应用 UML 顺序图的联锁测试用例生成方法*

高雪娟, 武晓春
(兰州交通大学 自动化与电气工程学院, 兰州 730070)

摘 要: 计算机联锁软件是安全完善度等级最高的软件,因此,在投入使用前对联锁软件的测试是必不可少的。为生成联锁软件测试用例,结合包含有充分测试信息的 UML 模型,提出一种以 UML 顺序图为主要模型的软件测试用例生成方法。通过提出有向图 SG 生成算法,将顺序图转换为 SG;然后基于特定的覆盖准则和采用深度优先搜索算法遍历 SG,得到场景的输入、预期输出、约束条件以及场景环境,生成测试用例;最后,以进路建立过程为例验证了该方法的可行性和正确性。

关键词: UML 顺序图; 软件测试; 测试用例生成; 联锁
中图分类号: TP311.5 **文献标志码**: A **文章编号**: 1001-3695(2013)09-2740-04
doi:10.3969/j.issn.1001-3695.2013.09.045

UML sequence based interlocking test case generation

GAO Xue-juan, WU Xiao-chun
(*School of Automation & Electrical Engineering*, *Lanzhou Jiaotong University*, *Lanzhou* 730070, *China*)

Abstract: Computer interlocking software is the highest safety integrity level software, so it is essential to test it before putting it into use. In order to get interlocking software test cases, combined with UML models with sufficient testing information, this paper proposed a method of test case generation based on UML sequence diagram. By proposing generating algorithm of directed graph SG, transferred sequence diagram into SG and traversed SG based on specific coverage criteria and by using depth-first algorithm. It generated the input, respected output, constraint and scenario context and these four constituted the test case. Finally, route setting example proves the feasibility and correctness of this method.

Key words: UML sequence diagram; software test; test case generation; interlocking

0 引言

计算机联锁软件是实时控制铁路车站信号设备和指挥行车作业的安全关键软件[1],要求具有很高的安全性和可靠性,满足铁路信号系统的故障—安全原则,因此,在投入使用前对计算机联锁软件进行测试是必不可少的重要环节。联锁软件测试的实质是针对所设计的一组测试用例执行联锁程序,以发现联锁程序的缺陷。联锁测试用例的设计与生成是软件测试的关键和难点,它决定着软件测试的质量,影响软件测试的效率[2]。

国外对于联锁测试用例的生成主要有:文献[3]从关系代数查询表示的规格说明中生成测试用例;文献[4]采用基于布尔规格说明的测试数据生成方法;此外,基于 Z 语言的测试用例生成也较成熟。但这些测试用例,因为形式化程度太高而通用性差,难以理解和使用。在国内,联锁软件测试用例是针对标准站由富有经验的调试工程师手工设计,对测试人员专业知识和依赖性强,影响测试的充分性和效率。由于安全软件的复杂性和联锁软件测试公正性等因素,第三方测试一般采用黑箱测试。UML(unified modeling language)是面向对象系统分析、设计的标准的建模语言,包括一系列视图和模型,从不同的层次和角度描述了软件系统的结构、行为以及软件的使用,因此基于 UML 模型的软件测试方法一直都受到研究人员的重视,并广泛应用于面向对象软件开发和测试中[5]。目前,基于 UML 顺序图的测试用例生成技术已经得到关注,并且取得了新的发展。例如文献[6]由带 OCL 约束的顺序图生成测试规格说明及事件序列,采用类别划分方法产生测试用例;文献[7]针对顺序图的测试场景进行研究;文献[8]中通过遍历结合类图、状态图生成的测试场景树得到测试用例;文献[9]将 UML 顺序图中的交互行为转换成 FSP 进程代数模型。本文以计算机联锁的进路建立为例,采用 UML 顺序图描述系统场景,以 UML 顺序图为主要模型,提出有向图 SG 生成算法,将顺序图转换为 SG,然后基于特定的覆盖准则和采用深度优先搜索算法遍历 SG,得到场景的输入、预期输出、约束条件以及场景环境,最后生成联锁测试用例。

1 UML 顺序图模型

UML 顺序图是交互图的一种形式,用于描述对象间动态的交互关系,具有良好的软件对象交互的表达性,着重体现消息交互的时间顺序。在图形上顺序图有两个轴,其中水平轴表示参与交互的各个对象,垂直轴表示时间。顺序图中的对象用

收稿日期: 2013-01-21; **修回日期**: 2013-03-12 **基金项目**: 国家自然科学基金资助项目(61164010)
作者简介:高雪娟(1990-),女,甘肃会宁人,硕士研究生,主要研究方向为软件测试、交通信息工程及控制(gaoxuejuan126@163.com);武晓春(1973-),女,副教授,硕士,主要研究方向为交通信息工程及控制.

带有垂直线的矩形框表示。垂直线称为对象的生命线,代表在对象之间的交互作用中该对象的生命周期。对象间的通信消息通过对象生命线之间的箭头表示,箭头上方标注消息名和一些控制信息。箭头分为实线箭头和虚线箭头两种。实线箭头表示一个实际的被触发的方法,虚线箭头仅表示一个方法的返回[10]。

为了能够准确地描述顺序图中的动态交互过程,消除由于自然语言的二义性造成的软件缺陷,本文将 UML 顺序图用一个四元组 $SD = \langle O, M, L, F \rangle$ 来表示。其中:

$O = \{O_1, O_2, \cdots, O_m\}$,是顺序图中对象的集合。

$M = \{\text{Message} \mid \text{Message} = \text{guard_condition} * \text{message_name} * \text{parameter}\}$ 是顺序图中消息的集合。顺序图中的每条消息都可以表示为:[监护条件]消息名(参数)。

L 是顺序图中位点的集合。位点是对象的生命线上发送或接收消息的点,每个位点都形如 $\langle O_i, l_i \rangle$,其中 $O_i \in O$,l_i 为对象 O_i 的位点。

$F: M \times \{s, r\} \rightarrow L$ 是顺序图中从消息到位点的一个函数关系。s 表示消息的发送,r 表示消息的接收。图 1 为一个简单的顺序图,其形式化定义如图 2 所示。

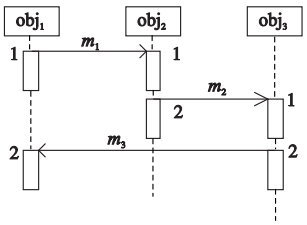

图1　一个简单的顺序图

$SD = \langle \{obj_1, obj_2, obj_3\}, \{m_1, m_2, m_3\}, \{\langle obj_{1,1} \rangle, \langle obj_{2,1} \rangle, \langle obj_{2,2} \rangle, \langle obj_{3,1} \rangle, \langle obj_{3,2} \rangle, \langle obj_{1,2} \rangle\}, \{F(m_1,s) = \langle obj_{1,1} \rangle, F(m_1,r) = \langle obj_{2,1} \rangle, F(m_2,s) = \langle obj_{2,2} \rangle, F(m_2,r) = \langle obj_{3,1} \rangle, F(m_3,s) = \langle obj_{3,2} \rangle, \langle F(m_3,r) = \langle obj_{1,2} \rangle \}\rangle$

图2　顺序图的形式化定义

2　基于 UML 顺序图的测试用例生成方法

基于顺序图的测试用例生成流程如图 3 所示。

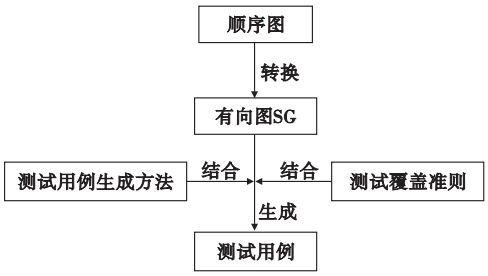

图3　基于顺序图的测试用例生成流程

基于顺序图的测试用例生成步骤如下:

a)添加约束到顺序图。

b)通过遍历顺序图中的所有有效事件序列,得到顺序图的所有场景,生成对应的有向图 SG。

c)定义测试覆盖准则和测试用例生成方法,生成测试用例。

2.1　顺序图生成有向图 SG 的方法

定义 1　有向图 $SG = \langle N_{SG}, \Sigma_{SG}, q_{0SG}, F_{SG} \rangle$。其中:

$N_{SG} = \{N_1, N_2, \cdots, N_m\}$ 是消息节点的集合,$N_j = \langle \text{messageName}, \text{fromObject}, \text{toObject}, [c(\,)] \rangle$,$N_j$ 是根据消息 m 建立的节点,fromObject 是消息的发送对象,toObject 是消息的接收对象,$c(\,)$ 是消息的监护条件;Σ_{SG} 是节点之间转换的边的集合;q_{0SG} 是有向图 SG 的起始节点,也是唯一的入口节点;F_{SG} 是有向图 SG 的终点集合。

定义 2　测试场景 TS。顺序图的测试场景 $TS = \{TS_1, TS_2, \cdots, TS_n\}$ 表示顺序图中的场景路径,包括正常场景和异常场景,其中:

TS_i:$\langle$ScnId, StartState, NodeSet, NextState$\rangle$,ScnId 表示测试场景的唯一标志号,StartState 表示 TS_i 执行前系统的初始状态,NodeSet 表示在测试场景 TS_i 中发生的消息节点的集合,NextState 表示执行完 TS_i 后系统所处的状态。

UML 顺序图转换为有向图 SG 的具体步骤如下:

a)初始化:N_{SG} = NULL;

b)结合顺序图中消息发送的顺序及约束条件,生成对应的消息节点 N 及节点之间的转换边 Σ;

c)将入度为 0 的节点的触发信息记为 q_0,将出度为 0 的节点的后置信息记为 F。

一个有向图 SDG 只能对应一个起点,但对于不同的测试场景,可能有一个或多个终点。

2.2　测试覆盖准则

测试覆盖标准是对软件测试充分性的度量,是测试结果有效性和可信性的凭证,任何测试都应满足一定的覆盖率。本文从可能存在的各对象上的操作错误、对象之间的消息交互错误以及场景错误出发,使用节点覆盖、转移边覆盖与全路径覆盖准则,遍历有向图 SG 获得相应的测试用例集。

准则 1　节点覆盖准则。SG 上的所有节点至少要访问过一次。

准则 2　转移边覆盖准则。SG 上的所有转移边至少要执行过一次。

准则 3　逻辑路径覆盖准则。SG 上的所有路径至少要执行过一次。

针对系统设计中隐含的操作错误,为了能够产生满足测试覆盖准则的测试用例,需要遍历顺序图中所对应的 SG,得到从初始节点到各个终止节点的所有路径,访问每条路径,获得相应的测试场景。

2.3　测试用例生成方法

为了生成满足测试用例覆盖准则的测试用例,本文采用深度优先搜索算法遍历有向图 SG,具体算法流程如图 4 所示。

在遍历有向图 SG 时,记录各个节点所包含的约束条件,包括对象的状态信息、分支条件以及场景的输入和最终输出,用逻辑与(&)将各个约束相连。对任意场景,测试用例为前置条件、输入事件、事件约束、期望的输出结果。

3　基于 UML 顺序图的联锁软件测试用例的生成

本文以进路处理过程为例说明基于 UML 顺序图模型的联锁软件测试用例的生成。

3.1　进路处理过程

联锁软件的进路处理过程是完成对已经搜索出来的进路到建立进路的过程,主要分为三个阶段。图 5 为进路处理过程顺序图。

1)选排一致检查及道岔控制命令生成　若当进路中的道岔位置符合进路要求时,执行下一任务;若道岔位置不符合要求,检查道岔所在区段是否空闲,道岔是否锁闭。若区段空闲

且道岔未锁闭,则转换道岔至规定的位置。

2)进路锁闭　检查进路中的所有区段是否空闲,若空闲,检查敌对进路是否建立。若没有建立敌对进路,则锁闭进路。

3)开放信号　若前两步满足,则开放信号,实现进路建立过程[11]。

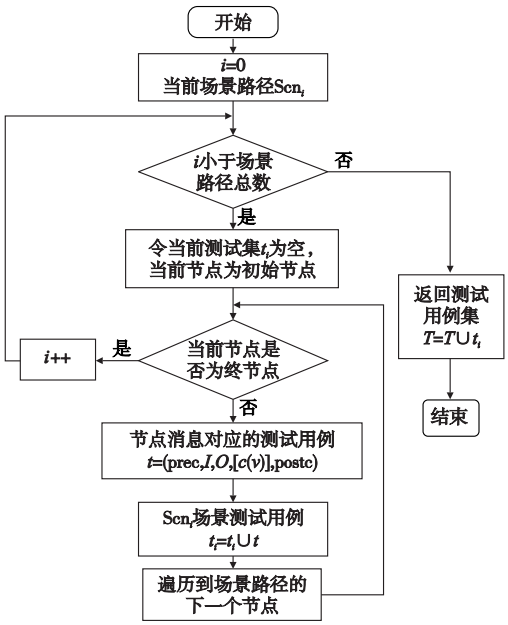

图4　测试用例生成方法流程

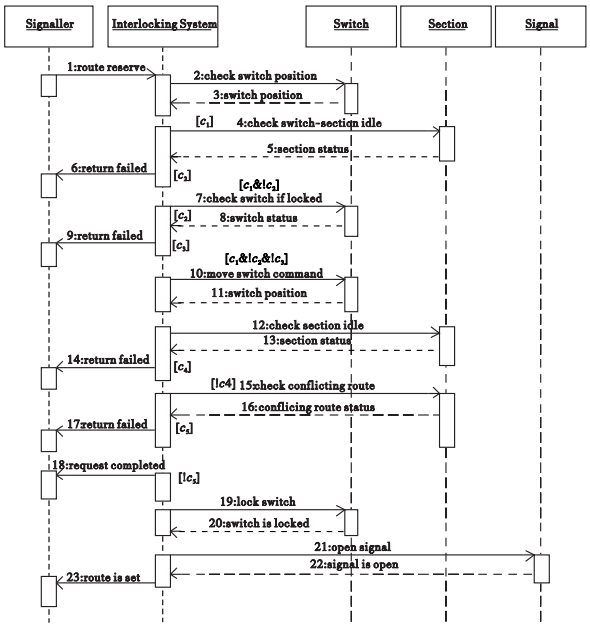

图5　进路处理过程顺序图

3.2　联锁软件测试用例的生成

图 5 中的进路处理过程顺序图对应的测试场景有八个,如图 6 所示,其中,s_i 表示 SG 中的节点,m_i 中的 i 对应顺序图中各消息的序号,各监护条件 c_i($i=1,\cdots,5$)表示含义如表 1 所示。

表 1　各监护条件表示意义

监护条件	表示意义	监护条件	表示意义
c_1	进路所包含的道岔位置不正确	c_4	进路包含区段不空闲
c_2	道岔所在区段有车占用	c_5	敌对进路已建立
c_3	道岔锁闭		

其对应的有向图 SG 如图 7 所示。其中 StateY 表示进路建立失败的终态,StateZ 表示进路建立成功的终态。有向图 SG 包含了生成测试用例所需要的信息,SG 中的每个节点对应顺序图中对象之间的交互,并且储存有监护条件,以及事件发生的前置条件和后置结果。

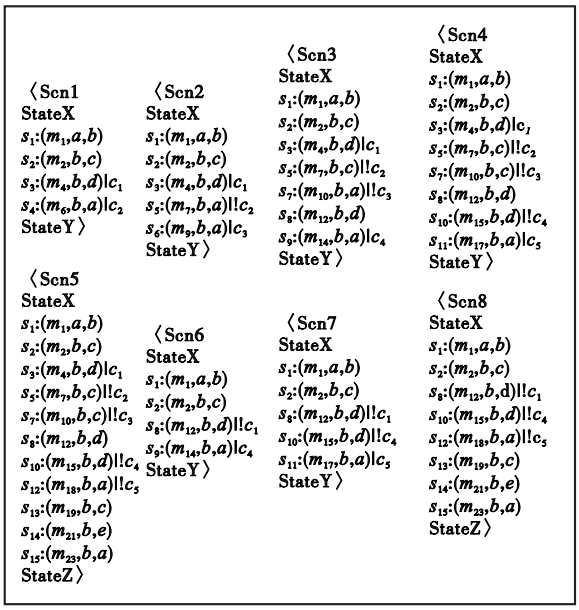

图6　进路处理过程顺序图对应场景图

根据测试用例生成算法遍历 SG 生成的测试用例如下:

测试用例名称:进路建立过程验证

前置条件:已经搜索出进路

测试用例:场景 1

输入:进路建立命令

环境条件:道岔位置不正确 and 道岔所在区段有车占用

方法调用序列:“route reserve”“check switch position”“check switch-section idle”“return failed”

预期结果:信息提示“道岔所在区段有车占用”,进路建立失败

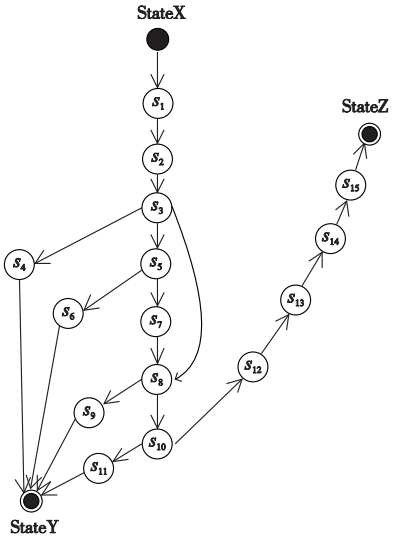

图7　进路处理顺序图对应的SG图

测试用例:场景 2

输入:进路建立命令

环境条件:道岔位置不正确 and 道岔所在区段空闲 and 道岔锁闭

方法调用序列:“route reserve”“check switch position”

“check switch-section idle”“check switch if locked”“return failed”

预期结果:信息提示“进路中有道岔锁闭”,进路建立失败

测试用例:场景 3

输入:进路建立命令

环境条件:道岔位置不正确 and 道岔所在区段空闲 and 道岔没有锁闭 and 进路中区段不空闲

方法调用序列:“route reserve”“check switch position”“check switch-section idle”“check switch if locked”“move switch command”“check section idle”“return failed”

预期结果:信息提示“进路中的区段有车占用”,进路建立失败

测试用例:场景 4

输入:进路建立命令

环境条件:道岔位置不正确 and 道岔所在区段空闲 and 道岔没有锁闭 and 进路中区段空闲 and 敌对进路已建立

方法调用序列:“route reserve”“check switch position”“check switch-section idle”“check switch if locked”“move switch command”“check section idle”“check conflicting route”“return failed”

预期结果:信息提示“敌对进路已建立”,进路建立失败

测试用例:场景 5

输入:进路建立命令

环境条件:道岔位置不正确 and 道岔所在区段空闲 and 道岔没有锁闭 and 进路中区段空闲 and 敌对进路没有建立

方法调用序列:“route reserve”“check switch position”“check switch-section idle”“check switch if locked”“move switch command”“check section idle”“check conflicting route”“request completed”“lock switch”“open signal”“route is set”

预期结果:进路建立成功

测试用例:场景 6

输入:进路建立命令

环境条件:道岔位置正确 and 进路中区段不空闲

方法调用序列:“route reserve”“check switch position”“check section idle”“return failed”

预期结果:信息提示“进路中区段有车占用”,进路建立失败

测试用例:场景 7

输入:进路建立命令

环境条件:道岔位置正确 && 进路中区段空闲 && 敌对进路已建立

方法调用序列:“route reserve”“check switch position”“check section idle”“check conflicting route”“return failed”

预期结果:信息提示“敌对进路已建立”,进路建立失败

测试用例:场景 8

输入:进路建立命令

环境条件:道岔位置正确 and 进路中区段空闲 and 敌对进路没有建立

方法调用序列:“route reserve”“check switch position”“check section idle”“check conflicting route”“request completed”“lock switch”“open signal”“route is set”

预期结果:进路建立成功

4　基于 UML 顺序图生成的联锁软件的测试用例结果分析

结合实例,可总结该基于 UML 测试用例生成方法相对于其他的测试用例生成方法有以下优点:

a)采用 UML 统一建模语言对联锁软件的需求进行定义,不仅使联锁逻辑关系更加清晰易懂,加强开发人员对于联锁系统的理解,有利于计算机联锁软件的开发与设计工作,还提供了证实系统正确性的潜力,使软件开发与测试用例生成同时进行,减少开发周期。

b)系统许多错误与系统的执行流程有关,测试要按照某一个流程执行才能发现这些错误。有向图 SG 覆盖了系统所有可能的执行流程,有利于测试用例覆盖这些错误。

c)在测试用例的生成过程中,将环境条件与输入、方法调用序列和预期输出作为最终的测试用例,最后将系统测试后的实际输出和方法调用序列与预期的输出和方法调用序列进行比较,从整体上验证系统的最终实现是否与设计一致。测试信息可由分析设计模型中直接导出,这也是基于 UML 测试的最大优势。

5　结束语

本文以联锁进路建立过程为例生成测试用例,基于节点、转移边、逻辑路径的覆盖准则满足联锁测试用例的条件需要。但由于只考虑了单个用例的测试用例生成,因此,研究考虑各用例之间的依赖关系,优化生成联锁软件的测试用例是未来的研究重点。

参考文献:

[1] 陈全贝,徐中伟. 联锁软件测试集可视化生成[J]. 武汉大学学报:理学版,2004,50(S1):45-48.

[2] 李娟,程光武,范多旺. 基于成对组合的遗传算法生成联锁测试用例[J]. 铁路计算机应用,2011,20(3):53-58.

[3] TSAI W T,VOLOVIK D,TKEEFE T F. Automated test case generation for programs specified by relational algebra queries[J]. IEEE Trans on Software Engineering,1990,16(3):316-324.

[4] WEYUKER E,GORADIA T,SINGH A. Automatically generating test case data from boolean specification[J]. IEEE Trans on Software Enginnering,1994,20(4):353-363.

[5] 张琛,段振华. 应用 UML 2.0 模型的测试用例生成方法[J]. 西安交通大学学报,2011,45(8):18-23.

[6] 江泽凡,王林章,李宣东,等. 基于 UML 顺序图的测试方法[J]. 计算机科学,2004,31(7):131-136.

[7] 黄陇,于洪敏,陈致明. 基于 UML 顺序图的测试场景自动生成研究[J]. 计算机应用与软件,2006,23(4):30-31,116.

[8] 李志强,邵培南,朱辉. 基于 UML 顺序图的测试用例生成[J]. 计算机工程,2010,36(22):58-60.

[9] 王曦,徐中伟. 基于场景分析的系统形式化模型生成方法[J]. 计算机科学,2012,39(8):136-163.

[10] 苏荟,张毅坤,姚海波,等. 基于 UML 顺序图的面向对象软件簇集测试[J]. 计算机工程,2005,31(24):78-79,101.

[11] 赵志熙. 计算机联锁系统技术[M]. 北京:中国铁道出版社,1999.