Implementing A Cohesive P-T Nets with Inhibitor Arcs in GPenSIM

Reggie Davidrajuh

Electrical and Computer Engineering
University of Stavanger
Stavanger, Norway
E-mail: reggie.davidrajuh@uis.no

Nejm Saadallah

IRIS Energy
International Research Institute of Stavanger
Stavanger, Norway
E-mail: nejm.saadallah@iris.no

Abstract - **Petri net is a family of modeling formalism; Place-Transition net - the original member of this family - though useful as a modeling tool, it often fails to model real-life discrete event systems. This is because it lacks modeling power. There are many extensions proposed to increase the modeling power; however, these extensions while increasing the modeling power they also decrease the decision (analytical) power. Cohesive Place-Transition net with Inhibitor Arcs (CPTI) is an extension which increases the modeling power while retaining the decision power. Hence, to enjoy the analysis capabilities of CPTI nets, one has to first determine whether a given Petri net model satisfies the CPTI constrains. General Purpose Petri Net Simulator (GPenSIM) is a toolbox running on MATLAB platform that is easy to use and extensible. The aim of this paper is to present the data structures and algorithms for implementation of CPTI in GPenSIM.**

Keywords - Petri nets; Cohesive Place-Transition nets with Inhibitor Arcs; coverability tree; GPenSIM

I. INTRODUCTION

Petri net is a family of modeling formalism for modeling, simulation and performance analysis of discrete event systems; Petri net is very useful for modeling discrete event systems as concurrency and conflict that are paramount in discrete event systems can be modeled effectively [1; 2].

Place-Transition net (PT-net) is the original member (or the earliest member) of this family; though PT-net is useful as a modeling tool for modeling simple discrete event systems, it fails to model real-life systems as it lacks modeling power. There are many extensions proposed to increase the modeling power; however, these extensions while increasing the modeling power they also decrease the decision-making (analytical) power. Cohesive Place-Transition net with Inhibitor Arcs (CPTI) is an extension that increases the modeling power while retaining the decision-making power [3].

General Purpose Petri Net Simulator (GPenSIM) is a toolbox of functions that is easy to use and extensible [4]. GPenSIM runs on MATLAB platform. GPenSIM can be freely downloaded from the website indicated in the reference section as [5].

In this paper: Firstly, PT-net, CPTI, and GPenSIM are introduced in the sections-II to IV, respectively. Section-V presents an algorithm for verifying a CPTI net in GPenSIM. Section-VI presents some of the important data structures for implementation of CPTI nets in GPenSIM. Finally, section-VII presents an application example that shows verification of a CPTI net implemented in GPenSIM. The scope of this paper is limited to implementation of CPTI in GPenSIM only.

II. PLACE-TRANSITION NETS

A Place-Transition net (PT-net) is defined as follows:
A PT-net is a bipartite graph that consists of four sets,

$$PT = (P, T, A, m),$$

Where, P is the set of places, T is the set of transitions, and A is the set of arcs (from places to transitions and from transitions to places). In addition, m - known as the marking - is a vector representing the number of tokens on the places. The arcs play the role of enabling a transition to fire if the places the arcs are emerging from has enough input tokens.

PT-net can be used to analyze some simple discrete event systems, such as checking for the existence of deadlocks, capacity limitation on the buffers (boundedness), etc. However, PT-net cannot model most of the real-life systems, as it lacks modeling power; its lack of modeling power is visible as it cannot perform the “zero-test” [6], in order to show its Turing equivalence [7]. Turing machines are the most general model that is able to model any discrete event system.

The fundamental extensions to increase modeling power are the *logical extensions* and the *color extension*. Some of the logical extensions are inhibitor arcs, transition priorities, and enabling functions (enabling functions associated with the transitions) [8]. Reference [9] shows that Petri Net with inhibitor arcs have the same representational power as Turing machines. Reference [8] proves that all three logical extensions - inhibitor arcs, transition priorities, and enabling functions - are equivalent.

Colored Petri Net is an extension where the tokens can be loaded with data; tokens loaded with data enables the modeling language to work with systems where synchronization, communication, and resource sharing are important [10].

A. PT-net with Inhibitor Arcs (PTI net)

In addition to the normal arcs, PTI net has inhibiting arcs too. A PTI net is defined as follows:

A PTI net is a bipartite graph that consists of two groups

$$PTI = (PT, Ai),$$

Where *PT* is the PT-net, and *Ai* is the set of inhibiting arcs (from places to transitions).

Because of the addition of inhibiting arcs, PTI net can model any discrete event system, as it has enough modeling power; it can perform the "zero-test" in order to show its Turing equivalence.

The inhibiting arcs play the opposite role of the normal arcs as such the inhibiting arcs *prevent* a transition from firing if the inhibiting place has enough tokens. Thus the "monotonicity" property in the PT nets (more the tokens in the places, more enabled the transitions are) is lost [11; 12]. The loss of monotonicity property makes the coverability tree – a very useful tool for analysis of PT nets – unusable in PTI nets [3]. *Cohesive Place-Transition net with Inhibitor Arcs (CPTI)* is an extension that while increasing the modeling power with the help of inhibiting arcs, it retains the monotonicity property; this is achieved by imposing some restrictions on the structure of the net. Thus, the coverability tree can be used for analysis of CPTI nets as well.

III. Cohesive Place-Transition nets with Inhibitor Arcs (CPTI)

A CPTI net is made of two types of *basic elementary structures*, which are connected together by inhibiting arcs. CPTI is defined as follows:

A CPTI net is a bipartite graph that consists of the following:

$$CPTI = (FES, CES, Ai),$$

Where:

1. *FES* is a set of (zero or more) *Flat Elementary Structures*,
2. *CES* is a set of (zero or more) *Circular Elementary Structures*, and
3. *Ai* is the set of inhibiting arcs.

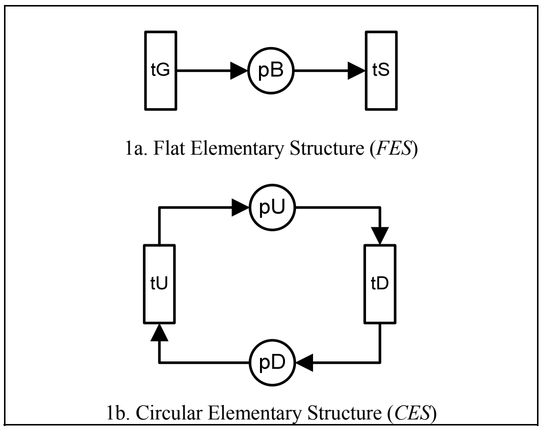
tG
pB
tS
1a. Flat Elementary Structure (*FES*)
pU
tU
tD
pD
1b. Circular Elementary Structure (*CES*)

Figure 1. The basic elementary structures of CPTI.

The figure-1a shows a *FES* structure consisting of two transitions tG (the generator transition) and tS (the sink transition). A *FES* also has a place pB; pB is the buffering place between the two transitions. Note that in a *FES*, transition tG functions as a cold-start generator, creating tokens and depositing in pB; whereas, tE does the opposite job, consuming tokens from pB and destroying them, as tE is a sink with no output places.

The figure-1b shows the second type of basic elementary structure known as the *CES*. There are two place-transition pairs in a *CES*, one transition tU and place pU in the forward (or Up) direction, and one transition tD and place pD in the reverse (or Down) direction. Note that a *CES* preserves the number of tokens in it, thus the two places pU and pD are a place invariant.

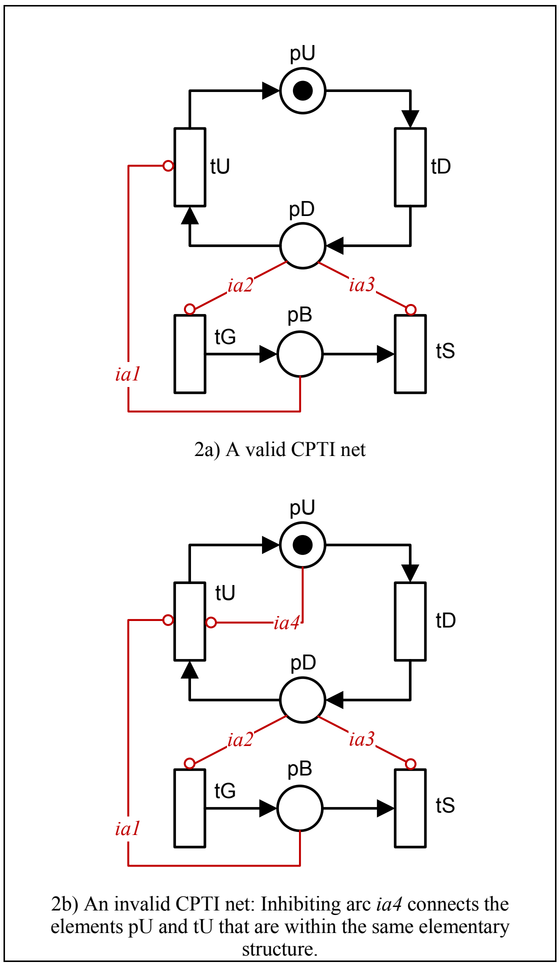
pU
tU
tD
pD
ia2
ia3
pB
tG
tS
ia1
2a) A valid CPTI net
pU
tU
ia4
tD
pD
ia2
ia3
pB
tG
tS
ia1
2b) An invalid CPTI net: Inhibiting arc *ia4* connects the elements pU and tU that are within the same elementary structure.

Figure 2. CPTI net consists of basic elementary structures connected together by inhibiting arcs.

An inhibiting arc *ai* in CPTI has a restriction: the source of the inhibiting arc (let's call it p1) and the destination of it (let's call it t1) must belong to different elementary structures (meaning, p1 and t1 cannot be part of the same elementary structure). Intuitively a *CES* represents a switch (on/off) types of situations, while a *FES* represents a so-called level control. For example, turn on or switch off a motor is modeled by a *CES*, while setting the rotational speed of the model (rpm set point) is modeled by a *FES*.

In the figure-2a, the three inhibiting arcs *ia1*, *ia2*, and *ia3* connects elements belonging different basic elementary structures, thus the net is a valid CPTI. However, in the figure-2b, the inhibiting arc *ia4* connects two elements pU and tU together which belongs to the same *CES*; thus, the net shown in the figure-2a is not a CPTI.

It is not the aim of this paper to explain how these two types of elementary structures (*CES* and *FES*) combined together by inhibiting arcs make them suitable to make us of the coverability tree. The interested reader is referred to the main source for CPTI, which is the Ph.D. done by Nejm Saadallah [3].

IV. GPENSIM

GPenSIM is a toolbox of functions, offering a Petri Net language that realizes Petri Net theory along with and a set of utility functions. In GPenSIM applications, the basic model is a Petri net model defined in a special Petri net language. GPenSIM - developed by the first author of this paper - satisfies four criteria such as [2; 4; 13]:

- Ease of use: anyone with limited mathematical and programming skills should be able to use the tool,
- Flexible: GPenSIM is applicable for modeling and simulation of discrete event systems in any domain (whether in engineering, business, or economics), as long as the system under scrutiny is discrete-event based.
- Extensible: the modeler can extend or replace any functionality available in the tool, and can also add newer functionality, and
- Compact simulation code.

In addition, GPenSIM also supports cost calculations and real-time control of discrete event systems.

A. GPenSIM support for Petri net extensions

Petri Net with a logical extension such as inhibitor arcs makes the other two extensions (transition priorities, and enabling functions) redundant [8]. Though redundant, GPenSIM supports all the three logical extensions so that the modeler may choose the extension that he finds more comfortable to use. GPenSIM supports the color extension too, as this extension provides facilities to build a more compact and modular Petri Net models [14].

The following section presents how CPTI extension is supported by GPenSIM.

V. ALGORITHM FOR VERIFYING CPTI NETS

In this section, an algorithm is given for verifying a CPTI net in GPenSIM. The subsections given below explains different parts of the algorithm. The program (MATLAB M-files) that realize the different steps of the algorithm are not given here due to brevity; interested readers are encouraged to visit the web page “Code for CPTI-GPenSIM, 2016” [15] for downloading the codes and experimenting with them.

A. The Overall Algorithm

The overall algorithm for verifying a CPTI net consists of the following steps:

Step-1. Verify whether inhibitor arcs exist. If there are no inhibitor arcs, then the net is not a CPTI net.

Step-2. Verify whether the preliminary conditions are met; the preliminary conditions are stated in the next subsection.

Step-3. Verify whether the *FES* structures (if exist) are valid.

Step-4. Verify whether the *CES* structures (if exist) are valid, and finally,

Step-5. Verify whether the inhibitor arcs connect elements from different basic elementary structures.

B. The Preliminary Conditions

For a Petri net to be a CPTI net, the following seven preliminary conditions must be satisfied:

Every place in the net ($\forall p \in P$) must satisfy the following conditions:

1. Exactly one input transition: $|\cdot p| = 1$
2. Exactly one output transition: $|p\cdot| = 1$, and
3. The input and out transitions must be different: $\cdot p \neq p\cdot$

Every transition in the net ($\forall t \in T$) must satisfy the following conditions:

4. A maximum of one input place: $|\cdot t| \leq 1$
5. A maximum of one output place: $|t\cdot| \leq 1$
6. There must be at least an input or an output place (both cannot be *null*): $|\cdot t| + |\cdot t| \geq 1$
7. If the input and output places exist, then they must be different. $\cdot t \neq \cdot t$

C. Verifying the FES structures

If there exist a transition without any input places, $|\cdot t| = 0$, then this transition (generator tG) could be part of a *FES*. The verification of the *FES* is simple: the output place pB of tG must have only one output transition, and this transition must be a sink (tS): pB = tG·, tS = pB·, where (tS·)=0.

In short, $|tG\cdot| = 1$ and $|(tG\cdot)\cdot| = 1$ and $|((tG\cdot)\cdot)\cdot| = 0$.

D. Verifying the CES structures

If there exist a transition with exactly one input place and one output place, $|\cdot t| = |t\cdot| = 1$, then this transition can be either tU or tD, could be part of a CES. The verification of the CES is also simple: the output place p1 of transition t must have only one output transition (t2). The transition t2 must also have only one output place p2. To complete the circle, the output transition of p2 must be t.

In short, $(((t\cdot)\cdot)\cdot)\cdot = t$.

E. Verifying the inhibitor arcs

For every inhibitor arc in the net, the source (place pi) and the destination (transition tj) should not belong to the same elementary structure:

$\forall ia \in IA, pi \in ESi, tj \in ESj, \rightarrow ESi \neq ESj.$

VI. Data Structures for Implementing CPTI Nets

As the data structures are very important for efficient implementation of algorithms, this section presents some of the important data structures for implementation of CPTI nets in GPenSIM. This section presents the data structures such as the incidence matrix (*A*), the inhibiting incidence matrix (*IA*), Elementary Structure Membership Table (*ESMT*), and the Elementary Structures Summary (*ESS*).

The matrices *A* and *IA* are created by the function '*pnstruct*' which is a standard function of GPenSIM. The tables *ESMT* and *ESS* are created by the function '*processCPTI*' which is part of the CPTI library of GPenSIM.

In the algorithm given in the previous section, data structures *A* and *IA* are used in all the five steps. During the steps 3 and 4, *ESMT* and *ESS* are created as a result. These two data structures are utilized in step-5.

A. The Incidence Matrix (A)

The incidence matrix *A* keeps the record of normal arc connections between places and transitions. *A* has two parts: the input incidence matrix (for arcs from places to transitions) and the output incidence matrix (for arcs from transitions to places). As an example, the incidence matrix *A* for the CPTI shown in the figure-2a is given below, in the figure-3.

	Input incidence matrix			*Output incidence matrix*		
	pB	pD	pU	pB	pD	pU
tD	0	0	1	0	1	0
tG	0	0	0	1	0	0
tS	1	0	0	0	0	0
tU	0	1	0	0	0	1

Figure 3. The incidence matrix *A*.

B. The Inhibitor Incidence Matrix (IA)

Just like the incidence matrix *A* represents the normal arcs between the elements of a net, the inhibitor incidence matrix *IA* represents the inhibitor arcs of a net. Taking the CPTI shown in the figure-2a, there are 3 inhibitor arcs, from pB to tU, from pD to tG, and from pD to tS; the weights of these three inhibitor arcs are of default value 1. Thus, the inhibitor incidence matrix *IA* will look like the one shown below, in the figure-4:

	pB	pD	pU
tD	∞	∞	∞
tG	∞	1	∞
tS	∞	1	∞
tU	1	∞	∞

Figure 4. The inhibiting incidence matrix *IA*.

The value of infinite (∞) in *IA* means nonexistent inhibitor arcs between the corresponding elements.

C. The Elementary Structure Membership Table (ESMT)

The Elementary Structure Membership Table (*ESMT*) shows the membership (affiliation) of each element in the net.

ESMT has one row for each place and each transition; the first rows of *ESMT* are for places, and the rest of the rows are for the transitions. *ESMT* has two columns: the first column represents the elementary structure to which the element belongs to, and the second column represents the type of the elementary structure (1= *FES*, and 2=*CES*).

The figure-5 shows the *ESMT* of the CPTI net shown in the figure-2a. There are only two elementary structures in the net; the elements tG, pB, and tS belongs to the first elementary structure (ES = 1), which is a *FES*. The rest of the elements (pD, pU, tD, and tU) belong to the second elementary structure (ES=2), which is a *CES*.

	Element Structure Number	*Element Structure Type*
pB	1	*FES*
pD	2	*CES*
pU	2	*CES*
tD	2	*CES*
tG	1	*FES*
tS	1	*FES*
tU	2	*CES*

Figure 5. The elementary structures membership table (*EMST*).

D. The Elementary Structures Summary (ESS)

The Elementary Structures Summary (*ESS*), as its name suggests, presents overall information about the elementary structures. ESS has a number of rows, each row describes the details of a specific elementary structure. Each row starts with a field indicating what type elementary structures it is; the rest of the row indicates which elements belonging to the specific elementary structure.

Element Structure Type	*The members of the element structure*			
FES	tG	pB	tS	*X*
CES	tU	pU	tD	pD

Figure 6. The elementary structures summary table (*ESS*).

The figure-6 shows the *ESS* of the CPTI net shown in the figure-2a. The *ESS* has two rows as there are only two elementary structures in the CPTI net. The first row is for the *FES* consisting of the elements tG, pB, and tS. The second row is for the *CES* consisting of the elements tU, pU, tD, and tD.

VII. Application Example

Let us take the CPTI net shown in the figure-2a as the application example. Implementation of a Petri net model of a discrete event system in GPenSIM usually needs at least two files [2]: the Main Simulation File (MSF), the Petri net Definition File (PDF).

The Petri net definition file (PDF) is the code implementation of the static Petri Net graph. The PDF for the example net is shown in the figure-7; the PDF declares the sets of places, transitions, the normal arcs, and the inhibitor arcs.

```
function [png] = ems2016_pdf()

png.PN_name = 'EMS2016 example';
% Declare the CES
C1_Ps = {'pU','pD'}; % set of places
C1_Ts = {'tU','tD'}; % set of transitons
C1_As = {'pU','tD',1,... % normal arcs
   'tD','pD',1, 'pD','tU',1, 'tU','pU',1'};

% Declare the FES
F1_Ps = {'pB'};       % set of places
F1_Ts = {'tG','tS'}; % set of transitions
F1_As = {'tG','pB',1,'pB','tS',1};%normal
arcs

% Declare complete set of all elements
png.set_of_Ps= [C1_Ps, F1_Ps]; % places
png.set_of_Ts= [C1_Ts, F1_Ts]; % trans
png.set_of_As= [C1_As, F1_As]; % normal
arcs

% set of all the INHIBITING arcs
png.set_of_Is = {'pD','tG',1, ...
  'pD','tS',1, 'pB','tU',1};
```

Figure 7. The Petri net definition file (PDF).

The main simulation file (MSF) is shown in the figure-8. In the MSF, firstly, the initial dynamics (e.g. initial tokens in places, firing times of transitions) are declared. Then, the net is checked whether it is a CPTI or not.

```
pns = pnstruct('ems2016_pdf'); % declare PDF
dyn.m0 = {'pU',1}; % initial tokens (marking)

% create the Petri net initial dynamics
pni = initialdynamics(pns, dyn);
processCPTI(pni); % process (verify) the net
```

Figure 8. The main simulation file (MSF).

A. *The Verification Results*

The results of the verification process is given below:

```
> Verifying the net "EMS2016 example" ....
>
> *** Inhibiting arcs: ***
> pD -O tG
> pD -O tS
> pB -O tU
>
> ***  The Elementary Structures: ***
> ES-1 (FES): tG -> pB -> tS
> ES-2 (CES):  -> tD -> pD -> tU -> pU ->
>
> This is a CPTI !
```

VIII. Conclusion

GPenSIM is a general purpose Petri net simulator that is used for modeling and simulation of discrete event systems, both in academia and in industries. GPenSIM is commended as the ideal tool for working with marked graphs [16].

This paper focuses on incorporating Cohesive Place-Transition Inhibitor (CPTI) net into GPenSIM.

References

[1] T. Murata, "Petri nets: Properties, analysis and applications," Proceedings of the IEEE, 77(4), 1989, pp. 541-580.

[2] R. Davidrajuh, "Developing a Petri Nets based Real-Time Control Simulator," International Journal of Simulation, Systems, Science & Technology (IJSSST), vol. 12, issue. 3, pp. 28-38, November 2012

[3] N. Saadallah, "Scheduling Drilling Processes with Petri Nets", PhD Thesis, Universitety of Stavanger, Stavanger, Norway, 2013.

[4] R. Davidrajuh, "Developing a new Petri net tool for simulation of discrete event systems". In 2008 Second Asia International Conference on Modelling & Simulation (AMS) (pp. 861-866). IEEE.

[5] GPensim website, "General Purpose Petri Net Simulator (GPenSIM)". Available: http://www.davidrajuh.net/gpensim/

[6] J. L. Peterson, Petri Net Theory and the Modeling of Systems. NJ, USA: Prentice-Hall, 1981.

[7] J. C. Shepherdson and H. E. Sturgis, "Computability of recursive functions," J. ACM, 10, April 1963, pp. 217-255.

[8] G. Ciardo, "Toward a Definition of Modeling Power for Stochastic Petri Net Models," Proc. International Workshop on Petri Nets and Performance Models, 1987, pp. 54-62

[9] T. Agerwala, A complete model for representing the coordination of asynchronous processes. Baltimore, Maryland: Hopkins Computer Research Report 32, Johns Hopkins University, 1974

[10] Y. G. Saffar, M. Jamali, and M. Neshati, "Colored Petri Nets (CPN)". Available: www.cs.ubc.ca/~jamalim/resources/CPN.ppt

[11] A. Finkel. The minimal coverability graph for Petri nets. 12th International conference on Application and Theory of PetriNets, 1993, London, UK.

[12] L. Ghomri and H. Alla, "Modeling and analysis using hybrid Petri nets. Nonlinear Analysis: Hybrid Systems", 1(2), 2007, pp. 141-153.

[13] R. Davidrajuh and B. Lin, "Exploring airport traffic capability using Petri net based model," Expert Systems with Applications, 38(9), pp. 10923-10931.

[14] K. Jensen, Colored Petri Nets. Vol. I., II., III. Second edition. Springer, Berlin, 1997.

[15] Code for CPTI-GPenSIM 2016, Available: http://davidrajuh.net/gpensim/Code-for-CPTI.htm

[16] A. Cameron, M. Stumptner, N. Nandagopal, W. Mayer, and T. Mansell, "Rule-based peer-to-peer framework for decentralised real-time service oriented architectures," Science of Computer Programming, 97, 2015, pp. 202-234.