Object Enhanced Time Petri Nets

Dept. of Automation
Technical University of Cluj-Napoca Romania

1. Goals

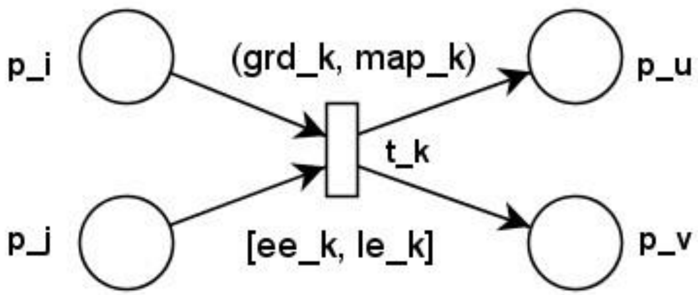
OETPN describes:

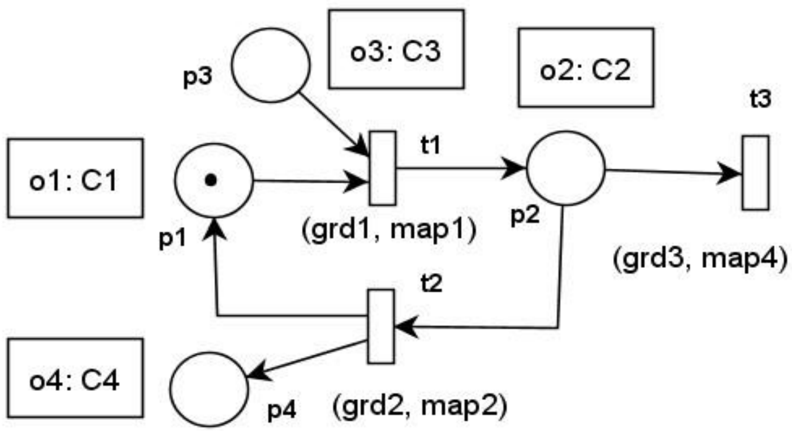
- OOP structure,
- behavior,
- the interaction between objects,
- information flows and task migration,
- modeling the object behaviors and their distributed communication.

OETPN models:

- synchronization,
- concurrency,
- synchronous and asynchronous communication,
- distributed implementation

2. OETPN





$$OETPN = (P, T, Pre, Post, Inp, Out, Types, type, \boldsymbol{Grd}, \boldsymbol{Map}, \boldsymbol{Eet}, \boldsymbol{Let}, \Lambda, \boldsymbol{M}, init, end)$$

Tokens:

- the tokens are different objects
- all the places have assigned a fixed token type
- There are two types of tokens:
  - passive tokens (references to objects)
  - The active objects are threads of executions, When their execution ends, they become passive objects (the run() method ends or interrupted)
- When a new token is set in a place where another one currently exists, the older one is replaced by the newest
- if a place has no token, it is ϕ (i.e. null)

3. OETPN Framework

OETPN executor algorithm:

Input: ***Pre***, ***Post***, M^0, *P*, *T*, *D*, *Grd&Map*, Out, Inp;

Initialization: $M = M^0$; *execList = empty;*

repeat

wait(event);

if event is *tic* **then**

* decrease the Delays of the transitions in *execList*;

else

receive(Inp);

* update **M**;

end if;

repeat

for all t_i *2 T* do

if there is met at least one *grd* in the t_i Grd_i list

for M(p), $p \in {}^o t_i$, **then**

* move the tokens of ${}^o t_i$ from **M** to M_t;

* add t_i to *execList*;

$Delay[t_i] = \delta(t_i)$;

end if;

end for;

for all t_i in execList do

if ($Delay[t_i]$ is 0) **then**

* remove *ti* from *execList*;

* calculate the tokens for **M** in t_i^o;

* remove the tokens from M_t for all ${}^o t_i$;

* set the tokens in t_i^o and start the active tokens;

end if

if $t_i \in Out$ **then**

send(Out);

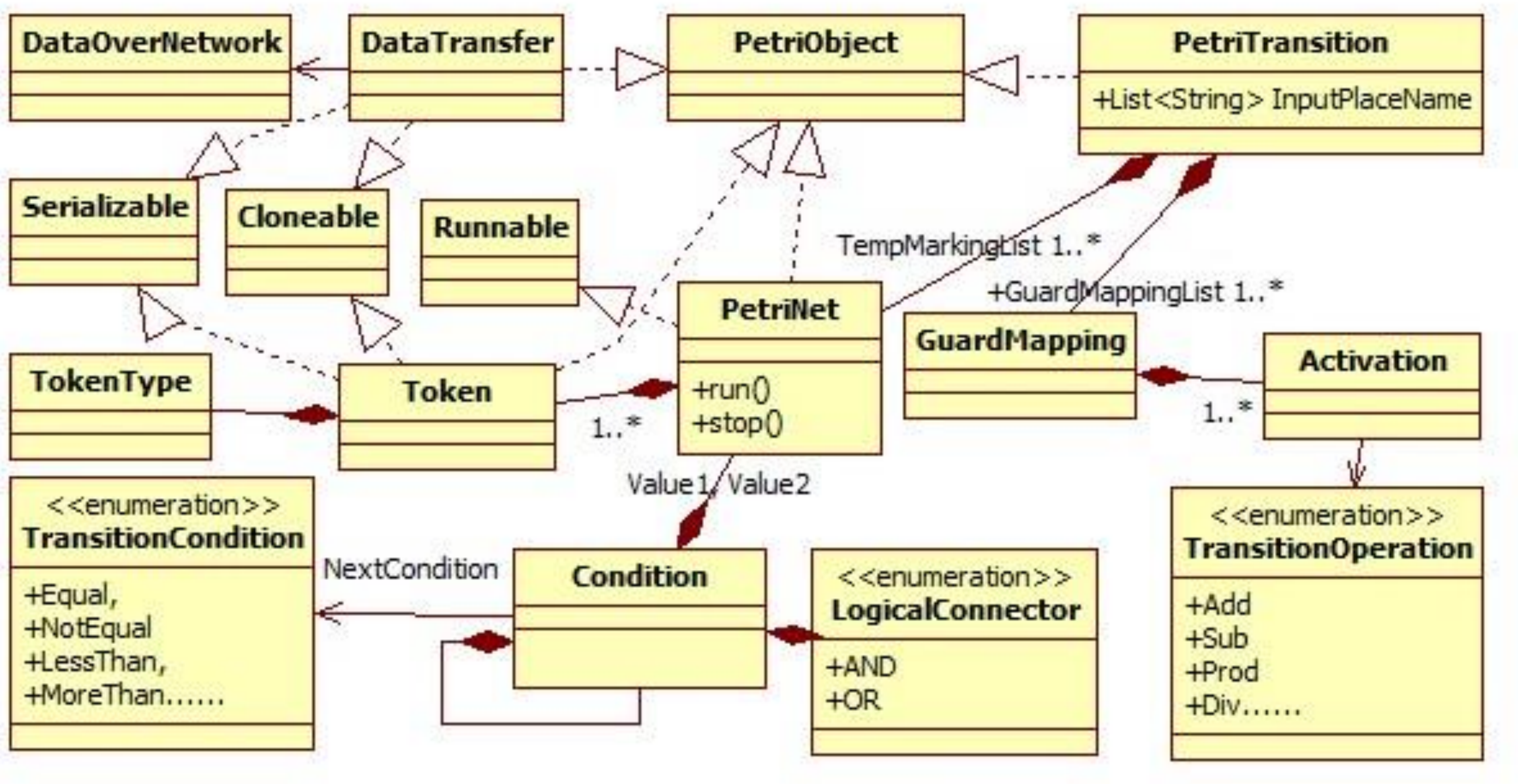
end if

end for

until there is no transition that can be executed;

until the time horizon;

END algorithm;


DataOverNetwork
DataTransfer
PetriObject
PetriTransition
+List<String> InputPlaceName
Serializable
Cloneable
Runnable
TempMarkingList 1..*
+GuardMappingList 1..*
PetriNet
+run()
+stop()
GuardMapping
Activation
1..*
TokenType
Token
1..*
Value1, Value2
<<enumeration>>
TransitionCondition
+Equal,
+NotEqual
+LessThan,
+MoreThan......
NextCondition
Condition
<<enumeration>>
LogicalConnector
+AND
+OR
<<enumeration>>
TransitionOperation
+Add
+Sub
+Prod
+Div......

The transition, guard and map synthesis

Transition algorithm:

Input: $^{o}t_i$, t^{o}_i, g_i, Pre, Post, M^0, P, T, D, Grd&Map, network_flag, Out, Inp;

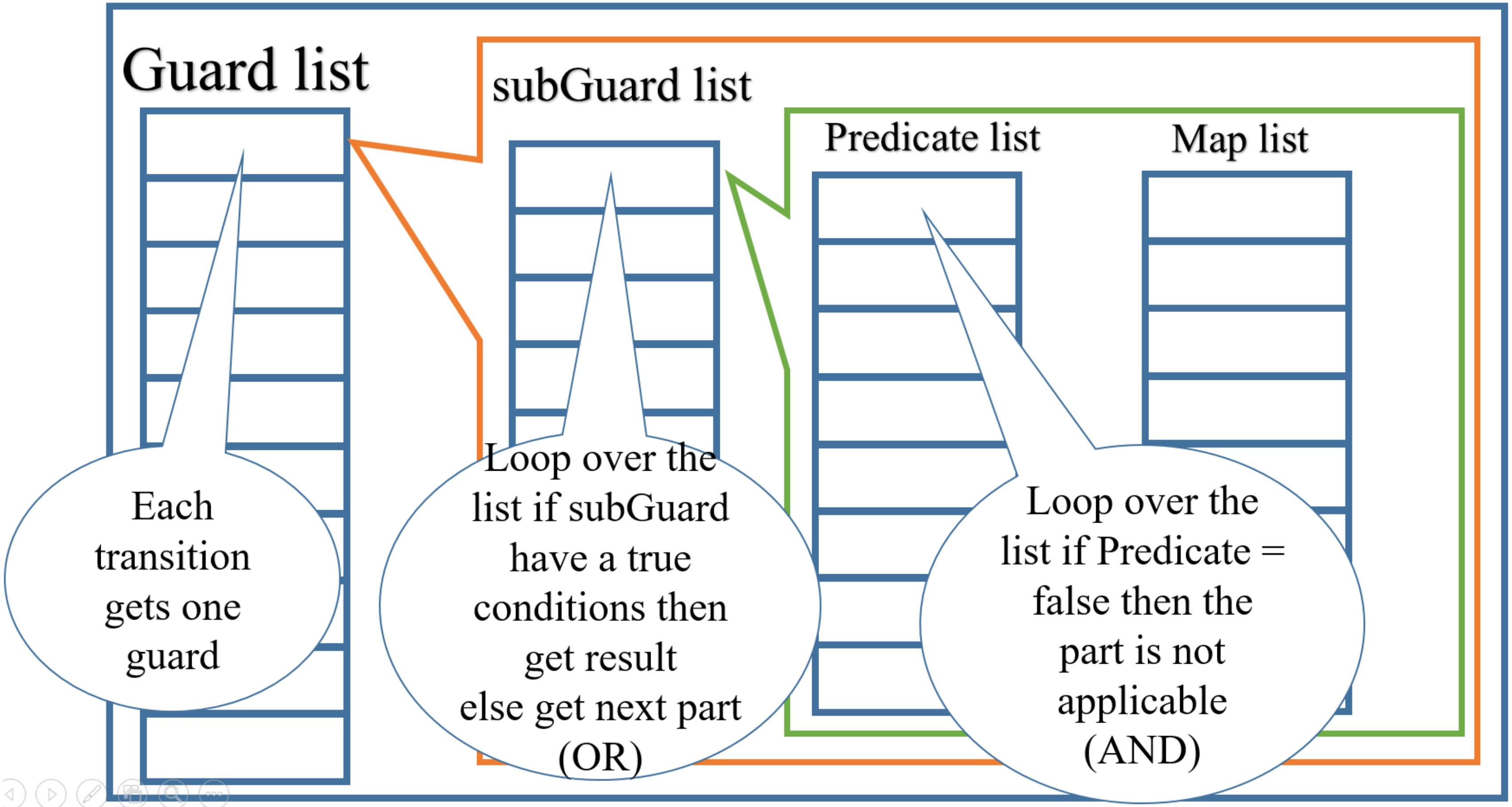
Initialization: $M = M^0$, execList = empty;

```
for all gᵏᵢ ∈ gᵢ do
for all πˡi ∈ gᵏᵢ
    if !πˡi then
                    exit  //covers (and) operation
    end if
end for
for all p ∈ tᵒ  // the count of mapping list is equal the Post
        if map = object then //a voctor of float or a Boolean object
      *Set object to Mi
            if network_flag = true then
         Send object over network
                    end if
      else
        if map = active OETPN  then
           *Start new thread with sub OETPN
        else
            *Set passive OETPN marking to Mi  //this case of OETPN is passive
                      end if
    end if
end for
End for
```

A transition guard and map example:

guard: (M(p1)!=null) AND (M(p2)<1);   map1: M(p3)=M(p2)

predicate part   connector   predicate part



Experiments:

A. Parent and child creation

The parent places have the types:

- $type(p_1) = type(p_3) = type(p_4) = type(p_5) = float$; the tokens are denoted by x_1, x_3, x_4 and x_5 respectively.
- $type(p_2) = OETPN$; the token is denoted by PN_2

The child places have the types:

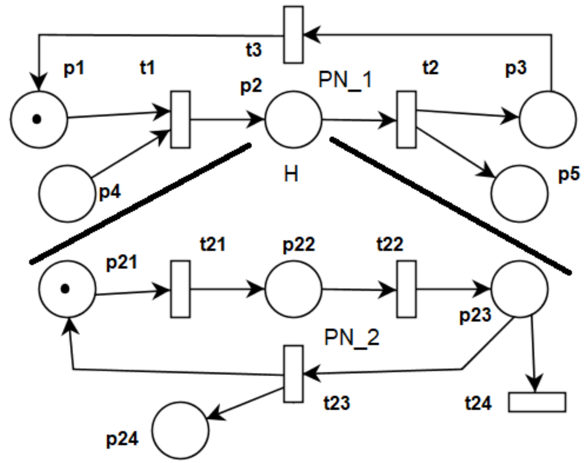
- $type(p_{21}) = type(p_{22}) = type(p_{23}) = type(p_{24}) = float$; the tokens are denoted by x_{21}, x_{22}, x_{23} and x_{24} respectively.

The parent guards and mappings are:

- $grd_1^1 = (x_1 \neq \varphi)$ AND $(x_4 \leq 1)$; map_1^1 instantiates the child $M(p_2) = PN_2$ being an active object with the marking $M^2 = [x_4, 0, 0, 0]$ (i.e. $M(p_{21}) = M(p_4)$ and the running state true).
- $grd_1^2 = (x_1 \neq \varphi)$ AND $(x_4 > 1)$; map_1^2 instantiates the child $M(p_2) = PN_2$ being a passive object with the marking $M^2 = [x_4, 0, 0, 0]$ (i.e. $M(p_{21}) = M(p_4)$ and running state *false*.
- All the rest transitions have the guards *true* and the mappings copy the values of the transition input places in its output places.

The child guards and mappings are:

- $grd_{21} = true$; map_{21}: $x_{22} = x_{21}$;
- $grd_{22} = true$; map_{22}: $x_{23} = x_{22} + 0.1$;
- $grd_{23} = x_{23} < 2$; map_{23}: $x_{21} = x_{23}$;
- $grd_{24} = x_{23} \geq 2$; **StopPetriNet()**



B. Concurrent task execution

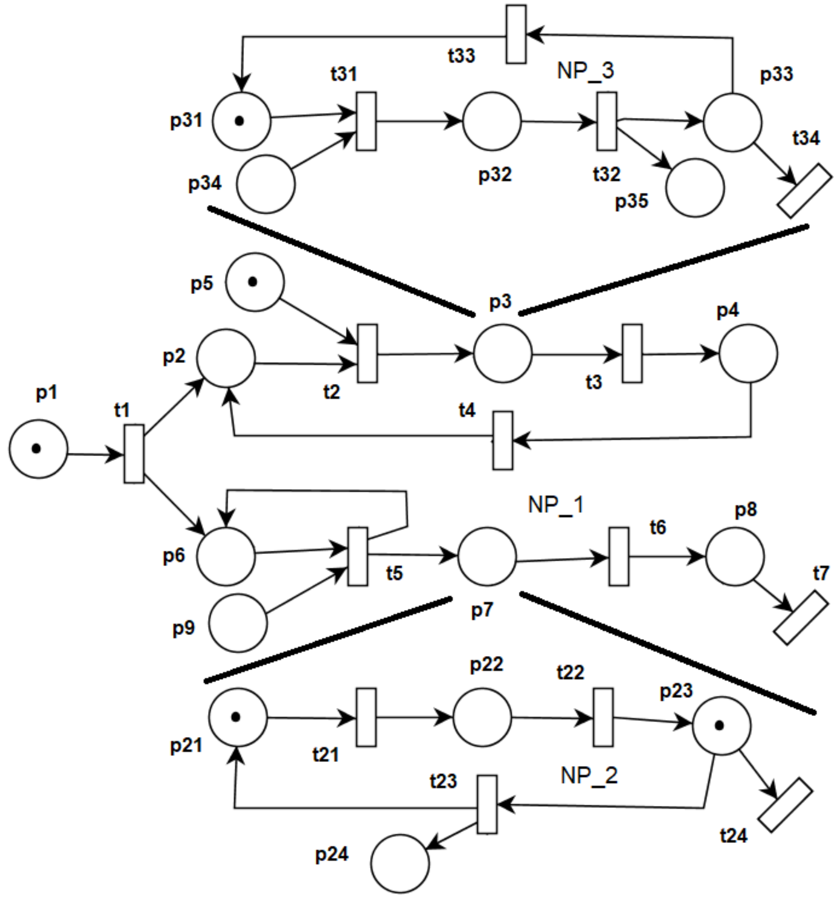
The child **NP₂** is similar to the previous example with the difference that its output place is linked to NP_3 input place p_{34}.

The types of the **NP1** places p_3 and p_7 are tasks: type(p_3) = NP_3 and type(p_7) = NP_2. The rest of the places are of the types float.

The parent's (NP1) places p_5 and p_9 are linked to the keyboard input channel. The guards and the mappings of the transitions t_2 and t_6 are similar to the previous example.

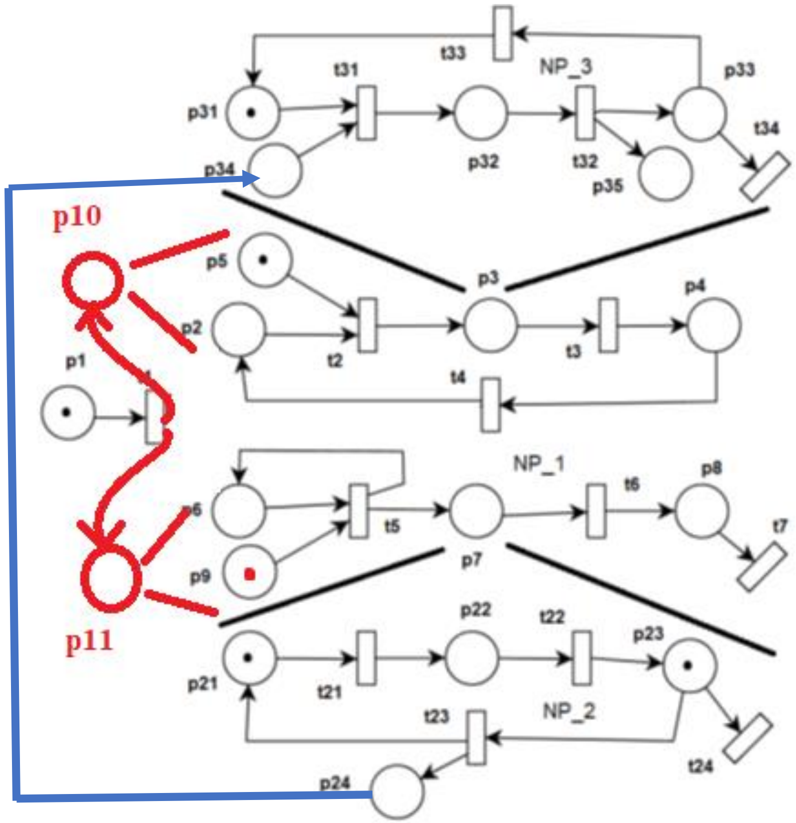
The guards and mappings of NP_3 are:

- grd_{31} = *true*; map_{31}: $x_{32} = x_{31} + x_{34}$.
- grd_{32} = *true*; map_{32}: $x_{33}=x_{32}$, $x_{35}=x_{32}$;
- $grd_{34} = x_{33} > 3$; **StopPetriNet()**
- $grd_{33} = x_{33} \leq 3$; map_{33}: $x_{31} = x_{33}$;



In implementation:

NP_1 is divided into two threads of execution, so p1, t1, p10, and p11 are considered the starting thread, the upper pard is a thread starts from p10 and the lower starts from p11 just as shown in the figure below:



References:

[1] Tiberiu S. Letia, and Dahlia Al-Janabi, "Object Enhanced Time Petri Nets Models", IEEE International Conference on Automation, Quality and Testing, Robotics (AQTR), IEEE, DOI: 10.1109/AQTR.2018.8402743, May 2018.

[2] The OETPN Framework: https://bitbucket.org/dahliajj/oetpn_oertpn_framework/src/master/