UML Statecharts 的模型检验方法*

董 威+, 王 戟, 齐治昌

(国防科学技术大学 计算机学院,湖南 长沙 410073)

An Approach of Model Checking UML Statecharts

DONG Wei+, WANG Ji, QI Zhi-Chang

(School of Computer, National University of Defence Technology, Changsha 410073, China)

+ Corresponding author: Phn: 86-731-4573673, Fax: 86-731-4573637, E-mail: dong.wei@263.net

http://www.nudt.edu.cn

Received 2002-03-22; Accepted 2002-08-14

Dong W, Wang J, Qi ZC. An approach of model checking UML Statecharts. *Journal of Software*, 2003,14(4): 750~756.

Abstract: UML (unified modeling language) has been widely used in software development. Verifying if a UML model meets the required properties has become a key issue. An approach of model checking UML Statecharts is given in this paper. At first, the UML Statecharts is structurally expressed by extended hierarchical automata. Then, the deduction rules of operational semantics are defined. The correctness of operational semantics can be ensured through finding the maximal non-conflict transition set. For the system with infinite runs, the operational semantics can be mapped to a Büchi automaton. Linear temporal logic properties of the system can be verified based on the automata theory of model checking. The method of verifying a complex multi-object system modeled by Statecharts and collaboration diagram is also presented in this paper.

Key words: UML (unified modeling language); Statecharts; model checking; ω-automata

摘 要: 统一建模语言 UML 已广泛应用于软件开发中,验证 UML 模型是否满足某些关键性质成为一个重要问题.提出了对 UML Statecharts 进行模型检验的方法.首先用扩展层次自动机结构化地表示 UML Statecharts,然后给出其操作语义,通过寻找最大无冲突迁移集可以保证语义的正确性.对于具有无穷运行的系统,该操作语义可以映射到一个 Büchi 自动机.使用基于自动机理论的模型检验方法来验证 UML Statecharts 的线性时态逻辑性质,并给出方法验证由 Statecharts 和协同图建模的复杂多对象系统.

关键词: UML (unified modeling language);Statecharts;模型检验;ω自动机

中图法分类号: TP311 **文献标识码**: A

统一建模语言 UML(unified modeling language)[1]是一种描述能力强大且含义直观的可视化建模语言.基于

* Supported by the National Natural Science Foundation of China under Grant Nos.69973051, 60233020, 90104007 (国家自然科学基金); the National High-Tech Research and Development Plan of China under Grant No.2001AA113202 (国家高技术研究发展计划(863)); the Huo Ying-Dong Education Foundation of China under Grant No.71064 (霍英东青年教师基金)

第一作者简介: **董威**(1976－),男,陕西咸阳人,博士,讲师,主要研究领域为软件自动验证,软件测试.

UML 的开发方法和软件建模环境已经广泛应用于各种软件开发过程中,其中包括航空航天、武器装备、汽车等关键领域的系统开发.UML Statecharts 是 Harel Statecharts[2]的一种变体,刻画了对象在其生命周期中的行为和状态变迁,在软件分析和设计建模过程中占有重要的地位,而对其进行正确性验证以判断设计规范是否满足目标需求也成为一个关键问题.

模型检验(model checking)是一种重要的自动验证技术,主要通过显式状态搜索或隐式不动点计算来验证有穷状态并发或实时系统的模态/命题性质.近来已出现了一些对 Statecharts 进行模型检验的研究.文献[3]以扩展层次自动机(extended hierarchical automaton,简称 EHA)作为中间模型,把 Harel Statecharts 转换为模型检验工具 SPIN 或 SMV 的输入语言来进行验证.文献[4]把 UML Statecharts 转换为 SPIN 的输入语言.文献[5]用已有验证环境 JACK 的标准数据格式表示 UML Statecharts 对应的标记迁移系统(labeled transition system,简称 LTS).上述方法都是把 UML Statecharts 翻译为现有模型检验工具的输入语言后加以验证的,这对于 Statecharts 模型不是最自然、最有效的方法,翻译之后会引入较多冗余,而且由于语义的不同需要额外的处理.

本文首先对 EHA 的定义进行了一些扩充,从而包括 UML Statecharts 的一些主要特点和基本功能,并对 EHA 的操作语义进行了相应的改进.然后对从 EHA 得到的、以状况(stastus)为基础的 LTS 直接用基于自动机理论的模型检验方法验证 UML Statecharts 的线性时态逻辑(linear temporal logic,简称 LTL)性质.多数已有的 Statecharts 模型检验方法把系统中的并发对象看作是一个 Statecharts 中的并发状态,这对于存在多个并发对象的复杂系统并不自然.本文给出一种方法验证由 Statecharts 和协同图建模的多对象系统,每个对象被看作是一个异步进程.我们已经开发出一个环境:UML-MC,可以验证建模工具 Rational Rose 或 I-Logix Rhapsody 建立的 UML 模型.

1 用 EHA 表示 UML Statecharts

UML Statecharts 的语法和语义在文献[1]中都是以精确的自然语言来描述的,文献[5,6]给出了形式化定义.图 1 是一个 TV 控制器的例子,其中 TV_SYS 和 ON 是 AND-状态,TV,USER,OFF,IMAGE 和 SOUND 是 OR-状态,其余为 BASIC-状态.$t1$ 和 $t2$ 是层间迁移.

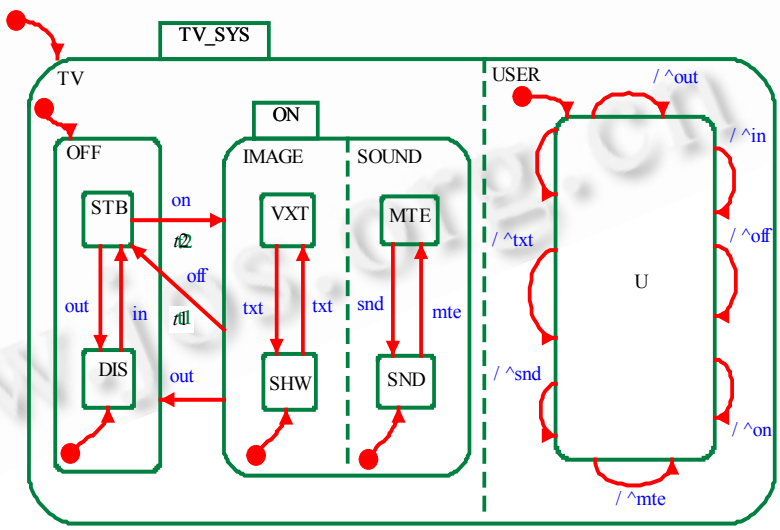

Fig.1 Statecharts of TV controller
图 1 TV 控制器的 Statecharts

根据语义解释执行 UML Statecharts 时需要一个动作语言,在 UML 标准中未对其进行定义.我们定义了一个基本的动作语言,能够对整型和布尔变量进行赋值和算术运算;可以定义==,<,>,<>等条件表达式,并能进行逻辑运算;可通过 ^*obj.e* 这种形式向对象 *obj* 发送事件 *e*(^*e* 表示发送给任何可以接受 *e* 的对象).

模型检验要求得到被验证系统的 LTS,这就需要对 UML Statecharts 定义一种结构化的语义.层间迁移是体现 UML Statecharts 强大、灵活功能的一个重要因素,但却破坏了 UML Statecharts 的结构化,极大地复杂化了操作语义.文献[7]把 EHA 作为 UML Statecharts 的中间模型.EHA 可以被看作 Statecharts 的抽象语法,抽取掉了语法细节,而只保留 Statecharts 的关键部分,且以一种结构化的方式表示出来.本文对 EHA 进行了一些扩充,以包括 UML Statecharts 的一些主要特点和一些基本功能,如能响应外部事件的开放模型;动作语言除了产生事件以外,

还可以对整型和布尔变量进行运算;为每个状态添加入口和出口动作等.

1.1 从UML Statecharts到EHA

EHA 由简单的顺序自动机组成,状态通过精化函数映射到一组对其进行细化的(并发)顺序自动机.

定义 1(顺序自动机). 顺序自动机 A 是一个四元偶$(\sigma_A,s^0_A,\lambda_A,\delta_A)$,$\sigma_A$ 是有穷的状态集合,s^0_A是初始状态,λ_A 是有穷的迁移标记集合,$\delta_A\subseteq\sigma_A\times\lambda_A\times\sigma_A$ 是迁移关系.

定义 2(扩展层次自动机,EHA). EHA H 是一个五元偶(F,E,ρ,A_0,V),F 是一个有穷的顺序自动机集合,$\forall A_1,A_2\in F,\sigma_{A1}\cap\sigma_{A2}=\varnothing$;$E$ 是有穷事件集合;V 是变量集合;$\rho:\cup_{A\in F}\sigma_A\to 2^F$ 是精化函数,给出 F 的一个树型结构满足:(1) 存在惟一的根自动机 $A_0\in F$,使得不存在 $s\in\cup_{A\in F}\sigma_A,A_0\in\rho(s)$;(2) 每个非根自动机恰有一个父状态:对于$\forall A\in F\backslash\{A_0\},\exists_1 s\in\cup_{A'\in F\backslash\{A\}}\sigma_{A'},A\in\rho(s)$;(3) 不存在环路:$\forall S\subseteq\cup_{A\in F}\sigma_A,\exists s\in S,S\cap(\cup_{A\in\rho(s)}\sigma_A)=\varnothing$.

在把 UML Statecharts 转换为 EHA 的过程中,层间迁移提升到它离开和进入的最高层状态之间,从而变为非层间迁移.为了不改变语义,需要对迁移标记进行扩充,添加受限源状态 sr 和确定目标状态 td 这两个集合.sr 将迁移的使能限制在源状态下的一个格局中,td 决定了迁移使系统进入目标状态时有哪些子状态也同时进入.事件集合 E 和变量集合 V 分别由 UML Statecharts 中所有事件和变量组成.

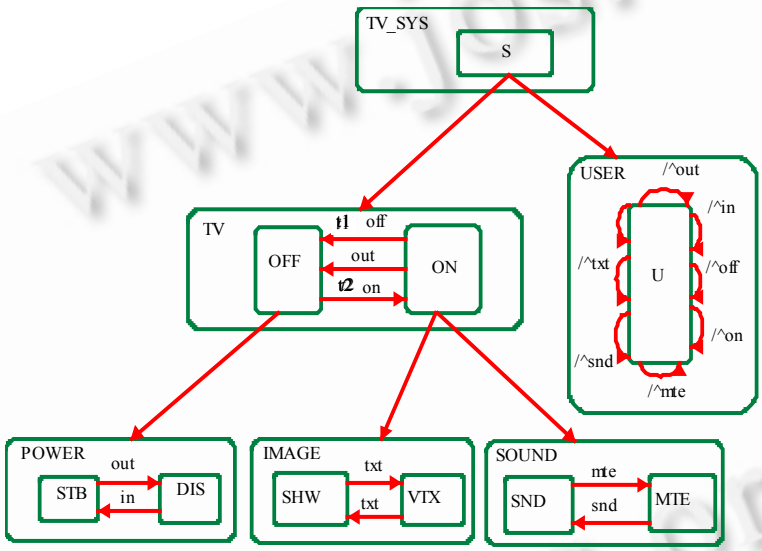

Fig.2 Corresponding EHA of TV controller
图 2 TV 控制器对应的 EHA

图 2 是图 1 中 Statecharts 对应的 EHA,其中 $F=\{TV\_SYS,TV,USER,POWER,IMAGE,SOUND\}$,$\rho(S)=\{TV,USER\}$,$\rho(OFF)=\{POWER\}$,$\rho(ON)=\{IMAGE,SOUND\}$.迁移 $t1$ 的 td 为 $\{STB\}$,$t2$ 的 sr 也为 $\{STB\}$.图中粗框的状态为其顺序自动机的初始状态.

层次自动机的全局状态由格局来表示,它由包含的顺序自动机的某些局部状态组成.$Conf_H$表示由EHA H中所有的有效格局组成的集合.

定义 3(格局). H 的一个格局是一个集合 $Conf\subseteq\cup_{A\in F}\sigma_A$ 使得:(1) $\exists_1 s\in\sigma_{A_0}$ 满足 $s\in Conf$;(2) $\forall s,A$,若 $s\in Conf$ 且 $A\in\rho(s)$,则$\exists_1 s'\in A,s'\in Conf$.

1.2 EHA的操作语义

EHA 的操作语义可以用一个 LTS 来定义,包含通过迁移相关联的一组状况(status).每个状况由格局和当前的环境组成,环境包括当前事件队列中存在的事件及其顺序和当前所有变量的取值,它们分别称为事件环境和变量环境.UML 标准中没有对事件队列的类型进行限定,这里也同样不固定事件队列的类型.对于事件集合 X,用 ΘX 表示在 X 上某种类型的事件队列(例如 FIFO 队列或集合等)所有可能的结构.对于变量集合 Y,ΩY 表示对 Y 中的变量所有可能的赋值.

图 1 中 TV 的例子不响应 Statecharts 以外的事件,是一个封闭模型.能对外部实体发出的消息进行响应的模型称为开放模型,例如,如图 3 所示的哲学家就餐问题中的哲学家 Statecharts.以前对 Statecharts 进行模型检验的研究多数基于封闭模型[4,5],但当系统中存在多个并发对象交互时,封闭模型语义就不足以描述了.下面,我们基于文献[7]的工作,用 LTS 对经过扩充的 EHA 定义操作语义,这里考虑开放模型(包括了封闭模型).在其操作语义中,当处于某一个状况时,除了从事件队列中选择所有可使能某些迁移的事件以外,还要从环境事件队列中非确

定选择所有可使能某些迁移的事件.这是因为,对于单个 Statecharts,其环境的行为是不可预料的.这样,所有可能的后继状况都被找到,从中非确定地选择下一个到达的状况.例如对于图 3,当处于格局 {*Hungry*,*WaitLeft*,*WaitRight*} 时,除了考虑当前事件队列中的事件和完成事件以外,还需要考虑若事件 *LeftForkGranted* 或者 *RightForkGranted* 属于当前环境事件队列的情况.

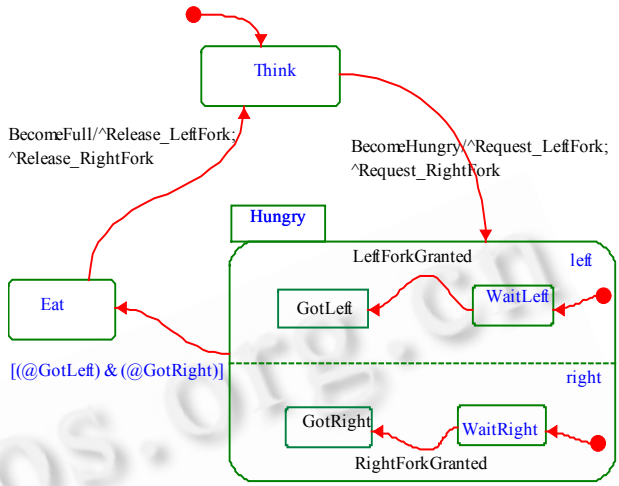

Fig.3 Statecharts of a philosopher
图 3 哲学家的 Statechart

定义 4(EHA 的操作语义). 一个 EHA H 的操作语义是一个 LTS $Ts=(S,s_0,L,\rightarrow)$,其中 $S=Conf_H\times\Theta X\times\Omega Y$ 是状况集合,$s_0=(C_0,E_0,V_0)$ 是初始状况,$L:S\rightarrow 2^{AP}$ 为每个状况标注一组原子命题,$\rightarrow\subseteq S\times S$ 是迁移关系.

Ts 中的一个迁移是 Statecharts 中通过选择最大无冲突迁移集完成的一个 RTC 步[1].关系"$\rightarrow$"通过一组演绎规则进行定义.我们基于文献[7]的结果,针对开放模型、变量环境和出口、入口动作等,扩充定义了这些规则.这些操作语义规则将在附录中给出.

2 验证 UML Statecharts 的 LTL 性质

时态逻辑已在实践中表现出对规约并发系统的性质非常有用,因为它们可以在不明确引入时间的情况下描述事件发生的顺序.描述 UML Statecharts 性质的 LTL 公式包含 3 种形式的原子命题:(1) x rop c,其中 rop 是关系符(例如>,<,≠等),当变量 x 和常量 c 满足 rop 时命题为真;(2) $@s_i$,其中 s_i 是状态名,当 UML Statecharts 的当前活跃格局包括 s_i 时命题为真;(3) ^e,当前状况的事件队列中包含事件 e 时为真.建立 LTL 公式的规则及其语义可参见文献[8].对于 TV 的例子,LTL 公式 φ_1:**G**(^*off* ->**X**(@*OFF*))表示,不论何时,当 *off* 事件产生时,系统在下一步中都位于状态 OFF.

ω-正规序列是描述并发进程和其他程序的一种很自然的方式,具有描述ω-正规序列的能力对程序验证是至关紧要的.接收ω-正规语言的 Büchi 自动机文献[9]为一个五元偶 $A=(S,s_0,\Sigma,\rho,F)$,其中 S 是状态集合,$s_0\in S$ 是初始状态,Σ是字母表,$\rho:S\times\Sigma\rightarrow 2^S$ 是一个非确定的迁移函数,$F\subseteq S$ 是接收状态集合.无穷字$\alpha=a_1a_2\ldots$Büchi 自动机 A 中的一个路径为一个无穷序列 $s_0,s_1,\ldots$,其中 s_0 是初始状态且对所有 $i\geq 1$ 有 $s_i\in\rho(s_{i-1},a_i)$.对于一个路径 $s_0,s_1,\ldots$,当有某一接收状态无穷次经常重复出现,即对某一个 $s\in F$ 有无穷多个 i 使 $s_i=s$ 时,该路径是可接收的.当在 A 中有无穷字α的一个可接收路径时,α对 A 是可接收的.从 Büchi 自动机的定义可知,一个 Büchi 自动机非空当且仅当有 $s\in F$ 从初始状态可达且从自身可达.

要验证 UML Statecharts 的无穷行为的性质,首先要把其 EHA 的操作语义模型 $Ts=(S,s_0,L,\rightarrow)$转换为一个等价的 Büchi 自动机 $A_s=(S,s_0,\Sigma,\rightarrow_s,S)$.其中$\Sigma=\cup_{s\in S}L(s)$;若$(s,t)\in\rightarrow$,则 $t=\rightarrow_s(s,L(t))$;接收集为整个状态集 S;状态集和初始状态保持不变.

自动机方法的一个重要理论基础是时态逻辑公式 f 可以转换为一个表示相同无穷状态序列的ω自动机(如 Büchi 自动机).R.Gerth 和 D.Peled 等人提出了一个为 LTL 公式否定命题建立 Büchi 自动机的高效算法[10],它能接受所有不满足 f 的无穷字.

假定 $A_{\neg f}=(V,v_0,\Sigma,\rho,F)$是为公式$\neg f$建立的 Büchi 自动机,$A_s=(S,s_0,\Sigma,\rightarrow_s,S)$是从 UML Statecharts 的操作语义得到的 Büchi 自动机,则 $A_s\times A_{\neg f}$是根据下面规则得到的一个 Büchi 自动机(S',s_0',Σ,τ,F'):

- $S'=S\times V$ 为状态集;
- $s_0'=(s_0,v_0)$为初始状态;

- Σ与 A_s 和 $A_{\neg f}$中的字母集相同;
- 迁移函数$\tau:(S\times V)\times\Sigma\to 2^{S\times V}$定义为:$(s_2,v_2)\in\tau((s_1,v_1),a)$当且仅当 $v_2\in\rho(v_1,a)$且 $s_2\in\to_s(s_1,a)$;
- $F'=S\times F$ 为接收状态集.

于是,UML Statecharts 的模型检验问题的主要步骤总结如下:(1) 把 UML Statecharts 表示为相应的 EHA 模型;(2) 根据 EHA 的操作语义得到对应的 LTS 并转换为 Büchi 自动机 A_s;(3) 为线性时态逻辑公式 f 的否定命题建立 Büchi 自动机 $A_{\neg f}$;(4) 计算 A_s 和 $A_{\neg f}$的乘积自动机;(5) 检查该乘积自动机所能接受的语言是否非空.

所得到的乘积自动机接收所有可被自动机 A_s 接收但违背公式 f 的路径.乘积自动机的非空问题可以通过检查是否存在至少一个环路包含一个接收状态.解决该问题的一个经典方法是 Tarjan 的深度优先搜索检查图的强连通[11],但也可以把它简化为简单的环路检测.系统满足 f 当且仅当乘积自动机为空,否则可以给出违反该性质的反例路径.我们已经开发出一个验证环境 UML-MC,可以对 Rational Rose 或 I-Logix Rhapsody 建立的 Statecharts 进行模型检验.通过对 TV 控制器模型的验证,可以确定该设计规范满足前面所给的性质φ_1.

3 验证复杂多对象系统

上面对 UML Statecharts 进行验证时只考虑了单独的一个 Statecharts,即使一个系统中有多个并发对象,它们也被当作一个 Statecharts 中的并发状态来建模.这种方法使得复杂系统模型的结构变得很不自然,并且 UML Statecharts 中并发状态的语义使得这些相对独立的对象的活动复杂化.UML 协同图可以描述相互协作的并发对象间的动态交互关系和对象之间存在的某些联结关系[1].协同图可以分为两种形式:实例层和规范层.我们在这里用规范层的协同图描述系统中存在的并发对象以及相互之间的关联、通信和数目对应关系,各个对象之间通过发送事件进行通信和同步.

图 4 中给出了哲学家就餐问题的规范层协同图.哲学家类的 Statecharts 在图 3 中已给出,它从驱动对象接收 *BecomeHungry* 和 *BecomeFull* 事件,并从资源管理器接收 *LeftForkGranted* 和 *RightForkGranted* 事件,这些都是外部事件.

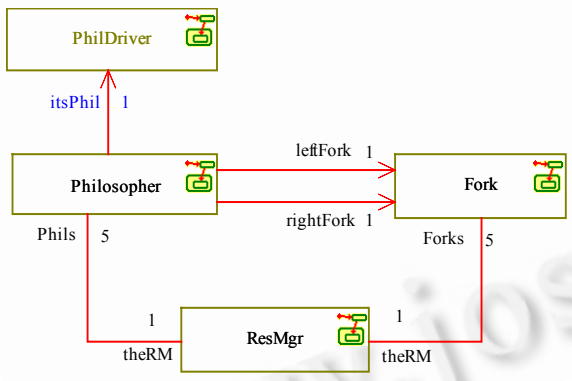

Fig.4 Collaboration diagram of philosopher dinner model
图 4 哲学家就餐模型的协同图

这种描述只能从整体上反映整个系统和各对象之间的关系,不能以一种严格的方式从对象的行为推导出整个系统的精确模型,而这是验证所必需的.考虑到许多系统中并发对象相对都较为独立,对象之间只通过消息传递进行交互,我们在此把每个对象当作系统中的一个进程,进程之间是用异步方式来通信的,每个进程的一步执行就是一个 RTC 步.

假定协同图中 n 个对象的 Statecharts 对应的操作语义分别为 $M_n=(S_n,s_{n_0},L_n,\to_n)$,则可以得到整个系统的标记迁移图 $M=(S,s_0,L,\to)$,其中:

(1) $S\subseteq S_1\times\ldots\times S_n$;

(2) $s_0=(s_{1_0},\ldots,s_{n_0})$;

(3) 对任意$(s_1,\ldots,s_n)\in S$,$L((s_1,\ldots,s_n))=L_1(s_1)\cup\ldots\cup L_n(s_n)$;

(4) 对任意$(s_1,\ldots,s_k,\ldots,s_n)\in S$,若$(s_k,s_k')\in\to_k$ $(1\le k\le n)$且该迁移对应的 RTC 步的激发事件满足下面条件之一:(i) 激发事件是完成事件;(ii) 激发事件是该对象的内部事件;(iii) 激发事件为外部事件,且已被某个对象 $M_i(1\le i\le n,i\ne k)$产生并且未被消耗掉;(iv) 激发事件为外部事件且由协同图中所有对象之外的环境产生,那么$(s_1,\ldots,s_k',\ldots,s_n)\in S$ 且$((s_1,\ldots,s_k,\ldots,s_n),(s_1,\ldots,s_k',\ldots,s_n))\in\to$.

由此得到的系统的 LTS M 就可以用第 2 节中的方法进行验证.例如,对于哲学家就餐问题可以验证性质 **G**(^*Philosopher*[*i*].*BecomeHungry*->**F**(@*Philosopher*[*i*].*Eat*)).该 LTL 公式表示,如果某一个哲学家饿了,那么他肯定可以在以后吃到食物.对于死锁问题,比如每个哲学家都持有左边的叉子而等待右边的叉子,也可以被检

测到.

4 结 论

已有的几种对 UML Statecharts 进行模型检验的方法都是在把其翻译为现有模型检验工具的输入语言之后再验证,这对于 Statecharts 模型不是最自然、最有效的方法.本文给出通过 EHA 结构化地描述 UML Statecharts 及其语义的方法,然后用基于自动机理论的模型检验方法验证 UML Statecharts 的 LTL 性质,这是一种较为直接和自然的方法.此外,我们还给出了方法验证用 Statecharts 和协同图建模的多对象系统.

References:

[1] OMG unified modeling language specification (Version 1.4). Needham: Object Management Group, Inc., 2001.

[2] Mikk E, Lakhnech Y, Petersohn C, Siegel M. On formal semantics of Statecharts as supported by STATEMATE. In: Duke D, Evans A, eds. Proceedings of the 2nd BCS-FACS Northern Formal Methods Workshop. London: Springer-Verlag, 1997.

[3] Mikk E, Lakhnech Y, Siegel M, Holzmann GJ. Implementing Statecharts in PROMELA/SPIN. In: Proceedings of the Workshop on Industrial-Strength Formal Specification Techniques (WIFT'98). Washington, DC: IEEE Computer Society, 1998, 90~101. http://www.computer.org/cspress/CATALOG/pr00081.htm.

[4] Lilius J, Paltor IP. vUML: a tool for verifying UML models. In: Hall R, Tyugu E, eds. Proceedings of the 14th IEEE International Conference on Automated Software Engineering. Washington, DC: IEEE Computer Society, 1999. 255~258.

[5] Gnesi S, Latella D. Model checking UML statecharts diagrams using JACK. In: Proceedings of the 4th IEEE International Symposium on High-Assurance Systems Engineering. Washington, DC: IEEE Computer Society, 1999. 46~55. http://www.computer.org/cspress/CATALOG/pr00418.htm.

[6] Lilius J, Paltor IP. The semantics of UML state machines. Technical Report, No.273, Turku: Turku Centre for Computer Science, TUCS, 1999.

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

[8] Puneli A. The temporal semantics of concurrent programs. Theoretical Computer Science, 1981,13:45~60.

[9] Courcoubetis C, Vardi M, Wolper P, Yannakakis M. Memory-Efficient algorithms for the verification of temporal properties. Formal Methods in System Design, 1992,1(2-3):275~288.

[10] Gerth R, Peled D, Vardi MY, Wolper P. Simple on-the-fly automatic verification of linear temporal logic. In: Dembinski P, Sredniawa M, eds. Proceedings of the 15th International Symposium on Protocol Specification Testing and Verification. Boston: Kluwer Academic Publishers, 1995. 3~18.

[11] Clarke EM, Grumberg O, Peled DA. Model Checking. Cambridge, MA: MIT Press, 1999.

附录: EHA 操作语义的演绎规则

本附录基于文献[7]并针对本文中对 EHA 所作的扩充给出 EHA 的语义规则.对于迁移 t,令 $Src(t)$和 $Tar(t)$分别表示 t 的源状态和目标状态,$Sr(t)$为 t 的受限源状态集合,$Td(t)$为确定目标状态集合.对于 $A\in F$,A 下面的自动机、状态和迁移分别定义为 $SE(A)=\{A\}\cup(\cup_{A'\in(\cup_{s\in\sigma_A}\rho(s))}SE(A'))$,$S(A)=\cup_{A'\in SE(A)}\sigma_{A'}$,$T(A)=\cup_{A'\in SE(A)}\delta_{A'}$.

定义 5(状态优先顺序). 对于 $s,s'\in S(A_0)$,$s\prec s'$当且仅当 $s'\in S(\rho(s))$.用$\preccurlyeq$表示$\prec$的自反闭包.

迁移的优先级通过一个函数π和一个基于状态优先顺序定义的偏序<得到.若 t 的优先级比 t'低,则记为 $\pi t<\pi t'$.当迁移 t 和 t'冲突时,记作 $t\#t'$.

操作 $add(E,e)$表示把 e 加入事件环境 E 后得到的结构;$join(E,E')$表示 E 和 E'的合并;$Sel(E,e,E')$表示 E'是把 e 从 E 中选择出来后的结构,选择策略依赖于特定结构的语义;nil 表示空结构;对于序列 $r\in E^*$,$new(r)$表示包含 r 中元素的结构(同样,$new(r)$中的元素之间关系依赖于特定结构的语义).例如,如果使用集合,则 $add(E,e)=E\cup\{e\}$,$join(E,E')=E\cup E'$等.定义外部环境事件集合 $Env\subseteq X$ 是由 EHA 外部产生且可被接收的所有事件.对于动作序

列$\xi=a_1,a_2,\ldots,a_n$,$next(V,\xi)$表示在变量环境为V的情况下按顺序执行完ξ中的动作以后得到的变量环境.定义6给出了开放系统规则.

定义6(开放系统).

$$\frac{Sel(E,e,E'')\quad H\uparrow\varnothing::(C,\{e\},V)\rightarrow_L(C',E',V')\wedge E'''\backslash E'\subseteq Env}{(C,E,V)\rightarrow_L(C',join(E'',E'''),V')}$$

在上面使用了一个辅助关系$A\uparrow P::(C,E,V)\rightarrow_L(C',E',V')$对层次自动机$A$的迁移进行建模,其中$L$是顺序自动机$A$中被选择激发的迁移集合;$P$是一个迁移集合,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.该关系表示当L中没有比P中优先级更低的迁移时,A在状况(C,E,V)下可以执行L转移到状况(C',E',V').

令$LE_A(C,E,V)$表示在状况(C,E,V)下顺序自动机A中所有使能迁移的集合,$E_A(C,E,V)=\cup_{A'\in SE(A)}LE_{A'}(C,E,V)$.对于状态$s$和集合$K\subseteq S(\rho(s))$,若满足对于任意$s''\in K$有$s\preccurlyeq s''$,$K$的闭包$cl(s,K)$定义为集合$\{s'|\exists s''\in K,s\preccurlyeq s'\preccurlyeq s''\}$.令$\xi(t)$为激发并执行迁移$t$的过程中所依次执行的动作序列,$\xi_1+\xi_2$为两个序列的顺序连接.$\rightarrow_L$的演绎系统的3条规则如下:

(1) 前进规则:

$$\frac{t\in LE_A(C,E,V)\quad \nexists t'\in P\cup E_A(C,E,V),\pi t<\pi t'}{A\uparrow P::(C,E,V)\rightarrow_{\{t\}}(cl(Tar(t),Td(t)),\cup_{a\in\xi(t)}new(a),next(V,\xi(t)))}$$

(2) 组合规则:

$$\frac{\begin{array}{l}\{s\}=C\cap\sigma_A\quad \rho(s)=\{A_1,\ldots,A_n\}\neq\varnothing\\ \wedge_{1\le j\le n}(A_j\uparrow(P\cup LE_A(C,E,V))::(C,E,V)\rightarrow_{L_j}(C_j,E_j,V_j))\\ (\cup_{1\le j\le n}L_j=\varnothing)\Rightarrow(\forall t\in LE_A(C,E,V),\exists t'\in P,\pi t<\pi t')\end{array}}{A\uparrow P::(C,E,V)\rightarrow_{\cup_{1\le j\le n}L_j}(\{s\}\cup(\cup_{1\le j\le n}C_j),join_{1\le j\le n}(E_j),next(V,+_{t\in\cup_{1\le j\le n}L_j}\xi(t)))}$$

(3) 暂停规则:

$$\frac{\{s\}=C\cap\sigma_A\quad \rho(s)=\varnothing\quad \forall t\in LE_A(C,E,V),\exists t'\in P,\pi t<\pi t'}{A\uparrow P::(C,E,V)\rightarrow_{\varnothing}(\{s\},\mathrm{nil},V)}$$

前进规则指出,如果A的某个迁移使能且优先级足够高,那么迁移被激发,并相应地到达一个新状况.组合规则说明自动机A怎样把迁移的执行委派到子自动机,且这些迁移是如何向上传播的.如果在A中没有足够高优先级的迁移,此外,没有可以委派执行迁移的子自动机,那么A不得不"暂停",这由暂停规则来表示.UML语义中选择最大无冲突迁移集L的过程可以描述为:对于所有$L\subseteq T(A)$,存在(C',E',V')使得$A\uparrow P::(C,E,V)\rightarrow_L(C',E',V')$,当且仅当$L$在集合包含下是满足下面性质的一个最大集合:(1) L是无冲突的,即$\forall t,t'\in L,\neg(t\#t')$;(2) L中所有的迁移在当前状况使能,即$L\subseteq E_A(C,E,V)$;(3) 当前状况下,在L以外没有迁移使能且比L中某个迁移的优先级更高,即$\forall t\in L,\nexists t'\in E_A(C,E,V),\pi t<\pi t'$;(4) $\forall t\in L,\nexists t'\in P,\pi t<\pi t'$.

通过在每个状况选择最大无冲突迁移集L,该操作语义满足UML标准[1]中对Statecharts语义的非形式化定义.因此,通过层次自动机可以用结构化方式正确地表示UML Statecharts.