FUZZY TIMED OBJECT-ORIENTED PETRI NET

Hua Xu and Peifa Jia
State Key Laboratory of Intelligent Technology and Systems, Tsinghua University, Beijing, 100084, P. R. China

Abstract: The goal of this work is to extend a model of timed object-oriented Petri nets (TOPN) to allow modeling and analyzing dynamic systems with timing effect on system information. In the proposed Fuzzy timed object-oriented Petri net (FTOPN), we attach temporal fuzzy sets to each transition objects accounting for the aging of information. In particular, we investigate a new way to represent and deal with timing effect in dynamic systems. FTOPN also supports learning similar to that in fuzzy timed Petri net (FTPN). Finally, we use FTOPN to model a real decision making model of our cooperative multiple robot system (CMRS) to demonstrate its following benefits: independent training for its supporting object abstraction and size reconfiguration for its object granularity control function.

Key words: Petri nets; Fuzzy sets; Object-oriented method; Timing effect; Learning

1. INTRODUCTION

Petri nets [1], [2] have been widely used to model various discrete event systems [3]. Characterized as concurrent, asynchronous, distributed, parallel, nondeterministic, and/or stochastic [1], Petri nets have gained more and more applications. However, when they are used to analyze and model practical systems in different fields, models may be too complex to be analyzed. These years, object-oriented concepts have been introduced into Petri nets and object-oriented Petri nets (OPN) are presented such as HOONet [4], OBJSA [5], COOPN/2 [6] and LOOPN++ [7], which are suggested on the base of colored Petri Net (CPN) [8]. Then for providing the ability of modeling time critical systems, timed hierarchical object

oriented Petri net (TOPN) [9] is proposed on the base of HOONet [4]. It supports temporal description and object-oriented concepts. Modeling features in TOPN support abstracting complex systems, so the corresponding models can be simplified effectively.

On the other hand, although Petri nets can be used to model and analyze different systems, they fail to model the timing effects in dynamic systems. Recently, fuzzy timed Petri net (FTPN) [10] has been presented and it has solved this modeling problem, which is on the base of temporal fuzzy sets and Petri nets. However, similar to the general Petri Nets, FTPN may also meet the complexity problem, when it is used to model complex dynamic systems.

In this paper, fuzzy timed object-oriented Petri net (FTOPN) is proposed on the base of TOPN and FTPN, whose aim is to solve the timing effects and other modeling problems of dynamic systems. This paper is organized as the following. In Section 2, the preliminary notions of TOPN are reviewed. Section 3 presents our FTOPN. Section 4 discusses the learning of FTOPN. In section 5, FTOPN is used to model the decision making procedure of our cooperative multiple robot system (CMRS) to demonstrate its model abstraction and reconfiguration benefits. Finally, the conclusion and future work can be found in section 5.

2. PRELIMINARY NOTIONS OF TOPN

Formally TOPN is a four-tuple (OIP, ION, DD, SI), where (OIP,ION,DD) is an ordinary object Petri net-"HOONet" [4] and *SI* associates a *static* (*firing*) temporal *interval* SI: $\{o\}\rightarrow[a, b]$ with each object o, where a and b are rationals in the range $0\leq a\leq b\leq+\infty$, with $b\neq+\infty$. The four parts in TOPN have different function roles. *Object identification place* (OIP) is a unique identifier of a class. *Internal timed object net* (ION) is a net to depict the behaviors (methods) of a class. *Data dictionary* (DD) declares the attributes of a class in TOPN. And *static time interval function* (SI) binds the temporal knowledge of a class in TOPN. There are two kinds of places in TOPN. They are common places (represented as circles with thin prim) and abstract places (represented as circles with bold prim). Abstract places are also associated with a static time interval. Because at this situation, abstract places represent not only firing conditions, but also the objects with their own behaviors. So, abstract places (TABP) in TOPN also need to be associated with time intervals. One problem to be emphasized is that the tokens in abstract places need to have two colors at least. Before the internal behaviors of an abstract place object are fired, the color of tokens in it is one color (represented as hollow token in this paper). However, after fired, the

color becomes the other one (represented as liquid token in this paper). At this time, for the following transitions, it is just actually enabled. There are three kinds of transitions in TOPN. The timed primitive transition (represented as rectangles with thin prim) (TPIT), timed abstract transition (represented as rectangles with bold prim) (TABT) and timed communication transition (represented as rectangles with double thin prim) (TCOT).

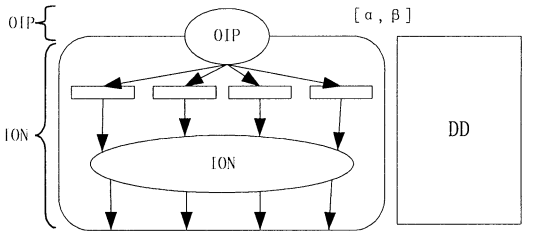

Figure 1. The General Structure of TOPN

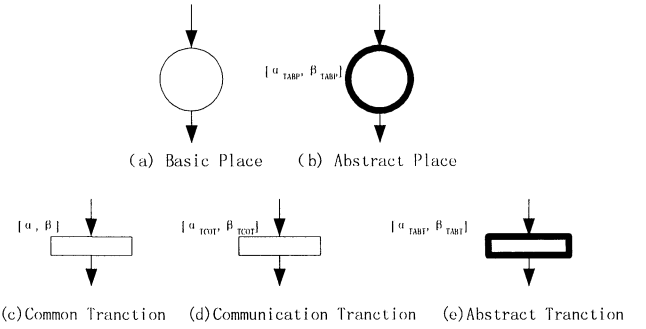

Figure 2. Places and Transactions in TOPN

Static time intervals change the behavior of TOPN just similar to a time Petri net in the following way. If an object o with SI(o)=[a, b] becomes enabled at time I_0, then the object o *must be* fired in the time interval [I_0+a, I_0+b], unless it becomes disabled by the removal of tokens from some input place in the meantime. The *static earliest firing time* of the object o is a; the *static latest firing time* of o is b; the *dynamic earliest firing time* (EFT) of t is I_0+a; the *dynamic latest firing time* (LFT) of t is I_0+b; the *dynamic firing interval* of t is [I_0+a, I_0+b].

The state of TOPN (Extended States-"ES") is a 3-tuple, where ES=(M, I,

path) consists of a marking M, a firing interval vector I and an execution path. According to the initial marking M_0 and the firing rules mentioned above, the following marking at any time can be calculated. The vector--"I" is composed of the temporal intervals of enabled transitions and TABPs, which are to be fired in the following states. The dimension of I equals to the number of enabled transitions and TABPs at the current state. The firing interval of every enabled transition or TABP can be got according to the calculation formula of EFT and LFT in TOPN [9].

For enabling rules in TOPN, two different situations exist. A transition t in TOPN is said to be enabled at the current state (M, I, path), if each input place p of t contains at least the number of *solid tokens* equal to the weight of the directed arcs connecting p to t in the marking M. If the TABP object is marked with a *hollow token*, it is enabled. At this time, its ION is enabled. After the ION has been *fired*, the tokens in TABP are changed into *solid ones*.

We say that an object *o* is *fireable* in state (M, I, path) if it is enabled, and if it is legal to fire *o* next. This will be true if and only if the EFT of *o* is less than or equal to the LFT of all other enabled transitions. Of course, even with strong time semantics, *o*'s being fireable in state (*M*, I, path) does not necessarily mean that *t will* fire in the time interval I.

3. FUZZY TIMED OBJECT-ORIENTED PETRI NETS

Similar to FTPN [10], fuzzy set concepts are introduced into TOPN [9]. Then FTOPN is proposed, which can describe fuzzy timing effect in dynamic systems.

Definition 1: FTOPN is a six-tuple, FTOPN= (OIP, ION, DD, SI, R, I) where

1) Suppose OIP=(oip, pid, M_0, status), where oip, pid, M_0 and status are the same as those in HOONet [4] and TOPN [9].
   - oip is a variable for the unique name of a FTOPN.
   - pid is a unique process identifier to distinguish multiple instances of a class, which contains return address.
   - M_0 is the function that gives initial token distributions of this specific value to OIP.
   - status is a flag variable to specify the state of OIP.
2) ION is the internal net structure of FTOPN to be defined in the following. It is a variant CPN that describes the changes in the values of attributes and the behaviors of methods in FTOPN.
3) DD formally defines the variables, token types and functions (methods) just like those in HOONet [4] and TOPN [9].

4) SI is a static time interval binding function, SI: {OIP}→Q^*, where Q^* is a set of time intervals.
5) R: {OIP} → r, where r is a specific threshold.
6) I is a function of the time *v*. It evaluates the resulting degree of the abstract object firing. □

Definition 2: An internal object net structure of TOPN, ION = (P,T,A,K,N,G,E,F,M_0)

1) P and T are finite sets of places and transitions with time restricting conditions attached respectively.
2) A is a finite set of arcs such that P∩T=P∩A=T∩A=Φ.
3) K is a function mapping from P to a set of token types declared in DD.
4) N, G, and E mean the functions of nodes, guards, and arc expressions, respectively. The results of these functions are the additional condition to restrict the firing of transitions. So they are also called *additional restricting conditions*.
5) F is a special arc from any transitions to OIP, and notated as a body frame of ION.
6) M_0 is a function giving an initial marking to any place the same as those in HOONet [4] and TOPN [9]. □

Definition 3: A set of places in TOPN is defined as P=PIP∪TABP, where

1) Primary place PIP is a three-tuple: PIP =(P,R,I), where
   - P is the set of common places similar to those in PN [1], [2].
2) Timed abstract place (TABP) is a six-tuple: TABP= TABP(pn, refine state, action, SI, R, I), where
   - pn is the identifier of the abstract timed place.
   - refine state is a flag variable denoting whether this abstract place has been refined or not.
   - action is the static reaction imitating the internal behavior of this abstract place.
3) SI, R and I are the same as those in *Definition 1*.□

Definition 4: A set of transitions in TOPN can be defined as T= TPIT∪TABT∪TCOT, where

1) Timed primitive transition TPIT = TPIT (BAT, SI), where
   - BAT is the set of common transitions.
2) Timed abstract transition TABT= TABT (tn, refine state, action, SI), where
   - tn is the name of this TABT.
3) Timed communication transition TCOT=TCOT (tn, target, comm type, action, SI).
   - tn is the name of TCOT.
   - target is a flag variable denoting whether the behavior of this TCOT has been modeled or not. If target="Yes", it has been modeled. Otherwise, if target="No", it has not been modeled yet.

- comm type is a flag variable denoting the communication type. If comm type ="SYNC", then the communication transition is synchronous one. Otherwise, if comm type="ASYN", it is an asynchronous communication transition.

4) *SI is* the same as that in *Definition 1.*

5) *refine state* and *action* are the same as those in *Definition 3.* □

Similar to those in FTPN [10], the object t fires if the foregoing objects come with a nonzero marking of the tokens; the level of firing is inherently continuous. The level of firing (z(*v*)) assuming values in the unit interval is governed by the following expression:

$$z(v) = (\mathop{T}_{i=1}^{n}(r_i \rightarrow x_i(v'))sw_i)tI(v) \quad (1)$$

where T (or t) denotes a t-norm while "s" stands for any s-norm. "*v*" is the time instant immediately following *v'*. More specifically, $x_i(v)$ denotes a level of marking of the i^{th} place. The weight w_i is used to quantify an input coming from the i^{th} place. The threshold r_i expresses an extent to which the corresponding place's marking contributes to the firing of the transition. The implication operator ($\rightarrow$) expresses a requirement that a transition fires if the level of tokens exceeds a specific threshold (quantified here by r_i).

Once the transition has been fired, the input places involved in this firing modify their markings that is governed by the expression

$$x_i(v)=x_i(v')t(1-z(v)) \quad (2)$$

(Note that the reduction in the level of marking depends upon the intensity of the firing of the corresponding transition, z(*v*).) Owing to the t-norm being used in the above expression, the marking of the input place gets lowered. The output place increases its level of tokens following the expression:

$$y(v)=y(v')sz(v) \quad (3)$$

The s-norm is used to aggregate the level of firing of the transition with the actual level of tokens at this output place. This way of aggregation makes the marking of the output place increase.

The FTOPN model directly generalizes the Boolean case of TOPN and OPN. In other words, if $x_i(v)$ and w_i assume values in {0, 1} then the rules governing the behavior of the net are the same as those encountered in TOPN.

4. LEARNING IN FTOPN

The parameters of FTOPN are always given beforehand. In general, however, these parameters may not be available and need to be estimated just like those in FTPN[10]. The estimation is conducted on the base of some

experimental data concerning marking of input and output places. The marking of the places is provided as a discrete time series. More specifically we consider that the marking of the output place(s) is treated as a collection of target values to be followed during the training process. As a matter of fact, the learning is carried in a supervised mode returning to these target data.

The connections of the FTOPN (namely weights w_i and thresholds r_i) as well as the time decay factors α_i are optimized (or trained) so that a given performance index $\boldsymbol{Q}$ becomes minimized. The training data set consists of (a) initial marking of the input places $x_i(0),\ldots,x_n(0)$ and (b) target values—markings of the output place that are given in a sequence of discrete time moments, that is target(0), target(1),..., target(K).

In our FTOPN, the performance index $\boldsymbol{Q}$ under discussion assumes the form of the following sum:

$$\boldsymbol{Q}=\sum_{k=1}^{K}(t\arg et(k)-y(k))^2 \tag{4}$$

where the summation is taken over all time instants (k =1, 2,... , K).

The crux of the training in FTOPN models follows the general update formula being applied to the parameters:

$$\textbf{param}(\text{iter}+1)=\textbf{param}(\text{iter})-\gamma\nabla_{\textbf{param}}\boldsymbol{Q} \tag{5}$$

where γ is a learning rate and $\square_{\textbf{param}}\boldsymbol{Q}$ denotes a gradient of the performance index taken with respect to all parameters of the net (here we use a notation **param** to embrace all parameters in FTOPN to be trained).

In the training of FTOPN models, marking of the input places is updated according to the following form:

$$\tilde{x_i} = x_i(0)T_i(k) \tag{6}$$

where $T_i(k)$ is the temporal decay. And Ti(k) complies with the following form. In what follows, the temporal decay is modeled by an exponential function,

$$T_i(k)=\begin{cases}\exp(-\alpha_i(k-k_i)) & if \quad k>k_i,\\ 0 \quad others\end{cases} \tag{7}$$

The level of firing of the place can be computed as the following:

$$z=(\underset{i=1}{\overset{n}{T}} \quad ((r_i \rightarrow \tilde{x_i})sw_i)) \tag{8}$$

The successive level of tokens at the output place and input places can be calculated as:

$$y(k) = y(k-1)\sigma z,\ x_i(k) = x_i(k-1)\tau(1-z) \tag{9}$$

We assume that the initial marking of the output place y(0) is equal to zero, y(0)=0. The derivatives of the weights w_i are computed as follows:

$$\frac{\partial}{\partial w_i}(target(k) - y(k))^2 = -2(target(k) - y(k)\frac{\partial y(k)}{\partial w_i}) \tag{10}$$

where i=1,2,..., n. Note that y(k+1)=y(k)σz(k).

5. A MODELING EXAMPLE

In cooperative multiple robot systems (CMRS), every robot is controlled according to different system information such as other robot states, its own states and task assignment. As the information may not be available from all sensors or sources at the same time moment, the one that occurs earlier needs to be discounted over time as becoming less relevant. That is to say, information timing effects exist in this kind of dynamic systems. However, in the control of every robot system, every kind of information is required simultaneously. As the information readings could come at different time instants and be collected at different sampling frequency, we encounter an inevitable timing effect of information collected by the system and sensors. It becomes apparent that its relevance is the highest at the time moment when the system sensor captures it but then its relevance has to be discounted over the passage of time. This is an effect of aging that has to be viewed as an integral part of the model. So FTOPN is used to model our CMRS. At the same time, FTOPN can reduce the model complexity and can model complex decision making processes in different levels, because of the OO abstraction concept supported in FTOPN. It triggers interest in the class of the FTOPN.

5.1 CMRS Example

In our experiment, there are two cooperative robots. FTOPN is used to model the information fusion process in the decision making of scheduling robot in every robot. Because the model is hierarchical, only the highest level of the model is depicted in Figure.3.

In the model of Figure.3, 3 place objects are used to represent 3 kinds of information to be fused. Each kind of information may include different detailed contents. For example, "other robot state" may include other robots' working state, location, speed, movement direction, etc al. So every kind of information is also an abstract object. On the other hand, the relative firing

temporal interval is [a, b] of the object. The information should be sampled and processed in this relative interval. So does command sending. If the relative time exceeds it, the information should be sampled again and task should be reassigned. In the model, one transaction object represents the information fusion process. The timing effect on the fusion is depicted in Figure.4. The information "other robot state" and "own state" complies with the rule in Figure.4 (1). The other information complies with Figure.4 (2). After the fusion, a new command will be sent in this relative interval. The command to be sent is also a place object, which include robot schedule and control commands.

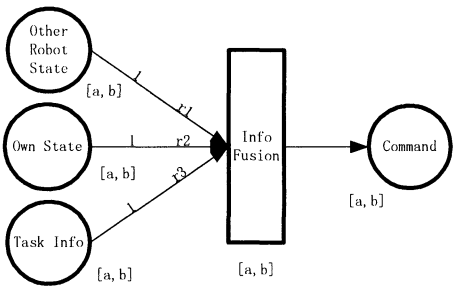

Figure3. The FTOPN Model

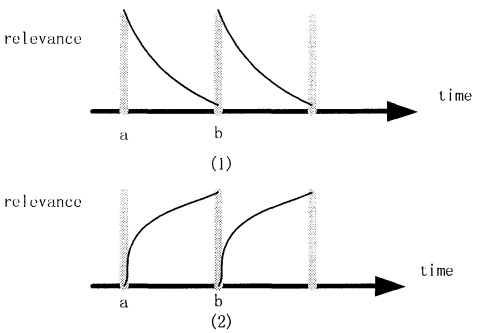

Figure 4. The Relevance

What's more, all the objects in Figure.3 can also be depicted in details by FTOPN. For example, the object—"Other Robot State" in Figure.3 can also be modeled concretely with FTOPN. The detailed model of the object is depicted in Figure.5. It is also an independent fuzzy reduction process. According to the modeling and analysis requirements, the detailed model can be unfolded directly in the model of Figure.3. At the same time, its training

can be conducted independently. It can also be reduced independently and the reduction results will be used as the believing effect of the corresponding object in the higher level of the FTOPN model in Figure.3.

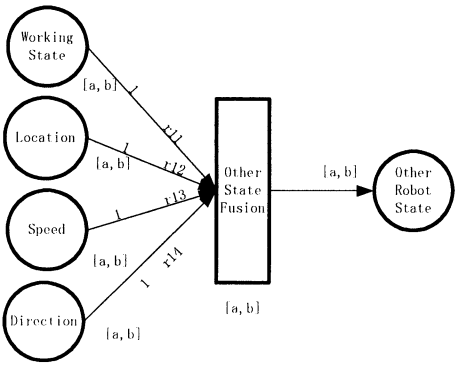

Figure 5. The Object-"Other Robot State" Model

After completing the FTOPN model, the learning algorithm of FTOPN can be used to train our model and adjust it to fulfill the practical requirements.

5.2 Application Analysis

From the view of the former FTOPN modeling example, objects in FTOPN model can be abstracted. They can be modeled and represented in other levels independently. At the same time, the training and fuzzy reduction can also be conducted independently. So for the abstraction concepts supported, the model complexity has been reduced effectively because of abstraction concepts in FTOPN. And the fuzzy reduction procedures have been simplified. Essentially, hierarchical modeling idea in FTOPN is to control model size by abstracting objects in FTOPN model. In nature, OO abstraction concepts are used to control fuzzy knowledge granularity in FTOPN. Because OO concepts are supported in FTOPN, the abstract objects can be unfolded or abstracted in FTOPN model flexibly. Our modeling focus can also be paid upon the important parts.

A comparative analysis between FPN, PN and neural network is conducted in [11]. Table.1 summarizes the main features of the fuzzy timed Object-oriented Petri nets and contrast these with the structures with which the proposed constructs have a lot in common, namely FPN and TFPN. It becomes apparent that FTOPN combine the advantages of both FPN in terms of their learning abilities and the glass-style of processing (and architectures) of Petri nets with the abstraction of OO concepts.

Table.1 Object Petri nets, Fuzzy Petri nets and Fuzzy Time Object-oriented Petri nets: a comparative analysis

Characteristics	Object Petri nets	Fuzzy Petri Nets	Fuzzy Timed Object Oriented Petri nets
Learning Aspects	From non-existent to significantly limited (the same as those of common Petri nets).	Significant learning abilities parametric optimization of the connections of the net. Structural optimization can be exercised through a variable number of the transitions utilized in the network.	Significant learning abilities as well as FPN. Distributed learning (training) abilities are supported in different independent objects on various system model levels.
Knowledge Representation Aspects	Glass Box or black box style knowledge representation supporting as a result of abstracting a given problem (problem specification) onto the structure of the net in different levels. Well-defined semantics of places and transitions	Transparent knowledge representation (glass box processing style) the problem (its specification) is mapped directly onto the topology of the fuzzy Petri net. Additionally, fuzzy sets deliver an essential feature of continuity required to cope with continuous phenomena encountered in a vast array of problems (including classification tasks)	Glass Box Style (Transparent Knowledge Representation) and Black Box Processing is supported at the same time. The problem (its specification) is mapped directly onto the topology of FTOPN. Knowledge representation granularity reconfiguration reacts on the reduction of model size and complexity.

6. CONCLUSIONS AND FUTURE WORKS

Timing effect is a usual phenomenon in dynamic systems especially in time critical systems. In order to model, analyze and simulate this kind of systems, this paper proposes fuzzy timed object-oriented Petri net (FTOPN) on the base of TOPN [9] and FTPN [4]. Temporal fuzzy sets are used in FTOPN to describe the timing effect and evaluation levels can be got according to the information arriving time and specific fuzzy relevance function. What's more, compared with FTPN (or FPN) models, the model size and reduction complexity of FTOPN models can be reduced by

controlling object granularity because of supporting OO concept in FTOPN. Every abstract object in FTOPN can train and reduce independently according to the modeling and analysis requirements for OO concepts supported in FTOPN. The validity of this modeling method has been demonstrated by using it in the simulation of the decision information fusion process in our CMRS.

State analysis needs to be studied in the future. Xu [9] has proposed an extended State Graph to analyze the state change of TOPN models. With the temporal fuzzy sets introduced into TOPN, the certainty factor about object firing (state changing) needs to be considered in the state analysis.

ACKNOWLEDGEMENTS

This work is jointly supported by the National Nature Science Foundation for Youth Fund (Grant No: 60405011) and China Postdoctoral Foundation for China Postdoctoral Science Fund (Grant No: 20040350078).

REFERENCES

1. Tadao Murata, "Petri Nets: Properties, Analysis and Applications", Proceedings of IEEE, Vol.77, No.4, April 1989, pp.541-580
2. James L. Peterson, Petri Net Theory and the Modeling of Systems, Prentice-Hall, 1991
3. John O.Moody, Panos J. Antsaklis, Supervisory Control of Discrete Event Systems Using Petri Nets, Kluwer Academic Publishers, 1998:1-4
4. Jang-Eui Hong, Doo-Hwan Bae, Software Modeling And Analysis Using a Hierarchical Object-oriented Petri net, Information Sciences 130(2000), 133-164
5. E. Battiston, F.D. Cindio, G. Mauri, OBJSA Nets: a class of high-level nets having objects as domains, in: APN'88, Lecture Notes in Computer Science, vol. 340, 1988, pp. 20±43.
6. Biberstein, D. Buchs, An object-oriented specification language based on hierarchical algebraic Petri nets, in: Proceedings of the IS-CORE Workshop Amsterdam, September 1994 (and TR: EPFL-DI 94-76)
7. C. Lakos, C. Keen, LOOPN++: a new language for object-oriented Perti nets, Technical Report R94-4, Networking Research Group, Univesity of Tasmania, Australia, April 1994.
8. Kurt Jensen, Coloured Petri Nets: Basic Concepts, Analysis methods and Practical Use, 1992, Berlin: Springer,1:65-85
9. Hua Xu, Studies on Timed Hierarchical Object-oriented Petri Net and Its Application, Doctor Thesis, Tsinghua University, 2003
10. Pedrycz, Witold; Camargo, Heloisa: Fuzzy timed Petri nets, Fuzzy Sets and Systems Volume: 140, Issue: 2, December 1, 2003, pp. 301-330
11. Witold Pedrycz, Generalized fuzzy Petri nets as pattern classifiers, Pattern Recognition Letters 20 (1999):1489—1498