We are IntechOpen, the world's leading publisher of Open Access books Built by scientists, for scientists

4,800
Open access books available

122,000
International authors and editors

135M
Downloads

Our authors are among the

154
Countries delivered to

TOP 1%
most cited scientists

12.2%
Contributors from top 500 universities

CLARIVATE ANALYTICS BOOK CITATION INDEX INDEXED

WEB OF SCIENCE™

Selection of our books indexed in the Book Citation Index in Web of Science™ Core Collection (BKCI)

Interested in publishing with us?
Contact book.department@intechopen.com

Numbers displayed above are based on latest data collected.
For more information visit www.intechopen.com

12

Timed Hierarchical Object-Oriented Petri Net

Hua Xu
State Key Laboratory of Intelligent Technology and Systems
Tsinghua National Laboratory for Information Science and Technology
Department of Computer Science and Technology,
Tsinghua University, Beijing 100084,
P. R. China

1. Introduction

Petri nets (Murata, 1989) (Peterson, 1991) have been widely used to model various discrete event systems (Moody & Antsaklis, 1998). Characterized as concurrent, asynchronous, distributed, parallel, nondeterministic, and/or stochastic (Murata, 1989), Petri nets have gained more and more applications. However, when they are used to analyze and model systems of different domains, the shortages of this kind of formal method still exist. Basic Petri nets lack temporal knowledge description, so they have failed to describe the temporal constraints in time critical or time dependent systems. The introduction of temporal knowledge into Petri nets has increased not only the modeling power but also the model complexity (Wang et al. 2000). The improved models of Petri nets (Wang, 1998) include Timed Petri Net (Ramchandi, 1974), Stochastic Timed Petri Net (Florin etc al., 1991) and Time Petri Net (TPNs) (Merlin & Farber, 1976). In TPNs (Merlin & Farber, 1976), each bar has two times specified. The first time denotes the minimal time that must elapse from the time that all the input conditions of a bar are enabled until this bar can fire. The other time denotes the maximum time that the input conditions can be enabled and the bar does not fire. After this time, the bar must fire. In general, these two times give some measures of minimal and maximal execution times of the bars.

The reachability (coverability) analysis is one of the main analysis methods for Petri nets (Murata, 1989), in which the coverability tree is always used. It permits the automatic translation of behavioral specification models into a state transition graph made up of a set of states, a set of actions, and a succession relation associating states through actions (Bucci & Vivario, 1995). That is to say, it involves essentially the enumeration of all reachable markings or their coverable markings. This representation makes such properties as deadlock and reachability (Zhou, 1995) explicit, and allows the automatic verification of ordering relationships among task execution times (Tsai et al., 1995).

Although the reachability analysis method can be used for all nets, it is only limited to "small" nets due to the complexity of the state-space explosion. The same thing also happens in the analysis of TPN models. Sloan et al. (Sloan & Buy, 1996) developed several reduction rules for TPN analysis that work at an individual transition level. These reduction rules help to reduce the complexity of TPN analysis to some extent. However, it is not a trivial work to automatically search the preconditions of applying these reduction rules for a

Source: Petri Net,Theory and Applications, Book edited by: Vedran Kordic, ISBN 978-3-902613-12-7, pp. 534, February 2008, I-Tech Education and Publishing, Vienna, Austria

complex TPN. Wang et al. proposed the compositional time Petri nets and the corresponding component-level reduction rules (Wang et al. 2000). Each of the reduction rules transforms a TPN component to a small one while maintaining the net's external observable timing properties. The application of these rules will dramatically reduce the size of a TPN. However, all of the methods or models only reduce the complexity after the model becomes complex. It can not avoid the complexity to the best of its ability according to the analysis requirements when it is modeled.

These years, the usefulness of the object-oriented concepts has been recognized, because it allows us to describe systems easily, intuitively and naturally. These years, the object-oriented formal methods such as object Petri nets (OPN) (Bastide, 1995), VDM++ (Harel & Gery , 1996), Object-Z (Schuman, 1997), etc are suggested. Among the studies, the research on OPN has been focused on the extending Petri net formalism to OPN such as HOONet (Hong & Bae, 2000), OBJSA (Battiston et al. 1988), COOPN/2 (Biberstein & Buchs, 1994) and LOOPN++ (Lakos & Keen, 1994), which are suggested on the base of colored Petri Net (CPN) (Jensen, 1992). Object-oriented Petri net (OPN) can model different systems easily, intuitively and naturally. Abstraction is one of OPN characters compared with basic Petri nets. OPN can model various systems hierarchically and the models can be analyzed even if they have not been completed. So the complexity of OPN models can be simplified at the beginning of modeling stage according to the analysis requirements. Although the results of such studies have shown promise, these nets do not fully support time critical (time dependent) system modeling and analysis, which may be complex, midsize or even small. When time critical systems with any sizes are modeled, it requires formal modeling and analysis method to support temporal description and object-oriented concepts. That is to say, TPN and OPN need to be combined.

Firstly, this chapter formally proposes a high-level Petri net called timed hierarchical object-oriented Petri net (TOPN) (Xu & Jia, 2006) (Xu & Jia, 2005-2), which supports not only temporal description but also OO concepts. On one hand, TOPN has extended a model of Object-Oriented Petri Nets to allow modeling and analyzing complex time critical systems. Modeling features in TOPN support abstracting complex systems, so the corresponding models can be simplified effectively. In the proposed TOPN, a duration is also attached to each object accounting for the minimal and maximal amount of time between which that the behavior of the object can be completed once fired. On the other hand, this chapter also addresses the problem of the state analysis of TOPN models, what makes it possible to judge the model consistency at a given moment of time. On the base of Yao's extended state graph (ESG) (Yao, 1994), TOPN extended state graph (TESG) is presented for incremental reachability analysis for temporal behavior analysis. In particular, a new way is investigated to represent and deal with the objects with temporal knowledge.

Secondly, in order to extend a model of TOPN to allow modeling and analyzing dynamic systems with timing effect on system information, fuzzy concept is introduced into TOPN and fuzzy timed object-oriented Petri net (FTOPN) (Xu & Jia, 2005-1) is proposed. Temporal fuzzy sets are attached to each transition objects in TOPN accounting for the aging of information. In particular, a new way is investigated to represent and deal with timing effect in dynamic systems. FTOPN also supports learning similar to that in fuzzy timed Petri net (Pedryz & Camargo, 2007). FTOPN is also used to model a real decision making procedure of one 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.

Finally, in order to model CMRS, a CMRS modeling method called fuzzy timed agent based Petri nets (FTAPN) (Xu & Jia, 2007) is proposed on the base of FTOPN, because it can be regarded as a kind of multi-agent system (MAS) and the agent is also a special kind of object. FTAPN can be used to model and illustrate both the structural and dynamic aspects of CMRS. Supervised learning is supported in FTAPN. As a special type of high-level object, agent is introduced, which is used as a common modeling object in FTAPN models. The proposed FTAPN can not only be used to model CMRS and represent system aging effect, but also be refined into the object-oriented implementation easily. At the same time, it can also be regarded as a conceptual and practical artificial intelligence (AI) tool for multi-agent system (MAS) into the mainstream practice of software development.

This chapter has just been arranged as the following. Section 1 makes a quick review of the relative study of Petri Nets. In section 2 of this chapter, it justifies the need for defining TOPN through interpreting how to combine the time restricting information with HOONet. An informal and intuitive behavior semantics of TOPN has been introduced in section 3. Then, in section 4, the constructing algorithm of reachability tree is presented, which can support most of the property analysis of TOPN. In section 5, FTOPN is proposed on the base of TOPN, and FTOPN has been used to model and analyze the decision procedure of one CMRS. Then, FTAPN is presented on the base of FTOPN and it is used to model and analyze one CMRS to demonstrate its effectiveness in section 6. Section 7 concludes the work in this chapter and suggests further research issues in the future.

2. The basic concepts of TOPN

In this section, some important basic concepts of Petri nets are firstly reviewed. Then the definitions of TOPN are presented. At the same time, the enabling rules and the firing rules of TOPN are presented.

2.1 A brief review of basic Petri nets

In this subsection, we will quickly review some key definitions. A more general discussion on Petri nets can be found in Peterson's book (Peterson, 1991) and in the excellent survey article by Murata (Murata, 1989).

A Petri net is a five-tuple PN= (P, T, F, W, M_0) where P and T are the node sets and F is the edge set of a directed bipartite graph, and M_0: P→N is called the initial marking (or initial state) of PN. (We use N to denote the set {0, 1, 2…}.) We call P the set of places of PN and T the set of transitions of PN. In diagrams, we will show places as circles and transitions as bars. Formally, $F \subseteq (P \times T) \cup (T \times P)$ and F is called the flow relation (or edges) of PN. W: F → {1, 2, 3…} and it is called the weight of a flow. In general, a marking of PN associates a nonnegative integer number of markers or tokens with each place.

For net PN= (P, T, F, W, M_0), we use the following symbols and notations for the sets of predecessors and successors of a place p∈P and transition t∈T.

- •t={p | (p, t)∈F} =the set of input places of t,
- t•={p | (t, p) ∈F}=the set of output places of t,

- •p={t| (t, p)∈F}=the set of input transitions of p,
- p•= {t| (p, t) ∈F} =the set of output transitions of p.

A transition is enabled when all its input places have at least one token. When an enabled transition t is fired, a token is removed from each input place of t and a token is added to each output place; this gives a new marking. For net PN= (P, T, F, W, M_0), the language of PN, denoted as L (PN), is the set of all legal sequences ω∈T* of transition firings starting from marking M_0.

Petri net PN= (P, T, F, W, M_0) is safe if M_0 : P→{0, 1}, and if all markings reachable by legal sequences of transition firings from the initial marking have either zero or more tokens in every place.

2.2 High-level Petri nets

There are different definitions and terminology of TPN and OPN. In this chapter, our work is based on the Merlin's TPN and Hong's OPN which is called HOONet (Hong & Bae, 2000). A time Petri net is also a tuple *TPN*=(*PN*, *SI*). PN is a basic Petri net. And SI is a mapping called a static interval, SI: T→Q* ×(Q*∪∝), where Q* is a set of nonnegative rational numbers.

HOONet is a high-level Petri net supporting the representation scheme of object-oriented concepts. A HOONet model is represented as Petri-net form for an object and has components to represent a unique name, attributes and its behaviors (methods) of an object.

Definition 1: HOONet is defined with a tuple HOONet= (OIP, ION, DD), where

1. OIP (object identification place) is a special place which is defined as a tuple, OIP=(*oip*,*pid*,M_0,*status*), where
   - *oip* is a unique name of a HOONet model.
   - *pid* is a unique process identifier that distinguishes the multiple instances of an object.
   - M_0 is an initial marking function.
   - *status* is a flag variable (either pre value or post value) to represent the specific states of OIP.
2. ION (internal object net) is a variant of CPN (colored Petri nets) that represents the internal behaviors of an object, which is defined as a tuple ION=(*P*,*T*,*A*,*C*,*N*,*G*,*E*,*F*,M_0), where
   - *P, T* and *A* are finite sets of places, transitions and arcs respectively.
   - *C, N, G* and *E* mean the functions of a color set, a node, a guard and an arc expression, respectively. They are the same as defined in (Jensen, 1992).
   - *F* is a special arc from transitions to OIP, and depicted as those a rim of ION, and
   - M_0 is a function giving initial marking to specific places.
3. DD (data dictionary) contains the declarations of variable, token types, and functions per a HOONet model using standard CPN ML (Jensen, 1992). □

Definition 2: A set of place types in HOONet, P=(P_i,P_a), where

1. Primitive place P_i is a basic type of places that represent the local states of a system, the same as in general CPN (Jensen, 1992).

2. Abstract place P_a= (*pn, refine_state, action*) is a place type which represents abstract states, where
    - *pn* is the name of an abstract place.
    - *refine_state* is a flag variable denoting the refinement of an abstract place.
    - *action* is a static reaction that imitates the internal behaviors of an abstract place. □

Definition 3: A set of transition types in HOONet, T={T_i, T_a, T_c},where
1. Primitive transition T_i is a basic transition type in general CPN
2. Abstract transition T_a = (tn, refine_state, action), where
    - tn is the name of an abstract transition.
    - refine_sate and action have the same meanings as in the definition of the abstract place.
3. Communicative transition T_c =(tn, target, ctype, action) is a transition type that represents calling a method, where
    - tn is the name of a communicative transition.
    - target is a flag variable denoting whether the method called from Tc, is modeled (a "yes" value) or not (a "no" value).
    - ctype is also a flag variable denoting whether the interaction of Tc is synchronous (a "SYNC" value) or asynshronous (an "ASYN" value).
    - action is the static reaction that reflects the execution results of the called method.

The variable *ctype* with its "SYNC" value denotes that the caller waits for the result from the called method. With "ASYN" value, the token is duplicated. Each of the duplicated tokens is transferred to the called object and the next place in its net, respectively.

2.3 Timed hierarchical object-oriented Petri net

The purpose of designing timed hierarchical object-oriented Petri net (TOPN) is to aid in the modeling and analysis of real time systems and bridge the gap between the formal treatment of object-oriented Petri nets and temporal reduction approach for the modeling, analysis, and prototyping of complex time critical systems.

A TOPN model is a variant HOONet representation that corresponds to the class with temporal property in object-oriented paradigm. Like the HOONet, TOPN is composed of four parts: 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.

Definition 4: TOPN is a four-tuple: TOPN= P (OIP, ION, DD, SI), where:
1. OIP=(oip, pid, M_0, status), oip, pid, M_0 and status are the same as those in HOONet.
    - oip is a variable for the unique name of a TOPN.
    - 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 TOPN 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 TOPN.
3. DD formally defines the variables, token types and functions (methods) just like those in HOONet (Hong & Bae, 2000).
4. SI is a static time interval binding function, SI: {OIP}→Q*, where Q* is a set of time intervals.□

According to the *definition 4*, the general structure of TOPN is shown in Fig.1. In time critical systems, time relates to events. While in Petri net, events occur and originate from system behaviors. And system behaviors stem from the behavior properties of objects in TOPN. These objects include transitions, abstract places and other TOPN objects. So not only transitions, but also all TOPN objects including abstract places, etc need to be restricted by time condition.

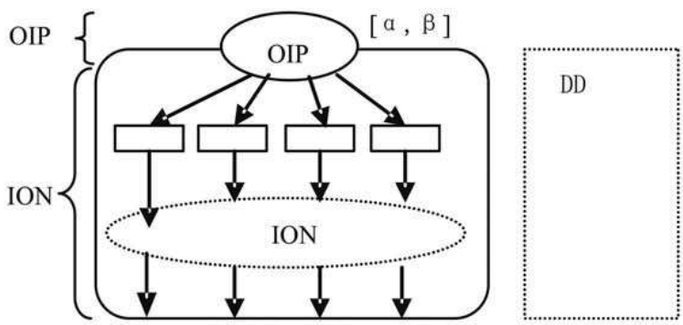


Fig. 1. The General Structure of TOPN

An event in a time critical system can be thought of as an interval [s, t] on the time line where s is its starting endpoint and t is its terminating endpoint, having a duration given by t-s≥0. The special case of time interval where t=s is a *point event*. Otherwise, it is an *interval event*. In the corresponding time interval [s, t] of event firing, s is the earliest firing time (EFT) and t is the latest firing time (LFT). In the changes of TOPN behavior, events are regarded as interval events. The temporal knowledge in TOPN is represented as time intervals.

Similar to HOONet, TOPN is also a kind of hierarchical net. In TOPN, the whole TOPN model is also an object, and it is always regarded as an abstract place object. Its realizing details are depicted in ION. Inside the ION, abstract objects may also be included. The realizing details of these objects can also be depicted as a TOPN. The definition of ION is just like the following.

Definition 5: 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 (Hong & Bae, 2000). □

Similar to common OPNs, basic OPN components and additional restricting conditions are included in the detailed ION structure. The basic OPN components may include common components (transition and place) and abstract components. If the model needs to be analyzed in details, the abstract components in ION should be refined. At the same time, the ION is unfolded. The following definitions of abstract components in TOPN are the base of refining abstract component. The abstract components in TOPN include timed abstract transitions, timed abstract communication transitions and timed abstract places.

Definition 6: A set of places in TOPN is defined as P=PIP∪TABP, where

1. PIP is the set of primitive places similar to those in PNs (Murata, 1989) (Peterson, 1991).
2. Timed abstract place (TABP) is a four-tuple: TABP= TABP(pn_{TABP}, refine $state_{TABP}$, $action_{TABP}$, SI_{TABP}), where
   - pn_{TABP} is the identifier of the abstract timed place.
   - refine $state_{TABP}$ is a flag variable denoting whether this abstract place has been refined or not.
   - $action_{TABP}$ is the static reaction imitating the internal behavior of this abstract place.
   - SI_{TABP} is also a static time interval binding function from a set of TABPs to a set of static time intervals. □

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) described in Fig.2. 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 in TOPN also need to be associated with time intervals.

Generally, the abstract place—TABP is always represented as a kind of abstract form in higher layers in TOPN models. At this time, "refine state" dedicates that it is in abstract form. However, if "refine state" denotes that it is in the refined state, TABP will have been refined into the corresponding TOPN which is defined in lower layers.

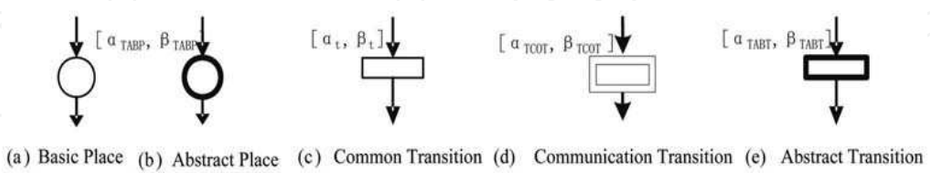


Fig. 2. Places and Transitions in TOPN

Definition 7: A set of transitions in TOPN can be defined as T= TPIT∪TABT∪TCOT, where

1. Timed primitive transition TPIT = TPIT (BAT, SI_{TPIT}), where
   - BAT is the set of common transitions.

- SI_{TPIT} is a static time interval binding function, SI: {TPIT}→Q*, where Q* is a set of time intervals.

4. Timed abstract transition TABT= TABT (tn_{TABT}, refine $state_{TABT}$, $action_{TABT}$, SI_{TABT}), where
   - tn_{TABT} is the name of this TABT.
   - refine $state_{TABT}$ is a flag variable denotes whether this TABT has been refined or not.
   - $action_{TABT}$ is the static reaction imitating the internal behavior of the TABT.
   - SI_{TABT} is a static time interval binding function, SI: {TABT}→Q*, where Q* is a set of time intervals.
5. Timed communication transition TCOT=TCOT (Tn_{TCOT}, $target_{TCOT}$, comm $type_{TCOT}$, $action_{TCOT}$, SI_{TCOT}).
   - Tn_{TCOT} is the name of TCOT.
   - $target_{TCOT}$ is a flag variable denoting whether the behavior of this TCOT has been modeled or not. If $target_{TCOT}$ ="Yes", it has been modeled. Otherwise, if $target_{TCOT}$ ="No", it has not been modeled yet.
   - comm $type_{TCOT}$ is a flag variable denoting the communication type. If comm $type_{TCOT}$ ="SYNC", then the communication transition is synchronous one. Otherwise, if comm $type_{TCOT}$ ="ASYN", it is an asynchronous communication transition.
   - $action_{TCOT}$ is the static reaction imitating the internal behavior of this TCOT.
   - SI_{TCOT} is a static time interval binding function, SI: {TCOT}→Q*, where Q* is a set of time intervals. □

Just like those in HOONet, there are three kinds of transitions in TOPN. The timed primitive transition (represented as rectangles with thin prim), timed abstract transition (represented as rectangles with bold prim) and timed communication transition (represented as rectangles with double thin prim). They are depicted in Fig.2. Different transitions represent different system behaviors. So, temporal intervals are associated with all of these transitions in TOPN. Abstract transitions are also TOPN objects. They can be refined in lower layers.

The definition of abstract transitions mentioned above is also a kind of abstract form in higher layers of TOPN models, when "refine state" indicates it is in the abstract form. While, if "refine state" denotes it is in refined state, the corresponding abstract transition should be refined into the corresponding TOPN which is defined in lower layers.

From the definitions mentioned above, TOPN are hierarchical just like the structure of object models. In the higher levels of the model, its components may be in abstract form and the model is simple. In the unfolded model where the abstract components are refined, the TOPN model may be complex, but the realizing details are clear. So, according to the analysis requirements, users can analyze the TOPN models in different layers, even if the detailed realization in lower layers have not been completed yet.

3. Behavior semantics of TOPN

3.1 Execution paths

State changes relate to the events in TOPN. However, events may stem from transition firing or TABP behaviors. The state changes in TOPN relate to the schedule and the associated

temporal interval tightly. In order to analyze the dynamics of TOPN, the definition of schedule and path is given in the following.

Definition 8: In Petri net N, if the state Mn is reachable from the initial state M_0, then there exists a sequence of fired transitions from M_0 to Mn. This sequence is called a path or a schedule ω from M_0 to Mn. It can be represented as:

$$\text{Path} = \{M_0,t_1,M_1,\ldots,t_n,M_n\} \text{ or } \omega = \{M_0,t_1,M_1,\ldots,t_n,M_n\}$$

$$t_i \in N.T;\ 1 \le i \le n$$

And the schedule set of Petri net N with initial marking M_0 is represented as $L(N,M_0)$. □

Just like those in TPN (Merlin & Farber, 1976) (Harel & Gery, 1996), if the number of solid tokens residing in the input place equals or exceeds the weight of the input arc, the forward transition is enabled. However, when one TABP is marked by enough hollow tokens compared with the weight of internal arcs in its refined TOPN, it is also enabled at this time. After its internal behaviors have completed, the color of tokens residing in it become from hollow to solid, which are similar to those in common places. So TABPs also manifest actions in TOPN. An extended definition of path in TOPN is given in the following, in which TABP is extended into the schedule.

Definition 9: If the state Mn is reachable from the initial state M_0, then there exists a sequence of marked abstract places and fired transitions from M_0 to Mn. This sequence is called a path or a schedule ω from M_0 to Mn. It can be represented as:

$$\text{Path} = \{PA_1, PA_2, \ldots, PA_n\} \text{ or } \omega = \{PA_1, PA_2, \ldots, PA_n\}$$

where $PA_i \in T \cup TABP$ and $1 \le i \le n$.

Definition 10: Let t be a TOPN transition and let $\{PA_1, PA_2, \ldots, PA_n\}$ be a path, add t_i into the path is expressed as $\{PA_1, PA_2, \ldots, PA_n\} + t = \{PA_1, PA_2, \ldots, PA_n, t\}$.

Let p be an abstract place and let $\{PA_1, PA_2, \ldots, PA_n\}$ be a path, add p into the path is expressed as $\{PA_1, PA_2, \ldots, PA_n\} + p = \{PA_1, PA_2, \ldots, PA_n, p\}$, where $PA_i \in T \cup TABP$ and $1 \le i \le n$.

Definition *11*: For a TOPN N with schedule ω, we denote the state reached by starting in N's initial state and firing each transition in ω at its associated time φ(N,ω). The time of φ(N,ω) is the global firing time of the last transition in ω.

When the relative time belongs to the time interval attached to the transition or the TABP and the corresponding object is also enabled, then it can be fired. If a transition has been fired, the marking may change like that in PN (Wang, 1998). If a TABP is fired, then the hollow token(s) change into solid token(s), and the tokens still reside in the primary place. At this time, the new relative time intervals of every object are calculated like those in (Harel & Gery, 1996).

3.2 Enabling rules and firing rules

State changes in TOPN stem from the behavior executions in TOPN. The execution of a TOPN depends on two main factors. Firstly, it is the number and distribution of tokens in

the TOPN. Tokens reside in the places and control the execution of the transition. Secondly, its execution depends on the definition of execution time represented as time intervals. A TOPN executes by firing transitions.

The dynamic behavior can be studied by analyzing the distribution of tokens (markings) in TOPN. So the enabling rule and firing rule of a transition in TOPN are introduced in the following, which govern the flow of tokens.

Enabling Rule:

1. A transition t in TOPN is said to be enabled 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:
   $M(p) \geq I(t, p)$ for any p in P, the same as in ordinary Petri nets, where M(p) is the marking of the place p and I(t, p) is the weight of the input arc from the place p to the transition t.
2. If the place is TABP, it will be marked with a hollow token and TABP is enabled. At this time, the ION of the TABP is enabled. After the ION is executed, the tokens in TABP are changed into solid ones. □

Firing Rule:

1. For a transition:
   a. An enabled transition in TOPN may or may not fire depending on the additional interpretation (Merlin & Farber, 1976) (Bucci & Vivario, 1995) (Harel & Gery, 1996), and
   b. The relative time θ, relative to the absolute enabling time τ, is not smaller than the earliest firing time (EFT) of transition t_i, and not greater than the smallest of the latest firing time (LFT) of all the transitions enabled by marking M (Hong & Bae, 2000):
      $$\text{EFT of } t_i \leq \theta \leq \min(\text{LFT of } t_k)$$
      where k ranges over the set of transitions enabled by M, the same as (Hong & Bae, 2000).
   c. After a transition t_i (common one or abstract one) in TOPN is fired at a time θ, TOPN changes to a new state. The new states can be computed as the following:

- The new marking M′ (token distributions) can be computed as the following:
  If the output place of t_i is TABP,
  then M′(p)= attach (*, (M(p)-I(t_i,p)+O(t_i,p)));
  else M′(p)=M(p)-I(t_i,p)+O(t_i,p);
  The symbol "*" attached to the markings of TABP represents as hollow tokens in TABP.
- The computation of the new firing interval I′ is the same as those in (Harel & Gery, 1996), (Yao, 1994), as
  $$I' = (\max(0, EFT_k - \theta_k), (LFT_k - \theta_k))$$
  where EFT_k and LFT_k represents the lower and upper bound of interval in I corresponding to t_k in TOPN, respectively.
- The new path can be computed as path′ = path + t_i .

2. For a TABP
   a. The relative time θ should satisfy the following conditions:
      $$\text{EFT of } t_i \leq \theta \leq \min(\text{LFT of } t_k)$$

where t_k belongs to the set of the places and transitions which have been enabled by M.

b. After a TABP p in TOPN is executed at a time θ, TOPN states change. The new marking can be computed as the following.

- The new markings are changed for the corresponding TABP p, as
  M′(p)= remove_attach (*, M(p))
  The symbol "*" is removed from the marking of TABP. Then the marking is the same as those of common places. The change represents that the internal actions of TABP have been finished. Tokens of TABP have been changed into solid ones.
  To compute the new time intervals is the same as that mentioned above.
- The new path can be decided by path′ = path + p. □

When the number of tokens satisfies the conditions of enabling rule, the corresponding transitions or TABPs are enabled. Only if the corresponding objects are enabled and the relative time is in the time interval, can the objects be fired. The relative firing time may be stochastic, but it is after EFT and before LFT. In TOPN, the firing procedures are considered to be instantaneous and their execution delay can be considered in the time interval of execution conditions.

4. Reachability analysis

4.1 Analysis algorithm

The purpose of TOPN is to aid in modeling and analysis of complex time critical systems. From the point of TOPN definition, TOPN can describe the temporal constraints in time critical systems. Then the model analysis method especially reachability analysis, need to be discussed. In order to analyze TPN (Yao, 1994) models, Yao has presented extended state graph (ESG) to analyze TPN models. On the base of ESG, an extended TOPN state graph has been presented in this section, into which temporal reasoning has also been introduced.

In a TOPN model, an extended state representation "ES" is 3-tuple, where ES=(M, I, path) consisting 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 state. 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 formula of I′.

Definition 12: A TOPN extended state graph (TESG) is a directed graph. In TESG, nodes represent TOPN model states. In TESG, there is an initial node, which represents the TOPN model initial state. Arcs denote the events, which make model state change. There are two kinds of arcs from one state ES to another one ES′ in TESG.

1. The state change from ES to ES′ stems from the firing of the transition t_i. Correspondingly, there is a directed arc from ES to ES′, which is marked by t_i.
2. If the internal behavior of the TABP—"p_i" makes the TOPN model state change from ES to ES′, then in TESG there is also a directed arc from ES to ES′. It is marked by p_i. □

On the base of Petri net analysis method (PN and TPN) and the definition of TESG, the TESG of one TOPN model can be constructed by the following step:

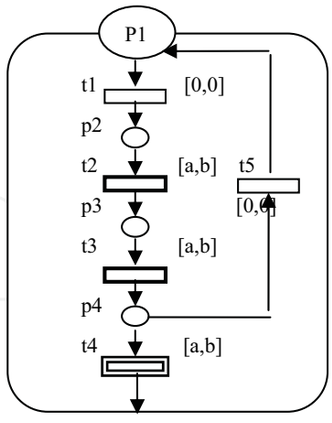
Step 1) Use the initial state ES_1 as the beginning node of TESG, where $ES_1=(M_0, [0,0], \Phi)$.

Step 2) Mark the initial state "New".

Step 3) While (there exist nodes marked with "new") do
Step 3.1) Choose a state marked with "new".
Step 3.2) According to the enabling rule, find the enabled TOPN objects at the current state and mark them "enabled".
Step 3.3) While (there exist objects marked with "enabled") do
Step 3.3.1) Choose an object marked with "enabled".
Step 3.3.2) Fire this object and get the new state ES_2.
Step 3.3.3) Mark the fired object "fired" and mark the new state ES_2 "new".
Step 3.3.4) Draw a directed arc from the current state ES_1 to the new state ES_2 and mark the arc with the name of the fired object and relative firing temporal constraint.
// The internal "while" cycle ends.
Step 3.4) Mark the state ES_1 with "old".
// The external "while" cycle ends.

TESG describes state changes in TOPN models. In TESG, not only state changing sequence, but also dynamic temporal constraints and execution paths related to state changes have all been described in TESG. TESG constructing procedure is also a TOPN model reachability analysis procedure. So if the TESG of one TOPN model has been depicted, the corresponding reachability has also been analyzed.
Similar to the state analysis in TPN, when the TESG of one TOPN model has been completed, the TPN consistency determination theorem can be used to judge the consistency of TOPN models. So the consistency of time critical system can be checked. The theorem can be referenced to Yao's paper (Yao, 1994).

4.2 A modeling and analysis example




```
Var +CT = boolean; /* Transferring Tag */
/*CT is set to "T" in the transition--
"DataFusion" */
Var +Time=Integer; /* Current Relative
Time*/
TT C = with hollow | solid;
TCOT (ComTransf)={
      Fun(CT= =F □ (a≤Time≤b) ):
        ComTransf () □ CT=F □ Mark(p1,C);
      };
/* Mark(P,C): Mark the place P with C。 */
TABT (StateCol)={
      Fun(CT= = F □ (a≤Time≤b)):
        OwnStateCol()□ M(p5,C);
  };
/* M(P,C): Mark the place P with C。 */
Mark(Place,C)={
  Fun(Place is a TABP □ (αp≤Time≤βp)):
      OIP(Place) □ M(Place,C);
  Fun(Place is not in N.TABP): M(Place,C);
};/*mark different places*/
```

Fig. 3. The TOPN Model

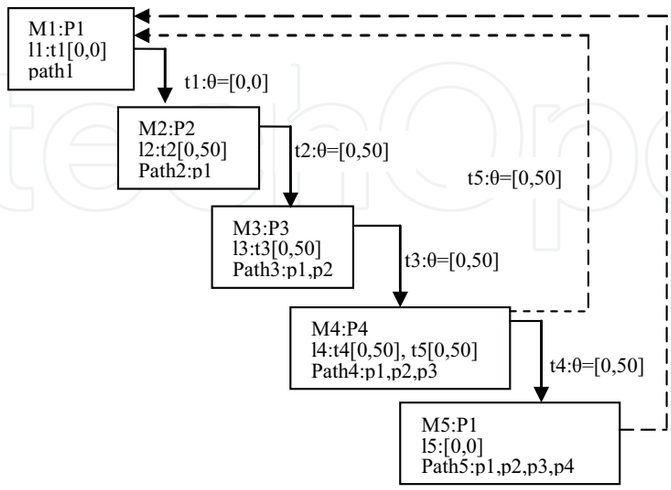


Fig. 4. The TESG of the Decision Model

In distributed cooperative multiple robot systems (CMRS), every robot makes control and schedule decisions according to different system information such as other robot states, its own states and task assignment. The decision making procedure can be divided into 3 main phases. In the first phase, the decision making module collects the above information. For the information mentioned above, every kind of information may include different detailed information. For example, velocity, movement direction and location need to be considered in its and other robot's states. The task to be completed in the future is considered in the task assignment. As the information may not be available from all sensors or sources at the same time moment, the temporal constraint about the information collection is needed. This collection procedure should be completed in 50 unit time. In the second phase, information fusion based method is used to make control and schedule decisions of every robot. To complete the information fusion aim, every kind of information is required simultaneously. It may last for about 50 unit time. Finally, the decision results are transformed to other system modules. The transferring procedure will last for about 50 unit times. In this control procedure, the decision conditions and temporal constraints need to be considered simultaneously, so TOPN is chosen to model this decision making module. Fig.3 has shown the TOPN model of CMRS decision model and its data dictionary respectively. Then Fig.4 has given the state analysis by means of TESG. From the TESG, the design logical errors can be excluded. According to the Yao's consistency judging theorem and the TESG, the TOPN model in Fig.3 is consistent.

5. Fuzzy timed object-oriented Petri net

Although Petri nets can be used to model and analyze different systems, they fail to model the timing effects in dynamic systems. Fuzzy timed Petri net (FTPN) (Pedrycz & Camargo, 2003) 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 with the complexity problem, when it is used to model complex dynamic systems. In this section, 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.

5.1 Basic Concept

Similar to FTPN (Pedrycz & Camargo, 2003), fuzzy set concepts are introduced into TOPN (Xu & Jia, 2005-2) (Xu & Jia, 2006). Then FTOPN is proposed, which can describe fuzzy timing effect in dynamic systems.

Definition 13: 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 (Hong & Bae, 2000) and TOPN (Xu & Jia, 2006).
   - 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 (Hong & Bae, 2000) and TOPN (Xu & Jia, 2006).
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 13: 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 conditions 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 (Hong & Bae, 2000) and TOPN (Xu & Jia, 2006). □

Definition 14: 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 (Murata, 1989) (Peterson, 1991).
2. Timed abstract place (TABP) is a six-tuple: TABP= TABP(p_n, refine state, action, SI, R, I), where
   - p_n 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 15: 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 a 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 (Pedrycz & Camargo, 2003), 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:

$$z(v) = (\overset{n}{\underset{i=1}{T}}(r_i \rightarrow x_i(v'))\, sw_i)\, t\mathrm{I}(v) \tag{1}$$

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 (→) 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

$$x_i(v)=x_i(v')t(1-z(v)) \tag{2}$$

(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:

$$y(v)=y(v')sz(v) \tag{3}$$

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.

5.2 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(Pedrycz & Camargo, 2003). 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 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 Q under discussion assumes the form of the following sum:

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

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:

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

where γ is a learning rate and $\nabla_{\mathbf{param}}Q$ 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 $T_i(k)$ complies with the following form. In what follows, the temporal decay is modeled by an exponential function,

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

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

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

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

$$y(k) = y(k\text{-}1)sz,\ x_i(k) = x_i(k\text{-}1)t(1\text{-}z) \tag{9}$$

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:

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

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

5.3 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.

5.3.1 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 Fig.5.

In the model of Fig.5, 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 Fig.6. The information "other robot state" and "own state" complies with the rule in Fig.6 (1). The other information complies with Fig.6 (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 includes robot schedule and control commands.

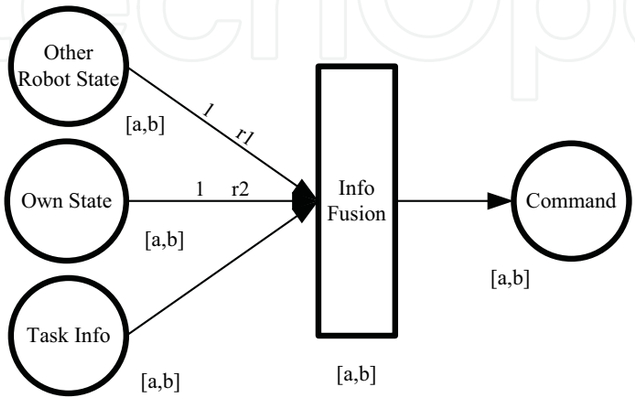


Fig. 5. The FTOPN Model

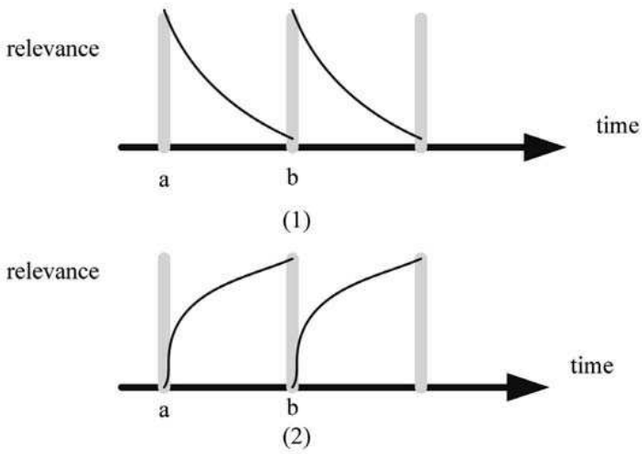


Fig. 6. The Relevance

What's more, all the objects in Fig.5 can also be depicted in details by FTOPN. For example, the object—"Other Robot State" in Fig.5 can also be modeled concretely with FTOPN. The detailed model of the object is depicted in Fig.7. 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 Fig.5. 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 Fig.5.

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

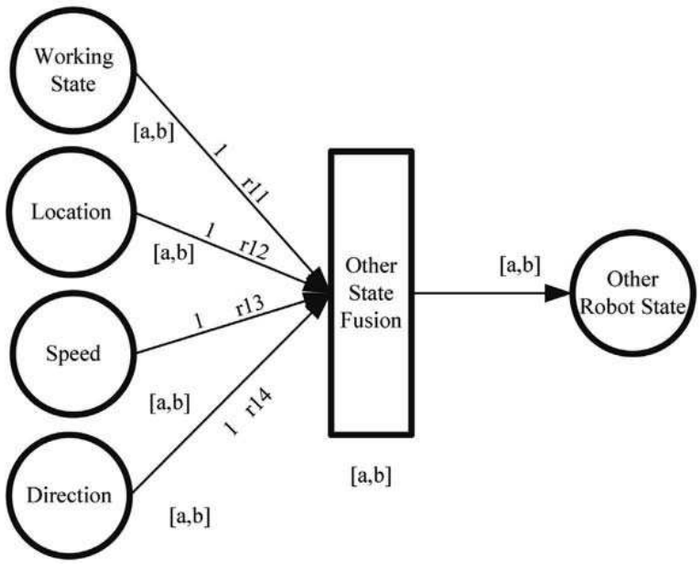


Fig. 7. The Object-"Other Robot State" Model

5.3.2 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 the abstraction concepts in FTOPN. And the fuzzy reduction procedures have been simplified. Essentially, hierarchical modeling idea in FTOPN is to the 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 (Pedrycz, 1999). Table.1 summarizes the main features of the fuzzy timed Object-oriented Petri nets and contrasts 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.

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 are 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.

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

6. Fuzzy timed agent based Petri net

As a typical multi-agent system (MAS) in distributed artificial intelligence (Jennings et al., 1998), when CMRS is modeled, some difficulties are met with. For modeling this kind of MAS, object-oriented methodology has been tried and some typical agent objects have been proposed, such as *active object*, etc (Guessoum & Briot, 1999). However, agent based object models still can not depict its structure and dynamic aspects, such as cooperation, learning, temporal constraints, etc(Jennings et al., 1998). This section proposes a high level PN called fuzzy timed agent based Petri net (FTAPN) on the base of FTOPN (Xu & Jia, 2005-1)

6.1 Agent object and FTAPN

The *active object* concept (Guessoum & Briot, 1999) has been proposed to describe a set of entities that cooperate and communicate through message passing. To facilitate implementing *active object* systems, several frameworks have been proposed. ACTALK is one of the typical examples. ACTALK is a framework for implementing and computing various *active object* models into one object-oriented language realization. ACTALK implements asynchronism, a basic principle of *active object* languages, by queuing the received messages into a mailbox, thus dissociating message reception from interpretation. In ACTALK, an *active object* is composed of three component classes: *address*, *activity* and *activeObject* (Guessoum & Briot, 1999).

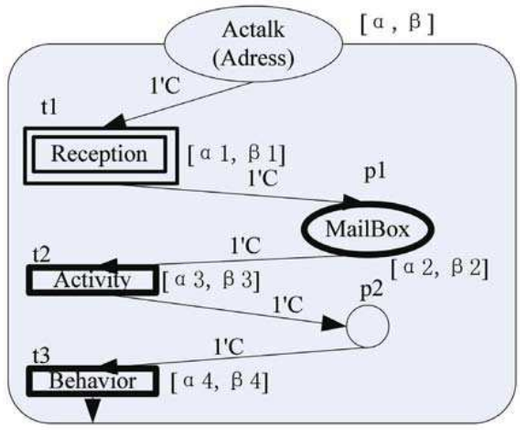


Fig. 8. The FTOPN Model of ACTALK

ACTALK model is the base of constructing *active object* models. However, *active object* model is the base of constructing multi-agent system model or agent system model. So, as the modeling basis, ACTALK has been extended to different kinds of high-level agent models. Because of this, ACTALK is modeled in Fig.8 by FTOPN.

In Fig.8, OIP is the describer of the ACTALK model and also represents as the communication address. One communication transition is used to represent as the behavior of message reception. According to the communication requirements, it may be synchronous or asynchronous. If the message has been received, it will be stored in the corresponding mail box, which is one first in and first out queue. If the message has been received, the next transition will be enabled immediately. So mail box is modeled as abstract place object in FTAPN. If there are messages in the mail box, the following transition will be enabled and fired. After the following responding *activity* completes, some *active behavior* will be conducted according to the message.

Fig.8 has described the ACTALK model based on FTOPN on the macroscopical level. The detailed definition or realization of the object "*Activity*" and "*Behavior*" can be defined by FTOPN in its parent objects in the lower level. The FTOPN model of ACTALK can be used as the basic agent object to model agent based systems. That is to say, if the agent based model—ACTALK model is used in the usual FTOPN modeling procedure, FTOPN has been

extended to *agent based modeling methodology*. So it is called *fuzzy timed agent based Petri net* (*FTAPN*).

6.2 Learning in FTAPN

The parameters of FTAPN are always given beforehand. In general, however, these parameters may not be available and need to be estimated just like those in FTPN (Pedrycz & Camargo, 2003). 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 out 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 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 FTAPN, the performance index Q under discussion assumes the following form.

$$Q=\sum_{k=1}^{K}(t\arg et(k)-y(k))^2 \tag{11}$$

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 in the following equation being applied to the parameters:

$$\mathbf{param}(\text{iter}+1)=\mathbf{param}(\text{iter})-\gamma\nabla_{\mathbf{param}}Q \tag{12}$$

where γ is a learning rate and $\nabla_{\mathbf{param}}Q$ 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 equation:

$$\tilde{x_i} = x_i(0)T_i(k) \tag{13}$$

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

$$T_i(k)=\begin{cases}\exp(-\alpha_i(k-k_i)) & if \quad k>k_i,\\ 0 & others\end{cases} \tag{14}$$

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

$$z=(\overset{n}{T}_{i=1}\quad((r_i\rightarrow\tilde{x_i})sw_i)) \tag{15}$$

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

$$y(k) = y(k-1) \mathbf{s} z,\ x_i(k) = x_i(k-1) \mathbf{t} (1-z) \tag{16}$$

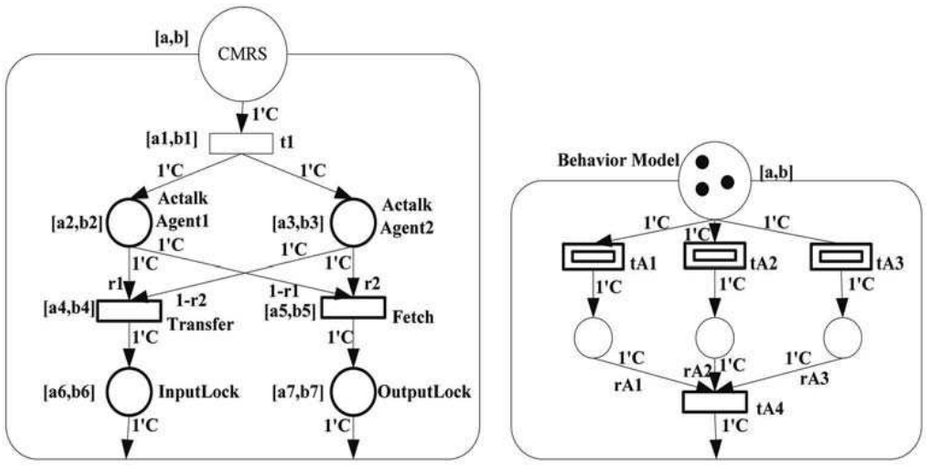
We assume that the initial marking of the output place y(0) is equal to zero, y(0)=0. The derivatives of the weights w_i are computed as the form in the following equation:

$$\frac{\partial}{\partial w_i}(t \arg et(k) - y(k))^2 = -2(t \arg et(k) - y(k)\frac{\partial y(k)}{\partial w_i}) \tag{17}$$

where i=1,2,…, n. Note that y(k+1)=y(k)$\mathbf{s}$z(k).

6.3 A Modeling example

In manufacturing integrated circuits, usually there is a Brooks Marathon Express (MX) CMRS platform made up of two transferring robots. These two cooperative robots are up to complete transferring one unprocessed wafer from the input lock to the chamber and fetch the processed wafer to the output lock. Any robot can be used to complete the transferring task at any time. If one robot is up to transfer one new wafer, the other will conduct the other fetching task. They will not conflict with each other. Fig. 9 depicts this CMRS FTAPN model, where two agent objects (ACTALK) is used to represent these two cooperative robots.



(a) The Agent Based FTAPN Model (b) The Behavior Model in Every Agent

Fig. 9. The FTAPN Model

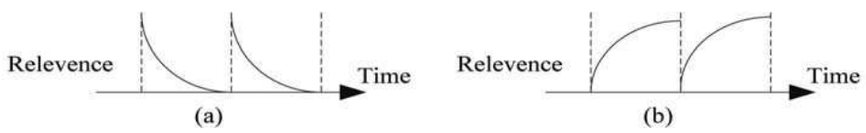


Fig. 10. The Relevance

Fig. 9 (a) has depicted the whole FTAPN model. The agent object—"ACTALK" is used to represent every robot model. Different thresholds are used to represent the firing level of the behavior conducted by the corresponding robot (agent). They also satisfy the unitary requirements and change according to the fuzzy decision in the behavior of every agent in Fig. 9 (b). In the model of Fig. 9 (b), three communication transition objects are used to represent the behavior for getting different kinds of system states. These states include the state of the other robot, its own goal and its current state, which can be required by the conductions of the communication transitions t_{A1}, t_{A2} and t_{A3}. When one condition has been got, the following place will be marked. In order to make control decisions (transition object t_{A4}) in time, all of these state parameters are required in the prescriptive time interval. However, the parameters of the arrival times comply with the rule in Fig. 10 (a). The other two kinds of information comply with that in Fig. 10 (b). After the decision, a new decision command with the conduction probability will be sent in this relative interval and it also affects which behavior will be conducted by updating the threshold in Fig. 10 (a).

6.4 Application aspects of FTAPN

Owing to the nature of the facet of temporal knowledge, fuzzy sets and object-oriented concepts in this extension of PN, they become viable models in a wide range of engineering problem augmenting the already existing high level Petri nets, cf. (Hong & Bae, 2000) (Wang, 1998). Two main and commonly categories of models are worth elaborating here.

6.4.1 Models of multi-agent systems

The multi-agent paradigm and subsequently a variety of models are omnipresent in a number of areas. In a nutshell, in spite of the existing variety of improved models, it still lacks a powerful modeling method, which can bridge the gap between model and practical implementations. Petri nets come with objects, temporal knowledge and fuzzy sets can use active objects to model generic agents with situatedness, autonomy and flexible. This helps us to use the object to reduce the complexity of MAS systems and the dynamic learning and decision to support the autonomy of agents.

6.4.2 Models of complex real-time systems

In models of complex real-time systems as usually encountered in industrial practice, the scale of the system module may be too complicated to be analyzed and the readings of different system state or sensors may be available at different time. The former may lead to the state explosion, while the latter needs adjustment of relevance of the information gathered at different time scales. The object models with temporal information degradation (aging) help to abstract complicated model and quantify the confidence of the inferred results.

7. Conclusion

Firstly, a high-level Petri net called timed hierarchical object-oriented Petri net (TOPN) is studied deeply in this chapter.

For modeling complex time critical systems and analyzing states, TOPN is proposed firstly. The work is based on the following work: Hong's hierarchical object-oriented Petri net (HOONet) (Hong & Bae, 2000), Marlin's timed Petri net (Merlin & Farber, 1976)) and Yao's extended state graph (Yao, 1994).With the introduction of temporal knowledge in TOPN, the temporal constraints need to be considered in state analysis. A state analysis method—"TOPN extended state graph (TESG)" for TOPN has also been presented in this chapter. Not only state analysis, but also consistency can be analyzed by means of TESG. On the other hand, TOPN can model complex time critical systems hierarchically. So analysis of properties and state change becomes much easier. A decision making example modeled by TOPN has been used to illustrate the usefulness of TOPN.

In the future research of TOPN, temporal reasoning and TOPN reduction rules will be studied, which can be used to refine and abstract TOPN models with preserving timing property.

Secondly, fuzzy timed object-oriented Petri net (FTOPN) is presented on the base of TOPN.

Timing effect is also 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 (Xu & Jia, 2006) and FTPN (Pedrycz & Camargo, 2003). 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 be trained and reduced 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 which can analyze the FTOPN and FTAPN models, needs to be studied in the future research. With the temporal fuzzy sets introduced into TOPN, the certainty factor about object firing (state changing) needs to be considered in the state analysis.

Finally, agent concepts are introduced into FTOPN and fuzzy timed agent based Petri net (FTAPN) is proposed in this chapter.

Cooperative multi robot system is a kind of usual manufacturing equipments in manufacturing industries. In order to model, analyze and simulate this kind of systems, this paper proposes fuzzy timed agent based Petri net (FTAPN) on the base of FTOPN (Xu & Jia, 2005-1) and FTPN (Pedrycz & Camargo, 2003). In FTAPN, one of the *active objects*—ACTALK is introduced and used as the basic agent object to model CMRS, which is a typical multi-agent system. Every abstract object in FTOPN can be trained and reduced independently according to the modeling and analysis requirements for OO concepts supported in FTOPN. The validity of this modeling method has been used to model Brooks CMRS platform in manufacturing IC. The FTAPN can not only model complex MAS, but also be refined into the object-oriented implementation easily. It has provided a methodology to overcome the development problems in agent-oriented software engineering. At the same time, it can also be regarded as a conceptual and practical artificial

intelligence (AI) tool for integrating MAS into the mainstream practice of software development.

State analysis needs to be studied in the future. An extended State Graph (Xu & Jia, 2006) has been proposed to analyze the state change of TOPN models. With the temporal fuzzy sets introduced into FTAPN, the certainty factor about object firing (state changing) needs to be considered in the state analysis.

8. Acknowledgement

This work is jointly supported by the National Nature Science Foundation (Grant No: 60405011, 60575057) and the China Postdoctoral Foundation for China Postdoctoral Science Fund (Grant No: 20040350078) in China.

9. References

Bastide, R.(1995). *Approaches in unifying Petri nets and the object-oriented approach*, Proceedings of the 1st International Workshop on Object-oriented Programming and Models of Concurrency, http://wrcm.dsi.unimi.it/PetriLab/ws95/home.html, Turin, Italy, June, 1995

Battiston, E., Cindio, F.D., Mauri, G.(1988). *OBJSA Nets: a class of high-level nets having objects as domains,* Proceedings of APN'88, Lecture Notes in Computer Science, Vol.340, pp.20-43

Biberstein, O., Buchs, D.(1994). *An object-oriented specification language based on hierarchical algebraic Petri nets*, Proceedings of the IS-CORE Workshop, Amsterdam, Holland, September 1994

Bucci, G., Vivario, E.(1995). *Compositional validation of time-critical systems using communicating time Petri nets,* IEEE Transactions on Software Engineering, Vol.21, No.12, pp.969-992

Florin, G., Fraize, C., Natkin, S.(1991). *Stochastic Petri nets: Properties, applications and tools*, Microelectron. Reliab., Vol. 31, No. 4, pp. 669–697

Guessoum, Z., Briot, J.P.(1999). *From active objects to autonomous agents,* IEEE Concurrency, Vol.7, No.3, pp. 68 – 76

Harel, D., Gery, E.(1996). *Executable object modeling with statechart,* Proceedings of the 18th International Conference on Software Engineering, Germany, March 1996, pp. 246±257.

Hong, J.E., Bae, D.H.(2000). *Software Modeling And Analysis Using a Hierarchical Object-oriented Petri net,* Information Sciences, Vol.130, pp.133-164

Jennings, N.R., Sycara, K., Wooldridge, M.(1998). *A Roadmap of Agent Research and Development,* Autonomous Agents and Multi-Agent Systems, Vol.1, pp.7–38

Jensen, K.(1992). *Coloured Petri Nets: Basic Concepts, Analysis methods and Practical Use,* Springer, ISBN : 3-540-60943-1, Berlin, German

Lakos, C., Keen, C.(1994). *LOOPN++: a new language for object-oriented Perti nets,* Technical Report R94-4, Networking Research Group, Univesity of Tasmania, Australia, April 1994

Merlin, P., Farber, D.(1976). *Recoverability of communication protocols – Implication of a theoretical study* , IEEE Transactions on Communications, Vol.COM-24, pp.1036-1043

Moody, J.O., Antsaklis, P.J. (1998). *Supervisory Control of Discrete Event Systems Using Petri Nets,* Kluwer Academic Publishers, ISBN-10: 0792381998, MA, USA

Murata, T.(1989). Petri Nets: Properties, Analysis and Applications. *Proceedings of IEEE,* Vol.77, No.4, (April 1989) pp.541-580

Pedrycz, W.(1999). *Generalized fuzzy Petri nets as pattern classifiers,* Pattern Recognition Letters, Vol.20 , pp.1489-1498

Pedrycz, W., Camargo, H.(2003). *Fuzzy timed Petri nets*, Fuzzy Sets and Systems, Vol.140, No. 2, pp. 301-330

Peterson, J.L.(1991). *Petri Net Theory and the Modeling of Systems,* Prentice-Hall, ISBN:0-13-661983-5, N.J., USA

Ramchandani, C.(1974). *Analysis of Asynchronous Concurrent Systems by Timed Petri nets,* Massachusetts Institute of Technology, Project MAC, Technology Report 120, 1974

Schuman, S.A.(1997). *Formal Object-oriented Development,* Springer, ISBN-10: 3540199780, Berlin, German.

Sloan, R., Buy, U.(1996). *Reduction Rules for Time Petri Nets*, Acta Inform, Vol.33, pp.687-706

Tsai, J., Yang, S., Chang, Y.(1995). *Timing constraint Petri nets and their application to schedulability analysis of real-time system specifications,* IEEE Transactions On Software Engineering, Vol.21, No.1, pp.32-49

Wang, J.(1998). *Timed Petri Nets – Theory and Application,* Kluwer Academic Publishers, ISBN : 0-7923-8270-6, Boston , USA

Wang, J. ; Deng, Y., Zhou, M.(2000). *Compositional time Petri nets and reduction rules,* IEEE Transactions on Systems, Man and Cybernetics (Part B), Vol. 30, No.4, (Aug. 2000) pp. 562 -572

Xu, H., Jia, P.F.(2005-1). *Fuzzy Timed Object-Oriented Petri Net,* Artificial Intelligence Applications and Innovations (Proceedings of AIAI2005), Springer, pp.148-160, N.Y., USA

Xu, H., Jia, P.F.(2005-2). *Timed Hierarchical Object-oriented Petri Net,* GESTS International Transaction on Computer Science and Engineering, Vol.24, No.1, pp.65-76

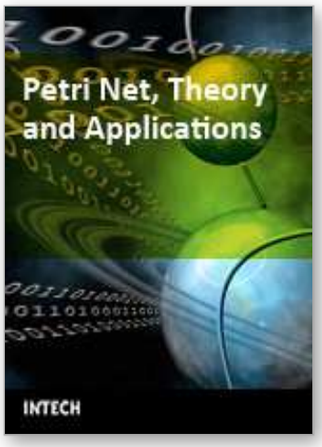
Xu, H., Jia, P.F.(2006). *Timed Hierarchical Object-Oriented Petri Net-Part I: Basic Concepts and Reachability Analysis,* Lecture Notes In Artificial Intelligence (Proceedings of RSKT2006), Vol. 4062, pp.727-734

Xu, H., Jia, P.F.(2007). *A Novel Modeling Method for Cooperative Multi-Robot Systems Using Fuzzy Timed Agent Based Petri Nets,* LNCS (Proceedings of ICCS2007), Vol.4488, pp.956-959

Yao, Y.L.(1994). *A Petri Net Model for Temporal Knowledge Representation and Reasoning,* IEEE Transactions On Systems, Man and Cybernetics, Vol.24, pp.1374-1382

Zhou, M.C.(1995). *Petri Nets in Flexible and Agile Automation,* Kluwer Academic Publishers, ISBN : 0792395573 , MA, USA

IntechOpen



Petri Net, Theory and Applications

Edited by Vedran Kordic

ISBN 978-3-902613-12-7
Hard cover, 534 pages
Publisher I-Tech Education and Publishing
Published online 01, February, 2008
Published in print edition February, 2008

Although many other models of concurrent and distributed systems have been de- veloped since the introduction in 1964 Petri nets are still an essential model for concurrent systems with respect to both the theory and the applications. The main attraction of Petri nets is the way in which the basic aspects of concurrent systems are captured both conceptually and mathematically. The intuitively appealing graphical notation makes Petri nets the model of choice in many applications. The natural way in which Petri nets allow one to formally capture many of the basic notions and issues of concurrent systems has contributed greatly to the development of a rich theory of concurrent systems based on Petri nets. This book brings together reputable researchers from all over the world in order to provide a comprehensive coverage of advanced and modern topics not yet reflected by other books. The book consists of 23 chapters written by 53 authors from 12 different countries.

INTECH
open science | open minds

InTech Europe
University Campus STeP Ri
Slavka Krautzeka 83/A
51000 Rijeka, Croatia
Phone: +385 (51) 770 447
Fax: +385 (51) 686 166
www.intechopen.com

InTech China
Unit 405, Office Block, Hotel Equatorial Shanghai
No.65, Yan An Road (West), Shanghai, 200040, China
中国上海市延安西路65号上海国际贵都大饭店办公楼405单元
Phone: +86-21-62489820
Fax: +86-21-62489821

IntechOpen