一种形式化的基于TTCN的测试执行方法*

郝瑞兵 吴建平 史美林

(清华大学计算机系 北京 100084)

摘要 探讨基于形式化技术的测试执行方法是开展形式化的协议一致性测试活动的关键环节. 本文提出了一种形式化的基于测试描述语言TTCN的操作语义的测试执行方法,并使用标号变迁系统刻画了这一方法的整个执行过程,同时讨论了这个方法的一个具体实现. 这种形式化的基于TTCN的测试执行方法非常适合于构造通用的协议测试系统,同时也是进行测试集的自动验证的有效手段.

关键词 协议一致性测试,形式化技术,TTCN,测试执行.

中图分类号 TP393,TP311

协议一致性测试的目的是为了验证协议实现是否满足协议标准的一致性需求. 当前的协议一致性测试活动基本上是依照国际标准化组织所制定的"协议一致性测试的方法和框架"[1]进行的. 这个标准共包括7个部分,分别对协议一致性测试的过程、测试方法、测试集的组织及其描述给出了详细的规定. 随着形式化技术在协议工程学领域的应用不断增多,越来越多的专家正在研究形式化技术在协议一致性测试活动中的可应用性.[2~4]由于ISO 9646中所描述的一致性测试过程是基于用自然语言描述的协议标准进行的,因此ISO和CCITT共同开展了一个称为"一致性测试中的形式化方法"的研究项目[5],目的是研究如何基于用形式描述语言描述的协议标准进行一致性测试活动. 但是这些研究活动主要集中于讨论形式化方法在测试集生成、测试管理和测试方法等方面的应用,对于协议一致性测试的关键环节"一致性评价过程(Conformance Assessment Process)"则很少涉及. 按照ISO 9646的定义,一致性评价过程是实现协议一致性测试过程的主要环节,而测试执行MOT(means of test)是实现这一环节的主要工具. 因此探讨形式化方法在测试执行这一过程中的应用具有很重要的现实意义,它是构造基于形式化方法的协议一致性测试活动的基础.

在目前已实现的协议一致性测试系统中,测试执行的设计一般都未基于形式化模型,而且具体的执行过程也各不相同. 现有的测试执行方法一般可以分为2类:①基于编译的测试执行方法,在这种方法中,测试集首先被编译成可执行的机器码,然后由测试执行逐个的运

* 本文研究得到国家自然科学基金资助. 作者郝瑞兵,1970年生,博士,主要研究领域为计算机网络体系结构,协议工程学. 吴建平,1953年生,教授,主要研究领域为计算机网络,协议工程学,分布式系统. 史美林,1938年生,教授,主要研究领域为计算机网络,协议工程学.

本文通讯联系人:郝瑞兵,北京 100084,清华大学计算机系网络实验室

本文1996-05-27收到修改稿

行测试例.②基于解释的测试执行,即边解释边执行测试例.这2种方法各有优缺点:前者要花很长的编译时间,不易于测试集的修改,但运行效率较高;后者由于是纯粹的串行解释执行,效率不高,但是有利于实现测试集的修改及测试过程监控.这2类方法还有一个共同的特点就是都采用某种自定义的内部格式描述测试集,因此其可扩充性受到了限制,对测试系统有很强的依赖性.

本文提出了一种形式化的基于TTCN的测试执行方法,并设计了它的实现,在这种方法中,测试执行是基于使用TTCN描述的测试集直接解释执行的,它具备以下几个特点:

· 整个测试执行过程完全遵循ISO 9646的定义,所有用到的概念、测试方法和测试执行的全过程符合ISO 9646的规定.

· 直接对使用TTCN描述的测试集进行解释执行,执行过程是使用TTCN操作语义定义的形式化的TTCN机的具体实现,独立于具体的被测协议实现.

· 并行的测试例解释与测试例执行过程,以提高测试执行的效率,同时保留灵活的测试集管理及方便的测试过程监控.

构造形式化的基于TTCN的测试执行方法,有如下的重要意义:

· 它是形式化的一致性测试活动的重要组成,基于这种执行方法,可以设计通用的、独立于具体被测协议及具有良好可扩充性的协议一致性测试系统.

· 基于TTCN操作语义所构造的TTCN机可以用来对TTCN描述的测试集进行自动验证,从而降低人工验证的费用,增强测试集的正确性.

· 探讨这种测试执行方法在多方测试和互操作测试中的可应用性.

本文第1节对所使用到的形式化工具"标号变迁系统"做一简单的描述;第2节详细讨论形式化的基于TTCN操作语义的测试执行方法,给出了TTCN机的定义;第3节基于形式化模型TTCN机构造了一个具体的实现,并讨论了其中的几个关键技术;第4节总结全文并对这种形式化的测试执行方法的应用前景进行了讨论.

1 标号变迁系统

标号变迁系统LTS(labeled transition system)是对系统或过程的行为进行模型化时常用的数学工具,同时也是LOTOS和CCS的语义模型,我们选用LTS作为对TTCN操作语义进行形式化描述的数学工具,下面首先给出其定义:

定义1. 标号变迁系统是一个四元式$\langle S,L,T,s_0\rangle$,其中

(1) S是一个状态的集合;

(2) L是一个外部可观察动作的集合;

(3) $T\subseteq S\times\{L\cup\{\tau\}\cup\{\delta\}\}\times S$是变迁的集合,其中$\tau$表示不可观察的内部动作,$\delta$表示成功的结束.$T$中的元素$(s,u,s')$可以记为:$s\xrightarrow{u}s'$,其中$s,s'\in S,u\in L\cup\{\tau\}\cup\{\delta\}$;

(4) s_0为初态.

"迹(*Trace*)"是标号变迁系统中常见的概念,定义2给出了迹及其表示方法的定义.

定义2. 令$\langle S,L,T,s_0\rangle$为一标号变迁系统,$L'=L\cup\{\tau\}\cup\{\delta\}$包含了全部动作,$s,s',s_1,s_2,\ldots,s_n,s_{n+1}\in S$,令$\sigma=u_1\cdot u_2\cdot\ldots\cdot u_n$是一串$L'$中标号的接续,'$\cdot$'表示接续运算,则$\sigma$

被成为 L' 上的一个迹(*Trace*). L'^* 表示 L' 上所有可能的迹的集合,同时我们引入下面有关迹的表示方法:

(1) 如果 $s=s_1\xrightarrow{u_1}s_2\xrightarrow{u_2}\cdots\xrightarrow{u_n}s_{n+1}=s'$,则 $s\xrightarrow{\sigma}s'$.

(2) 如果 $s=s_1\xrightarrow{\tau^*}s_2\xrightarrow{u_1}s_3\xrightarrow{\tau^*}s_4=s'$,则 $s\overset{u_1}{\Rightarrow}s'$, $u_1\in L$.

(3) 如果 $s=s_1\overset{u_1}{\Rightarrow}s_2\overset{u_2}{\Rightarrow}\cdots\overset{u_n}{\Rightarrow}s_{n+1}=s'$,则 $s\overset{\sigma}{\Rightarrow}s'$.

(4) 如果$\exists$ s',有 $s\xrightarrow{\sigma}s'$,则 $s\xrightarrow{\sigma}$.

(5) 如果$\exists$ s',有 $s\overset{\sigma}{\Rightarrow}s'$,则 $s\overset{\sigma}{\Rightarrow}$.

2 形式化的基于 TTCN 的测试执行

树表结合表示法 TTCN(tree and tabular combined notation)是 ISO 制定的协议测试集的描述语言,它有严格定义的语法与操作语义. 现有的一些标准测试集都是使用 TTCN 描述的,为了提高测试执行的通用性与灵活性,克服已有的测试执行方法的缺点,我们提出了一种形式化的以 TTCN 描述的测试集为操作对象,基于 TTCN 操作语义进行解释执行的测试执行方法,简称为形式化的基于 TTCN 的测试执行. 这种方法将整个测试执行过程划分为 2 个阶段,即测试例的解释阶段和测试例的执行阶段.[6]在解释阶段完成由 TTCN 描述的测试例到 TTCN 求值树的转化,TTCN 求值树的定义见文献[1]的第 3 部分;执行阶段实际上是在一个下面所定义的 TTCN 机上运行 TTCN 求值树,依据被测协议实现的响应,得到测试判决的过程.

2.1 测试例的解释阶段

在测试例的解释阶段,要完成对 TTCN 描述的测试例的语法检查、静态语义检查和到 TTCN 求值树的转换. 对测试例的语法检查和静态语义检查应基于 ISO 9646 中 TTCN 的巴克斯语法和静态语义的定义来完成. TTCN 求值树的构造是解释阶段所要完成的主要任务,也是我们形式化描述的对象.

TTCN 求值树是 TTCN 测试例的机器内部表示格式,它以树形结构表示测试例内事件的层次关系,树中的每个节点表示一个测试事件. 前后顺序的事件用父子关系的树节点来表达,而选择关系的事件用兄弟关系的树节点表达. 为了加快执行阶段的效率,在构造求值树的时候还要把 TTCN 测试例中的一些复杂结构用简单结构来替代,这些工作包括:

(1) 将测试例的缺省行为附加在求值树内;

(2) 用布尔表达式和 GOTO 语句替代 REPEAT 语句;

(3) 扩展测试例内的附接(Attach)树.

最后得到的求值树可以用定义 3 中给出的测试行为表达式来描述.

定义 3. 测试行为表达式 TBE(test behavior expression)的语法定义如下:

$B=_{def}$**stop** | **exit** | id? a;B | id! a;B | B□B | B>>B|

[q];B | (I:=val);B | **start** tid val;B | **stop** tid; B | **timeout** tid;B

其中运算符的优先级是';' > '□' > '>>',stop 表示停止求值,exit 表示成功的退出;id? a;B 和 id! a;B 分别表示在接口 id 处的接收事件 a 和发送事件 a;';'表示顺序执行;

'□'表示选择;q 是布尔表达式,其取值为 TRUE 或 FALSE;I:=val 为赋值运算,I 为标识符,val 为值;'>>'表示附接表达式运算;**start** 和 **stop** 分别表示计时器的启动和停止,**timeout** tid 表示计时器超时.

上面所定义的 TBE 实际上是 TTCN 求值树的代数形式表述,它描述的测试例在语义上与 TTCN 描述的测试例完全相同.因篇幅有限,这里省略其详细的证明.所以解释过程也可以看作是保持语义未改变的从 TTCN 描述的测试例到 TBE 描述的测试例的转换,这里所提到的测试例是指经过上面所说的简化处理的测试例.下面引入转换函数的定义:

定义 4. 令 $TBEs$ 表示所有测试行为表达式 TBE 的集合,TCs 是 TTCN 描述的测试例的集合,t 是 TCs 中的一个测试例,$t\in TCs$,解释过程可以被定义为 TCs 上的一个函数 $Texpr$,它将测试例 t 转换为测试行为表达式 B_t,并保持语义的不变性:

$$Texpr:TCs\rightarrow TBEs.$$

$$Texpr(t)=B_t.\quad B_t\in TBEs.$$

我们通过几个例子来看看从 TTCN 格式的测试例到 TBE 的转换.

例 1:顺序执行与选择执行的转换.

```
pco1! a
    pco1? b
    pco2? c
    ? timeout t1
⇒ pco1! a;( pco1? b;stop□pco2? c;stop□timeout t1;stop)
```

例 2:附接(Attach)树的扩展.

```
pco1! a                    B:  pco2? d
      +B                          pco1! e
      pco2? c                pco1? f (inconc)
⇒ pco1! a; (pco2? d; pco1! e; exit□ pco1? f; stop) >> pco2? c
```

例 3:循环语句转换为递归定义.

```
pco1? a    L1
      pco1? b
      →L1
⇒ L1= pco1? a; pco1? b; L1
```

2.2 测试例的执行阶段

在得到测试例的 TBE 格式描述之后,便进入测试例的执行阶段.在这个阶段,执行器基于 TTCN 的操作语义,即按照 TBE 的操作语义,完成各种事件,并根据被测协议实现的响应做出最终的测试判决.为了形式化的描述基于 TTCN 操作语义的执行过程,我们首先引入 TTCN 机(TTCN-M)的定义,然后在 TTCN-M 上用标号变迁系统给出 TBE 的操作语义定义.

定义 5. 一个 TTCN 机 TTCN-M,是一个能够完成用 TBE 描述的测试例 t 的求值过程的虚拟机.TTCN-M 的状态可以用一个三元式$\langle ctl,sto,env\rangle$来表示,定义如下:

(1) 控制部分 ctl. ctl 为要执行的测试例的 TBE 表达式,

$$ctl=Texpr(t)\in TBEs;$$

(2) 存储部分 sto. $sto=\{(i,v)\mid(i,v)\in MEM=Ident\times Value\}$,其中 $Ident$ 是标识符的集合,$Value$ 是值的集合.sto 中的元素为(标识符,值)对,表示了测试集中的变量、常量及其

对应的值;

(3) 环境部分 env. $env=\{(qid,i,o)\mid(qid,i,o)\in ENV=ID\times\{Input^*\}\times\{Output^*\}\}$, *TTCN-M* 与环境的交互作用通过多个交互点实现,每个交互点由一对入出队列描述. 其中 $ID=\{pcoid\}\cup\{timer\}$,包含了所有的交互点的名称,$pcoid$ 表示观察控制点,$timer$ 用于表示计时器的队列,入队列存放超时计时器,出队列中为启动的计时器. 每个队列中的元素或者是协议数据单元(*PDUs*)、或者是抽象服务原语(*ASPs*)、或者是计时器. $Input,Output\in\{PDUs\}\cup\{ASPs\}\cup\{tid\}$.

下面我们通过定义测试例 t 所对应的测试行为表达式 B_t 的操作语义来刻画执行器的运行过程.

定义 6. 测试行为表达式 B_t 的操作语义可以用一个标号变迁系统 $LTS(B_t)$ 来描述:

$$LTS(B_t)=_{def}\langle S,L,Tran,s_0\rangle,$$

其中(1) $S=\{(ctl,sto,env)\}\subseteq TBEs\times MEM\times ENV$,包含了运行测试例的 *TTCN* 机的所有可能的状态;

(2) $L'=L\cup\{\tau\}\cup\{\delta\}$,是所有可能的动作集合,$?x$ 表示一个接收事件,$!x$ 表示一个发送事件;

(3) $s_0=(B_t,sto_0,env_0)$,$B_t=Texpr(t)$,sto_0,env_0 分别为存储部分和环境部分的初始状态;

(4) $Tran\subseteq S\times L'\times S$,包含了满足下述规则的变迁:

$(exit,sto,env)\xrightarrow{\delta}(stop,sto,env)$;

$(id?\ a;B,sto,env)\xrightarrow{?a}(B,sto,env[(id,a\cdot i,o)/(id,i,o)])$,

其中 $env[X/Y]=_{def}(env-\{X\})\cup\{Y\}$;

$(id!\ a;B,sto,env)\xrightarrow{!a}(B,sto,env[(id,i,o)/(id,i,o\cdot a)])$,

$(B_1\square B_2,sto,env)\xrightarrow{\mu}(B_1',sto',env')$,

如果$(B_1,sto,env)\xrightarrow{\mu}(B_1',sto',env')$,

$(B_1\square B_2,sto,env)\xrightarrow{\mu}(B_2',sto',env')$,

如果$(B_2,sto,env)\xrightarrow{\mu}(B_2',sto',env')$,

$(B_1>>B_2,sto,env)\xrightarrow{\mu}(B_1'>>B_2,sto',env')$,

如果$(B_1,sto,env)\xrightarrow{\mu}(B_1',sto',env')$,$\mu\in L\cup\{\tau\}$;

$(B_1>>B_2,sto,env)\xrightarrow{\tau}(B_2,sto',env')$,

如果$(B_1,sto,env)\xrightarrow{\delta}(B_1',sto',env')$;

$([q];B,sto,env)\xrightarrow{\tau}(B,sto,env)$,if $q=TRUE$;

$([q];B,sto,env)\xrightarrow{\tau}(stop,sto,env)$,if $q=FALSE$;

$([I:=v];B,sto,env)\xrightarrow{\tau}(B,sto[(I,x)/(I,v)],env)$,

其中 $sto[(I,x)/(I,v)]=(sto-\{(I,x)\})\cup\{(I,v)\}$;

$(start\ tid\ value;B,sto,env)\xrightarrow{\tau}(B,sto,env')$,

其中 $env'=env[(timer,i,o)/(timer,(tid,value)\cdot i,o)]$;

$(stop\ tid;B,sto,env)\xrightarrow{\tau}(B,sto,env')$,

其中 $env'=env[(timer,i,o)/(timer,(tid,0)\cdot i,o)]$;

$(timeout\ tid;B,sto,env)\xrightarrow{\tau}(B,sto,env')$,

其中 $env'=env[(timer,i,o_1\cdot(tid,0)\cdot o_2)/(time,i,o_1\cdot o_2)]$;$o_1\cdot(tid,0)\cdot o_2$表示超时队列中含有计时器 tid.

在定义6中我们利用标号变迁系统在TTCN机上描述了TTCN的操作语义,实际上也定义了基于TTCN操作语义的测试执行过程.测试执行的最终目的是为了获得测试判决,从而形成一致性测试报告,因此我们还需要对测试判决进行形式化的描述.

定义7. B_t 是测试例 t 的测试行为表达式,$LTS(B_t)$定义了 B_t 的操作语义,令

$$LTS(B_t)=\langle S,L,Tran,s_0\rangle,$$

其中 $s_0=(B_t,sto_0,env_0)$.

$Trace(t)=_{def}\{\sigma\mid s_0\xRightarrow{\sigma}\}$.

$Pass(t)=_{def}\{\sigma\mid\sigma\in Trace(t)$且相应测试事件序列在 t 中的测试判决为 $Pass\}$,

$Fail(t)=_{def}\{\sigma\mid\sigma\in Trace(t)$且相应测试事件序列在 t 中的测试判决为 $Fail\}$,

$Inconc(t)=_{def}\{\sigma\mid\sigma\in Trace(t)$且相应测试事件序列在 t 中的测试判决为 $Inconclusive\}$.

$Trace(t)$定义了执行测试例 t 时所有可能的由可观察动作组成的迹.每个测试例都有测试目的,依据测试目的定义了3个互斥子集 $Pass(t)$,$Fail(t)$,$Inconc(t)$,分别包含了测试判决为 $Pass$,$Fail$,$Inconclusive$ 的迹.

定义8. 令函数 Log 模拟测试例 t 在TTCN-M上的一次运行,它执行测试例 t 并给出执行过程中可观察动作的记录:

$Log:TCs\rightarrow L^*$, TCs 为测试例的集合,L^* 为 L 上所有可能的迹的集合,

$Log(t)=_{def}\sigma$,其中 $\sigma\in Trace(t)$.

定义9. 令Log函数如上定义,我们引入测试判决函数的定义:

$Verdict:L^*\times TCs\rightarrow\{Pass,Fail,Inconc\}$,

$Verdict(\sigma,t)=_{def}v$,其中 $\sigma=Log(t)$,$v\in\{Pass,Fail,Inconc\}$.

$Verdict(\sigma,t)=Pass$ 如果 $\sigma\in Pass(t)$,

$Verdict(\sigma,t)=Fail$ 如果 $\sigma\in Fail(t)$,

$Verdict(\sigma,t)=Inconc$ 如果 $\sigma\in Inconc(t)$.

至此,我们使用形式化的方法定义了整个基于TTCN操作语义的测试解释执行过程.

3 形式化的基于TTCN的测试执行的实现

在由清华大学设计的协议一致性测试系统PCTS中(见图1),其核心部件测试执行子系统TE便是基于我们在第2节中提出的测试执行方法实现的.在实际测试过程中,我们发现这种基于TTCN操作语义的测试执行方法在保证测试系统的协议独立性和提供方便的测试集管理和测试过程监控方面是非常有用的.在本节中,我们给出该方法在测试系统中的

结构设计并讨论几个关键的技术问题.

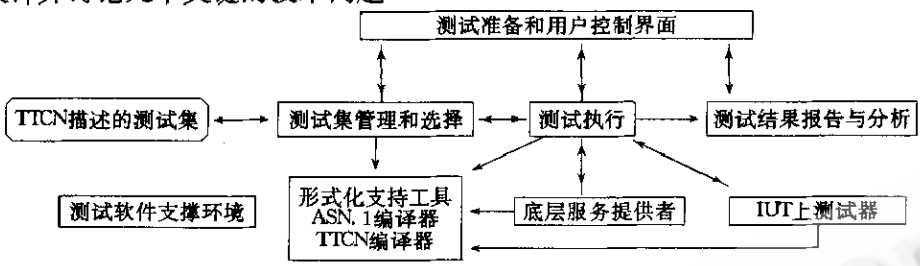

图1　协议一致性测试系统总体结构

3.1　PCTS 中 TE 的结构设计

PCTS 中测试执行子系统 TE 是形式化的基于 TTCN 的测试执行方法的一个具体实现,图 2 给出了它的内部结构及其与周围环境和其他子系统的交互作用. 整个测试执行子系统由解释器、执行器和接口队列组成.

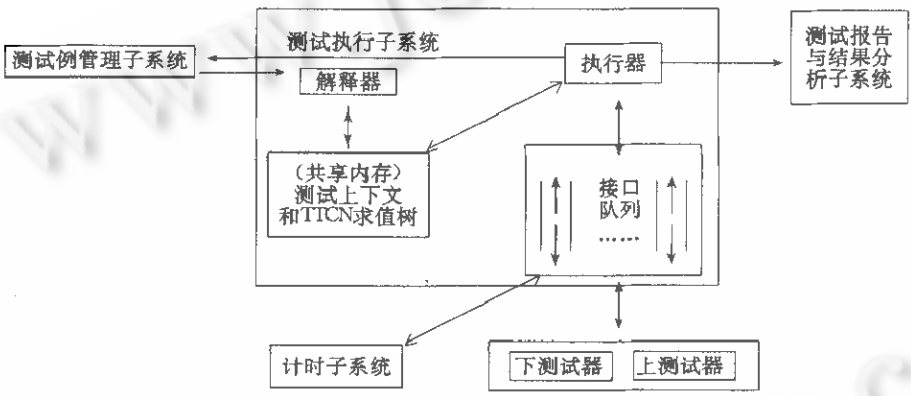

图2　测试执行子系统结构设计

解释器完成由 TTCN 描述的测试例到 TTCN 求值树的转化,它由测试例管理子系统获得下一个将要执行的测试例并对其进行解释,测试例管理子系统依据文献[7]中提出的测试例定序关系完成下一个测试例的选择,TTCN 求值树存放在共享内存中供执行器使用;解释器依据 TTCN 求值树,基于 TTCN 的操作语义完成执行过程,它通过与上下测试器的接口完成接收和发送事件,同时改变共享内存中的测试上下文(即 TTCN 机中的存储部分),最后形成测试判决,送至测试报告与结果分析子系统供生成一致性测试报告时使用,同时将执行结果回送至测试例管理子系统供选择下一测试例时使用;接口部分实现了测试执行子系统与计时子系统、上下测试器之间的信息传递.

3.2　关键技术

在这里我们讨论在实现测试执行子系统时使用的 2 个关键技术:并发执行的解释器和执行器,统一的消息机制.

为了加快解释执行的效率,我们采用多线程技术实现了解释器与执行器的并发执行. 解释器与执行器分别作为单独的线程实现,二者之间的同步通过信号量的 P,V 操作实现. 解释器不断地从测试例管理子系统获得下一个将要执行的测试例,对其进行转换并存放在共享内存中;同时执行器则执行一个已在共享内存中的 TTCN 求值树,形成测试判决. 这种两级流水线技术可以明显地提高由于纯粹串行解释执行所降低的效率.

单独地实现执行器与上下测试器或计时子系统的每个接口是件很费力的工作,因此在TE的实现中我们规定了一种统一的接口消息传递机制,并利用一个消息实体实现整个接口的功能.在这种消息机制中,每个消息都带有源和目的的标识,消息实体为每个消息接收者设立一个队列,用于存放属于同一接收者的消息.消息实体负责接收从各个消息发送者发出的消息,存放至相应的队列;同时处理消息接收请求,返回对应消息队列上的头一个消息.利用这种方法,我们可以统一的设计测试执行器、上下测试器和定时子系统的对外接口,使整个PCTS具有很好的可扩充性.

4 总 结

本文提出了一种形式化的基于TTCN的测试执行方法,并利用标号变迁系统刻画了它的解释执行过程,同时讨论了它在清华大学设计的协议一致性测试系统PCTS中的实现.利用PCTS,我们对X.25 LAPB、PLP、TP0－2、PAD协议、会话层和表示层协议以及X.400协议进行了完整的测试,对这种测试执行方法进行了充分的验证.在实际测试过程中,这种测试执行方法的执行效率比普通的串行解释执行要快30～40%,同时它还提供了良好的测试过程监控和方便的测试集管理能力.基于这种测试执行方法的测试系统完全独立于具体的被测协议,具有良好的通用性和可扩充性,为进行协议一致性测试活动提供了强有力的工具.

形式化的基于TTCN的测试执行方法,基于协议的形式化描述的测试生成[3],以及形式化的一致性测试框架[5]结合在一起,完整地描述了整个形式化的协议一致性测试过程,为构造通用的一致性测试系统奠定了良好的理论基础.同时抽象的TTCN机也可应用于TTCN抽象测试集的验证活动,目前协议测试集一般是手工设计的,对其进行验证是一件很费力的事情.利用TTCN机,我们可以模拟一个测试例所有可能的运行情况,并根据测试集中的规定给出测试判决,依据协议标准我们可以对各种运行情况是否合法进行检查,从而完成测试例的验证活动.形式化的基于TTCN的测试执行方法不仅适用于协议的一致性测试过程,对于互操作测试和多方测试也是适用的.但是如何实现多个上下测试器和主测试器之间的协调过程,还是一个值得探讨的问题.

目前,ISO在TTCN的基础上又制定了并发TTCN的描述语言,如何实现基于并发TTCN的解释执行是我们目前正在研究的一个课题.

参考文献

1 ISO. Conformance Testing Methodology and Framework. IS-9646. ISO,1991. CCITT X.290-X.294.

2 Hao Ruibing, Wu Jianping, Chanson Samual T. Design and implementation of an automatic test suite generator. Chinese Journal of Advanced Software Research,1994,1(2):152～165.

3 Jan Tretmans. A formal approach to conformance testing. In: Proceeding of International Workshop on Protocol Testing System(IWPTS), Leidschendam Netherlands. 1994. 261～280.

4 Tretmans, Gerrit Jan. A formal approach to conformance testing. Dissertation, University of Twente, Enschede, The Netherlands, 1992.

5 ISO/IEC JTC1/SC21 N6201. Formal methods in conformance testing. working draft. Project 1.21.54. ISO,1992.

6 Wang Yaming, Wu Jianping, Hao Ruibing. An approach to TTCN-based test execution. In: Proceeding of IWPTS

VII, Japan, 1994. 299～306.

7 Tian Jining, Wu Jianping. Test management and TTCN based test sequencing. In: Proceeding of IWPTS VIII, France, 1995. 427～442.

A FORMAL APPROACH TO TTCN-BASED TEST EXECUTION

HAO Ruibing　WU Jianping　SHI Meilin

(*Department of Computer Science　Tsinghua University　Beijing*　100084)

Abstract　Research on formal test execution methods is an important phase in the activity of formal protocol conformance testing. This paper proposes a formal test execution approach based on test notation language TTCN's operational semantics and described it by using labeled transition system, and also discusses an implementation of this method. This formal approach is very suitable for the construction of general protocol testing system, and can be an effective way for automatic test suite verification.

Key words　Protocol conformance test, formal technique, TTCN, test execution.

Class number　TP393, TP311