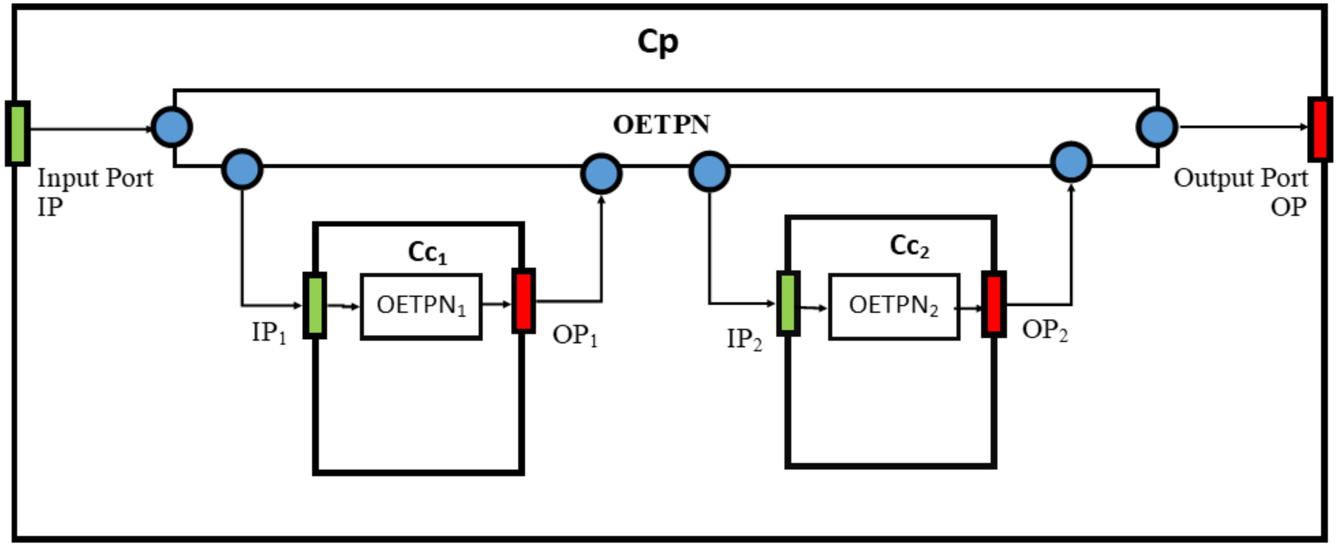
4.10 Object Enhanced Time Petri Net Capsules (OETPN-Cs)

They are kinds of components integrating OETPN models and other OETPN-C with particular features.

Goal: OETPN-Cs can be used for specification, design, verification and implementation of concentrated or distributed reactive software applications.

Capabilities: An OETPN-C can describe and implement asynchronous and synchronous reactions to external or internal events, using its concurrent tasks or another OETP-Cs that it integrates.

OETPN-C Structure



An OETPN-C could have:

- ***Input ports*** for receiving synchronous or asynchronous information from sensors or another OETPN-C of the types:
    - *Float* in the domain [-1,1]
    - Integers
    - Objects of specified types.
    - The input ports can be synchronous or asynchronous storing synchronous information or asynchronous information.
- ***Output ports*** for sending synchronously or asynchronously information, to another OETPN-C or actuators, of the types:
    - *Float* in the domain [-1,1]
    - *Integers*
    - *Object* of the specified types.

- The output ports can be synchronous sending information at the tics, or asynchronous sending further the information immediately.

- ***OETPN*** linked to Input ports, output ports and another OETPN-Cs that it integrates. It can contain:
  - Asynchronous tasks,
  - Synchronous tasks.
  - They can intercommunicate through inside places of different specified types.

The input ports can store synchronous information that requires the reaction at the OETPN-C ticks (generated by its own real-time clock), or asynchronous information that requires immediate reaction.

The integrated OETPN model could have two kinds of transitions: asynchronous (T_a) and synchronous (T_s). $T = T_a \cup T_s$. The asynchronous transitions can be executed in any moment of time when its input places fulfil the enabling conditions, unlike synchronous transitions that can be executed only at the ticks. The synchronous transitions can be delayed with integer delay $\{0, 1, 2, \ldots\}$.

Behavior

The OETPN-C behavior is implemented by the integrated OETPN model.
The OETPN model reacts synchronously or asynchronously to:

- An external event signaled through an input port (from capsule frontier or an integrated capsule). The event can be set in an asynchronous port or a synchronous one.
- A tick event of the capsule real-time clock.

The reaction to an asynchronous event is immediately processed by asynchronous transitions or by synchronous transitions at the ticks. When a new token is asynchronously injected in a place, only the asynchronous transactions are verified for enabling.
The reaction to synchronous events are performed by synchronous transitions at the ticks. When a new token is synchronously injected in a place, all kinds of transitions a verified for enabling.
As flowing,

OETPN-C executor algorithm:
Input: ***Pre***, ***Post***, $M_0, P, T = T_a \cup T_s, D,$ ***Grd&Map***, Out,
$Inp = Inp_a \cup Inp_s$;
Initialization: $M = M^0$, *execList* $= empty$;

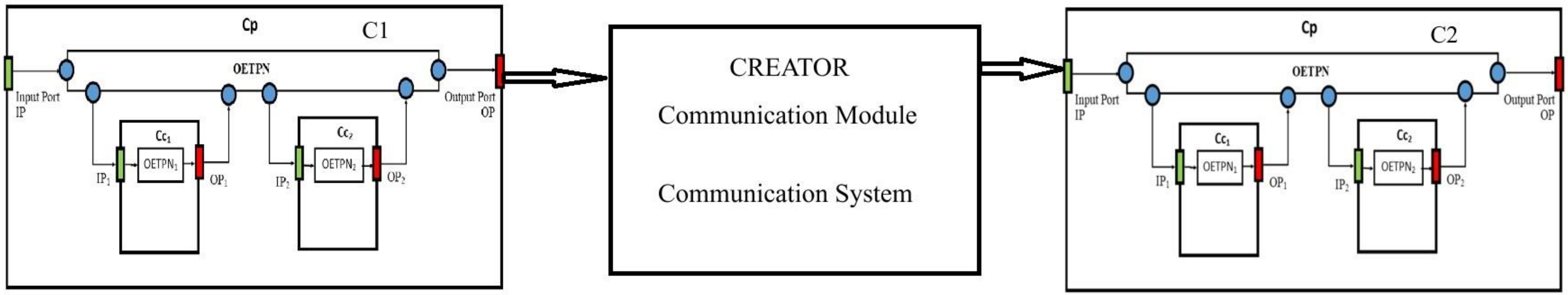
repeat
  wait(event);
  if event is $tick$ **then**
    * decrease the Delays of the transitions in *execList*;
    repeat
      for all $t_i \in T$ **do**
        if there is met at least one *grd* in the t_i Grd_i list for M(p), $p \in {}^o t_i$, **then**
          * move the tokens of ${}^o t_i$ from 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 t_i 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 $t_i^o \in Out$ **then**
          $send(Out)$;
        end if
      end for
    until there is no transition that can be executed;
  else
    $receive(Inp_a)$;
    * update M;
    repeat
      for all $t_i \in T_a$ **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**
          * remove the tokens of ${}^o t_i$ from M;
          * calculate and add the tokens of t_i^o to M;
          if $t_i^o \in Out$ **then**
            send(Out);
          end if
        end if;
      end for;
    until there is no transition in T_a that can be executed;
  end if;
until the time horizon;
END algorithm;

Implementation

Each OETPN-C is executed by its thread (OETPN-C executor). Each of its OETPN-C children is created by the parent and executed by a thread.

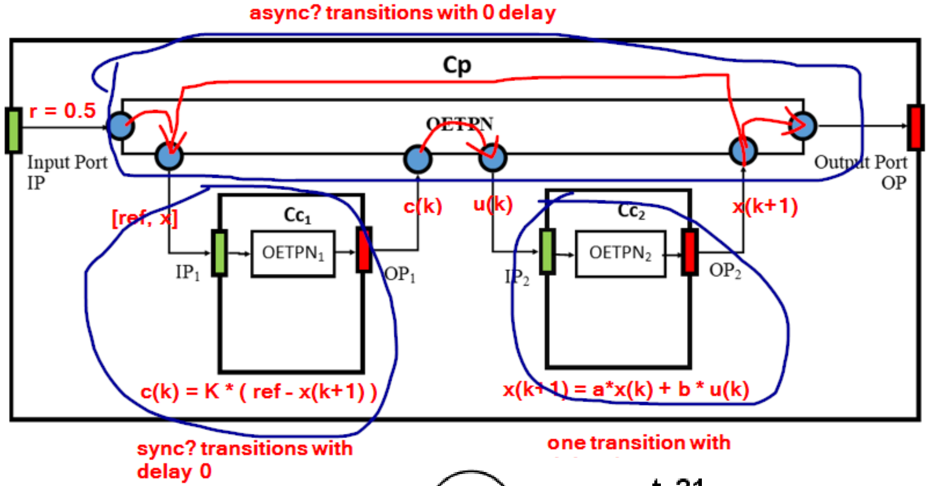
The inter OETPN-C communication at the same level is performed by capsules creators (OETPN-C parent, main thread) or the linked operation systems.

The methods *send(Out)* and *receive(Inp)* are used for this purpose.



Exemples

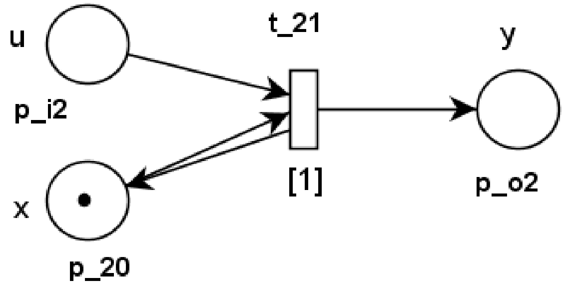
Implement on a dual cores system:



0) Control of a first order system with a proportional controller.

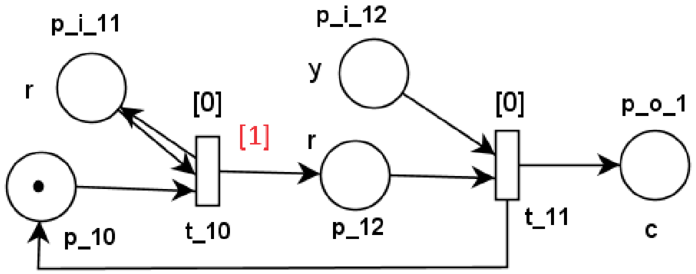
Cc2: plant model:

$x(k+1)=a\cdot x(k) + b\cdot u(k); y=x(k+1); x(0)=0.1;$



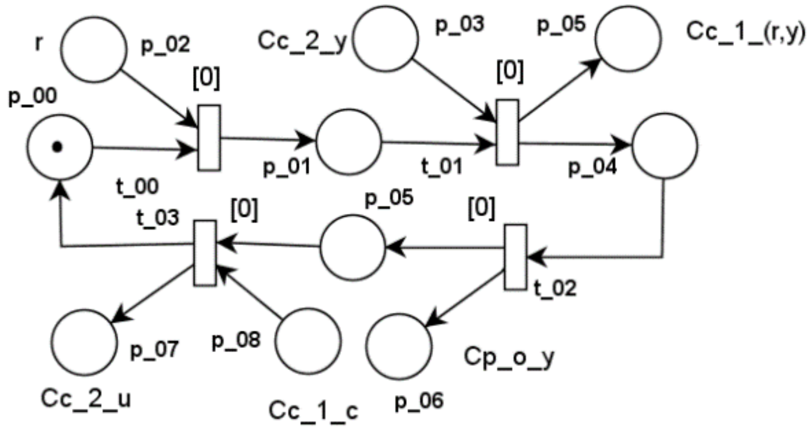
Cc1: Controller) model

$c(k)= K\cdot(r(k)-y(k));$



Cp: Model

Read *r* from operator and *y* from plant (Cc2).
Send to controller *(r,y)*. Display on screen *(r,c) or (r,c,y)*.
Get *c* from controller and apply *c* on Cc2 (plant) through input *u*.



Analiza temporală a exemplului 0

Ritmul schimbării timpului este dat în exemplul curent doar de Cc2 *plant*. Cp si Cc1 se supun modificărilor date de temporizarea din plant Cc2. Cp se execută cu perioada 1 (u.t.) dacă se ignoră lipsă lui *r* în intrarea tranziției t_00.

Cp se poate concepe şi să execute concurent modificarea referinței cu controlul execuțiilor componentelor Cc1 și Cc2.

Cc2 (Plant) are o temporizare intrinsecă (naturală). Cc1 ar trebui să aibă propriul său ritm pe care să-l impună şi pentru Cc2. Pentru așa ceva ar trebui să se schimbe temporizarea tranziției t_10 cu [1]. $OETPN_1$ și $OETPN_2$ nu sunt sincronizate deși ambele evoluează cu o perioadă de 1 u.t. (unitate de timp) dacă tranziția t_00 are şi o condiție de gardă care ignoră lipsa jetonului *r* în p_02.

1) Modelul unui controller multi-variabil. Cp primește două intrări de erori (e_1, e_2) şi generează la ieșire două semnale de control (c_1, c_2). Cc_1 care realizează $c_1 = F_1(e_1)$, iar Cc_2 care realizează $c_2 = F_2(e_2)$ lucrează în paralel. Duratele de calcul ale celor două funcții sunt d_1 și respectiv d_2.

Varianta 1: Când ambele semnale sunt disponibile, Cp aplică la ieșire simultan comenzile (c_1, c_2). Constrângerea temporală: comanda este aplicată la $maxim\{d_1, d_2\}$.

Varianta 2: Dacă ambele semnale sunt disponibile, după o secundă de la încărcarea portului de intrare cu (e_1, e_2) se generează la ieșire (c_1, c_2). Condiții temporale: $d_1 < 1$ secundă, $d_2 < 1$ secundă.

2) Cp primește la intrare (e_1, e_2) și generează la ieșire c. Cc$_1$ realizează $c_1= F_1(e_1)$, iar Cc$_2$ realizează $c = F_2(e_2,c_1)$. Prin urmare controllerele Cc$_1$ şi Cc$_2$ lucrează în cascadă. Duratele de timp pentru F_1 și F_2 sunt cele de mai sus.
Varianta 1: Comanda se aplică imediat. Întârzierea temporală este: $d_1 + d_2$.
Varianta 2: Comanda se aplică după o secundă. Constrângerea temporală este: $d_1 + d_2 < 1$ sec.

3) Cp primește la intrare (e_1, e_2) și generează la ieșire (c_1, c_2). Cc$_1$ realizează $a_1= F_1(e_1, e_2)$, iar Cc$_2$ realizează $a_2= F_2(e_2, c_1)$. La ieșire se aplică $c_1= F_3(a_1, a_2)$ și $c_2= F_4(a_1, a_2)$. Duratele pentru F_1, F_2, F_3 şi F_4 sunt d_1, d_2, d_3 şi respectiv d_4.
Varianta 1: Comenzile se aplică imediat cu întârzierea: $max\{d_1, d_2\} + max\{d_3, d_4\}$.
Varianta 2: Comenzile se aplică după o secundă. Constrângerea temporală: $max\{d_1, d_2\} + max\{d_3, d_4\}< 1sec.$

4) Dacă variabilele (fie acestea x şi y) implicate sunt de tip sincron (deci au întârziere de cel puțin o unitate de timp), relația dintre ele este $y(k+d)= F(x(k)$.

5) Dacă variabilele implicate sunt de tip asincron (fără întârziere între ele) şi valoarea ceasului de timp este k, relația dintre ele este $y(k)=F(x(k))$. Se presupune că evenimentul care a generat (modificat) valoarea lui x s-a produs într-un moment de timp între k și $(k+1)$.

6) Dacă variabila x este de tip asincron, iar variabile modificată y la momentul k este de tip sincron, relația dintre ele este $y(k)= F(x)$ cu modificarea lui y în momentul producerii

evenimentului care generează (modifică pe) *x*. Se presupune că *y* are întârziere zero față de *x*, prin urmare timpul nu se modifică, deși valoare lui *y* se modifică.

Problemă: Discernerea precedenței dintre două evenimente de intrare asincrone care s-au produs între aceleaşi ticuri ale ceasului.
Se pot folosi 6 variabile x, x', y, y', z, v. Evenimentul de intrare e_1 modifică valorile variabilelor x, x' şi v. Producerea evenimentului e_1 realizează (x,x') =F_1(x, e_1); (x,v)=F_3(x,y',e_1). Producerea evenimentului e_2 modifică variabilele y, y' şi z realizând relațiile (y,y')=F_2(y, e_2); (y,z)=F_4(y, x',e_2). Folosind valorile perechii (v,z) la momentul de timp (k+1) se poate discerne care a fost mai întâi e_1 sau e_2.

Variabilele
Toate variabilele aplicației sunt de tip vector *Fuzzy Logic (FL). Fuzzy Logic Set (FLS)*. El reprezintă in jeton.
FLS={NL, NM, ZR, PM, PL}
Toate variabilele sunt în domeniul *[-1,1]*. Se dau formule de scalare pentru intrări şi ieșiri.
Jetonul (tip unar) de valoare „1" este un caz particular: *1=[0,0,0,0,1]*. Adică *Pozitiv Large* este 1.
Variabilele sunt modificate doare de mappingurile modelelor OETPN.
Mappingurile pot fi de tip *FLRS (Fuzzy Logic Rule Set)* combinate cu operații aritmetice de tip: +, – , ·, ∴
Variabilele pot fi stocate în locații (engl.: places) sau în porturile de intrare sau de ieşire.

Variabilele pot fi modificate doar de tranzițiile modelelor OETPN.

Propunere: Variabilele de tip sincron să fie adnotate cu „1”, iar cele de tip asincron cu „0”. Semnificația este: reacția la o intrare sincronă este tratată de o tranziție care poate avea întârzierea diferită de zero. Reacția la o intrare asincronă cere o acțiune imediată (aperiodic), deci cu întârziere „0” unități de timp.

Parametrii unei componente

Parametrii modelului OETPN sunt: numărul n al tranzițiilor, numărul m al locațiilor, matricele *Pre*, *Post*, vectorii *Inp*, *Out*, *Delay*, marcajele locațiilor din canalele de intrare care sunt sincrone sau asincrone şi tranzițiile care sunt sincrone sau asincrone, precum şi ponderile W_i ale tranzițiilor t_i. Mappingurile tranzițiilor sunt de asemenea parametrii.

Pentru FLETPN acestea sunt $FLRS_i$. Platforma curentă are mappinguri simplificate acceptând doar operații aritmetice şi FLRs.

Parametrii componentelor pot fi setați offline de către *Operator 2* sau de către *Genotype Mapping* înaintea fiecărui nou test. Fiecare componentă are metode pentru setarea parametrilor.

Simulator

Are porturi de intrare cu *Training Data* şi *Genetic Algorithm*.

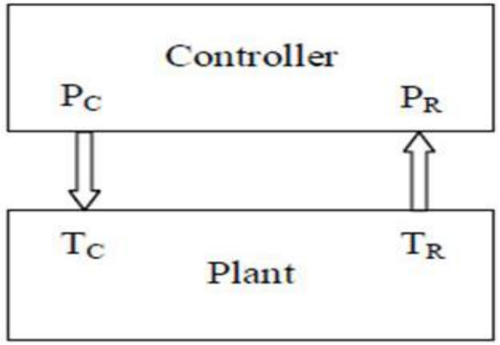
Are porturi de ieşire cu *Individuaal Evaluator*.

Alimentează cu variabile de intrare portul de intrare al componentei principale (numită *mainComponent*) folosind informațiile din *Training Data*.

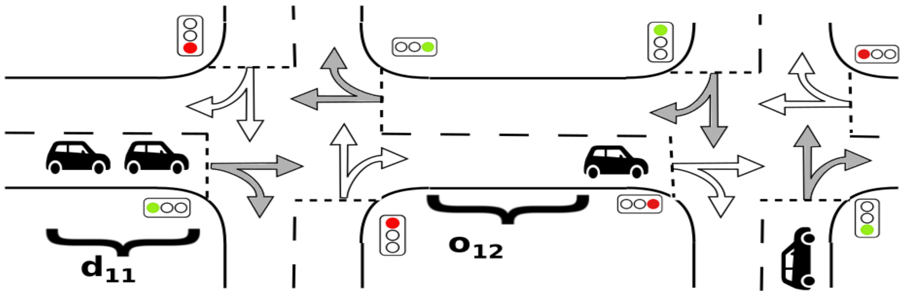
Lansează execuția lui *Plant Model*. Furnizează datele de ieșire la *Individual Evaluator*.

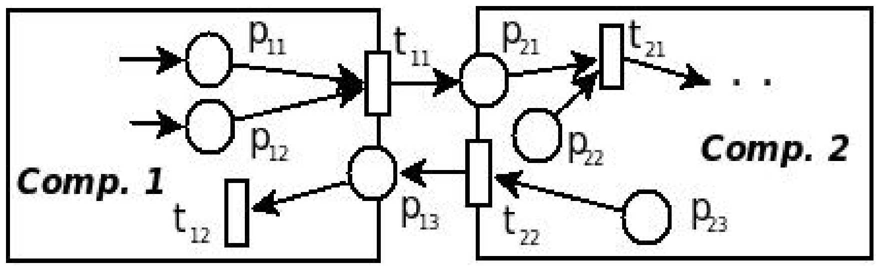
Unified Enhanced Time Petri Nets

- To obtain models suitable for solving problems arising in hybrid control applications
- Models capable to describe the
  - reactions to
    - asynchronous events
    - continuous changing of some measured variables
  - fulfilling some temporal and performance requirements



Reactive programs ➔ event-driven approach
Reactive systems ➔ message-driven approach





Requirements of reactive applications

Reactive applications → local: event-driven
→ distributed: message-driven

Models capable to describe:

- Reaction to different event with different intensities
- Asynchronous reactions to continuous variables

Models capable to describe:

- The structure
- The concurrent dynamic behavior
- Reaction to internal and external events
- Selection based on internal or external results
- Synchronizations
- Temporal behavior with
    - Fixed and
    - Variable delays

- **Scalable**
- **Resilient**
- **Elastic**
- **Responsive**

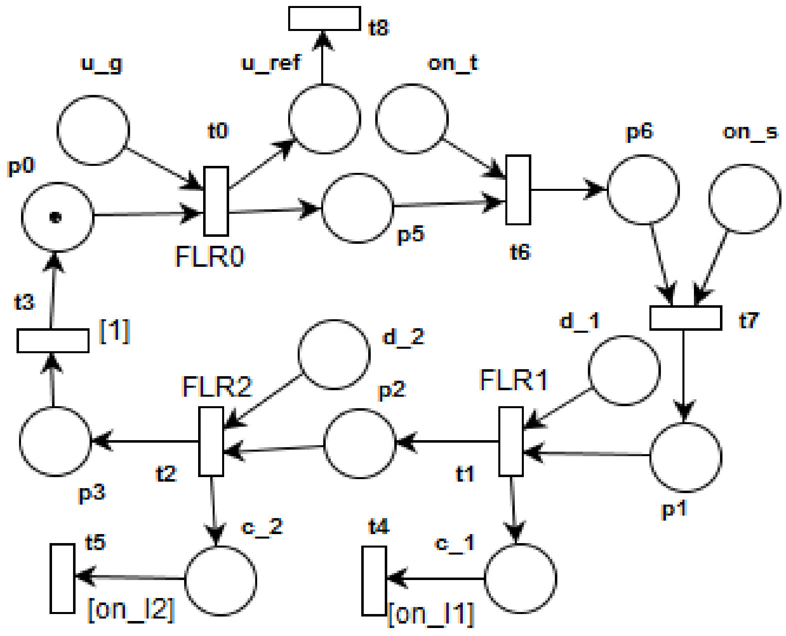
Justifications of UETPN:

1) FLETPN models comfortably describe the conditions for transition enabling.
2) FLETPN models unify the description of different kinds of variables. They can describe real value numbers.
3) FLETPN models describe logic operations, but some applications need arithmetic operations: +, -, * and /.
4) UETPN models combine the FLETPN features with arithmetic operations.
5) UETPNs can describe *time-discrete systems* and *discrete event systems* as well in the same model.
6) The operations maintain the variable values in the same domain [-1,1].

Features of the development approach

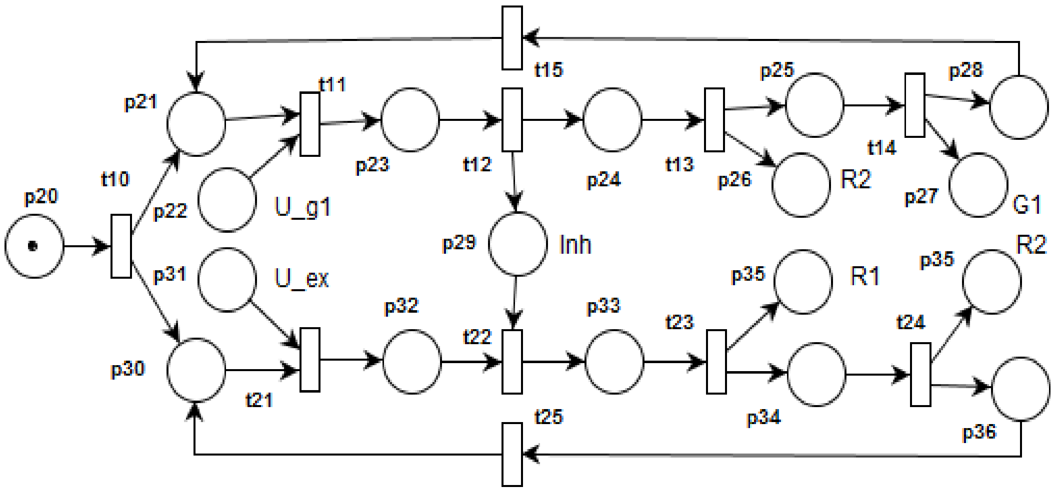
The current approach extends the classical Petri net models to become capable for:

- Handling continuous and fuzzy logic variables,
- Performing simple arithmetic operations,
- Control (split, join, select or block) the execution of sequences of events,
- Making decisions based on tokens set in the transition input set or the lack of them,
- Describing asynchronous and synchronous concurrent executions and
- Synchronizations of the dynamic behavior with external and internal events, or fixed and variable delays.



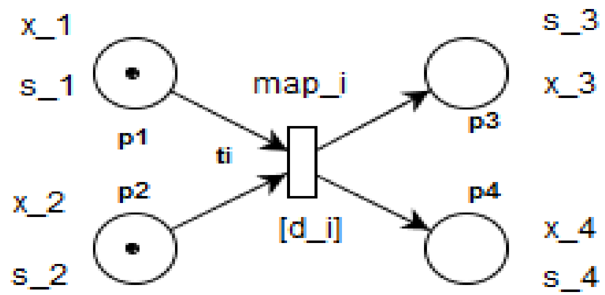
Advantages of the UETPN models

- Single kind of place and transitions,
- Simple implementation
- Can be included into components
- Do not need conditional expression as thresholds
- Include features available in different kinds of Petri Nets (PNs)
  - Inhibitor arcs
  - Reset arcs
  - Variable delays
- Easy to be synthesized due to their simple structure



UETPN Models

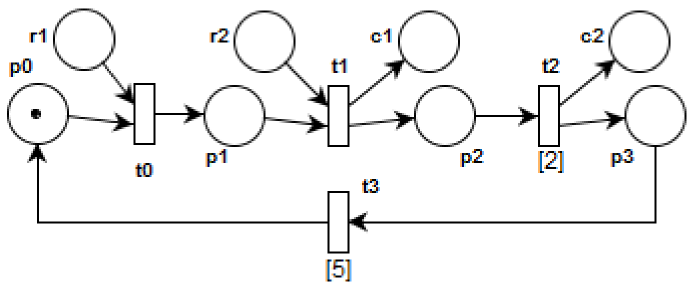
The current approach endows the previous FLETPN models with the capability to describe arithmetic operations with real numbers besides the logical operations and maintaining the previous features.



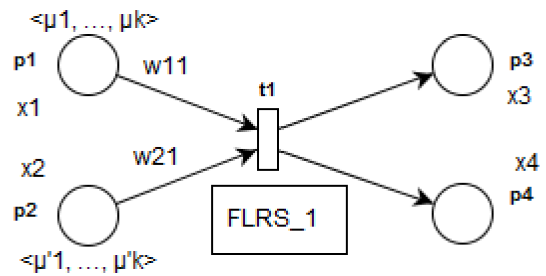
To achieve the UETPN models, the FLETPN models have been modified as follows:

- Each place p_k has assigned a real number variable x_k.
- Each variable x_k has a specified scale factor x_k. The domain of the variable is bounded to the continuous interval $[-s_k,s_k]$.
- A variable x_k can be involved in an arithmetic operation or a fuzzy logic operation.
- Each transition t_i has assigned a mapping ^{o} defining the operations between the input variables of the input place set $^o t_i$ and the output variables of the output place set t^o_i:

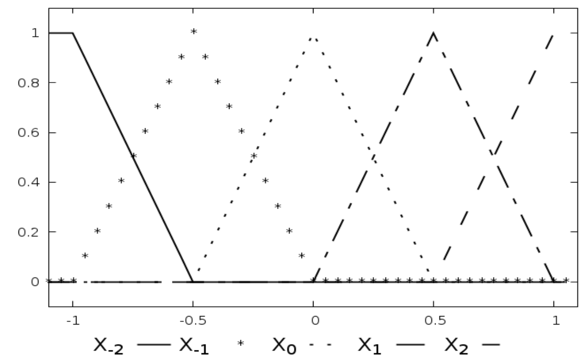
Related Petri Nets



- Standard Petri Nets
- Time Petri Nets
- Hybrid Petri Nets
- Fuzzy Logic Petri Nets
- ***Fuzzy Logic Enhanced Time Petri Nets***
- **Unified Enhanced Time Petri Nets**



FS = {X_{-2}, X_{-1}, X_0, X_1, X_2}
EFS = {X_{-2}, X_{-1}, X_0, X_1, X_2, Φ}
μ= <μ_{-2}, μ_{-1}, μ_0, μ_1, μ_2>



IF (x_1 is X_{-2}) & (x_2 is X_0) THEN (x_3 is X_2) AND (x_4 is Φ)

x_k is Φ ← it is not known the value of x_k at the current moment of time

x_1/x_2	X_{-2}	X_{-1}	X_0	X_1	X_2
X_{-2}	X_{-2},ϕ	X_2,ϕ	X_{-2},X_2	X_0,X_2	X_2,X_2
X_{-1}	X_{-1},ϕ	X_2,ϕ	X_2,X_2	X_1,X_2	X_{-2},X_2
X_0	X_1,X_2	X_1,X_2	X_0,X_2	X_{-1},X_2	X_1,X_2
X_1	X_0,X_2	X_1,X_2	X_{-2},X_2	ϕ,X_{-1}	ϕ,X_1
X_2	X_{-2},X_2	X_0,X_2	X_0,X_2	ϕ,X_1	ϕ,X_1

Definition of UETPN

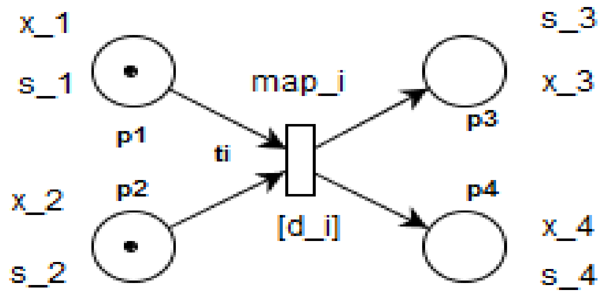
$$UETPN = (P, T, pre, post, D, X, S, X^{o}, EFS, Map, \mathbf{FLRS}, Inp, Out, \alpha, \beta, \delta, \mathbf{M})$$

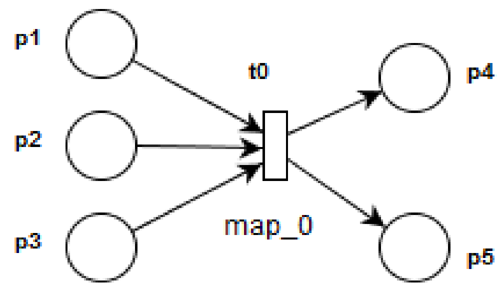
$D = \{d_0, d_1, ..., d_m\}$, time delays;
$X = \{x_0, x_1, ..., x_m\}$; x_k real number variable
$Map = \{map_0, map_1, ..., map_n\}$
EFS: extended fuzzy set = $\{X_{-2}, X_{-1}, X_0, X_1, X_2, \Phi\}$
FLRS: fuzzy logic rule set

$$map_i : ([-s_1, s_1] \cup \phi) \times ([-s_2, s_2] \cup \phi) \rightarrow ([-s_3, s_3] \cup \phi) \times ([-s_4, s_4] \cup \phi).$$

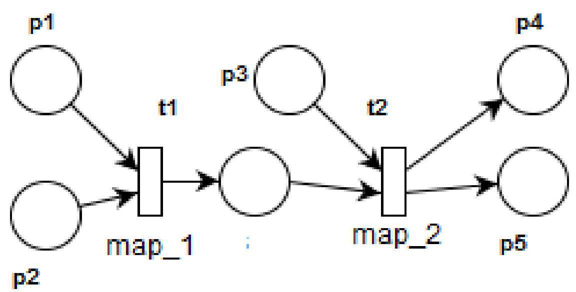
op = $\{\wedge, +, -, *, /\}$

x_1/x_2	X_{-2}	X_{-1}	X_0	X_1	X_2
X_{-2}	X_2, X_2	X_2, X_2	X_2, X_2	X_2, X_1	X_2, X_0
X_{-1}	X_2, X_2	X_2, X_2	X_2, X_1	X_2, X_0	X_2, X_{-1}
X_0	X_2, X_2	X_2, X_1	X_2, X_0	X_2, X_{-1}	X_2, X_{-2}
X_1	X_2, X_1	X_2, X_0	X_2, X_{-1}	X_2, X_{-2}	X_2, X_{-2}
X_2	X_2, X_0	X_2, X_{-1}	X_2, X_{-2}	X_2, X_{-2}	X_2, X_{-2}
ϕ	ϕ, ϕ	ϕ, ϕ	X_2, ϕ	ϕ, X_0	ϕ, ϕ

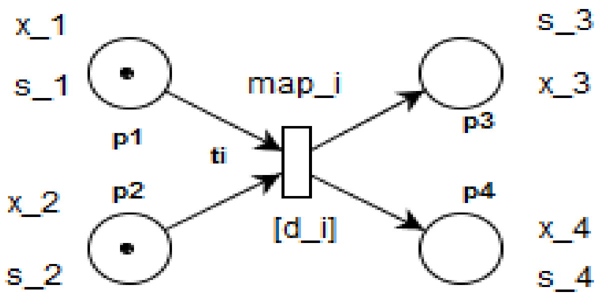


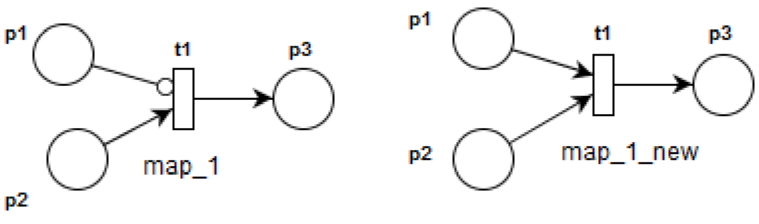


a) Required UETPN

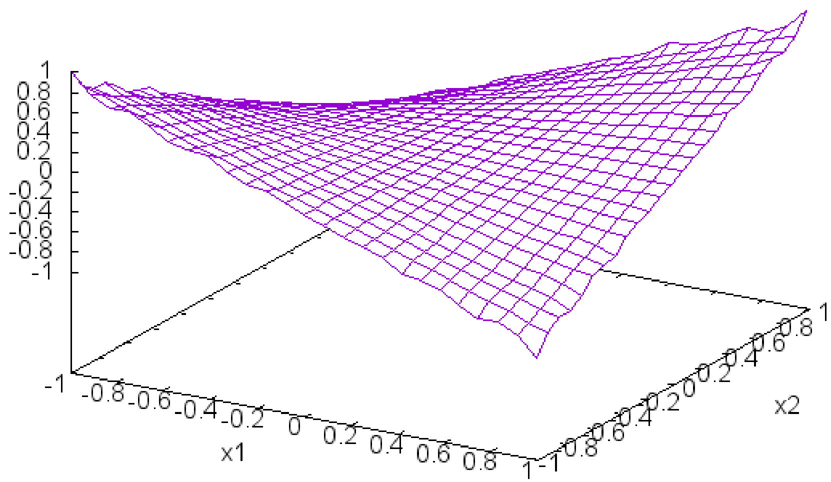


b) Developed UETPN





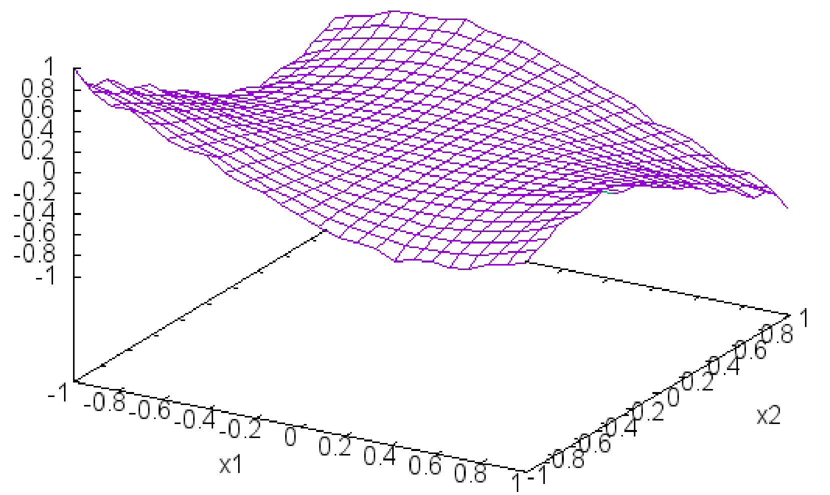
a) PN with inhibitor arcs
b) Equivalent UETPN



$$x_3 = map_i(x_1, x_2) = (x_1 \circ x_2) \star FL_{MT}(x_1, x_2)$$

where $\circ$ is in *op* = {∧ ,+, -, *, /}

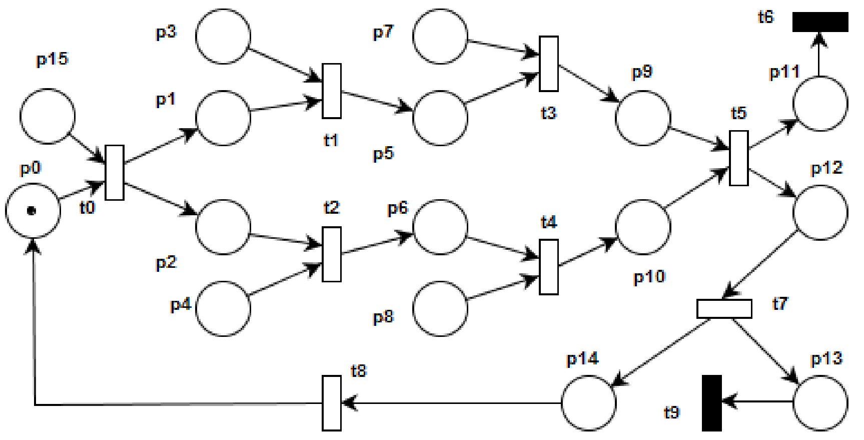
x_1/x_2	X_{-2}	X_{-1}	X_0	X_1	X_2
X_{-2}	X_2, X_2	X_2, X_2	X_2, X_2	X_2, X_1	X_2, X_0
X_{-1}	X_2, X_2	X_2, X_2	X_2, X_1	X_2, X_0	X_2, X_{-1}
X_0	X_2, X_2	X_2, X_1	X_2, X_0	X_2, X_{-1}	X_2, X_{-2}
X_1	X_2, X_1	X_2, X_0	X_2, X_{-1}	X_2, X_{-2}	X_2, X_{-2}
X_2	X_2, X_0	X_2, X_{-1}	X_2, X_{-2}	X_2, X_{-2}	X_2, X_{-2}
ϕ	ϕ, ϕ	ϕ, ϕ	X_2, ϕ	ϕ, X_0	ϕ, ϕ



Admissibility and simulation

Inference rules for t_5 (otimized by GA)

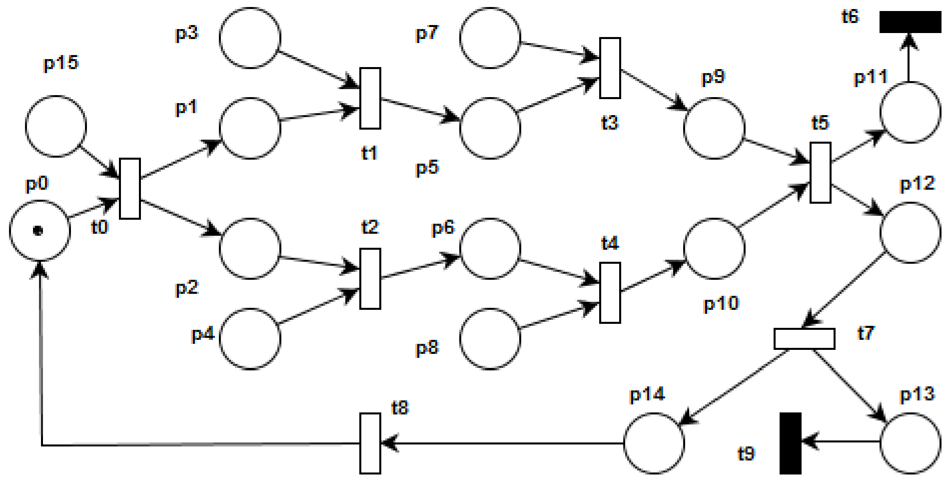
x_1/x_3	X_{-2}	X_{-1}	X_0	X_1	X_2
X_{-2}	X_1, X_0	X_{-2}, ϕ	X_{-1}, ϕ	ϕ, X_1	X_2, X_{-1}
X_{-1}	X_2, X_{-1}	X_0, ϕ	X_2, X_1	X_1, X_0	X_2, X_{-1}
X_0	X_1, X_{-2}	X_1, X_0	X_2, ϕ	X_0, X_2	ϕ, X_2
X_1	X_{-2}, X_2	X_2, X_{-2}	X_2, X_1	X_1, X_1	X_2, X_2
X_2	X_0, X_{-2}	ϕ, X_{-2}	X_2, X_0	ϕ, ϕ	X_2, X_2



A transition is enabled if and only if there is at least one rule in the mapping table that can be activated. If more than one rule can be activated, it is applied the ***fuzzy inference procedure.***

Analysis of UETPN models

- Verification of the basic PN structure
- Verification of the temporal behavior
- Verification of functionality



Reachability graph of the classical PN

Enhanced Time Petri Net based Language (ETPNL)

$$\sigma_2 = [t_0(x_{15},\phi) * [(t_1(x_3,\phi) * t_3(x_7,\phi))\&(t_2(x_4,\phi) * t_4(x_8,\phi))] * t_5(\phi, x_{11}) * t_7(\phi, x_{13})]\#t_8(\phi,\phi)$$

Inference rules for t_5 (otimized by GA)

x_1/x_3	X_{-2}	X_{-1}	X_0	X_1	X_2
X_{-2}	X_1, X_0	X_{-2}, ϕ	X_{-1}, ϕ	ϕ, X_1	X_2, X_{-1}
X_{-1}	X_2, X_{-1}	X_0, ϕ	X_2, X_1	X_1, X_0	X_2, X_{-1}
X_0	X_1, X_{-2}	X_1, X_0	X_2, ϕ	X_0, X_2	ϕ, X_2
X_1	X_{-2}, X_2	X_2, X_{-2}	X_2, X_1	X_1, X_1	X_2, X_2
X_2	X_0, X_{-2}	ϕ, X_{-2}	X_2, X_0	ϕ, ϕ	X_2, X_2

The mapping tables constraint the UETPN model behavior. They can implement thresholds.

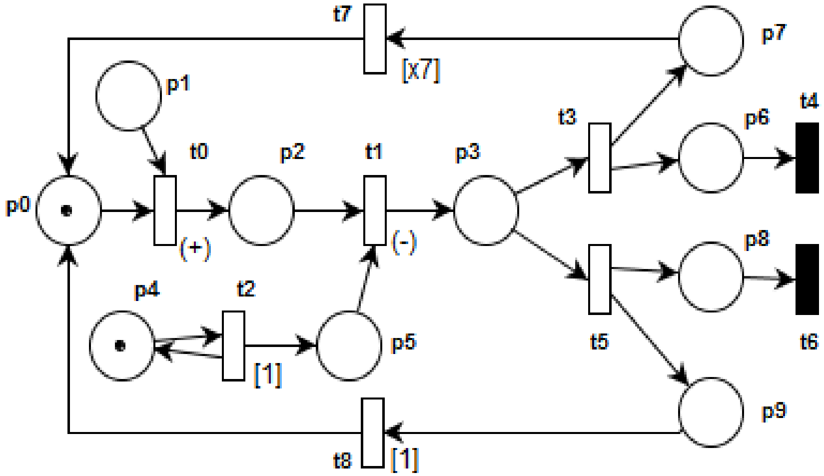
Homework: Integrate the UETPN models in OETPN-C.

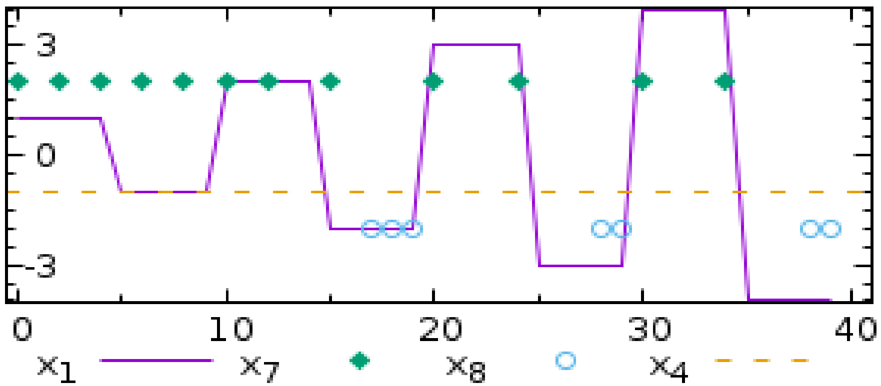
Questions:

- **Place types?**
- **Guards?**
- **Mappings?**
- **Input ports?**
- **Output ports?**

Example of utilization 1

Conflict with variable delay





$M^0 = [0, \varphi, \varphi, \varphi, -1, \varphi, \varphi, \varphi, \varphi]$

Input signal in p_1: rectangular signal amplified at each 10 t.u.

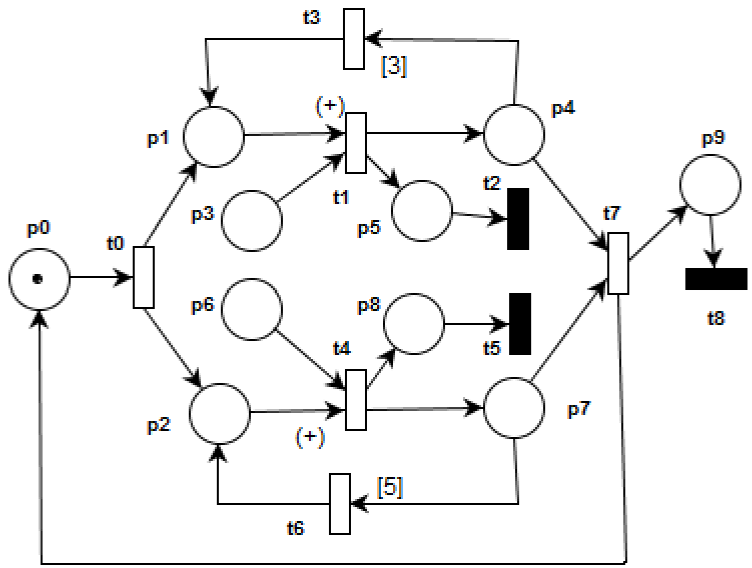
- $t_0 : x_2 = x_0 + x_1$
- $t_1 : x_3 = x_2 - x_5$
- t_3 is executed if $x_3 > 0$
- t_4 is executed if $x_3 < 0$

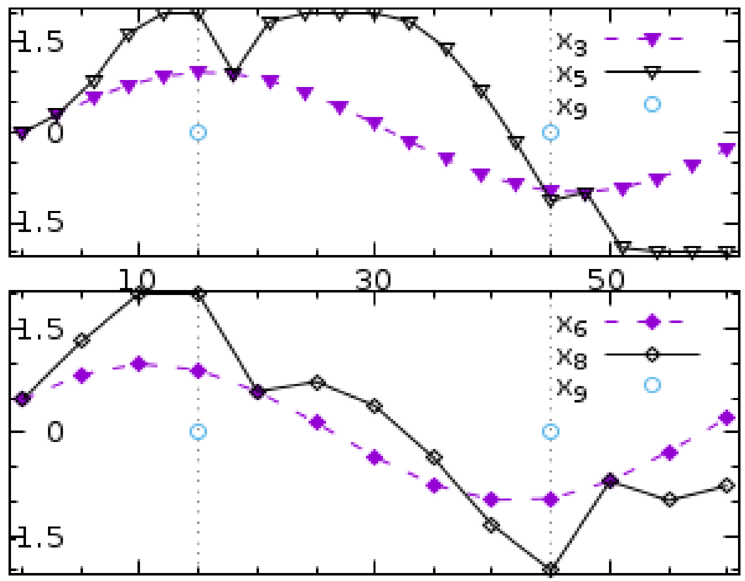
Joined MTs of the transitions t_3 and t_5

	X_{-2}	X_{-1}	X_0	X_1	X_2
t_3	ϕ, ϕ	ϕ, ϕ	ϕ, ϕ	X_1, X_1	X_2, X_2
t_5	X_{-2}, X_{-2}	X_{-1}, X_{-1}	ϕ, ϕ	ϕ, ϕ	ϕ, ϕ

Example of utilization 2

Two loops with different periods





$M^0 = [0, \varphi, \varphi, \ldots, \varphi]$

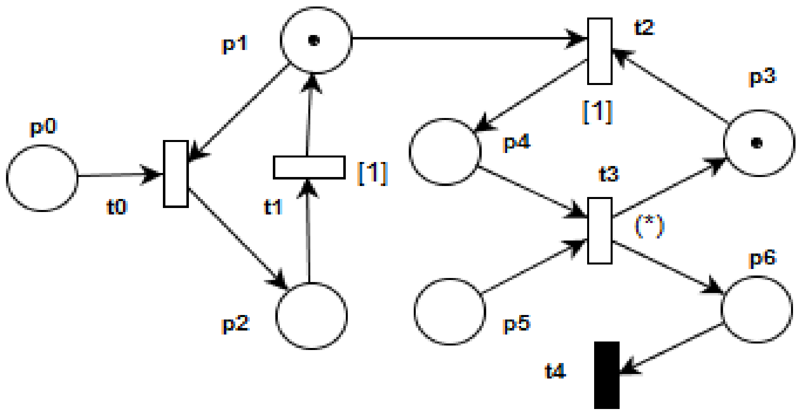
t_7 is executed if:

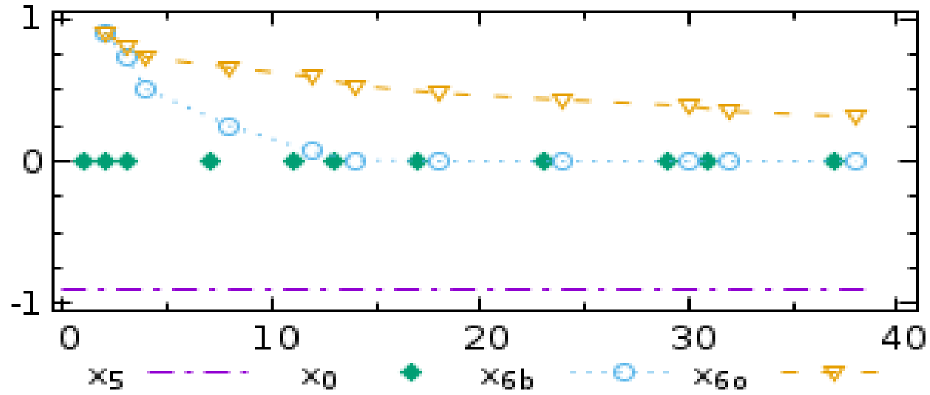
$$(x_4 > 1 \quad and \quad x_5 > 1) \quad or \quad (x_4 < -1 \quad and \quad x_5 < -1).$$

Mapping table of transition t_7

x_4/x_7	X_{-2}	X_{-1}	X_0	X_1	X_2
X_{-2}	X_0, X_0	ϕ, ϕ	ϕ, ϕ	ϕ, ϕ	ϕ, ϕ
X_{-1}	ϕ, ϕ	ϕ, ϕ	ϕ, ϕ	ϕ, ϕ	ϕ, ϕ
X_0	ϕ, ϕ	ϕ, ϕ	ϕ, ϕ	ϕ, ϕ	ϕ, ϕ
X_1	ϕ, ϕ	ϕ, ϕ	ϕ, ϕ	ϕ, ϕ	ϕ, ϕ
X_2	ϕ, ϕ	ϕ, ϕ	ϕ, ϕ	ϕ, ϕ	X_0, X_0

Example of utilization 3
Inhibitor arc and bending of multiplication





Example 3. The mapping table for t_2

x_1/x_3	X_{-2}	X_{-1}	X_0	X_1	X_2	ϕ
X_{-2}	ϕ	ϕ	ϕ	ϕ	ϕ	ϕ
X_{-1}	ϕ	ϕ	ϕ	ϕ	ϕ	ϕ
X_0	ϕ	ϕ	ϕ	ϕ	ϕ	ϕ
X_1	ϕ	ϕ	ϕ	ϕ	ϕ	ϕ
X_2	ϕ	ϕ	ϕ	ϕ	ϕ	ϕ
ϕ	X_{-2}	X_{-1}	X_0	X_1	X_2	ϕ

TABLE 4. Inference rules for third example T3

x_1/x_3	X_{-2}	X_{-1}	X_0	X_1	X_2
X_{-2}	X_2, X_2	X_2, X_2	X_2, X_2	X_1, X_1	X_0, X_0
X_{-1}	X_2, X_2	X_2, X_2	X_1, X_1	$X_0 X_{,0}$	X_{-1}, X_{-1}
X_0	X_2, X_2	X_1, X_1	X_0, X_0	X_{-1}, X_{-1}	X_{-2}, X_{-2}
X_1	X_1, X_1	X_0, X_0	$X{-1}, X_{-1}$	X_{-2}, X_{-2}	X_{-2}, X_{-2}
X_2	X_0, X_0	X_{-1}, X_{-1}	X_{-2}, X_{-2}	X_{-2}, X_{-2}	X_{-2}, X_{-2}

*

****

END

****

*