一种符号化执行的实时系统一致性测试生成方法

万勇兵[1,2],徐中伟[1,2],梅 萌[1]
(1.同济大学电子与信息工程学院,上海 201804;2.嵌入式系统与服务计算教育部重点实验室,上海 201804)

摘 要: 系统一致性测试用来验证和确认系统实现的正确性.针对实时系统在进行数据处理时受到时间约束,容易导致状态空间爆炸的问题,提出一种符号化测试生成方法.首先对符号变迁系统和时间自动机进行扩展,建立一种新的符号语义模型 TSIOSTS,基于该模型定义了时间一致性关系(tioco);然后以 tioco 关系为指导,结合符号化执行策略,生成被测系统模型的时间符号化测试行为树,并转化为测试用例;最后将提出的理论和方法应用于 CTCS-3 列控系统临时限速服务器的一致性测试中,验证了该方法的可行性和有效性.

关键词: 实时系统; 一致性测试; 时间安全输入输出符号变迁系统; 符号执行; 测试用例生成
中图分类号: TP301 **文献标识码:** A **文章编号:** 0372-2112 (2013) 11-2276-09
电子学报 URL: http://www.ejournal.org.cn **DOI:** 10.3969/j.issn.0372-2112.2013.11.026

A Symbolic Execution Method for Conformance Test Generation of Real-Time System

WAN Yong-bing[1,2], XU Zhong-wei[1,2], MEI Meng[1]
(1. *School of Electronics & Information Engineering, Tongji University, Shanghai* 201804, *China*;
2. *The Key Laboratory of Embeddedsystem and Service Computing Ministry of Education, Shanghai* 201804, *China*)

Abstract: Conformance test is performed to verify and validate the correctness of a system implementa- tion. In view of easily causing the problem of state space explosion, and due to time constraint in data processing for real-time system, a symbolic test generation method is proposed. Firstly, symbolic transition systems and timed automata are extended to establish a semantic model TSIOSTS, based on which an extension of timed conformance relation tioco is defined. Then, with tioco and the symbolic execution strategy, a symbolic timed behavior tree of the system model under test is yielded and transformed into test cases. Finally, the proposed method is applied to conformance test of temporary speed restriction server of the CTCS-3 train control system, and the results present feasibility and validity of the method.

Key words: real-time system; conformance test; time safety input-output symbolic transition system; symbolic test generation; test case generation

1 引言

实时系统(Real-Time Systems,RTS)是一种能够在指定的时间内完成系统功能并对外部或内部、同步或异步时间做出响应的系统,其系统的正确性不仅取决于系统计算的逻辑结果,还依赖于产生这个结果的时间[1].一致性测试是当前验证和确认实时系统正确性的重要技术手段之一,其目的是确认被测系统实现(IUT)与其系统规范(specification)是否一致.为了测试实时系统的一致性,不仅要考虑系统的输入输出行为,还要检查这些行为是否在其规定的时间内发生.在实时系统的一致性测试理论和方法上,已取得了不少研究成果[2~12].其中:文献[2]提出一种实时系统一致性测试框架.该方法用非确定性 TAIO 对系统规范进行建模,并采用 on-the-fly 策略生成并执行测试用例.文献[5]利用 TSIOA 模型来描述实时系统行为,采用符号状态拆分算法和抽象时间迁移去除技术,生成满足时间延迟极值覆盖标准的测试用例.文献[6,7]结合符号变迁系统和时间自动机提出一种符号时间自动机模型,描述系统的输入输出数据流和实时行为,并扩展得到符号一致性关系 sioco,在此基础上生成符号一致性测试用例.文献[9]提出一种基于时间 Petri 网的实时系统一致性测试方法,利用模型

收稿日期:2012-11-26;修回日期:2013-04-27;责任编辑:覃怀银
基金项目:国家自然科学基金(No.61075002,No.61273180);"十二五"国家科技支撑资助项目(No.2011BAG01B03);铁道部科技重点资助项目(No.2009X002-A)

检测工具 TINA 优化执行时间,生成时间一致性测试用例.

也有一些学者提出将实时系统模型转换为不含时间约束的形式模型,以消除时间约束对系统行为造成的影响,然后采用时间无关的测试生成方法.文献[10]将被测系统的 TA 模型转换为有限状态变迁图,再将变迁图转换为 FSM 模型,最后采用基于 FSM 的方法生成测试用例;文献[11]提出一种实时测试和非实时符号化测试组合的测试策略,该方法仅考虑了系统的输入输出行为,虽实现了系统时间模型与非时间模型的转换,但描述模型时间约束的能力受到极大限制.研究表明,由于实时系统引入了时间维,使得被测系统的行为空间变得非常庞大,加上系统中的时间约束也增加了确定系统行为的难度.因此,如何解决系统行为和时间约束之间的关系,避免状态空间爆炸问题仍是学术界和业界的研究热点之一.

本文主要针对如何解决系统行为和时间约束之间的关系,避免状态空间爆炸问题展开,提出一种符号化执行的实时系统一致性测试生成方法:首先对文献[2,12]中的模型进行扩展,给出时间安全输入输出符号变迁系统(Time Safety Input-Output Symbolic Transition System-TSIOSTS),采用统一符号语义模型对系统行为和时间约束进行描述,以准确刻画系统行为与其时间约束之间的关系,用抽象数值替代具体数值对变量参数进行符号化描述,极大地减少了状态空间的容量;然后从扩展语义模型出发,定义时间一致性关系,结合符号执行策略对变量参数和时间约束进行抽象分析,提出一种基于时间一致性关系的测试生成算法,生成 IUT 模型的时间符号化测试行为树,并转化为测试用例,在一定程度上有效地缓解了状态空间爆炸问题;最后通过实例表明该方法的可行性和有效性.

2　一致性测试框架

一致性测试是系统生命周期中的重要环节.根据文献[13]中对一致性测试方法的规定,测试标准包括 3 大部分,如图 1 所示:左边分支表示 IUT 及其测试环境的配置,右边分支表示测试实现过程,根据由不同功能特征点所组成的系统需求规范,以一致性关系和测试目的为测试依据,经过一致性测试生成过程,生成抽象测试用例集(ATS);并通过测试用例的选择和参数化转化为可执行的测试用例集(ETS);最后将所得 ETS 与 IUT 进行交互执行,并根据测试执行的结果进行测试判决.与上述过程相对应,本文第 3 节给出一种形式化模型来描述被测系统的需求规范,第 4 节基于该模型提出时间一致性关系,第 5 节研究基于一致性关系的测试生成和执行问题,第 6 节给出实例研究.

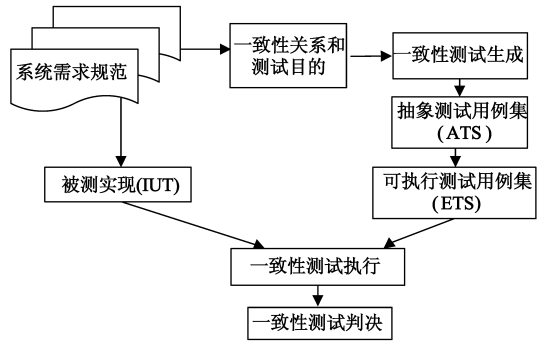

图1　一致性测试的结构

3　基于 TSIOSTS 的形式化建模

3.1　时间安全输入输出符号变迁系统

定义 1(TSIOSTS)　一个时间安全输入输出符号变迁系统 M 是一个 7 元组$(Q, V, D, q_0, \sum, C, T)$,其中:$Q$ 是有限非空的状态集合,$q_0 \in Q$ 是初始状态;V 是一个有限类型变量集;D 是有限可数数据集,对于 $x \in V \cup D$ 用 type(x)表示数据类型;$\sum = \sum^{?} \cup \sum^{!} \cup \sum^{\tau}$ 是一组非空、有限的符号集合,$\sum^{?}$ 表示输入活动,$\sum^{!}$ 表示输出活动,$\sum^{\tau}$ 表示内部活动;C 是一组有限的时钟集合$\{d_1, d_2, \cdots, d_{|N|}\}$,$|N|$是时钟数,$v_i \in R^{+}$(非负实数)表示 d_i 的时钟值.T 是一组有限变迁集合(q, a, P, R, y, q'),其中:$q, q' \in Q$ 分别代表源状态和目标状态;$a \in \sum$表示输入或输出活动;P 是一个线性不等式的布尔合取范式 $P(v)$,表示时间约束;$R \subseteq C$ 表示在该变迁使能后重置为 0 的时钟集合(在本文提出的模型中,假设变迁的时间约束的形式均为 $\wedge (v_i \cong d)$,其中 $\cong \in \{>, <, \geqslant, \leqslant, =\}$,$d \in R^{+}$);$y \in \{\text{lazy}, \text{delayable}\}$表示变迁使能的延迟时间类型;变迁$(q, a, P, R, y, q') \in T$ 也可记作:$q \xrightarrow{a[p]/R} q'$.

本文定义 2 类变迁使能的延迟时间类型[14]:(1) lazy:输入活动变迁均属于这类变迁,这是因为输入活动是由外部环境控制,也许不会发生,也称为"非强制输入";(2)delayable:输出活动变迁均属于这类变迁,规定在变迁使能的有效时间内,相应的输出活动必须发生.当没有特别规定,我们假定输出活动变迁使能的延迟时间为 delayable,输入活动变迁使能的延迟时间为 lazy.

图 2 是一个用 TSIOSTS 描述的实时系统的例子.其中:"?"表示输入动作,"!"表示输出动作.$\sum = \{a, b, c, d, e, f, g\}$表示系统的活动集,其行为受到时间约束 $C = \{t_1, t_2\}$.$Q = \{q_0, q_1, q_2, q_3, q_4, q_5, q_6, q_7\}$为系统状态集合,系统从 q_0 状态开始,初始条件 P 为 true.

定义 2(TSIOSTS 的语义)　一个 TSIOSTS 模型 $M = (Q, V, D, q_0, \sum, C, T)$的语义可以描述为一个时间输

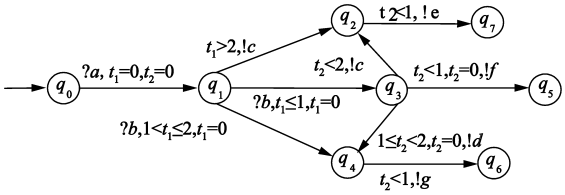

图2 简单TSIOSTS模型

入输出标号迁移系统(TIOLTS),即:$[[M]]=(S,s_0,\sum,T)$,其中 $S=Q\times V\times(C\to R^{\geq 0})$是状态集合,对于状态 $s=(q,v,\psi)$,有 $q\in Q$ 表示位置,$x\in V$ 表示在变量集 V 中的一个映射 $v(x)$,$\psi: C\to R^{\geq 0}$表示对时间赋值,0 表示该时钟重置为零;$s_0=\{(q_0,v,\psi)\mid\varphi(v)=\text{true},\overline{0}\}$是初始状态集;$\sum=\Lambda\cup D$ 是活动集合,其中 $\Lambda=\{(\alpha,\gamma)\mid a\in\sum,\gamma\in\Gamma_{\text{sig}(a)}\}$是离散活动集合,$D=R^{\geq 0}$时间活动集合,$\Lambda$ 分为两部分:$\Lambda^{?}$表示输入活动集合,$\Lambda^{!}$ 表示输出活动集合,Λ^{τ} 代表内部活动;$T\in S\times(\sum\cup R^{+})\times S$ 代表变迁集合,包含时间活动变迁和离散活动变迁,其中(1)时间活动变迁$(q,v,\psi)\xrightarrow{d}(q,v',\psi')$描述了时间的自然流逝,如果存在 $0\leqslant d_1\leqslant d_2\leqslant d$,$d\in R^{+}$表示延迟时间,$\psi=\psi+(d_1,d_2,\cdots,d_n)$,即在 d 时间单位内,TSIOSTS 上没有发生输入输出活动,位置保持不变;(2)离散活动变迁形式为:$(q,v,\psi)\xrightarrow{a}(q',v',\psi')$,即如果存在一个变迁 $t:(q,a,P,R,y,q')\in T$,满足时间约束 $P(P(v)=\text{true})$,则系统通过活动 a 从状态(q,v,ψ)变迁使能到状态(q',v',ψ').其中 v'是由 v 的时钟更新函数得到,定义如下:

$$v'=\text{Update}_R(v)=_{\text{def}}v_i'=\begin{cases}v_i, & d_i\notin R\\ 0, & d_i\in R\end{cases}$$

具备以上的语义模型后,我们可以给出基于TSIOSTS 模型的系统行为描述,本文用时间迹(timed traces)描述模型的活动行为.为了方便讨论,假设TSIOSTS模型 M 的语义为$[[M]]=(S,s_0,\sum,T)$,令 s,s',$s_i\in S$;$a_1,a_2,\cdots,a_n\in\sum$,$\varepsilon$ 表示空活动,$\sigma\in(\sum)^{*}$表示一组离散活动和时间可伸缩活动的序列,定义活动行为如下:

(1)$s\xrightarrow{\rho(e)}s'$,表示系统在状态 s 进行数据处理$\rho(e)$后进入状态 s';

(2)$s\overset{\sigma}{\Rightarrow}s'=_{def}\exists\, s_0,s_1,\cdots,s_i\in S,s\overset{\sigma}{\Rightarrow}s_0\xrightarrow{a_1}s_1\xrightarrow{a_2}\cdots\xrightarrow{a_i}s_i\overset{\sigma}{\Rightarrow}s'$,表示系统从状态 s 经过某个活动序列到达状态s';

(3)$s\xrightarrow{\sigma}=_{\text{def}}\exists\, s',s\xrightarrow{\sigma}s'$,$\sigma$ 是一个时间迹;

(4)s afte r $\sigma=_{\text{def}}\{s'\in S:s\overset{\sigma}{\Rightarrow}s'\}$,表示状态 s 经历某个活动序列σ 到达某些状态s'的集合;

(5)$\text{out}(s)=_{\text{def}}\{a\in\sum^{!}\cup R^{+}\mid\exists\, s':s\xrightarrow{a}s'\}$,表示状态 s 下所有可能的输出集合;

(6)$\text{traces}(s)=_{\text{def}}\{\sigma\in(\sum\cup R^{+})^{*}\mid\exists\, s':s\xrightarrow{\sigma}s'\}$,表示状态 s 下所有可能的可观察时间迹的集合.

在时间迹中,相邻的两个延时活动可以合并,称为"时间可加性".即 $s_1\xrightarrow{d_1d_2}s_2$ 可表示为 $s_1\xrightarrow{d_1+d_2}s_2$.因此,一个时间迹可以用如下形式表示:$\sigma=d_0a_0d_1a_1\cdots d_na_n(a_i\in\sum,\ d_i\in R^{+})$.实际上,$d_i$ 是外部可观察动作 a_{i-1}和 a_i 之间的相对时间间隔.在本文中只考虑这种时间迹.

一方面,如果 TSIOSTS 模型 M 在任何位置都可以接受任何的活动,则称 M 是完全的,即$\forall\, s\in S,b\in\sum:s\xrightarrow{b}$.另一方面,如果 M 在任何位置,或是在内部活动之后,都可以接受任何的输入,则称 M 是输入完全的(input-complete),即:$\forall\, s\in S,b\in\sum^{?}:s\Rightarrow$.为使模型 M 输入完全,假定在某个状态下,如果发生规范模型中未说明的输入动作,则模型保持原有状态不变,即在模型中引入自环(self-loop)变迁.

当以下条件满足时,模型 M 被认为是确定的:(1)系统没有内部活动,即$\sum^{\tau}=\varnothing$;(2)系统最多只有一个初始位置,即$|s_0|=1$;(3)对所有的 $q\in Q$,对带有相同活动 a 的不同变迁,$t_1:(q,a,P_1,R_1,y_1,q_1')$和 $t_2:(q,a,P_2,R_2,y_2,q_2')$,时间约束 P_1 和 P_2 是互斥的(即 $P_1\wedge P_2$ 是不满足的).

4 时间一致性关系

在一致性测试中,一致性关系和测试目的在整个测试过程中起着指导性作用.一致性测试通过一致性关系把系统规范 Spec 和被测系统实现 IUT 关联在一起,根据测试目的,经过一致性测试生成过程,生成测试用例集,并通过执行测试用例来给出测试判决.在本文中,给出一种基于时间测试目的(Timed Test Purpose,TTP)的测试生成方法来指导测试,即在测试过程中检验 IUT 在规定时间内的预期行为.我们设定时间测试目的是一个特定的 TSIOSTS 模型,其形式化描述如下:

定义 3(时间测试目的) 给定一个带有活动集$\sum$的系统规范的 TSIOSTS 模型 M,其时间测试目的是一个带有特殊位置集(Accept$\in Q_{\text{TTP}}$)的 TSIOSTS 模型 $\text{TTP}=<Q_{\text{TTP}},V_{\text{TTP}},D_{\text{TTP}},q_0^{\text{TTP}},\sum_{\text{TTP}},C_{\text{TTP}},T_{\text{TTP}}>$,其中 TTP 是完全和非循环的,且与 M 相容,即$\sum_{\text{TTP}}=\sum_S$.特别的,对$\forall\, q\in Q_{\text{TTP}}$,有 $P_{\text{TTP}}(q)=\text{true},C_{\text{TTP}}=C_S$.

一般来说,时间测试目的代表着我们试图验证的系统属性.而对于一个实时系统的被测实现,该属性即是一组在规定的时间范围内,系统与外部环境的交互行为的活动

集.这里增加的状态 Accept 用来表示用 TTP 描述的测试想定(test scenario)满足该性质.图 3 描述了一个实时系统(图 2)的时间测试目的.用它来选择一个测试想定,接受以下一个测试序列:一个输入活动 a 发生后,在不大于 1s 的时间内输入活动 b,最后在 1s 内输出活动 e.

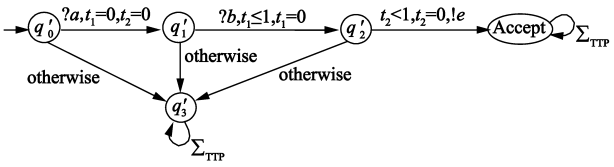

图3　图2中规范说明的时间测试目的

对于 IUT 来说,执行一致性测试时必须在其接口处观察其对激励的响应,从而确定其基于测试所产生的外部响应与规范说明提供的结果是否满足一致性关系.因此,本文只考虑基于规范的时间一致性关系[2,6],且给出以下测试想定:(1)被测实现是输入完全,确定的,并且与规范说明有相同的外部接口;(2)被测实现,时间测试目的,测试用例都可以用 TSIOSTS 模型进行描述;(3)输入输出活动是无丢失无延迟的,即输入输出活动的传输时延近似为 0,其与规范中的时间约束相比,可忽略不计.以下采用基于时间迹的语义对基于 TSIOSTS 模型的时间一致性关系进行形式化的定义.

定义 4(时间一致性关系)　一个被测实现 IUT 与一个规范说明 Spec 满足时间一致性关系 tioco,记为 IUT tioco Spec,当且仅当:

$$\forall \sigma \in \mathrm{Trace(Spec)}, \mathrm{Out}(\mathrm{IUT\ after}\ \sigma) \subseteq \mathrm{Out}(\mathrm{Spec\ after}\ \sigma)$$

5　基于 TSIOSTS 的一致性测试

为降低状态空间爆炸给测试用例生成带来的难度,大多数研究采用状态等价关系来对系统进行简化.这里,通过扩展文献[5]得到基于区图的实时系统一致性测试生成方法,其基本思想均是将模型中的一个状态和一个时间区域一起构成一个符号状态以生成有限状态模型.所不同的是本文提出的方法以 TTP 为指导,找到 ITU 中所有可能的时间迹.针对模型中时间的连续性,将一组等价的时间迹表示为一个参数化的时间迹,极大地减少了模型中的状态数量.然后检查其是否符号 sioco 关系,结合符号执行策略将不满足执行条件的分枝裁剪掉,同时生成满足条件的测试用例,避免了对无限数据域(或时间域)的数值枚举,在一定程度上缓解了状态空间爆炸问题.文章将基于 TSIOSTS 的一致性测试分为测试执行和测试选择两个过程.

5.1　计算同步积

基于时间一致性关系和时间测试目的的实时系统一致性测试,首先通过计算得到规范说明 Spec 和时间测试目的 TTP 的同步积 STP(见算法 1).然后,对所得结果进行符号化执行,选择确定的结果和导致系统进入 Accept 状态的可能时间迹;最后,将选取的时间迹转化成测试用例.

算法 1　计算同步积算法

输入:规范说明的 TSIOSTS 模型 $S=(Q,V,D,q_0,\sum,C,T)$

　　时间测试目的 TSIOSTS 模型 $\mathrm{TTP}=(Q_{\mathrm{TTP}},V_{\mathrm{TTP}},D_{\mathrm{TTP}},q^0_{\mathrm{TTP}},\sum_{\mathrm{TTP}},C_{\mathrm{TTP}},T_{\mathrm{TTP}})$

输出:同步积 $\mathrm{STP}=(Q_{\mathrm{STP}},V_{\mathrm{STP}},D_{\mathrm{STP}},q^0_{\mathrm{STP}},\sum_{\mathrm{STP}},C_{\mathrm{STP}},T_{\mathrm{STP}})$

1: Init(Q_R)→∅ and Init(Q_H)→∅; // Q_R 表示可达状态集,Q_H 表示已访问状态集

2: Let $q^0_{\mathrm{STP}}:=(q^0_S,q^0_{\mathrm{TTP}})$;

3: AddState(q^0_{STP}, Q_{STP}); // 将同步积的初始状态 q^0_{STP} 添加到状态集 Q_{STP} 中

4: Let $C_{\mathrm{SP}}:=(C_S\cup C_{\mathrm{TTP}})$;

5: AddState(q^0_{SP}, Q_R);

6: while($Q_R \setminus Q_H \neq \varnothing$) do

7:　　q:= SelectState((q_1, q_2), $Q_R \setminus Q_H$); // 从状态集 $Q_R \setminus Q_H$ 中取出状态(q_1, q_2)

8:　　AddState(q, Q_H);

9:　　if ($q_1 \xrightarrow{a[P_1]/d_1}{}_s q' \in T_s$) and $q_2 \xrightarrow{a[P_2]/d_2}{}_{\mathrm{TTP}} q_2' \notin T_{\mathrm{TTP}}$ then;

10:　　AddState((q_1', q_2), Q_R);

11:　　AddState($(q_1,q_2) \xrightarrow{a[P_1]/d_1}{}_{\mathrm{STP}} (q_1', q_2)$) to T_{STP};

12:　end-if

13:　if ($q_1 \xrightarrow{a[P_1]/d_1}{}_s q' \in T_s$) and $q_2 \xrightarrow{a[P_2]/d_2}{}_{\mathrm{TTP}} q_2' \in T_{\mathrm{TTP}}$ then;

14:　　AddState((q_1', q_2'), Q_R);

15:　　AddState($(q_1,q_2) \xrightarrow{a[P_1\&P_2]/(d_1\cup d_2)}{}_{\mathrm{STP}} (q_1', q_2')$) to T_{STP};

16:　end-if

17: end-while

下面分析算法 1 的复杂度.算法中第 6 行的 while 循环对每个可达状态进行计算,其最坏情况下,同步积中的状态数为($|Q_S|\cdot|Q_{\mathrm{TTP}}|$),变迁数为($|Q_S|+|Q_{\mathrm{TTP}}|$).因为 $|T_{\mathrm{TTP}}|$ 远远小于 $|T_S|$,所以算法 1 总的时间复杂度为 $\sum_{i=1}^{n}((|Q_S|\cdot|Q_{\mathrm{TTP}}|)\cdot|T_S|)$,即算法的时间复杂度与系统规范规模和测试目的规模成正比.图 4 所示是由图 2 描述的系统规范和图 3 描述的一个时间测试目的得到的同步积.

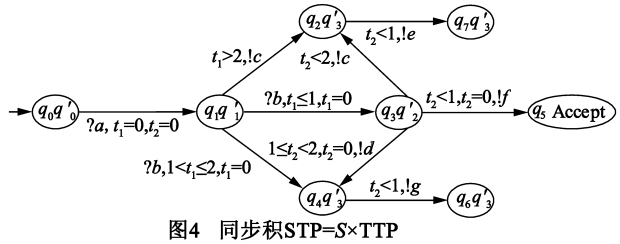

图4　同步积STP=S×TTP

5.2 测试执行

符号执行的基本思想是使用符号值代替具体的数据作为程序输入来模拟程序的执行[15].在执行程序的过程中采集与输入相关的路径约束条件并进行求解,得出符号变量的具体值.本文在符号执行的基础上进行,对系统 TSIOSTS 模型中的活动参数和变量,以及所有可能产生的时间迹都使用符号值代替具体的数据值,即采用参数化的时间迹、基于区的符号状态(Zone-Based Symbolic State,ZSS)、符号变迁(Symbolic Transition,ST)和符号通信活动(Symbolic Communication Action,SCA)来描述模型中所有动作变迁行为产生的时间迹[16,17].

定义 5(基于区的符号状态) TSIOSTS 模型 $M=(Q,V,D,q_0,\Sigma,C,T)$的一个 ZSS 是一个四元组 $\eta=(q,\pi,\gamma,Z)$,其中:(1)$q\in Q$ 是 W 中的一个状态;(2)π 是路径条件,用布尔表达式约束其符号变量必须满足该条件;(3)γ 是变量和活动参数到其符号值的一个映射关系 $\gamma: V\rightarrow V_{\text{Symb}}$,其中 $V\cap V_{\text{Symb}}=\varnothing$;(4)$Z$ 是一个时间区域,对于时间参数 d,用 $d_i\in Z(d)$来表示一个时钟约束的解集.这里我们假定对于 $d_1,d_2\in Z(d)$,由于属于同一个时间区域,它们具有相同的活动行为,即如果状态(q,d_1)接受一个时间迹 σ,那么(q,d_2)也同样接受 σ.

定义 6(符号变迁) 一个 ST 是一个三元组 st=(η,sca,η'),对 $\forall$ st$\in$ST,有 $\eta,\eta'\in S$,分别表示源 ZSS 和目标 ZSS,sca$\in$SCA 是一个符号通信活动.

定义 7(符号通信活动) 一个 SCA 是一个三元组 sca=$(a,\mu_{\text{sca}},\gamma_{\text{sca}})$,其中:(1)$a\in\Sigma$是 TSIOSTS 模型中的一个活动;(2)$\mu_{\text{sca}}$是一个全局标识列表,用来表示一个 sca 的活动参数;(3)γ_{sca}是列表 μ_{sca}中从起始活动参数名到符号通信活动全局标识的一个映射关系.

对具有时间约束的实时系统进行一致性测试,其测试用例是一个符号化的测试行为树.为生成时间一致性测试用例,基于 TSIOSTS 模型的符号化执行必须考虑数据和时间的约束关系.从被测系统模型的全局初始状态 S_0 出发构造测试行为树.测试行为树的节点用 ZSS 表示,边用 SCA 来表示.

定义 8 (时间符号化测试行为树,STBT)一个 STBT 用一个四元组 stbt=$(S,\text{SCA},\eta_0,\text{ST})$表示,其中:$S$ 是一组有限的基于区的符号状态集,SCA 是一组有限符号通信活动集,η_0 是一个基于区的初始符号状态,ST 是一组有限符号变迁集.

在文献[18]中提出符号变迁系统的符号执行算法,但未考虑时间因素的影响.文本在此基础上对符号执行算法进行扩展.扩展算法将待执行的 TSIOSTS 模型 M 作为输入,输出的结果是一个 STBT.对于 STBT,每条边用 sca 描述一个测试行为$(a,\mu_{\text{sca}},\gamma_{\text{sca}})$;其中 a 代表一个输入/输出动作符号.每个节点用 ZSS 描述一个测试行为的初始/结束状态.算法 2 描述了构造一个 STBT 的过程:即从初始状态 η_0 出发,经过时间迹 σ 后到达目标状态 η'.其中 IsSatisfiable(π')表示状态 η'满足路径条件 π',并且时间区域 Z'不为空;IsBoundReached(q')表示与状态 q'关联的当前路径中 ZSS 的数量没有超过指定的上限值;IsPrune(η')表示对出现状态重复情况的 η'进行裁剪,以减少符号通信状态的数量,其都是执行过程中提供的支持服务,由于篇幅关系,在此不作赘述.

算法 2 符号执行扩展算法

输入:同步积 STP=$(Q_{\text{STP}},V_{\text{STP}},D_{\text{STP}},q^0_{\text{STP}},\sum_{\text{STP}},C_{\text{STP}},T_{\text{STP}})$

输出:时间符号化测试行为树 STBT=$(S,\text{SCA},\eta_0,\text{ST})$

1: Letγ_0: = $\{V\cup D\}$; //初始化符号变量

2: Letη_0: = $(q_0,\text{true},\gamma_0,Z_0)$; //定义 TR 的初始符号状态,$\pi_0$ = true 为路径的初始条件

3: AddState(stbt,η');

4: Init(S_{UH})$\rightarrow\{\eta_0\}$; //将未被访问过的状态添加到未访问的状态集 S_{UH}中

5: while($S_{\text{UH}}\neq\varnothing$) do

6: DeleteState(S_{UH},η); //删除访问过的符号状态,然后循环处理与模型中状态 q 相关联的变迁

7: for each$(q,a,P,R,y,q')\in T$ do

8: μ_{sca}: = $V_{\text{Symb}}\cup D$;

9: γ_{sca}: = $V\rightarrow V_{\text{Symb}}$;

10: sca: = sca$\cup(a,\mu_{\text{sca}},\gamma_{\text{sca}})$; //得到符号活动集,以便在后面计算目标符号状态

11: π': = $\pi\wedge\sigma(\sigma_{\text{sca}}(P^D))$;

12: γ': = $\gamma\circ\gamma_{\text{sca}}\circ R^D$; //"$\circ$"表示函数组合,$R^D$ 代表时间设定

13: Z': = $(R^C\cap\vec{Z})\cup\text{Update}(R^C)$; //$\vec{Z}$ 表示时间自然流逝

14: η': = $<q',\pi',\sigma',Z'>$; //得到目标符号状态

15: if(IsSatisfiable(η')$\wedge\neg$(IsBoundReached(q'))$\wedge$IsPrune(η')) then

16: S_{UH}: = $S_{\text{UH}}\cup\{\eta'\}$; //满足上述条件后,将目标符号状态添加到未访问的状态集中

17: AddState(stbt,η');

18: AddTransition(stbt,st);

19: end-if

20: end-for

21: end-while

对算法 2 的复杂度进行分析.算法 2 中第 5 行的 while 循环对每个未被访问的状态进行计算,至到未被访问的状态集为空,一般情况下,该状态数为($|Q_{\text{STP}}|$),变迁数为($|T_{\text{STP}}|$).所以算法 2 总的时间复杂度为 $\sum_{i=1}^{n}(|Q_{S_{\text{TP}}}|\cdot|T_S|)$.图 5 给出了由图 4 中经过符号执行后获取的时间符号化测试行为树 STBT.

5.3 测试生成

通过符号化执行确认所有可能的时间迹之后,需要选择一个导致 Accept 状态出现的 STBT 子树,称为测试树(TT),即随机选取一个满足 Accept 可达状态 $\eta^A=(q^A,\pi^A,\gamma^A,Z^A)$向前搜索直到 STBT 的根节点为止,文献[18]描述了 Forward Traversal 方法的具体过程.最后生成满足时间一致性关系的测试用例,只需将满足条件的测试树(S,SA,η_0,T)转换成一个测试用例 TC=(Q_{TC},V_{TC},D_{TC},q^0_{TC},$\sum_{TC}$,C_{TC},T_{TC},Verdict).见算法 3,测试用例的数据在 STBT 中用符号值表示,其符号值可作为测试用例的变量和参数.这里假定 $\eta_0=(q_0,\pi_0,\gamma_0,Z_0)$是 STBT 的初始状态,初始条件 π_0 为 true,位置集为 Q,初始位置是 q_0,字符列表为$\bigcup_{<a,\mu_{sca},\gamma_{sca}>\in SCA}a$,时钟集为 C_{SP}.

对所有的符号变迁(η,sa,η')∈ST 进行处理,其每一个变迁在测试用例中可以得到一个新的变迁(q,a,P,R,y,q')∈T.对于源符号状态 η,新变迁的活动是一个带有参数 μ_{sca}的符号通信活动 sca=(a,μ_{sca},γ_{sca});通过该活动到达目标符号状态 η',需要同时满足路径条件和活动的时间约束;需被重置为零的时钟 R,通过时钟赋值更新函数 Update()重新定义;延迟时间类型 y 与 STP 中的活动保持一致.最后,通过计算测试断言函数Verdict(a)得到满足 tioco 的测试用例,即 Verdict(a)=pass 表示 IUT 在经过 σ 的交互后产生模型M 允许的输出,IUT 与 M 一致;反之,Verdict(a)=fail 为 M 不允许的输出,说明 IUT 与 M 不一致.

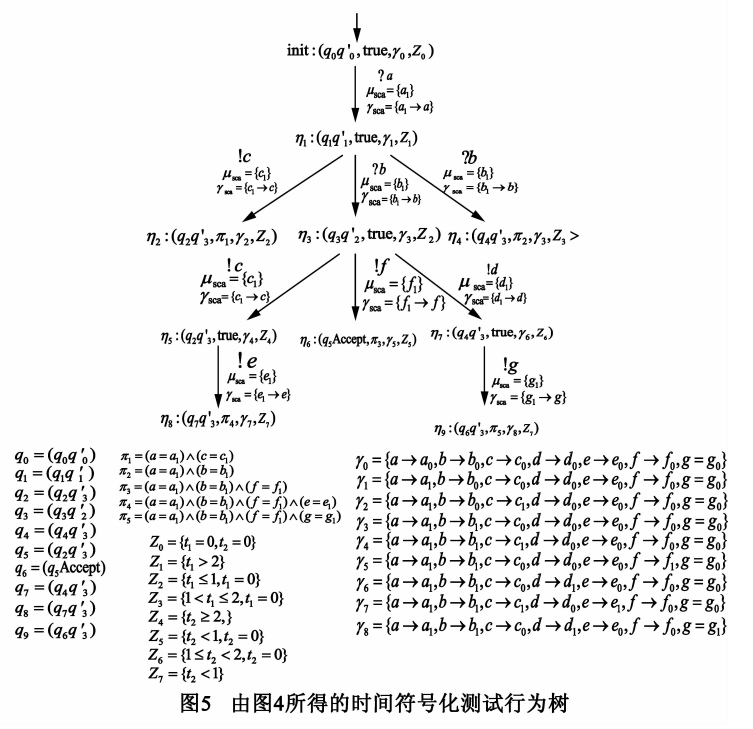

图5　由图4所得的时间符号化测试行为树

算法 3　符号执行树转换为测试用例算法

输入:同步积 STP=(Q_{STP},V_{STP},D_{STP},q^0_{STP},$\sum_{STP}$,C_{STP},T_{STP})
　　测试树 TT=(S,SCA,η_0,ST)

输出:测试用例 TC=(Q_{TC},V_{TC},D_{TC},q^0_{TC},$\sum_{TC}$　,C_{TC},T_{TC},Verdict)

1: Init(V_{TC})→V_{Symb} and Init(D_{TC})→{x|x∈V∪D};
2: Let $\eta_0=(q_0,\text{true},\gamma_0,Z_0)$;//指定 η_0 为测试树的初始符号状态
4: Q_{TC}:=S;
5: q^0_{TC}:=η^0;
6: $\sum_{TC}$:=$\bigcup_{(a,\mu_{sca},\tilde{a}_{sca})\in SCA}a$;
7: C_{TC}:=C_{STP};
8: for each<η,sca,η'>∈T_{ZSS} do
9:　　q:=η;
10:　　a:=(a,μ_{sca},φ_{sca});//将带有参数 μ_{sca}的符号通信活动表示输入/输出活动
11:　　P:=$\pi\cup R^C$;//需要满足的路径条件和时间约束
12:　　R:=Update(R^C);//通过时钟赋值更新函数对时钟重新定义
13:　　y:=y_{STP};
14:　　q':=η';
15:　　AddTransition (TC,(q,a,P,R,y,q'));
16:　　AddVerdict(TC,Verdict(a));//Verdict(a)={pass,fail}为测试断言函数
17: end-for

算法 3 中第 8 行的 for 循环对 ZSS 中每个变迁进行计算分析,一般情况下,为 STBT 中导致 Accept 状态出现的测试树的变迁数(|T_S|).所以算法 3 总的时间复杂度为 $\sum_{i=1}^{n}(|T_S|)$.图 6 给出了从图 5 的 STBT 中获得的测试用例.

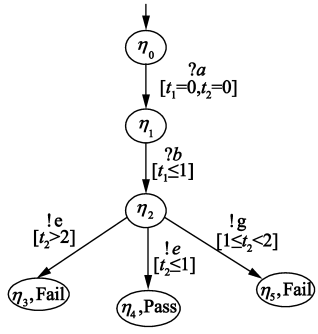

图6　从图5的ZSET中获得的测试用例

6 应用举例

下面以 CTCS-3 列控系统临时限速服务器(TSRS)为例,将上述模型和方法应用到该系统的一致性测试中,以说明本文提出的测试生成算法.

6.1 临时限速服务器

临时限速(TSR)是指线路固定限速以外的、具有实效性的限速,包括:施工、维修引起的计划性限速;自然灾害、设备故障引起的突发性限速等.TSRS 是一个典型的实时系统,其主要功能包括:对 TSR 的存储、校验、删除、设置、执行和取消,以及限速设置时机的辅助提示;

临时限速命令的一致性检测和维护及与管辖范围内的列控中心(TCC)和无线闭塞中心(RBC)及相邻的 TSRS 等地面设备建立通信等.本节以 TSRS 的临时限速命令执行功能为例说明本文提出的实时系统一致性测试方法.在临时限速命令执行过程中,由集中调度中心(CTC)下发限速命令(SetTSR),TSRS 对该命令进行校验并返回校验回执信息.若 CTC 在 3 个 T_{cycle}内没有收到 TSRS 的反馈信息,则重发该信息.对校验成功的限速命令,CTC 根据提示,选取并激活即将执行的 TSR 命令并向 TSRS 发送验证限速命令(ActTSR).TSRS 对接收到的命令进行拆分和转换为相应目标设备所识别的验证命令.目标 TCC 和 RBC 分别对接收到的命令进行有效性判断并反馈验证结果信息给 TSRS.当全部设备验证成功,TSRS 向 CTC 发送验证通过信息,若存在任一设备验证失败或 3 个 T_{cycle}内未返回验证结果,则向 CTC 返回限速验证失败.对验证成功的 TSR 命令,CTC 向 TSRS 发送执行限速命令(ExcTSR).TSRS 将命令分发给目标 TCC 和 RBC 执行并将执行信息反馈给 TSRS.TSRS 对执行结果进行综合判定,若存在任一设备执行失败或 3 个 T_{cycle}内未返回执行结果,则向 CTC 返回执行失败;若全部设备执行成功,则向 CTC 返回执行成功信息.

6.2 TSRS 建模

如图 7 是 TSRS 临时限速命令执行功能的 TSIOSTS 模型 $M=(Q,V,D,q_0,\sum,C,T)$.模型中 $\sum^{?}=\{$SetTSR, ActTSR, ExcTSR$\}$, $\sum^{!}=\{$SaveTSR, SetNo, ActNo, ExcNo, ActSucc, ExcSucc, Err, Delay$\}$, $Q=\{q_0, q_1,\cdots,q_{12}\}$, $D=\{$tsr, msg$\}$, $V=\{0<\text{tsr}\leqslant 300;\text{msg}=m_i:i=1,2,3,4\}$, $C=\{t_1,t_1,t_3\}$.其中,变量 tsr 表示有效地限速命令值,msg 表示回执的错误类型,初始状态为 q_0,表示被测系统内没有临时限速命令.

6.3 测试生成

被测系统的 TSIOSTS 模型建立之后,给出一个如图 8 所示的时间测试目的 TTP,采用第 5 节给出的相应算法,生成时间符号化测试行为树,并得到如图 9 所示的测试用例.

对于图 9 中的变量 tsr 的参数,我们定义一个数据集为$\{45,60,80,120,160,200,250,300,350\}$.而对于时间 t 的参数,理论上可取 0 到 R^+间的任意值,但通常在实际测试时,我们会指定一个上限值.本文将时间区域划分为 $t=0$、$0<t<3$ 和 $t\geqslant 3$ 三个部分,即测试时,可以在这 3 个区域中任意选取 t 的参数值进行测试,以提高测试的覆盖度.

另一方面,在实际测试时,我们采用图 10 所示的测试架构.图中 Control 部分表示系统接收到输入活动后响应输出,实现与外部环境的通信;Clock 部分用来处理

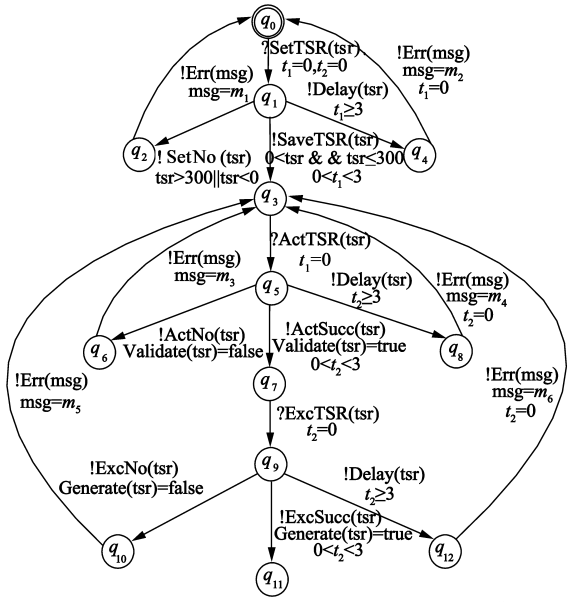

图7 TSRS临时限速命令执行功能的TSIOSTS模型

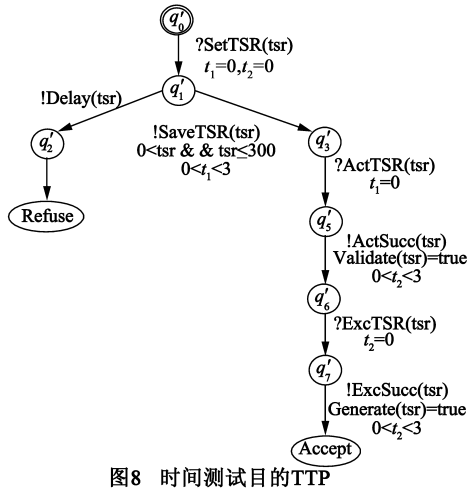

图8 时间测试目的TTP

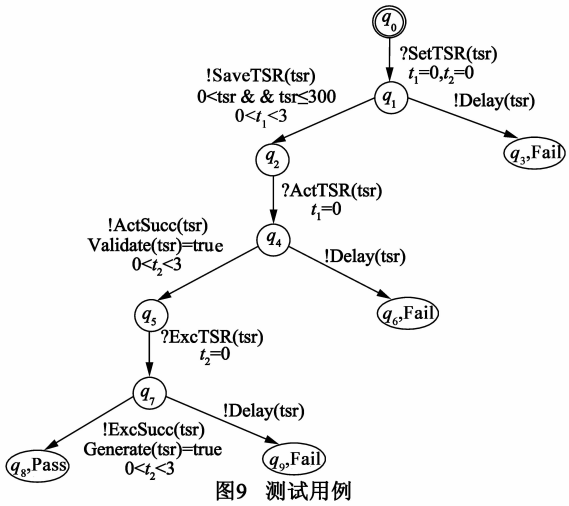

图9 测试用例

系统规定的时间变量,当前者收到一个输入活动时,检查输入是否满足该活动的时间约束,并将结果信息传给 Control 部分,同时时间重置为零.这种测试架构较好地实现时钟复位,使测试系统更精确的获取更新后的时钟信息.

6.4 测试分析

我们选用某 TSRS 设备作为被测对象,对其 TSR 功能进行测试,其结果见表 1.

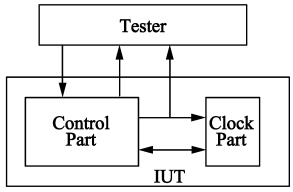

图10　测试架构

测试情况分析:TSRS 系统一致性测试过程是循序渐进的,前一阶段的测试用例首先被执行.在测试用例的覆盖率方面,我们对各项功能的测试覆盖了所有的状态变迁.考虑到外部仿真环境和物理设备的限制以及系统对并发性的支持不足,对时钟约束范围进行了适当放大,如规范约束为 $0 < t < 3$ 的,在实现中设定为 $0 < t < 4$.从测试结果可见,针对各项功能特征生成的测试用例基本上可以正常通过.但由于不存在可单独作用于其中某事件的行为发生,有时候也会导致某个状态不可测或 Fail 的测试用例出现.

表 1　TSRS 功能测试分析表

被测对象功能特征	TSR 拟定	TSR 校验	TSR 存储	TSR 执行	TSR 取消
所用测试用例数	55	45	45	40	35
通过个数	55	45	43	39	35
通过率	100%	100%	95.6%	97.5%	100%

7 结论

为了扩大形式化测试在实时系统领域的应用,本文提出了一种符号化执行的实时系统一致性测试生成方法.首先通过形式建模建立系统形式化模型,基于该模型定义了时间一致性关系以指导系统测试生成,在此基础上引入了符号执行机制,对所有可能产生的时间迹使用符号值代替具体的数值,充分体现了符号化模型的优势,避免了由于数据枚举所造成的空间爆炸.最后通过实例验证了该方法的可行性和有效性.

下一步,我们准备优化并扩展本文提出的形式化测试框架,以更好的解决实时系统中的数据与时间的约束关系,同时考虑测试生成过程中数据的建模和测试执行过程中的数据选择问题,进一步提高该测试框架的实用性.

参考文献

[1] Abdeslam E, Rachida D, Ferhat K. Timed Wp-method: testing real-time systems[J]. IEEE Transactions on Software Engineering, 2002, 28(11): 1023 – 1038.

[2] Moez K, Stavros T. Conformance testing for real-time systems [J]. Formal Methods in Systems Design, 2009, 34(3): 238 – 304.

[3] Constant C, Thierry J, Marchand H. Integration formal verification and conformance testing for reactive system [J]. IEEE Transactions on Software Engineering, 2007, 33(8): 558 – 574.

[4] 胡宇,吴建平,等. PPP 一致性测试研究[J].电子学报, 2002, 30(8): 1242 – 1245.
HU Yu, WU Jian-ping, et al. PPP conformance testing research [J]. Acta Electronica Sinica, 2002, 30(8): 1242 – 1245. (in Chinese)

[5] 陈伟,薛志云,赵琛,李明树.一种基于时间自动机的实时系统测试方法[J].软件学报, 2007, 18(1): 62 – 73.
CHEN Wei, XUE Yun-Zhi, ZHAO Chen, LI Ming-Shu. A method for testing real-time system based on timed automata [J]. Journal of Sofware, 2007, 18(1): 62 – 73.

[6] Rachel C O, Tim G. A practical and complete algorithm for testing real-time systems [A]. Proc of the 5th International Symposium on Formal Techniques in Real-Time and Fault-Tolerant Systems[C]. Berlin: Springer-Verlag Press, 1998. 251 – 261.

[7] Styp S V, Bohnenkamp H, Schmaltz J. A conformance testing relation for symbolic timed automata [A]. Lecture Notes in Computer Science[C]. Berlin: Springer-Verlag Press, 2010. 243 – 255.

[8] 李书浩,王戟,齐治昌.一种面向性质的实时系统测试方法[J].电子学报, 2005, 33(5): 827-834.
LI Shu-hao, WANG Ji, QI Zhi-chang. An approach to property-oriented testing of real-time systems[J]. Acta Electronica Sinica, 2005, 33(5): 827 – 834. (in Chinese)

[9] Adjir N, Pierre S S, Kamel M R. Time optimal real-time test case generation using prioritized time petri net[J]. Proceedings of First International Conference on Advances in System Testing and Validation Lifecycle[C]. New York: IEEE Press, 2009. 110 – 116.

[10] Krichen M, Tripakis S. State identification problems for timed automata[A]. Proceedings of the 17th International Conference on Testing of Communicating Systems [C]. Berlin: Springer-Verlag Press, 2005. 175 – 191.

[11] Khoumsi M. Complete test graph synthesis for symbolic real-time systems [J]. Electronic Notes in Theoretical Computer Science, 2005, 30(5): 79 – 100.

[12] Thierry J. Symbolic model-based test selection[J]. Electronic Notes in Theoretical Computer Science, 2009, 240(7): 167 – 184.

[13] ISO/IEC. Information Technology, Open Systems Interconnection, Conformance Testing Methodology and Framework [S]. ISO/IEC 9646, 1991.

[14] 王之梁,尹霞,景传明.一种形式化的实时协议互操作性测试方法[J].中国科学 E 辑,2008,38(10):1614 – 1635.
[15] King J C. Symbolic execution and program testing[J]. Communications of the ACM,1976,19(7):385 – 394.
[16] Gaston C, Gall P L, Rapin N, Touil A. Symbolic execution techniques for test purpose definition[A]. Lecture Notes in Computer Science[C]. Berlin: Springer-Verlag Press,2006. 1 – 18.
[17] Andrade W L, Machado P D L, Thierry J. Abstracting time and data for conformance testing of real time systems[A]. The 4th Internal Conference on Software Testing, Verification and Validation Workshop[C]. New York: IEEE Press,2011. 9 – 17.
[18] Elisabeth J, Martin W, et al. When BDDs fail: conformance testing with symbolic execution and SMT solving[A]. The 3rd International Conference on Software Testing, Verification and Validation Workshop[C]. New York: IEEE Press,2010. 479 – 488.

作者简介

万勇兵(通信作者) 男,1981 年 11 月出生于江西南昌.现为同济大学电子与信息工程学院计算机科学系博士研究生.主要研究方向为安全软件形式化建模,仿真与测试.
E-mail: wybingsh@163.com

徐中伟 男,1964 年 6 月出生于江苏无锡.现为同济大学电子与信息工程学院信息与通信工程系教授、博士生导师.从事安全软件验证、测试与评估方面的研究工作.
E-mail: xuzhongweish@163.com