AKADEMIA GÓRNICZO-HUTNICZA
IM. STANISŁAWA STASZICA W KRAKOWIE

AGH UNIVERSITY OF SCIENCE
AND TECHNOLOGY

Coloured Petri nets - CPN

Tomasz Szmuc

Agenda

1. Multisets and their properties
2. Coloured Petri net and its execution
3. Time multisets
4. Timed coloured Petri net
5. Example of modelling

www.agh.edu.pl

Multisets and their properties 1/5

Multiset notion was introduced to describe multicopy property of set elements.

The theory is therefore over classical sets theory, by an extention of membership function:

- Set theory range of the membership function is {0,1}
- Multiset theory the range is interval [0,*k*] (non-negative integer) – *k* is a finite number
- Fuzzy sets theory range is [0,1] (real)

The membership function defines number of copiies of any element constituting given multiset.

www.agh.edu.pl

Multisets and their properties 2/5

Definition
A pair **(*S, m*)** is ***a* multiset**, where ***S*** is nonempty set and ***m*** is function $m \in [S \to N]$. Any such pair is called multiset over set *S*. Any nonnegative integer $m(s) \in N$ defines number of instances of *s* in the multiset.
Multiset is usually represented in the form of sum:

$$\sum_{s \in S} m(s)`s$$

The following notation is introduced:

- S_{MS} - set of all multisets over set *S*
- Any nonnegative integer $m(s)$ ($s \in S$) is called coefficient of element s .
- Element s belongs to multiset (s, m) iff $m(s) > 0$. The fact is also denoted shortly $s \in m$.

www.agh.edu.pl

Multisets and their properties 3/5

Definition

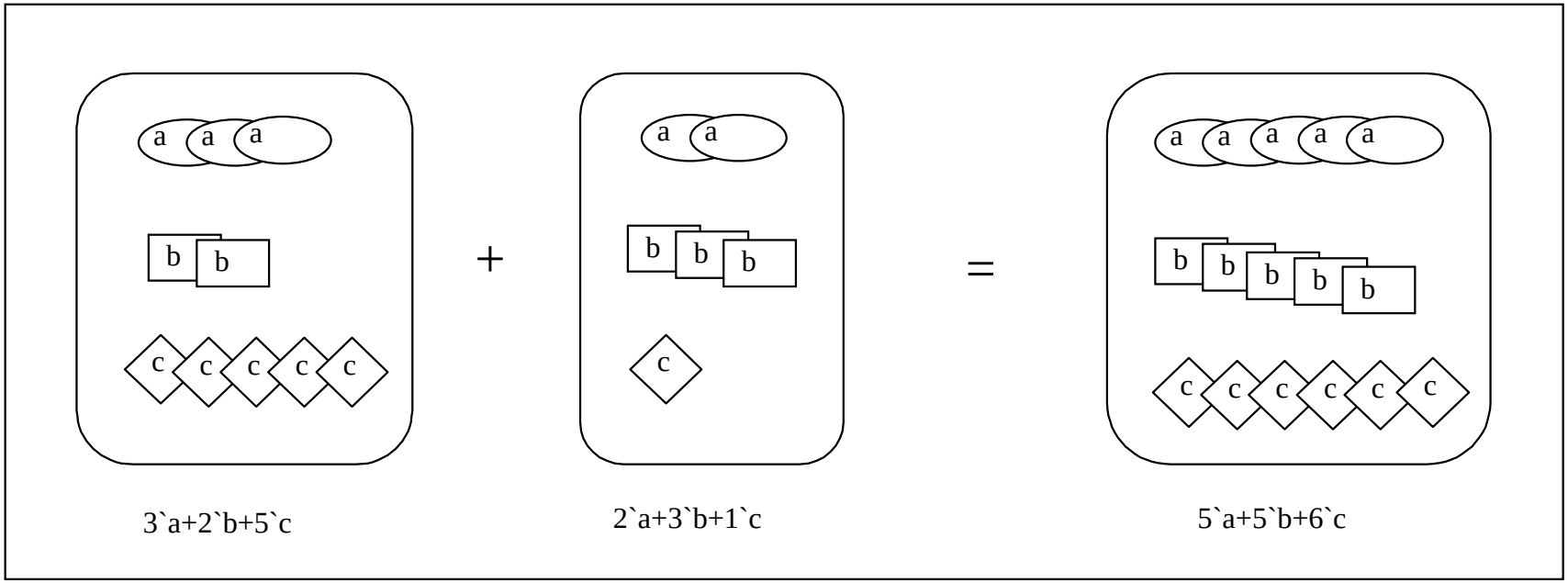
Adding, scalar multiplying, comparison and *power of multiset* are defined as below. Let consider multisetets: $m, m_1, m_2 \in S_{MS}$, and $n \in N$.

- Adding : $m_1 + m_2 = \sum_{s \in S}\big(m_1(s) + m_2(s)\big)`s$
- Scalar multiplying $n \cdot m = \sum_{s \in S}\big(n \cdot m(s)\big)`s$
- Comparison $m_1 \neq m_2 \Leftrightarrow (\exists s \in S)\, m_1(s) \neq m_2(s)$
  $m_1 \leq m_2 \Leftrightarrow (\forall s \in S) m_1(s) \leq m_2(s)$
- Power of multiset $|m| = \sum_{s \in S} m(s)$
- If $m_1 \leq m_2$ then difference of the multisets is defined as:
  $m_2 - m_1 = \sum_{s \in S}\big(m_2(s) - m_1(s)\big)`s$

If $|m| = \omega$, then multiset is Infinite, finite otherwise.

www.agh.edu.pl

Multisets and their properties 4/5



Example of addition of multisets

Multisets and their properties 5/5

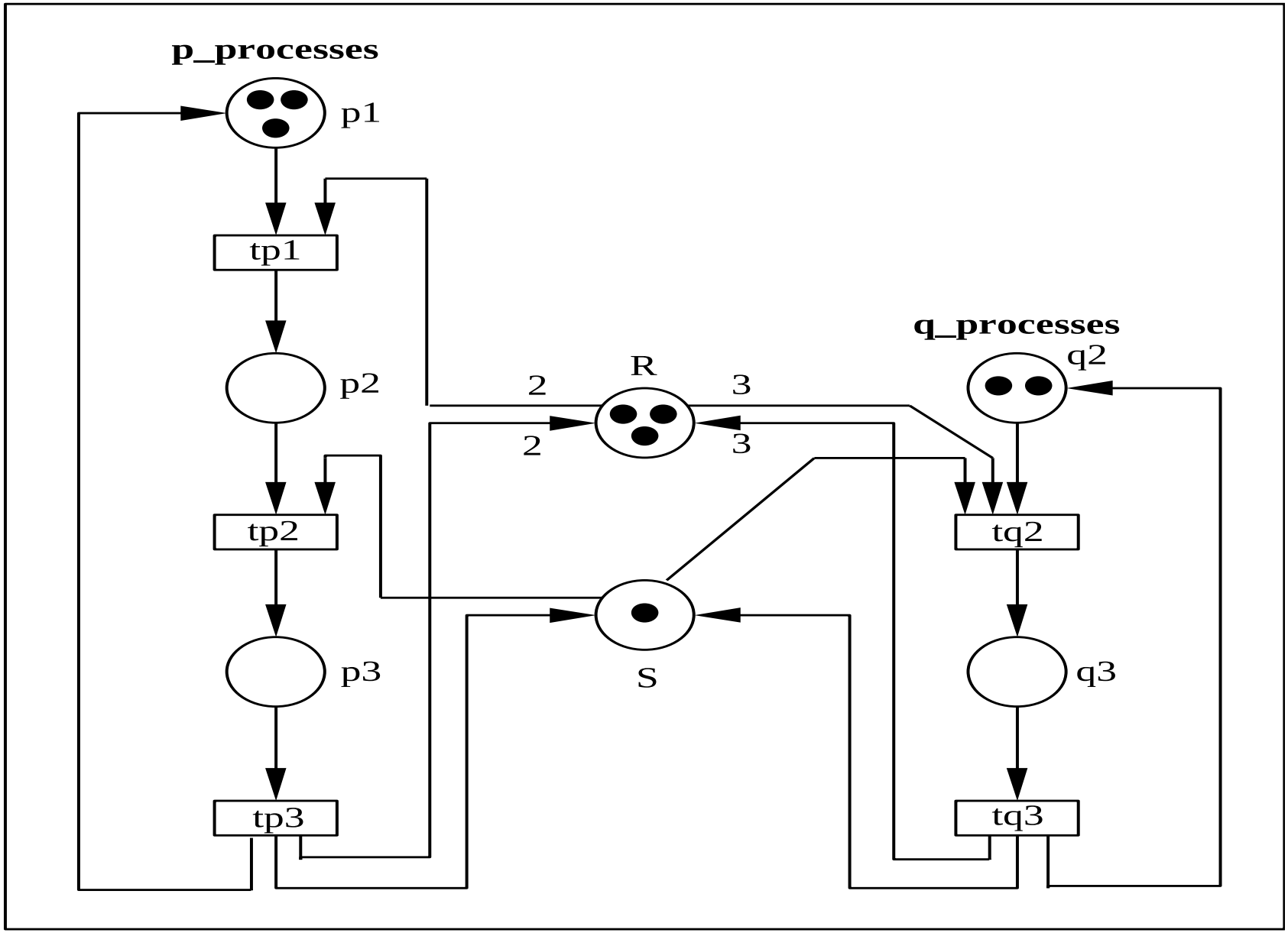
Proposal

For multisets $m, m_1, m_2, m_3 \in S_{MS}$ and non-negative integers $n, n_1, n_2 \in N$ the following properties are held:

1. Commutative property $m_1 + m_2 = m_2 + m_1$
2. Associative property $m_1 + (m_2 + m_3) = (m_1 + m_2) + m_3$
3. Additive identity $m + \emptyset = m$
4. Multiplicative identity $1 \cdot m = m$
5. $0 \cdot m = \emptyset$
6. $n \cdot (m_1 + m_2) = (n \cdot m_1) + (n \cdot m_2)$
7. $(n_1 + n_2) \cdot m = (n_1 \cdot m) + (n_2 \cdot m)$
8. $n_1 \cdot (n_2 \cdot m) = (n_1 \cdot n_2) \cdot m$
9. $|m_1 + m_2| = |m_1| + |m_2|$
10. $|n \cdot m| = n \cdot |m|$

www.agh.edu.pl

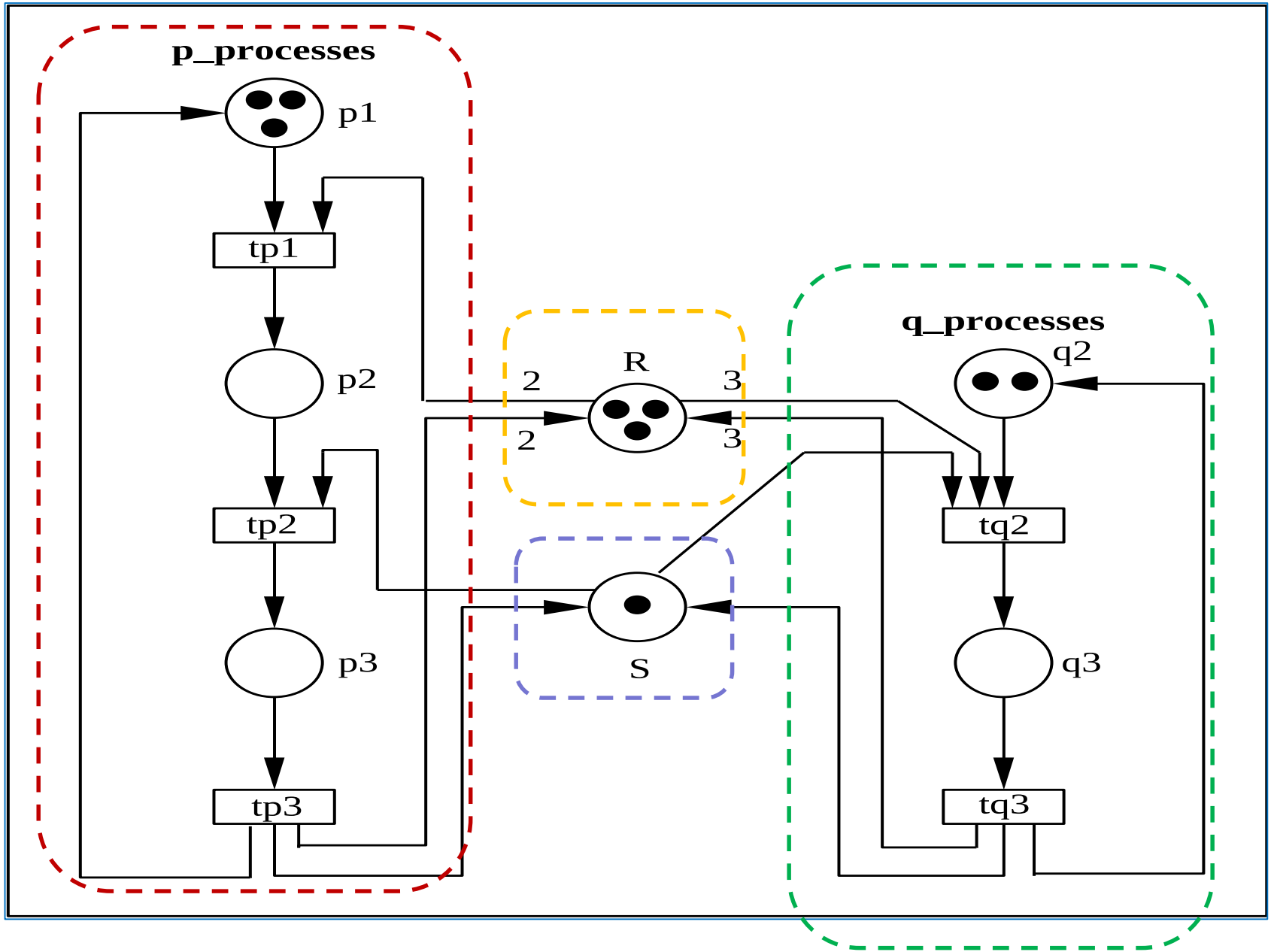
Towards Coloured Petri Nets



Processes *p*, *q* using resources *R* and *S*

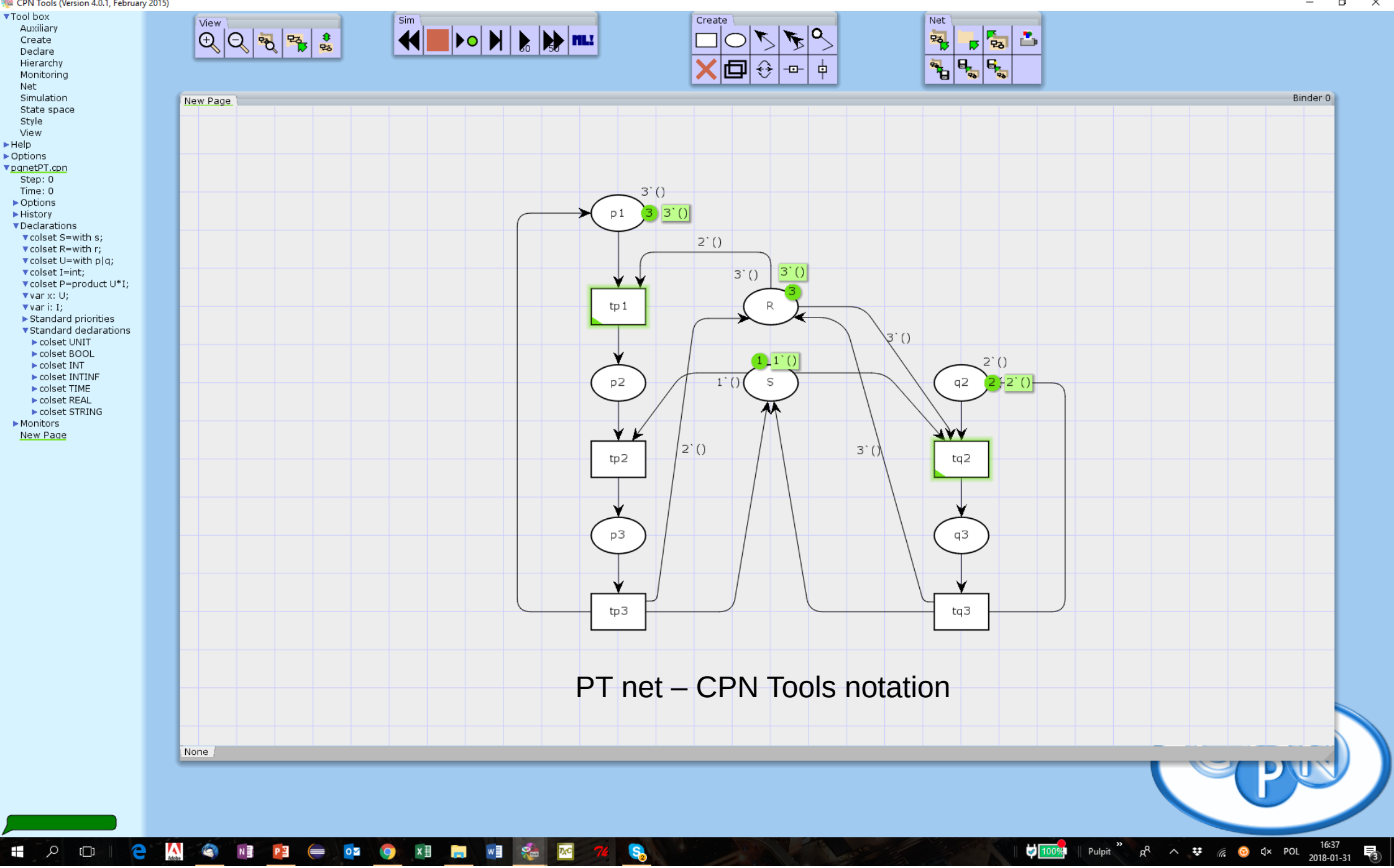
www.agh.edu.pl

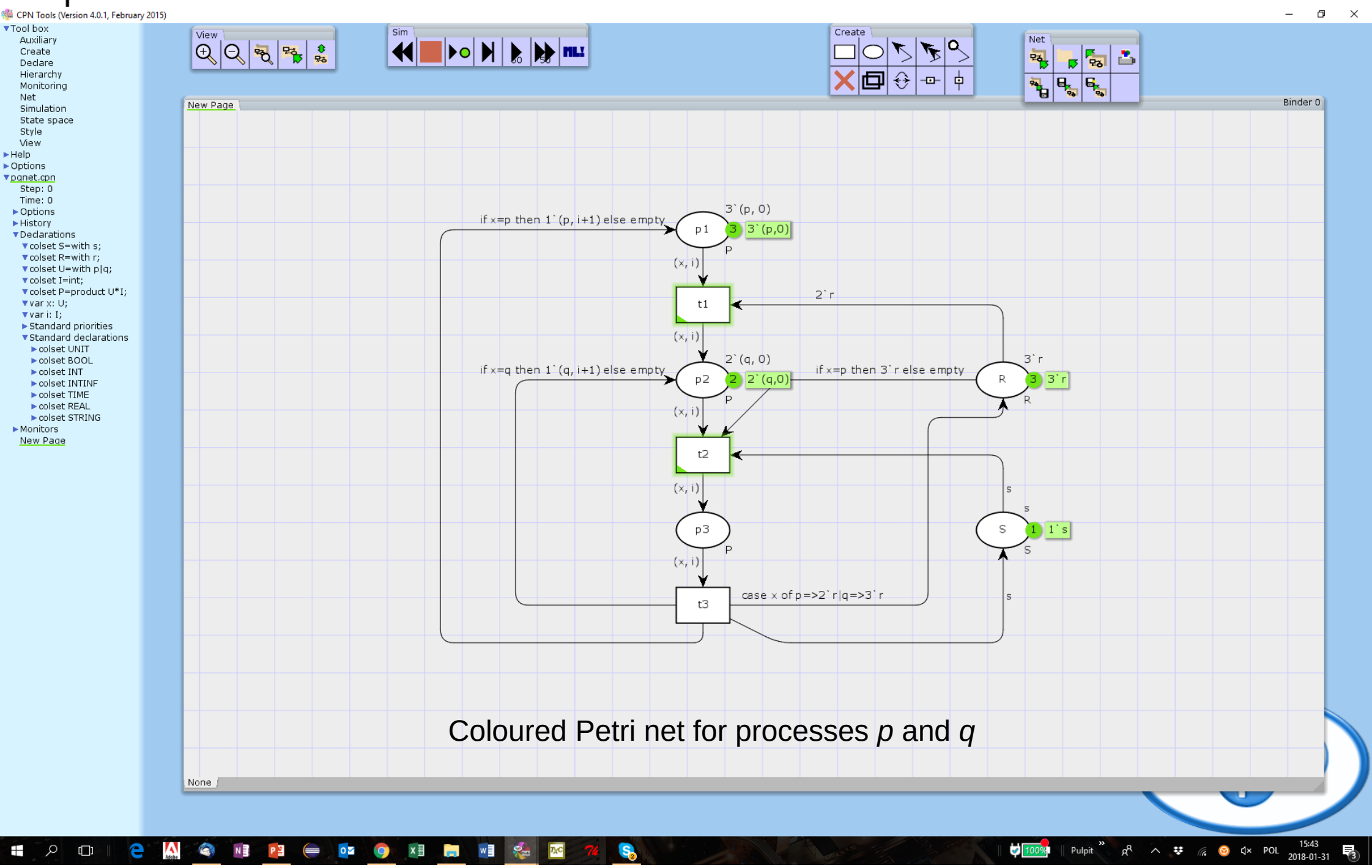
Towards Coloured Petri Nets



Processes *p*, *q* using resources *R* and *S*

www.agh.edu.pl


CPN Tools (Version 4.0.1, February 2015)
Tool box
Auxiliary
Create
Declare
Hierarchy
Monitoring
Net
Simulation
State space
Style
View
Help
Options
pqnetPT.cpn
Step: 0
Time: 0
Options
History
Declarations
colset S=with s;
colset R=with r;
colset U=with p|q;
colset I=int;
colset P=product U*I;
var x: U;
var i: I;
Standard priorities
Standard declarations
colset UNIT
colset BOOL
colset INT
colset INTINF
colset TIME
colset REAL
colset STRING
Monitors
New Page
View
Sim
Create
Net
New Page
Binder 0
3`()
p1
3
3`()
2`()
3`()
3`()
tp1
R
3
3`()
1
1`()
1`()
S
p2
2`()
q2
2
2`()
tp2
2`()
3`()
tq2
p3
q3
tp3
tq3
None
PT net – CPN Tools notation



Coloured Petri net for processes *p* and *q*

CPN definition 2/4

Definition

Coloured Petri net is a tuple $CPN = (\Sigma, P, T, N, C, G, E, I)$, where

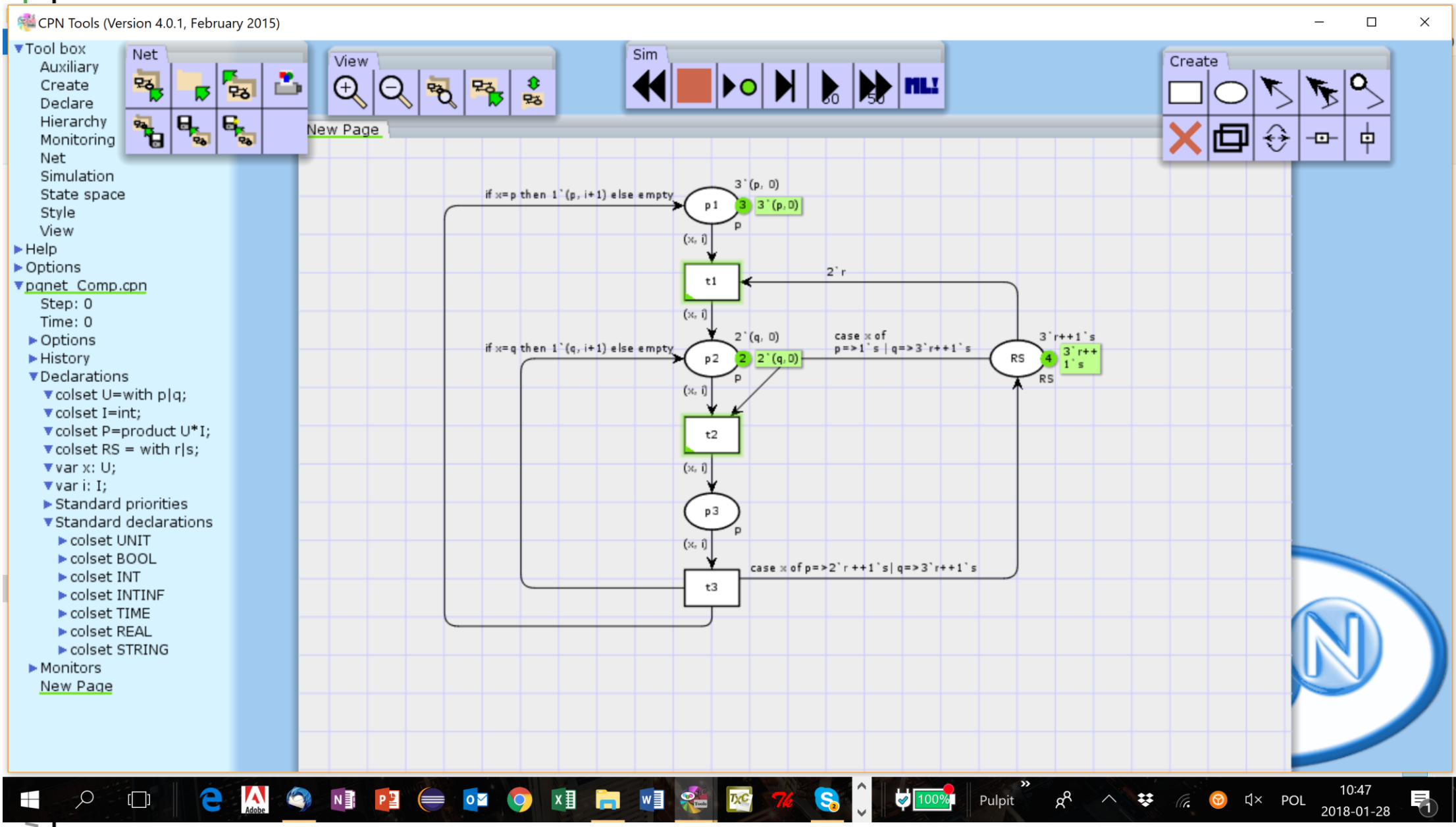
1. Σ – nonempty, finite set of types called **set of coloures**;
2. P – finite **set of places**;
3. T – finite **set of transitions**;
4. A – finite **set of arcs**, such that:
$$P \cap T = T \cap A = T \cap A = \emptyset;$$
5. N – **function of arcs**, $N : A \to P \times T \cup T \times P$;
6. C – **function of colours**, $C : P \to \Sigma$;
7. G – **function of guards**, $G : T \to EB$, where EB is a set of boolean expressions, satisfying the condition:
$$(\forall t \in T) Type\big(G(t)\big) = B \ \wedge Type(Var(G(t)) \subseteq \Sigma;$$
8. E – **function of arcs expressions**, $E : A \to EC$, where EC is a set of expressions with values in colures multitisets, satisfying the condition:
$(\forall a \in A) Type\big(E(a)\big) = C(p(a))_{MS} \wedge Type(Var(E(a)) \subseteq \Sigma,$
where $p(a)$ is a place of $N(a))$ arc;
9. I – **initialization function**, $I : P \to ECP$, where ECP is a set of expression satisfying the condition:
$$(\forall p \in P)\ Type\big(I(p)\big) = C(p))_{MS}.$$

CPN definition 3/4

Explanations

- Type of variable v is denoted by $Type(v)$. Type of value of expression $expr$ is denoted by $Type(expr)$ correspondingly.
- Set of all variables in expression $expr$ is denoted as $Var(expr)$. For a set of variables V assign of values for every variable (from the set) is called as **binding of the set of variables**.
- Value of expression $expr$ for binding b is denoted by $expr < b >$. It is assumed that set of variables $Var(expr)$ is a subset of the set of binding variables.

www.agh.edu.pl


CPN Tools (Version 4.0.1, February 2015)
Tool box
Auxiliary
Create
Declare
Hierarchy
Monitoring
Net
Simulation
State space
Style
View
Help
Options
pqnet_Comp.cpn
Step: 0
Time: 0
Options
History
Declarations
colset U=with p|q;
colset I=int;
colset P=product U*I;
colset RS = with r|s;
var x: U;
var i: I;
Standard priorities
Standard declarations
colset UNIT
colset BOOL
colset INT
colset INTINF
colset TIME
colset REAL
colset STRING
Monitors
New Page
Net
View
Sim
Create
New Page
if x=p then 1`(p, i+1) else empty
p1
3`(p, 0)
3`(p,0)
p
(x, i)
t1
2`r
(x, i)
if x=q then 1`(q, i+1) else empty
p2
2`(q, 0)
2`(q,0)
p
case x of p=>1`s | q=>3`r++1`s
RS
3`r++1`s
3`r++
1`s
RS
(x, i)
t2
(x, i)
p3
p
(x, i)
case x of p=>2`r ++1`s| q=>3`r++1`s
t3
100%
Pulpit
POL
10:47
2018-01-28

Execution of Coloured Petri Net

Auxiliary notions 1/6

Definition

Function b defined over $Var(t)$ is called as **binding of transition t iff** $\left(\forall v \in Var(t)\right)\ b(v) \in Type(v)\ \wedge\ G(t) < b >$.

Set of all bindings for t is denoted by $B(t)$.

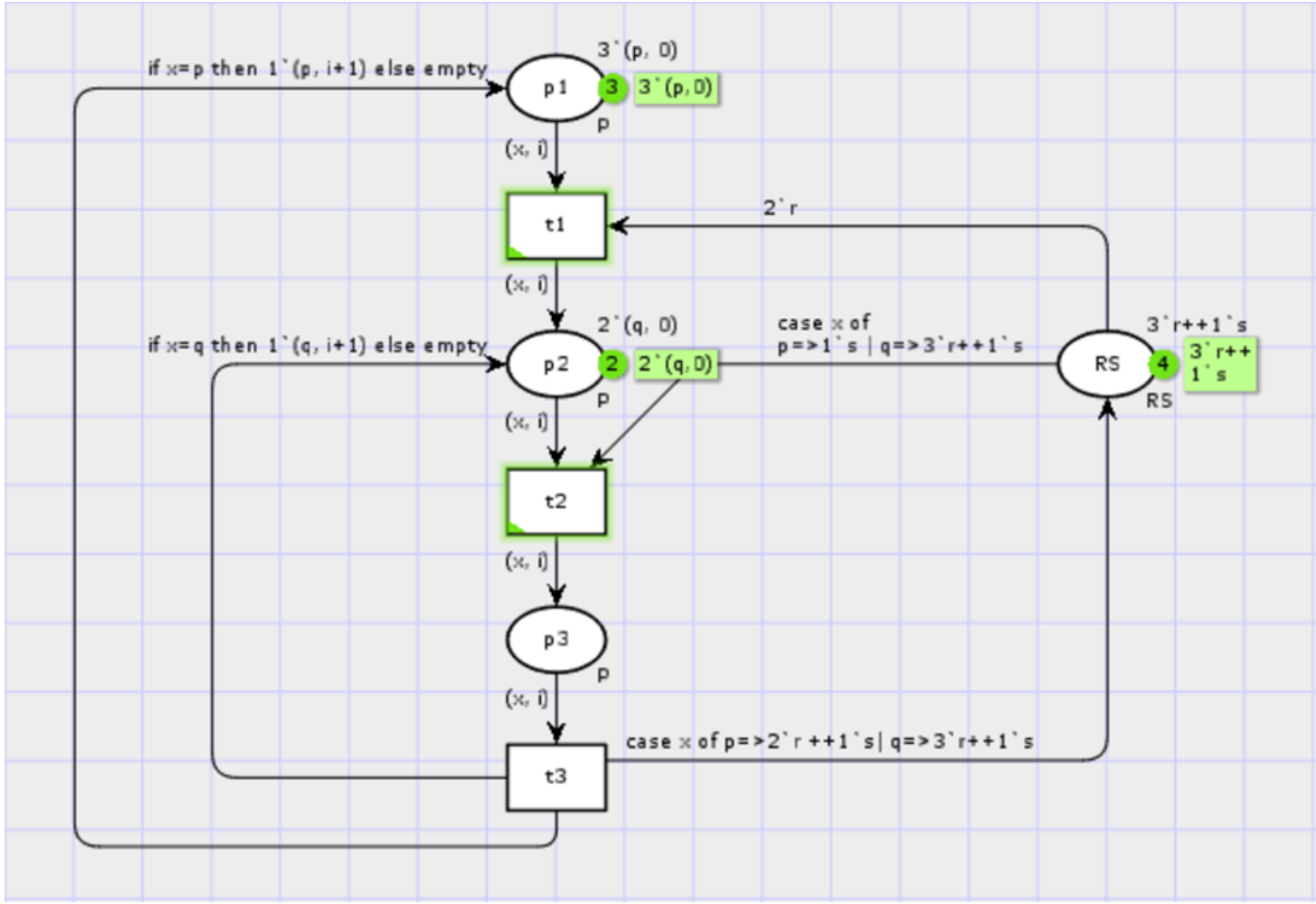
$Var(t) = \left\{v \middle| v \in Var\left(G(t)\right)\ \vee\ \left(\exists a \in A(t)\right) v \in Var\left(E(a)\right)\right\}$.

$E(x_1, x_2) = \sum_{a \in A(x_1, x_2)} E(a)$, where $(x_1, x_2) \in (P \times T \cup T \times P)$

www.agh.edu.pl

Auxiliary notions 2/6

Binding of transition t is any set of values of guard (of the transition) variables and variables in expressions labelling arcs surrounding transition – for which guard $G(t)$ is true.



www.agh.edu.pl

Auxiliary notions 3/6

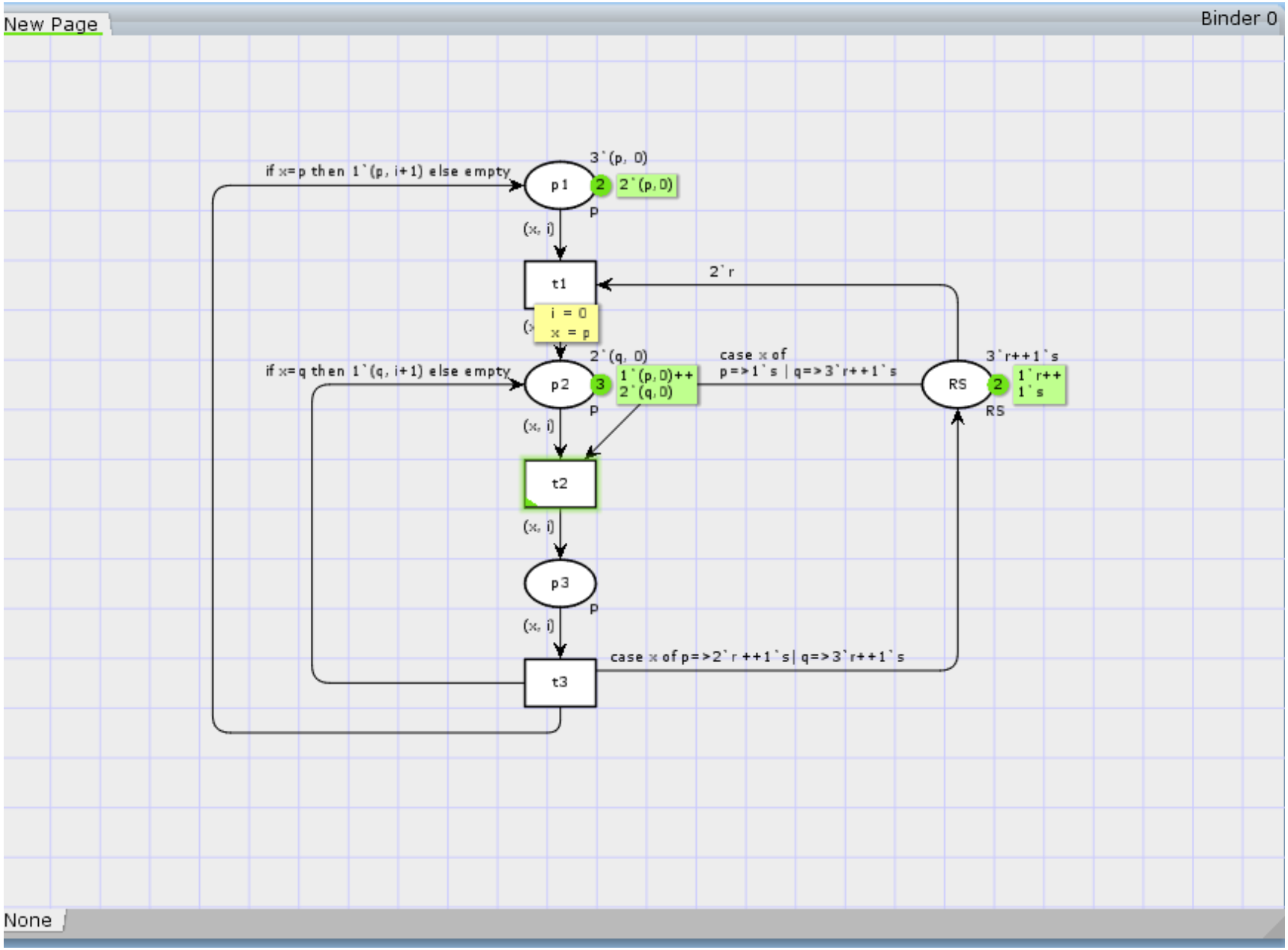


Fig. Marking of net after execution of transition $t1$

www.agh.edu.pl

Auxiliary notions 4/6

Definition

Token element is any pair (p, c), where $p \in P$ and $c \in C(p)$.
Binding element is any pair (t, b), where $t \in T$ and $b \in B(t)$.
Set of all token elements is denoted by $\boldsymbol{TE}$, and set of all binding elements is denoted by $\boldsymbol{BE}$.
Marking is a **multiset over** $\boldsymbol{TE}$ and step is nonempty finite multiset over BE.
Initial marking m_0 is marking obtained by evaluation of initialization function:

$$(\forall (p, c) \in TE)\ \ m_0(p, c) = (I(p))(c).$$

Set of all markings is denoted by S and set of all steps by Y correspondingly.

www.agh.edu.pl

Auxiliary notions 5/6. Intuitions

Token element (p, c) consists of colour and place to which the colour is assigned (admissible). Any multiset will be built using such colours. These multisets are called **markings**.

Step (for given transition) is any multiset over set of all bindings, i.e. multiset over values of variables satisfying the guard condition.

www.agh.edu.pl

Auxiliary notions 6/6

Definition
Any **step Y is enabled** in marking M **iff** the condition below is satisfied:

$$(\forall p \in P) \quad \sum_{(t,b)\in Y} E(p,t) < b > \ \leq \ M(p)$$

www.agh.edu.pl

Definition 1/2

Definition

If a transition t is enabled in marking M_1, then execution of the transition may occur, i.e. change of marking M_1 into marking M_2, defined in the following way:

$$(\forall p \in P)\ (M_2(p) = M_1(p) - \sum_{(t,b)\in Y} E(p,t) < b > + \sum_{(t,b)\in Y} E(t,p) < b >).$$

Marking M_2 is **directly reachable** from marking M_1. This fact is denoted $M_1 \overset{Y}{\rightarrow} M_2$ or $M_1\ [y > M_2$ (older notation).

www.agh.edu.pl

Definition 2/2

Definition
Finite occurrence (executions) sequence is a sequence of markings and relevant steps:

$$M_1 \overset{Y_1}{\rightarrow} M_2 \overset{Y_2}{\rightarrow} M_3 \cdots \overset{Y_n}{\rightarrow} M_{n+1}\,.$$

Infinite occurrence sequence is denoted:

$$M_1 \overset{Y_1}{\rightarrow} M_2 \overset{Y_2}{\rightarrow} M_3 \cdots\,.$$

www.agh.edu.pl

Colour sets (colset declarations)

The CPN ML contains predefined basic colours (types) inherited from standard ML.

- `colset UNIT = unit; //single value written ()`
- `colset BOOL = bool;`
- `colset INT = int;`
- `colset INTINF = intinf;`
- `colset TIME = time;`
- `colset REAL = real;`
- `colset STRING = string;`

The basic (standard) colours are used to define structured colours.

www.agh.edu.pl

Structured colours

```
colset DATA = string;
colset NO = int;

colset NOxDATA = product NO * DATA;
//records with two fields
colset DATAPACK = record seq:NO * data:DATA;
//package of records
colset ACKPACK = NO;
colset PACKET = union Data:DATAPACK + Ack:ACKPACK;
//union of two colour sets, i.e. union of the two
//packages
colset RESULT = with success | failure | duplicate;
```

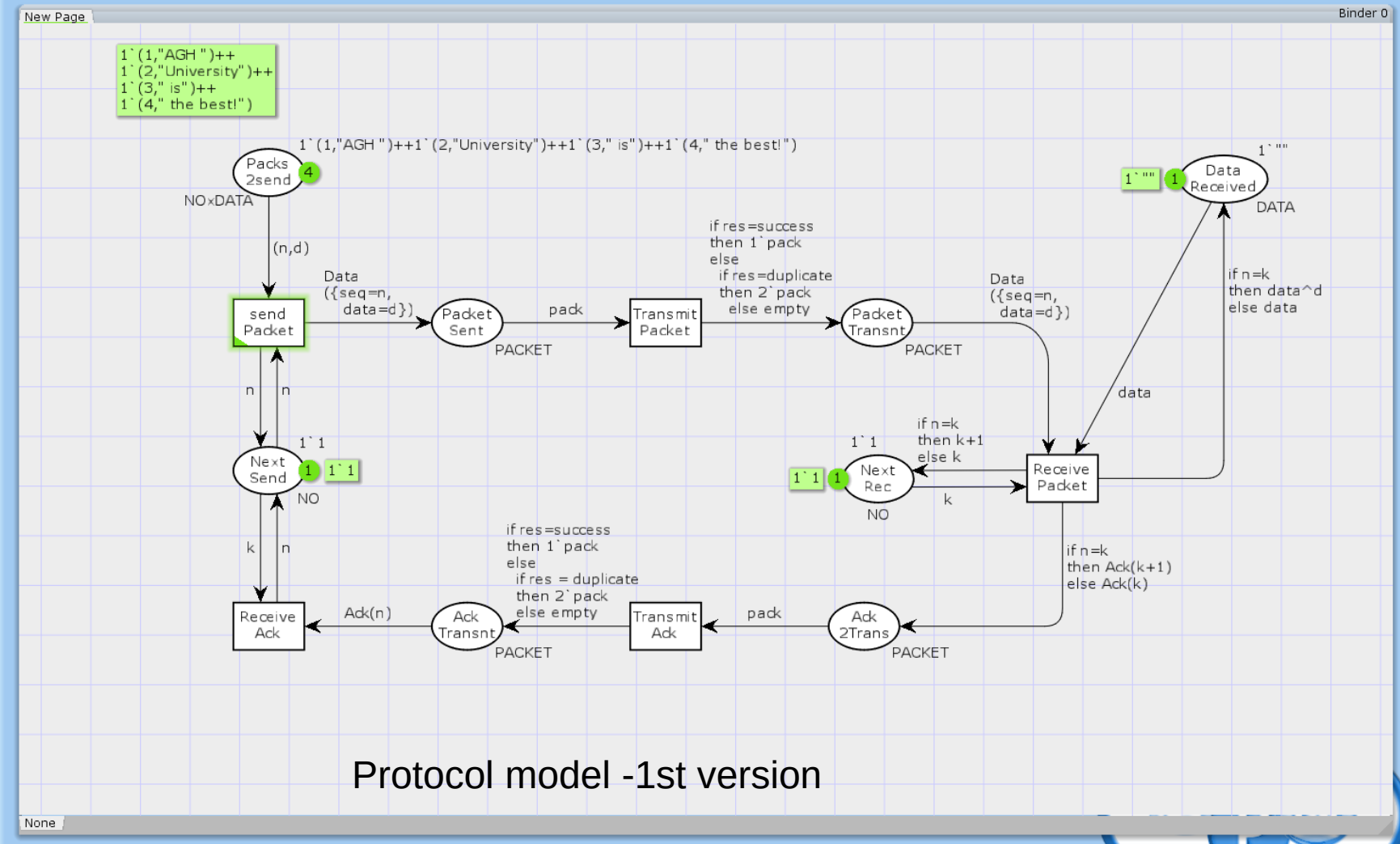
www.agh.edu.pl

CPN Tools (Version 4.0.1, February 2015)

- Tool box
  - Auxiliary
  - Create
  - Declare
  - Hierarchy
  - Monitoring
  - Net
  - Simulation
  - State space
  - Style
  - View
- Help
- Options
- ColoursNet1.cpn
  - Step: 0
  - Time: 0
  - Options
  - History
  - Declarations
    - Standard priorities
    - Standard declarations
      - colset UNIT
      - colset BOOL
      - colset INT
      - colset INTINF
      - colset TIME
      - colset REAL
      - colset STRING
      - colset DATA = string;
      - colset NO=INT;
      - colset NOxDATA = product NO*DATA;
      - colset DATAPACK = record seq:NO * data:DATA;
      - colset ACKPACK = NO;
      - colset PACKET = union Data:DATAPACK + Ack:ACKPACK;
      - colset RESULT = with success | failure | duplicate;
      - var res: RESULT;
      - var k, n:NO;
      - var data, d:DATA;
      - var pack: PACKET;
  - Monitors
  - New Page

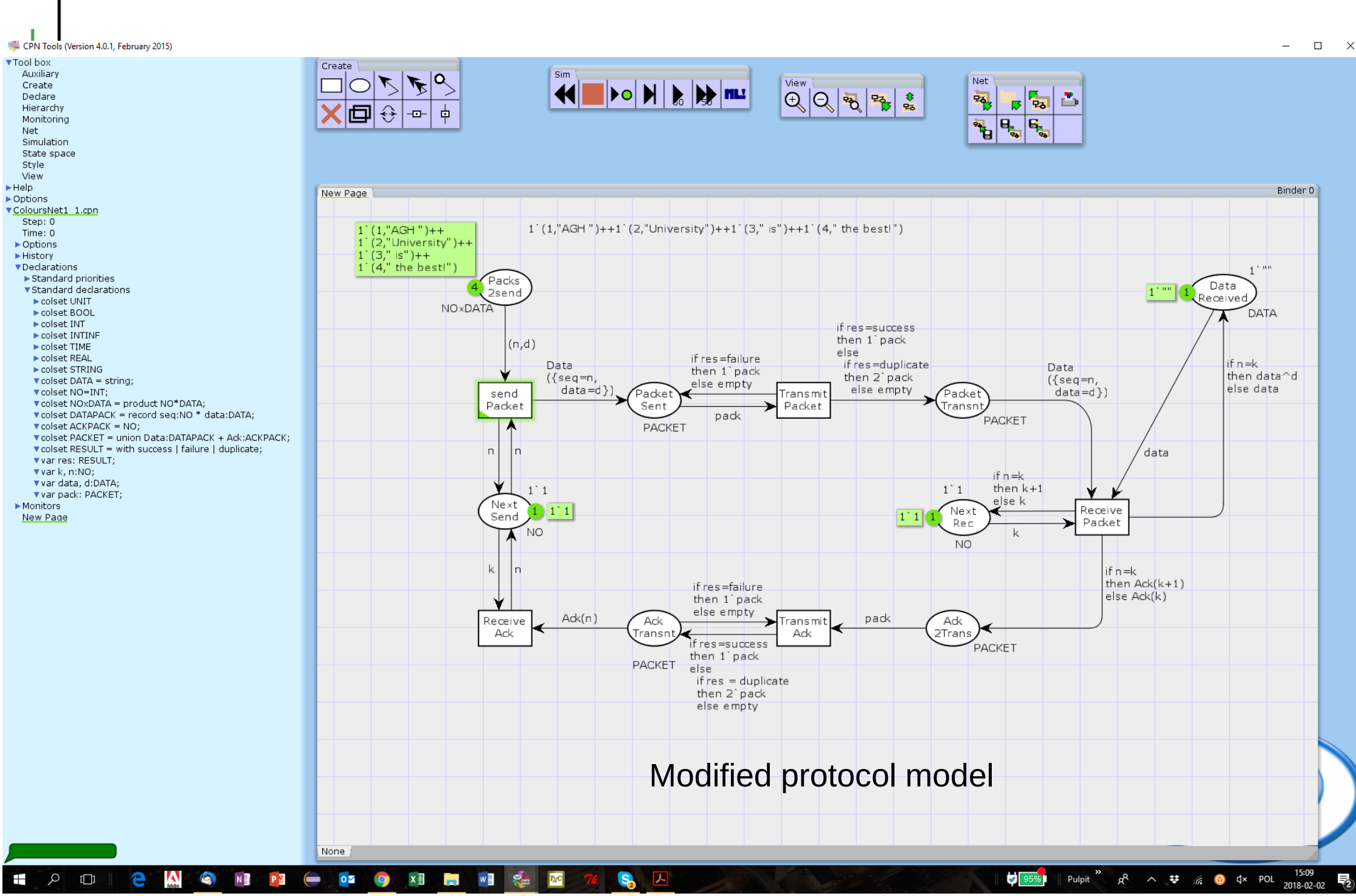
Create | Sim | View | Net

New Page — Binder 0



Protocol model -1st version

None

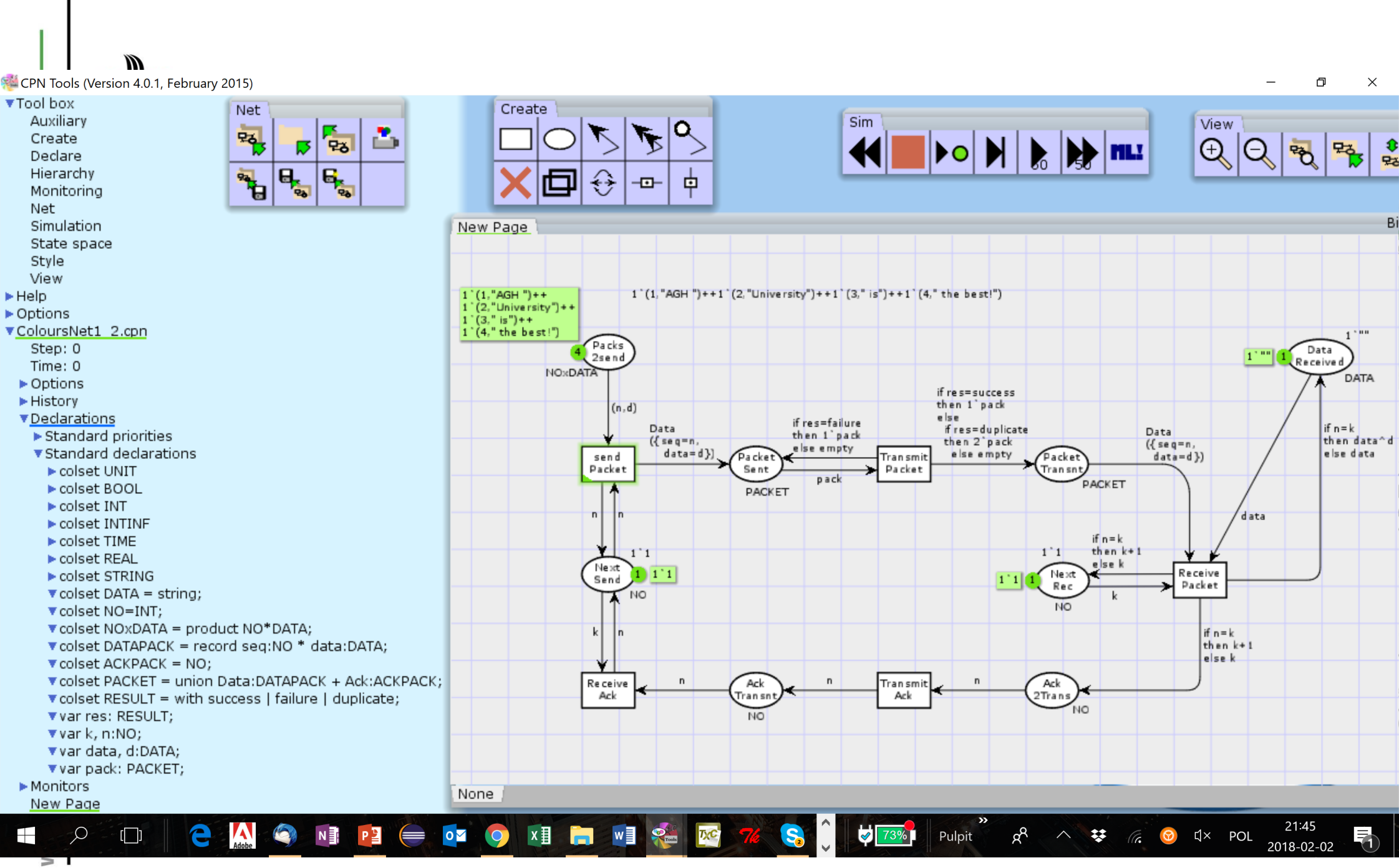

CPN Tools (Version 4.0.1, February 2015)
Tool box
Auxiliary
Create
Declare
Hierarchy
Monitoring
Net
Simulation
State space
Style
View
Help
Options
ColoursNet1_1.cpn
Step: 0
Time: 0
Options
History
Declarations
Standard priorities
Standard declarations
colset UNIT
colset BOOL
colset INT
colset INTINF
colset TIME
colset REAL
colset STRING
colset DATA = string;
colset NO=INT;
colset NOxDATA = product NO*DATA;
colset DATAPACK = record seq:NO * data:DATA;
colset ACKPACK = NO;
colset PACKET = union Data:DATAPACK + Ack:ACKPACK;
colset RESULT = with success | failure | duplicate;
var res: RESULT;
var k, n:NO;
var data, d:DATA;
var pack: PACKET;
Monitors
New Page
Create
Sim
View
Net
New Page
Binder 0
1`(1,"AGH ")++
1`(2,"University")++
1`(3," is")++
1`(4," the best!")
1`(1,"AGH ")++1`(2,"University")++1`(3," is")++1`(4," the best!")
Packs 2send
NO×DATA
(n,d)
send Packet
Data ({seq=n, data=d})
Packet Sent
PACKET
if res=failure then 1`pack else empty
pack
Transmit Packet
if res=success then 1`pack else if res=duplicate then 2`pack else empty
Packet Transnt
PACKET
Data ({seq=n, data=d})
Data Received
1`""
DATA
if n=k then data^d else data
data
n
n
Next Send
1`1
NO
k
n
Receive Ack
Ack(n)
Ack Transnt
PACKET
if res=failure then 1`pack else empty
if res=success then 1`pack else if res = duplicate then 2`pack else empty
Transmit Ack
pack
Ack 2Trans
PACKET
if n=k then Ack(k+1) else Ack(k)
Next Rec
NO
if n=k then k+1 else k
k
Receive Packet
None
Modified protocol model

Functions

May be used in: guards, arc expressions and initial markings.

Expressions in industrial models may be complicated, therefore it is more convenient to define functions and then use their mnemonic names it the corresponding inscriptions.

www.agh.edu.pl



Simplified CPN – no failure in sending Ack

CPN Tools (Version 4.0.1, February 2015)

View
Help
Options
ColoursNet1_Fun_2.cpn
Step: 0
Time: 0
Options
History
Declarations
Standard priorities
Standard declarations
colset UNIT
colset BOOL
colset INT
colset INTINF
colset TIME
colset REAL
colset STRING
colset DATA = string;
colset NO=INT;
colset NOxDATA = product NO*DATA;
colset DATAPACK = record seq:NO * data:DATA;
colset ACKPACK = NO;
colset PACKET = union Data:DATAPACK + Ack:ACKPACK;
colset RESULT = with success | failure | duplicate;
var res: RESULT;
var k, n:NO;
var data, d:DATA;
var pack: PACKET;
fun UpdSeq (n,k) =
if n=k
then k+1
else k;
fun ResTrans (res,pack) =
if res =success then 1`pack
else
if res = duplicate then 2`pack
else empty;
fun FailTrans (res,pack) =
if res=failure then 1`pack
else empty;
Monitors

Net
Create
Sim
View

New Page

1`(1,"AGH ")++
1`(2,"University")++
1`(3," is")++
1`(4," the best!")

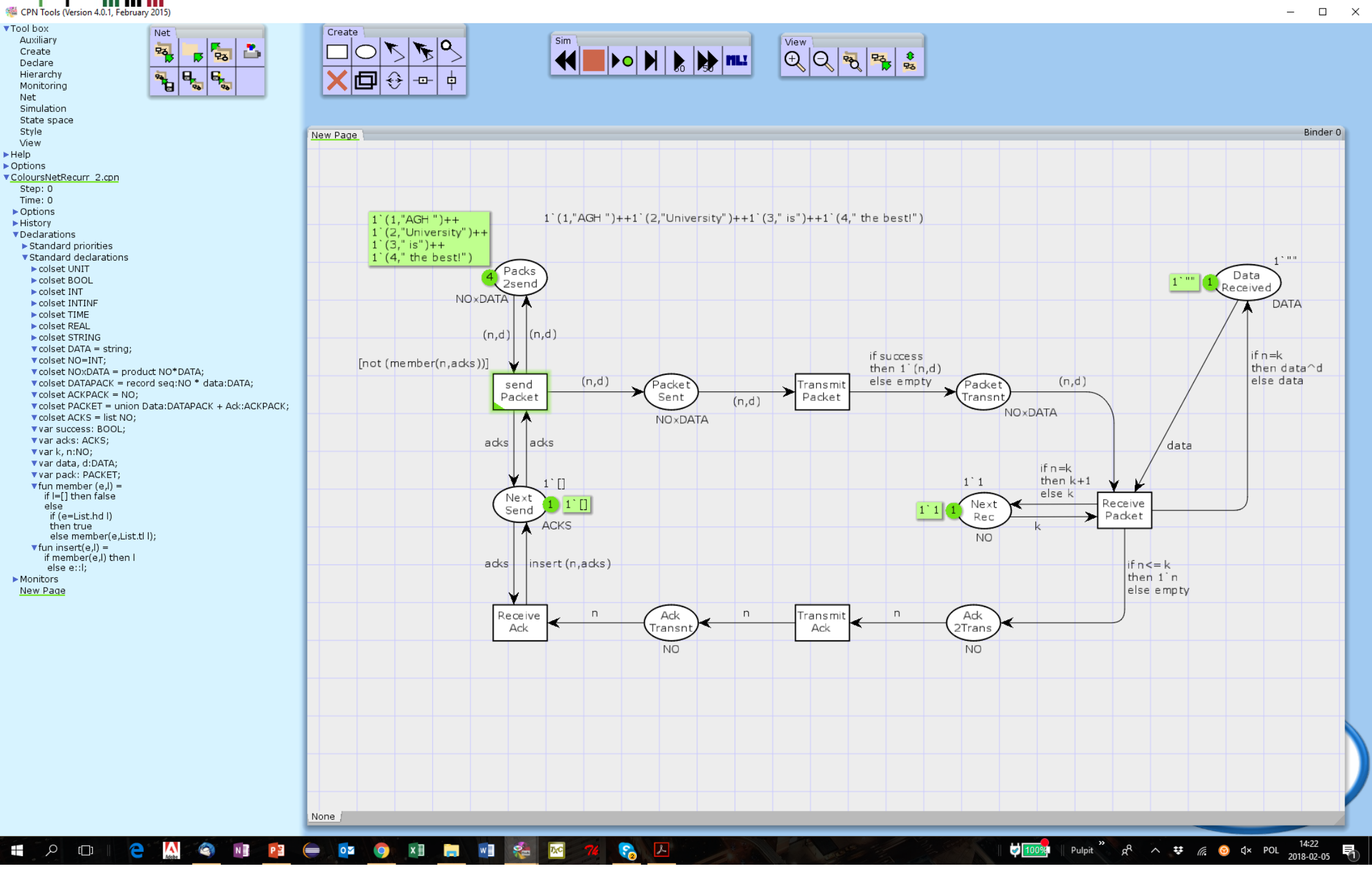
1`(1,"AGH ")++1`(2,"University")++1`(3," is")++1`(4," the best!")

Packs 2send
NOxDATA
(n,d)
send Packet
Data ({seq=n, data=d})
Packet Sent
PACKET
FailTrans (res,pack)
pack
Transmit Packet
ResTrans (res,pack)
Packet Transnt
PACKET
Data ({seq=n, data=d})
Data Received
1`""
DATA
if n=k then data^d else data
data
Next Send
1`1
NO
Next Rec
UpdSeq(n,k)
k
Receive Packet
UpdSeq(n,k)
Receive Ack
Ack Transnt
Transmit Ack
Ack 2Trans

None

Pulpit
POL
21:43
2018-02-02

Recursion and lists



Recursion and lists

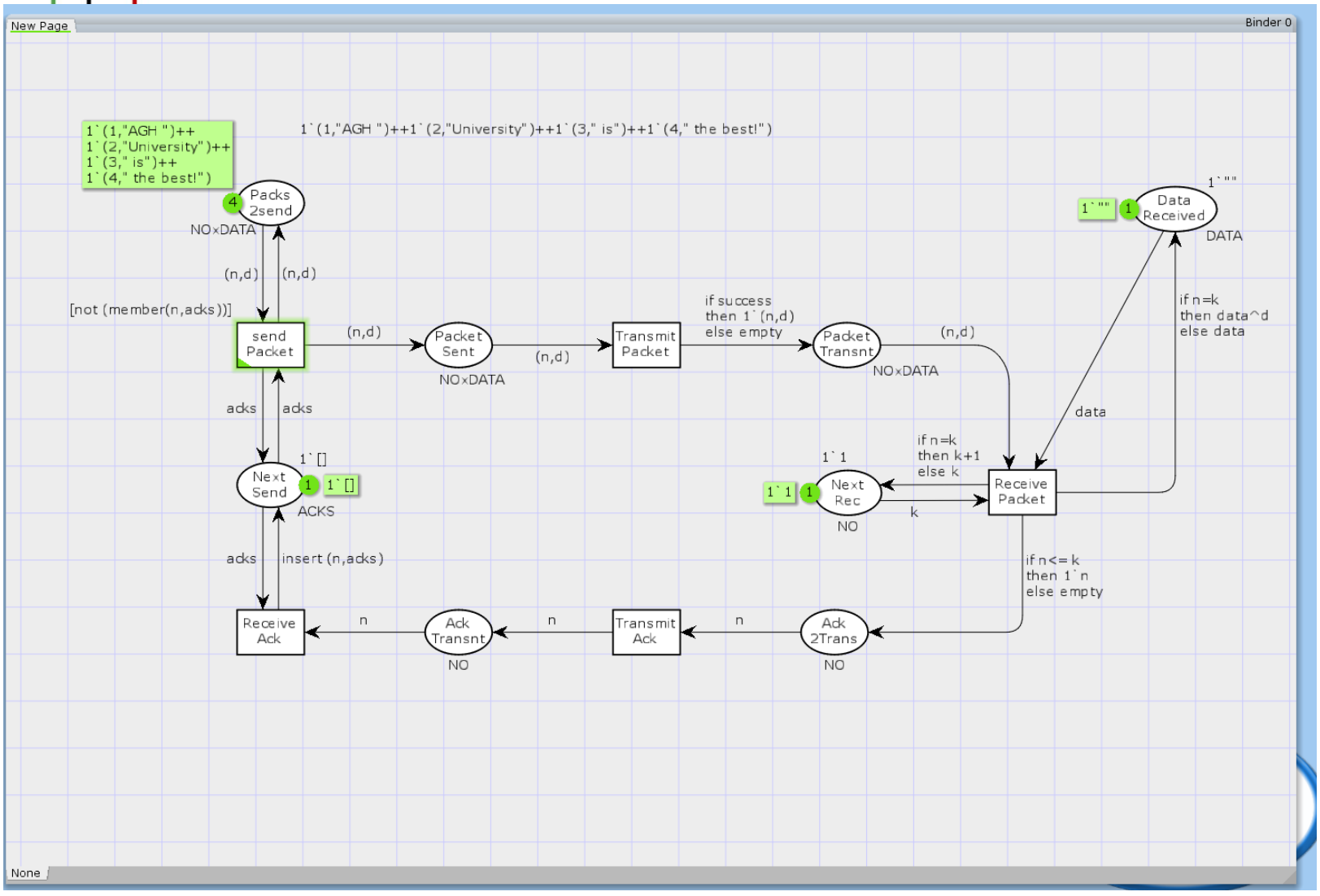
```
fun member(e,l) =
  if l = []  then false
  else
     if (e = List.hd l) then true
     else member(e, List.tl l);


fun insert(e,l) =
  if member(e,l) then l
  else e::l;
```

www.agh.edu.pl

Recursion and lists

```
fun member(e,l) =
  if l = []
  then false
  else
     if (e = List.hd l)
     then true
     else member(e, List.tl l);
```




```
fun insert(e,l) =
   if member(e,l)
   then l
   else e::l;
```

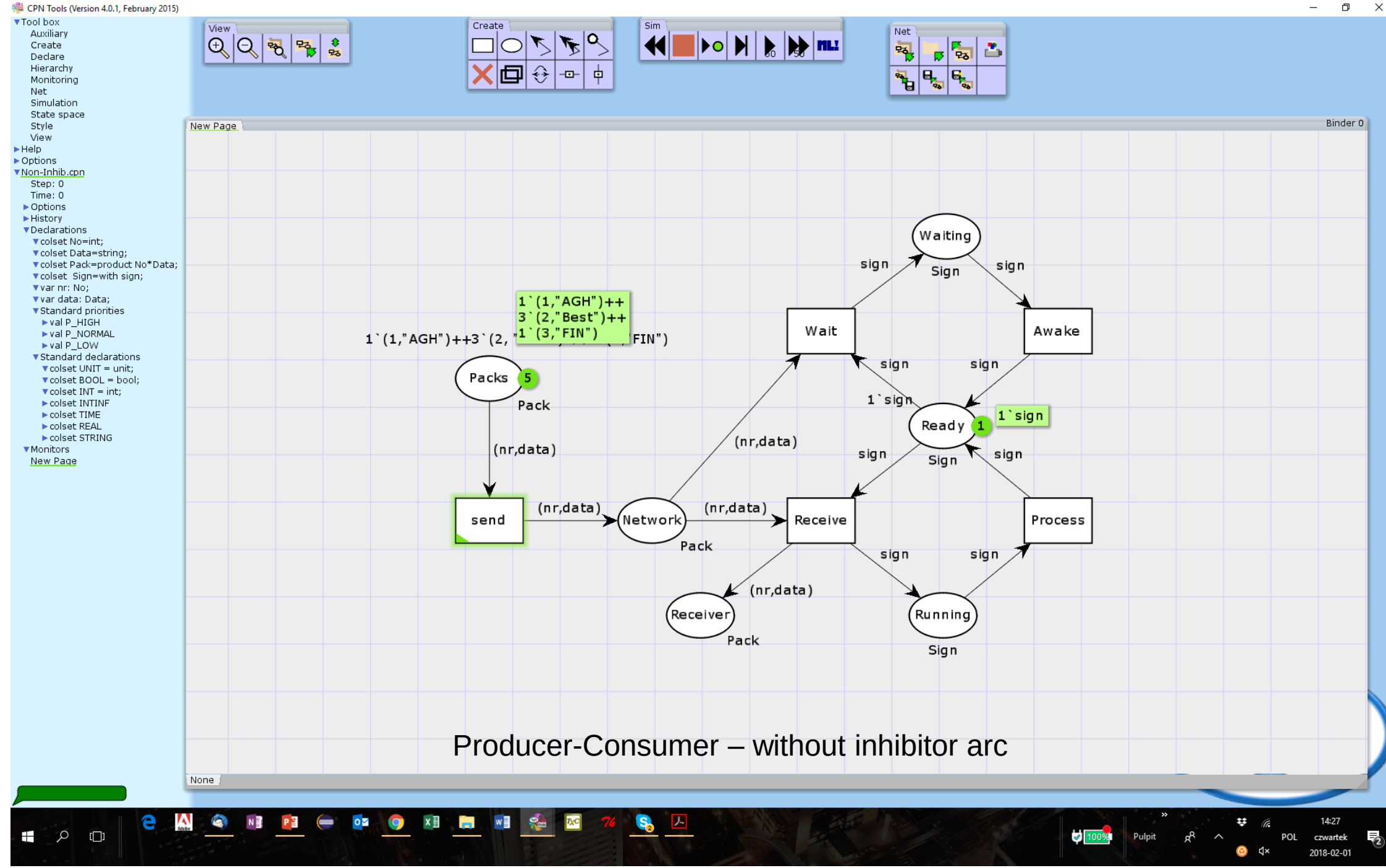
Recursion and lists

Another way for implementation of memeber function

```
fun member (e,[]) = false
    | member (e,x::l) =
      if (e = x)
      then true
      else member (e,l);
```

```
fun member (_,[]) = false //_ matches any value
    | member (e,x::l) =
      if (e = x)
      then true
      else member (e,l);
```

www.agh.edu.pl


CPN Tools (Version 4.0.1, February 2015)
Tool box
Auxiliary
Create
Declare
Hierarchy
Monitoring
Net
Simulation
State space
Style
View
Help
Options
Non-Inhib.cpn
Step: 0
Time: 0
Options
History
Declarations
colset No=int;
colset Data=string;
colset Pack=product No*Data;
colset Sign=with sign;
var nr: No;
var data: Data;
Standard priorities
val P_HIGH
val P_NORMAL
val P_LOW
Standard declarations
colset UNIT = unit;
colset BOOL = bool;
colset INT = int;
colset INTINF
colset TIME
colset REAL
colset STRING
Monitors
New Page
View
Create
Sim
Net
New Page
Binder 0
1`(1,"AGH")++
3`(2,"Best")++
1`(3,"FIN")
Waiting
Sign
sign
Wait
Awake
Packs
5
Pack
(nr,data)
1`sign
Ready
1
1`sign
send
Network
Receive
Process
Receiver
Running
Pack
Sign
None
Producer-Consumer – without inhibitor arc

Timed Coloured Petri Net - TCPN

Time models 1/2

Time extensions in „classical” Petri nets were introduced in several ways. The two direct possibilities are based on assigning some additional features to transitions or places correspondingly.

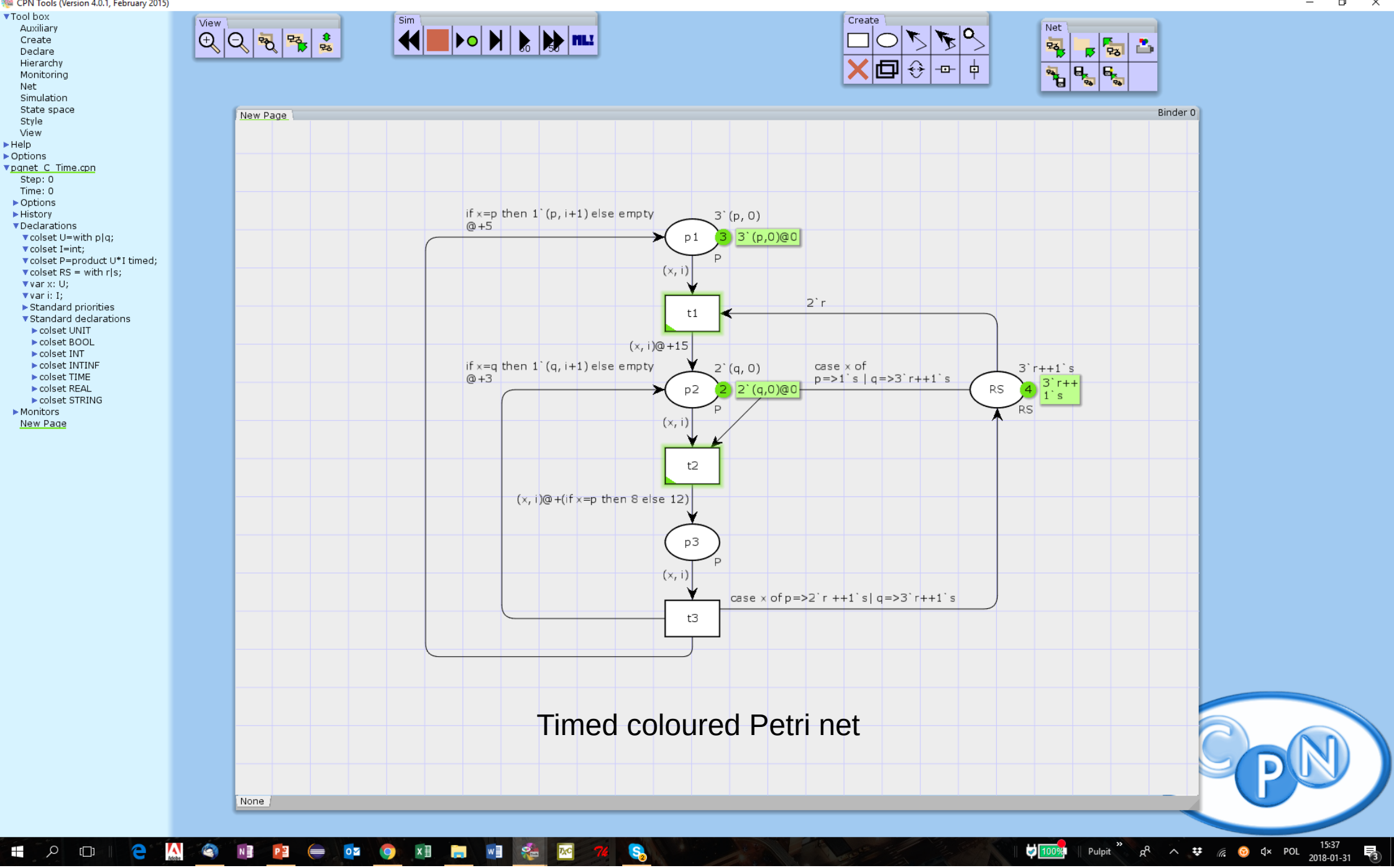
1. **Assigning of time to transition** is a natural way, because transition is interpreter as an operation, and therefore the time may represent time needed for execution of the operation. Execution of transition (change of markings) may be carried out after the specified period (of time) measured since the beginning of enabling state.
2. **Assigning time to place** interpreted as a minimal period of time when any token has to be located in the place. If for a given transition **many input places** exist, then the transition is enabled when **the longest time** in the related place is passed.

www.agh.edu.pl

Time models 2/2

3. Preparation time and execution time. Two times are assigned to every transition: **preparation time** and **execution time**. During execution markers from input places are absorbbed by the related transition.
4. Time interval assigned to transition. When $[a, b]$ interval is assigned, then enabled transition may be executed after **at least a time** units and **no later then after the time defined by b**.

www.agh.edu.pl


CPN Tools (Version 4.0.1, February 2015)
Tool box
Auxiliary
Create
Declare
Hierarchy
Monitoring
Net
Simulation
State space
Style
View
Help
Options
pqnet_C_Time.cpn
Step: 0
Time: 0
Options
History
Declarations
colset U=with p|q;
colset I=int;
colset P=product U*I timed;
colset RS = with r|s;
var x: U;
var i: I;
Standard priorities
Standard declarations
colset UNIT
colset BOOL
colset INT
colset INTINF
colset TIME
colset REAL
colset STRING
Monitors
New Page
View
Sim
Create
Net
New Page
Binder 0
if x=p then 1`(p, i+1) else empty
@+5
3`(p, 0)
p1
3
3`(p,0)@0
P
(x, i)
t1
2`r
(x, i)@+15
if x=q then 1`(q, i+1) else empty
@+3
2`(q, 0)
p2
2
2`(q,0)@0
P
case x of
p=>1`s | q=>3`r++1`s
RS
4
3`r++
1`s
RS
3`r++1`s
(x, i)
t2
(x, i)@+(if x=p then 8 else 12)
p3
P
(x, i)
case x of p=>2`r ++1`s| q=>3`r++1`s
t3
None
Timed coloured Petri net
Pulpit
POL
15:37
2018-01-31

Definition
Timed multiset tm over nonempty set S is a function $tm \in [S \times R \to N]$, such that sum:

$$tm(s) = \sum_{r \in R} tm(s, r)$$

is finite for every $s \in S$.

Non-negative value of $tm(s)$ is called number of occurrences of element s in finite multiset tm.

List $tm[s] = [r_1, r_2, \cdots, r_{tm(s)}]$ contains values of time $r \in R$, having non-zero values of coefficients $(tm)s) \neq 0$). Every r appears $tm(s, r)$ times in the list, which is ordered by time points, i.e. $r_i \leq r_{i+1}$ for every $i \in 1..tm(s) - 1$.

www.agh.edu.pl

Definition

Let us consider two increasingly ordered lists: $a = [a_1, a_2, \cdots, a_n]$ and $b = [b_1, b_2, \cdots, b_m]$.

We say, that $a \geq b$ iff $n \geq m$ and $a_i \leq b_i$ for every $i = 1,2, \cdots, m$.

If $a \geq b$ for a, b defined as above, difference $a - b$ is defined as below:

1. Length of the list is $n - m$;
2. Items of the list are defined by the algorithm:

for $j = 1,2, \cdots, m$ **do**
$r_j = max(a_i \leq b_j)$;
$a \coloneqq a \setminus \{r_j\}$, i.e. remove r_j from the list a ;
end.

$$r_j := \max_{a_i \in a} (a_i \leq b_j)$$

a = [57, 82, 103, 117, 134, 146]

b = [98, 136, 145]

c = [84, 138]

(a - b) = [57, 103, 146]

(a - b) – c = [57, 103, 146] – [84, 138] =
= [146]

www.agh.edu.pl

$$r_j := \max_{a_i \in a} (a_i \leq b_j)$$

a = [57, 82, 103, 117, 134, 146]

b = [98, 136, 145]

c = [84, 138]

(a-c) = [57, 103, 117, 146]

(a-c) – b = [57, 103, 117, 146]
– [98, 136, 145] = **[146]**

(a - b) – c = (a - c) – b = 146

www.agh.edu.pl

Definition

Comparison and subtraction of timed multisets is defined using operation described in the previous notions. For every two multisets $tm_1, tm_2 \in S_{TMS}$:

- $tm_1 \leq tm_2 = (\forall s \in S)\ tm_1[1] \leq tm_2[s]$
- If $tm_1 \leq tm_2$, then subtraction is defined as below:
  $tm_2 - tm_1 = \sum_{s \in \in S}\big(tm_2(s) - tm_1(s)\big)`s@(tm_2[s] - tm_1[s])\,.$

www.agh.edu.pl

Definition
Timed coloured Petri net is a triple $TCPN = (CPN, R, r_0)$, where:

1. CPN is defined as in the previous definition, including assumption that functions $E(a)$ and $I(p)$ are extended to timed multisets over $C(p(a))$ and $C(p)$ correspondingly
2. R is a set of time values, the so-called **time stamps**. R is a subset of positive real number containing 0 and closed w.r.t. + operation.
3. $r_0 \in R$ is **start time**.

www.agh.edu.pl

Definition

Marking of timed Petri net is a time multiset over TE. Initial marking M_0 is obtained by evaluation of initial expression:

$$(\forall p \in P)\ M_0(p) = I(p)_{r_0}.$$

Any pair (M, r) where M is marking and r time value is called **state** of timed Petri net. Initial state is denoted (M_0, r_0).

Definition

Step Y **is enabled in state** (M_1, r_1) and in **time** r_2, if the following conditions are satisfied:

1. $(\forall p \in P) \ \sum_{(t,b)\in Y} E(p,t) < b >_{r_2} \ \leq M_1(p)$
2. $r_1 \leq r_2$.
   r_2 is the smallest element in R for which there exists step Y satisfying the above mentioned (1 and 2) conditions.

Definition

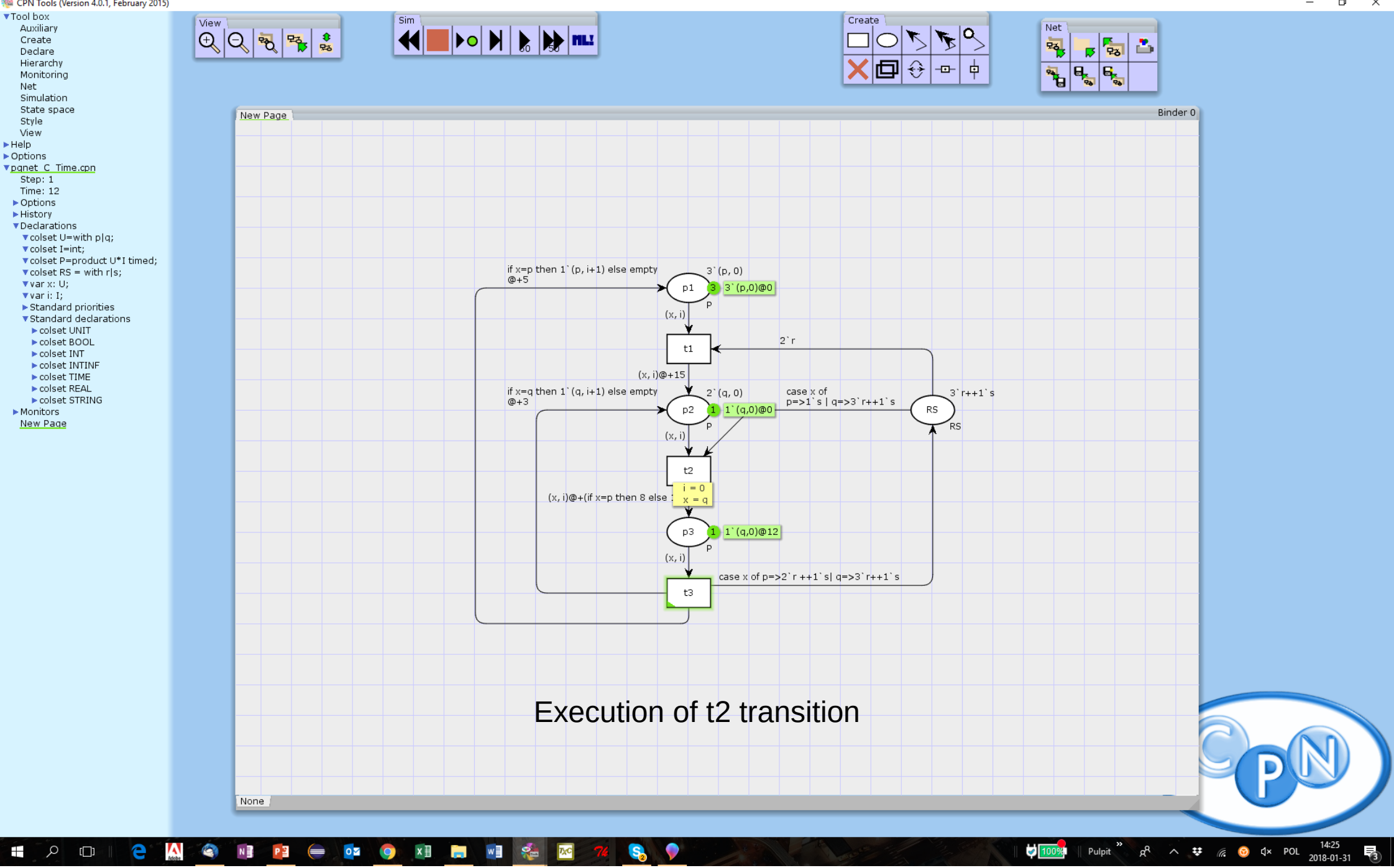
If step Y is enabled is state (M_1, r_1), then in time r_2 change of state (M_1, r_1) into (M_2, r_2) may appear, where M_2 is defined as below:

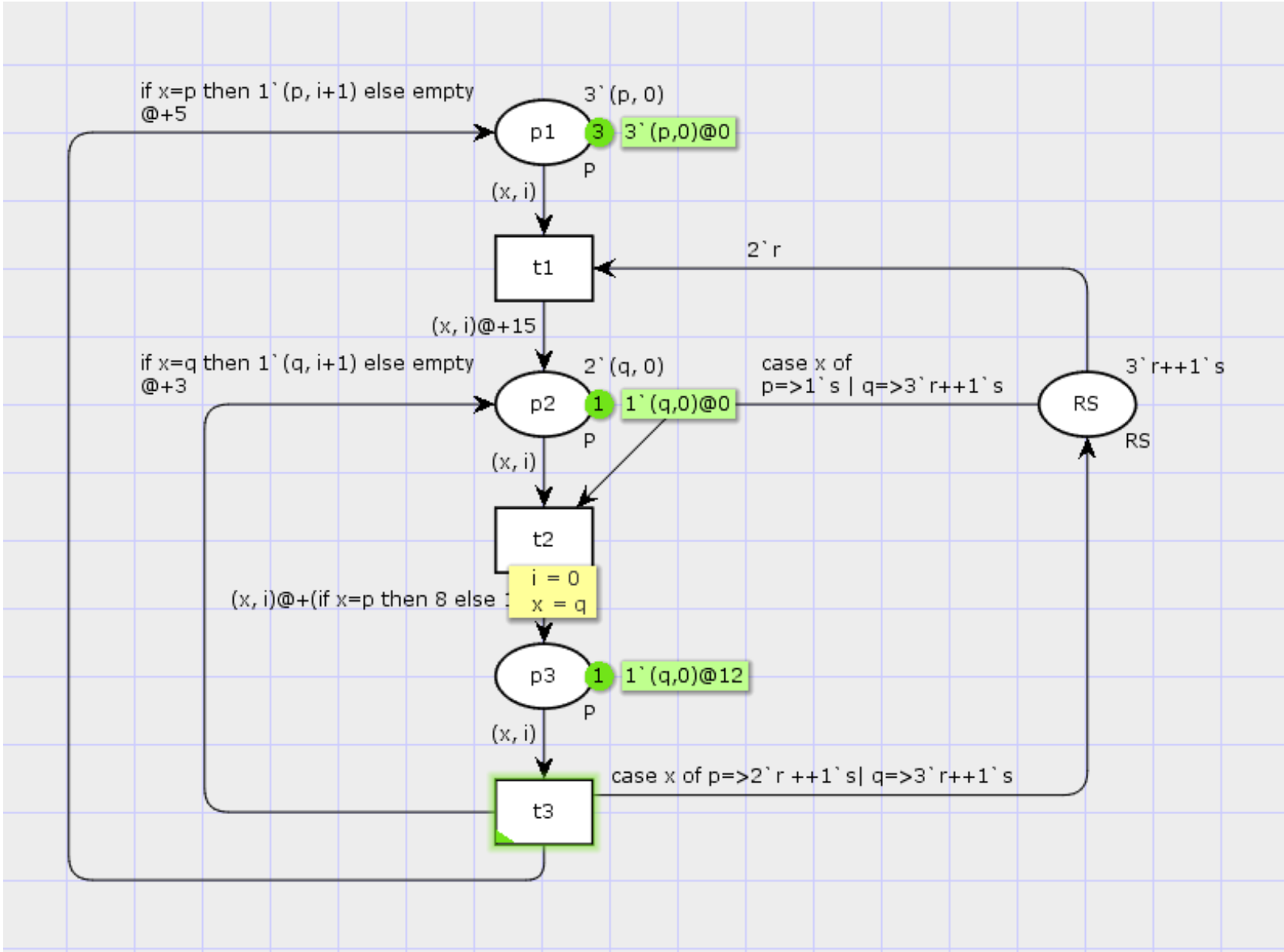
$$(\forall p \in P)\ M_2(p) = M_1(p) - \sum_{(t,b)\in Y} E(p,t) < b >_{r_2} + \sum_{(t,b)\in Y} E(t,p) < b >_{r_2}$$

State (M_2, r_2) is directly accessible from the state (M_1, r_1) by performing (execution) of step Y in time r_2. The fact is denoted:

$(M_1, r_1) \overset{Y, r_2}{\longrightarrow} (M_2, r_2)$.

www.agh.edu.pl


CPN Tools (Version 4.0.1, February 2015)
Tool box
Auxiliary
Create
Declare
Hierarchy
Monitoring
Net
Simulation
State space
Style
View
Help
Options
pqnet_C_Time.cpn
Step: 1
Time: 12
Options
History
Declarations
colset U=with p|q;
colset I=int;
colset P=product U*I timed;
colset RS = with r|s;
var x: U;
var i: I;
Standard priorities
Standard declarations
colset UNIT
colset BOOL
colset INT
colset INTINF
colset TIME
colset REAL
colset STRING
Monitors
New Page
View
Sim
Create
Net
New Page
Binder 0
if x=p then 1`(p, i+1) else empty
@+5
3`(p, 0)
p1
3`(p,0)@0
P
(x, i)
t1
2`r
(x, i)@+15
if x=q then 1`(q, i+1) else empty
@+3
2`(q, 0)
p2
1`(q,0)@0
P
case x of
p=>1`s | q=>3`r++1`s
RS
3`r++1`s
RS
(x, i)
t2
i = 0
x = q
(x, i)@+(if x=p then 8 else
p3
1`(q,0)@12
P
(x, i)
case x of p=>2`r ++1`s| q=>3`r++1`s
t3
None
Execution of t2 transition



State after execution of transition $t2$

www.agh.edu.pl

Example of modelling

Let us consider process simulating environment of lift controller, i.e. behaviour of lift motor and floor sensors.

Translation form SDL/UML2.X diagram into timed coloured Petri net.

www.agh.edu.pl

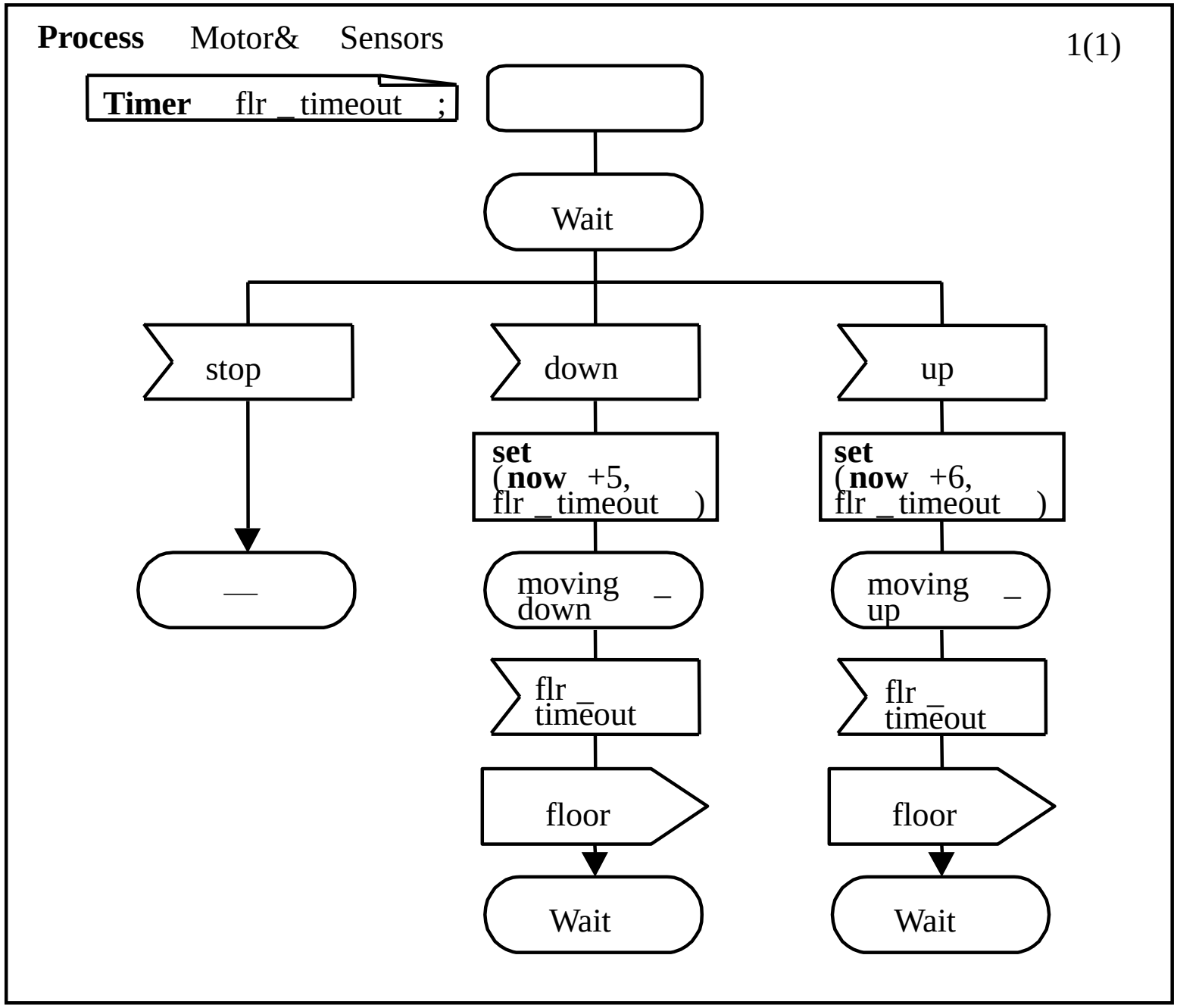
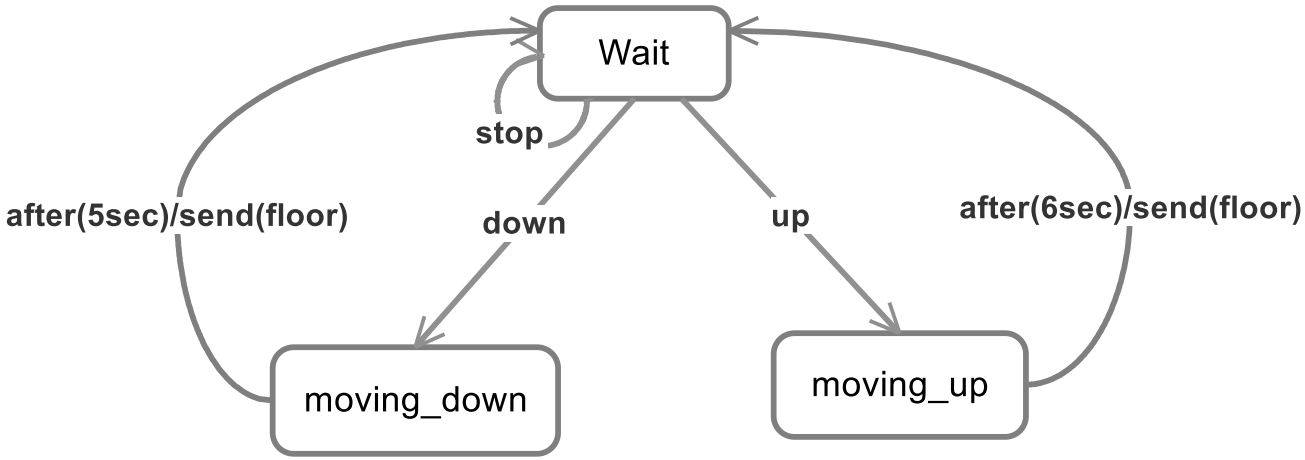
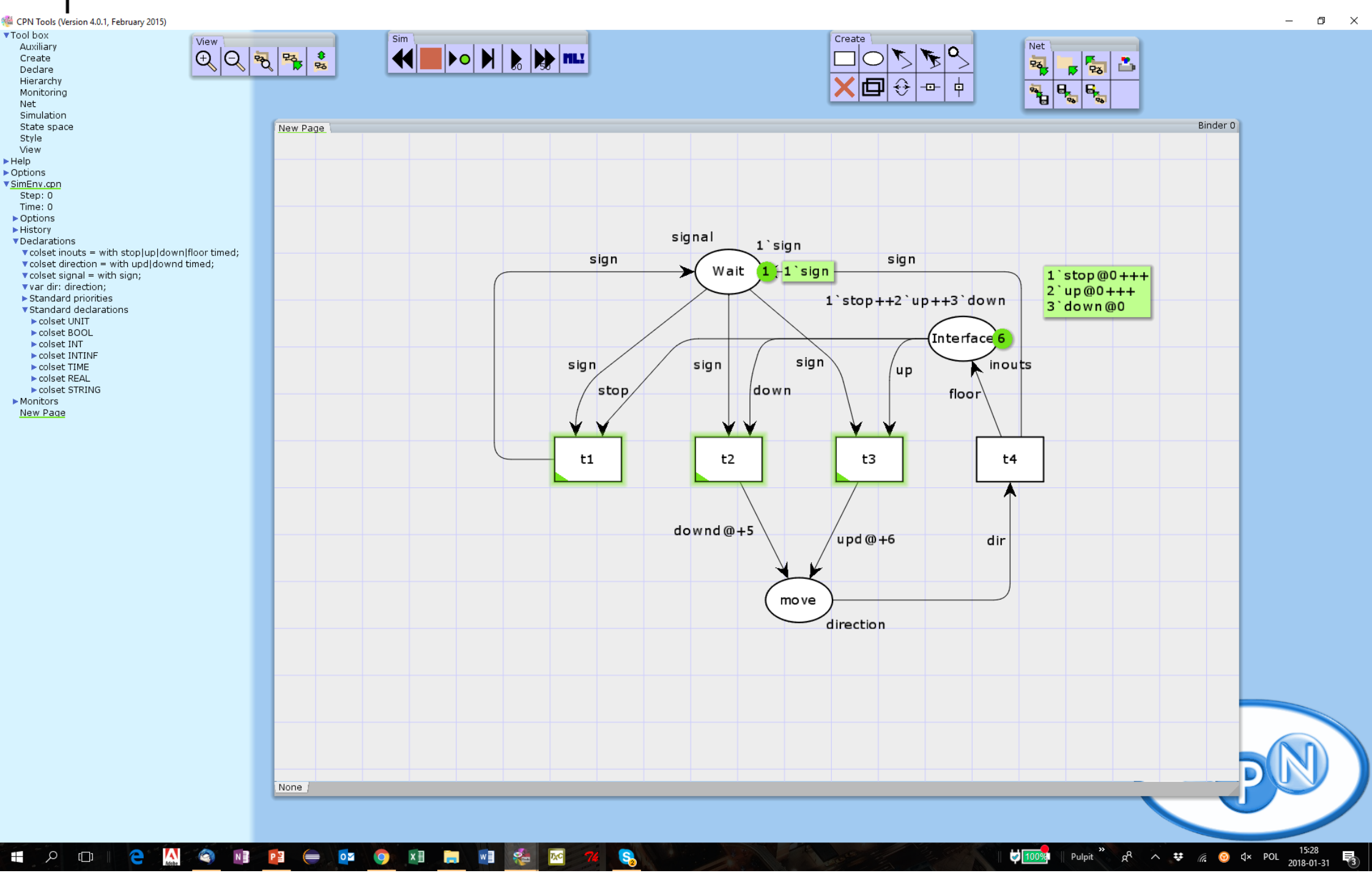


Diagram defined using SDL (Finite State Machine)

www.agh.edu.pl

Process Motor&Sensors – UML notation

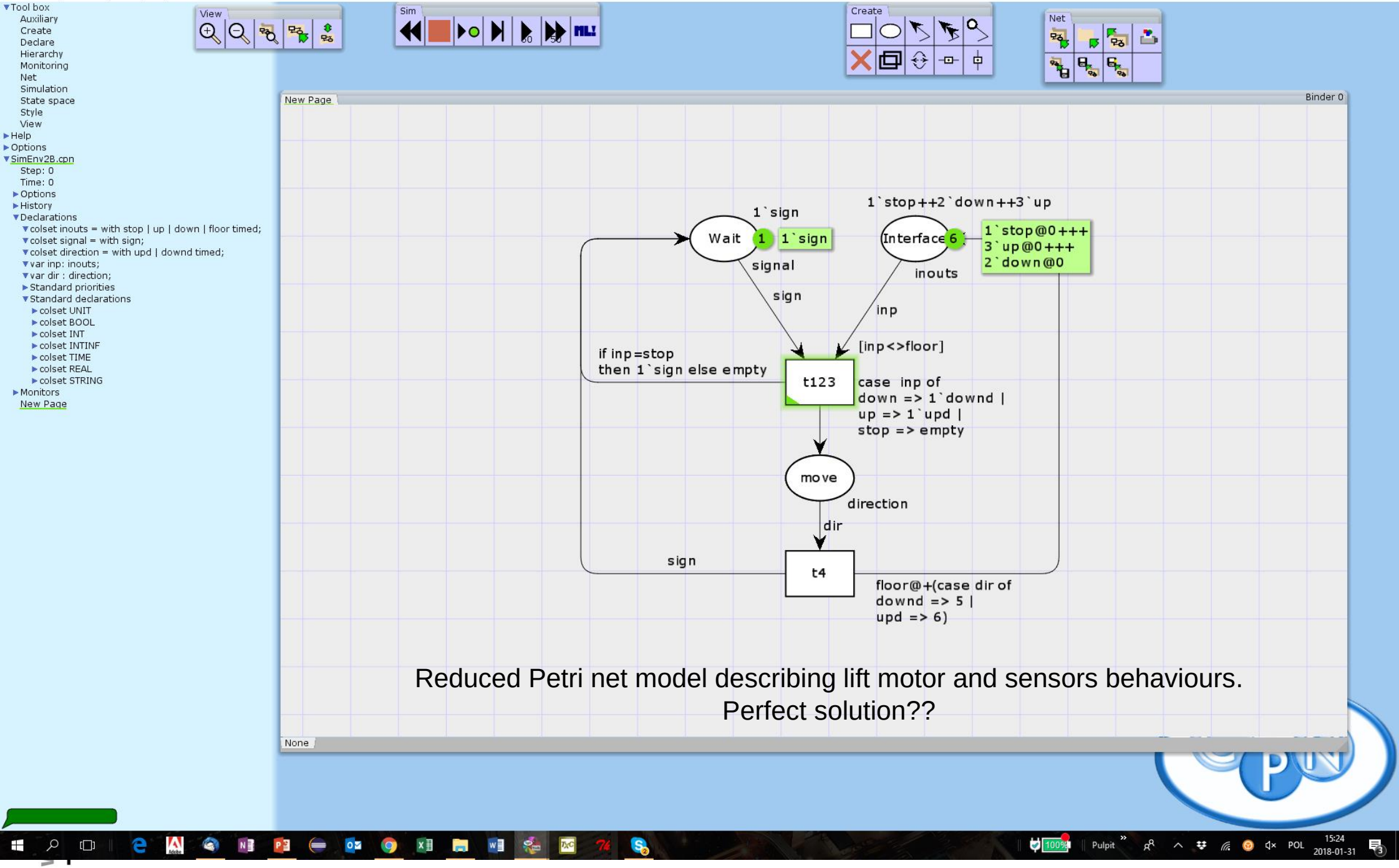


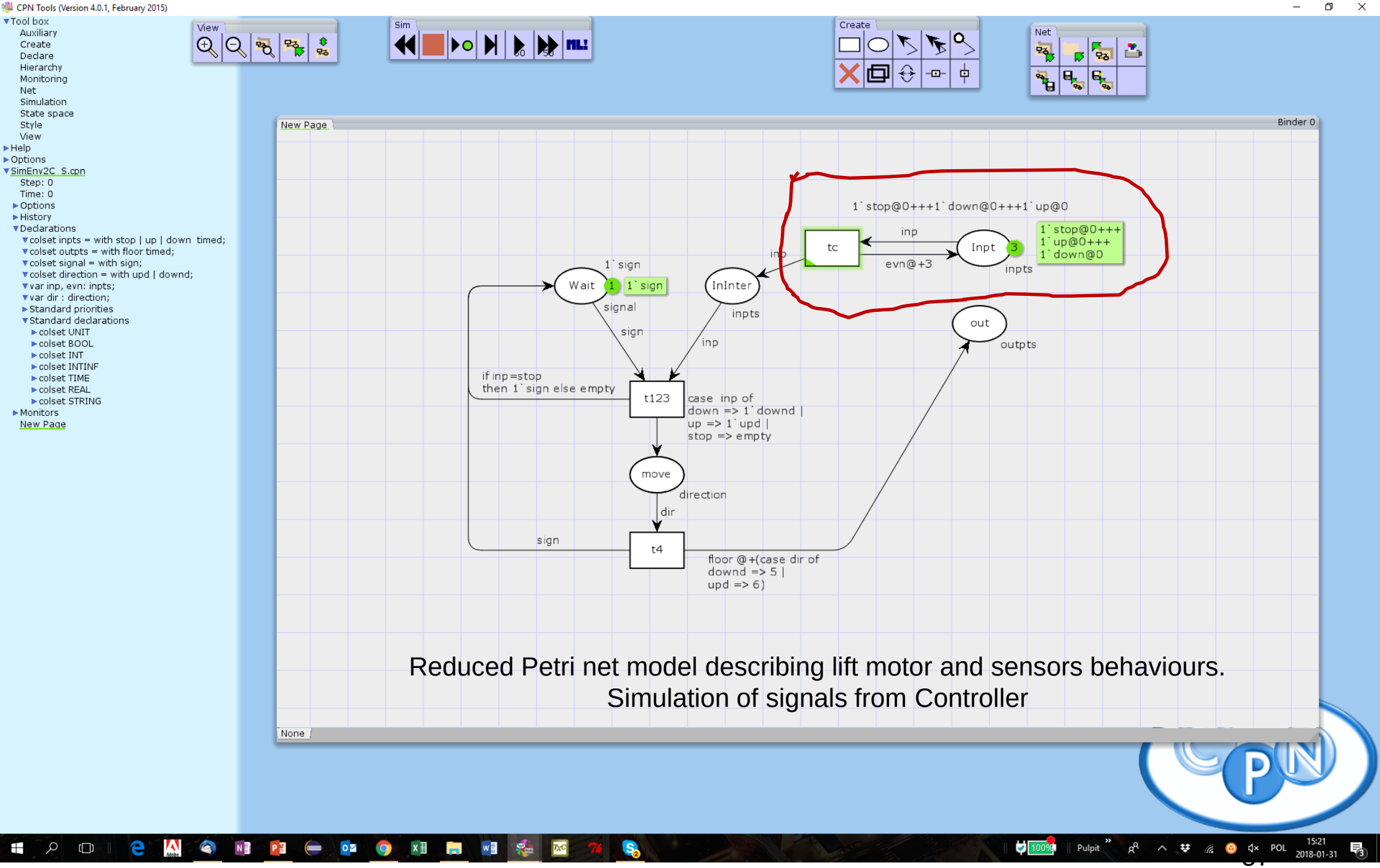

CPN Tools (Version 4.0.1, February 2015)
Tool box
Auxiliary
Create
Declare
Hierarchy
Monitoring
Net
Simulation
State space
Style
View
Help
Options
SimEnv.cpn
Step: 0
Time: 0
Options
History
Declarations
colset inouts = with stop|up|down|floor timed;
colset direction = with upd|downd timed;
colset signal = with sign;
var dir: direction;
Standard priorities
Standard declarations
colset UNIT
colset BOOL
colset INT
colset INTINF
colset TIME
colset REAL
colset STRING
Monitors
New Page
View
Sim
Create
Net
New Page
Binder 0
signal
1`sign
sign
Wait
1
1`sign
sign
1`stop++2`up++3`down
1`stop@0+++
2`up@0+++
3`down@0
Interface
6
sign
stop
sign
down
sign
up
inouts
floor
t1
t2
t3
t4
downd@+5
upd@+6
dir
move
direction
None

Translation rules

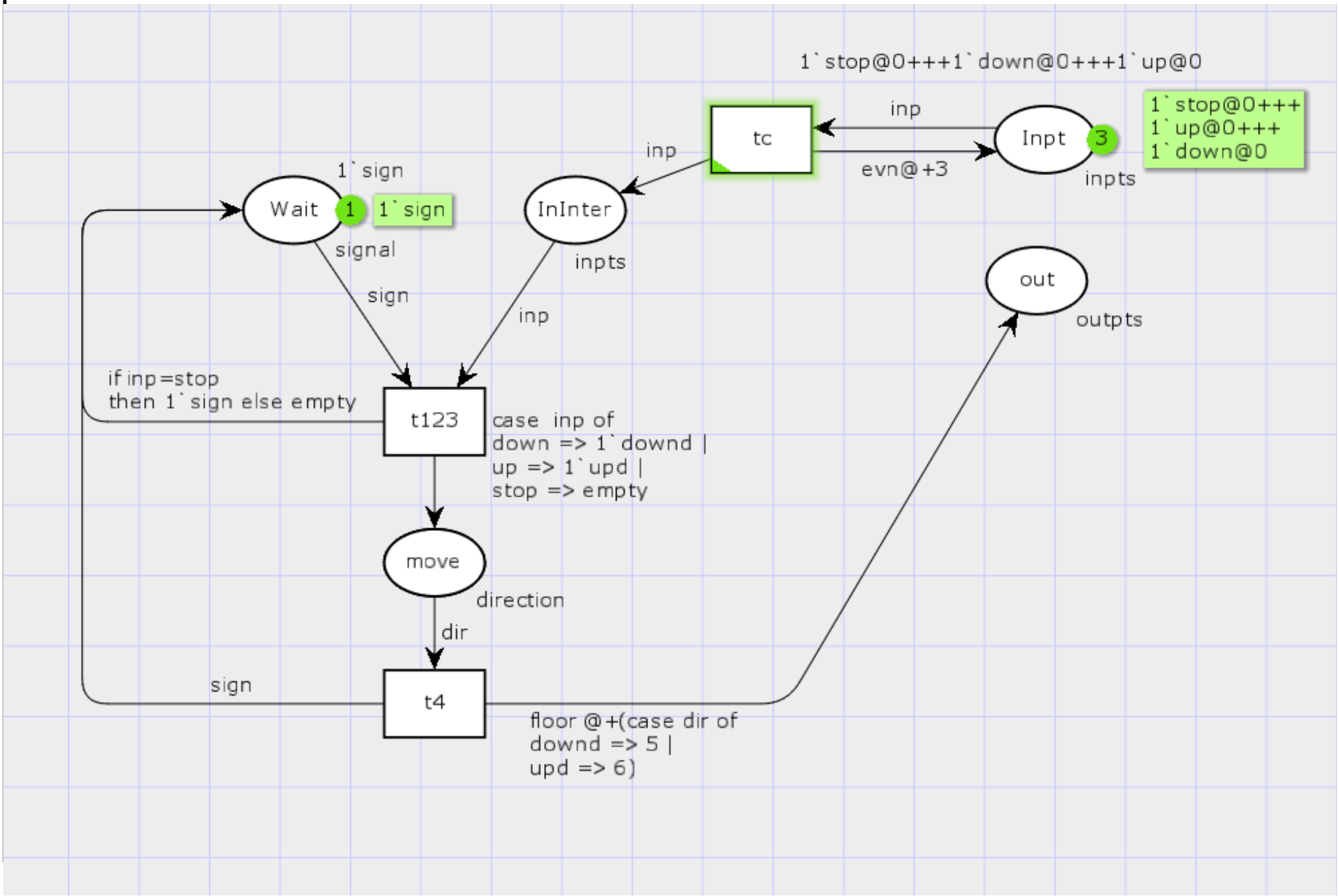
1. Assign of states into related Petri net places.
2. Define colours relevant to input and output signals.
3. Define interface (for input/output signals) place – here between the simulation model and controller (*Interface*).
4. Define transitions between the corresponding states.
5. Define (draw) input arcs for every transition taking into account order described input signals. Define additional colour(s) for synchronization ($sign$).
6. Define (draw) output arcs for every transition taking into account described order, synchronization and time requirements. In the later case output arcs are labelled additionally by the corresponding time stamps.
7. Reduce the constructed net, in particular by grouping homomorphic branches (see next slide).

www.agh.edu.pl


Tool box
Auxiliary
Create
Declare
Hierarchy
Monitoring
Net
Simulation
State space
Style
View
Help
Options
SimEnv2B.cpn
Step: 0
Time: 0
Options
History
Declarations
colset inouts = with stop | up | down | floor timed;
colset signal = with sign;
colset direction = with upd | downd timed;
var inp: inouts;
var dir : direction;
Standard priorities
Standard declarations
colset UNIT
colset BOOL
colset INT
colset INTINF
colset TIME
colset REAL
colset STRING
Monitors
New Page
View
Sim
Create
Net
New Page
Binder 0
1`sign
Wait
1
1`sign
signal
sign
1`stop++2`down++3`up
Interface
6
1`stop@0+++
3`up@0+++
2`down@0
inouts
inp
[inp<>floor]
if inp=stop
then 1`sign else empty
t123
case inp of
down => 1`downd |
up => 1`upd |
stop => empty
move
direction
dir
sign
t4
floor@+(case dir of
downd => 5 |
upd => 6)
Reduced Petri net model describing lift motor and sensors behaviours.
Perfect solution??
None



Reduced Petri net model describing lift motor and sensors behaviours.
Simulation of signals from Controller


1`stop@0+++1`down@0+++1`up@0
tc
inp
Inpt
3
inpts
1`stop@0+++
1`up@0+++
1`down@0
evn@+3
inp
1`sign
Wait
1
1`sign
InInter
signal
inpts
sign
inp
out
outpts
if inp=stop
then 1`sign else empty
t123
case inp of
down => 1`downd |
up => 1`upd |
stop => empty
move
direction
dir
sign
t4
floor @+(case dir of
downd => 5 |
upd => 6)