# 基于 Petri 网的 UML 状态图的形式化模型

郭 峰 姚淑珍

(北京航空航天大学 计算机学院，北京 100083)

**摘 要**：提出一种可以准确描述 UML 状态图动态特征的形式化模型 SC_Net. 首先给出了 UML 状态图的形式化语法定义，其中用状态集合、转移集合、事件集合、条件集合、活动集合、对象集合和变量集合，定义了一系列辅助函数描述 UML 状态图特征，用确定目标状态和受限源状态表示层次关系，用开放事件和封闭事件表示对象之间的消息. 基于 C_Net 定义了描述 UML 状态图动态语义的 Petri 网模型 SC_Net，既能描述状态图中的控制部分，又能描述状态图中的数据处理部分，并给出了从 UML 状态图到 SC_Net 的转换步骤，便于实现自动转换过程. 最后以柔性制造系统的一个实例说明 SC_Net 能用于分析 UML 状态图的性质.

**关 键 词**：UML 状态图；Petri 网；形式化语义

**中图分类号**：TP 311.5；TP 393

**文献标识码**：A **文 章 编 号**：1001-5965(2007)02-0248-05

## Formal models of UML statechart diagrams based on Petri nets

Guo Feng Yao Shuzhen

(School of Computer Science and Technology, Beijing University of Aeronautics and Astronautics, Beijing 100083, China)

**Abstract**: A formal model named SC_Net which can precisely describe the dynamic features of UML statechart diagrams was presented. Firstly a formal syntactic of UML statechart diagrams was given and a series of auxiliary functions to describe the structure features of UML statechart diagrams were presented in which target determinator and source restriction were used to describe the inter-level transitions and open events and close events were used to describe the communication among UML statechart diagrams. Then, based on C_Net an extended Petri net model named SC_Net was defined which can be used for the semantic models of UML statechart diagrams. Besides the control parts, SC_Net also included the data process parts which was lacked in the research area of formal semantics of UML statechart diagrams. The translation process from UML statechart diagrams to SC_Net was defined. At last, an example in flexible manufacture systems was given. Results show that SC_Net can be used to verify the properties of the UML statecharts.

**Key words**: UML statecharts; Petri nets; formal semantics

UML 状态图是经典的 Statechart[1] 的面向对象变体，UML 规范[2] 用元模型描述模型的语法，用自然语言描述了行为模型的执行语义，这种方法限制了 UML 状态图在关键领域的应用. 因此，对于 UML 状态图形式化语义的研究一直是一个开放的话题. 目前定义 UML 状态图形式化语义的方法主要可以分为 2 种. 一种是操作语义方法，如文献[3－5]等，借助层次自动机(EHA，Extended Hiarichical Automata)定义 UML 状态图的操作语义，通过模型检测方法分析模型的性质. 此种方法的优点是采用 EHA 能够很好地描述 UML 状态图的层间转移引起的冲突和优先级等概念，但是不能准确描述多对象合作情况下出现的真并发现象. 另一种是指称语义方法，通过定

**收稿日期**：2006-01-05

**基金项目**：航空科学基金资助项目(01F51052)

**作者简介**：郭 峰(1972－)，男，山东宁津人，博士生，guofeng_buaa@163.com.

义语法转换规则,将 UML 状态图转换为另外的形式化模型,如 Petri 网[6-7]、进程代数、Z 语言等.指称语义方法尤其是基于 Petri 网的方法,可以很好地利用 Petri 网在真并发语义方面的优势,但是目前的方法对于 UML 层间转移引起的并发和冲突等概念缺少深入研究.

Petri 网是一种系统的数学和图形的建模和分析工具,适用于对具有并发、同步、冲突等特点的系统进行模拟和分析,广泛应用于复杂系统的设计与分析中.本文以 Petri 网作为 UML 状态图的语义模型,定义了一种可以精确描述 UML 状态图语义的 Petri 网模型 SC_Net,给出了从 UML 状态图到 SC_Net 的转换规则.这项工作将成为对 UML 状态图的性质进行自动化分析的基础.

## 1　UML 状态图的形式化语法定义

本文在相关研究的基础上(如文献[8]中给出的 UML 状态图的形式化定义),结合文献[4]中受限源状态与确定目标状态的概念,给出 UML 状态图的一种形式化语法.

**定义 1**　UML 状态图是一个七元组:

$M=(S,T,E,G,A,O,V)$.其中,$S$ 是状态集合,$T$ 是转移集合,$E$ 是事件集合,$G$ 是转移上的条件集合,$A$ 是活动集合,$O$ 是对象集合,$V$ 是变量集合.在此基础上定义一系列辅助函数:

Name:$S\cup T\rightarrow$String,表示状态的名称.

Child:$S\rightarrow\rho(S)$,$\rho(S)$表示 $S$ 的幂集,Child 函数表示一个状态可以有多个直接子状态.

Sub:$S\rightarrow\rho(S)$,表示一个状态的所有子状态.

Kind:$S\rightarrow\{$simple, final,seq,conc $\}$.

Initial:$S\rightarrow\{0,1\}$,标识状态为初始状态.

Source:$T\rightarrow S$,定义转移的源状态.转移最外层的状态是其源状态,转移最内层的状态,也就是 UML 规范中定义的源状态,在本文称为受限源状态.

Target:$T\rightarrow S$,定义转移的目标状态.

SourceR:$T\rightarrow S$,定义转移的受限源状态.

TargetD:$T\rightarrow\rho(S)$,定义转移的确定目标状态.转移进入复合状态时,最外层的状态称为目标状态,最内层的状态称为确定目标状态,也就是 UML 规范中定义的目标状态.

Triger:$T\rightarrow E$,定义转移的触发事件.

Guard:$T\rightarrow G$,是布尔表达式,布尔表达式中的变量在变迁的动作中进行计算.

Receiver:$(T\rightarrow E)\rightarrow O$,每一个动作表达式都可能有一个对应的对象.

Event:$T\rightarrow E$,定义输出事件.一个转移上的活动,或者是向其它对象发送消息,或者是执行内部操作.

Activity:$T\rightarrow A$,定义了变迁上的活动.

Enter:$T\rightarrow A$,定义了状态的入口动作.

Exit:$T\rightarrow A$,定义了状态的出口动作.

Read:$T\rightarrow\rho(V)$,定义了每个转移上活动表达式的输入变量.

Write:$T\rightarrow\rho(V)$,定义了每个转移上活动表达式的输出变量.

Open:$E\rightarrow$boolean,0 表示该事件为封闭事件,也就是说该事件出现在状态图内部或者协作对象的状态图中;1 表示该事件是开放事件,表示该对象来自环境之中.

Type:$V\rightarrow$TYPE, TYPE 是有限非空类型集,包括用户自定义的类型.

**定义 2**　$C(t)$表示与 $t$ 并发的转移集合,满足条件:$\forall t'\in C(t)$, $\mathrm{Event}(t)=\mathrm{Event}(t')$, $\mathrm{Sub}(\mathrm{Source}(t))\wedge\mathrm{Sub}(\mathrm{Source}(t'))=\varnothing$.

**定义 3**　$F(t)$表示与 $t$ 冲突的转移集合,满足条件:$\forall t'\in F(t)$, $\mathrm{Event}(t)=\mathrm{Event}(t')$, $\mathrm{Sub}(\mathrm{Source}(t))\wedge\mathrm{Sub}(\mathrm{Source}(t'))\neq\varnothing$.

**定义 4**　优先级:$\forall t,t'\in F(t)$,如果两者源状态交集不为空,且 $\mathrm{Sub}(\mathrm{Source}(t))\subset\mathrm{Sub}(\mathrm{Source}(t'))$,则 $t$ 的优先级高于 $t'$,记为 $t'<t$.

## 2　UML 状态图的 Petri 网模型

本文基于 C_Net[9]和抑制弧[10]的相关概念,定义适合描述 UML 状态图语义特征的 SC_Net. C_Net通过进一步给库所赋值,使 Petri 网具备了处理非流动资源的能力,适合描述 UML 状态图中转移之上和状态内部的活动对变量的操作.抑制弧适合描述 UML 状态图中冲突变迁之间的优先级关系.因此,SC_Net 主要有 3 个优点:适合描述多对象合作下的真并发现象;适合处理层间转移引起的冲突和优先级问题;适合描述状态图中的计算.

**定义 5**[9]　六元组 $N=(S_v,S_p,T,F,R,W)$称为 C_Net 的条件是:

$$S_p\cup S_v\cup T\neq\varnothing\wedge S_p\cap S_v=\varnothing\wedge S_p\cap T=\varnothing\wedge S_v\cap T=\varnothing\wedge F\subseteq S_p\times T\cup T\times S_p\wedge R,$$

$$W\subseteq S_v\times T\wedge\mathrm{dom}(F)\cup\mathrm{cod}(F)=F\cup T\wedge S_v=$$

$$\mathrm{dom}(R\cup W)\wedge S_v\cup\mathrm{cod}(R\cup W)\subseteq T$$

其中，$S_p$是控制库所集；$S_v$是变量集；$T$是变迁集；$R$，$W$分别是读、写关系；$F$是流关系.

**定义6** 令$T$为变迁集合，$T$的复合变迁表示为$t'=\mathrm{Composite}(T)$，结果是$\forall t\in T$：${}^\cdot t'{}^\cdot=\cup_{\forall t\in T}{}^\cdot t{}^\cdot$，$e(t')=\cup_{\forall t\in T}e(t)$，$B(t')=\cup_{\forall t\in T}B(t)$，$A(t')=\cup_{\forall t\in T}A(t)$.

在C_Net(Computing Net)[9]的基础上定义SC_Net(StateChart Net).

**定义7** 一个C_Net是SC_Net的充分必要条件是：

1) $\exists\,\mathrm{in}T,\mathrm{out}T\in T$，$\exists\,\mathrm{env}\in S_p$，${}^\cdot\mathrm{env}=\varnothing$，${}^\cdot\mathrm{in}T=\{\mathrm{env}\}$，$\mathrm{out}T^\cdot=\{\mathrm{env}\}$；

2) $\exists\,\mathrm{in}P,\mathrm{out}P\in S_p$，${}^\cdot\mathrm{in}P=\{\mathrm{in}T\}$，$\mathrm{out}P^\cdot=\{\mathrm{out}T\}$.

**定义8** $\Sigma=(S_v,S_p,T,F,R,W,I,K,E,P,B,A,M_t,M_0)$称为SC_Net系统，如果$C_N=(S_v,S_p,T,F,R,W)$为SC_Net，$I$，$K$，$E$，$P$，$M_t$，$M_0$，$B$，$A$依次为抑制弧集合，容量函数，权函数，类型函数，变迁标识，初始标识，哨函数，操作函数，

1) $N=(S_p,S_v,T,R,W,F)$是C_Net；

2) $P:\{\mathrm{in}P,\mathrm{out}P,\mathrm{env}\}\cup S_v\cup T\to\mathrm{TYPE}$是$S$上的类型函数，$\forall s_p\{\mathrm{in}P,\mathrm{out}P,\mathrm{env}\}$，$P(s_p)$定义了控制库所中token的类型，对于变量库所，$P(S_v)$定义了变量库所的类型；

3) $I$是抑制弧集合，$I\subseteq S_p\times T$，$|I|\geqslant 0$；

4) $\forall s_p\in\{\mathrm{in}P,\mathrm{out}P,\mathrm{env}\}$，$K:s_p=\infty$；

$\forall s_p\in S_p-\{\mathrm{in}P,\mathrm{out}P\}$，$K:s_p=1$；

5) $W:F\to E\cup\{1\}\cup\{x,y\}$，$x,y$是变量；

6) $B:T\to\mathrm{BoolExpr}$，$B$是$T$上的哨函数；

7) $A:T\to\mathrm{Assignment}$，$\forall t\in T$，$A(t)$给出了$t$的操作细节；

8) 变迁标识$M_t$，初始标识$M_0$以及标识$M$的定义请参考文献[9].

## 3 UML状态图到SC_Net转换规则

状态图本身的状态用格局描述，格局包含当前所有活动的状态.对于复合状态，同样存在以其处于活动状态的子状态表示的格局.下面给出的格局计算函数的定义借鉴了文献[11]中的方法.

Asc函数计算一个状态的所有可能的格局.

$$\mathrm{asc}(s):=\begin{cases}\{s\} & \mathrm{Kind}(s)\in\{\mathrm{simple},\mathrm{final}\}\\ \cup_{s'\in\mathrm{Child}(s)}\mathrm{asc}(s') & \mathrm{Kind}(s)=\mathrm{seq}\\ \otimes_{s'\in\mathrm{Child}(s)}\mathrm{asc}(s') & \mathrm{Kind}(s)=\mathrm{conc}\end{cases}$$

Isc计算一个状态的初始格局：

$$\mathrm{isc}(s):=\begin{cases}\{\lambda\} & \mathrm{Kind}(s)=\mathrm{final}\\ \{s\} & \mathrm{Kind}(s)=\mathrm{simple}\\ \mathrm{isc}(s'\mid s'\in\mathrm{Child}(s),\mathrm{Initial}(s')) & \mathrm{Kind}(s)=\mathrm{seq}\\ \otimes_{s'\in\mathrm{Child}(s)}\mathrm{isc}(s') & \mathrm{Kind}(s)=\mathrm{conc}\end{cases}$$

Exsc计算转移离开源状态时可能的格局：

$$\mathrm{exsc}(s,t):=\begin{cases}\{s\} & \mathrm{Kind}(s)\in\{\mathrm{simple},\mathrm{final}\}\\ \mathrm{asc}(s') & \mathrm{Kind}(s)=\mathrm{seq},\ s'=\mathrm{sr}(t)\\ \mathrm{asc}(s) & \mathrm{SourceRestriction}(t)=\varnothing\\ \mathrm{asc}(s')\otimes(\otimes_{s''\in\mathrm{Child}(s)}\mathrm{asc}(s'')) & \mathrm{Kind}(s)=\mathrm{conc}\\ & \mathrm{SourceR}(t)\subset\mathrm{Child}(s')\\ & \mathrm{SourceR}(t)\not\subset\mathrm{Child}(s'')\end{cases}$$

Ensc计算转移进入目标状态时的格局：

$$\mathrm{ensc}(s,t):=\begin{cases}\{\lambda\} & \mathrm{Kind}(s)=\mathrm{final}\\ \{s\} & \mathrm{Kind}(s)=\{\mathrm{simple}\}\\ \mathrm{isc}(s')\mid_{s'=\mathrm{td}(t)} & \mathrm{Kind}(s)=\mathrm{seq}\\ (\otimes_{s'\in\mathrm{td}(s)}\mathrm{isc}(s'))\otimes(\otimes_{s''\in\mathrm{Child}(s),\mathrm{Child}(s'')\wedge\mathrm{td}(s)=\varnothing}\mathrm{isc}(s'')) & \mathrm{Kind}(s)=\mathrm{conc}\end{cases}$$

基于上述格局定义函数，按照下列步骤可以将状态图转换为SC_Net：

1) $Z=\{S_p,S_v,T,F,R,W\}$

其中

$S_p=\{s_p\mid s_p\in S,\ \mathrm{kind}(p)\in\{\mathrm{simple},\mathrm{final}\}\}$

$T=\{t_{\mathrm{from.\,tr.\,to}}\mid\mathrm{from}\in\mathrm{exsc}(s,t),\mathrm{tr}\in T,\mathrm{to}\in\mathrm{ensc}(s,t)\}$

$S_v=\{s_v\mid s_v\in V\}$

$F=\{\langle s_p,t_{\mathrm{from.\,tr.\,to}}\rangle\mid t_{\mathrm{from.\,tr.\,to}}\in T,s_p\in S_P,s_p\in\mathrm{from}\}\cup\{\langle t_{\mathrm{from.\,tr.\,to}},s_p\rangle\mid t_{\mathrm{from.\,tr.\,to}}\in T,s_p\in P,s_p\in\mathrm{to}\}$

$R=\{\langle s_v,t_{\mathrm{from.\,tr.\,to}}\rangle\mid t_{\mathrm{from.\,tr.\,to}}\in T,s_v\in\mathrm{Read}(\mathrm{tr})\}$

$W=\{\langle t_{\mathrm{from.\,tr.\,to}},s_v\rangle\mid\mathrm{t}_{\mathrm{from.\,tr.\,to}}\in T,s_v\in\mathrm{Write}(\mathrm{tr})\}$

2) 在$N$中增加$\mathrm{in}T,\mathrm{out}T,\mathrm{env}$，同时增加这些库所和变迁连接的弧：

$\mathrm{Add}(F,\{\mathrm{env},\mathrm{in}T\},\{\mathrm{out}T,\mathrm{env}\})$

$\mathrm{Add}(R,\{\mathrm{id},\mathrm{int}\},\{\mathrm{id},\mathrm{out}T\})$

$\mathrm{Add}(W,\{\mathrm{in}T,\mathrm{id}\},\{\mathrm{out}T,\mathrm{id}\})$

令

$\mathrm{Triger}(t_{\mathrm{from.\,tr.\,to}})=\mathrm{Triger}(\mathrm{tr})$

If $\mathrm{Open}(\mathrm{Triger}(t_{\mathrm{from.\,tr.\,to}}))$

then $\mathrm{Add}(F,\langle\mathrm{in}P,t_{\mathrm{from.\,tr.\,to}}\rangle)$

令

$\mathrm{Event}(t_{\mathrm{from.\,tr.\,to}})=\mathrm{Event}(\mathrm{tr})$

If Event($t_{\text{from. tr. to}}$)≠null

then Add($F$,⟨$t_{\text{from. tr. to}}$,out$P$⟩)

3) 定义资源库所的类型:

$P(s_v)$=Type($s_v$), $P$(in$P$)=$E$, $P$(out$P$)=$E$

4) 给弧加标记:

$W$(in$P$, $t_{\text{from. tr. to}}$)={$x$|$x$∈Triger(tr)}

$W$($t_{\text{from. tr. to}}$,out$P$)={$e$|$e$∈Event(tr)}

$W$(in$T$, in$P$)=$x$,$W$(out$P$, out$T$)=$y$

$W$(env, in$P$)=$x$,$W$(out$T$, env)=$y$

5) 给变迁加标记:

$B$($t_{\text{from. tr. to}}$)=Guard(tr)

$A$($t_{\text{from. tr. to}}$)=Activity(tr)

$B$(in$T$)="$x$ in Out$E$";

$B$(out$T$)="$y$ in In$E$"

其中,Out$E$ 是状态图的输出事件集合;In$E$ 是状态图的输入事件集合.

6) 对于并发转移,执行合并操作:

∀ $T'$∈$\rho$($C$($t$)),$t'$=Composite($T'$)

∀ $p$∈$\cdot t'$

if $p^{\cdot}$∉($C$($t$)−$T$)

then ∀$t$∈$C$($t$)　Add($I$, {$p$,$t$})

7) 对于冲突转移,如果存在优先级高的变迁,增加抑制弧:

∀ $t'$∈$F$($t$)

if $t'$>$t$

then ∀ $p$∈$\cdot t'$ add($I$,{$p$,$t$})

为了准确分析模型的性质而将非形式化的 UML 状态图转换为形式化的 Petri 网的过程实际上是一种指称语义的定义方式. 这种语义是否恰当,如何证明,可以借鉴形式化语义研究中的结论,这方面的问题是本文后续的研究内容.

## 4　实例分析

图 1 和图 2 是柔性制造系统中的中 2 个对象的状态图. 图 1 中的任务规划对象根据负责生成规划,图 2 中的任务控制对象负责控制规划的执行. 为了避免图形过于复杂,仅仅在任务控制对象中从"发送目标"状态和"决策"状态之间的转移上描述了对数据的处理,在其它转移上,也可以采用类似的描述方法,并且可以用同样的方法转换为

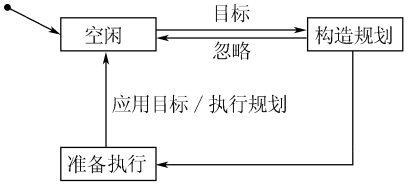


图 1　任务规划对象状态图

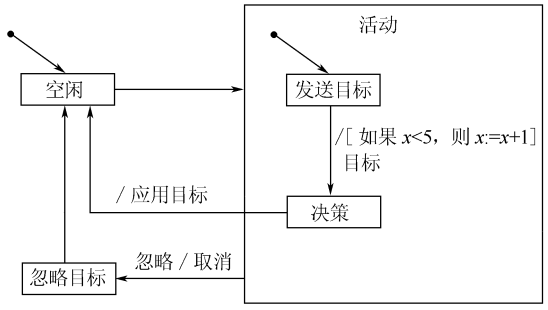


图 2　任务控制对象状态图

SC_Net 的对数据的处理部分. 在任务控制对象中,$x$ 代表有限的系统资源,任务控制对象首先执行 $x$+1 的操作,当满足 $x$<5 这一条件时,向任务规划对象发出"目标"消息,任务规划对象根据这一目标生成完成任务的规划.

按照第 3 节的转化规则,分别转换为图 3 和图 4 所示的 SC_Net. 任务控制对象包括对数据的操作,其对应的 SC_Net 中反映了这一特征,变量 $x$ 转换为变量库所,变迁 $t_2$ 相应增加了赋值语句. 由此可见, SC_Net 可以准确地描述 UML 状态图的状态变化和数据处理 2 个方面的特征,这是现有的 UML 状态图形式化语义描述方法中所欠缺的. 同时,SC_Net 可以描述对象之间的通信机制,并且多个对象可以独立、并发地执行,对象之间的通信和并发在 UML 规范中没有做规定,本文给出的描述是自然的容易被理解的,与面向对象系统的实际执行情况是相符的.

根据文献[8]中的结论,C_Net 系统的安全性和进展性是其主要的分析方法,SC_Net 作为 C_Net 的限制解释,同样受到这一结论的限制. 但是在对 UML 状态图进行转换之后,在某些情况下可能没有或者有意屏蔽掉数据处理部分,在这些情况下仍然可以采用传统的可达树、可达图或不变量分析方法分析模型的性质. 例如,如果屏蔽图 4 的数据处理部分,将图 3 和图 4 两个对象的 SC_Net 通过合并 env 库所,合成为一个包含 2 个对

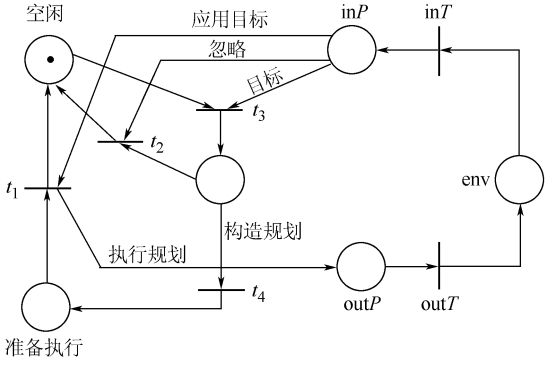


图 3　任务规划对象 SC_Net

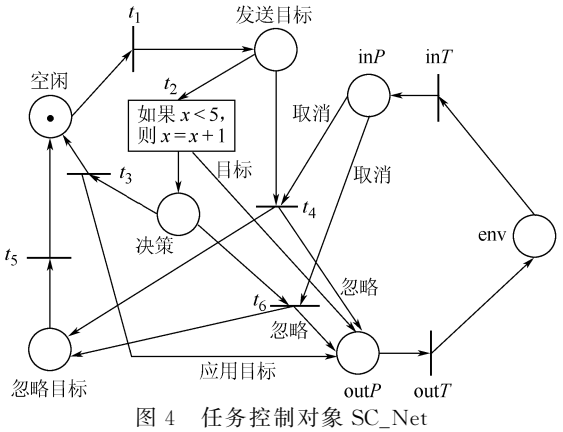


图4 任务控制对象 SC_Net

象的 Petri 网模型,通过分析,可以发现该模型存在死锁的可能:任务控制对象 SC_Net 执行变迁序列 $t_1$,$t_2$,$t_6$,而任务规划对象 SC_Net 在变迁 $t_3$ 发生之后,变迁 $t_2$,$t_4$ 都有可能发生,如果变迁 $t_4$ 发生,之后无法处理任务控制对象发出的"忽略"消息,也无法处理任务控制对象进入下一循环之后发出的其它消息,一方面致使任务规划对象进入死锁状态,另一方面出现对象之间的消息无法响应的情况.由此可知图1和图2给出的状态图模型不满足正确性,需要修改之后重新进行验证.产生这个问题的原因是图1本身存在二义性,解决这一问题的关键在于用户如何定义对"忽略"消息的处理,如果不允许在"构造规划"状态之后处理"忽略"消息,则可以增加抑制弧(in$P$,$t_4$),标记为"忽略",表示当存在"忽略"消息时不允许 $t_4$ 发生,如果允许在"构造规划"状态之后处理"忽略"消息,则可以在图1中在"准备执行"状态和"空闲"状态之间增加一个由"忽略"消息触发的转移.

## 5 结束语

本文基于 C_Net 提出了一种可以形式化描述 UML 状态图语义的 Petri 网模型 SC_Net,尤其是对于冲突、优先级和复合状态的进入与离开等概念给出了形式化的说明.在 SC_Net 的基础上,可以借助 Petri 网工具,算法分析 UML 状态图的性质.本文的研究可以结合 Petri 网的形式化分析能力和 UML 状态图的用户友好特性,给用户提供强大的建模及模型分析能力.在本文的后续工作中,将进一步研究 SC_Net 的状态空间构造算法以及其性质分析方法,同时进一步研究 UML 其它行为模型的 Petri 网表示方法,在此基础上研究 UML 行为模型的正确性和一致性问题.

## 参考文献(References)

[1] Harel D. Statecharts: a visual formalism for complex systems[J]. Science of Computer Programming, 1987, 8(3): 231-274

[2] OMG unified modelling language version 1.5[EB/OL]. Object Management Group, 2003, http://www.omg.org/uml/

[3] Wieringa R, Broersen J. A minimal transition system semantics for lightweight class and behavior diagrams[C]// Broy M, Coleman D, Maibaum T, et al. Proceedings of the ICSE98 Workshop on Precise Semantics for Software Modeling techniques. Munich: Technical University Munich, 1998:129-151

[4] Latella D, Majzik I, Massink M. Towards a formal operational semantics of UML statechart diagrams[C]//Ciancarini P, Fantechi A, Gorrieri R. Proceedings of the 3rd International Conference on Formal Methods for Open Object-Oriented Distributed Systems. Boston:Kluwer Academic Publishers, 1999:15-18

[5] Lilius J, Paltor P I. The semantics of UML state machines [R]. TUCS Technical Report No 273, 1999

[6] Baresi L, Pezzè M. On formalizing UML with high-level Petri nets[C]//Cindio F De, Agha G. Concurrent Object-oriented Programming and Petri Nets: Advances in Petri Nets. New york: Springer-Verlag, 2001:276-304

[7] Saldhana J A, Shatz S M, Hu Zh X. Formalization of object behavior and interactions from UML models[J]. International Journal of Software Engineering and Knowledge Engineering, 2001, 11(6): 643-673

[8] Eshuis R, Wieringa R. Requirements-level semantics for UML statecharts[C]//Smith S F, Talcott C L. Processings of Fourth International Conference on Formal methods for open object-based distributed systems. Boston: Kluwer Academic Publishers, 2000: 121-140

[9] 袁崇义. Petri 网原理与应用[M]. 北京:电子工业出版社, 2005: 213-225
Yuan Chongyi. The theory and application of Petri nets [M]. Beijing: Publishing House of Electronics Industry, 2005:213-225(in Chinese)

[10] Busi N. Analysis issues in Petri nets with inhibitor arcs[J]. Theoretical Computer Science, 2002, 275(1/2): 127-177

[11] Störrle H. Models of software architecture[D]. Munich: Institute for Information, Ludwig Maximilians University, 2000

(责任编辑:卢 硕)