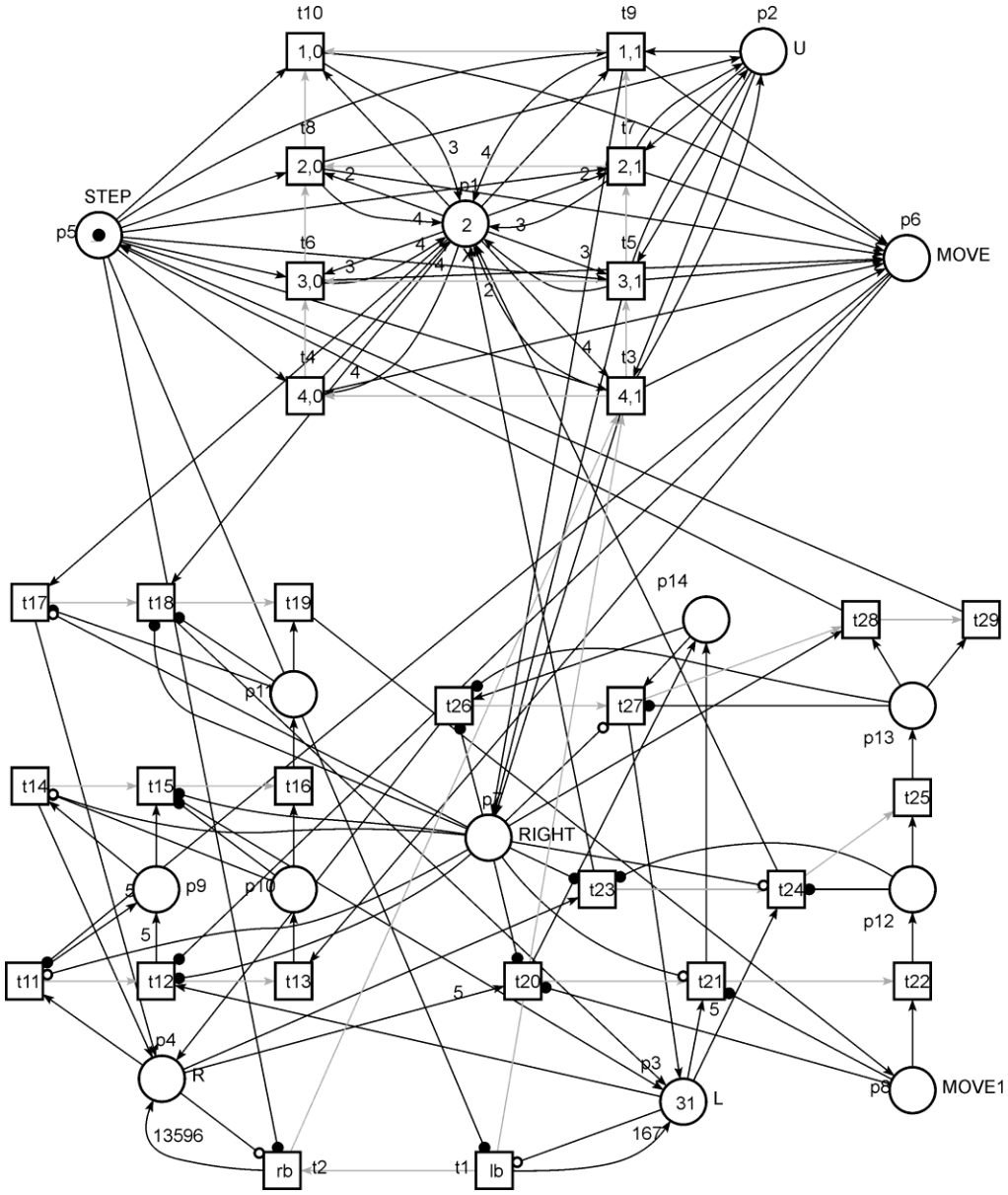
Subnet MA5LR



Subnet MD5LR



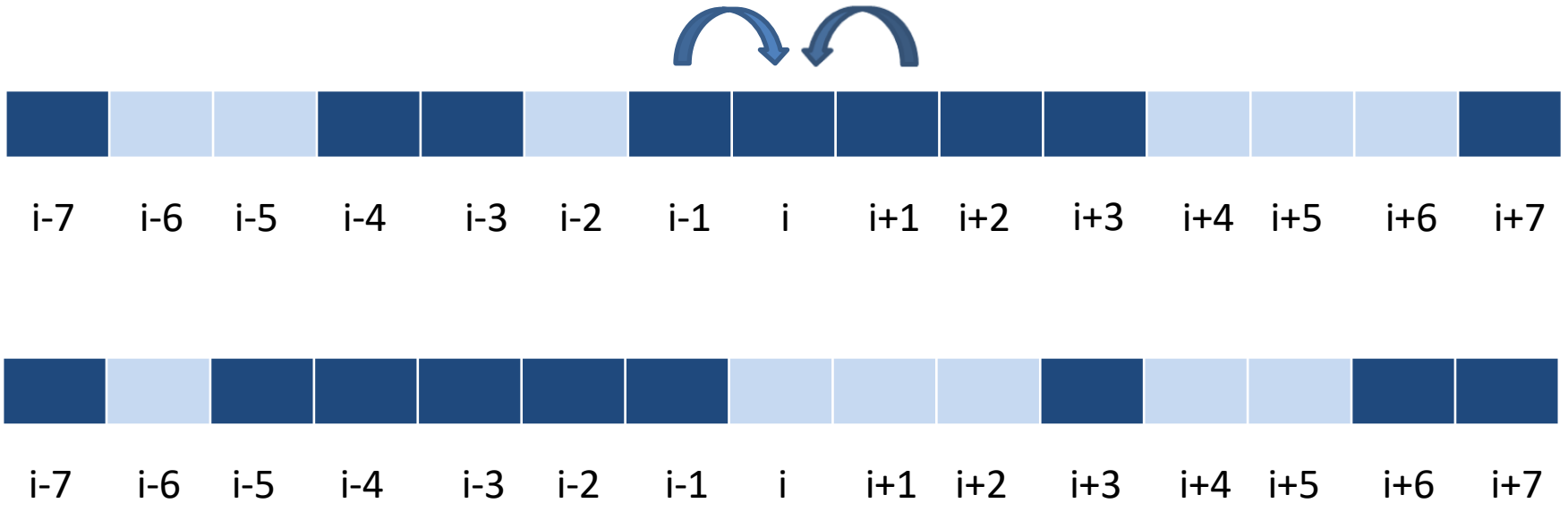
UPN(14,29) in graphical form



Trace of UPN(14,29) running

Step	Configuration	Code of	
		state U	tape (L,X,R)
0	$u_1, \ldots 00\emptyset 1\ 00\emptyset 1\ \mathbf{000\underline{1}}\ 0\not{1}\emptyset\emptyset 0\not{1}\ 0\not{1}\emptyset\emptyset 0\not{1} \ldots$	0	(31,2,0)
	$u_1, \ldots 00\emptyset 1\ 00\emptyset 1\ \mathbf{000\underline{1}\ 01\emptyset 00\not{1}}\ 0\not{1}\emptyset\emptyset 0\not{1} \ldots$	0	(31,2,13596)
1	$u_2, \ldots 00\emptyset 1\ 00\emptyset 1\ \mathbf{00\underline{0}1\ 01\emptyset 00\not{1}}\ 0\not{1}\emptyset\emptyset 0\not{1} \ldots$	1	(6,1,67984)
2	$u_1, \ldots 00\emptyset 1\ 00\emptyset 1\ \mathbf{00\underline{1}1\ 01\emptyset 00\not{1}}\ 0\not{1}\emptyset\emptyset 0\not{1} \ldots$	0	(34,4,13596)
3	$u_1, \ldots 00\emptyset 1\ 00\emptyset 1\ \mathbf{00\underline{1}1\ 01\emptyset 00\not{1}}\ 0\not{1}\emptyset\emptyset 0\not{1} \ldots$	0	(6,4,67984)
4	$u_1, \ldots 00\emptyset 1\ 00\emptyset 1\ \mathbf{0\underline{0}11\ 01\emptyset 00\not{1}}\ 0\not{1}\emptyset\emptyset 0\not{1} \ldots$	0	(1,1,339924)
5	$u_1, \ldots 00\emptyset 1\ 00\emptyset 1\ \mathbf{\underline{0}011\ 01\emptyset 00\not{1}}\ 0\not{1}\emptyset\emptyset 0\not{1} \ldots$	0	(0,1,1699623)
	$u_1, \ldots 00\emptyset 1\ \mathbf{000\underline{1}\ 0011\ 01\emptyset 00\not{1}}\ 0\not{1}\emptyset\emptyset 0\not{1} \ldots$	0	(167,1,1699623)
6	$u_1, \ldots 00\emptyset 1\ \mathbf{00\underline{0}1\ 0011\ 01\emptyset 00\not{1}}\ 0\not{1}\emptyset\emptyset 0\not{1} \ldots$	0	(33,2,8498118)
7	$u_2, \ldots 00\emptyset 1\ \mathbf{00\underline{0}1\ 0011\ 01\emptyset 00\not{1}}\ 0\not{1}\emptyset\emptyset 0\not{1} \ldots$	1	(6,3,42490594)
8	$u_2, \ldots 00\emptyset 1\ \mathbf{000\underline{1}\ 0011\ 01\emptyset 00\not{1}}\ 0\not{1}\emptyset\emptyset 0\not{1} \ldots$	1	(31,4,8498118)
9	$u_2, \ldots 00\emptyset 1\ \mathbf{0001\ \underline{0}011\ 01\emptyset 00\not{1}}\ 0\not{1}\emptyset\emptyset 0\not{1} \ldots$	1	(157,3,1699623)
10	$u_2, \ldots 00\emptyset 1\ \mathbf{0001\ \underline{0}011\ 01\emptyset 00\not{1}}\ 0\not{1}\emptyset\emptyset 0\not{1} \ldots$	1	(786,3,339924)
11	$u_2, \ldots 00\emptyset 1\ \mathbf{0001\ 00\underline{1}1\ 01\emptyset 00\not{1}}\ 0\not{1}\emptyset\emptyset 0\not{1} \ldots$	1	(3931,4,67984)
12	$u_2, \ldots 00\emptyset 1\ \mathbf{0001\ 001\underline{1}\ 01\emptyset 00\not{1}}\ 0\not{1}\emptyset\emptyset 0\not{1} \ldots$	1	(19657,4,13596)
13	$u_2, \ldots 00\emptyset 1\ \mathbf{0001\ 0011\ \underline{0}1\emptyset 00\not{1}}\ 0\not{1}\emptyset\emptyset 0\not{1} \ldots$	1	(98287,1,2719)
14	$u_1, \ldots 00\emptyset 1\ \mathbf{0001\ 0011\ 1\underline{1}\emptyset 00\not{1}}\ 0\not{1}\emptyset\emptyset 0\not{1} \ldots$	0	(491439,4,543)
...	...	...	...

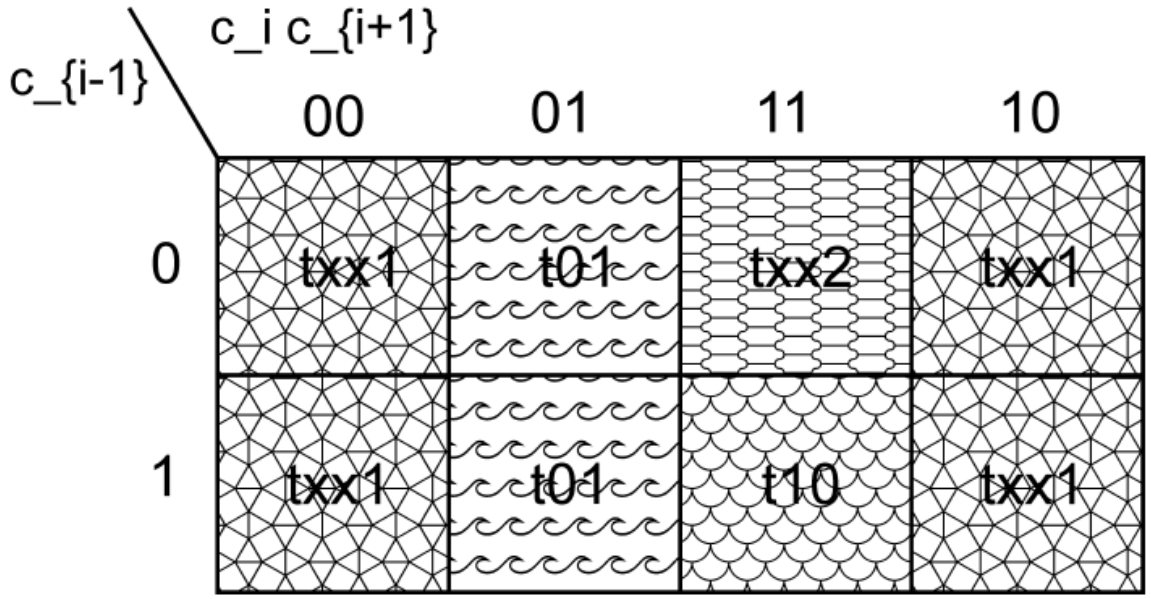
III. Simulating Linear Cellular Automaton 110



Rules 110:

$$R(0,0,0)=0 \quad R(0,1,0)=1 \quad R(1,0,0)=0 \quad R(1,1,0)=1$$
$$R(0,0,1)=1 \quad R(0,1,1)=1 \quad R(1,0,1)=1 \quad R(1,1,1)=0 \qquad (1)$$

Minimization of CA110 function



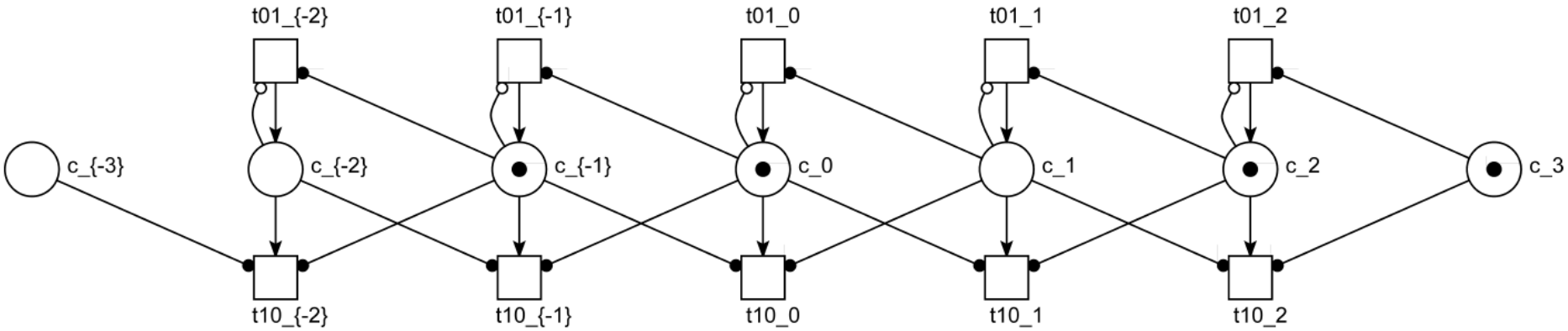
$$t10 = c_{i-1}c_ic_{i+1},$$

$$t01 = \bar{c}_{i-1}\bar{c}_ic_{i+1} \vee c_{i-1}\bar{c}_ic_{i+1} = \bar{c}_ic_{i+1},$$

$$txx = \bar{c}_{i-1}\bar{c}_i\bar{c}_{i+1} \vee \bar{c}_{i-1}c_i\bar{c}_{i+1} \vee \bar{c}_{i-1}c_ic_{i+1} \vee c_{i-1}\bar{c}_i\bar{c}_{i+1} \vee c_{i-1}c_i\bar{c}_{i+1} =$$

$$= \bar{c}_{i+1} \vee \bar{c}_{i-1}c_ic_{i+1}.$$

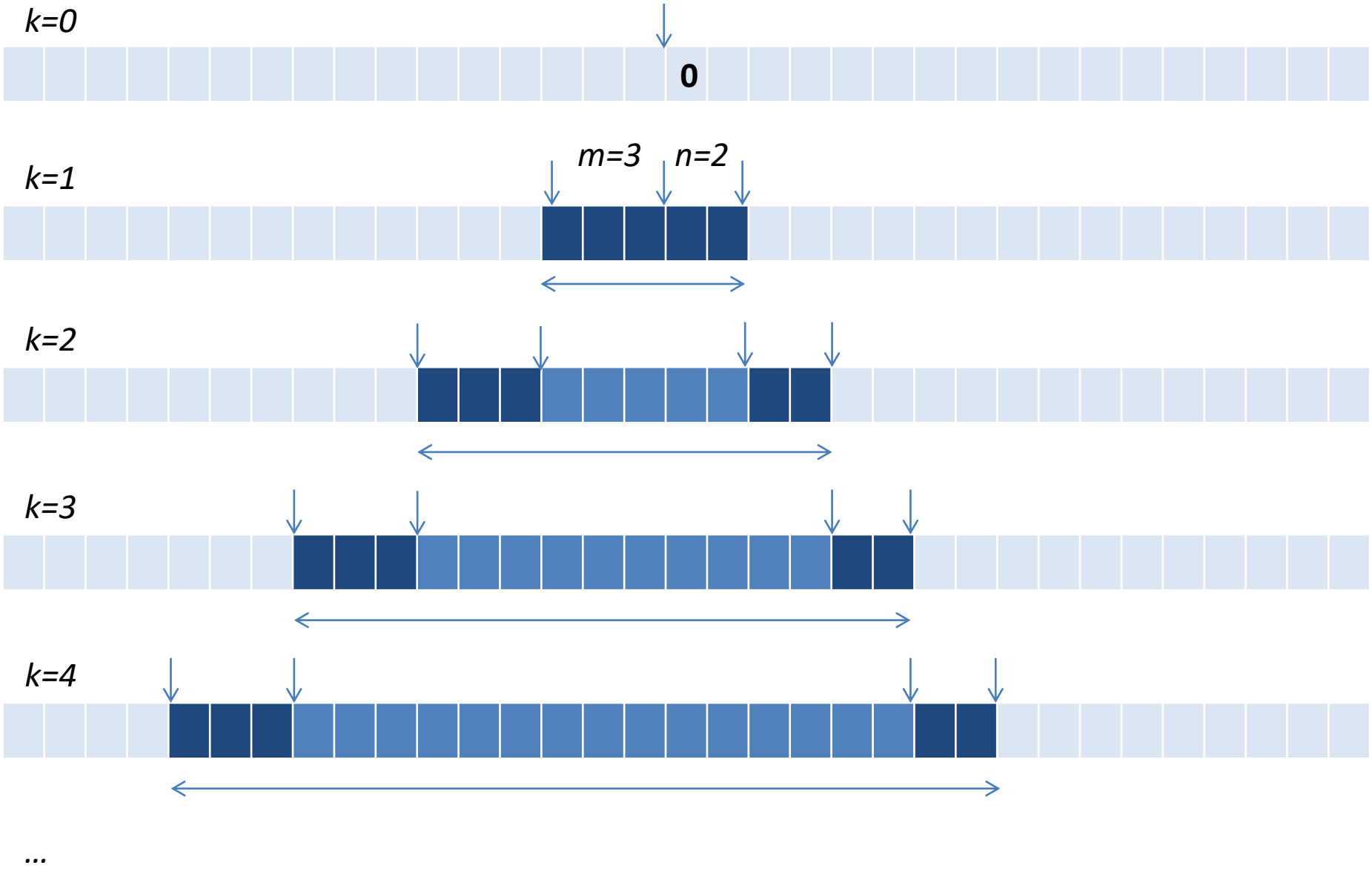
Synchronous PN with Inhibitor and Read Arcs



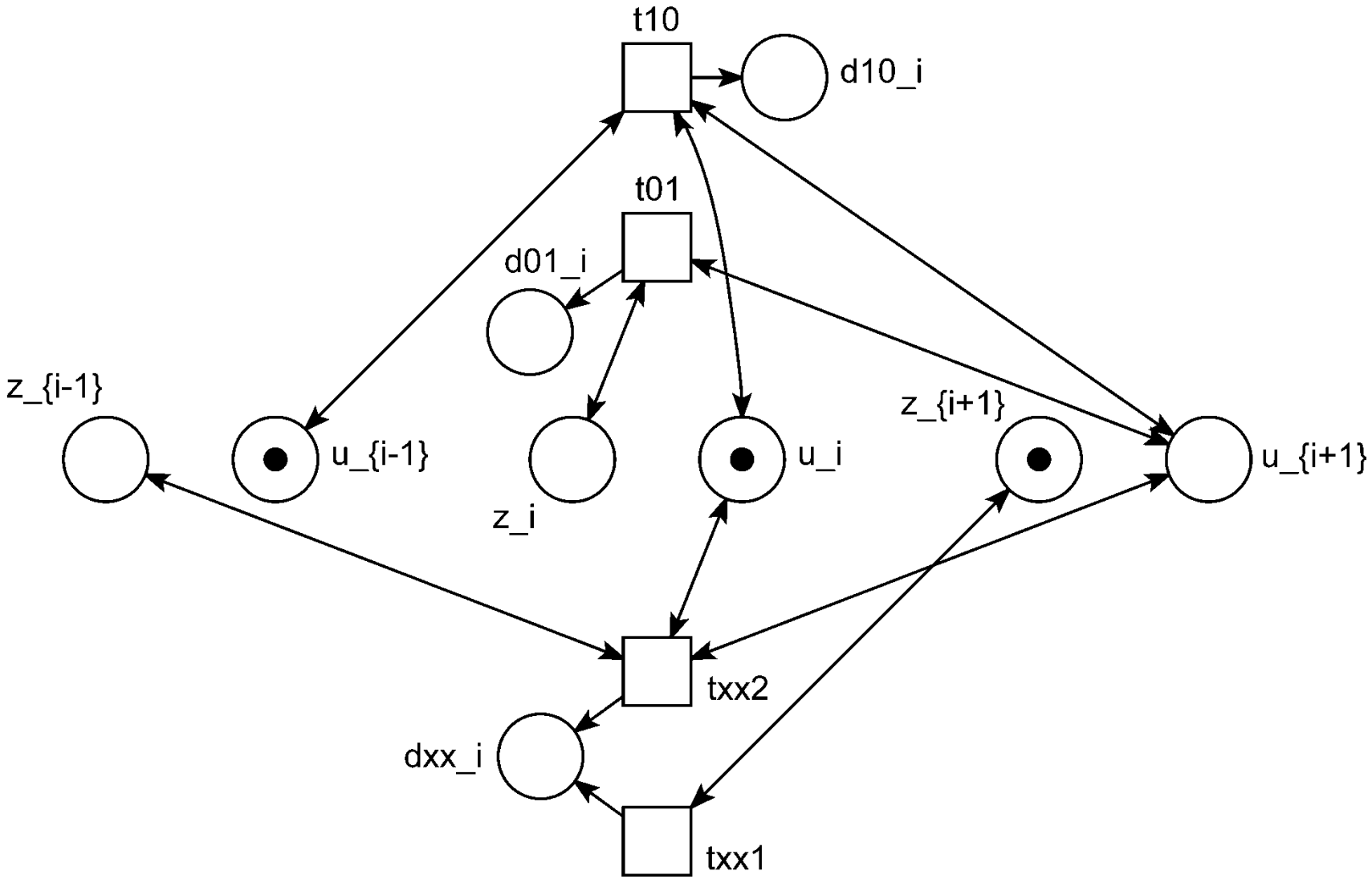
Parametric expression (PE):

$$\left(\begin{array}{l} t01_i : c_i = 0, c_{i+1} > 0 \rightarrow c_i, \\ t10_i : c_{i-1} > 0, c_i, c_{i+1} > 0 \rightarrow \end{array} \right)$$

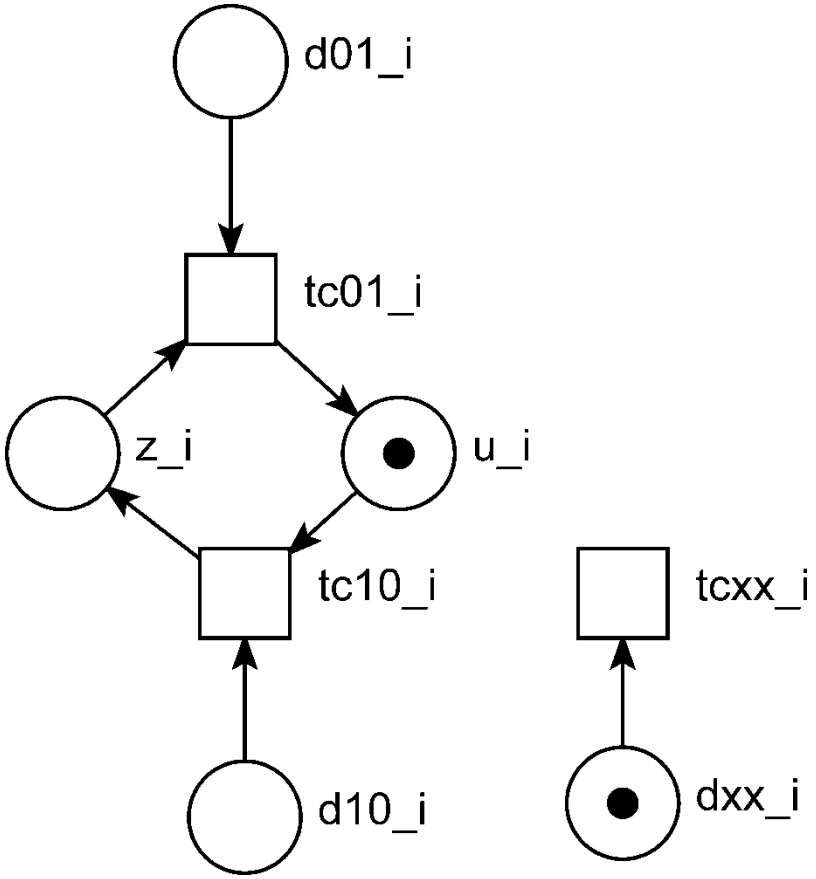
Expanding Traversals of the Cell Array for CA110 Simulation by Asynchronous Nets



DS_i – Calculate the State Difference



CS_i – Change the Cell State



Parametric Specification of DS_i and CS_i

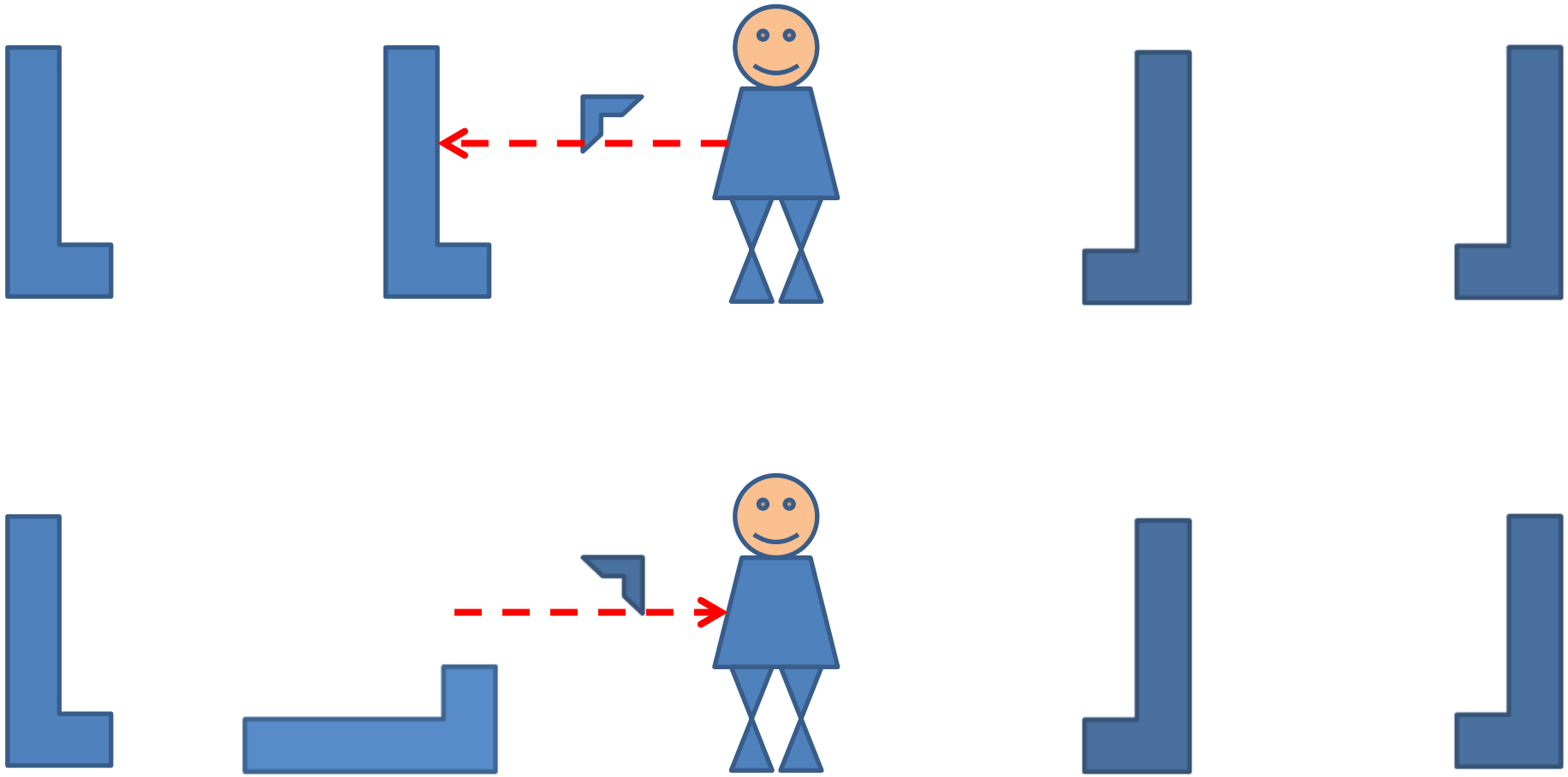
DS_i

$$\left(\begin{array}{l} t01_i : z_i, u_{i+1} \rightarrow z_i, u_{i+1}, d01_i, \\ t10_i : u_{i-1}, u_i, u_{i+1} \rightarrow u_{i-1}, u_i, u_{i+1}, d10_i, \\ txx1_i : z_{i+1} \rightarrow z_{i+1}, dxx_i, \\ txx2_i : z_{i-1}, u_i, u_{i+1} \rightarrow z_{i-1}, u_i, u_{i+1}, dxx_i \end{array} \right)$$

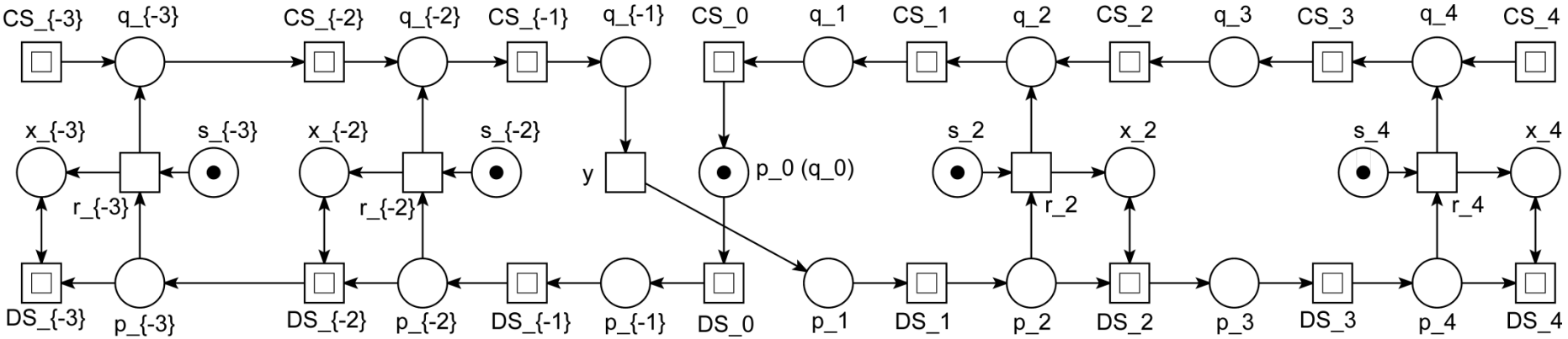
CS_i

$$\left(\begin{array}{l} tc01_i : d01_i, z_i \rightarrow u_i, \\ tc10_i : d10_i, u_i \rightarrow z_i, \\ tcxx_i : dxx_i \rightarrow \end{array} \right)$$

Boomerang and Barriers of UPN(9,12,Inf)



Boomerang and Barriers Net Picture



An example for *m = 1*, *n = 2*

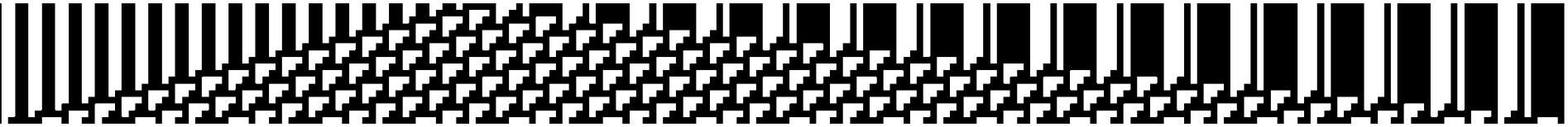
The fragment represents 8 cells

Boomerang and Barriers Net

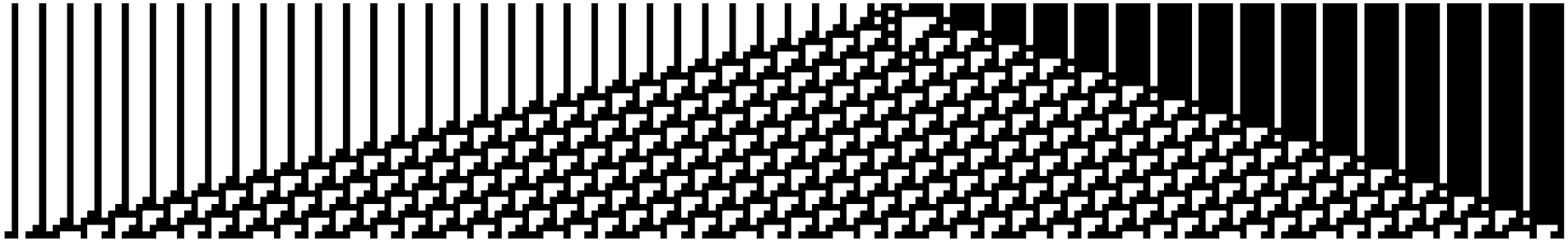
$$
\left(
\begin{array}{l}
DS_0 : p_0 \to p_{-1}, \\
CS_0 : q_1 \to p_0, \\
y : q_{-1} \to p_1, \\
p_0 = 1, \\
\left(\begin{array}{l} DS_i : p_i \to p_{i+d(i)}, \\ CS_i : q_{i+d(i)} \to q_i, \end{array} \right) : \\
i = -1 \vee (i < -1 \wedge |i+1| \; mod \; m \neq 0) \vee (i > 0 \wedge i \; mod \; n \neq 0), \\
\left(\begin{array}{l} DS_i : p_i, x_i \to p_{i+d(i)}, x_i, \\ CS_i : q_{i+d(i)}, \to q_i, \\ r_i : s_i, p_i \to x_i, q_i, \\ s_i = 1, \end{array} \right) : \\
(i < -1 \wedge |i+1| \; mod \; m = 0) \vee (i > 0 \wedge i \; mod \; n = 0), \\
d(i) = \left\{ \begin{array}{ll} -1, & i < 0, \\ 1, & i \geq 0 \end{array} \right.
\end{array}
\right)
$$

Visual Simulation of Ether

The left word *“1001”* and the right word *“1011111000”*;
m = 4 and *n = 10*



The left word *“0001”*, central word *“01110”*, and right word *“111110”*;
m = 4 and *n = 0* interchanges with *n = 6*



Simulating TMs which Simulate CA110

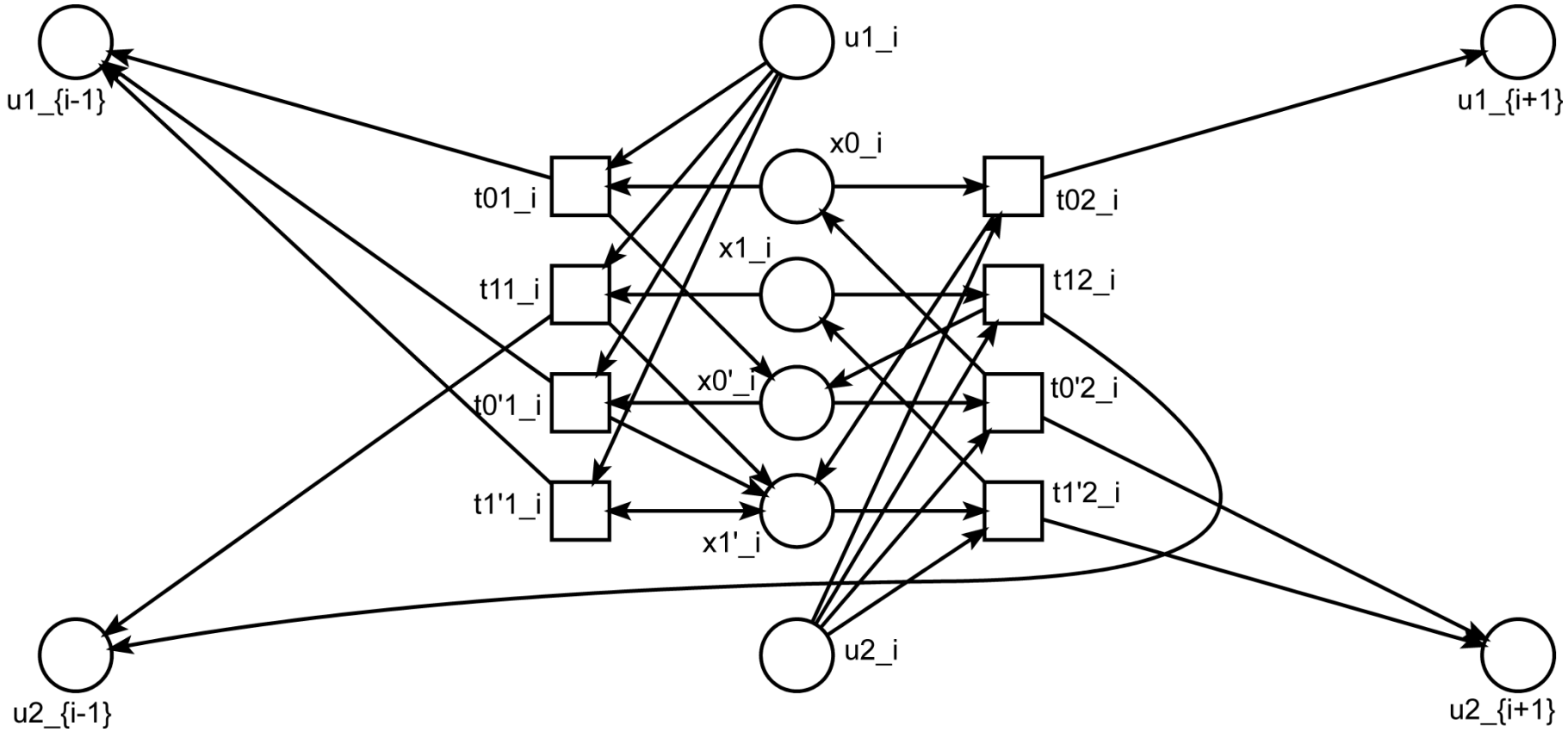
Neary & Woods' weakly universal TM with 2 states and 4 symbols WUTM(2,4)

	u_1	u_2
0	$\emptyset\ L\ u_1$	$\not{1}\ R\ u_1$
1	$\not{1}\ L\ u_2$	$\emptyset\ L\ u_2$
$\emptyset$	$\not{1}\ L\ u_1$	$0\ R\ u_2$
$\not{1}$	$\not{1}\ L\ u_1$	$1\ R\ u_2$

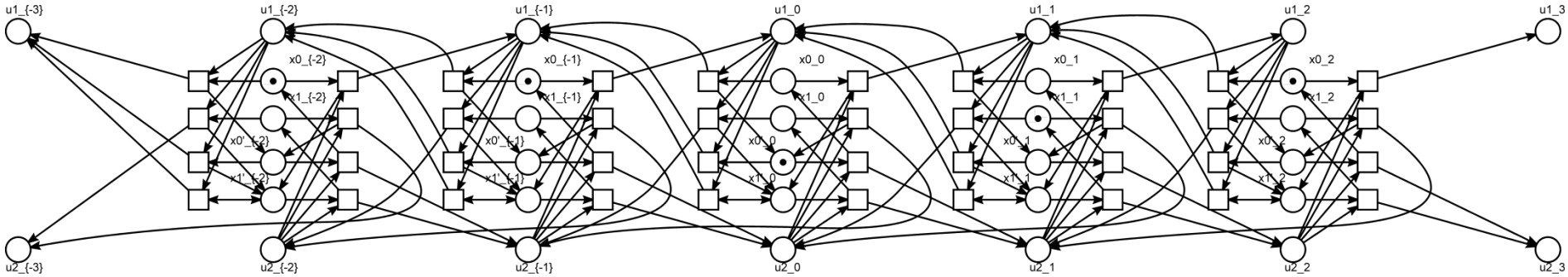
The left blank word: $w_l = 00\emptyset\,1$

The right blank word: $w_r = 0\not{1}\,\emptyset\,\emptyset\,0\not{1}$

Model of WUTM(2,4) Cell

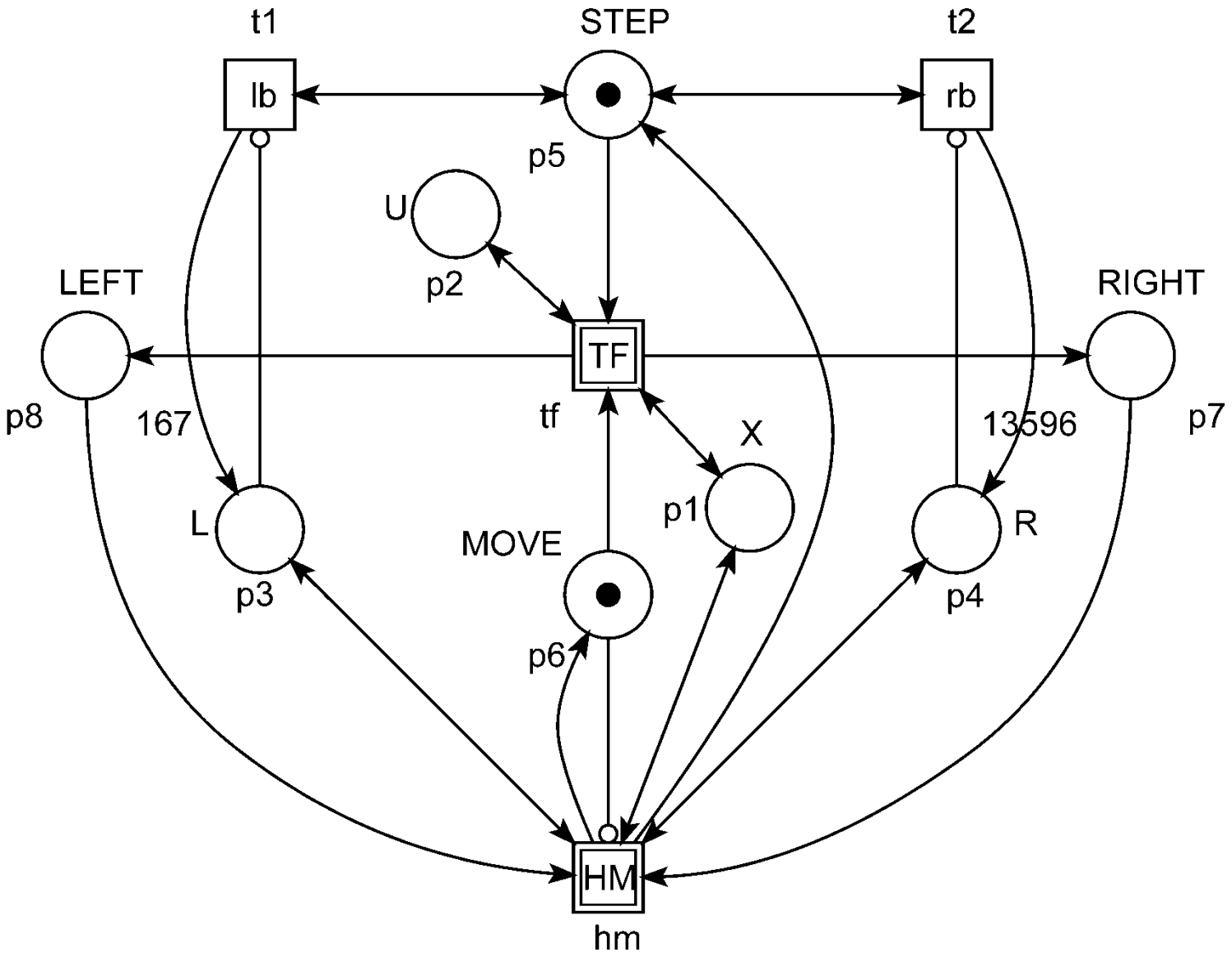


Model of WUTM(2,4) – UPN(6,8,inf)

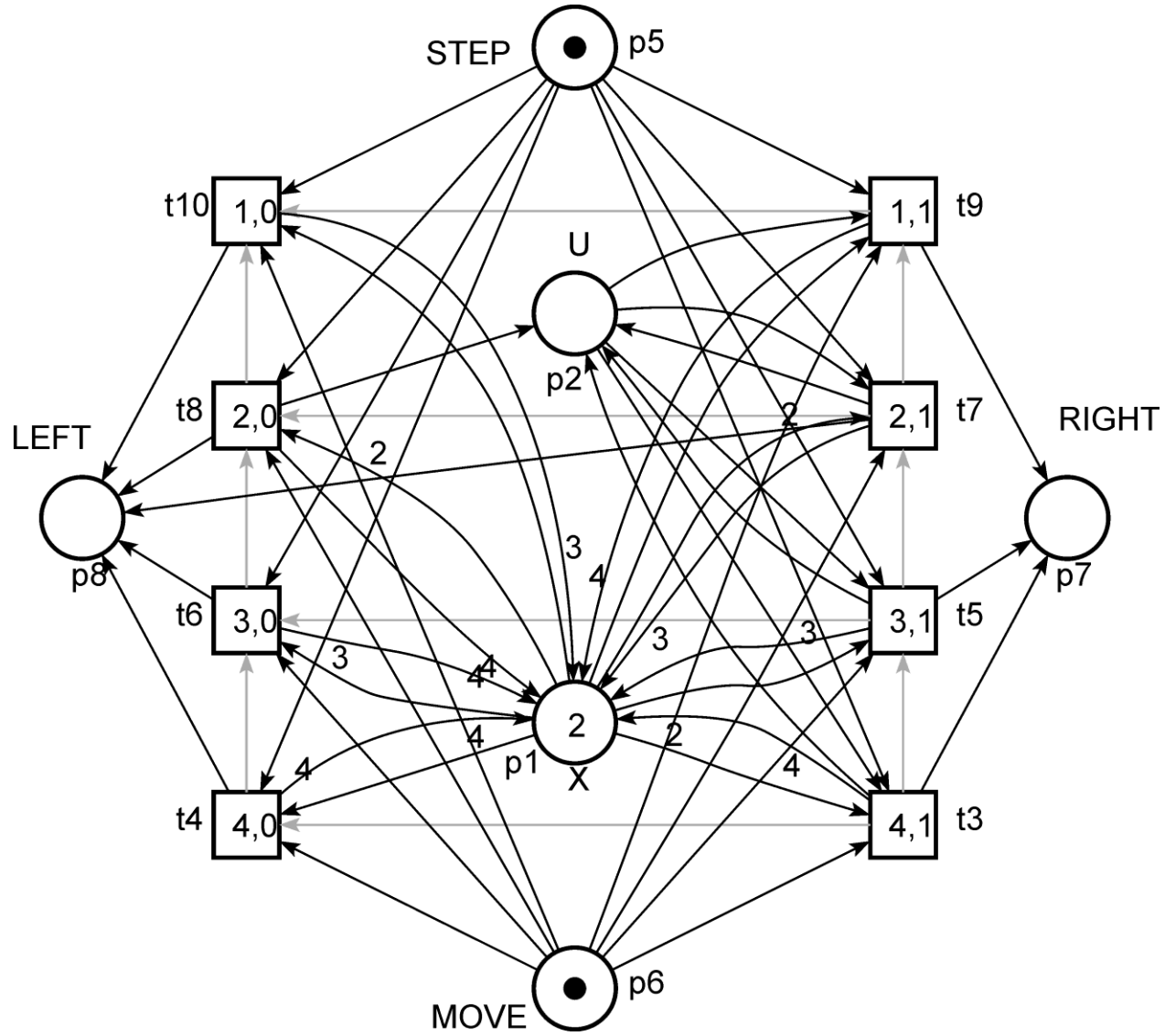


$$
\begin{pmatrix}
t(0,1)_i : x0_i, u1_i \to x\emptyset_i, u1_{i-1}, \\
t(0,2)_i : x0_i, u2_i \to x\not{1}_i, u1_{i+1}, \\
t(1,1)_i : x1_i, u1_i \to x\not{1}_i, u2_{i-1}, \\
t(1,2)_i : x1_i, u2_i \to x\emptyset_i, u2_{i-1}, \\
t(\emptyset,1)_i : x\emptyset_i, u1_i \to x\not{1}_i, u1_{i-1}, \\
t(\emptyset,2)_i : x\emptyset_i, u2_i \to x0_i, u2_{i+1}, \\
t(\not{1},1)_i : x\not{1}_i, u1_i \to x\not{1}_i, u1_{i-1}, \\
t(\not{1},2)_i : x\not{1}_i, u2_i \to x1_i, u2_{i+1},
\end{pmatrix}
$$

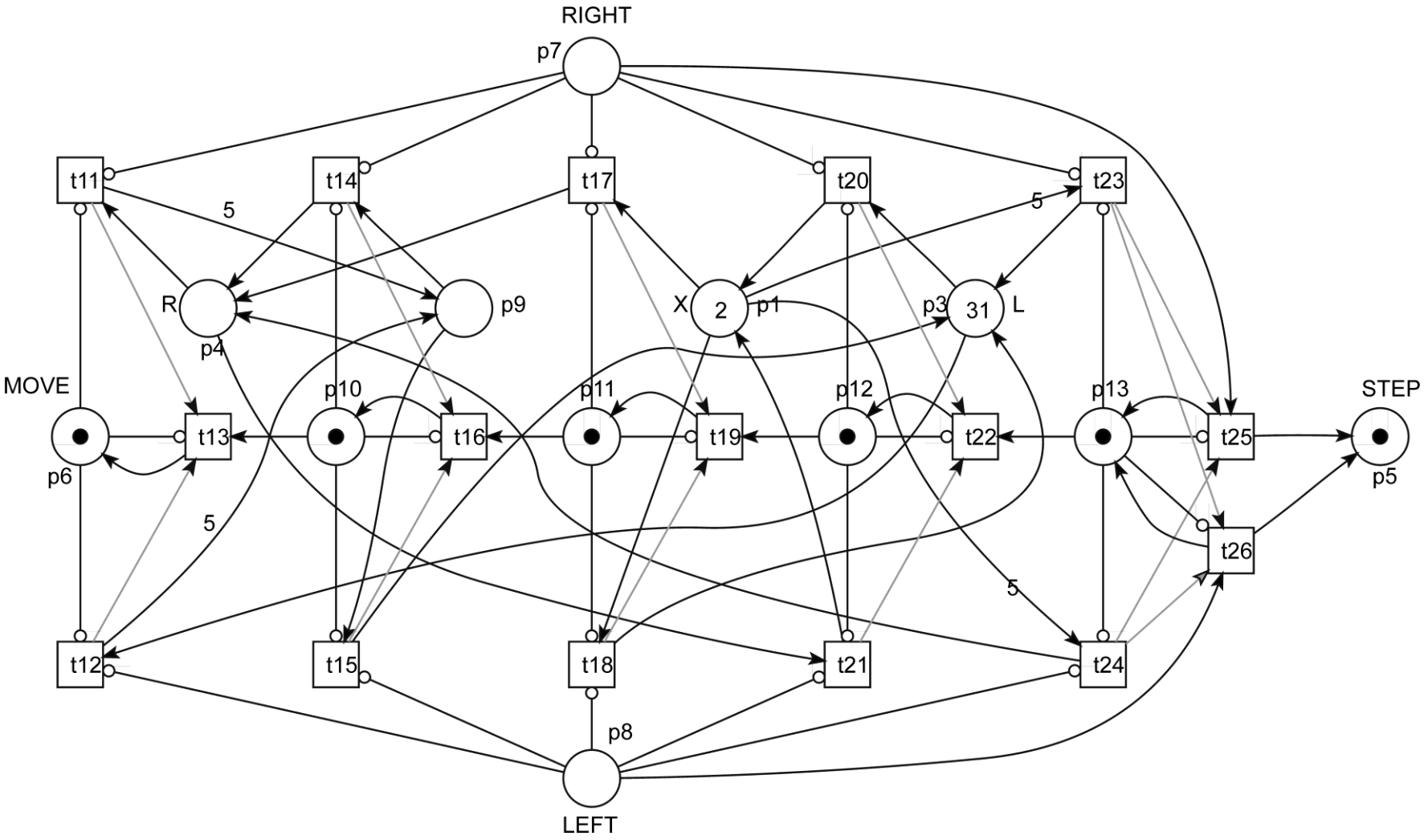
IV. Universal Sleptsov Net USN(13,26)



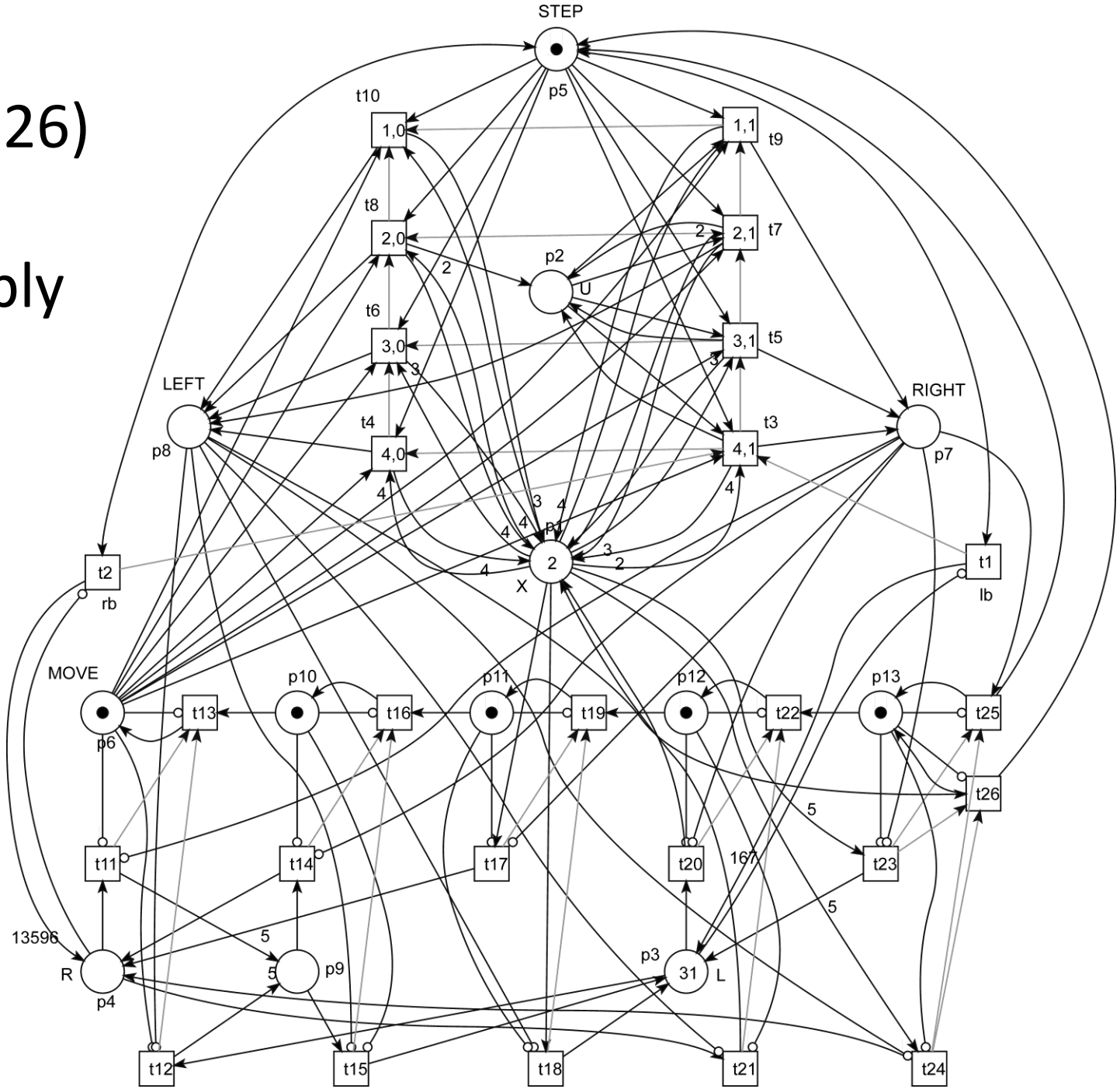
Transition function subnet TF



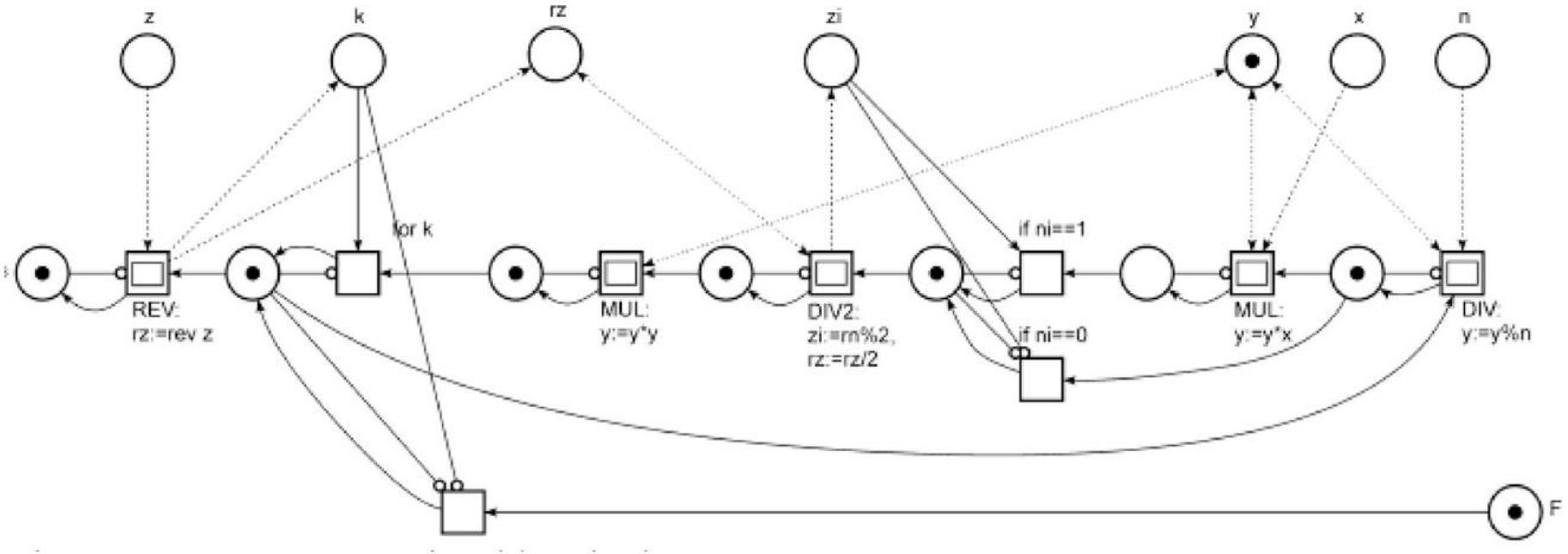
Head move subnet HM



USN(13,26) Final Assembly



V. Examples of SN programs: RSA encoding/decoding



$$y = x^z \bmod n$$

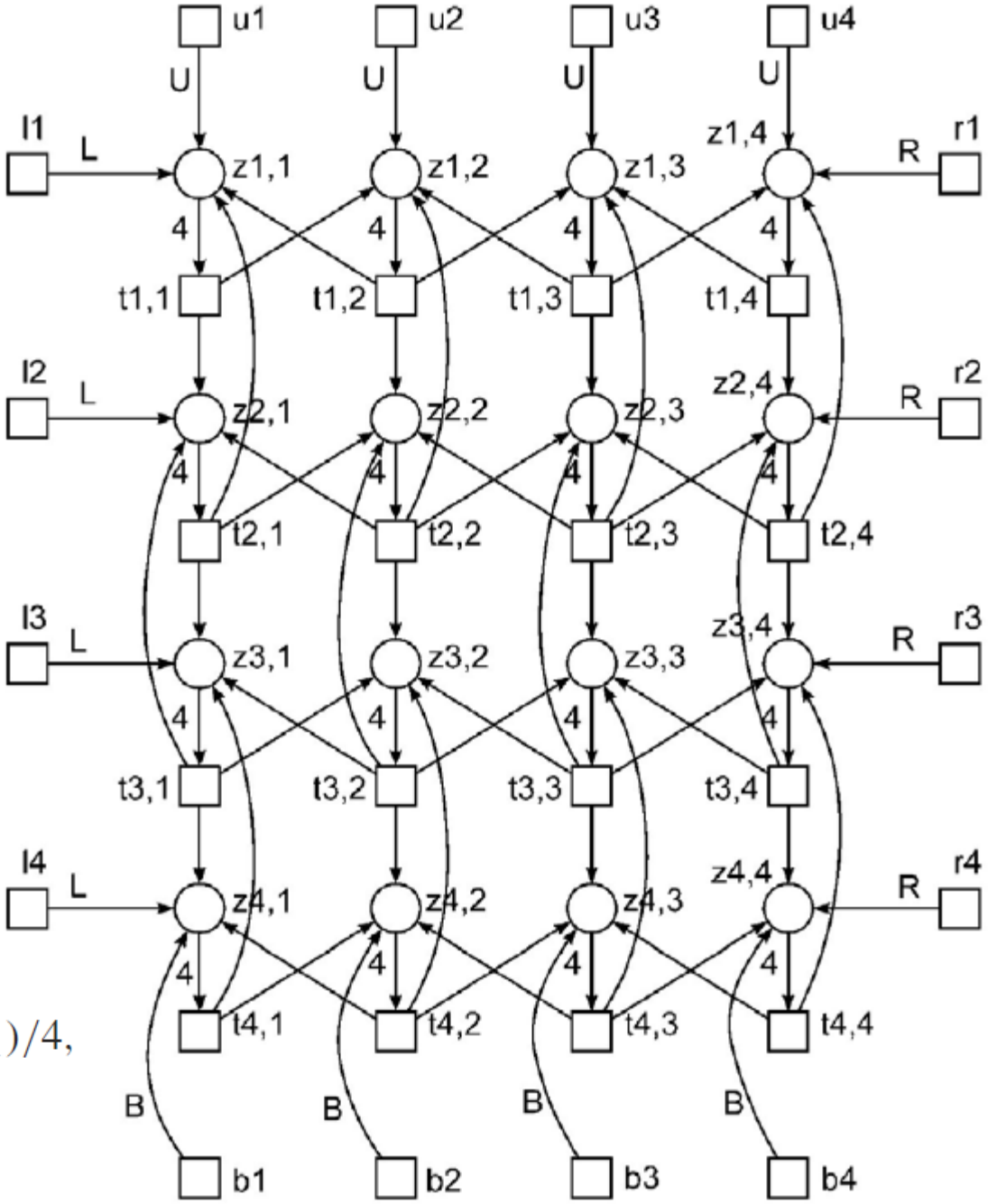
$$x^z = x^{(\ldots((z_k \cdot 2 + z_{k-1}) \cdot 2 + z_{k-2}) \ldots + z_2) \cdot 2 + z_1}$$

$$= \left(\ldots \left(\left(\left(x^{z_k} \right)^2 \cdot x^{z_{k-1}} \right)^2 \cdot x^{z_{k-2}} \right)^2 \cdot \ldots \cdot x^{z_2} \right)^2 \cdot x^{z_1}$$

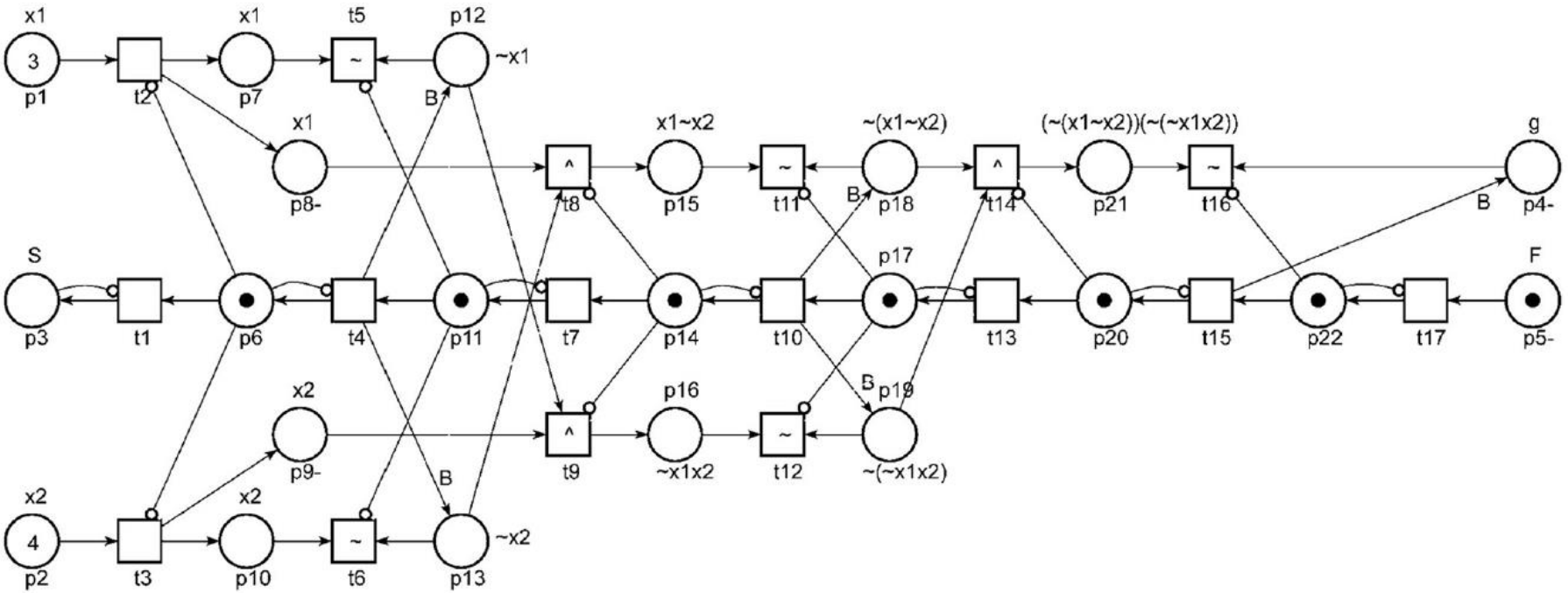
Examples of SN programs: Solving Laplace equation

$$\frac{\partial^2 \varphi}{\partial x^2} + \frac{\partial^2 \varphi}{\partial y^2} = 0$$

$$\varphi_{i,j} = (\varphi_{i-1,j} + \varphi_{i+1,j} + \varphi_{i,j-1} + \varphi_{i,j+1})/4,$$
$$\varphi_{i,j} = \varphi(x_i, y_j).$$

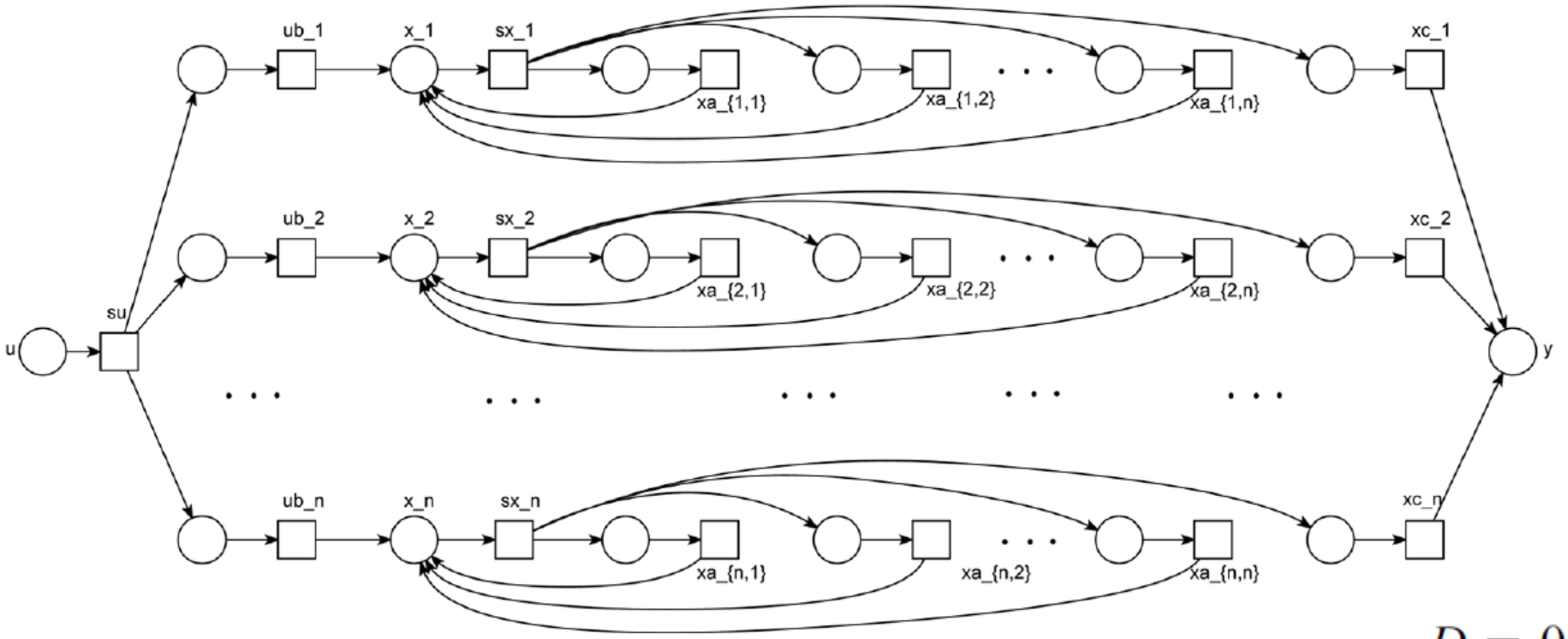


Examples of SN programs: Computing Fuzzy Logic Function



$$\varphi = x_1\bar{x}_2 \vee \bar{x}_1 x_2$$

Examples of SN programs: Discrete-Time Linear Control in Two Tacts



$$D = 0$$

$$\begin{cases} x(k+1) = Ax(k) + Bu(k) \\ y(k) = Cx(k) + Dy(k) \end{cases}$$

Conclusions

- Petri nets run exponentially slower with regard to Turing machines while Sleptsov nets run fast
- Universal Sleptsov net is a prototype of a processor in the Sleptsov net paradigm of computing; the smallest contain 39 nodes and run in polynomial time
- Sleptsov net computing offers: graphical concurrent language, formal verification of concurrent programs, fine granulation of parallel processes, massively parallel computations

http://member.acm.org/~daze

Basic References

- Zaitsev D.A. Universal Sleptsov Net, International Journal of Computer Mathematics, 2017
- Zaitsev D.A. Simulating Cellular Automata by Infinite Petri Nets, Journal of Cellular Automata, 2017
- Zaitsev D.A., Jürjens J. Programming in the Sleptsov Net Language for Systems Control, Advances in Mechanical Engineering, 2016, Vol. 8(4), 1–11.
- Zaitsev D.A. Sleptsov Nets Run Fast, IEEE Transactions on Systems, Man, and Cybernetics: Systems, 2016, Vol. 46(5), 682–693.
- Zaitsev D.A. Paradigm of Computations on the Petri Nets, Automation and Remote Control, 2014, Vol. 75(8), 1369–1383.
- Zaitsev D.A. Toward the Minimal Universal Petri Net, IEEE Transactions on Systems, Man, and Cybernetics: Systems, 2014, Vol. 44(1), 47–58.
- Zaitsev D.A. Inhibitor Petri Net Executing an Arbitrary Given Markov Normal Algorithm, Automatic Control and Computer Sciences, 2012, Vol. 46(7), 345–355.
- Zaitsev D.A. Universal Petri net, Cybernetics and Systems Analysis, 2012, Vol. 48(4), 498–511.

#SleptsovNets

#СетиСлепцова