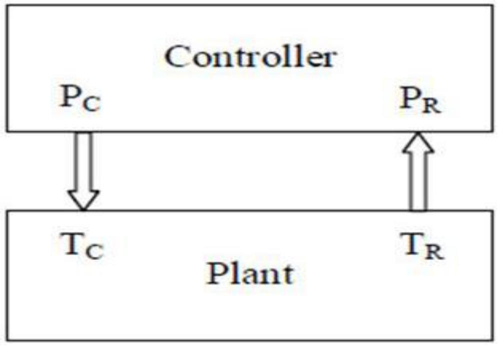
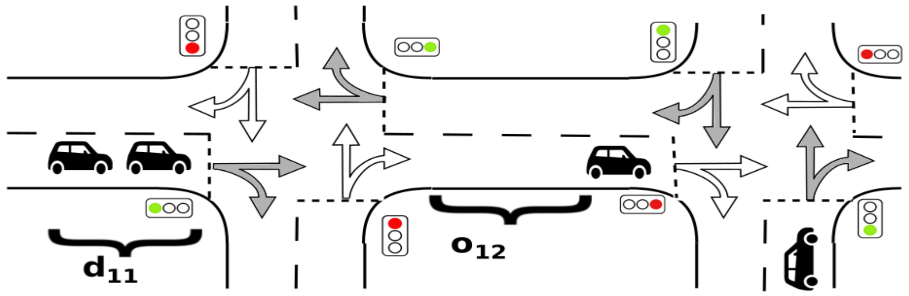
Integration of Fuzzy Logic in Object Enhanced Time Petri Nets

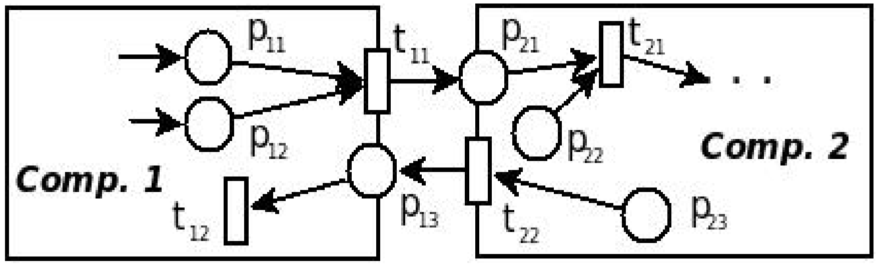
⇨ Unified Enhanced Time Petri Nets (UETPNs)

- 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**

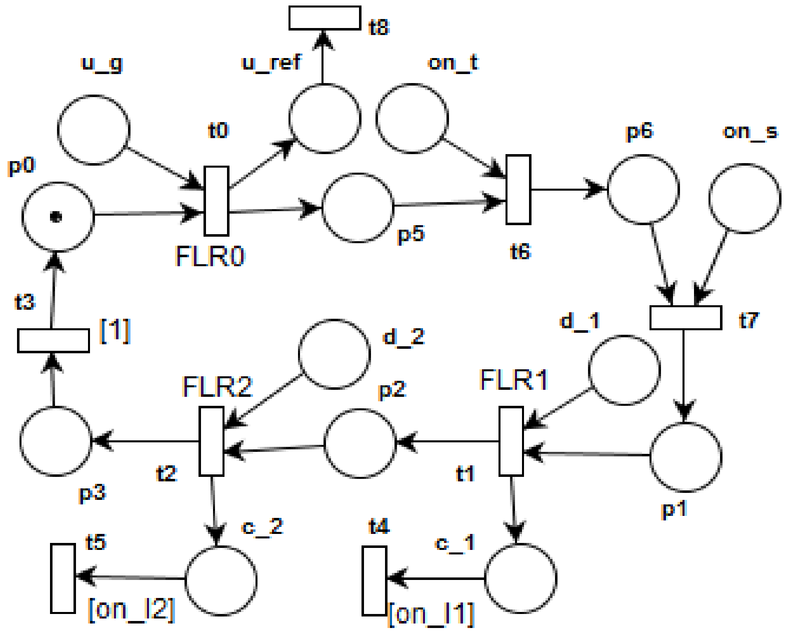
Justifications of integration:

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) OETPN models combine the FLETPN features with arithmetic operations.
5) OETPNs 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

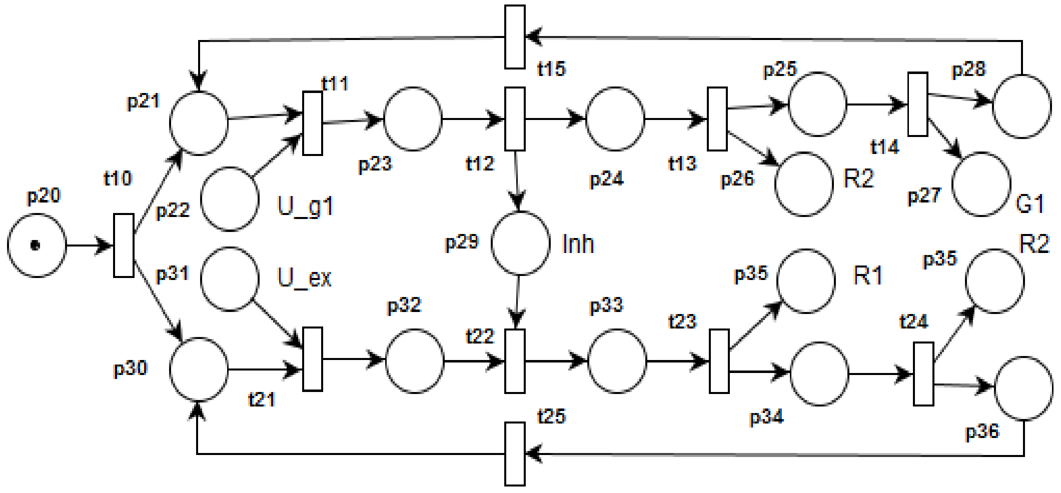
The current approach extends the classical Petri net models to become capable for:

- 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.



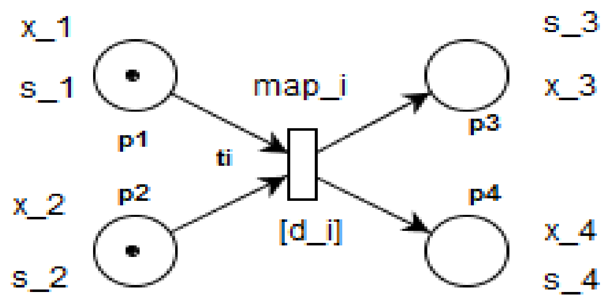
Advantages of the OETPN models with Fuzzy Logic

- 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



OETPN 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.

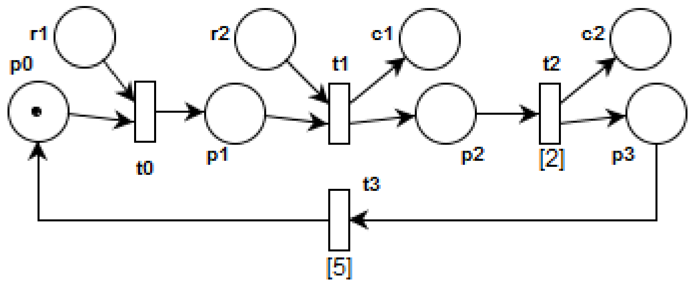


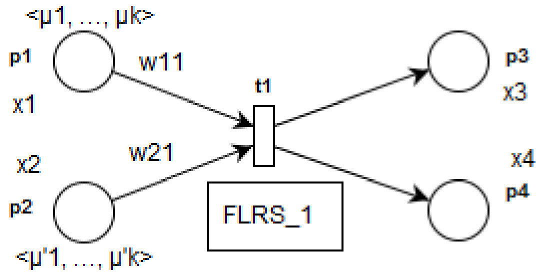
To achieve the OETPN models, the FLETPN models have been modified as follows:

- 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:

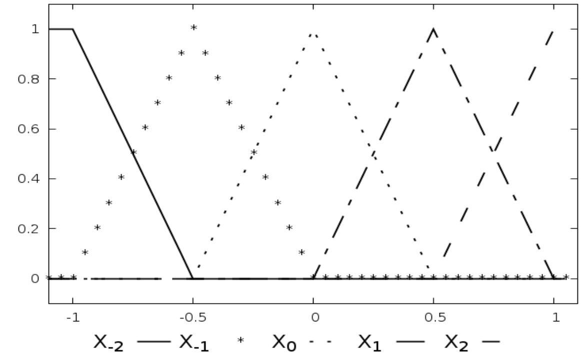
Related Petri Nets

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



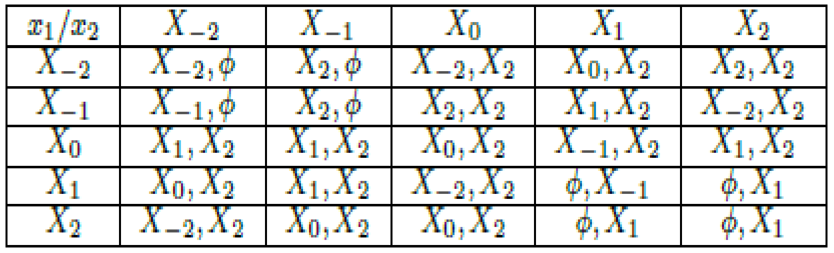


FS = {X_{-2}, X_{-1}, X_0, X_1, X_2}
EFS = {X_{-2}, X_{-1}, X_0, X_1, X_2, Φ}
μ= <μ_{-2}, μ_{-1}, μ_0, μ_1, μ_2>



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

OETPN integrating FL ➔ UETPN

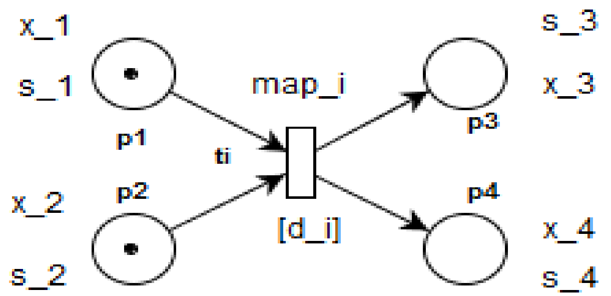
$$UETPN = (P, T, pre, post, D, X, S, X^o, EFS, Map, \mathbf{FLRS}, Inp, Out, \alpha, \beta, \delta, \mathbf{M})$$

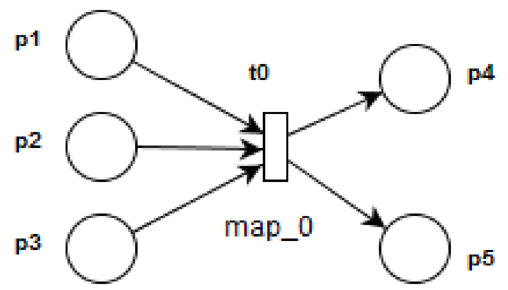
D = $\{d_0, d_1, \ldots, d_m\}$, time delays;
X = $\{x_0, x_1, \ldots, x_m\}$; x_k real number variable
Map = $\{map_0, map_1, \ldots, map_n\}$
EFS: extended fuzzy set = $\{X_{-2}, X_{-1}, X_0, X_1, X_2, \Phi\}$
FLRS: fuzzy logic rule set

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

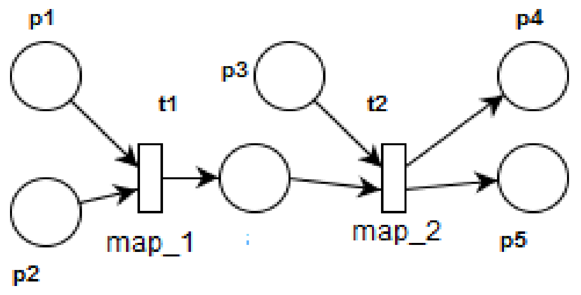
op = $\{\wedge, +, -, *, /\}$

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

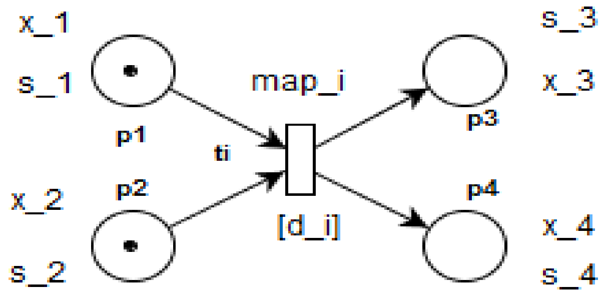


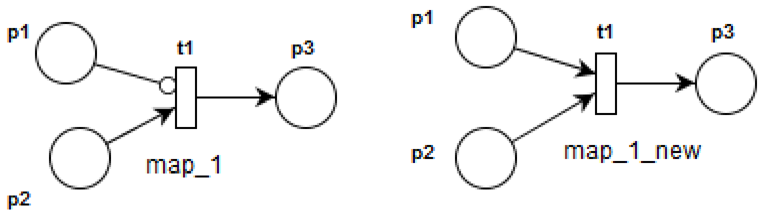


a) Required UETPN

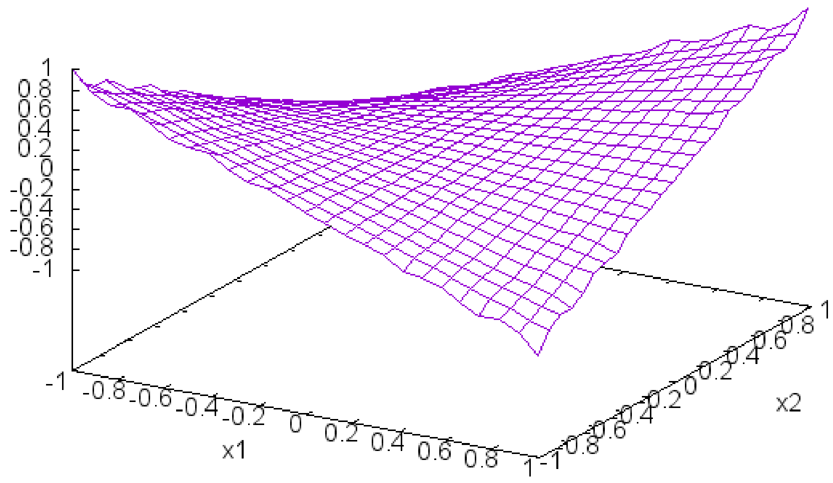


b) Developed UETPN





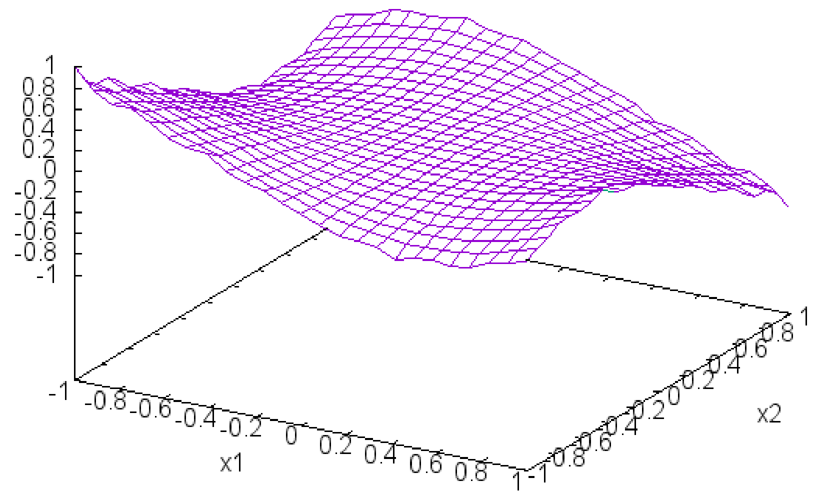
a) PN with inhibitor arcs b) Equivalent UETPN



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

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



Homework: Integrate the UETPN models in OETPN.

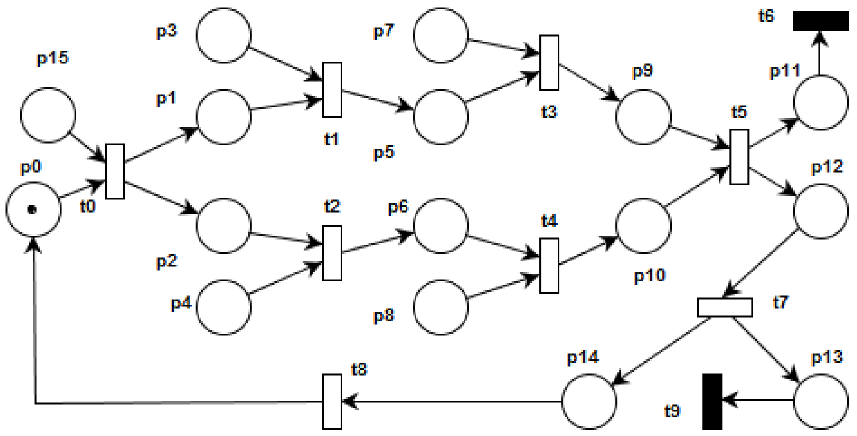
Questions:

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

Admissibility and simulation

Inference rules for t_5 (otimized by GA)

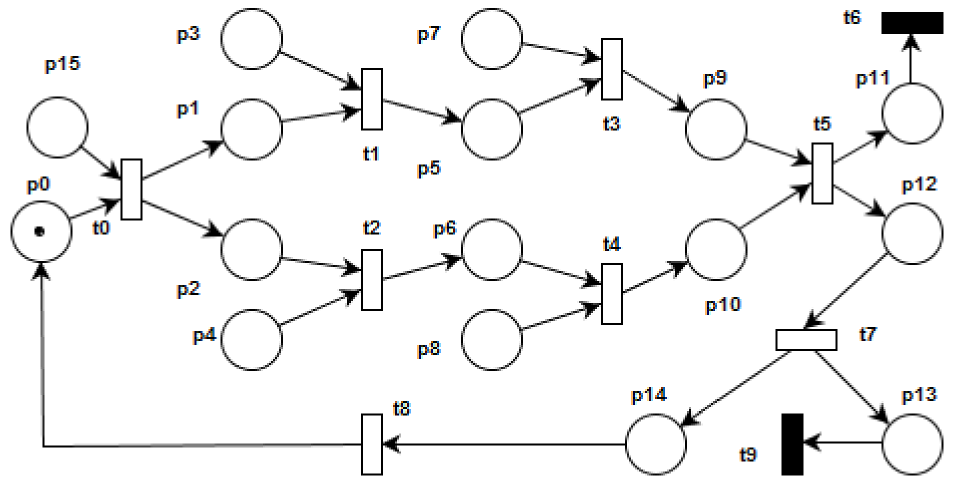
x_1/x_3	X_{-2}	X_{-1}	X_0	X_1	X_2
X_{-2}	X_1, X_0	X_{-2}, ϕ	X_{-1}, ϕ	ϕ, X_1	X_2, X_{-1}
X_{-1}	X_2, X_{-1}	X_0, ϕ	X_2, X_1	X_1, X_0	X_2, X_{-1}
X_0	X_1, X_{-2}	X_1, X_0	X_2, ϕ	X_0, X_2	ϕ, X_2
X_1	X_{-2}, X_2	X_2, X_{-2}	X_2, X_1	X_1, X_1	X_2, X_2
X_2	X_0, X_{-2}	ϕ, X_{-2}	X_2, X_0	ϕ, ϕ	X_2, X_2



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 OETPN 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)

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

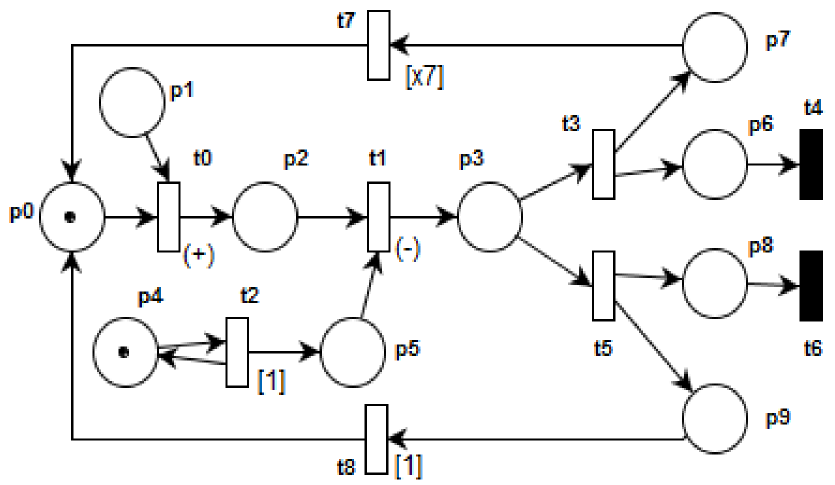
Inference rules for t_5 (otimized by GA)

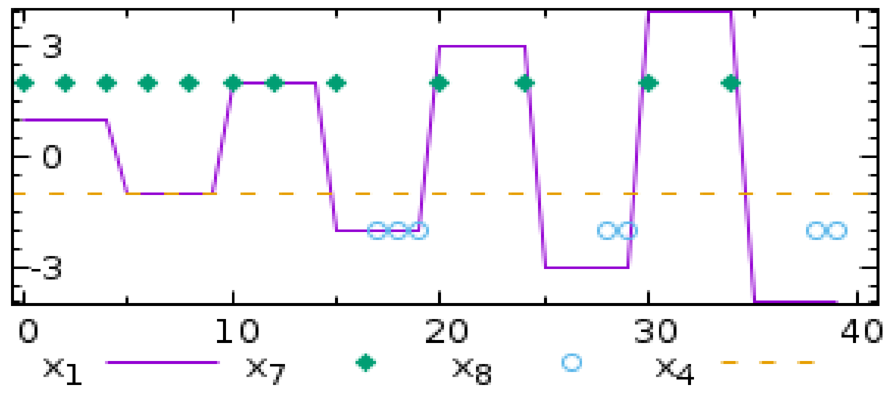
x_1/x_3	X_{-2}	X_{-1}	X_0	X_1	X_2
X_{-2}	X_1, X_0	X_{-2}, ϕ	X_{-1}, ϕ	ϕ, X_1	X_2, X_{-1}
X_{-1}	X_2, X_{-1}	X_0, ϕ	X_2, X_1	X_1, X_0	X_2, X_{-1}
X_0	X_1, X_{-2}	X_1, X_0	X_2, ϕ	X_0, X_2	ϕ, X_2
X_1	X_{-2}, X_2	X_2, X_{-2}	X_2, X_1	X_1, X_1	X_2, X_2
X_2	X_0, X_{-2}	ϕ, X_{-2}	X_2, X_0	ϕ, ϕ	X_2, X_2

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

Example of utilization 1

Conflict with variable delay



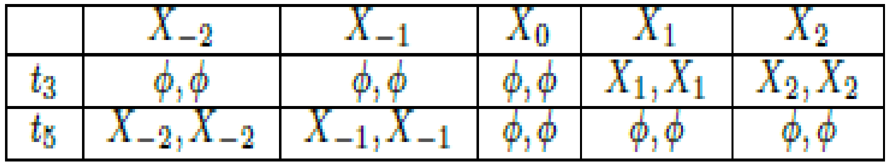


$M^0 = [0, \varphi, \varphi, \varphi, -1, \varphi, \varphi, \varphi, \varphi]$

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 $x_3 < 0$

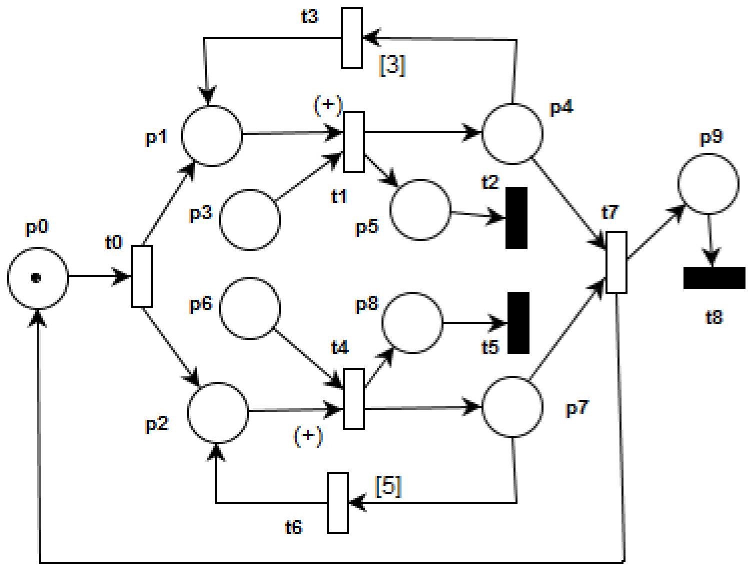
Joined MTs of the transitions t_3 and t_5

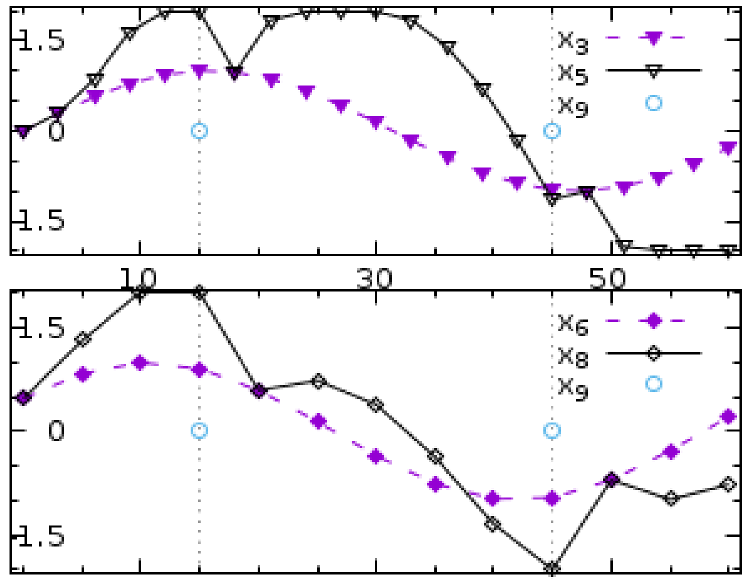


	X_{-2}	X_{-1}	X_0	X_1	X_2
t_3	ϕ, ϕ	ϕ, ϕ	ϕ, ϕ	X_1, X_1	X_2, X_2
t_5	X_{-2}, X_{-2}	X_{-1}, X_{-1}	ϕ, ϕ	ϕ, ϕ	ϕ, ϕ

Example of utilization 2

Two loops with different periods





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

t_7 is executed if:

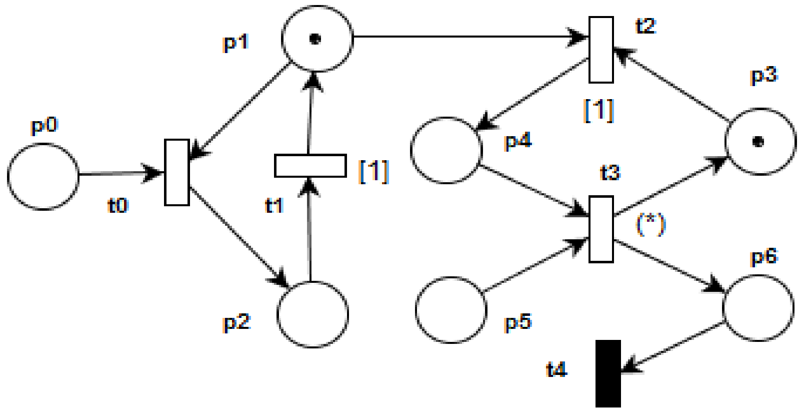
$$(x_4 > 1 \quad and \quad x_5 > 1) \quad or \quad (x_4 < -1 \quad and \quad x_5 < -1).$$

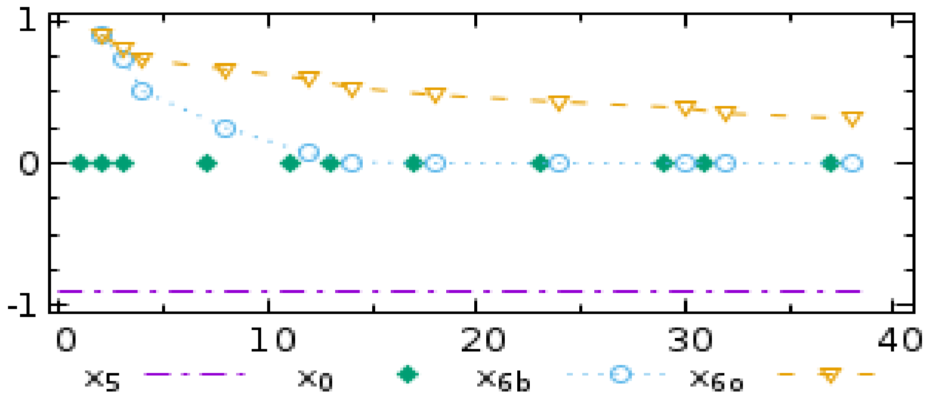
Mapping table of transition t_7

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

$x1/x3$	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}

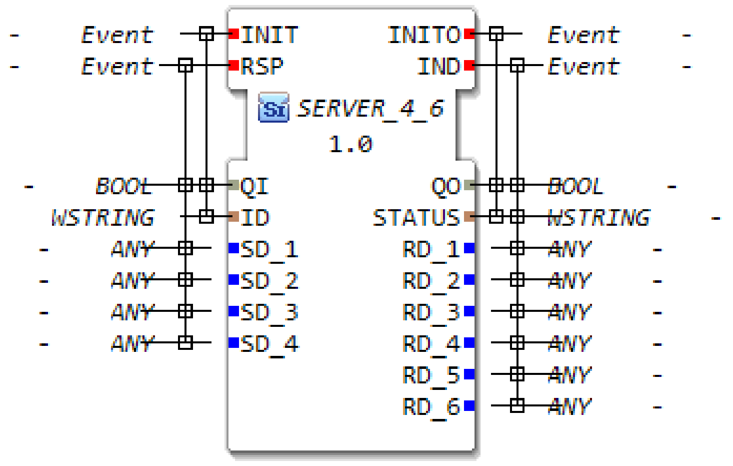
4.10 Object Enhanced Time Petri Net Capsules (OETPN-Cs)

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

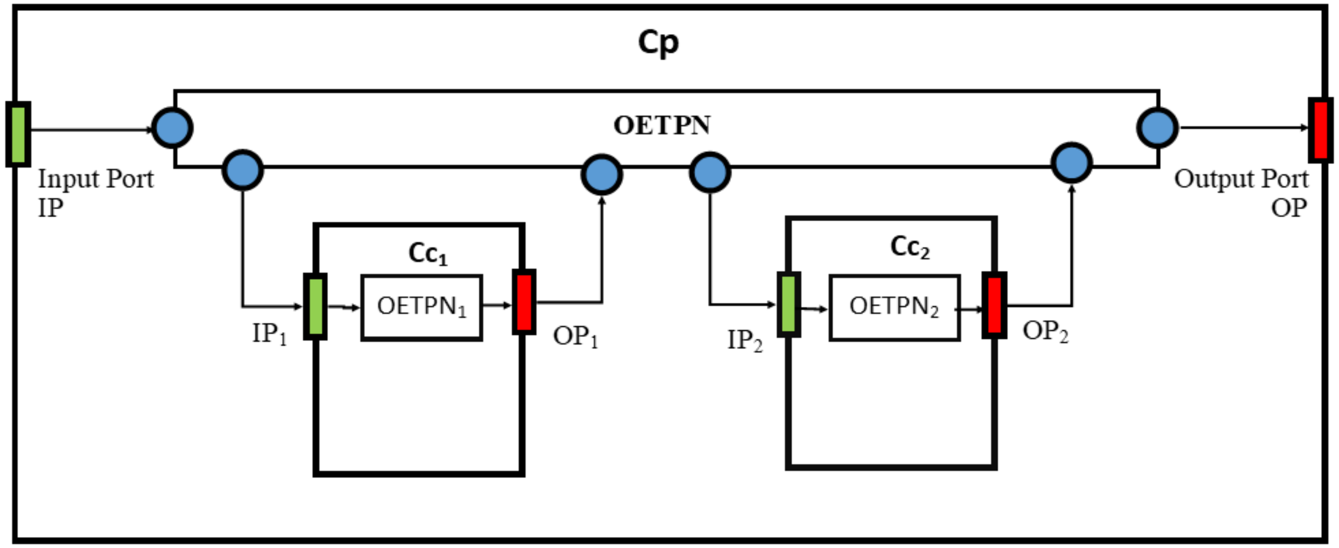
Goal: OETPN-Cs can be used for specification, design, verification and implementation of concentrated or distributed reactive software applications.

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.

IEC 61499 standard Function blocks have some similar capabilities to OETPN-C.



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, …}.

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 transitions 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,

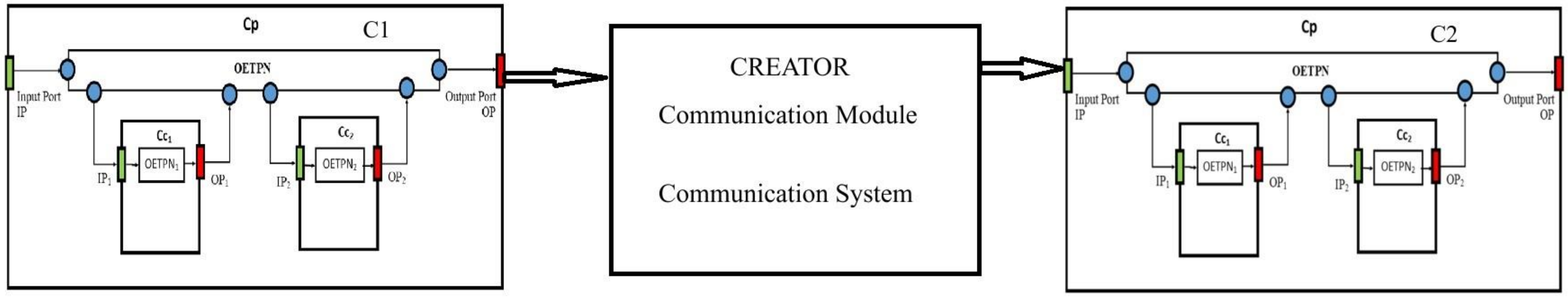
```
OETPN-C executor algorithm:
Input: Pre, Post, $M_0, P, T = T_a \cup T_s, D$, Grd&Map, Out,
$Inp = Inp_a \cup Inp_s$;
Initialization: $M = M^0$, execList = empty;
repeat
  wait(event);
  if event is $tick$ then
    * decrease the Delays of the transitions in execList;
    repeat
      for all $t_i \in T$ 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.

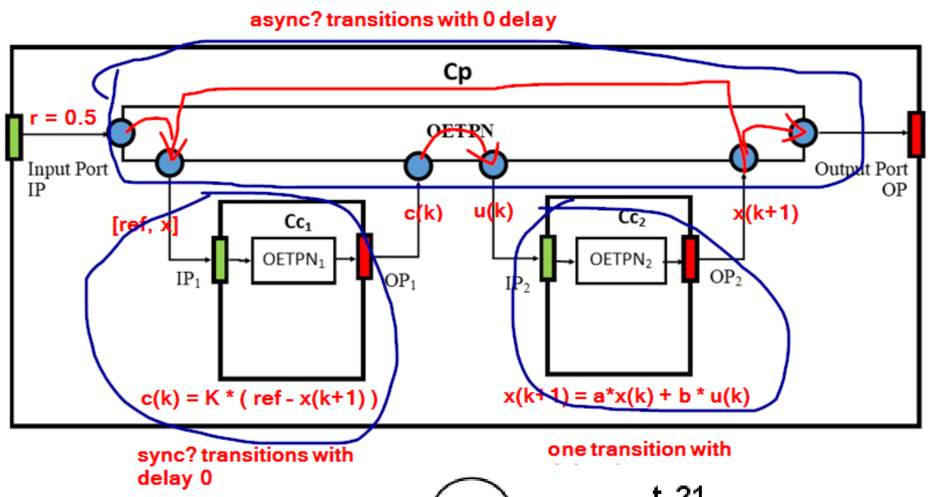
The methods *send(Out)* and *receive(Inp)* are used for this purpose.



Exemples

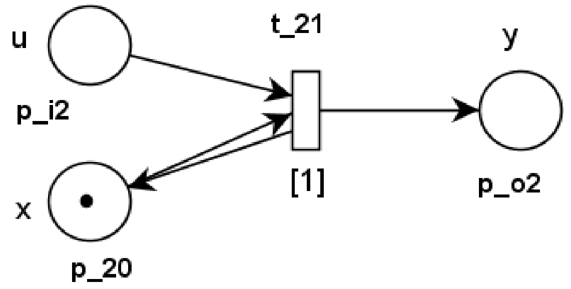
Implement on a dual cores system:

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



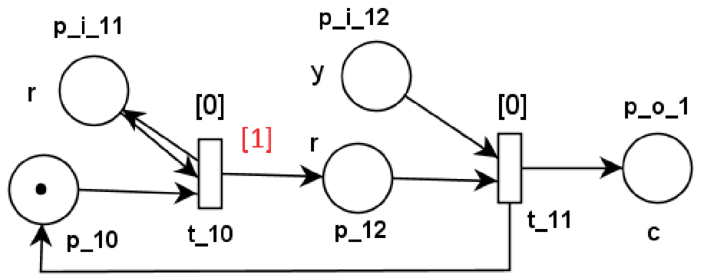
Cc2: plant model:

x(k+1)=a·x(k) + b·u(k); y=x(k+1); x(0)=0.1;

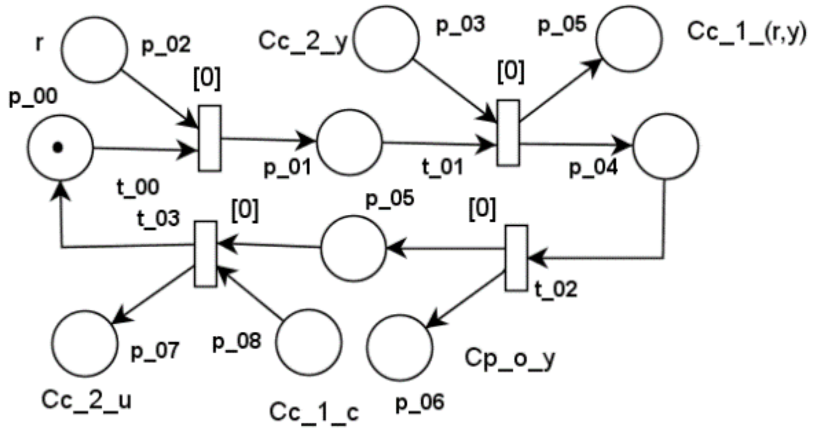


Cc1: Controller) model

c(k)= K·(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*.

HomeWork

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.

Question: *What are the differences between components integrating OETPN models and OETPN-Capsules?*

Try to design the precedent problem with components integrating the same OETPN models!

4.11 Digital Twins (DTws)

DTw definitions

DTws are consistent constructions and representations of physical objects and systems for their dynamic simulations, monitoring, analysis and control.

They take input data from real systems and produce prediction information of the real system evolutions.

According to ***Industrial Internet Consortium***:
“a digital twin is a formal digital representation of some asset, process or system that captures attributes and behaviors of that entity suitable for communication, storage, interpretation or processing within a certain context.”

DTw = virtual twin = digital avatar

Definition: A DTw is a real-time dynamic ***surreal mapping*** of a physical entity system

A DTw displays four typical technical features:

- virtual-real mapping,
- real-time synchronization,
- symbiotic evolution, and
- closed-loop optimization.

The differences between DTws and model simulations consist of:

- The simulations are performed at their specified pace, but may not have a one-to-one analog correspondence with real physical objects. Simulation predicts the future state of a physical system.
- DTws are linked to sensors of real equipment, and they replicate the real world or systems consistently changing their states with the represented physical entities.
- While the basic uses of simulations are offline, DTws are used online.
- DTWs determine the current states and the past states of the physical system.
- DTws can predict the future evolution of the physical system.

Digital model: once they are created, there is no information interchange between digital models and physical object (or model).
Digital shadow: there is data flowing from the physical object to the digital model.
Digital twin: there is data flows between the physical object and the digital object,

DTw Utilization and Applications

- Optimal state control
- Output control
- Model predictive control
- Production quality control
- Real-time monitoring, process evaluation
- Optimization of process parameters
- Optimal-tuning control
- Routing control
- Fault prediction
- Diagnosing
- Maintenance of equipment
- Management decisions
- Prescription of actions

Applications can be found in:

- Manufacturing
- Energy
- Transportation
- Health-care
- Agriculture
- Aircraft operation
- Unmanned vehicle control etc.

DTw Methods of utilization:

- The digital plant is corrected using the real plant, and the real controller is improved using the digital controller
- Adaptive model tracking
- Performance prediction for plant state measurable and unmeasurable
- Performance retention
- Fault tolerant control

“Simultaneously, the DTw can predict and analyze the behavior of simulated objects, diagnose faults and issue warnings, locate and record problems, and achieve optimal control.”

DTws can be used in the distribution planning, activity scheduling, resource allocation and monitoring, robust process control and maintenance of resources.

DTw Characteristics and Features

DTWs can be composed by hierarchical, association and heterarchical (peer-to-peer) organization.

DTWs enable computational and analytic models.

Online and offline information are used for achieving the specified goals.

To ensure interoperability of digital twins and their information sources synchronized communication means are involved.

Dtws are dynamic and need the interaction with the physical world. For DTw control systems, there are changes of information between the physical subsystem and the virtual subsystem.

DTws fulfil the features:

- Connectivity enabling interactions of DTws.
- Deployment from edge to clouds. Some polymorphic DTws can be required for their interaction.
- Interoperability to exchange information and to mutually its use.
- Security requirements for entire system protection.

DTw Architectures

OETPN-C: a novel design framework for the development of Digital Twin (DTw) models for process- and motion-control applications.
DTWs can be developed from different points of view:

- subsystems and components,
- process replication,
- equipment modelling,
- shop floor simulation,
- manufacturing facility, and
- multi-factory simulation.

DTws use computational and analytical models such as:

- Fuzzy logic
- Neural networks
- Wavelet analysis
- Support vector machines etc.

DTws can be developed based on:

- physics-based,
- data-driven,
- machine learning,
- parametric models, etc.

DTw capabilities:

- Anomaly detection ← safety systems & security systems
- Waiting time prediction ← elevator performance
- Parameter prediction ← manufacturing

DTw capabilities ←→ model-based simulation methods

DTw Capabilities vs. Competence

Capabilities: can link input and output channels
Competence: DTw reacts to its inputs with performances exceeding specified thresholds.

Challenges:

- *hallucinations* in foundation models (i.e., content generated by the foundation model that is not based on factual information)
- how to assess the *fidelity* of digital twins

DTw Problems:

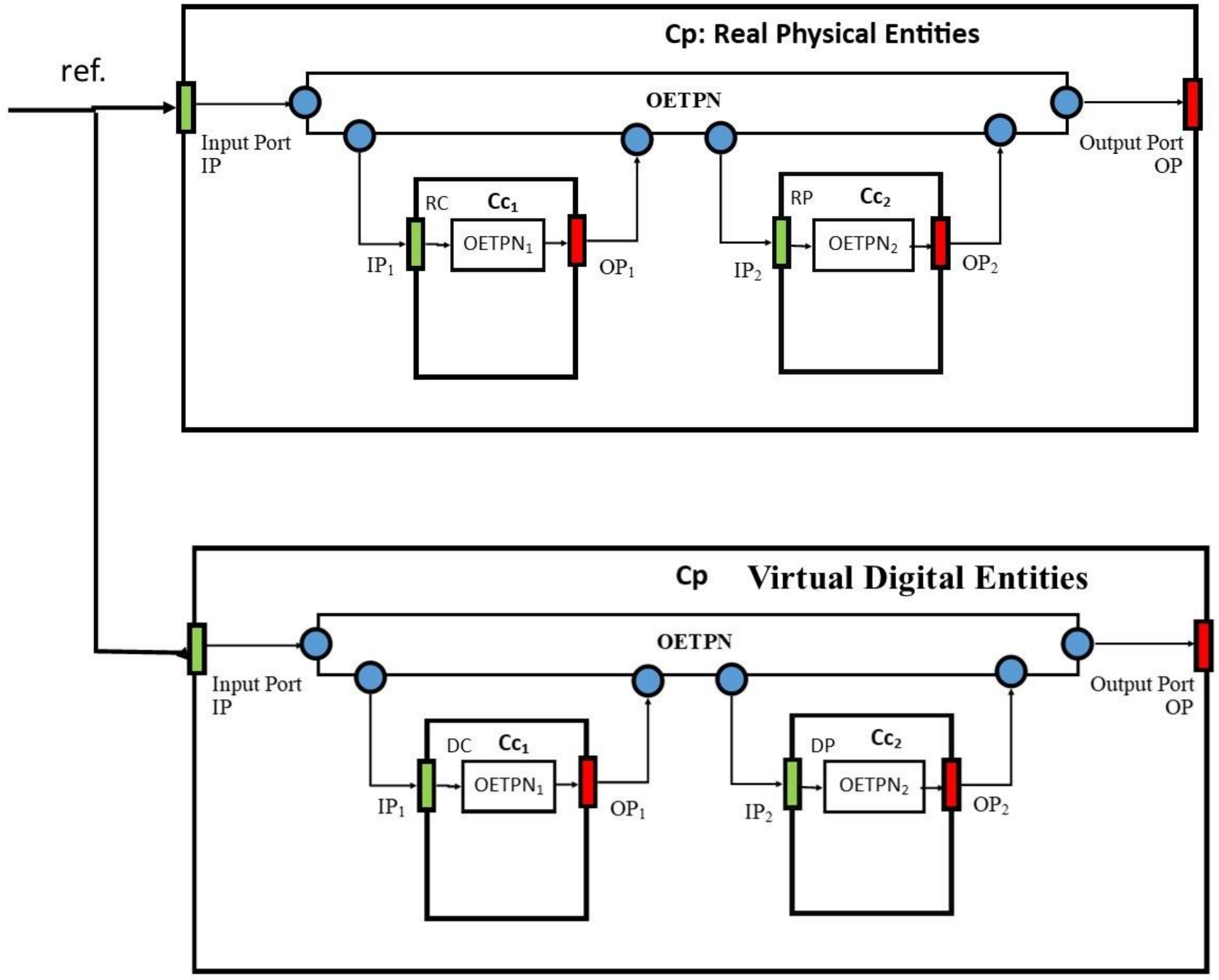
- The real-time interaction between physical entities and DTw.
- R-T interactions between distributed DTws.

The implementations are based on:

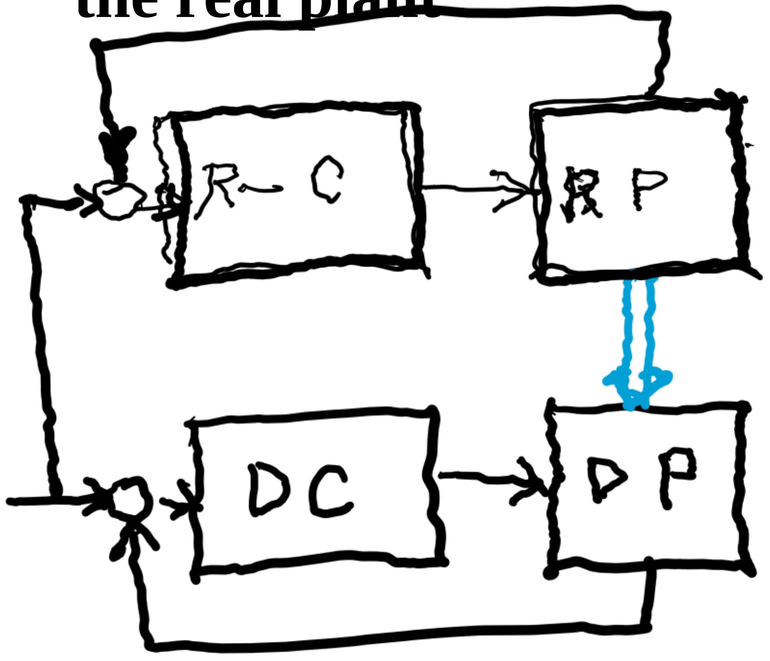
- Digital Twin Prototype (DTP) and
- Digital Twin Instance (DTI).

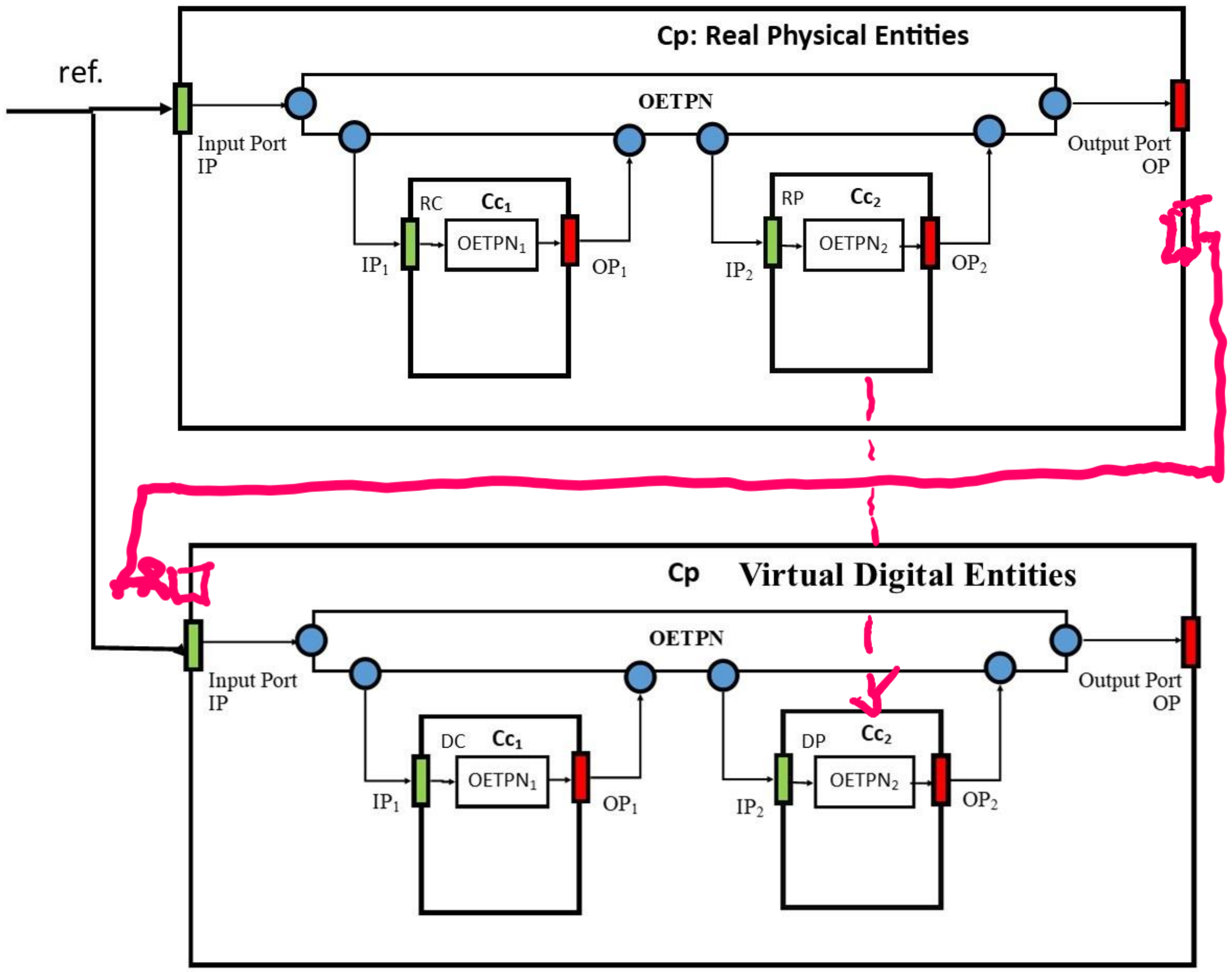
DTw’s are operated on or in a ***Digital Twin Environment (DTE***).

Digital Shadow



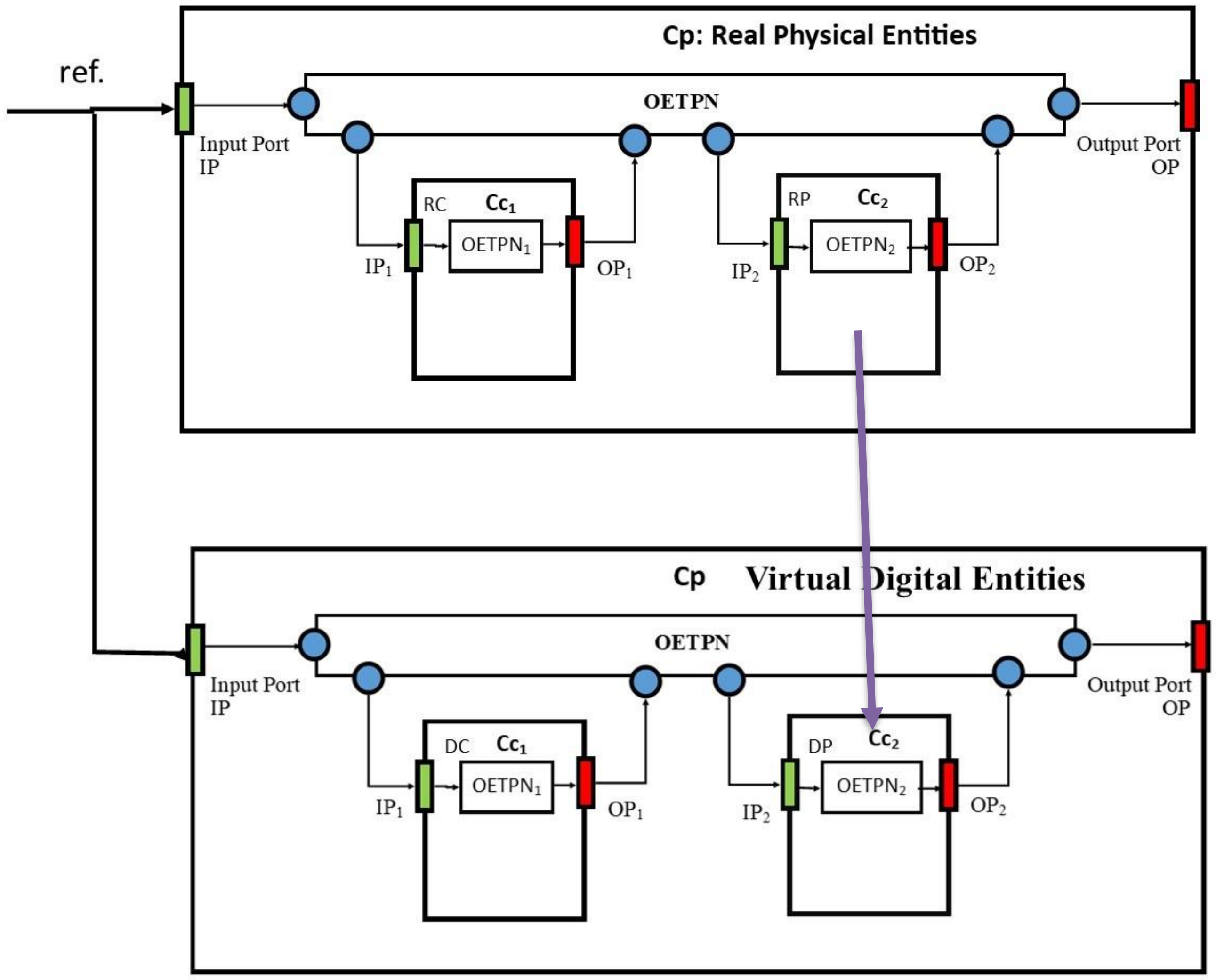
Adaptive model tracking of the real plant





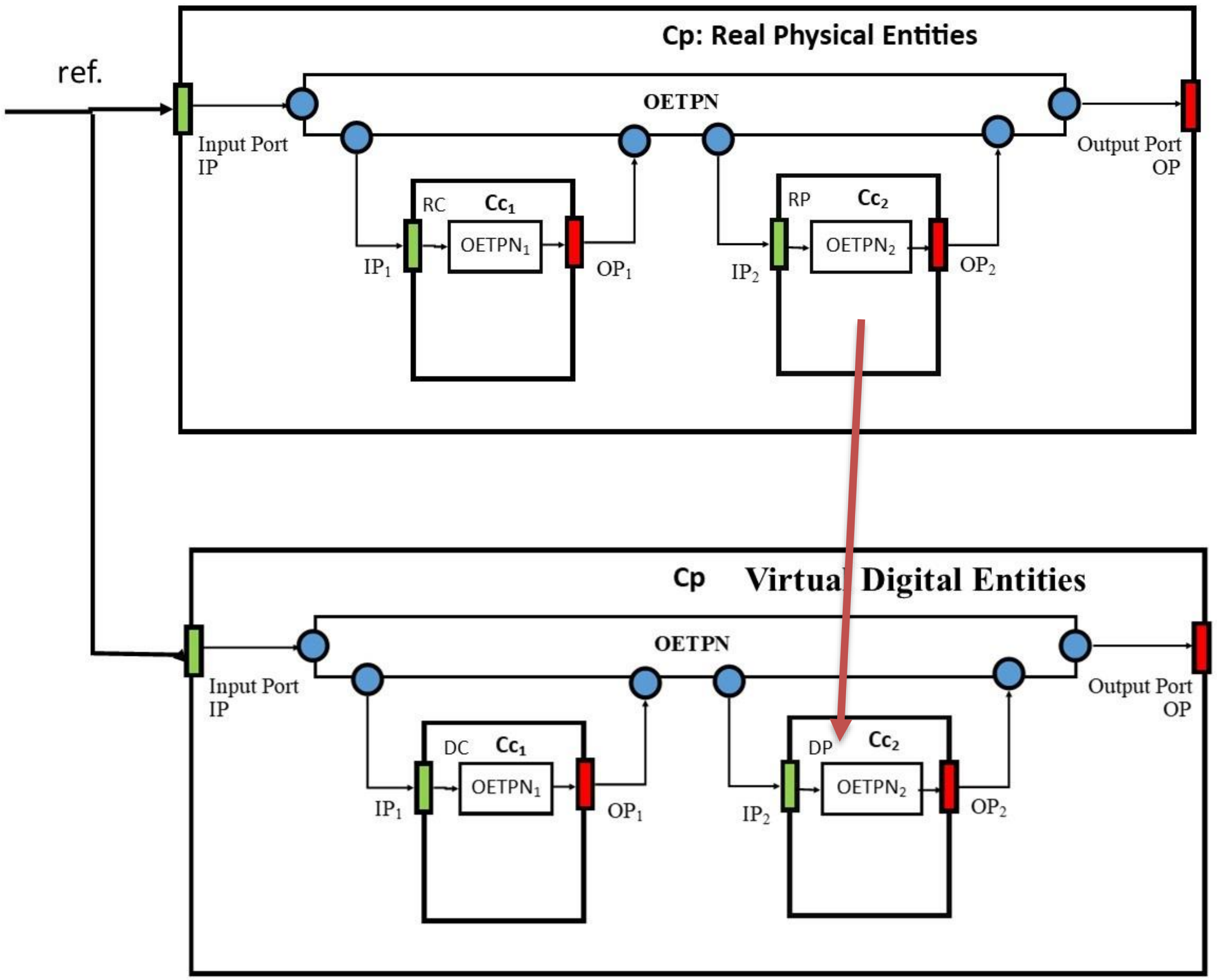
a) Adaptive model tracking of the real-plant

Link:
 Real Plant → Virtual Plant can be achieved by ports.
No direct link!



b) Performance prediction of the real plant

DTw can predict the evolution of the real plant in predicted condition.

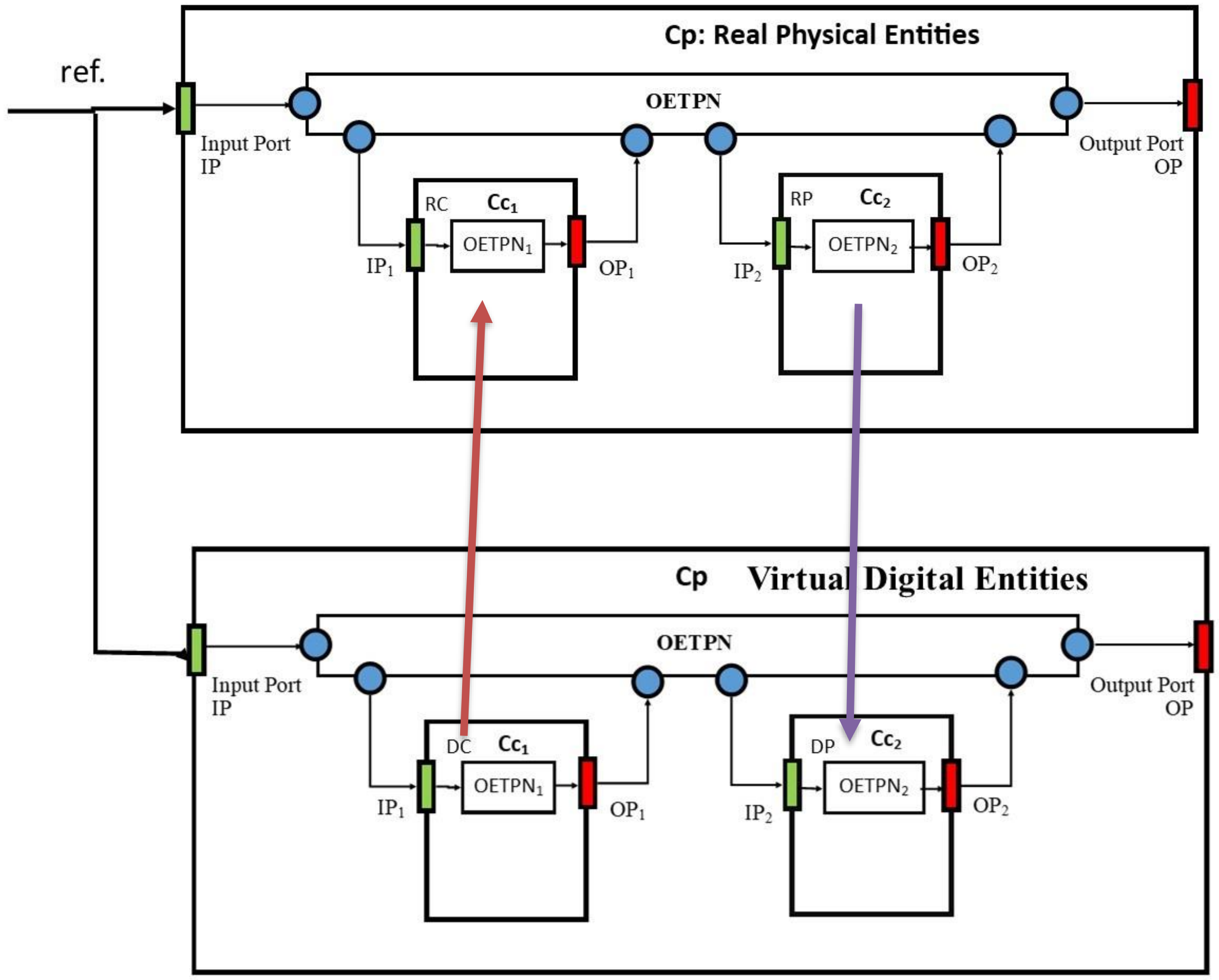


c) Sensor-fault tolerant control of the real plant

DTw can avoid the sensor fault by correcting some signals using the possible answers in given situations (or state).

d) Controller-fault tolerant control

The digital controller can replace the real controller providing the correct decisions.



*

***

****

*** END ***

****

***

*