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, 𝑮𝒓𝒅, 𝑴𝒂𝒑, 𝑬𝒆𝒕, 𝑳𝒆𝒕, 𝛬, 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****, M0, P, T, D, Grd&Map,* Out, Inp;

*Initialization: M = M0; execList = empty;*

**repeat**

*wait(event);*

**for** all *ti* in execList do

**if** (*Delay[ti]* is 0) **then**

\* remove *ti* from *execList*;

\* calculate the tokens for **M** in ;

\* remove the tokens from *Mt* for all *oti*;

\* set the tokens in and start the active tokens;

**end if**

**if** ti ∈ *Out* **then**

*send(Out);*

**end if**

**end for**

**until** there is no transition that can be executed;

**until** the time horizon;

**END** algorithm;

**if** event is *tic* **then**

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

**else**

*receive(Inp);*

\* update **M**;

**end if;**

**repeat**

**for** all *ti 2 T* do

**if** there is met at least one *grd* in the *ti Grdi* list

**for** M(p), *p* ∈ *oti*, **then**

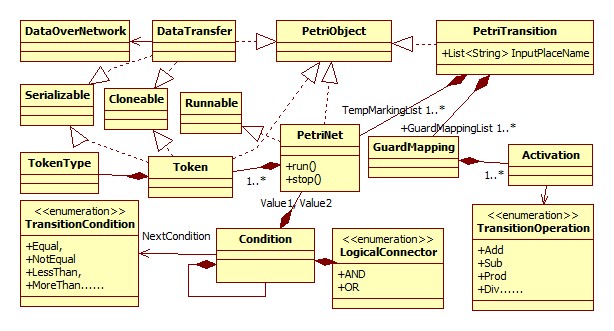
\* move the tokens of *oti* from **M** to *Mt*;

\* add *ti* to *execList*;

*Delay[ti] = δ (ti);*

**end if**;

**end for**;



# The transition, guard and map synthesis

*Transition algorithm:*

Input: oti, toi, gi, Pre, Post, M0, P, T, D, Grd&Map, network\_flag, Out, Inp;

Initialization: M = M0, 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**

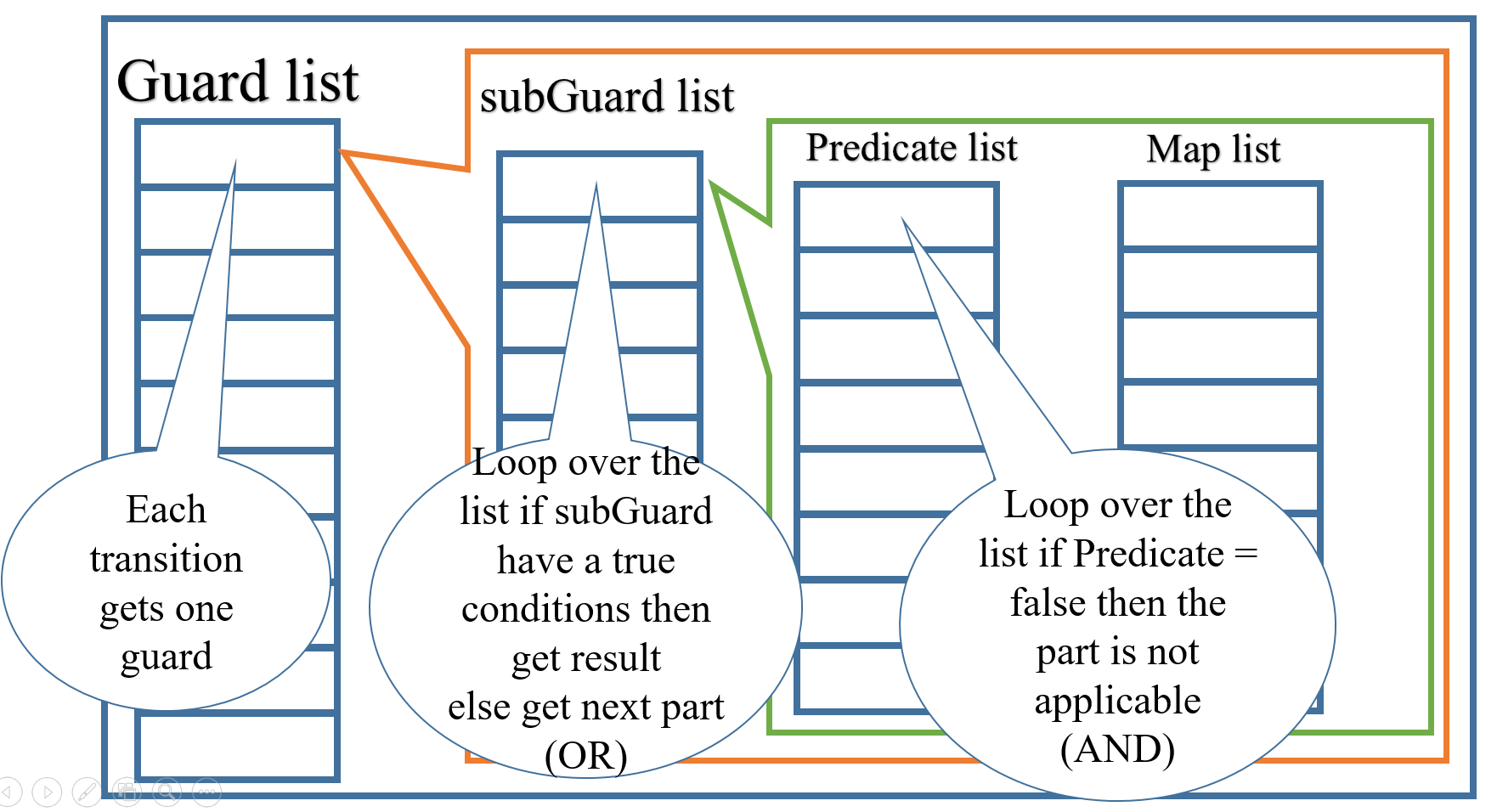
**End for**

**A transition guard and map example:**

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

predicate connector predicate

part part



**Experiments:**

**A. Parent and child creation**

The parent places have the types:

* type(p1) = type(p3) = type(p4) = type(p5) = float; the tokens are denoted by x1, x3, x4 and x5 respectively.
* type(p2) = OETPN; the token is denoted by PN2

The child places have the types:

* type(p21) = type(p22) = type(p23) = type(p24) = float; the tokens are denoted by x21, x22, x23 and x24 respectively.

The parent guards and mappings are:

* grd11= (x1 ≠ φ) AND (x4 ≤ 1); map11 instantiates the child M(p2) = PN2 being an active object with the marking M2 = [x4, 0, 0, 0] (i.e. M(p21) = M(p4) and the running state true).
* grd12= (x1 ≠ φ) AND (x4 > 1); map12 instantiates the child M(p2) = PN2 being a passive object with the marking M2 = [x4, 0, 0, 0] (i.e. M(p21) = M(p4) 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:

* grd21 = true; map21: x22 = x21;
* grd22 = true; map22: x23 = x22 + 0.1;
* grd23 = x23 < 2; map23: x21 = x23;
* grd24 = x23 ≥ 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 NP3 input place p34.

The types of the **NP1** places p3 and p7 are tasks: type(p3) = NP3 and type(p7) = NP2. The rest of the places are of the types float.

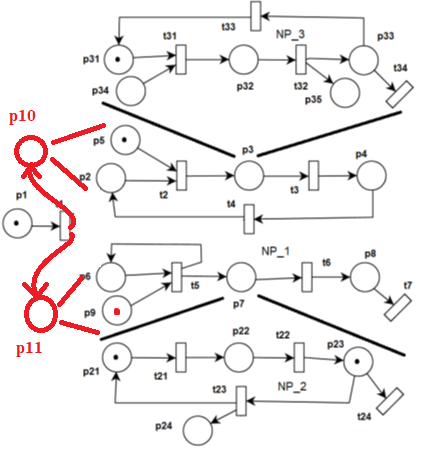
The parent’s (NP1) places p5 and p9 are linked to the keyboard input channel. The guards and the mappings of the transitions t2 and t6 are similar to the previous example.

The guards and mappings of NP3 are:

* grd31 = *true*; map31: x32 = x31 + x34.
* grd32 = *true*; map32: x33=x32, x35=x32;
* grd34 = x33 > 3; **StopPetriNet()**
* grd33 = x33 ≤ 3; map33: x31 = x33;

**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/>