http://ansinet.com/itj

ITJ

ISSN 1812-5638

INFORMATION TECHNOLOGY JOURNAL

ANSInet
Asian Network for Scientific Information
308 Lasani Town, Sargodha Road, Faisalabad - Pakistan

Information Technology Journal 8 (1): 95-100, 2009
ISSN 1812-5638

Logic Petri Nets and Equivalency

[1,2]Y.Y. Du and [1]B.Q. Guo
[1]College of Information Science and Engineering,
Shandong University of Science and Technology, Qingdao 266510, China
[2]The State Key Laboratory of Computer Science, Institute of Software,
Chinese Academy of Sciences, Beijing 100080, China

Abstract: Logic Petri nets (LPN) can describe and analyze batch processing function and passing value indeterminacy in cooperative systems and its practical applications are shown with some nontrivial examples. This study focuses on the analysis of the modeling power of LPNs and the equivalency between LPNs and Petri nets with inhibitor arcs (IPN). The equivalency is proved formally and a constructing algorithm of equivalent IPNs from LPNs is proposed based on the disjunctive normal forms of logic input/output expressions. Moreover, the size of an LPN model is smaller than that of the equivalent IPN model.

Key words: Logic petri net, petri nets with inhibitor arcs, equivalency, constructing algorithm, logic expression

INTRODUCTION

Petri nets (PNs) (Murata, 1989) are a well-founded procedure modeling technique and have been used to model and analyze all kinds of procedures with applications ranging from protocols, hardware and embedded systems to flexible manufacturing systems and business processes. PNs are a graphical language. As a result, PNs are intuitive and easy to learn. Also, the semantics of PNs and several enhancements (color, time, hierarchy) have been defined formally (Wang *et al*., 2000).

PNs provide a solid framework for modeling and analyzing workflow processes (Vander Aalst, 2000; Derks *et al*., 2001). Workflow management is becoming a mature technology that can be applied within organizations and across organizational boundaries. It is very important to use an established framework for modeling and analyzing workflow processes (Vander Aalst, 1998), since processes are a main factor in workflow management systems. However, the existing Petri-net-based approaches are mainly used to investigate functional and secure properties of workflows. With rapid increase in the number of workflow processes where multiple organizations are involved, a batch processing function is urgently needed in cooperative systems, i.e., a batch data of the same type from different organizations need to be processed before a deadline. Also, the arrival times of these data are indeterminable, i.e., some data arrive before a given deadline, but others may not arrive. One usually requires that, to enhance the efficiency of workflows, all arrived data be processed at a same time slice, while late received data after the deadline be processed in the next workflow cycle. This situation is called passing value indeterminacy (Yuyue and Changium, 2002; Du *et al*., 2007). Therefore, a good formalism of cooperative systems should be able to model and analyze both the batch processing function and the passing value indeterminacy.

However, it is very difficult to apply Petri nets to the description of the batch processing function and the indeterminacy (Hao and Wang, 2003). Because the processing transition doesn't been fired until all its input places contain tokens strictly. However, in that situation the transition needs a flexible trigger condition, i.e., some of its input places containing tokens are also able to fire the transition. Petri nets with weights of flow arcs may offer the batch processing function, but cannot easily represent the data arrival indeterminacy as the weights are not less than 1. Petri nets with inhibitor arcs (IPN) have the modeling power of a Turing machine (Suzuki and Lu, 1989). Although IPNs may model the batch processing function and the data arrival indeterminacy, the models of systems with these properties are very complex. To tackle the above problems, therefore, a new formal method-Logic Petri Net (LPN), an extend Petri net with logic transitions, is proposed and applied efficiently to the modeling and analysis of stock trading systems and electronic commerce systems (Yuyue and Changium, 2002, 2003; Du *et al*., 2007). It is illustrated sufficiently that LPNs can model explicitly and represent compactly the batch processing function and the data arrival indeterminacy and mitigate the state explosion problem to some extent.

Corresponding Author: Y.Y. Du, College of Information Science and Engineering, Shandong University of Science and Technology, Qingdao 266510, China

LPNs sort the indeterminacy from interactive across organizational boundaries into the input and output indeterminacy and describe them respectively by logic input and output transitions. Firing them is subject to the constraints from logic expressions. In fact, the indeterminacy is implied by the logic expressions attached to logic transitions. Doubtless this will enhance the expression ability of PNs and cater to more application needs. Moreover, LPNs are able to reduce the size of system models. The objective of this study is to investigate the equivalence between LPNs and IPNs and to propose a constructing algorithm of an equivalent IPN from an LPN.

LOGIC PETRI NETS

To apply LPNs to the description of cooperative systems with the batch processing function and the passing value indeterminacy, two types of places in LPNs, control and data places are introduced. Control places are used to store control tokens within an organization, while data places to store data tokens forwarded from other organizations. Thereby, data places are also called interface places. Also, there are two kinds of logic transitions in LPNs, logic input and output transitions. Logic transitions are restricted by logic expressions that can efficiently describe the batch processing function and the passing value indeterminacy, such as the indeterminacy of arrival orders or actual trading quantities in electronic commerce. The transitions cannot be usually expressed in high-level PNs (Atluri and Huang, 2000), since the type of the data passed or generated is explicitly fixed by the variable symbol sums.

Assume that this research is based on the elementary net system, i.e., the capacity of each place is limited to 1. A brief introduction to PNs and IPNs is reviewed as follows.

Definition 1: PN = (P, T, F, M_0) is a PN if and only if

- P is a finite nonempty set of places;
- T is a finite nonempty set of transitions;
- $P \cap T = \varnothing$;
- $F \subseteq (T \times P) \cup (P \times T)$ is a set of arcs, i.e. flow relations;
- M_0: $P \to \{0, 1\}$ is the initial marking.

In the graphic representation, places are drawn as circles or ellipses, transitions as bars. The flow relations between the nodes (i.e., places and transitions) are represented as directed arcs and the tokens of the marking as dots inside places. This study uses the following symbols for the pre-set and post-set of a node $x \in P \cup T$: ${}^{\bullet}x = \{y|(y, x) \in F\}$, $x^{\bullet} = \{y|(x, y) \in F\}$, respectively. $R(M_0)$ represents a set of all markings reachable from M_0. A transition t is enabled if each $p \in {}^{\bullet}t$ contains one token at the current marking $M \in R(M_0)$. If t is enabled, it can fire and a new marking M′ is generated from the current marking M, represented by $M[t>M'$, where when $p \in {}^{\bullet}t\text{-}t^{\bullet}$, $M'(p) = M(p)\text{-}1$; when $p \in t^{\bullet}\text{-}{}^{\bullet}t$, $M'(p) = M(p)+1$; otherwise, $M'(p) = M(p)$.

Definition 2: IPN = (P, T, F, I, M_0) is a PN with inhibitor arcs if and only if

- P is the set of places;
- T is the set of transitions;
- $F \subseteq (P \times T) \cup (T \times P)$ is a set of the edges (flow relations);
- $I \subset P \times T$ is the inhibitor arc set, $I \cap F = \phi$ and $(p, t) \in I$ is represented as a non-directed arc with a small circle at t side;
- M_0: $P \to \{0,1\}$ is the initial marking of IPN;
- Transition firing rules: $\forall t \in T$, t is enabled if $\forall p \in P$: $(p, t) \in F \Rightarrow M(p) \geq 1$ and $(p, t) \in I \Rightarrow M(p) = 0$. Firing t will results in a new marking M′, denoted by $M[t>M'$. And $\forall p \in {}^{\bullet}t$ and $(p, t) \in F$: $M'(p) = M(p)\text{-}1$; $\forall p \in t^{\bullet}$: $M'(p) = M(p)+1$; otherwise: $M'(p) = M(p)$.

In the following, LPNs are first defined formally. Then an example is given to show expressive ability of logic transitions.

Definition 3: LPN = (P, T, F, I, O, M_0) is a logic PN if and only if

- (P, T, F, M_0) is a PN;
- T includes three subsets of transitions, i.e., $T = T_G \cup T_I \cup T_O$, $\forall t \in T_I \cup T_O$: ${}^{\bullet}t \cap t^{\bullet} = \phi$ where
  - T_G denotes a set of traditional transitions;
  - T_I denotes a set of logic input transitions;
  - T_O denotes a set of logic output transitions;
  - I is a mapping from a logic input transition to a logic input expression, i.e., $\forall t \in T_I$, $I(t) = f_I(t)$;
- O is a mapping from a logic output transition to a logic output expression, i.e., $\forall t \in T_O$, $O(t) = f_O(t)$;
- Transition firing rules:
  - $\forall t \in T_G$, the firing rules of t are the same as in PNs;
  - $\forall t \in T_I$, t is enabled only if $f_I(t)|_M = {}^{\bullet}T^{\bullet}$. $M[t> M'$, $\forall p \in {}^{\bullet}t$, if $M(p) = 1$, then $M'(p) = M(p)\text{-}1$; $\forall p \in t^{\bullet}$: $M'(p) = M(p)+1$; $\forall p \notin {}^{\bullet}t \cup t^{\bullet}$: $M'(p) = M(p)$.
  - $\forall t \in T_O$, t is enabled only if $\forall p \in {}^{\bullet}t$: $M(p) = 1$. Firing t will generate a new marking M′, notation $M[t>M'$ and $\exists S \subseteq t^{\bullet}$, S must satisfy $f_O(t)|_{M'} = {}^{\bullet}T^{\bullet}$; $\forall p \in S$: $M'(p) = M(p)+1$; $\forall p \in {}^{\bullet}t$: $M'(p) = M(p)\text{-}1$; $\forall p \notin t^{\bullet} \cup {}^{\bullet}t$ or $\forall p \in t^{\bullet}\text{-}S$: $M'(p) = M(p)$.

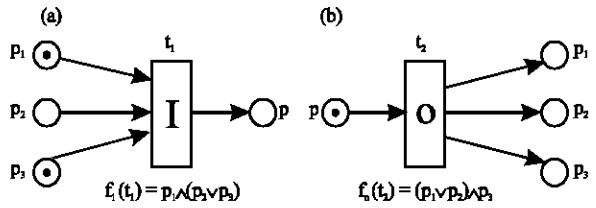

Fig. 1: Representation of logic input\output transitions in LPNs, (a) in put and (b) out put

Usually such S is not unique, but only one case is selected and the selection is random. The randomness of S can describe the output indeterminacy of cooperative systems.

In Fig. 1, t_1/t_2 is a logic input/output transition and it is restrained by a logic input/output expression $f_I(t_1)/f_O(t_2)$. p_1 is the control place; p_2 and p_3 are the data places. According to logic input expression $f_I(t_1) = p_1 \wedge (p_2 \vee p_3)$, in Fig. 1a t_1 is enabled if place p_1 has one token and one or each of places p_2 and p_3 includes one token. In Fig. 1b, p_3 is the control place; p_1 and p_2 are the data places. t_2 is enabled once p includes one token and firing t_2 will generate one token in place p_3 and one or each of places p_1 and p_2 respectively based on logic output expression $f_O(t_2) = (p_1 \vee p_2) \wedge p_3$.

In the application situation, when the control place in Fig. 1a contains a token, starting the business processing (transition t_1) requires at least one of the data places whose tokens come from other organizations contains tokens, instead of each of the data places must contain tokens rigidly in general Petri nets. The situation in Fig. 1b is similar to that in Fig. 1a. So, LPNs can describe clearly the batch processing function and the indeterminacy of data arrival with the corresponding logic expressions. In fact, the logic expressions can enhance the representation ability of LPNs. With respect to LPN expressivity, this study will prove that there is the equivalence between LPNs and IPNs.

EQUIVALENCE OF LPNS AND IPNS

In order to prove the equivalence between LPNs and IPNs, their isomorphism and equivalent definitions are introduced first.

Definition 4 (Isomorphism): Let $\Sigma_1 = (P, T, F, I, O, M_0)$ be an LPN and $\Sigma_2 = (P, T', F', I', M_0)$ an IPN. RG (Σ_i) is the reachable marking graph of Σ_i and R_i is the vertex set of RG (Σ_i), i = 1, 2. If there exists a bijective function $f: R_1 \to R_2$, such that $\forall M_1, M_2 \in R_1$, $t \in T$, $M_1[t> M_2, \Rightarrow \exists t' \in T'$, $f(M_1)[t' > f(M_2)$. Then RG(Σ_1) and RG(Σ_2) are isomorphic.

Definition 5 (Equivalence): Let $\Sigma_1 = (P, T, F, I, O, M_0)$ be an LPN and $\Sigma_2 = (P, T', F', I', M_0)$ an IPN. Σ_1 and Σ_2 are equivalent if and only if RG(Σ_1) and RG (Σ_2) are isomorphic.

Theorem 1: For any LPN, there exists an equivalent IPN.

Proof: Given an LPN $\Sigma_1 = (P, T, F, I, O, M_0)$ and IPN $\Sigma_2 = (P, T', F', I', M_0)$ is equivalent to Σ_1.

For $\forall t \in T$, we transform each transition of Σ_1 into a corresponding transition or a subnet of Σ_2 on the basis of the following three cases.

$t \in T_G = T - T_I \cup T_O$: Let $t \in T'$; $\forall p \in P$, if $(p, t) \in F$, then $(p, t) \in F'$; and if $(t, p) \in F$, then $(t, p) \in F'$.

$t \in T_I$: Let ${}^\bullet t = \{p_{i1}, p_{i2}, \ldots, p_{ik}\}$, $f_I(t)$ is its logic input expression. t is enabled once the value of $f_I(t)$ is true on the basis of the token change of places p_{i1}-p_{ik}. In fact, there may be several situations of places p_{i1}-p_{ik} where $f_I(t)$ is true. However, $f_I(t)$ can be transformed into a disjunctive normal form and the disjunctive normal form is unique, i.e., each disjunctive clause in the disjunct is a conjunct which consists of all positive or negative places p_{i1}-p_{ik} and each conjunct corresponds to a situation of places p_{i1}-p_{ik} where $f_I(t)$ is true.

Thus, the disjunctive normal form of $f_I(t)$ can be represented as follows:

$$f_I(t) = \bigvee_{l=1}^{m} \bigwedge_{j=1}^{k} \tau_{lj}$$

where, $\tau_{lj} = p_{ij}$ or $\neg p_{ij}$. $f_I(t)$ is true if and only if one of m disjunctive clauses is satisfied. Any two disjunctive clauses are mutually exclusive. In the following, m transitions in Σ_2 corresponding to m kinds of the true value states are constructed in the disjunctive normal form respectively. That is, a logic input transition t with logic expression $f_I(t)$ in Σ_1 can be represented equivalently by a subnet including m transitions in Σ_2 and m transitions correspond to m disjunctive clauses in the disjunctive normal form of $f_I(t)$ respectively. The subnet is constructed in detail as follows:

For any disjunctive clause $\bigwedge_{j=1}^{k} \tau_{lj}$ $l \in \{1, 2, \ldots, m\}$

Assume that the disjunctive clause corresponds to a transition t_l' in Σ_2, i.e., $t_l' \in T'$. Then the arc set of this subnet is defined. $\forall j \in \mathbb{N}$, if $\tau_{lj} = p_{ij}$, then $(p_{ij}, t_l') \in F'$; if $\tau_{lj} = \neg p_{ij}$, then $(p_{ij}, t_l') \in I'$.

If $\forall p \in t^{\bullet}$, then $(t_i', p) \in F'$. Therefore, firing a logic input transition of Σ_1 corresponds to firing a transition of Σ_2, i.e., if a logic input transition is enabled in Σ_1, then there must be an enabled transition in the corresponding subnet of Σ_2 and it is unique.

$t \in T_O$: Let $t^{\bullet} = \{p_{i1}, p_{i2}, \ldots, p_{ik}\}$, $f_O(t)$ is its logic output expression. t is enabled once its input place p includes a token. Firing t will generate a token in all or some output places and must assure that $f_O(t)$ is true at the next state of places p_{i1}-p_{ik}..

Similarly, $f_O(t)$ can be transformed into a unique disjunctive normal form. Without loss of generality, assume that the disjunctive normal form consists of r disjunctive clauses. Thus,

$$f_o(t) = \bigvee_{l=1}^{r} \bigwedge_{j=1}^{k} \tau_{lj}$$

where $\tau_{lj} = p_{ij}$ or $\neg p_{ij}$. $f_O(t)$ decides all possible states of places p_{i1}-p_{ik} and each disjunctive clause corresponds to a possible state in which $f_O(t)$ is true. Thus r transitions in Σ_2 corresponding to r kinds of the possible states are constructed in the disjunctive normal form respectively. That is, a logic output transition t with logic expression $f_O(t)$ in Σ_1 can be represented equivalently by a subnet including r transitions in Σ_2 and r transitions correspond to r disjunctive clauses in the disjunctive normal form of $f_O(t)$ respectively. The subnet is built as follows.

For any disjunctive clause,

$$\bigwedge_{j=1}^{k} \tau_{lj}$$

$l \in \{1, 2, \ldots, r\}$. Assume that the disjunctive clause corresponds to a transition t_l' in Σ_2, i.e., $t_l' \in T'$. Then the arc set of this subnet is defined. $\forall j \in \mathbb{N}$, if $\tau_{lj} = p_{ij}$, then $(t_l', p_{ij}) \in F'$; if $\tau_{lj} = \neg p_{ij}$, then $(t_l', p_{ij}) \notin F'$. $\forall p \in {}^{\bullet}t$, then $(p, t_l') \in F'$. Therefore, firing a logic output transition of Σ_1 corresponds to the occurrence of a transition in Σ_2, i.e., firing a logic output transition t is functionally equivalent to the occurrence of one of transitions t_1'- t_r'.

Therefore, the place set P in Σ_1 is the same as that in Σ_2. But $T \neq T'$, $F \neq F'$ and $|T| \leq |T'|$, $|F| \leq |F'|$, where $|T|$ denotes the size of set T. Since Σ_1 and Σ_2 have the same initial marking M_0, the equivalence between Σ_1 and Σ_2 are now proved based on the reachable marking graph. In Σ_1, for $\forall M_1, M_2 \in R(\Sigma_1)$, $t \in T$ and $M_1[t> M_2$. Then there is a mapping function f: $R(\Sigma_1) \rightarrow R(\Sigma_2)$ based on the construction of Σ_2, such that $f(M_1) = M_1$ and $f(M_2) = M_2$ and in Σ_2, if $t \in T_G$, then $\exists t' \in T'$: $t' = t$ and $f(M_1)$ $[t' > f(M_2)$; if $t \in T_I \cup T_O$, then $\exists t_l' \in T'$: $t_l' = t$ and $f(M_1)$ $[t_l' > f(M_2)$. Accordingly, f is an identity mapping and satisfies injective and surjection requirements at $M \in R(M_0)$. That means that Σ_1 and Σ_2 have the same behavior characteristics. Moreover, the structure of Σ_2 is unique since the disjunctive normal form is only one. So f is a bijective function and $RG(\Sigma_1)$ and $RG(\Sigma_2)$ are isomorphic.

Consequently, Σ_1 and Σ_2 are equivalent based on definition 5. The conclusion holds.

From Theorem 1 and the construction of Σ_2, the following constructing algorithm of an equivalent IPN from an LPN can be obtained.

Algorithm 1 (Constructing Algorithm): Transforming an LPN into an equivalent IPN

Input: an LPN $\Sigma_1 = (P, T, F, I, O, M_0)$

Output: an equivalent IPN $\Sigma_2 = (P, T', F', I', M_0)$

(1) IPN.P = LPN.P;
(2) For each transition t in $LPN.T_G$
(3) $IPN.T' = IPN.T' \cup \{t\}$;
(4) For each p in ${}^{\bullet}t$
(5) $IPN.F' = IPN.F' \cup \{(p, t)\}$;
(6) End for
(7) For each p in $t^{\bullet}$
(8) $IPN.F' = IPN.F' \cup \{(t, p)\}$;
(9) End for
(10) End for
(11) For each t in $LPN.T_I$
(12) For each disjunctive clause $\bigwedge_{j=1}^{k} \tau_{lj}$ of $f_l(t)$
(13) $IPN.T' = IPN.T' \cup \{t_l'\}$;
(14) For each p_{ij} in ${}^{\bullet}t$
(15) If $\tau_{lj} = p_{ij}$ then $IPN.F' = IPN.F' \cup \{(p_{ij}, t_l')\}$;
(16) else $IPN.I' = IPN.I' \cup \{(p_{ij}, t_l')\}$;
(17) End for
(18) For each p in $t^{\bullet}$
(19) $IPN.F' = IPN.F' \cup \{(t_l', p)\}$;
(20) End for
(21) End for
(22) End for
(23) For each t in $LPN.T_O$
(24) For each disjunctive clause $\bigwedge_{j=1}^{k} \tau_{lj}$ of $f_O(t)$
(25) $IPN.T' = IPN.T' \cup \{t_l'\}$;
(26) For each p_{ij} in $t^{\bullet}$
(27) If $\tau_{lj} = p_{ij}$ then $IPN.F' = IPN.F' \cup \{(t_l', p_{ij})\}$;
(28) End for
(29) For each p in ${}^{\bullet}t$
(30) $IPN.F' = IPN.F' \cup \{(p, t_l')\}$;
(31) End for
(32) End for
(33) End for

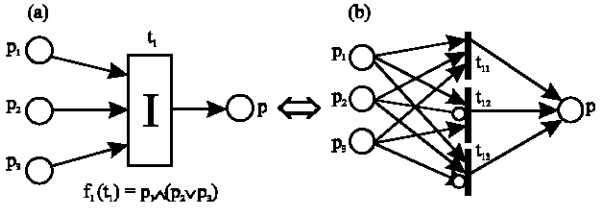

Fig. 2: Equivalent IPN subnet of a logical input transition, (a) logical Input transition and (b) equivalent IPN subnet

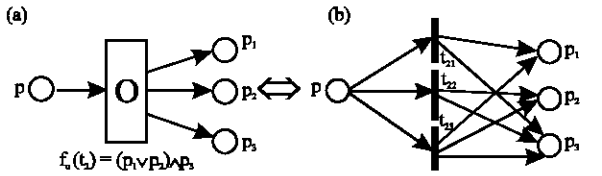

Fig. 3: Equivalent IPN subnet of a logical output transition (a) logical output transition and (b) equivalent IPN subnet

For any LPN, therefore, its equivalent IPN can be easily built based on theorem 1 and algorithm 1. In the following, the transformation processes from a logic transition in LPN to an equivalent subnet in IPN are illustrated with two examples (Fig. 2, 3).

Example 1: Transforming a logic input transition in LPNs to an equivalent subnet in IPNs. In Fig. 2a, t_1 is a logic input transition in an LPN and $f_I(t_1) = p_1 \wedge (p_2 \vee p_3)$ is its logic input expression. $t_1^\bullet = \{p\}$ and $^\bullet t_1 = \{p_1, p_2, p_3\}$. $f_I(t_1)$ is transformed into a disjunctive normal form as follows.

$$
\begin{aligned}
f_I(t_1) &= p_1 \wedge (p_2 \vee p_3) \\
&= (p_1 \wedge p_2) \vee (p_1 \wedge p_3) \\
&= ((p_1 \wedge p_2) \wedge (p_3 \vee \neg p_3)) \vee ((p_1 \wedge p_3) \wedge (p_2 \vee \neg p_2)) \\
&= (p_1 \wedge p_2 \wedge \neg p_3) \vee (p_1 \wedge \neg p_2 \wedge p_3) \vee (p_1 \wedge p_2 \wedge p_3)
\end{aligned}
$$

By means of the proof process of theorem 1 and algorithm 1, logic input transition t_1 can be transformed equivalently into the subnet of an IPN in Fig. 2b. From the proof process of theorem 1, three disjunctive clauses of $f_I(t_1)$ correspond with three transitions in the IPN subnet, i.e., $(p_1 \wedge p_2 \wedge \neg p_3) \rightarrow t_{11}$, $(p_1 \wedge \neg p_2 \wedge p_3) \rightarrow t_{12}$ and $(p_1 \wedge p_2 \wedge p_3) \rightarrow t_{13}$. That is, the disjunctive normal form of $f_I(t_1)$ decides fully the structure of the corresponding IPN subnet. (p_3, t_{11}) and (p_2, t_{12}) are two inhibitor arcs. The place sets in Fig. 2a and Fig. 2b are the same, i.e., $^\bullet t_1 \cup t_1^\bullet = {}^\bullet t_{11} \cup t_{11}^\bullet \cup {}^\bullet t_{12} \cup t_{12}^\bullet \cup {}^\bullet t_{13} \cup t_{13}^\bullet = \{p_1, p_2, p_3, p\}$.

Example 2: Transforming a logic output transition in LPNs to an equivalent subnet in IPNs. In Fig. 3a, t_2 is a logic output transition in an LPN and $f_O(t_2) = (p_1 \vee p_2) \wedge p_3$ is its logic output expression. $t_2^\bullet = \{p_1, p_2, p_3\}$ and $^\bullet t_2 = \{p\}$. The disjunctive normal form of $f_O(t_2)$ is deduced as follows.

$$
\begin{aligned}
f_O(t_2) &= (p_1 \vee p_2) \wedge p_3 \\
&= (p_1 \wedge p_3) \vee (p_2 \wedge p_3) \\
&= ((p_1 \wedge p_3) \wedge (p_2 \vee \neg p_2)) \vee ((p_2 \wedge p_3) \wedge (p_1 \vee \neg p_1)) \\
&= (p_1 \wedge \neg p_2 \wedge p_3) \vee (\neg p_1 \wedge p_2 \wedge p_3) \vee (p_1 \wedge p_2 \wedge p_3)
\end{aligned}
$$

Similarly, Fig. 3b shows the equivalent IPN subnet of t_2. By the constructing method of IPNs, three disjunctive clauses of $f_O(t_2)$ correspond with three transitions in the IPN subnet. Therefore, the disjunctive normal form of $f_O(t_2)$ decides fully the structure of the corresponding IPN subnet.

From the examples, it is a fact that the size of an IPN model can be reduced in its equivalent LPN model, since many transitions are capsulized into an equivalent logic transition. Their modeling power and behaviors are same, because IPNs and LPNs have the same reachable marking graph in essence and IPNs and LPNs are isomorphic.

CONCLUSION

With the rapid progress of cooperative and network extent of interorganizational workflows, batch processing function and passing value indeterminacy of workflow processes need often to be considered in the design of workflow systems. Moreover, formal modeling and analysis technologies should have the modeling ability of these functions. To deal with the problems, a new extension of PNs, i.e., LPNs, is proposed in (Yuyue and Changium, 2002; Du *et al*., 2007) and its practical application is illustrated with some nontrivial examples. This study focuses on the analysis of the modeling power of LPNs and the equivalency between LPNs and IPNs. The equivalency is proved formally and a constructing algorithm of equivalent IPNs from LPNs is proposed based on the disjunctive normal forms of logic input/output expressions. Moreover, the size of LPN models is smaller than that of the equivalent IPN models. Further study will be the investigation of fundamental properties of LPNs, such as liveness, state equivalency and reachability. Moreover, the analysis technologies of LPNs will be studied and its developmental tool is considered and designed.

ACKNOWLEDGMENTS

This study was supported in part by the National Natural Science Foundation of China under Grant 60773034 the Taishan Scholar Construction

Project of Shandong Province, China; the National Basic Research Program of China (973 Program) under Grants 2007CB316502 and 2004CB318001-03; and the Open Project of the State Key Laboratory of Computer Science, Institute of Software, Chinese Academy of Sciences under Grant SYSKF0804.

REFERENCES

Atluri, V. and H.K. Huang, 2000. A petri net based safety analysis of workflow authorization models. J. Comput. Security, 8: 209-240.

Derks, W., J. Dehnert, P. Grefen and W. Jonker, 2001. Customized atomicity specification for transactional workflows. Proceedings 3rd International Symposium on Cooperative Database Systems for Advanced Applications, April 23-24, Beijing, China, pp: 155-164.

Du, Y.Y. and C.J. Jiang, 2002. Formal representation and analysis of batch stock trading systems by logical petri net workflows. Lect. Notes Comput. Sci., 2495: 221-225.

Du, Y.Y. and C.J. Jiang, 2003. Towards a workflow model of real-time cooperative systems. Formal Methods and Software Engineering: 5th International Conference on Formal Engineering Methods, ICFEM 2003, Singapore, LNCS., 2885, November 5-7, Springer-Verlag, Berlin, pp: 452-470.

Du, Y.Y., C.J. Jiang and M.C. Zhou, 2007. Modeling and analysis of real-time cooperative systems using Petri nets. IEEE Trans. Syst., Man Cybernetics-Part A: Syst. Hum., 37: 643-654.

Hao, K. and B. Wang, 2003. Non-determined Petri nets. Mini-Micro Syst., 24: 582-584 (In Chinese).

Murata, T., 1989. Petri nets: Properties, analysis and applications. Proceedings of the IEEE, 77: 541-580.

Suzuki, I. and H. Lu, 1989. Temporal Petri nets and their application to modeling and analysis of a handshake daisy chain arbiter. IEEE Trans. Comput., 38: 696-704.

Van der Aalst, W.M.P., 1998. The application of Petri nets to workflow management. J. Circuit Syst. Comput., 8: 21-66.

Van der Aalst, W.M.P., 2000. Loosely coupled interorganizational workflows: Modeling and analyzing workflows crossing organizational boundaries. Inform. Manage., 37: 67-75.

Wang, J.C., Y. Deng and G. Xu, 2000. Reachability analysis of real-time systems using time Petri nets. IEEE Trans. Syst. Man Cybernet B Cybernet, 30: 725-736.