定义及验证 UML Statechart 图中的数据流语义

陆公正[1],吴澜波[2],张广泉[3]
LU Gong-zheng[1],WU Lan-bo[2],ZHANG Guang-quan[3]

1.苏州市职业大学 计算机工程系,江苏 苏州 215104
2.苏州卫生职业技术学院 检验药学系,江苏 苏州 215009
3.苏州大学 计算机科学与技术学院,江苏 苏州 215006
1.Department of Computer Engineering,Suzhou Vocational University,Suzhou,Jiangsu 215104,China
2.Department of LAB TEC and Pharmacy,Suzhou Health College,Suzhou,Jiangsu 215009,China
3.School of Computer Science and Technology,Soochow University,Suzhou,Jiangsu 215006,China
E-mail:lugz@jssvc.edu.cn

LU Gong-zheng,WU Lan-bo,ZHANG Guang-quan.Definition and verification of semantics of data flow in UML Statecharts.Computer Engineering and Applications,2009,45(24):56-59.

Abstract: After being integrated with data flow objects,traditional UML Statecharts are not suited for modeling and the verification of the data flow in the workflows due to the lack of the exact semantics of the data flow.To solve the problem,LTS is selected as the semantic field,and the semantics of data flow are defined with SOS in two steps so as to lay the foundation for the verification of the data flow of the workflow.Based on this,temporal logic formula is used to express the properties that the data flow must satisfy,before the verification,an algorithm transforming the UML Statecharts model into reachable state transition graph is given,finally the correctness of the data flow is verified by model checking technique.
Key words: Unified Modeling Language(UML);UML Statecharts;semantics of data flow;temporal logic;verification;model checking

摘 要:在传统的 UML Statechart 图中加入了数据流对象后,因为 UML Statechart 图缺乏精确的数据流语义,所以不适合应用 UML Statechart 图对工作流中的数据流进行建模并验证其正确性。为了解决这一问题,选择标记转换系统(LTS)作为语义域,并用结构化操作语义(SOS)分两步定义了 UML Statechart 图的数据流语义,为工作流中的数据流正确性验证奠定了基础。在此基础上,使用时序逻辑公式表示数据流所需满足的性质,在验证数据流的正确性之前,给出了将它的 UML Statechart 图模型转化为可达状态迁移图的算法,最后通过模型检测算法验证数据流的正确性。
关键词:统一建模语言(UML);UML Statechart 图;数据流语义;时序逻辑;验证;模型检测
DOI:10.3778/j.issn.1002-8331.2009.24.018 **文章编号:**1002-8331(2009)24-0056-04 **文献标识码:**A **中图分类号:**TP311

1 引言

工作流管理模型中,具有这样的数据流机制:管理作为工作流中活动输入的数据和提供作为活动输出的数据;确保工作流模型中活动提供的数据对于模型中需要它的其他活动是有效的;提供确保从一个活动到另一个活动的数据流一致性机制。因此对于数据流的一致性归根结底还是要从语义上去保证。

Harald 根据着色 Petri 网的语义定义了 UML 活动图的数据流语义[1],可以说这是间接的语义定义方法,在进行数据流分析时还需把活动映射到着色 Petri 网, 对于数据流的验证来说极其不便。该文从 UML 状态图自身出发,在其中加入数据流对象,定义 UML 状态图的语法,选择标记转换系统作为语义域,并通过结构化操作语义分两步定义了UML 状态图的数据流语义,为工作流中数据流的正确性验证提供了理论基础。

2 具有数据流对象的 UML 状态图

2.1 具体语法

UML 状态图的基本元素是状态、迁移和动作。状态分为伪状态、基本状态和复合状态。伪状态包括初始状态、终止状态、深度历史和浅度历史状态,该文只涉及前两者;复合状态包括 AND-状态和 OR-状态,它们又包括多个子状态。迁移包括源状态、目标状态、触发事件、布尔条件和动作。

首先,需要在 UML 状态图中加入数据流对象,数据流用对象结点和对象流表示[2]。采用一个对象结点和两个对象流表示依附于控制流的数据流,这种方式很方便,它可以先描述控制

基金项目:国家自然科学基金(the National Natural Science Foundation of China under Grant No.60073020);江苏省高校自然科学研究项目(the Natural Science Foundation of Jiangsu Province of China under Grant No.05KJB520119)。
作者简介:陆公正(1981-),男,讲师,CCF 会员,主要研究方向为形式化方法;吴澜波(1984-),男,助教,主要研究方向为教育技术;张广泉(1965-),男,博士,教授,硕士生导师,CCF 高级会员,主要研究方向为形式化方法、软件工程、XYZ 系统等。
收稿日期:2008-05-08 **修回日期:**2008-08-31

流,然后有选择地根据需要加入数据流。如图 1 所示,用依附于控制流的矩形表示数据结点,数据流没有显式表示。

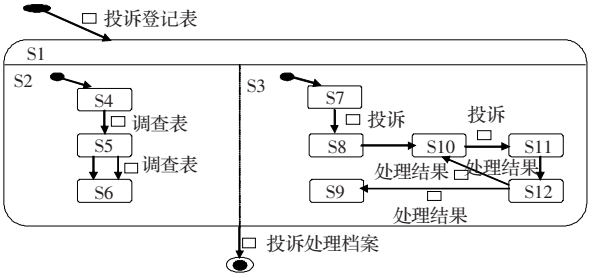

图 1　顾客投诉工作流模型

2.2　抽象语法

定义 1　UML 状态图是六元组 $SC=(S,ON,OF,\rho,type,T)$[3-7],其中:

(1)S 是非空有穷状态集;

(2)ON 是数据结点集,表示状态间输入输出的数据;

(3)OF 是数据流集,表示数据传输的路径;

(4)$\rho:S\to 2^S$ 是状态精化函数,刻画状态间的层次(父/子)关系,$\rho(s)$定义状态 s 的子状态集,在 UML Statechart 图中存在唯一根状态 $root\in S$,且 $\forall s\in S:root\notin\rho(s)$。存在初始状态 $init\in S$ 且 $init\in\rho(root)\wedge\rho(init)=\varnothing$;

(5)$type:S\to\{AND,OR,BASIC,PSEUDO\}$是状态类型函数,其中:

①如果 $\rho(s)\neq\varnothing\wedge type(s)=AND$,$s$ 是 AND 状态,当处于状态 s 时,实际上处于 s 的所有子状态,$\forall s_1\in S:(type(s_1)=AND\Rightarrow\forall s_2\in\rho(s_1):type(s_2)=OR)$;

②如果 $\rho(s)\neq\varnothing\wedge type(s)=OR$,$s$ 是 OR 状态,当处于状态 s 时,实际上处于 s 的一个子状态;

③如果 $\rho(s)=\varnothing\wedge type(s)=BASIC$,$s$ 是基本状态;

④如果 $\rho(s)=\varnothing\wedge type(s)=PSEUDO$,$s$ 是伪状态;

(6)$T\subseteq 2^S\times Lab\times 2^S$ 是所有转换集,对象流集 $OF\subseteq T$。其中 Lab 是迁移标记,它由元组$(source,event,guard,action,on,target)$表示,为了叙述方便,定义 $source(t)$是 t 的源状态,$event(t)$是触发事件,$guard(t)$是监视条件,$action(t)$是动作,$on(t)$是数据结点,$target(t)$是目标状态,其中 $guard$、$action$ 和 on 可以为空。

定义 2　对于 $s,s'\in S$,s 优先于 s'表示为 $s'<s$ 当且仅当 $s\in\rho(s')$。

定义 3　对于 $t,t'\in T$,t 优先于 t'表示为 $\pi t'<\pi t$ 当且仅当 $source(t)\in\rho(source(t'))$。

定义 4　对于 $t,t'\in T$,t 与 t'相冲突表示为 $t\#t'$当且仅当 $t\neq t'\wedge((source(t)<source(t'))\vee(source(t')<source(t)))$。

定义 5　s 的所有当前活跃子状态称为格局,由函数 $Conf:S\to 2^S$ 定义,它计算状态 s 的完全当前格局:

$$conf(s)=\begin{cases}s & \text{if } type(s)=BASIC\\ s\bigcup_{i=1}^{k}conf(s_i) & \text{if } type(s)=AND\wedge s_i\in\rho(s)\\ s\cup conf(s_r) & \text{if } type(s)=OR\wedge s_r\in\rho(s),s_r\text{是默认的当前活跃子状态}\end{cases}$$

2.3　UML 状态图的数据流语义

2.3.1　入口和出口动作

当 UML 状态图的转换 t 发生时,执行动作集合,并传输数据对象:首先执行 $source(t)$的退出动作序列(以内部优先的方法),$source(t)$的数据沿着数据流向外传递,然后执行 t 的动作部分,数据沿着数据流向内传递到 $target(t)$,最后执行 $target(t)$的入口动作序列(以外部优先的方法)。

如果从状态 s_i 到状态 s_j,带有动作 α 和数据 on 的转换发生,那么执行动作序列 $exit(s_i)\bullet trans(on)\bullet\alpha\bullet trans(on')\bullet entry(s_j)$,$\bullet$是连接动作序列的操作符,其中 $trans()$是传递数据的过程,on'可能仍为 on。

定义 6　设 A 为 UML 状态图的所有动作集合,函数 $entry:S\to A^*$ 和 $exit:S\to A^*$ 定义如下:

$$entry(s)=\begin{cases}en \text{ if } type(s)=BASIC,en\text{是状态}s\text{的入口动作}\\ en\bullet entry(s_1)\bullet\cdots\bullet entry(s_k) \text{ if } type(s)=AND\wedge s_1\cdots s_k\in\rho(s)\\ en\bullet entry(s_r) \text{ if } type(s)=OR\wedge s_r\in\rho(s),s_r\text{是默认的当前活跃子状态}\end{cases}$$

$$exit(s)=\begin{cases}ex \text{ if } type(s)=BASIC,ex\text{是状态}s\text{的出口动作}\\ exit(s_1)\bullet\cdots\bullet exit(s_k)\bullet ex \text{ if } type(s)=AND\wedge s_1\cdots s_k\in\rho(s)\\ exit(s_r)\bullet ex \text{ if } type(s)=OR\wedge s_r\in\rho(s),s_r\text{是默认的当前活跃子状态}\end{cases}$$

2.3.2　语义描述

选择 LTS(标记转换系统)作为语义域并使用 SOS(结构化操作语义)[8-9]定义 UML 状态图的数据流语义。下面分两步定义它的数据流语义,第一步先定义辅助语义 AUX:SC→LTS,第二步使用 AUX 定义最终语义 DF:SC→LTS。

定义 7　辅助语义 AUX 由标记转换系统 LTS 给出,它是四元组$(S,s_0,L,\to)$,其中:

(1)S 是格局的集合;

(2)S_0 是初始格局;

(3)$L=EVE\times ACT^*\times GUD\times ON$ 是标记集合,其中 EVE、ACT、GUD、ON 分别是事件集合、动作集合、监视条件集合和数据结点集合;

(4)$\to\subseteq Conf_{all}\times L\times Con_{fall}$ 是转换关系,$Conf_{all}$ 是全部$<Conf,ON>$的集合。

假设集合 P 为所有迁移的集合,它对每个被激发的迁移提供一个约束,表示 P 中不能存在优先级更高的迁移;LE 表示当前格局中所有使能迁移的集合;转换关系→由 5 个 SOS 规则给出(图 2)。

定义 8　最终语义 DF 由标记转换系统 LTS$(S,s_0,L',\to)$给出:

(1)S 是格局集合;

(2)s_0 是初始格局;

(3)$L'=EVE^*\times ACT^*\times GUD\times ON^*$ 是标记集合;

(4)$\to\subseteq Conf_{all}\times L\times Conf_{all}$ 是转换关系。

最终语义的标记集合 L'和辅助语义的标记集合 L 之间的不同是最终语义的转换针对于触发事件序列和数据序列。设事件序列为 ε,数据序列为 θ。首先,定义它们的选择函数 $sele()$和 $seld()$,$sele(\varepsilon)=(e,\varepsilon')$表示在事件序列中选择事件 e,剩下事件序列 ε',$seld(\alpha,\theta)=(on,\theta')$表示根据动作 α 在数据序列中选择数据 on,剩下数据序列 θ'。接下来定义它们的合并函数 $joine()$和 $joind()$,$joine(e,\varepsilon')$表示把事件 e 和事件序列 ε'合并成新的事件序列,$joind(on,\theta')$表示把数据 on 和数据序列 θ'合并成新的数据序列。

例如规则 $\dfrac{s\xrightarrow{e}_t s'}{<C,on>\xrightarrow[\alpha]{e}_t<C',on'>}$,在最终语义下它们应该为

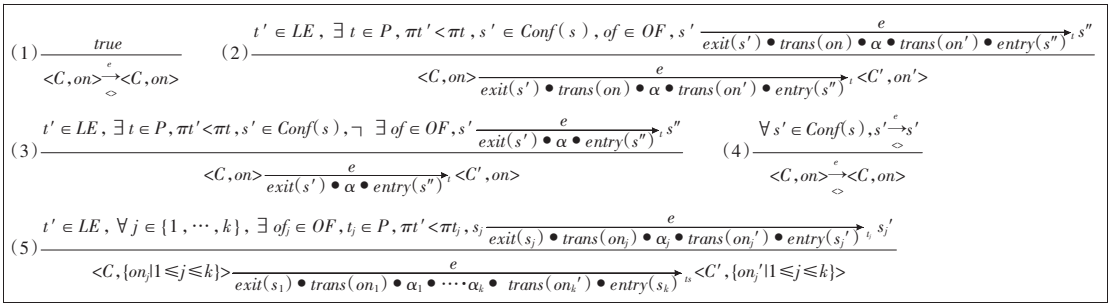

图 2 SOS 规则

如下形式：

$$\frac{s\xrightarrow{e}_t s'}{<C,\theta>\xrightarrow[joine(\alpha,\varepsilon')]{\varepsilon}_t<C',joind(on',\theta')>}\ sele(\varepsilon)=(e,\varepsilon'),seld(\alpha,\theta)=(on,\theta')$$

3 数据流性质表示

为了对数据流的正确性进行验证，使用时序逻辑公式表示数据流所需满足的性质。时序逻辑是对命题逻辑的扩展，它引入了与时间相关的算子，例如□（必然算子）、◇（终于算子）、○（下一时刻算子）和 u（直到算子）。这里所使用的是 CTL（计算树逻辑），它的语法和语义[10]如下。

定义 9 CTL 的结构由三元组 $M=(S,L,\lambda)$给出：

（1）S 是格局集；

（2）$L:S\to 2^{AP}$ 是标记函数，它赋给每一个格局为真的原子命题集合；

（3）λ 是迁移集。

CTL 公式主要有以下几种标准形式，其他的公式都可以转化为这几种形式。

定义 10 CTL 公式 $\phi::=p|\text{false}|\phi_1\to\phi_2|\exists\bigcirc\phi_1|\exists(\phi_1 u\phi_2)|\forall(\phi_1 u\phi_2)$，其中 $p\in AP,\phi_1,\phi_2$ 是 CTL 公式。

定义 11 对于 CTL 结构 M、格局 C 和公式 ϕ，$(M,C)\models\phi$ 定义如下：

$C\models p$ 当且仅当 $p\in L(C)$；

$C\not\models$false 是 $\models$ 的否定；

$C\models\phi_1\to\phi_2$ 当且仅当 $C\not\models\phi_1$ 或者 $C\models\phi_2$；

$C\models\exists\bigcirc\phi$ 当且仅当对于某一 $(C,C')\in\lambda$ 的格局 $C'\in S$ 有 $C'\models\phi$；

$C\models\exists(\phi_1 u\phi_2)$ 当且仅当对于某一有 $C=C_0$ 的路径 $(C_0,C_1,\cdots)$，对于某一 $i\geqslant 0,C_i\models\phi_2$ 且对于 $0\leqslant j<i$ 有 $C_j\models\phi_1$；

$C\models\forall(\phi_1 u\phi_2)$ 当且仅当对于每一有 $C=C_0$ 的路径 $(C_0,C_1,\cdots)$，对于某一 $i\geqslant 0,C_i\models\phi_2$ 且对于 $0\leqslant j<i$ 有 $C_j\models\phi_1$。

为了保证数据流的正确性，必须满足[11]：（1）在到达格局 $<s,on>$ 之前，$<s,on>$ 的所有前继格局必须成功到达且成功输出数据；（2）不允许在并发结构的两个不同分支上存取相同的数据项。其中（1）可以理解为最终可以到达格局 $<s,on>$，CTL 公式表示为 $\forall\diamond<s,on>$，可以转化为标准形式 $\forall(\text{true}\ u<s,on>)$，（2）可以理解为在并发结构的两个分支上不存在写冲突，当两个格局的转换上的动作为写操作时 CTL 公式表示为 $\forall\square\neg(<s_i,on_i>\wedge<s_j,on_i>)$，转化为标准形式 $\neg\exists(\text{true}\ u(<s_i,on_i>)\wedge<s_j,on_i>)$。

4 数据流正确性验证

4.1 可达状态迁移图

为了验证数据流的正确性，首先要根据 UML Statechart 图的数据流语义把 UML Statechart 图模型转化为可达状态迁移图，下面给出转化算法：

（1）把初始状态 i 加入初始格局 $Conf_0$，把它的所有活跃子状态也加入 $Conf_0$，子状态中的数据与 i 中的数据相同；

（2）从事件队列中删除当前触发事件，得到可以被当前事件使能的所有迁移的集合；

（3）检查迁移集合中是否存在冲突的迁移，如果存在则根据迁移间的优先关系进行触发；

（4）检查是否存在数据流，如果不存在，则根据迁移关系到达下一状态；否则，判断数据是否正确输出，完成执行动作（序列）后所得到的数据是否是下一状态所需要的输入，如果都满足则顺利到达下一状态，否则停留在原状态；

（5）如果新格局中没有新状态则算法终止，生成错误信息；

（6）如果格局中都是终止状态，则算法结束，否则跳到（2）。

这里提到的状态是 $<s,on>$ 形式，如果其中任一项不同，则称它为与 $<s,on>$ 不同的状态。

4.2 数据流性质验证

通过上述转化算法，可以得到与 UML Statechart 图模型等价的可达状态迁移图，所以要证明 UML Statechart 图模型满足用 CTL 公式表示的性质 F，只需证明等价的可达状态迁移图 G 满足 CTL 公式 F，而 $G\models F$ iff $Conf_0\models F$[12]。

下面给出验证数据流正确性的模型检测算法，算法的基本思想是为每个格局 $Conf$ 标注在它为 true 时的 F 的子公式，如果最后以公式 F 标注初始格局 $Conf_0$，那么工作流模型满足性质 F。其中 $label(Conf)$ 表示为格局 $Conf$ 标注命题集合，$sub(F)$ 表示 F 的所有子公式，$G,Conf\models F$ iff $F\in label(Conf_0)$。

```
Function CheckCTL( )
for each φ∈sub(F) do
    Case φ
        φ∈AP:skip
        φ=¬f:for each Conf∈Conf_all do
                if f∉label(Conf) then label(Conf)=label(Conf)∪φ
endif
                repeat
        φ=f∨g:for each Conf∈Conf_all do
                if f∈label(Conf) or g∈label(Conf) then label(Conf)=label(Conf)∪φ
                endif
```

repeat

$\phi=\exists\bigcirc f$:for each $Conf_1, Conf_2 \in Conf_{all}$ do

if $(Conf_1, Conf_2) \in R$ and $label(Conf_2)=f$ then

$label(Conf_1)=label(Conf_1)\cup\phi$ endif

repeat

$\phi=\exists(f\,\boldsymbol{u}\,g)$:for each $Conf_i, Conf_j \in Conf_{all}$ do

if there exists a path from $Conf_i$ to $Conf_j$ and $label(Conf_j)=g$ and $label(Conf_i)=f$ then $label(Conf_i)=label(Conf_i)\cup\phi$ endif

repeat

$\phi=\forall(f\,\boldsymbol{u}\,g)$:for each $Conf_i, Conf_j \in Conf_{all}$ do

for every path from $Conf_i$ to $Conf_j$ do

if $label(Conf_j)=g$ and $label(Conf_i)=f$ then

$label(Conf_i)=label(Conf_i)\cup\phi$ endif

repeat

repeat

if $F \in label(Conf_0)$ then return YES else return NO endif

ENDCheckCTL

Function VerDataflow( )

//如果是性质(1)

if property1 then call CheckCTL( ) endif

//如果是性质(2)

if property2 then

for each $<s_i, on_i>, <s_j, on_j>$ do

//判断数据项是否相等并属于写资源集合 WS,如果是则调用函数 CheckCTL( )检查

if $on_i==on_j$ and $on_i \in WS$ and $on_j \in WS$ then call CheckCTL( ) endif

EndVerDataflow

5　相关工作分析与比较

现有工作流建模方法有工作流图、Petri 网和 UML 等,取得较多成果的是工作流图和 Petri 网。工作流图的优点是简单、直观,与自然工作流过程很相近,但缺点是缺乏形式化语义,不利用对其建立的工作流模型进行验证。而 Petri 网是一种形式化工具,具有坚实的数学基础,只是它需要有一定理论知识的人才能理解。UML 现在已经成为面向对象的标准,它也是一种标准建模语言,在各个领域都得到广泛应用,其中的 UML Statechart 图描述功能强大、结构紧凑且具有较高程度的形式化语义,它在工作流建模和验证方面的应用前景被人们所看好。目前,工作流图和 Petri 网主要使用在工作流中的控制流的建模和验证,而完整的工作流验证除了控制流验证之外,还需要进行数据流和时序约束等方面的验证。工作流图和 Petri 网没有提供描述数据流的相关机制,跟它们相比,UML 中的活动图较适合描述数据流,但由于它是半形式化的,不利于进行验证。而 UML Statechart 是基于状态转换的,能够通过自动验证技术进行验证,但因为其缺乏精确的数据流语义,所以该文定义了它的数据流语义,为数据流的验证奠定了基础。数据流验证是一项复杂和困难的工作,该文只是针对它所需满足的正确性标准进行了验证,将来还需要针对数据流的其他方面例如类型匹配等进一步深入研究下去。

6　结束语

工作流建模和验证已经成为工作流的重要研究领域之一,基于 UML Statechart 图的方法又在其中占有重要地位,但在进行工作流的数据流正确性验证方面,由于 UML Statechart 图缺乏精确的数据流语义,难以利用 UML Statechart 对数据流进行建模和验证。为了解决这一问题,选择标记转换系统(LTS)作为语义域并通过结构化操作语义(SOS)分两步定义了 UML Statechart 图的数据流语义,这样为基于 UML Statechart 图的工作流的数据流正确性验证奠定了基础。在验证数据流正确性的过程中,应用了模型检测技术,它的优点是精确、自动化程度高并能在出错时提供反例。把形式化技术应用在工作流的验证方面,必将推动形式化技术的发展,而形式化技术的进一步发展又将提供解决验证工作流正确性过程中所遇到问题的新途径和新方法。

参考文献:

[1] Harel D,Naamad A.The STATEMATE semantics of Statecharts[J].ACM Transactions on Software Engineering and Methodology,1996,5(4):293-333.

[2] Storrle H.Semantics and verification of data flow in UML 2.0 activities[J].Theoretical Computer Science,2005,127(4):35-52.

[3] 李留英,王戟,齐治昌.UML Statechart 图的操作语义[J].软件学报,2001,12(12):1864-1873.

[4] 蒋慧,林东,谢希仁.UML 状态机的形式语义[J].软件学报,2002,13(12):2244-2250.

[5] Beeck M.Formalization of UML-Statecharts[C]//Gogolla M.UML 2001.Berlin:Springer-Verlag,2001:406-421.

[6] Varro D.A formal semantics of UML Statecharts by model transition systems[C]//ICGT2002.Berlin:Springer-Verlag,2002:378-392.

[7] Schettini A M,Peron A,Tini S.A comparison of Statecharts step semantics[J].Theoretical Computer Science,2003,290(1):465-498.

[8] Bhaduri P,Remash S.Model checking of statechart models:Survey and research directions[C]//CoRR,cs.SE/0407038,2004.

[9] Latella D,Majzik I,Massink M.Automatic verification of a behavioural subset of UML Statechart digrams using the SPIN model-checker[J].Formal Aspects of Computing,1999,11(6):637-664.

[10] Alur R,Courcoubetis C,Dill D L.Model-checking in dense real-time[J].Information and Computation,1993,104(1):2-34.

[11] 林信男.维护工作流程时间限制一致性之研究[D].高雄:国立中山大学,2000.

[12] Berard B,Bidoit M,Finkel A,et al.Systems and software verification:Model-checking techniques and tools[M].Berlin:Springer-Verlag,2001,104(1):2-34.