软件学报 ISSN 1000-9825, CODEN RUXUEW
Journal of Software,2016,27(3):562−579 [doi: 10.13328/j.cnki.jos.004981]

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

安全苛刻系统测试语言中的测试设备协同语句*

吕江花[1], 高世伟[1], 马世龙[1], 孙 波[2], 李先军[1]

[1](北京航空航天大学 计算机学院,北京 100191)

[2](中国空间技术研究院,北京 100094)

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迫切,支持可信性评估的数据主要来自于测试.为了保证测试数据的可靠性和正确性,特别是对安全苛刻系统这类复杂系统,手工测试实际不可行.研发测试语言是实现自动化测试的有效途径,也是安全苛刻系统自动化测试发展的必然趋势.针对安全苛刻系统通用测试语言应独立于具体设备包括被测安全苛刻系统、测试设备的应用需求,对安全苛刻系统测试中的测试设备协同语句展开研究.针对安全苛刻系统测试中测试设备协同任务中的高阶性、实时性等特点,通过给出测试语言中测试设备协同相关类型、设备协同表达式,定义测试设备协同语句,并通过设备协同表达式求值定义设备协同语句的操作语义规则.最后,对语句的正确性给出相关证明,从而支持安全苛刻系统测试过程中测试设备协同过程的动态性和开放性,支持安全苛刻系统测试语言的通用性.

关键词: 可信性;自动化测试;测试设备协同;操作语义;安全苛刻系统

中图法分类号: TP311

中文引用格式: 吕江花,高世伟,马世龙,孙波,李先军.安全苛刻系统测试语言中的测试设备协同语句.软件学报,2016,27(3):562−579. http://www.jos.org.cn/1000-9825/4981.htm

英文引用格式: Lü JH, Gao SW, Ma SL, Sun B, Li XJ. Equipment collaboration in general test languages of safety critical system. Ruan Jian Xue Bao/Journal of Software, 2016,27(3):562−579 (in Chinese). http://www.jos.org.cn/1000-9825/4981.htm

Equipment Collaboration in General Test Languages of Safety Critical System

LÜ Jiang-Hua[1], GAO Shi-Wei[1], MA Shi-Long[1], SUN Bo[2], LI Xian-Jun[1]

[1](School of Computer Science and Engineering, Beihang University, Beijing 100191, China)

[2](China Academy of Space Technology, Beijing 100094, China)

Abstract: The trustworthiness of safety-critical systems (SCS) is very important. Assessing the trustworthiness mainly depends on data from test. In order to ensure the reliability and validity of test data, especially for such complex systems, manual testing is infeasible in practice. Development of test languages as effective way to implement automatic testing is inevitable trend for automatic testing of SCS. As in general test language for SCS, testing should be independent of specific equipment, including SCS (SCS under test) and test equipment. In the paper, the issues of equipment collaboration are discussed. Aiming at high order and real time characteristics of equipment collaboration during testing, types and expressions of equipment collaboration involved in test of SCS are proposed, and the syntax of statements of equipment collaboration is defined. Then by defining the evaluating rules of these equipment collaboration expressions, semantic rules of statements of equipment collaboration are specified, and related properties are proved to show the

* 基金项目: 国家自然科学基金(61300007); 北京航空航天大学软件开发环境国家重点实验室基金(SKLSDE-2015ZX-09, SKLSDE-2014ZX-06); 国家科技支撑计划(2013BAH46F00)

Foundation item: National Natural Science Foundation of China (61300007); State Key Laboratory of Software Development Environment Foundation (Beihang University) (SKLSDE-2015ZX-09, SKLSDE-2014ZX-06); National Key Technology Support Program (2013BAH46F00)

收稿时间: 2015-07-14; 修改时间: 2015-10-20; 采用时间: 2015-11-27; jos 在线出版时间: 2016-01-05

CNKI 网络优先出版: 2016-01-05 16:39:55, http://www.cnki.net/kcms/detail/11.2560.TP.20160105.1639.007.html

soundness of these semantic rules. This work demonstrates that the equipment collaboration is dynamic and open, and the test languages of SCS can be general.

Key words: trustworthiness; automatic test; testing equipment collaboration; operational semantics; SCS

在安全苛刻系统测试中,测试过程是部署的测试任务通过各种测试设备实现对被测产品的测试,测试任务通过向测试设备发送指令,测试设备执行指令后向被测产品发送激励信号,被测产品响应激励后改变自身状态参数并进行反馈,测试任务通过获取指定的测试参数值,进行评判的过程.测试过程如图 1 所示.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,这些设备之间的交互需要通过传送设备操作指令触发被测产品的功能操作,因此,硬件指令是协同的主要内容.安全苛刻系统的测试过程即是测试设备之间、被测产品之间以及测试设备与被测产品之间的协同过程,安全苛刻系统自动化测试语言的核心是设备协同机制及其相关设备协同语句.

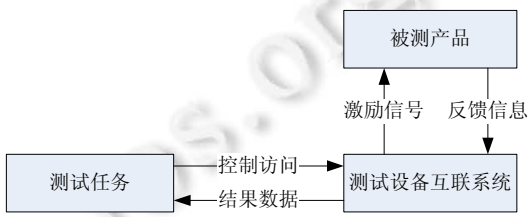

Fig.1 Process of SCS testing

图 1 安全苛刻系统测试过程

在安全苛刻系统测试中,不同型号的被测产品测试所涉及的测试设备不同,为支持测试系统的通用性(独立于测试数据、独立于被测产品、独立于测试设备),即,测试语言的通用性,安全苛刻系统自动化测试中的设备协同需具有通用性.特别是当前软硬件技术的高速发展,安全苛刻系统测试模式已经由单被测产品测试模式转变为多被测产品并行测试.在这种批产化网络并行测试需求下,测试系统需要支持各种测试设备和被测产品动态加入和退出.为了支持新的测试设备类动态加入自动化测试系统,并支持设备协同过程可定制,测试设备协同的动态性和开放性是通用安全苛刻系统测试语言设备协同研究中的关键问题.

在安全苛刻系统测试中,由于设备间交互内容不仅仅是一阶的数据信息,高阶的设备操作指令更是设备交互的主要内容,是典型的高阶协同系统[1,2].传统的做法是:将这些高阶的设备指令编码为一阶数据,按照一阶数据的协同来处理,简化难度.但带来的问题是测试系统和硬件设备的耦合度高,通用性差,难以通用于不同的安全苛刻系统型号的测试中,甚至难以用于同一类安全苛刻系统型号的升级测试中.另一方面,由于安全苛刻系统测试中激励反馈数据实时传输,测试设备协同过程也表现出时间约束性和时间协同性.

现有的设备协同模型主要可以分为以下两类:一类是设备间的间接协同,主要有设备远程访问和设备共享,其特点是设备类型和数量都比较少,支持用户协作,通过设备用户的协同间接实现设备间的协同行为;另一类是设备间简单协同模型,代表性有 BPEL[3],WSFL[4],YSWL[5]和 GSEL[6],这类模型是将设备服务封装为 Web 服务,利用 Web 服务的组合实现设备的协同.

在当前应用中,对设备协同的需求呈现动态性、开放性、高阶性以及实时性等特点,设备作为协同的主体,设备间协同流程实时动态,因而需要建立设备协同机制描述设备间的协同行为.在设备协同系统中,由于使用设备协同流程建模语言对设备协同流程进行描述,所以能支持设备协同的自动化.因此,设备协同建模语言是当前设备协同的发展趋势.

本文作者所在的课题组通过对现有设备协同系统及其关键技术总结[7],针对大规模设备协同应用的特点,给出了一种支持大规模照明设备协同语言[8],抽象设备操作,给出设备操作的解释执行机制,支持照明设备基于时间的协同.研究成果应用于奥运中心区景观照明设备协同系统项目 DCS(large-scale lighting device collaborative system),通过设备协同流程描述,实现了奥运中心区景观数字化绿色照明.

在安全苛刻系统测试中,为支持测试设备协同的动态性和开放性,设备协同语言需要与具体设备解耦合.同时,由于安全苛刻系统的特殊性,对实时具有较高要求,设备协同还具有实时性,支持设备协同操作的时间约束和时间协同.因此,为支持安全苛刻系统测试语言的通用性,需要研究安全苛刻系统测试语言中的设备协同机制.本文在文献[1,2]作者给出的安全苛刻系统自动化测试语义模型基础上,给出安全苛刻系统测试语言中测试设备协同语句及其相关证明,支持可以严格定义测试设备协同的通用安全苛刻系统测试语言.本文的重点是介绍测试语言中关于设备协同方面的内容,包括设备协同类型、设备协同表达式及其计算规则等.常规程序设计语言中常见的类型、表达式、语句等定义不再赘述.

1 测试设备协同相关类型

通用安全苛刻系统测试语言中,类型除了包括整型、布尔型、字符型、实型、数组等通常的类型,还包括关于设备协同的特殊类型,分别是测试数据类型、测试设备类型以及被测产品类型.

1.1 测试数据类型

测试数据类型包括测试指标类型、测试参数类型以及测试指令类型等.

(1) 测试指标类型

在测试中,测试指标表示了某个组件或系统的某个功能的正常工作状态的标准值或范围,一般是通过反映其功能的状态参数含义和参数值来评判功能是否正常.状态参数的含义是指参数在组件或者系统中的工作状态含义.不同阶段,同一状态参数的含义不同,其指标也可能不同.用 *Name* 表示参数名,*Sem* 表示参数含义,*StVal* 表示参数值,测试指标类型 *Standard* 可定义为测试参数名 *Name*、含义 *Sem* 和测试指标值 *StVal* 的元组表,见如下形式:

$$Standard=(Name\times Sem\times StVal)^{*}.$$

在安全苛刻系统测试中,测试指标可分为测试设备指标 *DevStandard* 和被测产品指标 *ProdStandard* 两类.即:

$$DevStandard=DParam\times Sem\times DStVal, ProdStandard=PParam\times Sem\times PStVal.$$

DParam 为测试设备状态参数名,*PParam* 为被测产品状态参数名,*DStVal* 和 *PStVal* 分别为测试设备和被测产品指标值.指标值可为通常语言中的数据类型或其上各种构造类型值.表 1 为某安全苛刻系统-航天器测试数管分系统中的部分测试指标.

Table 1 A system test indicator value table in testing of spacecraft

表 1 航天器测试数管分系统中的部分测试指标

指标名称	含义	正常指标范围	
		加电状态	未加电状态
VS1	遥控 *A* 机 PSK 幅度检测	0～5V	0V
VS2	遥控 *B* 机 PSK 幅度检测	0～5V	0V
VS8	数管计算机 DC/DC+12.6	12～13.6v	0～1V
VS9	数管计算机 DC/DC−12.6	−13.6～−12.0v	0～1V
…	…	…	…
S176	虚拟信道优先级标识——VC4 优先级	1	0
S188	卫星出入境标识	0 境内 1 境外	
S157	PK13 源包使能禁止状态	1	0

(2) 测试参数类型

测试参数表征了某一个组件或者系统所处的实际工作状态.参数名用于参数的标识,参数值表示参数处于的实际工作状态.由于参数含义在指标中给出,故此处可省略.安全苛刻系统测试中,测试参数既有被测产品状态参数、测试设备状态参数,还有其他参数形式.为了区分这些不同的参数,测试参数 *Param* 定义为由测试参数标识 *Label* 和参数名组成.

$$Param=(Label\times ParamName).$$

Label 可为整型或者其他类型,用于标识该参数所属类, *ParamName* 为测试参数名.例如,被测产品状态参数可以表示为 *PParam*=(“*Prod*”,*ParamName*),测试设备状态参数 *DParam*=(“*Dev*”,*ParamName*)以及其他状态参数 *SParam*=(“*Other*”,*ParamName*)等.例:在某航天器测试中,测试参数名以”S”开头命名的为测试设备状态参数,参数 S022 代表测试设备计算机 *A* 机供电+5.3 检测.当 S022 的值为 0 时,表示计算机 *A* 机没有供 5.3V 电压;当 S022>4.5 时,表示计算机 *A* 机供电 5.3V.

(3) 测试指令类

在安全苛刻系统测试中,通过发送指令,测试人员根据被测产品的状态参数改变情况判断被测产品是否响应并正常执行该指令,从而判断被测产品的某一组件或系统的某一功能是否正常.测试指令分为测试设备指令和被测产品指令两类:被测产品指令的执行将改变自身状态参数值;测试设备指令的执行在改变自身状态参数值的同时,还将产生激励信号改变被测产品状态参数.因此,指令 *OpId* 由指令由标识 *Label* 和指令名 *OpName* 组成,其形式表示为

$$OpId=(Lable\times OpName).$$

指令标识用于区分指令,指令名用于标识指令.这样,被测产品的指令操作可表示为 *ProdOpId*=(“*Prod*”,*OpName*),测试设备指令可表示为 *DevOpId*=(“*Dev*”,*OpName*).例:在某航天器综合测试中,*SETP*(20)[*DevRequest*(0012,*set*,*r*]表示将标识为 0012 的测试设备参数值初始化,回令为 *r*,等待回令时间为 20s.

以上 3 类数据是安全苛刻系统测试中测试数据的基本内容.在测试语言的执行过程中,可能还会涉及其他形式的数据,如测试流程执行相关数据以及管理数据等.关于此部分的内容见文献[9],在此不再详细论述.

1.2 被测产品类型

在安全苛刻系统测试中,多被测产品并行测试,不同的被测产品,其指令集也不同.被测产品类型定义为

Product=(被测产品标识,被测产品操作指令集).即,*Product*=(*ProdId*,*ProdOpS*).

1.3 测试设备类型

测试设备端的测试设备由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,对测试设备系统进行分层架构.如图 2 所示: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 管理具体一类测试设备,给出具体测试设备的访问路由.

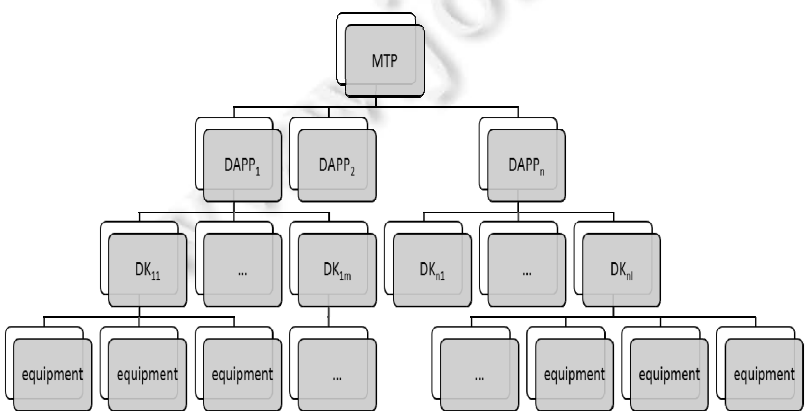

Fig.2 Architecture of test equipment

图 2 测试设备端架构图

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、协同交互与统一访问机制.表 2 为某航天器测试中各种测试设备类示例.

Table 2 Test equipment class in some spacecraft testing

表 2 某航天器测试中各种测试设备类

测试设备类	测试设备类名
频率计	Cymometer
模块化电源	DCModule
程控直流电源	DCPower
电子负载	Load
逻辑分析仪	LogicAna
…	…
波形发生器	Waveform
信号发生器	Generator

测试设备类型定义为

Dev=(测试设备标识,测试设备类,测试设备操作集,测试设备所属的设备级设备网关).

即 *Dev*=(*DevId*,*DevKind*,*DevOpS*,*DAPPName*).例如,某航天器测试设备频率计设备的部分操作指令集见表 3.

Table 3 Instruction set of a spacecraft cymometer

表 3 某航天器测试设备频率计设备操作指令集

测试类名	设备操作指令名	操作指令参数	…
Cymometer //注释: 频率计	*CymometerInit* //注释:频率计初始化操作.	*int CymometerInit*(*ViRsrc resourceName*, *ViSession*pnHandle*,*char*pErrMsg*); //注释: 输入参数描述: 1. *resourceName*,指定将要初始化的仪表的程控接口地址.例如,“GPIB0::22” 输出参数描述: 1. *pnHandle*,返回仪表控制句柄. 2. 返回状态:0,成功;<0,出现错误. 3. *pErrMsg*,如果出现错误,保存当前错误信息.	…
	CymometerReset //注释:频率计复位操作.	*int CymometerReset*(*ViSession nHandle*,*char*pErrMsg*); //注释: 输入参数描述: 1. *nHandle*,仪表控制句柄. 输出参数描述: 1. 返回状态:0,成功;<0,出现错误. 2. *pErrMsg*,如果出现错误,保存当前错误信息.	…
	CymometerConfigActiveChannel //注释:多端口仪表设置当前使用通道操作.	*int CymometerConfigActiveChannel*(*ViSession nHandle*, *int nChannel*,*char*pErrMsg*); //注释: 输入参数描述: 1. *nHandle*,仪表程控句柄. 2. *nChannel*,通道号:1,通道 1;2,通道 2. 输出参数描述: 1. *ErrMsg*,如果出现错误,保存错误提示信息. 2. 返回状态:0,成功;<0,出现错误.	…
	…	…	
	get_TD_infos //注释:取得测试驱动程序缺省配置信息操作.	*void get_TD_infos*(*int*pnInstrType*,*unsigned char*pInstrName*, *int*pnInterfaceType*,*int*pnAddress*); //注释: 输入参数描述:无 输出参数描述: 1. *pnInstType*,当前 TD 所对应的仪表类型. 2. *pInstName*,当前 TD 所对应的仪表名称. 3. *pnInterfaceType*,当前 TD 所对应的仪表接口所属的类别. 4. *pnAddress*,当前 TD 所对应的仪表缺省逻辑地址.	

基于上述类型定义,在安全苛刻系统测试语言中,就可以声明关于被测产品、测试设备的变量或常量,其语法格式与常用的声明格式一致.

2 测试设备协同语句

在安全苛刻系统测试语言中,设备协同语句主要是由关于测试设备协同表达式构成.下面首先介绍设备协同表达式,在此基础上给出设备协同语句的语法定义.

2.1 设备协同表达式

在安全苛刻系统测试中,依赖专业测试人员根据相关业务逻辑编写测试用例,不同的被测产品涉及不同的业务知识,测试方法也不同.安全苛刻系统是复杂系统,在测试过程中,需对被测产品进行分级测试,使得对该产品的测试用例按照产品的功能部件逐渐细化,最终分解为相对独立的最小测试单元集合.这些测试单元在功能上具有一定完整性,在测试过程中具有高复用性的特点,称为原子测试任务,简称测试原子.测试原子内部封装了基本测试设备协同行为.

一般情况下,测试设备协同原子(简称测试原子)具有前置条件,当前置条件满足时,测试原子表达式才能被求值.同时,由于实时性要求,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,测试原子表达式可抽象表示(即,测试原子封装规范)如下:

$$Atom=Precond\rightarrow AName(Time\text{-}Restriction)[TestP].$$

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;*Time-Restriction* 为测试原子执行时间约束;*TestP* 为测试原子内部封装的基本测试设备协同行为,这些基本测试设备协同行为定义如下:

$$TestP::=DevRequest|DevData|DevColGuard|DevColData|Judge(Num\times Val)|Wait(Num).$$

其中,

(1) 设备请求基本表达式 *DevRequest* 定义为

DevRequest::=*DevG*(设备名,设备操作标识,参数值,回令,被测产品标识).

例如 *DevG*(*d*,*op*,*vs*,*ack*,*pid*),其含义是:请求测试设备 *d* 执行操作 *op*,参数为 *vs*,执行结果将向被测产品 *pid* 发送激励信号,回令为 *ack*.

(2) 设备数据请求基本表达式 *DevData* 定义为

DevData::=*DevV*(设备名,参数名,变量名,回令).

例如 *DevV*(*d*,*pn*,*x*,*ack*),表示测试任务向设备 *d* 请求其状态参数 *pn* 的值并存于变量 *x* 中.

(3) 设备指令交互基本表达式 *DevColGuard* 定义为

DevColG(设备名,被测产品标识,操作指令标识,参数值,回令).

例如 *DevColG*(*d*,*prod*,*op*,*vs*,*ack*),表示通过设备 *d* 向被测产品 *prod* 发送指令 *op*,参数为 *vs*.

(4) 设备数据交互基本表达式 *DevColData* 定义为

DevColV(设备名,被测产品标识,参数名,变量名,回令).

例如 *DevColV*(*d*,*prod*,*pn*,*x*,*ack*),表示通过设备 *d* 向被测产品 *prod* 请求其状态参数 *pn* 的值存于 *x* 中,*ack* 为原语执行回令,“真”表示交互成功,“假”为失败.

(5) 判参表达式 *Judge*(参数名,参数值),含义是判断该参数值是否符合标准.例如,*Judge*(*pn*,*pv*)表示判读值 *pv* 是否在参数 *pn* 的标准值范围内.

(6) 时间等待表达式 *Wait*(时间区间值),含义是等待一段时间.例如,*Wait*(*n*)表示等待 *n* 个时间段.

2.2 设备协同语句

设备协同语句由测试协同表达式构成.由于涉及实时约束,当由设备协同原子复合构成设备协同语句,这些复合操作也受实时约束影响.为此,定义如下具有时间约束的复合操作: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.其中,

- 时间约束串行 $ce1;_t ce2$ 指的是:只有 $ce1$ 的执行满足时间约束,且结果为真时,后续的 $ce2$ 才会被继续求值;否则中止,直接返回结果为假;
- 时间约束并行 $ce1||_t ce2$:当且仅当两个设备协同表达式 $ce1$ 和 $ce2$ 的求值均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 $b\rightarrow ce1,ce2$:当 b 结果为真,表达式求值结果为 $ce1$ 的结果值;否则为 $ce2$ 的结果值;
- 时间约束循环 $\langle b\rightarrow ce\rangle$:求值不仅依赖循环中 b 的结果,而且还与 ce 的结果值有关.当 b 为假或 ce 的结果值为假时,均结束循环求值,返回假的结果值.

这样,设备协同语句 *ColStm* 定义为

$$ColStm::=Skip|Atom|ColExp;_t ColExp|ColExp||_t ColExp|(BoolExp\rightarrow ColExp,ColExp)|(\langle BoolExp\rightarrow ColExp\rangle).$$

其中,*Skip* 表示空表达式.

从安全角度看,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,防止错误的测试操作对被测产品和测试设备的损坏,从而也保证了测试过程的安全.

3 测试设备协同语句的操作语义

本节通过定义测试语言中设备协同表达式的抽象执行机[10],给出测试设备协同表达式的求值规则,严格定义安全苛刻系统自动化测试中测试语言设备协同语句的执行过程.

由于在安全苛刻系统测试中,分别有测试人员、测试设备、被测产品等 3 类实体,测试人员借助测试设备完成对被测产品的功能测试、性能测试和接口测试.根据被测产品的需求描述设计测试任务,通过测试设备对被测产品进行测试,判读获取的测试数据以完成测试任务.在测试程序执行过程中,测试设备协同表达式求值过程通过访问测试设备协同系统驱动测试设备对被测产品进行测试,如图 3 所示.

因此,测试环境 *TestEnv* 可以按照上述架构定义如下:

$$TestEnv=(TEQPTEnv,PRODEnv)$$
$$TEQPTEnv=(DevsOpSet,DevsInfo,DevsState)$$
$$PRODEnv=(ProdsOpSet,ProdsState)$$

测试环境由测试设备环境 *TEQPTEnv* 和被测产品环境 *PRODEnv* 组成;测试设备环境由测试设备操作抽象集 *DevsOpS* 和测试设备状态集 *DevsState* 组成;被测产品环境由被测产品操作集 *ProdsOpSet* 和被测产品状态集 *ProdsState* 组成;测试设备状态集由测试设备及其上的设备状态组成;被测产品状态集由被测产品及其上的产品状态组成.相关定义如下:

$$DevsOpSet=DevKind\rightarrow DevOpS$$
$$ProdsOpSet=ProdId\rightarrow ProdOpS$$
$$DevsState=DevId\rightarrow DevC$$
$$ProdsSate=ProdId\rightarrow ProdC$$
$$DevOp=DVal^*\times DevC\rightarrow DevC\times Stimu$$
$$DevC=DParam\rightarrow DVal$$
$$Stimu=(ProdId\times PParam)\rightarrow PVal$$
$$ProdOp=PVal^*\times ProdC\rightarrow ProdC$$
$$ProdC=PParam\rightarrow PVal$$

通用安全苛刻系统测试语言中的设备协同表达式求值抽象机的状态格局 *MState* 由测试设备协同表达式、外部环境 *ExeEnv* 和测试环境 *TestEnv* 组成,定义为

$$MState=\langle Tescoexp,ExeEnv,TestEnv\rangle.$$

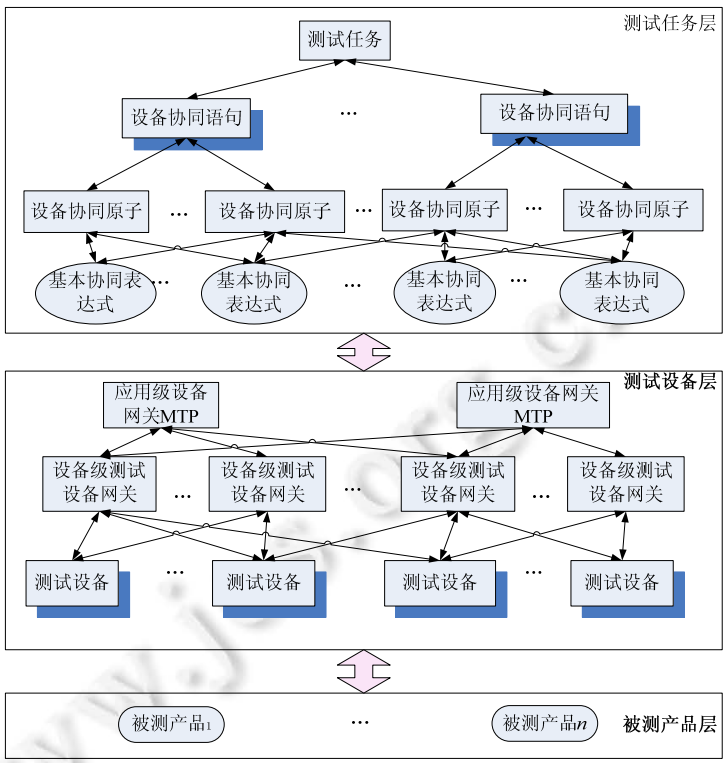

Fig.3 Architecture of testing process of SCS

图 3 安全苛刻系统测试过程架构图

外部环境 $ExeEnv=(Env,Timer)$由一般程序设计语言中语句的执行环境 $Env=Var\to Val$ 和实时时钟 $Timer$ 组成.求值过程还包括一些辅助函数,定义如下:

(1) 时间约束函数

为刻画设备协同原子语句和上述时间约束控制结构的语义,定义时间约束函数监测设备协同原子语句的执行过程.令设备协同原子表达式的求值过程定义为函数 $F=Atom\times Context\times Timer\to Result\times Context$,其中,$Timer$ 为时钟,$Context$ 为设备协同原子的上下文执行环境.时间约束函数 TR 定义为

$$F\times StartTime\times Time\text{-}Restriction\times Timer\times Context\to Result\times Context.$$

其中,$Time\text{-}Restriction$ 为设备协同原子的时间约束,$StartTime$ 为设备协同原子开始执行时间,对 $f\in F,st\in Starttime,tr\in Time\text{-}Restriction,\tau\in Timer,C\in Context,TR(f,st,tr,\tau,C)$定义如下:

(i) 若在 τ 时钟下存在时刻 $t,st\leqslant t<st+tr,f$ 终止,并且执行结果 $f=(r,C')$,则 $TR(f,st,tr,\tau,C)=(r,C')$;

(ii) 否则,$TR(f,st,tr,\tau,C)=(\text{false},C)$.

(2) 激励响应函数

由于测试设备操作改变的不仅仅是测试设备自身状态,而且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,若被测产品激励响应过程定义为函数 H,则 H 定义为 $H=ProdId\times Stimu\times ProcC\to ProcC$.对 $pid\in ProdId,s\in Stimu,\gamma\in ProcC,H(pid,s,\gamma)=\gamma'$.其中,$\gamma'_{pid}=\gamma_{pid}\{s\},\gamma'=(\gamma_1,...,\gamma'_{pid},...,\gamma_n)$,$\{\cdot\}$为置换操作.即,根据激励更新指定被测产品的状态参数值.

(3) 协同协议检查函数

基于安全考虑,在协同执行时还要定义协议检查函数,用于检测设备协同任务中的协同操作是否符合协议规范,防止未定义的操作对被测产品带来破坏.

若定义协议检查函数 C:$Dev\times Op\times Params\times OpSet\to BOOL$,对 $d\in Dec,f\in Op,ps\in Params,\chi\in OpSet$,$C(d,f,ps,\chi)$=true,即,接入设备 d 的操作集中有符合 f、参数 ps 的出现;否则,$C(d,f,ps,\chi)$=false.

设 $\sigma\in Env,\tau\in Timer,\chi\in DevsOpSet,\delta\in ProdsOpSet,\rho\in DevsState,\gamma\in ProdsState$,定义符号"⇒"为抽象机转换.设 $time$ 为时钟 τ 的系统取时变量,以下给出设备协同表达式的求值规则.

3.1 设备协同表达式的求值规则

根据设备的分层结构,设备协同表达式的求值分为 3 层,分别是设备协同任务层、设备层和被测产品层.设备协同表达式的求值分为设备协同原子的求值和其封装的基本设备协同表达式的求值.

当设备协同原子表达式的前置条件满足时,设备协同原子在时间约束函数 TR 的作用下,开始设备协同原子内部基本设备表达式的求值.对设备协同原子 $atom=p\to a(rt)[testp]$,上下文 $Context=(ExeEnv,TestEnv)$.当前置条件不满足时,返回结果为 false,上下文不变.

$$\langle p\to a(rt)[abody],(\sigma,\tau),(\chi,\rho),(\delta,\gamma)\rangle\Rightarrow$$
$$\langle p,(\sigma,\tau)\rangle\to TR(\langle abody,(\sigma,\tau),(\chi,\rho),(\delta,\gamma)\rangle,\tau(time),rt,(\sigma,\tau),(\chi,\rho),(\delta,\gamma)),$$
$$\langle \text{false},(\sigma,\tau),(\chi,\rho),(\delta,\gamma)\rangle.$$

其中,$abody$ 为该设备协同原子内部的基本设备协同表达式,其求值规则如下:

(1) 基本设备协同求值规则

① 判参规则 $Judge(v,\upsilon)$:判断状态参数 v 的值是否符合 υ.υ 可以表征某一范围值或者某一具体值.根据文献[11],其判断都可以归为等式逻辑的判断操作,故用 $\sigma(v)=?\upsilon$ 形式表示状态参数 v 是否符合 υ.

$$\langle Judge(v,\upsilon),(\sigma,\tau),(\chi,\rho),(\delta,\gamma)\rangle\Rightarrow\langle\sigma(v)=?\upsilon,(\sigma,\tau),(\chi,\rho),(\delta,\gamma)\rangle.$$

② 等待规则 $Wait(n)$:等待时间 n:

$$\langle Wait(n),(\sigma,\tau),(\chi,\rho),(\delta,\gamma)\rangle\Rightarrow\langle\kappa(n,\tau(time),\tau),(\sigma,\tau),(\chi,\rho),(\delta,\gamma)\rangle,$$
$$\kappa(n,ct,\tau)=\tau(time)-ct<n\to\kappa(n,ct,\tau),\text{true}.$$

③ 设备请求规则 $DevG(d,g,as,ack,pid)$:请求测试设备操作,首先检查是否符合协议:不符合则返回回令 false;若符合协议,则请求指定某类设备转到设备层求值.

$$\frac{C(d,g,as,\chi)\to\text{false}}{\langle DevG(d,g,as,ack,pid),(\sigma,\tau),(\chi,\rho),(\delta,\gamma)\rangle\Rightarrow\langle\text{false},(\sigma,\tau),(\chi,\rho),(\delta,\gamma)\rangle}$$
$$\frac{C(d,g,as,\chi)\to\text{true}}{\langle DevG(d,g,as,ack,pid),(\sigma,\tau),(\chi,\rho),(\delta,\gamma)\rangle\Rightarrow(\text{true},(\sigma\{\text{true}/ack\},\tau),\langle(g,as,pid),(\chi,\rho),(\delta,\gamma)\rangle^d)}$$

④ 设备访问规则 $DevV(d,di,pn,v,ack)$:返回回令.转到设备层求值,以获取该参数值.

$$\langle DevV(d,di,pn,v,ack),(\sigma,\tau),(\chi,\rho),(\delta,\gamma)\rangle\Rightarrow(\text{true},(\sigma\{\text{true}/ack,w/v\},\tau),(\chi,\rho),(\delta,\gamma)),$$

其中,$(w,(\chi,\rho),(\delta,\gamma))=\langle(di,pn),(\chi,\rho),(\delta,\gamma)\rangle^d$.

⑤ 设备指令交互规则 $DevColG(d,pid,g,as,ack)$:通过测试设备将被测产品的指令传给被测产品.首先检查协议是否符合,不符合则返回 false 回令,符合则转到设备层继续求值.

$$\frac{C(pid,g,as,\delta)\to\text{false}}{\langle DevColG(d,pid,g,as,ack),(\sigma,\tau),(\chi,\rho),(\delta,\gamma)\rangle\Rightarrow\langle\text{false},(\sigma,\tau),(\chi,\rho),(\delta,\gamma)\rangle}$$
$$\frac{C(pid,g,as,\delta)\to\text{true}}{\langle DevColG(d,pid,g,as,ack),(\sigma,\tau),(\chi,\rho),(\delta,\gamma)\rangle\Rightarrow(\text{true},(\sigma\{\text{true}/ack\},\tau),\langle(pid,g,as),(\chi,\rho),(\delta,\gamma)\rangle^d)}$$

⑥ 设备数据交互规则 $DevColV(d,pid,pn,v,ack)$:返回回令后,转到设备层继续求值.

$$\langle DevColV(d,pid,pn,v,ack),(\sigma,\tau),(\chi,\rho),(\delta,\gamma)\rangle\Rightarrow(\text{true},(\sigma\{\text{true}/ack,w/v\},\tau),(\chi,\rho),(\delta,\gamma)),$$

其中,$(w,(\chi,\rho),(\delta,\gamma))=\langle(pid,pn),(\chi,\rho),(\delta,\gamma)\rangle^d$.

(2) 设备层求值规则

① 设备请求规则:从该类设备 d 中选择合适测试设备执行该指令.指令的执行将改变测试设备的状态,同时产生对被测产品的激励信号 H,转到被测产品层求值以相应激励.

$$\langle(g,as,pid),(\chi,\rho),(\delta,\gamma)\rangle^{d}\Rightarrow((\chi,\rho'),\langle s,\delta,\gamma\rangle^{pid}),$$

其中,$di=lookupid(d)$,$(s,\rho'_{di})=g(as,\rho_{di})$.

② 设备访问规则:访问某类测试设备 dk 中测试设备标识为 id 的状态参数 pn 的值存于 v 中.

$$\langle(di,pn),(\chi,\rho),(\delta,\gamma)\rangle^{d}\Rightarrow(\rho_{di}(pn),(\chi,\rho),(\delta,\gamma)).$$

③ 设备指令交互规则:该类设备中,选择合适的测试设备传送该指令给被测产品;转到被测产品层继续求值.

$$\langle(pid,g,as),(\chi,\rho),(\delta,\gamma)\rangle^{d}\Rightarrow((\chi,\rho),\langle(g,as),(\delta,\gamma)\rangle^{pid}),$$

其中,$di=lookupid(d)$.

④ 设备数据交互规则:通过测试设备 d 访问被测产品.

$$\langle(pid,pn),(\chi,\rho),(\delta,\gamma)\rangle^{d}\Rightarrow((\chi,\rho),\langle(pn,\delta,\gamma)\rangle^{pid}).$$

(3) 被测产品层求值规则

① 设备请求规则:被测产品响应激励改变自身状态.

$$\langle s,(\delta,\gamma)\rangle^{pid}\Rightarrow(\delta,H(pid,s,\gamma)).$$

② 设备指令交互规则:被测产品接收到指令后执行该指令,并改变自身状态.

$$\langle(g,as),(\delta,\gamma)\rangle^{pid}\Rightarrow(\delta,\gamma'_{pid}),\gamma'_{pid}=(g(as),\gamma_{pid}).$$

③ 设备数据交互规则:被测产品的状态参数 pn 的值存于 v 中.

$$\langle(pn,\delta,\gamma)\rangle^{pid}\Rightarrow(\gamma_{pid}(pn),\delta,\gamma).$$

3.2 设备协同语句的语义规则

基于上述设备协同表达式的求值规则,给出如下设备协同语句的操作语义规则:

① 空规则:

$$\langle skip,(\sigma,\tau),(\chi,\rho),(\delta,\gamma)\rangle\Rightarrow\langle\text{true},(\sigma,\tau),(\chi,\rho),(\delta,\gamma)\rangle.$$

② 时间约束串行";$_t$"规则:对两个设备协同语句的时间约束串行,先执行第 1 个语句,当其返回结果为真时,继续后面的语句;否则直接返回.

$$\frac{\langle coe1,(\sigma,\tau),(\chi,\rho),(\delta,\gamma)\rangle\Rightarrow\langle\text{true},(\sigma',\tau),(\chi,\rho'),(\delta,\gamma')\rangle}{\langle coe1;_t col2,(\sigma,\tau),(\chi,\rho),(\delta,\gamma)\rangle\Rightarrow\langle coe2,(\sigma',\tau),(\chi,\rho'),(\delta,\gamma')\rangle}$$

$$\frac{\langle coe1,(\sigma,\tau),(\chi,\rho),(\delta,\gamma)\rangle\Rightarrow\langle\text{false},(\sigma,\tau),(\chi,\rho),(\delta,\gamma)\rangle}{\langle coe1;_t coe2,(\sigma,\tau),(\chi,\rho),(\delta,\gamma)\rangle\Rightarrow\langle\text{false},(\sigma,\tau),(\chi,\rho),(\delta,\gamma)\rangle}.$$

③ 时间约束并行"$\|_t$"规则:对两个设备协同语句的时间约束并行,只有当两个语句的执行返回结果均为真时返回成功;否则返回失败.

$$\frac{\langle coe1,(\sigma,\tau),(\chi,\rho),(\delta,\gamma)\rangle\Rightarrow\langle\text{true},(\sigma',\tau),(\chi,\rho'),(\delta,\gamma')\rangle}{\langle coe1\|_t coe2,(\sigma,\tau),(\chi,\rho),(\delta,\gamma)\rangle\Rightarrow\langle coe2,(\sigma',\tau),(\chi,\rho'),(\delta,\gamma')\rangle}$$

$$\frac{\langle coe1,(\sigma,\tau),(\chi,\rho),(\delta,\gamma)\rangle\Rightarrow\langle\text{false},(\sigma,\tau),(\chi,\rho),(\delta,\gamma)\rangle}{\langle coe1\|_t coe2,(\sigma,\tau),(\chi,\rho),(\delta,\gamma)\rangle\Rightarrow\langle\text{false},(\sigma,\tau),(\chi,\rho),(\delta,\gamma)\rangle},$$

$$\frac{\langle coe1,(\sigma,\tau),(\chi,\rho),(\delta,\gamma)\rangle\Rightarrow\langle\text{true},(\sigma1,\tau),(\chi,\rho1),(\delta,\gamma1)\rangle,\langle coe2,(\sigma,\tau),(\chi,\rho),(\delta,\gamma)\rangle\Rightarrow\langle\text{true},(\sigma2,\tau),(\chi,\rho2),(\delta,\gamma2)\rangle}{\langle coe1\|_t coe2,(\sigma,\tau),(\chi,\rho),(\delta,\gamma)\rangle\Rightarrow\langle\text{true},(\sigma1,\tau),(\sigma1\oplus\sigma2,\tau),(\chi,\rho1\oplus\rho2),(\delta,\gamma1\oplus\gamma2)\rangle}$$

其中,$\oplus=\lambda xy.(x\cap y)\bullet(x-(x\cap y))\bullet(y-(x\cap y))$.

④ 时间约束选择$(b\rightarrow coe1,coe2)$规则:时间约束选择同一般语言中的选择语句.

$$\frac{\sigma(b)\Rightarrow\text{true}}{\langle(b\rightarrow coe1,coe2),(\sigma,\tau),(\chi,\rho),(\delta,\gamma)\rangle\Rightarrow\langle coe1,(\sigma,\tau),(\chi,\rho),(\delta,\gamma)\rangle},$$

$$\frac{\sigma(b)\Rightarrow\text{false}}{\langle(b\rightarrow coe1,coe2),(\sigma,\tau),(\chi,\theta,\rho),(\delta,\gamma)\rangle\Rightarrow\langle coe2,(\sigma,\tau),(\chi,\rho),(\delta,\gamma)\rangle}.$$

⑤ 时间约束循环$(\langle b\rightarrow cole\rangle)$规则:时间约束循环与一般语言中的循环语句不同的是:当入口条件满足进

入循环体时,若循环体的执行返回成功则继续;否则循环终止,返回失败.

$$\frac{\sigma(b)\Rightarrow \text{true},\langle coe,(\sigma,\tau),(\chi,\rho),(\delta,\gamma)\rangle\Rightarrow\langle \text{true},(\sigma',\tau),(\chi,\rho'),(\delta,\gamma')\rangle}{\langle(\langle b\rightarrow coe\rangle),(\sigma,\tau),(\chi,\rho),(\delta,\gamma)\rangle\Rightarrow\langle\langle b\rightarrow coexp\rangle,(\sigma',\tau),(\chi,\rho'),(\delta,\gamma')\rangle}$$

$$\frac{\sigma(b)\Rightarrow \text{false}}{\langle(b\rightarrow cole),(\sigma,\tau),(\chi,\rho),(\delta,\gamma)\rangle\Rightarrow\langle \text{false},(\sigma,\tau),(\chi,\rho),(\delta,\gamma)\rangle}$$

$$\frac{\langle coe,(\sigma,\tau),(\chi,\rho),(\delta,\gamma)\rangle\Rightarrow\langle \text{false},(\sigma,\tau),(\chi,\rho),(\delta,\gamma)\rangle}{\langle(\langle b\rightarrow cole\rangle),(\sigma,\tau),(\chi,\rho),(\delta,\gamma)\rangle\Rightarrow\langle \text{false},(\sigma,\tau),(\chi,\rho),(\delta,\gamma)\rangle}$$

4 设备协同语句语义的正确性证明

由于安全苛刻系统系统自动化测试具有时间约束,扩展标记转换系统,记 $\xrightarrow{\alpha}_{\iota}$ 表示实时条件转换,ι 为实际执行时间,α 为归约标记. $L\xrightarrow{\alpha}_{\iota}R$ 表示转换左部 L 执行动作 α,经过时间 ι 后,归约为右部 R. $\xrightarrow{\alpha}_{\iota}{}^{*}$表示多步归约.归约标记 α::=$\varepsilon|Cop$,其中,ε 为哑动作,Cop 为协同动作.Cop 定义为

$$Cop::=a[d!(f,as,pid)]|a[d?(p,v)]|a[d!(pid,g,as)]|a[d?(pid,pn,v)],$$

其中,$a[d!(f,as,r,pid)]$表示设备协同原子 a 发生了向测试设备 d 发送指令 f、参数为 as 的 c 设备访问动作;$a[d?(di,p,v)]$表示设备协同原子 a 发生了向测试设备 d 中的具体测试设备 di 请求参数 p 的值存于 v 中的设备访问动作;$a[d!(pid,g,as)]$表示设备协同原子 a 发生了通过测试设备 d 向被测产品 pid 发送指令 g、参数为 as 的设备指令协同动作;$a[d?(pid,pn,v)]$表示设备协同原子 a 发生了通过测试设备 d,向被测产品 pid 请求参数 pn 的值 v 的设备数据协同动作.

在此基础上,给出设备协同任务的标记转换系统:

(1) 设备请求规则:请求设备响应发送的指令.

$$\langle a(T)[DevG(d,f,as,ack,pid)],(\sigma,\tau),(\chi,\rho),(\delta,\gamma)\rangle\xrightarrow{a[d!(f,as,pid)]}_{\iota}$$
$$\begin{cases}\langle \text{true},(\sigma\{\text{true}/ack\},\tau),(\chi,\rho),(\delta,\gamma')\rangle, & \text{if } \iota<T, \text{ where } \gamma'=H(pid,s,\gamma),(s,\rho)=f(as,\rho)\\ \langle \text{false},(\sigma,\tau),(\chi,\rho),(\delta,\gamma)\rangle, & \text{otherwise}\end{cases}.$$

(2) 设备访问规则.

$$\langle a(T)[DevV(d,di,pn,v,ack)],(\sigma,\tau),(\chi,\rho),(\delta,\gamma)\rangle\xrightarrow{a[d?(di,dn,\upsilon)]}_{\iota}$$
$$\begin{cases}\langle \text{true},(\sigma\{(\text{true}/ack),(\upsilon/v)\},\tau),(\chi,\rho),(\delta,\gamma)\rangle, & \text{if } \iota<T\\ \langle \text{false},(\sigma,\tau),(\chi,\rho),(\delta,\gamma)\rangle, & \text{otherwise}\end{cases}.$$

(3) 设备协同指令规则:设备与被测产品间的指令协同,通过设备向被测产品发送指令.

$$\langle a(T)[DevColG(d,pid,g,as,ack)],(\sigma,\tau),(\chi,\rho),(\delta,\gamma)\rangle\xrightarrow{a[d!(pid,g,as)]}_{\iota}$$
$$\begin{cases}\langle \text{true},(\sigma\{\text{true}/ack\},\tau),(\chi,\rho),(\delta,\gamma')\rangle, & \text{if } \iota<T, \text{ where } \gamma'_{pid}=(g(as),\gamma_{pid})\\ \langle \text{false},(\sigma,\tau),(\chi,\rho),(\delta,\gamma)\rangle, & \text{otherwise}\end{cases}.$$

(4) 设备协同数据规则:设备与被测产品间的数据协同,通过设备获取被测产品状态信息.

$$\langle a(T)[DevColV(d,pid,pn,v,ack)],(\sigma,\tau),(\chi,\rho),(\delta,\gamma)\rangle\xrightarrow{a[d!(pid,p,\upsilon)]}_{\iota}$$
$$\begin{cases}\langle \text{true},(\sigma\{(\text{true}/ack),(\gamma_{pid}(pn)/v)\},\tau),(\chi,\rho),(\delta,\gamma)\rangle, & \text{if } \iota<T\\ \langle \text{false},(\sigma,\tau),(\chi,\rho),(\delta,\gamma)\rangle, & \text{otherwise}\end{cases}.$$

以下为复合操作规则,复合操作规则的含义见第 3.2 节.对$\Gamma\in Context$:

(5) 时间约束串行复合操作规则.

$$\frac{\langle coe1,\Gamma\rangle\xrightarrow{\alpha}_{\iota}\langle \text{true},\Gamma'\rangle}{\langle coe1;_{t}\ coe2,C\rangle\xrightarrow{\alpha}_{\iota}\langle coe2,C'\rangle}.$$

$$\frac{\langle coe1,\Gamma\rangle\xrightarrow{\alpha}_{\iota}\langle \text{false},\Gamma\rangle}{\langle coe1;_{t}\ coe2,\Gamma\rangle\xrightarrow{\alpha}_{\iota}\langle \text{false},\Gamma\rangle}.$$

(6) 时间约束并行复合操作规则.

$$\frac{\langle coe1,\Gamma\rangle \xrightarrow{\alpha}_t \langle \text{true},\Gamma'\rangle}{\langle coe1 \|_t\ coe2,C\rangle \xrightarrow{\alpha}_t \langle coe2,C'\rangle}$$

$$\frac{\langle coe1,\Gamma\rangle \xrightarrow{\alpha}_t \langle \text{false},\Gamma\rangle}{\langle coe1 \|_t\ coe2,C\rangle \xrightarrow{\alpha}_t \langle \text{false},C\rangle}$$

$$\frac{\langle coe2,\Gamma\rangle \xrightarrow{\alpha}_t \langle \text{true},\Gamma'\rangle}{\langle coe1 \|_t\ coe2,C\rangle \xrightarrow{\alpha}_t \langle coe1,C'\rangle}$$

$$\frac{\langle coe2,\Gamma\rangle \xrightarrow{\alpha}_t \langle \text{false},\Gamma\rangle}{\langle coe1 \|_t\ coe2,C\rangle \xrightarrow{\alpha}_t \langle \text{false},C\rangle}$$

(7) 时间约束选择复合操作规则.

$$\frac{\Gamma(b)=\text{true},\langle coe1,\Gamma\rangle \xrightarrow{\alpha}_t \langle r,\Gamma'\rangle}{\langle (b\to coe1,coe2,\Gamma)\rangle \xrightarrow{\alpha}_t \langle r,\Gamma'\rangle}$$

$$\frac{\Gamma(b)=\text{false},\langle coe2,\Gamma\rangle \xrightarrow{\alpha}_t \langle r,\Gamma'\rangle}{\langle (b\to coe1,coe2,\Gamma)\rangle \xrightarrow{\alpha}_t \langle r,\Gamma'\rangle}$$

(8) 时间约束循环复合操作规则.

$$\frac{\Gamma(b)=\text{true},\langle coe,\Gamma\rangle \xrightarrow{\alpha}_t \langle \text{true},\Gamma'\rangle}{\langle (\langle b\to coe\rangle,\Gamma)\rangle \xrightarrow{\alpha}_t \langle \text{true},\Gamma'\rangle}$$

$$\frac{\Gamma(b)=\text{false}}{\langle (\langle b\to coe\rangle,\Gamma)\rangle \xrightarrow{\varepsilon}_t \langle \text{false},\Gamma\rangle}$$

$$\frac{\langle coe,\Gamma\rangle \xrightarrow{\alpha}_t \langle \text{false},\Gamma\rangle}{\langle (\langle b\to coe\rangle,\Gamma)\rangle \xrightarrow{\alpha}_t \langle \text{false},\Gamma'\rangle}$$

容易证明,有如下引理成立:

引理 1. 对任意设备协同语句式 s 和上下文环境Γ,若$\langle s,\Gamma\rangle \Rightarrow^* \langle R,\Gamma'\rangle$,则 $\langle s,\Gamma\rangle \xrightarrow{\varepsilon}_t{}^* \langle r,\Gamma'\rangle$.

证明:首先证明测试设备协同原子满足结论.根据定义,测试设备协同原子为 $pc\to a(rt)[tp]$,其求值为

$$\langle p\to a(rt)[abody],\Gamma\rangle \Rightarrow \langle p,(\sigma,\tau)\rangle \to TR(\langle abody,\Gamma\rangle,\tau(time),rt,\Gamma),\langle \text{false},\Gamma\rangle,$$

其中,$abody$ 为基本设备协同表达式.当前置条件不满足时,测试设备协同原子不执行,返回结果为 false,满足结论.由于判断参数规则和等待规则不涉及设备协同行为,故结论显然成立.因此,以下主要证明当前置条件为真时,带有设备协同行为的测试原子的执行过程满足结论.对测试原子中的基本设备协同表达式分情形证明.

(1) 设备请求时,即,$abody=DevG(d,g,as,ack,pid)$时:

$$\langle a(rt)[DevG(d,g,as,ack,pid],\Gamma\rangle \Rightarrow TR(\langle DevG(d,g,as,ack,pid,\Gamma\rangle,\tau(time),rt,\Gamma).$$

当不满足时间约束时,根据时间约束函数 TR 的定义,直接返回 false;当满足时间约束时,当操作不符合协议时,根据表达式求值规则,则该请求不被响应,直接返回失败.即:

$$\langle DevG(d,g,as,ack,pid,\Gamma\rangle \Rightarrow \langle \text{false},\Gamma\rangle,$$

则有:

$$\langle a(rt)[DevG(d,g,as,ack,pid],\Gamma\rangle \Rightarrow^* \langle \text{false},\Gamma\rangle.$$

当操作符合协议时,有:

$$\begin{aligned}\langle DevG(d,g,as,ack,pid),(\sigma,\tau),(\chi,\rho),(\delta,\gamma)\rangle &\Rightarrow (\text{true},(\sigma\{\text{true}/ack\},\tau),\langle (g,as,pid),(\chi,\rho),(\delta,\gamma)\rangle^d)\\ &\Rightarrow (\text{true},(\sigma\{\text{true}/ack\},\tau),((\chi,\rho),\langle s,\delta,\gamma\rangle^{pid}))\\ &\quad \text{其中},di=lookupid(d),(s,\rho'_{di})=g(as,\rho_{di})\\ &\Rightarrow (\text{true},(\sigma\{\text{true}/ack\},\tau),((\chi,\rho),(\delta,H(pid,s,\gamma))))\end{aligned},$$

则有:

$$\langle a(rt)[DevG(d,g,as,ack,pid],(\sigma,\tau),(\chi,\rho),(\delta,\gamma)\rangle \Rightarrow^* \langle \text{true},(\sigma\{\text{true}/ack\},\tau),((\chi,\rho),(\delta,H(pid,s,\gamma)))\rangle.$$

综上,根据第 3 节的规则(1)设备请求规则,均满足.

(2) 设备访问时,即,当 $abody=DevV(d,di,pn,v,ack)$时:

$$\langle a(rt)[DevG(d,g,as,ack,pid],\Gamma\rangle \Rightarrow TR(\langle DevV(d,di,pn,v,ack),\Gamma\rangle,\tau(time),rt,\Gamma).$$

同样地,当不满足时间约束时,根据时间约束函数 TR 的定义,直接返回 false;当满足时间约束时:

$$\langle DevV(d,di,pn,v,ack),(\sigma,\tau),(\chi,\rho),(\delta,\gamma)\rangle \Rightarrow (\text{true},(\sigma\{\text{true}/ack,w/v\},\tau),(\chi,\rho),(\delta,\gamma)),$$

其中,$(w,(\chi,\rho),(\delta,\gamma))=\langle(di,pn),(\chi,\rho),(\delta,\gamma)\rangle^{d}$.

而根据$\langle(di,pn),(\chi,\rho),(\delta,\gamma)\rangle^{d}=(\rho_{di}(pn),(\chi,\rho),(\delta,\gamma))$,则有:

$$\langle a(rt)[DevG(d,g,as,ack,pid],(\sigma,\tau),(\chi,\rho),(\delta,\gamma)\rangle \Rightarrow (\text{true},(\sigma\{\text{true}/ack,\rho_{di}(pn)/v\},\tau),(\chi,\rho),(\delta,\gamma).$$

综上,根据第 3 节的规则(2),满足.

(3) 设备指令交互时,即,当 $abody=DevColG(d,pid,g,as,ack)$时:

$$\langle a(rt)[DevColG(d,pid,g,as,ack)],\Gamma\rangle \Rightarrow TR(\langle DevColG(d,pid,g,as,ack),\Gamma\rangle,\tau(time),rt,\Gamma).$$

当不满足时间约束时,根据时间约束函数 TR 的定义,直接返回 false;当满足时间约束时,当操作不符合协议时,根据表达式求值规则,则该请求不被响应,直接返回失败.即:

$$\langle DevColG(d,pid,g,as,ack),\Gamma\rangle \Rightarrow \langle \text{false},\Gamma\rangle,$$

则有:

$$\langle a(rt)[DevG(d,g,as,ack,pid],\Gamma\rangle \Rightarrow^{*} \langle \text{false},\Gamma\rangle.$$

当操作符合协议时,有:

$$\begin{aligned}\langle DevColG(d,pid,g,as,ack),(\sigma,\tau),(\chi,\rho),(\delta,\gamma)\rangle &\Rightarrow (\text{true},(\sigma\{\text{true}/ack\},\tau),\langle(pid,g,as),(\chi,\rho),(\delta,\gamma)\rangle^{d})\\ &\Rightarrow (\text{true},(\sigma\{\text{true}/ack\},\tau),((\chi,\rho),\langle(g,as),(\delta,\gamma)\rangle^{pid}))\\ &\quad \text{其中},di=lookupid(d)\\ &\Rightarrow (\text{true},(\sigma\{\text{true}/ack\},\tau),((\chi,\rho),(\delta,\gamma'_{pid}))),\text{其中},\gamma'_{pid}=(g(as),\gamma_{id})\end{aligned},$$

则有:

$$\langle a(rt)[DevG(d,g,as,ack,pid)],(\sigma,\tau),(\chi,\rho),(\delta,\gamma)\rangle \Rightarrow^{*} \langle \text{true},(\sigma\{\text{true}/ack\},\tau),((\chi,\rho),(\delta,\gamma'_{pid}))\rangle.$$

综上,根据的规则(3)设备请求规则,均满足.

(4) 设备数据交互时,即,当 $abody=DevColV(d,pid,pn,v,ack)$时:

$$\langle a(rt)[DevColV(d,pid,pn,v,ack),\Gamma\rangle \Rightarrow TR(\langle DevColV(d,pid,pn,v,ack),\Gamma\rangle,\tau(time),rt,\Gamma).$$

同样地,当不满足时间约束时,根据时间约束函数 TR 的定义,直接返回 false;当满足时间约束时:

$$\langle DevColV(d,pid,pn,v,ack),(\sigma,\tau),(\chi,\rho),(\delta,\gamma)\rangle \Rightarrow (\text{true},(\sigma\{\text{true}/ack,w/v\},\tau),(\chi,\rho),(\delta,\gamma)),$$

其中,$(w,(\chi,\rho),(\delta,\gamma))=\langle(pid,pn),(\chi,\rho),(\delta,\gamma)\rangle^{d}$;

$$\langle(pid,pn),(\chi,\rho),(\delta,\gamma)\rangle^{d} \Rightarrow ((\chi,\rho),\langle(pn,\delta,\gamma)\rangle^{pid}) \Rightarrow (\gamma_{pid}(pn),\delta,\gamma),$$

则有:

$$\langle a(rt)[DevColV(d,pid,pn,v,ack),(\sigma,\tau),(\chi,\rho),(\delta,\gamma)\rangle \Rightarrow (\text{true},(\sigma\{\text{true}/ack,\gamma_{pid}(pn)/v\},\tau),(\chi,\rho),(\delta,\gamma).$$

综上,根据第 3 节的规则(4),满足.

因此,构成设备协同语句的测试设备协同原子的执行是正确的.通过测试设备协同原子上复合操作,包括时间约束串行、时间约束并行、时间约束选择以及时间约束循环构成设备协同语句.根据复合操作的定义,结论显然.证毕. □

引理 1 也说明了设备协同语句语义收敛性和实现的正确性.

引理 2. 对任意设备协同语句 s 和上下文环境Γ,若 $\langle s,\Gamma\rangle \xrightarrow{\varepsilon}_{t}^{*} \langle r,\Gamma'\rangle$,则$\langle s,\Gamma\rangle \Rightarrow^{*} \langle r,\Gamma'\rangle$.

该引理的证明类似引理 1 的证明.引理 2 也说明了设备协同语句实现的可靠性.

根据引理 1 和引理 2,有如下定理:

定理. 对任意设备协同语句 s 和上下文环境Γ,若$\langle s,\Gamma\rangle \Rightarrow^{*} \langle r,\Gamma'\rangle$,当且仅当 $\langle s,\Gamma\rangle \xrightarrow{\varepsilon}_{t}^{*} \langle r,\Gamma'\rangle$.

通过上述证明,说明本文的测试设备协同语句是正确可靠的.

5 测试设备协同应用例

航天器作为典型的安全苛刻系统,以下给出航天器测试的应用实例.基于上述设备协同模型,可设计航天器通用测试语言,实现航天器通用测试系统,该系统分为测试编辑环境、测试执行环境和测试评估环境等 3 个子系统.

测试编辑环境子系统以航天器通用测试语言中的语法为基础,面向测试人员,为其提供组织编写测试任务的用户友好环境.

测试执行环境是测试任务的运行环境,提供了测试任务的解释执行、过程监视、流程控制和结果判别功能.图 4 为航天器测试执行环境组织结构图.该系统分为 3 个层次,分别为测试执行系统接口层、多测试执行引擎管理层、测试执行引擎实例层.此部分以航天器通用测试语言的语义为基础,其中,测试语言中测试语句关键是测试设备协同语句,在测试执行引擎模型中,主要基于测试设备协同语句的操作语义规则,实现了上述测试设备协同语句,封装为测试程序,向上层提供测试测试程序执行结果.

测试执行环境主要由以下几个功能部分组成.

- 测试执行系统接口层

用于外界的接口定义,接收用户请求转发给测试执行引擎管理模块;从测试程序数据库中提取测试程序,提交给多测试执行引擎管理模块,并从测试执行引擎管理模块处接收处理后的测试程序执行结果,并将该结果保存到测试数据库中.

- 多测试执行引擎管理层

用于多测试过程并行执行的管理,多个测试执行引擎实例并行执行多测试程序.在接收新测试程序后,创建并启动新的测试执行引擎实例;在测试执行过程中,监控测试执行引擎实例的状态,确保各测试执行引擎实例处于 Active 状态并运转正常.当某个测试程序执行结束时,销毁相应的测试执行引擎实例,并回收该实例所占用的测试资源.

- 测试执行引擎实例层

测试执行引擎实例在多测试执行引擎管理模块控制下,解释执行测试程序.测试执行引擎实例包括 3 层,分别是测试执行引擎接口层、测试执行引擎管理层和测试执行引擎层:

  - 测试执行引擎接口层是测试执行引擎实例与外界的接口定义层,负责从外界接收测试程序,并将其递交给测试执行引擎层中的测试程序管理模块;从测试执行引擎管理模块处接收用户控制请求,并将请求转发给执行引擎流程控制模块;从执行引擎层中接收处理后的测试程序执行结果,并将该结果递交给测试执行引擎管理模块;
  - 测试执行引擎管理层则管理单个测试执行引擎实例;
  - 测试执行引擎层:测试执行引擎实例每次负责执行一个测试程序.在执行过程中,基于设备协同语句的操作语义规则,将测试设备协同原子以及测试设备协同原子上的复合操作解释执行,完成设备的协同任务,在此基础上调用测试数据与测试资源,完成测试程序;同时接收用户控制命令,实时改变测试执行引擎状态.当测试程序执行结束时,将测试程序结果返回给测试执行引擎管理层.

- 测试评估环境

实现对测试数据、测试内容与测试结果的查询、分析、统计与显示,自动生成各分系统的测试报告,并为测试结果分析和可靠性评估提供支持.

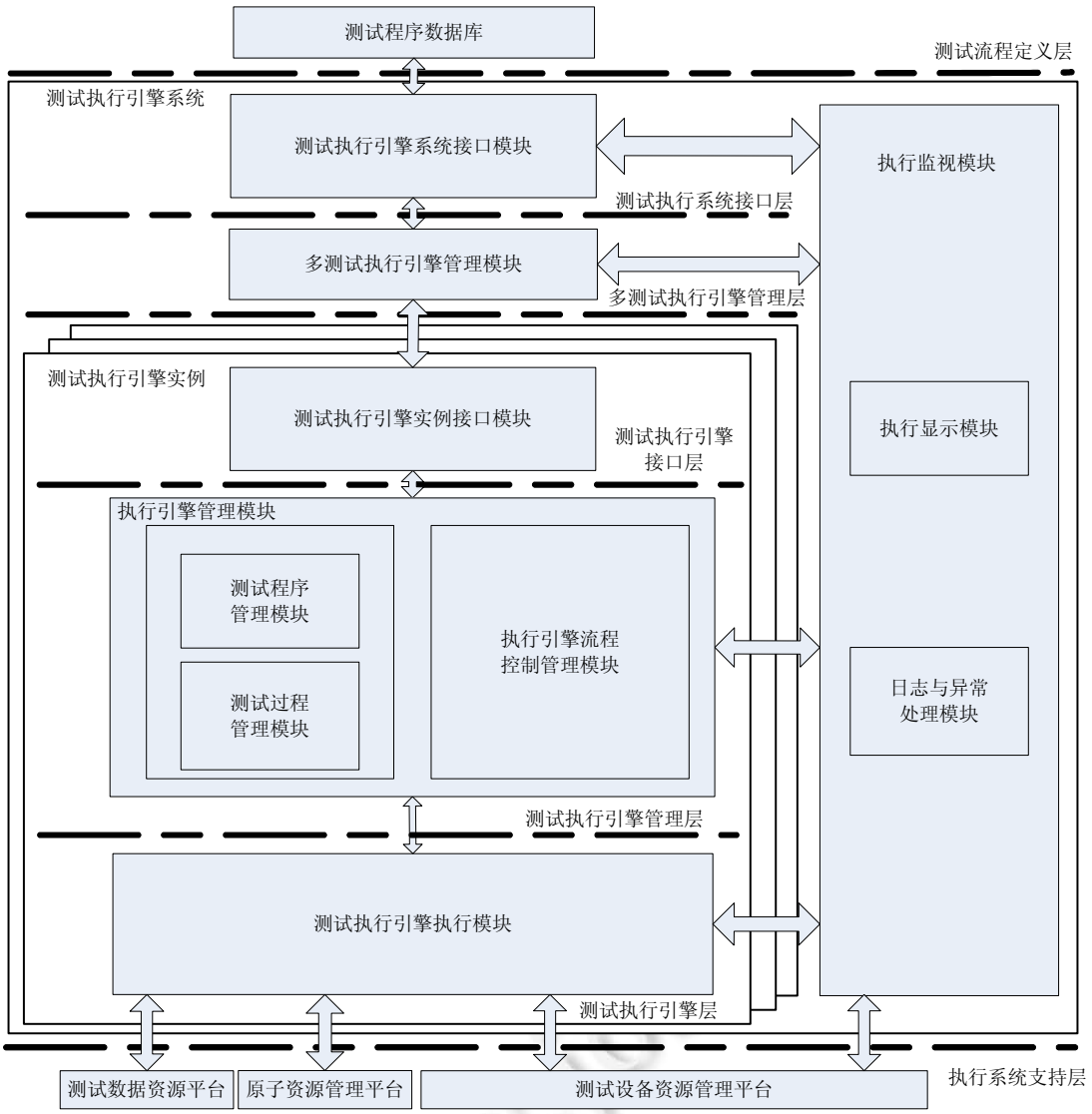

Fig.5 Architecture of test execution system of spacecraft

图 5 航天器测试系统测试执行环境子系统结构图

航天器测试系统已经成功应用于多个型号航天器实际测试中,实现对测试任务、测试设备与测试结果的有效控制与管理,提高航天器测试的准确性和效率.

例如在某航天器系统级综合测试中,需要支持实时处理数据,流量大约 136Mbps,支持 18 种数据类型,并且数据格式定义复杂,数据来源多样化;测试设备约有 20 多种类型,数量达数百台;航天器的基本测试方法约 30 个;每个航天器有数百个测试用例.根据测试语言中设备协同原子的封装规范,相应的实例化约 30 个设备协同原子,分别是文字描述、图片说明、用户暂停继续交互、用户填写结论交互、自等待、发指令(离散指令、比例指令、注入指令和设置命令)、判别静态参数、判参数工程值、判参数变化规律、监视 VC 计数比、监视 PK 时间差、监视 VC、监视 PK、时间参数判别原子、存储 PK、比较 PK、延时遥测数据比对、延时遥测原码比对、实时遥测原码比对、实时遥测数据关系判别、参数线性关系判别等设备协同原子.

基于这些设备协同原子及其上的复合操作即组织测试程序,部署航天器各类测试用例,完成航天器的综合测试.以下简要给出几个实例的构造和解释执行过程.

例 5.1:航天器测试中,遥控指令的某一类离散指令 LSTC 是向航天器发送离散遥控指令,判别接收到的离散指令回令信息后,返回执行是否正确的标识.

对于该测试过程,可定义为设备协同语句:

$$LSTC(\langle 等待回令时间\rangle)[DevColG(\langle 测试设备型号\rangle,\langle 航天器型号\rangle,\langle 指令号\rangle,\langle 回令\rangle)].$$

例如,$LSTC(20)[DevColG(0012,012,0018,ack)]$表示发送 0018 号离散指令,等待回令时间为 20s,*ack* 为回令标识.若在某测试环境下$(\sigma,\tau),(\chi,\rho),(\delta,\gamma)$,若时间约束满足,并满足协同协议.该表达式的求值过程简化描述为

$$\begin{aligned}LSTC(20)[DevColG(0012,012,0018,ack)] &= (\text{true},(\sigma\{\text{true}/ack\},\tau),\langle(012,0018),(\chi,\rho),(\delta,\gamma)\rangle^{0012})\\ &= (\text{true},(\sigma\{\text{true}/ack\},\tau),((\chi,\rho),\langle(0018),(\delta,\gamma)\rangle^{012}))\\ &= (\text{true},(\sigma\{\text{true}/ack\},\tau),((\chi,\rho),(\delta,\gamma'_{012}))),\gamma'_{012}=(0018,\gamma_{012}).\end{aligned}$$

例 5.2:判别参数工程值,是判断指定参数序号的遥测工程值是否在指定范围内.执行过程是将参数序号发送给测试设备,通过测试设备向被测航天器发出请求,并取回指定参数序号对应的被测航天器状态的参数工程值,判别该参数工程值是否符合给定的期望下限与期望上限约束:如果符合,则函数返回 true;否则,继续获取参数工程值,并判别.

该测试过程可描述为

$$GETVALUE(10)[DevColV(did,pid,pn,pv,ack)];_t JUDGE(2)[Judge(pv,low,high)],$$

其中,*did* 为测试设备型号,*pid* 为航天器型号,*pn* 为参数,*low* 为期望下限,*high* 为期望上限.*GETVALUE* 和 *JUDGE* 分别为获取工程值和判断工程值的为设备协同原子,10 和 2 分别是时间约束.

若上述表达式表示为函数 *STATICPARAMCHECKER*,即:

$$\begin{aligned}&STATICPARAMCHECKER(did,pid,pn,low,high)\overset{def}{=}\\ &GETVALUE(10)[DevColV(did,pid,pn,pv,ack)];_t JUDGE(2)[Judge(pv,low,high)].\end{aligned}$$

例如,通过测试设备 0012,判断 012 航天器的 S165 参数对应的参数工程值是否在[190,192]范围内,可具体描述为 STATICPARAMCHECKER(0012,012,S165,190,192).对给定测试环境$\langle(\sigma,\tau),(\chi,\rho),(\delta,\gamma)\rangle$,假设归约时间满足,并且协同协议满足,根据归约规则,该表达式的求值过程简化描述为

$$\begin{aligned}&STATICPARAMCHECKER(0012,012,S165,190,192)=\\ &GETVALUE(10)[DevColV(0012,012,S165,pv,ack)];_t JUDGE(2)[Judge(pv,190,162)]=\\ &(\text{true},(\sigma\{\text{true}/ack\},\tau),\langle(012,S165,pv),(\chi,\rho),(\delta,\gamma)\rangle^{0012});_t JUDGE(2)[Judge(pv,190,162)]=\\ &(\text{true},(\sigma\{\text{true}/ack\},\tau),(\chi,\rho),\langle(S165,v),\gamma\rangle^{012});_t JUDGE(2)[Judge(pv,190,162)]=\\ &(\text{true},(\sigma\{\text{true}/ack\},\tau),(\chi,\rho),(\delta,\gamma\{\gamma_{012}(S165)/v\}));_t JUDGE(2)[Judge(pv,190,162)]=\\ &JUDGE(2)[Judge(\gamma_{012}(S165),190,162)].\end{aligned}$$

若值在此范围内,则返回 true;否则为假.

本文研究的通用测试语言已经用于中国空间技术研究院航天器自动化测试系统,可以支持并行模式下的批产化测试,支持设备松耦合,已经完成北斗导航系列至少 3 种类型 12 颗卫星并行测试.在测试中,测试设备约有 20 多种类型,数量达数百台,每个航天器有数百个测试用例.根据测试语言中设备协同原子的封装规范,相应的实例化约 30 种设备协同原子,航天器上的测试用例即由这些基本设备协同原子组成的基本测试语句通过复合操作复合而成.与过去手工或手动航天器测试方法相比较,提高了测试效率,缩短了测试周期,保证了测试准确度,降低了测试成本.

6 与其他相关工作比较分析

关于设备协同模型方面,文献[12]给出一种基于本体的统一设备模型定义,由于采用本体描述设备,因而独立于具体设备类型,可以实现多种不同类型设备的抽象,但是其缺乏对设备能力的描述.文献[13,14]提出了以文

档为中心的分布式智能对象系统,利用文档封装智能对象的功能特征,在一定程度上解决了智能对象异构性的问题,但是它缺乏对于设备交互能力方面的描述.在 ECHONET 模型[15]中,把每种设备作为一种类型,每种类型都有明确的属性和访问方法,每个设备供应商为了能够联入这个网络,必须实现 ECHONET 规范中定义的接口.当设备变化的时候,其对应的规范也相应变化,因此缺乏对设备的统一描述.文献[16]基于 Atlas 的硬件平台提出了设备描述语言 DDL,实现了对 Atlas 平台设备的描述,但是这些研究都无法实现对设备的统一描述.

现有的设备协同建模语言多是基于已有的工作流语言进行扩展支持设备协同,如 BPEL,WSFL,YSWL,GSFL 等.如,GRIDCC 中的设备协同执行系统即是采用业务流程执行语言 BPEL 实现设备协同.然而对于这类工作流语言,由于缺乏对设备操作和设备协同的执行机制描述,难以实现设备间的直接协同,协同系统与设备间的耦合度高、灵活性差、扩展性低.

在作者及其所在课题组的前期研究工作中,使用流程建模语言实现了对设备协同流程的描述,并能够支持设备协同的自动化.设计了面向设备协同应用领域的流程建模语言[8],将设备操作抽象为单独语句,并给出了设备操作的解释执行机制;同时,考虑到时间操作在设备协同中的作用,设计了专门的时间操作语句,支持基于时间的设备协同.

本文中的测试设备协同语句支持设备的统一描述和访问,实现与具体设备解耦合,支持设备的动态加入和退出;可灵活编写设备协同流程,实现设备协同的可定制;设备协同语句中增加了时间约束,实现测试设备协同流程的实时约束.

7 结 论

为了支持安全苛刻系统测试中测试设备协同流程的动态性和开放性,支持通用安全苛刻系统测试语言研发,针对测试设备协同中的高阶协同性和实时性,通过测试指标、测试参数以及测试操作指令类型等给出了测试设备协同涉及的相关类型定义.通过基本测试协同表达式、设备协同原子表达式以及具有时间约束的复合算子定义,给出了测试设备协同语句的语法定义.通过定义测试设备协同表达式的求值规则,定义了测试设备协同语句的语义规则,描述了设备协同执行过程,支持设备协同流程的设计,并证明了设备协同语句语义规则的收敛性和正确性.最后,通过实例说明了测试设备协同语句的应用和执行过程描述.

未来工作将在此基础上进一步给出测试设备协同的类型系统[17],研究设备协同的语义完整性;将研究安全苛刻系统测试过程中设备协同呈现的特点扩展到一般应用系统中,如物联网中的大规模设备协同中等.

References:

[1] 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 with English abstract). http://www.jos.org.cn/1000-9825/4412.htm [doi: 10.13328/j.cnki.jos.004412]

[2] Lü JH, Ma SL, Li XJ, Song JG. A high order collaboration and real time formal model for automatic testing of safety critical systems. Frontiers of Computer Science, 2015,9(4):495−510.

[3] Curbera F, Goland Y, Klein J, Leymann F, Roller D, Thatte S, Weerawarana S. Business process execution language for Web services. 2003. http://www.ibm.com/ developerworks/library/ws-bpel/

[4] Leymann F. Web services flow language (WSFL1.0). 2001. http://www.ibm.com/software/solutions/webservices/pdf/WSFL.pdf

[5] van Der Aalst WMP, Aldred L, Dumas M, ter Hofstede AHM. Design and implementation of the YAWL system. In: Proc. of the 16th Int'l Conf. on Advanced Information Systems Engineering. Berlin, Heidelberg: Springer-Verlag, 2004. 142−159.

[6] Krishnan S, Wagstrom P, von Laszew ski G. GSFL: A workflow framework for grid services 2002. http://www-unix.globus.org/cog/papers/gsf-lpaper.pdf

[7] Chen F, Rong XH, Deng P, Ma SL. A survey of device collaboration technology and system software. Acta Electronica Sinica, 2011, 39(2):440−447 (in Chinese with English abstract).

[8] Chen F, Rong XH, Deng P. A large-scale device collaborative process design meta-model and case study. In: Proc. of the 2nd Int'l Conf. on Advanced Computer Theory and Engineer. New York: ASME, 2009. 601−608.

[9] Chen WW, Ma SL. EDRM: A unified approach for enterprise data resource management. Int'l Journal of Computer Science and Network Security, 2007,7(1):119−126.

[10] Landin PJ. The mechanical evaluation of expressions. Computer Journal, 1964,6(4):308−320.

[11] Li W, Ma SL. Limits of theory sequences over algebraically closed fields and applications. Discrete Applied Mathematics, 2004, 136(1):23−43.

[12] Mcmullen D, Reichherzer T. Identity and functionality in the common instrument middleware architecture. Applied Ontology, 2006, 3:928−942.

[13] Kawsar F, Nakajima T, Park JH, Jeong YS. A document based framework for smart object systems. In: Proc. of the 2nd Int'l Conf. on Future Generation Communication and Networking (FGCN 2008). IEEE, 2008. 178−183.

[14] Kawsar F, Fujinami K, Nakajima T. Prottoy middleware platform for smart object systems. Int'l Journal of Smart Home, 2008,2(3): 1−18.

[15] Consortium E. ECHONET Specification ver. 2.11 Part I. ECHONET Overview, 2002.

[16] Chen C, Helal A. Device integration in SODA using the device description language. In: Proc. of the 9th Annual Int'l Symp. on Applications and the Internet (SAINT 2009). IEEE, 2009. 100−106.

[17] Pierce BC, Wrote; Ma SL, Sui YF, *et al*., Trans. Types and Programming Languages. Beijing: Electronic Industry Press, 2005 (in Chinese).

附中文参考文献:

[1] 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 [doi: 10.13328/j.cnki.jos.004412]

[7] 陈峰,荣晓慧,邓攀,马世龙.设备协同技术及其系统软件研究综述.电子学报,2011,39(2):440−447.

[17] Pierce BC,著;马世龙,眭跃飞,等译.类型和程序设计语言.北京:电子工业出版社,2005.

吕江花(1975－),女,山东潍坊人,博士,讲师,主要研究领域为软件形式化,安全苛刻系统自动化测试.

李先军(1977－),男,博士,主要研究领域为安全苛刻系统自动化测试.

高世伟(1978－),男,硕士,主要研究领域为安全苛刻系统自动化测试.

孙波(1974－),男,博士,教授,博士生导师,主要研究领域为航天器自动化测试.

马世龙(1953－),男,博士,教授,博士生导师,CCF 会员,主要研究领域为海量计算,软件形式化方法,设备协同.