

International Humanitarian University

<http://mgu.edu.ua>

Sleptsov Net Computing

Dmitry Zaitsev

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

Write Programs

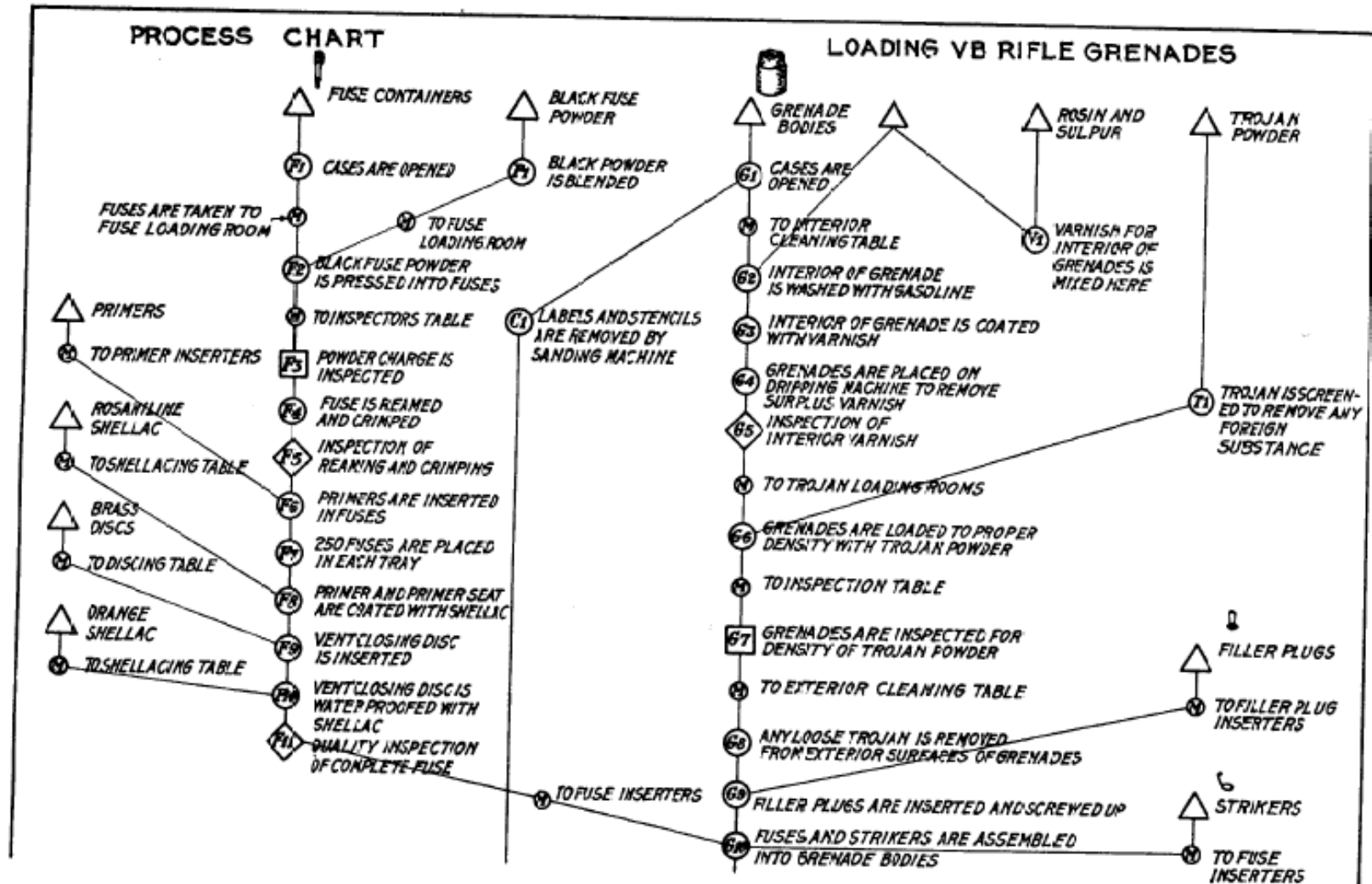
or

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

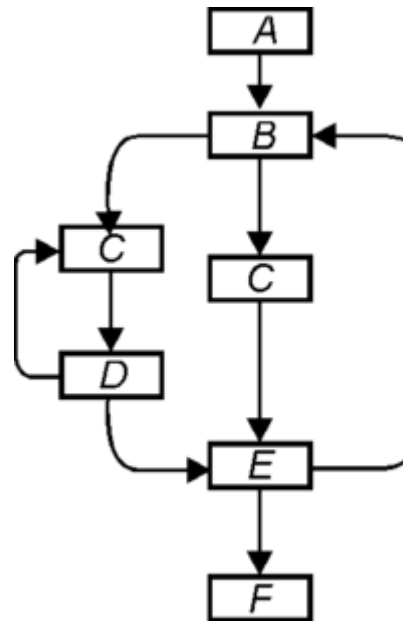
Frank and Lillian Gilbreth example



Program schemata



Yanov, 1958



Martiniuk, 1961

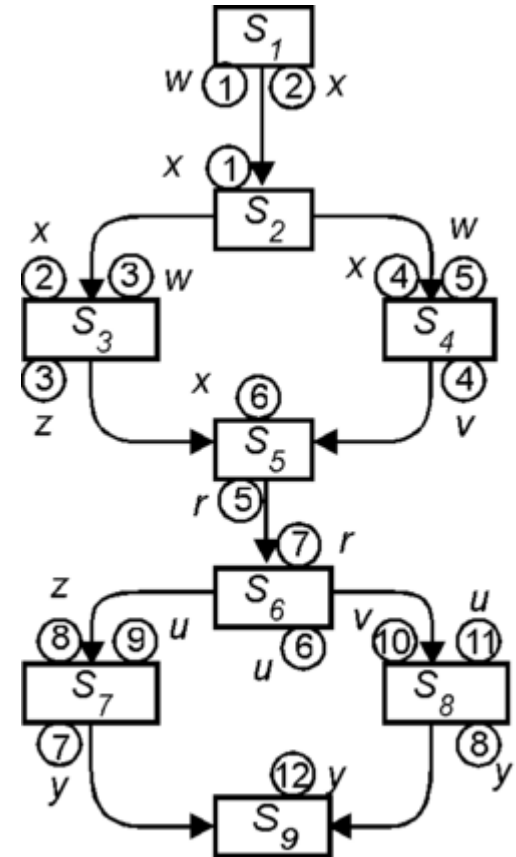


Схема Лаврова

Lavrov, 1961

R-technology of programming

```

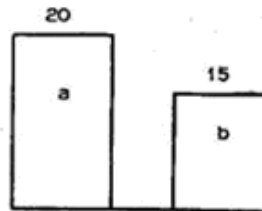
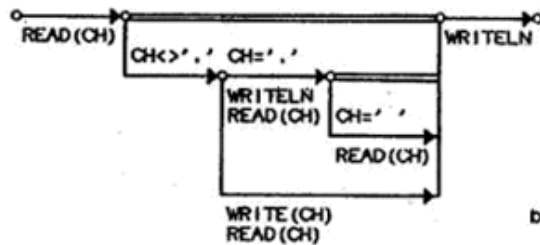
PROGRAM PRINID(INPUT,OUTPUT);
VAR CH:CHAR;
BEGIN
  READ(CH);
  WHILE CH<>'.' DO
    IF CH='.' THEN
      BEGIN
        WRITELN;
        READ(CH)
        WHILE CH='.' DO
          READ(CH)
        END
      END
    ELSE
      BEGIN
        WRITE(CH);
        READ(CH)
      END
    END;
  WRITELN
END.

```

```

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

```

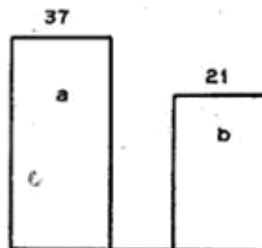
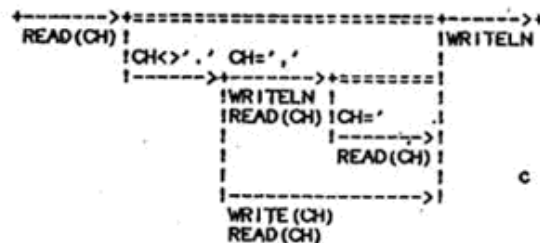


Число строк на экране
дисплея или на листинге

```

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

```



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

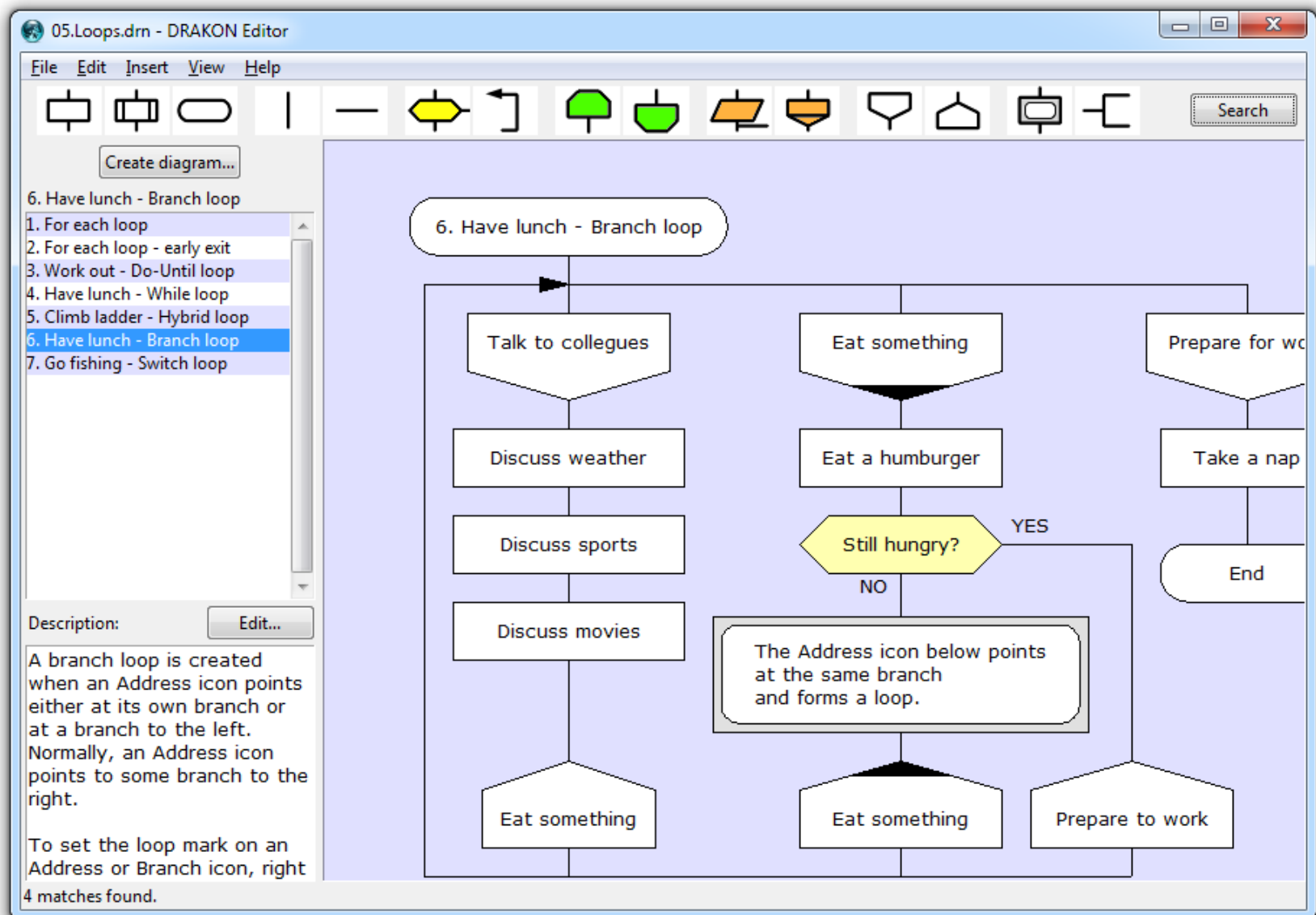
d

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

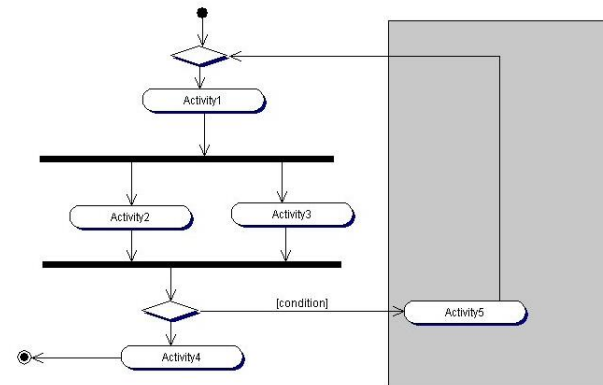
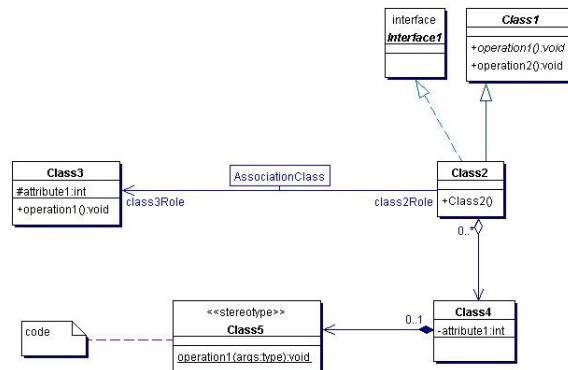
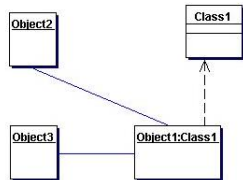
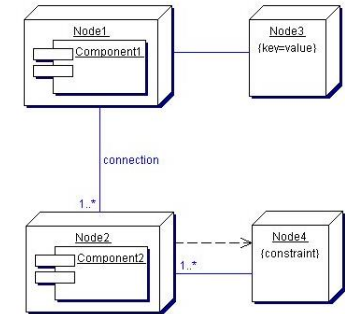
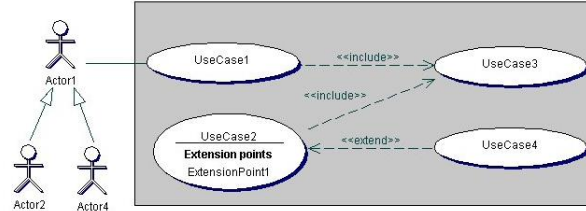
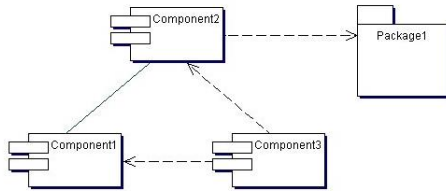
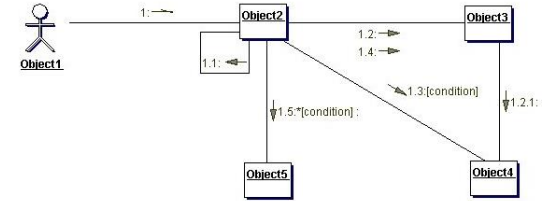
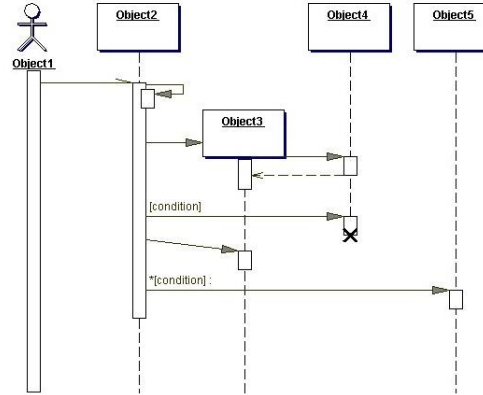
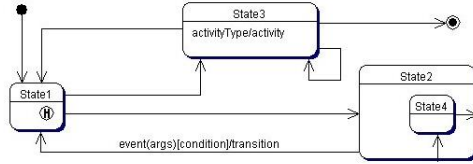
Modern Visual Programming

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

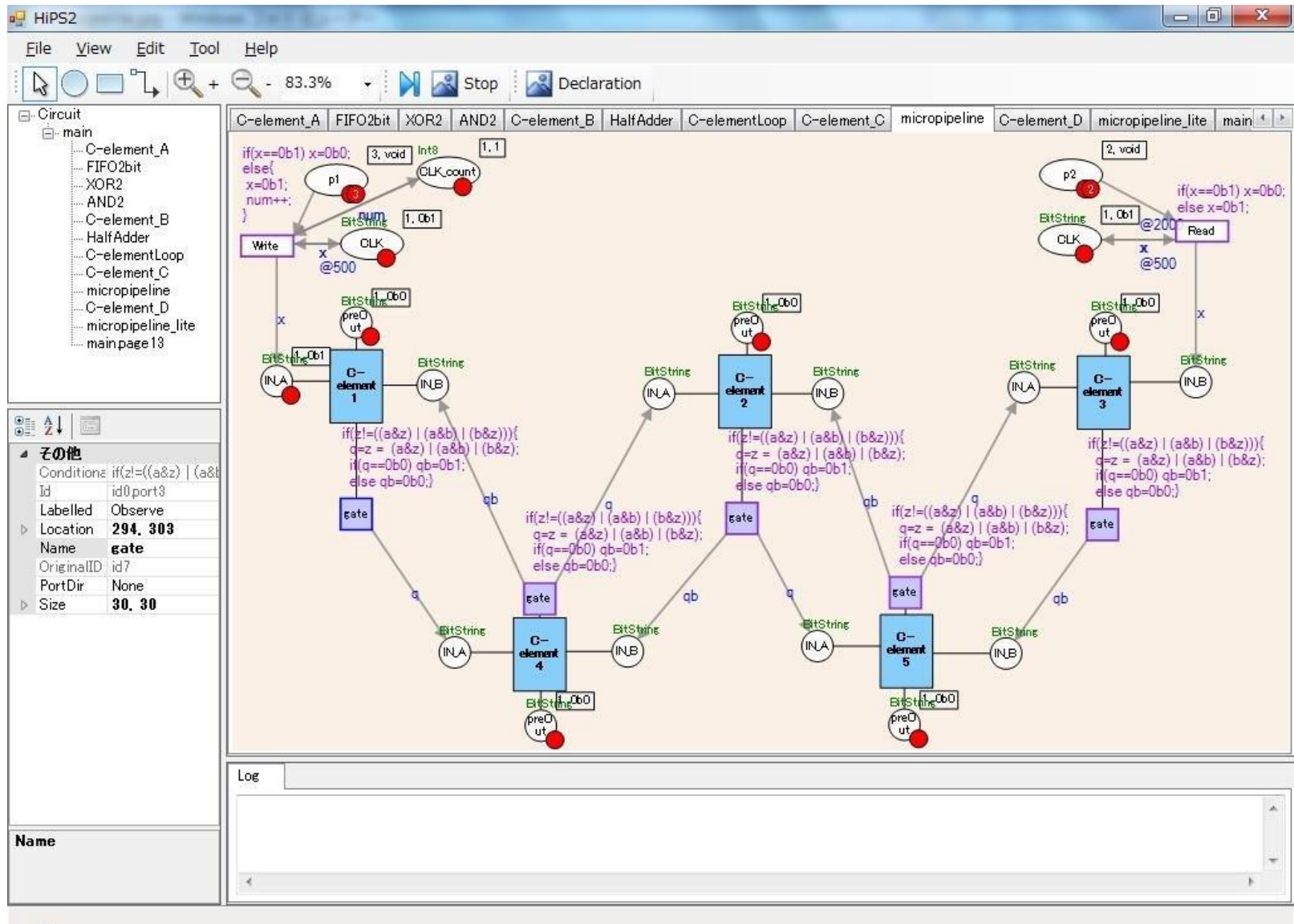
DRAKON



UML



Programming on Petri nets



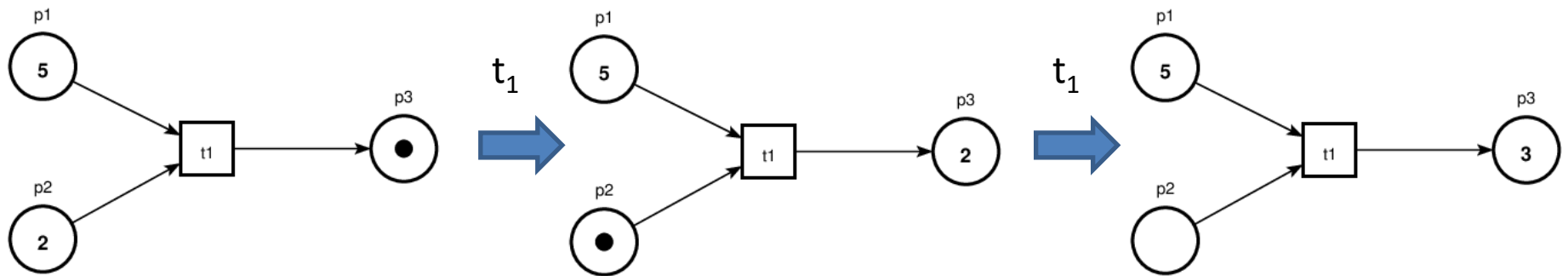
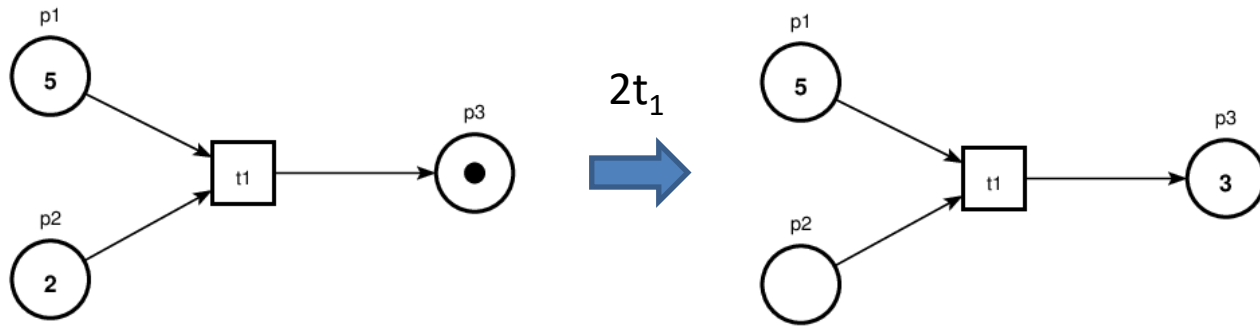
Path to a uniform concept

- 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

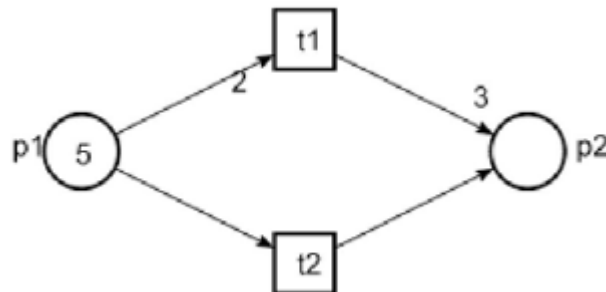
Transition firing strategy

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

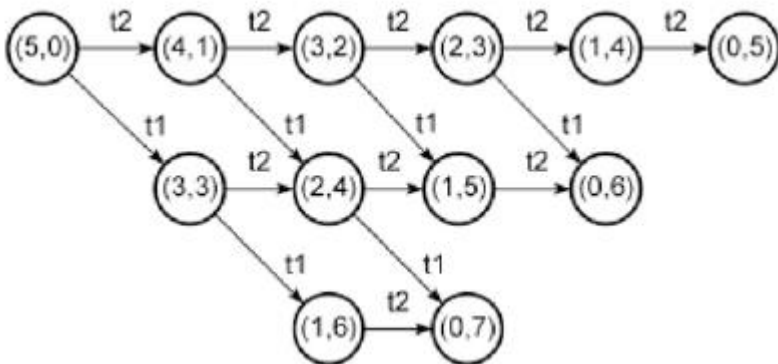
Sleptsov net vs Petri net



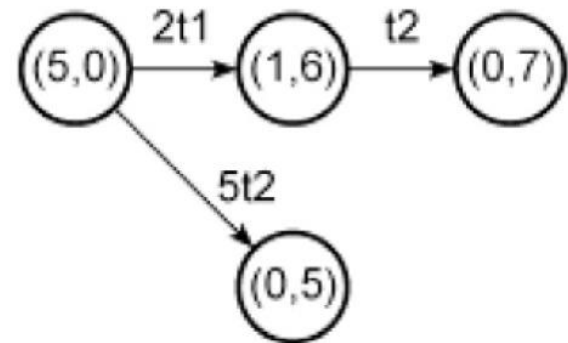
Sleptsov Net – Multiple Firing



Reachability graphs



Petri net



Sleptsov net

Sleptsov Nets Run Fast

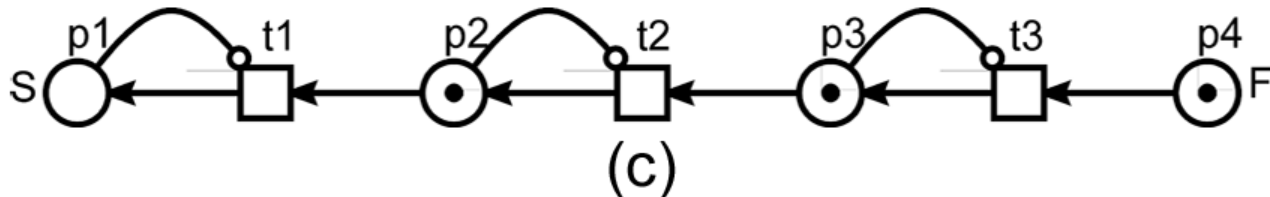
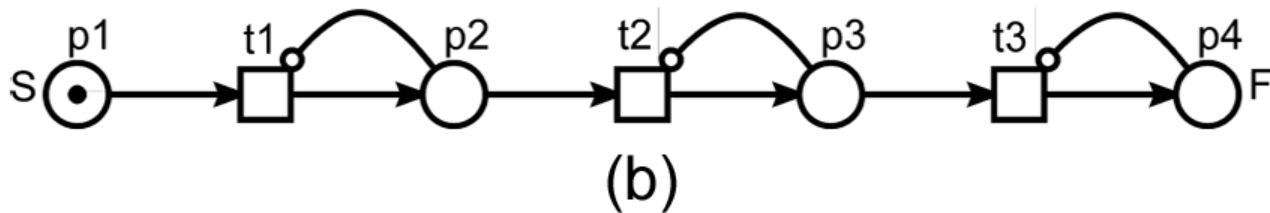
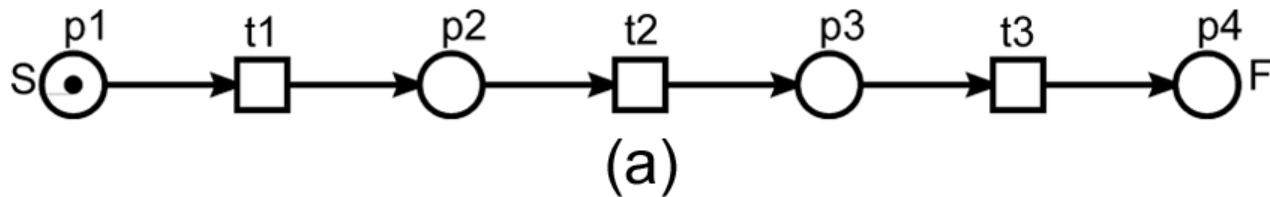
$$x \succ y = \begin{cases} x/y, & \text{if } y > 0 \\ 0, & \text{if } y = -1, x > 0 \\ \infty, & \text{if } y = -1, x = 0. \end{cases}$$

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

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	$(x / y) \cdot (2 \cdot y + 2) + (x \% y) + y + 4$	$39 \cdot (\log_2 x - \log_2 y) + 19$

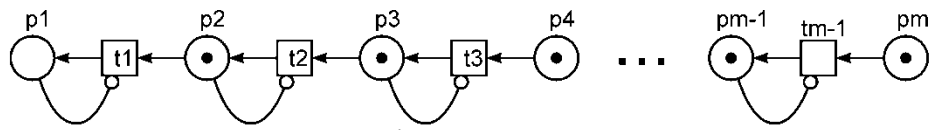
I. Peculiarities of programming in SNs



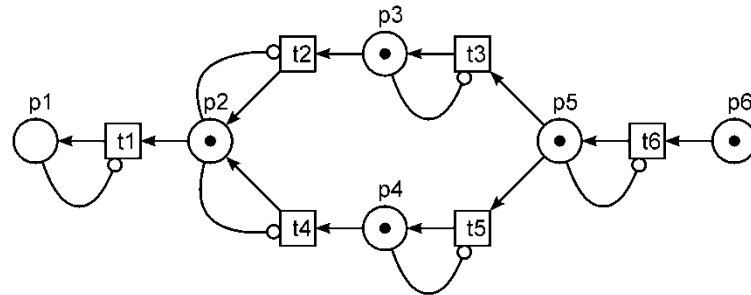
A. Reversed control flow (c)

B. Using inhibitor arcs to control a transition firing

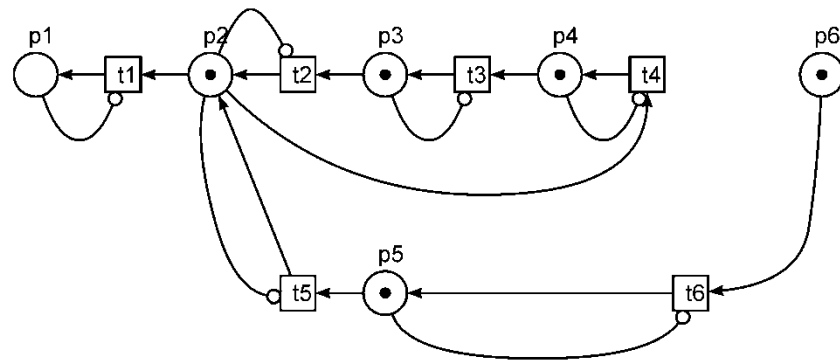
Basic operators



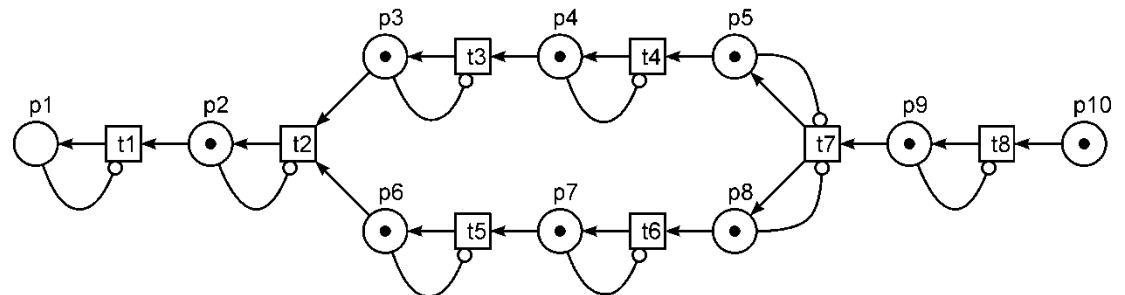
a) sequence;



b) branching;

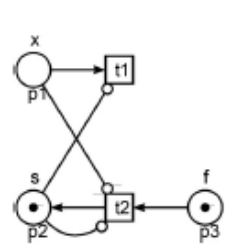


c) cycle;



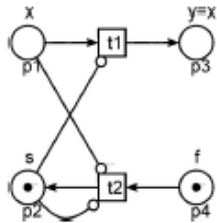
d) Parallel execution (split t_2 and join t_7)

Basic subnets (subroutines)



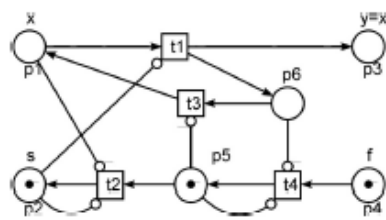
CLEAN:

$x := 0$



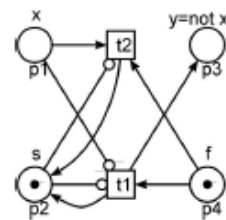
MOVE:

$y := x, x := 0$



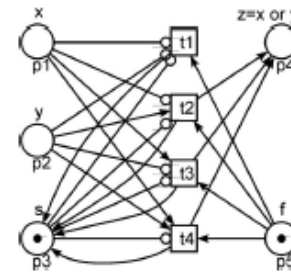
COPY:

$y := x$



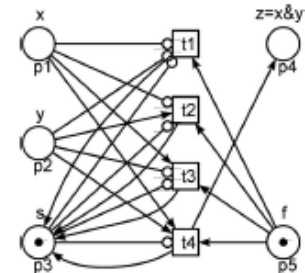
NOT(x):

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



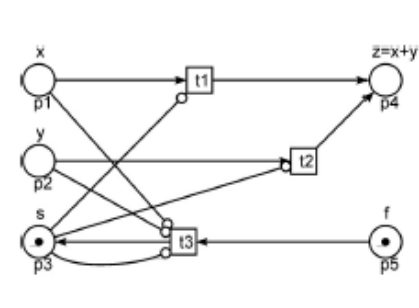
OR(x,y)

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



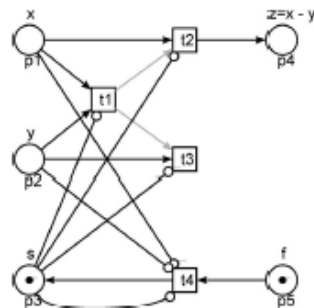
AND(x,y):

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



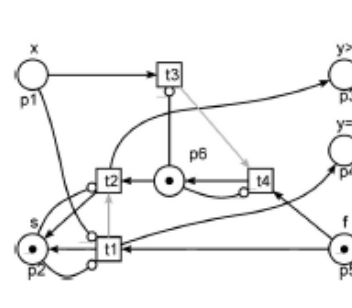
ADD(x,y):

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



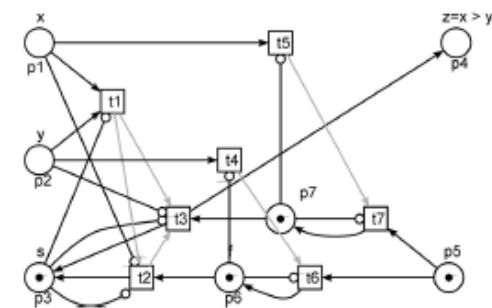
SUB(x,y):

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



GT0(x):

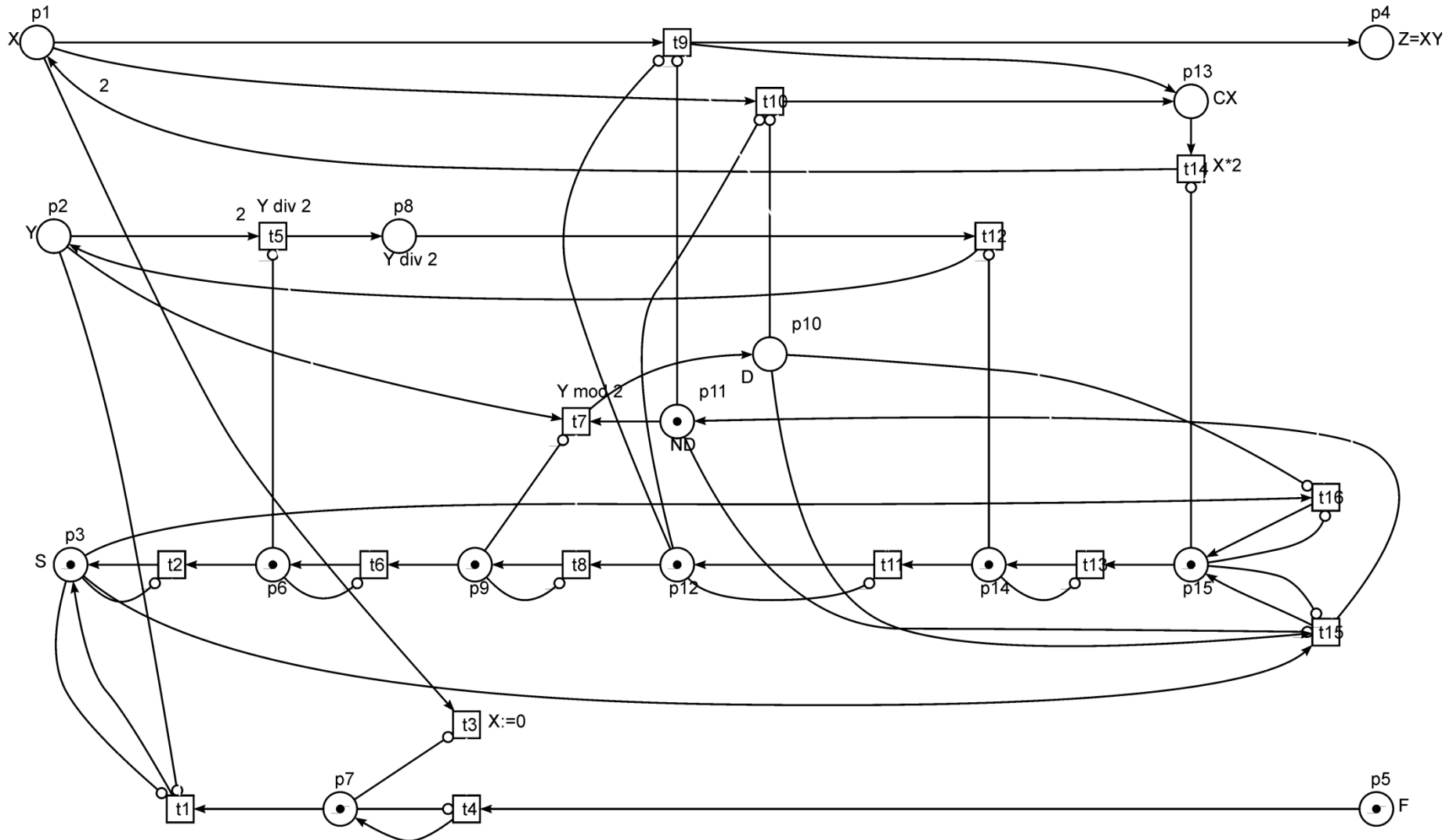
$y := (x > 0), z := (x = 0)$



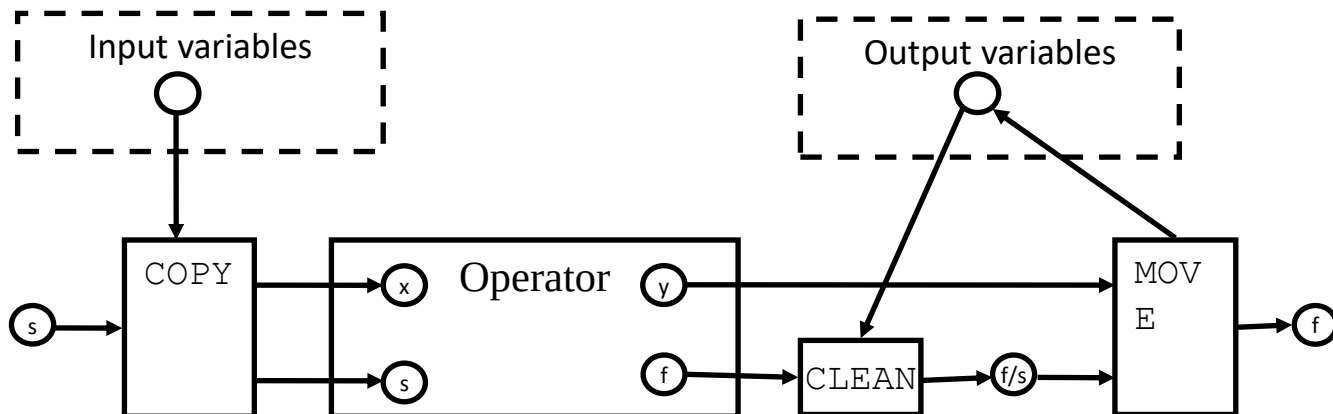
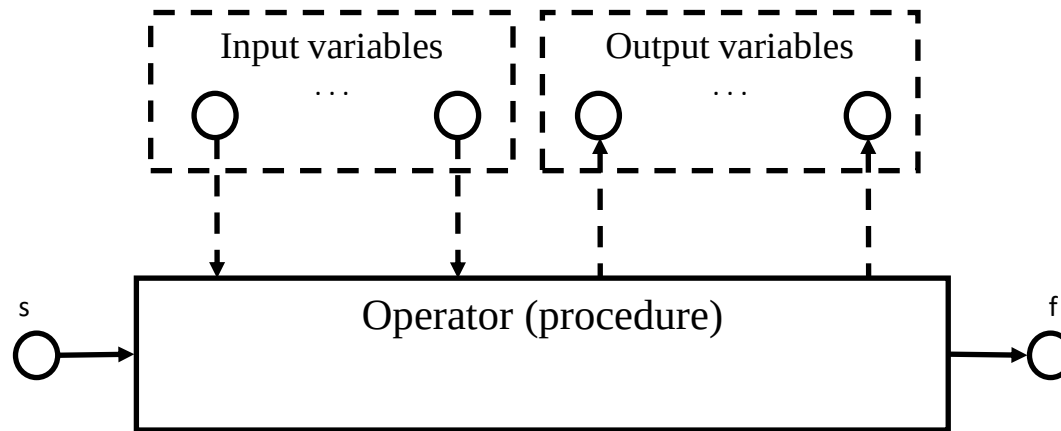
GT(x,y):

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

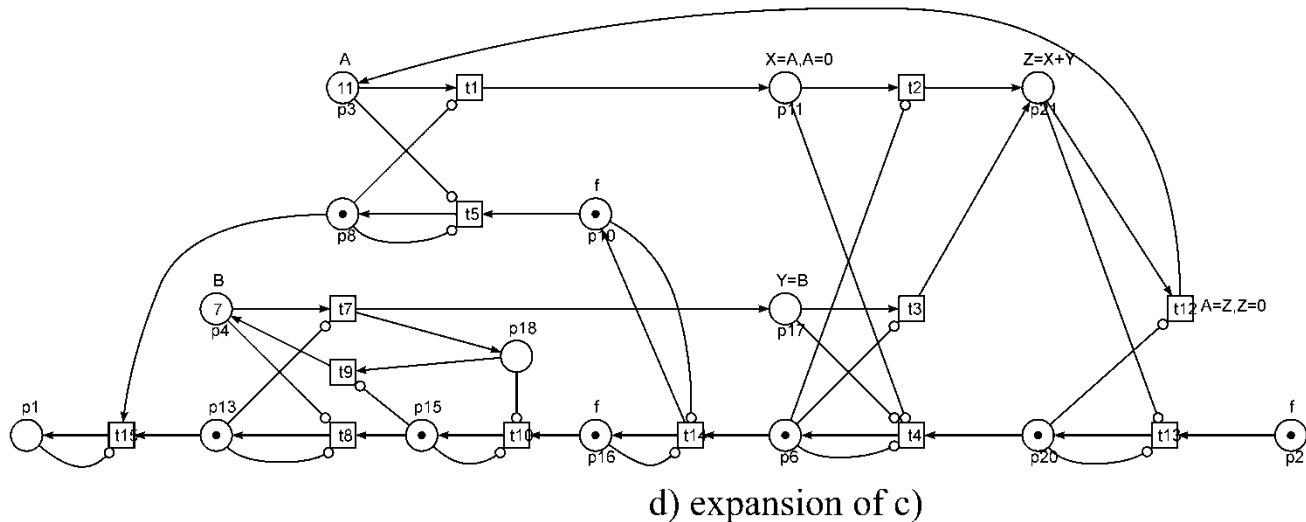
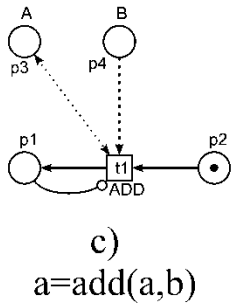
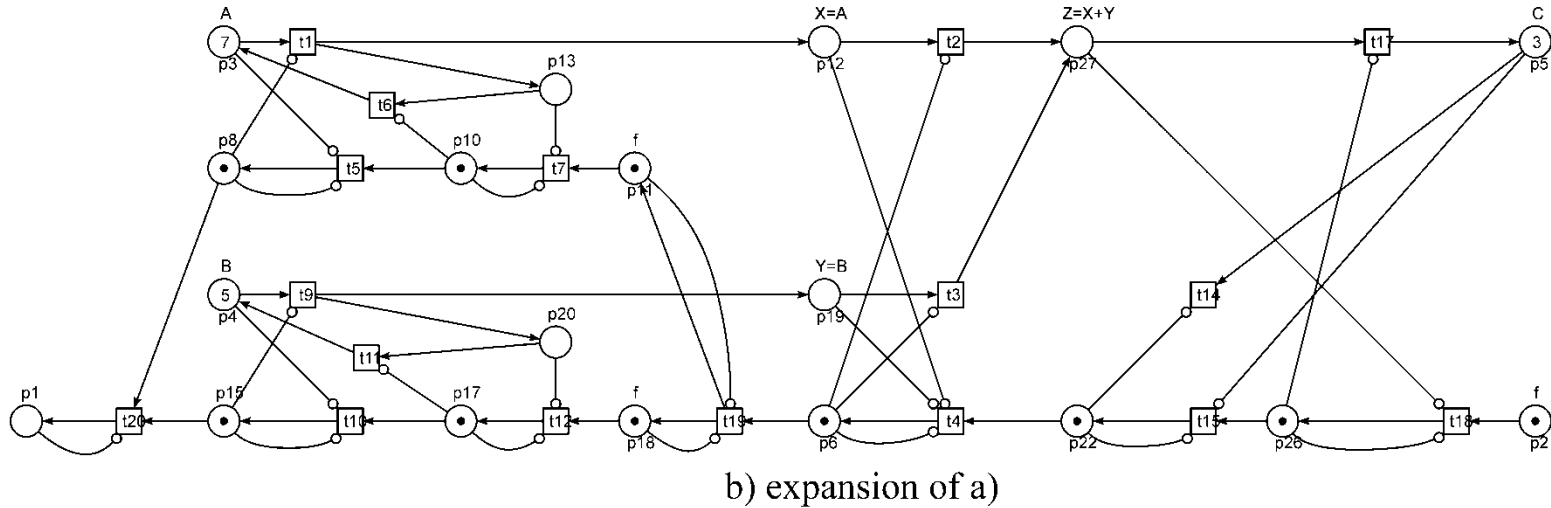
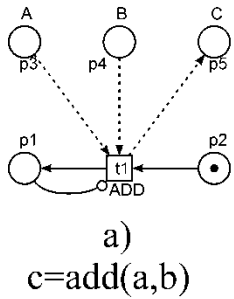
Fast multiplication in Sleptsov nets



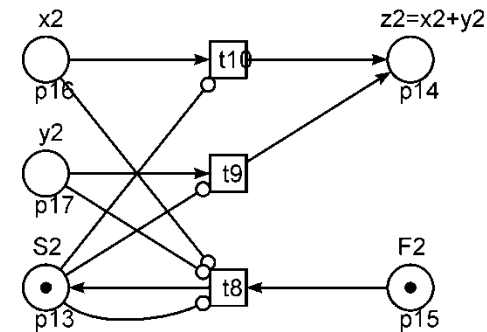
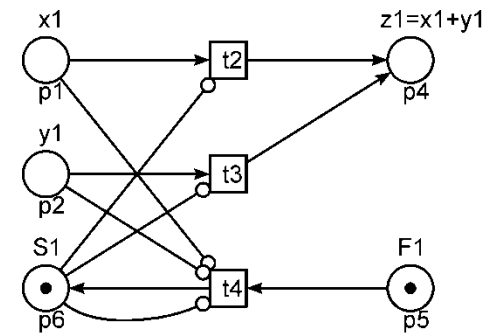
Work with variables



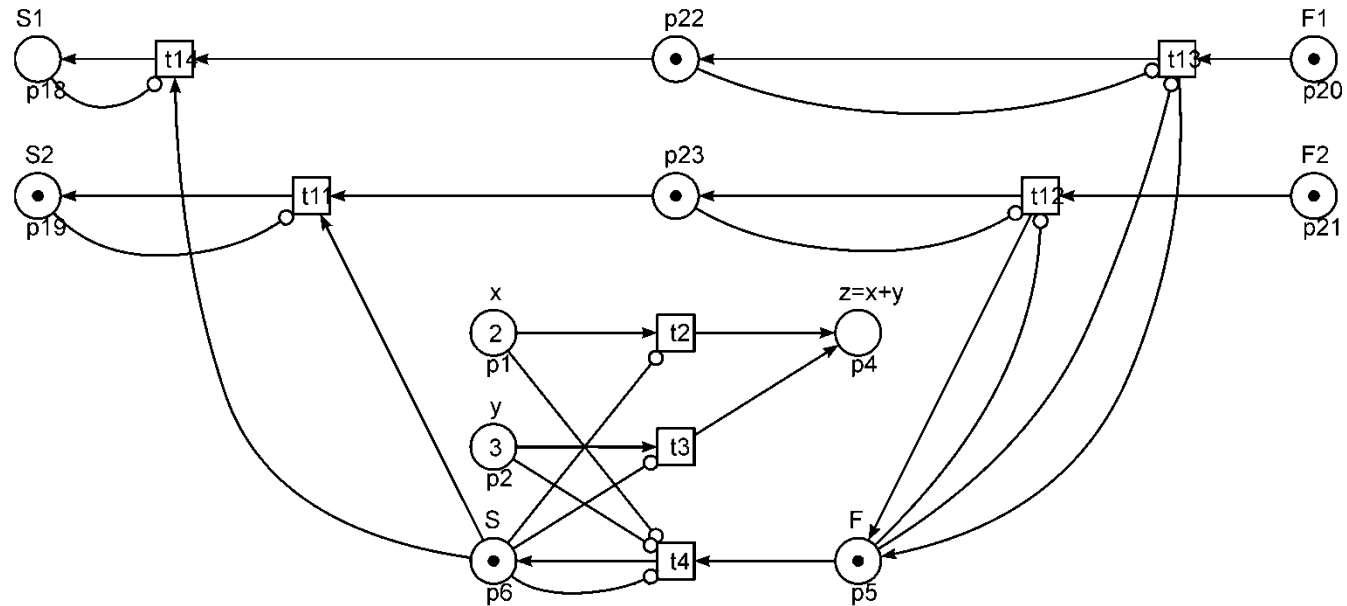
Expansion of dashed/dotted arcs



Subnets (routines) calls

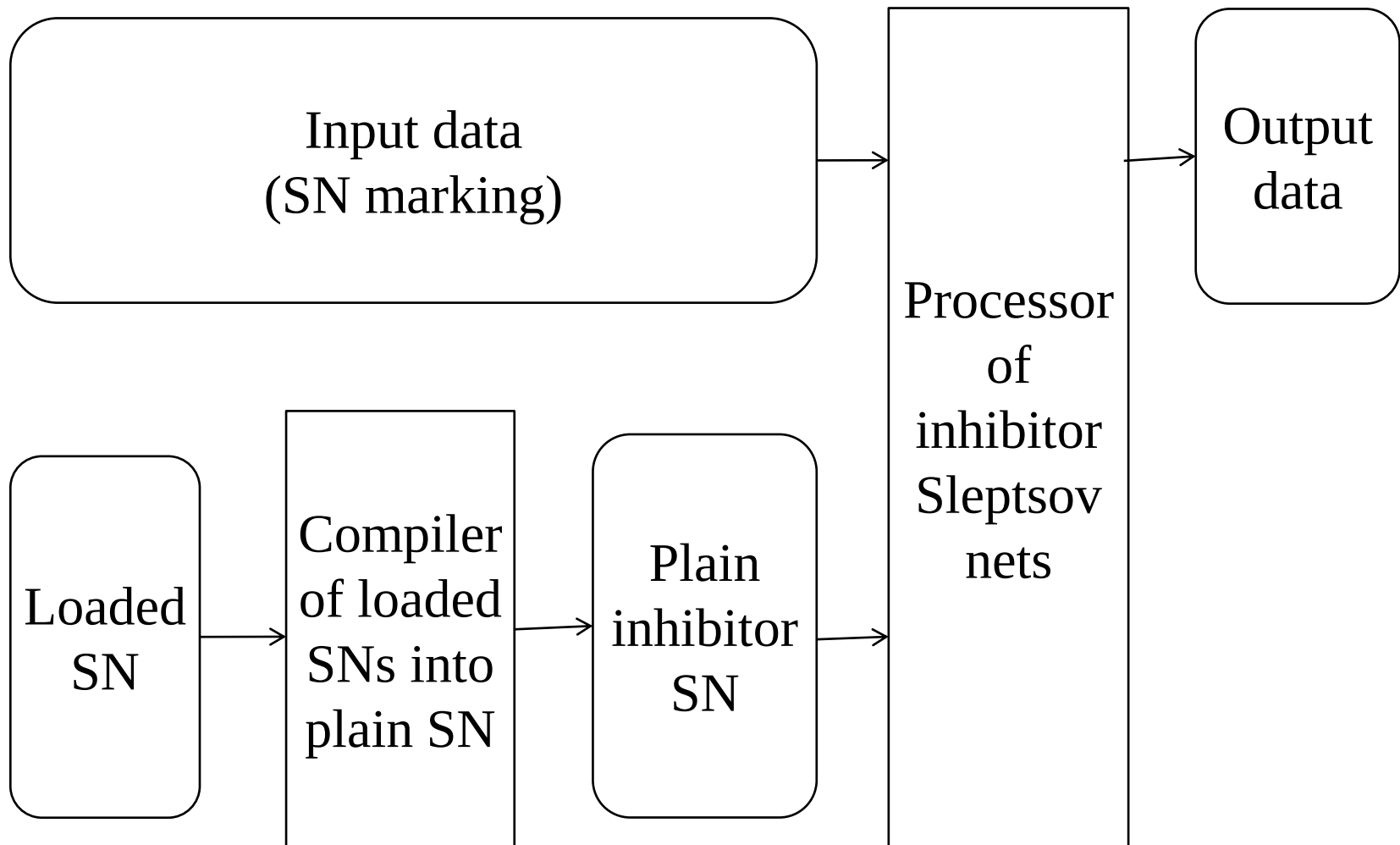


a) inline



b) call-return

Sleptsov Net Paradigm of Computing



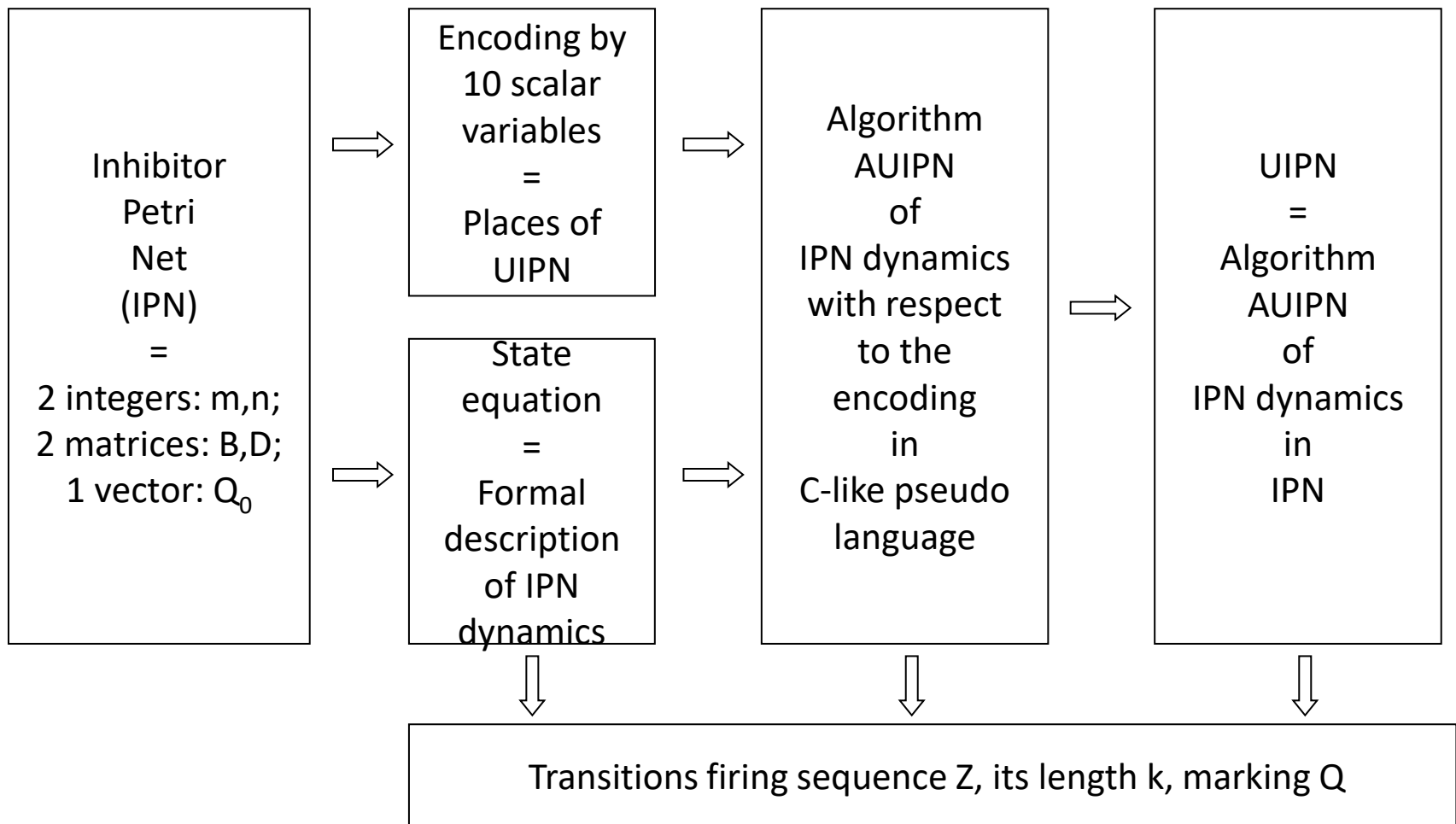
Universal Sleptsov Net Concept



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

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

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

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

Encodings

Encoding of a vector (marking)

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

Encoding of a matrix (incidence)

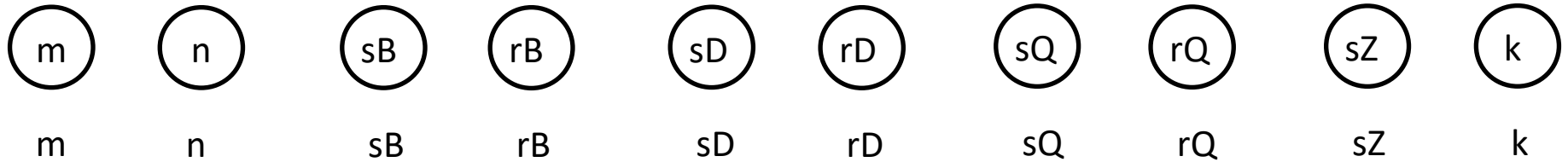
$$s = \varphi(A) = \sum_{i=0}^{n-1} \sum_{j=0}^{m-1} r^{(m \cdot i + j)} \cdot a_{i,j}, \quad r = \max_{i,j} a_{i,j} + 1$$

Recursive encoding

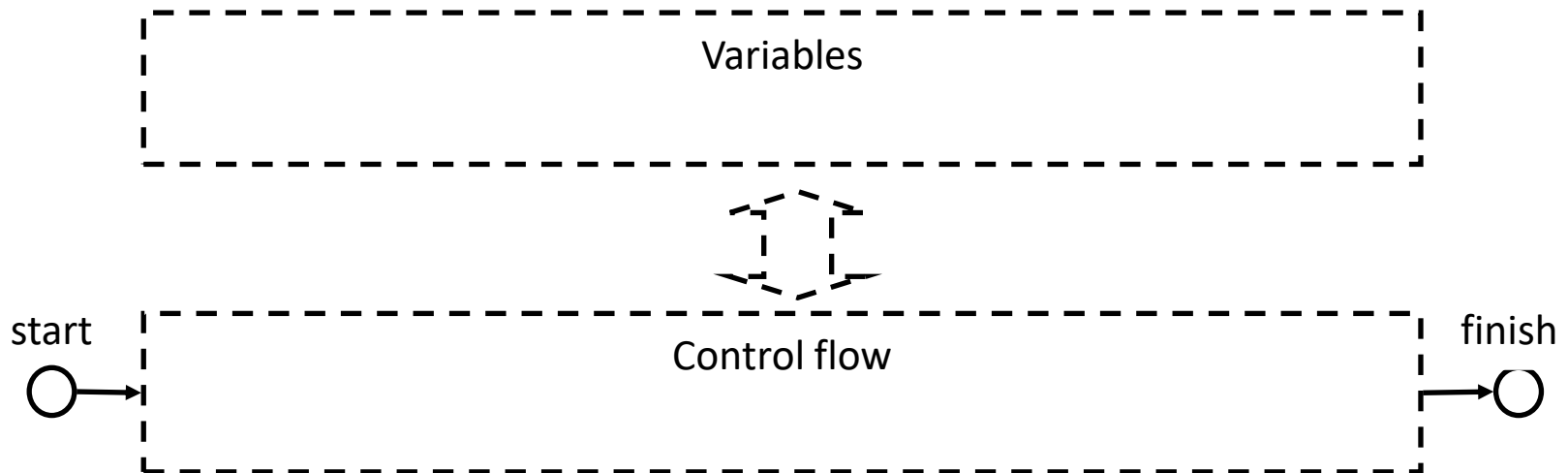
$$s_v = s_{v-1} \cdot r + a_{i,j}, s_0 = a_{n-1,m-1}$$

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

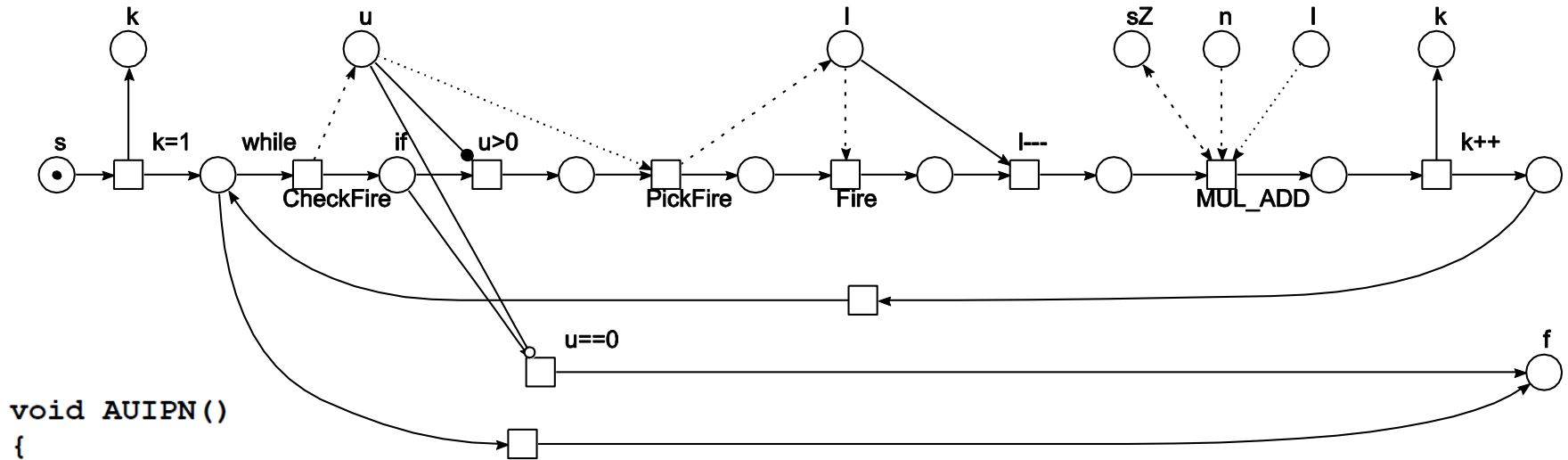
Composition of data and control flow



Code of a given Petri net

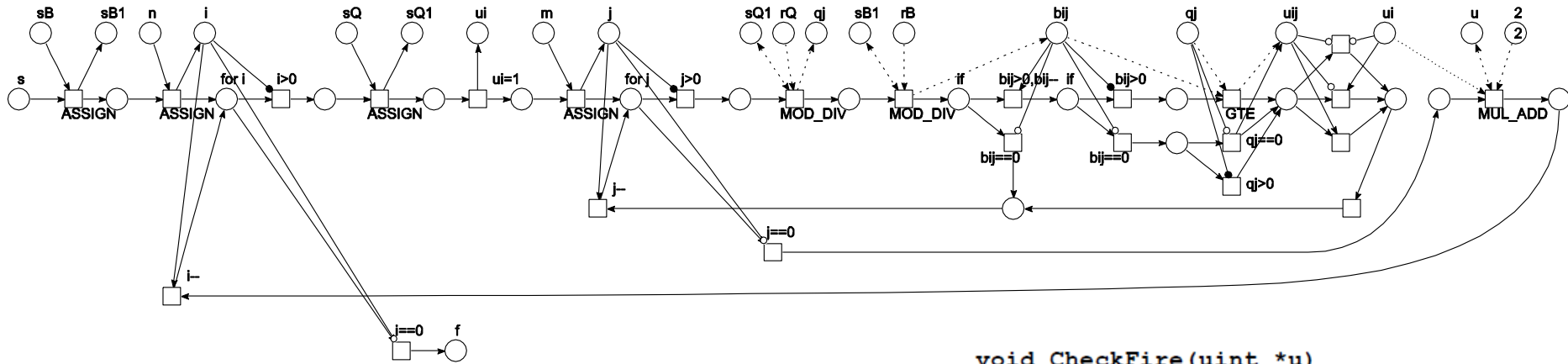


UIPN



```
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();  
}
```

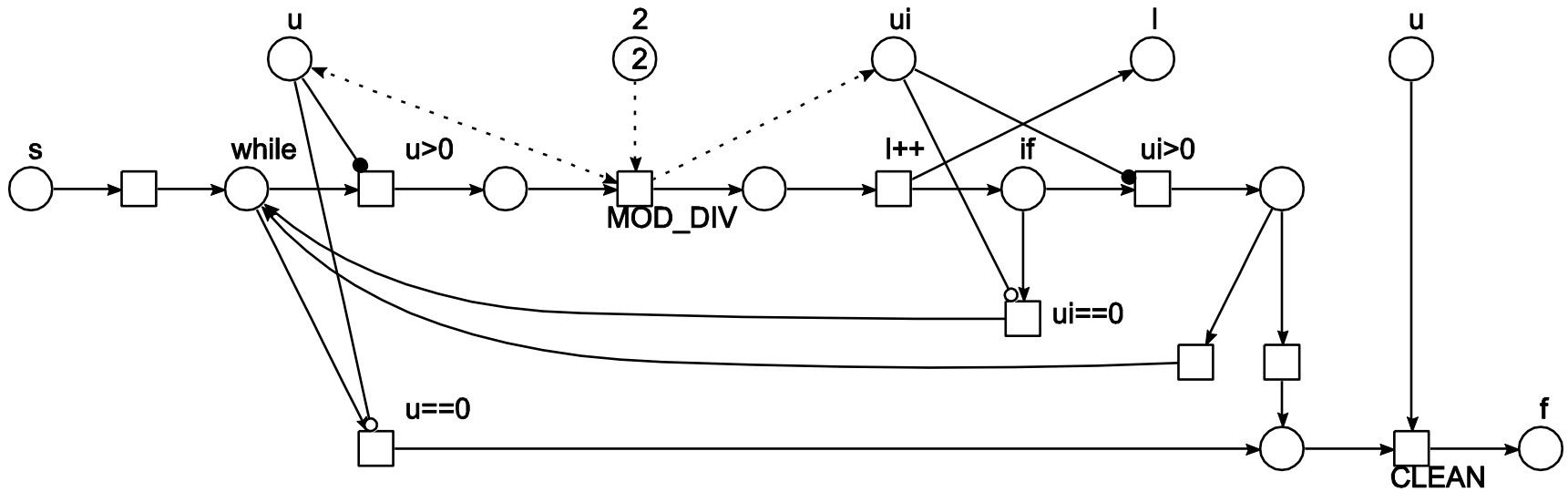
CheckFire



```

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);
    }
}
    
```

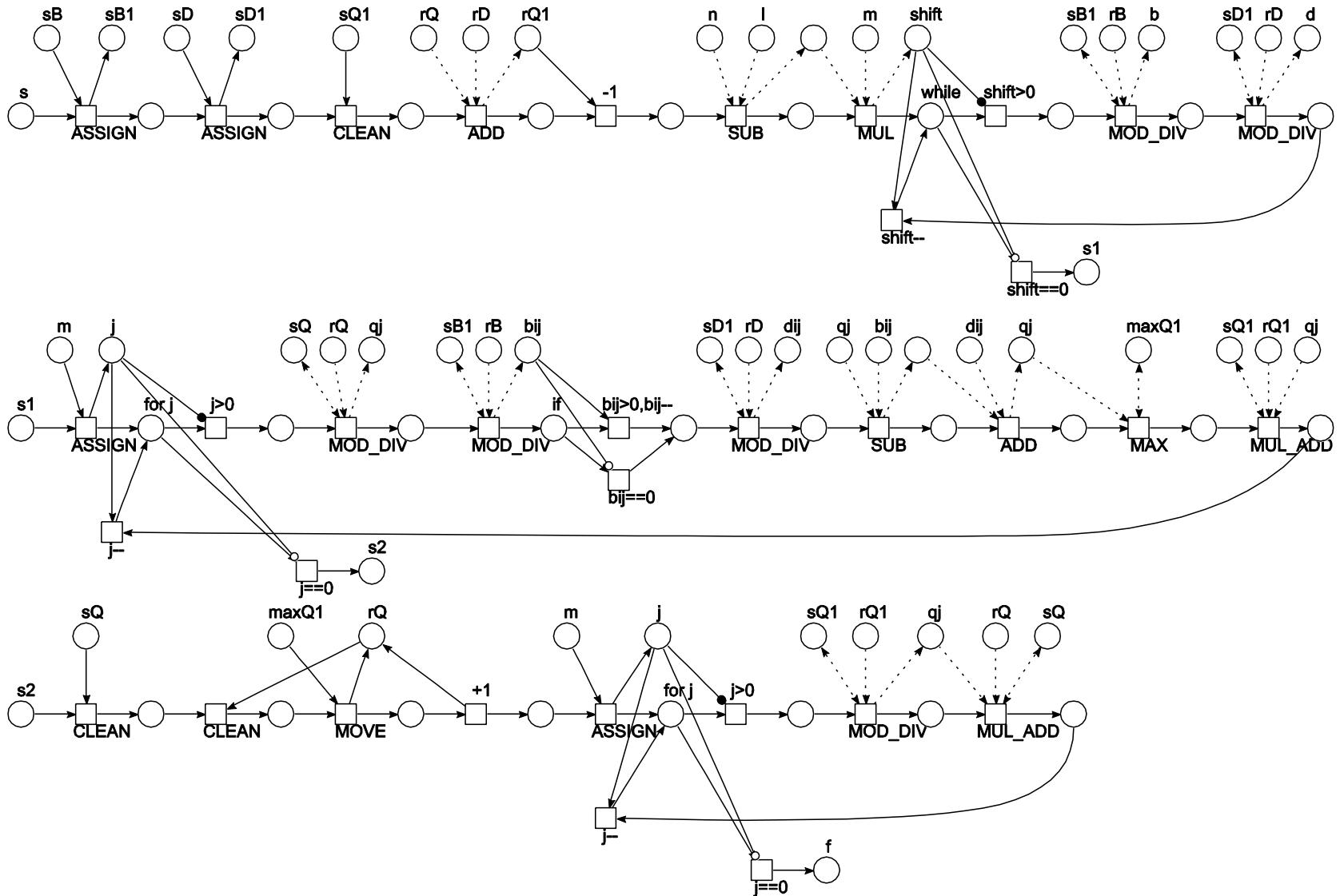
PickFire



```

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



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	646549072061101455668889034663481743952654	3	19352259085292454555975681	2

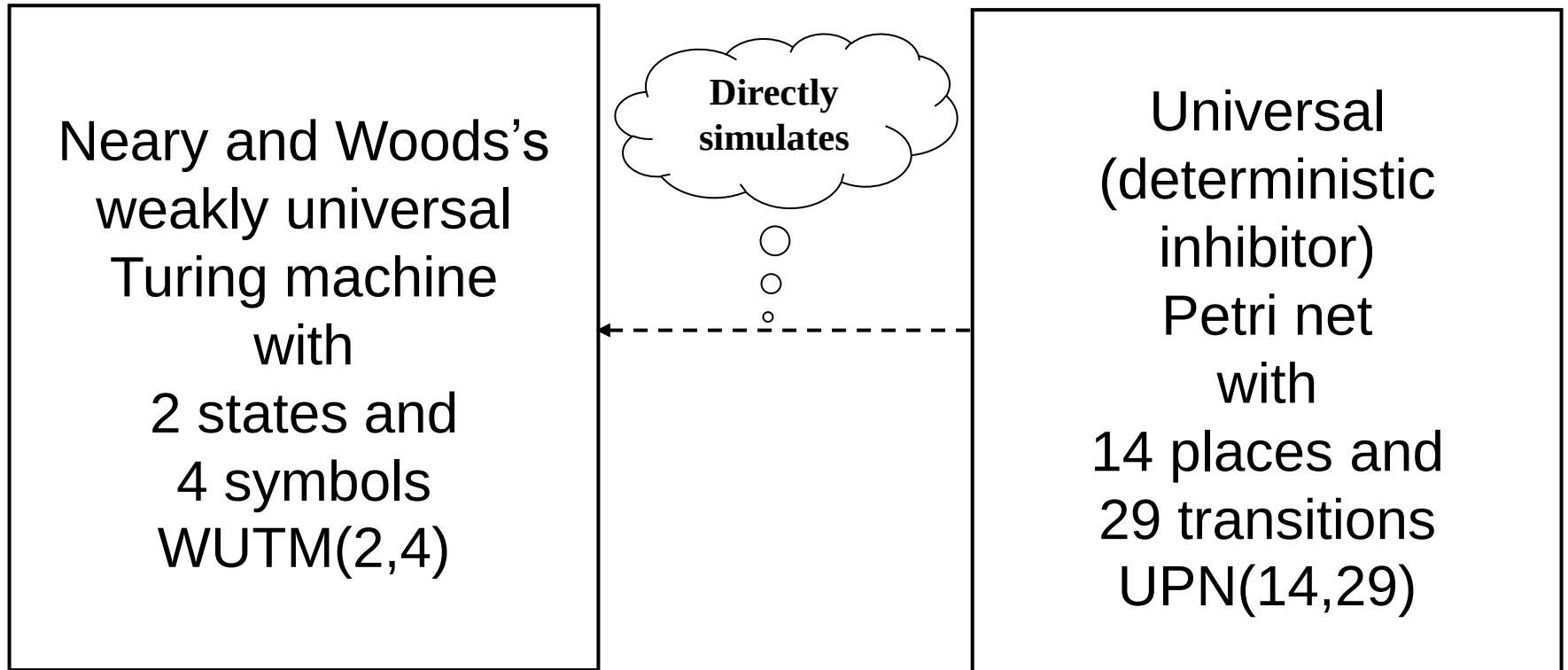
Marking

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

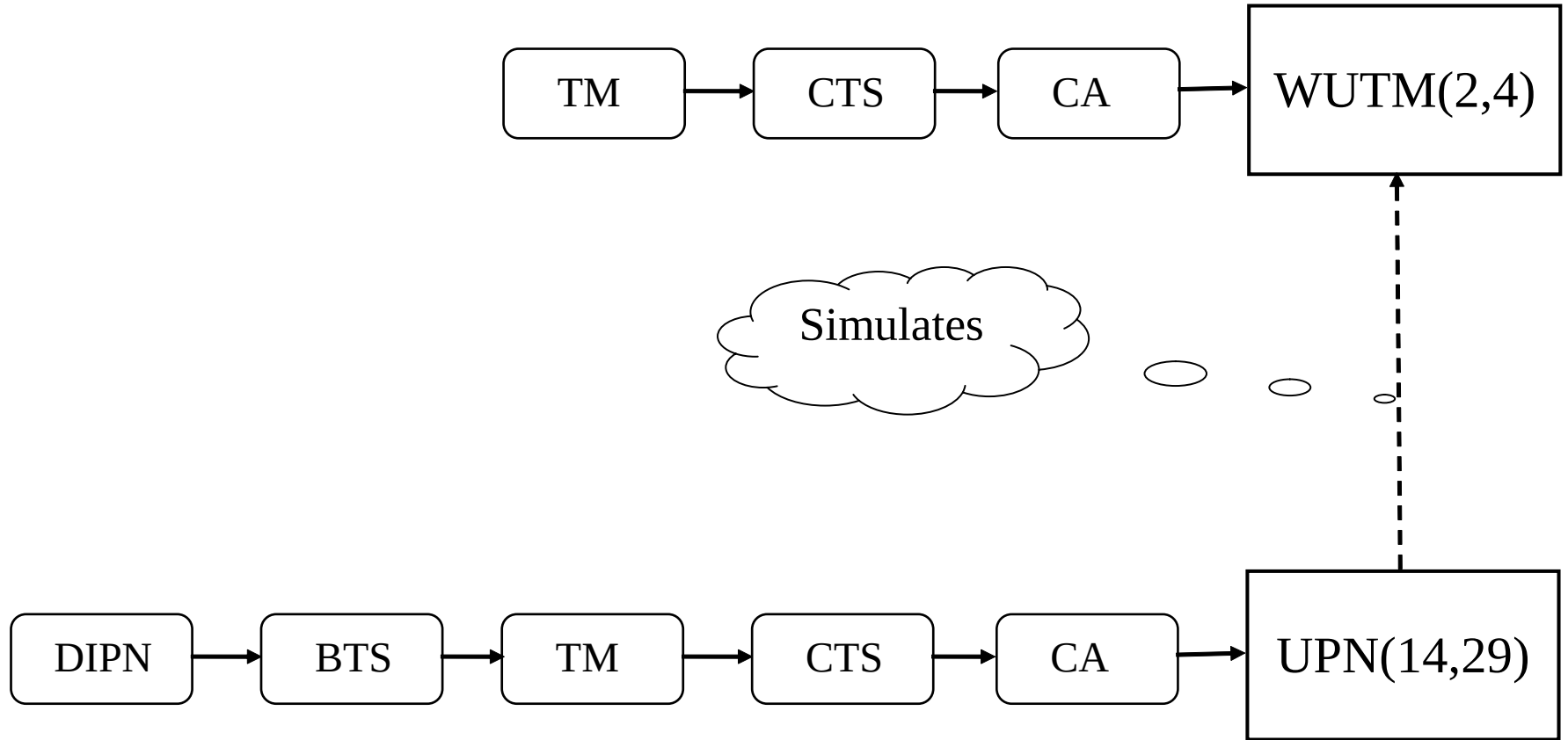
Transitions firing sequence

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	109815712212339723705298	26

II. UPN(14,29)



Chains of translations

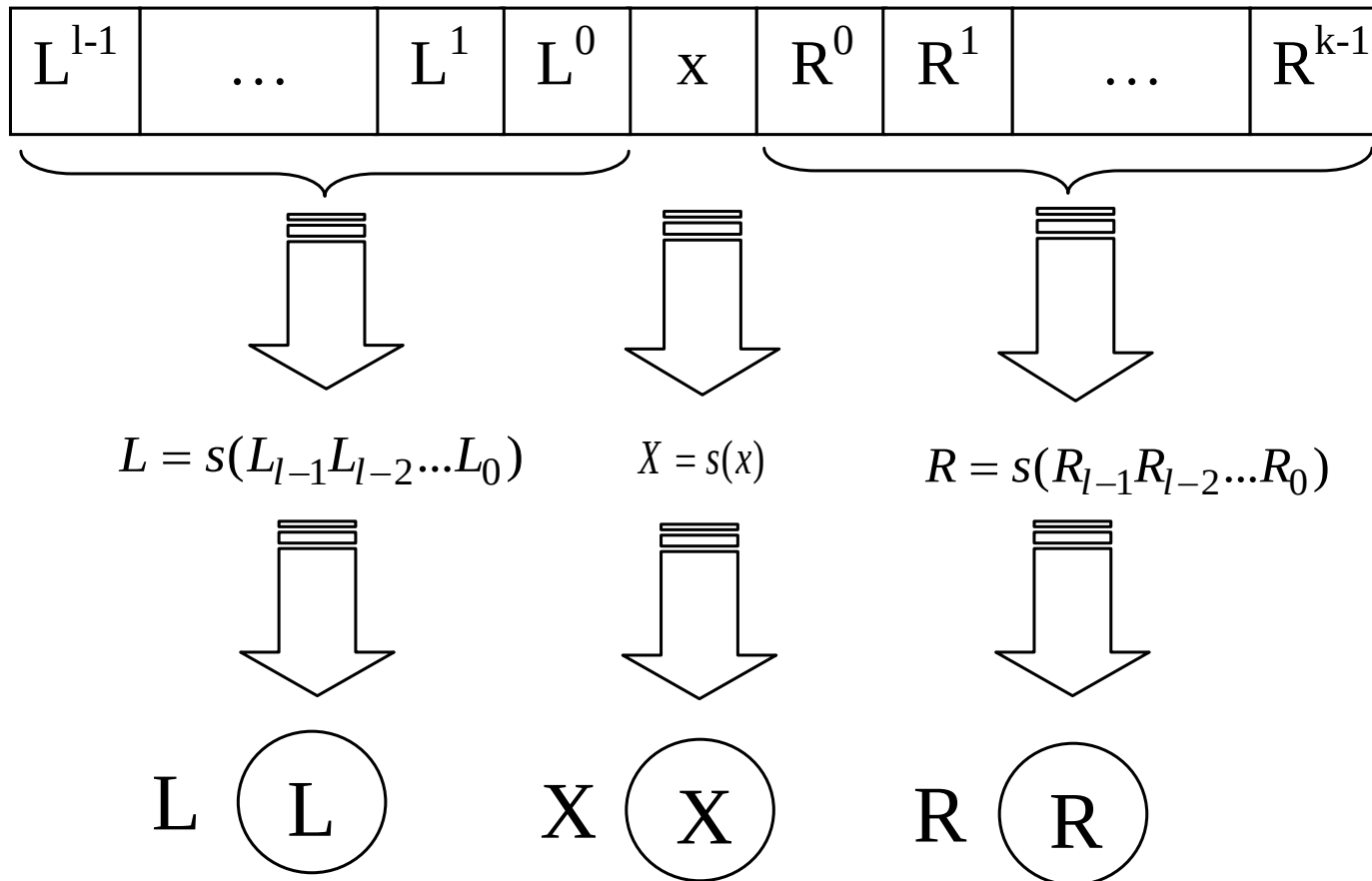


Encoding of states, symbols, and transition function

$\Sigma \backslash \Omega$		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	$4, left, 0$	$1, right, 1$
λ	4	$4, left, 0$	$2, right, 1$

Tape encoding

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



Encoding of blank words

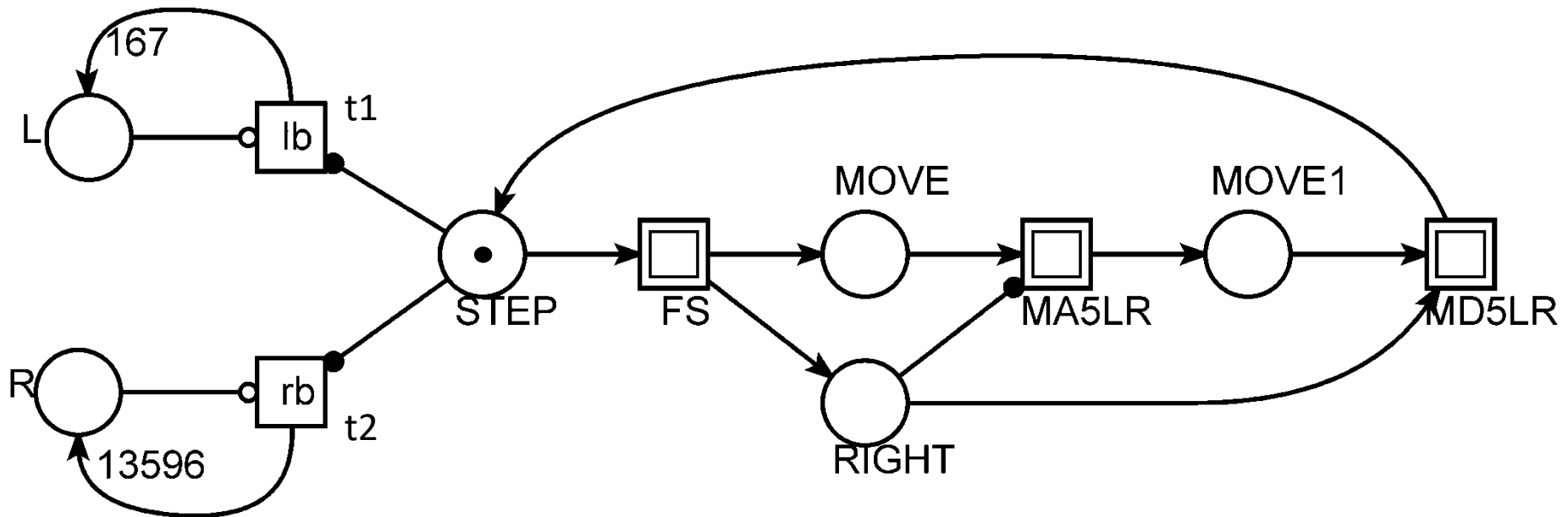
$$w_l = 00\emptyset 1$$

$$w_r = 0\cancel{1}\emptyset\emptyset 0\cancel{1}$$

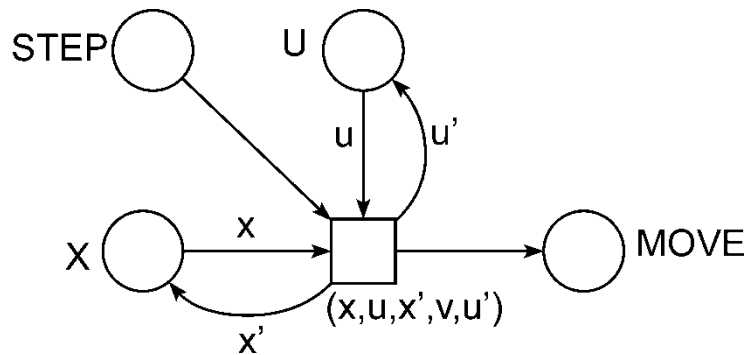
$$sw_l = s(w_l) = ((s(0) \cdot rX + s(0)) \cdot rX + s(\emptyset)) \cdot rX + s(1) = ((1 \cdot 5 + 1) \cdot 5 + 3) \cdot 5 + 2 = 167$$

$$\begin{aligned} sw_r = s(w_r) &= (((((s(\cancel{1}) \cdot rX + s(0)) \cdot rX + s(\emptyset)) \cdot rX + s(\emptyset)) \cdot rX + s(\cancel{1})) \cdot rX + s(0)) \\ &= (((((4 \cdot 5 + 1) \cdot 5 + 3) \cdot 5 + 3) \cdot 5 + 4) \cdot 5 + 1 = 13596. \end{aligned}$$

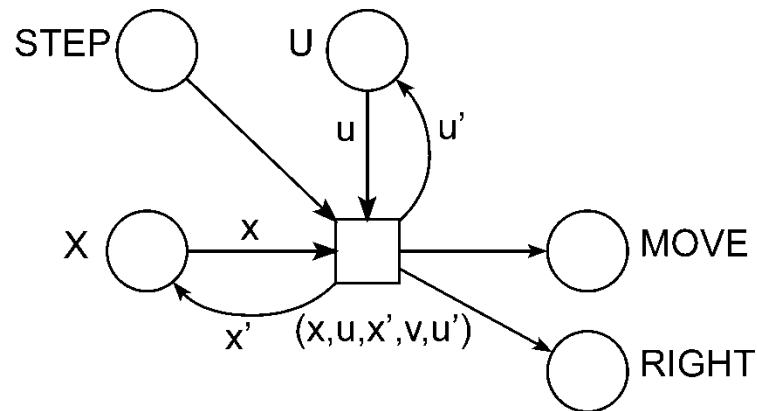
General arrangement of UPN(14,29)



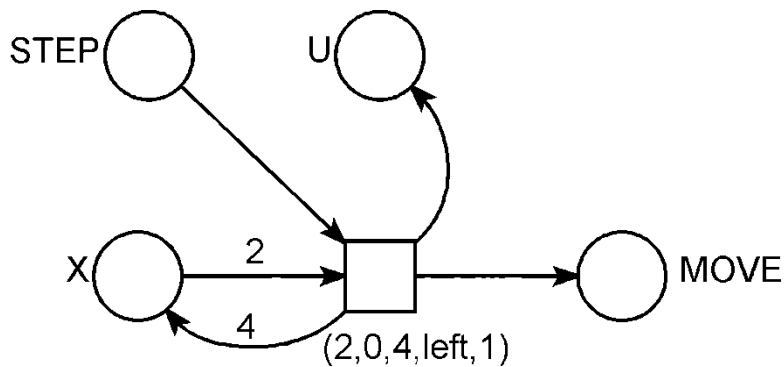
Simulating TM instruction



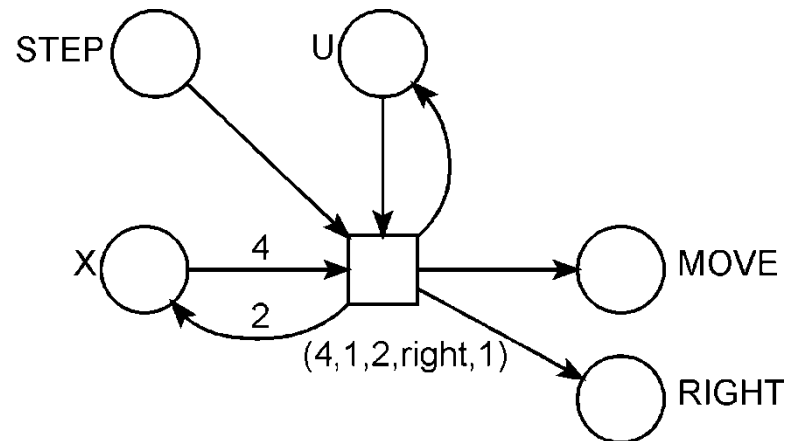
an instruction with the left move



an instruction with the right move

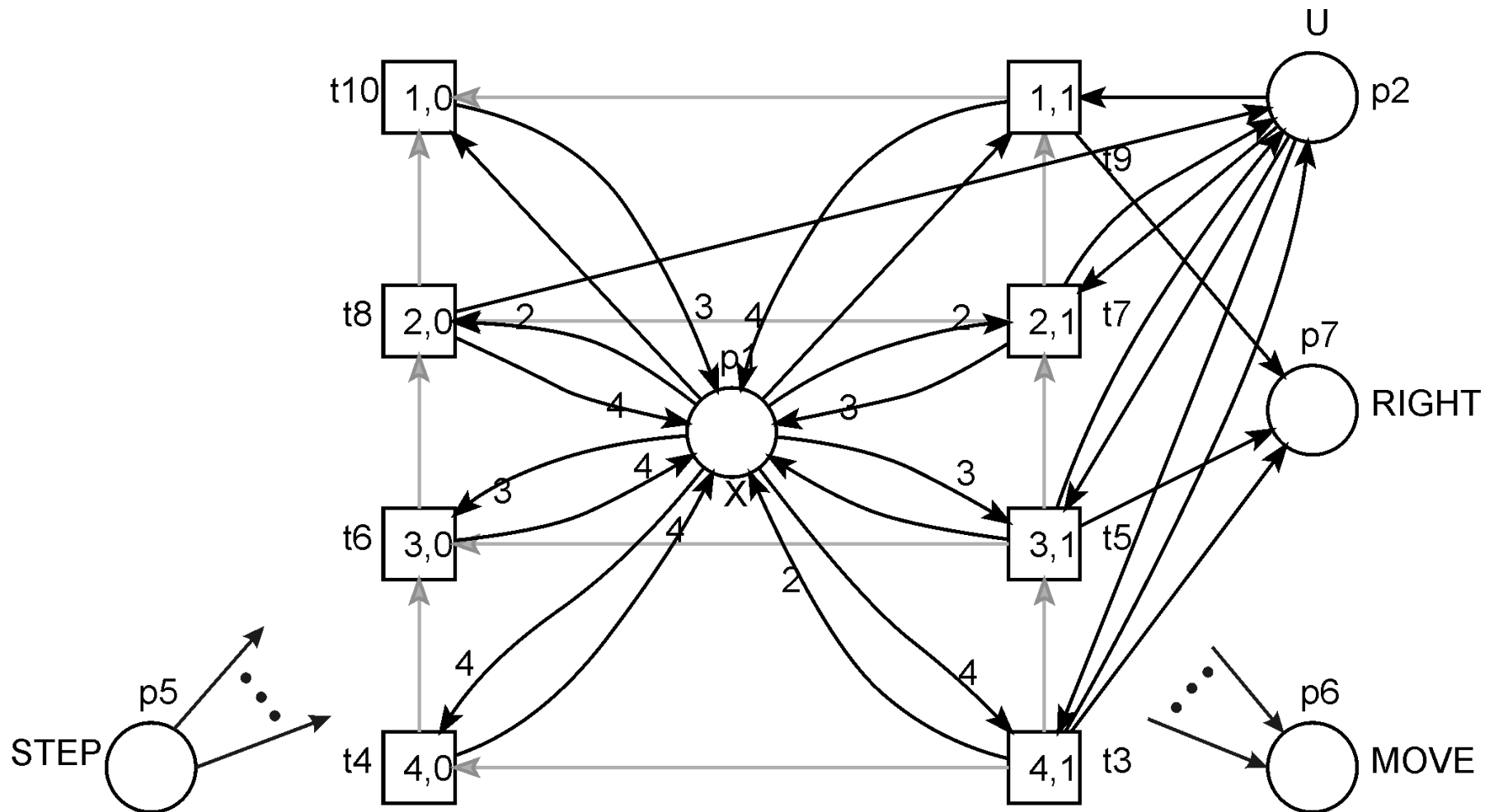


$(2, 0, 4, \text{left}, 1)$

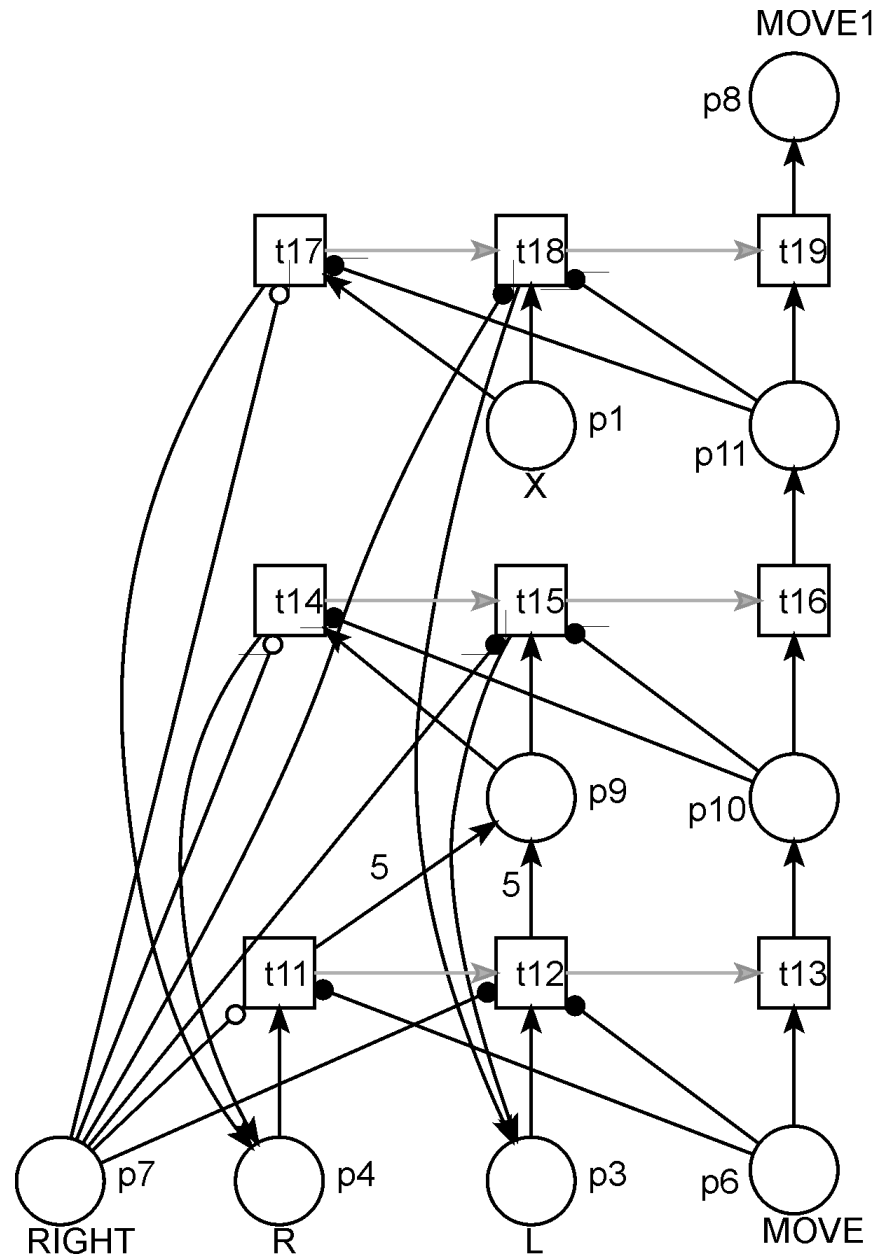


$(4, 1, 2, \text{right}, 1)$

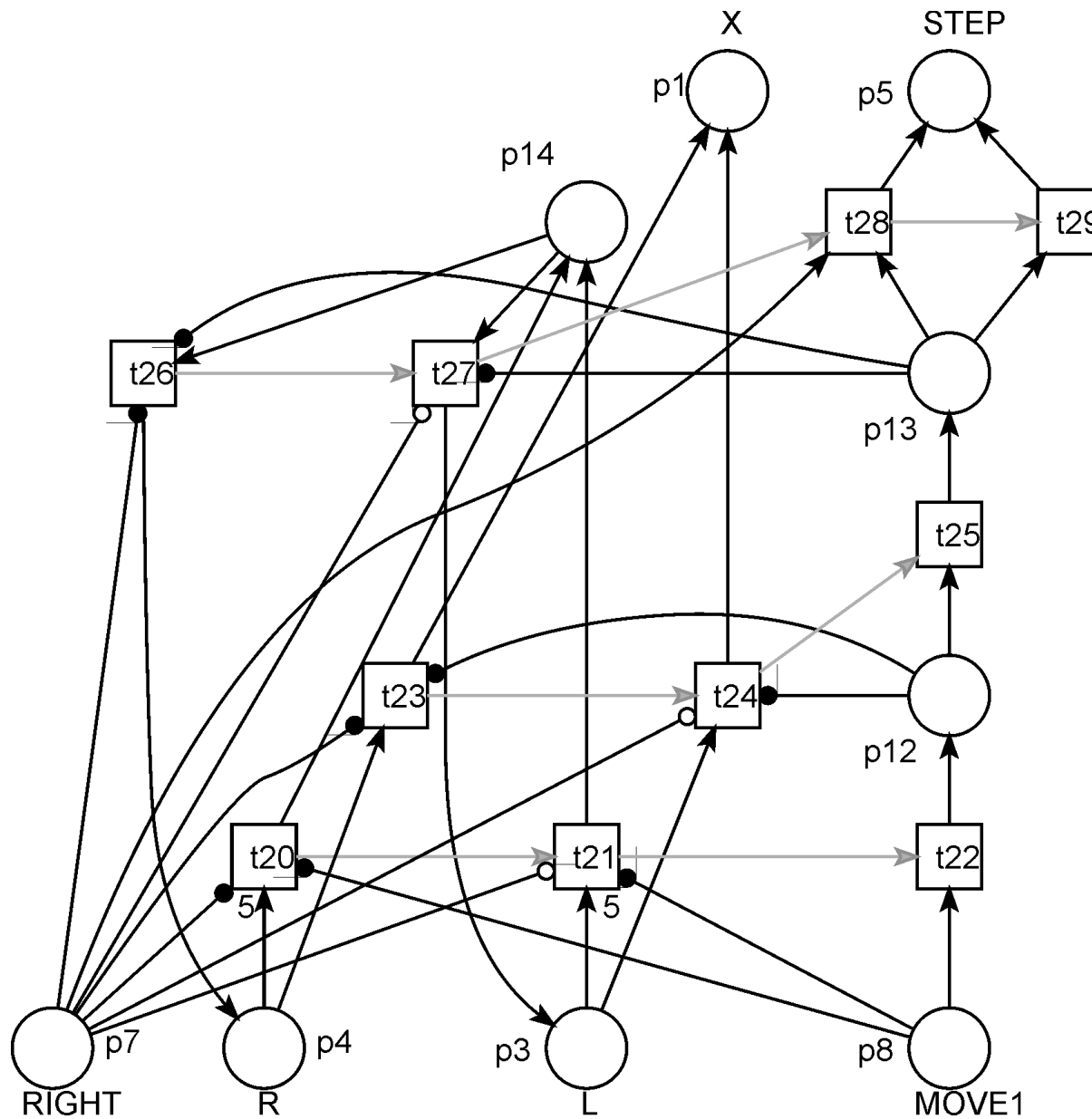
Subnet FS simulating WUTM(2,4) transition function



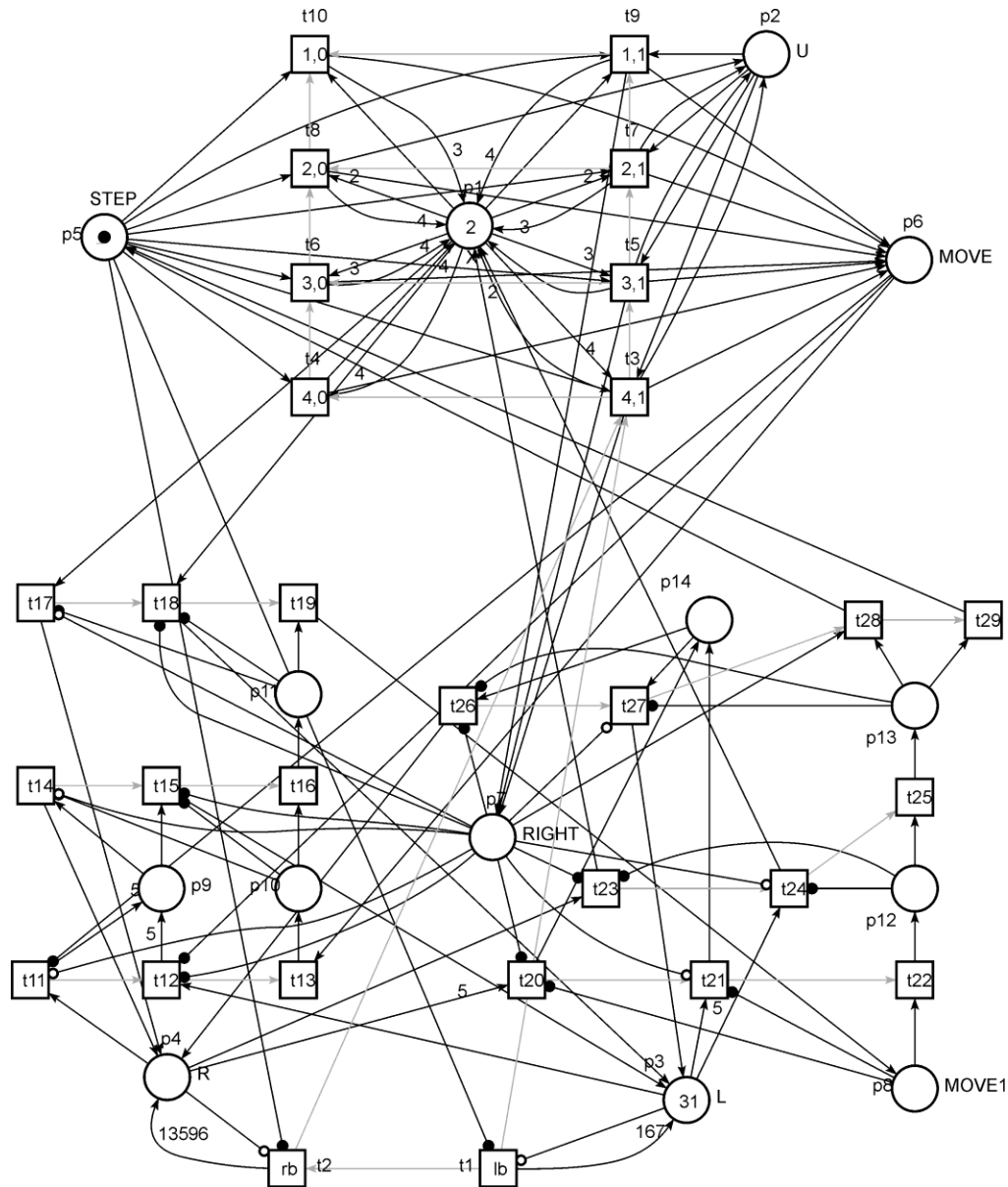
Subnet MA5LR



Subnet MD5LR



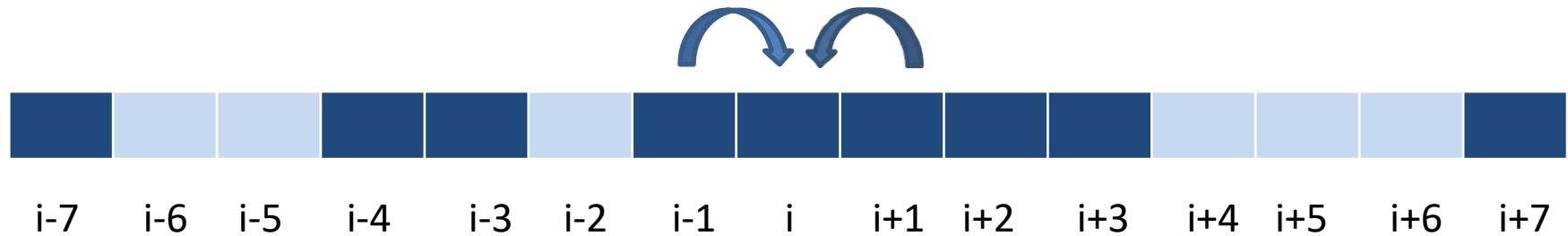
UPN(14,29) in graphical form



Trace of UPN(14,29) running

Step	Configuration	Code of	
		state U	tape (L, X, R)
0	$u_1, \dots 00\emptyset 1 \ 00\emptyset 1 \ \mathbf{000\underline{1}} \ 0\cancel{1}\emptyset\emptyset 0\cancel{1} \ 0\cancel{1}\emptyset\emptyset 0\cancel{1} \dots$	0	(31,2,0)
	$u_1, \dots 00\emptyset 1 \ 00\emptyset 1 \ \mathbf{000\underline{1}} \ \mathbf{0\underline{1}\emptyset\emptyset 0\cancel{1}} \ 0\cancel{1}\emptyset\emptyset 0\cancel{1} \dots$	0	(31,2,13596)
1	$u_2, \dots 00\emptyset 1 \ 00\emptyset 1 \ \mathbf{000\underline{1}} \ \mathbf{0\underline{1}\emptyset\emptyset 0\cancel{1}} \ 0\cancel{1}\emptyset\emptyset 0\cancel{1} \dots$	1	(6,1,67984)
2	$u_1, \dots 00\emptyset 1 \ 00\emptyset 1 \ \mathbf{00\underline{1}\cancel{1}} \ \mathbf{0\underline{1}\emptyset\emptyset 0\cancel{1}} \ 0\cancel{1}\emptyset\emptyset 0\cancel{1} \dots$	0	(34,4,13596)
3	$u_1, \dots 00\emptyset 1 \ 00\emptyset 1 \ \mathbf{00\underline{1}\cancel{1}} \ \mathbf{0\underline{1}\emptyset\emptyset 0\cancel{1}} \ 0\cancel{1}\emptyset\emptyset 0\cancel{1} \dots$	0	(6,4,67984)
4	$u_1, \dots 00\emptyset 1 \ 00\emptyset 1 \ \mathbf{00\underline{1}\cancel{1}} \ \mathbf{0\underline{1}\emptyset\emptyset 0\cancel{1}} \ 0\cancel{1}\emptyset\emptyset 0\cancel{1} \dots$	0	(1,1,339924)
5	$u_1, \dots 00\emptyset 1 \ 00\emptyset 1 \ \mathbf{00\underline{1}\cancel{1}} \ \mathbf{0\underline{1}\emptyset\emptyset 0\cancel{1}} \ 0\cancel{1}\emptyset\emptyset 0\cancel{1} \dots$	0	(0,1,1699623)
	$u_1, \dots 00\emptyset 1 \ \mathbf{000\underline{1}} \ \mathbf{00\underline{1}\cancel{1}} \ \mathbf{0\underline{1}\emptyset\emptyset 0\cancel{1}} \ 0\cancel{1}\emptyset\emptyset 0\cancel{1} \dots$	0	(167,1,1699623)
6	$u_1, \dots 00\emptyset 1 \ \mathbf{000\underline{1}} \ \mathbf{\emptyset\emptyset\underline{1}\cancel{1}} \ \mathbf{0\underline{1}\emptyset\emptyset 0\cancel{1}} \ 0\cancel{1}\emptyset\emptyset 0\cancel{1} \dots$	0	(33,2,8498118)
7	$u_2, \dots 00\emptyset 1 \ \mathbf{000\underline{1}} \ \mathbf{\emptyset\emptyset\underline{1}\cancel{1}} \ \mathbf{0\underline{1}\emptyset\emptyset 0\cancel{1}} \ 0\cancel{1}\emptyset\emptyset 0\cancel{1} \dots$	1	(6,3,42490594)
8	$u_2, \dots 00\emptyset 1 \ \mathbf{000\underline{1}} \ \mathbf{\emptyset\emptyset\underline{1}\cancel{1}} \ \mathbf{0\underline{1}\emptyset\emptyset 0\cancel{1}} \ 0\cancel{1}\emptyset\emptyset 0\cancel{1} \dots$	1	(31,4,8498118)
9	$u_2, \dots 00\emptyset 1 \ \mathbf{000\underline{1}} \ \mathbf{\emptyset\emptyset\underline{1}\cancel{1}} \ \mathbf{0\underline{1}\emptyset\emptyset 0\cancel{1}} \ 0\cancel{1}\emptyset\emptyset 0\cancel{1} \dots$	1	(157,3,1699623)
10	$u_2, \dots 00\emptyset 1 \ \mathbf{000\underline{1}} \ \mathbf{00\underline{1}\cancel{1}} \ \mathbf{0\underline{1}\emptyset\emptyset 0\cancel{1}} \ 0\cancel{1}\emptyset\emptyset 0\cancel{1} \dots$	1	(786,3,339924)
11	$u_2, \dots 00\emptyset 1 \ \mathbf{000\underline{1}} \ \mathbf{00\underline{1}\cancel{1}} \ \mathbf{0\underline{1}\emptyset\emptyset 0\cancel{1}} \ 0\cancel{1}\emptyset\emptyset 0\cancel{1} \dots$	1	(3931,4,67984)
12	$u_2, \dots 00\emptyset 1 \ \mathbf{000\underline{1}} \ \mathbf{00\underline{1}\cancel{1}} \ \mathbf{0\underline{1}\emptyset\emptyset 0\cancel{1}} \ 0\cancel{1}\emptyset\emptyset 0\cancel{1} \dots$	1	(19657,4,13596)
13	$u_2, \dots 00\emptyset 1 \ \mathbf{000\underline{1}} \ \mathbf{00\underline{1}\cancel{1}} \ \mathbf{0\underline{1}\emptyset\emptyset 0\cancel{1}} \ 0\cancel{1}\emptyset\emptyset 0\cancel{1} \dots$	1	(98287,1,2719)
14	$u_1, \dots 00\emptyset 1 \ \mathbf{000\underline{1}} \ \mathbf{00\underline{1}\cancel{1}} \ \mathbf{\underline{1}\underline{1}\emptyset\emptyset 0\cancel{1}} \ 0\cancel{1}\emptyset\emptyset 0\cancel{1} \dots$	0	(491439,4,543)
...	...	...	...

III. Simulating Linear Cellular Automaton 110



Rules 110:

$$\begin{array}{llll}
 R(0, 0, 0) = 0 & R(0, 1, 0) = 1 & R(1, 0, 0) = 0 & R(1, 1, 0) = 1 \\
 R(0, 0, 1) = 1 & R(0, 1, 1) = 1 & R(1, 0, 1) = 1 & R(1, 1, 1) = 0
 \end{array} \quad (1)$$

Minimization of CA110 function

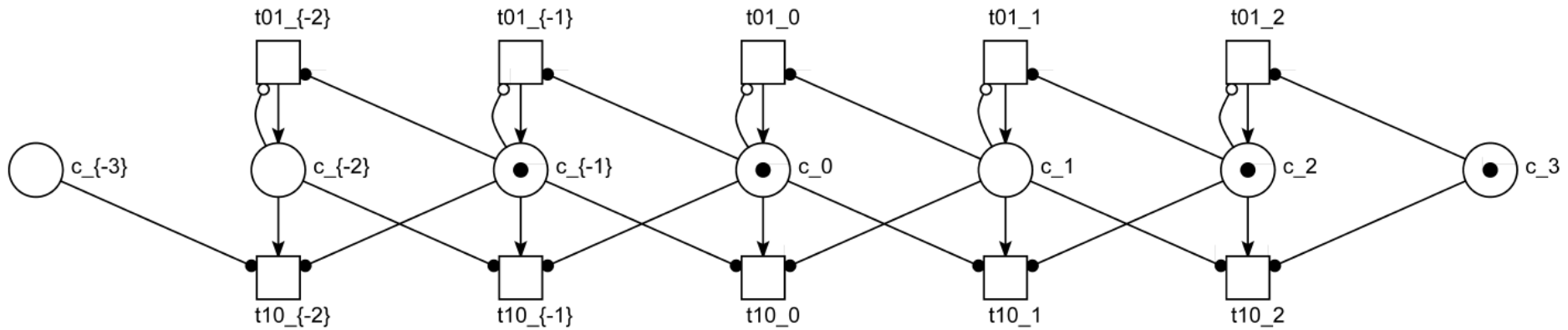
		$c_i c_{i+1}$			
		00	01	11	10
c_{i-1}	0	$txx1$	$t01$	$txx2$	$txx1$
	1	$txx1$	$t01$	$t10$	$txx1$

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

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

$$\begin{aligned} txx &= \bar{c}_{i-1}\bar{c}_i\bar{c}_{i+1} \vee \bar{c}_{i-1}c_i\bar{c}_{i+1} \vee \bar{c}_{i-1}c_i c_{i+1} \vee c_{i-1}\bar{c}_i\bar{c}_{i+1} \vee c_{i-1}\bar{c}_i c_{i+1} \\ &= \bar{c}_{i+1} \vee \bar{c}_{i-1}c_i c_{i+1}. \end{aligned}$$

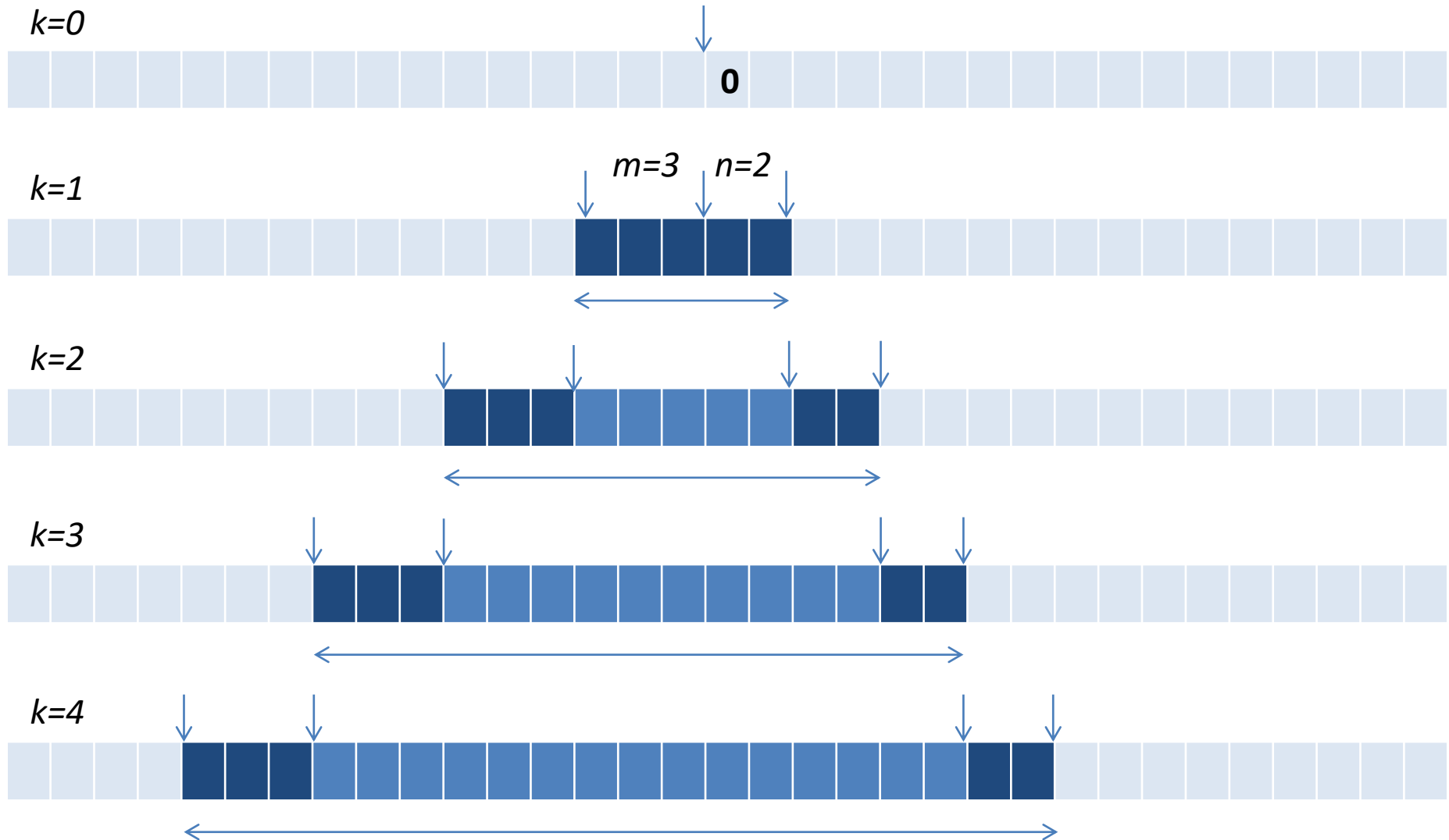
Synchronous PN with Inhibitor and Read Arcs



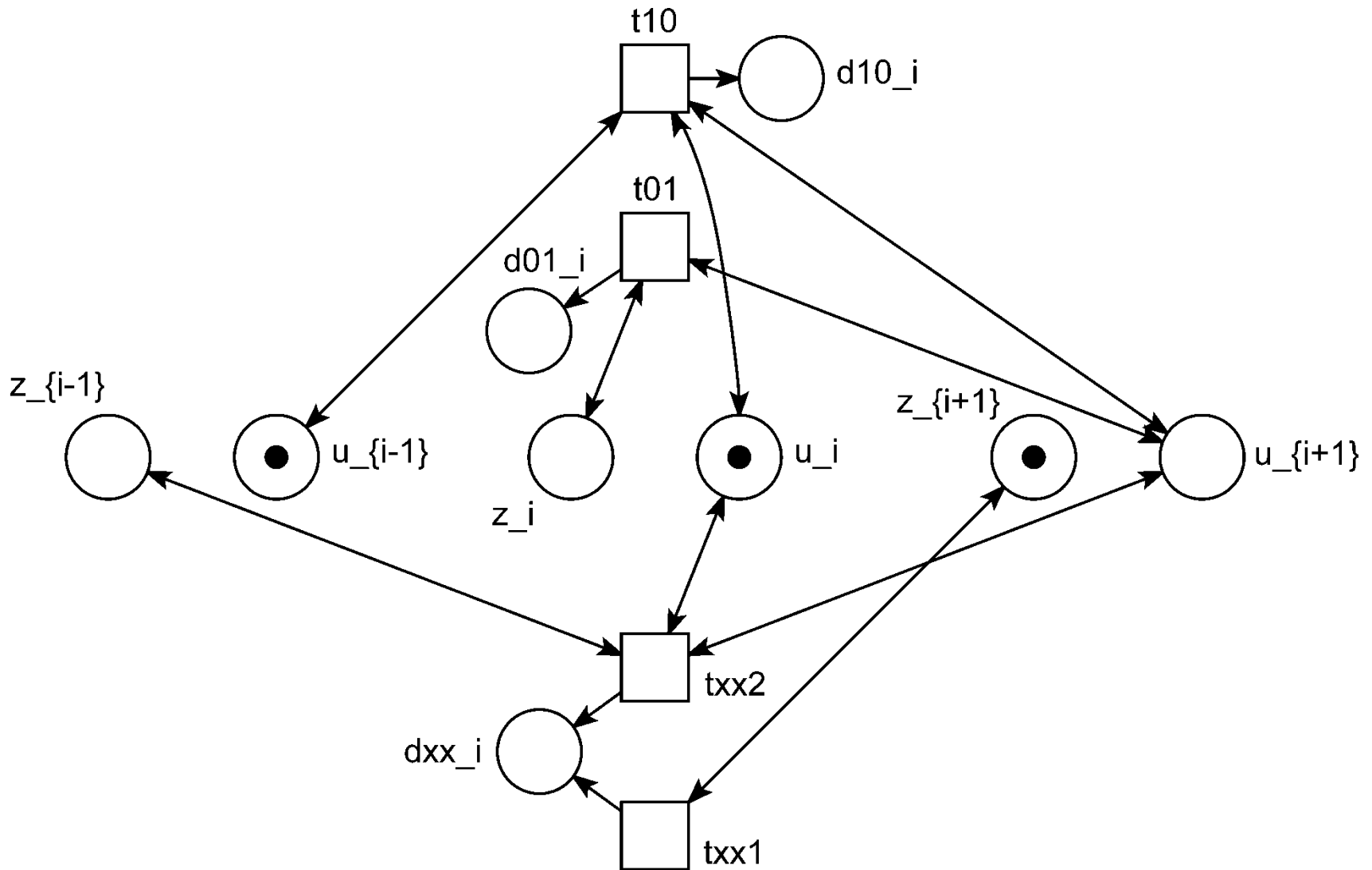
Parametric expression (PE):

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

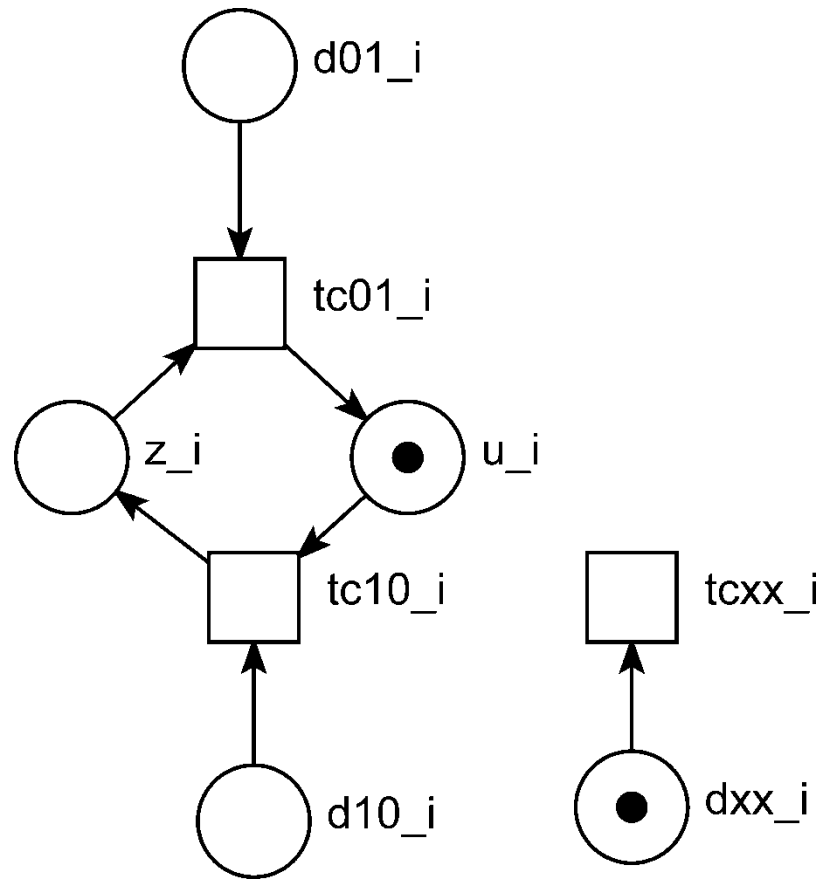
Expanding Traversals of the Cell Array for CA110 Simulation by Asynchronous Nets



DS_i – Calculate the State Difference



CS_i – Change the Cell State



Parametric Specification of DS_i and CS_i

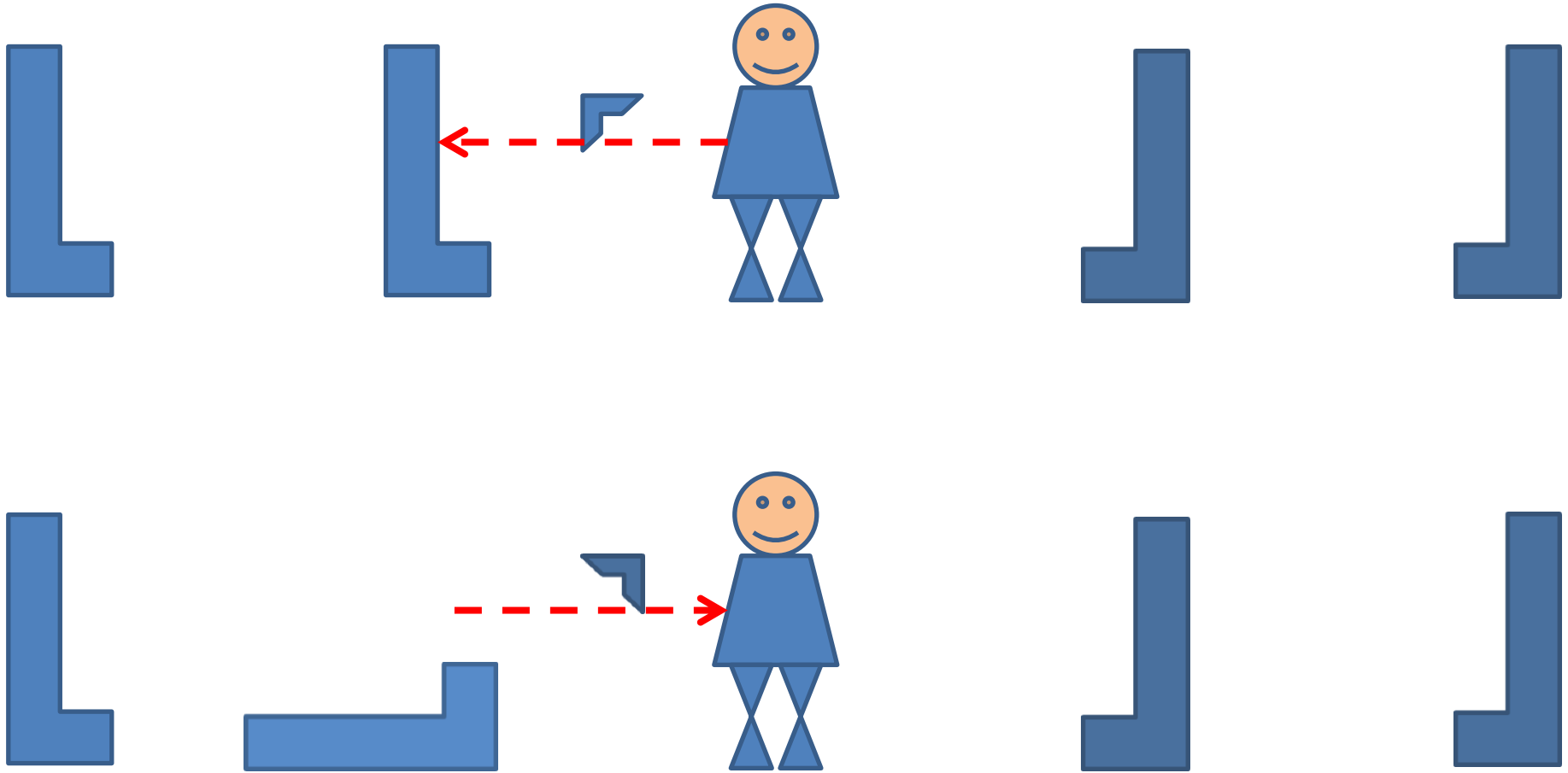
DS_i

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

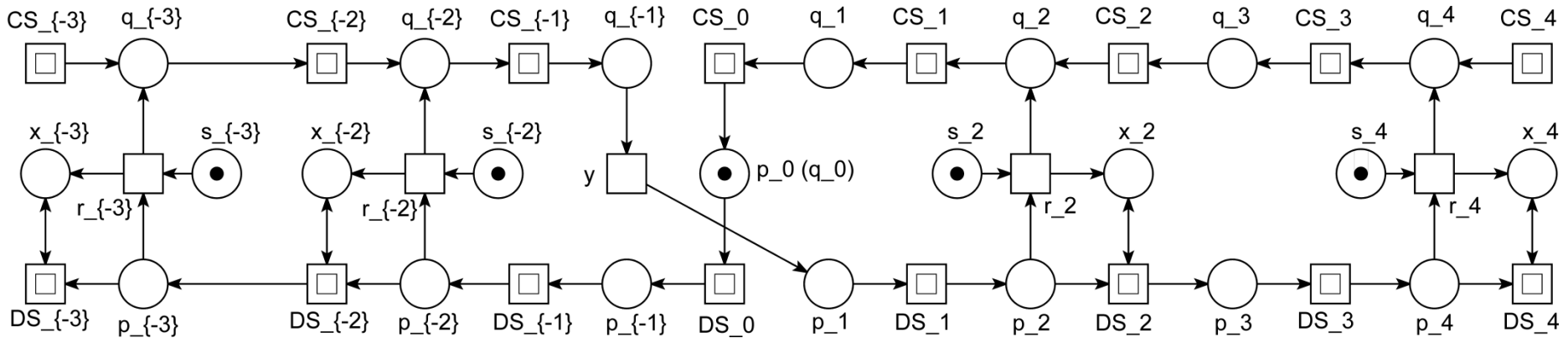
CS_i

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

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



Boomerang and Barriers Net Picture



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

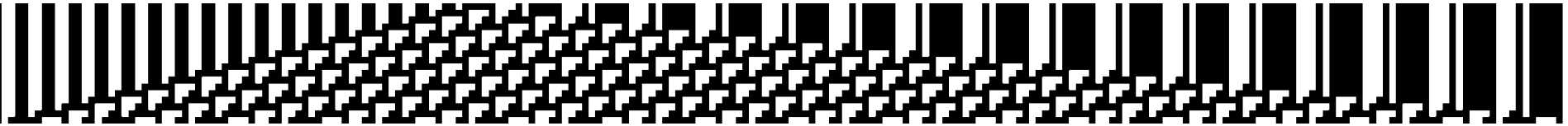
The fragment represents 8 cells

Boomerang and Barriers Net

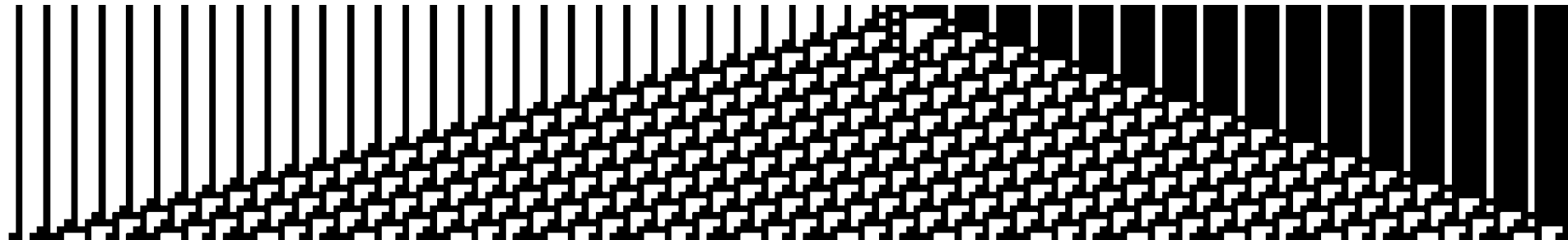
$$\left(\begin{array}{l}
 DS_0 : p_0 \rightarrow p_{-1}, \\
 CS_0 : q_1 \rightarrow p_0, \\
 y : q_{-1} \rightarrow p_1, \\
 p_0 = 1, \\
 \left(\begin{array}{l} DS_i : p_i \rightarrow p_{i+d(i)}, \\ CS_i : q_{i+d(i)} \rightarrow q_i, \end{array} \right) : \\
 i = -1 \vee (i < -1 \wedge |i+1| \bmod m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

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

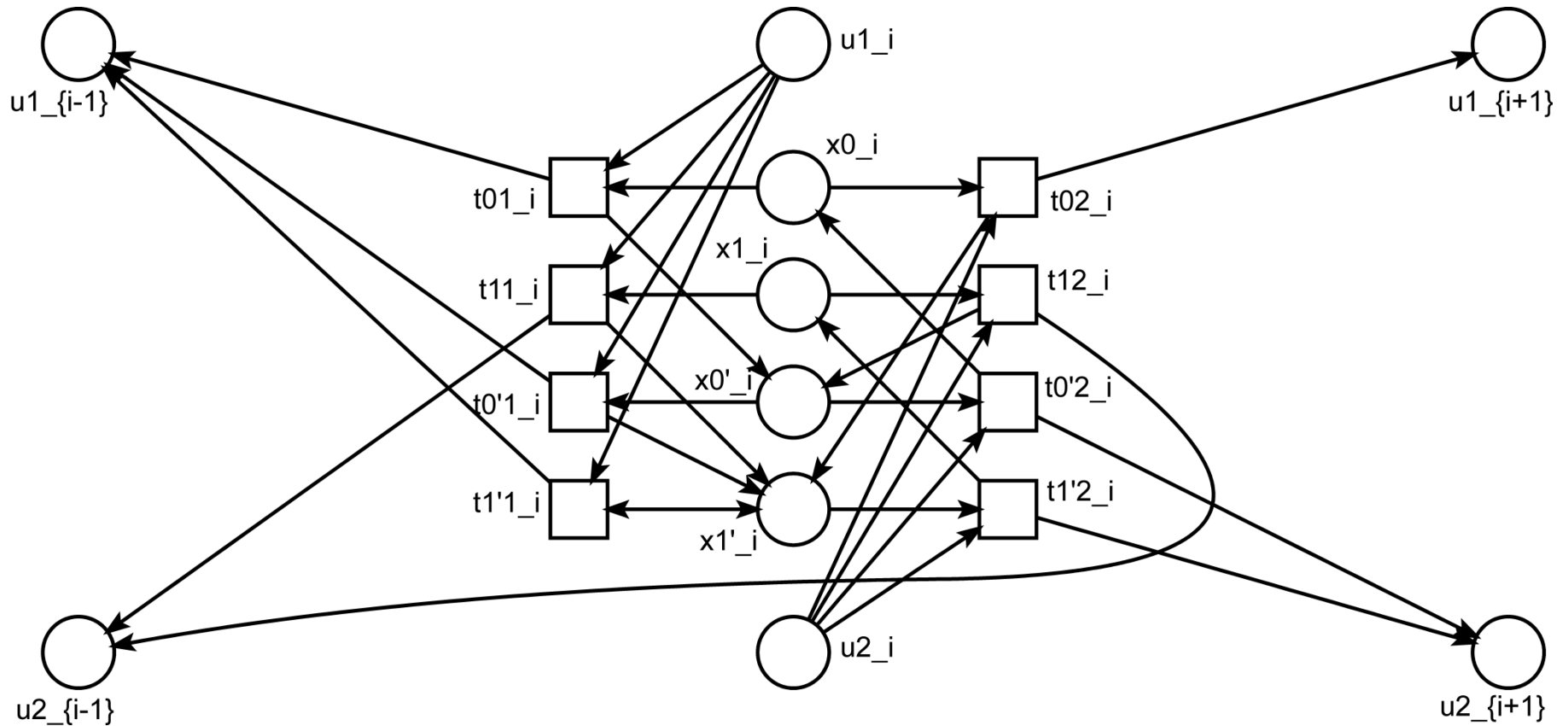
Neary & Woods' weakly universal TM with 2 states and 4 symbols
WUTM(2,4)

	u_1	u_2
0	$\emptyset \ L \ u_1$	$\cancel{1} \ R \ u_1$
1	$\cancel{1} \ L \ u_2$	$\emptyset \ L \ u_2$
$\emptyset$	$\cancel{1} \ L \ u_1$	$0 \ R \ u_2$
$\cancel{1}$	$\cancel{1} \ L \ u_1$	$1 \ R \ u_2$

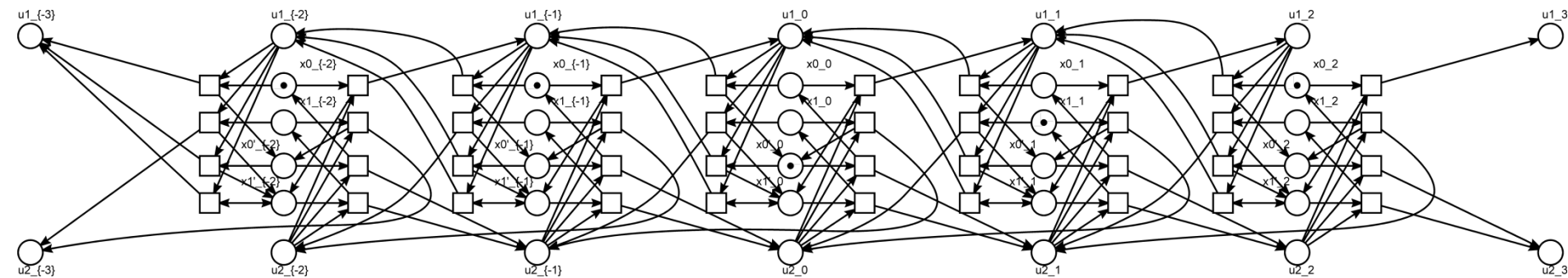
The left blank word: $w_l = 00\emptyset 1$

The right blank word: $w_r = 0\cancel{1} \emptyset \emptyset 0\cancel{1}$

Model of WUTM(2,4) Cell

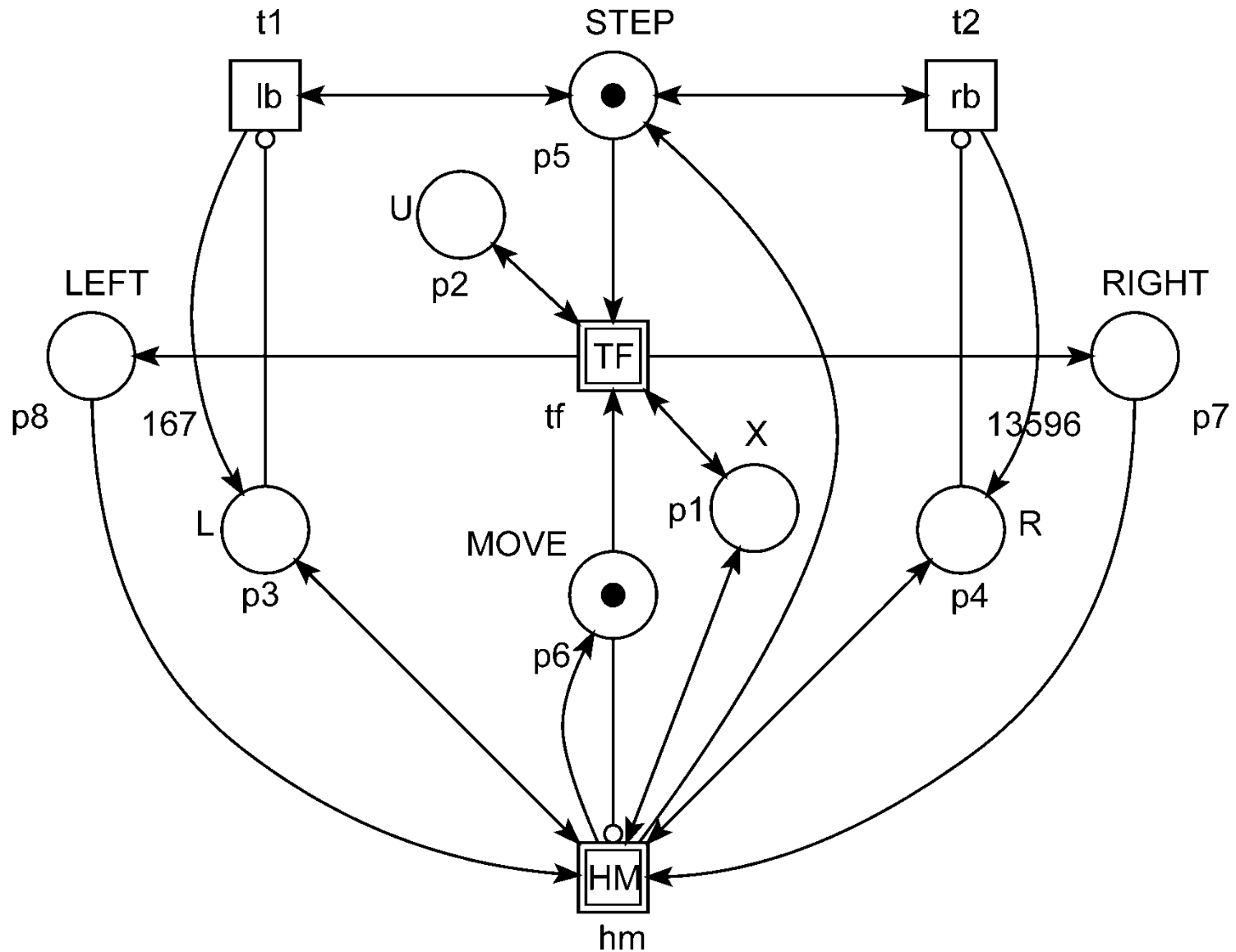


Model of WUTM(2,4) – UPN(6,8,inf)

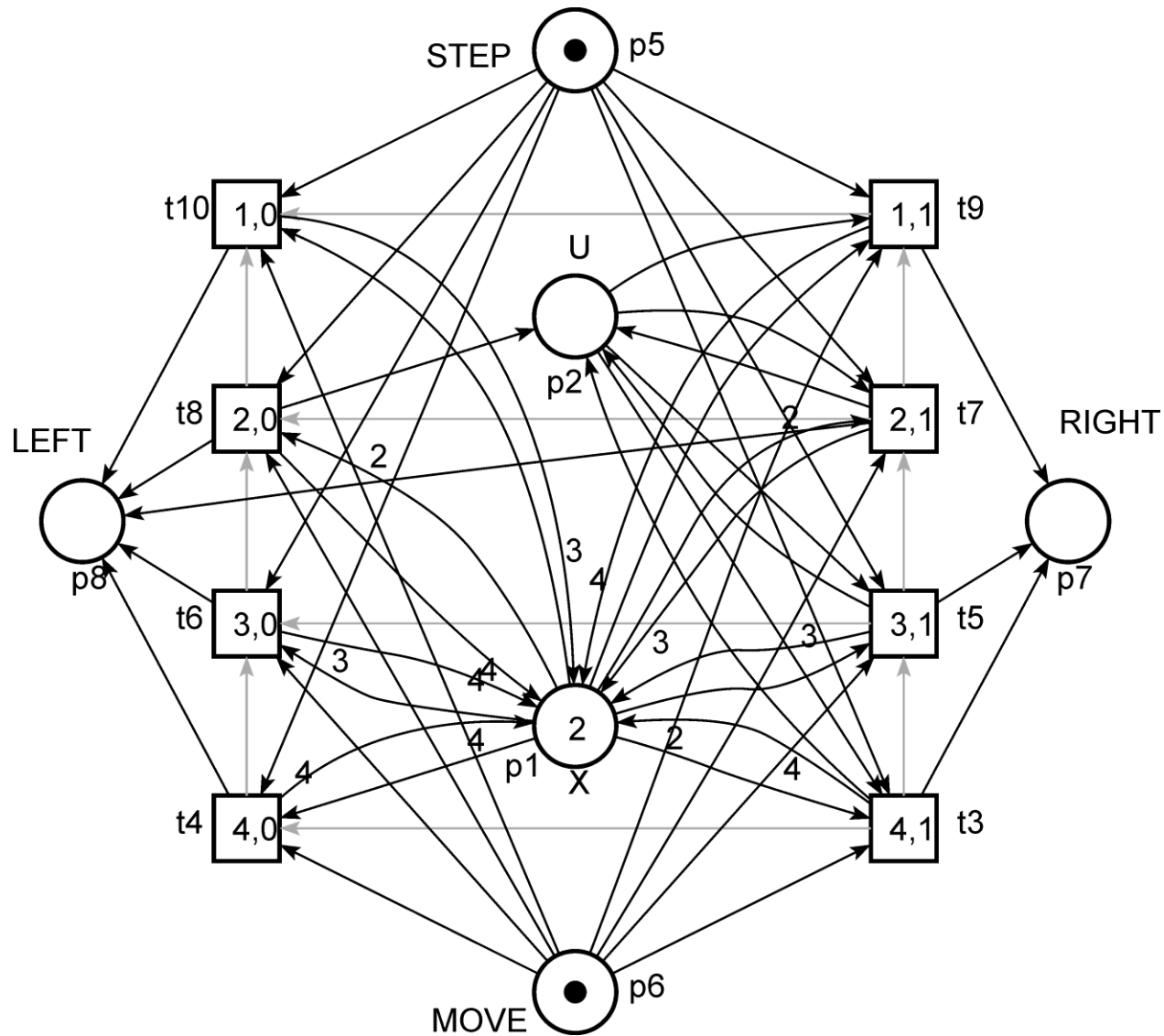


$$\left(\begin{array}{l} t(0, 1)_i : x0_i, u1_i \rightarrow x\emptyset_i, u1_{i-1}, \\ t(0, 2)_i : x0_i, u2_i \rightarrow x\cancel{1}_i, u1_{i+1}, \\ t(1, 1)_i : x1_i, u1_i \rightarrow x\cancel{1}_i, u2_{i-1}, \\ t(1, 2)_i : x1_i, u2_i \rightarrow x\emptyset_i, u2_{i-1}, \\ t(\emptyset, 1)_i : x\emptyset_i, u1_i \rightarrow x\cancel{1}_i, u1_{i-1}, \\ t(\emptyset, 2)_i : x\emptyset_i, u2_i \rightarrow x0_i, u2_{i+1}, \\ t(\cancel{1}, 1)_i : x\cancel{1}_i, u1_i \rightarrow x\cancel{1}_i, u1_{i-1}, \\ t(\cancel{1}, 2)_i : x\cancel{1}_i, u2_i \rightarrow x1_i, u2_{i+1}, \end{array} \right)$$

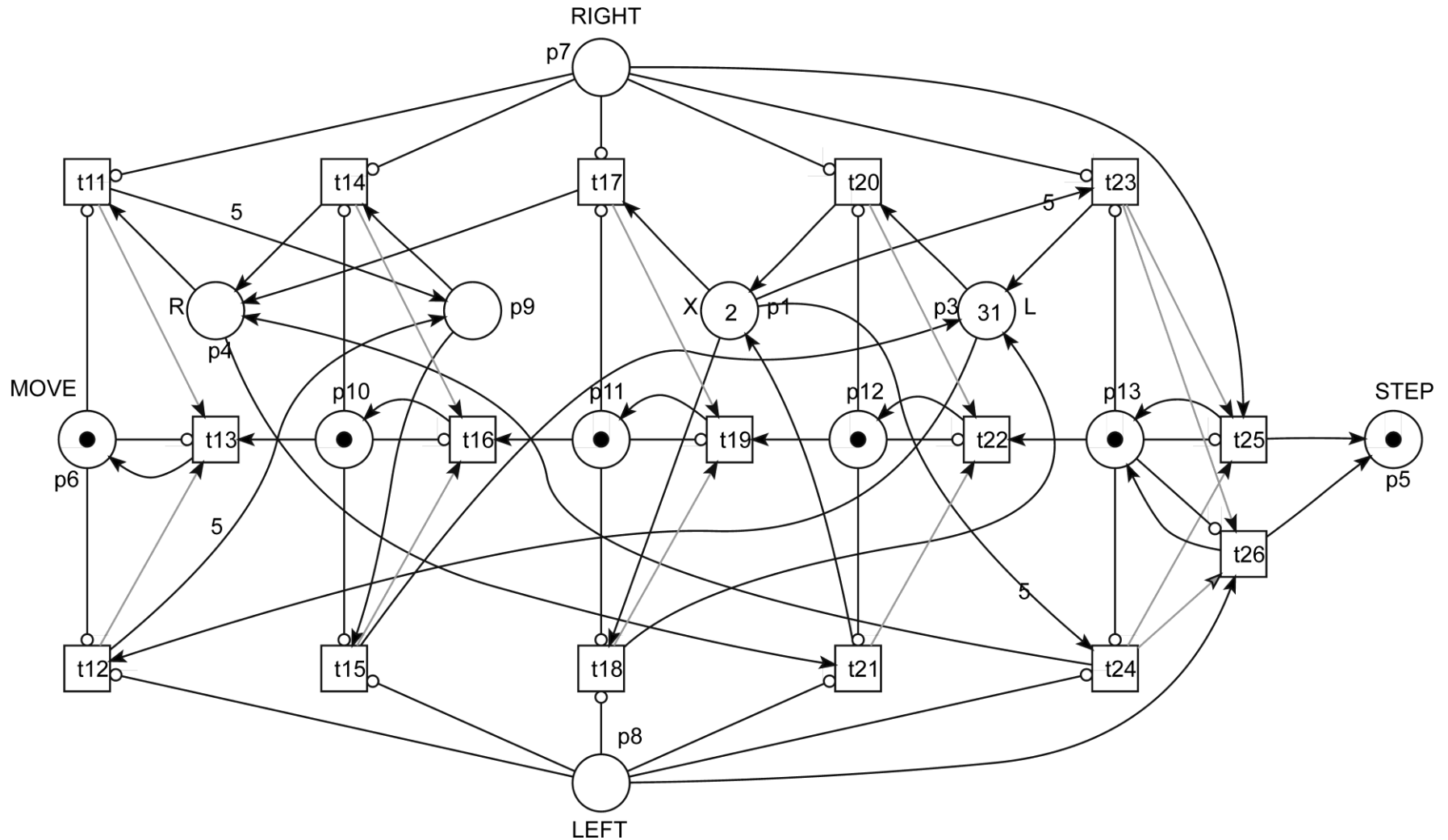
IV. Universal Sleptsov Net USN(13,26)



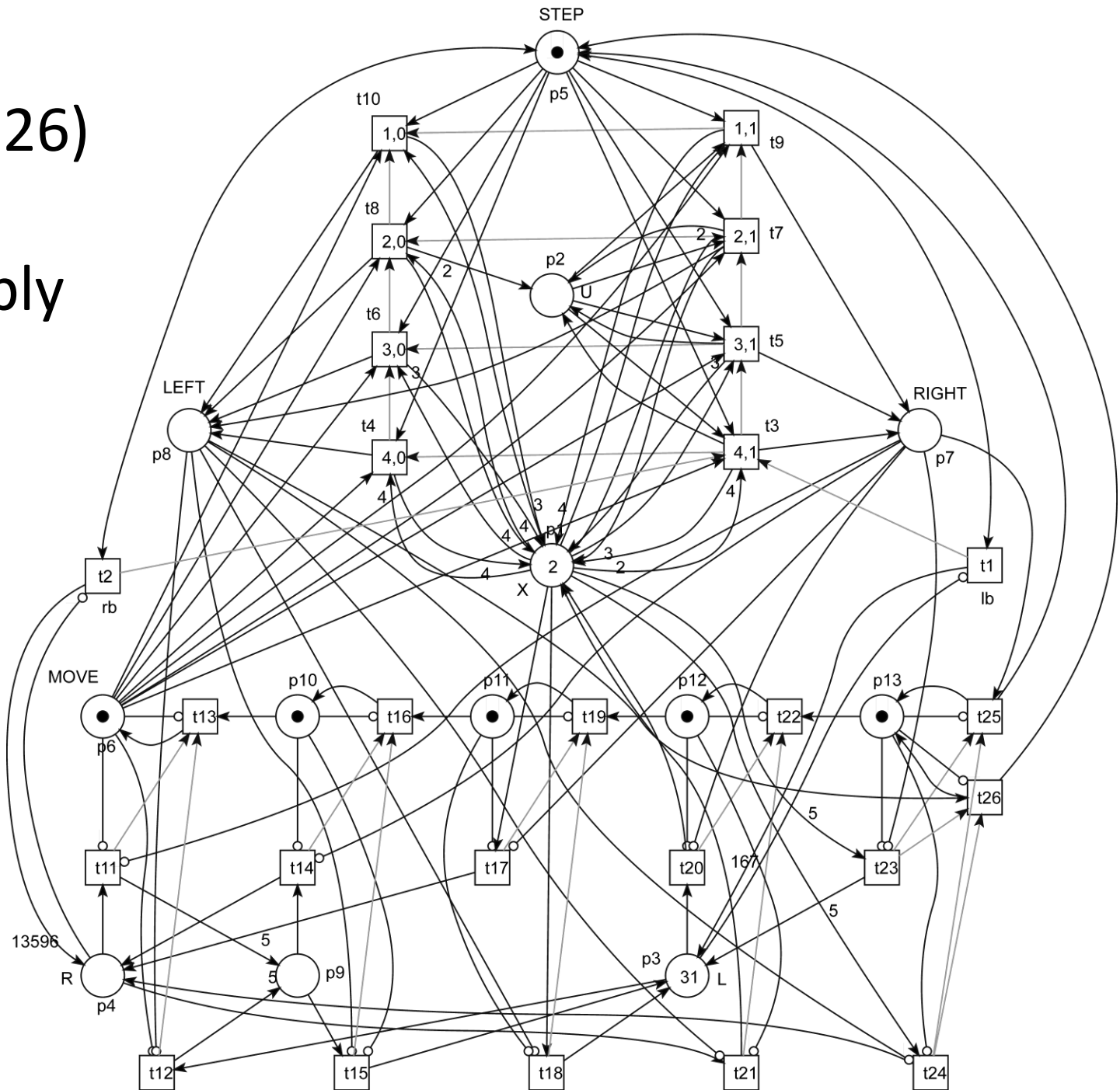
Transition function subnet TF



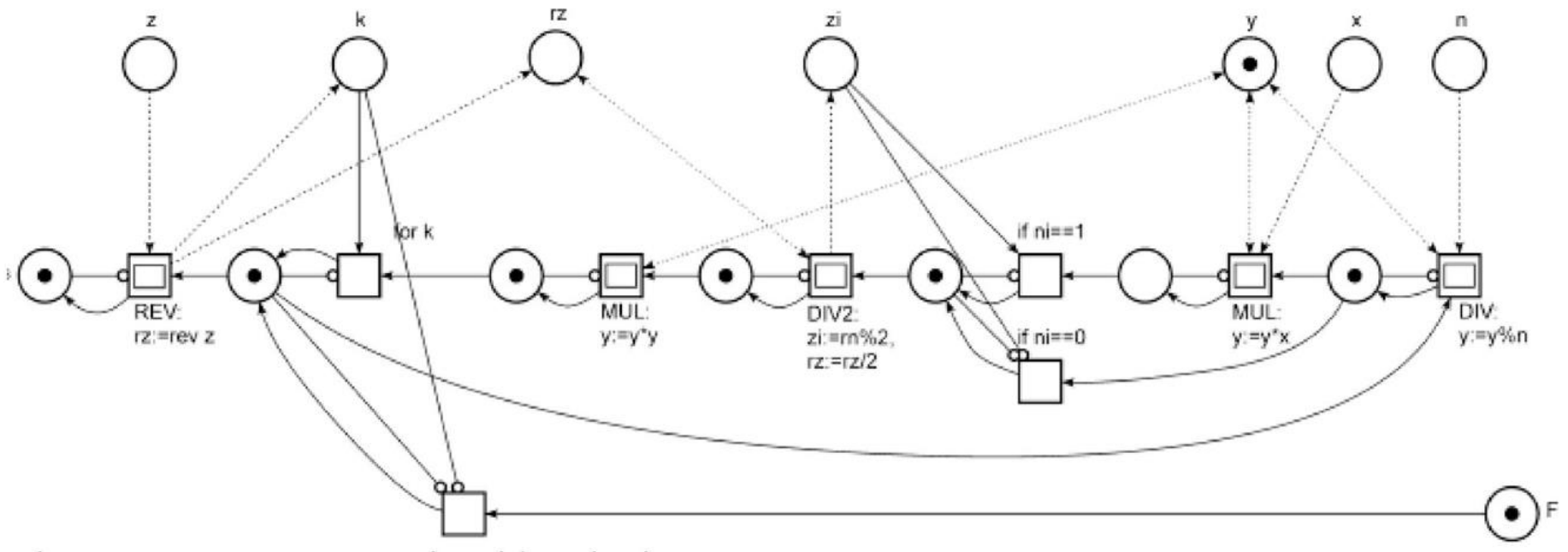
Head move subnet HM



USN(13,26) Final Assembly



V. Examples of SN programs: RSA encoding/decoding



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

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

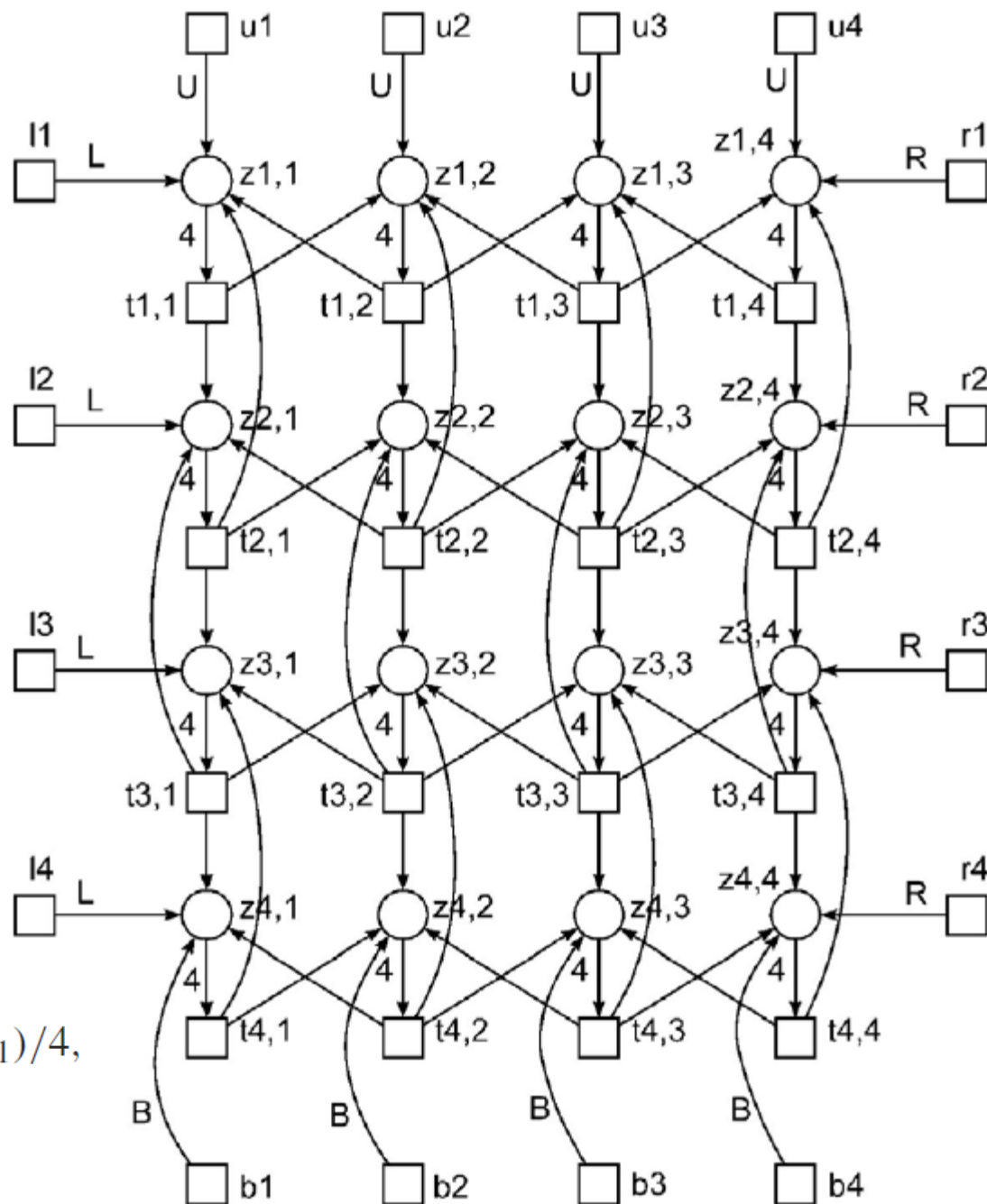
$$= \left(\dots \left(\left((x^{z_k})^2 \cdot x^{z_{k-1}} \right)^2 \cdot x^{z_{k-2}} \right)^2 \dots \cdot x^{z_2} \right)^2 \cdot x^{z_1}$$

Examples of SN programs: Solving Laplace equation

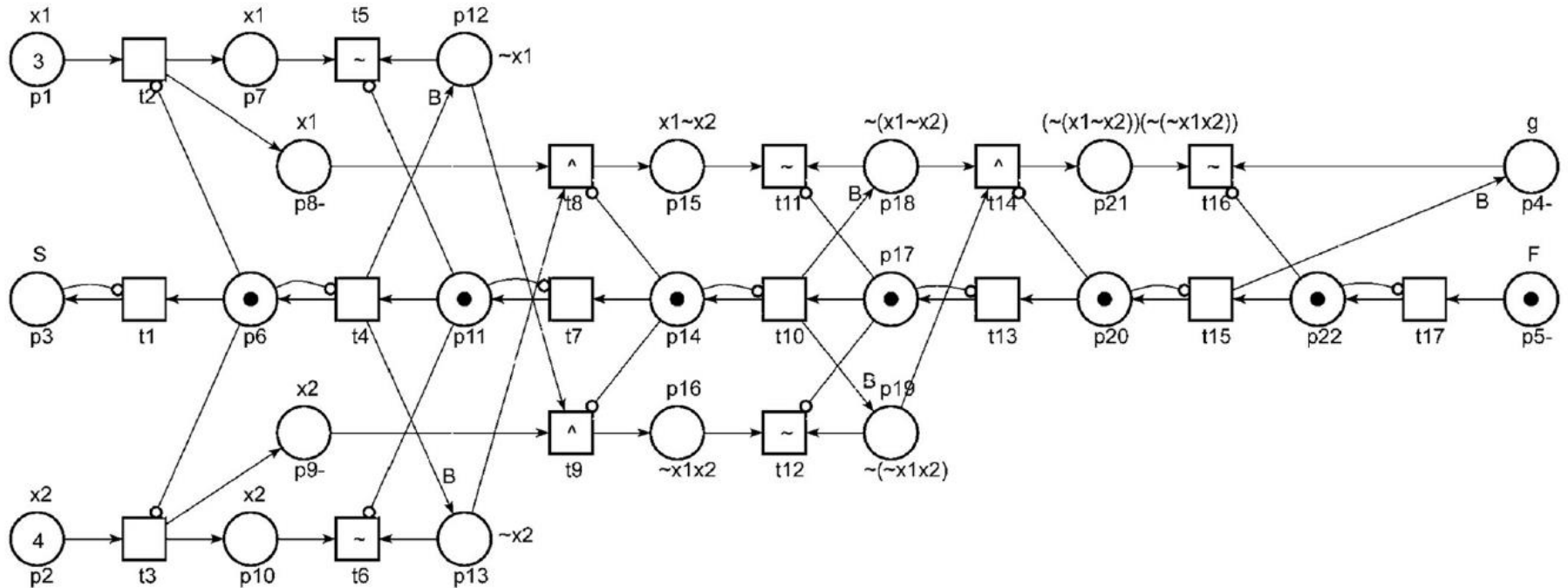
$$\frac{\partial^2 \varphi}{\partial x^2} + \frac{\partial^2 \varphi}{\partial y^2} = 0$$

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

$$\varphi_{i,j} = \varphi(x_i, y_j).$$



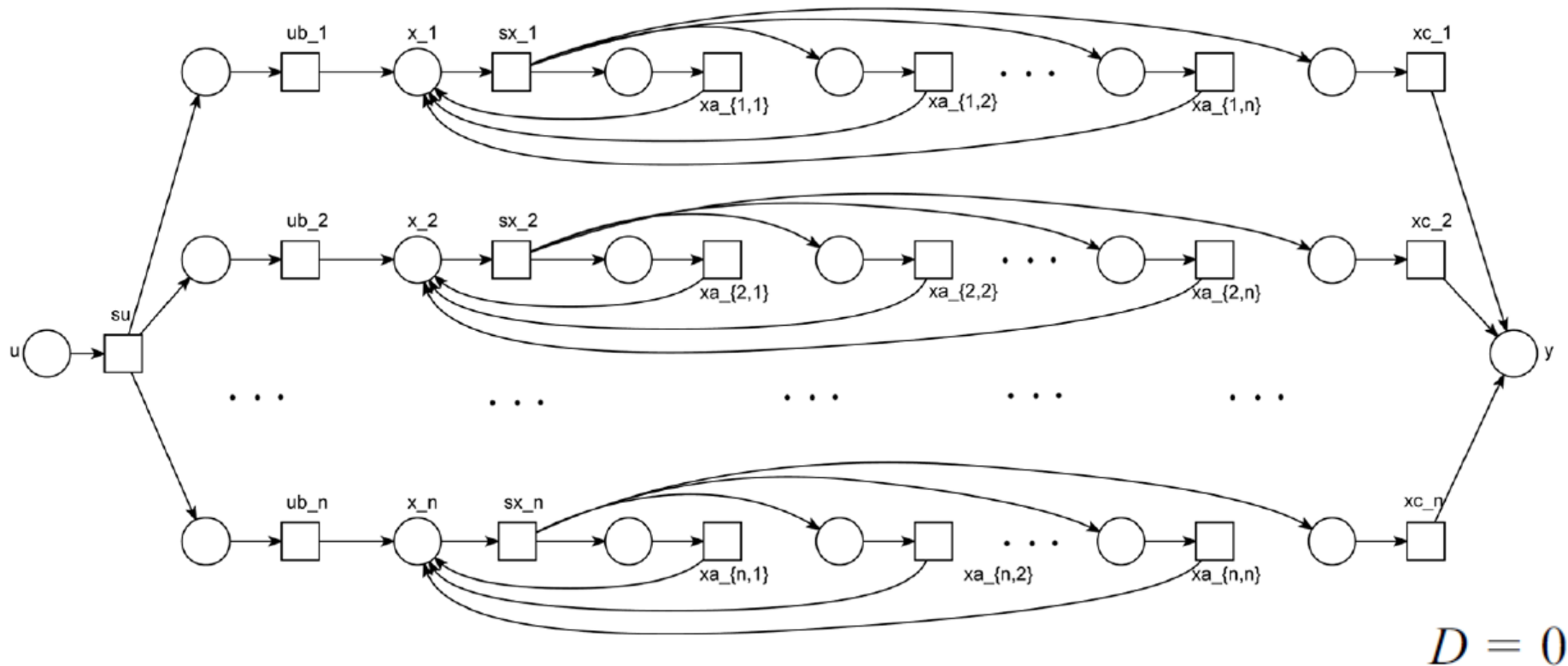
Examples of SN programs: Computing Fuzzy Logic Function



$$\varphi = x_1\bar{x}_2 \vee \bar{x}_1x_2$$

Examples of SN programs:

Discrete-Time Linear Control in Two Tacts



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

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

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