Two modeling methods for signaling pathways with multiple signals using UPPAAL

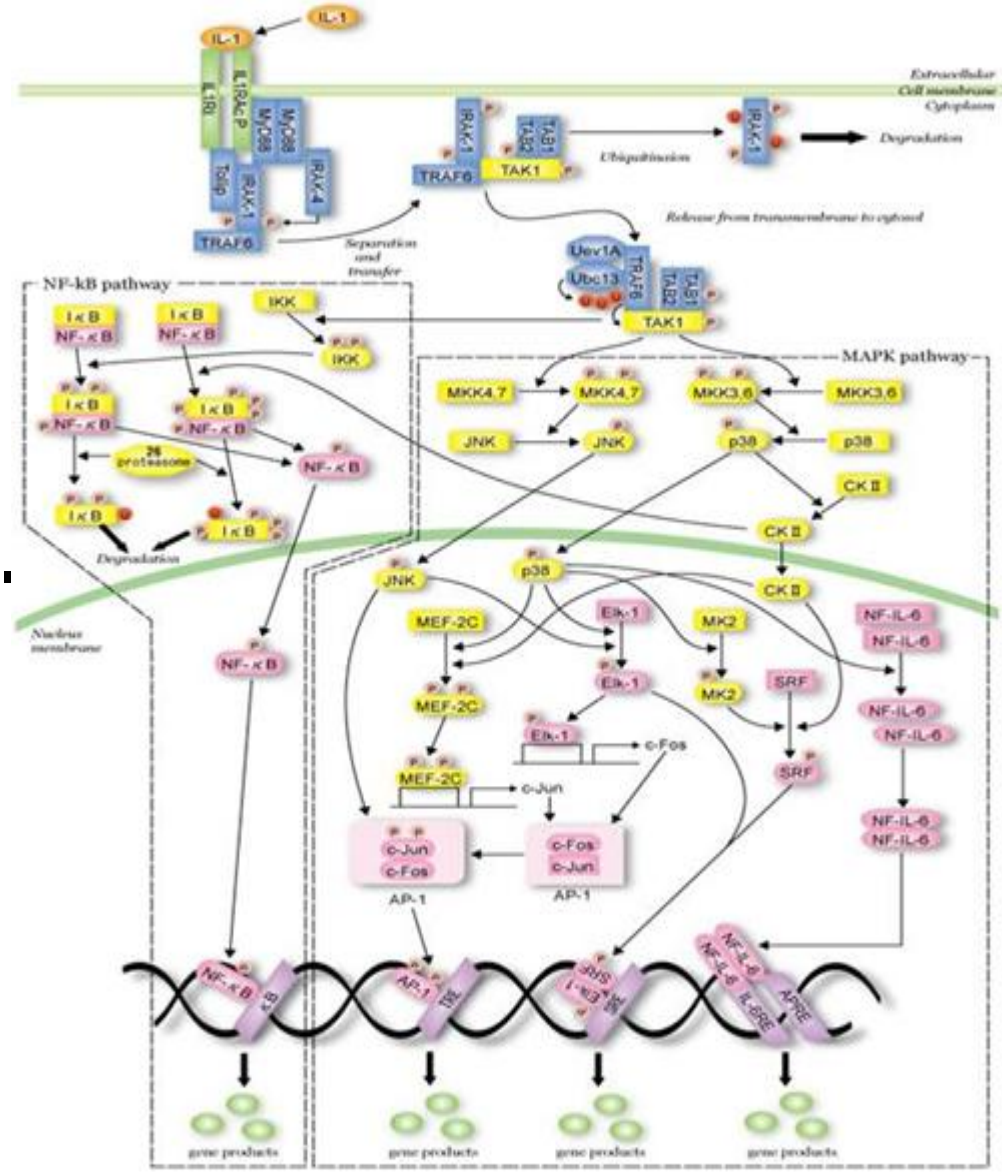
Shota Nakano, Shingo Yamaguchi
Graduate School of Science and Engineering,
Yamaguchi University, Japan

Contents

How do we model signaling pathways with multiple signals?
Multiple automata VS
Single automaton with variables

1. Introduction

Signaling pathway is a signaling mechanism.
→large and complex



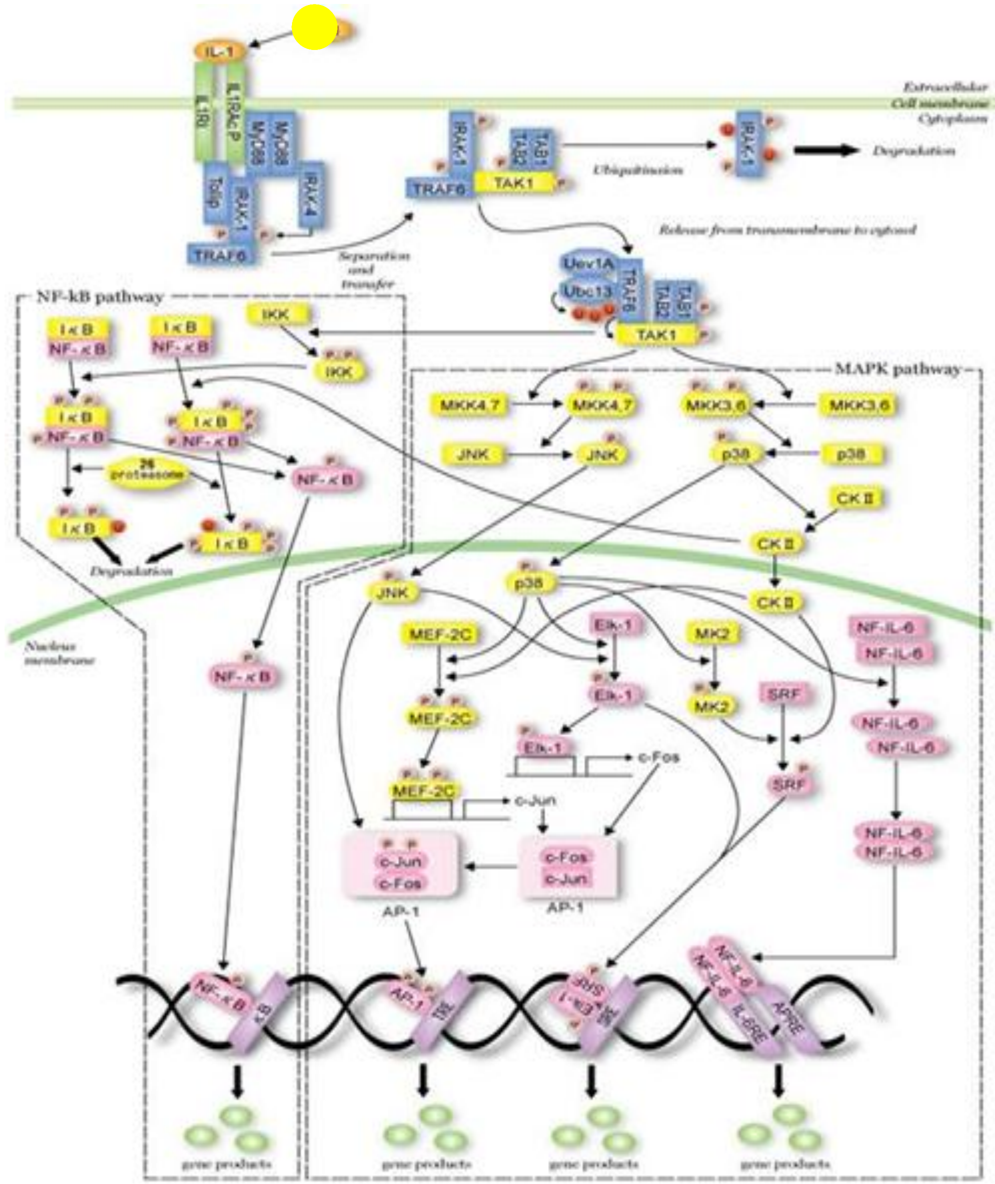
Some researchers have applied model checking.
Model checking is
- Automatic
- Quite fast

1．2 What is “multiple signals”?

A ligand joins a receptor repeatedly.

There are two or more, i.e. multiple, signals flowing in a signaling pathway.

We analyze the signaling pathway more precisely.



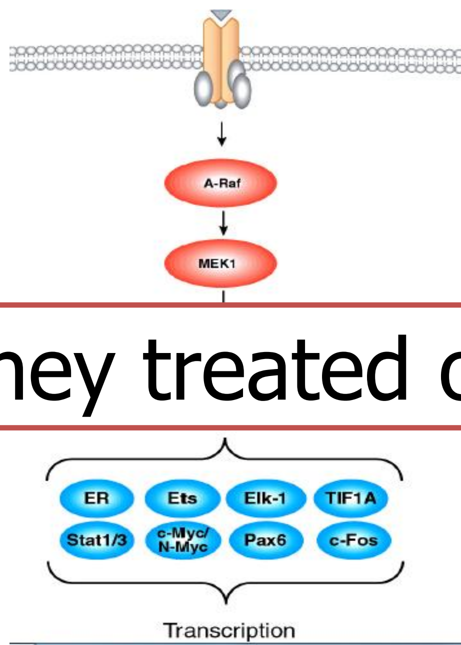
1．3（1） Related works

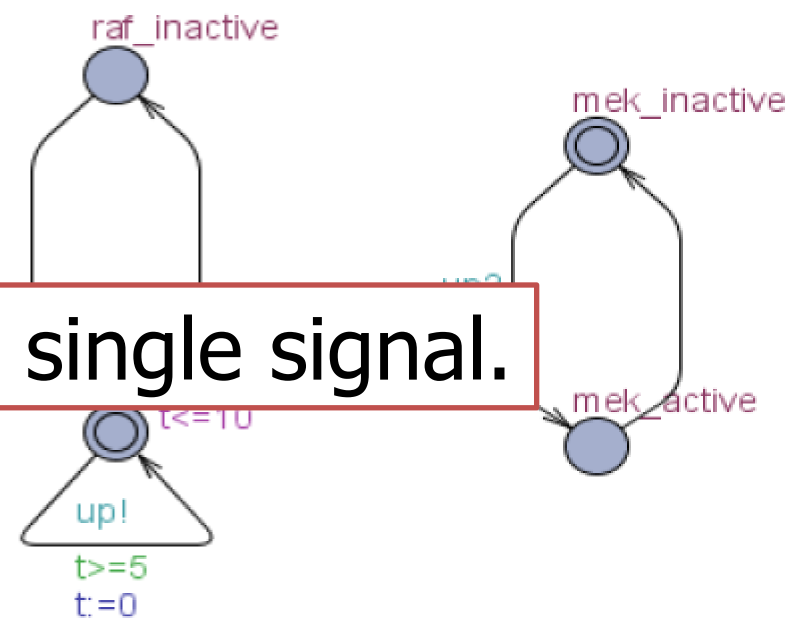
[Chabrier et al. 2003] applied NuSMV to perform model checking on mammalian cell cycle control. They analyzed reachability, pathway, and stability. The time concept is not incorporated.

•N. Chabrier, F. Fages, “Symbolic model checking of biochemical networks,” Lecture Notes in Computer Science 2602, pp.149–162, 2003.

1．3（2） Related works

[Bos et al. 2010] applied UPPAAL to perform model checking on MAPK/ERK pathway.
They analyzed reachability.
The time concept is incorporated by using clocks.



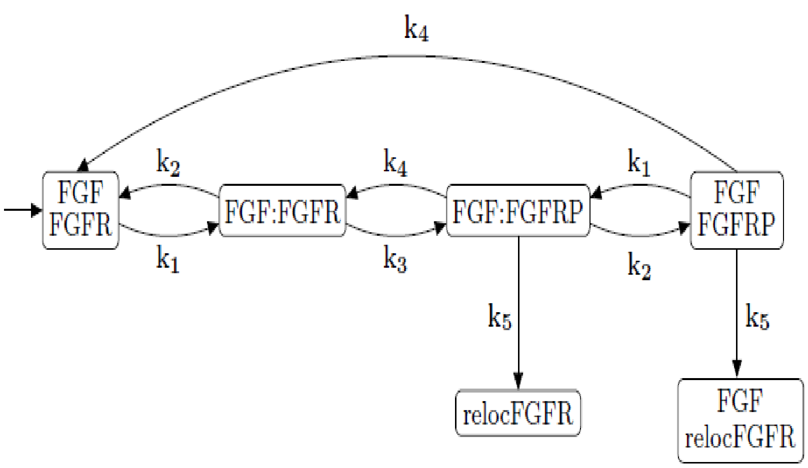


They treated only a single signal.

•W.J.Bos , “Interactive Signaling Network Analysis Tool”, Master Thesis, University of Twente, 2009.

1．3(3)　Related works

[Kwiatkowska et al. 2010] applied PRISM to perform model checking on FGF signaling pathway. They analyzed probability.
The time concept is incorporated by using variables.



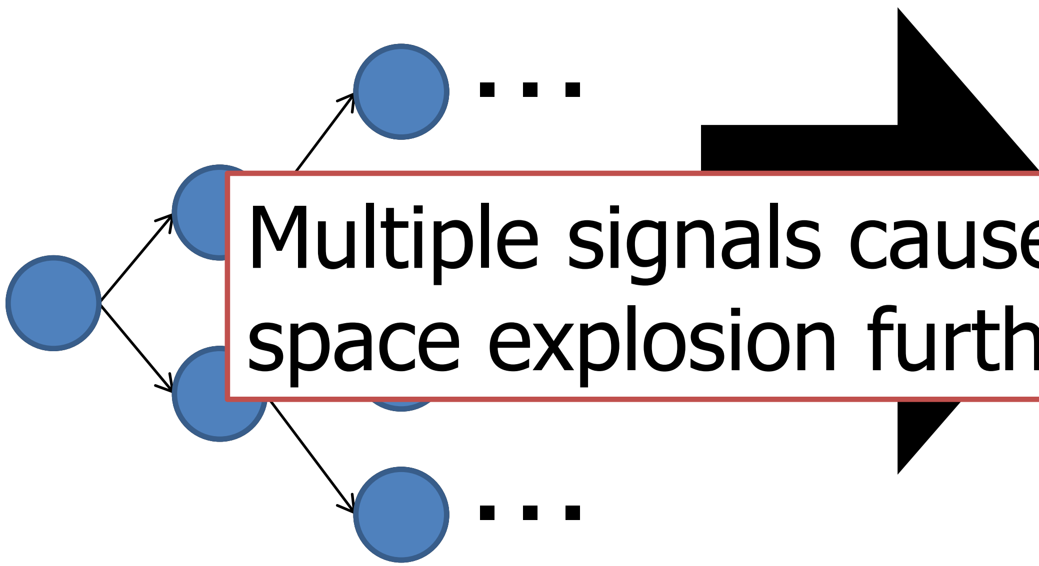

```
const int N; // initial amount of MAPK

// stochastic reaction rates
const double a7=1/N; const double d7=150; const double k7=150;
const double a8=1/N; const double d8=150; const double k8=150;
const double a9=1/N; const double d9=150; const double k9=150;
const double a10=1/N; const double d10=150; const double k10=150;

module MAPK

    k : [0..N] init N; // quantity of MAPK
    k_kkpp : [0..N] init 0; // quantity of MAPK:MAPKK-PP
    kp : [0..N] init 0; // quantity of MAPK-P
    kp_kkpp : [0..N] init 0; // quantity of MAPK-P:MAPKK-PP
    kp_ptase : [0..N] init 0; // quantity of MAPK-P:MAPK phosphatase
    kpp : [0..N] init 0; // quantity of MAPK-PP
    kpp_ptase : [0..N] init 0; // quantity of MAPK-PP:MAPK phosphatase

    // reaction 7 (MAPK is activated by MAPKK-PP)
    [a_k_kk] k>0 & k_kkpp<N → a7 * k : (k_kkpp'=k_kkpp + 1) & (k'=k − 1);
    [d_k_kk] k<N & k_kkpp>0 → d7 * k_kkpp : (k_kkpp'=k_kkpp − 1) & (k'=k + 1);
    [k_k_kk] k_kkpp>0 & kp<N → k7 * k_kkpp : (k_kkpp'=k_kkpp − 1) & (kp'=kp + 1);
    // reaction 8 (MAPK-P is deactivated by MAPK phosphatase)
    [a_k_ptase] kp>0 & kp_ptase<N → a8 * kp : (kp_ptase'=kp_ptase + 1) & (kp'=kp − 1);
    [d_k_ptase] kp<N & kp_ptase>0 → d8 * kp_ptase : (kp_ptase'=kp_ptase − 1) & (kp'=kp + 1);
    [k_k_ptase] kp_ptase>0 & k<N → k8 * kp_ptase : (kp_ptase'=kp_ptase − 1) & (k'=k + 1);
    // reaction 9 (MAPK-P is activated by MAPKK-PP)
    [a_k_kk] kp>0 & kp_kkpp<N → a9 * kp : (kp_kkpp'=kp_kkpp + 1) & (kp'=kp − 1);
    [d_k_kk] kp<N & kp_kkpp>0 → d9 * kp_kkpp : (kp_kkpp'=kp_kkpp − 1) & (kp'=kp + 1);
    [k_k_kk] kp_kkpp>0 & kpp<N → k9 * kp_kkpp : (kp_kkpp'=kp_kkpp − 1) & (kpp'=kpp + 1);
    // reaction 10 (MAPK-PP is deactivated by MAPK phosphatase)
    [a_k_ptase] kpp>0 & kpp_ptase<N → a10 * kpp : (kpp_ptase'=kpp_ptase + 1) & (kpp'=kpp − 1);
    [d_k_ptase] kpp<N & kpp_ptase>0 → d10 * kpp_ptase : (kpp_ptase'=kpp_ptase − 1) & (kpp'=kpp + 1);
    [k_k_ptase] kpp_ptase>0 & kp<N → k10 * kpp_ptase : (kpp_ptase'=kpp_ptase − 1) & (kp'=kp + 1);

endmodule
```

•M. Kwiatkowska, G. Norman,D. Parker, “Probabilistic model checking for systems biology,” Symbolic Systems Biology, Jones and Bartlett, 2010.

1．4　State−space explosion

Model checking can comprehensively analyze all states of a model.

If there is an exponential growth in the state space of the model.



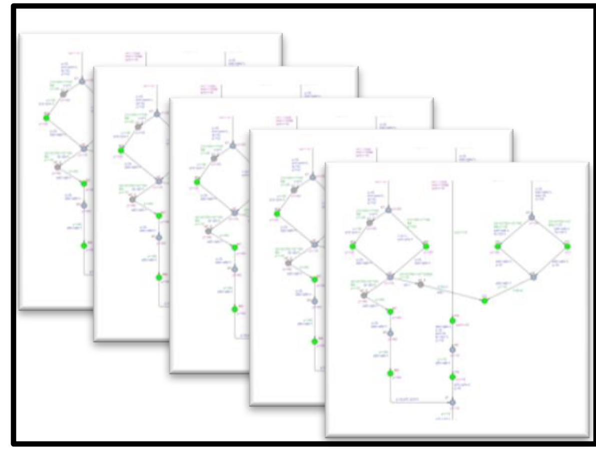
The state-space

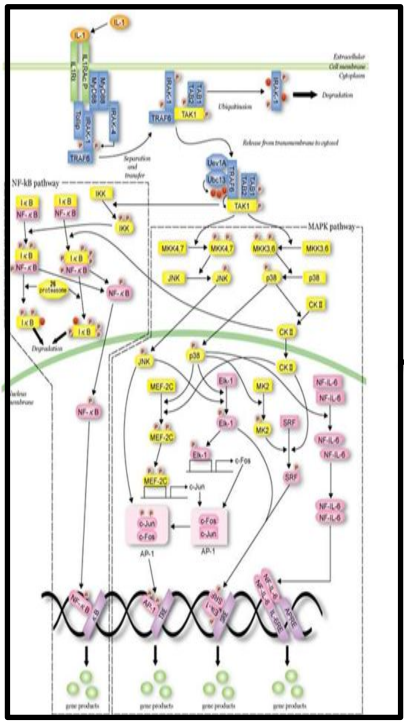
may

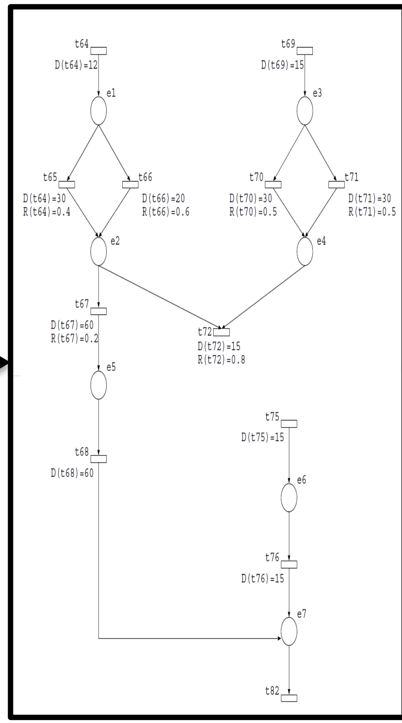
Multiple signals cause the state-space explosion further.

Purpose

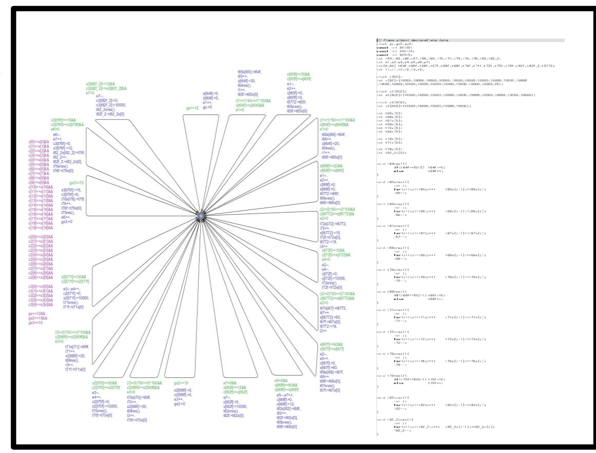
The purpose of our research is to prove the correctness of a Petri net model of a signaling pathway.





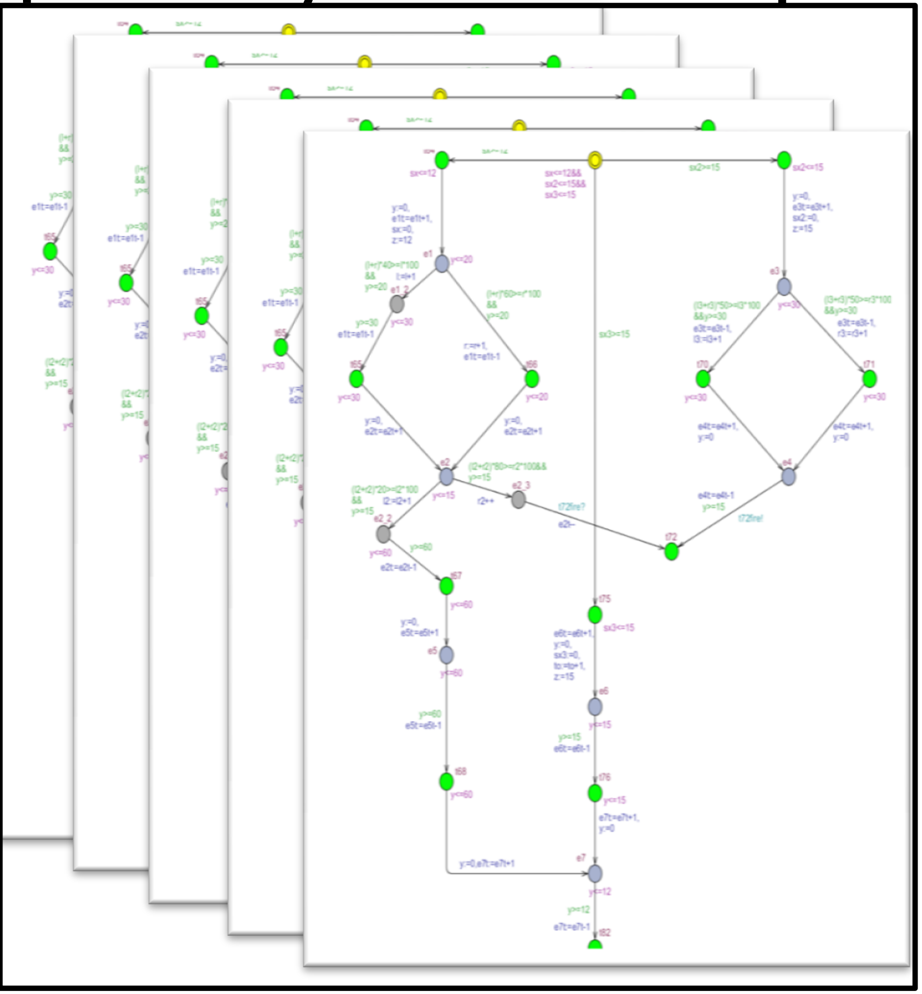


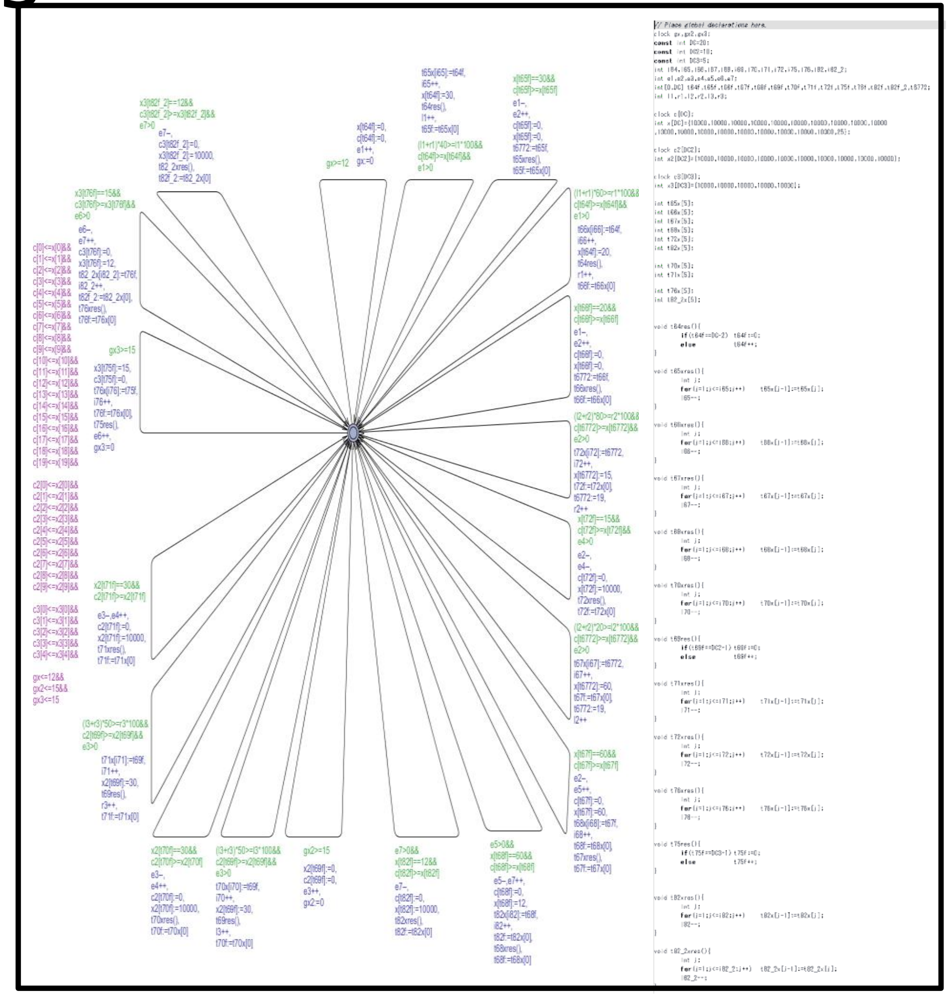
Our proposed methods



This work

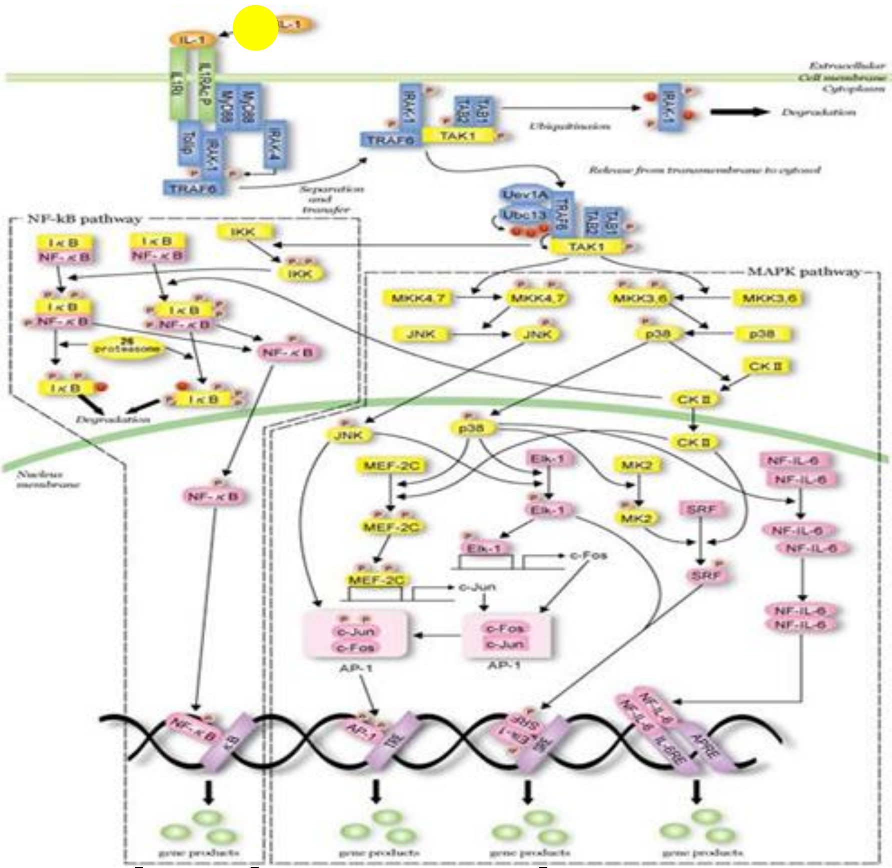
We propose two modeling methods for signaling pathways with multiple signals.

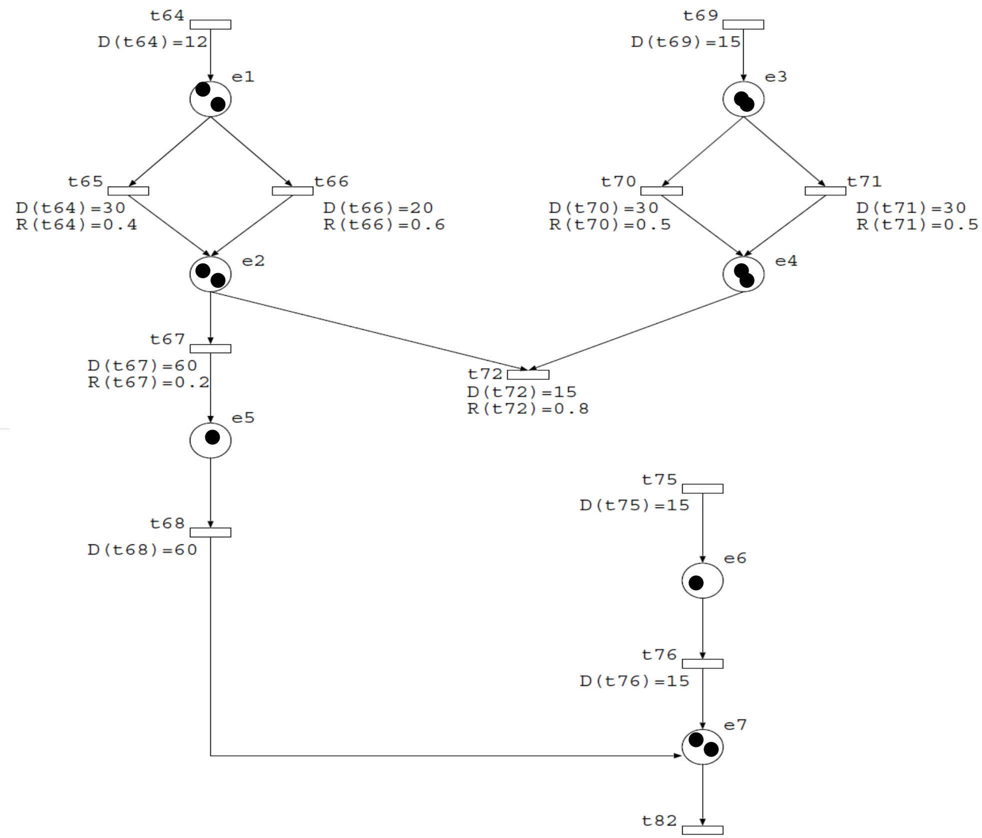




2. Preliminary

2. 1（1） Petri net model of signaling pathway



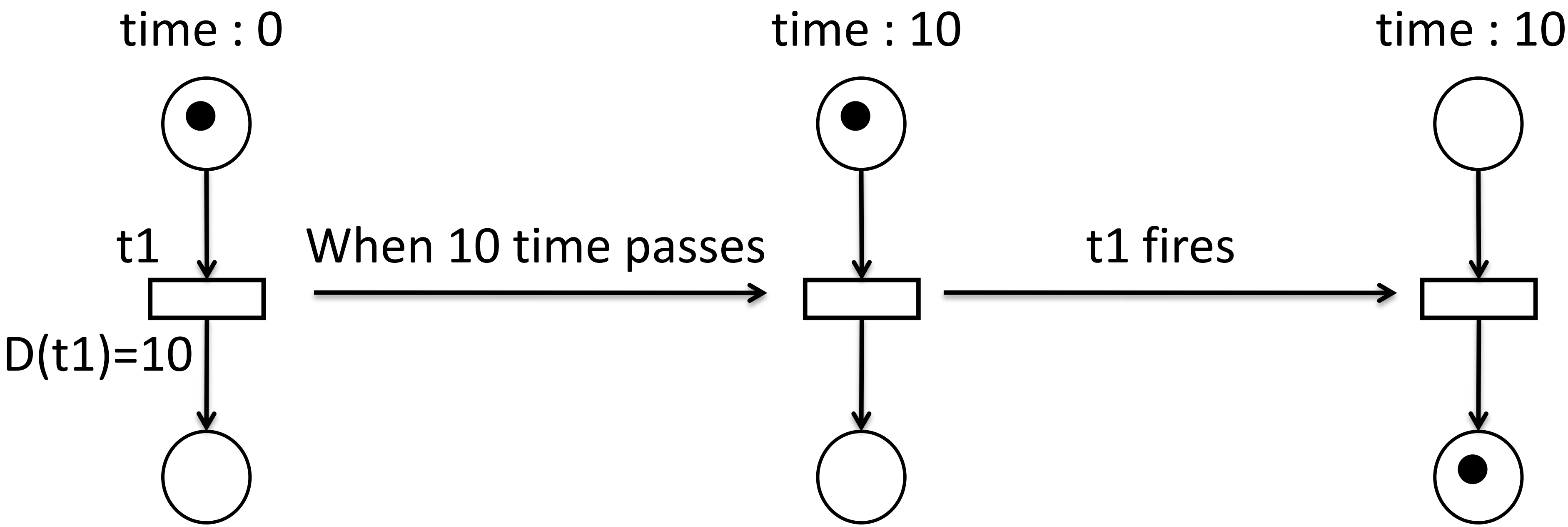


Multiple signals are represented as multiple tokens.

2. 1(2) The structural characteristics

Static elements	Places	○
Active elements	Transitions	▭
The relations between corresponding static elements and active elements.	Directed arcs	→
Signals	Tokens	●
Reaction times	Firing delay times	D(t)
Amount of material used for the reaction	Firing rate	R(t)

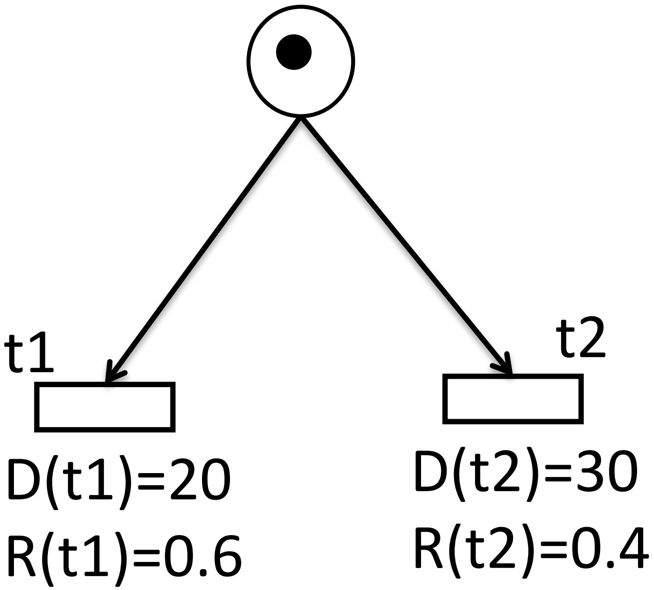
2．1（3） Firing delay times



When D(ti) passes, ti fires to remove the reserved tokens from each input places of ti and put non-reserved tokens into each output places of ti.

It uses multiple server.

2．1（4） Firing rates



$$\sum_{t \in p^\bullet} R(t) = 1$$

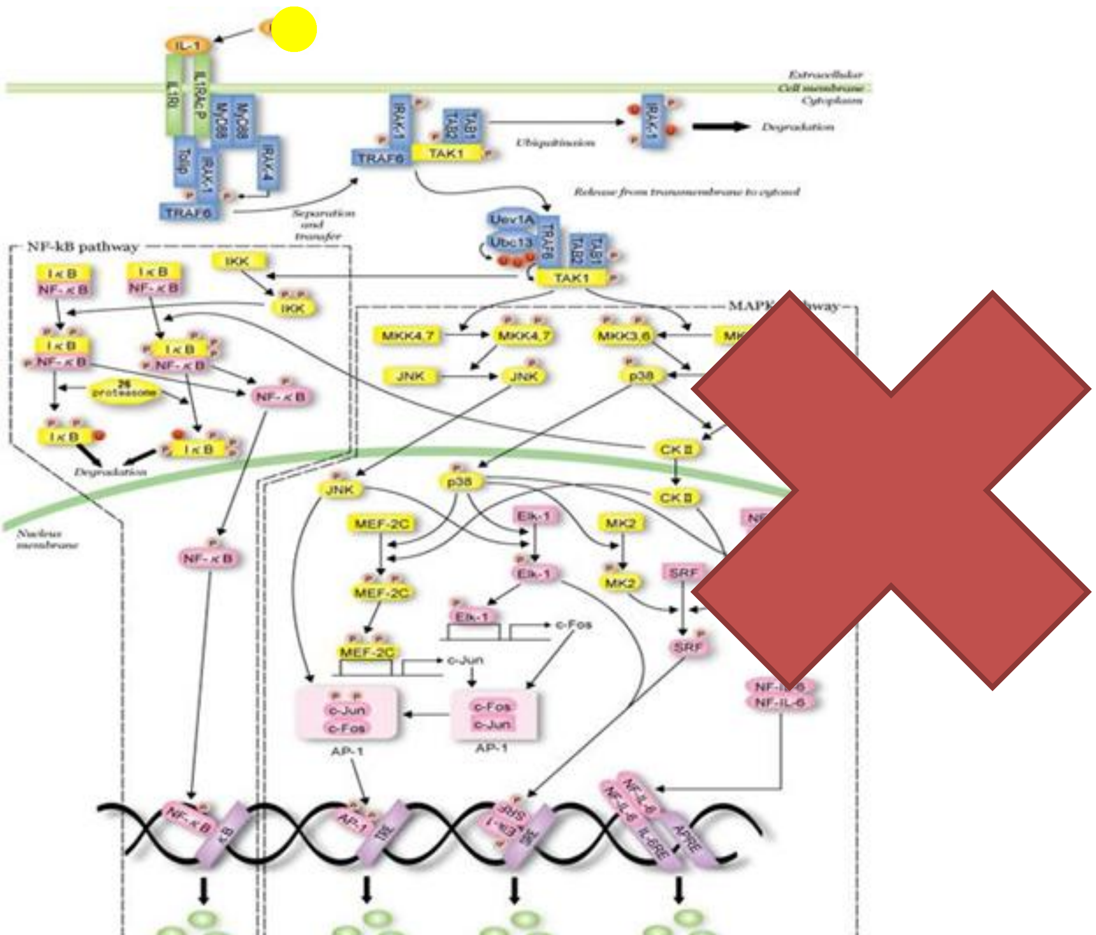
Let X(t) be the firing count of t.

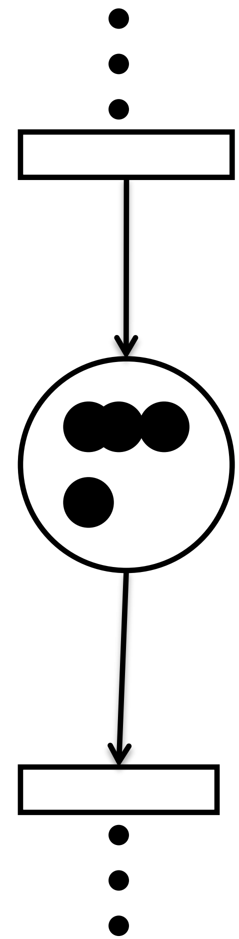
$$\forall t \in p^\bullet : \frac{X(t)}{\sum_{t' \in p^\bullet} X(t')} \leq R(t)$$

Each output transitions can't fire over R(t).

Transition t1 can't fire over firing rate 0.6 and transition t2 can't fire over firing rate 0.4 .

2. 2（1） Retention





Firing delay times and firing rates must be given exactly.

2. 2(2) The correctness of the model

The correctness of the Petri net model of signaling pathways must be examined.

↓

Model checking can comprehensively analyze all states of a model. The correctness is guaranteed.

↓

We use UPPAAL to verify because the Petri net model includes time concept.

2．3（1） Automaton model of UPPAAL

A timed automaton is a 6-tuple (L, l_0 ,C, Act, E, I).

L : Locations

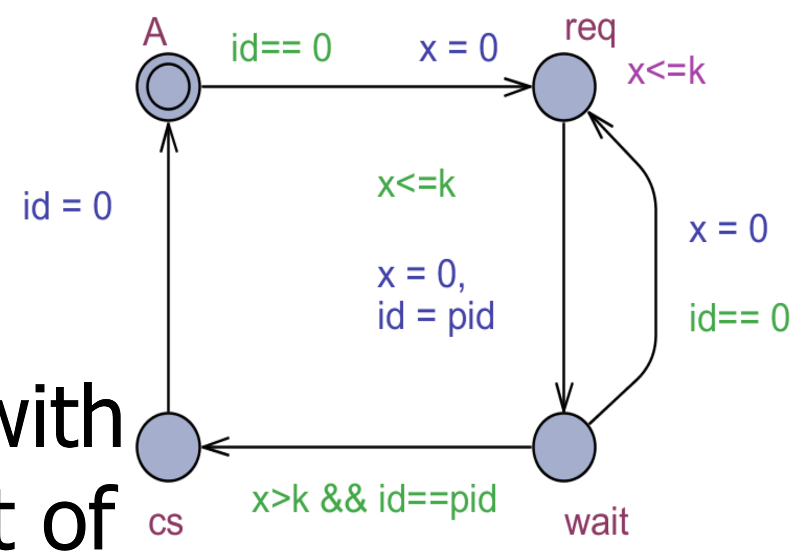
$l_0 \in$ L : The initial location

C : Clocks

E ⊆ L × Act × B(C) × 2^C × L

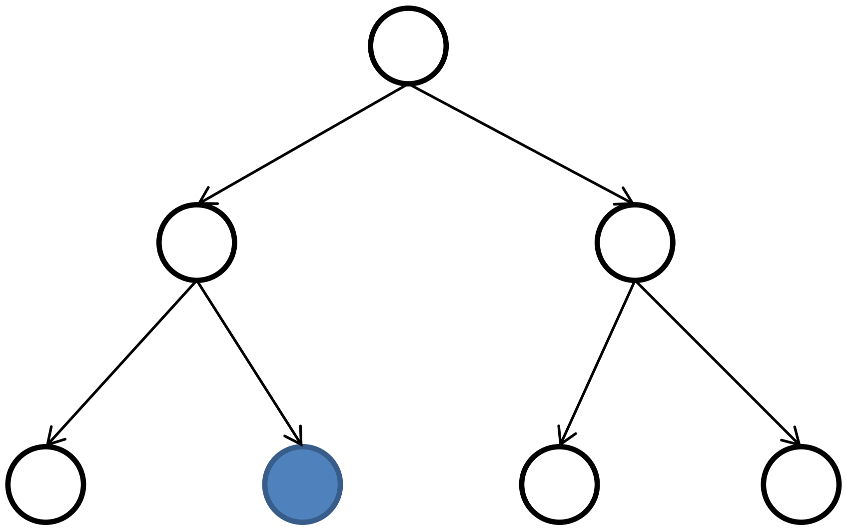
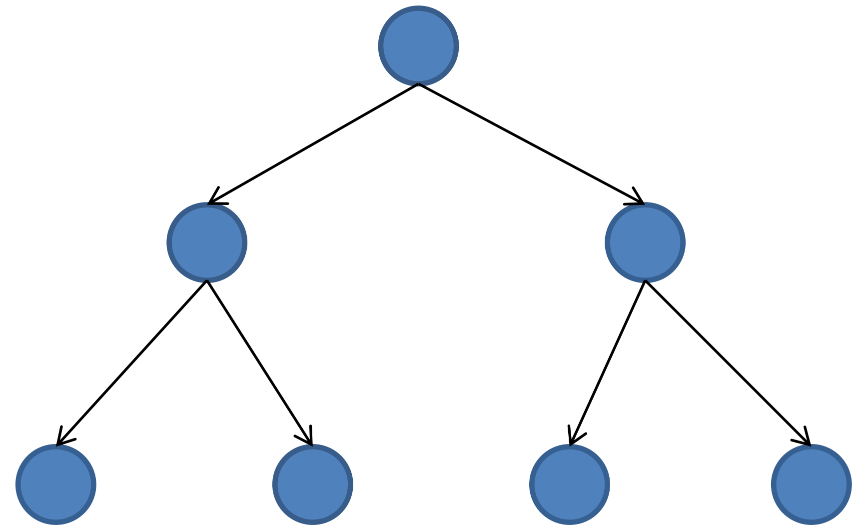
: Edges between locations with an action, a guard and a set of clocks to be reset

I : L → B(C) : Invariants to a location. An invariant is an expression that satisfies the following conditions.



2. 3(2) TCTL on UPPAAL

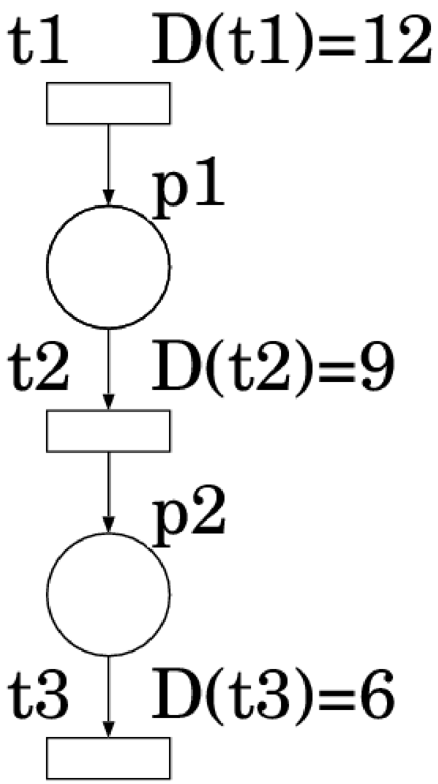
We can verify whether the model with clock variables satisfies the property by running model checking.

E<> p (There exists a path where p eventually holds).	A[] p (For all paths p always holds)
	

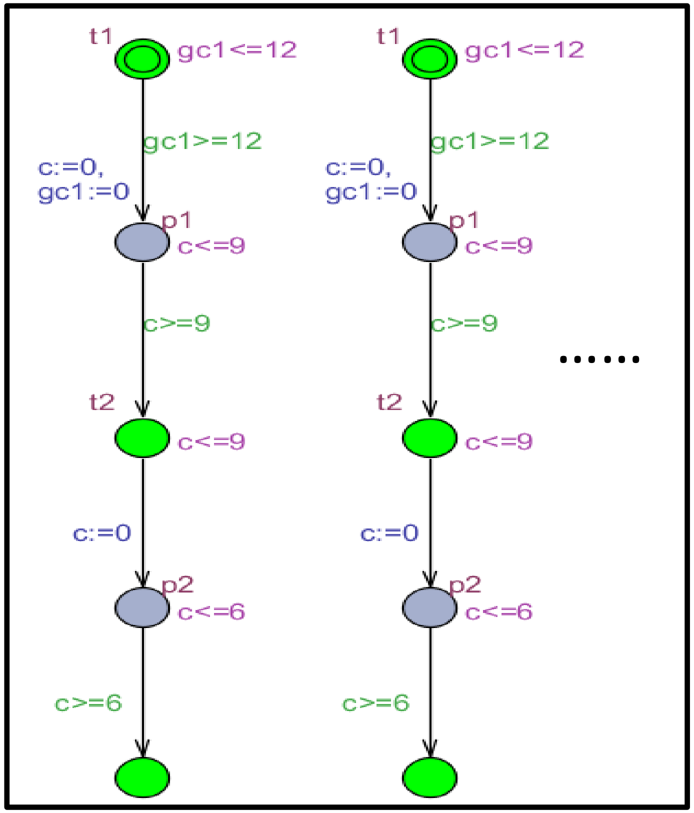
3. Two modeling methods for signaling pathways with multiple signals

- Representation models
- Algorithms
- Pattern lists

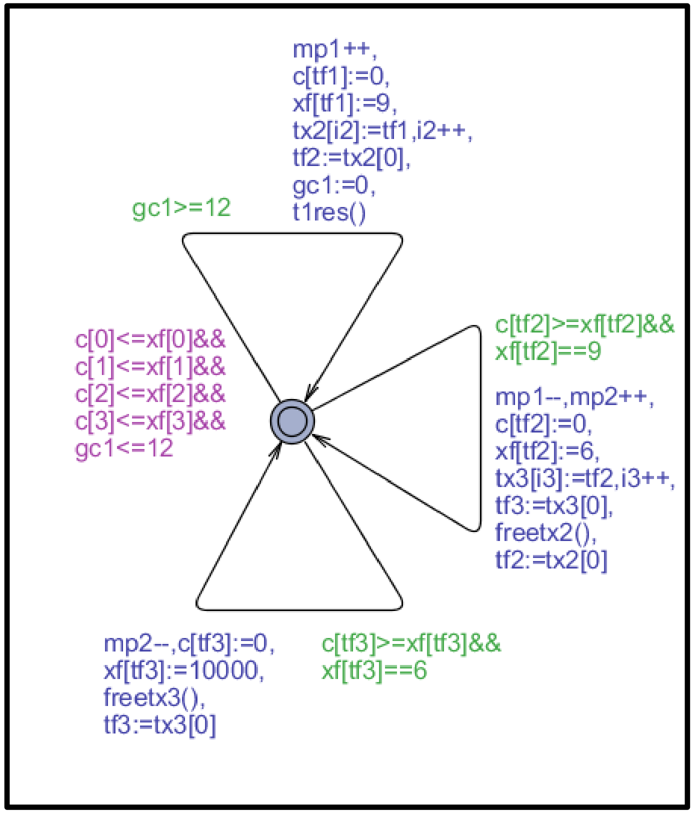
3. 1 Each representation model



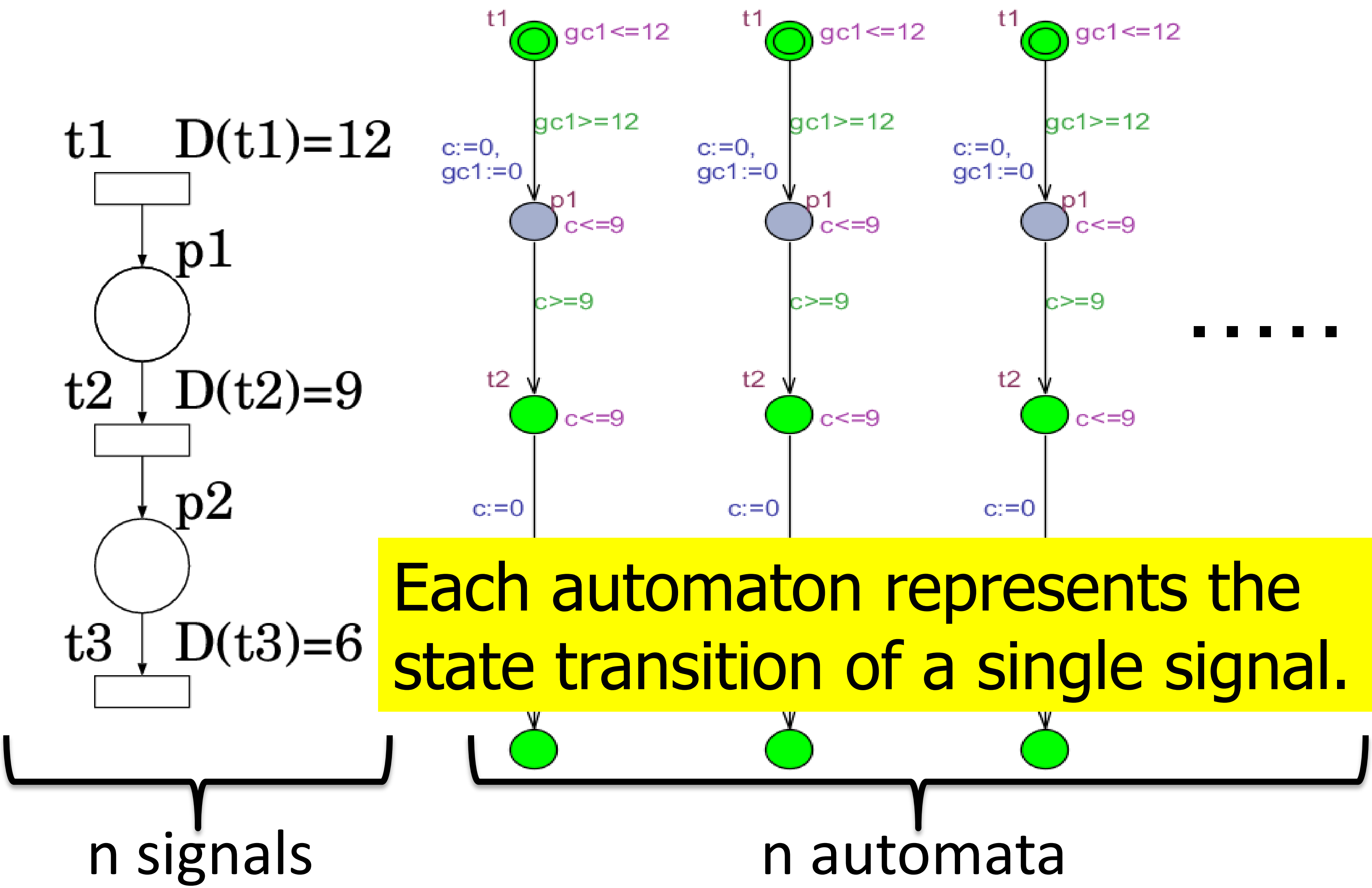
Multiple automata model



Single automaton model

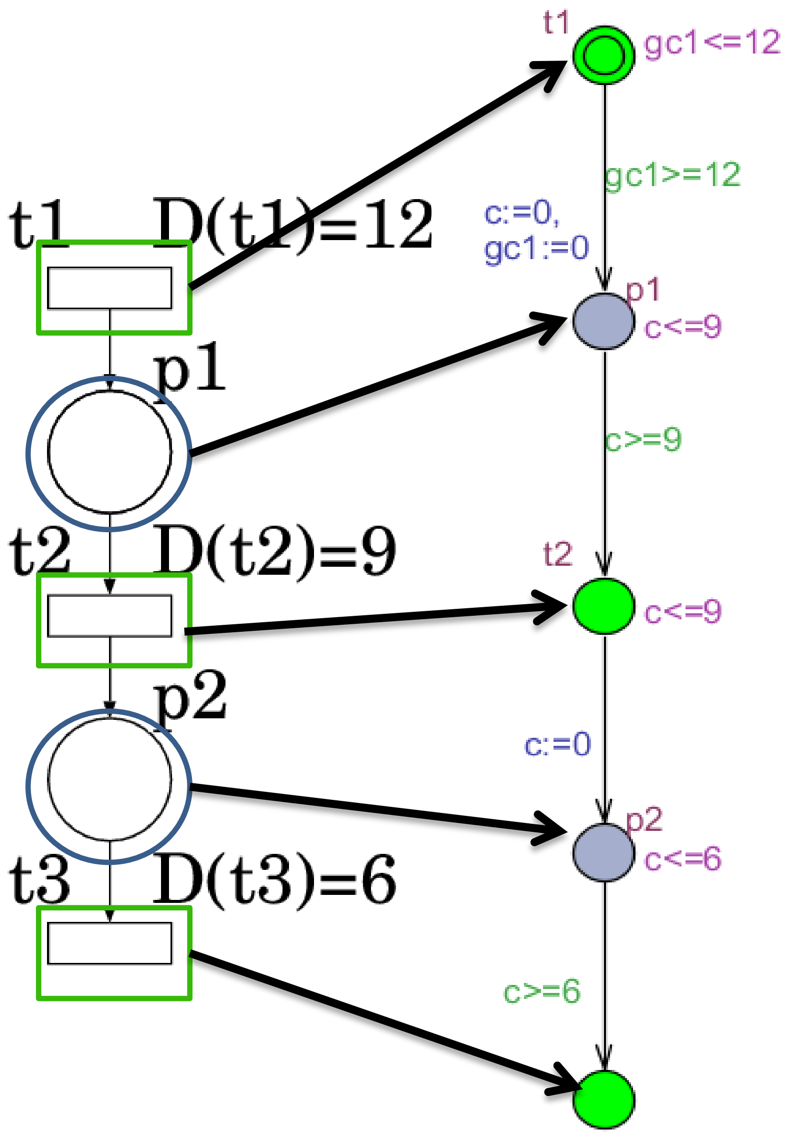


3. 2(1) Multiple automata model



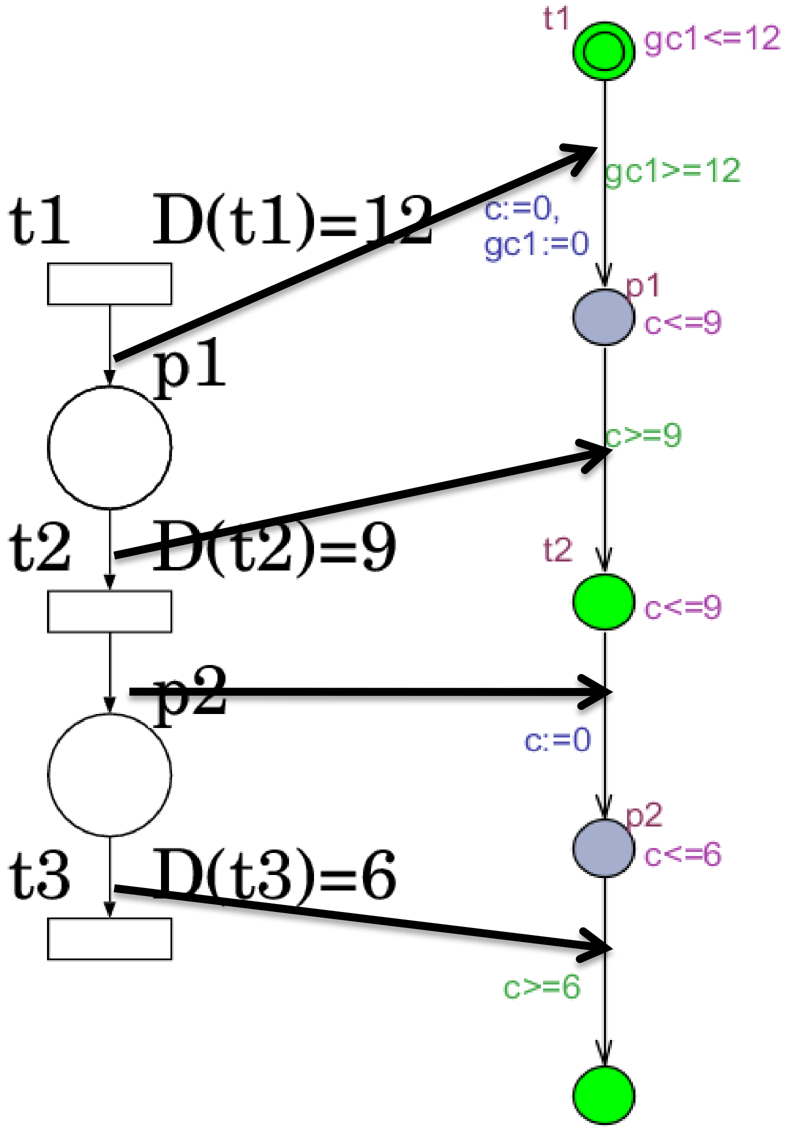
Each automaton represents the state transition of a single signal.

3. 2(2) Components



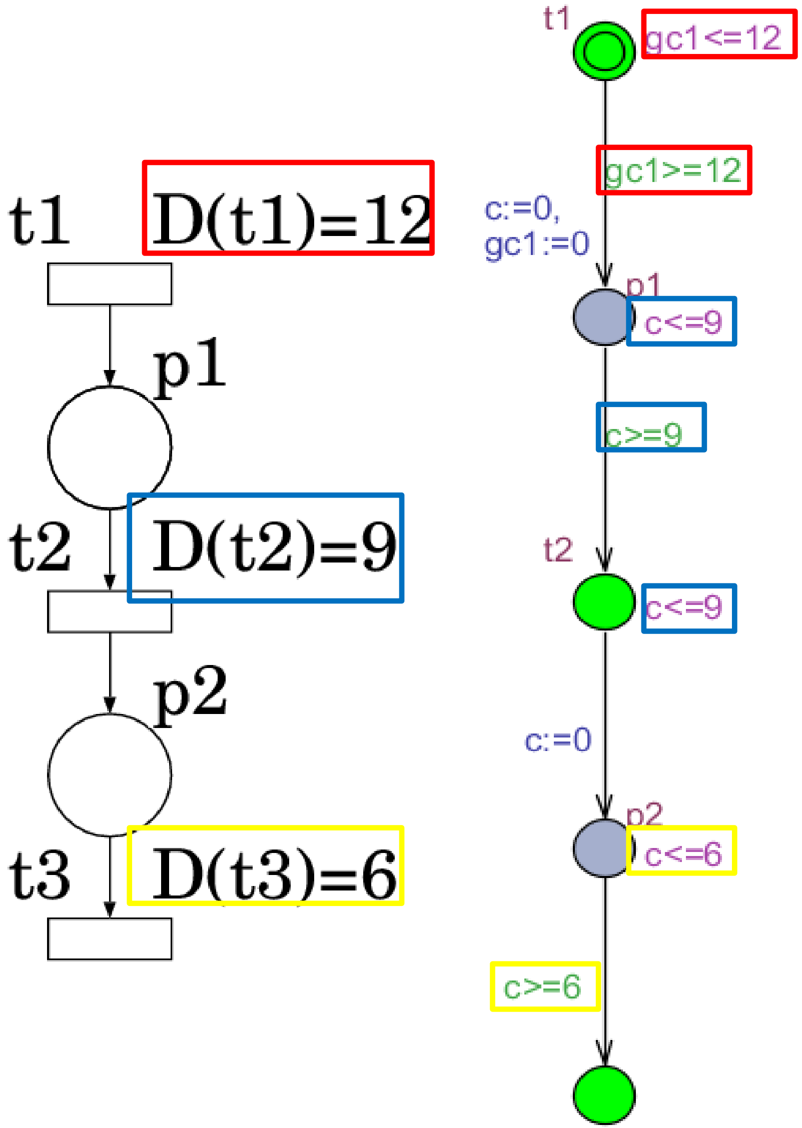
- A places or a transition is represented as a location.
- An arc is represented as an edge.

3．2(2)　Components



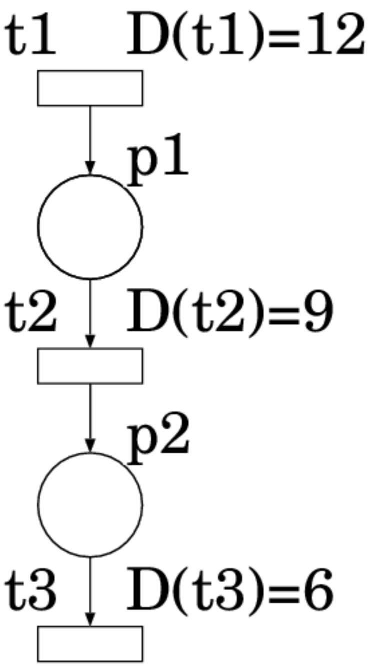
- A places or a transition is represented as a location.
- An arc is represented as an edge.

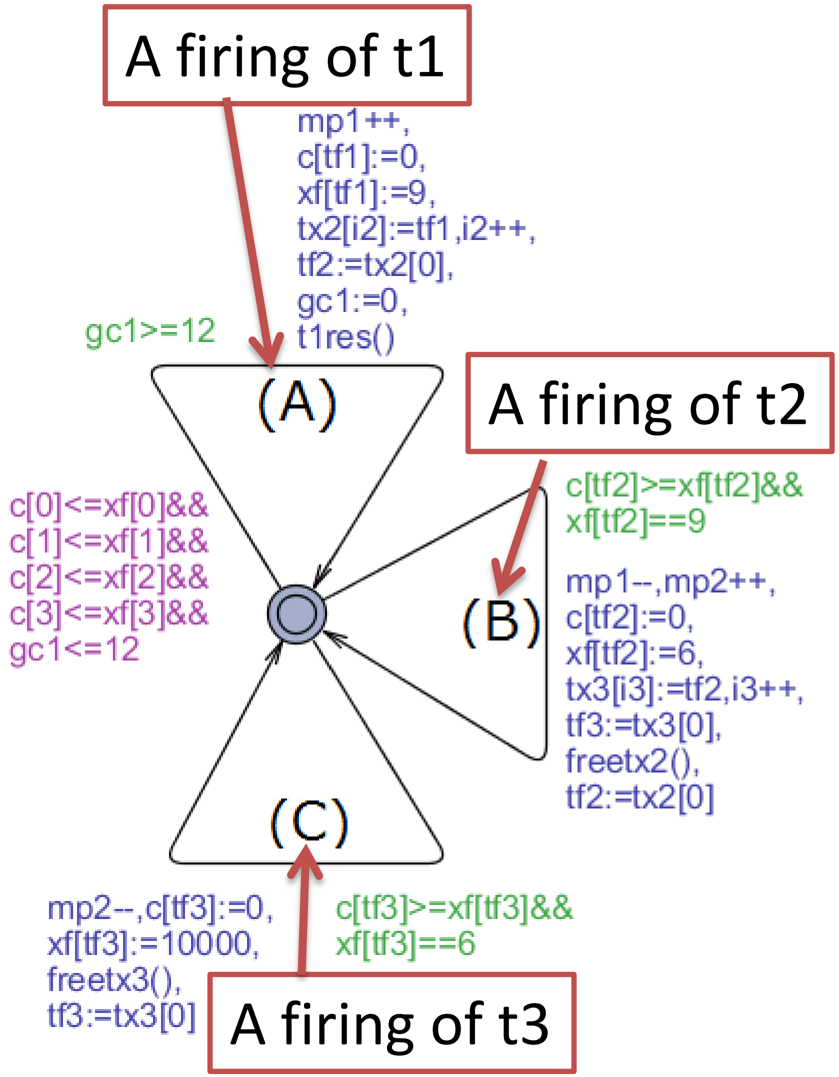
3. 2(3) Firing delay times



Firing delay time is implemented by using a location invariant and a guard.

3．3 Single automaton model



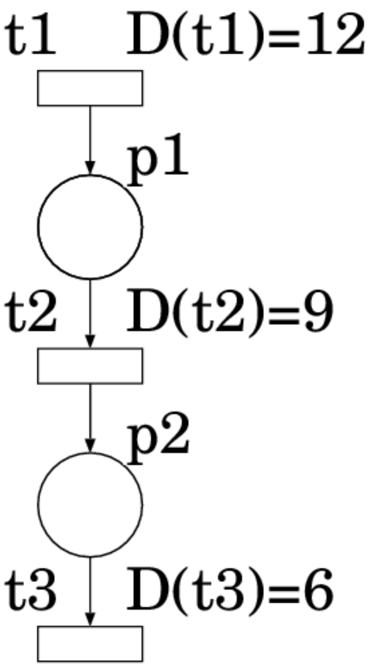


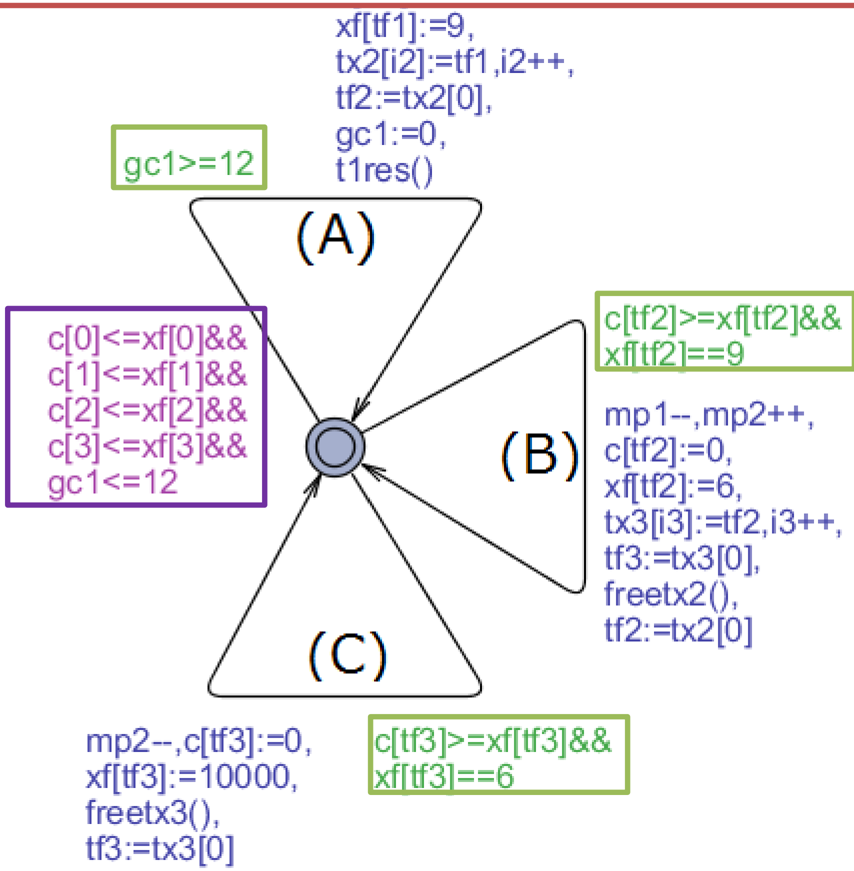

```
// Place global declarations here.
clock gx,gc1;
clock c[4];
int xf[4]={2000,10000,10000,10000};
int tx2[3];
int tx3[3];
int i2,i3;
int tf1=1,tf2,tf3;
int mp1,mp2;

void t1res(){
        if(tf1==3)      tf1=1;
        else            tf1++;
}
void freetx2(){
        int j;
        for(j=1;j<=i2;j++){
                tx2[j-1]:=tx2[j];
        }       i2--;
}
void freetx3(){
        int j;
        for(j=1;j<=i3;j++){
                tx3[j-1]:=tx3[j];
        }       i3--;
}
```

The markings

3．3 Single automaton model

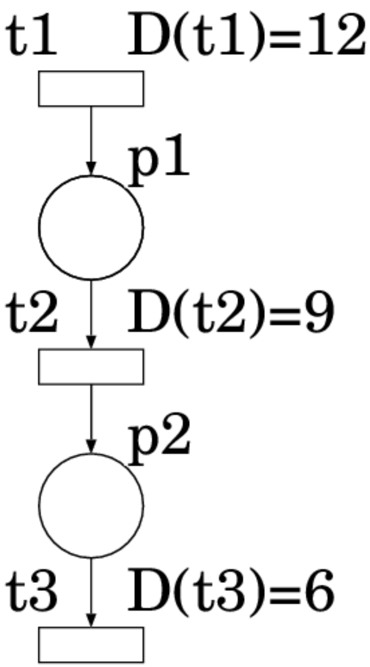
Firing delay time is implemented by using a location invariant and a guard.

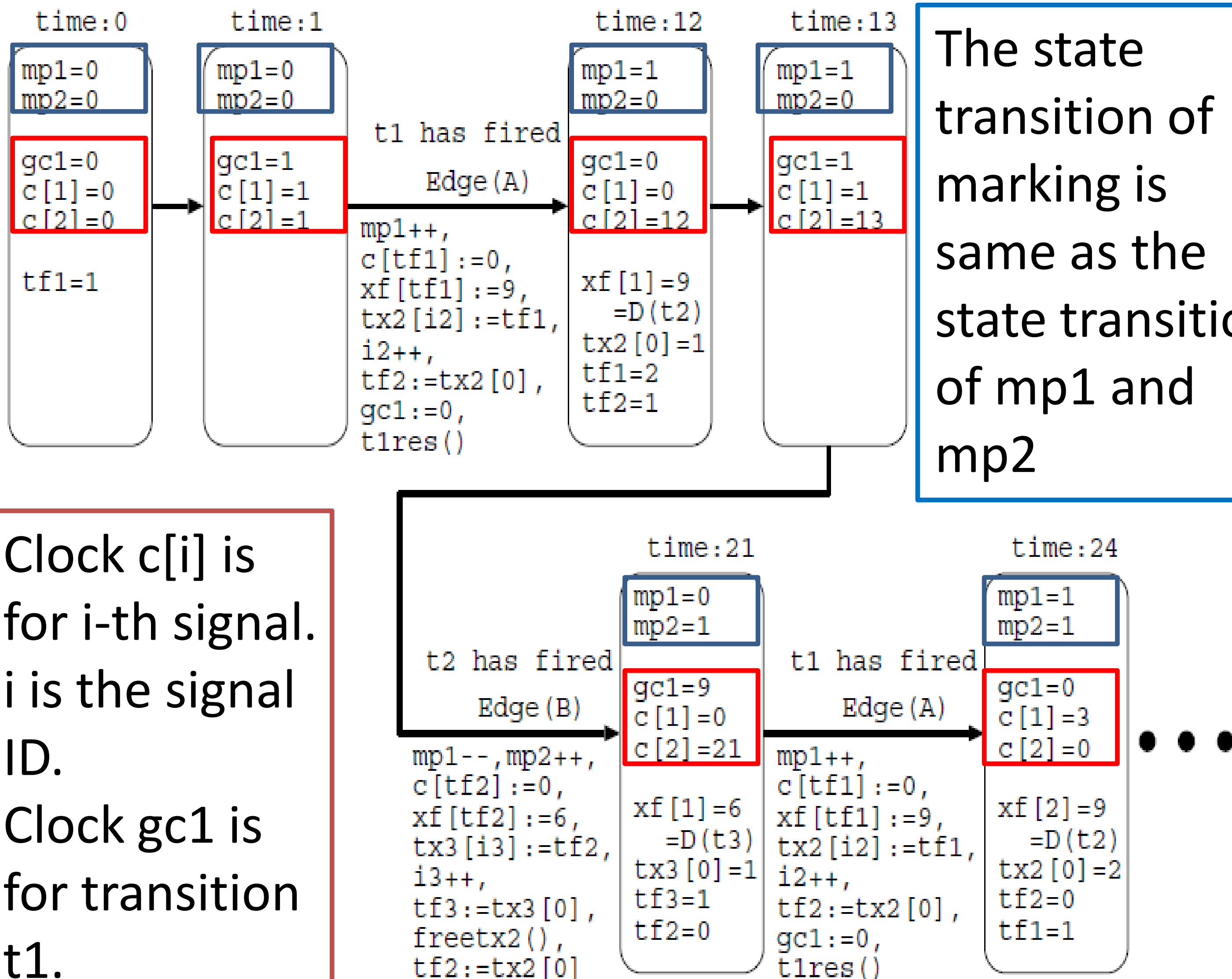





```
clock c[4];
int xf[4]={2000,10000,10000,10000};
int tx2[3];
int tx3[3];
int i2,i3;
int tf1=1,tf2,tf3;
int mp1,mp2;

void t1res(){
        if(tf1==3)      tf1=1;
        else            tf1++;
}
void freetx2(){
        int j;
        for(j=1;j<=i2;j++){
                tx2[j-1]:=tx2[j];
        }       i2--;
}
void freetx3(){
        int j;
        for(j=1;j<=i3;j++){
                tx3[j-1]:=tx3[j];
        }       i3--;
}
```

3．3(1)　State transition

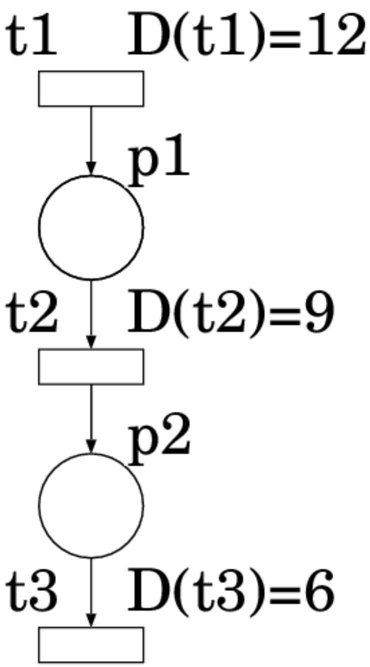


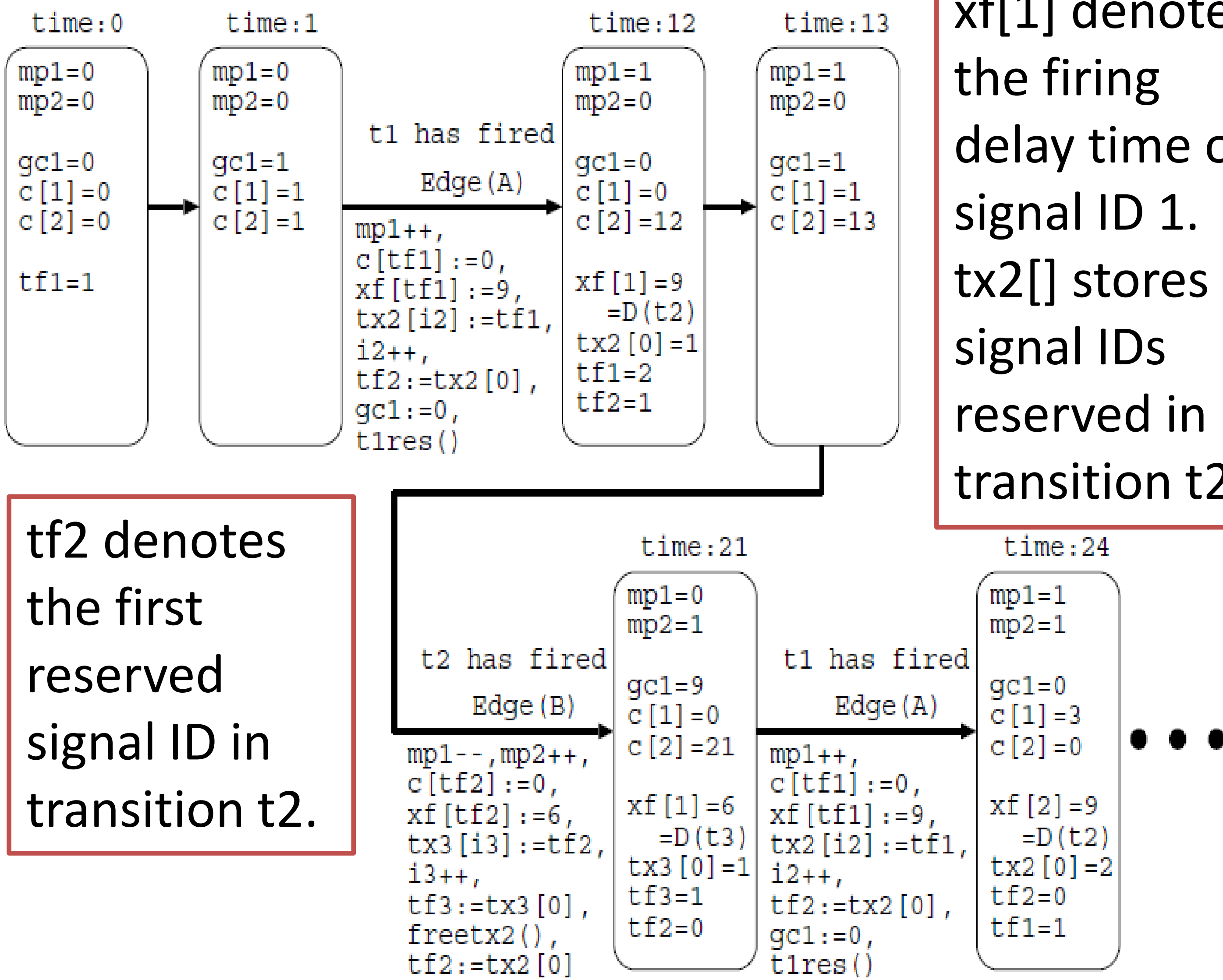


The state transition of marking is same as the state transition of mp1 and mp2

Clock c[i] is for i-th signal. i is the signal ID.
Clock gc1 is for transition t1.

3．3（1） State transition

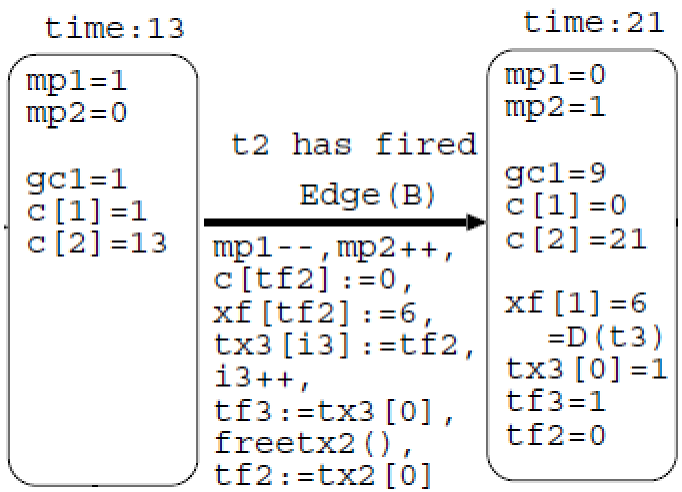


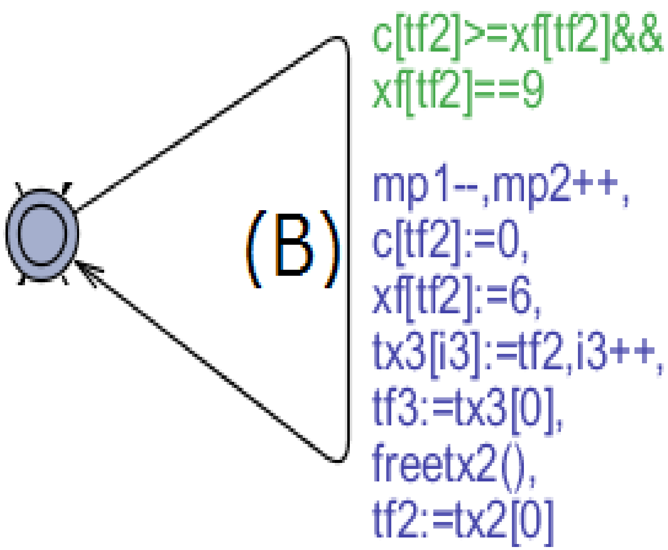


xf[1] denotes the firing delay time of signal ID 1. tx2[] stores signal IDs reserved in transition t2.

tf2 denotes the first reserved signal ID in transition t2.

3. 3(2) A firing of transition t2





In a firing of transition t2,

•xf[tf2]:=6

→Setting the firing delay time of transition t3

•tx3[i3]:=tf2

→Reserving the signal to transition t3

•tf3:=tx3[0]

→Removing the first reserved signal ID of transition t3

3. 4 Multiple automata modeling method

The algorithm to the modeling of a single path Petri net model.

<<Transformation to Multiple Automata Model>>

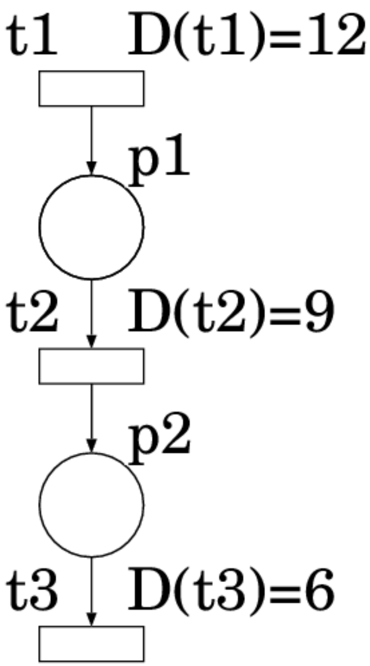
Input: Petri net model $TPNR = (P, T, \mathcal{E}, D, R)$, number n of signals

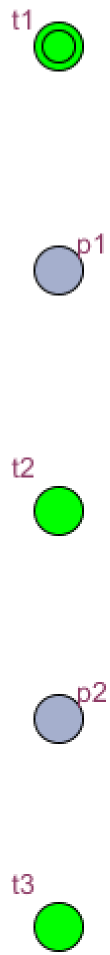
Output: Multiple automata model $A_1, A_2, \cdots, A_n$

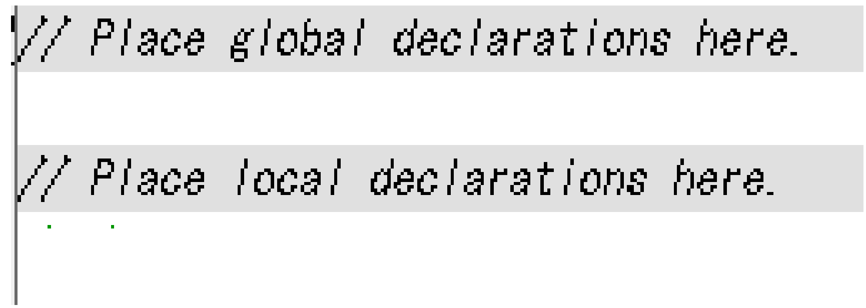
Foreach $i = 1$ to n, make A_i according to the following:

1. $L \leftarrow P \cup T$
2. $C \leftarrow \{\texttt{gc1},\ \texttt{c}i\}$
3. $E \leftarrow \{(p, \emptyset, (\texttt{c}i\texttt{>=}\ D(t)), \emptyset, t) |\ (p, t) \in A\}$
   $\cup \{(t, (\texttt{gc1:=0},\ \texttt{c}i\texttt{:=0}), (\texttt{gc1>=}D(t)), \{\texttt{gc1},\ \texttt{c}i\}, p) |\ |{}^{\bullet}t| = 0, (t, p) \in A\}$
   $\cup \{(t, (\texttt{c}i := 0), \emptyset, \{\texttt{c}i\}, p) |\ |t^{\bullet}| > 0, (t, p) \in A\}$
4. $I \leftarrow \{(t, (\texttt{gc1<=}D(t_1))) |\ t \in T, |{}^{\bullet}t| = 0\}$
   $\cup \{(t, (\texttt{c}i\texttt{<=}D(t))) |\ t \in T, |t^{\bullet}| > 0\}$
   $\cup \{(p, (\texttt{c}i\texttt{<=}D(t))) |\ p \in P$, t is the output transition of $p\}$

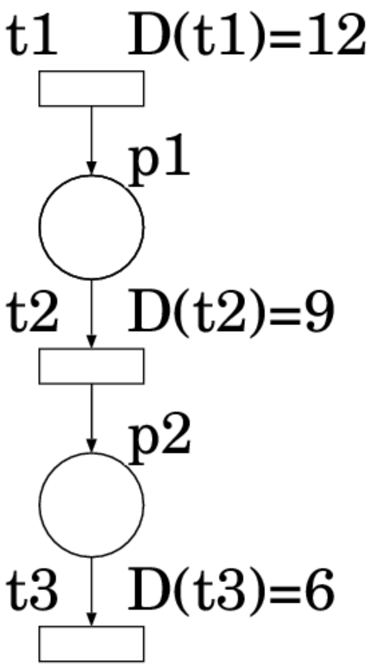
3. 4(1) Step1 : Make locations implementing places and transitions

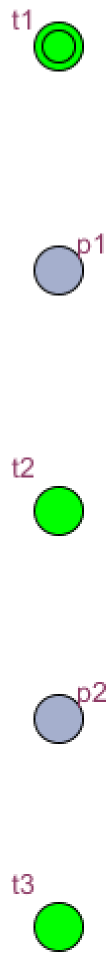






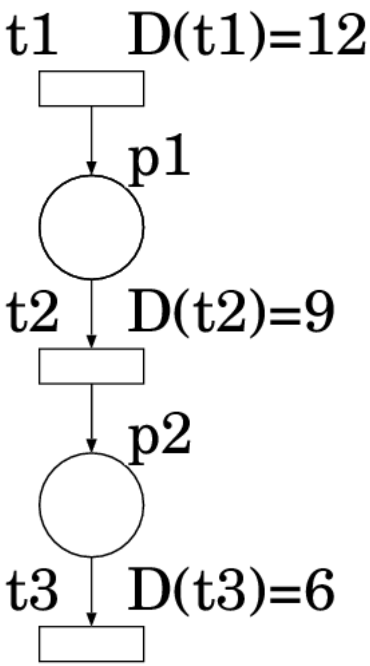
3. 4(2) Step2 : Declare clocks

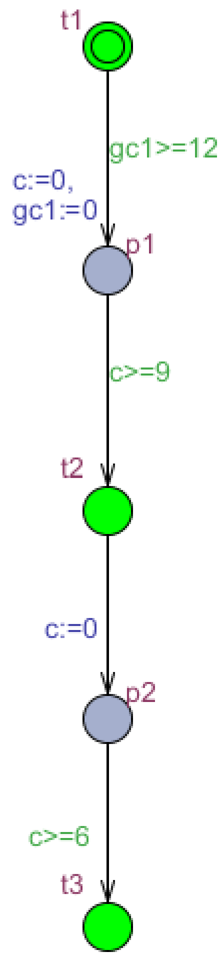





```
// Place global declarations here.
clock gc1;
// Place local declarations here.
clock c;
```

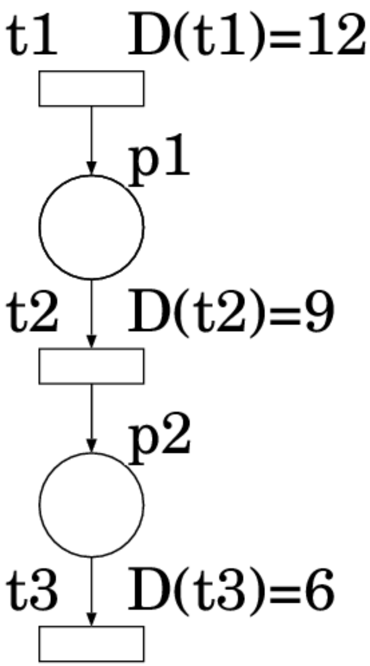
3. 4(3) Step3 : Make edges between locations

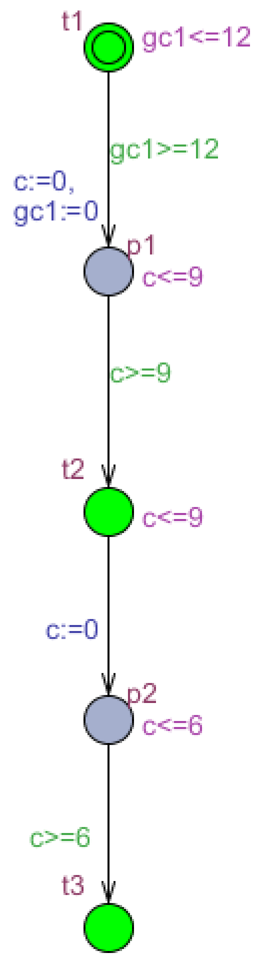





```
// Place global declarations here.
clock gc1;
// Place local declarations here.
clock c;
```

3. 4(4) Step4 : Describe invariants






```
// Place global declarations here.
clock gc1;
// Place local declarations here.
clock c;
```

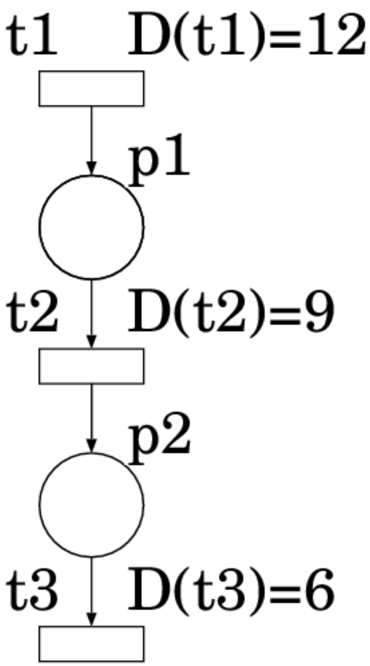
3．5 Single automaton modeling method

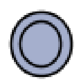
The algorithm to the modeling of a single path Petri net model.

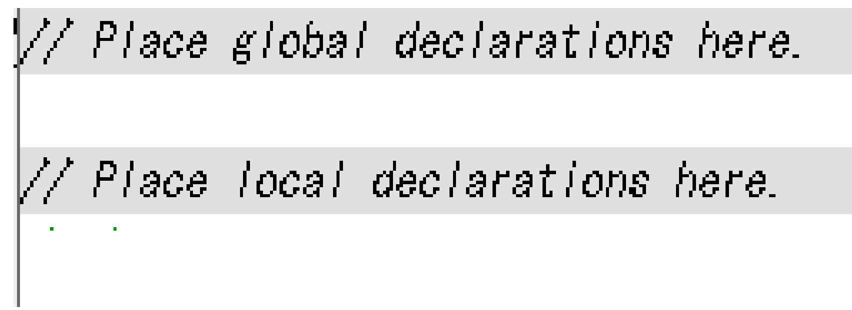
<<Transformation to Single Automaton Model>>
Input: Petri net model $TPNR = (P, T, \mathcal{E}, D, R)$, number n of signals
Output: Single automaton model A, variables mp1, mp2, $\cdots$, mp$|P|$, tf1, tf2, $\cdots$, tf|T|, xf[], tx1[], tx2[], $\cdots$, tx$|T|$[].

1. $L \leftarrow \{l_0\}$
2. $C \leftarrow$ {gc1, c[1], c[2],$\cdots$, c[k]}
3. $E \leftarrow$ {(l_0, (mp1++, c[tf1]:=0, xf[tf1]:=$D(t_2)$, tx2[i2]:=tf1, i2++, tf2:=tx2[0], gc1:=0, t1res()), (gc1>=$D(t_1)$), {gc1, c[tf1]}, l_0)} $\cup$ {(l_0, (mp$|P|$--, c[tf|T|]:=0, xf[tf$|T|$]:=10000, freetx|T|(), tf$|T|$:=tx$|T|$[0]), (c[tf$|T|$]>=xf[tf$|T|$] && xf[tf$|T|$]==$D(t_{|T|})$)), {c[tf$|T|$]}, l_0)} $\cup \bigcup_{i=2 to |T|-1}$ {(l_0, (mp($i-1$)--, mp$i$++, c[tf$i$]:=0, xf[tf$i$]:=$D(t_{(i+1)})$,tx($i+1$)[i($i+1$)]:=tf$i$,i($i+1$)++, tf($i+1$):=tx($i+1$)[0], freetx$i$(), tf$i$:=tx$i$[0]), (c[tf$i$]>=xf[tf$i$]&&xf[tf$i$]==$D(t_i)$), {c[tf$i$]}, l_0)}
   t1res() resets c[] and xf[].
   freetx$i$() organizes reserved signal IDs of transition t_i.
4. $I \leftarrow$ {(l_0, (c[0]<=xf[0]&&c[1]<=xf[1]&&$\cdots$ &&c[k]<=xf[k]&& gc1<=$D(t_1)$))}

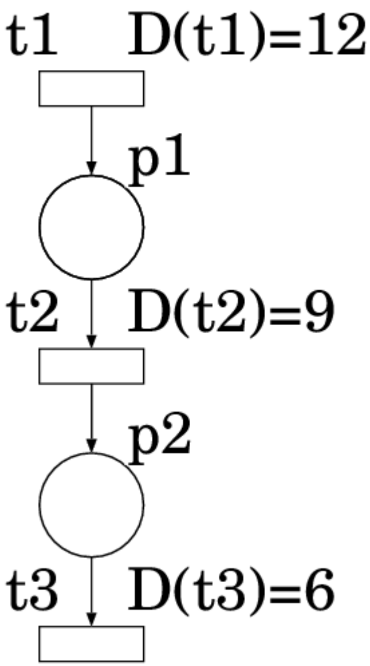
3. 5(1) Step1 : Make a single location

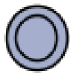






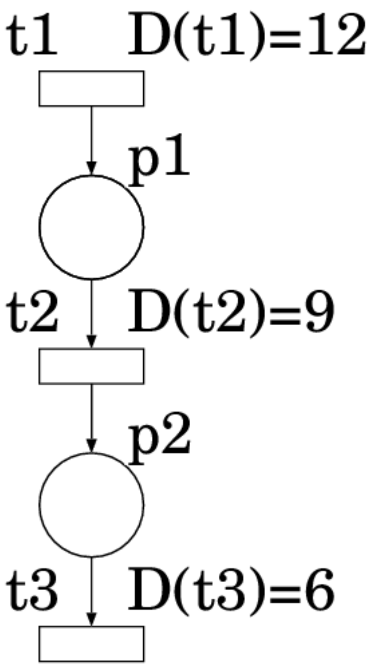
3．5(2)　Step2：Declare clocks and variables

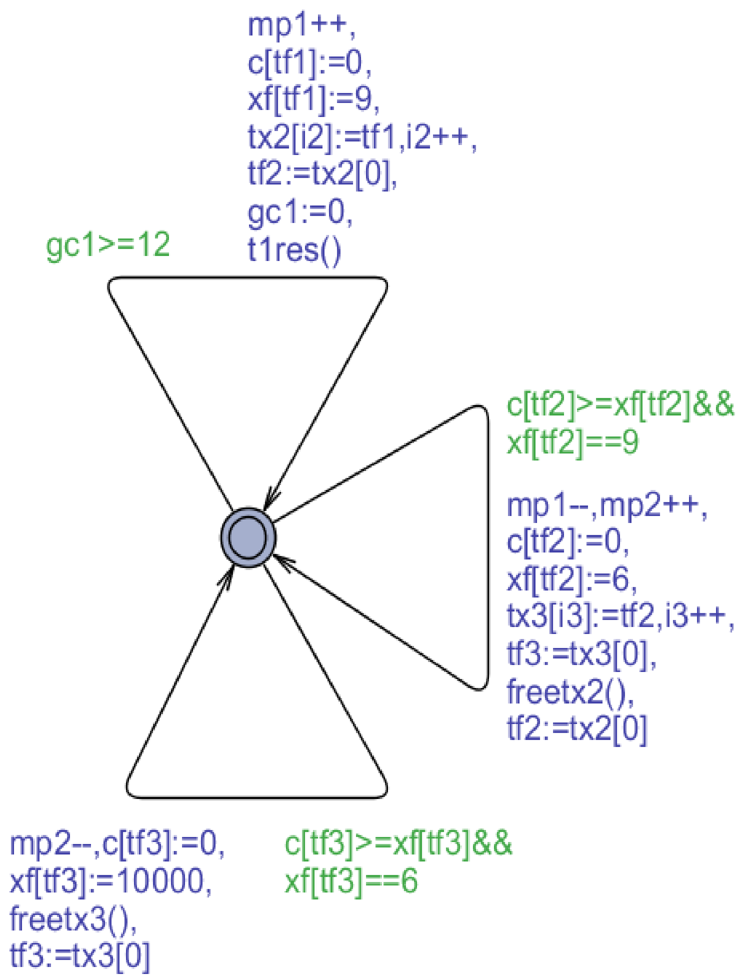




```
// Place global declarations here.
clock  gc1;
clock c[4];
int xf[4]={2000,10000,10000,10000};
int tx2[3];
int tx3[3];
int i2,i3;
int tf1=1,tf2,tf3;
int mp1,mp2;

void t1res(){
        if(tf1==3)      tf1=1;
        else            tf1++;
}
void freetx2(){
        int j;
        for(j=1;j<=i2;j++){
                tx2[j-1]:=tx2[j];
        }       i2--;
}
void freetx3(){
        int j;
        for(j=1;j<=i3;j++){
                tx3[j-1]:=tx3[j];
        }       i3--;
}
```

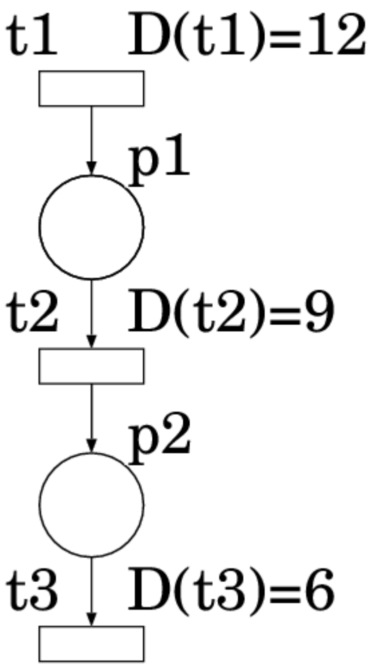
3. 5(3) Step3 : Make edges of self loop to the location

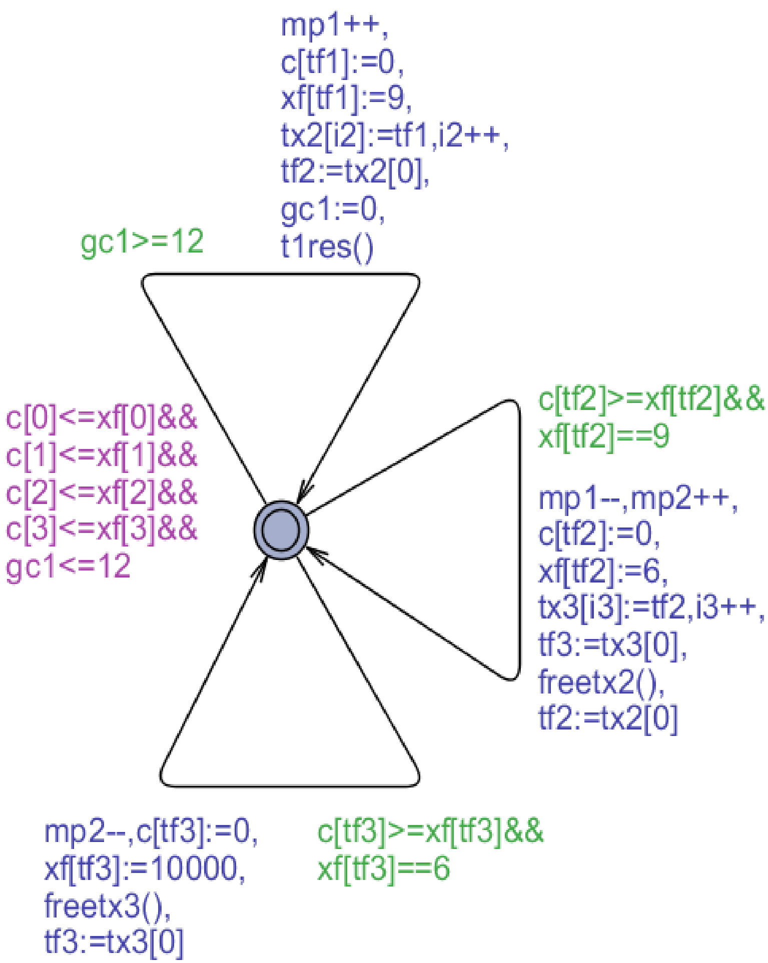





```
// Place global declarations here.
clock gc1;
clock c[4];
int xf[4]={2000,10000,10000,10000};
int tx2[3];
int tx3[3];
int i2,i3;
int tf1=1,tf2,tf3;
int mp1,mp2;

void t1res(){
        if(tf1==3)      tf1=1;
        else            tf1++;
}
void freetx2(){
        int j;
        for(j=1;j<=i2;j++){
                tx2[j-1]:=tx2[j];
        }       i2--;
}
void freetx3(){
        int j;
        for(j=1;j<=i3;j++){
                tx3[j-1]:=tx3[j];
        }       i3--;
}
```

3. 5(4) Step4 : Describe an invariant to the location

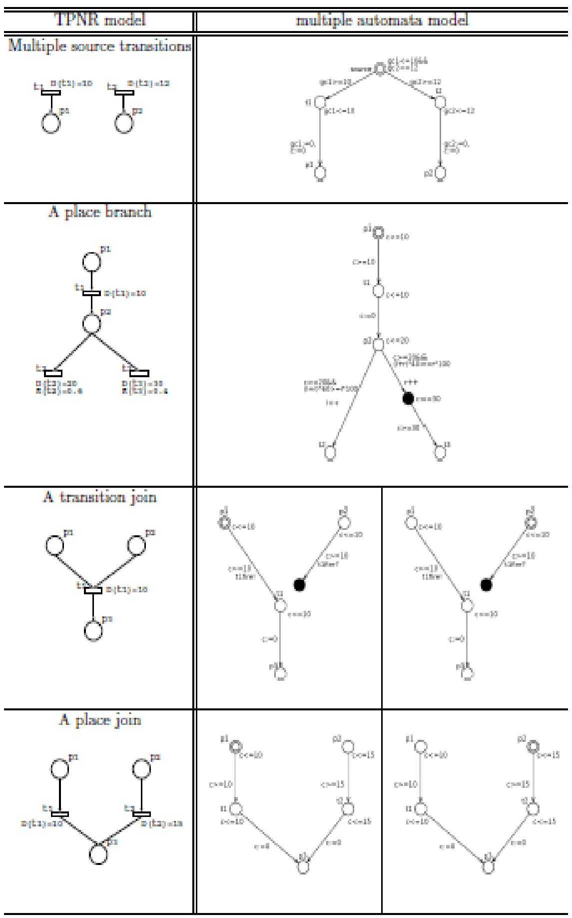


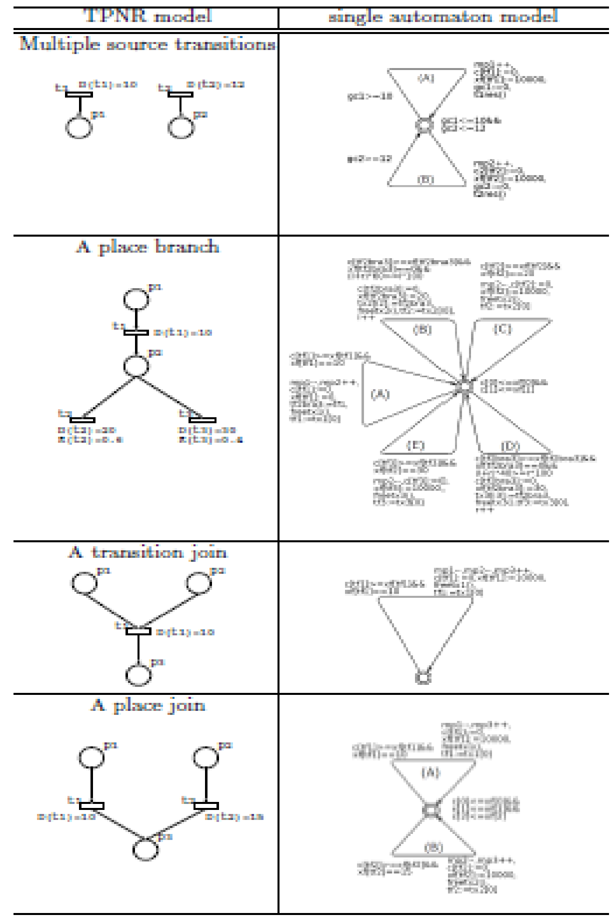



```
// Place global declarations here.
clock  gc1;
clock c[4];
int xf[4]={2000,10000,10000,10000};
int tx2[3];
int tx3[3];
int i2,i3;
int tf1=1,tf2,tf3;
int mp1,mp2;

void t1res(){
        if(tf1==3)      tf1=1;
        else            tf1++;
}
void freetx2(){
        int j;
        for(j=1;j<=i2;j++){
                tx2[j-1]:=tx2[j];
        }       i2--;
}
void freetx3(){
        int j;
        for(j=1;j<=i3;j++){
                tx3[j-1]:=tx3[j];
        }       i3--;
}
```

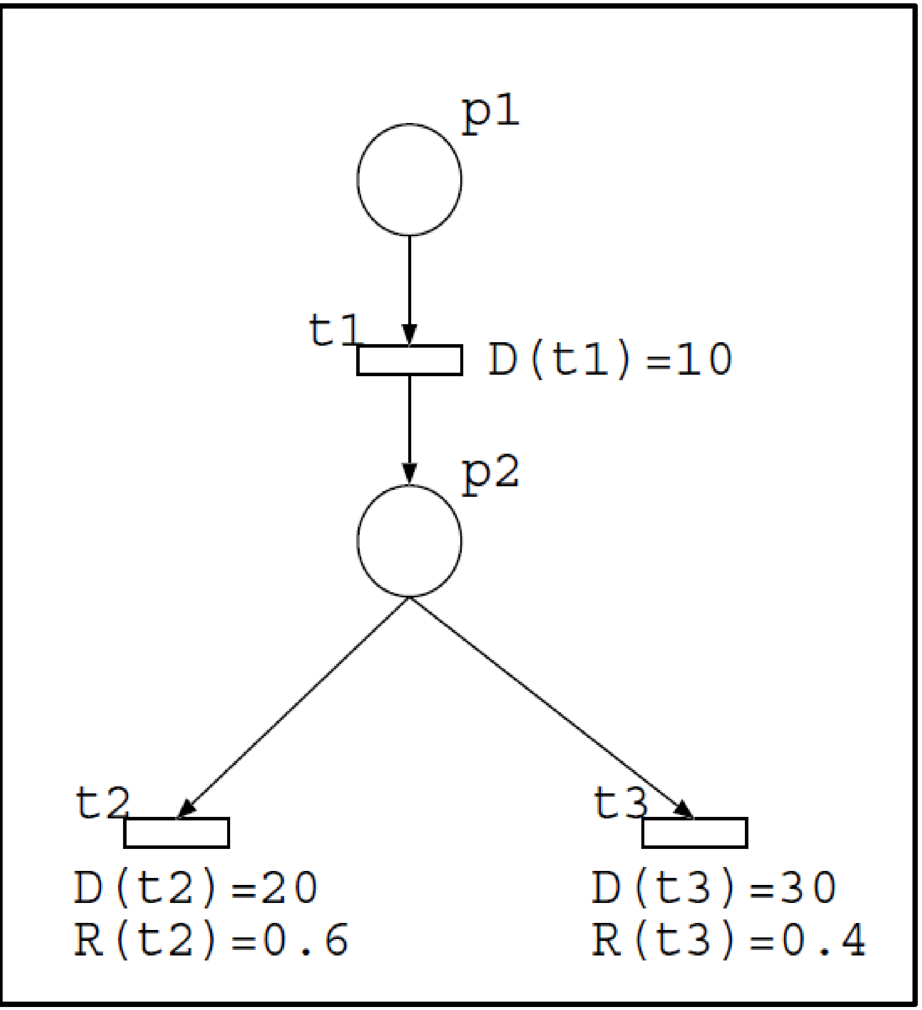
3. 6 Pattern lists

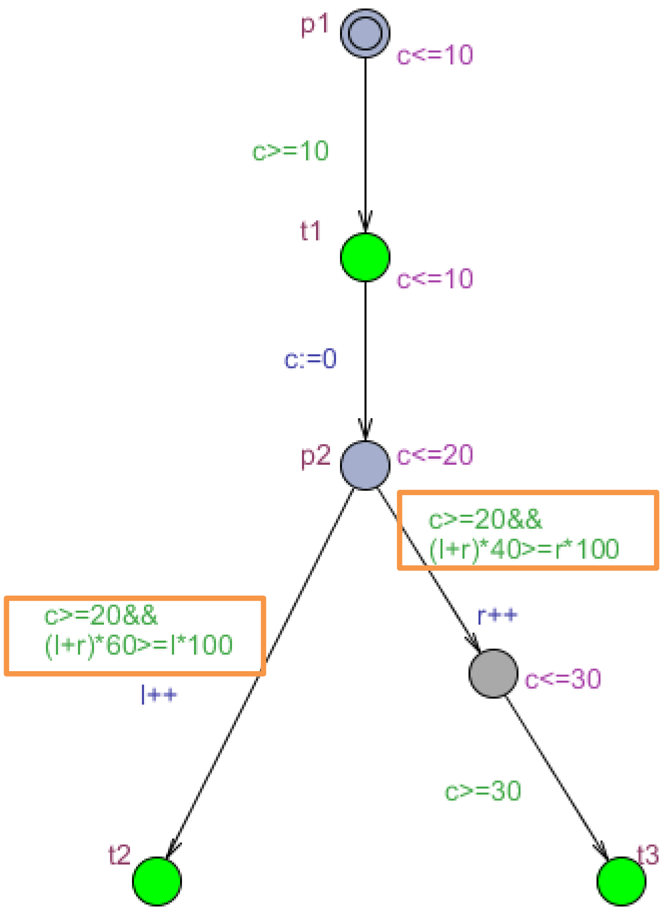
We made the pattern lists to ease the transformation.





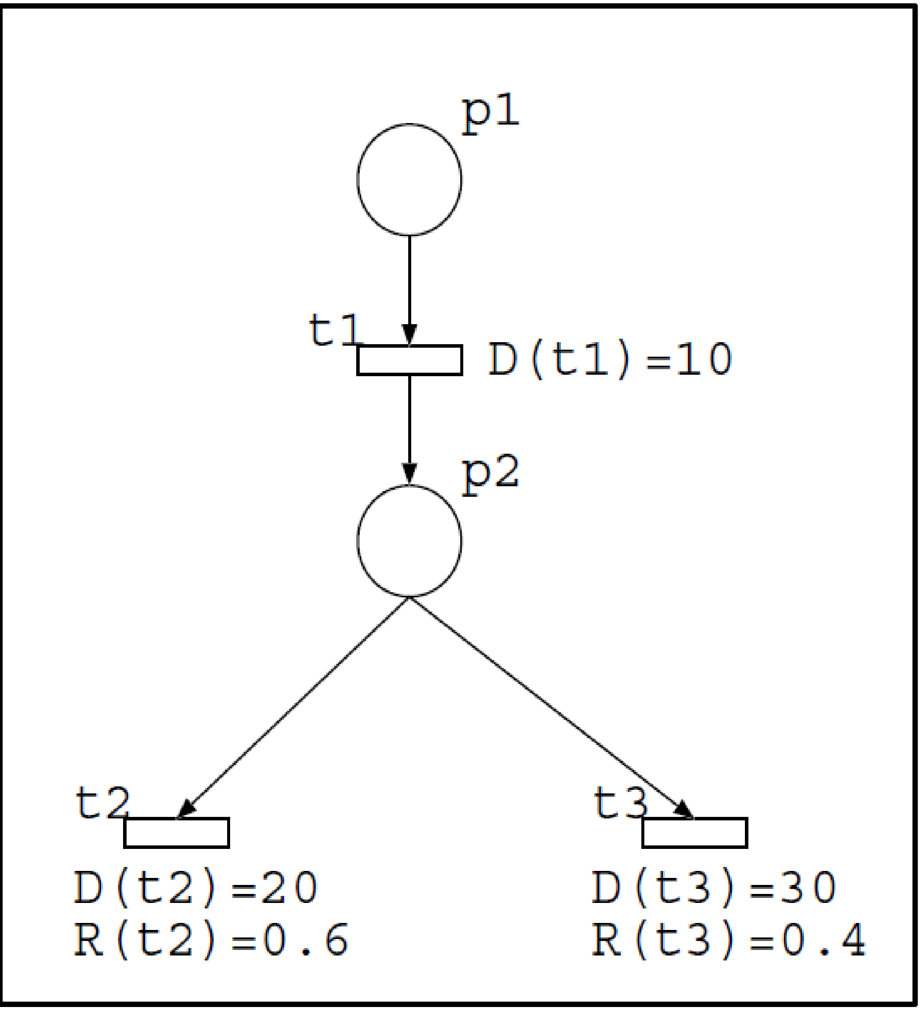
3. 6（1） A place branch Multiple automata model

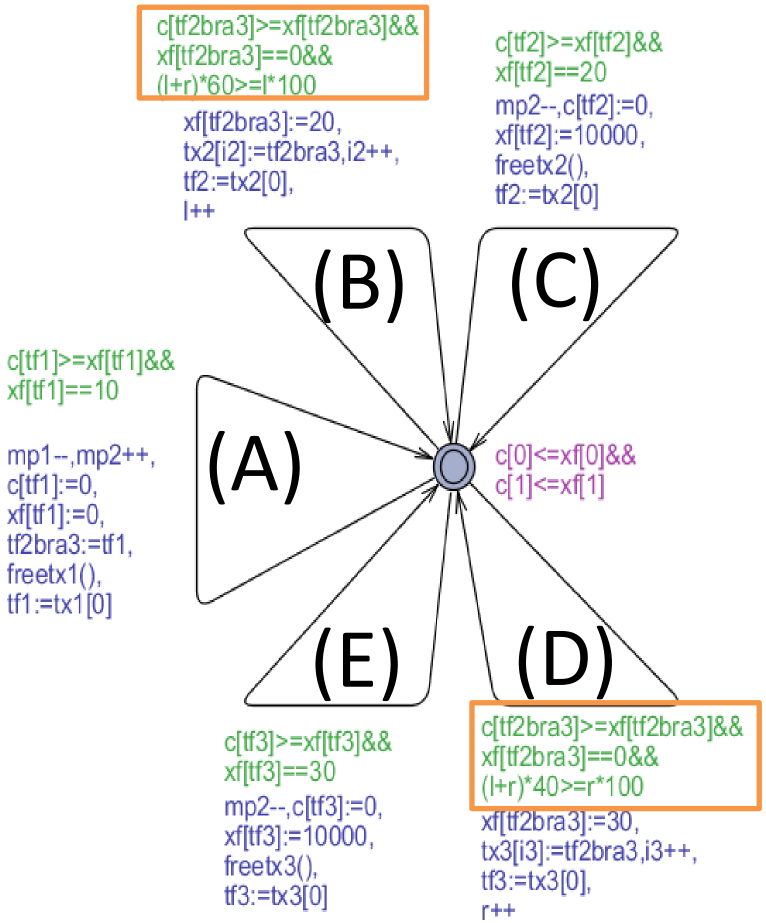




Limiting the firing count by a guard.

3. 6(2) A place branch Single automaton model





Limiting the firing count by a guard.

4. Evaluation

We compare the two modeling methods in terms of the following:

- Model size
- Model checking

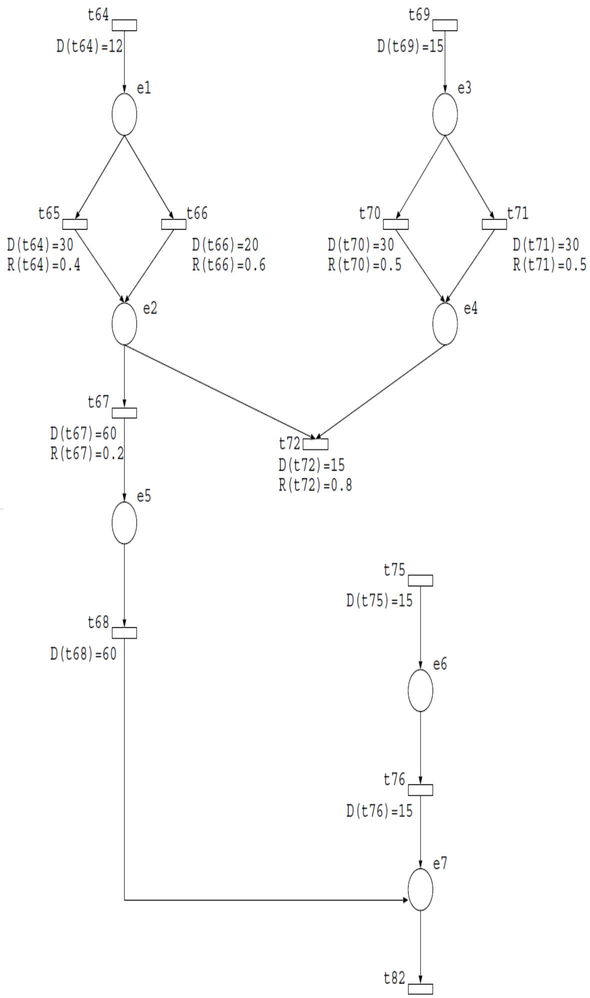
Then we discuss two modeling methods.

4．1 Model checking

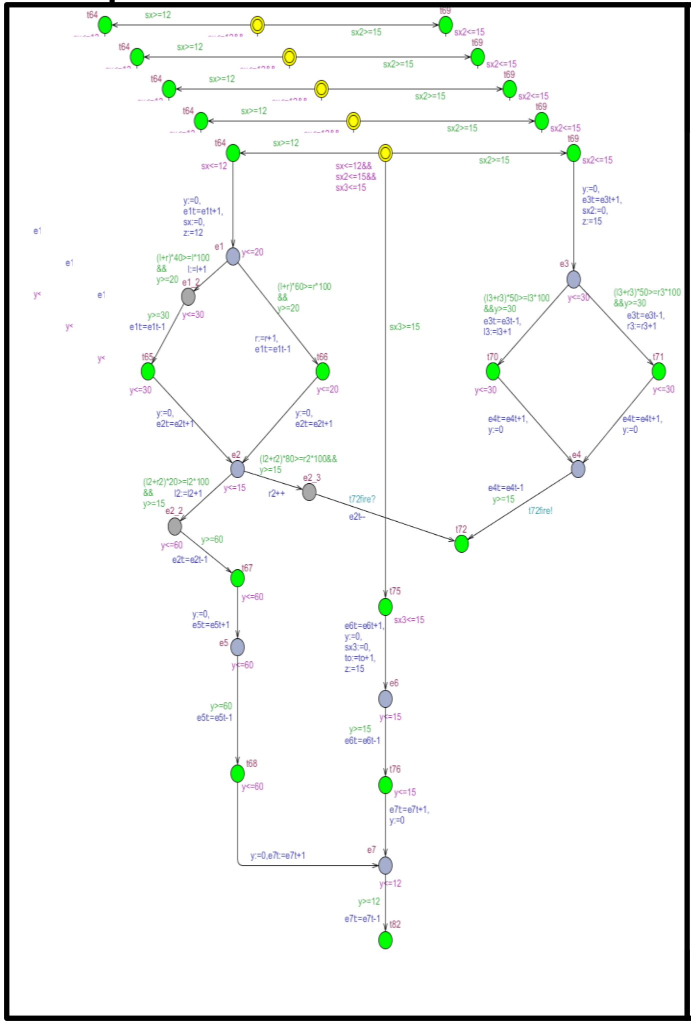
The correctness of the Petri net model of signaling pathways must be examined.
We can check the correctness by model checking.

- Reachability : E<> M(p1)>0 && c==30
  There exist a path where a signal is on p1 when clock c is 30.
- No retention : A[] M(p1)<=5
  The number M(p1) of signals for place p1 is always 5 or less.

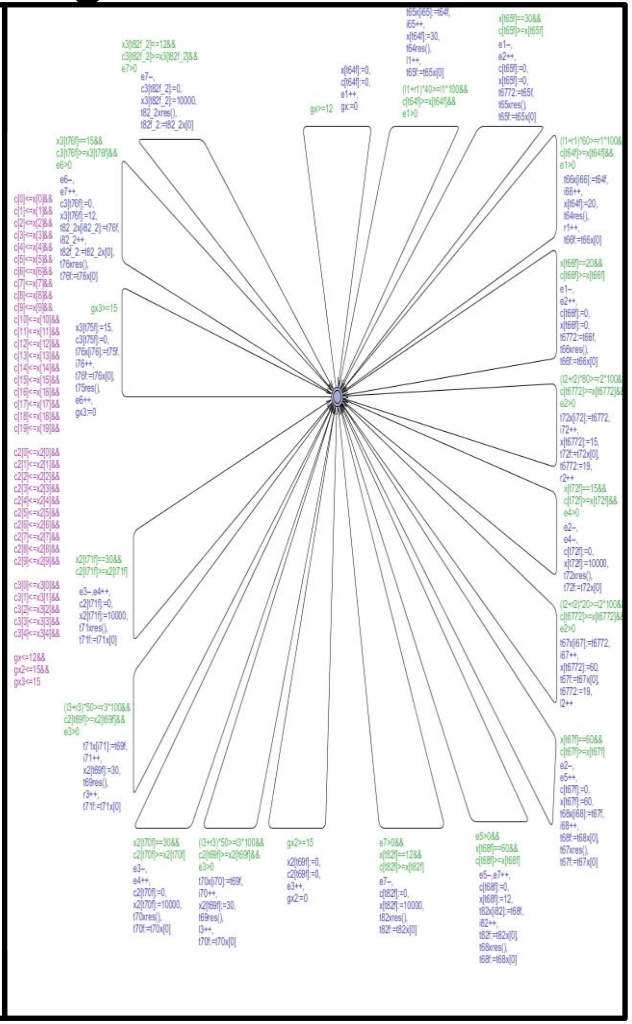
4．2 Application example1: IL-1 signaling pathway |P|=7, |T|=12



Multiple automata model

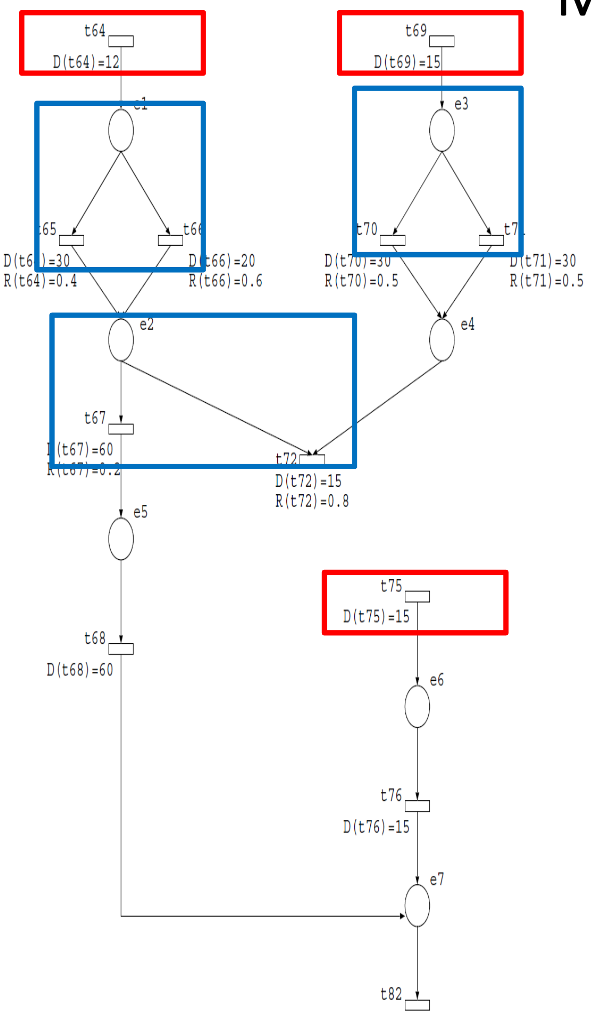


Single automaton model

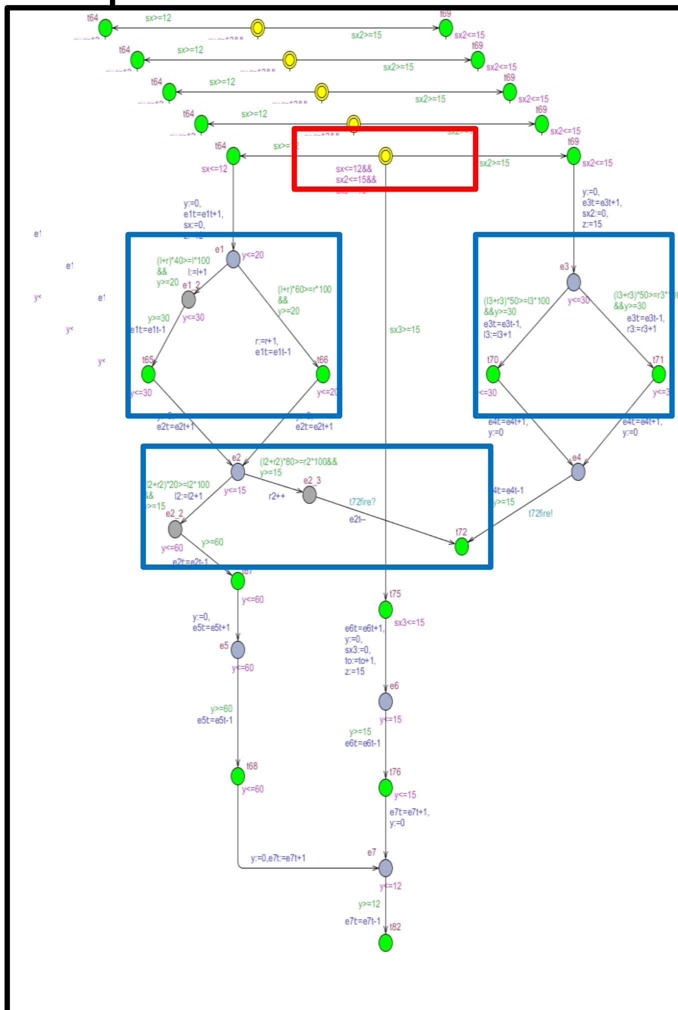


4. 2(1) Using pattern lists.

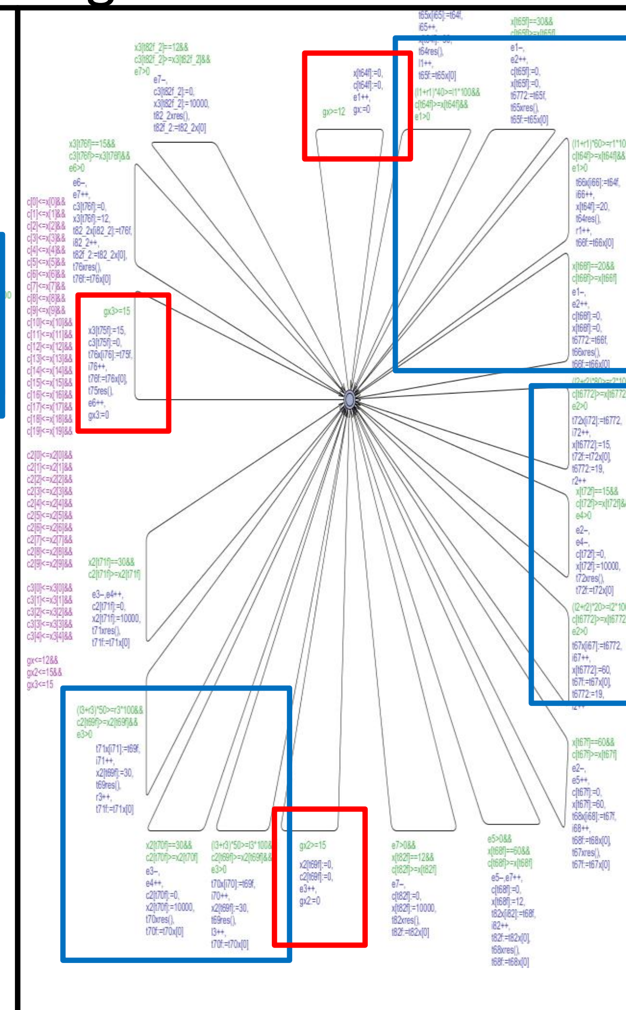
Multiple source transitions. Place branches.



Multiple automata model

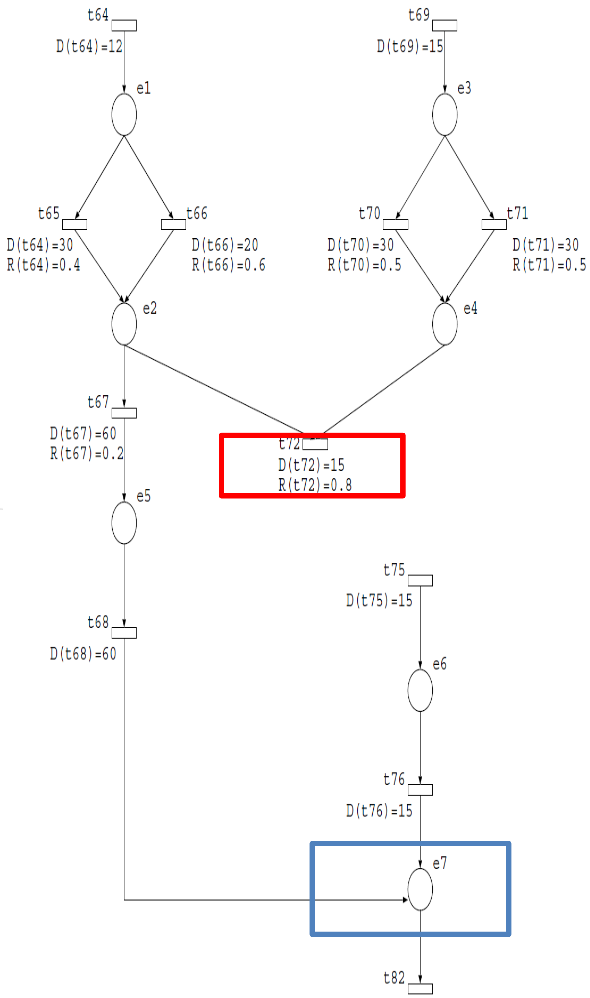


Single automaton model

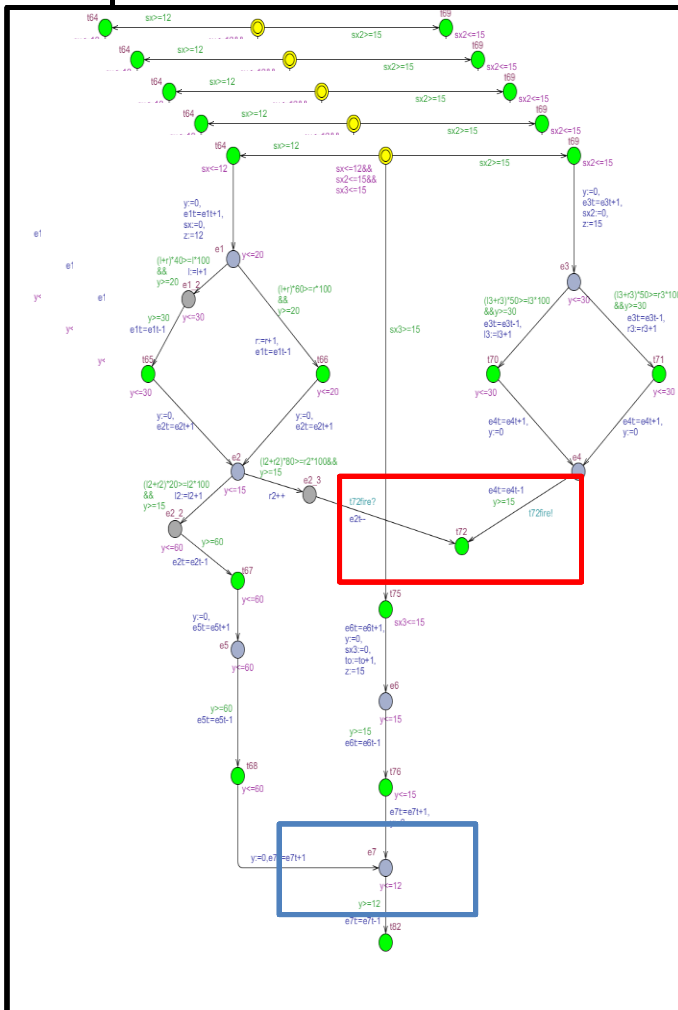


4．2（1） Using pattern lists

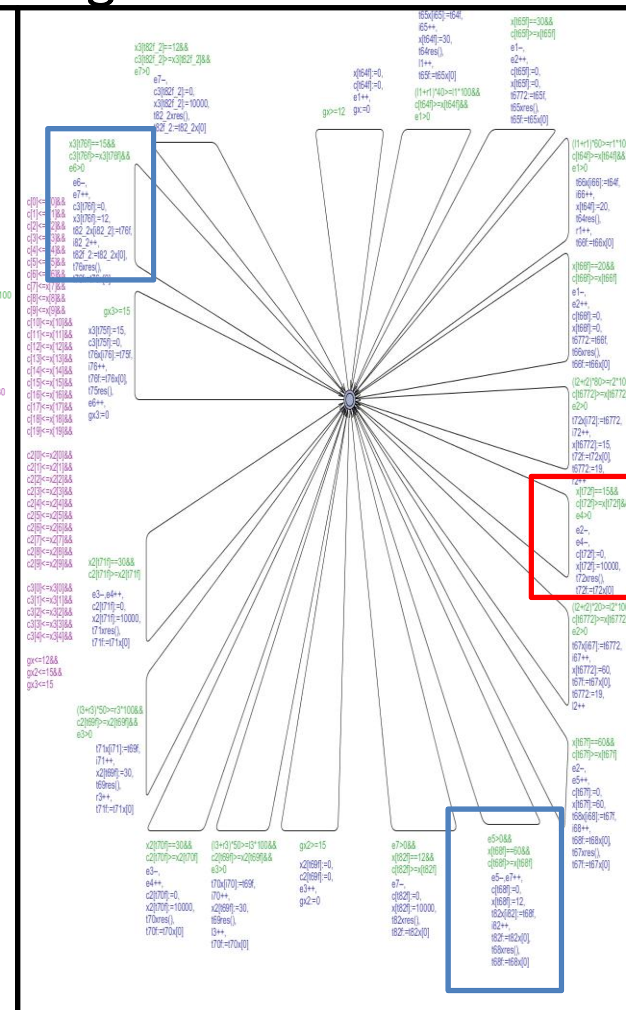
Transition joins. Place joins.



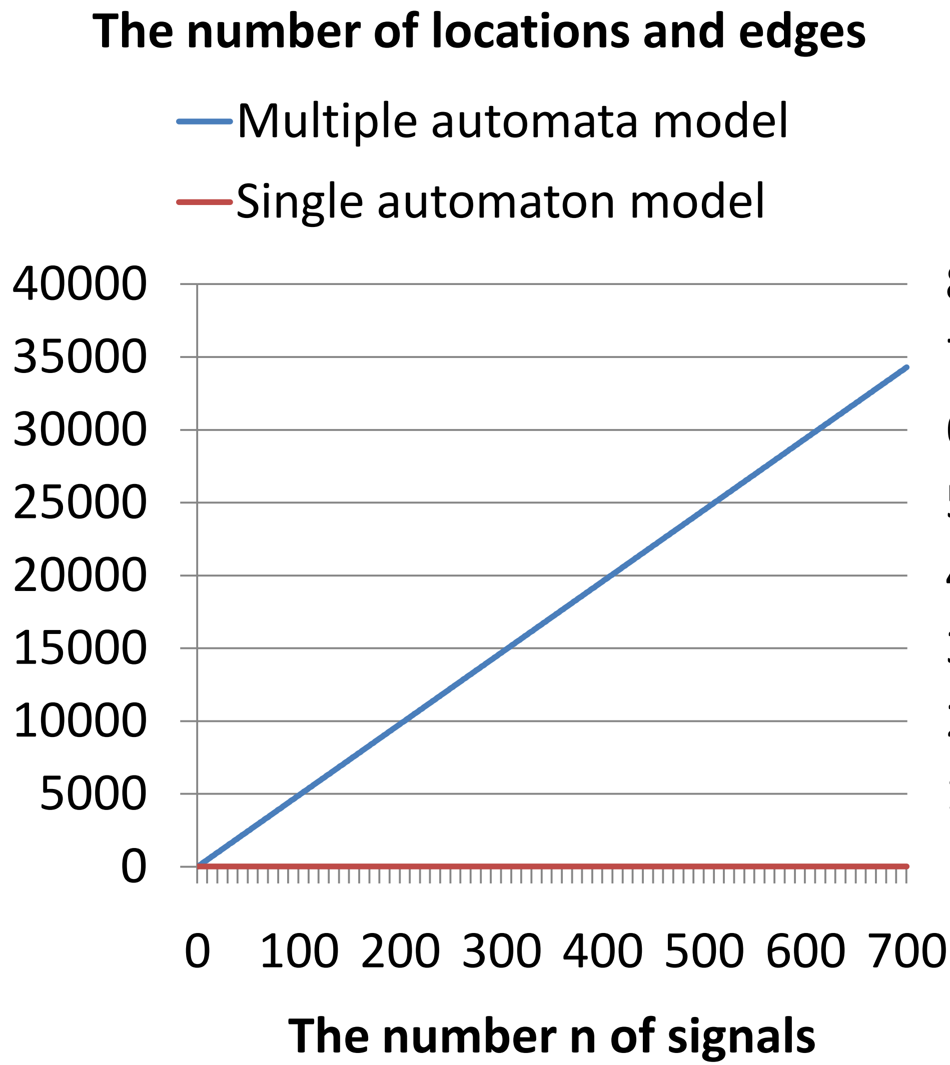
Multiple automata model



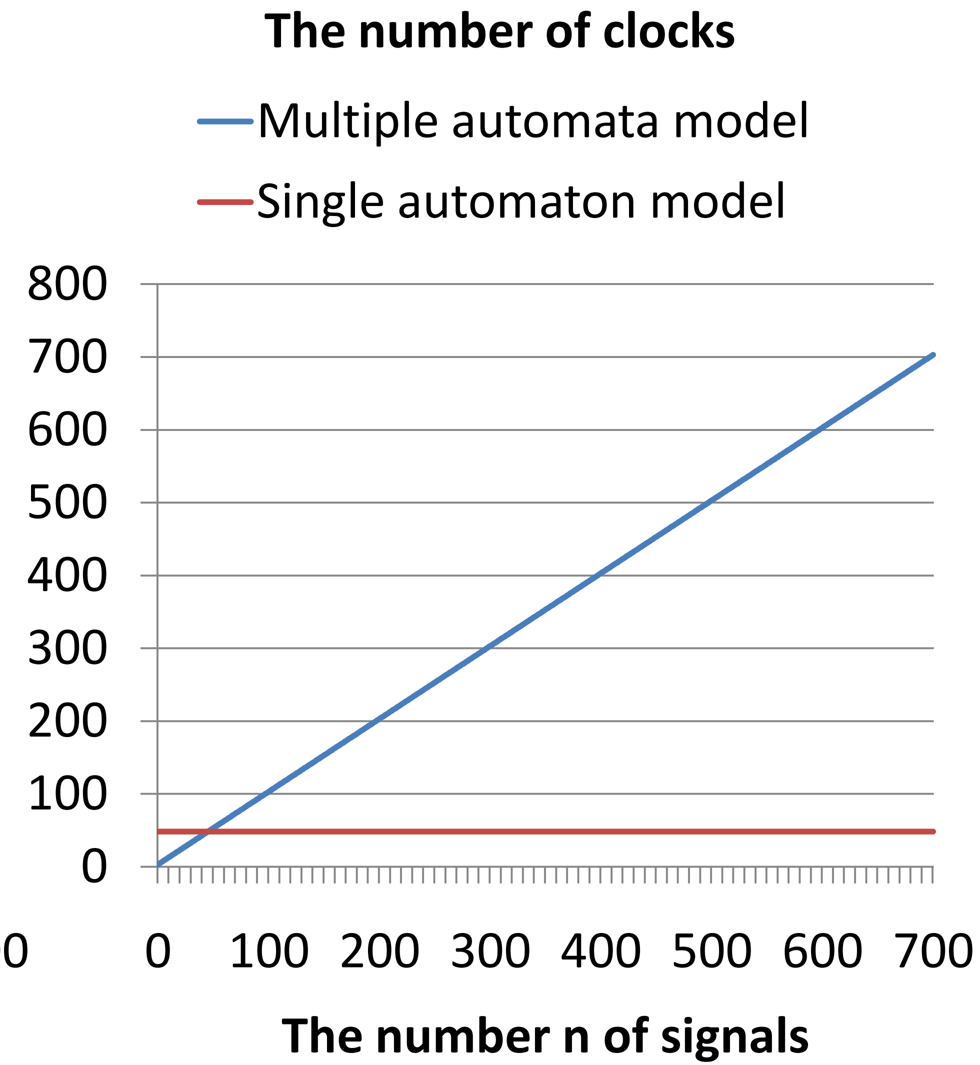
Single automaton model



4. 2(2) Comparison of model size



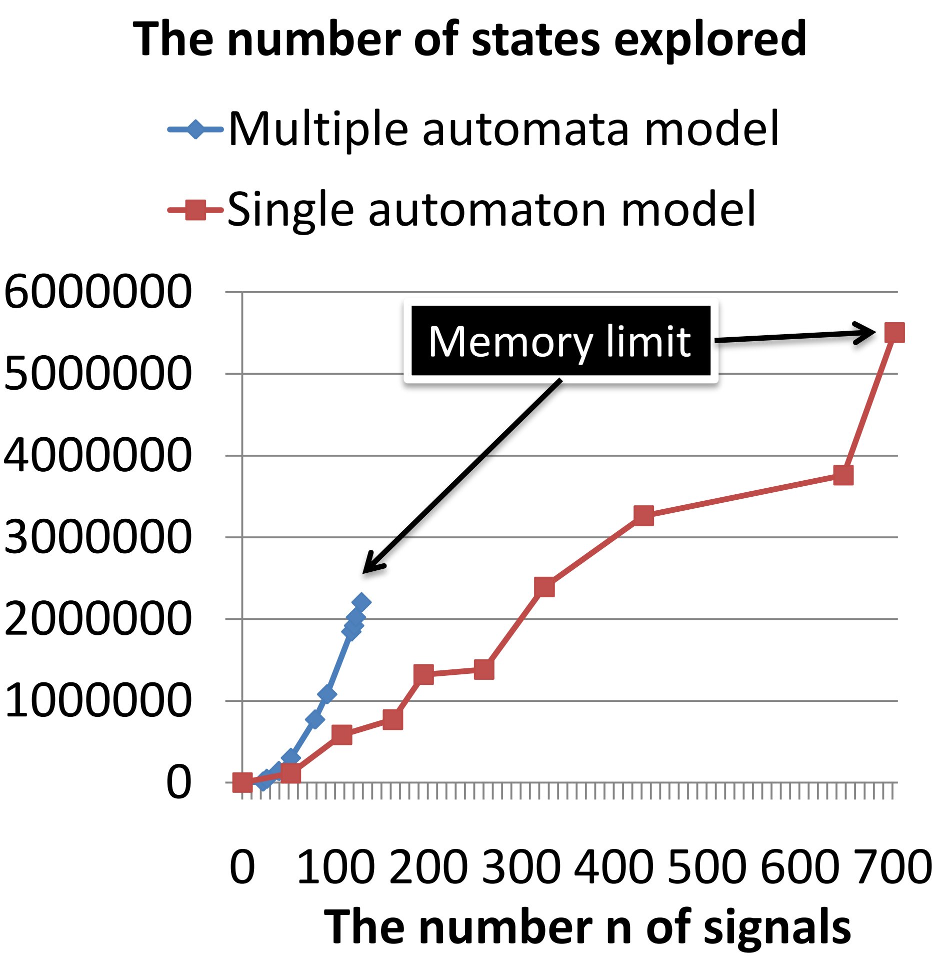
Variables : 6

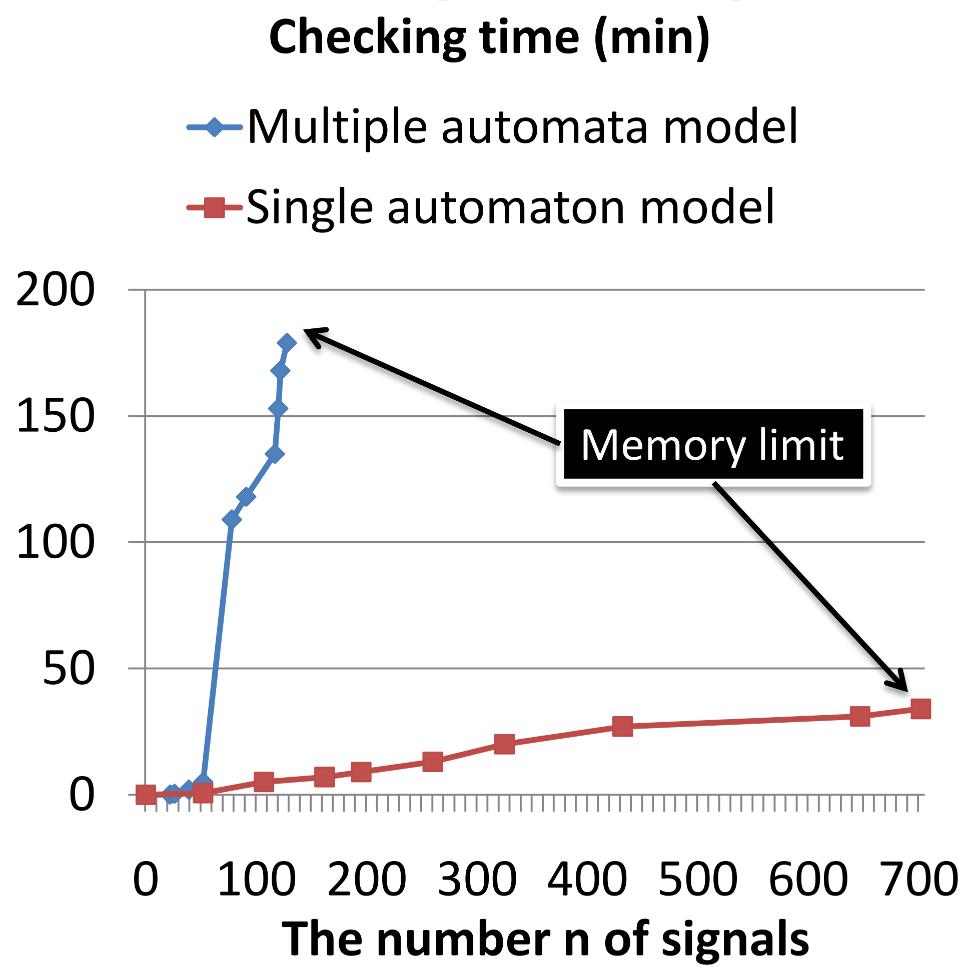


Variables : 53

4. 2(3) Comparison on model checking

We analyzed the retention property on the PC with CPU Xeon 2.13GHz and memory 3.2Gbyte.





4．3 Demonstration

4. 4(1) Discussion on modeling

The two methods have the same expressive power.

- The size of the multiple automata model is larger than that of the single automaton model.

- The number of variables of the single automaton model is larger than that of the multiple automata model.

4. 4(2) Discussion on model checking

- The single automaton model can analyze more signals than the multiple automata model.

- The single automaton model can check with half of the number of states explored, checking time than the multiple automata model.

Signaling pathways with multiple signals should be represented as not automata but variables.

5. Conclusion

- We proposed two modeling methods for signaling pathways with multiple signals.
- We applied these methods to an example.

- The model size to be analyzed can be increased by devising of modeling method.

Future work
- To develop a method treating transition branches.
- To prove whether our method can be applied to time Petri net model.