反应式系统面向性质测试的方法框架

李书浩[1], 王 戟[1], 董 威[1,2], 齐治昌[1]
(1. 国防科技大学计算机学院, 湖南长沙 410073; 2. 武汉大学软件工程国家重点实验室, 湖北武汉 430072)

摘 要: 提出了一种反应式系统选择性测试方法. 该方法根据描述待测系统的 UML Statecharts 模型和描述系统功能属性的时序逻辑公式生成有针对性的测试序列. 据此实现了一个面向性质的测试工具. 实验表明, 该方法可以将测试资源集中于用户关注的系统行为. 经扩展后, 该方法可用于实时系统.

关键词: 软件测试; 反应式系统; UML Statecharts; 时序逻辑
中图分类号: TP311 文献标识码: A 文章编号: 0372-2112 (2004) 12A-2222-04

A Framework of Property-Oriented Testing of Reactive Systems

LI Shu-hao[1], WANG Ji[1], DONG Wei[1,2], QI Zhi-chang[1]
(1. *School of Computing, National University of Defense Technology, Changsha, Hunan* 410073, *China*;
2. *State Key Laboratary for Sofware Engineering, Wuhan University, Wuhan, Hubei* 430072, *China*)

Abstract: A selective testing method for reactive systems is presented. UML Statecharts is employed to model the system under test (SUT) and temporal logic is used to specify the functional property. Then targeted test sequences are derived from the model according to the given property. This method has been implemented in a property-oriented testing tool. Experimental results show that testing efforts can be focused on behaviors of interest of the SUT. With the appropriate extensions, this method can also be applied to real-time systems.

Key words: software testing; reactive system; unified modeling language (UML) statecharts; temporal logic

1 引言

反应式系统在任何时刻都要对可能出现的事件作出适当反应. 由于"激励-响应"在反应式系统中占主要地位, 因此这类系统常常包含大量复杂的控制行为. 作为经典 Statecharts 的面向对象变体, UML Statecharts[1] 描述问题直观、有力. 随着基于模型开发方法的提出和推广, 越来越多反应式系统基于 Statecharts 模型进行开发, 因此基于 Statecharts 的测试变得日益重要.

人们通常希望在项目预算允许范围内对待测系统(SUT)进行尽可能全面而深入的测试, 以期发现尽可能多的错误. 对基于 Statecharts 的测试而言, 由于存在层次、并发、广播通讯和数据变量, Statecharts 常常包含无穷多的行为. 多数场合下, 理想情况的"穷尽测试"难以做到, 因此产生测试用例必须遵循一定的覆盖准则. 现有基于 Statecharts 的测试方法[2~4] 都试图在"全面的"覆盖准则下对 SUT 作"全面"测试. 具体而言, 文[2]采用了状态、格局、迁移等控制流覆盖和 all-def、all-use 等数据流覆盖准则; 文[3]采用了迁移、全谓词、迁移对覆盖等准则; 而文[4]的测试生成方法是 Wp 方法, 相当于采用了迁移覆盖、状态覆盖等.

在许多场合, 测试人员可能更关注 SUT 的某些特定功能性质, 希望重点围绕这些性质对 SUT 进行测试; 或者在项目时间、预算紧张情况下, 只允许针对某些关键性质进行深入测试. 在这些场合, 进行面向性质测试, 即把测试资源集中用于 SUT 中与指定性质相关的行为上就显得非常必要. 然而, 已有的面向性质测试方法[5,6] 大多是基于程序的测试. 由于它们主要采用程序分析和插装技术, 因此难以用于基于模型的测试. 此外, 这些方法在性质描述上也受到诸多限制. 虽然文[7]的方法可以对系统规约进行测试, 但它所采用的是较低层次系统规约, 不适合描述较大规模的、有并发行为的、带数据变量的反应式系统.

本文提出了一种用于反应式系统的选择性测试方法. 令系统 Statecharts 模型为 M, 待测时序属性为 P, 假设 M 满足 P. 我们只为模型 M 中使性质 P 的前件为真的那些行为生成测试用例, 从性质 P 构造 test oracle, 将测试用例精化后作用于 SUT, 以判断 SUT 是否满足性质 P (图 1). 一般来说, 使性质 P 前件为真的模型行为(待测行为)远远少于模型 M 的全部行为, 因此为了让功能测试达到同样"深度"(即测试序列中回

收稿日期: 2004-09-02; 修回日期: 2004-11-22
基金项目: 国家自然科学基金(No. 60233020, No. 90104007, No. 60303013); 国家 863 项目(No. 2001AA113202, No. 2001AA113190); 武汉大学软件工程国家重点实验室开放基金(No. SKLSE03-08); 国防科技大学计算机学院基金"基于构件的软件开发与测试"

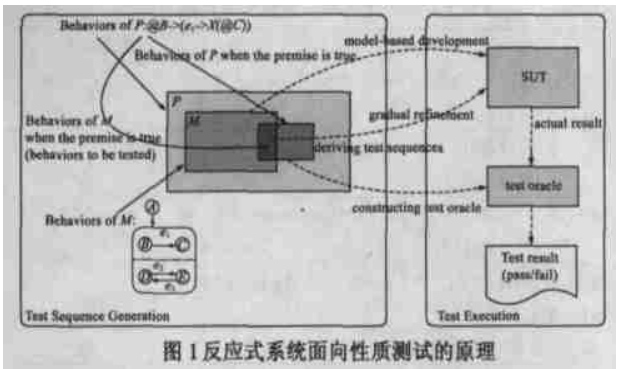

图 1 反应式系统面向性质测试的原理

路的允许出现次数),面向性质测试比非面向性质测试需要更少的开销;而在同等开销下,面向性质测试比非面向性质测试进行得更加深入.

2 反应式系统建模与待测性质描述

图 2 是 UML Statecharts(以下简称 Statecharts)的一个例子,它简要描述了航天器推进子系统(SPS)在变轨前后的加速行为.为了简化问题,我们对 Statecharts 作适当限定:暂不考虑各种伪状态;假定事件不带参数;不考虑迁移优先级;不考虑层间迁移和复合迁移.我们还假设 Statecharts 中所有环境变量都是离散的,其取值集合是有穷的.我们用异步模型[2,8]定义 Statecharts 的语义.Statecharts 的执行以 step 为语义单位.一个 step 是被一个外部输入事件或者一个内部(局部)事件所激发的最大无冲突使能迁移集.我们用"格局"表示 Statecharts 同时所处的状态的集合.从初始格局出发的一个 step 序列称为一个运行(run).一个 statechart 所有可能的运行构成了该模型的行为.

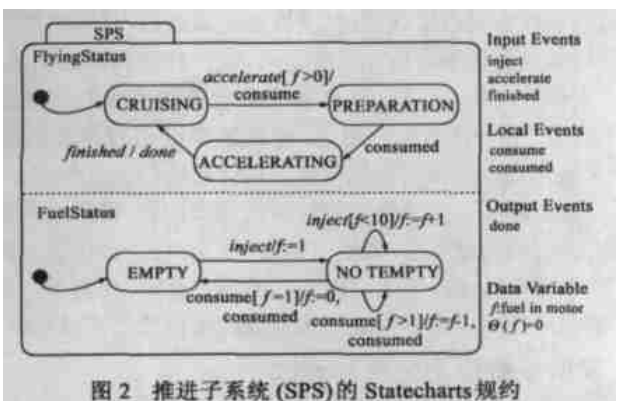

图 2 推进子系统(SPS)的 Statecharts 规约

尽管 Statecharts 描述能力很强,但由于它具有层次、并发、广播通讯机制,因此直接从 Statecharts 生成测试用例的复杂性很高,当系统中含有数据变量时尤其如此.由于扩展有限状态机(EFSM)能对反应式系统的控制结构和数据结构建模,而且其形式比较简洁,因此本文用行为等价的 EFSM 表示 Statecharts,然后在 EFSM 上进行测试生成.

本文采用如下方法将 Statecharts 转换成对应的 EFSM:(1)将 Statecharts 格局和上一 step 产生的内部事件作为 EFSM 的状态;(2)将 Statecharts 的可能 step 作为 EFSM 的迁移.称这样得到的 EFSM 是该 Statecharts 的一个范式规约.图 3 是推进子系统对应的 EFSM.经过 EFSM 变量取值标注和 EFSM 修剪后,EFSM 中不可达迁移和不可达状态(图 3 中虚线表示)被剔除.这样,修剪后的 EFSM 包含与原 Statecharts 完全相同的行为[9].如果我们通过测试能推断出 EFSM 满足某给定逻辑性质,那么就说明 EFSM 的任何一个运行都满足该性质,因此相应 Statecharts 也满足该性质.

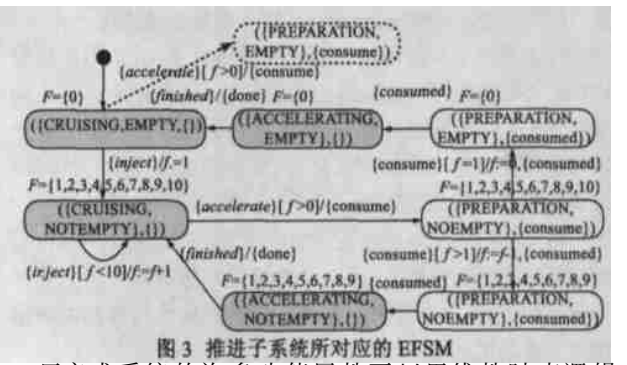

图 3 推进子系统所对应的 EFSM

反应式系统的许多功能属性可以用线性时序逻辑(LTL)[10]公式描述.本文允许 LTL 包含三种原子命题:(1)状态命题"$@s_i$",其中 s_i 是 Statecharts 中的状态名,当 Statecharts 当前状况(我们用"状况"表示 Statecharts 的〈格局,变量解释,上一 *step* 产生的内部事件〉信息)的格局包含 s_i 时命题为真;(2)关系命题"$v_i\ rop\ c_i$",其中 v_i 是环境变量,*rop* 是关系运算符(例如=,≥,< 等),c_i 是常量,当 Statecharts 当前状况的变量解释使得 v_i 和常量 c_i 满足 *rop* 时命题为真;(3)事件命题"e_i",其中 e_i 是外部输入事件或输出事件,当 e_i 是外部输入事件或 Statecharts 当前状况的输出事件时命题为真.

LTL 公式归纳定义如下:

$$\varphi ::= p \mid \neg\varphi \mid \varphi_1 \wedge \varphi_2 \mid \varphi_1 \vee \varphi_2 \mid X\varphi \mid F\varphi \mid G\varphi \mid \varphi_1\, U\varphi_2 \mid \varphi_1 R\varphi_2$$

其中 p 是原子命题,$\boldsymbol{X}$(neXt),$\boldsymbol{F}$(Future),$\boldsymbol{G}$(Globally),$\boldsymbol{U}$(Until),$\boldsymbol{R}$(Release)是时态算子,其具体语义见[10].

例如,$\boldsymbol{F}$(@NOTEMPTY∧($f=2$)∧*done*),$\boldsymbol{G}$(*inject* $\vec{}$ $\boldsymbol{X}$($f\geqslant 1$)),$\boldsymbol{G}$(($f=1$) $\vec{}$ (*accelerate* $\vec{}$ X(@EMPTY)))都是 LTL 公式.其含义比较直观.例如,上述第 3 条公式表示:如果 Statecharts 当前稳定状况(如果一个状况对应的局部事件集为空,那么就说该状况是稳定的)下环境变量 f 值为 1,那么激励 *accelerate* 到达后,下一稳定状况 Statecharts 将处于 EMPTY 状态;这一命题总成立.

3 测试生成与测试选择

为了测试 EFSM 是否满足给定的 LTL 性质,必须列出使性质前件为真的所有(可能无穷多、无穷长)EFSM 路径.我们用正规表达式(或其集合)表示这些路径.

对于状态命题 $@s_i$,我们寻找从 EFSM 初始状态到那些其格局包含 s_i 的所有 EFSM 稳定状态的所有路径.我们将 EFSM 视为一个 FSM,将求得的 EFSM 稳态视为 FSM 的接收状态,上述问题转化为寻找 FSM 所接收的语言的问题.给定一个 FSM,构造与该 FSM 所接收语言对应的正规表达式(或其集合)的算法是已知的[11].

对于关系命题 $v_i\ rop\ c_i$,先寻找 EFSM 中那些变量取值集合能够满足此命题的稳态,然后用上面的算法求出其路径表达式.如果路径表达式中的某个迁移执行一次后会修改变量

v_i 值,那么称该迁移为变量 v_i 的影响迁移. 为关系命题构造路径表达式时,我们对路径中变量 v_i 的影响迁移的出现次数进行规划,使得最后的 v_i 值与 c_i 满足 rop 关系.

对于输出事件命题,我们先找出产生该输出事件的迁移(可能有多个),然后找出从迁移源状态向后算起的第一个稳态,接着生成从 EFSM 初态到这些稳态的路径表达式.

以上是原子命题的路径表达式构造算法. 对于用户按照 LTL 定义构造出的一般形式的 LTL 公式,我们先按照公式构成规则识别出公式的各部分,然后为作为前件的各子公式分别构造路径表达式或表达式集合,最后合成出整个公式的路径表达式或表达式集合.

由于传统的控制流覆盖和数据流覆盖等准则是用来度量"全面"测试的充分性的,而不是度量用户感兴趣行为的测试充分性的,因此本文提出一组用于度量面向性质测试的测试深入程度的覆盖准则.

EFSM 路径表达式中可能存在回路. 由于一条回路可能嵌入到另一回路中,所以我们只考虑不出现重复状态的回路,即基本回路. 一个基本回路的执行如果改变某个与指定性质相关的变量的值,那么称该回路是基本影响回路. 沿基本(影响)回路运行次数越多,表明 SUT 的功能测试进行得越深入. 我们定义如下覆盖准则:

- 基本路径覆盖 如果在得到的路径表达式中,每条基本路径都在某个测试序列中出现,则称该测试集满足基本路径覆盖.
- ***k***-度基本影响回路覆盖 如果在得到的路径表达式中,每个基本影响回路都分别在某个测试序列中出现 i 次($i = 0, 1, 2, \cdots, k$),则称该测试集满足 k-度基本影响回路覆盖.
- ***k***-度基本回路覆盖 如果在得到的路径表达式中,每个基本回路都分别在某个测试序列中出现 i 次($i = 0, 1, 2, \cdots, k$),则称该测试集满足 k-度基本回路覆盖.

上述三条准则是从系统功能测试深度的角度定义的. 显然,k-度基本影响回路覆盖包含基本路径覆盖,而 k-度基本回路覆盖包含 k-度基本影响回路覆盖. 测试人员可以根据项目预算和测试深度要求来选择适当的 k 值.

4 实验结果与讨论

4.1 实验结果

我们实现了一个反应式系统面向性质的测试工具. 工具以 Statecharts 模型、时序逻辑性质、测试覆盖准则为输入. 我们在 SPS 案例上进行了实验. 选取"2-度基本影响回路覆盖"准则,针对若干性质进行测试,所需的测试集大小如表 1 所示. 不难看出,待测性质越宽松,所需测试集越大. 极端情况下,如果不采用面向性质的测试方法(即待测性质为"None"),那么测试集大小为 40. 由此可见,在相同测试深度下,面向性质测试大大降低了所需的测试序列数目.

4.2 讨论

(1) 方法可用性

为了能用于更大规模系统,本文方法需要辅之以各种化简与优化技术. 在状态命题路径表达式构造过程中,可以隐藏 EFSM 中无关状态和迁移,这样大大减小了构造路径表达式的复杂性. 此外,可以事先对 Statecharts 进行切片[12].

表 1 面向性质测试所需的测试集大小

待测性质	测试集大小
$F((@\text{NOTEMPTY}) \wedge (f = 1) \rightarrow (accelerate \rightarrow X(@\text{ACCELERATING})))$	1
$G((@\text{NOTEMPTY}) \wedge (f = 1) \rightarrow (accelerate \rightarrow X(@\text{ACCELERATING})))$	4
$G((@\text{NOTEMPTY}) \wedge (f \leqslant 2) \rightarrow (accelerate \rightarrow X(@\text{ACCELERATING})))$	7
$G((@\text{NOTEMPTY}) \rightarrow (accelerate \rightarrow X(@\text{ACCELERATING})))$	8
None(即不针对任何性质进行测试)	40

(2) 方法适用性

本文方法可以从系统模型、模型语义、待测性质等方面予以替换或扩充. 除了 UML Statecharts 之外,本方法修改后也能用于经典 Statecharts 和其它许多基于 EFSM 的模型. 所用语义也可以是较宽松的同步模型语义[8]. 用于描述待测性质的逻辑也可以是诸如 CTL 的其它时序逻辑.

(3) 与模型检验的关系

为了保证"模型 M 满足性质 P"这一假设成立,需要使用模型检验技术. 具体而言,在进行面向性质测试之前,先调用模型检验工具[13]对指定性质进行验证,看该性质是否满足. 若性质满足,再进行面向性质测试;否则,进行诊断:到底是模型 M 中存在错误,还是指定性质 P 错误?并作相应修正.

当性质 P 是平凡性质(即前件永假)时,无法为 P 的前件生成相应测试序列. 为了防止这种情况发生,对于指定性质的前件,我们先调用模型检验工具进行检验. 一旦发现前件永假,那么立即认为该性质是平凡性质,不再进行进一步处理. 再对整条性质进行检验,看整条性质是否永假. 只有当性质的前件可满足且整条性质亦可满足时,才进行面向性质测试.

近年来,利用模型检验技术从系统规约生成测试用例的工作较多[14,15]. 其基本思想是:将覆盖准则用时序逻辑公式(或其集合)表示,利用模型检验工具针对这些公式生成 counterexample/witness,然后将它们转换成测试用例. 所采用的覆盖准则包括:控制流覆盖,数据流覆盖,变异覆盖,目标谓词覆盖等. 这些都不是面向特定功能属性的覆盖准则. 作为下一步工作,我们可以利用模型检验技术生成面向性质的测试用例.

5 实时系统面向性质的测试

UML Statecharts 经实时扩充后可以描述含有非平凡时间约束的实时系统. 我们的扩充方式是让迁移具有"[$lower$, $upper$] $e_1 \vee e_2 \vee \cdots \vee e_n$ [$guard$]/ $assignments$, $actions$"的形式,其中[$lower$, $upper$]是外部事件发生的时间区间. 待测的实时性质可以用实时逻辑(例如命题线性时间逻辑 PLTL)描述,例如 $\boldsymbol{G}((@\text{NOTEMPTY} \wedge (m = 1)) \rightarrow (coffee \rightarrow ([3, 5]\ accept \rightarrow \boldsymbol{F}_{[1,2]}(@\text{IDLE}))))$. 扩充后的模型仍然采用异步模型语义. 根据这种语义,将扩充后的 Statecharts 模型用行为等价但又较易测试的 EFSM 表示.

给定模型和性质后,就可以生成针对性测试序列. 实时系

统的测试选择包括第一阶段的测试序列生成(得到"抽象测试集")和第二阶段的测试选择,即为测试序列中的时间区间确定时间延迟的具体值(得到"实时测试集").

表 2　实时系统面向性质测试所需的测试集大小

待测性质	抽象测试集	实时测试集
$\boldsymbol{F}$(@NOTEMPTY∧($m=1$)$\rightarrow$($coffee\rightarrow$([3, 5] $accept\rightarrow F_{[1,2]}$(@IDLE))))	1	3
$\boldsymbol{G}$(@NOTEMPTY∧($m=1$)$\rightarrow$($coffee\rightarrow$([3, 5] $accept\rightarrow F_{[1,2]}$(@IDLE))))	4	168
$\boldsymbol{G}$(@NOTEMPTY∧($m\leqslant 2$)$\rightarrow$($coffee\rightarrow$([3, 5] $accept\rightarrow F_{[1,2]}$(@IDLE))))	7	201
$\boldsymbol{G}$(@NOTEMPTY∧($coffee\rightarrow$([3, 5] $accept\rightarrow F_{[1,2]}$(@IDLE))))	8	204
(None)	40	935

我们以实时扩充后的自助咖啡机(CVM)为例进行了实验.在第一阶段,选取"2-度基本影响回路覆盖";在第二阶段,选取粒度为 1 的"时间网格覆盖".结果如表 2 所示.不难看出,对于实时系统,待测性质越宽松,所需测试集越大.

6　结论

本文提出了一种用于反应式系统的选择性测试方法,该方法针对用户给定的时序逻辑公式,从反应式系统模型生成测试序列.当用户仅关注系统的某些特定性质,或者在项目时间、预算紧张情况下只能围绕关键性质进行深入测试时,本文方法能很好地满足其需求.

面向性质测试的进一步研究内容包括:(1)参数事件系统的面向性质测试;(2)混成系统的面向性质测试,允许系统中含有连续变量;(3)多对象系统的面向性质测试,这对于大规模系统非常重要;(4)进一步的经验研究与方法性能评价.

参考文献:

[1] OMG. Unified Modeling Language: Superstructure[S]. version 2.0. Object Management Group, 2003.

[2] H S Hong, Y G Kim, S D Cha, D H Bae, H Ural. A test sequence selection method for statecharts[J]. Journal of Software Testing, Verification, and Reliability, 2000, 10(4): 203-227.

[3] J Offutt, A Abdurazki. Generating Tests from UML Specifications[A]. Proc. 2nd Intl. Conf. on the Unified Modeling Language[C]. Fort Collins, Colorado, USA, 1999. 416-429.

[4] K Bogdanov, M Holcombe. Testing from statecharts using the wp-method[A]. Proc. 2nd Int. Workshop on Formal Approaches to Testing of Software[C]. Brno, Czech, 2002. 19-33.

[5] G Fink, M Bishop. Property based testing: A new approach to testing for assurance[J]. ACM SIGSOFT Software Engineering Notes, 1997, 22(4): 74-80.

[6] B Marre, A Arnould. Test sequence generation from LUSTRE descriptions: GATeL[A]. Proc. 15th IEEE Int Conf on Automated Software Engineering[C]. Grenoble, France, 2000. 229-237.

[7] J C Fernandez, L Mounier, C Pachon. Property oriented test case generation[A]. Proc. 3rd Int Workshop on Formal Approaches to Software Testing[C]. Montreal, Canada, 2003. 147-163.

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

[9] S Li, J Wang, Z C Qi. Property-oriented test generation from UML statecharts[A]. Proc. 19th IEEE Int Conf on Automated Software Engineering[C]. Linz, Austria, 2004. 122-131.

[10] Z Manna, A Pnueli. The Temporal Logic of Reactive and Concurrent Systems: Specification[M]. Springer-Verlag, New York, 1991.

[11] H R Lewis, C H Papadimitriou. Elements of the Theory of Computation[M]. 2nd ed. Prentice Hall, Upper Saddle River, NJ, 1998.

[12] J Wang, W Dong, Z C Qi. Slicing hierarchical automata for model checking UML statecharts[A]. Proc. 4th Int Conf on Formal Engineering Methods[C]. Shanghai, China, 2002. 435-446.

[13] W Dong, J Wang, X Qi, Z C Qi. Model checking UML statecharts[A]. Proc. 8th Asiarr-Pacific Software Engineering Conference[C]. Macau, China, 2001. 363-370.

[14] H S Hong, S D Cha, I Lee, O Sokolsky, H Ural. Data flow testing as model checking[A]. Proc. 25th Int Conf on Software Engineering[C]. Portland, Oregon, USA, May 2003. 232-242.

[15] D Beyer, A J Chlipala, T A Henzinger, R Jhala, R Majumdar. Generating tests from counterexamples[A]. Proc. 26th IEEE Int Conf on Software Engineering[C]. Edinburgh, UK, 2004. 326-335.

作者简介:

李书浩　男, 1976 年生于湖北汉川, 博士生, 主要从事软件质量保障与测试、软件开发方法学等方面的研究. E-mail: shuhao_li@yahoo.com.

王　戟　男, 1969 年生于上海, 博士, 教授, 博士生导师, 主要从事高可信软件技术、软件工程、语义 Web 等方面的研究.

董　威　男, 1976 年生于陕西咸阳, 博士, 讲师, 主要从事高可信软件技术、软件工程等方面的研究.

齐治昌　男, 1942 年生于辽宁锦州, 教授, 博士生导师, 主要从事软件工程、计算机教育等方面的研究.