Translated from: Zaitsev D.A. Petri Nets and Modelling of Systems: textbook for laboratory training. Odessa: ONAT, 2007, 42 p. In Ukr. https://dimazaitsev.github.io/pdf/pnmsua.pdf

D.A.Zaitsev

Petri nets and modeling of systems

Students' book for practical and laboratory training

Practical lesson 1

Development of models for simple objects

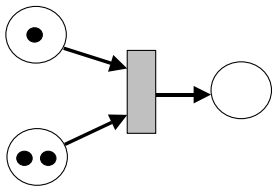
Goal of lesson: study basic elements of Petri nets, rules of nets' behavior, master the technique of models development for asynchronous processes and systems

Preparation: for execution of lesson it is required to know basic elements of Petri nets: place, transition, arc, token, and also rules of transition fire; representation of sequential, conveyer, alternative, parallel processes, consumable, recoverable and partially recoverable resources; dynamics of Petri net with multiply arcs.

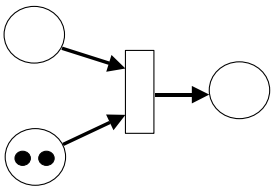
Task 1

To study the rules of Petri net transitions firing: for a given Petri net (fragment of net) point out the fireable transitions and marking, which will be obtained in the result of different transitions fire.

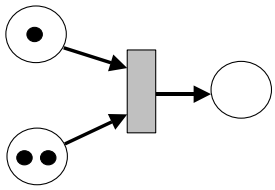
Example



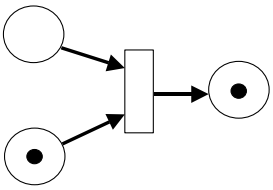
Transition is permitted



Transition isn't permitted



After firing

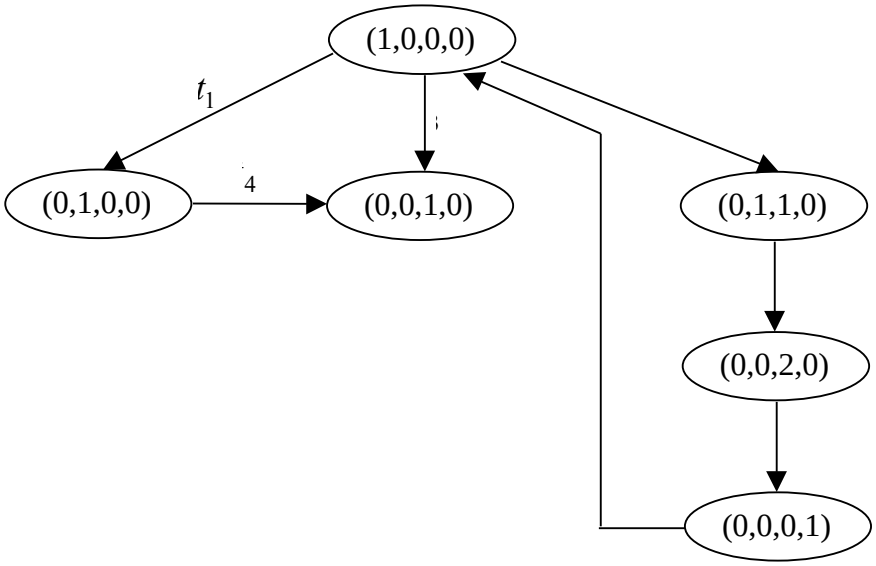


Task 2

Construct the graph of reachable markings for a given Petri net.

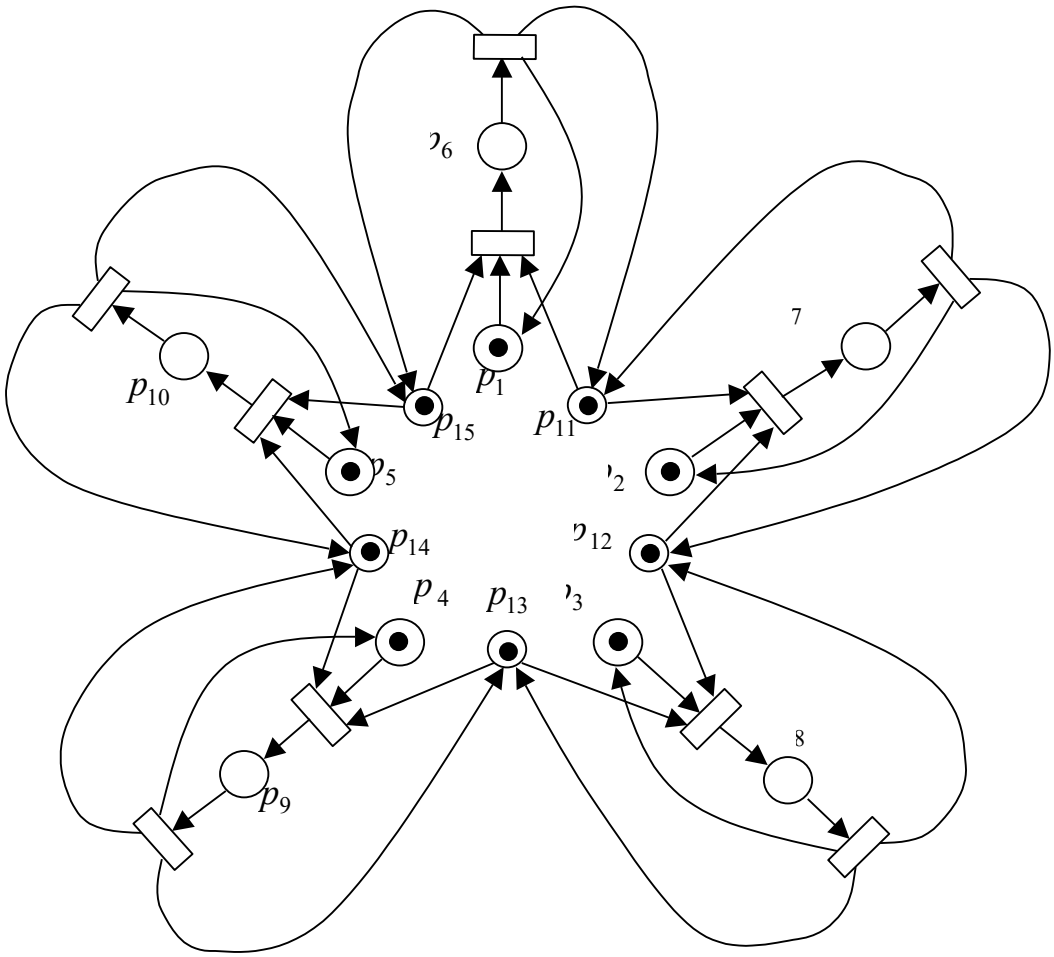
Example

Net 28, Appendix 1.



Task 3
Construct the model of a given object

Example
Task about dinning phosphors (Appendix 2. Task 1).



In the net constructed the places $p_1 - p_5$ model the thinking philosophers and places $p_6 - p_{10}$ model thinking philosophers.

Variants of tasks
Appendices 1,2.

Test questions
1. List the basic elements of Petri net.
2. Point out the basic areas of Petri nets application.
3. Formulate the firability conditions for Petri net transition.
4. What the firing of Petri net transition consists in?
5. Describe briefly the process of Petri net dynamics.
6. Point out the means of Petri net dynamics representation.
7. What is the graph of reachable markings of Petri net?
8. What Petri nets are named by ordinary?

Laboratory lesson 1

Simulation of Petri net dynamics

Goal of lesson: study the technology of graphical input and editing of Petri nets in the environment of simulation system, master the simplest manner of models investigation with help of simulation of their dynamics in stepwise and automatic mode.

Preparation: for execution of lesson it is required to know the basic elements of Petri nets, representation of various types of processes and resources relations, facilities of computer simulation system concerned with input and editing of nets and imitation of their dynamics.

Order of lesson's execution
1. Run simulation system.
2. Load sequentially nets phil1, phil2, phil3.
3. Run firable transitions watching the results of firing.
4. Run net in automatic mode.
5. Input and investigate early studied nets (on the order of teacher).
6. Construct the model of pointed object.
7. Input and investigate model with the help of imitation of its dynamics.

Guidelines to lesson's execution
Use simulation system Tina (Appendix 3.1). At first load early developed nets. Run "tools - stepper" and imitate the dynamics of nets in stepwise mode studying the rules of transition firability and firing.

At input of nets provide vivid allocation of elements. At investigation of nets with the help of simulation of dynamics pay attention to the question is it possible to fire all the transitions, is there unlimited accumulation of tokens into places, are there sequences of transitions firing, which may be repeated.

Variants of tasks
Appendices 1,2.

Test questions
1. Formulate the condition of Petri net transition firebility.
2. What the firing of Petri net transition consists in?
3. Briefly describe the process of Petri net behavior.
4. Point out forms of vivid representation of Petri net dynamics.
5. What is the graph of reachable markings of Petri net?
6. List the basic function of simulation system Tina.
7. What modes of Petri net dynamics imitation provide the simulation system?

Practical lesson 2

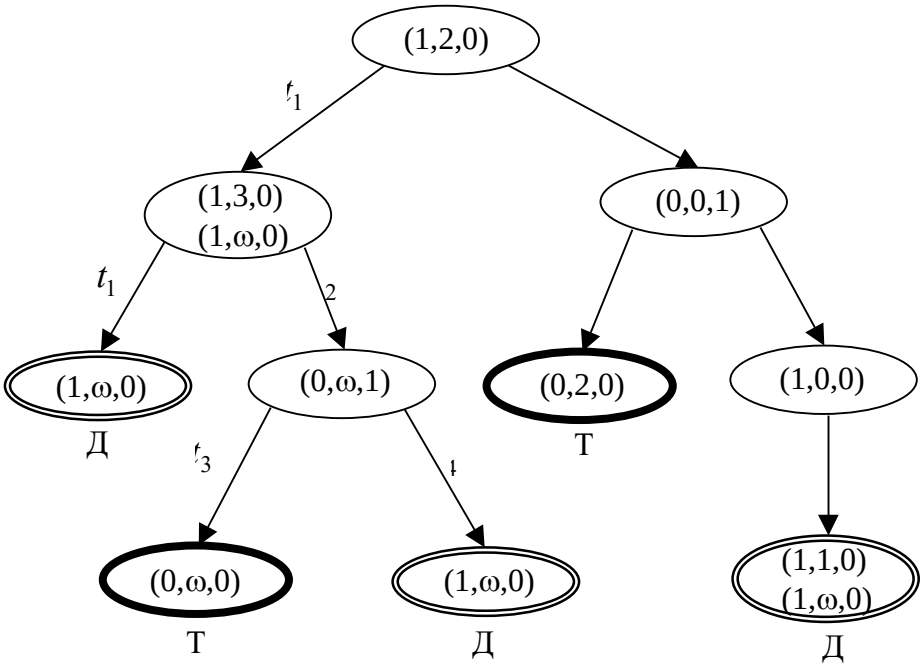
Analysis of Petri net properties using coverable tree

Goal of lesson: study the algorithm of coverable marking tree construction for a given Petri net and also the technique of investigation the basic properties of Petri nets using coverable tree.

Preparation: for execution of lesson it is required to know basic properties of Petri nets: boundeness, safeness, reachability, liveness, consistence's and also properties of pseudomarkings and the algorithm of coverability tree construction.

Task 1
Construct the coverable marking tree for a given Petri net

Example
Net 29, Appendix 1.



Task 2
Using coverable markings tree determine is a given net bounded, safe, consistent, live.

Example
- the net is unbounded as the tree contains symbols ω;
- the net is potentially live as the tree contains arcs labeled by the symbols of each transition;
- the net is not live as the tree contains terminal (dead-lock) nodes;
- the net is not persistent as firing of transition t_3 in the marking $(0,0,1)$ makes impossible the subsequent firing of the firable in this marking transition t_4.

Task 3
With the help of coverable markings tree determine the reachability of pointed markings in Petri net.

Example
- the marking (0,7,1) is reachable in net; really, the tree contains the coverable marking (0,ω,1) on which the sequence of transitions' firing $t_1^7 t_2$ may be constructed leading the net into pointed marking;
- the marking (0,1,7) is unreachable in net so the tree does not contains the coverable marking for it.

Variants of tasks
Appendix 1.

Test questions
1. Point out the peculiarities of pseudomarkings construction for Petri net.
2. List the basic steps of the algorithm for coverability tree of Petri net construction.
3. What types of vertices contains the tree of coverable markings?
4. In what case the concrete marking of place is changed by symbol ω?
5. What properties of Petri nets may be determined using the coverable markings tree?
6. What is the way to determine the liveness of net with the help of the coverability tree?
7. What is the way to determine the reachability of marking with the help of coverability tree?
8. What size may possess the tree of coverable markings?

Laboratory lesson 2

Investigation of boundness of Petri net models

Goal of lesson: study the technique of boundeness of Petri net models investigation with the help of simulation system Tina

Preparation: for execution of the lesson it is required to know basic properties of Petri nets, properties of pseudomarkings and also functions of simulation system.

Order of lesson's execution

1. Investigate the boundness of the nets studied at previous lessons (on the order of teacher).
2. Construct automatically the tree of coverable markings.
3. Determine the boundness of places and the whole net.
4. Make yourself sure in presence (absence) of properties listed with the help of imitation of Petri net dynamics.
5. Construct and investigate Petri net model for pointed object; fulfill the choice of capacities of storage.

Guidelines to lesson's execution

Use simulation system Tina (Appendix 3.1). You may input Petri net either in graphical editor of Petri net or construct it automatically on the information about connection among elements. Graphical input is implemented via mouse buttons in the combination with "Ctrl", "Alt", "Shift" keys. The following example describes the structure of Petri net 29, Appendix 1:

```
net n29
tr t1 p1 -> p1 p2
tr t2 p1 p2*2 -> p3
tr t3 p3 -> p2*2
tr t4 p3 -> p1
pl p1 (1)
pl p2 (2)
pl p3 (0)
```

Type of file is .net; exact description of its format is contained in file net/FORMATS of Tina. Just load this file and run function "draw". Then you may correct the picture of net moving elements. For construction of coverability tree run "tools - reachabilyty" and choose verbose format of coverability tree. For pointed net we obtain the following result:

```
CLASSES:
0 : p1 p2*2
1 : p1 p2*w
2 : p2*w p3
3 : p2*w
4 : p3
5 : p2*2
6 : p1
REACHABILITY GRAPH:
0 -> t1/1, t2/4
1 -> t1/1, t2/2
2 -> t3/3, t4/1
3 ->
4 -> t3/5, t4/6
5 ->
6 -> t1/1
```

Since markings contain symbol ω, net is unbounded.

Variants of tasks
Appendices 1,2.

Test questions
1. List basic properties of Petri nets?
2. What nets are named bounded and safe?
3. What the property of safeness consists in?
4. What the property of Petri net liveness consists in?
5. What marking of Petri net is named deadlock?
6. Point the peculiarities of Petri net pseudomarkings construction.
7. Formulate the basic steps of the algorithm for Petri net coverability tree construction.
8. What properties of Petri net may be determined using the tree of coverable markings?

Practical lesson 3

Analysis of Petri nets properties with the help of fundamental equation and linear invariants

Goal of lesson: master the technique of fundamental equation resolution with the help of unimodular transformations and calculation of invariants with the help of Toudic method and also usage of invariants and fundamental equation's solutions for the analysis of Petri nets properties.

Preparation: for execution of lesson it is required to know basic properties of Petri nets, Toudic's method, method of reduction of integer matrix to Smith normal form with the help of unimodular transformations.

Task 2
Investigate the reachability of a given marking with the help of fundamental equation of Petri net.

Example
Net 30, Appendix 1.
Let $\mu = (2,0,1,1,4,0)$.
Let's solve the fundamental equation of the net: $\overline{\mu} = \overline{\mu}_0 + C \cdot \overline{\sigma}$. The general integer solution has the form: $\overline{\sigma} = \begin{bmatrix} 2 \\ 0 \\ 1 \\ 1 \\ 4 \\ 0 \end{bmatrix} + k_1 \cdot \begin{bmatrix} 1 \\ 1 \\ 0 \\ 0 \\ 0 \\ 1 \end{bmatrix} + k_2 \cdot \begin{bmatrix} 2 \\ 0 \\ 2 \\ 1 \\ 6 \\ 1 \end{bmatrix}$, where $k_1, k_2 -$ an arbitrary nonnegative integer numbers. On the minimal solution we may construct the following permitted sequence of transitions firing $t_1 t_5^3 t_3 t_4 t_1 t_5$. Therefore, the marking $\overline{\mu}$ is reachable in the Petri net.

Task 2
Investigate boundness and safeness of net with the help of p-invariants.

Example
Let's solve the equation $\bar{x} \cdot C = 0$. Invariants of places have the form $\bar{x} = k \cdot (6,1,3,3,1)$, where k is an arbitrary natural number.

The net is p-invariant as, for instance, the invariant $(6,1,3,3,1)$ contains only nonzero components for all the places. Consequently, the net is conservative and bounded. Weighted sum of tokens is equal to $\bar{x} \cdot \mu_0 = 9$. Marking of places does not exceed (2,9,3,3,9).

Task 3
Investigate liveness and consistence of processes with the help of t-invariants.

Example
Let's solve the equation $C \cdot \bar{y} = 0$. Invariants of transitions have the form $\bar{y} = k_1 \cdot \begin{bmatrix} 1 \\ 1 \\ 0 \\ 0 \\ 0 \\ 1 \end{bmatrix} + k_2 \cdot \begin{bmatrix} 2 \\ 0 \\ 2 \\ 1 \\ 6 \\ 1 \end{bmatrix}$, where k_1, k_2 – an arbitrary nonnegative integer numbers.

The net is t-invariant as, for instance, the invariant $\begin{bmatrix} 3 \\ 1 \\ 2 \\ 1 \\ 6 \\ 2 \end{bmatrix}$ contains only natural components. To the pointed invariants corresponds the consistent sequence of transitions' firing $\sigma = t_1 t_2 t_6 t_1 t_3 t_4 t_1 t_5^6 t_3 t_6$, which contains all the transitions of the net; consequently, the net is potentially live.

Variants of tasks
Appendix 1.

Test questions
1. Describe the structure of the fundamental equation of Petri net.
2. What is the place invariant of Petri net?
3. What is the transition invariant of Petri net?
4. Describe the properties of invariant Petri nets.
5. What are the basic steps of Toudic method?

6. What transformations of matrices are named unimodular?
7. What is the matrix in Smith normal form?
8. Characterize the complexity of linear systems' solution in integer and nonnegative integer numbers.

Laboratory lesson 3

Search of deadlocks in model-based systems

Goal of lesson: master the technique of deadlocks' search in asynchronous parallel processes with the help Petri net simulation system

Preparation: for execution of the lesson it is required to know basic properties of Petri nets, especially, liveness and deadlock-freeness, master the methods of net invariants, be able to apply invariants to investigate the liveness.

Order of lesson's execution

1. Investigate the presence of deadlocks in previously studied nets.
2. Construct net into simulation system.
3. Find invariants of the net.
4. Conclude about basic properties of the net and about of deadlocks presence.
5. Try to find a deadlock with the help of imitation of net's dynamics; write the corresponding sequence of transitions' firing.
6. Construct the model of pointed object.
7. Investigate model concerned with presence of deadlocks.

Guidelines to lesson's execution

Use simulation system Tina (Appendix 3.1). Input net either in graphical editor or in text file and construct net automatically.

Execute structural analysis of the net: "tools - structural analysis". For the net 30, Appendix 1 we obtain the following result:

```
P-SEMI-FLOWS GENERATING SET ----------------------------------
invariant
p1*6 p2 p3*3 p4*3 p5
T-SEMI-FLOWS GENERATING SET ----------------------------------
consistent
t1*2 t3*2 t4 t5*6 t6
t1 t2 t6
```

Therefore, the net is invariant and consistent. Further imitation of net's dynamics and construction of graph of reachable markings allows the conclusion that net does not contains deadlocks. Notice that direct construction of graph of reachable marking is possible only for bounded nets and moreover for non-huge nets.

Variants of tasks

Appendix 1,2.

Test questions

1. List the basic properties of Petri nets?
2. What the property of Petr net liveness consists in?
3. What marking is named by deadlock?
4. What is the place invariant of Petri net?
5. What is the transitions invariant of Petri net?
6. Characterize properties of invariant Petri nets.
7. What is the way to apply Petri net invariants for the search of deadlocks?

Practical lesson 4

Analysis of Petri nets properties with the help of reduction

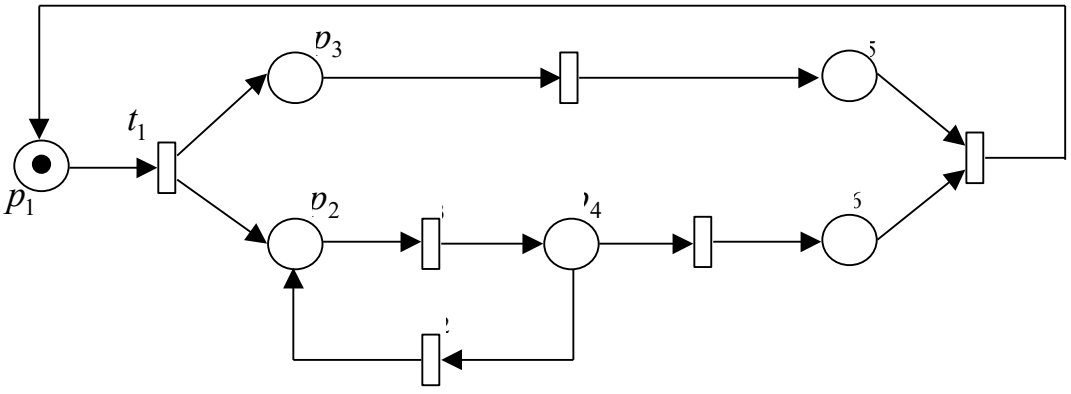
Goal of lesson: master experience of Petri net dimension decreasing (reduction) with the help of transformations saving basic properties of nets.

Preparation: for execution of the lesson it is required to know basic properties of Petri nets, set of reduction rules and conditions of their application for decreasing of Petri net dimension.

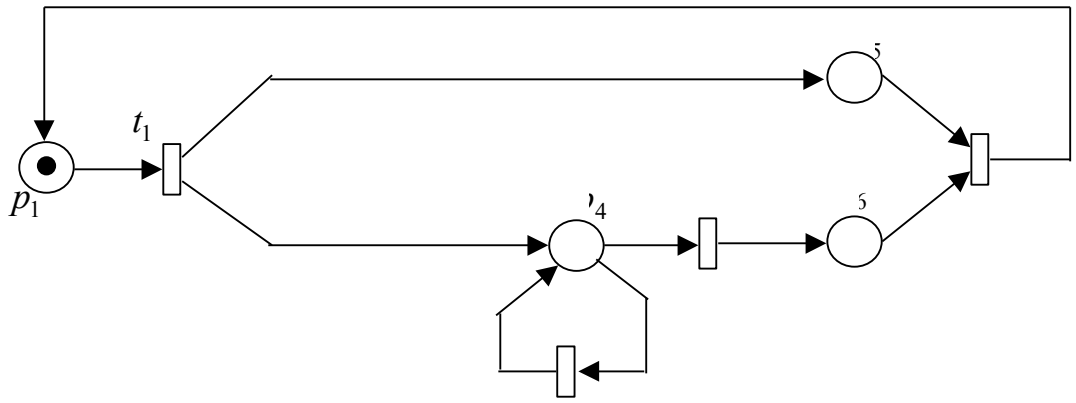
Mission

1. Execute a few separate transformations for each of the reduction rules.
2. Implement the reduction of pointed nets.
3. Investigate properties of nets before and after the reduction; make sure in preservation of properties.
4. Implement the reduction with the help of algebraic transformations of state equation.
5. Investigate real-life objects' models with the help of reduction.

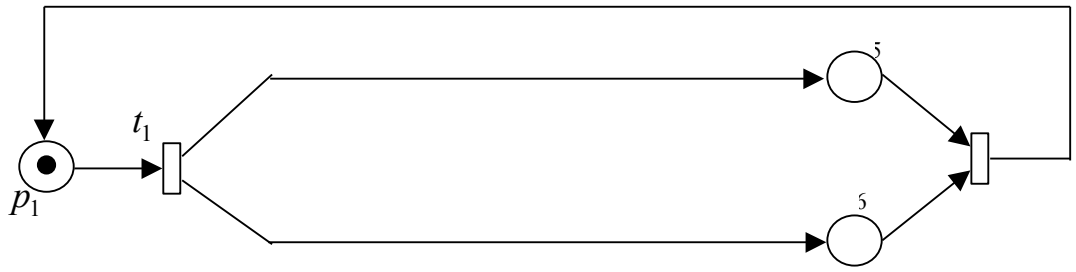
Example



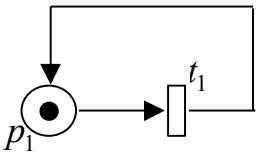
$R_1(p_3)R_1(p_2)$



$R_3(t_2)R_1(p_4)$



$R_2(p_5)R_1(p_6)$



Obtained net is live, bounded, safe; consequently, the source net possesses pointed properties.

Variants of tasks
Appendix 1.

Test questions
1. What the reduction of Petri nets consists in?
2. Point the basic rules of Petri net reduction.
3. List the basic properties of Petri nets.
4. What nets are named by bounded and safe?
5. What the property of Petri net safeness consists in?
6. What the property of Petri net liveness consists in?
7. What marking of Petri net is named by deadlock?

Practical lesson 5

Construction the generating family of functional subnets

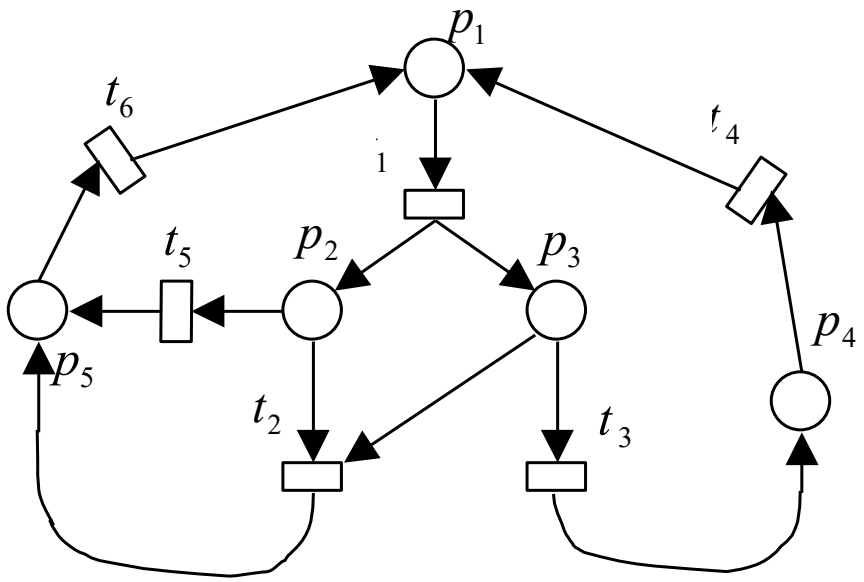
Goal of lesson: master the methods of a given Petri net decomposition into functional subnets.

Preparation: for execution of the lesson it is required to know definitions of functional subnet and minimal functional subnet as well as algorithmic and analytic methods of decomposition.

Mission

1. Construct the system of logical equations defining functional subnet. Find its solutions.
2. Implement the construction of generating family of functional subnets with the help of ad-hoc algorithm of decomposition.
3. Construct a few nets, which are the sum of minimal subnets.
4. Construct the net of minimal functional subnets; construct directed and undirected graph of decomposition.
5. Decompose net by Deborah plug-in of Tina (Appendix 3.1).

Example



System, which defines functional subnet, has the following form:

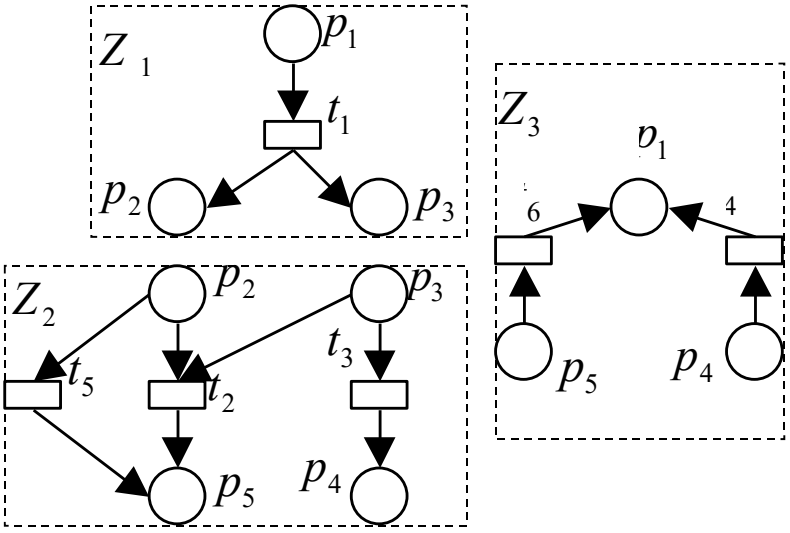
$$\begin{cases} \tau_1 \equiv (\tau_6\tau_4\tau_1 \vee \bar{\tau}_6\bar{\tau}_4\tau_1)(\tau_1\tau_5\tau_2 \vee \tau_1\bar{\tau}_5\bar{\tau}_2)(\tau_1\tau_2\tau_3 \vee \tau_1\bar{\tau}_2\bar{\tau}_3) \\ \tau_2 \equiv (\tau_1\tau_5\tau_2 \vee \bar{\tau}_1\tau_5\tau_2)(\tau_1\tau_2\tau_3 \vee \bar{\tau}_1\tau_2\tau_3)(\tau_5\tau_2\tau_6 \vee \tau_5\tau_2\bar{\tau}_6) \\ \tau_3 \equiv (\tau_1\tau_2\tau_3 \vee \bar{\tau}_1\tau_2\tau_3)(\tau_3\tau_4 \vee \tau_3\bar{\tau}_4) \\ \tau_4 \equiv (\tau_3\tau_4 \vee \bar{\tau}_3\tau_4)(\tau_1\tau_4\tau_6 \vee \bar{\tau}_1\tau_4\tau_6) \\ \tau_5 \equiv (\tau_1\tau_5\tau_2 \vee \bar{\tau}_1\tau_5\tau_2)(\tau_5\tau_2\tau_6 \vee \tau_5\tau_2\bar{\tau}_6) \\ \tau_6 \equiv (\tau_5\tau_2\tau_6 \vee \bar{\tau}_5\bar{\tau}_2\tau_6)(\tau_1\tau_4\tau_6 \vee \bar{\tau}_1\tau_4\tau_6) \end{cases}$$

Construct the solution of the system:

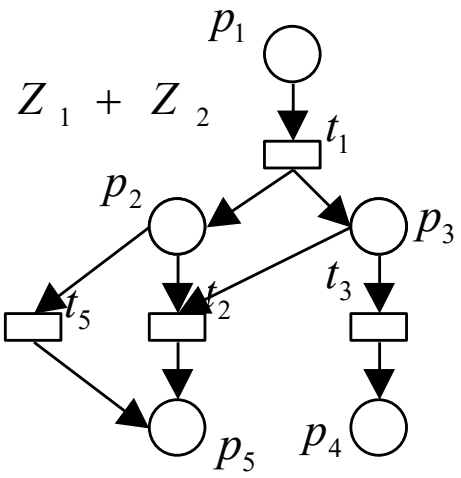
$$\bar{\tau}_1\bar{\tau}_2\bar{\tau}_3\bar{\tau}_4\bar{\tau}_5\bar{\tau}_6 \vee \tau_1\bar{\tau}_2\bar{\tau}_3\bar{\tau}_4\bar{\tau}_5\bar{\tau}_6 \vee \bar{\tau}_1\bar{\tau}_2\bar{\tau}_3\tau_4\bar{\tau}_5\tau_6 \vee \bar{\tau}_1\tau_2\tau_3\bar{\tau}_4\tau_5\bar{\tau}_6 \vee$$

$$\tau_1\bar{\tau}_2\bar{\tau}_3\tau_4\bar{\tau}_5\tau_6 \vee \tau_1\tau_2\tau_3\bar{\tau}_4\tau_5\bar{\tau}_6 \vee \bar{\tau}_1\tau_2\tau_3\tau_4\tau_5\tau_6 \vee \tau_1\tau_2\tau_3\tau_4\tau_5\tau_6$$

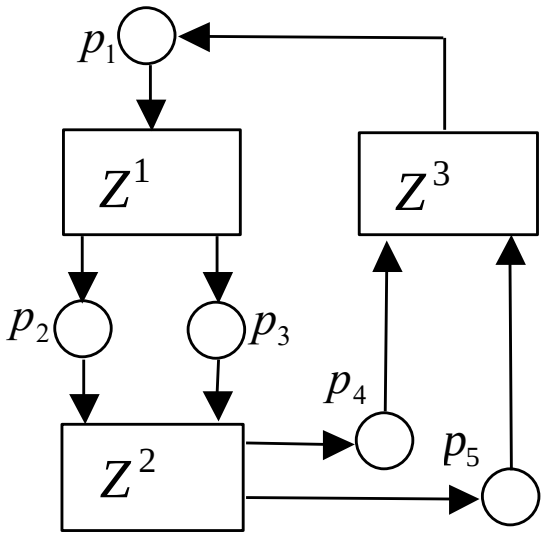
Notice that, the first term corresponds to empty subnet, next three terms correspond to minimal subnets:



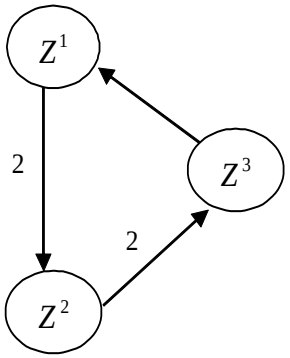
remaining terms describe sums of minimal subnets, for instance, sixth term describes the following subnet:

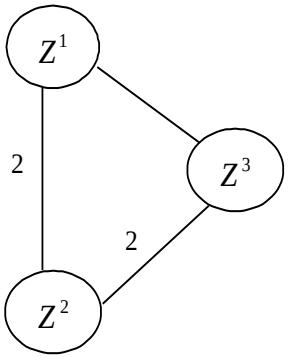


for representation of decomposition the net of functional subnets:



as well as directed and undirected graphs may be used:





Variants of tasks
Appendix 1,2.

Test questions
1. What Petri net is named by functional?
2. What is the functional subnet of a given Petri net?
3. May a minimal functional subnet to contain another functional subnets?
4. What functional subnet is defined by?
5. What is the generating family of functional subnets?
6. What is the way to create the system of equations, defining functional subnets?
7. Point the basic steps of decomposition algorithm.
8. What is the complexity of decomposition algorithm?

Laboratory lesson 4

Construction and investigation of telecommunication protocols' models

Goal of lesson: master skill of telecommunication protocols verification

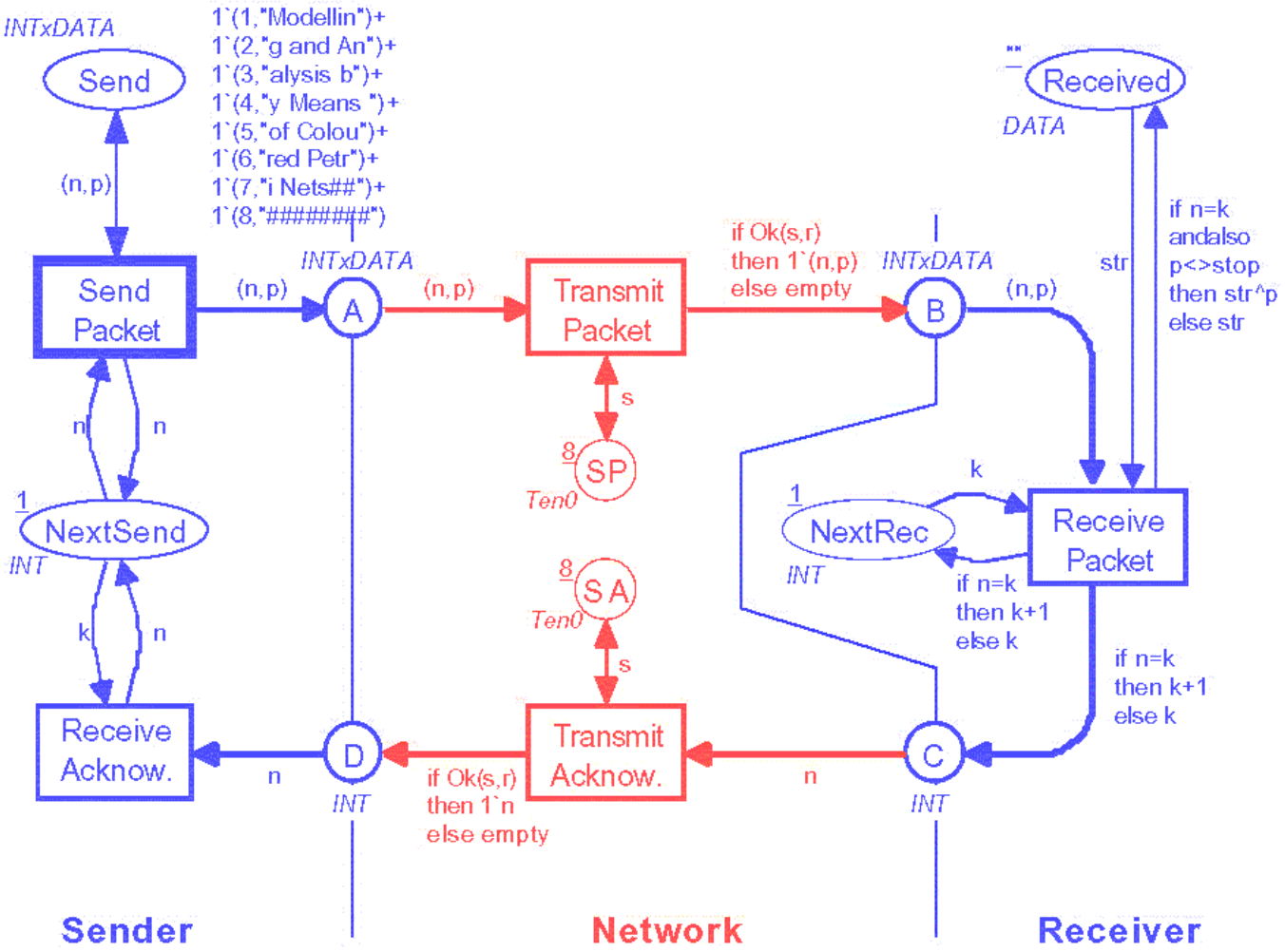
Preparation: for execution of the lesson it is required to know basic classes of Petri nets, possess the knowledge about telecommunication protocols, master skill of CPN models construction, know the exact specifications of protocols.

Order of lesson's execution
1. Study the training model of protocol.
2. Construct the colored Petri net into environment of simulation system.
3. Execute the imitation of net's dynamics.
4. Determine the boundness and liveness of model.
5. Construct the model of a given protocol.
6. Input model into CPN Tools.
7. Execute the verification of protocol with the help of imitation and state space methods.

Guidelines to lesson's execution
For execution of the lesson we use CPN Tools (Appendix 3.2) and Tina (Appendix 3.2) simulation systems. CPN Tools allows the construction detailed models, which reflect all the peculiarities of source standard specifications but for investigation of models only

imitation of net's dynamics and state space methods may be applied. Tina provides formal methods such as coverability tree, linear invarians, and decomposition. But we have to represent the model with only low-level Petri net which is concerned with abstraction from details of standard specifications.

Telecommunication protocol is a set of rules, which defines exactly the order of systems' interaction with the goal of information exchange. Let's consider the simplest protocol of messages transmission with acknowledgements from the standard set of CPN Tools examples:




```
color INT = int;
color DATA = string;
color INTxDATA = product INT * DATA;
var n, k : INT;
var p,str : DATA;
val stop = "########";

color Ten0 = int with 0..10;
color Ten1 = int with 1..10;
var s : Ten0;  var r : Ten1;
fun Ok(s:Ten0,r:Ten1) = (r<=s);
```

Transmitting packets are situated in place **Send** and consist of number of packet and data, which are represented by text string. Place **NextSend** defines the number of current packet. In the places **Received** the assembling of data is executed out of correctly received packets: text strings are concatenated. Until acknowledgement the retransmission of packed is executed. Transmission of packages is terminated with processing of special terminal string **stop**. Network is simulated by the pair of transitions: **TransmitPacket** for transmission of messages and **TransmitAcknow** for transmission of acknowledgements. It's possible the lost of messages and acknowledgements during transmission. The probability of successful transmission is given by marking of places **SP** and **SA**.

At execution of the lesson it is required to supply the protocol with timed characteristics, add a delay before retransmission of packet (timeout), provide the transmission of packets with different order and following assembling of them.

For investigation of model with formal methods we recommend a simple transformation of model into low-level Petri net with abstraction from characteristics of net's elements.

Variants of tasks

Variant	Time of packets' transmission	Time of acknowledgements' transmission	Timeout	Probability of successful package transmission	Probability of successful acknowledgement transmission
1	4000	1000	6000	0.9	0.9
2	5000	1000	9000	0.8	0.9
3	2000	1000	4000	0.8	0.8
4	3000	2000	7000	0.7	0.8
5	1000	1000	4000	0.7	0.9
6	5000	2000	8000	0.6	0.7
7	4000	1000	7000	0.8	0.9
8	3000	2000	8000	0.9	0.9
9	2000	1000	5000	0.6	0.6
10	1000	1000	8000	0.7	0.6
11	5000	4000	9000	0.8	0.9
12	4000	3000	9000	0.9	0.7
13	3000	1000	6000	0.6	0.9
14	2000	1000	5000	0.7	0.7
15	1000	1000	3000	0.8	0.9
16	4000	1000	8000	0.6	0.8

Test questions

1. What is telecommunication protocol?
2. What telecommunication protocols do you know?
3. What is the goal of telecommunication protocols investigation?
4. What the protocols verification consists in?
5. What properties have to possess Petri net modeling the correct protocol?
6. What methods are applied for reliable transmission of information through unreliable channels?

7. List and describe the tools of system CPN Tools?
8. What is the way to represent times in CPN Tools?

Laboratory lesson 5

Construction and investigation of switched networks models

Goal of lesson: master the technique of switched networks investigation with the help of colored Petri nets.

Preparation: for execution of the lesson it is required to know basic classes of Petri nets and basis of switched networks technology, master habits of models construction, know organization of simulation system in details, be able to recognize the properties of net which have to possesses the model of correctly working device.

Order of lesson's execution

1. Draw the structural scheme of network for your variant and write its parameters.
2. Input in CPN Tools submodels of basic devices: switch, server, workstation.
3. Debug submodels of devices separately putting tokens into contact places and watching their traces.
4. Construct the model of switched network with a given structure; main page will unite the submodels of devices.
5. Debug the model in stepwise and automatic modes of simulation.
6. Input and debug the submodel of a measuring workstation.
7. Measure the response time putting measuring workstation into various segments of the network.
8. Make your conclusion about network simulated.

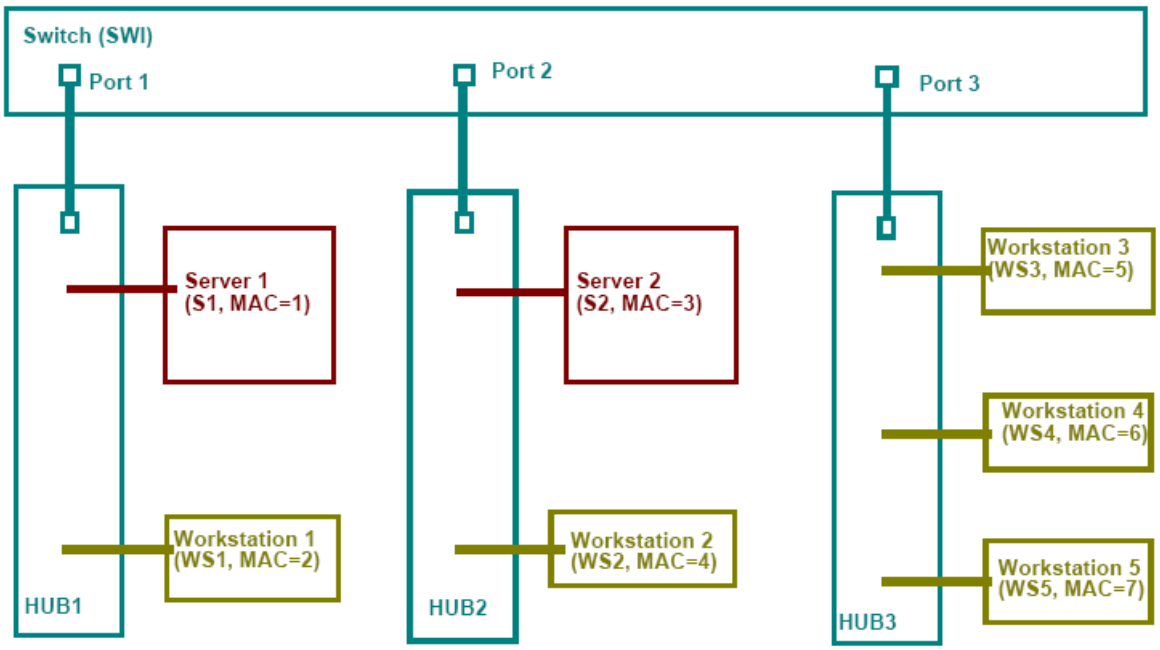
Guidelines to lesson's execution

For construction and investigation of models we use CPN Tools (Appendix 3.2). Lets consider an example of model construction for a given switched Ethernet LAN structure. Hubs are dumb passive equipment aimed only at the connection of devices as wires. The base element of the Local Area Network (LAN) Ethernet (IEEE 802.x) is a switch of frames:

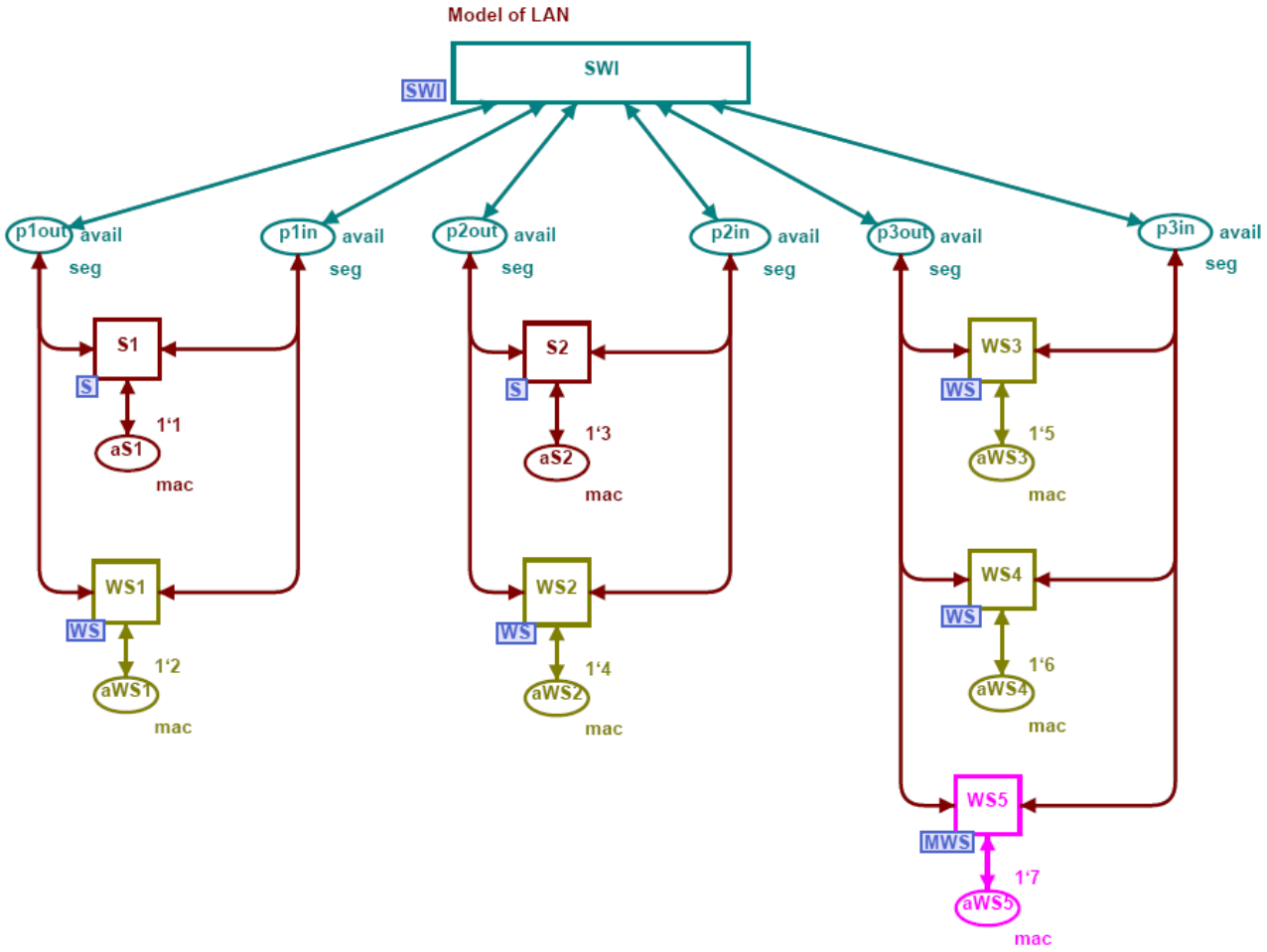
Logically a switch is constituted of a set of ports. LAN segment (for example, made up via hub) or terminal equipment such as workstation or server may be attached to each port. The task of a switch is the forwarding of incoming frame to the port that the target device is connected to. The usage of a switch allows for a decrease in quantity of collisions so each frame is transmitted only to the target port and results in an increased bandwidth. Moreover the quality of information protection rises with a reduction of ability to overhear traffic. As a rule, the Ethernet works in a full-duplex mode now, which allows simultaneous transmission in both directions. To determine the target port number for the incoming frame a static or dynamic switching table is used. This table contains the port number for each known Media Access Control (MAC) address. Static switching tables will be considered.

Consider the given scheme of LAN:

Scheme of sample switched LAN



Its elements are: switch (SWI), hubs (HUB), servers (S), workstations (WS). We model MAC addresses with integer numbers. Top page of the model has the following form:



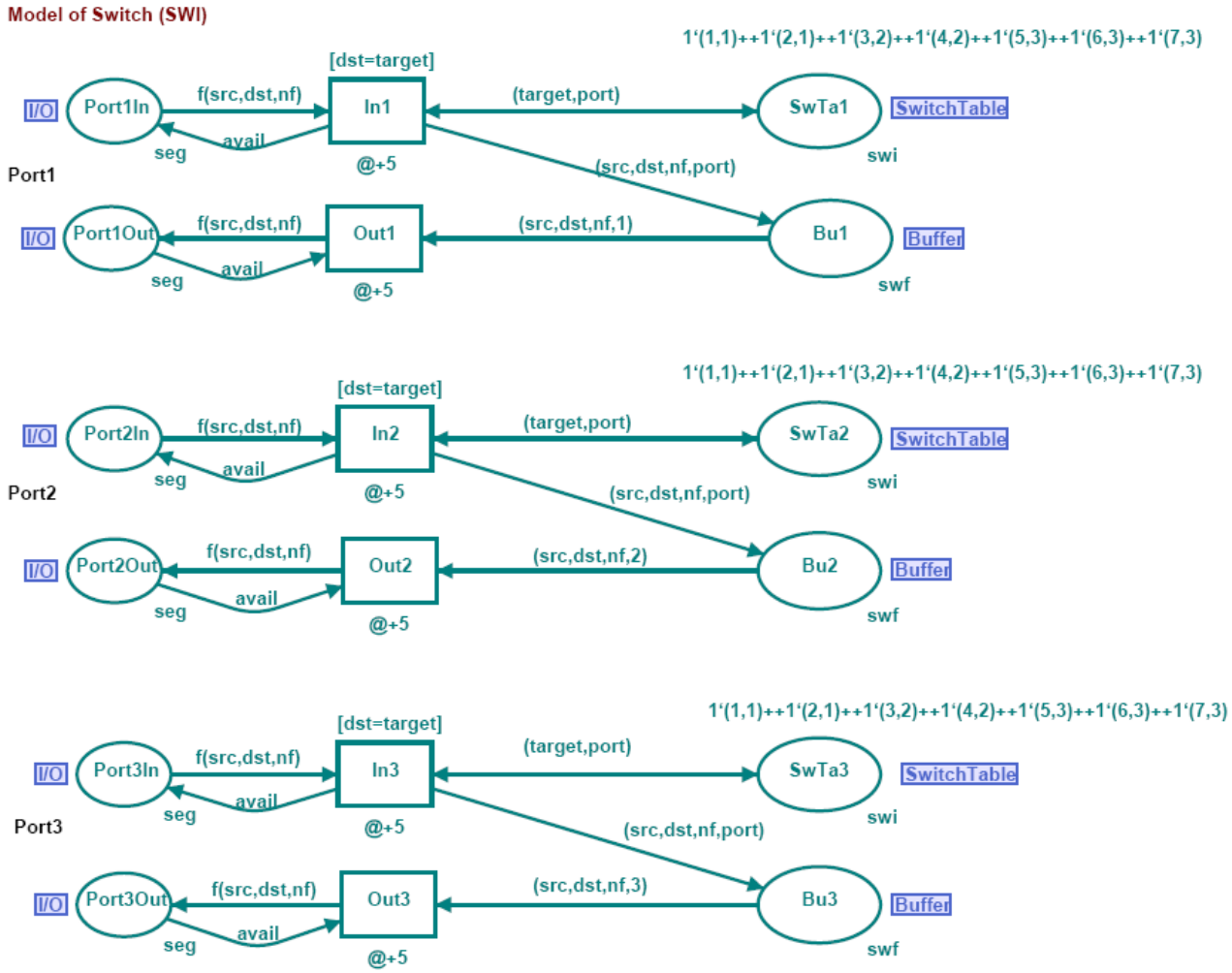
The elements of this model are sub models of: Switch (**SWI**), Server (**S**), Workstation (**WS**) and Measuring Workstation (**MWS**). Workstations **WS1-WS4** are the same type exactly **WS**, whereas workstation **WS5** is the type **MWS**. It implements the measuring of network response time. Servers **S1** and **S2** are the same type exactly **S**. Hubs are a passive equipment and have not an independent model representation. The function of hubs is modelled by common use of the corresponding places **p*in** and **p*out** by all the attached devices. Declarations of colors, variables and function are as the following:

```
color mac = INT timed;
color portnum = INT;
color nfrm = INT;
color sfrm = product nfrm * INT timed;
color frm = product mac * mac * nfrm timed;
color seg = union f:frm + avail timed;
color swi = product mac * portnum;
color swf = product mac * mac * nfrm * portnum timed;
color remsv = product mac * nfrm timed;
var src, dst, target: mac;
var port: portnum;
var nf, rnf: nfrm;
var t1, t2, s, q, r: INT;
color Delta = int with 1000..2000;
fun Delay() = Delta.ran();
color dex = int with 100..200;
fun Dexec() = dex.ran();
color dse = int with 10..20;
fun Dsend() = dse.ran();
color nse = int with 10..20;
fun Nsend() = nse.ran();
fun cT()=IntInf.toInt(!CPN'Time.model_time)
```

Each server and workstation has it's own MAC address represented in places **aS***, **aWS***. A switch has separate places for input (**p*in**) and output (**p*out**) frames for each

port. It represents the full-duplex mode of work. Bidirected arcs are used to model the carrier detection procedures. One of the arcs checks the state of the channel, while another implements the transmission. The Ethernet MAC address is modelled with integer number (colour **mac**). The frame is represented by a triple **frm**, which contains source (**src**) and destination (**dst**) addresses, and also a special field **nfrm** to enumerate the frames for the calculation of response time. We abstract of other fields of frame stipulated by standard of Ethernet. The colour **seg** represents unidirectional channel and may be either available for transmission (**avail**), or busy with transmission of a frame (**f.frm**). It is represented with a **union** type of colour. Notice that the descriptor **timed** is used for tokens, which take part in timed operations such as delays or timestamps.

Consider the submodel of switch:

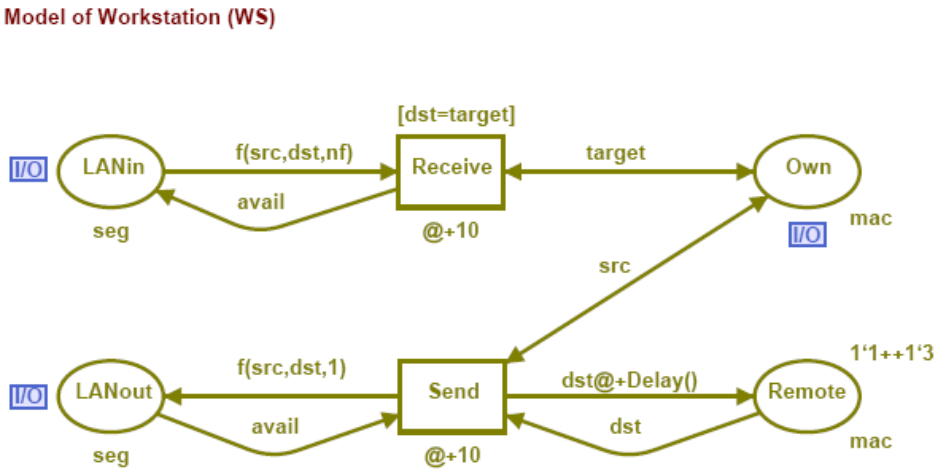


The colour **swi** represents records of switching table. It maps each known MAC address (**mac**) to the number of port (**nport**). The colour **swf** describes the switched frames, waiting for output buffer allocation. The field **portnum** stores the number of the target port. The places **Port*In** and **Port*Out** represent input and output buffers of the ports correspondingly. The fusion place **SwitchTable** models the switching table; each token in this place represents the record of the switching table. For instance, token **1`(4,2)** of the initial marking means that the host with MAC address 4 is attached to port 2. The fusion place **Buffer** corresponds to the switched frames' buffer. Notice that a fusion place (such as **SwitchTable** or **Buffer)** represents a set of places. The fusion place **SwitchTable** is represented with places **SwTa1, SwTa2, SwTa3**. The fusion place **Buffer** is represented with places **Bu1, Bu2, Bu3**. It allows the convenient modelling of switches with an arbitrary number of ports avoiding numerous cross lines.

The transitions **In*** model the processing of input frames. The frame is extracted from the input buffer only in cases where the switching table contains a record with an address that equals to the destination address of the frame (**dst=target**); during the frame displacement the target port number (**port**) is stored in the buffer. The transitions **Out*** model the displacement of switched frames to the output ports' buffers. The inscriptions of input arcs check the number of the port. The fixed time delays (@+5) are assigned to the operations of the switching and the writing of the frame to the output buffer.

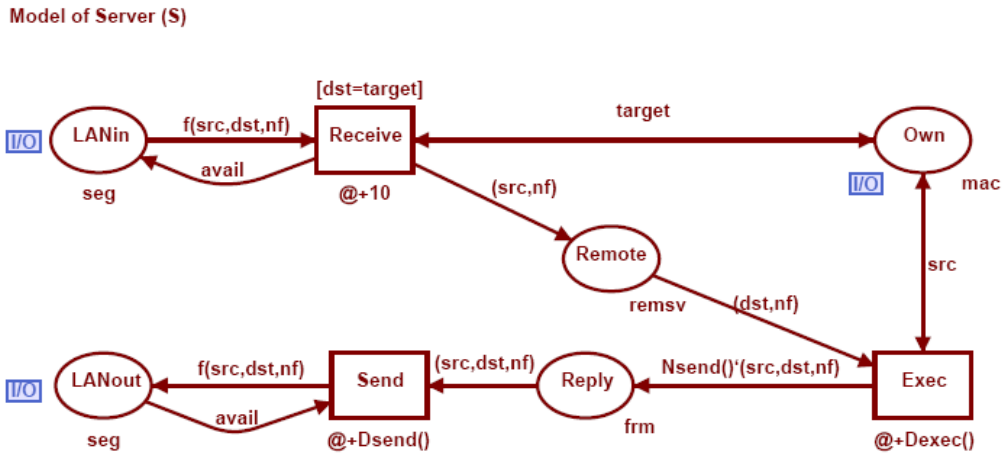
It is necessary to explain the CSMA procedures of LAN access in more detail. When a frame is extracted from the input buffer by transition **In*,** it is replaced with the label **avail**. The label **avail** indicates that the channel is free and available for transmission. Before the transition **Out*** sends a frame into a port, it analyses if the channel is available by checking the token **avail**.

Regarding the peculiarity of the traffic's form we shall separate workstations and servers. For an accepted degree of elaboration we consider periodically repeated requests of workstations to servers with random uniformly distributed delays. On reply to an accepted request a server sends a few packets to the address of the requested workstation. The number of packets sent and the time delays are uniformly distributed random values. Consider the submodel of workstation:



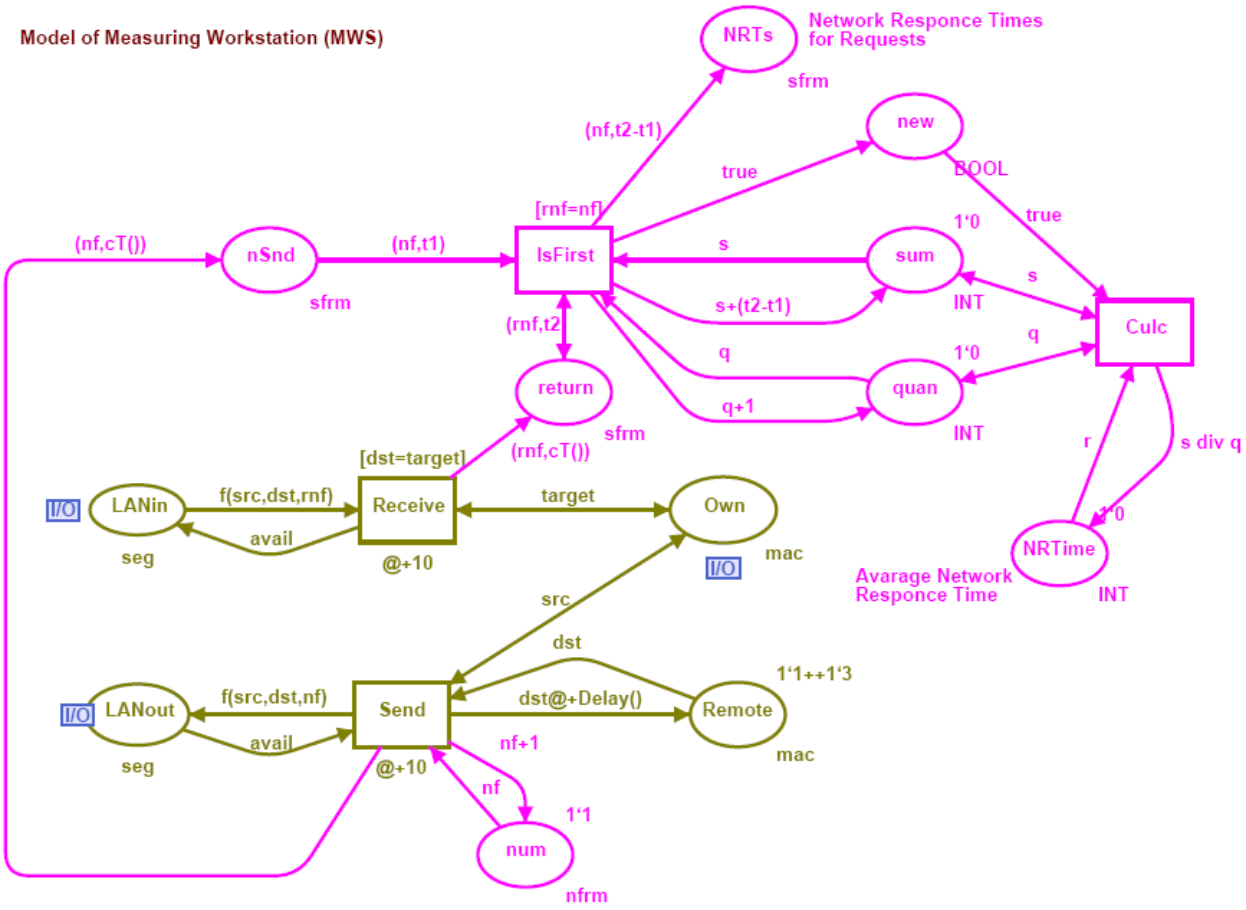
The places **LANin** and **LANout** model the input and output channels of the local area network correspondingly. The workstation listens to the network by means of transition **Receive** that receives frames with the destination address, which is equal to the own address of the workstation (**dst=target**) saved in the place **Own**. The processing of received frames is represented by the simple absorption of them. The workstation sends periodic requests to servers by means of transition **Send**. The servers' addresses are held in the place **Remote**. After the sending of a request the usage of the server's address is locked by the random time delay given by the function **Delay()**. The sending of the frame is implemented only if the LAN segment is free. It operates by checking place **LANout** for a token **avail**. In such a manner the workstation interacts with a few servers holding their addresses in the place **Remote**.

Submodel of server is as the following:



The listening of the network is similar to the model of the workstation but it is distinct in that the frame's source address is held in the place **Remote**. The transition **Exec** models the execution of the workstation's request by a server. As a result of the request execution the server generates a random number **Nsend()** of the response frames, which are held in the place **Reply**. Then these frames are transmitted into the network by the transition **Send**. Notice that the request number **nf** is stored in the place **Remote** also. It allows us to identify the response with the same number as the request.

To measure the network response time we use special measuring submodel of workstation:



Each frame of a workstation's request is enumerated with a unique number contained in the place **num**. The time, when the request was sent, is stored in the place **nSnd**. The function **cT()** calculates the current value of the model's time. The place **nSnd** stores a pair: the frame's number **nf** and the time of request **cT()**.

The place **return** stores the timestamps of all the returned frames. As the network response time we consider the interval of time between the sending of the request and

receiving the first frame of response. This value is stored in place **NRTs** for each responded request. The transition **IsFirst** determines the first frame of response. The inscription of the arc, connecting the transition **IsFirst** with the place **NRTs**, calculates the response time (t2-t1).

A residuary part of the measuring elements calculates the average response time. The places **sum** and **quant** accumulate the sum of response times and the quantity of accepted responses correspondingly. The arrival of a new response is sensed by the place **new** and initiates the recalculation of average response time with the transition **Culc**. The result is stored in the place **NRTime**.

Variants of tasks

Variant	Servers	Segments	Workstations			Workstation	Server	
			P1	P2	P3	Wt	St	Sq
1	2	3	2	2	3	2000-3000	50	12
2	3	2	2	2		3000-4000	60	10
3	2	3	3	2	1	2000-4000	40	15
4	3	2	3	2		4000-5000	20	20
5	2	3	3	3	2	3000-4000	30	17
6	2	2	4	2		3000-5000	70	8
7	2	3	2	4	2	2000-5000	10	19
8	2	2	3	4		5000-7000	50	5
9	2	3	2	2	3	3000-4000	20	11
10	3	2	2	2		4000-7000	30	14
11	2	3	3	2	1	2000-7000	40	10
12	3	2	3	2		3000-7000	10	18
13	2	3	3	3	2	5000-8000	30	13
14	2	2	4	2		7000-8000	60	4
15	2	3	2	4	2	4000-8000	20	10
16	2	2	3	4		3000-8000	40	7

P1, P2, P3 – number of workstations in segments (on the ports of switch);
Wt – interval of workstation's requests;
St – time of request execution by server;
Sq – number of frames in the answer of server.

Test questions

1. What elements switched network consists of.
2. Describe briefly the frame's structure for Ethernet.
3. What the functions of Ethernet switch consists in?
4. What are the peculiarities of traffics for workstations and servers?
5. Formulate the peculiarities of colored Petri nets.
6. What is the way to describe of tokens' type in colored Petri nets?
7. What additional characteristics have transitions and places of colored Petri net?
8. What additional characteristics have arcs of colored Petri net?
9. What is the goal of places' fusion in CPN Tools?

10.What is the way of hierarchical nets construction in CPN Tools?

Laboratory lesson 6

Construction and investigation of models for automatic control

Goal of lesson: master the skill of construction and debugging of automatic control systems represented by Petri nets.

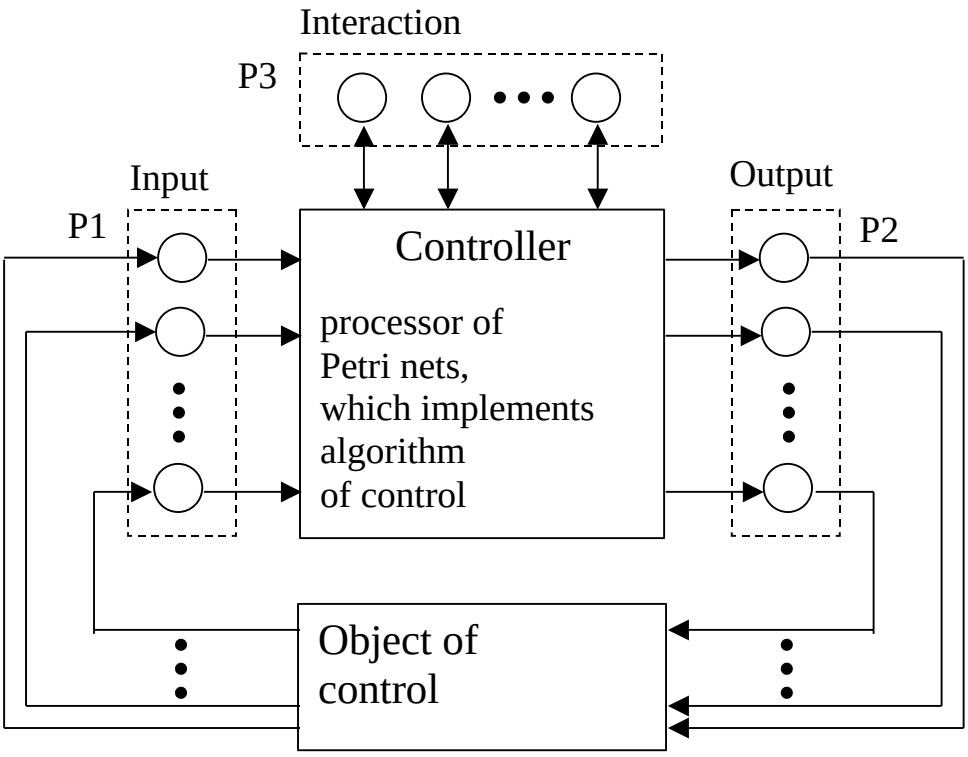
Preparation: for completion of the lesson, it is required to know basic classes of Petri nets, master the attainments of models' construction, study the description of automatic control system.

Order of lesson's execution

1. Study the model of automatic control of robot-manipulator.
2. Construct the corresponding Petri net into the environment of simulation system.
3. Execute an imitation of net's dynamics.
4. Determine the basic properties of net.
5. Construct the models of given control system and object of control.
6. Execute standalone debugging of control system.
7. Execute the complex debugging of control system together with the model of controlling object.
8. Investigate the properties of complex model.

Guidelines to lesson's completion

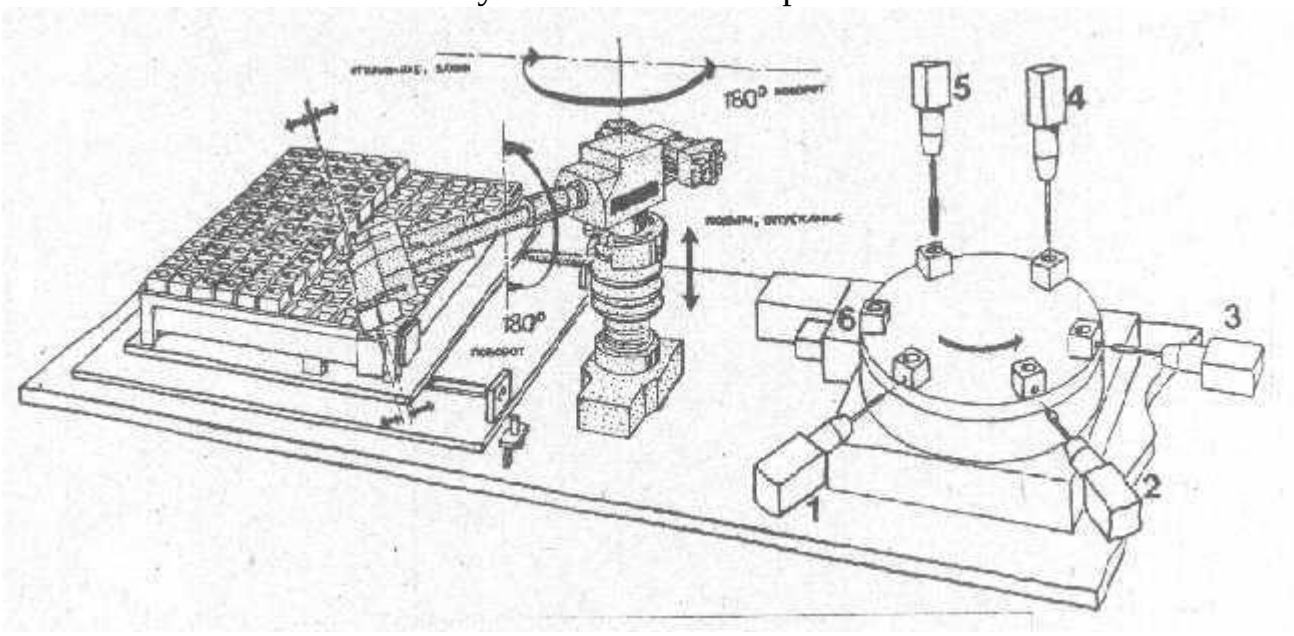
Let us consider the peculiarities of construction of automatic control systems based on a controller, which contains a processor of Petri nets (hardware of software):



Input places (P1) are directly connected with sensors of controlled object, output places (P2) are connected to executing devices (actuators); places P3 provides connection with operator and interaction with external control systems.

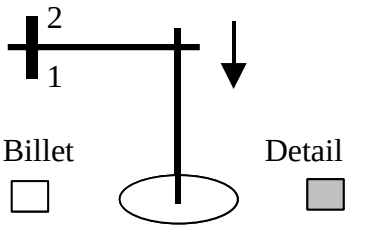
The significant achievement of describing technique is the possibility for representation both the algorithm of control and model of controlling object by Petri net. This allows the complex debugging of system.

Let's construct the control system of robot-manipulator:

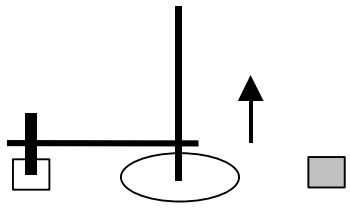


Robot interacts with storage of billets/details and rotary table on which the processing of billets is executed. Robot consists of pole, lever and head with two claws. Pole may rise up or lower down. There are two end sensors of upper and lover positions of the pole. Drive executes two commands: raise pole and lower pole. Lever rotates on 180 degrees and fixes two extreme positions over rotary table and over storage with the help of end sensors. Drive executes two commands: rotate left and rotate right. Head has two claws situates on opposite sides of it. It rotates on 180 degrees fixing in two extreme positions with the help of end sensors. Drive executes commands: rotate left and rotate right. Each of claws may either open or closed.

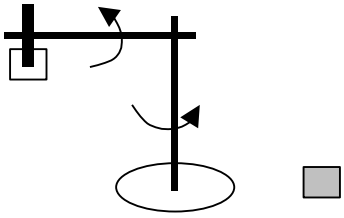
Initial position of parts is as follows: pole in the extreme upper position, lever in the extreme right position over storage, claw 1 directed down. Basic loop of robot's functioning consists in the following sequence of operations: lower pole down, grasp billet, raise pole up, rotate lever to the left and rotate head to the right, lower pole down, grasp detail by claw 2, raise pole up, rotate head to the right, lower pole down, open claw 1 for placement billet on rotary table, raise pole up, rotate lever to the right and rotate head to the left, lower pole down, open claw 2 for placement detail in storage, raise pole up, rotate head to the right. Let us represent described operations by the following sequence of pictograms:



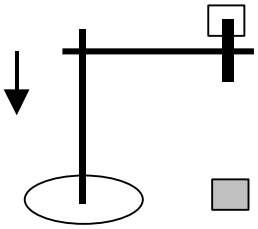
Lower pole down



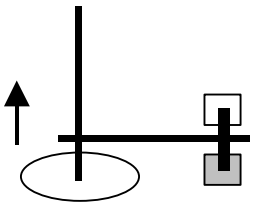
Grasp by claw 1
Raise pole up



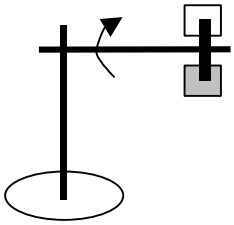
Rotate lever to the left
Rotate head to the left



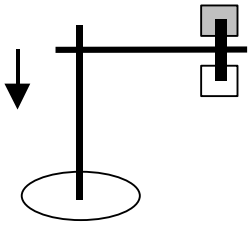
Lower pole down



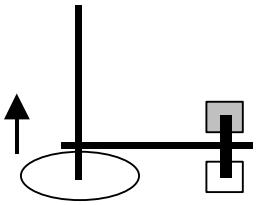
Grasp by claw 2
Raise pole up



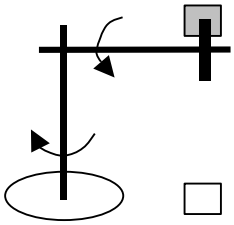
Rotate head to the right



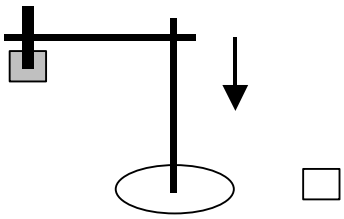
Lower pole down



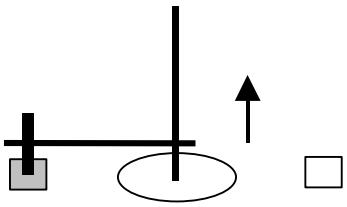
Open claw 1
Raise pole up



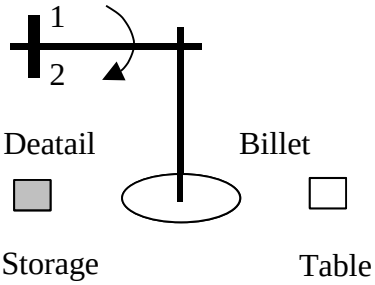
Rotate lever to the right
Rotate head to the left



Lower pole down



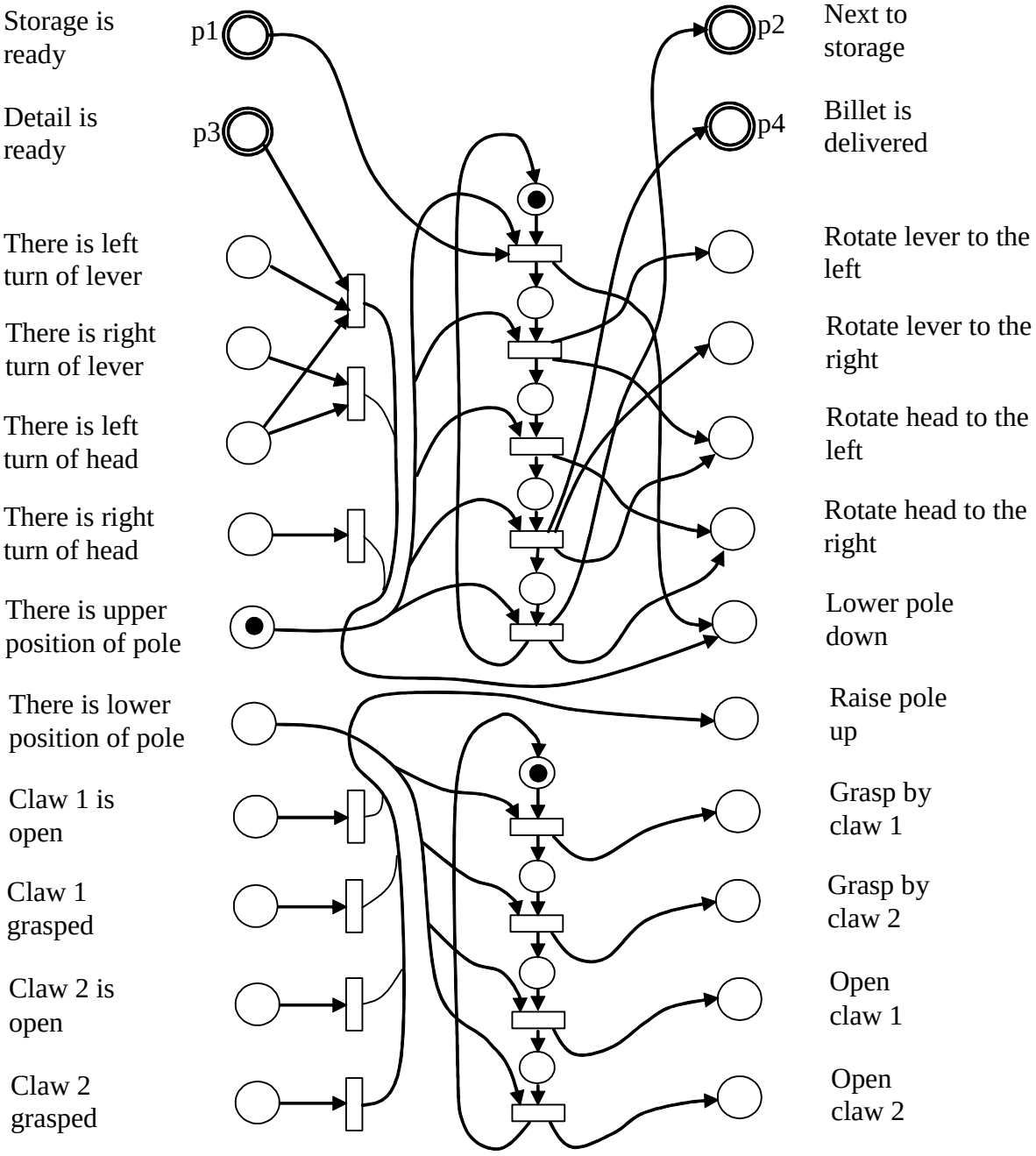
Open claw 2
Raise pole up



Rotate head to the right

Storage of billets/details constitutes rectangular box with cells for billets/details. Movements of storage in two directions provide the disposition of the required cell under the extreme right position of head. Rotary table provides the processing of detail in five positions and also an additional sixth position for loading/unloading under the extreme left position of head.

Let's represent the algorithm of robot's control:



Places p1, p2 are aimed to interaction with storage and places p3, p4 for interaction with rotary table. In the simplest case the model of the object may consists of a set of separate transitions representing the correct execution of command and firing of sensor. For instance, for pole it consists of transition connecting output place "lower pole down" with input place "there is lower position of pole".

It's required to input constructed algorithm into simulation system Tina (Appendix 3.2), debug standalone and together with the model of the controllable object.

Construct the models of control for storage and rotary table. Debug them standalone and together with the model of robot.

Investigate the properties of Petri net models constructed; prove the correctness of control algorithms with the help of system Tina (Appendix 3.2).

Variants of tasks

Variant	Size of storage		Size of rotary table
	Width	Length	
1 17	2	4	3
2 18	3	4	4
3 19	4	4	5
4 20	2	3	6
5	2	4	3
6	3	4	4
7	4	4	5
8	2	3	6
9	2	4	3
10	3	4	4
11	4	4	5
12	2	3	6
13	2	4	3
14	3	4	4
15	4	4	5
16	2	3	6

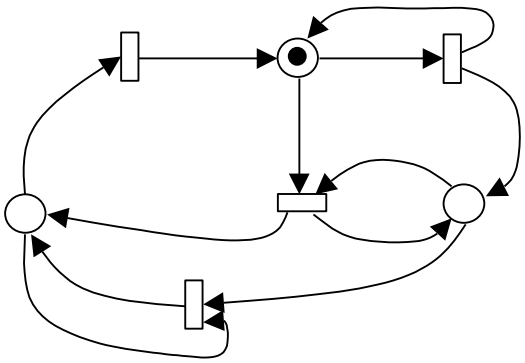
Test questions

1. What elements the standard scheme of control system consists of?
2. In what manner the interaction of controller and controllable object is implemented?
3. What the advantages of Petri nets application for realization of control systems consist in?
4. What properties have to possess a correct algorithm of control?

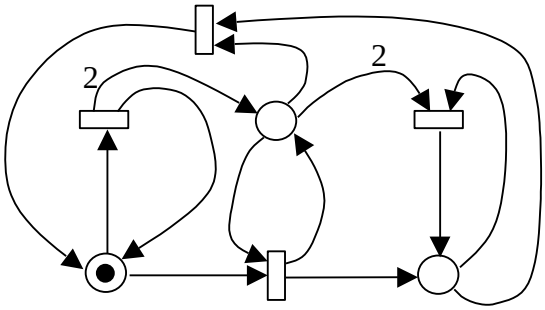
Appendix 1

Collection of Petri nets for investigation

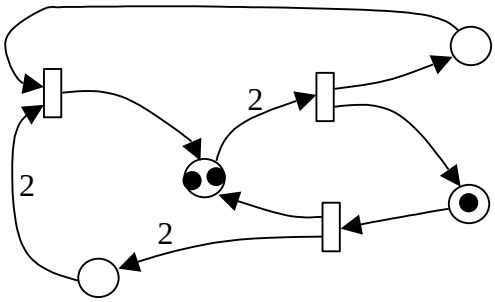
1



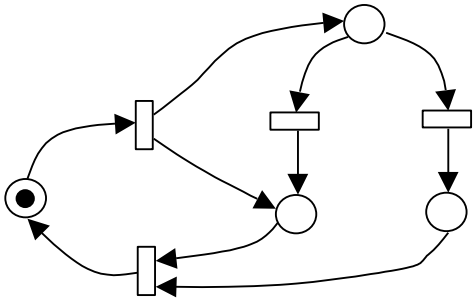
2



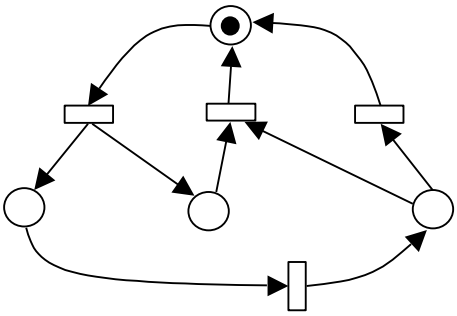
3



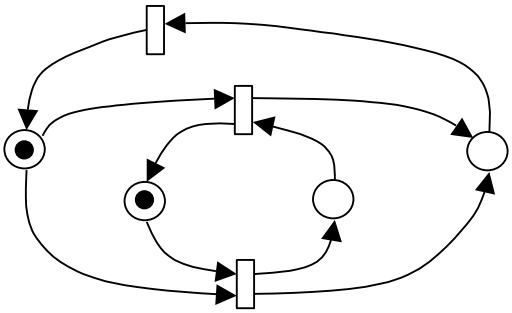
4



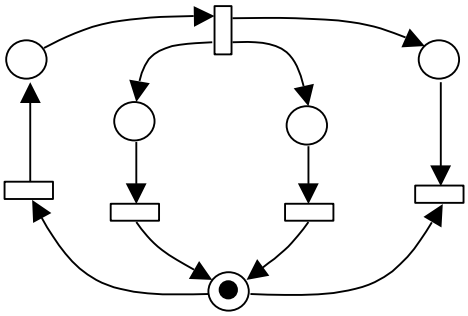
5



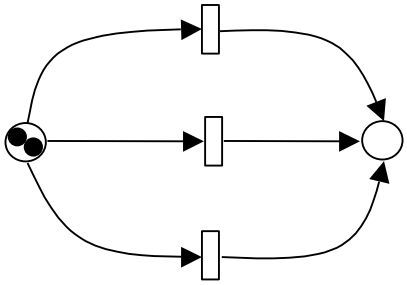
6



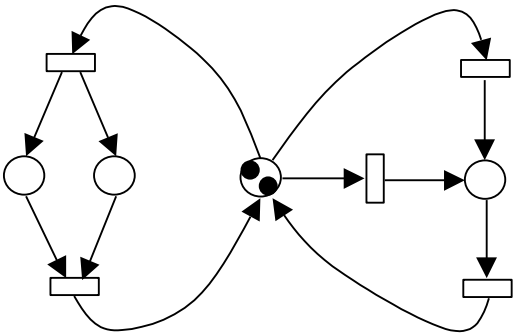
7



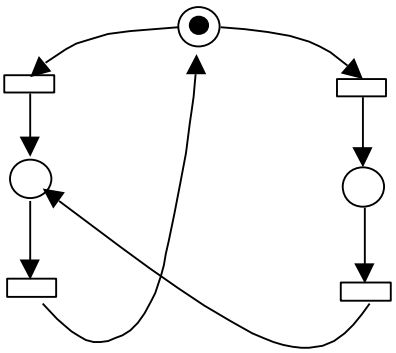
8



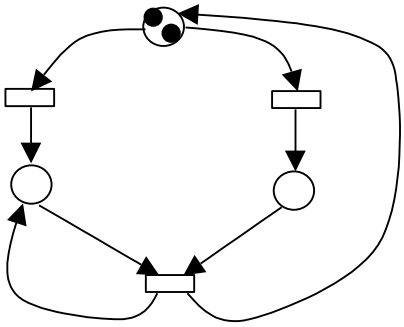
9



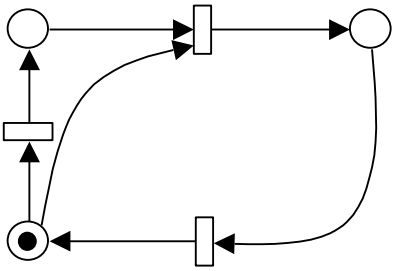
10



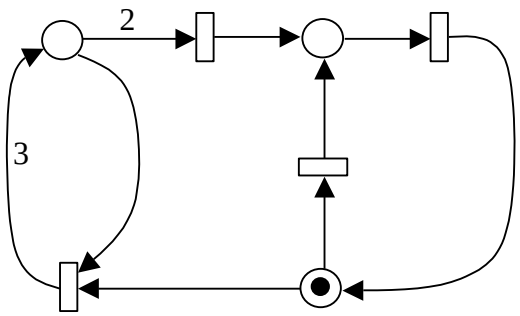
11



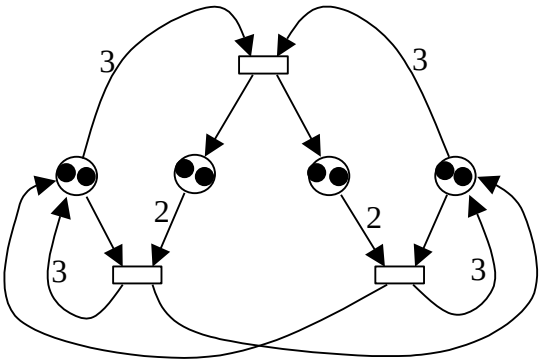
12



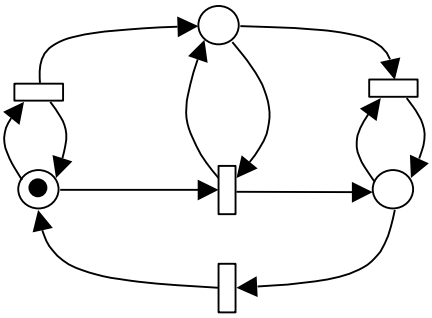
13



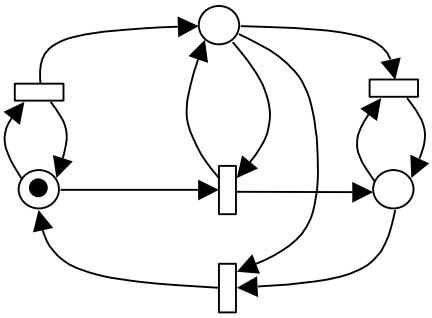
14



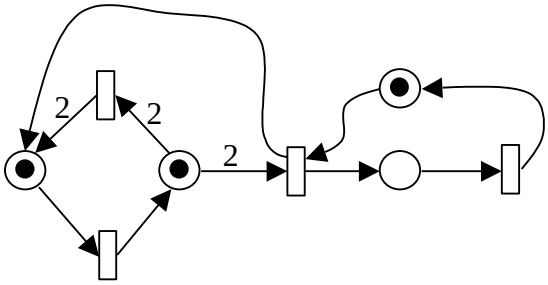
15



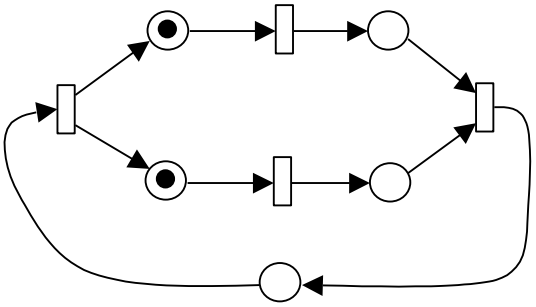
16



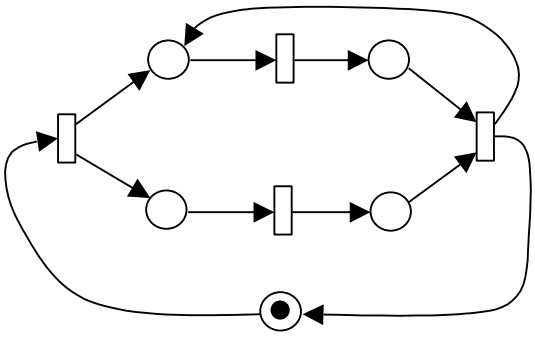
17



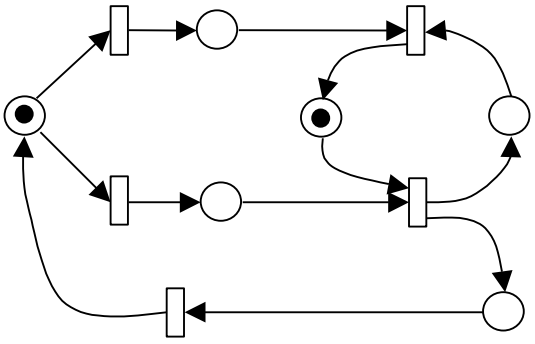
18



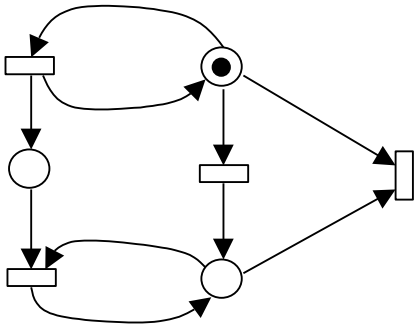
19



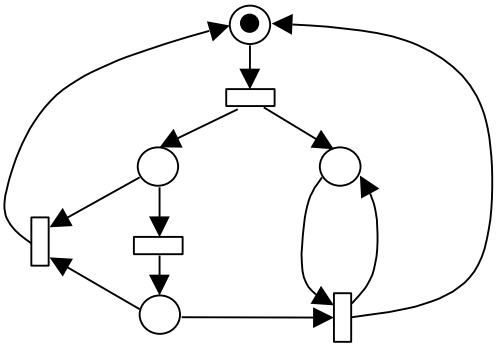
20



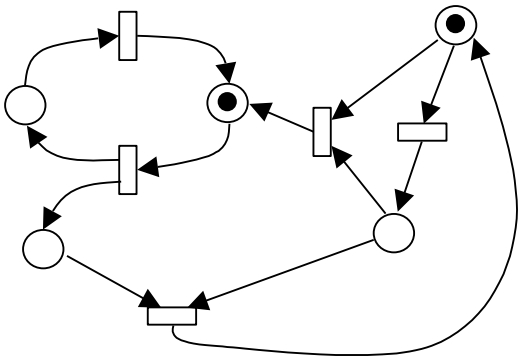
21



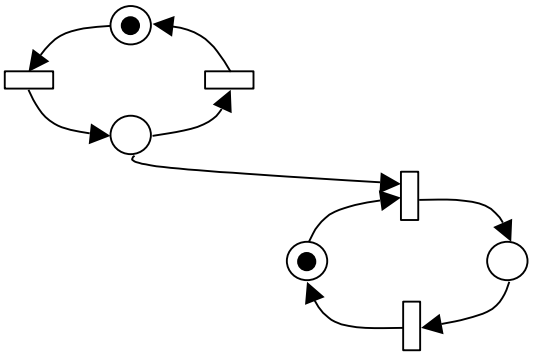
22



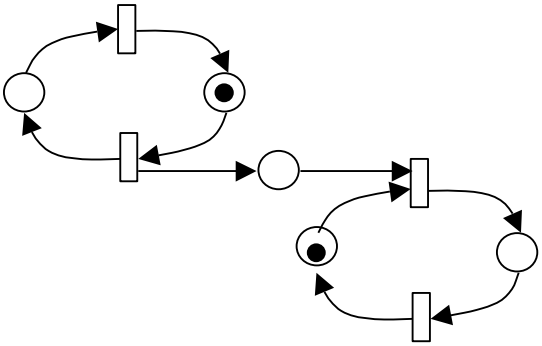
23



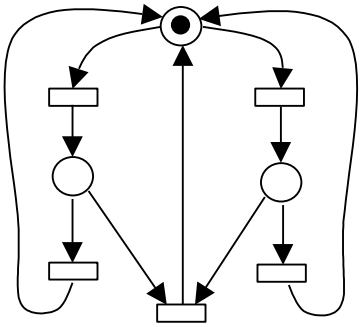
24



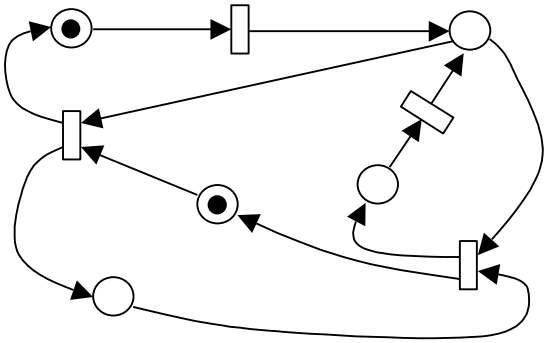
25



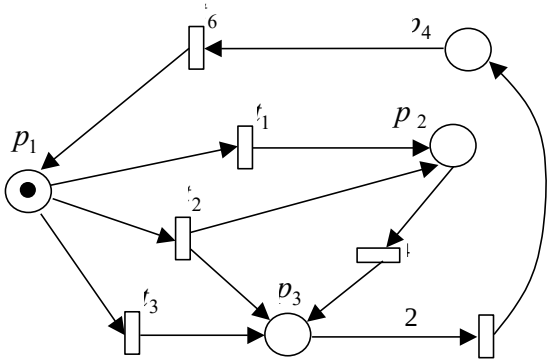
26



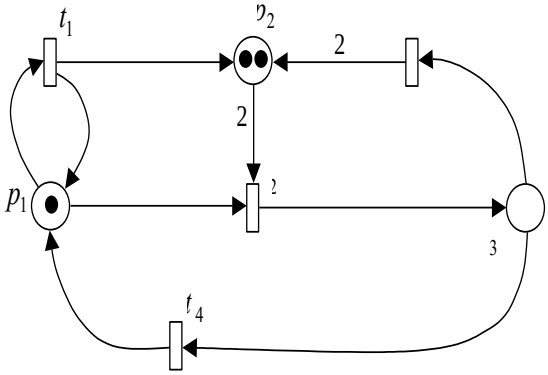
27



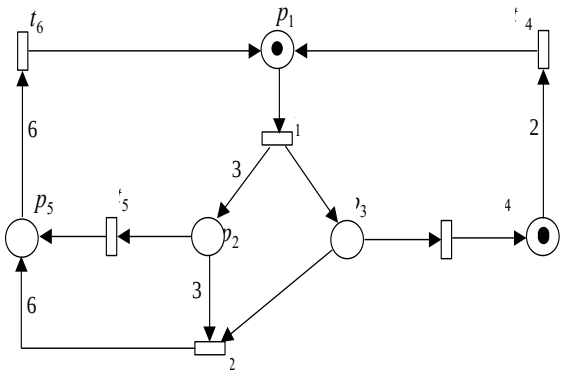
28



29



30



Appendix 2

Individual tasks for systems' models construction

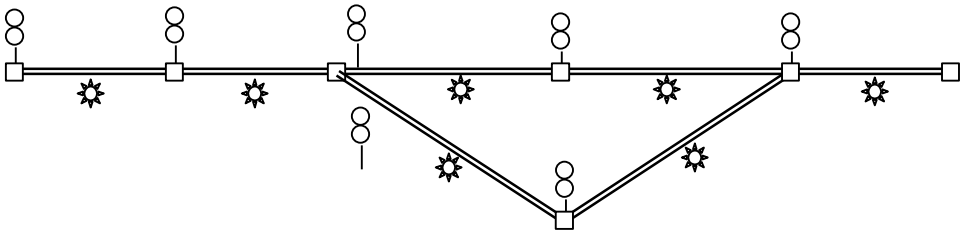
1. Dining philosophers task. Five philosophers take a rest in a five stars hotel. Each of philosophers may either to be thinking or to be eating come onto dining room. In dining room a round table is situated where there are places for each philosopher, five plates, five forks situated between plates and a dish of spaghetti in the center of the table. Each philosopher to take a diner take simultaneously two forks and

commit the meal, on the finish of which he put forks on the table and leaves the dining room for thinking.

2. Represent the model of task 1 about dining philosophers for the case when they take sequentially left and right forks.
3. Represent the model of task 2 for the case when philosopher is invited to the table only in the case he is not require forks taken by already dining philosophers.
4. Represent the model of task 3 for the case when invitations to the table are not obligatory to fulfillment by philosopher.
5. Construct the model of processes servicing into computer, which have two processors, hard disk and printer.
6. Construct the model for interaction of three processes one of which writes messages to buffer and another two process messages and put the results into output buffer.
7. Represent the model of task 6 at limitations on buffers' size.
8. Represent the model of task 7 at an arbitrary given number of writing and reading processes.
9. Construct the elevator's model for four-floor house. Call buttons are situated at each floor.
10. Represent the model of task 9 with the limitations on the number of passengers.
11. Construct the model of the sitting for an examination by the group of four pupils to one professor.
12. Construct the model of barber's shop, in which three barbers work and there are five sitting places for the waiting of clients.
13. Represent the model of task 12 for the case when client did not get a place can either be waiting upright or go away unshorn.
14. Construct the model of safe traffic on the railway section represented with the scheme. Traffic is one-way. Section consists of spans with light signals (!) and busyness sensors (*).

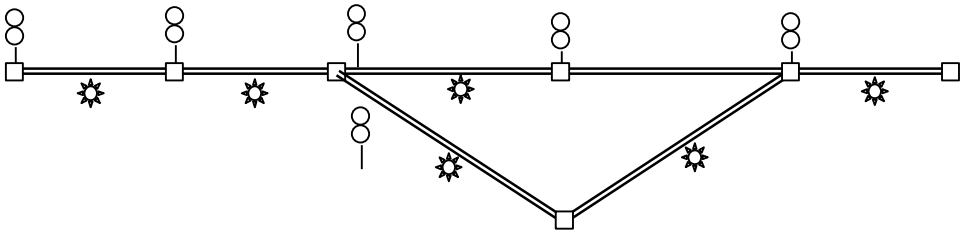
A.

14
18



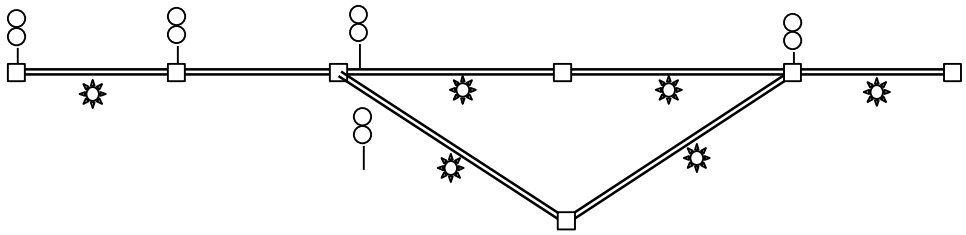
B.

15
19



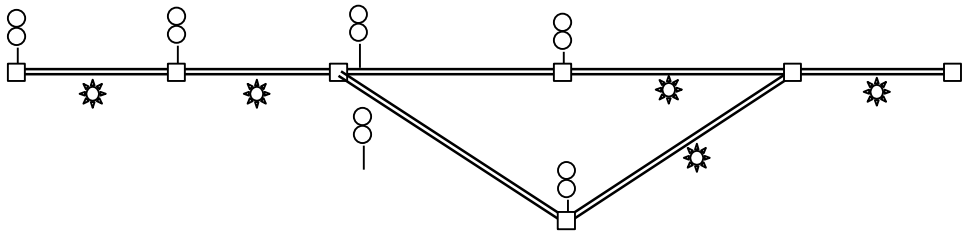
C.

16
20



D.

17



~~15~~. Construct the model of nuclear reaction when from each atom in which bombed with neutron is produced a new atom and eight neutrons.

Appendix 3
Brief descriptions of using simulation systems

A.3.1. Simulation system Tina.

System Tina (www.tina.fr/laas) was developed in LAAS, Toulouse. Tina allows the graphical edition of low-level and timed Petri nets; simulation the dynamics of Petri net – game of tokens; analysis the model with such formal methods as reachability and coverability tree, linear invariants, decomposition into functional subnets.

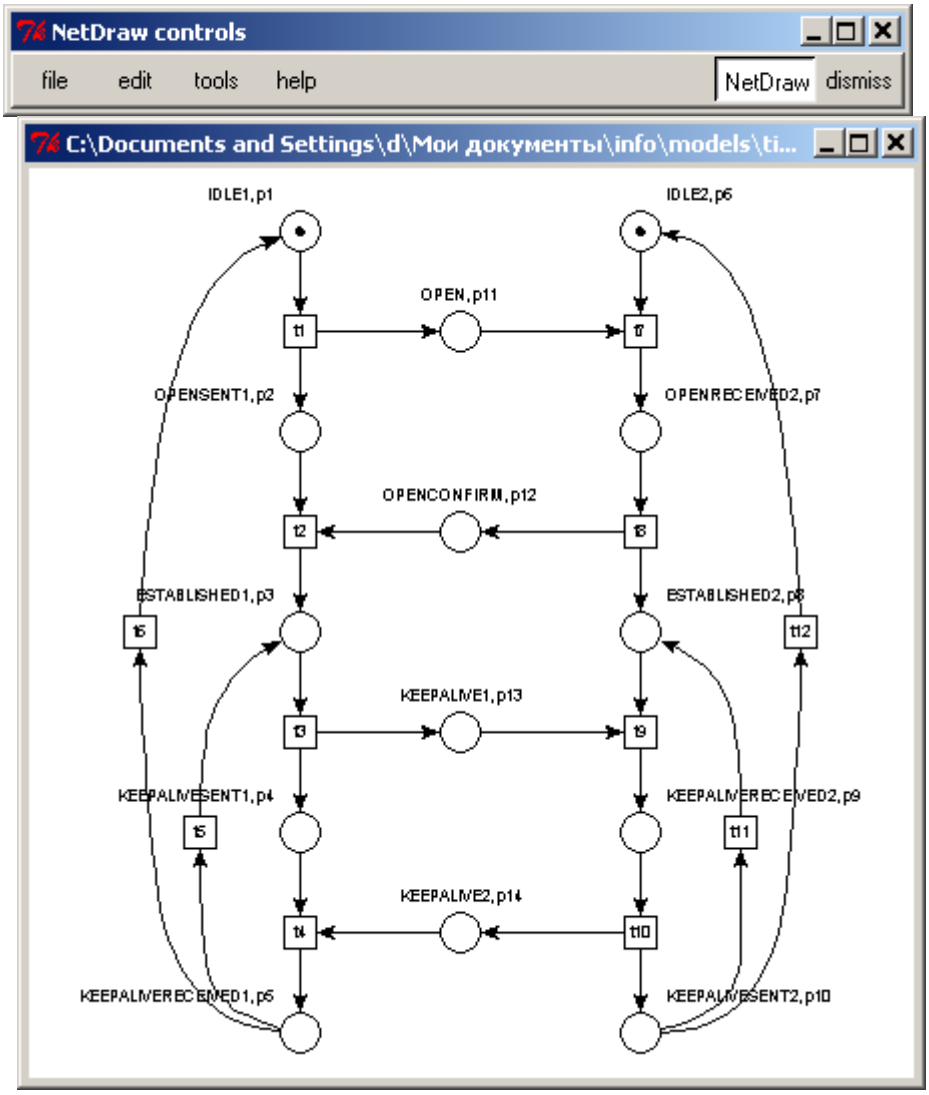
Graphical editor of Petri nets allows the drawing of places, transition and arcs with buttons of mouse as well as input of elements' names and attributes. Moreover, Tina includes facilities for automatic construction of Petri net picture on the abstract description of net.

Two formats of Petri net files are supported: graphical (.ndr) and abstract (.net). In the simplest case an abstract description of net has the following structure:

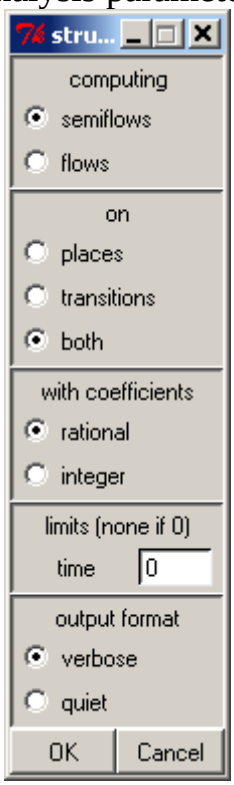
net <petri net name>
tr <transition name> <list of input places> -> <list of output places>
…
pl (<initial marking>)
…

Multiplicity of arc is written after the name of place separated with asterisk symbol.

Let's consider an example of Tina window with Petri net loaded:



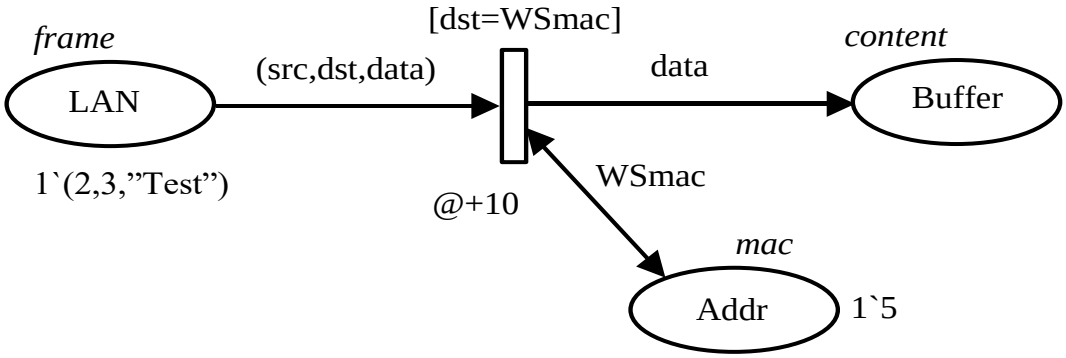
Graphical editor of nets bears the name "nd" (Net Draw) another components of Tina tool set are running from "tools" submenu of nd and have their own windows with parameters. For example, window for structural analysis parameters has the following form:



A.3.2. Simulation system CPN Tools.

System CPN Tools (www.daimi.au.dk/CPNTools) was developed in University of Aarhus, Denmark. System contains graphical editor of models; embedded compiler of ML language, which is used for description of net elements' attributes; simulator, which allows the imitation of net's dynamics in stepwise and automatic modes; analyzer, which provides the construction and analysis of model's state space. Moreover CPN Tools contains the facilities for hierarchical models construction with substitution of transitions and places' fusion.

Elements of colored Petri net except of names have the following attributes: place – type (color) of tokens, initial and current marking; input arc of transition – pattern for input tokens choice; transition – predicate for firable condition named guard, time of firing; output arc of transition – pattern for creation of output tokens. Let's consider the fragment of net modeling the process of frames extraction directed to the workstation with pointed address:

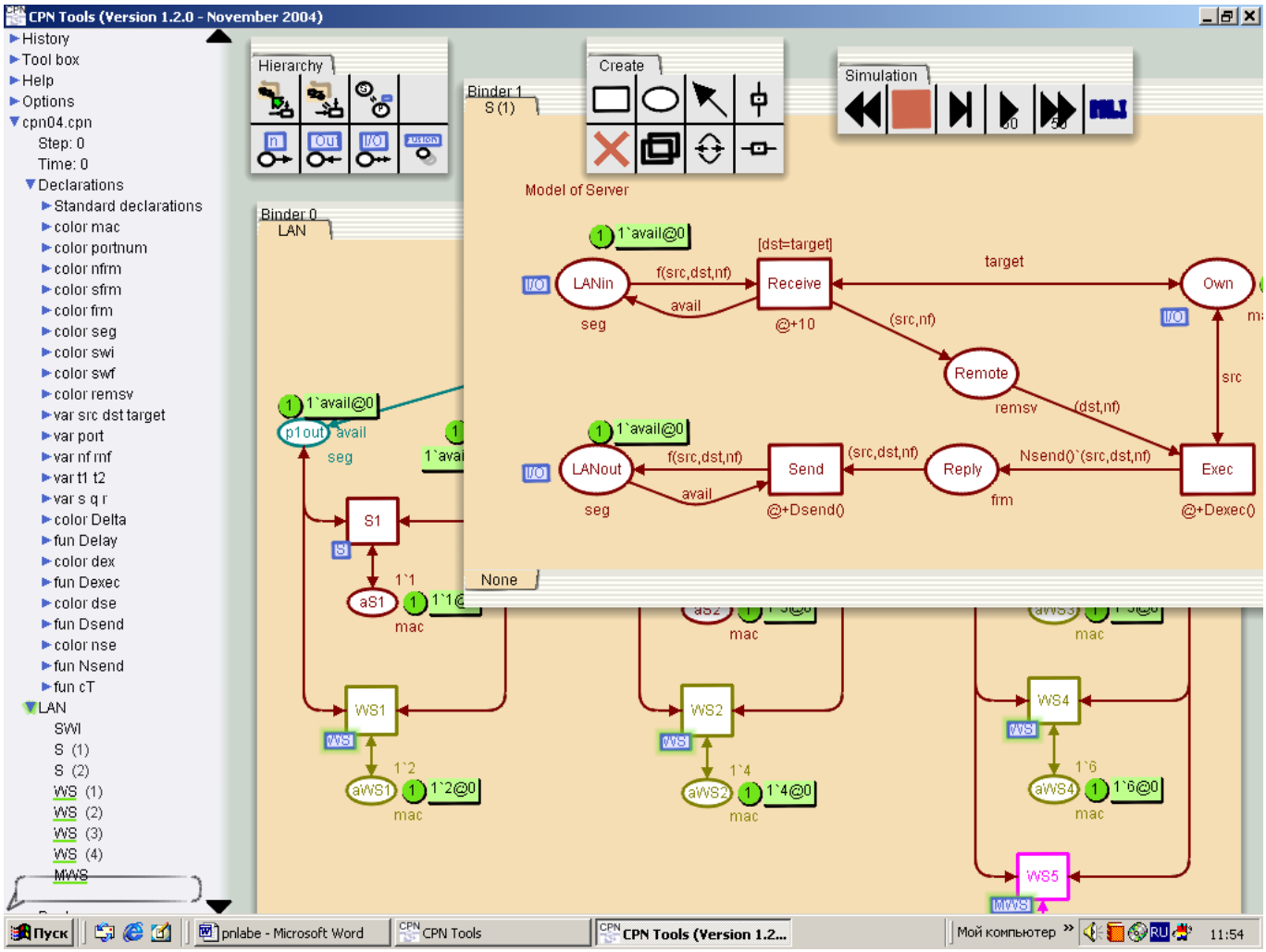



```
color mac = int;
color content = string;
color frame = product mac * mac  * content;
var src, dst, WSmac : mac;
var data : content;
```

For the simplicity of representation the network addresses of computers are modeled by integer number. Frame consists of sender address (**src**), receiver address (**dst**) and data. Checking of receiver address is executed and in the case coincidence the data is put to buffer. Presence of tokens in place is represented with multiset where the number defines multiplicity and then follows the description of token. Traditionally data types are named by colors.

Time characteristics are defined by expressions of the form *@+Delay*, where *Delay* represents the time delay. In the simplest case delays are associated with transitions of net. In more complex situations individual tokens may be delayed with the time expressions on the output arcs of transitions.

For hierarchical construction of model the substitution of transitions are used. Model consists of pages. At substitution, the transition with substitution tag is replaced by the net situated in another page of the model. Let's consider an example of CPN Tools screen:



Left part of screen contains tools, descriptions and pages of model. Pages may be visualized by moving them into the right part of screen. Three tools sets: Create, Simulation and Hierarchy are shown.

CPN Tools has an excellent help system, which uses standalone help sufficient in the most cases and facility to retrieve records from remote on-line help.

List of recommended literature

1. Peterson J.: Petri Net Theory and the Modelling of Systems, Prentice Hall, 1981
2. Sleptsov A.I., Jurasov A.A. Automated development of management and control systems for flexible manufactures // Kiev, Tehnika, 1986, 160p.
3. Murata T.: Petri Nets: Properties, Analysis and Applications. In: Proceedings of the IEEE, April 1989, Vol. 77, 541-580.
4. Jensen K. Colored Petri Nets – Basic Concepts, Analysis Methods and Practical Use. Springer-Verlag, Vol. 1-3, 1997.
5. Beaudouin-Lafon M., Mackay W.E., Jensen M. et al. CPN Tools: A Tool for Editing and Simulating Coloured Petri Nets. LNCS 2031: Tools and Algorithms for the Construction and Analysis of Systems, 2001, 574-580. (www.daimi.au.dk/CPNTools)
6. Girault C., Valk R.: Petri nets for systems engineering. Springer-Verlag, 2003, 607 p.
7. Diaz M. (dir.) Les résaux de Petri. Hermés, Traité IC2, Paris, 2001
8. Berthomieu B., Ribet O.-P., Vernadat F. The tool TINA - construction of abstract state space for Petri nets and Time Petri nets. International Journal of Production Research, 2004 (www.laas.fr/tina)