THE CONCEPTION OF CONCURRENT PETRI NET AND ITS SYNTH+ESIS

Henryk Piech, Wioletta Skibińska

Institute of Computer and Information Science, Czestochowa University of Technology, Poland
e-mail:hpiech@adm.pcz.czest.pl

Abstract. Petri net is form of bipartite graph. Schemes in form of Petri (PT) net permit modeling systems, objects, automata etc. Petri model becomes virtual prototype of represented system. Natural phenomena in PT nets is concurrent realized actions. It is guarantee by organization of fired transitions system. That are realized sequentially as singly or grouped procedures. In our approach we propose treat placements in standard connections with input and output system transitions. It is deeply form of concurrent because of unify structure of joining with all others placements. In this conception it's also possibility to fix sequence of fired transitions. Proposed concurrent PT net expand possibility of functional model dealing by invariant combinations of weights structures.

Introduction

Petri nets are used in concurrent modeling. With help of Petri net convention we can model distributed, real-time, operation systems [1-3]. Usually we support projecting process by algebraic equations [4-6]. Models are often complex hence their equivalent projecting mathematical analysis need complicated procedures [7, 8]. We try to unify Petri net structure combining input, output transitions with central node (placement). Such crated element has concurrent function structure and set of transition's parameters [9-11]. Prepared PT net scheme guarantee all possible combination of connection between placements and transitions. Prospectively, we want to elaborate system of organization optimal form and sequence of fired transitions. In present moment we should define transition activation process with preceding without supporting analysis. It is obviously far from optimal approach [12-15]. Additionally we propose algorithmic (but not algebraic) synthesis system. The weights matrix is results of synthesis process [16-18].

1. Definition of concurrent PT net and its parameters

Concurrent PT net is connected with transitions $t1, t2, \ldots, tn$ which are fired simultaneously (Fig. 1). Hence, we can depict them in form of one transition (low scheme in Figs 1 and 2). So every placement is joined with several transitions on upper scheme and with one transition on lower scheme. Obviously arcs have different weights: *w*1,*w*2,…,*wn*.

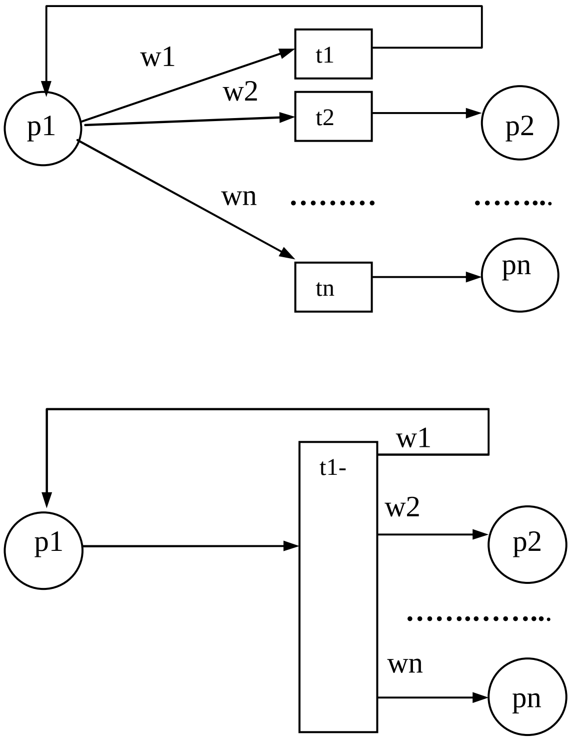

Fig. 1. Concurrent Petri net - conception of structure for decreasing number of tokens in p1

Proposed functioning of concurrent PT structure from Figure1 can be described as follows:

$$\begin{array}{ll} M'(p(out(t)) = M(p(out(\mathrm{t}))+w & \text{if } \; M(p(in(t)) \geq w \\ M'(p(out(t)) = M(p(out(\mathrm{t})) & \text{if } \; M(p(\mathrm{int}(t)) < w \\ M'(p(in\ (t)) = M(p(int(t))-w & \text{if } \; M(p(in(t)) \geq w \\ M'(p(in\ (t)) = M(p(int(t)) & \text{if } \; M(p(int(t)) < w \end{array}$$

where:
M(*p*) - number of tokens,
p(*out*(*t*)) - placement on output of *t* transition (successor),
p(*in*(*t*)) - placement on input of *t* transition (predecessor).
To supplement of full functioning structure performed in Figure 1 it's enough to change direction (arrows) on arcs.

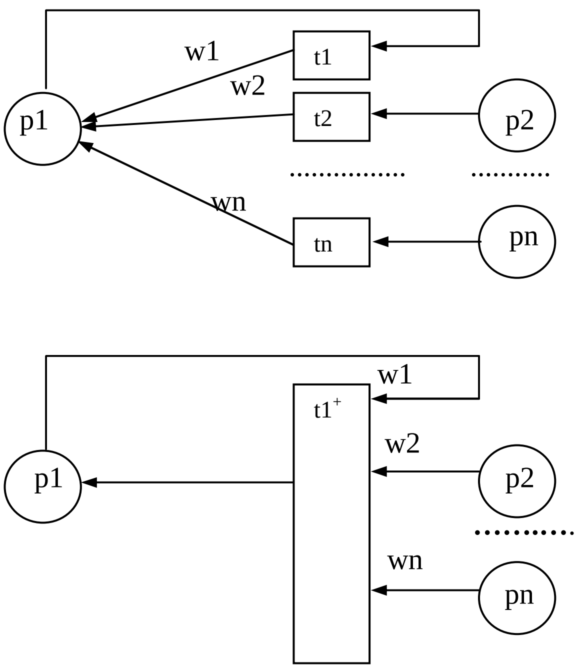

Fig. 2. Supplement of base structure for increasing number of tokens in p1

Joined structures is performed in Figure 3.

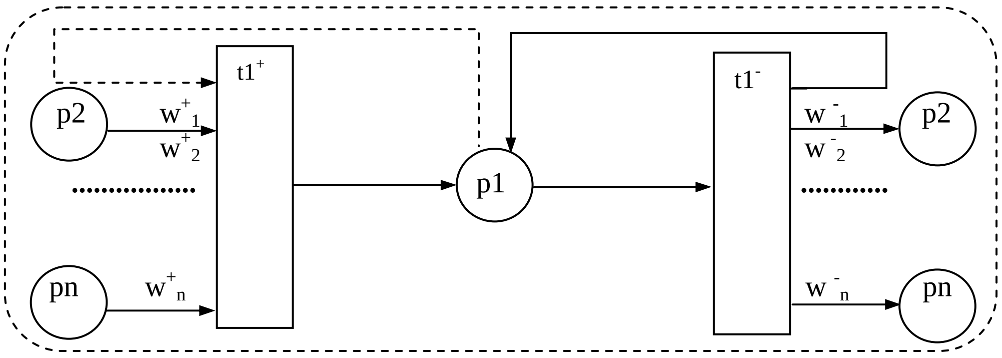

Fig. 3. Joined structures with excluding double feedback (doted line) - base cell

Generally we assume that number of tokens is positive and not fractional. In this case we can describe functioning process basing on both structures from Figures 1 and 2:

$$M'(p)=M'(p(out(t^{+})\ and\ p(int(t^{-}))= M(p(out(t^{+})\ and\ p(int(t^{-})) + \sum_{i=1}^{n} w_i^{+} - \sum_{i=1}^{n} w_i^{-}$$

$$\text{if } \left(\sum_{i=1}^{n} w_i^{+} - \sum_{i=1}^{n} w_i^{-} \right) \geq 0$$

$$M'(p(out(t^{+})\ and\ p(int(t^{-}))= M(p(out(t^{+})\ and\ p(int(t^{-}))$$

$$\text{if } \left(\sum_{i=1}^{n} w_i^{+} - \sum_{i=1}^{n} w_i^{-} \right) < 0 \qquad (1)$$

where: $w_i^{+}= w_i$ if $M(p(in(t^{+}))\geq w_i$ else $w_i^{+} = 0$
$w_i^{-}= w_i$ if $M(p(in(t^{-}))\geq w_i$ else $w_i^{-} = 0$

The full structure of concurrent PT net consist of *n* base cells (Fig. 4).

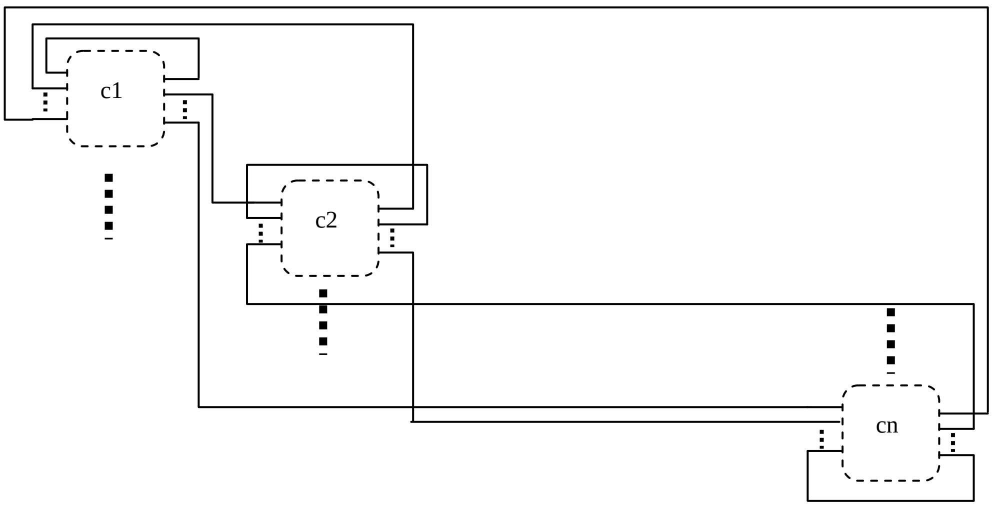

Fig. 4. Connections in concurrent PT net structure. It is conventional form because of cells overlapping in practice

Information about weights is contained in transition matrix T ($n \times n$) (there appears second index because of n cells):

$$T = \begin{pmatrix} w_{1,1} & w^{-}_{1,2} & \dots & w^{-}_{1,n} \\ w^{+}_{2,1} & w_{2,2} & \dots & w^{-}_{2,n} \\ \dots & \dots & \dots & \dots \\ w^{+}_{n,1} & w^{+}_{n,2} & \dots & w_{n,n} \end{pmatrix}$$

where:
$w^{+}_{i,j}$ - weight of predecessor *j* placement transition (from *i* placement),
$w^{-}_{i,j}$ - weight of successor *i* placement transition (to *j* placement).

In our conception we can fire simultaneously both transition ti^{+} and ti^{-} or only one of them. But we don't stay in position that it isn't allowed to fire more transitions at the same time, though it isn't consider in this work.
For defining token change in *i* placement after firing ti^{+} and ti^{-} we exploit *i* row and *i* column of matrix *T* according (1).

2. Assumption to synthesis of concurrent PT net

The main problem of synthesis consists in calculate all weights of matrix *T*. There we have n^2 variables. In point of view of area of searching solution (*s*) it is good information but in point of defining algorithm (or mathematical apparatus) it is more complicate. Let's look on initial data for synthesis net in assumption that we want to model automata functioning on base of several sequenced states: $St(k), k = 1,2,\ldots m$. Firstly, let's look on attributes with describe every state: $a(j), j = 1,2,\ldots,v$. We create matrix of attributes in sequenced states for which is added second index (regarding state number): $a_{j,k}$.

$$S = \begin{pmatrix} a_{1,1} & a_{1,2} & \ldots & a_{1,m} \\ a_{2,1} & a_{2,2} & \ldots & a_{2,m} \\ \multicolumn{4}{c}{\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots} \\ a_{v,1} & a_{v,2} & \ldots & a_{v,m} \end{pmatrix}$$

The assumption connected with matrix *S* consists in choosing set of placements (cells) with cardina*lity equals* $v(n = v)$*: token of every placement* (*cell*) *described one attribute:* $M^{(k)}(p_j) = a_{j,k}$. Every change of state follows after firing succeeding transition. Most of real automata return to initial set of attributes: $a(j,1) = a(j,m)$, $j = 1,2,\ldots,v$.

Next problem bases on question; in which way are involved variables: $w^{+}_{i,j}$ and $w^{-}_{i,j}$in ordered states. It can be showed in pictorial form in Figure 5. These form is adequate to table *T*. In succeeding transition firing are exploited previous (according sequence of cells connections) determined weights.

Next assumptions is connected with ranges of weights. For simplify, we use the same ranges for all weights, and integer their figure: $rw = [wl, wu]$. Hence, they will change from *wl* to *wu* by one. It has sense to assume *apriori* that low bound will be equal zero: $wl = 0(ru = [0,wu])$. Such result means that adequate connection is absent.

	$p1$	$p2$	$p3$	……	$p(v)$	$M(p)$
$p1$	▒	■	■	▒ ……	▒	$M(p1)$
$p2$	■	▨	■	▨ ……	▨	$M(p2)$
$p3$	■	■	▦	▦ ……	▦	$M(p3)$
……	▒ ……	▨ ……	▦ ……	……	……	……
$p(v)$	▒	▨	▦	……		$M(p(v))$

Fig. 5. Illustration of variables involving (common places - black fields) in process of crating solution about state attributes (after determining first three tokens: $M(p1)$, $M(p2)$, $M(p3)$)

The assumption according algorithmic organization of finding set of solutions consist in exploitation n*n cycles (loops):

```
for w[1,1]=0 to wu do
  for w[1,2]=0 to wu do
   ....................................
      for w[1,n]=0 to wu do
        for w[2,1]=0 to wu do
           for w[2,2]=0 to wu do
                .....................................
                   for w[2,n]=0 to wu do


                   ....................................
                      for w[n,1]=0 to wu do
                         for w[n,2]=0 to wu do
                            ....................................
                               for w[n,n]=0 to wu do
                                  begin
                                       Procedure Tokens_Calculation ($M^{(k)}(p_l)$);
                           ProcedureCompare_with_Model_Assumptions ($M^{(k)}(p_l) = a_{l,k}$)
                                 end;
```

Fig. 6. Algorithmic structure for organizing solution searchin.

The complexity of this algorithm is more then $O(wu^{n*n})=O(exp(n^2ln(n)))$ because procedures Tokens_Calculation and Compare_with_Model_Assumptions contain inner cycles (every of them).

3. Practice realization of synthesis concurrent PT nets

In algorithmic variant of synthesis realization we can obtain several equivalent solutions or none solution. It depend of number of states of modeled object (automata). Every new state of model functioning increases number of equations about $n = p$. This information is important in theoretical approach basing on resolving system of equations [ ]. Algorithmic approach can be generally performed in form presented in Figure 7. Notation “data of model states $a_{i,j}$ “refers to set of model-object attributes in all its states. The block “generation set of weights” means that in cyclic stages will be sequentially create new set of weights in different variations. Number of variations is equal wu^{n*n}. For token calculation we use formula (1). The block “comparing with given data” means that state by state will be checked all model’s attributes with set of current tokens. Full “agreement” means that for all states, all attributes fit to tokens. Process of tokens creation has sequential character because result from previous state becomes data to current tokens set definitions.

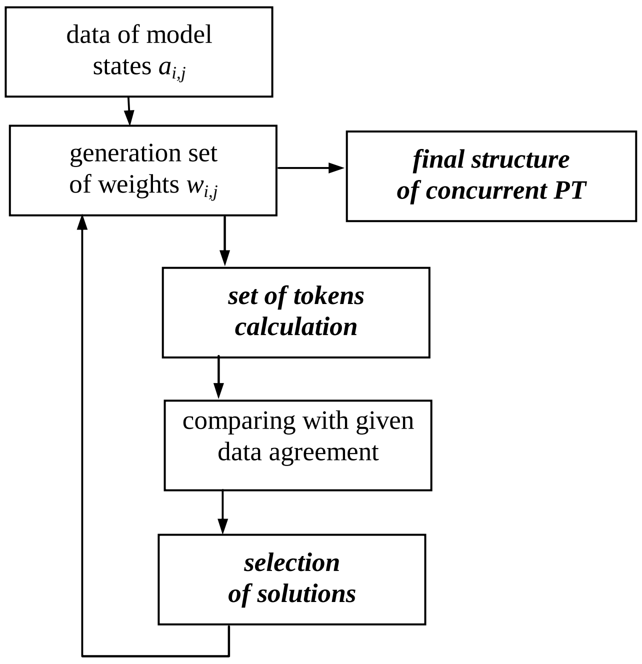
data of model states $a_{i,j}$
generation set of weights $w_{i,j}$
final structure of concurrent PT
set of tokens calculation
comparing with given data agreement
selection of solutions

Fig. 7. Idea scheme of concurrent PT net synthesis

In algorithmic approach we can easy overcame problems with solution (matrix of weights) searching. When appears such problem we can exploit two convention:

– increase range of weights ($wu = wu+d$, where $d\in\mathbf{N}$ and $d>0$),
– introduce new fiction placement (connected with fiction attribute of statement): $n = n+1(v = v+1)$.

Last convention is connected with adding inequation in not strong Figure: $M(p_{n+1}) \geq 0$. This remark has obviously irrelevant character because of its multivariant conceptions. Returning to algorithm organization it is based on combinatorial variation structure. We will present such composition in two variants:

1) basing on conversion into *wu*-nominal number system,
2) basing on generation full set of variations

In both cases we try shortly and compact performed algorithmic structure from Figure 6.

Let's start from first variant (Fig. 8):

```
for x:=0 to wu^{n*n}–1 do
 begin
   z:=x;
   for y:=1 to n*n do
     begin
       w[((y–1) div n)+1,((y–1) mod n)+1]:=z mod wu;
       z:=z div wu
     end;
       Procedure Tokens_Calculation ($M^{(k)}(p_l)$);
       Procedure Compare_with_Model_Assumptions ($M^{(k)}(p_l)=a_{l,k}$)
 end;
```

Fig. 8. Algorithmic structure for organizing solution searching (variant 1)

Description of algorithm start from organizing cycles for all numbers of possible situations connected with weights changing. During analysis every situation we use Euclid's approach systematically dividing n^2 times number x by *wu* transforming decimal x into *wu*-nominal x. Rest of division define particular weight value (z *mod* wu). The indices of weight $w[i,j]$ is defined as $i = ((y–1)$ *div* $n)+1$ and $j = ((y–1)$ *mod* $n)+1$. For every defined situation are provided token creation according (1): **Procedure** Tokens_Calculation ($M^{(k)}(p_l)$. After then are checked agreement with model states attributes: **Procedure** Compare_with_Model_Assumptions ($M^{(k)}(p_l)=a_{1,k}$).

In second variant (Fig. 9) is created set of variations. It is connected with increasing last weight element until it don't exceed upper bound. After it takes place all next elements (with parameter *y*) will zero out. Simultaneously previous element (with parameter *z*) is increased under condition that it is less then upper bound.

Let's start from zero up all weights elements. It is assumed that last weight elements is changed most quickly ($x=n*n$). When series of last elements are equal *wu* sequentially is exploited jump label "leb". When all weighs achieve level *wu* ($i:=wu^{n*n}$) then it is the last state for analyzing tokens in reference to given model attributes. This variant is more complex because contains additional cycle with parameter *y*.

Realizing procedure Tokens_Calculation ($M^{(k)}(p_l)$) we have to check *m* states of tokens when the previous state is data base for next state (Fig. 10).

```
    for x:=1 to n*n do
   w[((x–1) div n)+1,((x–1) mod n)+1] =0;
x:=n*n; z:=x;
for i:=1 to wu^(n*n) do
begin
 if w[((x–1) div n)+1,((x–1) mod n)+1] +1 ≤wu
  then w[((x–1) div n)+1,((x–1) mod n)+1]:= w[((x–1) div n)+1,((x–1) mod n)+1]+1
  else
   begin
    leb: z:=z–1;
    if w[((z–1) div n)+1,((z–1) mod n)+1] +1 ≤wu)
     then
      begin
       w[((z–1) div n)+1,((z–1) mod n)+1]:= w[((z–1) div n)+1,((z–1) mod n)+1]+1
            for y:=x+1 to n*n do w[((y–1) div n)+1,((y–1) mod n)+1]:=0;
          end
        else goto leb;
      end;
    Procedure Tokens_Calculation (M^(k)(p_l));
    Procedure Compare_with_Model_Assumptions (M^(k)(p_l)=a_(l,k))
    end;
```

Fig. 9. Algorithmic structure for organizing solution searching (variant 2)

```
    for j:=1 to n do
    M[j,1]:=a[j,1];
    for i:=1 to m–1 do
      for j:=1 to n do
     M[j,i+1]:=M[j,i];
        for k:=1 to n do
          begin
       if M[k,i]≥ w[k,j] then u[k,j]:=w[k,j] else u[k,j]:=0;
       if M[j,i ]≥ w[j,k] then u[j,k]:=w[j,k] else u[j,k]:=0;
           M[j,i+1]:=M[j,i]+u[k,j]–u[j,k]
          end
```

where: i - number of state,
j - number of placement,
k - number of ingredient weight,
M[j,i] - value of token,
a[j,i] - value of attribute.

Fig. 10. Algorithmic structure for calculation tokens values

Procedure Compare_with_Model_Assumptions ($M^{(k)}(p_l)=a_{l,k}$) is very simple too (Fig. 11).

```
for i:=2 to m do
  for j:=1 to n do
    if  M[j,i] ≠ a[j,i] then go to neg
"agreement - there are found adequate set of weights";
…………………
neg: "not agreement- there aren't found adequate set of weights";
```

Fig. 11. Algorithmic structure for checking attributes. All transitions are fired in particular states of attributes

Usually we obtain more then one solution - set of weights fulfilling "agreement" conditions. In this case we chose set with the minimal number of connections (with maximal number of $w_{i,j}$ ***= 0) and minimal total sum of tokens ("final structure of concurrent PT" in Fig. 7).***

4. Applying the method of concurrent PT net synthesis in modeling process

Ten attributes of 5 states of given object are presented in Figure 12. In the example transitions are fired pairwise (ti^+, ti^-) for every central placement (see Fig. 2) and sequentially from 1 to n cell.

i	a(i,0)	a(i,2)	a(i,3)	a(i,4)	a(i,5)
1	4	3	2	1	0
2	1	3	5	7	9
3	2	1	0	0	0
4	9	6	3	0	0
5	2	4	6	8	10
6	5	7	9	11	13
7	4	2	1	0	0
8	5	3	1	0	0
9	7	8	9	10	11
10	3	5	5	7	9

Fig. 12. Object attributes in all states

The results in form of weights matrix are put in table in Figure 13.

	p1	p2	p3	p4	p5	p6	p7	p8	p9	p10
p1	0	2	0	0	0	2	0	0	0	0
p2	0	0	1	0	1	0	0	0	1	0
p3	2	0	0	1	0	1	0	0	0	0
p4	0	0	0	0	2	0	0	0	0	3
p5	0	2	1	0	0	0	1	0	0	0
p6	1	0	0	0	0	0	0	0	2	0
p7	0	0	0	0	0	2	0	1	0	1
p8	0	1	1	0	0	0	1	0	0	0
p9	0	0	0	0	3	0	0	0	0	1
p10	0	0	0	1	0	0	0	0	2	0

Fig. 13. Table of weights

According Figure 14 we build final structure of concurrent Petri net.

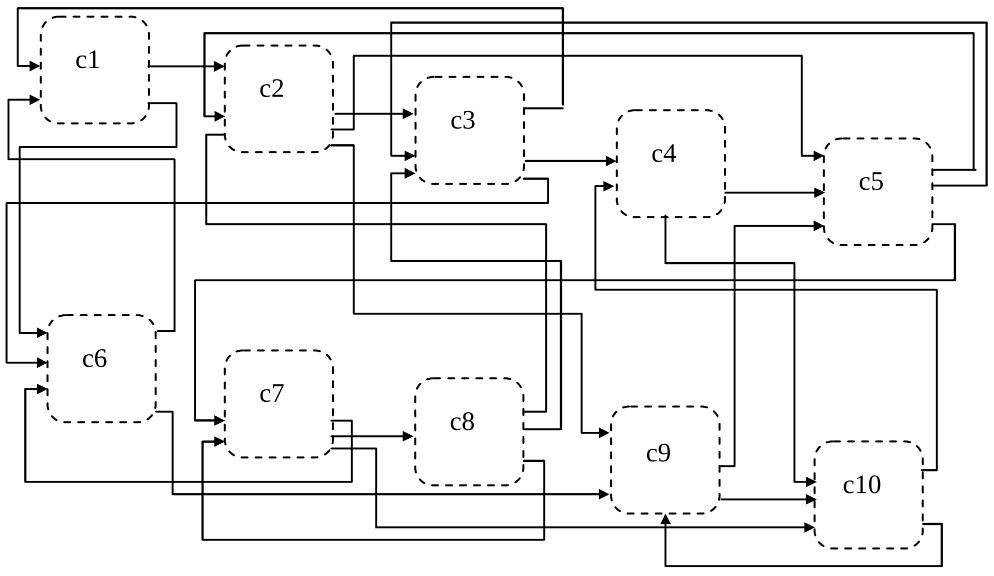

Fig. 14. Final structure of concurrent PT net - example

The sequences of fired cells be different. The form of fired transitions can has single of grouped character (as in our example) according to particular object states. It obviously influents on procedure Tokens_Calculation.

Conclusions

Concurrent PT net permit on comfortable algorithmic synthesis structure modeling object with given set of states. We can realized this process in different way in depend on chronology and character of fired transitions. Hence we obtain different result - set of weights. The structure of algorithm in Figure 8 suggests the possibility to realize organizing process in parallel variant dividing range [1, wu^{n*n}] into possessed number of processors.

References

[1] Cortadella J., Jakovlev A., Rosenberg G., Concurrency and hardware design: Advances in Petri nets, Springer-Verlag, New York 2002, 2549.

[2] Dadda L., The synthesis of Petri nets for controlling purposes and the reduction of their complexity, Euromicro, 2002.

[3] Murata T., Petri Nets: properties, analysis, and applications, IEEE 1989, 77, 4, 541-580.

[4] Berthelot G., Transformations and decompositions of nets, Advanced in Petri nets, Springer-Verlag, London 1987, 250, 359-377.

[5] Chiola G., On the structural and behavioral characterization of P/T nets, International Workshop in Petri nets and Performance Model, Toulouse 1993, 66-75.

[6] Jensen K., Rozenberg G., High-level Petri Nets - theory and application, Springer-Verlag, Berlin 1991.

[7] Szpyrka M., Fast and flexible modeling of real-time systems with RTCP- nets, Computer Science 2004, 81-94.

[8] Zuberek W.M., Timed Petri nets, definitions, properties, and applications, Microelectronics and Reliability 1991, 31, 4, 627-644.

[9] Bowden F.D.J., Modeling time in Petri nets, Workshop on Stochastic Models in Engineering, ACM Press, New York 1994, 228-239.

[10] Commoner F., Deadlocks in Petri Nets. Applied Data Research Inc., Wakefield, 1972.

[11] Marsan M.A., Balbo G., Conte G., Donatelli S., Franceschinis G., Modeling with generalized stochastic Petri Nets, John Wiley and Sons, New York 1995

[12] Cerone A., Maggiolo-Schettini A., Time based expressivity of time Petri nets for system specification, Theoretical Computer Science 1999, 216, 1-53.

[13] Hollyday M.A., Vernon M.K., A Generalized timed Petri model for performance analysis, IEEE, Transaction of Software Engineering 1987, SE-13, 12, 1297-1310.

[14] Peterson J.L., Petri net theory and the modeling of systems, Prentice Hall, New York 1981.

[15] Petri C.A., Advanced Course on General Net Theory of Processes and Systems, Springer-Verlag, London 1979.

[16] Memmi G., Vautherin J., Analyzing nets in invariant method, Advanced in Petri nets, Springer-Verlag, London 1987, 300-336.

[17] Valmari A., Petri Net Newsletter 1994, 46, 6-14.

[18] Yakovlev A., Gomes L., Hardware designed Petri nets, Kluwer Academic, Publishers, Norwell, US 2000.

[19] Samolej S., Szmuc T., Time extensions of Petri nets for modeling and verification of hard real-time systems, Computer Science 2002, 55-76.