基于DL-Safe规则的UML状态图形式化和一致性验证

何红悦， 宋自林， 周 波

(解放军理工大学指挥自动化学院 南京 210007)

摘要：为了对UML状态图进行形式化验证，将状态图中的语义分为静态语义和动态语义两部分，用描述逻辑知识库表示静态语义，用DL-Safe规则表示动态语义.研究了检查UML状态图一致性的算法，该算法能够用DL-Safe规则对知识库进行推理达到检查状态图一致性的目的，最后分析了算法的可行性.

关键词：UML状态图；描述逻辑；DL-Safe规则；一致性

中图分类号：TP 181 **文章编号**：1671－6841(2009)01－0094－05

0 引言

统一建模语言(Unified Modeling Language，UML)是对象管理组织推荐的第3代面向对象语言.系统设计人员可以使用UML定义的多种框图从静态、动态和结构3个方面对系统进行建模分析[1].UML中有5个框图供设计人员进行动态建模，状态图是其中重要的一个，它不仅可以建模表示系统的动态语义，还可以使用正向工程来创建可运行的系统[1].系统设计人员在用状态图进行建模时会遇到状态冲突、孤立状态等不一致性问题，但现有的UML工具并不支持检查这些不一致性.模型一致性是系统设计人员对系统进行正确建模的必要前提，因此，对模型进行一致性检查至关重要.目前，国内外学者针对状态图的形式化验证提出了多种研究方法：文献[2]首先将状态图进行时间扩展，然后将其转换为时间自动机，再使用模型检测技术对其进行验证；文献[3]中提出了一种基于扩展层次自动机的状态图形式化验证方法；文献[4]则将状态图中的动态语义用描述逻辑表示出来，通过推理工具来验证其一致性.

DL-Safe规则是一种特殊的Horn规则，解决了OWL-DL知识库和规则结合起来推理的不可判定性问题，OWL-DL知识库和DL-Safe规则结合起来进行推理是可判定的[5].OWL-DL语言的语义表示能力同描述逻辑SHOIN(D)是相同的[6]，并且有软件Protégé支持从描述逻辑到OWL-DL本体知识库的转换，因此，描述逻辑SHOIN(D)知识库和DL-Safe规则结合起来进行推理也是可判定的.作者用描述逻辑知识库和DL-Safe规则对状态图中的语义进行形式化，然后通过推理检查状态图的一致性.

1 UML状态图语义

状态图向用户展示了一个对象或系统的状态机.状态机则指定了一个对象或系统在其生命周期内的状态序列，及引起状态序列中状态之间转换的相关触发事件.文献[1]将一个基本的状态机分为状态(States)和转换(Transitions)两部分.状态是对象在其生命周期内的一种处境，在该处境下对象完成某一动作或等待某一事件发生.状态包含名称、进入/退出结果、内部转换、子状态和延期事件五部分，可分为简单状态、复杂状态、初始状态和终止状态.对于UML2.0中的状态分支和状态合并也称为状态的特例，它们和终止状态一起被称为伪状态.本文只考虑简单状态、初始状态和终止状态.转换是状态与状态之间的关系，它表示对象在

收稿日期：2008-12-01

基金项目：国家自然科学基金资助项目，编号10501053.

作者简介：何红悦(1985－)，男，硕士研究生，主要从事数据库应用技术研究，E-mail：hehy2008@sina.com；通讯联系人：宋自林(1944－)，男，教授，博士生导师，主要从事数据库及知识表示研究.

某种条件下由一个状态转移到另一状态,包含源状态、目标状态、触发事件、守护条件和后效五部分.

对于状态图,人们更多地关注于状态之间的合理转换,通过状态的合理转换来实现系统目标.因此,本文忽略了和状态图转换无关的一些语义信息,如状态内部的转换和动作等,只考虑状态和转换内部的源状态、目标状态、触发事件和守护条件.

由于忽略状态内部包含的语义,并且状态机内的状态都属于同一对象,可以从状态机内的所有状态提取出一个状态类 State,然后把所有的状态表示为该类的个体,同理可以提取守护条件类 GuardCondition 和触发事件类 EventTrigger,将所有的守护条件和触发事件声明为相应类的个体,将源状态和目标状态看作是状态转换关系中的一种角色表示.声明关系 Transition 表示状态与状态之间的转换关系.虽然守护条件和触发事件是转换的组成部分,但转换是在两个状态之间进行的,因此状态和守护条件与触发事件也有关系,将这些关系定义为 RelatedCondition 和 RelatedEvent,RelatedCondition 表示状态和守护条件之间的关系,RelatedEvent 表示状态和触发事件之间的关系.

通过以上分析可以将状态机定义如下:$SM'=\langle$ State, GuardCondition, EventTrigger, Transition, RelatedCondition, RelatedEvent, $S, G, E\rangle$.其中,State, GuardCondition, EventTrigger 分别是从状态机中提取出的状态类、守护条件类和触发事件类,Transition, RelatedCondition, RelatedEvent 分别是从状态机中提取出的状态之间的转换关系、状态与守护条件之间的关系和状态与触发事件之间的关系,Transition(a,b)表示状态 a 可以转换到状态 b,RelatedCondition(a,c)表示状态 a 与守护条件 c 相关,包含了 a 作为源状态和目标状态两种情况,RelatedEvent(a,e)表示状态 a 与触发事件 e 相关,也包含了 a 作为源状态和目标状态两种情况,S, G, E 分别是由状态类、守护条件类和触发事件类的个体组成的相应集合,其中,初始状态 Initial 和终止状态 Final 也作为状态类的个体包含在集合 S 中.对于无守护条件的转换定义特殊的个体 NullGC 表示此处的守护条件,NullGC $\in G$.对于无触发事件的转换定义特殊的个体 NullET 表示此处的触发事件,NullET $\in E$.

由上述定义得到的状态机 SM' 只包含了状态机内的静态语义,无法表示出状态之间的动态转换关系,即状态 a 在什么条件下可以转换到状态 b,同时也不能表示出触发事件的先后顺序.对于状态机中的动态语义本文用规则表示,首先定义一个类 Active 表示活动的状态,定义关系 Before 表示触发事件的先后顺序关系.定义规则如下:

规则 1　Active$(b)\leftarrow$
Active$(a)\wedge$Transition(a,b)
$\wedge$RelatedCondition(a,c)
$\wedge$RelatedCondition(b,c)
$\wedge$RelatedEvent(a,e)
$\wedge$RelatedEvent(b,e)
$\wedge$RelatedCondition$(b,c1)$
$\wedge$RelatedEvent(a,e)
$\wedge$RelatedEvent(b,e)
$\wedge$Transition(b,d)
$\wedge$RelatedCondition$(b,c2)$
$\wedge$RelatedCondition$(d,c2)$

规则 2　Before$(e,f)\leftarrow$
Active$(a)\wedge$Transition(a,b)
$\wedge$RelatedCondition$(a,c1)$
$\wedge$RelatedEvent(b,f)
$\wedge$RelatedEvent(d,f)

规则 1 表示活动的状态 a 在触发事件 e 发生并且守护条件 c 存在时可以装换到下一状态 b.规则 2 表示若活动状态 a 可以经过状态 b 转换到状态 d,则其中涉及到的触发事件 e、f 的执行顺序是 e 要先于 f 执行.令 $R=\{$规则 1,规则 2$\}$.

综上所述将一个完整的状态机的语义可以定义为:$SM=\langle SM', R\rangle$,其中,SM' 表示状态机的静态语义,R 表示状态机的动态语义.

2　用描述逻辑表示状态图静态语义

由于状态图中的静态语义只包含类和关系以及它们的个体,因此可以用描述逻辑知识库来表示.对 SM' 中的类和关系的定义可以用 Tbox 中的公理表示如下:

$State \subseteq \forall RelatedCondition \cdot GuardCondition$
$State \subseteq \forall RelatedEvent \cdot EventTrigger$
$State \subseteq \forall Transition \cdot State$
$GuardCondition \subseteq \forall RelatedCondition^{\sim} \cdot State$
$EventTrigger \subseteq \forall RelatedEvent^{\sim} \cdot State$

其中,$RelatedCondition^{\sim}$、$RelatedEvent^{\sim}$分别为关系 RelatedCondition、RelatedEvent 的逆关系.

对 SM'中的个体集合可以用描述逻辑知识库中的 Abox 表示,对集合 S、G、E 中的每一个个体都添加相应的类公理和属性公理,例如对 S 中的个体 a,要添加 State(a)类公理,同时添加 Transition(a,b)、RelatedCondition(a,c)、RelatedEvent(a,e)属性公理,其中,b 是任一与 a 有转换关系的状态个体,且 b 是作为目标状态,c 是任一与 a 有关系的守护条件个体,e 是任一与 a 有关系的触发事件的个体.以图 1 为例,可以用描述逻辑知识库将其中的静态语义表示如下:

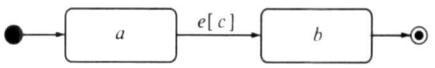

图 1 简单状态图

Fig.1 Simple state diagram

$State \subseteq \forall RelatedCondition \cdot GuardCondition$
$State \subseteq \forall RelatedEvent \cdot EventTrigger$
$State \subseteq \forall Transition \cdot State$
$GuardCondition \subseteq \forall RelatedCondition^{\sim} \cdot State$
$EventTrigger \subseteq \forall RelatedEvent^{\sim} \cdot State$
State(Initial), State(Final), State(a), State(b)
GuardCondition(NullGC), GuardCondition(c)
EventTrigger(NullET), EventTrigger(e)
RelatedCondition(Initial, NullGC)
RelatedEvent(Initial, NullET)
RelatedEvent(a, NullET)
RelatedEvent(a, e)
RelatedCondition(a, NullGC)
RelatedCondition(a, c)
RelatedEvent(b, NullET)
RelatedEvent(b, e)
RelatedCondition(b, NullGC)
RelatedCondition(b, c)
RelatedCondition(Final, NullGC)
RelatedEvent(Final, NullET)
Transition(Initial, a)
Transition(a, b)
Transition(b, Final)

3 用 DL-Safe 规则表示状态图动态语义

由描述逻辑知识库表示的 SM'和规则集合 R 结合起来进行推理具有不可判定性,需要将规则集合 R 中的规则转换为 DL-Safe 规则[7].DL-Safe 规则一个特殊的 Horn 规则,是在 Horn 规则上添加限制条件构成的.设 KB 为一个描述逻辑 SHOIN(D)知识库,N_c 为概念集合,N_{Ra}为抽象关系集合,N_{Rc}为具体关系集合.将 $A(s)$中 $A \in N_c$ 或 $R(s,t)$中 $R \in N_{Ra} \cup N_{Rc}$称为一个 DL 项.若规则 r 中的任一变量在规则 r 的规则体中都非 DL 项存在,则称规则 r 为 DL-Safe 规则.

参考 DL-Safe 规则的定义可知,规则集合 R 中的 2 个规则都不是 DL-Safe 规则,因为规则中的变量在规则体中相关的谓词全是 DL 项.为了将规则表示为 DL-Safe 规则,对规则中的任一变量 x,在规则体中添加非 DL 项谓词 $O(x)$,相应地还要在相关知识库中的 Abox 中对规则涉及到的变量个体 y 添加 $O(y)$,Abox 中的 $O(y)$表示个体 y 是知识库中明确存在的个体.使用上述方法将规则集合 R 中的规则修改为 DL-Safe 规则,修改后的 DL-Safe 规则如下:

规则 1 $Active(b) \leftarrow Active(a) \wedge Transition(a,b) \wedge RelatedCondition(a,c) \wedge RelatedCondition(b,c) \wedge RelatedEvent(a,e) \wedge RelatedEvent(b,e) \wedge O(a) \wedge O(b) \wedge O(c) \wedge O(e)$

规则 2 $Before(e,f) \leftarrow Active(a) \wedge Transition(a,b) \wedge RelatedCondition(a,c1) \wedge RelatedCondition(b,c1) \wedge RelatedEvent(a,e) \wedge RelatedEvent(b,e) \wedge Transition(b,d) \wedge RelatedCondition(b,c2) \wedge RelatedCondition(d,c2) \wedge RelatedEvent(b,f) \wedge RelatedEvent(d,f)$

$$\wedge O(a) \wedge O(b) \wedge O(d) \wedge O(e) \qquad \wedge O(f) \wedge O(c1) \wedge O(c2)$$

相应地对知识库中的 Abox 添加相关个体 y 的非 DL 项谓词 $O(y)$,这样得到 SM'的知识库,和修改后的规则集合 R 结合起来推理就具有了可判定性,可以通过推理来检查状态图的一致性.

4　检查一致性

4.1　检查状态的可达性

某一状态的可达性指初始状态通过转换可以到达该状态.规则 1 中定义 Active 类表示活动状态类.由于状态图默认第一个活动状态为初始状态,如果状态图中的某一状态能够在初始状态为活动状态的情况下,经过推理得到该状态也为活动状态,则表明该状态是可达的.算法 1 如下:

1)将初始状态设定为活动状态,即向知识库的 Abox 中添加 Active(Initial)公理.

2)对知识库进行推理:

①若推理产生了新的 Active(a)公理,将该公理添加到知识库的 Abox 中,重新执行步骤 2).

②若推理没有产生新的 Active(a)公理,推理结束.

3)对 Abox 中的每个类公理 State(a),检查是否存在 Active(a).若存在状态个体 a,在 Abox 中有 State(a)公理,但没有 Active(a)公理,表明状态 a 是不可达的.

4.2　检查状态的终止性

将状态的终止性定义为该状态经过转换可以到达终止状态.和检查状态的可达性类似,若将状态图中的某一状态设定为活动状态,在此条件下,经过推理可以得出终止状态为活动状态,表示该状态是可终止的.算法 2 如下:

1)删除知识库 Abox 中的所有 Active(a)公理,$a \in S$.

2)对要检查终止性的状态个体 b,向 Abox 中添加 Active(b)公理.

3)对知识库进行推理.

①若推理产生了新的 Active(a)公理,将该公理添加到知识库的 Abox 中,重新执行步骤 3).

②若推理没有产生新的 Active(a)公理,推理结束.

4)检查 Abox 中是否有 Active(Final)公理,若有该公理,表明状态 b 是可终止的;否则,表明 b 是不可终止的.

4.3　算法可行性分析

描述逻辑知识库和规则结合起来推理是不可判定的[5-6].文献[7]提出了 DL-Safe 规则,并理论证明了描述逻辑知识库和 DL-Safe 规则结合起来推理的可判定性.在此基础上本文提出了检查状态图一致性的算法,算法中的主体是对知识库和 DL-Safe 规则实施推理,且算法每次执行时推理的次数不会超过状态个体的数目.因为推理是可判定的,状态机中状态个体数目是有限的,所以算法是可行的.

5　结束语

本文分析认为状态图中的语义可以分为静态和动态两部分.对于静态语义可以用描述逻辑知识库表示,对于动态语义可以用 DL-Safe 规则表示,并且二者结合起来推理是可判定的,可以通过不同的 DL-Safe 规则来表示状态图不同的动态语义.

参考文献:

[1]　Booch G, Rumbaugh J, Jacobson I. UML 用户指南[M]. 2 版. 邵维忠,译. 北京:人民邮电出版社, 2006.

[2]　张广泉,陆公正,戎玫. 时间 UML-Statecharts 建模的工作流时序约束的一致性验证[J]. 计算机科学, 2006, 33(11): 98-101.

[3]　刘晓建,李战怀. 基于扩展层次自动机的 UML 状态图完备性和一致性检验[J]. 微电子学与计算机, 2008, 25(1): 39-44.

[4]　Straeten Van Der R. Inconsistency management in model-driven engineering[D]. Brussels: Vrije University, 2005.

[5] Horrocks I, Patel-Schneider P E. A proposal for an OWL rules language[C]//Proceedings of the 13th Int'l World Wide Web Conference. New York: ACM Press, 2004.

[6] Levy A Y, Rousset M C. Combining Horn rules and description logics in CARIN[J]. Artificial Intelligence, 1998, 104(1/2): 165-209.

[7] Motik B, Sattler U, Studer R. Query answering for OWL-DL with rules[C]//Proceedings of the 3rd International Semantic Web Conference. Berlin: Springer, 2004: 549-563.

Formalization and Consistency Checking of UML-Statechart Based on DL-Safe Rule

HE Hong-yue, SONG Zi-lin, ZHOU Bo

(*Institute of Command Automation, PLA University of Science & Technology, Nanjing* 210007, *China*)

Abstract: The semantic information of UML-Statechart is divided into static aspect and dynamic aspect. The static aspect is expressed by a knowledge base of description logics, and the dynamic aspect is expressed by DL-Safe rule. An algorithm is proposed for checking the consistency of UML-Statechart, which can use the DL-Safe rule to reason the knowledge base. Finally, the feasibility of the algorithm is analyzed theoretically.

Key words: UML-Statechart; description logic; DL-Safe rule; consistency

---

(上接第93页)

Online Filtering Method and Optimization Based on SMO and E-mail Fingerprint

ZHU Qing-rong[1], DONG Shou-bin[2], CHEN Bin[1]

(1. *School of Computer Science and Engineering, South China University of Technology, Guangdong* 510640, *China*; 2. *Guangdong Computer Network Key Lab, Guangdong* 510640, *China*)

Abstract: The finger features vectoring and the SVM filtering method are proposed on the spam, and an online SVM spam filter called FSVM is designed and implemented, which attains the corresponding performance as the classic method in the online spam filtering. In view of the computing speed of SVM filtering, a dynamic example set reduction method(DFSVM) is given out for SVM filtering method based on the original SMO algorithm, which can greatly reduce the computing cost and keep the corresponding performance. The experiment and comparison test running on the standard corpus is given out in the actual mail system. It is proved that the optimization can improve the accuracy of SVM classification.

Key words: spam filtering; SVM; conditional relax; dynamic subset