维普资讯 http://www.cqvip.com

分层的面向对象 Petri 网模型分析*)

Analysis of Hierarchical Object-oriented Petri Net Model

郑 红 李师贤

(中山大学计算机科学系 广州510275)

Abstract Petri nets are widely used to analyze and model various systems for their powerful analysis ability and mature theory. Object-oriented techniques such as encapsulation, inheritance, polymorphism etc, are also applied in modeling systems, but they lack of formalization. This paper introduces the hierarchical object-oriented Petri net (HOONet) supporting the concepts of object-oriented, and analyzes its formal syntax and semantics in detail.

Keywords HOONet , Object-oriented , Formal specification , Object model

1. 引言

面向对象技术具有模块化、继承、多态性和消息传递等特征,它利用"对象"这个概念,将数据与在其上的操作封装在一起,直接作为可独立存在的建模实体,通过"消息传递"来完成对象之间的联系,若干个对象通过消息传递可以有机地组成一个系统,易于实现系统建模[1]。但随着网络的发展,系统的结构也日趋复杂。运用面向对象的方法构造复杂大系统时,常常要构造大量的对象实体,它们之间的关系错综复杂,十分繁琐,且诸多对象之间隐含有并发性,稍一疏忽便容易出错。另外,面向对象技术形式化的描述和分析还不太完善,一些问题难以得到严格的规范化结论[2]。而 Petri 网[3,5]作为一种描述具有分布、并发、异步特征的系统的图形工具,深刻、简洁地刻画了那些含有并行操作成分的系统的动态性质,在描述分析系统并发行为、动态特性等方面有较强的能力。因此,如何将面向对象技术和 Petri 网结合起来,以便充分利用面向对象技术建模的方便性和 Petri 网的强大理论分析能力一直是广泛关注的课题[4]。

2. 分层的面向对象 Petri 网 HOONet[4]

HOONet 由三个部分组成:对象标识库所(OIP),它是类的唯一的标识;内部对象网(ION)也是一个网,描述一个类的行为(方法);数据字典(DD),用以声明类的属性。

2.1 HOONet 定义

定义 2.1 HOONet 是一个3元组,HOONet = (OIP, ION, DD)满足如下条件:

1. 对象识别库所 OIP 是一个专用的库所,定义为四元组,OIP = (oip, pid, M_0, status),其中:oip 是 HOONet 的变量的名字;pid 是一个进程识别器,以区别一个类的多个实例,pid 包含返回地址;M_0是具有指定托肯值的 OIP 标识;status 是一个标志变量,说明 OIP 的状态。

2. 内部对象网 ION 描述属性值的变化和行为的变化。

3. 数据字典 DD 用来声明变量、托肯类型和函数。

HOONet 的结构如图1所示。

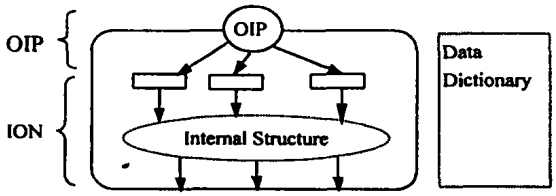

图1 HOONet 的一般结构

定义 2.2 HOONet 内部结构,ION = ($P, T, A, H, N, G, E, F, M_0$)其中,$P$ 和 T 分别是库所和变迁集。A 是有限的弧集,$P \cap T = P \cap A = T \cap A = \Phi$。$H$ 是从 P 到在 DD 中声明过的托肯类型集的映射。N、G 和 E 分别是结点、防护和弧表达式。F 是从任一变迁到 OIP 的特定的弧。M_0是任一库所的初始标识。

用 HOONet 特征建模时,可表示其中抽象信息(如抽象状态和抽象动作)和子系统之间的互操作。

定义 2.3 HOONet 库所集是二元组成 P = (PIP, ABP),其中,初始库所 PIP 是一个基本库所,表示系统的局部状态,类似于基本 Petri 网。抽象库所 ABP = (pn, refine-state, action)表示抽象状态。其中,pn 是抽象库所名;refine-state 是一个标志变量,表示 ABP 的细化;action 是模拟内部 ABP 的行为的静态反应。

抽象库所 ABP 用来表示抽象信息(状态),可以在建模过程中进一步细化。根据细化的信息(状态)是否已用 HOONet 建模,标志变量"refine_state"的值分别为 true 或 false。在 HOONet 模型中抽象库所用粗线圈表示。

定义 2.4 HOONet 的变迁集 T,有三类变迁:T = {PIT, ABT, COT},其中,初始变迁 PIT 是一基本变迁,和基本 Petri 网一样。抽象变迁 ABT = (tn, refine-state, action),其中,tn 是抽象变迁名;refine-state 是一个标志变量,表示 ABT 的细化;action 是模拟 ABT 内部的行为的静态反应。通讯变迁 COT 是表示方法调用的变迁,COT = (tn, target, comm-type, action),其中,tn 是通讯变迁名;target 是标志变量,表示是由 COT 调用的方法被模块化,值为(yes 或 no);comm-type 也是标志变量,表示是否 COT 的互作用是同步的(值为"SYNC")或异步的(值为"ASYN");action 是映射方法调用

*)本文得到国家自然科学基金委项目 NSFC 与 RGC 联合资助基金资助(批准号:79910161989)。郑 红 博士研究生,研究方向:Petri 网应用,CORBA 与软件体系结构。李师贤 教授,博士生导师,研究方向:软件工程,形式语义学等。

维普资讯 http://www.cqvip.com

的执行结果的静态反应。

通常，当一个模型包含有抽象信息(行为和状态)时，该模型表示的系统行为是很难模拟的。但是，具有抽象信息的HOONet模型可用代数表达式描述的变量"action"定义抽象信息或行为的前置条件。该变量用于获取模拟效果，如同抽象库所或抽象变迁的实际发生。在HOONet模型中，ABT和COT被分别用粗线框和双线框表示。

HOONet的托肯类型在数据字典DD中描述，分为两类：简单类型和抽象类型。简单类型等同于CPN中的托肯类型，抽象类型由简单类型组成。抽象类型用于表示抽象库所的状态。抽象类型声明为"complex"或""record"类型。用抽象信息描述的HOONet模型的托肯类型也应该用抽象的托肯类型描述，因为即使托肯类型的信息很详细，用它描述模型中抽象行为的状态也是不够的。设计者可以在抽象层就详细地声明托肯类型，但是，这时描述的抽象状态并不精确，可能在进一步的细化过程中会发生变化。抽象类型的托肯用前缀"complex"声明。在细化的模型中，抽象类型的托肯分解为子类型(简单类型)托肯。DD用简单的文本语法定义，假定一个HOONet模型"Customer"，其DD可定义如下：

```
Var CX=Boolean;/* buy 或 not
TT C=complex;
……
```

2.2 HOONet的行为语义

HOONet的行为语义支持绑定、发生等。HOONet模型的行为如图2所示。HOONet模型左边的"Client"包含抽象库所A_p，右边模型表示抽象库所的细化。当托肯交给OIP左边模型的"Client"时，T1，T2，T3中的一个变迁授权发生。在抽象库所A_p的入口处，托肯被传给细化模型的OIP，这时变量"refine_state"在2.3节中定义，其值为true(图2中步骤1)。细化模型的内部行为完成后(步2)，托肯传给细化的模型OIPA_p(步3)，然后，执行结果返回到抽象库所A_p(步4)。

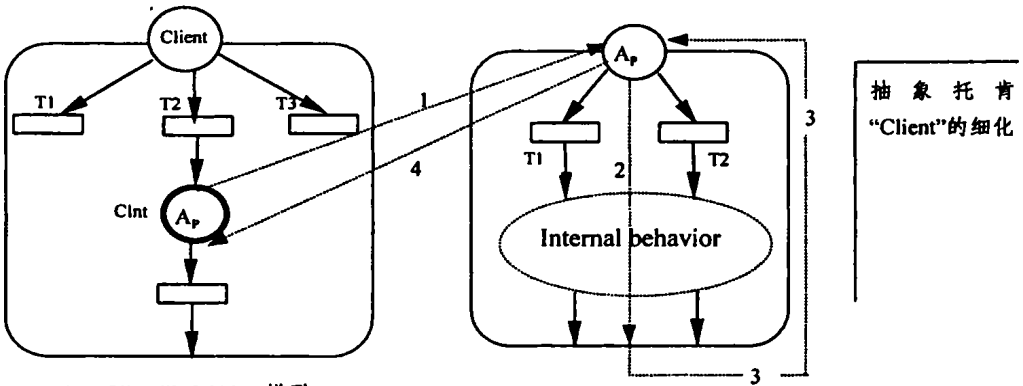

图2 HOONet行为的逻辑概念

2.2.1 OIP行为语义 HOONet模型的OIP的绑定和授权是由OIP的状态变量约束的，OIP的行为语义定义如下：

if status="pre"，$\forall t \in$ OIP·且$\forall v \in$ Var(t)：$B(t)$

if status="post"，$M(X)$或$M(X\cdot)$使得OIP.pid.return $=X$

这里，Var(t)表示防护函数和t的弧表达式的变量集合，$B(t)$是t的绑定。

2.2.2 抽象库所的行为语义 根据抽象是否细化，抽象库所有不同的行为。如果抽象是细化的，行为的绑定和发生从抽象到它的细化模型时出现。否则，ABP的后序，即ABP·的变迁将授权，ABP的行为语义定义：

if refine-state=true，M(OIP)使得ABP.pn= OIP.oip

if refine-state=false，R(action)；$\forall t \in$ ABP·：$B(t)$

其中，R(action)表示给ABP的变量"action"表达式赋值。

2.2.3 抽象变迁的行为语义 抽象是否细化决定了抽象变迁有不同的行为。如果抽象是细化的，绑定和发生行为从抽象到细化模型时出现，否则，ABT·的一个库所会做标记，ABT的行为语义定义如下：

if refine-state=true，M(OIP)使得ABT.pn = OIP.oip

if refine-state=false，R(action)；$+M(ABT\cdot)\wedge -M(ABT\cdot)$

其中，$+M(ABT\cdot)\wedge -M(ABT\cdot)$表示由ABT启动发生的输出库所的托肯减去输入库所的托肯的和。

2.2.4 通讯变迁的行为语义 根据不同通讯类型以及交互模型是否得出，HOONet模型(如类、对象等)之间有不同的互操作。COT的行为语义：

```
if target = true,
  if sync="SYNC",M(OIP)使得 COT.tn = OIP.oip
  if sync="ASYN",M(OIP)使得 COT.tn = OIP.oip
  +M(COT·)∧−M(COT·)
if traget=false,
  if sync="SYNC",R(action);+M(COT·)∧−M(·COT)
  if sync="ASYN",+M(COT·)∧−M(·COT)
```

2.3 继承组件的语义

相对于其它面向对象Petri网，HOONet更支持面向对象技术的继承机制，HOONet模型定义的一些通用的共同属性，它的子模型可以继承，从而减少了导出模型的复杂性。HOONet的继承概念和语义将在3.3节用一例子做详细的解释。

3. HOONet的面向对象表示

HOONet支持面向对象概念如类/对象、信息隐藏、抽象层次和继承等，因此具有更强的表达能力。

3.1 信息隐藏

面向对象模型中的类是一个HOONet模型，它由一个对象识别器OIP、一个数据字典DD(对应数据)和一个内部对象网ION(对应行为)组成。HOONet中的OIP是公共接口，有

维普资讯 http://www.cqvip.com

唯一的外部引用名，OIP 也是一个库所，它开始和终止模型的内部行为。DD 是一个数据字典，它涉及变量、函数和用于描述 ION 内部行为的托肯类型的详细定义。HOONet 的结构的组件完全支持信息隐藏概念，HOONet 模型相当于一个黑盒，其内部的细节对外不可见。

3.2 抽象

系统设计时，抽象强调对象是什么和做什么。HOONet 模型中，用库所、变迁及托肯表示抽象。对库所、变迁及托肯的抽象有助于保持网的动态结构及分析具有抽象状态和行为的网模型，概念上，抽象库所 ABP、抽象变迁 ABT 和抽象托肯 ATN 分别表示对象状态的抽象、方法行为的抽象、托肯类型的抽象。

抽象和细化的关系可用它们之间的射定义。射表明抽象的约束条件是保留在细化模型(Refined Model，RM)中。对于形式化的 HOONet，函数并不以射为条件，这表明抽象部件之间可视行为和它的细化部件是等同的，同时也说明抽象的托肯类型完全包含在细化的数据字典 DD 中。

当抽象库所 ABP 细化时，应该保留在细化模型中的约束条件是抽象库所 ABP 的托肯类型和输入、输出弧的表达式；对于抽象变迁 ABT，输入和输出库所的托肯类型和 ABT 的防护表达式都是约束条件。HOONet 模型中可视射定义如下：

定义3.1 可视射，$\forall C \in \{ABT, ABP, ATN\} C \Rightarrow RM$，满足如下要求：(其中，$\Rightarrow$表示细化为)

1. 对于任意的 ABP 和 ABT 的托肯类型，

$\forall p \in ABP, TYPE(p) \subseteq TYPE(TT_{RM})$，并且

$\forall t \in ABT, \{TYPE(\cdot t) \cup TYPE(t \cdot)\} \subseteq TYPE(TT_{RM})$，

其中，TT_{RM}是由 RM 的 DD 定义的所有托肯类型集，TYPE(p)表示每一个库所 p 到托肯类型集的映射。

2. 对于 ABP 的所有输入输出弧的表达式 a_i，a_0，以及所有 ABT 的 a_i 和 ABT·的 a_0和 ABT·的 $a_0 \underset{a_i \cdot a_0}{Y}$ $Var(expr) \subseteq Var(OIP_{RM})$

3. 对于所有的 ABT 的防护表达式，t ABT，$G(t) \equiv Y_{\forall u \in OIP} \cdot G(t_i)$，$G(t)$是 t 的所有防护函数集

用抽象和细化渐增分层对系统进行建模，可以减少在分析和设计过程中的复杂性。

3.3 继承

继承是基于层次关系上的类之间的属性和操作的共享。类可进一步细化为子类。每个子类结合或继承它的父类的所有属性并且增加自己特有的属性。父类的属性不必在子类中重复描述。

HOONet 模型中继承的概念如图3所示。图3说明了子类 Customer 继承了所有父类 Person 的属性。表达继承的符号“:<”。继承类 Customer 包含一个增加的行为由“aspect”变迁发生，还包括一个细化的行为由“license”变迁发生。

实例化继承类时，子类的数据字典 DD 扩展以包含父类的 DD。$E(OIP, t)\langle b \rangle$授权的变迁集也被扩展。继承机制中要注意父类属性的重定义(overriding)。子类重定义的特征(如 customer 模型中“license”)细化和替代了父类重定义的特征(如 person 模型中“license”)。由于重定义细化了具有同名的方法，但并不允许重定义这个特征符号或形式，因此要通过动态绑定确定重定义方法及哪一个方法可被绑定。

定义3.2 继承父类需满足如下条件：假定 I、S 和 N 分别表示子类、父类和重定义的方法。

1. $TT_I = TT_S \cup TT_N$，TT_I、TT_S、TT_N 分别是子类、父类及重定义方法的托肯类型集。

2. 授权变迁集，$ET_I = \{t_i \mid \forall t_N, E(OIP_I, t)\langle b \rangle\} \cup \{t_i \mid \forall t_i, E(OIP_S, t)\langle b \rangle\}$，且

3. 如果 $t_i \in T_N \wedge t_i \in T_S$，$\exists t_i, t_i \in ET_I \alpha ((t_i, b) \in Y \wedge (t_i, b) \notin Y)$。

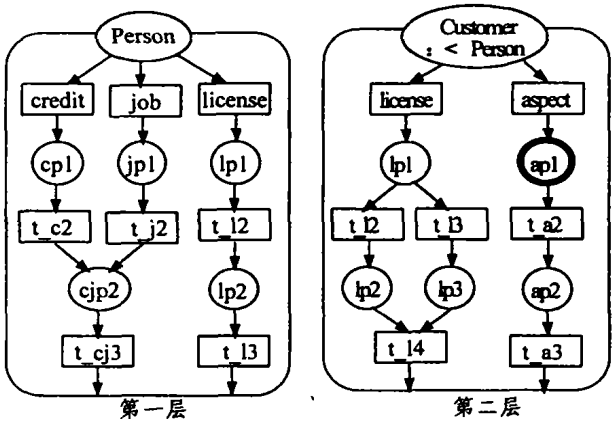

图3 HOONet 模型的继承

3.4 参数多态

支持多态意味着可能根据输入的托肯类型值动态地绑定一个托肯类型。实际上，动态绑定可看作为分发机制，根据不同的托肯类型值选择(动态地)合适的行为。在 HOONet 模型中，抽象库所 ABP 或者抽象变迁 ABT 的变量 refine_state 的为真时，并且通讯变迁 COT 调用外部类的 OIP 时出现多态。抽象部件的动态绑定分为两步：首先，复杂的托肯类型分解为几个 OIP 的托肯类型；然后根据复杂类型的值，定义 OIP 的托肯值，例如图3中的情形。一个抽象的库所“ap1”是表示智力或生理上消费者状态是否正常的库所，(如 TT C=complex with normal|abnormal)，当 C 的托肯值是 normal，并且正常状态表示一个消费者的精神状态时，托肯类型 C 可分解为：

```
TT C=record with{
    IntQ=[0…110]|[111…130]|[131…150];
    EmoQ=[0…3]|[4…7]|[8…10];
    SpeK=trouble|common|fluent;}
```

其次，在 ap1的细化类型中，每一类型的托肯值指定为{(IntQ,([111…130]),(EmoQ,[4…7]),(SpeK,common)。如果物理状态是正常的，C 的托肯类型可分解为：

```
TT C=record with{
    Height=short|average|tall;
    Weight=light|middle|heavy;
    Heart=low|center|high;}
```

并且，在细化模型中，托肯值指定为{(Height,average),(Weight,middle),(Heart,center)}。用这种方式，根据参数变量的值(智力或生理上的)，一个托肯类型能够和不同的子类型绑定。

通过继承创建的类的属性和方法也是一种多态。

3.5 对象之间的消息传递

HOONet 模型中对象之间的消息传递是由 COT 传递一个托肯到被调用对象的 OIP 实现。根据输入的 OIP 的托肯值，决定 OIP·的哪一个的变迁能够授权和发生。一个指定的方法结束，通过它自己的 OIP，将执行结果传给 COT。这里有两点要注意：

(下转第10页)

维普资讯 http://www.cqvip.com

DFIFO 模型最为接近，但是 DFIFO 考虑了接收缓冲区和消息延迟，比 FIFO 模型更能反映实际系统的特性，尤其是机群系统的特性。

4 DFIFO 消息通信模型的应用

DFIFO 消息通信模型，是我们正在开发的机群系统并行程序调试环境--DENNET(DEbugging eNvironmeNt for parallEl programs on clusTer systems)[5]的基础，它主要应用于以下几个方面：

1)记录/重放调试 依据 DFIFO 消息通信模型，可以对并行程序中的不确定性事件进行排序，从而解决调试并行程序时的不确定性问题。在 DENNET 中，我们已经实现了基于向量时钟的执行/重放功能，可以对并行程序进行确定性的调试[5]。

2)调试器对并行程序执行的干扰特性的判定 并行程序具有不确定性的特征，调试器的介入会影响并行程序的不确定性行为，这就是我们常说的干扰效应。有了 DFIFO 消息通信模型，可以方便地研究调试器对并行程序的干扰特性，进而搞清在什么情况下有干扰，在什么情况下没有干扰。我们设计了一个判定算法，它可以在调试程序的过程中实时地告知程序员本次调试活动是否对并行程序的行为造成了干扰[6]。

3)检查点设置/回卷恢复时全局一致性状态的确定 在 DENNET 中，拟实现检查点设置/回卷恢复功能，以支持大规模长时间运行程序的调试。回卷恢复时，关键是要确定全局一致性的状态，我们结合记录的检查点设置方法和 DFIFO 消息通信模型来确定回卷恢复时的全局一致性状态。

4)性能调试中的性能预测 性能调试是 DENNET 系统的重要功能[7]，我们拟采用解析建模的方法实现有限的性能预测功能[8]，解析建模的对象(并行程序)同样以 DFIFO 消息通信模型为基础。

本文主要是介绍我们提出的 DFIFO 消息通信模型，上述几方面的内容将会有专门的文章单独讨论，在此不再详细叙述。

结论 本文讨论了机群系统中消息通信的模型，首先用形式化的方法，总结了常用的几种消息通信模型：同步消息通信模型、异步消息通信模型、FIFO 消息通信模型、CO 消息通信模型和 RSC 消息通信模型，讨论了它们的一些性质及其相互关系。然后，本文提出了一个有限延迟的双 FIFO 消息通信模型。与现有的模型相比，DFIFO 消息通信模型更能反映机群系统的特性，如消息缓冲、消息延迟等，利用 DFIFO 消息通信模型可以对并行程序的执行进行更为细致而实际的分析与研究。从而为解决机群系统中其它的关键技术建立形式化的基础。

建立机群系统的消息通信模型，是研究机群系统中其它技术的基础。在设计调试器时，可以用上述的消息通信模型解决不确定性问题；也可以应用消息通信模型来分析研究调试器对并行程序的干扰特性；在检查点技术中，要应用消息通信模型来确定全局一致性的状态；机群系统的时钟问题涉及确定性调试、性能调试和容错技术等诸多方面的基础，建立机群系统的时钟需要应用消息通信模型；对并行程序进行性能预测，同样要建立并行程序的消息通信模型；在静态分析并行程序的不确定性因素时，建立高效而又符合实际的消息通信模型是非常重要的。本文对 DFIFO 消息通信模型在 DENNET 的设计和实现中的应用情况，作了简要描述。

参考文献

1 Shatz S M. Communication Mechanism for Programming Distributed Systems. Computer, 1984, 17: 21～28
2 Lamport L. Time, Clock and Ordering of Events in a Distributed System. Communication of ACM, 1978, 21(7)
3 Birman K, Joseph T A. Reliable Communication in the Presence of Failure. ACM Transactions on Computer Systems, 1987, 5: 47～76
4 Soneoka T, Ibaraki T. Logically Instantaneous Message Passing in Asynchronous Distributed Systems. IEEE Transactions on Computer, 1994, 43: 513～527
5 Liu Jian, Yu Hong-liang. Implementation of a Debugger for Parallel Program in Cluster System. The 7th Joint Intl. Computer Conference, 2000. 11, Shantou: Guangdong
6 Xiong Jianxin, et al. On-line Debugging of Parallel Program. Intl. Conf. on Parallel Algorithm(ICPA'95), Wuhan Oct. 1995
7 刘建，余宏亮，等. 并行程序性能调试环境模型. 中国计算机体系结构2000学术年会，哈尔滨：2000
8 赵刚，沈美明，郑纬民. 并行程序集成环境的特点及 IPCE2.0实现中的关键技术. 中国计算机体系结构2000学术年会，哈尔滨：2000

(上接第26页)

1. 同步或异步的方法调用：同步调用时，COT 必须收到调用的结果。因此，COT·的库所不能被标记，直到调用返回。同时，"target"值是"yes"。在异步调用时，即使托肯从 COT 传给了 OIP，COT·的库所仍然有标记，COT 的异步调用中，托肯传给 OIP 和 COT·，方法调用的结果保存在全局变量中，以便以后使用。

2. 嵌套调用：在调用期间，可能调用其它对象的方法或自己的方法。变量 pid 是表示方法调用的嵌套。当 COT 调用一个方法时，变量 pid 为传递托肯值，进程识别器识别由方法调用创建的类的实例，pid 的大小和子序列调用次数成正比。

3.6 对象实例化

当一个类初始化系统行为(在过程语言中相当于 main())，或一个类被其它类(或对象)调用时，该类的一个实例可创建。类的实例有识别器，它可区别变量 pid 的值。实例 pid 的值对应 main()函数，是自包含的，被调用实例的 pid 值返回指针给调用类。

总结 HOONet 在建模方法上采用细化和分层，用 HOONet 建模通过合适的接口支持信息隐藏；通过抽象库所、变迁和托肯支持抽象；通过动态绑定抽象托肯支持多态，通过类之间的共享支持继承；通过消息传递支持交互(interaction)。因此，HOONet 不仅具有 Petri 网的直观模拟系统动态行为的能力，同时也具有面向对象技术的优点。

参考文献

1 Kilov H, Ross J. Information Modeling—An Objected Oriented Approach. Prentice-Hall, 1994
2 Bastide R. Approaches in unifying Petri nets and the object-oriented approach. In: Proc. of the Intl. Workshop on Objected-oriented Programming and Models of Concurrency, 1995, 6
3 Peterson J. Petri Net Theory and The Modeling of Systems. Prentice Hall, 1981
4 Hong Jang-Eui, Bae Doo-Hwan. Software modeling and analysis using a hierachical object-oriented Petri net. Information Sciences, 2000
5 袁崇义. Petri 网原理. 北京：电子工业出版社，1998
6 张新晖，朱淼良，吴春明. 立体分层 Petri 网对象化专家系统实现. 小型微型计算机系统，2000, 3