On Properties of Concurrent System Based on Petri Net Language*

JIANG Chang-jun[1,2], LU Wei-ming[3]

[1](Department of Computer Science and Engineering, Tongji University, Shanghai 200092, China);
[2](Shandong Institute of Science and Technology, Taian 271019, China);
[3](Institute of Mathematics, The Chinese Academy of Sciences, Beijing 100080, China)
E-mail: cjjiang@online.sh.cn; wmlu@math20.math.ac.cn
http://www.tongji.edu.cn
Received September 18, 1999; accepted April 10, 2000

Abstract: In this paper, two language characterizations for weak liveness (free deadlock) and liveness of Petri nets are given. Some language properties of synchronous composed Petri nets are discussed. Based on Petri net language, a necessary and sufficient condition is given for live Petri net (bounded), and then the liveness preservation in a synchronous composed net is studied and a necessary and sufficient condition is obtained. Those results give a formal language method for net liveness testing and liveness controlling.

Key words: Petri net; concurrent system; liveness; synchronous composition; testing; controlling

Petri net has been widely used in concurrent system design, modeling, analyzing, and controlling since 1962, in which C. A. Petri introduced this powerful mathematical tool for studying concurrency. Now more and more people in the fields of computer science and technology and others are interested in it.

While a system is studied, it is the first step to model the system by Petri net, then to analyze the modeled system using net theory in order to know some important system properties about structure, dynamical behaviors, etc. Based on these properties it is possible to study system scheduling and controlling.

In system study, the liveness is basic and useful system property, i.e. scientists are interested in having a system with no deadlock or even more, no dead event. Unfortunately, a marking system in live is still an open problem in net theory although in some special net classes it was solved. References. [1~3] studied structure net, Ada net etc., and got a deadlock condition, i.e. a necessary condition for liveness only, Ref. [4] studied marked graph (T-graph) and got a necessary and sufficient condition for liveness, Ref. [5] got a necessary and sufficient condition for live state machine (S-graph), Ref. [6] extended the results of Refs. [4,5] and gave a necessary and suffi-

* This project is supported by the National Natural Science Foundation of China under Grant Nos. 69973029, 69933020 (国家自然科学基金), the National Key Basic Science Foundation of China under Grant No. G1998030604 (国家重点基础研究基金), the National Excellent Ph.D. Paper Author Foundation of China under Grant No. 199934 (全国优秀博士学位论文作者基金), the Key Basic Science Foundation of Shanghai of China (上海市重点基础研究基金), Shanghai Dawn Plan Foundation (上海市曙光计划基金), and the Excellent Young Scientist Foundation of Shandong Province of China (山东省优秀青年科学家基金).
JIANG Chang-jun was born in 1962. He received his Ph.D. degree from Institute of Automation, the Chinese Academy of Sciences. He is currently a professor at the Department of Computer Science and Engineering, Shanghai Tongji University. His research areas include Petri net, concurrent system, model checking and fuzzy reasoning. **LU Wei-ming** was born in 1941. He is a professor at the Institute of Mathematics, the Chinese Academy of Sciences. His research areas include Petri net, algorithm and complexity, concurrent system.

cient condition for live free choice net, Refs. [7] got a necessary and sufficient condition for live weighted T graph. Ref. [4~7] were based on net structure and initial marking, and Ref. [8] was based on reachable marking graph and got a necessary and sufficient condition for live bounded Petri net.

Based on Petri net language we give a necessary and sufficient condition for live Petri net (not strictly bounded) here, and then study the liveness presevation in a synchronous composed net and get a necessary and sufficient condition. Those results give us a formal language method for net liveness testing and liveness controlling, which is based on the same idea as in behavior semantics oriented system design methodology.

1 Basic Definitions and Terminology

3-tuple $N=(P,T;F)$ is a net iff:

(1) $P\cap F=\varnothing \wedge P\cup T\neq\varnothing$;

(2) $F\subseteq(P\times T)\cup(T\times P)$;

(3) $\mathrm{dom}(F)\cup\mathrm{cod}(F)=P\cup T$.

For $\forall x\in P\cup T$,

$$x=\{y\in P\cup T\mid(y,x)\in F\}$$

and

$$x^{\cdot}=\{y\in P\cup T\mid(x,y)\in F\}$$

are preset and postset of x, respectively.

4-tuple $PN=(P,T;F,M_0)$ is a Petri net iff:

(1) $N=(P,T;F)$ is a net;

(2) $M:P\to Z$ (set of non-negative integers) is marking of PN, where M_0 is an initial marking of PN;

(3) PN satisfies the following fire rules:

a) $\forall t\in T$, if $\forall p\in{}^{\cdot}t$, $M(p)\geqslant 1$, t is enabeled under M (denoted as $M[t>$);

b) If t is enabeled under M, then t can be fired. Let t be fired from M to M' (denoted as $M[t>M'$), then $\forall p\in P$

$$M'(p)=\begin{cases}M(p)+1 & \text{if } p\in t^{\cdot}-{}^{\cdot}t,\\ M(p)-1 & \text{if } p\in{}^{\cdot}t-t^{\cdot},\\ M(p) & \text{otherwise.}\end{cases}$$

Let M_0 and $M_k(k>0)$ be two markings of PN. If there exist a transition sequence $\sigma=t_1t_2\ldots t_k$ and a marking sequence $M_1,M_2,\ldots,M_{k-1}$, such that:

$$M_{i-1}[t_i>M_i,\ i=1,2,\ldots,k,$$

then σ is a fire sequence of PN and M_k is reachable from M_0 (denoted as $M_0[\sigma>M_k$). The set of reachable markings from M_0 is denoted as $R(M_0)$.

2 Liveness Descriptions

Definition 1[14]. Let $PN=(P,T;F,M_0)$ be a Petri net. PN is weak live (free-deadlock) iff $\forall M\in R(M_0)\ \exists t\in T:M[t>$.

Definition 2[14]. Let $PN=(P,T;F,M_0)$ be a Petri net. PN is live iff $\forall t\in T\forall M\in R(M_0)\ \exists M'\in R(M):M'[t>$.

Definition 3. Let $PN=(P,T;F,M_0)$ be a Petri net, and $G\subseteq R(M_0)$. Then

$$L(PN)=\{\sigma\in T^{*}\mid M_0[\sigma>\}$$

and

$$L_G(PN)=\{\sigma\in T^{*}\mid M_0[\sigma>M\wedge M\in G\}$$

are behavior language and recognition language of PN, respectively.

For $\forall\ \sigma\in T^*$ write $\&(\sigma)=\{x\mid x \text{ is a letter in } \sigma\}$.

Definition 4. Let $L\subset T^*$ be a language. Then

$$\overleftarrow{L}=\{\sigma\in T^*\mid\exists\ \sigma'\in T^*:\sigma\circ\sigma'\in L\},$$

$$\vec{L}=\{\sigma\in T^*\mid\exists\ \sigma'\in T^*:\sigma\circ\sigma'\in L\wedge|\sigma'|\neq 0\},$$

$$\overrightarrow{L}=\{\sigma\in T^*\mid\exists\ \sigma'\in T^*:\sigma\circ\sigma'\in L\wedge\&(\sigma')=T\},$$

are closed language, strict closed language and strong closed language, respectively.

Theorem 1. Let $PN=(P,T;F,M_0)$ be a Petri net. PN is weak live iff $L(PN)=L(\overrightarrow{PN})$.

Proof. ($\Rightarrow$) PN is weak live, then $\forall\ M\in R(M_0)$, $\exists\ t\in T$, such that $M[t>$, then $\forall\ \sigma\in L(PN)$, $M_0[\sigma>M$, $\exists\ t\in T\subseteq T^*$, such that $M_0[\sigma>M[t>$, that is $\sigma\circ t\in L(PN)$. Thus $\sigma\in L(\overrightarrow{PN})$, hence $L(PN)\subseteq L(\overrightarrow{PN})$. On the other hand, obviously, $L(\overrightarrow{PN})\subseteq L(PN)$. Hence $L(PN)=L(\overrightarrow{PN})$.

($\Leftarrow$) If $L(PN)=L(\overrightarrow{PN})$, then $\forall\ \sigma\in L(PN)$, $\exists\ \sigma'\in T^*$ such that $\sigma\circ\sigma'\in L(PN)$, and $|\sigma'|\neq 0$, then $\forall\ M\in R(M_0)$, $\exists\ t'\in T$, t' is the first letter in σ', such that $M_0[\sigma>M[t'>$, hence PN is weak live.

Theorem 2. Let $PN=(P,T;F,M_0)$ be a Petri net, then PN is live iff $L(PN)=L(\overleftarrow{PN})$.

Proof. ($\Rightarrow$) (contrary proof). Suppose $L(PN)\neq L(\overleftarrow{PN})$, then $\exists\ \sigma\in L(PN)$, consider the following two cases:

Case 1. If $\neg\exists\ \sigma'\in T^*$, such that $\sigma\circ\sigma'\in L(PN)$, thus $\sigma\notin L(\overleftarrow{PN})$. According to Theorem 1, $\exists\ M\in R(M_0)$, such that for $\forall\ t\in T$, we have $\neg(M[t>)$. Hence PN isn't live, contradiction.

Case 2. For $\forall\ \sigma'\in T^*$, if $\sigma\circ\sigma'\in L(PN)$, then $\&(\sigma')\neq T$. Hence $\exists\ t'\in T-\&(\sigma')$, such that $\exists\ M\in R(M_0)$, $\forall\ M'\in R(M)$, we have $M_0[\sigma>M'\wedge\neg(M'[t'>$, that is t isn't live. Hence PN isn't live, contradiction.

From Cases 1 and 2, $\sigma\in L(\overleftarrow{PN})$, that is $L(PN)\subseteq L(\overleftarrow{PN})$. On the other hand, $L(\overleftarrow{PN})\subseteq L(PN)$. Hence $L(PN)=L(\overleftarrow{PN})$.

($\Leftarrow$) Since $L(PN)=L(\overleftarrow{PN})$, hence $\forall\ \sigma\in L(PN)$, $\exists\ \sigma'\in T^*$, such that $\sigma\circ\sigma\in L(PN)\wedge\&(\sigma)=T$. For $\forall\ t\in T$, let $\sigma'=\sigma'_1\circ t\circ\sigma'_2$, that is $\forall\ M\in R(M_0)\exists\ M'\in R(M)$, such that $M_0[\sigma>M[\sigma'_1>M'[t>$. Hence PN is live.

Definition 6[12]. Let $PN=(P,T;F,M_0)$ be a Petri net and $G\subseteq R(M_0)$. PN is non-blocking for G iff $L(PN)=L_G(PN)$.

The condition that PN is non-blocking for G means each trace of PN can be extended to a recognition state in G.

Definition 7. Let $PN=(P,T;F,M_0)$ be a Petri net and $G\subseteq R(M_0)$.

(1) PN is strict non-blocking for G iff $L(PN)=L_G(\overleftarrow{PN})$.

(2) PN is strong non-blocking for G iff $L(PN)=L_G(\overleftarrow{PN})$.

The condition that PN is strict non blocking for G means each trace of PN can be non-emptily extended to a recognition state in G.

The condition that PN is strong non-blocking for G means each trace of PN can be extended to a recognition state in G, and each t in T can occur in extended part.

As non-blocking is an important concept in the supervisory control theory of discrete event dynamic systems (DEDS) and liveness is an important concept in the net theory. The relationship between non-blocking and liveness is interesting.

Theorem 3. Let $PN=(P,T;F,M_0)$ be a Petri net and $G\subseteq R(M_0)$ be a set of PN's recognition state.

(1) PN is non-blocking for G, then PN is closed *;

(2) PN is strict non-blocking for G, then PN is weak live;

(3) PN is strong non-blocking for G, then PN is live.

Proof. (1) Obvious.

(2) Easy to see

$$L_G(\vec{PN})\subseteq L(\vec{PN})\subseteq L(PN)$$

Since PN is strict non-blocking, then

$$L_G(\vec{PN})=L(PN),$$

hence $L(PN)=L(\vec{PN})$, that is PN is weak live.

(3) Derived from (2).

By these theorems liveness of Petri nets can be tested easier than before.

Example 1. PN_1 and PN_2 are two Petri nets in Fig. 1.

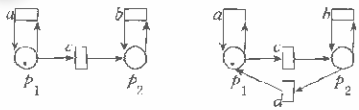

Fig. 1

For PN_1:

(1) Since $L(PN_1)=L(\overline{PN}_1)=\{a^*cb^*\}$, hence PN_1 is weak live.

(2) Get $\sigma=a^*c\subset(PN_1)$, if $\sigma'\in T$, such that $\sigma\circ\sigma'\in L(PN_1)$, thus $L(PN_1)\neq L(\vec{PN}_1)$. Hence PN_1 isn't live.

(3) Get $G_1=\{(1,0)^T\}\subset R(M_{01})$, then $\sigma=cb^*\in L(PN_1)$, but $\sigma\notin L_{G_1}(PN_1)$, then $\sigma\notin L_{G_1}(\vec{PN}_1)$, thus $L(PN_1)\neq L(\vec{PN}_1)$. Hence, PN_1 is strict blocking for G_1.

Get $G'_1=\{(0,1)^T\}\subset R(M_{01})$, then $L(PN_1)=L_{G'_1}(\vec{PN}_1)=\{a^*cb^*\}$, hence PN_1 is strict non-blocking for G'_1.

(4) Since PN_1 isn't live by Theorem 3(2), we know for $\forall\, G_1\subseteq R(M_{01})$, PN_1 is strong blocking for G_1.

For PN_2:

(1) Since $L(PN_2)=L(\overline{PN}_2)=\{((a^*cb^*d)^*)\}$, hence, PN_2 is live. Obviously, PN_2 is also weak live.

(2) Get $G_2=\{(1,0)^T\}\subset R(M_{02})$, then PN_2 is strict blocking for G_2, and it is also strong blocking for G_2.

If get $G'_2=R(M_{02})$, then PN_2 is strict non-blocking and strong non-blocking for G'_2.

In fact, we have the following consequence:

Corollary 1. Let $PN=(P,T;F,M_0)$ be a Petri net and $G\subseteq R(M_0)$. If PN is strong non-blocking for G, then PN is strict non-blocking for G.

Proof. Since PN is strong non-blocking for G. Hence, $L(PN)=L_G(\vec{PN})$. On the other hand, $L_G(\vec{PN})\subseteq L_G(\overline{\vec{PN}})\subseteq L(PN)$, hence $L(PN)=L_G(\overline{\vec{PN}})$, thus PN is strict non-blocking for G.

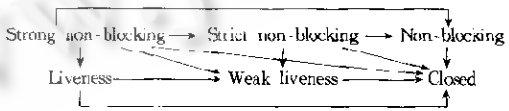

Fig. 2 Relation graph of liveness and non-blocking

Now, the following relation graph is obvious.

3 Liveness Control

In Ref. [10], system scheduling, control and synchronous composition were discussed. In this section there are some formal discussions.

---

* PN is closed iff $L(PN)=\overline{L(PN)}$.

Definition 8. Let $PN_i=(P_i,T_i;F_i,M_{0i})$, $i=1,2$, be two Petri nets. If $PN=(P,T;F,M_0)$ is a Petri net and

(1) $P_1\cap P_2=\varnothing$, $P=P_1\cup P_2$;

(2) $T_1\cap T_2\neq\varnothing$, $T=T_1\cup T_2$;

(3) $F=F_1\cup F_2$;

(4) $M_0(p)=\begin{cases}M_{01}(p), & \text{if } p\in P_1\\ M_{02}(p), & \text{if } p\in P_2.\end{cases}$

Then PN is a synchronous composition net of PN_1 and PN_2, denoted as $PN=PN_1OPN_2$.

Here O is an operator for system composition. In Ref. [11] there are more operators.

Definition 9. Let $T'\subseteq T$, $\sigma\in T^*$, and $\Gamma_{T\to T'}(\sigma)$ be the string obtained by deleting the letters in $T-T'$ from σ. Then it is a projection from T^* to T'^*.

Example 2. PN_1 and PN_2 are two Petri nets in Fig. 3. Petri net $PN=PN_1OPN_2$ is a synchronous composition net of PN_1 and PN_2.

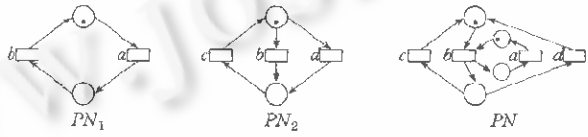

Fig. 3

Obviously, $L(PN_1)=\{(a\overline{b})^*\}$, $L(PN_2)=\{(b(c+\overline{d}))^*\}$, then

$$L(PN)=\{a(b((c+d)\,\|\,a))^*\}.$$

If $\sigma=abacbdabd\in L(PN)$, then

$$\Gamma_{T\to T_1}(\sigma)=ababab\in L(PN_1),\ \Gamma_{T\to T_2}(\sigma)=bcbdbd\in L(PN_2).$$

Theorem 4. Let $PN_i=(P_i,T_i;F_i,M_{0i})$, $i=1,2$, be Petri nets. Let $G_i\subseteq R(M_{0i})$ be sets of recognition states of PN_i, $i=1,2$. $PN=PN_1OPN_2$, and

$$G=\{(M_1^T\,M_2^T)^T\mid M_i\in G_i\wedge i=1,2\},$$

then

(1) $L(PN)=\Gamma^{-1}_{T_1\to T}(L(PN_1))\cap\Gamma^{-1}_{T_2\to T}(L(PN_2))$;

(2) $L_G(PN)=\Gamma^{-1}_{T_1\to T}(L_{G_1}(PN_1))\cap\Gamma^{-1}_{T_2\to T}(L_{G_2}(PN_2))$.

Proof. (1) For convenience, suppose the empty letter $\varepsilon\notin T$, and

(i) $M[\varepsilon>M'\Leftrightarrow M=M'$,

(ii) $\forall\, t\in T, t\circ\varepsilon=t$.

$\sigma\in L(PN)$ iff $\exists\, M_0,M_1,\ldots,M_k\in R(M_0)$, such that

$$M_0[\sigma(1)>M_1[\sigma(2)>M_2\ldots M_{k-1}[\sigma(k)>M_k,$$

where

$$\sigma=\sigma(1)\circ\sigma(2)\circ\ldots\circ\sigma(k).$$

Let $M_i=(M_{i1}^T,M_{i2}^T)^T$, $i=0,1,2,\ldots,k$, iff

$$\begin{aligned}&[((M_{i1}^T,M_{i2}^T)^T\geqslant((C_1^-\sigma(i+1))^T,0^T)^T)\wedge((M_{i+1,1}^T,M_{i+1,2}^T)^T=(M_{i1}^T,M_{i2}^T)^T+((C_1\underline{\sigma(i+1)})^T,0^T)^T)]\vee\\ &[((M_{i1}^T,M_{i2}^T)^T\geqslant(0^T,(C_2^-\underline{\sigma(i+1)})^T,)^T)\wedge((M_{i1}^T,M_{i2}^T)^T=(M_{i1}^T,M_{i2}^T)^T+((C_2\underline{\sigma(i+1)})^T,0^T)^T)]\vee\\ &[((M_{i1}^T,M_{i2}^T)^T\geqslant((C_1^-\sigma(i+1))^T,(C_2^-\sigma(i+1))^T)^T)\wedge\\ &((M_{i+1,1}^T,M_{i+1,2}^T)^T=(M_{i1}^T,M_{i2}^T)^T+((C_1^-\sigma(i+1))^T,(C_2^-\sigma(i-1))^T)^T)]\end{aligned}$$

where $C_j=C_j^+-C_j^-$ is the incidence matrix of PN_j[6], $\underline{\sigma(i+1)}$ indicates $|T_j|-$ vectors $(0\,0\ldots1\ldots0)^T$, $i=1,2,\ldots,k$; $j=1,2$,

iff

$$[(M_{i1}[\sigma_1(i+1)>M_{i+1,1})\wedge(M_{i2}-M_{i+1,2})\wedge(\sigma_1(i+1)-\sigma(i+1))\wedge(\sigma_2(i+1)-\varepsilon]\vee$$
$$[(M_{i2}[\sigma_2(i+1)>M_{i+1,2})\wedge(M_{i1}=M_{i+1,1})\wedge(\sigma_2(i+1)=\sigma(i+1))\wedge(\sigma_1(i+1)=\varepsilon]\vee$$
$$[(M_{i1}[\sigma_1(i+1)>M_{i+1,1})\wedge(M_{i2}[\sigma_2(i+1)>M_{i+1,2})\wedge(\sigma_1(i+1)=\sigma_2(i+1)=\sigma(i+1))]$$

iff

$$(M_0[\sigma_1>M_{k,1})\wedge(M_{i2}[\sigma_2>M_{k,2})\wedge(\sigma_j(1)\circ\sigma_j(2)\circ\cdots\circ\sigma_j(k)=\sigma_j)\wedge(j=1,2)$$

iff

$$(\sigma_j\in L(PN_j))\wedge(\sigma_j=\Gamma_{T\to T_j}(\sigma))\wedge(j=1,2)$$

iff

$$(\sigma\in(\Gamma^{-1}_{T_j\to T}(L(PN_j))\wedge(j=1,2)$$

iff $$\sigma\in\Gamma^{-1}_{T_1\to T}(L(PN_1))\cap\Gamma^{-1}_{T_2\to T}(L(PN_2)).$$

Hence (1) $L(PN)=\Gamma^{-1}_{T_1\to T}(L(PN_1))\cap\Gamma^{-1}_{T_2\to T}(L(PN_2))$.

(2) is true in the same way.

Corollary 2. Let Petri nets be PN_1,PN_2,PN and sets of recognition states be G_1,G_2,G, respectively. If $T_1=T_2$, then

(1) $L(PN)=L(PN_1)\cap L(PN_2)$,

(2) $L_G(PN)=L_{G_1}(PN_1)\cap L_{G_2}(PN_2)$.

The following properties are useful.

Property 1. $L(PN_1OPN_2)=L(PN_2OPN_1)$.

Proof. Follow the interchangeable set operations.

Property 2. $L(PN_1O(PN_1OPN_2))=L(PN_1OPN_2)$.

Proof.

$$L(PN_1O(PN_1OPN_2))=\Gamma^{-1}_{T_1\to T_1\cup(T_1\cup T_2)}(L(PN_1))\cap\Gamma^{-1}_{T_1\cup T_2\to T_1\cup(T_1\cup T_2)}(L(PN_1OPN_2))$$
$$=\Gamma^{-1}_{T_1\to T_1\cup T_2}((PN_1))\cap\Gamma^{-1}_{T_2\to T_1\cup T_2}(L(PN_2))=L(PN_1OPN_2).$$

Property 3. $L(PN_1O(PN_2OPN_3))=L((PN_1OPN_2)OPN_3)$.

Proof.

$$L(PN_1O(PN_2OPN_3))=\Gamma^{-1}_{T_1\to T_1\cup(T_2\cup T_3)}(L(PN_1))\cap\Gamma^{-1}_{T_1\cup T_2\to T_1\cup(T_2\cup T_3)}(L(PN_2OPN_3))$$
$$=\Gamma^{-1}_{T_1\to T_1\cup T_2\cup T_3}(L(PN_1))\cap\Gamma^{-1}_{T_2\to T_1\cup T_2\cup T_3}(L(PN_2))\cap\Gamma^{-1}_{T_3\to T_1\cup T_2\cup T_3}(L(PN_3))$$
$$=\Gamma^{-1}_{T_1\cup T_2\to(T_1\cup T_2)\cup T_3}(L(PN_1OPN_2))\cap\Gamma^{-1}_{T_3\to(T_1\cup T_2)\cup T_3}(L(PN_3))$$
$$=L((PN_1OPN_2)OPN_3).$$

Property 4. $L(PN_1OPN_2)=L(PN_1)\cap\Gamma^{-1}_{T_2\to T_1}(L(PN_2))$, if $T_2\subseteq T_1$.

Proof.

$$L(PN_1OPN_2)=\Gamma_{T_1\to T_1\cup T_2}(L(PN_1))\cap\Gamma^{-1}_{T_2\to T_1\cup T_2}(L(PN_2))$$
$$=\Gamma^{-1}_{T_1\to T_1}(L(PN_1))\cap\Gamma^{-1}_{T_2\to T_1}(L(PN_2))\qquad(\text{Since } T_2\subseteq T_1)$$
$$=L(PN_1)\cap\Gamma^{-1}_{T_2\to T}(L(PN_2)).$$

Property 5. If $\exists\ L\subseteq L(PN_1):\Gamma_{T_1\to T_2}(L)=L(PN_2)$, then

$$L(PN_1)\cap\Gamma^{-1}_{T_2\to T_1}(L(PN_2))=L.$$

Proof. (1)

$$L\subseteq L(PN_1)\wedge\Gamma_{T_1\to T_2}(L)=L(PN_2)$$
$$\Rightarrow\quad L\subseteq L(PN_1)\cap\Gamma^{-1}_{T_2\to T_1}(L(PN_2)).$$

(2) $\forall\ \sigma\in L(PN_1)\cap\Gamma^{-1}_{T_2\to T_1}(L(PN_2))$. If $\sigma\notin L$ and $L\subseteq L(PN_1)\wedge L\subseteq\Gamma^{-1}_{T_2\to T_1}(L(PN_2))$, then

$$\sigma\notin L(PN_1)\wedge\sigma\in\Gamma^{-1}_{T_2\to T_1}(L(PN_2)),$$

thus

$$\sigma \notin L(PN_1) \wedge \Gamma^{-1}_{T_2 \to T_1}(L(PN_2)),$$

contradiction, hence $\sigma \in L$. That is $L(PN_1) \cap \Gamma^{-1}_{T_2 \to T_1}(L(PN_2)) = L$.

From (1) and (2), the consequence is true.

In fact, the properties 1~5 are also true for language $L_G(PN)$. Now we are interested in getting the conditions for preserving liveness in system composition.

Definition 10. Let L_i be a language $T_i, i = 1,2$.

(1) L_1 and L_2 are non-conflict iff

$$\Gamma^{-1}_{T_1 \to T_1 \cup T_2}(\overline{L_1}) \cap \Gamma^{-1}_{T_2 \to T_1 \cup T_2}(L_2) \overline{\Gamma^{-1}_{T_1 \to T_2 \cup T_2}(L_1) \cap \Gamma^{-1}_{T_2 \to T_1 \cup T_2}(L_2)}.$$

(2) L_1 and L_2 are strict non-conflict iff

$$\Gamma^{-1}_{T_1 \to T_1 \cup T_2}(\vec{L}_1) \cap \Gamma^{-1}_{T_2 \to T_1 \cup T_2}(\vec{L})_2 \overrightarrow{\Gamma^{-1}_{T_1 \to T_2 \cup T_2}(L_1) \cap \Gamma^{-1}_{T_2 \to T_1 \cup T_2}(L_2)}.$$

(3) L_1 and L_2 are strict non conflict iff

$$\Gamma^{-1}_{T_1 \to T_1 \cup T_2}(\vec{L}_1) \cap \Gamma^{-1}_{T_2 \to T_1 \cup T_2}(\vec{L})_2 \overrightarrow{\Gamma^{-1}_{T_1 \to T_2 \cup T_2}(L_2) \cap \Gamma^{-1}_{T_2 \to T_1 \cup T_2}(L_2)}.$$

Non-conflict is an important concept in supervisory control theory of DEDS[12]. Here we generalize this concept to the concepts of strict non-conflict and strong non-conflict.

Theorem 5. Let Petri net $PN_i = (P_i, T_i; F_i, M_{0i})$, $i = 1,2$, be all live. If $PN = PN_1 OPN_2$, then

(1) PN is live iff $L(PN_1)$ and $L(PN_2)$ are strong non-conflict;

(2) PN is weak live iff $L(PN_1)$ and $L(PN_2)$ are strict non-conflict.

Proof. (1) PN is live iff $L(PN) = L(\overrightarrow{PN})$ (by Theorem 2) iff

$$\Gamma^{-1}_{T_1 \to T_1 \cup T_2}(L(PN_1)) \cap \Gamma^{-1}_{T_2 \to T_1 \cup T_2}(L(PN_2)) = \overrightarrow{\Gamma^{-1}_{T_1 \to T_2 \cup T_2}(L(PN_1)) \cap \Gamma^{-1}_{T_2 \to T_1 \cup T_2}(L(PN_2))}$$ (by Theorem 4)

iff

$$\Gamma^{-1}_{T_1 \to T_1 \cup T_2}(\overrightarrow{L(PN_1)}) \cap \Gamma^{-1}_{T_2 \to T_1 \cup T_2}(\overrightarrow{L(PN_2)}) = \overrightarrow{\Gamma^{-1}_{T_1 \to T_2 \cup T_2}(L(PN_1)) \cap \Gamma^{-1}_{T_2 \to T_1 \cup T_2}(L(PN_2))}$$ (by Theorem 2)

iff $L(PN_1)$ and $L(PN_2)$ are strong non-conflict.

(2) Derived from (1).

It is time to demonstrate the system liveness control.

Example 3. A Petri net is shown in Fig. 4.

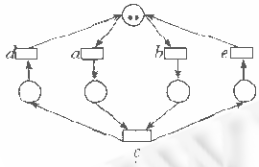

Fig. 4 PN

$$L(PN) = \overline{\{(a(bc(dae \mid ead))^* bcde \mid b(ac(ehd \mid dhe))^* aced)(aa + bb)\}},$$

obviously, $L(PN) \neq L(\overrightarrow{PN})$, hence PN isn't live. Thus we do control on PN.

The principle of the control is that the behaviors of the prime system are preserved under dissolving deadlock.

Control policy 1: The control S_1 is shown in Fig. 5. The closed system $PNOS_1$ is also shown in Fig. 5.

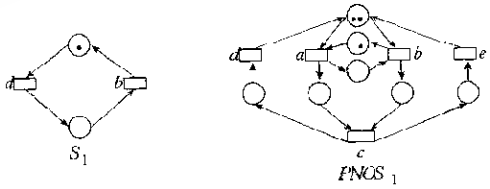

Fig. 5

Obviously, $L(S_1) = \{\overline{(ab)^*}\}$, hence the language of closed system is as follows:

$$L(PNOS_1) = \Gamma^{-1}_{T_{PN} \to T_{PN} \cup T_{S_1}}(L(PN)) \cap \Gamma^{-1}_{T_{S_1} \to T_{PN} \cup T_{S_1}}(L(S_1))$$

$$=L(PN)\bigcap\Gamma^{-1}_{T_{S_1}\to T_{PN}}(L(S_1)) \quad \text{(Since } T_{S_1}\subset T_{PN}\text{, by Property 4)}$$

$$=\{\overline{(a(bc(dae+ead))^*bc(de+ed))^*}\} \quad \text{(by Property 5)}$$

Since $L(\overleftarrow{PNOS_1})=L(\overrightarrow{PNOS_1})$, hence $PNOS_1$ is live.

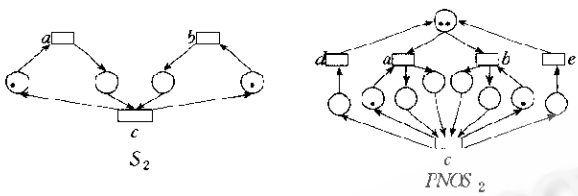

Fig. 6

Control policy 2: The control S_2 is shown in Fig. 6. Its closed system $PNOS_2$ is also shown in Fig. 6.

Obviously, $L(S_2)=\{\overline{((a\parallel b)c)^*}\}$, hence the language of closed system $PNOS_2$ is as follows:

$$L(PNOS_2)=L(PN)\bigcap\Gamma^{-1}_{T_{S_2}\to T_{PN}}(L(S_2))=\{\overline{((abc-bac)((dbe+edb)ac+(ead+dae)bc)^*(de+ed))^*}\}.$$

Since $L(\overleftarrow{PNOS_2})=L(\overrightarrow{PNOS_2})$, hence $\overleftarrow{PNOS_2}$ is also live.

It is easy to see:

$$\{\overline{(a(bc(dae+ead))^*bc(de+ed))^*}\}\subset\{\overline{((abc+bac)((dbe+ebd)ac-(ead+dae)bc)^*(de+ed))^*}\},$$

hence $L(PNOS_1)\subset L(PNOS_2)$.

In some sense the control policy 2 is better than the control policy 1.

4 Conclusion

The testing method of the liveness of Petri nets is studied based on language in this paper. The relations of some important properties between Petri net and DEDS are pointed out. The deadlock of system is dissolved by synchronous composition.

References:

[1] Bermond, J. C., Memmi, G. A graph theoretical characterization of minimal deadlock. In: The Proceedings of the 4th European Workship on Application and Theory of Petri Nets. Toulouse, France, 1983. 1~11.

[2] Ottein, H. A. I. Automatic deadlock analysis of parallel programs. International Computer Symposium, 1977,8(2):209~215.

[3] Murata, T., Shenker, B., Shartz, S. M. Detection of ada static dead-locks using Petri net invariants. IEEE Transactions on Software Engineering, 1989,SE-3:314~325.

[4] Commoner, F., Holt, A., Even, S., *et al*. Marked directed graphs. Journal of Computer and System Science, 1977,14(5):511~523.

[5] Lautenbach, K. Liveness in Petri net. ISF-Report 2, GMD, ST. Augustin, F. R. G., 1975. 1~75.

[6] Hack, M. Decidability questions for Petri nets [Ph. D. Thesis]. Massachusetts Institute of Technology, Cambridge, Massachusetts, 1975.

[7] Xu, A. G., Wu, Z. H. Analysis of liveness for weighted T graphs. Journal of Software, 1993,4(6):16~21 (in Chinese).

[8] Wu, Z. H. Analysis and implementation of liveness and fairness of bounded Petri nets. Chinese Journal of Computers, 1989,12(4):267~278 (in Chinese).

[9] Lu, W. M. Discussion of the means of token for live net. Science in China (Series A), 1988,17(8):776~784 (in Chinese).

[10] Lu, W. M., Lin, C. The Petri net model of production system. Acta Automation, 1993,19(3):290~299 (in Chinese).

[11] Jiang, C. J., Wu, Z. H. Net operations. Journal of Computer Science & Technololgy, 1992,7(4):333~344.

[12] Jiang, C. J. Petri net dynamic invariance. Science in China (Series E), 1997,27(6):605~611 (in Chinese).
[13] Wonham, W. M., Remadge, P. J. Modular supervisory control of discrete-event systems. Mathematics Control Signal System, 1998,1(1):13~30.
[14] Petri, C. A. Kommunikation mit automatien, Bonn. Institute fur Instru-mentelle Mathematik, Schriften des IIM Nr. 2. 1952.
[15] Best, E., Fernandez, C. Notations and terminology on Petri net theory. Petri net Newsletter, 1986,23:21~46.
[16] Giua, A., Cesare, F. D. Petri net structural analysis for supervisory control. IEEE Transactions RA-2, 1994.
[17] Valmari, A. Compositional Analysis with Place-bordered Subnets. LNCS, 815, 1994,815.
[18] Cheung, S. C., Kramer, J. Enhancing compositional reachability analysis with context constraints. ACM on Software Engineering Notes, 1993,5.
[19] Portinale, L. Exploiting T-invariant analysis in diagnostic reasoning on a Petri net model. LNCS, 1994,815.

附中文参考文献:

[7] 许安国,吴哲辉.加权T图的活性分析.软件学报,1993,4(6):16~21.
[8] 吴哲辉.有界公平Petri网的活性与公平性分析与实现.计算机学报,1989,12(4):267~278.
[9] 陆维明.活网的token意义判定.中国科学(A辑),1988,18(8):776~784.
[10] 陆维明,林闯.生产系统的Petri网模型.自动化学报,1993,19(3):290~299.
[12] 蒋昌俊.Petri网的动态不变性.中国科学(E辑),1997,7(6):605~611.

基于Petri网语言的并发系统性质研究

蒋昌俊[1,2], 陆维明[3]

[1](同济大学 计算机科学与工程系,上海 200092);
[2](山东科技大学 计算机科学与技术系,山东 泰安 271019);
[3](中国科学院 数学研究所,北京 100080)

摘要: 给出Petri网弱活性(无死锁)与活性的两个语言刻画,讨论了同步合成Petri网的语言性质.基于Petri网语言,给出了判定Petri网活性的充分必要条件.同时研究了Petri网同步合成过程中活性保持问题,给出保持活性的充分必要条件.这些结果为讨论网的活性测试和控制提供了形式语言的方法.

关键词: Petri网;并发系统;活性;同步合成;测试;控制
中图法分类号: TP301 **文献标识码**: A