

WSIZ, 2023

Sleptsov Net Computing

Dmitry Zaitsev

<http://daze.ho.ua>

Modeling by Petri and Sleptsov nets

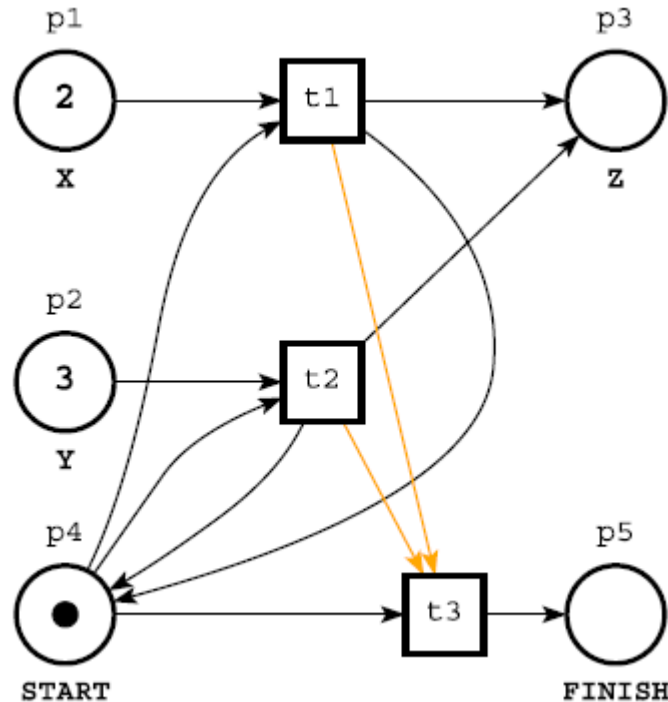
- Verification of protocols by Petri nets
- Model analysis methods. Composition of clans
- Analysis of Computational Grids and Clouds by Infinite Petri Nets
- Evaluation of System Performance by Colored Petri Nets
- Sleptsov Net Computing (SNC)

Recent references for SNC

- [Sleptsov Net Computing Resolves Modern Supercomputing Problems, The April 21, 2023, edition of ACM TechNews](#)
- [Dmitry Zaitsev \(2023\) Sleptsov Net Computing resolves problems of modern supercomputing revealed by Jack Dongarra in his Turing Award talk in November 2022, International Journal of Parallel, Emergent and Distributed Systems.](#)
- [<https://wsiz.edu.pl/blog-naukowy/sleptsov-net-computing-resolves-problems-of-modern-supercomputing-revealed-by-jack-dongarra-in-his-turing-award-talk-in-november-2022/>](#)
- [<http://daze.ho.ua/12-snc.html>](#)

Inhibitor or priority net is Turing-complete (a computationally universal system)

Tilak
Agerwala,
1974



An example of
priority net
 $z=x+y$

Write programs

or

Draw programs?

Modern visual programming

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

P-technology of programming

```

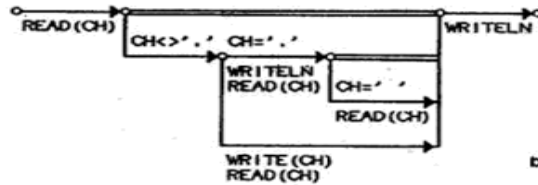
PROGRAM PRINID(INPUT,OUTPUT);
VAR CH:CHAR;
BEGIN
  READ(CH);
  WHILE CH<>' ' DO
    IF CH='.' THEN
      BEGIN
        WRITELN;
        READ(CH)
      WHILE CH='.' DO
        READ(CH)
      END
    ELSE
      BEGIN
        WRITE(CH);
        READ(CH)
      END;
    WRITELN
  END.

```

```

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

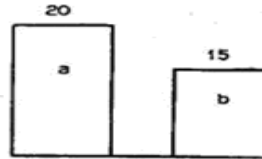
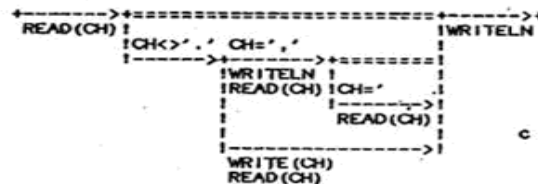
```



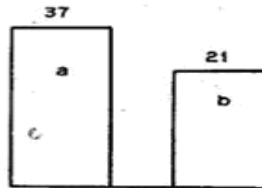
```

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

```



Число строк на экране
дисплея или на листинге

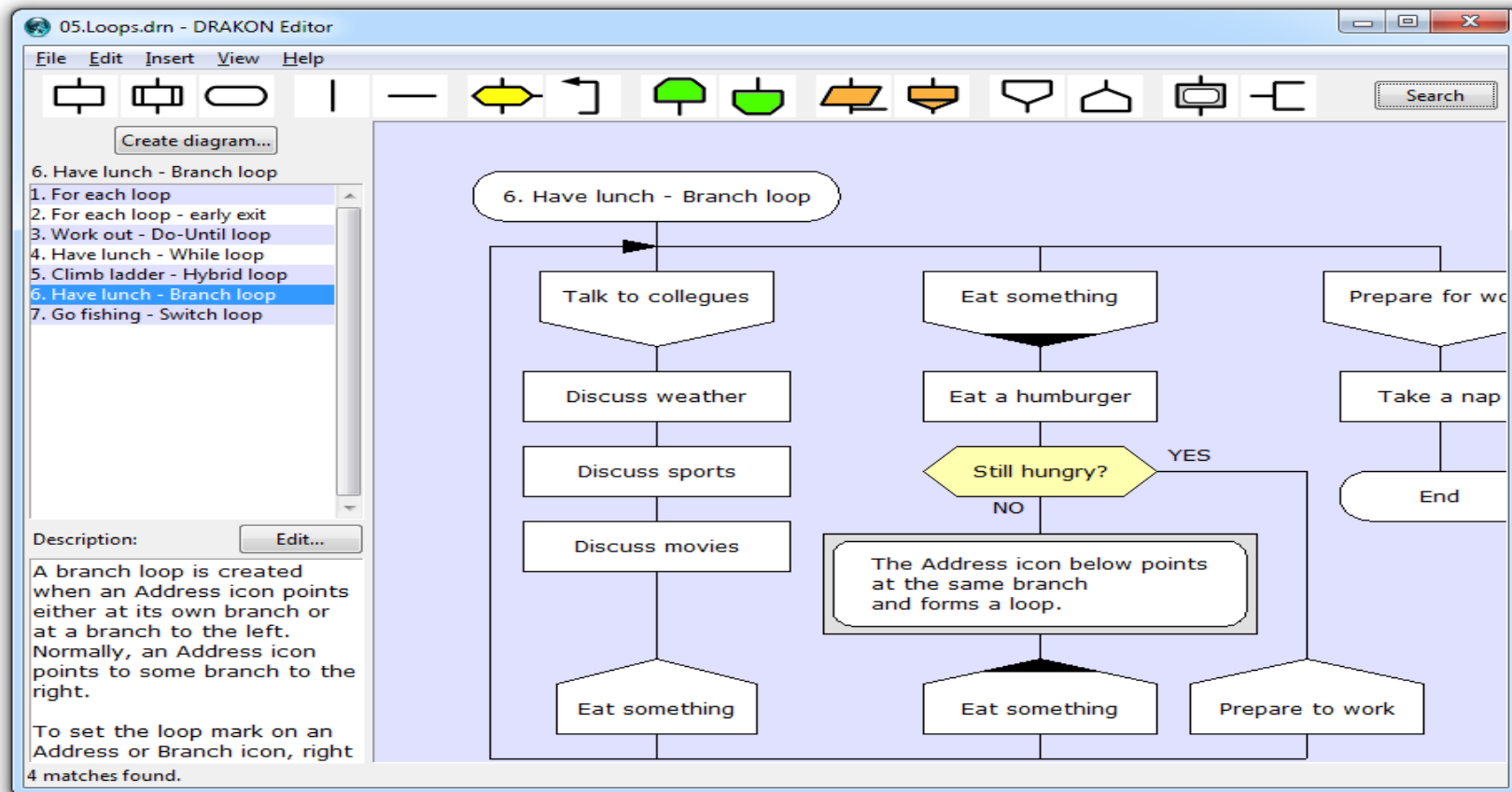


Число байтов памяти
(по управляющим структурам
программы) или скорость ввода

d

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

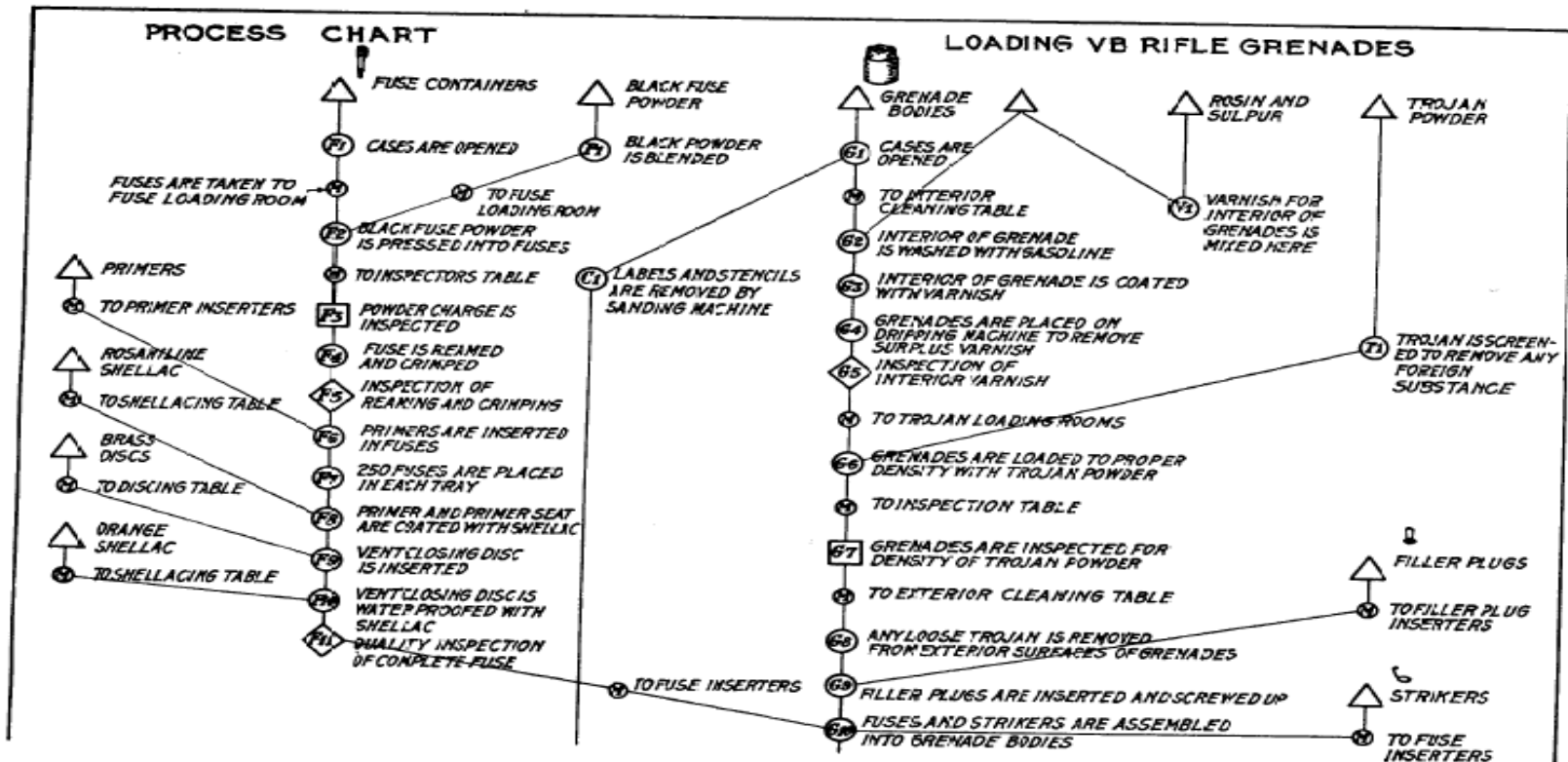
Drakon



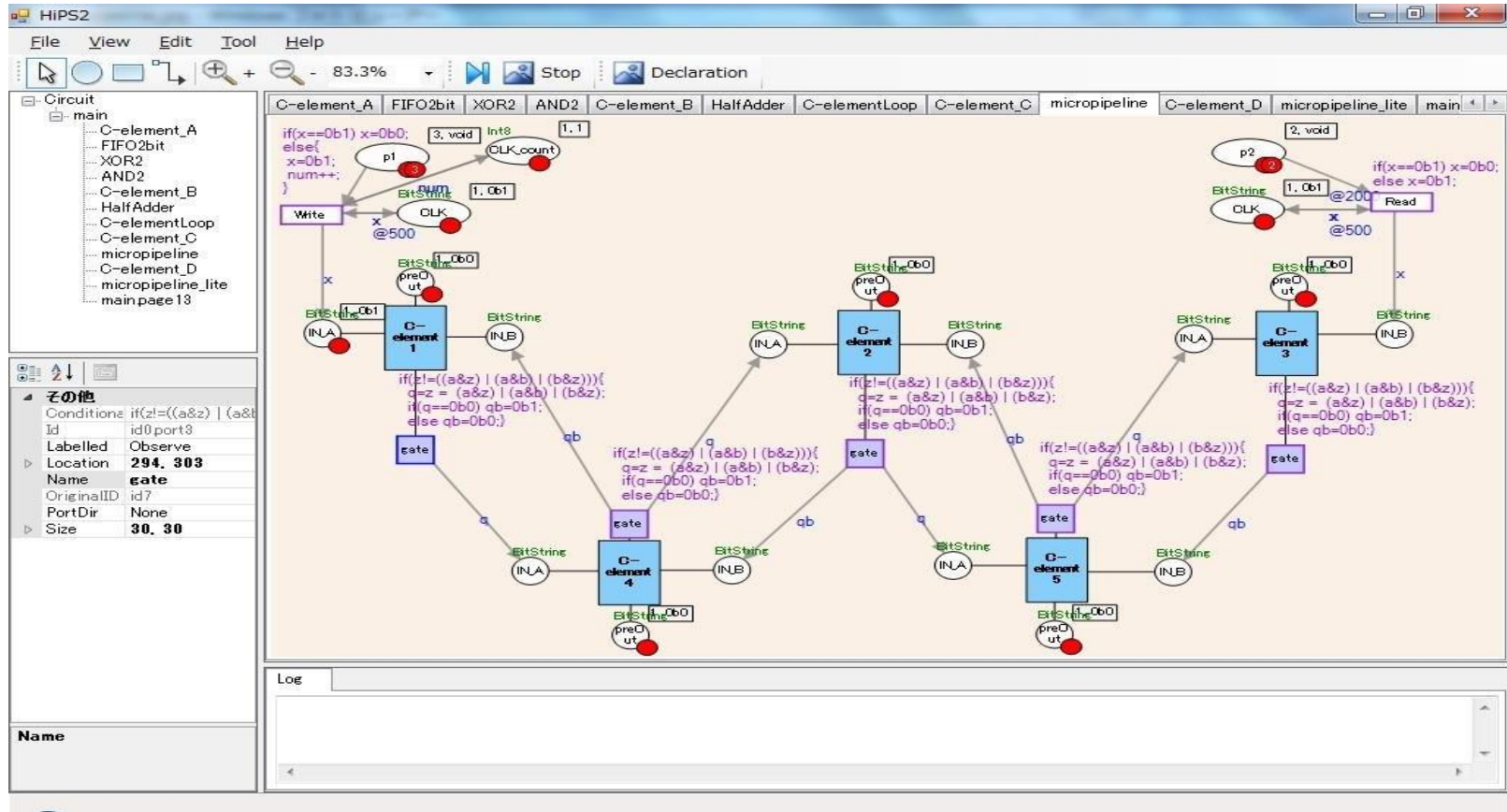
Schemes of processes and flow-charts

- Schemes of production processes, Frank and Lilian 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

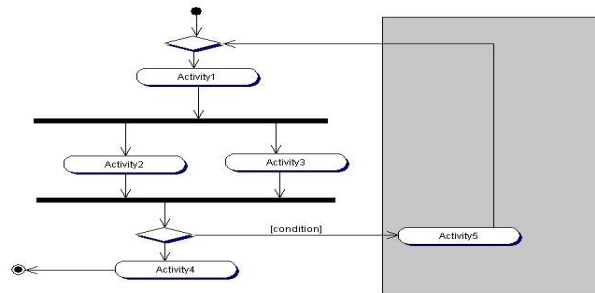
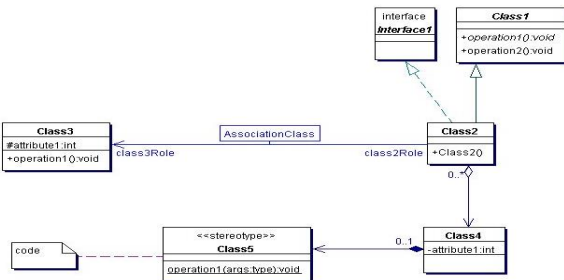
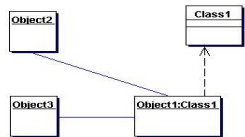
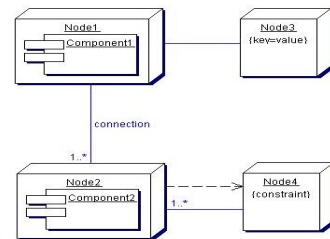
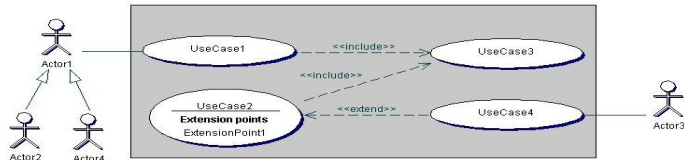
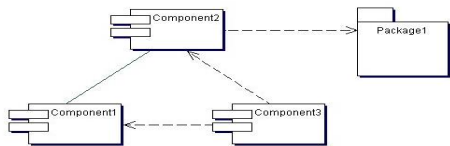
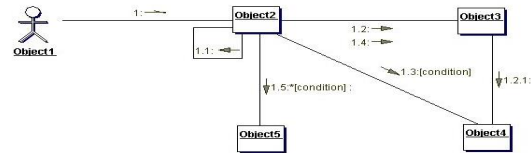
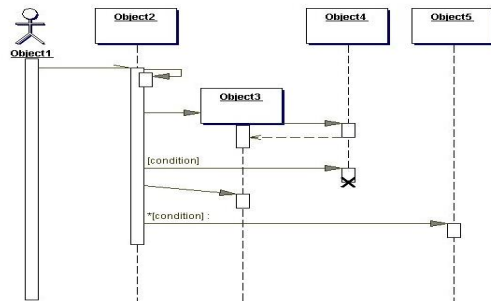
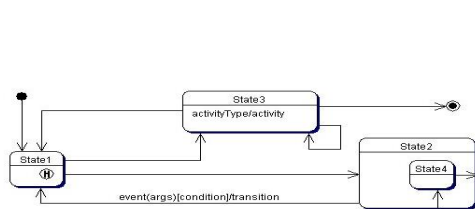
Frank and Lilian Gilbreth Processes Schemata



Programming on Petri Nets



UML



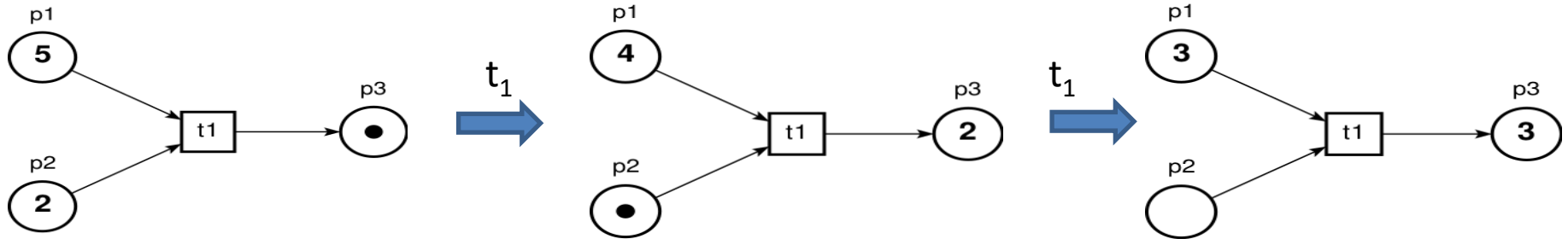
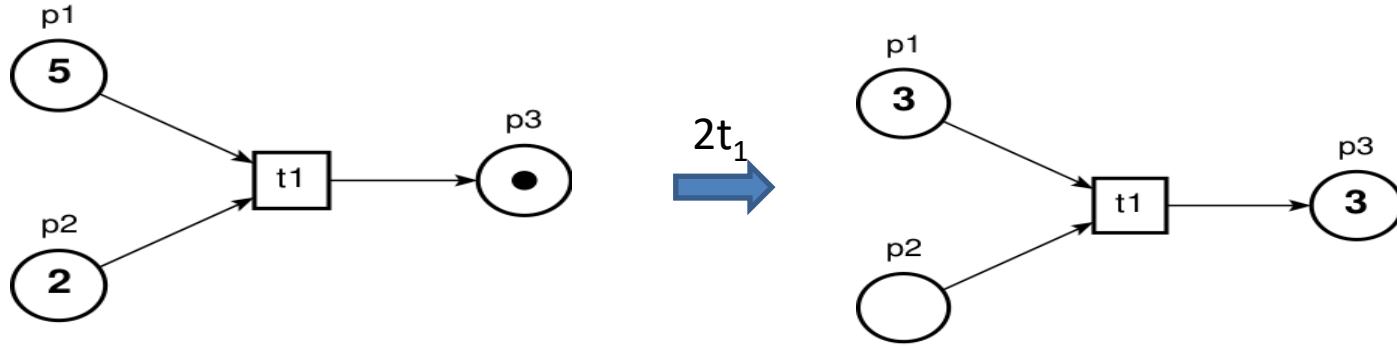
Way to uniform concept

- Textual programming
- Graphs loaded with textual programming language
- Completely graphical programming – nothing but graphs (text as comments only)
- Inhibitor (or priority) Petri net – fast uniform language of concurrent programming
- Mass parallel computations
- Fine granulation of parallel processes
- Hardware implementation as computing memory

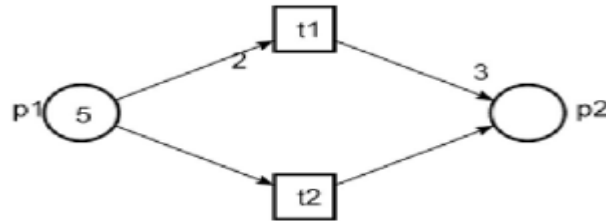
Transition firing rules

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

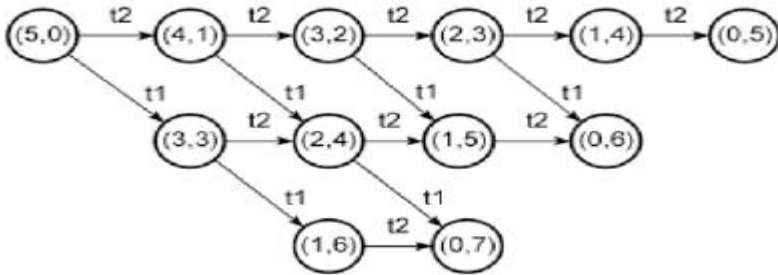
Comparing Sleptsov and Petri nets



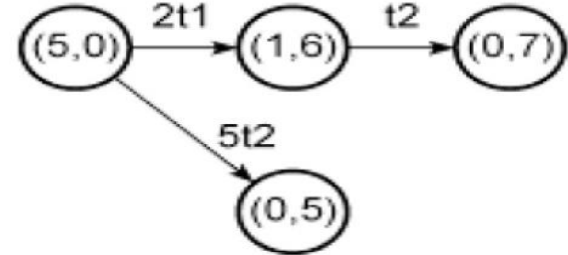
Sleptsov net – Multiple firing



Marking graphs



Petri net



Sleptsov net

Sleptsov nets run fast

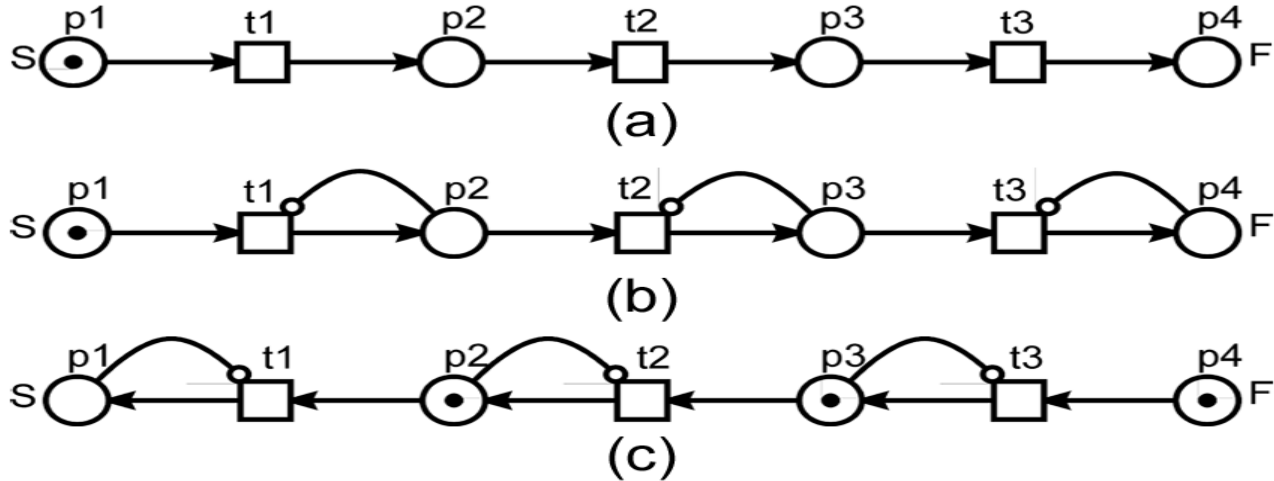
$$x \succ y = \begin{cases} x/y, & \text{if } y > 0 \\ 0, & \text{if } y = -1, x > 0 \\ \infty, & \text{if } y = -1, x = 0. \end{cases}$$

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

Comparing time of operations (linear scale – number of steps)

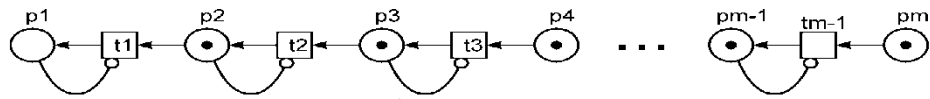
Operation	PN	SN
CLEAN	$x + 2$	2
MOVE	$x + 2$	2
COPY	$2 \cdot x + 3$	4
ADD	$x + y + 2$	3
SUB	$\max(x, y) + 3$	3
GT	$\max(x, y) + 3$	4
MUL	$y \cdot (2 \cdot x + 3) + x + 3$	$11 \cdot \log_2 y + 3$
DIV	$(x / y) \cdot (2 \cdot y + 2) + (x \% y) + y + 4$	$39 \cdot (\log_2 x - \log_2 y) + 19$

Peculiarities of programming in Sleptsov nets

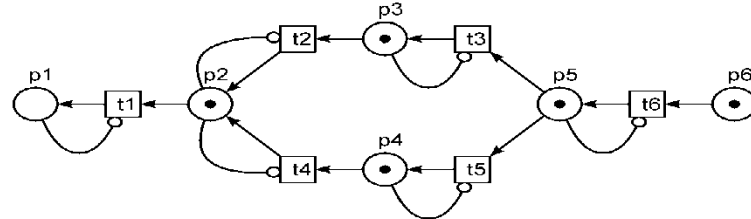


- Reverse control flow – moving “hole” (c)
- Using inhibitor arc to control the transition firing

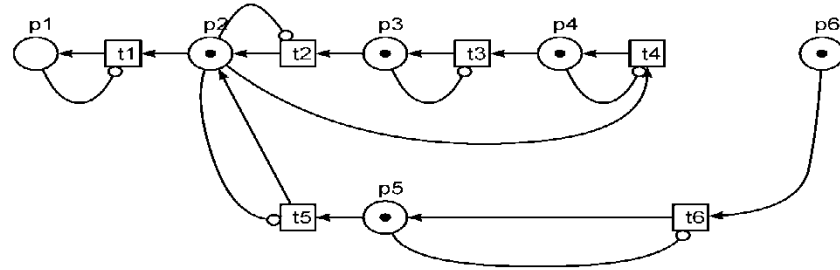
Basic statements



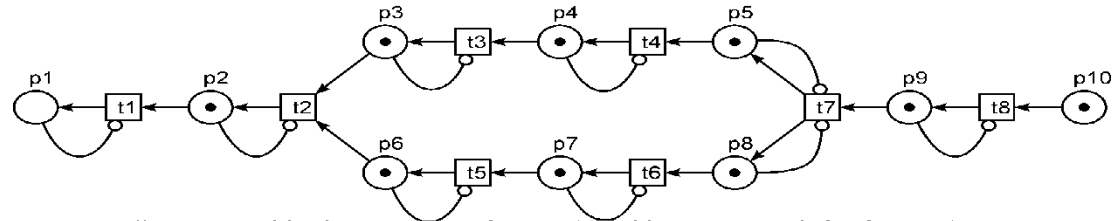
a) sequence;



b) branching;

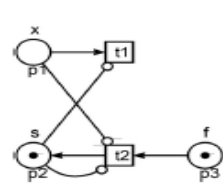


c) cycle;



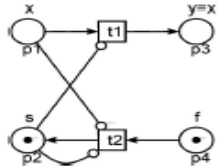
d) Parallel execution (split t_2 and join t_7)

Basic subnets (subroutines)



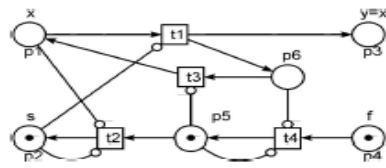
CLEAN:

$x := 0$



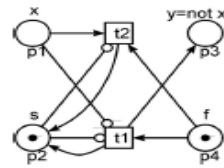
MOVE:

$y := x, x := 0$



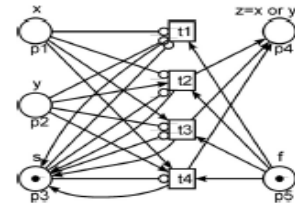
COPY:

$y := x$



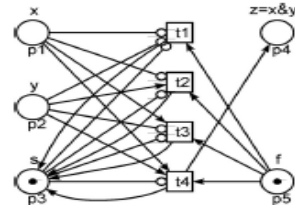
NOT(x):

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



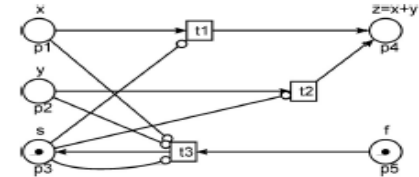
OR(x,y)

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



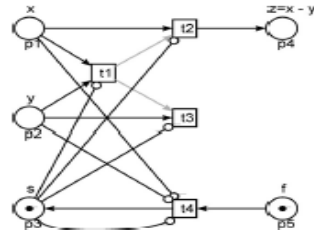
AND(x,y):

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



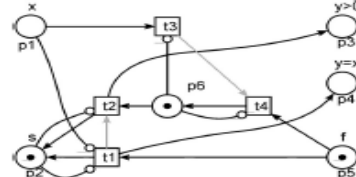
ADD(x,y):

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



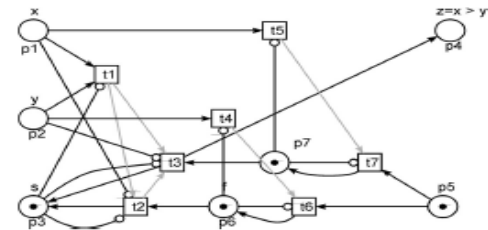
SUB(x,y):

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



GT0(x):

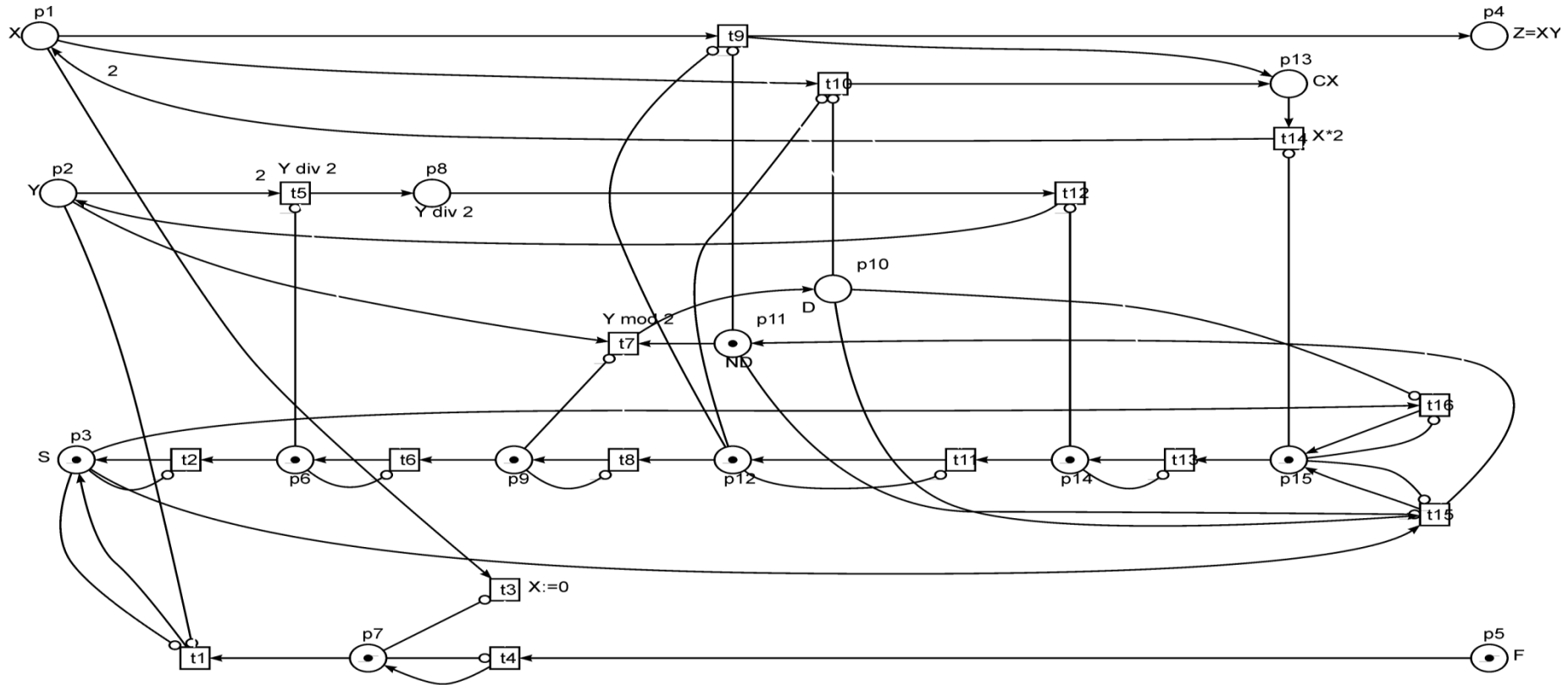
$y := (x > 0), z := (x = 0)$



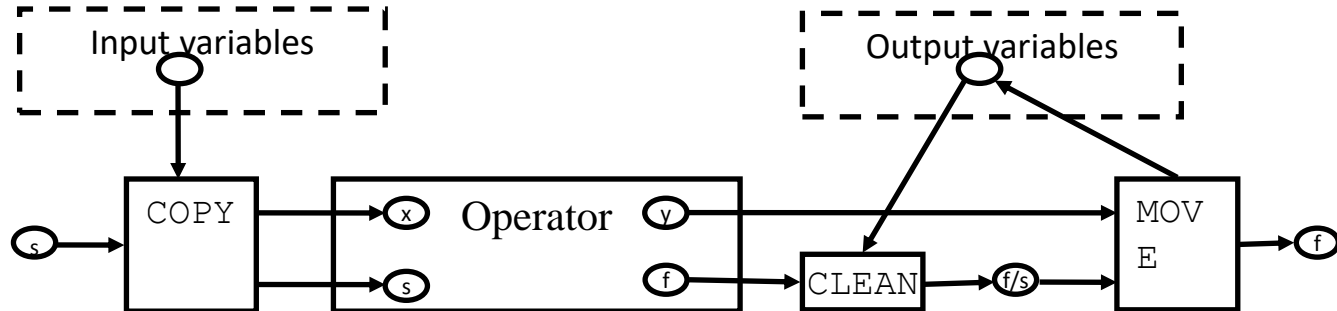
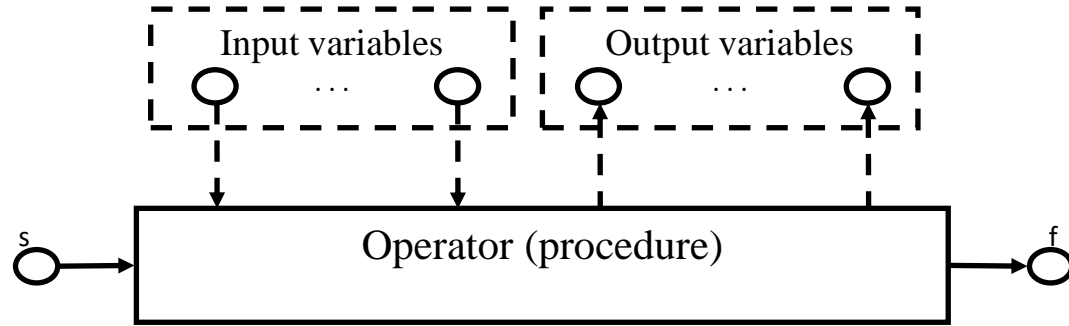
GT(x,y):

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

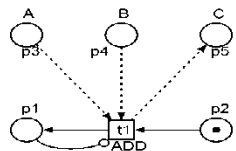
Fast multiplication on Sleptsov nets



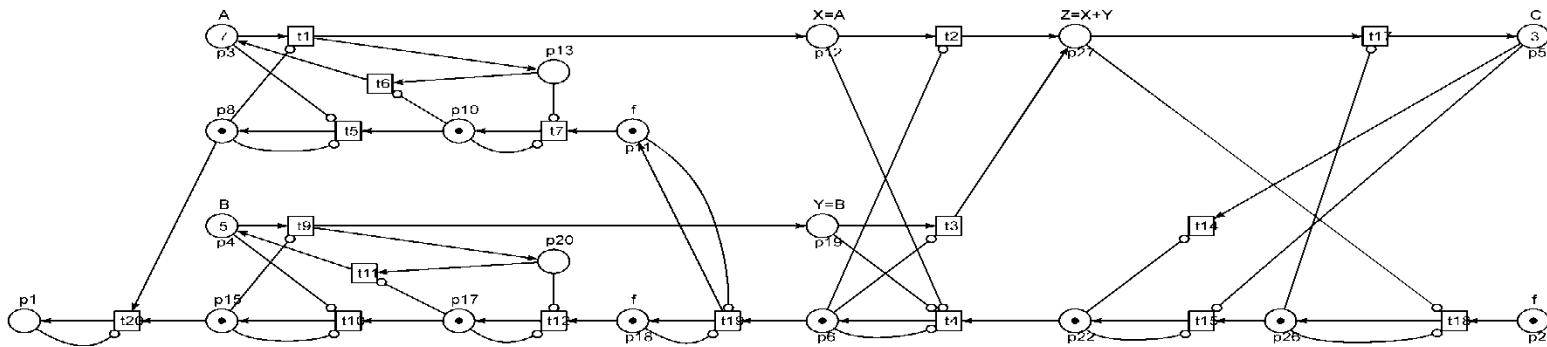
Working with variables



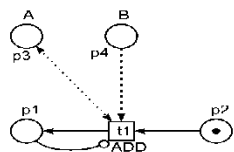
Expansion of dashed arcs



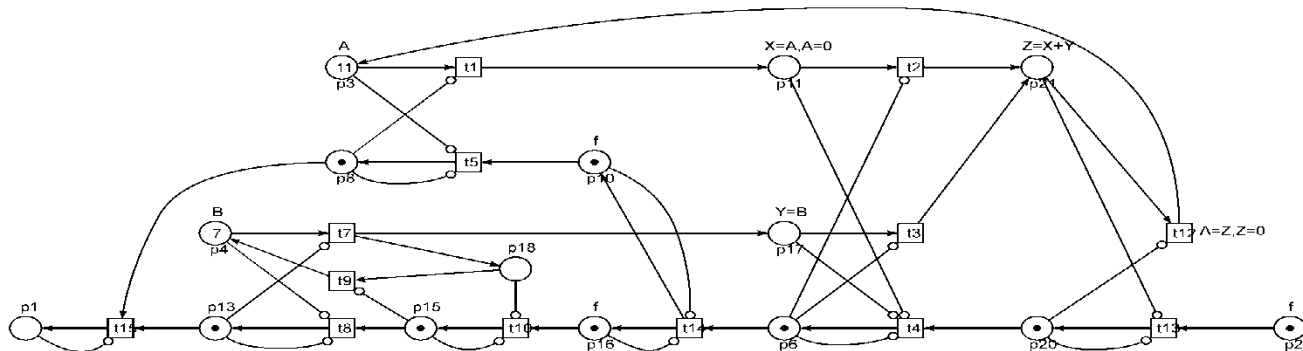
a)
 $c = \text{add}(a, b)$



b) expansion of a)

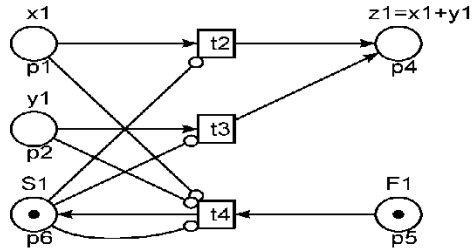


c)
 $a = \text{add}(a, b)$

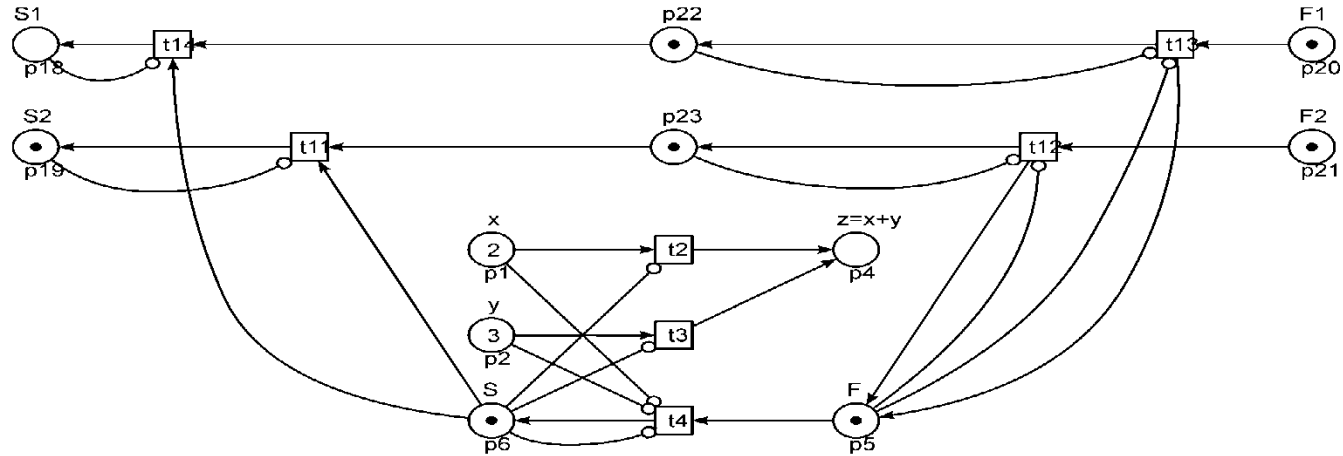


d) expansion of c)

Call of subnets

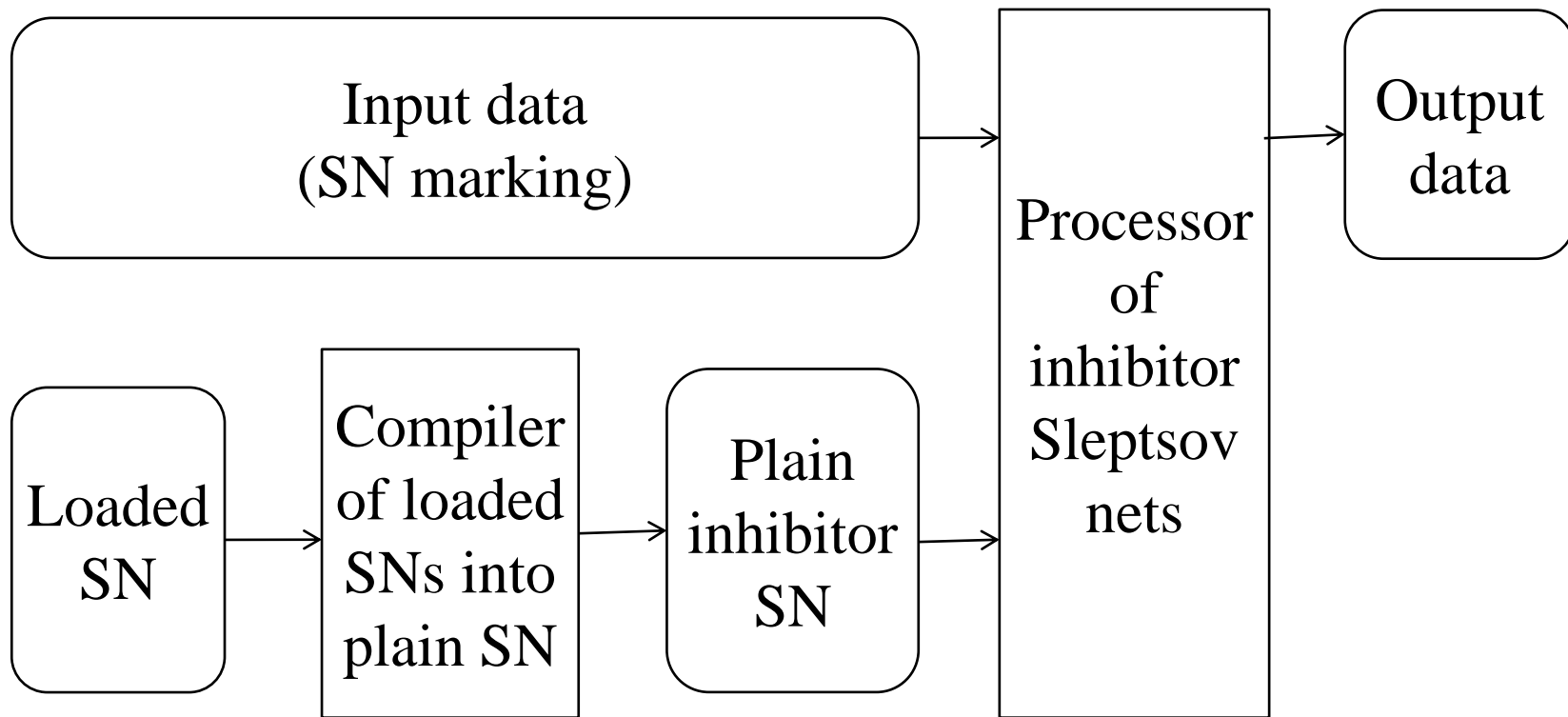


a) inline



b) call-return

Paradigm of Sleptsov net computing



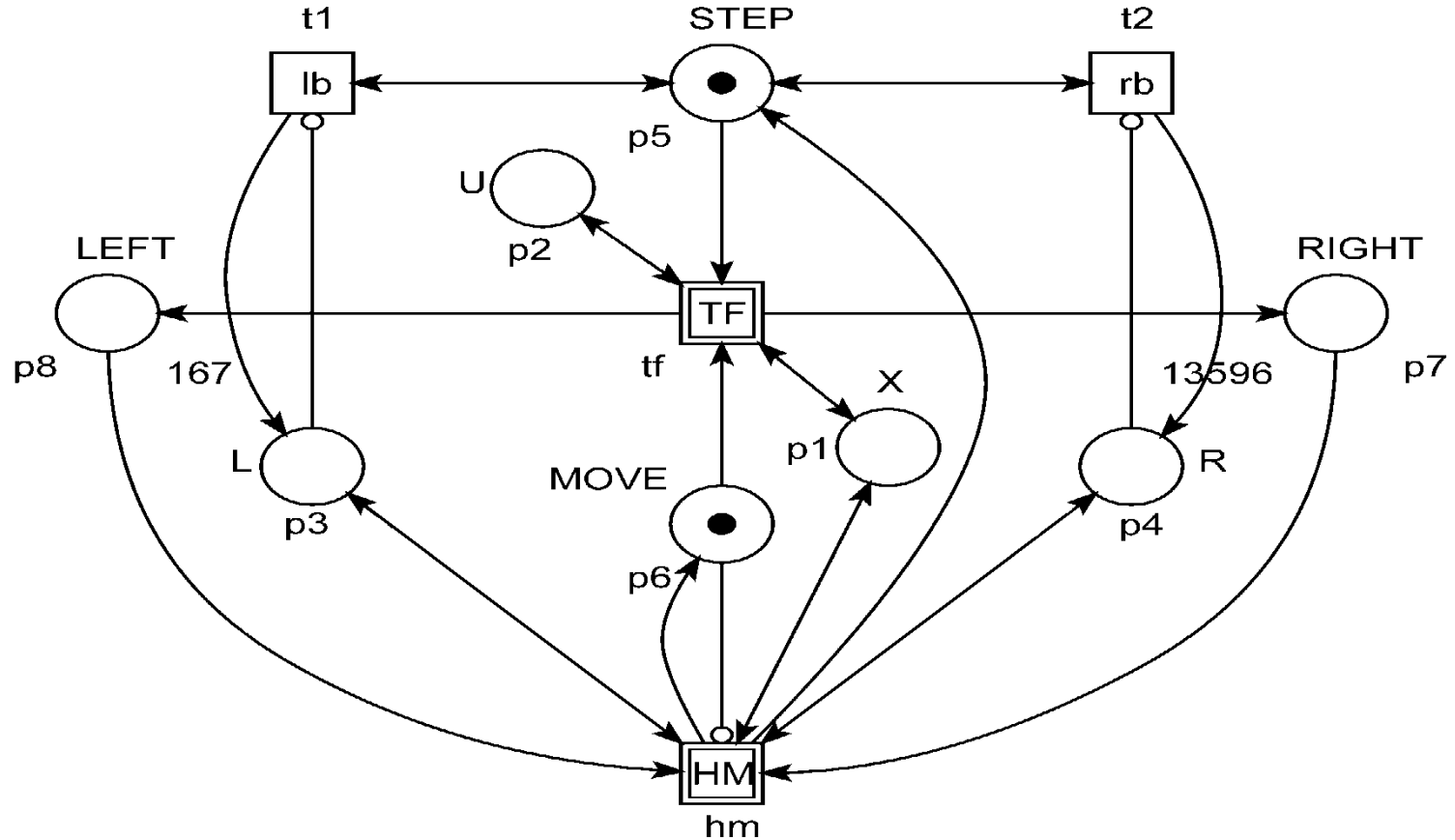
Concept of universal Sleptsov net



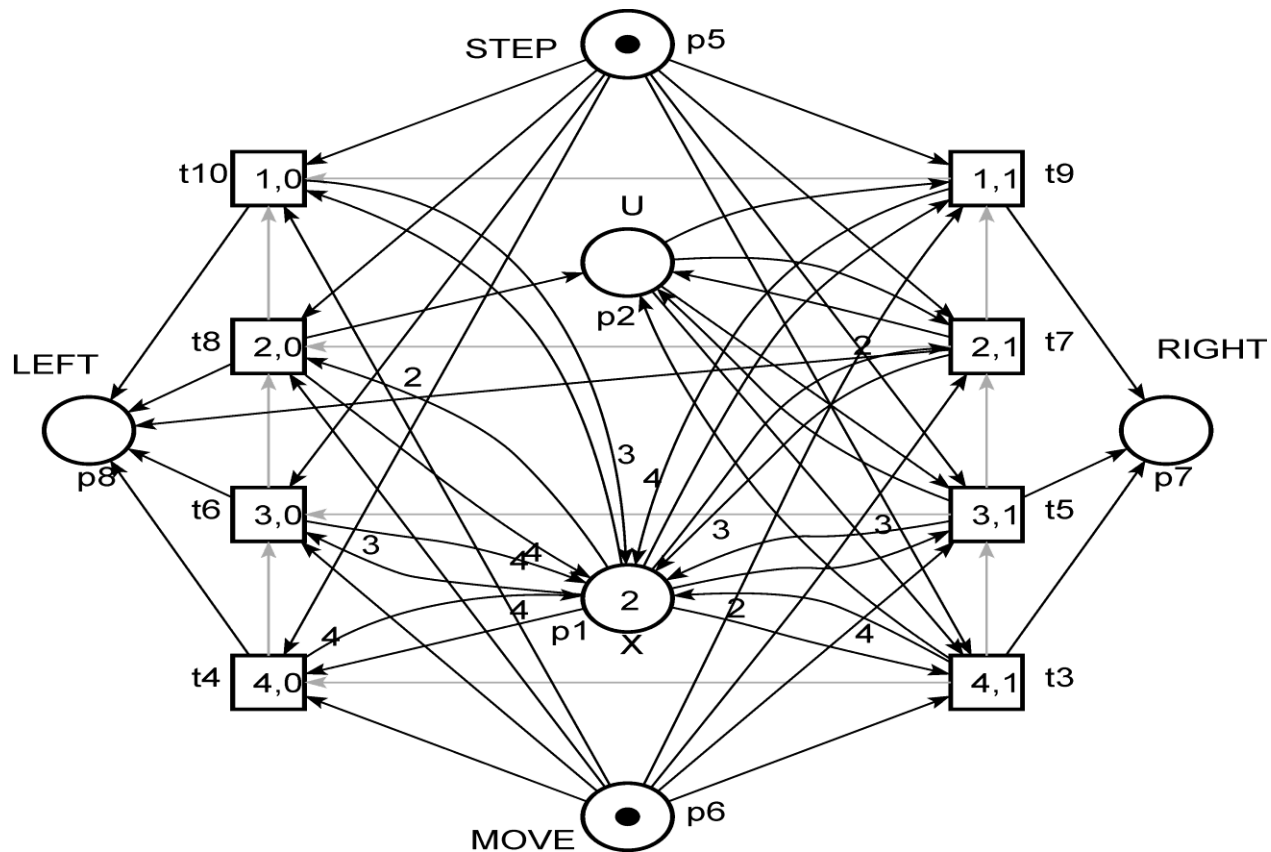
Universal Petri and Sleptsov nets

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

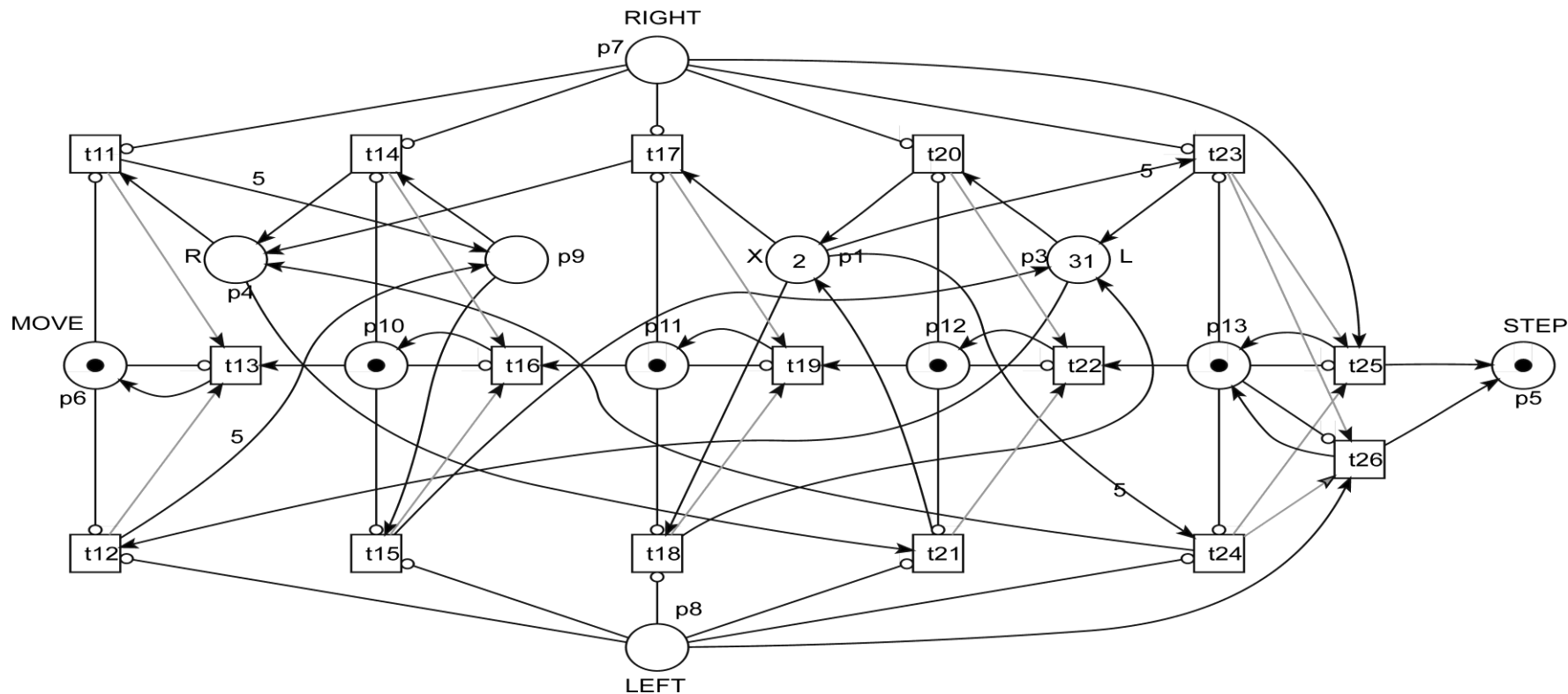
Universal Sleptsov net USN(13,26)



Transition Function Subnet TF

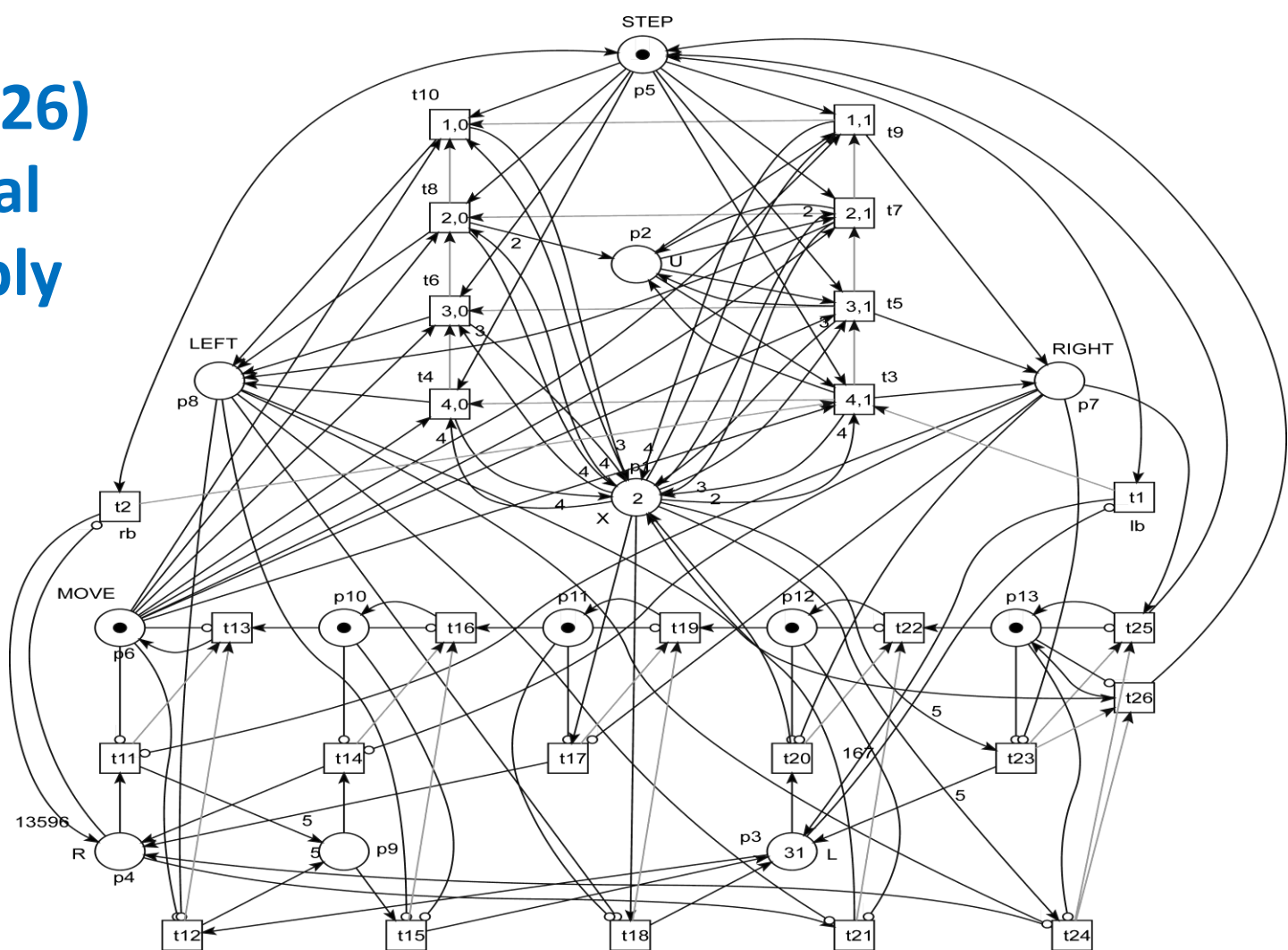


Head Move subnet HM

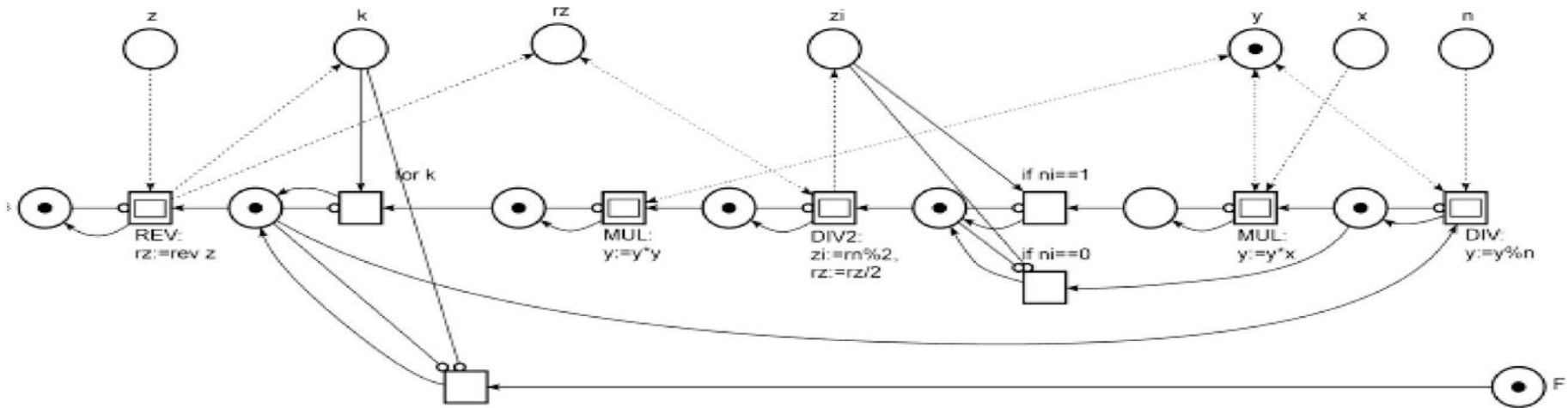


USN(13,26)

General assembly



Examples of programs: RSA encryption / decryption



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

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

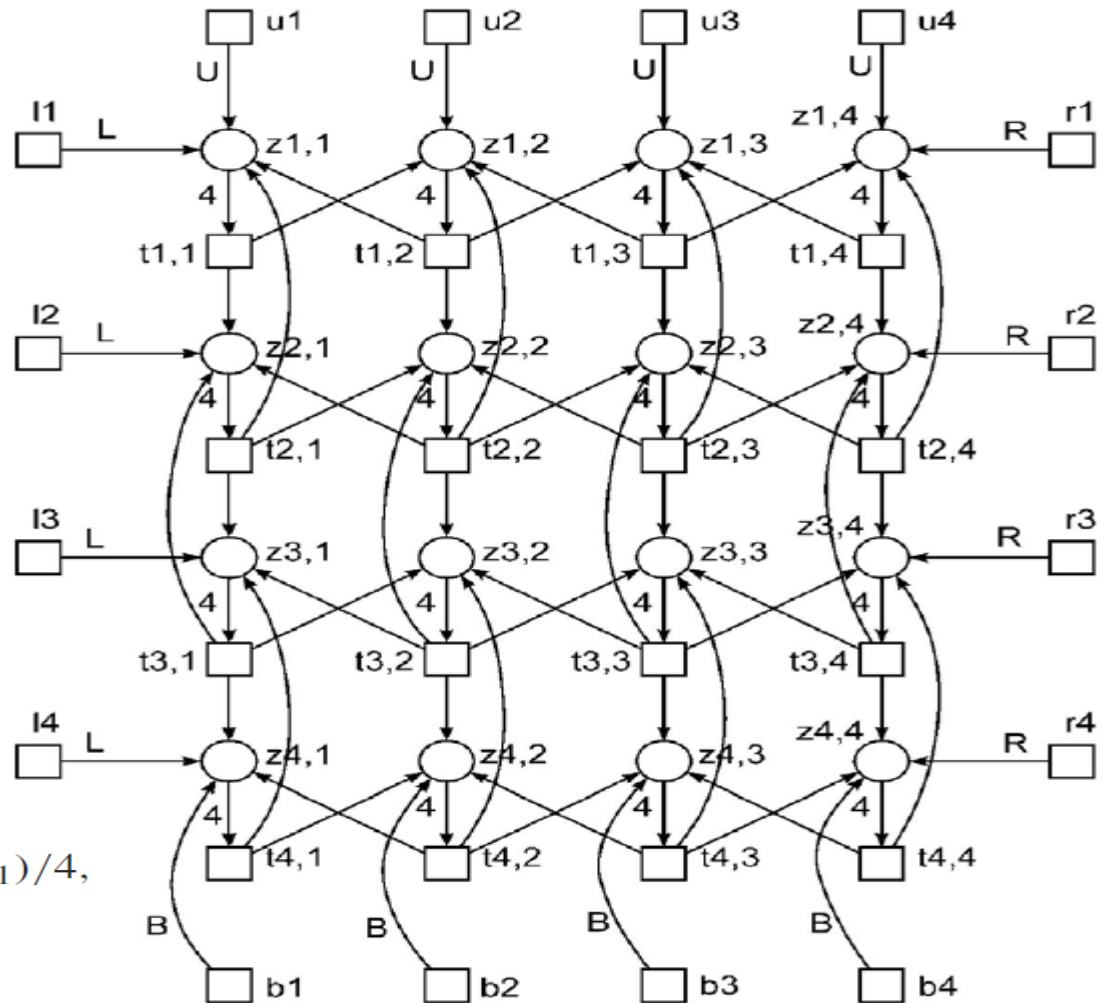
$$= \left(\dots \left(\left((x^{z_k})^2 \cdot x^{z_{k-1}} \right)^2 \cdot x^{z_{k-2}} \right)^2 \dots \cdot x^{z_2} \right)^2 \cdot x^{z_1}$$

Examples of programs: Solving Laplace equation

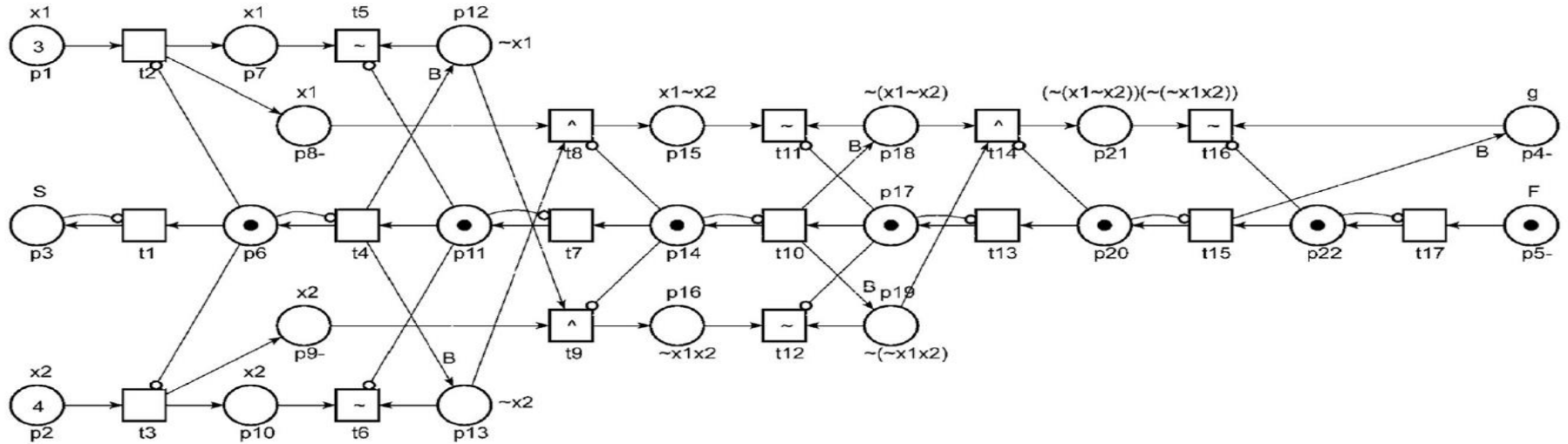
$$\frac{\partial^2 \varphi}{\partial x^2} + \frac{\partial^2 \varphi}{\partial y^2} = 0$$

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

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



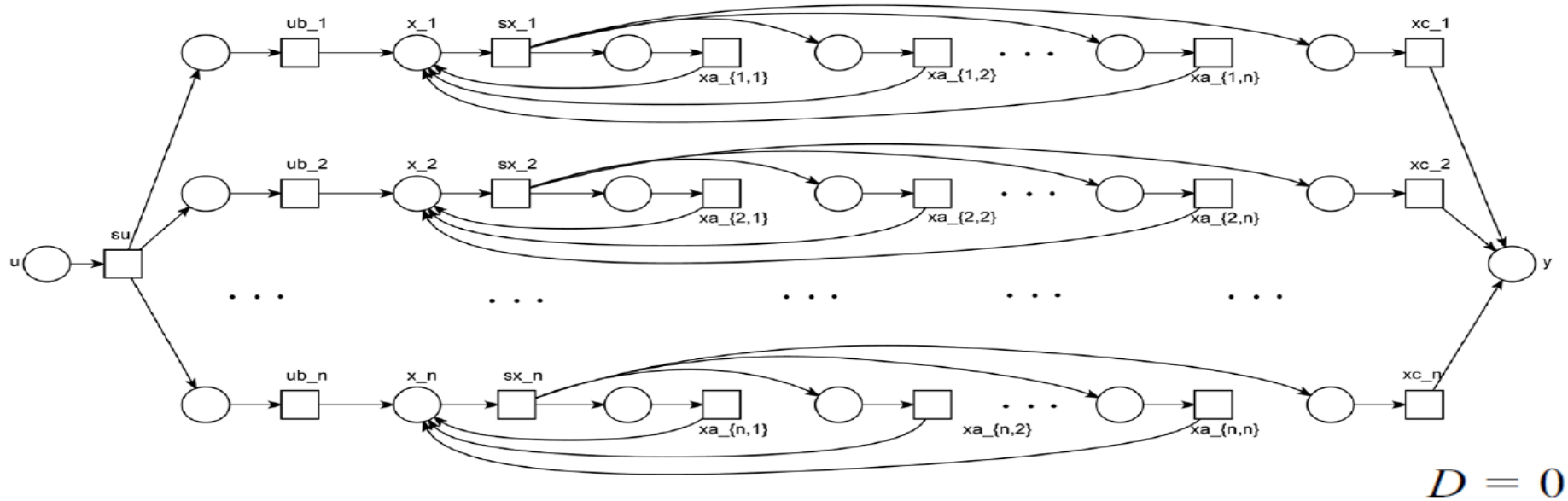
Examples of programs: Computing fuzzy logic functions



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

Examples of programs:

Linear control with discrete time in two ticks



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

Conclusions

- **Sleptsov nets run exponentially faster than Petri nets**
- **Universal Sleptsov net – prototype of a processor in paradigm of Sleptsov net computing; smallest universal net contains 39 nodes and runs in polynomial time**
- **Advantages of Sleptsov net computing: graphical language of concurrent programming, methods for verification of concurrent programs, fine granulation of parallel processes, mass parallel computations on computing memory**

References

- Zaitsev D.A. Sleptsov Nets Run Fast, [IEEE Transactions on Systems, Man, and Cybernetics: Systems, 2016, Vol. 46, No. 5, 682 – 693.](#)
- Zaitsev D.A., Jürjens J. Programming in the Sleptsov net language for systems control, [Advances in Mechanical Engineering, 2016, Vol. 8\(4\), 1-11.](#)
- Dmitry A. Zaitsev, Strong Sleptsov nets are Turing complete, [Information Sciences, Volume 621, 2023, 172-182.](#)