基于时间自动机的UML模型转换与验证研究

姬莉霞， 马建红

(郑州大学 软件技术学院 河南 郑州 450002)

摘要: 针对无法对UML模型进行形式化验证的问题,提出在元模型层将UML模型转换为时间自动机模型并进行验证的方法.形式化UML状态机的结构,抽象出UML和时间自动机的元模型,利用模型转换语言ATL对UML元模型和时间自动机元模型构造映射规则,实现UML模型到时间自动机模型的转换,在模型验证工具Uppaal中对转换结果进行形式化验证.最后进行实例研究,结果表明了此方法的有效性和先进性.

关键词: 统一建模语言; 模型验证; 模型转换; Uppaal; 时间自动机; 元模型

中图分类号: TP 311.5 **文献标志码:** A **文章编号:** 1671-6841(2013)01-0050-06

DOI: 10.3969/j.issn/1671-6841.2013.01.013

0 引言

基于模型的系统设计方法已经得到广泛的研究与应用,UML(unified modeling language)是一种面向对象的建模语言,运用统一的、标准化的标记和定义实现对软件系统进行面向对象的描述和建模,是应用最广泛的软件建模语言,被业界称为软件分析和设计的标准语言.但UML是非形式化的,无法对其模型进行形式化分析和验证,很难保证系统建模的安全性和正确性.形式化方法[1]是一种为硬件和软件系统提供精确规约、设计和验证的严格的数学技术和方法.为保证建模的正确性,对系统模型进行形式化验证十分必要.将系统的UML模型转换为形式化模型进行验证的方法是软件工程领域研究的热点[2].

模型验证是一种应用非常广泛的形式化验证技术,将UML与模型验证技术相结合能够有效避免软件设计错误,保证建模的正确性.本文结合国内外有关研究,提出一种将UML模型转换为时间自动机模型并进行验证的方法.首先将非形式化的UML类和状态机图转换为形式化的描述逻辑,接着基于元模型的转换思想,结合UML类和状态机语义抽象出UML元模型,同时根据时间自动机的形式化语义构造时间自动机元模型,通过模型转换语言ATL构造两者之间的转换映射规则,实现从UML模型到时间自动机模型的转换,并将转换得到的时间自动机模型在模型验证工具Uppaal中进行形式化验证,最后结合自动取款机(auto trade machine,ATM)对该方法进行实例研究和分析.

1 模型验证方法

模型验证的基本思想是采用状态空间搜索来验证给定的计算模型是否满足某种时态逻辑公式所表示的特定性质.在模型验证过程中,使用有穷状态模型建模,用某种时态逻辑公式表示性质规约,然后通过有效的搜索来验证所建立的模型是否满足该性质规约,这个问题是可判定的[3].与其他形式化方法比较,模型验证的优点非常显著,它有完整的数学理论基础,适用于软件工程、通信协议、嵌入式系统等多种领域.

Uppaal[4-5]是一种常用的实时系统模型验证工具,主要通过快速搜索机制来验证系统规范和可达性,其主要优点表现为高效性和便捷性.它适用于可以被描述为非确定的并行过程的时间自动机积的系统,已成功应用于多个领域.Uppaal使用简化的CTL(computational tree logic)描述系统模型需要满足的性质[4,6],它可以描述系统的安全性和活跃性,使用p和q表示系统的性质,则Uppaal中验证性质可表示为

收稿日期: 2012-11-28
基金项目: 河南省科技攻关计划项目,编号122102210518;河南省教育厅科学技术研究项目,编号12A520042.
作者简介: 姬莉霞(1979-),女,讲师,硕士,主要从事形式化方法与验证研究,E-mail:jilixia@zzu.edu.cn.

Prop ::= p | $E<>p$ | $A\blacksquare p$ | $E\blacksquare p$ | $A<>p$.

p 表示通往,p 为真,当且仅当在转换系统中存在从 p 到 q 的序列.

$E<>p$ 表示可能的,也就是某些路径中存在状态满足 p, $E<>p$ 为真,当且仅当在转换系统中从初始状态 s_0 开始存在一个序列 $s_0\rightarrow\cdots\rightarrow p$.

$A\blacksquare p$ 表示必然的,即所有路径的所有状态都满足 p,等价于 not $E<>$ not p.

$E\blacksquare p$ 表示潜在必然的,即某些路径中所有状态满足 p,在一个转换系统中,$E\blacksquare p$ 为真,当且仅当存在一个序列 $s_0\rightarrow\cdots\rightarrow s_i\rightarrow\cdots$,使得 p 在所有状态 s_i 中有效.

$A<>p$ 表示最终的,即所有路径中都有状态满足 p,等价于 not $E\blacksquare$ not p.

2　构建 UML 的形式化语义

UML 是非形式化的,为了对其进行有效的验证,必须给出精确的形式化语义[7]. 本文研究 UML 的一种轻量级形式化描述,即针对待解决问题忽略一些 UML 模型构造子,但这些处理不会对 UML 的概念语义一致性产生影响. 在构建 UML 的形式化语义之前,参考文献[7]的做法,先做出如下一些约定.

2.1　规约

1) 在本文方法中,一个 UML 模型中主要包含类和状态机图. 类不支持操作调用和子类,其静态结构由类图描述,其动态行为被定义为行为的状态机.

2) UML 为建模提供了 2 种同步机制: 通过信号通信或者对象在状态机中的正交区域通信,本文采用第 1 种方法,即对象之间通过使用信号进行通信,信号可以关联参数.

3) 在执行语义上,同一时间最多有一个完成转换(没有明确触发的转换) 可以被触发,这个限制保证每一步骤最多有一个区域改变状态,这有利于简单转换算法的实施.

4) UML 允许执行连续的活动状态,但本文不支持此做法,因为模型检查工具将会离散地执行模型.

5) 不支持历史伪态、交叉或连接伪态、进入或者退出活动状态.

2.2　状态机语义

状态机包含状态和状态之间的转换,参考文献[7-8]构造状态机的形式化语义.

定义 1　$H=\langle\sum_s,\sum_i,\sum_{ch},\sum_f,\sum_c,R,\text{top},\searrow\rangle$ 是一个状态层次当且仅当 $\searrow\ \sum_c\ R\cup R\ \sum$,并且 $\langle\sum\cup R,\searrow\rangle$ 是树根,它的树叶集合是 $\sum\setminus\sum_c$. 这里 $\sum$ 是状态集合,它包含简单状态集合 $\sum_s$、开始伪态集合 $\sum_i$、选择伪态集合 $\sum_{ch}$、结束状态集合 $\sum_f$ 和复合状态集合 $\sum_c$; R 是有限域集合; 子关系 $\searrow$ 表示,如果复合状态 c 的一个域 r 直接包含一个状态 s,那么 $c\searrow r$ 并且 $r\searrow s$.

定义 2　在给定状态层次上的一个转换是一个 5 元组,

$$t=\langle s,e,g,a,S'\rangle\quad(\sum\setminus\sum_f)\quad(E\cup\{u\})\quad L_g\quad L_a\quad 2^{\sum},$$

其中,s 是转换的源状态,记为 source(t) ; e 是转换的触发条件,记为 trigger(t) ; g 是转换的卫士(约束) 条件,记为 guard(t) ; a 是转换触发的更新动作,记为 effect(t) ; S' 是转换的目标集合,记为 target(t) .

在 UML 图形表示中,转换被表示为从源状态 source(t) 到目标状态 target(t) 的箭头. 转换伴有文本标签: trigger [guard] /effect,即“触发 [卫士] /动作”. 一个转换只有在其触发事件被调度时才能发生,触发trigger(t) 是一个信号,转换通过接收信号(完成转换接收特殊符号 u, u 在图中省略) 而触发. 转换还关联一个卫士 guard(t) ,缺省时默认为 true,卫士是布尔表达式,表示转换发生的先决约束条件. 转换可以伴随更新效果动作 effect(t) ,它在转换发生时被执行,缺省时不做任何更新.

定义 3　一个 UML 状态机是一个 4 元组 $\langle H,\phi,E,\text{deferrable}\rangle$,这里 H 是状态层次; ϕ 是 H 上的转换; E 是有限事件集合; deferrable: $\sum\ \ 2^E$,是从状态集合到事件集合 E 的子集的映射,表示可被延迟发生的事件.

3　模型转换

3.1　模型转换模式

模型转换在 MDE(model driven engineering) 中扮演着重要角色,传统的模型转换大多是基于模型层的转

换,因此转换规则和转换关系很难重用,而基于元模型的模型转换可以有效地解决此问题[2]. MDE 的模型转换遵循图 1 所示的统一模式[3],M1、M2 和 M3 分别表示模型层、元模型层和元元模型层. Tab 是一个转换程序,它自动创建从源模型 Ma 到目标模型 Mb 的转换. Tab、Ma 和 Mb 分别是遵从元模型 MMt、MMa 和 MMb 的定义,MMt 对应于转换语言的抽象语法. 3 个元模型 MMt、MMa 和 MMb 遵从元元模型 MMM 的定义.

本文使用 ATL[3] 进行模型转换,它是 ATLAS 转换语言,用于指定模型到模型的转换,是 ATLAS 模型管理架构 AMMA(ATLAS model management architecture) 平台的一部分,建立在的 OCL(object constraint language) 形式化之上. ATL 扮演模型转换模式中 MMt 的角色,它在元模型层定义转换规则,将 MMa 的元素映射到 MMb 上,并据此实现低层模型的自动转换.

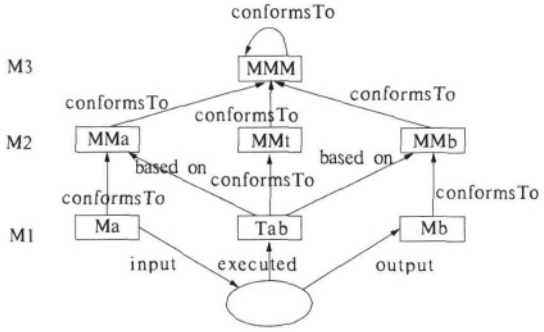

图 1 模型转换模式

Fig. 1 Model transformation pattern

3.2 UML 元模型

建模就是用模型元素来描述系统,通常需要捕捉系统的静态结构和动态行为. 在 UML 模型中,系统的静态结构由类图来表示,类图描述类以及类之间的关系;系统的动态行为由状态图进行描述,状态图描述一类对象的所有可能的状态以及事件发生时状态的转移条件.

根据 UML 模型的静态结构和动态行为的特征,结合类图元模型与状态机元模型,形成一个如图 2 所示的静态与动态相结合的 UML 元模型.

3.3 时间自动机元模型

Uppaal 中使用的模型是时间自动机,对 Alur 等提出的时间自动机[10] 进行了扩展,使之拥有用于同步的数据变量,并增加了紧急位置、坚定位置和紧急通道等.

定义 4 一个时间自动机是一个 8 元组,

$$TA = (L, l_0, C, A, E, I, V, T),$$

这里 L 是一个有限位置集合;$l_0 \in L$ 是开始位置集合;C 是有限时钟集合;A 是一个有限动作集合;

$$E \quad L \times \Phi(C) \times A \times 2C \times L$$

是边的集合,边一般伴随有动作、卫士条件和复位时钟集合;

$$I: L \rightarrow \Phi(C)$$

是位置的不变式;V 是有穷变量集合;

$$T: L \times A \rightarrow L$$

是一个转换函数.

虽然时间自动机拥有形式化定义,但进行模型转换必须构造其元模型,并实现 UML 元模型和时间自动机元模型的同构化. 图 3 给出了 Uppaal 中时间自动机的元模型,它与 UML 元模型是同构的,都遵从于 MMM 元元模型.

多个时间自动机的积组成时间自动机网络,它关联 1 个或多个时间自动机,所有的时间自动机都由 Uppaal 中的模板(Template) 表示. 一个模板可能包含若干个位置(Location)、边(Edge)、时钟(Clock) 和参数(Parameter). 位置又可能为开始位置(Initial Location)、紧迫位置(Urgent Location) 和坚定位置(Committed Location). 时钟、参数和数值等可构成各种表达式(Expression),包括布尔表达式(BoolExp)、同步表达式(SynExp) 和赋值表达式(AssignExp). 布尔表达式包括边的卫士条件(Guard) 和位置的不变式(Invariant);同步表达式是触发边转换的动作;赋值表达式是边转换后的更新(Update).

3.4 转换的映射

ATL 模型转换是基于图 1 所示的模型转换模式进行的,它在元模型层定义映射规则. 根据 UML 元模型以及时间自动机元模型的结构,将 UML 验证模型中的主要元素映射为时间自动机模型的元素,如表 1 所示.

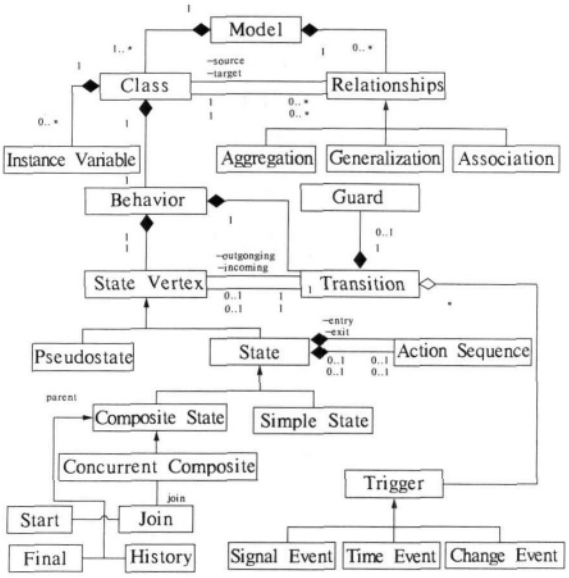

图 2　UML 元模型示意图

Fig. 2　Diagram of UML meta-model

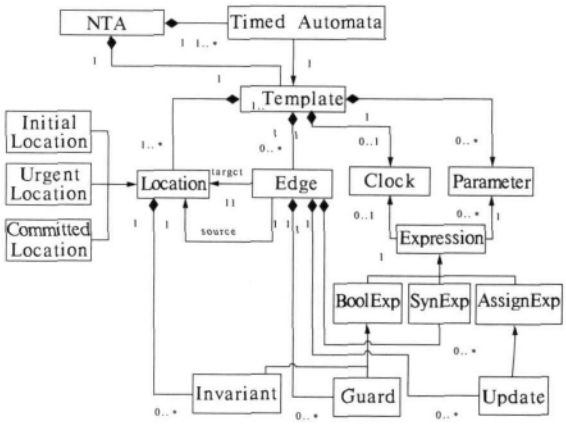

图 3　时间自动机元模型示意图

Fig. 3　Diagram of timed automata meta-model

表 1　UML 到价格时间自动机的映射

Tab. 1　Mapping from UML to TA

UML	时间自动机	说明
Class	Template	类是静态结构描述,映射为时间自动机的 Template
State Machine	Template	状态机包含 1 个或多个 Region,映射为 Template
State	Location	State 表示状态机的状态,映射为 TA 的位置 Location
Pseudostate	Initial Location	状态机的伪状态中的初始状态映射为 TA 的初始位置
Transition	Edge	转换边之间的映射
Guard	Guard	状态和位置卫士条件的映射
Trigger	SynExp	Trigger 表示触发变迁的动作,映射为 TA 边的事件 SynExp
Behavior	AssignExp	状态机中触发后引起的操作映射为 TA 中边的动作 AssignExp

4　实例研究

下边以 ATM 取款机的主要取款操作为例对前文方法进行实例研究. 该实例主要涉及 2 个类 ATM 和 Bank,类之间通过一对一的引用相互关联. 结合前文 UML 元模型,对 ATM 取款问题进行建模,得到图 4 所示的 ATM 和 Bank 的类和状态机图,它们共同构成验证模型基础.

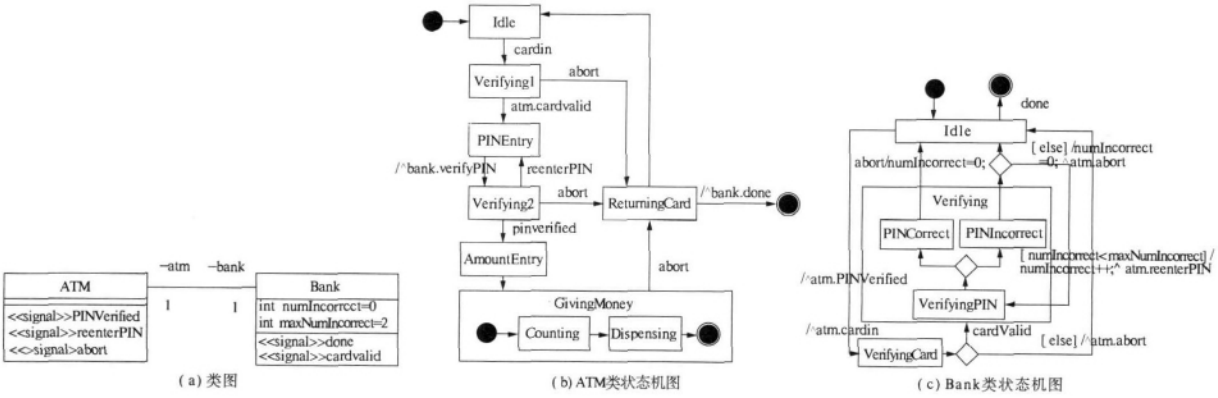

图 4　ATM 的 UML 模型

Fig. 4　UML model of ATM

接着进行模型转换,ATL是Eclipse环境下的一个插件,通过ATL的模型转换,将非形式化的UML模型转换为形式化的时间自动机模型,转换结果为xmi格式的文件.将生成的时间自动机模型的xmi文件转换成xml文件,并将其作为Uppaal的输入,得到时间自动机模型.转换后可以适当在模型中增加约束以提高系统的精确性,比如约束ATM机退卡10个时间单位后转入空闲状态,ATM机在确认钱数后60个时间单位内出钱等,调整后的时间自动机模型如图5所示.

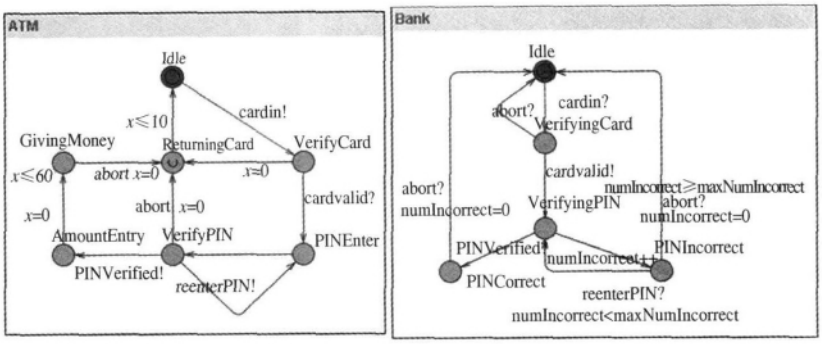

图5 ATM的时间自动机模型

Fig. 5 Timed automata model of ATM

在Uppaal的模拟器中可以对图5所示的时间自动机模型进行追踪模拟,得到多种运行轨迹,据此初步判断模型的安全性和正确性.接着在验证器中,通过简化的CTL对系统要满足的性质协议(如:A ▮ not deadlock,确保系统无死锁;A ▮ ATM.GivingMoney imply ATM.$x\leqslant 60$,ATM机在确认钱数后60个时间单位内出钱等)进行验证,性质的验证结果如图6所示.

与常用的将UML模型转换为PROMELA模型,并使用工具SPIN进行模型验证的方法相比较[11-12],本文方法可以对系统性质进行多方面验证,而不仅仅是验证系统是否出现不期望出现的交互行为.另外,Uppaal中可视化的时间自动机模型可进行直观的模拟和追踪,并且模型中的时钟可以弥补UML在时间控制方面的不足,更加适用于有时间约束的实时系统等复杂系统.

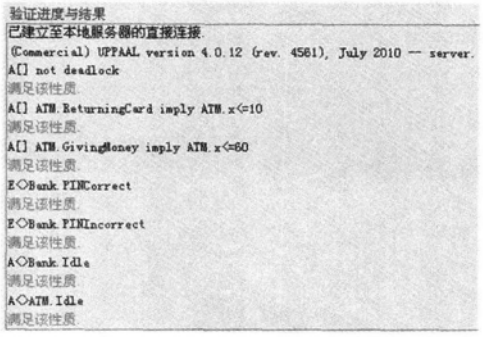

图6 验证结果

Fig. 6 Verification results

5 结束语

结合国内外有关研究,构造UML状态机的形式化语义,对UML和时间自动机的形式化语义进行元建模,利用模型转换语言ATL在元模型层完成从UML到时间自动机模型的转换.最后结合ATM取款机对该方法进行实例研究,对ATM机取款主要操作进行UML建模并完成模型转换,将得到的时间自动机模型在Uppaal中进行追踪模拟以及安全性和正确性验证.

今后将就此继续展开研究,后续工作重点主要是对UML交叉或连接伪态、进入或者退出活动状态等先进建模概念的转换研究,以及系统时间约束的进一步研究.

参考文献:

[1] Clarke E M, Wing J M. Formal methods: state of the art and future directions [J]. ACM Computing Surveys, 1996, 28(4): 626 – 643.

[2] 吉鸣, 黄志球, 祝义,等. 基于 MDA 的实时软件资源建模与模型转换的方法 [J]. 计算机科学,2011, 38(8): 137 – 141.

[3] 林惠民, 张文辉. 模型检测: 理论、方法与应用 [J]. 电子学报, 2002, 30(12A): 1907 – 1912.

[4] Gerd B, Alexandre D, Kim G L. A tutorial on Uppaal [C] //Proc of the 4th Int'l School on Formal Methods for the Design of Computer, Communication, and Software Systems. Heidelberg: Springer-Verlag, 2004: 200 – 236.

[5] 周清雷, 姬莉霞,王艳梅. 基于 UPPAAL 的实时系统模型验证 [J]. 计算机应用, 2004,24(9): 129 – 131.

[6] 姬莉霞,马建红,周清雷. 混杂系统的扩展时间自动机模型及验证研究 [J]. 计算机工程与应用, 2012,48(8): 73 – 78.

[7] Toni J, Jori D, Tommi J, et al. Model checking dynamic and hierarchical UML state machines [C] //Proceedings of MoDeV^{2}a. Heidelberg: Springer-Verlag, 2006: 94 – 110.

[8] 何红悦, 宋自林, 周波. 基于 DL-Safe 规则的 UML 状态图形式化和一致性验证 [J]. 郑州大学学报: 理学版, 2009, 41(1): 94 – 98.

[9] Jouault F, Allilaire F B J. ATL: a model transformation tool [J]. Science of Computer Programming Special Issue on Second Issue of Experimental Software and Toolkits(EST), 2008, 72(1/2): 31 – 39.

[10] Alur R, Dill D L. A theory of timed automata [J]. Theoretical Computer Science, 1994, 126(2): 183 – 235.

[11] Alexander K, Merz S. Model checking and code generation for UML state machines and collaborations [C] //Proc 5th Workshop on Tools for System Design and Verification. Reisenburg, 2002: 59 – 64.

[12] Timm S, Alexander K, Merz S. Model checking UML state machines and collaborations [J]. Electronic Notes in Theoretical Computer Science, 2001, 55(3): 357 – 369.

Research on Model Transformation and Model Checking of UML Based on Timed Automata

JI Li-xia, MA Jian-hong

(*Software Technology School, Zhengzhou University, Zhengzhou* 450002, *China*)

Abstract: There was no existing formal verification method for UML models. A method was developed to transform UML models into timed automata models that were verified in meta-model level. The structure of UML state machines was formalized and the meta-models of UML and timed automata were constructed in this method. Model transformation language ATL was used to construct mapping rules from UML meta-model to timed automata meta-model, and to transform UML models into timed automata models. The transformation result was verified by model checking tool Uppaal. A case was studied and it was shown that this method was effective and advanced.

Key words: UML; model checking; model transformation; Uppaal; timed automata; meta-model