软件学报 ISSN 1000-9825, CODEN RUXUEW
Journal of Software,2014,25(3):489−505 [doi: 10.13328/j.cnki.jos.004412]

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

安全苛刻系统自动化测试的形式化语义模型*

吕江花, 马世龙, 李先军, 高世伟

(软件开发环境国家重点实验室(北京航空航天大学 计算机学院),北京 100191)

通讯作者: 吕江花, E-mail: jhlv@nlsde.buaa.edu.cn

摘 要: 安全苛刻系统的可信性需求典型而迫切,其可信性评估和验证具有测试依赖性.安全苛刻系统一般是复杂系统,手工测试实际上不可行,发展自动化测试手段是必然趋势.针对安全苛刻系统测试过程自动化中存在的高阶协同、实时和时序性,以 Ambient 演算、CCS 演算、论域理论等为基础,给出测试过程的高阶协同定义,建立一种层次化演算模型,为测试过程提供一种信息化和自动化手段.模型通过对被测产品、测试设备与测试任务的抽象与组织,给出安全苛刻系统测试过程自动化的工作模式.最后,通过扩展标记转换系统定义,给出高阶协同行为的收敛性和正确性的证明,论证了模型的可计算性,验证了安全苛刻系统测试的可自动化.模型已应用于航天器的自动化测试中,并成为航天器测试行为的日常工作规范.

关键词: 安全苛刻系统;测试;自动化测试;设备协同;高阶演算;标记转换系统;实时

中图法分类号: TP311 **文献标识码**: A

中文引用格式: 吕江花,马世龙,李先军,高世伟.安全苛刻系统自动化测试的形式化语义模型.软件学报,2014,25(3):489−505. http://www.jos.org.cn/1000-9825/4412.htm

英文引用格式: Lü JH, Ma SL, Li XJ, Gao SW. Formal semantics model for automatic test of safety critical systems. Ruan Jian Xue Bao/Journal of Software, 2014,25(3):489−505 (in Chinese). http://www.jos.org.cn/1000-9825/4412.htm

Formal Semantics Model for Automatic Test of Safety Critical Systems

LÜ Jiang-Hua, MA Shi-Long, LI Xian-Jun, GAO Shi-Wei

(State Key Laboratory of Software Development Environment (School of Computer Science and Engineering, BeiHang University), Beijing 100191, China)

Corresponding author: LÜ Jiang-Hua, E-mail: jhlv@nlsde.buaa.edu.cn

Abstract: Safety critical systems (SCS) are very typical and crucial in trustworthiness study, and their evaluation and verification are test-dependent. Since SCS are usually complex and dramatically huge, manual test of SCS is unfeasible in practice, which makes automatic test approaches for SCS an important trend. This paper studies the characteristics of high order collaboration, real time and temporaries in SCS testing, and based on the domain theory, ambient and CCS calculus, it defines a formal semantics for automatic test of SCS, called AutTMSCS, which describes behaviors in SCS testing. Testing tasks, test equipments and products under test are abstracted and architected in three layers, and a procedure of automatic testing is provided in the model. Based on extended LTS, the convergency and correctness of the model are proved to demonstrate the computability of the model, which indicates that the testing process of SCS can be automated. The model had been practically applied to automatic test of spacecrafts, and the system developed based on the model had become the working platform of spacecrafts test service in daily usage.

Key words: safety critical systems (SCS); test; automatic test; equipment collaboration; high order; LTS; real time

随着计算系统的功能、结构、规模日趋庞大而复杂,系统是否可信已成为当前非常关注的问题[1].可信是指系统的行为及其结果是可预期的、行为状态是可监测的、行为结果是可评估的、行为异常是可控的[2,3].安全

* 基金项目: 国家自然科学基金(61300007, 61003016); 软件开发环境国家重点实验室开放基金(SKLSDE-2012ZX−28, SKLSDE-2013ZX-11)

收稿时间: 2012-09-13; 定稿时间: 2013-04-09

苛刻系统(safety critical system,简称 SCS)对可信性的需求典型而迫切,其系统功能一旦失效将引起财产、生命的重大损失以及环境可能遭到严重破坏.它广泛存在于航空航天、交通运输等领域.近年来,这类重大事故屡见不鲜,因而研究评估和验证安全苛刻系统的可信性具有现实意义.安全苛刻系统的可信性一般是通过测试来支持,因而获取准确可靠的测试结果数据至关重要.然而安全苛刻系统一般是复杂系统,人工测试方式一方面给测试带来不准确性,另一方面,很多测试也无法用人工方式进行,因此,测试的信息化和自动化已成为安全苛刻系统测试发展的主要趋势.自动化测试包括测试用例自动生成和测试过程的自动化,本文以测试过程的自动化为主要内容展开研究.关于具体的测试方法、技术和理论等则不在本文的讨论范畴中.

测试语言是实现测试过程信息化和自动化的根本途径,国内外针对安全苛刻系统测试分别设计和开发了不同的测试语言或测试脚本语言.如美国宇航局 NASA 的 GOAL[4]、欧洲宇航局 ESA 的 ETOL[5]、ISYS 公司发布的 STOL[6]、国际标准化组织 IEEE 的 ATLAS[7]、以色列 Technion 大学的 T++[8]、德国 Tech S. A. T 公司的航天系统测试脚本语言 ADS2[9].国内也展开广泛的研究,中国科学院光电研究院的 ATLMIC[10]、中国科学院空间科学与应用研究中心的卫星测试操作语言[11]、同济大学用于高速铁路仿真测试的 SED_SCS_STL[12]、北京航空航天大学用于航天器测试的 CATOL[13]等.这些有代表性的测试语言在自动化测试中发挥了重要作用.

安全苛刻系统的测试是由测试人员借助测试设备完成对被测产品的测试.根据被测产品的需求描述设计测试任务,驱动测试设备对被测产品进行测试,通过判读获取的测试数据完成测试,如图 1 所示.与以往相比,当前安全苛刻系统的测试面向新的测试需求,被测产品批量化生产网络并行测试对测试提出了更高的测试需求和更安全的可靠要求.同时,由于安全苛刻系统的特殊性和复杂性,其现有测试模型是在当时的技术环境下开发的,自动化程度低,硬件设备的耦合度高,开放性差,分系统之间融合度低,升级困难,很难满足新的测试需求.因此,由于安全苛刻系统领域的特殊性,需要深入分析和研究安全苛刻系统自动化测试过程中的相关理论、方法和技术,支持安全苛刻系统的自动化测试.

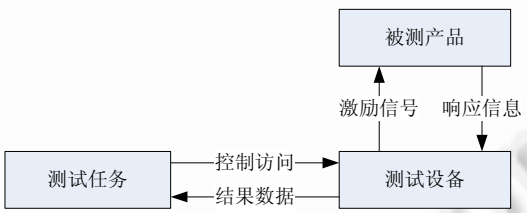


Fig.1 Testing process

图 1 测试过程

安全苛刻系统的自动化测试是通过测试任务、测试设备和被测产品三端交互协同完成的测试过程,包括这三端内部以及之间的交互协同.由于被测产品和测试设备的物理实体都是仪器和仪表等硬件设备,而这些硬件设备本身都是复杂系统,属于高阶对象,并且这些设备之间的交互需要通过传送设备操作指令(高阶数据)完成.另一方面,测试任务是通过测试设备访问被测产品,其中交互内容也主要是关于设备操作指令的高阶数据,因此,在安全苛刻系统自动化测试的协同过程中就表现出了交互内容和交互过程的高阶性,即协同对象和协同行为均具有高阶性.

为掌握实时的信息数据,安全苛刻系统一般具有实时性,测试中系统各部件之间不仅通过交互描述与功能相关的信息,而且还需要表达出时间方面的控制特征,并且由于系统部件之间的实时影响,测试任务上的控制结构与传统程序设计语言的控制结构的执行机制不同.

在安全苛刻系统测试中,被测产品状态参数在不同时刻具有不同值,以表征其不同的状态.不仅如此,被测产品状态参数的这些不同的取值在不同的时刻具有不同的参照关系,用于验证功能部件是否满足给定的功能.因此,安全苛刻系统的自动化测试过程具有时序性.

因此,安全苛刻系统的自动化测试过程呈现出上述的高阶协同性、实时性、时序性等特点,并且这些特性

纠结在一起贯穿在整个自动化测试过程中,从而使得对安全苛刻系统测试过程中的测试行为难以进行准确地刻画与分析,造成测试过程自动化困难.

近年来,作者参与了安全苛刻系统(包括卫星、飞船、大型武器装备任务系统)的自动化测试和评估验证等方面的大量实际项目工作,积累了一定的实际工作经验.针对上述问题,采用形式化方法为安全苛刻系统的自动化测试建立形式化模型,精确描述和定义安全苛刻系统的自动化测试过程.以此为基础,提出用于安全苛刻系统自动化测试的测试语言,设计实现的相关自动化测试系统和平台已运行于实际应用中,取得了很好的效果.

1 相关工作

不同的形式化方法在描述能力、应用领域等方面有不同的侧重.目前,已有的形式化模型分别刻画了高阶、实时、时序等特性中的部分特性.如:刻画高阶性的典型工作有高阶 CCS 演算[14]、高阶π演算[15]、高阶并发通信模型[16]、Ambient 演算[17]等;刻画实时性的有时间 CSP 模型[18]、时间自动机[19]、时段演算[20]等;刻画时序性的有分支时态逻辑 CTL[21]、线性时态逻辑 LTL[22]、μ演算逻辑[23]等;刻画高阶和实时性的高阶实时演算模型时间π演算模型[24,25],高阶多型π演算 THOπ-calculus[26]、高阶时段演算[27];刻画实时和时序性的实时时序模型 E-LOTOS[28,29]和 XYZ/E[30]等.本文主要讨论高阶协同性问题,故相关工作以高阶性模型展开论述.

在描述高阶性的模型中,由于并发模型中比较经典的模型有 CCS 模型[31]和π演算模型[32],目前的高阶演算模型主要是在二者的基础上扩展高阶定义,实现模型的高阶性.高阶 CCS 演算模型中,典型的有 Thomsen 的高阶通信系统演算 CHOCS[33]和 PlainCHOCS[34],二者均通过对 CCS 进行高阶通信扩展,即进程本身可以传递实现高阶定义,且这两个模型中没有一阶通信.CHOCS 和 PlainCHOCS 之间的不同在于:CHOCS 使用动态绑定,而 PlainCHOCS 使用静态绑定.

Sangiorgi 在 Thomsen 的工作基础上对π演算进行扩展,其中任意阶的进程都可以被传送.它将高阶与一阶区别开来的同时也带来了一些复杂性,该演算模型称为高阶π演算[15].

李未的并发演算 CC[16]是一种高阶并发通信系统的数学模型,把λ演算作为子理论并包含一阶通信系统演算 CCS、移动进程演算 CMP、高阶通信系统演算 CHOCS 的主要特征.在 CC 中,通信端口可为任意表达式,并且进程和通信端口都可以作为通信中传递的一阶对象.从而使 CC 不仅可以描述高阶通信行为而且还可以刻画通信网络的动态自修改行为.

Ambient 演算[17]作为一种描述移动并发的模型,其高阶性体现在通过移动动作实现进程的移动.在 Ambient 演算中,分布计算节点被描述为一个叫做 ambient 的封闭结构,通过在 ambient 中定义的 3 种基本动作 in,out, open 实现 ambient 的移动,描述移动计算过程.Ambient 演算是第一个成功地描述广域计算和移动计算的异步、高阶、并发演算,可看作是具有层次性、位置可移动和资源访问控制的π演算.

为了描述实时性,现有工作主要是在高阶模型上扩展实时性,以刻画系统行为与时间的关系.如:Degano 等人[24]对π演算的标号迁移系统 LTS 中的标号进行时间扩展,增加动作的完成时间信息和时间代价函数,定义了一种非时间良构的紧急同步的π演算时间操作语义.在该模型中,时间因素实际上被作为系统的一种特征属性信息而进行刻画;Berger[25]在π演算的基本语法中扩展计时器结构,用以描述时间与动作的关联关系.以整数表示时间,用时间步骤函数静态离散描述动作的执行过程.西北工业大学尤涛等人[26]以时态逻辑为基础,提出一种有时间特性的高阶多型π演算(THOπ-calculus),其细化定义时间种类,实现系统行为与时间的关联.该模型主要应用于构件式实时软件的建模与演化分析;中国科学院软件研究所詹乃军[27]用时段演算来刻画程序的实时行为.用高阶的时间函数来表示程序变量,建立高阶时段演算分析程序的实时性质.

在安全苛刻系统中,由于被测产品一般为复杂系统,在进行测试时需分级进行测试.为了支持测试模型的通用性,定义了一种独立于测试设备和被测产品的基本测试单元,称为测试原子.测试原子抽取和封装了基本的测试方法,在测试过程中具有可重用性和实时性的特点.测试任务即由这些测试原子组成.同时,被测产品和测试设备自身也是独立系统,这些系统进行交互完成测试的同时,内部既要根据交互信息,选择执行不同的操作来执行,又同时运行着其内部常规程序,并维护自身的状态;而且这些系统对外的接口具有不同的协议约束,并分别

隶属于不同的应用域内,例如加电设备、计量设备等由相关的设备组成的设备应用系统.因此,无论是测试任务还是被测产品、测试设备,采用传统进程形式来描述将极其繁琐并给实现带来极大的不便.

现有高阶演算模型中,高阶性主要体现在进程自身可以当做一阶值进行传递.由于高阶进程可以被传递,其副作用也使进程间的交互关系发生变化,带来进程执行次序和进程间交互关系的不确定性,这些不确定性难以通过静态方法进行分析和验证.而在安全苛刻系统测试中,由于安全可靠至关重要,这些不确定因素会给测试带来安全隐患.因此,上述基于进程的分布式模型用于描述设备协同过程时存在着不足,但 Ambient 演算作为一种描述移动并发的模型,其 ambient 封装结构值得借鉴.

2 安全苛刻系统自动化测试中的高阶协同性

协同是多个资源实体间通过信息交互协调一致完成某个功能或目标的过程.若定义 E 为资源实体域,D 为资源实体上的交互消息域,R 为系统输出(运行结果)域,则一个协同系统可以定义为 $CoSys(E,D,R)\xrightarrow{D}R$.一般协同系统中,D 为一阶论域,即交互消息为一阶数据.当交互消息域为高阶域时,我们将这样的协同系统称为高阶协同系统.

在安全苛刻系统的测试中,被测产品和测试设备操作指令作为高阶对象操作密集出现,是测试系统测试行为交互的主要内容.而在安全苛刻系统测试过程中,协同涉及测试任务、测试设备以及被测产品三端之间的协同,为支持系统的开放性,测试设备可以动态加入和退出,同时还要支持多种被测产品的测试,因此,协同中的这些高阶指令不能简单地处理为一阶数据交互,需要给出这些设备指令的高阶封装规范和交互协同机制.

在安全苛刻系统测试过程中,协同涉及测试任务、测试设备以及被测产品三端之间的协同,这些协同行为既有时间上的协同,又有控制上的协同.为清晰协同关系,支持高阶协同系统上的相关性质分析,本文采用层次架构,将自动化测试系统分为测试任务端、测试设备端、被测产品端三端,通过对这三端和端之间交互定义,刻画测试过程中不同的协同关系.采用这种层次化架构,系统很容易扩展到对多被测产品的批产化网络测试.自动化模型层次架构如图 2 所示.其中:测试任务端(TWPart)实现测试任务的组织和设计,实现对被测产品的自动化测试;测试设备端(TEQPTPart)支持上层测试任务对测试设备的访问和交互,为上层测试任务提供透明访问测试设备功能和屏蔽不同类设备的异构性,给出到具体测试设备的访问路由.被测产品端(PRODPart)完成测试设备到被测产品的交互,根据测试设备发来的激励、指令和请求,完成对被测产品状态参数的改变和获取.各端形式化定义如下:

定义 1(测试任务端). 测试任务端 $TWPart=(T\times TWEnv\times Timer)$.其中,域 T 表示测试任务端测试任务,域 $TWEnv$ 表示测试任务端测试任务的执行环境,域 $Timer$ 为实时环境时钟域.

定义 2(测试设备端). 测试设备端 $TEQPTPart=(TEQPT\times HChannel\times TInstructSet)$.其中,域 $TEQPT$ 为测试设备,域 $HChannel$ 为与测试任务端的交互通道,$TEInstructSet$ 为测试设备指令集域.

定义 3(被测产品端). 被测产品端 $PRODPart=(PROD\times PChannel\times PInstructSet)$.其中,域 $PROD$ 表示被测产品,域 $PChannel$ 是与被测产品的交互通道,域 $PInstructSet$ 表示被测产品的指令集域.

交互通道用于定义交互信息的形式,分为指令通道和数据通道,分别用于指令交互和数据交互.高阶的指令信息和一阶的数据信息形式化定义如下,是交互协议的形式表示:

定义 4(交互指令). 交互指令域 $GuardV=G\{Gd,Params\}$,其中,

- $G\{\}$为标志符,用于标识{}内的高阶指令;
- 域 $Gd=(DevKind\times Op)$是测试指令的形式表示,域 $DevKind$ 表示测试设备;域 Op 为具体指令的形式表示,是测试设备操作 $DevOp$ 或被测产品操作 $ProdOp$;
- 域 $Params$ 为操作参数域 $Param$ 上的序列.

定义 5(交互数据). 交互信息域 $DataV=PParam\times PVal$.由被测产品状态参数名域 $PParam$ 和参数取值域 $PVal$ 组成.

定义 6(被测产品操作). 被测产品操作 $ProdOp=PVal^*\times ProdC\rightarrow ProdC$ 为一个函数域,其中,域 $PVal^*$ 为被测产

品状态参数取值域 *PVal* 上的序列;函数域 *ProdC* 为被测产品的状态,*ProdC=PParam→PVal*,是关于被测产品状态参数名域到状态参数取值域的映射.

定义 7(测试设备操作). 测试设备操作域 *DevOp* 为函数域,$DevOp=DVal^{*}\times DevC\rightarrow DevC\times(PParam\times PVal)^{*}$,其中,域 $DVal^{*}$为测试设备操作参数取值域 *DVal* 上的序列;函数域 *DevC* 为测试设备的状态,*DevC=DParam→ DVal*,为测试设备操作参数名域 *DParam* 到值域 *DVal* 的映射.

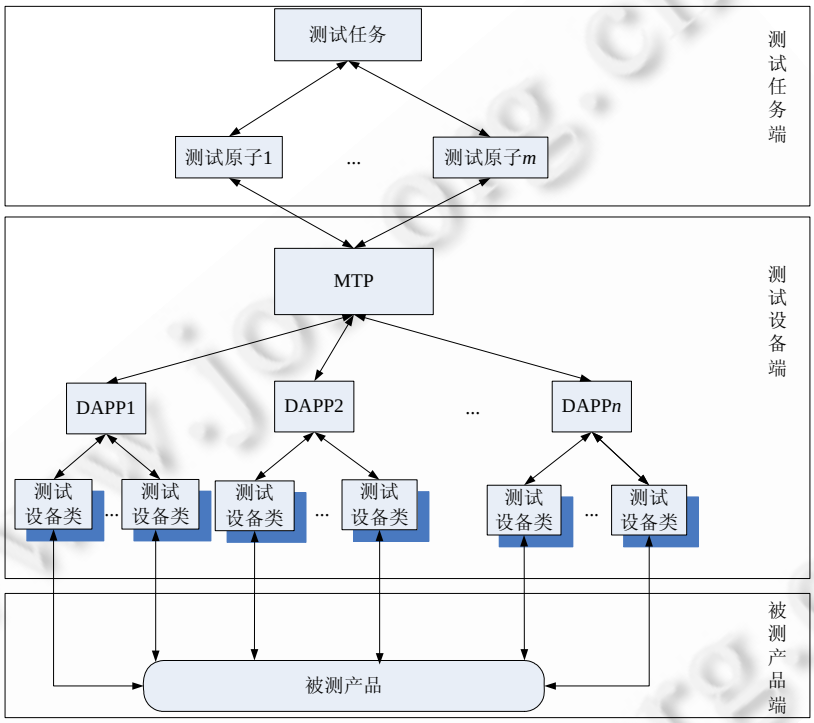


Fig.2 Layered architecture of automatic test model

图 2 自动化测试模型层次架构图

将被测产品参数取值定义为被测产品的状态,被测产品的操作改变的是被测产品的状态.这里将被测产品的操作定义为高阶函数形式,即给定操作的参数值和被测产品的当前状态,返回的是操作作用被测产品之后的新状态.通过对被测产品操作的功能抽象来表示被测产品操作.由于测试设备操作改变的不仅是测试设备自身状态,还将产生关于被测产品的激励信号,以触发被测产品改变自身状态,激励信号可定义为对被测产品状态中某些相关参数改变取值,定义为$(PParam\times PVal)^{*}$.

对给定的硬件设备,也就限定了该硬件设备上的所有操作,即设备和其操作集合之间存在着映射关系.下面给出了设备操作集域定义,表示测试设备、被测产品分别与其操作集合之间的映射关系.

定义 8(设备类操作集). 设备类操作集域 *OpEnv* 为函数域,*OpEnv=OpKind→OpS*,是设备类 *OpKind* 到设备操作集合 *OpS* 的映射.函数域 *OpS* 为设备操作名标识 *OpId* 到域 *Op* 的映射,即,*OpS=OpId→DevOp*.

这样,在实际应用中,对给定的具体测试设备或被测产品,均可根据上述定义,给出该应用中被测产品和具体测试设备操作的高阶形式化定义.

若定义 *SCSAutTM* 为安全苛刻系统的自动化测试系统,根据上述定义,*SCSAutTM* 为高阶协同系统:

$$SCSAutTM(\{TWPart,TEQPTPart,PRODPart\},GuardV+DataV,Result):$$

$$TWPart\times TEQPTPart\times PRODPart \xrightarrow{GuardV+DataV} Result.$$

3 安全苛刻系统自动化测试中的实时性

在安全苛刻系统自动化测试中,由于测试语言中基本元素为测试原子,测试原子的执行受时间约束,用以表明其内部测试行为执行的有效性.同时,测试原子上的复合操作也间接受时间影响.在一般语言系统中,系统行为改变系统状态,行为本身并没有结果,或者是结果无意义,主要用于系统调试.但安全苛刻系统自动化测试系统中,系统行为不仅改变系统自身状态,同时其本身的结果值也用于反映在时间约束下行为的有效性,即系统改变的状态是否正确.因此在测试模型中,测试原子及其上的复合操作均具有实时性.

若定义测试原子为 $Atom=Precond\rightarrow AName(Time\text{-}Restriction)[TestP]$,其中,$Time\text{-}Restriction$ 即为测试原子的执行时间约束.测试原子上的复合操作如串行、并行、选择和循环等也受时间约束影响,称为时间约束串行、时间约束并行、时间约束选择和时间约束循环.对于测试原子,这些时间约束复合操作的含义分别:

- 时间约束串行"$;_t$":测试原子之间时间约束串行指的是,只有当前一个测试原子的执行满足时间约束,且结果为真时,后续的测试原子才会被继续执行;否则中止,直接返回结果为假;
- 时间约束并行"$||_t$":当且仅当两个测试原子的执行满足时间约束且结果均为真时,并行运行的结果才为真;否则为假;
- 时间约束选择 if_t:当条件布尔表达式为真,其运行结果为第 1 个测试原子执行结果;否则为第 2 个的执行结果;
- 时间约束循环"$while_t$":执行不仅依赖循环中的布尔表达式结果,而且还与循环体中的测试原子有关,当进入循环后,只有当循环体中的测试原子执行满足时间约束且结果为真时,才会返回到循环入口,继续循环;否则,结束循环.

从另一角度看,当测试过程中出现错误时,测试程序会终止后续动作的执行,并防止错误的测试操作对被测产品和测试设备的损坏,从而也保证了被测产品测试的安全性.

为刻画上述测试原子及其上复合操作的语义,定义时间约束函数监测测试原子的执行过程.若测试原子的执行过程可用函数描述,则时间约束函数一高阶函数形式.设测试原子的执行过程函数 $\circlearrowleft^{atom}:Atom\times Context\times Timer\rightarrow Result\times Context,Timer$ 为时钟,$Context$ 为测试原子的上下文执行环境.时间约束函数 TR 定义为:$\circlearrowleft^{atom}\times StartTime\times Time\text{-}Restriction\times Timer\rightarrow Result,Time\text{-}Restriction$ 为测试原子的时间约束,$StartTime$ 为测试原子开始执行时间.对 $f\in\circlearrowleft^{atom},st\in Starttime,tr\in Time\text{-}Restriction,\tau\in Timer$,$TR(f,st,tr,\tau)$具体定义为:

(1) 若在 τ时钟下存在时刻 $t,st\leqslant t<st+tr$,f 收敛.即 $f_t=\ldots=f_{st+tr}=r$,则 $TR(f,st,tr,\tau)=r$;

(2) 否则,$TR(f,st,tr,\tau)$=false.

这样,通过 TR 函数时间约束测试原子的执行过程.

4 安全苛刻系统自动化测试模型 SCSAutTM**

4.1 被测产品PRODPart定义

被测产品端的主要功能是完成与测试设备端的交互协同.测试任务借助测试设备对被测产品进行测试,可以通过测试设备操作间接改变被测产品的状态信息.如卫星测试中,电源设备的加电操作产生对卫星的激励,使卫星通电.

定义 9(被测产品). 自动化测试模型中,被测产品端 $PRODPart$ 域中的被测产品 $PROD$ 域定义为 $PROD=(ACK\times ProcC\times ProdOpS)$,其中,$ACK$ 为被测产品的响应动作,其他定义同上.

被测产品与测试设备的交互将通过 $PChannel$ 完成.被测产品端的行为通过改变 $PChannel$ 和自身状态参数实现与测试设备端的交互协同.被测产品端交互通道 $PChannel=PSendGS\times PRetnVS$.其中,指令通道 $PSendGS=$

** 说明:(1) 在模型定义中,未给出的域定义则为常见域或根据应用领域给出具体定义;(2) 在模型定义的域函数定义部分中,未给出部分的定义则为$\perp$;(3) 对元组域 $A\in(T_1\times\ldots\times T_n)$,记 $A^1\in T_1,\ldots,A^n\in T_n$.

(*AName*×*GuardV*)*,数据通道 *PRetnVS*=(*AName*×*DataV*×*Label*),*AName* 为测试任务端的测试原子名,标志位 *Label* 用于标识一个交互动作是否已经完成.

被测产品端与测试设备端的交互过程为:

(1) 若接到关于自身的操作指令请求,则向测试设备端发送回令并执行具体操作;

(2) 若是来自测试设备的激励信号,则根据激励信号更新相应参数值,改变被测产品状态参数.由于激励是测试设备操作产生的,其回令已经由测试设备端收到测试设备操作时返回,故此时被测产品无需再返回回令;

(3) 若是被测产品状态参数请求,则将参数值传送给测试设备端.

被测产品端上的形式语义可表示为域 *PRODPart* 上语义映射函数 $\circlearrowleft_{M}^{Prodp}$,根据被测产品交互过程描述,其具体定义如图 3 所示.

$\circlearrowleft_{M}^{Prodp}$:*PROD*×*PChannel*→*ProcC*×*PChannel*
For (*ack*,*γ*,*I*)∈*PROD*, *ξ*∈*PChannel*
1) If $\xi^2(dk)=(a,dk,(pn,pv),0)$, then $\circlearrowleft_{M}^{Prodp}$【(*ack*,*γ*,*I*)】*ξ*=(*γ*,*ξ'*).
Where, *pv*=*γ*(*pn*), $\xi'^1(dk)=Update\ PRetnVS(a,dk,(pn,pv),1)$.
2) If $\xi^2(dk)=(a,dk,f,as))$, then $\circlearrowleft_{M}^{Prodp}$【(*ack*,*γ*,*I*)】*ξ*=(*γ'*,*ξ'*).
Where, $\gamma'=\gamma' f\ as\ \gamma$, $\xi'^2(dk)=Del\ PSendGS(a,dk,f,as,1)$.
3) If $\xi^2(dk)=PSendGS(a,G\{dk,-,psvl\})$, then $\circlearrowleft_{M}^{Prodp}$【(*ack*,*γ*,*I*)】*ξ*=(*γ*,*ξ'*).
Where, $\gamma'=\gamma\{psvl\}$, $\xi'^2(dk)=Del\ PSendGS(a,G\{dk,-,psvl\})$

Fig.3 Semantics of the layer of products under test

图 3 被测产品端语义

4.2 测试设备端TEQPTPart定义

测试设备端的主要功能是通过与测试任务端、被测产品端的交互支持测试任务完成对被测产品的测试.测试设备端将分别通过测试设备层的交互通道 *HChannel* 和被测产品层 *PChannel* 实现与测试任务端、被测产品的交互.同时,测试设备端还要完成交互中关于指令操作的静态解析功能,即检查交互信息是否符合协议,支持测试任务端和具体测试设备的交互.

测试设备端的测试设备由对被测产品测试中使用的各种测试仪器、测试系统等测试设备资源组成.测试设备的类型、数目众多,测试语言要独立于具体测试设备,需要与这些具体测试设备解耦合.为此,在测试语言中对测试设备进行分层:上层称为测试设备通信中间件(MTP),面向应用,旨在屏蔽前端测试应用功能级的复杂性,为测试任务提供透明的测试设备访问功能,其功能相当于应用级测试设备网关;其下层称为设备应用层(DAPP),屏蔽不同类设备的异构性,其功能相当于设备级测试设备网关;最底层为测试设备类层 *DK*,*DK* 管理具体的一类测试设备,给出具体测试设备的访问路由.通过这种分层架构,屏蔽底层测试设备的异构性,建立测试设备抽象描述、协同交互与统一访问机制.测试设备端架构如图 2 所示.

定义 10(**测试设备**). 自动化测试模型中,测试设备端 *TEQPTPart* 域中的测试设备 *TEQPT* 域定义为 *TEQPT*=(*MTP*×*DAPP*×*DK*×*BChannel*×*DChannel*),其中,MTP(middleware of test process)称为应用级测试设备网关,域 DAPP(device application)称为设备级测试设备网关,通道 *BChannel*=*BSendGS*×*BRetnVS* 为 MTP 与 DAPP 两层之间的交互通道,分为指令通道 *BSendGS*=(*AName*×*GuardV*×*Rtn*)*和数据通道 *BRetnVS*=(*AName*×*DataV*×*Label*)*表示来自测试原子的数据信息和指令信息.通道 *DChannel*=*DSendGS*×*DRetnVS* 为 DAPP 与 DK 两层之间的交互通道,分为指令通道 *DSendGS*=(*AName*×*GuardV*×*Rtn*)*和数据通道 *DRetnVS*=(*AName*×*DataV*×*Label*)*表示来自测试原子的数据信息和指令信息.其他定义同上.

测试设备层与测试任务层的交互通道 *HChannel*=*ASendGS*×*ARetnVS*,分为指令通道域 *ASendGS*=(*AName*×*GuardV*×*Rtn*×*Label*)*和数据通道域 *ARetnVS*=(*AName*× *DataV*×*Label*)*.返回值域 *Rtn*=*Bool*+(*Bool*×*String*),其值或者为 true,表示指令成功;或者为(false,errocode),表示指令执行失败,返回 false 和失败原因 errocode;标志位 *Label* 含义同上.

测试设备端的形式化语义表示为 *TEQPTPart* 域上的映射函数 $\circ\mathcal{M}^{Tep}$,定义为

$$\circ\mathcal{M}^{Tep}:TEQPTPart\times PChannel\times PInstructSet\rightarrow HChannel\times PChannel.$$

即,$\circ\mathcal{M}^{Tep}:TEQPT\times HChannel\times TInstructSet\times PChannel\times PInstructSet\rightarrow HChannel\times PChannel.$

这样,对于 $Mtp\in MTP,Dapp\in DAPP,Dk\in DK,\zeta\in BChannel,\eta\in HChannel,\varsigma\in DChannel,\rho^T\in TInstructSet,\xi\in PChannel,\rho^P\in PInstructSet$,测试设备端的语义函数 $\circ\mathcal{M}^{Tep}$ 具体定义为

$$\circ\mathcal{M}^{Tep}【(Mtp,Dapp,Dk,\zeta,\varsigma)】(\eta,\rho^T,\xi,\rho^P)=\circ\mathcal{M}^{Mtp}【Mtp】(\eta,\zeta)|\circ\mathcal{M}^{Dapp}【Dapp】(\zeta,\varsigma,\rho^T,\rho^P)|\circ\mathcal{M}^{Dk}【Dk】(\varsigma,\xi).$$

其中,函数 $\circ\mathcal{M}^{Mtp}$ 为应用级测试设备网关语义函数,$\circ\mathcal{M}^{Dapp}$ 为设备级测试设备网关语义函数,"|"表示并行.

若 $\circ\mathcal{M}^{Mtp}【Mtp】(\eta,\zeta)=(\eta',\zeta')$,$\circ\mathcal{M}^{Dapp}【Dapp】(\zeta,\varsigma,\rho^T,\rho^P)=(\zeta'',\varsigma')$,$\circ\mathcal{M}^{Dk}【Dk】(\varsigma,\xi)=(\varsigma',\xi')$

则 $\circ\mathcal{M}^{Tep}【(Mtp,Dapp,Dk,\zeta,\varsigma)】(\eta,\rho^T,\xi,\rho^P)=(\eta',\xi')$.

$\circ\mathcal{M}^{Mtp}$,$\circ\mathcal{M}^{Dapp}$ 和 $\circ\mathcal{M}^{Dk}$ 的具体定义见第 4.2.1 节~第 4.2.3 节.

$\circ\mathcal{M}^{M}:Mtp\times HChannel\times BChannel\rightarrow HChannel\times BChannel$

For $\eta\in HChannel$ and $\zeta\in BChannel$

$$1)\ \circ\mathcal{M}^{M}【m[a!(dk,dps)]】(\eta,\zeta)=\begin{cases}\Phi(\eta,\zeta') \text{ IF } (dk,dps,0)\in\eta^1(m,a) \text{ WHERE}\\ \qquad dapp=Analy(dk),\zeta''^1(m,dapp)=AddBRetnVS(a,dk,dps,0)\\ \qquad \Phi(\eta,\zeta')=\begin{cases}(\eta',\zeta'') \text{ IF } (a,dk,dps,1)\in\zeta''^1(m,dapp) \text{ WHERE}\\ \qquad\qquad \eta'^1(m,a)=UpdateARetnVS(a,dk,dps,1)\\ \qquad\qquad \zeta''^1(m,dapp)=DelBRetnVS(a,dk,dps,1)\\ \phi(\eta,\zeta) \text{ OTHERS}\end{cases}\\ (\eta,\zeta) \text{ OTHERS}\end{cases}$$

$$2)\ \circ\mathcal{M}^{M}【m[a?G\{dk,dpop,as\}]】(\eta,\zeta)=\begin{cases}\Psi(\eta,\zeta') \text{ IF } (dk,dpop,as,0)\in\eta^2(m,a) \text{ WHERE}\\ \qquad dapp=Analy(dk),\zeta''^2(m,dapp)=AddBRetnVS(a,dk,dpop,as,0)\\ \qquad \Psi(\eta,\zeta')=\begin{cases}(\eta',\zeta'') \text{ IF } (a,dk,dpop,as,1)\in\zeta''^2(m,dapp) \text{ WHERE}\\ \qquad\qquad \eta'^2(m,a)=UpdateASendGS(dk,dpop,as,(rtn,1))\\ \qquad\qquad \zeta''^1(m,dapp)=DelBRetnVS(a,dk,dps,1)\\ \phi(\eta,\zeta) \text{ OTHERS}\end{cases}\\ (\eta,\zeta) \text{ OTHERS}\end{cases}$$

$$3)\ \circ\mathcal{M}^{M}【m[Nil]】(\eta,\zeta)=(\eta,\zeta)$$

$\circ\mathcal{M}^{Mtp}:MTP\times HChannel\times BChannel\rightarrow HChannel\times BChannel$

$1)\ \circ\mathcal{M}^{Mtp}【Nil】(\eta,\zeta)=(\eta,\zeta)$

$2)\ \circ\mathcal{M}^{Mtp}【mtp||mtps】(\eta,\zeta)=\circ\mathcal{M}^{M}【mtp】(\varepsilon,\eta,\zeta)|\circ\mathcal{M}^{Mtp}【mtps】(\varepsilon,\eta,\zeta)$

Fig.4 Semantics of MTP of test equipment layer

图 4 测试设备端 MTP 层语义

4.2.1 应用级测试设备网关 MTP

MTP 端是由若干 MTP 进程以并行方式组成.MTP 端进程的主要功能是对 *HChannel* 中的指令和参数请求传递给测试设备的通道 *BChannel* 中,并将测试设备的返回结果通过 *BChannel* 回传给 *HChannel*.由于 MTP 端的时间约束是由测试任务端的测试原子来完成,即 MTP 端的实时性由测试任务端的实时性刻画,故此处 MTP 进程并行的执行同传统程序设计语言中的并行语义.若定义 *Mtp* 为 MTP 端进程,由于 MTP 端进程的功能是响应来自测试任务端的指令或参数请求,故进程 *Mtp* 和 *Mtps* 语法定义为:

Mtps::=*Nil*|*Mtp*||*Mtps*

Mtp::=*MName*[*MtP*]

MtP::=*Nil*|*AName*?*GuardV*|*AName*!*DataV*

其中,*Nil* 表示空动作;*a*?*gv* 和 *a*!*dv* 分别表示从测试任务端的测试原子 *a* 处获取指令 *gv* 和向测试原子 *a* 发送参数 *dv* 动作.

当 MTP 接收到来自测试任务层的数据请求时,通过对请求解析,将该请求发到指定的 DAPP 中并获取返回

值返回到测试任务层;当接收到来自测试任务层的指令请求时,也同样解析该指令,判断发送给哪类 DAPP,将该指令发送给该类 DAPP,并将获取的回令返回给测试任务层.若定义 $\circlearrowleft_{\mathcal{M}}^{M}$ 函数为 MTP 端进程的语义函数,$\circlearrowleft_{\mathcal{M}}^{M}$ 的定义如图 4 所示.其中,函数 *Analy* 为 MTP 端的解析函数,根据给定的测试设备类 DevKind 和测试设备类与设备级测试设备网关映射表 *DD*,找到该设备类所从属的设备级测试设备网关.*DD*=*DevKind*→*DAPPName*,其中,*DAPPName* 为设备级测试设备网关名域,*Analy*:*DevKind*×*DD*→*DAPPName*.

4.2.2 设备级测试设备网关 DAPP

设备级测试设备网关主要是检查和响应来自 MTP 端的指令操作或参数请求,即对 MTP 发来的请求进行解析,判断该请求是发送给哪类测试设备,并对其中交互信息进行协议检查,判断是否符合协议定义.定义函数 *Check*:*GName*×*Params*×*InstructSet*→*BOOL* 为协议检查函数,在指令集 *InstructSet* 检查指令 *GName* 及其参数 *Params* 是否符合定义.

当接收到数据请求时,将该请求通过指定类型的测试设备发送给被测产品;若接收到指令请求时,检查是否符合协议,符合则将该指令转发给指定设备(类),否则返回协议检查错误编码,回令置为 false.若设备级测试设备网关层 DAPP 的语义函数 $\circlearrowleft_{\mathcal{M}}^{Dapp}$,*DAck* 为 DAPP 层响应动作,具体定义如图 5 所示.

$\circlearrowleft_{\mathcal{M}}^{Dapp}$:*DAck*×*BChannel*×*DChannel*×*InstructSet*→*BChannel*

For $dack \in DAck$, $\rho=\rho^T\cup\rho^P$

1) if $(a,dk,dps,0)\in\zeta^1(m,dapp)$ $\circlearrowleft_{\mathcal{M}}^{Dapp}$【*dack*】

$$(\zeta,\varsigma,\rho^T,\rho^P)=\begin{cases}\phi(\zeta,\zeta') \text{ IF } (a,dk,dps,0)\in\zeta^1(m,dapp) \text{ WHERE}\\ \qquad \zeta'^1(dapp,dk)=AddDRetnVS(a,dk,dps,0)\\ \qquad \phi(\zeta,\varsigma)=\begin{cases}(\zeta',\varsigma'') \text{ IF } (a,dk,dps,1)\in\zeta^1(dapp,dk) \text{ WHERE}\\ \qquad\qquad \zeta'^1(m,dapp)=UpdateBRetnVS(a,dk,dps,1)\\ \qquad\qquad \varsigma''^1(dapp,dk)=DelDRetnVS(a,dk,dps,1)\\ \varphi(\eta,\zeta) \text{ OTHERS}\end{cases}\\ (\eta,\zeta) \text{ OTHERS}\end{cases}$$

2) if $(a,dk,dpop,as,0)\in\zeta^2(m,dapp)$

a) if $Check(dk,dpop,as,\rho)$=(false,errcode) $\circlearrowleft_{\mathcal{M}}^{Dapp}$【*dack*】$(\zeta,\varsigma,\rho^T,\rho^P)=(\zeta',\varsigma)$

where, $\zeta'^2(m,dapp)=Update\ DSendGS(a,G\{(dk,dpop),as\},((\text{false, errcode}),1))$.

b) if $Check(dk,dpop,as,\rho)$=true $\circlearrowleft_{\mathcal{M}}^{Dapp}$【(*dack*)】

$$(\zeta,\varsigma,\rho^T,\rho^P)=\begin{cases}\psi(\zeta,\zeta') \text{ IF } (a,dk,dps,0)\in\zeta^2(m,dapp) \text{ WHERE}\\ \qquad \zeta'^2(dapp,dk)=AddD\,\mathrm{Re}tnVS(a,dk,dps,0)\\ \qquad \phi(\zeta,\varsigma)=\begin{cases}(\zeta',\varsigma'') \text{ IF } (a,dk,dps,1)\in\zeta^2(dapp,dk) \text{ WHERE}\\ \qquad\qquad \zeta'^2(m,dapp)=UpdateBRetnVS(a,dk,dps,1)\\ \qquad\qquad \varsigma''^2(dapp,dk)=DelDRetnVS(a,dk,dps,1)\\ \varphi(\eta,\zeta) \text{ OTHERS}\end{cases}\\ (\eta,\zeta) \text{ OTHERS}\end{cases}$$

Fig.5 Semantics of DAPP of test equipment layer

图 5 测试设备端 DAPP 层语义

4.2.3 测试设备层 TE

测试设备层 TE 根据具体设备的状态信息,找到合适设备执行数据请求或指令请求.完成到具体测试设备的访问.

当接收到关于被测产品状态参数值的数据请求时,则找到合适的设备将该请求发给该被测产品并获取参数值;当接收到关于自身的操作指令时,则找到合适的设备执行该指令,并将执行该指令产生的激励信号发送给被测产品;当接收到关于被测产品的指令请求时,则通过具体设备将该指令发给被测产品,并将被测产品发来的回令返回给上层.设备层的进程 *Dev* 可定义为:

$$Dev::=DevKind[OpId,Vals]|DevKind(OpId,Vals)|DevKind(PParam),$$

其中,$d[f,as]$表示接受到关于测试设备的指令,d 是设备类名,f 是 d 上的操作,as 为参数值;$d(g,ps)$表示接受到,通过

测试设备 *d* 传送的关于被测产品的指令 *g*,*ps* 为参数值;*d*(*pn*)表示通过测试设备 *d* 请求被测产品状态参数 *pn* 的值.根据上述测试设备层的功能描述,其语义定义如图 6 所示.其中,*DevInfo* 为测试设备信息环境,用于标识测试设备的使用状态,*DevInfo*=*DevId*→{'*Busy*','*Idle*'},*DevId* 为测试设备名域.函数 *lookupId*:*DevKind*×*DevInfo*→*DevId*,*lookupId*(*dk*,ε)为在当前测试设备信息环境ε下,在设备类 *dk* 中寻找适合的具体测试设备名.

$$\supset_{M}^{Dev}:Dev\times DevInfo\times DevC\times OpEnv\times DChannel\times PChannel\rightarrow DChannel\times PChannel$$

For $\varepsilon\in DevInfo$, $\delta\in DevC$, $\chi\in OpEnv$

1) First order data collaboration: if $(a,dk,dps,0)\in\zeta^1(dapp,dk)\supset_{M}^{Dev}$【$(dk(dps),\varepsilon,\delta,\chi)$】

$$(\varsigma,\xi)=\begin{cases}\theta(\varsigma,\xi') \text{ IF } (a,dk,dps,0)\in\zeta^1(dapp,dk) \text{ WHERE}\\ \qquad \xi''^1(dk)=AddDRetnVS(a,dk,dps,0)\\ \qquad \theta(\varsigma,\xi)=\begin{cases}(\varsigma',\xi'') \text{ IF } (a,dk,dps,1)\in\xi^1(dk) \text{ WHERE}\\ \qquad\qquad \zeta'^1(dapp,dk)=UpdateDRetnVS(a,dk,dps,1)\\ \qquad\qquad \xi''^1(dk)=DelPRetnVS(a,dk,dps,1)\\ \theta(\varsigma,\xi) \text{ OTHERS}\end{cases}\\ (\varsigma,\xi) \text{ OTHERS}\end{cases}$$

2) High order operation collaboration

(i) if $dpop=(\text{“}dev\text{”},df)$ then $\supset_{M}^{Dev}$【$(dk[dpop,as],\varepsilon,\delta,\chi)$】$(\varsigma,\xi)=(\varsigma',\xi)$

where, $d=lookupId(dk,\varepsilon)$, $(\delta',psvl)=\chi\ dk\ dpop\ as\ \delta$, $\zeta'^2(dapp)=Add\ PSendGS(a,G\{dk,-,psvl\})$,

$\zeta'^2(dapp,dk)=Update\ DSendGS(a,dk,dpop,as,(\text{true},1))$.

(ii) if $dpop=(\text{“}prod\text{”},pf)\supset_{M}^{Dev}$【$(dk(dpop,as),\varepsilon,\delta,\chi)$】

$$(\varsigma,\xi)=\begin{cases}\theta(\varsigma,\xi') \text{ IF } (a,dk,dpop,as,0)\in\zeta^2(dapp,dk) \text{ WHERE}\\ \qquad \xi'^2(dk)=AddDRetnVS(a,dk,dpop,0)\\ \qquad \Theta(\varsigma,\xi')=\begin{cases}(\varsigma',\xi'') \text{ IF } (a,dk,dpop,as,(rtn,1))\in\xi^2(dk) \text{ WHERE}\\ \qquad\qquad \zeta'^2(dapp,dk)=UpdateDSentGS(a,dk,dpop,as,(rts,1))\\ \qquad\qquad \xi''^2(dk)=DelPSentGS(a,dk,dpop,as,(rtn,1))\\ \Theta(\varsigma,\xi) \text{ OTHERS}\end{cases}\\ (\varsigma,\xi) \text{ OTHERS}\end{cases}$$

Fig.6 Semantics of DK of test equipment layer

图 6 测试设备端 DK 层语义

测试原子进程为测试任务端基本测试进程,具体见本文第 4.3 节测试任务端定义部分.

4.3 测试任务端TWPart定义

4.3.1 TWPart 语法

在测试任务层,通过对基本测试方法与操作的规范与封装,定义测试过程中的基本测试动作——测试原子.测试原子通过其内部动作与测试设备、被测产品进行指令与数据交互,并根据接收到的测试数据信息进行判别,完成一个相对独立的测试过程.之后,按照被测产品功能部件之间的逻辑关系,定义测试原子上的控制结构组成测试原子序列.这样,关于被测产品的测试任务由组成测试任务的测试原子序列来描述.

一般情况下,测试原子具有前置条件,当前置条件满足时,测试原子才能执行.同时,由于实时性的影响,每个测试原子具有时间约束,表示测试动作在一定时间范围内的有效性.因此,在测试系统中,测试原子可抽象定义如下:

定义 11(测试原子). 自动化测试模型中,测试原子域 *Atom* 定义为 *Atom*=*Precond*→*AName*(*Runtime*)[*TestP*],其中,*Precond* 为布尔表达式,表示测试原子的前置条件;*AName* 为测试原子名;*Runtime* 为测试原子时间约束.联合域 *TestP* 为组成测试原子的内部动作,是由与测试设备端发生的交互动作 *ComA*、判别动作 *Judge* 和关于时间的等待动作 *Wait* 组成.即,*TestP*=*ComA*+*Judge*(*Num*×*Type*×*Val*)+*Wait*(*Num*).联合域 *ComA* 包括向测试设备端的 *MTP* 发送指令 *MName*!*GuardV* 和从 *MTP* 接收参数 *MName*?*DataV* 两种动作,即,*ComA*=*MName*?*DataV*+*MName*!*GuardV*.其中,"+"表示域的联合.

因此,测试任务端的测试任务 T 语法定义为

T::=*skip*|*AtomL*||$_t$*T*|*AtomL*;$_t$*T*

AtomL::=*skip*|*Atom*;$_t$*Atom L*|*Atom*||$_t$*AtomL*

|while$_t$ *BExp* do *AtomL od*

|if *BExp AtomL* else *AtomL*

其中,*skip* 表示空,*AtomL* 为由测试原子复合组成的测试原子序列.

4.3.2 TWPart 的语义

测试任务端根据被测产品的测试需求形成测试用例,通过与测试设备端、被测产品端的交互,返回测试结果,完成对被测产品的测试.因此,测试任务端的形式语义可表示 *TWPart* 上映射函数 $\circlearrowleft\mathcal{M}^{Twp}$,定义为

$$\circlearrowleft\mathcal{M}^{Twp}:TWPart\times TEQPTPart\times PRODPart\to Result.$$

即 $\circlearrowleft\mathcal{M}^{Twp}:(T\times TWEnv\times Timer)\times TEQPTPart\times PRODPart\to Result.$

测试任务端的测试行为由测试任务 T 完成,若定义测试任务 T 的语义函数为 $\circlearrowleft\mathcal{M}^{T}$,则 $\circlearrowleft\mathcal{M}^{T}=\circlearrowleft\mathcal{M}^{Twp}$.而测试任务 T 由测试原子序列构成,故首先给出测试原子的语义函数,接着,通过给出测试原子序列的语义函数定义 $\circlearrowleft\mathcal{M}^{T}$.

测试任务端测试原子的功能是通过其内部动作与测试设备和被测产品的交互完成测试原子的测试过程.因此在自动化测试模型中,定义 $\circlearrowleft\mathcal{M}^{Atom}$ 为测试原子的语义函数,其定义为 $\circlearrowleft\mathcal{M}^{Atom}:Atom\times TWEnv\times Timer\times TEQPTPart\times PRODPart\to Result$.其中,$TWEnv$ 为测试原子执行的外部环境,其他定义如上.若 $\iota\in TWEnv,\tau\in Timer,(Teqpt,\eta,\rho^T)\in TEQPTPart,(Prod,\xi,\rho^P)\in PRODPart$,则对一个测试原子 $pc\to a(rt)[tp]$,有:

$$\circlearrowleft\mathcal{M}^{Atom}【pc\to a(rt)[tp]】(\iota,\tau,(Teqpt,\eta,\rho^T),(Prod,\xi,\rho^P))=$$

$$\circlearrowleft\mathcal{M}【pc】\iota\tau\to TR((\circlearrowleft\mathcal{M}^{AtTs}【(a,tp)】(\tau,\eta)|\circlearrowleft\mathcal{M}^{Tep}【Teqpt】(\eta,\xi,\rho^T,\rho^P)|\circlearrowleft\mathcal{M}^{Prodp}【Prod】\xi),\tau(time),rt,\tau),\text{false}.$$

其中,$\circlearrowleft\mathcal{M}^{AtTs}$ 函数为测试原子 a 内部动作 tp 的执行语义函数,下面将具体讨论.

若 $\circlearrowleft\mathcal{M}^{AtTs}【(a,tp)】(\tau,\eta)=(r,\eta')$,$\circlearrowleft\mathcal{M}^{Tep}【Teqpt】(\eta,\xi,\rho^T,\rho^P)=(\eta',\xi')$,$\circlearrowleft\mathcal{M}^{Prodp}【Prod】\xi=(\gamma,\xi')$,则

$$\circlearrowleft\mathcal{M}^{AtTs}【(a,tp)】(\tau,\eta)|\circlearrowleft\mathcal{M}^{Tep}【Teqpt】(\eta,\xi,\rho^T,\rho^P)|\circlearrowleft\mathcal{M}^{Prodp}【Prod】\xi=r.$$

测试原子通过其内部动作改变与测试设备端的通道 *HChannel* 内容来实现与测试设备端的交互,将测试原子的指令或参数请求发送给 *HChannel*,并根据 *HChannel* 的返回结果给出测试原子的执行结果.测试原子内部动作的执行语义函数 M^{AtTs} 定义为 $AName\times TestP\times HChannel\times Timer\to Result\times HChannel$.即不考虑时间约束条件下,测试原子内部动作的执行过程描述.对一个测试原子 $a,\eta\in HChannel,\tau\in Timer$,其内部动作执行的语义为:

- $\circlearrowleft\mathcal{M}^{AtTs}:AName\times TestP\times HChannel\times Timer\to Result\times Hchannel$;
- $\circlearrowleft\mathcal{M}^{AtTs}【(a,nil)】\eta,\tau=(\text{true},\eta)$;
- $\circlearrowleft\mathcal{M}^{AtTs}【(a,Judge(n,v))】\eta,\tau=n=?v$;
- $\circlearrowleft\mathcal{M}^{AtTs}【(a,Wait(n))】\eta,\tau=\kappa(n,\tau(time),\tau)$,其中,

$$\kappa(n,ct,\tau)=\tau(time)-ct<n\to\kappa(n,ct,\tau),(\text{true},\eta);$$

- $\circlearrowleft\mathcal{M}^{AtTs}【(a,m?(dk,dps))】\eta,\tau=\Omega(m,a,\eta',\tau)$,其中,$\eta'^1(m,a)=(dk,dps,0)$,

$$\Omega(m,a,\eta,\tau)\begin{cases}(\text{true},\eta'')\ \text{IF}\ (dk,dps,1)\in\eta^1(m,a)\ \text{WHERE}\\ \qquad\eta''^1(m,a)=DelARetnVS((dk,dps,1));\\ \Omega(m,a,\eta,\tau)\ \text{OTHERS}\end{cases}$$

- $\circlearrowleft\mathcal{M}^{AtTs}【(a,m!G\{dk,dpop,ps\})】\eta,\tau=\Upsilon(m,a,\eta',\tau)$,其中,$\eta'^2(m,a)=(dk,dpop,ps,0)$,

$$\Upsilon(m,a,\eta,\tau)\begin{cases}(rtn^1,\eta'')\ \text{IF}\ (Gdk,dpos,ps,(rtn,1))\in\eta^2(m,a)\ \text{WHERE}\\ \qquad\eta''^2(m,a)=DelASendGS((Gdk,dpop,ps,(rtn,1)));\\ \Upsilon(m,a,\eta,\tau)\ \text{OTHERS}\end{cases}$$

对于判参操作 $Judge(n,v)$,判断状态参数 n 是否符合 v.v 可以表征某一范围值或者某一具体值.根据文献[35],其判断都可以归为等式逻辑的判断操作,故用 $n=?v$ 形式表示状态参数 n 是否符合 v.

根据测试原子的语义函数和测试原子上的复合操作含义,容易给出测试原子序列的语义函数 $\circ\mathcal{M}^{AtomL}$,主要包括测试原子上时序串行、并行、循环和选择等复合操作的语义:

- $\circ\mathcal{M}^{AtomL}$:$AtomL\times TWEnv\times Timer\times TEQPTPart\times PRODPart\to Result$;
- $\circ\mathcal{M}^{AtomL}$【$skip$】($\iota,\tau,Tep,Prodp$)=true;
- $\circ\mathcal{M}^{AtomL}$【$atom;_t atomL$】($\iota,\tau,Tep,Prodp$)=$\circ\mathcal{M}^{Atom}$【$atom$】($\iota,\tau,Tep,Prodp$)$\to\circ\mathcal{M}^{AtomL}$【$atomL$】($\iota,\tau,Tep,Prodp$), false;
- $\circ\mathcal{M}^{AtomL}$【$atomL_1||_t atomL_2$】($\iota,\tau,Tep,Prodp$)=$r_1\wedge r_2$,其中,
  $r_1=\circ\mathcal{M}^{AtomL}$【$atomL_1$】($\iota,\tau,Tep,Prodp$),$r_2=\circ\mathcal{M}^{AtomL}$【$atomL_2$】($\iota,\tau,Tep,Prodp$);
- $\circ\mathcal{M}^{AtomL}$【while$_t$ b do $atomL$ od】($\iota,\tau,Tep,Prodp$)=$\partial(\iota,\tau,Tep,Prodp)$,其中,
  $\partial(\iota,\tau,Tep,Prodp)=\circ\mathcal{M}$【$b$】($\iota,\tau$)$\to(r\to\psi(\iota,\tau,Tep,Prodp)$,true),true,
  其中,$r=\circ\mathcal{M}^{AtomL}$【$atomL$】($\iota,\tau,Tep,Prodp$);
- $\circ\mathcal{M}^{AtomL}$【if$_t$ b $atomL_1$ else $atomL_2$】($\iota,\tau,Tep,Prodp$)=$\circ\mathcal{M}$【b】(ι,τ)$\to\circ\mathcal{M}^{AtomL}$【$atomL_1$】($\iota,\tau,Tep,Prodp$), $\circ\mathcal{M}^{AtomL}$【$atomL_2$】($\iota,\tau,Tep,Prodp$).

这样,测试任务 T 上的语义函数 $\circ\mathcal{M}^t$ 定义为:

- $\circ\mathcal{M}^t$:$T\times TWEnv\times Timer\times TEQPTPart\times PRODPart\to Result$;
- $\circ\mathcal{M}^t$【$skip$】($\iota,\tau,Tep,Prodp$)=true;
- $\circ\mathcal{M}^t$【$atoml;_t t$】($\iota,\tau,Tep,Prodp$)=$r\to\circ\mathcal{M}^t$【t】($\iota,\tau,Tep,Prodp$),false,其中,$r=\circ\mathcal{M}^{AtomL}$【$atoml$】($\iota,\tau,Tep,Prodp$);
- $\circ\mathcal{M}^t$【$atoml||_t t$】($\iota,\tau,Tep,Prodp$)=$r_1\wedge r_2$,其中,$r_1=\circ\mathcal{M}^{AtomL}$【$atoml$】($\iota,\tau,Tep,Prodp$),$r_2=\circ\mathcal{M}^t$【$t$】($\iota,\tau,Tep,Prodp$).

5 安全苛刻系统自动化测试模型 SCSAutTM 相关性质

由于涉及高阶协同和 4 类时间约束复合操作,需阐述模型的正确性和收敛性.以下将给出模型的收敛性和正确性证明.

5.1 SCSAutTM的收敛性

定理 1. 在自动化测试模型中,对任意测试原子,其上的语义函数 $\circ\mathcal{M}^{Atom}$ 是收敛的.

证明:在自动化测试模型中,对一个测试原子 $pc\to a(rt)[tp]$,有:

$\circ\mathcal{M}^{Atom}$【$pc\to a(rt)[tp]$】($\iota,\tau,(Teqpt,\eta,\rho^T),(Prod,\xi,\rho^P)$)=

$\circ\mathcal{M}$【pc】$\iota\tau\to TR((\circ\mathcal{M}^{AtTs}$【$(a,tp)$】$(\tau,\eta)|\circ\mathcal{M}^{Tep}$【$Teqpt$】$(\eta,\xi,\rho^T,\rho^P)|\circ\mathcal{M}^{Prodp}$【$Prod$】$\xi),\tau(time),rt,\tau)$,false.

设:

- $G=\circ\mathcal{M}^{Atom}$【$pc\to a(rt)[tp]$】($\iota,\tau,(Teqpt,\eta,\rho^T),(Prod,\xi,\rho^P)$);
- $H=\circ\mathcal{M}^{AtTs}$【$(a,tp)$】$(\tau,\eta)|\circ\mathcal{M}^{Tep}$【$Teqpt$】$(\eta,\xi,\rho^T,\rho^P)|\circ\mathcal{M}^{Prodp}$【$Prod$】$\xi$);
- $F=TR(H,\tau(time),rt,\tau)$,

则 $G=\circ\mathcal{M}$【pc】$\iota\tau\to F$,false.

当前置条件不满足时,测试原子不执行,返回结果为 false,收敛.以下证明当前置条件为真时,测试原子的执行过程收敛.采用结构归纳法证明:

(1) 当 $tp=Skip$ 时,$\circ\mathcal{M}^{AtTs}$【(a,nil)】(τ,η)=(true,η),H=true,$F=TR$(true,$\tau(time),rt,\tau$)=true,收敛.

(2) 当 $tp=Wait$ 时,$\circ\mathcal{M}^{AtTs}$【$(a,Wait(n))$】$\eta,\tau=\kappa(n,\tau(time),\rho,\tau)$,其中,递归函数 $\kappa(n,ct,\tau)=\tau(time)-ct<n\to\kappa(n,ct,\tau)$,(true,$\eta$).由于时钟$\tau$为增函数,存在某时刻 $t\in TIME$,使 $t-ct>n$,即函数 κ 最终将收敛为(true,η),H=true,$F=TR$(true,$\tau(time),rt,\tau$)=true.若$\exists t'\in TIME$,设 $st=\tau(time)$,$t-ct<n$ 且 $t-st>rt$,则根据 TR 定义,F=false.故收敛.

(3) 当 $tp=Judge$ 时,$\circ\mathcal{M}^{AtTs}$【$(a,Judge(n,v))$】$\eta,\tau=((n=?v),\eta)$,$H=(n=?v)$,$F=TR((n=?v),\tau(time),rt,\tau)$.$F$ 的结果即为$(n=?v)$.故收敛.

(4) 当 $tp=m?(dk,dps)$时,$\circ\mathcal{M}^{AtTs}$【$(a,m?(dk,dps))$】$\eta,\tau=\Omega(m,a,\eta',\tau)$,其中,$\eta'^{1}(m,a)=(dk,dps,0)$,函数$\Omega$的含义见上.令 $st=\tau(time)$,$F=TR(H,st,rt,\tau)$,若$\exists t,t-st>rt$,H 函数未收敛,则据 TR 定义,F=false;若$\exists t,t-st<rt$,H 函数收敛,则收敛.

以下证明:在理想情况下,函数 $H=\Omega(m,a,\eta',\tau)|\circlearrowright^{Tep}【Teqpt】(\eta',\xi,\rho^T,\rho^P)|\circlearrowright^{Prodp}【Prod】\xi$收敛:

根据上述定义,对 $Teqpt=(Mtp,Dapp,Dk)$,有:

$$\circlearrowright^{Tep}【Teqpt】(\eta',\xi,\rho^T,\rho^P)=\circlearrowright^{Tep}【(Mtp,Dapp,Dk)】(\rho^T,\eta',\xi,\rho^P)$$
$$=\circlearrowright^{Mtp}【Mtp】(\eta',\zeta)|\circlearrowright^{Dapp}【Dapp】(\zeta,\xi,\rho^T,\rho^P)|\circlearrowright^{Dk}【Dk】(\varsigma,\xi).$$

对 $\circlearrowright^{Mtp}【Mtp】(\eta',\zeta)$:

- 若当 $Mtp\neq m[a!(dk,dps)]$时,即没有响应测试原子 a 的请求时,Φ函数不终止,$\Omega(m,a,\eta',\tau)$函数也将不终止.根据 *TR* 定义(步骤(2)),则 H=false 终止;
- 当 $Mtp=m[a!(dk,dps)]$时:
  - $\circlearrowright^{Mtp}【Mtp】(\eta',\zeta)=\Phi(\eta',\zeta')$;
  - $\circlearrowright^{Tep}【Teqpt】(\eta',\xi,\rho^T,\rho^T)=\Phi(\eta',\zeta')|\circlearrowright^{Dapp}【Dapp】(\zeta',\varsigma,\rho^T,\rho^P)|\circlearrowright^{Dk}【Dk】(\varsigma,\xi)$.

  其中,$\zeta'^1(m,dapp)=Add\ BRetnVS(a,dk,dps,0)$.

在 *DAPP* 层,根据给定定义,有 $\circlearrowright^{Dapp}【dack】(\zeta',\varsigma,\rho^T,\rho^P)=\phi(\zeta',\varsigma')$,其中,$\varsigma'^1(dapp,dk)=Add\ DRetnVS(a,dk,dps,0)$,$\phi$定义见上.此时有 $\circlearrowright^{Tep}【Teqpt】(\eta',\xi,\rho^T,\rho^T)=\Phi(\eta',\zeta')|\phi(\zeta',\varsigma')|\circlearrowright^{Dk}【Dk】(\varsigma,\xi)$.

在 *TE* 层,对 $\varepsilon\in DevInfo,\delta\in DevC,\chi\in OpEnv$,根据给定定义有 $\circlearrowright^{Dev}【(dk(dps),\varepsilon,\delta,\chi)】(\varsigma',\xi)=\theta(\varsigma',\xi')$,其中,$\xi'^1(dk)=Add\ DRetnVS(a,dk,dps,0)$,$\theta$定义见上.此时,$\circlearrowright^{Tep}【Teqpt】(\eta',\xi,\rho^T,\rho^T)=\Phi(\eta',\zeta')|\phi(\zeta',\varsigma')|\theta(\varsigma',\xi')$.

在 *PROD* 端,设$(ack,\gamma,\Upsilon)\in PROD,(pn,pv)=dps,\circlearrowright^{Prodp}【(ack,\gamma,\Upsilon)】\xi'=(\gamma,\xi'')$,其中,

$$pv=\gamma(pn),\xi''^1(dk)=Update\ PRetnVS(a,dk,(pn,pv),1).$$

因此,

$$H=\Omega(m,a,\eta',\tau)|\Phi(\eta',\zeta')|\phi(\zeta',\varsigma')|\theta(\varsigma',\xi')|(\gamma,\xi'')$$
$$=\Omega(m,a,\eta',\tau)|\Phi(\eta',\zeta')|\phi(\zeta',\varsigma')|(\varsigma'',\xi'')|(\gamma,\xi''),\varsigma''^1(dapp,dk)=Update\ DRetnVS(a,dk,(pn,pv),1),$$
$$\xi''(dk)=Del\ PRetnVS(a,dk,(pn,pv),1)$$
$$=\Omega(m,a,\eta',\tau)|\Phi(\eta',\zeta')|(\zeta'',\varsigma'')|(\varsigma'',\xi'')|(\gamma,\xi''),\zeta''^1(m,dapp)=Update\ BRetnVS(a,dk,(pn,pv),1),$$
$$\varsigma''^1(dapp,dk)=Del\ DRetnVS(a,dk,(pn,pv),1)$$
$$=\Omega(m,a,\eta',\tau)|(\eta'',\zeta'')|(\zeta'',\varsigma'')|(\varsigma'',\xi'')|(\gamma,\xi''),\eta''^1(m,a)=Update\ ARetnVS(dk,(pn,pv),1),$$
$$\zeta''^1(m,dapp)=Del\ BRetnVS(a,dk,(pn,pv),1)$$
$$=\Omega(m,a,\eta'',\tau)|(\eta'',\zeta'')|(\zeta'',\varsigma'')|(\varsigma'',\xi'')|(\gamma,\xi''),\eta''^1(m,a)=Del\ ARetnVS(a,dk,(pn,pv),1)$$
$$=\text{true}$$

故收敛.

但上述计算过程的递归函数不中止时,根据 *TR* 定义(步骤(2)),则 H=false 终止.

(5) 当 $tp=m!gv$ 时,其过程同步骤(4),可证明同样函数 H 收敛. □

定理 2. 在自动化测试模型中,由测试原子通过时间约束串行、时间约束并行、时间约束循环和时间约束选择复合操作组成的测试任务 T,其语义函数 $\circlearrowright^T$ 是收敛的.

证明:由定理 1,根据测试原子上时间约束串行、时间约束并行、时间约束循环和时间约束选择复合操作语义函数定义,容易证明.篇幅有限,此处略去. □

5.2 SCSAutTM的正确性

5.2.1 安全苛刻系统自动化测试过程的抽象描述

本节将给出安全苛刻系统自动化测试的抽象机,用于描述安全苛刻系统自动化测试的实际测试过程.由于安全苛刻系统自动化测试具有实时性,定义⊢表示转换的前提,测试任务与测试设备交互返回值α为系统转换标记,即"$\rightarrow_\alpha$","$\rightarrow$"表示无交互返回值;"$\rightarrow_{\vec{\alpha}}$"表示"$\rightarrow_\alpha$"∪"$\rightarrow$".下述用 $st\in TIME,\tau\in Timer\vdash$表示在实时环境 τ 和某一时刻 st 下的转换前提.$Timenotout:Timer\times Time\times Time\text{-}restriction\rightarrow Bool$ 函数用于监测计算过程是否超时:若超时,则为真;否则为假.实际测试过程的抽象语义描述为:

(1) 空规则:当测试原子 a 中的测试动作为空操作 *skip* 时,直接返回 true 结果.即:

$$st\in TIME,\tau\in Timer\vdash a(rt)[Nil]\rightarrow \text{true}.$$

(2) *Wait* 规则:当测试原子 a 中的测试动作为等待操作 $Wait(n)$时,若在等待操作执行过程中 a 超时,则终止等待动作;否则,等待指定的 n 个时间段.即:

$$st\in TIME,\tau\in Timer\vdash \frac{timenotout(\tau,st,rt)=\text{true},\tau(time)-st=n}{a(rt)[Wait(n)]\rightarrow \text{true}};$$

$$st\in TIME,\tau\in Timer\vdash \frac{timenotout(\tau,\tau(time),rt)=\text{false}}{a(rt)[Wait(n)]\rightarrow \text{false}}.$$

(3) *Judge* 规则:当测试原子 a 中的动作为判参操作 $Judge(n,v)$时,根据参数的名字和值 v 判断是否符合参数定义.若在执行过程中 a 超时,则终止判参操作,返回 false 结果;否则,返回判参结果.即:

$$st\in TIME,\tau\in Timer\vdash \frac{timenotout(\tau,\tau(time),rt)=\text{true}}{a(rt)[Judge(n,v)]\rightarrow (n=?v)};$$

$$st\in TIME,\tau\in Timer\vdash \frac{timenotout(\tau,\tau(time),rt)=\text{false}}{a(rt)[Judge(n,v)]\rightarrow \text{false}}.$$

(4) 交互指令规则:当测试原子 a 中的动作为发指令操作 $m!gv$ 时,a 将通过测试设备端的 *MTP* 将指令 gv 传给测试设备或被测产品,返回回令 true.同样地,若在执行过程中 a 执行超时,则终止该动作的执行,返回 false 结果.即:

$$st\in TIME,\tau\in Timer\vdash \frac{timenotout(\tau,\tau(time),rt)=\text{true}}{a(rt)[m!gv]\rightarrow_{(b,c)} b};$$

$$st\in TIME,\tau\in Timer\vdash \frac{timenotout(\tau,\tau(time),rt)=\text{false}}{a(rt)[m!gv]\rightarrow_{\vec{a}} \text{false}}.$$

(5) 交互参数规则:当测试原子 a 中的动作为取参数操作 $m!dv$ 时,a 将通过测试设备端的 *MTP* 向被测产品发出请求,取回参数值,返回 true.同样地,若在执行过程中 a 执行超时,则终止该动作的执行,返回 false 结果.即:

$$st\in TIME,\tau\in Timer\vdash \frac{timenotout(\tau,\tau(time),rt)=\text{true}}{a(rt)[m?dv]\rightarrow_{\omega} \text{true}};$$

$$st\in TIME,\tau\in Timer\vdash \frac{timenotout(\tau,\tau(time),rt)=\text{false}}{a(rt)[m?dv]\rightarrow_{\vec{a}} \text{false}}.$$

(6) 复合规则:对 $a,a_1,a_2\in AtomL$:

① 时间约束串行规则($;_t$-rule):$a_1;_t a_2=a_1\rightarrow a_2$,false;

② 时间约束并行规则($\|_t$-rule):$a_1\|_t a_2=a_1\wedge a_2$;

③ 时间约束循环规则(while$_t$-rule):while$_t$ b do a od$=\varphi$,其中,$\varphi=b\rightarrow(a\rightarrow\varphi$,false),false;

④ 时间约束选择规则(if$_t$-rule):if$_t$ b then a_1 else $a_2=b\rightarrow a_1,a_2$.

5.2.2 SCSAutTM 的正确性证明

定理 3. 在自动化测试模型中,对任意测试原子,其执行语义函数 $\circlearrowleft\mathcal{M}^{Atom}$ 的执行结果与测试原子的抽象语义一致.

证明:在自动化测试模型中,对一个测试原子 $pc\rightarrow a(rt)[tp]$,有:

$$\circlearrowleft\mathcal{M}^{Atom}【pc\rightarrow a(rt)[tp]】(\iota,\tau,(Teqpt,\eta,\rho^T),(Prod,\xi,\rho^P))=$$

$$\circlearrowleft\mathcal{M}【pc】\ \iota\tau\rightarrow TR((\circlearrowleft\mathcal{M}^{AtTs}【(a,tp)】(\tau,\eta)|\circlearrowleft\mathcal{M}^{Tep}【Teqpt】(\eta,\xi,\rho^T,\rho^P)|\circlearrowleft\mathcal{M}^{Prodp}【Prod】\xi),\tau(time),rt,\tau),\text{false}.$$

设:

- $G=\circlearrowleft\mathcal{M}^{Atom}【pc\rightarrow a(rt)[tp]】(\iota,\tau,(Teqpt,\eta,\rho^T),(Prod,\xi,\rho^P))$;
- $F=TR((\circlearrowleft\mathcal{M}^{AtTs}【(a,tp)】(\tau,\eta)|\circlearrowleft\mathcal{M}^{Tep}【Teqpt】(\eta,\xi,\rho^T,\rho^P)|\circlearrowleft\mathcal{M}^{Prodp}【Prod】\xi),\tau(time),rt,\tau)$,

则 $G=\circlearrowleft\mathcal{M}【pc】\ \iota\tau\rightarrow F$,false.

当前置条件不满足时,测试原子不执行,返回结果为 false.以下证明当前置条件为真时,测试原子的执行与测试过程中测试原子的抽象语义一致.采用结构归纳法证明:

(1) 当 $tp=Skip$ 时,$\circ\mathcal{M}^{ATS}$【(a,nil)】$(\tau,\eta)=(\text{true},\eta)$,$F=TR(\text{true},\tau(time),rt,\tau)=\text{true}$,符合测试原子的空规则.

(2) 当 $tp=Wait$ 时,$\circ\mathcal{M}^{ATS}$【$(a,Wait(n))$】$\eta,\tau=\kappa(n,\tau(time),\rho,\tau)$,其中,递归函数 $\kappa(n,ct,\tau)=\tau(time)-ct<n\rightarrow\kappa(n,ct,\tau)$,$(\text{true},\eta)$.由于时钟$\tau$为增函数,存在某时刻 $t\in TIME$,使 $t-ct>n$.即,函数 κ 最终将收敛为(true,η),$F=TR(\text{true},\tau(time),rt,\tau)=\text{true}$.若$\exists t'\in TIME$,设 $st=\tau(time)$,$t-ct<n$ 且 $t-st>rt$,则根据 *TR* 定义,$F=\text{false}$.符合 *Wait* 规则.

(3) 当 $tp=Judge$ 时,$\circ\mathcal{M}^{ATS}$【$(a,Judge(n,v))$】$\eta,\tau=(n=?v),F=TR((n=?v),\tau(time),rt,\tau)$.$F$ 的结果即为$(n=?v)$.符合 *Judge* 规则.

(4) 当 $tp=m?dv$ 时,见第 5.1 节.符合交互参数规则.

(5) 当 $tp=m!gv$ 时,见第 5.1 节.符合交互指令规则. □

定理 4. 在自动化测试模型中,由测试原子通过时间约束串行、时间约束并行、时间约束循环和时间约束选择复合操作组成的测试任务,其语义函数 $\circ\mathcal{M}^{T}$ 与测试任务的抽象语义一致.

证明:由定理 3,根据测试原子上时间约束串行、时间约束并行、时间约束循环和时间约束选择复合操作语义函数定义,容易证明.篇幅有限,此处略. □

6 结论及未来工作

基于安全苛刻系统的可信性需求,在已有工作的基础上,通过对安全苛刻系统自动化测试分析,采用指称语义形式化方法,为安全苛刻系统的自动化测试建立了一种层次化高阶实时时序演算模型.本文通过对测试过程中高阶协同性的分析和定义,以论域理论为基础,给出自动化测试模型.该模型是一种由测试任务端、测试设备端和被测产品端组成的层次化架构,通过对三端高阶域及其域上方程定义,给出三端的形式化语义,定义了自动化测试模型.通过这种层次化描述,刻画了自动化测试过程中测试动作之间的控制协同和时间协同关系.通过对被测产品、测试设备与测试任务的抽象与组织,给出安全苛刻系统测试的信息化工作模式,实现安全苛刻系统自动化测试系统的组织结构、测试流程等的标准化定义.最后,通过对模型中高阶域上方程的收敛性和正确性的证明论证了自动化测试模型高阶协同行为的可计算性,验证了安全苛刻系统测试的可自动化.该模型可直接推广到多被测产品的批产化网络测试中.由于篇幅所限,将另文讨论模型的实时性和时序性.

未来工作将继续给出模型的实时性和时序分析与讨论;研究基于此模型的测试语言的设计和实现,包括测试设备操作等高阶指令的封装规范以支持测试语言的通用性,测试语言的控制结构定义以及测试语言实现中的并行、实时、时序等关键问题,以支持语言的实时、时序性;研究安全苛刻系统测试的测试需求描述语言,由于测试需求描述具有时序性,测试需求中测试参数(即被测产品状态)具有时序性,研究测试需求中测试参数之间满足的时序规约关系,形成对测试需求的合理表示;同时,将研究带时序的测试需求可满足定义和验证推理系统,验证测试需求的可满足性等;研究测试过程中测试设备的协同问题,包括测试设备的冲突定义、冲突发生条件和冲突消解规则,使测试设备在测试过程中能够检测冲突和消解冲突等.

References:

[1] Liu K, Shan ZG, Wang J, He JF, Zhang ZT, Qin YW. Overview on major research plan of trustworthy software. Bulletin of National Natural Science Foundation of China, 2008,3:145−151 (in Chinese with English abstract).

[2] Zheng ZM, Ma SL, Li W, Wei W, Jiang X, Zhang ZL, Guo BH. Dynamical characteristics of software trustworthiness and their evolutionary complexity. Science China (Series F—Information Sciences), 2009,52(9):1651−1657. [doi: 10.1007/s11432-009-0137-2]

[3] Zheng ZM, Ma SL, Li W, Jiang X, Wei W, Ma LL, Tang ST. Complexity of software trustworthiness and its dynamical statistical analysis methods. Science China (Series F—Information Sciences), 2009,52(8):1328−1334. [doi: 10.1007/s11432-009-0143-4]

[4] Mitchell TR. A standard test language—GOAL. In: Proc. of the 10th Workshop on Design Automation. 1973. 87−96.

[5] Garner JT. Satellite Control: Comprehensive Approach. New York: John Wiley and Sons, 1996.

[6] Integral Systems Incorporation. EPOCHT&CSTOL programmer's reference manual. ISI-EPOCH-0094, 1992.

[7] IEEE Standard Coordinating Committee. ATLAS 2000 Requirements Document Revision 2.1. New York: IEEE, 1996.

[8] Gil J, Holstein B. T++: A test case generator using a debugging information based technique for source code manipulation. Tools-23: Technology of Object-Oriented Languages and Systems. 1997. 272. [doi: 10.1109/TOOLS.1997.654735]

[9] ADS2—Avionics development system 2nd generation. http://www.techsat.com/fileadmin/media/pdf/ADS2_ProductOverview/TechSAT_PD_ADS2_CN_1000.pdf

[10] Zhang JQ. Study of Automatic Test Language for Spacecraft Application System. Beijing: Graduate University of Chinese Academy of Sciences (Center for Space Science and Applied Research), 2005 (in Chinese with English abstract).

[11] Li ZY. Study of language for satellite testing and operation and its integrated support environment. Beijing: Graduate University of Chinese Academy of Sciences (Center for Space Science and Applied Research), 2004 (in Chinese with English abstract).

[12] Yu G, Xu ZW, Du JW. Research on scenario-event-driven simulation test script language for safety-critical software system. Journal of Computer Applications, 2010,30(2):374−379 (in Chinese with English abstract). [doi: 10.3724/SP.J.1087.2010.00374]

[13] Ma SL, Yu D. Spacecraft Automatic Test Language and System. National Defence Industry Press, 2011 (in Chinese).

[14] Thomsen B. A theory of higher order communication systems. Information and Computation, 1995,116:38−57. [doi: 10.1006/inco.1995.1004]

[15] Sangiorgi D. Expressing mobility in process algebras: First-order and higher-order paradigms [Ph.D. Thesis]. Edinburgh: University of Edinburgh, 1992.

[16] Li W, Wang JA. CC—A concurrent calculus for higher-order communicating systems. Journal of Beijing University of Aeronautics and Astronautics, 1992,3(3):12−18 (in Chinese with English abstract).

[17] Cardelli L, Gordon AD. Mobile ambients. Theoretical Computer Science, 2000,240(1):177−213. [doi: 10.1016/S0304-3975(99)00231-5]

[18] Reed GM, Roscoe AW. A timed model for communicating sequential processes. Theoretical Computer Science, Vol. 58. Pages 249−261. 1988.

[19] Alur R, Dill DL. A theory of timed automata. Theoretical Computer Science, 1994,126:183−235. [doi: 10.1016/0304-3975(94)90010-8]

[20] Li XS, Zhou CC. Duration calculi: An overview. Chinese Journal of Computers, 1994,17(11):842−851 (in Chinese with English abstract).

[21] Clarke EM, Emerson EA, Sistla AP. Automatic verification of finite state concurrent systems using temporal logic specffications. In: Proc. of the 10th Annual ACM Symp. on Principles of Programming Languages (POPL'83). New York: ACM Press, 1983. 117−126.

[22] Vardi MY, Wolper P. An automata theoretic approach to automatic program verification. In: Proc. of the 1st IEEE Symp. on Logic in Computer Science (LICS'86). Ins Alamitos: IEEE Computer Society, 1986. 322−331.

[23] Colim S. Modal and Temporal Properties of Processes. Chapter 5, Modal μ-calculus. New York, Berlin, Heidelberg: Springer-Verlag, 2001. 103−128.

[24] Degano P, Loddo JV, Priami C. Mobile processes with local clocks. Lecture Notes in Computer Science, 1996,1192:296−319. [doi: 10.1007/3-540-62503-8_14]

[25] Berger M. Basic theory of reduction congruence for two timed asynchronous π-calculi. Lecture Notes in Computer Science, 2004, 3170:115−130. [doi: 10.1007/978-3-540-28644-8_8]

[26] You T, Du CL, Wang XW, Zheng W. A new component-based real-time system based on timed high-order (THO) π calculus. Journal of Northwestern Polytechnical University, 2009,27(6):906−911 (in Chinese with English abstract).

[27] 詹乃军.高阶时段段演算及其完备性.中国科学(E 辑),2001,31(1):71−85.

[28] ISO/IEC. LOTOS—A formal description technique based on the temporal ordering of observational behavior. In: Proc. of the Int'l Standard 8807, Int'l Organization for Standardization, Information Processing Systems Open Systems Interconnection. Geneve, 1988.

[29] ISO/IEC. Information Technology—E-LOTOS. ISO/IEC Int'l Standard, 2001.

[30] Tang ZS, *et al*. A temporal logic language oriented toward software engineering. Beijing: Science Press, 1999.

[31] Milner R. A calculus of communicating systems. LNCS, 1980,92.

[32] Milner R, Parrow J, Walker D. A calculus of mobile processes (Parts I and II). Information and Computation, 1992,100:1−77. [doi: 10.1016/0890-5401(92)90008-4]

[33] Thomsen B. Calculi for higher order communicating systems [Ph.D. Thesis]. Department of Computing, Imperial College, 1990.

[34] Thomsen B. Plain CHOCS, a second generation calculus for higher-order processes. Acta Informatica, 1993,30:1−59. [doi: 10.1007/BF01200262]

[35] Li W, Ma SL. Limits of theory sequences over algebraically closed fields and applications. Discrete Applied Mathematics, 2004, 136(1):23−43. [doi: 10.1016/S0166-218X(03)00196-3]

附中文参考文献:

[1] 刘克,单志广,王戟,何积丰,张兆田,秦玉文.可信软件基础研究重大研究计划综述.中国科学基金,2008,3:145−151.

[10] 张建泉.面向航天应用的自动测试语言研究.北京:中国科学院研究生院(中国科学院空间科学与应用研究中心),2005.

[11] 李征宇.卫星测试操作语言及其集成支持环境研究.北京:中国科学院研究生院(中国科学院空间科学与应用研究中心),2004.

[12] 喻钢.场景一事件驱动的安全苛求软件系统仿真测试脚本语言研究.计算机应用,2010,30(2):374−379. [doi: 10.3724/SP.J.1087.2010.00374]

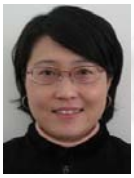
[13] 马世龙,余丹.航天器自动化测试语言及系统.北京:国防工业出版社,2010.

[16] 李未,王飚安.CC—高阶通信系统的并发演算.北京航空航天大学学报,1992,3(3):12−18.

[20] 李晓山,周巢尘.时段演算综述.计算机学报,1994,17(11):842−851.

[26] 尤涛,杜承烈,王小伟,郑炜.基于高阶时间π演算的构件式实时软件研究.西北工业大学学报,2009,27(6):906−911.

[30] 唐稚松,等.时序逻辑程序设计与软件工程.北京:科学出版社,1999.



吕江花(1975－),女,山东日照人,博士,讲师,主要研究领域为软件自动化,软件形式化方法.
E-mail: jhlv@nlsde.buaa.edu.cn

李先军(1977－),男,博士,主要研究领域为软件自动化测试方法.
E-mail: lixianjun@nlsde.buaa.edu.cn

马世龙(1953－),男,博士,教授,博士生导师,主要研究领域为网络环境下计算模型研究,海量信息处理计算模型研究,网格计算技术及其应用研究.
E-mail: slma@nlsde.buaa.edu.cn

高世伟(1983－),男,博士生,主要研究领域为自动化测试方法.
E-mail: ge89@163.com