基于网语言的 Ada 程序局部性质的分析和验证*

丁志军[1], 蒋昌俊[1,2]

[1](山东科技大学 信息科学与工程学院,山东 泰安 271019);
[2](同济大学 计算机科学与工程系,上海 200092)
E-mail: cjjiang@online.sh.cn
http://www.sdust.edu.cn

摘要: 旨在研究利用网语言讨论 Ada 程序性质和由此而引起的 Ada 网的状态爆炸问题.研究了 Ada 网的同步合成与分解,讨论了它们的语言性质,并利用这一结果分析和验证了 Ada 程序的安全性和活性,从而为复杂的 Ada 程序的分析与验证提供了一个新的有效途经.

关 键 词: Ada 网;同步合成;同步分解;网语言;分析;验证
中图法分类号: TP311 文献标识码: A

随着在各个领域中的广泛应用,并发程序已越来越引起人们的兴趣和重视.然而,由于并发程序所具有的并发特性,使得程序语义具有不确定性,从而使这方面的研究较顺序程序要困难得多,因此,相关的程序规范、设计与分析、验证方法已成为计算机理论界研究的重点与热点问题之一.近年来,许多学者利用并发分析工具 Petri 网对并发程序进行建模和分析[1],特别是对 Ada 任务程序的分析和构造[2,3].其中,基于可达性的分析方法受到了广泛的关注,原因是:(1) Petri 网的动态性质与并发程序的性质具有相似性,如死锁、活性等,而利用 Petri 网的可达图,可以很好地分析这些性质.(2) Petri 网语言的语义与我们讨论并发程序时所假设的语义都是交错语义,即“事件并行发生的效果和以任一顺序串行发生的效果一样”,这使得基于可达性的分析方法在描述并发程序时不会改变原程序的语义.而且基于可达性的分析方法由于实际上是给出了系统的运行轨迹和各可达状态,因此可以支持系统多种性质的分析和验证,包括安全性和活性(与 Petri 网的安全性和活性不同).(3) 基于可达性的分析手段很容易结合其他分析方法,如时序逻辑、进程代数等,从而提供更为有效的分析和验证方法.然而,基于可达性的分析方法的能力受到了状态爆炸的限制.已经证明,Ada 任务程序的可达状态随着任务的数目而呈指数增长,这就造成了我们利用可达性的分析手段进行分析和验证的困难.

为了解决状态爆炸问题,一些用于改善基于可达性的分析方法的复杂性的理论已经提出来了.其中,有些学者[4]利用网化简理论,删除或结合一些不必要的库所或变迁,仅保留反映程序性质的重要库所和变迁,从而减少可达状态的数目;另外一些理论[5]着重讨论程序中的并发部分,而避免探讨全部状态,从而降低了分析的复杂性;还有一些理论通过网分解理论,将较复杂的网系统通过分解,拆分成若干子网,分别加以讨论,最终得到原网系统的性质[6].文献[7]则结合上述各种方法进行网化简,并认为各种方法的结合使用能够更有效地降低分析复杂性.文献[8]对上述网化简方法进行了总结及评估,并给出了相关的测试数据.

文献[6]利用同步合成运算,将一个复杂的 Ada 网分解成若干个易于分析的子网,然后构造了分层可达图

* 收稿日期: 2001-01-15; 修改日期: 2001-03-19
基金项目: 国家自然科学基金资助项目(69973029;69933020);国家高技术研究发展计划资助项目(2001AA413020);国家重点基础研究发展规划973资助项目(G1998030604);国家杰出青年科学基金资助项目(60125205);教育部优秀青年教师教学科研奖励计划资助项目;全国优秀博士论文作者专项基金资助项目(199934);上海市科技发展计划重点基础研究基金资助项目(02DJ14064)
作者简介: 丁志军(1974－),男,江苏泰兴人,硕士,主要研究领域为 Petri 网理论,并行处理;蒋昌俊(1962－),男,安徽安庆人,博士,教授,博士生导师,主要研究领域为并发理论与并行处理,Petri 网理论,模型验证,模糊推理.

HRG,即具有根节点(原网可达图)、若干叶子节点(经过化简后的子网可达图)及其他非叶子节点(合成运算)的树状结构,可以完整地表示原网与子网可达图的构造关系,并利用分层可达图 HRG 进行了可达性分析和死锁检测.

本文借鉴上述思想,利用 Petri 网的同步合成运算,将一个复杂的 Ada 网分解成若干个易于分析的子网,并利用子网可达图及同步合成运算的性质求出原 Ada 网的可达图.并据此求出网语言,再通过其对程序性质进行分析.显然,求解网语言与求解可达图是等价的,但二者的内涵不同,状态图侧重于表述系统的运行效果,而网语言则反映了系统的运行轨迹,刻画了系统的行为动作,更有利于描述系统的动态行为.这对于分析并发程序性质是有帮助的.另外,利用网语言分析程序性质的工作鲜见论述,我们期盼在这方面能够作一些有益的探索.

1 同步合成运算的概念及性质

有关 Petri 网的基本概念、术语及性质请见文献[9,10],这里不再介绍.

近年来,组合化的设计思想日益得到人们的重视,尤其是对复杂系统的设计与分析,更能体现这一思想的重要性.文献[11~13]引入了同步合成运算,并研究了同步合成运算对 Petri 网的动态不变性,深入探讨了同步合成的动态性质:行为不变性和行为相关性,并应用于实际系统的分析与设计中.文献[14]对 PVM 程序进行建模,并利用同步合成运算,通过子程序耦合出大的程序,并分析其相关性质.

定义 **1**. 设 $N_i=(S_i,T_i;F_i),i=1,2$ 是两个网, $N_T=(S,T;F)$ 为网 N_1 与 N_2 的同步合成网,当且仅当

(1) $S=S_1\cup S_2,S_1\cap S_2=\varnothing$;

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

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

设 M_{0i} 是网 N_i 的初始标识,相应的 Petri 网记为 $\Sigma_i=(N_i,M_{0i}),i=1,2$.若 $M_0(s)=M_{0i}(s),s\in S_i$,则称 $\Sigma_T=(N_T,M_0)$ 为同步合成 Petri 网.

在上述文献中仅定义了两个子网的同步合成运算,而在讨论大系统时,这个概念是远远不够的.这里,我们将这个定义推广至 n 个子网,有如下定义.

定义 **2**(***n***-同步合成). 设 $N_i=(S_i,T_i;F_i),i=1,...,n$ 是 n 个网, $N_T^n=(S^n,T^n;F^n)$ 为网 $N_1,N_2,...,N_n$ 的 n-同步合成网,当且仅当

(1) $S^n=S_1\cup S_2\cup...\cup S_n,\forall i,j,i\neq j,S_i\cap S_j=\varnothing$;

(2) $T^n=T_1\cup T_2\cup...\cup T_n,\forall i$,至少存在一个 $j,i\neq j$,有 $T_i\cap T_j\neq\varnothing$;

(3) $F^n=F_1\cup F_2\cup...\cup F_n$.

设 M_{0i} 是网 N_i 的初始标识,相应的 Petri 网记为 $\Sigma_i=(N_i,M_{0i}),i=1,...,n$.若 $M_0^n(s)=M_{0i}(s),s\in S_i$,则称 $\Sigma_T^n=(N_T^n,M_0^n)$ 为 n-同步合成 Petri 网.

定义 **3**(***R***-图). 一个 R-图是五元组 (V,E,T,L,v_0) ,其中 (V,E) 是一个带标号的有限有向图, v_0 为根节点,有向图的每条边用变迁集合中的元素作为标号,映射 $L:E\to T$.

显然,Petri 网的可达图是一个 R-图.若干个 R-图可以合并为一个 R-图.

定义 **4**(***G***-合并). 设 $RG_1,RG_2,...,RG_n$ 是 n 个 R-图,其中 $RG_i=(V_i,E_i,T_i,L_i,v_{0i})$, $i=1,...,n$,则 n 个 R-图 $RG_1,RG_2,...,RG_n$ 的 G-合并($RG_1,RG_2,...,RG_n$)仍然是一个 R-图 $RG=(V,E,T,L,v_0)$,其中 V,E,T,L,v_0 分别定义如下:

(1) $V\subseteq V_1\times V_2\times...\times V_n$;

(2) $T=T_1\cup T_2\cup...\cup T_n$;

(3) $v_0=(v_{01},v_{02},...,v_{0n})$;

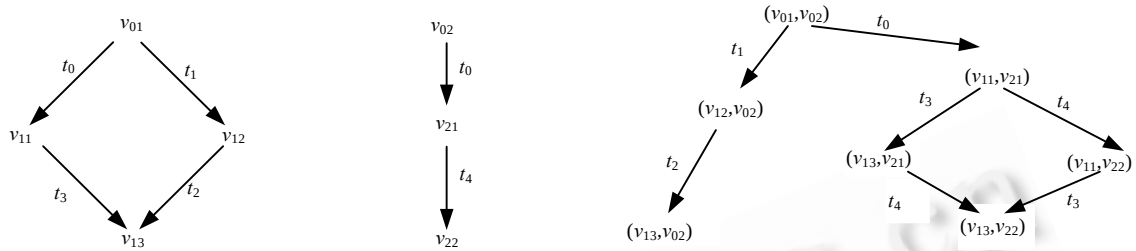
(4) V,E,L 递归定义如下:

(a) $v_0\in V$;

(b) 对于 $(v_1',v_2',...,v_n')\in V\times V_2\times...\times V_n$,若存在 $v=(v_1,v_2,...,v_n)\in V$,且存在 $t\in T_1\cup T_2\cup...\cup T_n$,使得对任意的 $i\in\{1,2,...,n\},(v_i,v_i')$,有 $L_i(v_i,v_i')=t$,或者 $(t\notin T_i)\wedge(v_i=v_i')$,那么, $v'=(v_1',v_2',...,v_n')\in V,(v,v')\in E,L(v,v')=t$.

(c) V,E,L 不包含除(a)、(b)以外所得到的其他任意元素.

例 1:设 $RG_i=(V_i,E_i,T_i,L_i,v_{0i}),i=1,2$ 是两个 R-图(如图 1(a)、(b)所示),其中 $t_0\in T_1\cap T_2$,则其 G-合并(RG_1,RG_2,…,RG_n)是一个 R-图 $RG=(V,E,T,L,v_0)$ (如见图 1(c)所示).



(a) RG_1 (b) RG_2 (c) G-Combine(RG_1,RG_2)

Fig.1

图 1

显然,上述定义及例 1 所采取的“分而治之”策略可以有效地降低求解可达图的复杂性,在本文中,我们正是利用上述方法求解 Ada 网的可达图的.

对于 n-同步合成 Petri 网的性质及语言,总可以通过两个子网的同步合成运算逐步迭代得出.下面,我们将刻画 n-同步合成 Petri 网的语言性质,同时,通过该定理的证明展示如何利用两个子网的同步合成运算迭代得到 n-同步合成 Petri 网性质.

定理 1. 设 $\Sigma_i=(S_i,T_i;F_i,M_{0i}),i=1,2$ 是两个 Petri 网,则同步合成 Petri 网 $\Sigma_T=(S,T;F,M_0)$ 的语言 $L(\Sigma_T)=\Gamma^{-1}_{T_1\to T}(L(\Sigma_1))\cap\Gamma^{-1}_{T_2\to T}(L(\Sigma_2))$.

证明见文献[12].

推论. 设 $\Sigma_i=(S_i,T_i;F_i,M_{0i}),i=1,...,n$ 是 n 个 Petri 网,则 n-同步合成 Petri 网 $\Sigma_T^n=(S^n,T^n;F^n,M_0^n)$ 的语言有

$$L(\Sigma_T^n)=\Gamma^{-1}_{T_1\to T^n}(L(\Sigma_1))\cap\Gamma^{-1}_{T_2\to T^n}(L(\Sigma_2))\cap...\cap\Gamma^{-1}_{T_n\to T^n}(L(\Sigma_n)).$$

证明:利用归纳法证明.

(1) 设 n=2 时,由定理 1 知,$L(\Sigma_T)=\Gamma^{-1}_{T_1\to T^2}(L(\Sigma_1))\cap\Gamma^{-1}_{T_2\to T^2}(L(\Sigma_2))$,即归纳基础成立.

(2) 假设 $n=k$ 时结论成立,即设 $\Sigma_i=(S_i,T_i;F_i,M_{0i}),i=1,...,k$ 是 k 个 Petri 网,则 k-同步合成 Petri 网 $\Sigma_T^k=(S^k,T^k;F^k,M_0^k)$ 的语言有

$$L(\Sigma_T^n)=\Gamma^{-1}_{T_1\to T^k}(L(\Sigma_1))\cap\Gamma^{-1}_{T_2\to T^k}(L(\Sigma_2))\cap...\cap\Gamma^{-1}_{T_k\to T^k}(L(\Sigma_k)).$$

下面需要证明当 $n=k+1$ 时,结论成立.

根据假设可知 k-同步合成 Petri 网 Σ_T^k 的语言为 $L(\Sigma_T^k)$,$k+1$ 子网语言是 $L(\Sigma_{k+1})$,则由定理 1 可知,$k+1$-同步合成 Petri 网 Σ_T^{k+1} 的语言为

$$\begin{aligned}L(\Sigma_T^{k+1})&=\Gamma^{-1}_{T^k\to T^{k+1}}(L(\Sigma_T^k))\cap\Gamma^{-1}_{T_{k+1}\to T^{k+1}}(L(\Sigma_{k+1}))\\&=\Gamma^{-1}_{T^k\to T^{k+1}}(\Gamma^{-1}_{T_1\to T^k}(L(\Sigma_1))\cap\Gamma^{-1}_{T_2\to T^k}(L(\Sigma_2))\cap...\cap\Gamma^{-1}_{T_k\to T^k}(L(\Sigma_k)))\cap\Gamma^{-1}_{T_{k+1}\to T^{k+1}}(L(\Sigma_{k+1}))\\&=\Gamma^{-1}_{T_1\to T^{k+1}}(L(\Sigma_1))\cap\Gamma^{-1}_{T_2\to T^{k+1}}(L(\Sigma_2))\cap...\cap\Gamma^{-1}_{T_k\to T^{k+1}}(L(\Sigma_k))\cap\Gamma^{-1}_{T_{k+1}\to T^{k+1}}(L(\Sigma_{k+1})).\end{aligned}$$

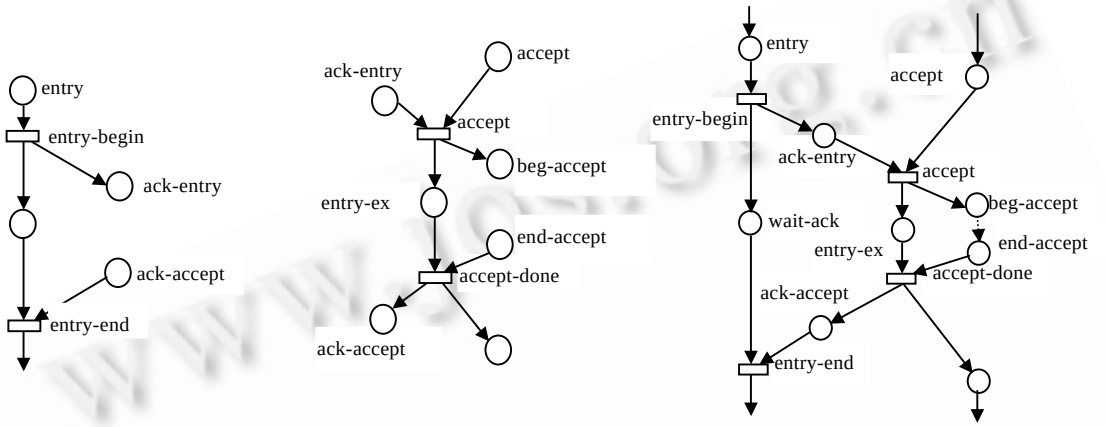
即 $n=k+1$ 时,结论成立.

综合(1),(2)两步,命题得证. □

2 Ada 程序的 Petri 网模型

文献[1]定义了 Ada 网,所谓 Ada 网,即指利用一系列转换规则将 Ada 任务程序转换成的 Petri 网模型.一个 Ada 网模型模拟了若干个通信有限状态自动机.Ada 任务中的通信是建立在汇聚机制上的,每个状态表示一个

任务的局部控制流,而通信的交互过程则通过连接不同状态机的节点来表示.同时,if-else,case 和 loop 等控制流结构也用网结构加以模拟并反映其对通信的影响.产生 Ada 网的转换规则主要是一系列对应 Ada 任务程序语句的 Petri 网模板,关于转换的细节,见文献[1~3].这里仅介绍其中通信部分的转换规则,而对于 Ada 网的同步合成运算也正是基于该转换规则.图 2(a)是一个 Ada 任务入口请求的 Petri 网模板,表示一个请求者执行 entry 语句调用的过程,根据入口调用语句的语义可知,其执行结果为本进程挂起,等待服务者的响应,以完成汇聚.图 2(b)是一个 Ada 任务接受语句的 Petri 网模板,描述了服务者执行 accept 语句的过程,在服务者执行到自己的接受语句时,若请求者还没有发出对应的入口调用,那么,服务者任务挂起,等待入口调用的发出.图 2(c)则反映了 Ada 任务的汇聚机制:一个任务中的入口请求语句与另一任务中的接受语句共同完成.



(a) (b) (c)

Fig.2

图 2

从图 2 中可以看出,Ada 任务程序的汇聚机制是通过外部库所 ack-entry,entry-ex,ack-accept 和变迁 accept,accept-done 在两个任务之间进行通信实现的.其中,变迁 accept,accept-done 起到了同步的作用,准确地表达了 Ada 程序汇聚机制的语义.我们对 Ada 网进行的同步合成运算也正是将这两个变迁作为共享变迁进行操作的.

下面对文献[4,7,8]引用的自动加油站问题进行 Ada 网模型建模,其源程序及 Ada 网模型(如图 3 所示)如下:

例 2:自动加油站问题.

```
1 task body Customer is
2 begin
3 loop
4   Operator.Prepay;
5   Pump.Start;
6   Pump.Finish;
7   accept Change;
8 end loop
9 end Customer
10 task body Pump is
11 begin
12 loop
13   accept Activate;
14   accept Start;
15   accept Finish do
16     Operator.Charge;
17   end Finish
18 end loop
19 end Pump
20 task body Operator is
21 begin
22 loop
23   select
24     accept Prepay do
25       Pump.Activate;
26     end Prepay;
27   or
28     accept Charge do
29       Customer.Charge;
30     end Charge;
31   end select;
32 end loop;
33 end Operator
```

研究 Ada 网本身的特性,可以更好地把握并发程序的性质,并简化网分析的复杂性.这在文献[1~4]均有讨论,归纳起来,其性质有如下几条:

(1) Ada 网是安全的 Petri 网;

(2) Ada 网中对应着程序语句的节点(库所和变迁)至多为线性数量个;

(3) Ada 网的初始标识仅在表示任务开始的库所中含有托肯;

(4) If,case 等确定性语句在网结构中表现为分支结构;

(5) 任务内部局部控制流中的确定性语句若不含通信语句,则不反映在 Ada 网中.

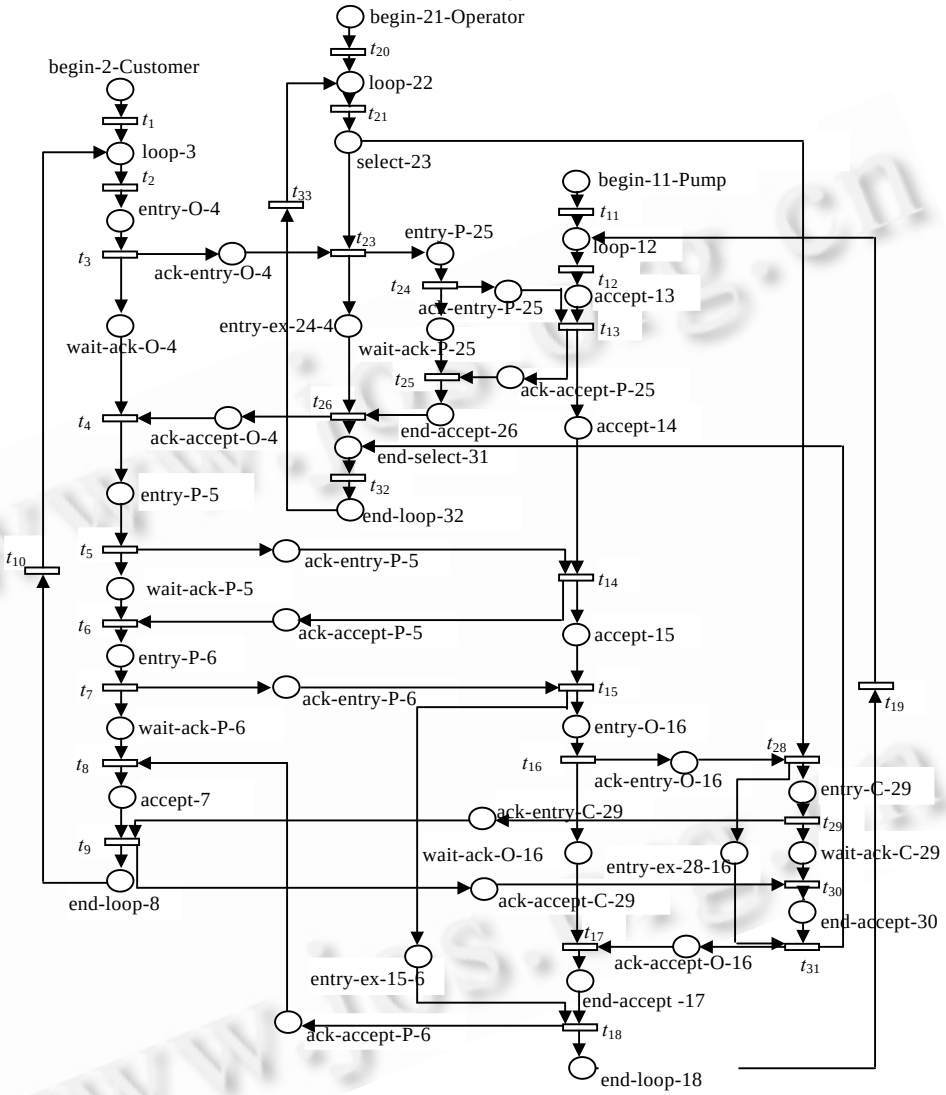


Fig.3 Ada net of source program

图 3 源程序的 Ada 网模型

综上所述,在 Ada 任务程序中,任务内部的局部控制流实际上对应着一段顺序程序,其性质的分析和验证可用传统的顺序程序分析方法进行解决.而 Ada 任务程序执行语义的不确定性,是由于多个进程的并发执行和任务间的消息传递的不确定性引起的.通过后继章节的讨论可知,其中汇聚机制的同步起到了关键作用,反映在 Ada 网中,即相关的同步变迁 accept,accept-done.

3 Ada 网的同步合成运算及其可达图求解

图 4(a)和图 4(b)是对图 2(c)进行同步合成运算所得到的两个子网.

事实上,通过同步合成运算,我们将通信部分分解为隶属于两个子网(任务)的顺序流,因此,利用同步合成运

算进行网分解后所得到的各个子网,分别对应着源程序中的各个任务.这样,通过独立地考察各个子网的可达图和性质,即对各个任务的局部控制流进行分析,考察相关性质,然后再利用同步合成运算的性质对原 Ada 网进行分析.

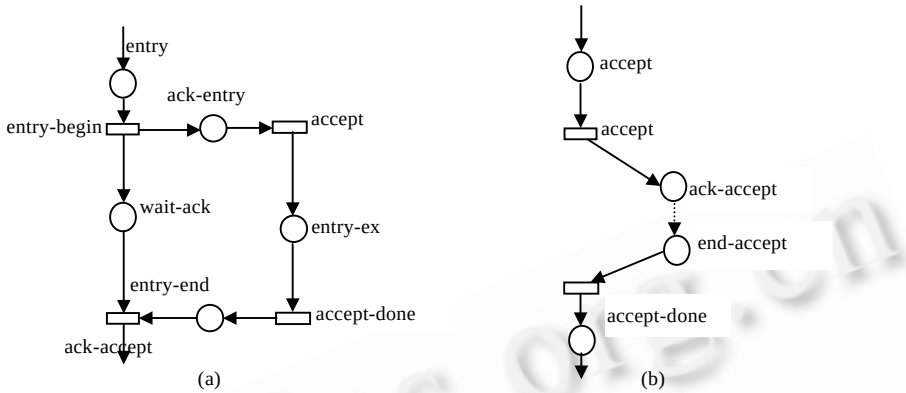


Fig.4
图 4

图 5 就是对图 3 所描绘的 Ada 网模型进行同步合成运算后所得的 3 个子网模型.图 6 所表示的就是 3 个子网的可达图,其中,我们进行了简单的化简,即将可达图上的连通分支(不包含共享变迁)约简为一个节点和一条边,很明显,这种化简不会改变原网的语言,但可以有效地缩减可达状态,进一步降低求解可达图的复杂性.具体示例如图 6 所示.

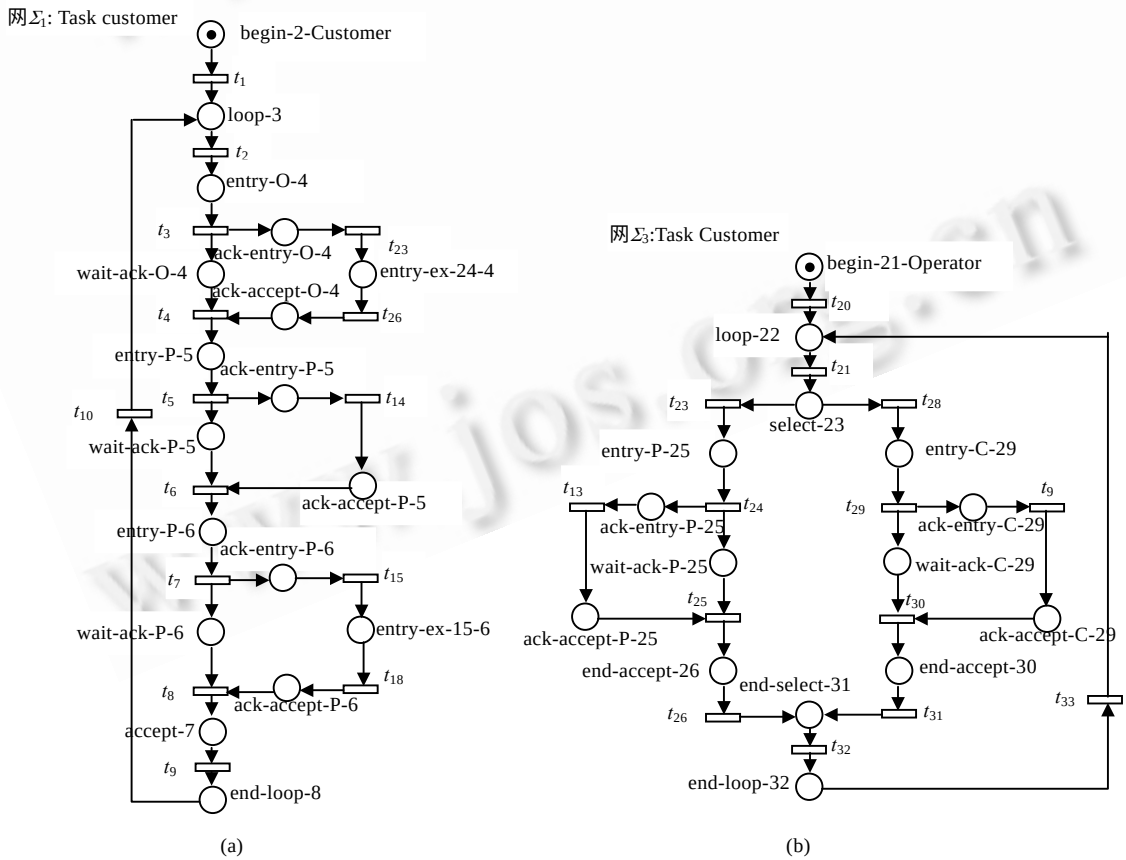


Fig.5
图 5

网Σ_2:Task Pump

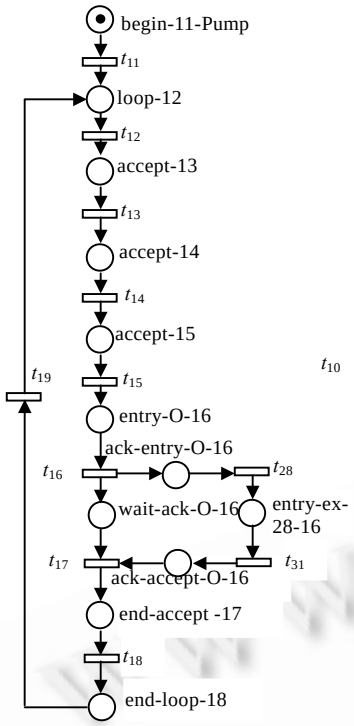


Fig.5(c)
图 5(c)

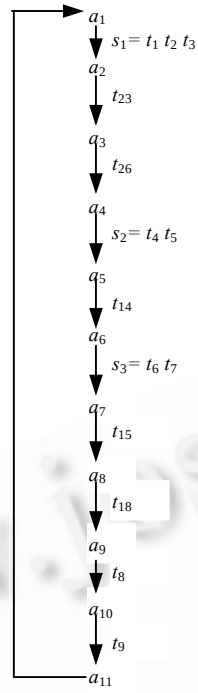


Fig.6 (a) $RG(\Sigma_1)$
图 6 (a) $RG(\Sigma_1)$

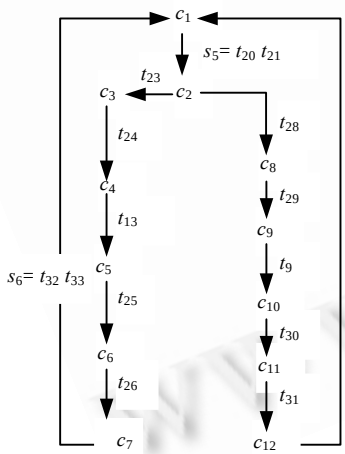


Fig.6(b) $RG(\Sigma_3)$
图 6(b) $RG(\Sigma_3)$

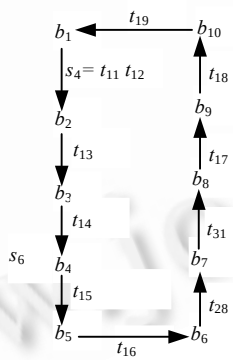


Fig.6(c) $RG(\Sigma_2)$
图 6(c) $RG(\Sigma_2)$

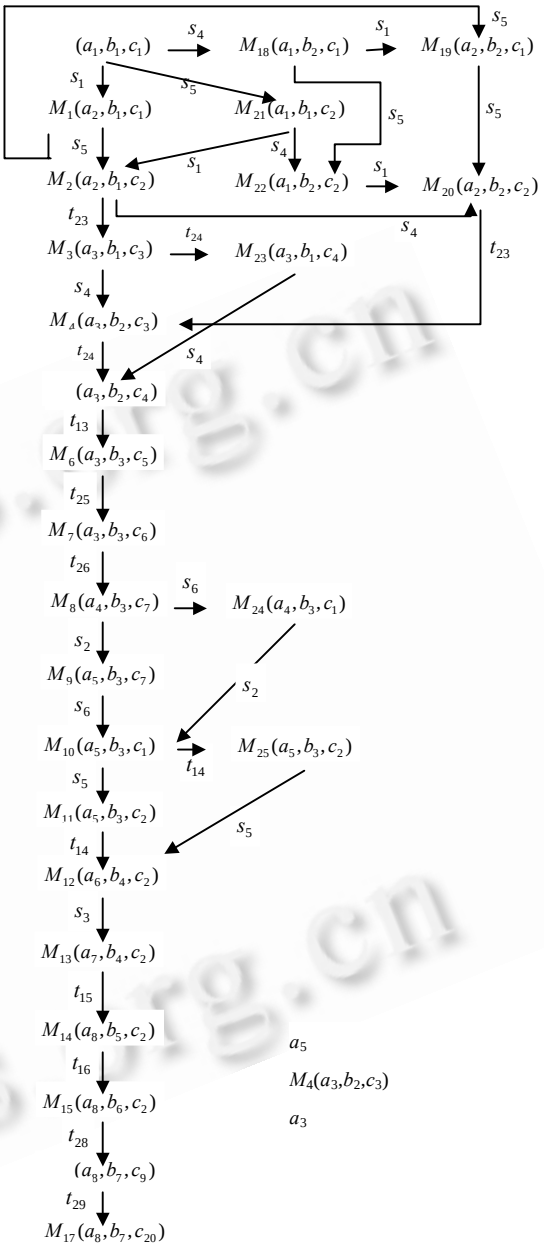


Fig.7
图 7

根据图 6 的 3 个子图,利用定义 4 和例 1 所叙述的方法,我们可以求出原 Ada 网的可达图,如见图 7 所示.

网语言中的每一元素在可达图 $RG(\Sigma)$ 中表现为从初始标识 M_0 起始的有向路上所对应的变迁引发序列.基于这一点,我们能够利用表达式 $M_i \rightarrow_\alpha M_j$ 表示存在有向路径由 $M_i \sim M_j$,其中路径上标号为 α.由此,我们可以借用类似自动机产生语言的方法推导出网语言的形式.推导过程如下:

$$M_0 = s_1M_1 + s_5s_1M_2 + (s_5s_4s_1 + s_4(s_1s_5 + s_5s_1))t_{23}M_4, \quad (1)$$

$$M_1 = s_5M_2 + s_4s_5t_{23}M_4. \quad (2)$$

综合式(1)、(2),有

$$M_0 = (s_1s_5 + s_5s_1)M_2 + (s_5s_4s_1 + s_1s_4s_5 + s_4(s_1s_5 + s_5s_1))t_{23}M_4. \quad (3)$$

又因为

$$M_2 = t_{23}M_3 + s_4t_{23}M_4,$$

$$M_3 = s_4M_4 + t_{24}s_4M_5,$$

$$M_4 = t_{24}M_5.$$

结合式(3)及上述各式,得到

$$M_0 = (s_1s_5 + s_5s_1)t_{23}s_4t_{24} + ((s_1s_5 + s_5s_1)s_4 + s_5s_4s_1 + s_1s_4s_5 + s_4(s_1s_5 + s_5s_1))t_{23}t_{24}M_5$$

$$M_0 = ((s_1s_5 + s_5s_1)s_4 + s_5s_4s_1 + s_1s_4s_5 + s_4(s_1s_5 + s_5s_1))t_{23}t_{24}M_5 + (s_1s_5 + s_5s_1)t_{23}s_4t_{24}M_5$$

$$= ((s_1 \otimes s_5)t_{23}s_4t_{24} + (s_1 \otimes s_4 \otimes s_5)t_{23}t_{24})M_5,$$

其中,符号 $\otimes$ 表示混排.

同理,可知

$$M_5 = t_{13}t_{25}t_{26}(s_2 \otimes s_6)M_{10},$$

$$M_{10} = (t_{14} \otimes s_5)M_{12},$$

$$M_{12} = s_3t_{15}t_{16}t_{28}t_{29}M_{17}.$$

综合上式,得到

$$M_0 = ((s_1 \otimes s_5)t_{23}s_4t_{24} + (s_1 \otimes s_4 \otimes s_5)t_{23}t_{24}) \cdot t_{13}t_{25}t_{26}(s_2 \otimes s_6)(t_{14} \otimes s_5)s_3t_{15}t_{16}t_{28}t_{29}M_{17}$$

将 s_i 用相应的变迁序列代替,我们就可得到

$$M_0 = ((t_1t_2t_3 \otimes t_{20}t_{21})t_{23}t_{11}t_{12}t_{24} + (t_1t_2t_3 \otimes t_{11}t_{12} \otimes t_{20}t_{21})t_{23}t_{24}) \cdot t_{13}t_{25}t_{26}(t_4t_5 \otimes t_{32}t_{33})(t_{14} \otimes t_{20}t_{21})t_6t_7t_{15}t_{16}t_{28}t_{29}M_{17}$$

由此,我们可以得到原 Ada 网语言.

定理 **2**. 设 Ada 网 $\Sigma = (S,T;F,M_0)$,如图 4 所示.其中, $M_0(S') = 1$ 并且对任意的 $s \notin S'$,有 $M_0(s) = 0$, $S' = \{begin-2-Customer, begin-11-Pump, begin-21-Operator\}$,则网语言 $L(\Sigma)$ 为

$$L(\Sigma) = ((t_1t_2t_3 \otimes t_{20}t_{21})t_{23}t_{11}t_{12}t_{24} + (t_1t_2t_3 \otimes t_{11}t_{12} \otimes t_{20}t_{21})t_{23}t_{24}) \cdot t_{13}t_{25}t_{26}(t_4t_5 \otimes t_{32}t_{33})(t_{14} \otimes t_{20}t_{21})t_6t_7t_{15}t_{16}t_{28}t_{29}.$$

以上,我们通过网的同步合成运算和子网可达图化简方法,有效地缩减可达状态,降低求解原网可达图的复杂性,并根据原网可达图形式化地得到了原网语言.下面,我们将利用网语言分析和验证系统性质.

4 利用网语言分析 Ada 程序性质

正如我们所熟知的,网语言能够准确刻画实际系统的动态行为,模拟系统的动作,给出了一个网的语言,也就等于得到了实际系统的运行轨迹,从而可以有效地分析系统性质.目前,较多采用的分析方法是利用网的可达图进行系统分析,但由于其侧重于描述系统的可达状态,因此,无法直接利用其分析系统的动态性质,特别是系统的活性性质(不同于 Petri 网的活性),所以,现在利用可达图分析并发程序的性质,主要集中于程序的死锁分析,但较少涉及程序的活性性质.本文利用网语言不仅可以有效地分析死锁等的安全性质,同时,可利用其分析程序的活性性质.

4.1 Ada程序的安全性

所谓安全性,是指某些状态永远不能发生,即程序永远不能进入不期望进入的状态.程序的安全性是第一位的,它关心的是可能性问题,要求所有可能的状态均满足一定的性质.在安全的前提下才能讨论活性.

定义 **5**. 一个 Ada 任务程序成为静态可执行的,当且仅当其对应的 Ada 网是弱活的[15].

满足定义 5 的程序总会执行某些语句,而不会停止下来,这就避免了死锁的发生.因此,程序的静态可执行性质属于安全性的范畴.当然,这也并不能保证所有的语句都能被执行,即程序仅反复执行某一部分语句,而无法得到进展,这也就是所谓的活锁.这将在后面加以叙述.

文献[11]研究了基于 Petri 网语言的活性刻画.在此,我们应用其给出程序静态可执行性质的判据.在讨论之前,先给出一个记号: $\forall \sigma \in T^*$,记 $\&(\sigma)=\{x \mid x$是σ中的字符$\}$.

定义 **6**. 设 $L \subset T^*$ 是一个语言,令

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

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

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

则称 $pref(L)$, $spref(L)$ 和 $stpref(L)$ 分别是 L 的闭语言、严格闭语言和强闭语言.

定理 **3**. 设 Petri 网 $PN=(S,T;F,M_0)$,则 PN 是弱活的充分必要条件是

$$L(PN)=spref(L(PN)).$$

证明见文献[11].

推论 **3.1**. 设 Ada 任务程序所对应的 Ada 网 $\Sigma=(S,T;F,M_0)$,其中 $M_0(S')=1$ 并且对任意的 $s \notin S'$,有 $M_0(s)=0$,则 Ada 任务程序是静态可执行的充分必要条件是

$$L(\Sigma)=spref(L(\Sigma)).$$

当然,程序的静态可执行性质条件要比程序的无死锁性强得多,一个正常终止的程序是不满足静态可执行性质的.因此,有必要讨论程序发生死锁的条件,便于我们分析程序的性质.这里需要强调的是,任何并发程序与顺序程序一样,都是单出口的,只是由于对于实时相应程序,如上面提到的自动加油站问题,终止性不具备任何意义,在建模中才没有描述单出口语句,但这并不代表其不是单出口的.同时,在 Ada 网语义中,变迁描述了语句的运行,因此,我们有下述定理.

定理 **4**.设 Ada 任务程序所对应的 Ada 网 $\Sigma=(S,T;F,M_0)$,其中 $M_0(S')=1$ 并且对任意的 $s \notin S'$,有 $M_0(s)=0$,则 Ada 任务程序发生死锁的充分必要条件是 $\exists \sigma \in L(\Sigma)$,其中 $\sigma=\sigma'' \circ t \wedge \forall \sigma'$,有 $\sigma \circ \sigma' \notin L(\Sigma)$,并且若 $\exists t_{end}$,有 $t \neq t_{end}$, t_{end} 表示单出口语句.

证明:结论是显然的, $\sigma=\sigma'' \circ t \wedge \forall \sigma'$,有 $\sigma \circ \sigma' \notin L(\Sigma)$,并且若 $\exists t_{end}$,有 $t \neq t_{end}$,表明程序执行有限步后,非正常地终止下来,即发生死锁.反之亦然. □

定理 2 所描述的语言满足定理 4 的条件,即该语言中的元素(变迁序列)均为有限长度,利用定理 4,易见例 2 的 Ada 任务程序发生了死锁.

基于定理 4,Ada 任务程序的死锁分析算法描述如下:

算法 **1**(死锁分析). Deadlock-Analysis.

输入: $\sigma \in L(\Sigma)$ 并且 $\forall \sigma'$,有 $\sigma \circ \sigma' \notin L(\Sigma)$;

输出:若 σ 是发生死锁的序列,则输出 False,否则,输出 True.

```
begin
while
 读入 σ 的一个字符 t
 if  t = t_end  then {程序执行了单出口语句,正常终止}
    输出 True;并跳出循环.
 elseif  t 为空字符  then {程序执行了有限步后,非正常终止,即发生死锁}
    输出 False,并跳出循环.
 endif
enddo
endbegin
```

可达图的分析方法一般是利用发生死锁时的可达标识,判断程序发生死锁时的执行情况,由于可达标识是各库所所含托肯的向量集,因此,判断起来稍嫌繁琐.这里,我们直接利用网语言及其同步合成运算性质进行判断,仅需找出发生死锁的变迁发生序列中分属各个子网变迁集合中顺序运行的最后一个变迁,就能了解程序发生死锁时的运行状态.对于例 2,根据定理 2,可以知道变迁 t_7, t_{16}, t_{29} 是分属各个子网变迁集合中顺序运行的最后

一个变迁,即网执行了上述变迁以后,进入死锁状态.根据 Ada 网的语义,易知此时任务 Customer 执行了 Pump.Finish 语句(加油完毕),等待响应,而任务 Pump 为了接受 Finish 调用,需要执行 Operator.Charge 语句(要求柜员机找零),同样,任务 Operator 为了接受 Charge 调用,需要执行 Customer.Charge 语句(要求顾客能够接受找零),而若接受该调用,则需执行任务 Customer 的接收语句,但其在 Pump.Finish 语句的后面,因此,程序发生了死锁.无法进展.

安全性虽然是第一位的,但这只是一种保障,而一个系统总是具有一定功能、解决一定问题的,一般地,这些是属于活性的范畴.对于并发程序也不例外.

4.2 活 性

所谓活性,是指某些好的状态最终会进入,即程序最终能进入期望中的状态.对于顺序程序而言,程序终止性和完全正确性是讨论得较多的活性性质.然而,在并发程序中,活性的内涵拓宽了,包括:每一个服务请求都最终得到响应;一个消息最终能达到其目的地;一个进程最终会进入临界区.

定义 **7**. 一个 Ada 任务程序是并发正确的,当且仅当其对应的 Ada 网是活的[15].

定理 **5**. 设 Petri 网 $PN=(S,T;F,M_0)$,则 PN 是活的充分必要条件是

$$L(PN)=stpref(L(PN)).$$

证明见文献[11].

推论 **5.1**. 设 Ada 任务程序所对应的 Ada 网 $\Sigma=(S,T;F,M_0)$,其中 $M_0(S')=1$ 并且对任意的 $s\notin S'$,有 $M_0(s)=0$,则 Ada 任务程序是并发正确的充分必要条件是

$$L(\Sigma)=stpref(L(\Sigma)).$$

程序的活锁一直是并发程序研究的重点和难点,并发正确性对应着程序无活锁性和响应性等重要性质.所谓无活锁性,即各进程不会进入循环等待状态,仅仅执行程序局部一些无意义的指令,而无法实现进展性.

定理 **6**. 设 Ada 任务程序所对应的 Ada 网Σ=$(S,T;F,M_0)$,其中 $M_0(S')=1$ 并且对任意的 $s\notin S'$,有 $M_0(s)=0$,则 Ada 任务程序发生活锁的充分必要条件是 $\exists\sigma\in L(\Sigma)$,其中 $\sigma=\alpha\circ\beta^*\wedge\exists t\in T$, 使得 $t\notin\&(\beta)\wedge\ \forall\sigma'$,有 $\sigma\circ\sigma'\notin L(\Sigma)$.

证明:结论是显然的, $\exists\sigma\in L(\Sigma)$,其中 $\sigma=\alpha\circ\beta^*\wedge\exists t\in T$, 使得 $t\notin\&(\beta)\wedge\ \forall\sigma'$,有 $\sigma\circ\sigma'\notin L(\Sigma)$ 表明程序在执行有限步后,进入局部循环,并且无法跳转出来.执行后继语句,即进入活锁.反之亦然. □

基于定理 6,Ada 任务程序的活锁分析算法描述如下:

算法 **2**(活锁分析). Livelock-Analysis.

输入: $\sigma\in L(\Sigma)$,并且 $\forall\sigma'$,有 $\sigma\circ\sigma'\notin L(\Sigma)$;

输出:若 σ 是发生活锁的序列,则输出 False,否则,输出 True.

(1) begin
(2) if σ 中不存在如 β 的子串 then {程序序列无循环,故不会发生活锁}
(3) 输出 True.
(4) else
(5) for i =1 to Length(β) do
(6) 读入 σ 的一个字符 t
(7) if $t\in T$ then $T=T-t$;{从变迁集中去掉元素 t }
(8) endfor
(9) if $T\neq\varnothing$ then {变迁集 T 不为空,即程序在局部循环中有未执行的语句,故存在局部死循环}
(10) 输出 False
(11) else {变迁集 T 为空,即程序在局部循环中执行了全部语句,为正常循环}
(12) 输出 True
(13) endif

(14) endif

(15) endbegin

限于篇幅,本文不再举例.

5 结 语

Petri 网作为并发工具已经被众多学者应用于并发程序性质的分析和验证,但主要是应用可达图和网不变量进行研究.本文首先利用网的同步合成运算,求解 Ada 网的可达图,并在保证语言不变的前提下,对子网可达图进行了化简,通过采用上述方法,有效地降低了求解可达图的算法复杂性,并保证了语言的不变性.其次,借用自动机作为语言产生器的方法,我们通过可达图形式化地产生了网语言,最终利用网语言分析了 Ada 任务程序的安全性和活性性质,给出了判断程序死锁和活锁的语言判据,定义了静态可执行性和并发正确性两个重要的反映程序特性的概念,并给出了其语言判据.

事实上,网语言作为描述网动态性质的重要工具,反映了系统的运行轨迹,刻画了系统的行为动作,更有利于描述系统的动态行为.特别地,网语言的语义和并发程序的语义是相同的,因此,可以准确地表述并发程序的动态行为,适于反映和分析系统的动态性质.但这方面的研究一直未能开展起来.本文在网语言的求解和利用网语言分析并发程序的性质方面进行了有益的探索,体现了网语言的独特优势.当然,在网语言的研究和应用方面,本文仅仅作了初步的探索,特别在保证网语言不变的前提下,进一步降低求解网语言的复杂性,还有待于研究.也期盼有更多的学者能够进行这方面的研究,丰富和完善网语言理论.

References:

[1] Shatz, S.M., Cheng, W.K. A Petri net framework for automated static analysis of Ada task behavior. Journal of System and Software, 1988,8(5):343~359.

[2] Murata, T., Shenker, B., Shatz, S.M. Detection of Ada static deadlocks using Petri net invariants. IEEE Transactions on Software Engineering, 1989,15(3):314~326.

[3] Shatz, S.M., Mai, K., Black, C., *et al*. Design and implementation of a Petri net-based tookit for Ada tasking analysis. IEEE Transactions on Parallel and Distributed System, 1990,1(4):424~441.

[4] Shatz, S.M., Tu, Sheng-ru, Murata, T. An application of Petri net reduction for Ada tasking deadlock analysis. IEEE Transactions on Parallel and Distributed Systems, 1996,7(12):1307~1322.

[5] Godefroid, P., Wolper, P. Using partial orders for the efficient verification of deadlock freedom and safety properties. Formal Methods in System Design. 1993,2(2):149~164.

[6] Notomi, M., Murata, T. Hierarchical reachability graph of bounded Petri nets for concurrent-software. IEEE Transactions on Software Engineering, 1994,20(5):325~336.

[7] Duri, S., Buy, U., Devwapalli, R., *et al*. Using state space reduction methods for deadlock analysis in Ada tasking. In: Proceedings of the 1993 International Symposium on Software Testing and Analysis. New York, NY: ACM Press, 1993. 51~60.

[8] Corbett, J.C. Evaluating deadlock detection methods for concurrency software. IEEE Transactions on Software Engineering, 1996,22(3):161~180.

[9] Peterson, J.L. Petri Net Theory and the Modeling of Systems. Englewood Cliffs: Prentice-Hall, 1981.

[10] Murata, T. Petri nets: propertier, analysis and applications. Proceedings of the IEEE, 1989,77(4):541~580.

[11] Jiang, Chang-jun. PN Machine Theories of Discrete Event Systems. Beijing: Science Press, 2000 (in Chinese).

[12] Jiang, Chang-jun. Dynamic invariance of Petri net. Science in China (Series E), 1997,27(5):567~573 (in Chinese).

[13] Jiang, Chang-jun. Complete sequence behavior invariance of synchronous composition nets. Journal of Applied Science, 2000,18(3):271~275 (in Chinese).

[14] Jiang, Chang-jun, Zhang, Zhao-qing, Qiao, Ru-liang. The design of parallel program based on PN machine. High Technology Letters, 1998,8(1):28~32 (in Chinese).

[15] Barkaoui, K. Verification in concurrent programming with Petri nets structural techniques. In: Proceedings of the 3rd International IEEE High Assurance System Engineering Symposium. 1998. 124~133. IEEE Computer Society Press, 1998. 124~133. http://www. computer.org/proceedings/hase/9221/9221toc.htm.

附中文参考文献:

[11] 蒋昌俊.离散事件动态系统的 PN 机理论.北京:科学出版社,2000.

[12] 蒋昌俊.Petri 网的动态不变性.中国科学(E 辑),1997,27(5):567~573.

[13] 蒋昌俊.同步合成网的完全顺序行为不变性.应用科学学报,2000,18(3):271~275.

[14] 蒋昌俊,张兆庆,乔如良.基于 PN 的并行程序设计方法.高技术通讯,1998,8(1):28~32.

Analysis and Verification of Local Properties of Ada Tasking Based on Net Language*

DING Zhi-jun[1], JIANG Chang-jun[1, 2]

[1](*Collage of Information Science and Engineering, Shandong University of Science and Technology, Taian* 271019, *China*);

[2](*Department of Computer Science and Engineering, Tongji University, Shanghai* 200092, *China*)

E-mail: cjjiang@online.sh.cn

http://www.sdust.edu.cn

Abstract: In this paper, the properties of Ada tasking and the correlative state explosion problem are discussed by using the net language. The synchronous composition and decomposition of Ada net is studied, as a result, net language properties are obtained, and the safeness and liveness properties of Ada tasking by using the Ada net language are analyzed and verified, which provide a new and useful way for analyzing and verifying the complex Ada tasking.

Key words: Ada net; synchronous composition; synchronous decomposition; net language; analysis; verification

* Received January 15, 2001; accepted March 19, 2001

Supported by the National Natural Science Foundation of China under Grant Nos.69973029, 69933020; the National High-Tech Research and Development Plan of China under Grant No.2001AA413020; the National Grand Fundamental Research 973 Program of China under Grant No.G1998030604; the National Science Foundation for Distinguished Young Scholars of China under Grant No. 60125205; the National Research Foundation of the Project for Cadre of Youth Teacher of Higher Education of China; the Special Foundation of National Excellent Doctoral Paper's Writer under Grant No.199934; the Grand Fundamental Research Foundation of Shanghai Science-Technology Development Project of China under Grant No.02DJ14064