软件学报 ISSN 1000-9825, CODEN RUXUEW
Journal of Software,2015,26(2):269−278 [doi: 10.13328/j.cnki.jos.004780]

E-mail: jos@iscas.ac.cn
http://www.jos.org.cn
Tel: +86-10-62562563

一种面向列车控制系统中安全攸关场景的测试用例自动生成方法*

陈 鑫[1,2], 姜 鹏[1,2], 张一帆[1,2], 黄 超[1,2], 周 岩[1,2]

[1](计算机软件新技术国家重点实验室(南京大学),江苏 南京 210023)

[2](南京大学 计算机科学与技术系,江苏 南京 210023)

通信作者: 陈鑫, E-mail: chenxin@nju.edu.cn

摘 要: 列车控制系统是一种安全攸关系统,为保证其安全性,要求测试过程对安全攸关场景中所有可能的运行进行完全的覆盖.现有的场景建模与测试用例自动生成方法不能完全满足这一技术需求.围绕列车控制系统的安全攸关场景建模以及测试用例自动生成方法展开研究,对 UML 活动图扩充了事件驱动机制和时间特性描述机制,以满足对安全攸关场景建模的需要,提出了简单路径覆盖准则以定义对场景中所有运行的完全覆盖,并针对这一覆盖准则给出了自动生成测试用例的方法.以地铁列车控制系统为研究对象展开实验,表明了该方法的有效性和局限性.

关键词: 安全攸关场景;场景建模;简单路径覆盖;测试用例自动生成

中图法分类号: TP311

中文引用格式: 陈鑫,姜鹏,张一帆,黄超,周岩.一种面向列车控制系统中安全攸关场景的测试用例自动生成方法.软件学报, 2015,26(2):269−278. http://www.jos.org.cn/1000-9825/4780.htm

英文引用格式: Chen X, Jiang P, Zhang YF, Huang C, Zhou Y. Method of automatic test case generation for safety-critical scenarios in train control systems. Ruan Jian Xue Bao/Journal of Software, 2015,26(2):269−278 (in Chinese). http://www.jos.org.cn/1000-9825/4780.htm

Method of Automatic Test Case Generation for Safety-Critical Scenarios in Train Control Systems

CHEN Xin[1,2], JIANG Peng[1,2], ZHANG Yi-Fan[1,2], HUANG Chao[1,2], ZHOU Yan[1,2]

[1](State Key Laboratory for Novel Software Technology (Nanjing University), Nanjing 210023, China)

[2](Department of Computer Science and Technology, Nanjing University, Nanjing 210023, China)

Abstract: The train control system is a safety-critical system. To assure its safety, it requires the testing process to cover all possible runs in its safety-critical scenarios. Existing methods of scenario modeling and test case generation cannot completely satisfy the requirement. The paper focuses on the methods of modeling safety-critical scenarios in train control system and the tools for automatically generating test cases for the system. UML activity diagram is extended with event-driven and time characteristic description mechanism to satisfy the requirement of modeling safety-critical scenarios. A simple path coverage criterion is also proposed to define the coverage of all possible runs in a scenario and a method is provided for automatic test case generation. The experiment on ground train control system shows the effectiveness and limitation of the proposed method.

Key words: safety-critical scenarios; scenario modeling; simple path coverage; automatic test case generation

列车控制系统是对列车运行进行实时控制的系统,它负责保障列车安全和快速的运行.一个完整的列车控制系统由车载系统和地面系统两部分组成,其中,车载系统包括车辆信号系统、列车自动防护系统、列车自动驾驶系统等,地面设备包括轨道电路、应答器、列车控制中心和通信网络等.在新一代的列车控制系统——基

* 基金项目: 国家重点基础研究发展计划(973)(2014CB340703); 国家自然科学基金(91318301, 91118002, 61321491, 61402222); 教育部高等学校博士学科点专项科研基金(20110091120058); 江苏省产学研项目(BY2014126-03)

收稿时间: 2014-07-01; 修改时间: 2014-10-31; 定稿时间: 2014-11-26

于通信的列车控制系统(communication based train control system,简称 CBTC)中,车载系统与地面系统之间通过多种无线通信技术进行通信,相互协同保证列车运行的安全.列车控制系统是一种典型的安全攸关系统,如何保证列车控制系统的安全性,是列控系统研发的核心技术问题.

列车控制系统的安全性需要遵循国际标准 IEC61508[1]的规定,这一标准将安全定义为“不存在不可接受的风险”.为满足这一安全性定义,需要对列车控制系统中所有的安全攸关场景进行充分的测试,以确认在所有安全攸关场景中,列车控制系统都能保证列车的运行不存在风险.这里,安全攸关场景是指列车控制系统的错误可能导致重大财产损失和人身伤亡的列车运行场景.在列车控制系统中,典型的安全攸关场景包括列车追踪运行、点式列车运行、临时限速运行、站台与列车屏蔽门联动控制、列车运行等级切换等.

对安全攸关场景进行充分测试,意味着对此场景下列车控制系统所有可能的运行序列进行完全覆盖.为驱动列车控制系统覆盖给定的一个运行,需要产生相应的测试数据序列.与通常的被测系统相比,列车控制系统具有其自身的技术特点:

- 首先,事件驱动是列车控制系统的运行时刻重要的技术特征.在列车控制系统中,各个子系统之间、子系统与物理设备之间通过响应各种类型的事件进行相互协同.
- 其次,作为一个实时系统,系统中各类事件和动作都需要满足相应的时刻约束和延时约束.

这些技术特征,决定了测试输入序列需要以事件作为基本形式;同时,输入事件之间除了需要遵守时序约束之外,还需要满足时刻约束和延时约束.为了满足对安全场景进行充分覆盖的要求,需要定义新的覆盖准则.针对大场景,为了满足充分覆盖准则的要求,需要研究测试数据自动生成方法,以提高测试覆盖度,降低测试开销.已有的场景建模与测试数据自动方法不能完全满足上述技术需求,因此,针对列车控制系统安全攸关软件的建模与测试数据自动生成方法是值得研究的问题.

基于上述分析,本文提出了一种面向列车控制系统安全攸关场景的测试输入生成方法.

- 该方法首先对 UML 活动图进行了扩展,引入对事件和时间进行表示的机制,以对列车控制系统安全攸关场景中事件驱动的运行和各种时刻、延时约束进行描述.
- 接下来,定义简单路径刻画安全攸关场景中每个可能的运行,并使用简单路径覆盖准则定义对安全场景进行完全覆盖的要求.
- 随后的自动测试输入生成方法首先对安全攸关场景模型进行遍历,以找到模型中的所有简单路径;然后针对每条简单路径,沿着其向前推进的轨迹,收集对应的输入事件序列,输出事件序列以及相关的时间约束,最终生成测试用例.

本文第 1 节给出面向列车控制系统安全攸关场景的建模方法.第 2 节定义简单路径覆盖准则以刻画对安全攸关场景的进行全覆盖的需求,并研究面向简单路径覆盖准则的测试输入的自动生成算法.第 3 节进行实例研究.第 4 节比较相关工作,最后是总结和对进一步工作的讨论.

1 安全攸关场景建模

首先给出一个安全攸关场景的例子来说明安全攸关场景建模的技术需求.下面的例子描述了地铁列车停车到站后,车载控制系统与地面控制系统协同,控制列车门与站台屏蔽门(platform screen doors,简称 PSD)的同步开关,以保证乘客安全上下车的场景.

列车到达站台,并在指定的停车窗内静止.所有车门及站台屏蔽门均处于关闭状态.车载子系统解锁列车车门以使其开启,并通过位置报告(position report,简称 PR)报文向 ATP 轨旁控制单元(wayside control unit,简称 WCU_ATP)发送开门授权.车载子系统根据驾驶员手动控制命令或自动车门控制命令,使用站台门报文发送 PSD 开启命令.WCU_ATP 只对与列车车门相对应的 2 号屏蔽门控制器(platform screen door controller,简称 PSDC)发送 PSD 开启命令.其他 PSDC 不会接收到该开门命令.在商定的时刻, PSD 和列车车门同时开启.如果在列车开门后,屏蔽门未能成功打开或超过 10s 未收到 WCU_ATP 返回报文,则会触发警报,需要进行应急处理.当 PSD 开启后,WCU_ATP 通过 MA 报文设定站台屏蔽门行车授权移动区域(PSD run authorization zone,简称 PSD

RAUZ)为限制区域,以禁止列车的移动.当停站时间结束后,车载子系统根据驾驶员手动控制命令或自动车门控制命令,使用站台门报文发送 PSD 关闭命令.WCU_ATP 对 2 号 PSDC 发送 PSD 关闭命令.列车车门与 PSD 一起关闭.PSDC 报告“PSD 关闭且锁闭”的状态.列车车门同样对车载子系统报告其状态信息.当所有车门关闭且 PSD RAUZ 变为允许区域后,车载子系统允许列车启动.当所有 PSD 关闭且锁闭后,WCU_ATP 立即使用移动授权(movement authorization,简称 MA)报文改变 PSD RAUZ 状态为允许状态,以允许列车离站.从车载系统发出指令至 PSD 系统之间的时延最大为 1s,从 PSD 系统接收到指令到门动作的时延小于 0.5s.

在这一场景中,车载子系统、ATP 轨旁控制单元、站台屏蔽门控制系统之间使用消息进行通信.特定种类的消息达到事件,驱动各个子系统执行对应的动作.站台屏蔽门控制系统和站台屏蔽门之间的协同,也是事件驱动机制.站台屏蔽门通过发送开启和关闭事件,向站台屏蔽门控制系统报告自己的状态.同时,屏蔽门控制系统通过发送开启和关闭命令,驱动站台屏蔽门执行开关的动作.

作为一种实时系统,上述场景中同时存在着关于时刻和延时的两种约束.例如,为保证乘客不会误入列车与站台之间的空隙,场景中列车车门和站台屏蔽门需要在商定的时刻同时开启.PSDC 在整个乘客交换过程中需要保证屏蔽门处于开启状态.

活动图是 UML 中进行场景建模的首选工具,它使用活动描写场景中的一个动作或者处理步骤,使用转换连接各个活动,表示活动之间的时序关系,通过转换连接起来的一系列活动构成一个处理过程.采用分叉和汇合来表示并发执行的处理过程,使用判定来表示不同条件会导致不同的后续活动.活动图中还引入了泳道区分场景中的不同对象,以便刻画活动与活动之间的从属关系.

针对列车控制系统中的安全攸关场景,UML 活动图并不能完全满足其建模的技术需求.比如,活动图中上一个动作结束后,可以沿着转换直接启动下一个动作执行,缺乏对事件驱动机制的显示描写机制;同时,标准的 UML 活动图中没有描述时间的机制,也就无法对场景中的时刻约束和时间约束进行描述.

针对上述问题,本文对标准的 UML 活动图进行扩展,使其能够满足为安全攸关场景建模的需要.

- 首先引入了事件驱动机制,允许在转换上添加形如“⟨*E*:事件描述⟩”的描述,表示这一转换是被“⟨·⟩”描述的事件驱动的.对于没有事件描述的转换,则遵从活动图原来的语义,即,上一活动完成后,下一活动自然启动.
- 为了描述时间约束,引入形如“*t*=@(事件)”的记录事件发生的时间 *t*.这样,时刻就可以用对应事件上的“*t*”进行刻画.对于延时,则可以分别记录相关事件时刻 t_1,t_2,用 t_2-t_1 进行描述.

我们对扩展了上述描述机制的活动图给出如下的定义:

定义 1. 一个扩展的活动图 *D* 是一个多元组 $D=(A,Tr,W,T,G,E,F,a_I,a_F,TC)$,其中,

- $A=\{a_1,a_2,\ldots,a_i\}$是活动的一个有穷集合.
- $Tr=\{tr_1,tr_2,\ldots,tr_j\}$是转换的一个有穷集合.
- $S=\{s_1,s_2,\ldots,s_n\}$是一组泳道的集合.
- $T=\{t_1,t_2,\ldots,t_k\}$是时刻的一个有穷集合,通过“*t*=@(事件)”记录的事件发生时刻.
- $G=\{g_1,g_2,\ldots,g_j\}$是卫式条件的一个有穷集合,g_i 是对应于转换 tr_i 的卫式条件(为 true 时可省略).
- $E=\{e_1,e_2,\ldots,e_m\}$是事件的一个有穷集合,且 e_i 位于跨泳道的转换 tr_i 上,表示事件 e_i 驱动转换 tr_i.
- $F\subset(A\times Tr)\cup(Tr\times A)$是一个有穷的流关系.
- $a_I\in A$ 是初始节点;$a_F\in A$ 是终止节点.初始节点对应唯一的转换,即,有且仅有一个转换 *tr* 满足$(a_I,tr)\in F$.初始节点 a_I 之前没有转换,终止节点 a_F 之后没有转换.即,$\forall tr\in Tr^{\bullet},(tr,a_I)\notin F\wedge(a_F,tr)\notin F$.
- $TC=\{tc_1,tc_2,\ldots,tc_l\}$是一个时间约束的有穷集合,其中,时间约束 tc_i 中出现的时刻变量一定包含在 *T* 中.

基于扩展后的活动图,前述列车进站停车后,车门控制场景可用图 1 来表示.

场景中的两个时间约束可用如下方法来表示:

- @(*b* 时刻开门命令到达站台屏蔽门控制器)–@(车载子系统.发出 *b* 时刻开门命令)≤1;
- @(打开屏蔽门)–@(*b* 时刻开门命令到达站台屏蔽门控制器)≤0.5.

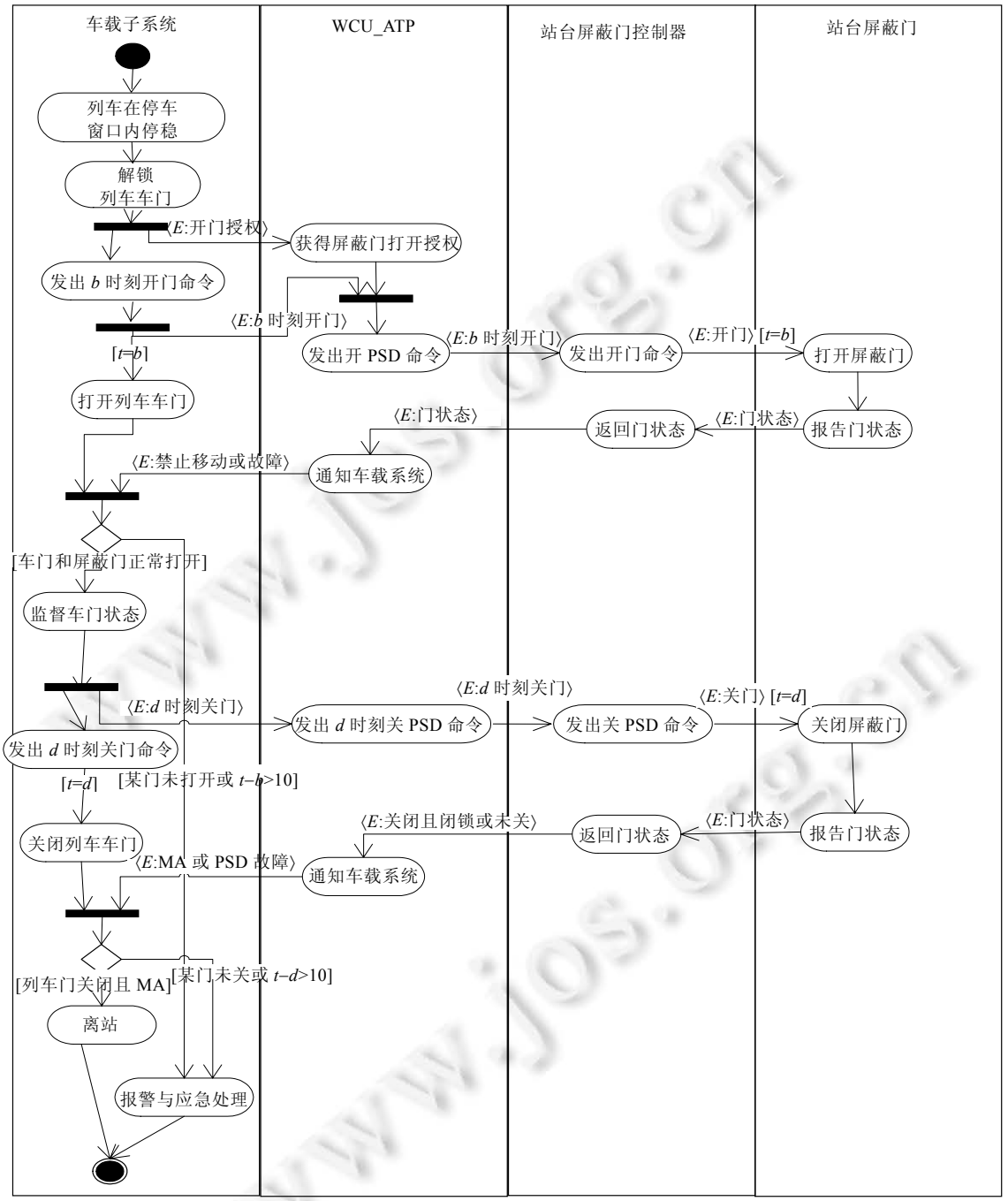


Fig.1 Scenario model of the door control when a train stops

图 1 列车停站车门控制场景模型

2 测试用例自动生成方法

2.1 安全攸关场景的覆盖准则

由于活动图中允许存在循环和并发活动,因此,这样的活动图中可能的执行序列是无限的.测试过程不可能

覆盖无穷多的执行,因此必须仔细考虑如何定义有效的覆盖准则,以反映对安全攸关场景中所有可能执行进行覆盖这一要求.由于循环可被展开成为无限长的序列,为了控制展开的长度,限定场景的一个执行中循环只展开一次.考虑到列车控制系统是一个物理上并发执行的系统,规定在执行的每一步,位于不同并发分支上的转换可以同时向前推进一步,即并发执行.这样的限定可以保证场景中的每个转换都可以按照先后顺序被执行至少一次,同时,由于不需要交错执行来覆盖所有的并发行为,可以大幅减少需要覆盖的路径的数目,控制测试的成本.

根据这样的想法,首先需要定义扩展活动图上的一步并发执行.并发执行的定义需要用到下面的几个辅助符号:给定一个扩展活动图 D,D 上的一个状态 μ 是一个 D 上活动的集合,即,$\mu \in D.A$;引入符号 ${}^\bullet tr$ 和 $tr^\bullet$ 分别表示转换 tr 的前状态集和后状态集,它们分别定义为 ${}^\bullet tr=\{a\in A|(a,t)\in F\}$ 和 $tr^\bullet=\{a\in A|(t,a)\in F\}$;用 $enabled(\mu)$ 表示可以从状态 μ 上发出的转换的集合,显然,$tr\in enabled(\mu)$,如果 ${}^\bullet tr\subseteq\mu$.

定义 2. 给定一个扩展活动图 $D=(A,Tr,S,T,G,E,F,a_I,a_F,TC)$,$\mu$ 是 D 中的一个状态,状态 μ 上的一步并发执行 τ 是一个满足下面条件的转换 $tr_1,tr_2,\ldots,tr_m$ 的集合:

- $\forall i,1\leqslant i\leqslant m\cdot tr_i\in enabled(\mu)$;
- $\forall i,j(1\leqslant i<j\leqslant m)\cdot {}^\bullet tr_i\cap {}^\bullet tr_j=\varnothing$;
- $\forall tr\in(enabled(\mu)-\{tr_1,tr_2,\ldots,tr_m\})$,存在一个 $tr_i(1\leqslant i\leqslant m)$,满足条件 ${}^\bullet tr\cap {}^\bullet tr_i\neq\varnothing$.

从状态 μ 出发,经过一步并发执行,得到了新状态 μ'.一步并发执行的结果可由下式计算得到:

$$\mu'=\bigcup_{i-1}^{m}((\mu-{}^\bullet tr_i)\cup tr_i^\bullet).$$

为了便于使用,后文中用 $fire(u,\tau)$ 表示并发执行的结果 μ'.

安全攸关场景的一次并发执行对应了从初始节点到终止节点的一条路径,路径中的每一步都是满足定义 2 的一步并发执行.

定义 3. 给定一个扩展活动图 $D=(A,Tr,W,T,G,E,F,a_I,a_F,TC)$,$D$ 上的一条路径 ρ 是一个由活动和执行组成的序列:

$$\rho=\mu_0\xrightarrow{\tau_0}\mu_1\xrightarrow{\tau_1}\ldots\xrightarrow{\tau_{n-1}}\mu_n.$$

其中,$\mu_0=\{a_I\}$,$\mu_n=\{a_F\}$,并且 $\forall 1\leqslant i\leqslant n\cdot u_i=fire(u,\tau)$.

如果路径中不存在循环,则称这样的路径为简单路径.从执行的角度来看,即,同一个状态不能在执行中出现两次及以上.根据这个观察,可以如下定义简单路径:

定义 4. 设 ρ 是活动图 $D=(A,Tr,S,T,G,E,F,a_I,a_F,TC)$ 上的一条路径,ρ 是一条简单路径,当且仅当对路径中任意两步执行 τ_i 和 τ_j,其中任意的两个转换 $tr\in\tau_i$ 和 $tr'\in\tau_j$ 都满足条件 ${}^\bullet tr\cap {}^\bullet tr'=\varnothing$.

在图 1 所示的场景中,有:

$\{a_I\}\rightarrow$\{列车在停车窗口内停稳\}$\rightarrow$\{解锁列车门\}$\rightarrow$\{发出 b 时刻开门命令,获得屏蔽门打开授权\}$\rightarrow$
\{发出 b 时刻开门命令,发出开 PSD 命令\}$\rightarrow$\{发出 b 时刻开门命令,发出开门命令\}$\rightarrow$
\{打开列车车门,打开屏蔽门\}$\rightarrow$\{打开列车车门,报告门状态\}$\rightarrow$\{打开列车车门,返回门状态\}$\rightarrow$
\{打开列车车门,通知车载系统\}$\rightarrow$\{监督车门状态\}$\rightarrow$
\{发出 d 时刻关门命令,发出 d 时刻关 PSD 命令\}$\rightarrow$\{发出 d 时刻关门命令,发出关 PSD 命令\}$\rightarrow$
\{关闭列车门,关闭屏蔽门\}$\rightarrow$\{关闭列车门,报告门状态\}$\rightarrow$\{关闭列车门,返回门状态\}$\rightarrow$
\{关闭列车门,通知车载系统\}$\rightarrow$\{离站\}$\rightarrow\{a_F\}$是对应于场景中正常运行的一条简单路径.

从这个例子可以看到,我们定义的并发执行可以精简由于并发行为带来的测试的路径数量.路径的数量主要受被测系统内部判定分支的数量影响.这样,简单路径可以很好地覆盖测试场景中的各种处理逻辑,而对并发行为导致的复杂性进行了很好的控制.

下面使用简单路径覆盖来定义对安全攸关场景的覆盖:

定义 5. 一个扩展活动图 $D=(A,Tr,S,T,G,E,F,a_I,a_F,TC)$ 描述了一个安全攸关场景,称安全攸关场景中所有的执行被覆盖,如果活动图 D 中所有的简单路径都已被覆盖.

2.2 测试用例生成方法

根据上节的定义,对安全攸关场景的完全覆盖问题转化成为对扩展活动图上所有简单路径的覆盖问题.为生成对应的测试用例,首先需要找出安全有关场景中所有可能的执行,即,扩展活动图上所有的简单路径.

算法 1. 简单路径集合生成算法.

输入:扩展的活动图 *D*.

输出:简单路径集合 *sPaths*.

1. *path*:=⟨{a_I}⟩;
2. *sPaths*:=∅;
3. DO
4. *node*:=*path* 的最后一个节点
5. IF 从 *node* 出发不存在一个并发执行到后继 $node_1$
6. THEN 移除 *path* 中最后一个节点和最后一个并发执行;
7. ELSE BEGIN
8. *node*:=从 *node* 出发经过一步并发执行 τ 得到的后继节点;
9. IF *node*=a_F THEN
10. BEGIN
11. 向 *path* 的尾部添加并发执行 τ 和节点 *node*;
12. *sPaths*:=*sPaths*∪*path*;
13. END
14. IF 从 *node* 出发可以将 *path* 扩展成为一条简单路径 THEN
15. 向 *path* 的尾部添加并发执行 τ 和节点 *node*;
16. END
17. WHILE *path*:=⟨·⟩;
18. RETURN *sPaths*

算法的基本思想是,基于深度优先搜索对扩展活动图进行遍历.遍历从起始节点开始,使用并发执行向前推进.变量 *path* 存储了从起始节点推进到当前节点经过的所有节点.如果已经推进到终止节点,则变量 *path* 加上终止节点就是一条简单路径.这时,将找到的简单路径添加到简单路径集合 *sPaths* 中去.如果从推进到的节点出发,可以将当前路径片断 *path* 继续推进到简单路径,则将当前推进到的节点加入到 *path* 的尾部,以便在下次循环中继续向前推进;如果无法继续推进,则从 *path* 尾部移除当前节点,进行回退,在下次循环中回到上一个节点进行推进.

获得简单路径集合之后,针对集合中每一条简单路径,采用如下的方法生成对应的测试用例:首先,明确该场景中被测系统(system under test,简称 SUT)的边界,被测的某一子系统或多个系统应处于边界之内,其他子系统作为外部环境位于边界之外;接着,沿着简单路径从起始节点向终止节点并发执行.沿路收集外部环境向被测系统发送的事件作为测试输入.收集被测系统向环境的输出作为观察到的结果序列.与上述收集到的事件有关的时间约束也一并收集起来,作为输入和输出需要满足的约束.

这里生成的测试用例中应当包括 3 个部分:SUT 边界之外在特定时间、满足特定约束情况下传递给 SUT 的控制信息,在特定时间点对 SUT 的观察结果,在时间上所需满足的特定的约束.给出本文对测试用例的定义如下:

定义 6. 与给定的扩展活动图 $D=(A,Tr,S,T,G,E,F,a_I,a_F,TC)$ 中一条路径 ρ 相关的一个测试用例 c 是一个三元组 $c=(I,CN,O)$,其中,

- I 是形如 $(s_m,e_n,t_1)\rightarrow(s_{m+1},e_{n+1},t_2)\rightarrow\ldots\rightarrow(s_{m+k},e_{n+k},t_k)$ 的一个输入序列,(s_i,e_j,t_k) 表示在 t_k 时刻,泳道 s_i 向被测系统发送事件 e_j;

- $CN=\{tc_1,tc_2,\ldots,tc_o\}$是时间约束的集合,其中,$tc_i$为时间约束;
- O是形如$(r_m,e_n,t_1)\rightarrow(r_{m+1},e_{n+1},t_2)\rightarrow\ldots\rightarrow(r_{m+k},e_{n+k},t_k)$的一个观测结果序列,$(r_i,e_j,t_k)$表示在$t_k$时刻可以观察到泳道$r_i$从被测系统接收事件$e_j$.

在上述定义中,观测结果序列表示生成的测试用例在给定的满足约束的输入之下得到的预期结果,集合CN仅包含与输入消息序列和观察结果序列有关的事件序满足的时间约束条件.

算法 2 以简单路径和被测系统所占用的泳道为输入,输出该路径对应的测试用例.

算法 2. 测试用例集生成算法.

输入:扩展活动图$D=(A,Tr,S,T,G,E,F,a_I,a_F,TC)$上的一条简单路径$\rho$,被测系统占用的泳道集合$SUT$.

输出:ρ对应的一个测试用例c.

1. $step:=\tau_0$;
2. DO
3. DO
4. $tr:=step$ 中第 1 个未处理的 tr;
5. IF tr 是从不属于 SUT 的泳道向属于 SUT 的泳道的发送事件 THEN
6. BEGIN
7. 向 $c.I$ 中添加当前的事件;
8. 向 $c.CN$ 中添加与这一事件有关的约束;
9. END
10. IF tr 是从属于 SUT 的泳道向不属于 SUT 的泳道的发送事件 THEN
11. BEGIN
12. 向 $c.O$ 中添加当前的事件;
13. 向 $c.CN$ 中添加与这一事件有关的约束;
14. END
15. WHILE $step$ 没有未处理的 tr;
16. $step:=\rho$中的下一步并发执行τ;
17. WHILE $step$ 不为空;
18. 去掉 $c.CN$ 中这样的约束,此约束包含了不在 $c.I$ 和 $c.O$ 中的事件;
17. RETURN c;

在图 1 的场景中,以车载子系统为被测系统,运用算法 2 为系统正常运行对应的简单路径生成测试用例 c.其中,

- $c.I=(WCU_ATP,$禁止移动$,t_3)\rightarrow(WCU_ATP,\mathrm{MA},t_5)$;
- $c.O=(WCU_ATP,$开门授权$,t_1)\rightarrow(WCU_ATP,b$ 时刻开门$,t_2)\rightarrow(WCU_ATP,d$ 时刻关门$,t_4)$;
- $c.CN=\{t_3-b<10,t_5-d<10\}$.

如果想要驱动车载系统的执行覆盖其他简单路径,可以改变 t_2 或者 t_3 的值,使得延时约束得不到满足;也可以改变 $c.I$ 中的输入事件,报告站台屏蔽门系统故障.

3 实例研究

基于上述方法,我们设计实现了一个原形工具.它以扩展的活动图为输入,可以自动遍历场景找出所有的简单路径,再逐一为简单路径生成测试用例.我们以国产地铁列车控制系统作为研究对象,选取其中的 10 个安全攸关场景进行建模和测试用例生成,实验结果见表 1.需要指出的是:由于地铁列车控制系统十分庞大和复杂,涉及到众多子系统和设备,我们对这些安全攸关场景进行了一定程度的简化,去除了部分与设备有关的故障处理过程,重点保留了与系统交互和协同相关的处理过程.同时,我们人工计算了场景中应该需要覆盖的可能的执行

的数量,并把它们与简单路径进行了数量比较.

Table 1 Experimental results

表 1 实验结果

场景	生成测试用例数/简单路径数	应覆盖可能执行数量
点式列车运行	36	37
列车追踪运行	66	70
临时限速运行	24	24
列车间隔调整	34	36
故障引发紧急制动	42	44
自动驾驶与 ATP 监督下人工驾驶相互转换	68	74
ITC 模式列车进出站	22	24
CTC 模式列车进出站	32	37
ITC 下列车折返	26	30
CTC 下列车折返	36	38

实验结果表明:我们的方法可以有效地找出场景中的简单路径,并有效地生成对应的测试用例集合.同时我们也注意到,安全攸关场景中不可避免地有环形路径的存在.这是因为在失效或者故障发生后,系统会按照既定的策略进行几次重试或者故障恢复,这些执行也需要被覆盖,以确认重试和故障恢复是安全的.可以考虑在后面的工作中,允许环路进行有限次的展开,以处理这种情况.

4 相关工作

场景建模方法以及基于场景的测试用例生成方法一直是学术界研究的热点问题.Ryser等人[2,3]使用带标注的 UML 状态图对场景进行建模,标注可以对前置条件、后置条件、数据范围和数据值以及一些非功能性需求进行描述.同时,引入了依赖关系表格表示场景之间的依赖关系.他们通过遍历图中路径的方式找到需要的测试用例集合.

Lettrari 等人[4]对 UML 时序图进行扩展,添加了形如 if-then-else 的分支语句支持分支场景处理、时序图的激活模式和激活条件表示场景的动态激活以及消息传递时间机制表示消息传递过程中的时间约束.他们提供了基于 Rhapsody UML model 的原型工具,支持对场景的建模.

Tsai 等人[5,6]采用事件(event)、活动(action)和相关的前置/后置条件(pre-/post-conditions)的序列进行场景建模.先基于事件、活动、前置/后置条件的序列为每个子系统生成场景规约,再结合子系统之间的交互产生整个系统的场景,并采用随机方法生成测试用例.

Marchetti 等人[7]研究了对一个医疗系统和人力资源系统进行场景测试的实例,他们使用 UML 时序图进行建模,给出了通过定义消息序列、分析可能的子用例、定义设置分类、确定分支选择、确定选择间的约束关系等几个步骤生成测试用例的算法.该工作为两个实例的集成测试过程生成了所需的测试用例,发现了一些缺陷.

Kim 等人[8]研究了基于用例模型为实时的面向对象软件进行场景建模并生成测试用例的方法,他们引入用例的行为矩阵作为用例行为可执行序列的集合,以反映实时系统的行为特性,对场景进行建模.然后提出了基于场景的计划,用以度量并指导生成测试场景集合的一个有序排列.这一工作应用在一个实时不间断电源系统上.

Castillos 等人[9]提出了一种针对 UML/OCL 行为模型的基于场景的测试方法.他们引入正则表达式语法描述场景,可以方便地描述方法调用、欲被激活的活动和需要满足的即时状态,以及场景中的重复或选择结构.通过这个表达能力很强的场景建模语言,可以方便地描述组成场景和约束模型执行的操作序列.该工作最后基于一款商业工具“Test Designer”,提供自己的测试用例生成工具.

Salas 等人[10]提出了一种面向异步系统中场景的测试方法.他们指出:对于异步系统,场景建模难点在于如何处理好自身的不确定性和通信信道.他们的工作使用了事件结构作为基本描述手段,并对控制事件和可观察的事件进行区分,通过所提取出的仅含有控制事件的序列来生成测试用例.

Dadeau 等人[11]实现了一个名为 jSynoPSys 的工具.该工具采用 B machine 作为场景建模语言.该方法首先

描述场景的执行过程,以正则表达式描述系统的操作以及场景展开时系统所需达到的中间状态;然后,基于BZ-Testing-Tools 引擎,从模型出发进行有界测试生成.此项工作的特点在于,将用例的自动化生成工作转化为约束求解问题.

Auguston 等人[12]提出了一种支持实时系统的自动化场景生成方法,系统的场景被定义为一个事件的集合,该集合中的事件有两种基本关系:优先和包含.该方法提出一种“事件语法”以描述事件的迹,并以此描述系统的行为.最后,根据事件的迹自动生成测试用例.

从测试目标来看,上述研究工作可以分成两类:以事件为基础的场景模型[4,7,9,10,12]主要关注场景中多个参与者之间的交互行为,其对应的测试方法以检测交互行为的错误为主要目标;另一类场景模型[2,3,5,6,8,11]通过场景模型刻画与系统功能相关的操作序列,这类测试方法强调对功能进行覆盖,主要检测系统功能方面的错误.本文的研究工作通过场景模型同时对系统的功能和行为进行刻画,所提出的测试方法关注对系统所有可能的运行进行覆盖,以检测其中功能和行为是否能够满足规约的要求.就安全攸关场景的测试需求而言,本文方法发现错误的能力更强.同时,在上述工作中,事件机制、时间表示机制和并发执行机制对本文工作提供了很好的参考价值.

基于模型的形式推理验证工作也是保证安全攸关场景下系统运行安全性的重要手段.Lei 等人[13]采用混成自动机为列车控制系统中的追踪运行场景建模,并采用在线验证的方法对每个控制周期内的控制参数进行验证,保证前后列车不会发生碰撞.基于类似的技术,他们[14]还针对激光刀系统给出了激光手术场景中,防止氧气浓度过高引发起火燃烧的验证技术.Platzer 等人[15]基于混成逻辑系统构造安全攸关场景模型,采用推理技术对安全性进行验证.基于混成系统的建模与验证工作的主要技术瓶颈在于:非线性混成系统的验证非常困难,现在的工作都需要通过线性系统进行近似,现有验证算法能够处理的系统规模也比较有限.混成逻辑推理过程基本依靠人工手动进行,自动化非常困难,对人员的要求非常高.此外,无论是使用混成系统还是混成逻辑建模,都需要很高的技巧,难以很快被工业界接受和使用.可以预见,在今后相当长的时间内,形式建模与验证技术只能作为一种辅助手段,测试技术仍然是安全性保障中最主要的技术手段.

5 总 结

对安全攸关场景中各种执行进行充分的测试,是保障安全攸关系统安全性最主要的手段.本文提出了一种扩展 UML 活动图为列车控制系统中安全攸关场景建模的方法,并基于简单路径覆盖给出了对安全攸关场景进行覆盖的测试覆盖准则,并给出了关于该准则的测试用例自动生成方法.实例研究表明了该覆盖准则和方法的有效性.进一步的工作将考虑对简单路径覆盖准则进行适当扩展,允许有限次地展开循环的部分,以包含更多的需覆盖的运行.还包括对时段约束的处理以及相应的测试用例生成方法.同时,也考虑将模型与被测程序代码相结合,自动生成具体测试用例的方法.

References:

[1] IEC 61508 series. 2010. http://www.iec.ch/functionalsafety/

[2] Ryser J, Glinz M. Using dependency charts to improve scenario-based testing. In: Proc. of the 17th Int’l Conf. on Testing Computer Software. 2000. 46−57.

[3] Ryser J, Glinz M. A scenario based approach to validating and testing software systems using statecharts. In: Proc. of the 12th Int’l Conf. on Software and Systems Engineering and their Applications. 1999. 71−80.

[4] Lettrari M, Klose J. Scenario-Based monitoring and testing of real-time UML models. In: Proc. of the 4th Int’l Conf. on the Unified Modeling Languages, Concepts, and Tools. Berlin, Heidelberg: Springer-Verlag, 2001. 317−328. [doi: 10.1007/3-540-45441-1_24]

[5] Tsai WT, Yu L, Saimi A, Paul R. Scenario-Based object-oriented test frameworks for testing distributed systems. In: Proc. of the 9th IEEE Workshop on Future Trends of Distributed Computing Systems. IEEE Press, 2003. 288−294. [doi: 10.1109/FTDCS.2003.1204349]

[6] Tsai WT, Yu L, Liu XX, Saimi A, Xiao Y. Scenario-Based test case generation for state-based embedded systems. In: Proc. of the 2003 IEEE Int'l Conf. on Performance, Computing, and Communications. IEEE Press, 2003. 335−342. [doi: 10.1109/PCCC.2003.1203716]

[7] Marchetti E, Schilders L, Winfield S. Scenario-Based testing applied in two real contexts: Healthcare and Employability. In: Proc. of IEEE the 4th Int'l Conf. on Software Testing, Verification and Validation Workshops. IEEE Press, 2011. 89−98. [doi: 10.1109/ICSTW.2011.64]

[8] Kim RYC, Joo BG, Kim KC, Joen BK. Scenario based testing and test plan metrics based on a use case approach for real time ups (uninterruptible power system). In: Proc. of Int'l Conf. on Computational Science and Its Applications. Berlin, Heidelberg: Springer-Verlag, 2003. 646−655. [doi: 10.1007/3-540-44843-8_71]

[9] Castillos KC, Botella J. Scenario based test generation using test designer. In: Proc. of IEEE the 4th Int'l Conf. on Software Testing, Verification and Validation Workshops. IEEE Press, 2011. 79−88. [doi: 10.1109/ICSTW.2011.93]

[10] Salas PP, Krishnan P. Automated software testing of asynchronous systems. Electronic Notes in Theoretical Computer Science, 2009,253(2):3−19. [doi: 10.1016/j.entcs.2009.09.048]

[11] Dadeau F, Tissot R. jSynoPSys—A scenario-based testing tool based on the symbolic animation of B machines. Electronic Notes in Theoretical Computer Science, 2009,253(2):117−132. [doi: 10.1016/j.entcs.2009.09.055]

[12] Auguston M, Michael JB, Shing MT. Environment behavior models for scenario generation and testing automation. In: Proc. of the 1st Int'l Workshop on Advances in Model-Based Testing. New York: ACM Press, 2005. 1−6. [doi: 10.1145/1083274.1083284]

[13] Bu L, Wang Q, Chen X, Wang L, Zhang T, Zhao JH, Li XD. Toward online hybrid systems model checking of cyber-physical systems' time-bounded short-run behavior. ACM SIGBED Review, 2011,8(2):7−10. [doi: 10.1145/2000367.2000368]

[14] Li T, Tan F, Wang QX, Bu L, Cao JN, Liu X. From offline towards real-time: A hybrid systems model checking and CPS co-design approach for medical device plug-and-play collaborations. IEEE Trans. on Parallel and Distributed Systems, 2014,25(3):642−652. [doi: 10.1109/TPDS.2013.50]

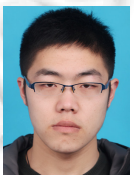
[15] Platzer A, Quesel J. European train control system: A case study in formal verification. In: Proc. of the 11th Int'l Conf. on Formal Engineering Method. Berlin, Heidelberg: Springer-Verlag, 2009. 246−265. [doi: 10.1007/978-3-642-10373-5_13]

陈鑫(1975－),男,辽宁大连人,博士,讲师,主要研究领域为软件工程,软件测试,验证技术.

姜鹏(1987－),男,硕士,主要研究领域为软件工程,软件测试,形式化方法.

张一帆(1989－),男,博士生,主要研究领域为软件工程,软件开发,设备驱动程序.

黄超(1988－),男,博士生,主要研究领域为信息物理融合系统的建模、验证、仿真、控制.



周岩(1991－),男,硕士生,主要研究领域为软件工程,软件测试,形式化方法.