ISSN1000-9825 Journal of Software 软件学报 2000,11(4):427~434

A Formal Approach to Conformance Testing of Internet Routing Protocols*

BI Jun WU Jian-ping

(Department of Computer Science and Technology Tsinghua University Beijing 100084)

E-mail: jbi@bell-labs.com

Abstract Concurrent TTCN (tree and tabular combined notation) is a test notation that can handle concurrent test behaviors. This paper proposes an approach to the conformance testing of distributed routing protocols based on concurrent TTCN. It first discusses the test architecture for the routing protocol entity, and then the concurrent TTCN based test system design is presented. Finally, this paper introduces the test suite design.

Key words Conformance testing, distributed system, routing protocol, internet, concurren. TTCN.

OSPF now is the most important routing protocol and is widely used in the Internet. The quality of the OSPF products would influence the stability of networks and it is really necessary to give the conformance test assessment of OSPF routers. However, the routing functions are realized by the cooperation of distributed routers. Then how to formally specify and test the concurrent behaviors of OSPF router is a challenge to the traditional protocol test methods and test systems. In Refs. [1,2], some problems of conformance testing in distributed systems were studied. The entity in distributed systems is more complex and general than traditional peer-to-peer protocol entity. An entity in distributed systems is associated with a set of interaction points and a set of concurrent behaviors at these interaction points. In this paper, we present a specification model called concurrent external behavior expression (CEBE) to specify the entity. CEBE could directly specify the concurrent external behaviors at different interaction points of an entity and reduce the internal states, actions and data. Therefore it is an ideal model to formally specify the entity in a complex distributed system and to generate concurrent TTCN (C-TTCN) test suite. Moreover, we formally define test system based on C TTCN operational semantics. The above methods have been put into practice by using an integrated test system PITS and an

* This research is supported by the National Natural Science Foundation of China (国家自然科学基金, Nos. 69473011 and 69682002). **BI Jun** was born in March, 1972. He is a postdoctoral researcher of High Speed Networks Research Department, Networking Laboratory, Bell Laboratories, USA. He received the Ph.D. degree at Department of Computer Science, Tsinghua University in 1999. He was invited as a referee for some international journals and conferences, such as International Journal of Computer Communications (UK), Journal of Computer Networks (North-Holland), Journal of Parallel and Distributed Computing (USA), IEEE International Conference on Communications, IFIP IWTCS, and IFIP FORTE/PSTV. His research interests are high speed computer networks, Internet routing, and protocol engineering. **Wu Jian-ping** was born in October, 1953. He is a professor and doctoral supervisor of the Department of Computer Science, Tsinghua University. He is the director of China Education and Research Networks (CERNET), Head of China Education and Research Networks Experts Committee, Director of Networks Research Center, Tsinghua University, member of "863" high-tech Experts Committee, and Chair or member of program committee of many international conferences. His current research areas include networks architecture, networks management, networks security and protocol engineering.

Manuscript received 1998-07-24, accepted 1998-11-17.

automatic test generation tool TUGEN to test CISCO 4700 router.

The rest of this paper is organized as follows. In Section 1, we give out the abstract test architecture. Then we discuss the test system design and test suite generation in Sections 2 and 3 respectively. Finally, we conclude this paper in Section 4.

1 Abstract Test Architecture

The Tree and Tabular Combined Notation (TTCN) is recommended by ISO to describe abstract test suite. C-TTCN is an extension of TTCN, that allows the test system to execute a test case by several test components (TCs) running in parallel. A tester consists of exactly one main test component (MTC) and any number of parallel test components (PTCs). TCs are linked by coordination points (CPs) capable of conveying coordination messages (CMs). Communications of TCs with the environment are through the points of control and observation (PCOs). C-TTCN is very suitable for the specification of a system consisting of several testers, while this is the common case in routing testing. The test architecture is a description of the environment in which IUT (Implementation under Test) is tested. For a router entity, there are k concurrent behaviors which happen at interaction points. If and only if all these behaviors conform to the behaviors of its specification, it is a conformance implementation. Then the test architecture should deal with these concurrent behaviors at several interaction points. We construct the abstract test architecture according to this feature. It consists of a test system, an entity under test, a test context, several PCOs and several interaction points. The test system is the implementation of a test suite written in C-TTCN. It carries out the experiments by executing test cases and observing results. The interactions of the IUT with its environment take place at interaction points (IPs). The test system communicates with the IUT indirectly; through the test context. The points, where the tester communicates with the test context and in this way indirectly controls and observes the IUT, are called the Points of Control and Observation (PCOs). Execution of a test case starts with the execution of MTC. It is the concern of MTC to set up all PTCs, and to manage all PCOs and CPs to be connected to. PTCs can be created by MTC on demand. A 'create' operation associates a PTC with a behavior tree. The newly created PTC starts execution of its assigned behavior tree concurrently with MTC. MTC may explicitly terminate a PTC by executing a 'down' operation.

2 C-TTCN Based Test System

2.1 Preliminaries

Definition 2.1. A Labeled Transition System (LTS) is a 4-tuple $\langle S, L, T, s_0\rangle$. (1) S is a countable, nonempty set of states. (2) L is a countable set of observable events. (3) $T\subseteq S\times(L\cup\{\tau\})\times S$ is a set of transitions, where τ denotes an unobservable event. Element (s,u,s') in T can also be written as: $s\xrightarrow{u}s'$, where $s, s'\in S$, $u\in L\cup\{\tau\}$. (4) $s_0\in S$ is the initial state.

We use $LTSs(L)$ to denote the set of all possible labeled transition systems over L. Trace is a common concept in LTS, Definition 2.2 gives out its definition and some useful notation.

Definition 2.2. Let $\langle S, L, T, s_0\rangle$ be an LTTS. $L'=L\cup\{\tau\}$ contains all observable and unobservable events. $s, s', s_1, s_2, \ldots, s_n, s_{n+1}\in S$, $u_1, u_2, \ldots, u_n\in L'$. Let $\sigma=u_1\cdot u_2\cdot\ldots\cdot u_n$ be a sequence of labels in L'. '$\cdot$' denotes concatenation, then σ is said to be a trace over L'. L'^* denotes the set of all possible traces over L'. We further have the following notations:

- if $s=s_1\xrightarrow{u_1}s_2\xrightarrow{u_2}\ldots\xrightarrow{u_n}s_{n+1}=s'$, then $s\xrightarrow{\sigma}s'$;
- if $s=s_1\xrightarrow{\tau^*}s_2\xrightarrow{u}s_3\ldots\xrightarrow{\tau^*}s_4=s$, then $s\xRightarrow{u}s$, $u\in L$, τ^* is the concatenation of zero or more τ;

- if $s=s_1\overset{u_1}{\Rightarrow}s_2\overset{u_2}{\Rightarrow}\cdots\overset{u_n}{\Rightarrow}s_{n+1}=s'$, then $s\overset{\sigma}{\Rightarrow}s'$;
- if $\exists\ s'$, $s\overset{\sigma}{\rightarrow}s'$, then $s\overset{\sigma}{\rightarrow}$;
- if $\exists\ s'$, $s\overset{\sigma}{\Rightarrow}s'$, then $s\overset{\sigma}{\Rightarrow}$.

In LTS model, events in L are treated in the same way no matter what they mean. In order to distinguish input events from output events, a kind of LTS called input-output transition system is introduced.

Definition 2.3. An Input-Output Transition System (IOTS) p is an LTS in which the set of events L is partitioned into input events L_i and output events L_o, where elements in L_i are events accepted by this LTS from its environment, and elements in L_o are events sent to its environment by this LTS. IOTS p satisfies: $p\in LTSs(L)$, $L=L_i\cup L_o$ and $L_i\cap L_o=\varnothing$. $IOTSs(L_i,L_o)$ represents the set of all possible input-output labeled transition systems over input events L_i and output events L_o. $IOTSs(L_i,L_o)\subseteq LTSs(L_i\cup L_o)$.

Definition 2.4. Input-queue operator is a unary operator $[\cdot]\ll\sigma_i: IOTSs(L_i,L_o)\rightarrow IOTSs(L_i,L_o)$, where $\sigma_i\in L_i^*$. Let $TS\in IOTSs(L_i,L_o)$, then $[TS]\ll\sigma_i$ is defined by the axiom $A1$ and inference rules $I1$, $I2$ and $I3$:

$A1$: $[TS]\ll\sigma_i\overset{a}{\rightarrow}[TS]\ll\sigma_i\cdot a$, $a\in L_i$,

$I1$: if $TS\overset{\tau}{\rightarrow}TS$ then $[TS]\ll\sigma_i\overset{\tau}{\rightarrow}[TS]\ll\sigma_i$,

$I2$: if $TS\overset{a}{\rightarrow}TS$ then $[TS]\ll a\cdot\sigma_i\overset{\tau}{\rightarrow}[TS]\ll\sigma_i, a\in L_i$,

$I3$: if $TS\overset{y}{\rightarrow}TS$ then $[TS]\ll\sigma_i\overset{y}{\rightarrow}[TS]\ll\sigma_i$, $y\in L_o$.

We will name $[S]\ll\sigma_i$ as the Input-Buffered Transition System (IBTS). For a common communication system, we can model its behavior by only one Input-Buffered Transition System. While considering communication system whose function is performed by several concurrent parts, we need a group of IBTSs to model its behaviors. Then we introduce a formal definition of communication system in the following.

Definition 2.5. A communication system consisting of n concurrent entities can be modeled by an IOTS Σ: $\langle S_\Sigma,\{Ext\}\times N\times L_{\Sigma,i}\cup N\times\{Ext\}\times L_{\Sigma,o},Tr,s_{\Sigma,0}\rangle$, where (1) S_Σ is the state set of this IOTS, element in S_Σ can be represented by $\langle s_1,s_2,\ldots,s_n\rangle$, where $1\leqslant i\leqslant n$, s_i is the state of i-th IBTS that is defined over input actions $ID\times ID\times L_i$ and output actions $ID\times ID\times L_o$, $ID=\{Ext,1,2,\ldots,n\}$; (2) $L_{\Sigma,i}$, $L_{\Sigma,o}$ are the set of input actions and the set of output actions of this communication system respectively; (3) Ext is the identification of the outside environment of this communication system; (4) $N=\{1,2,\ldots,n\}$, each number in N represents an IBTS; (5) $s_{\Sigma,0}$ is the initial state, $s_{\Sigma,0}=\langle s_{1,0},s_{2,0},\ldots,s_{n,0}\rangle$, $s_{i,0}$ is the initial state of i-th IBTS; (6) Transitions set Tr is inferred from the following axiom and rules:

$A1$: $s\overset{(Ext,i,a)}{\longrightarrow}s'$, where $s=\langle\ldots,[S_i]\ll\sigma_i,\ldots\rangle$ and $s'=s([S_i]\ll\sigma_i/[S_i]\ll\sigma_i\cdot a)$,

$I1$: if $[S_i]\ll a\cdot\sigma_i\overset{\tau}{\rightarrow}[S_i]\ll\sigma_i$, then $s\overset{\tau}{\rightarrow}s'$, where
$s=\langle\ldots,[S_i]\ll a\cdot\sigma_i,\ldots\rangle$ and $s'=s([S_i]\ll a\cdot\sigma_i/[S'_i]\ll\sigma_i)$,

$I2$: if $[S_i]\ll\sigma_i\overset{(i,y,y)}{\longrightarrow}[S_i]\ll\sigma_i$, then $s\overset{\tau}{\rightarrow}s'$, where
$s=\langle\ldots,[S_i]\rangle\ll\sigma_i,\ldots,[S_i]\ll\sigma_i\ldots\rangle$ and $s'=s([S_i]\ll\sigma_i/[S'_i]\ll\sigma_i,\ [S_j]\ll\sigma_j/[S_j]\ll\sigma_j\cdot y)$,

$I3$: if $[S_i]\ll\sigma_i\overset{(i,Ext,y)}{\longrightarrow}[S'_i]\ll\sigma_i$, then $s\overset{(i,Ext,y)}{\longrightarrow}s'$, where
$s=\langle\ldots,[S_i]\ll\sigma_i,\ldots\rangle$ and $s'=s([S_i]\ll\sigma_i/[S'_i]\ll\sigma_i)$,

$I4$: if $[S_i]\ll\sigma_i\overset{\tau}{\rightarrow}[S_i]\ll\sigma_i$, then $s\overset{\tau}{\rightarrow}s'$, where
$s=\langle\ldots,[S_i]\ll\sigma_i,\ldots\rangle$ and $s'=s([S_i]\ll\sigma_i/[S'_i]\ll\sigma_i)$,

note: $s([S_i]\ll\sigma_i/[S_i]\ll\sigma_i\cdot a)$ means substitution $[S_i]\ll\sigma_i$ with $[S_i]\ll\sigma_i\cdot a$.

2.2 The C-TTCN based test system design

We will use "communication system" in definition 2.5 to formally describe the interaction between test

system and IUT. To design the C-TTCN based test system, we should study the operational semantics of C-TTCN first. In a C-TTCN test case, the behavior of each test component is expressed with a Test Behavior Tree (TBE), which is a tree-like presentation of temporal relations between test events.

Definition 2.6. TBE $B=_{def} stop|exit|id?\ a;B|id!\ y;B|B[]B|B\gg B|[q];B|(I:=val);B|(B)|$ start tid val; $B|$ cancel tid; $B|$ timeout tid; $B|$ create *id* $B|$ done *id*.

Definition 2.7. A Test Component (TC) is a virtual machine that can perform the evaluation of test behavior expressions. It can be modeled by an IBTS, $[(ctl,sto)]\ll\sigma_i$. (1) *ctl* is a TBE specifying the behaviors of this TC. (2) *sto* is the test context of this test component, $sto=\{(i,v)|(i,v)\in MEM=Ident\times Value\}$. Ident and Value are the sets of identifiers and values respectively. Elements in *sto* are pairs of identifier and value, each element corresponds to a variable or constant in test suite and its value. (3) σ_i is the input-buffered queue of this test component. The initial state of this TC is $TC_0=[(TBE_{TC},sto_0)]\ll\varepsilon$, where TBE_{TC} is the complete behavior specification of this TC. sto_0 denotes the initial situation of the test context and ε represents that the input-buffered queue is empty.

Definition 2.8. A Test System (TS) is a system that can perform the evaluation of a test case specified in C-TTCN. It contains one main test component and zero or more optional parallel test components. Its running can be modeled by a communication system: $\langle S_{TS},\{Ext\cup Tim\}\times N\times L_{TS,i}\cup N\times\{Ext\cup(Tim)\times L_{TS,o},Tran,s_{TS,0}\rangle$ where (1) the state of *TS* in S_{TS} is the combination of the states of n test components: $(TC_1,TC_2,\ldots,TC_n)$, TC_i represents the state of test component i, $TC_i=[(TBE_i,sto)]\ll\sigma_i$; (2) $s_{TS,0}$ is the initial state of this *TS*, $s_{TS,0}=\langle TC_{i,0}|1\leqslant i\leqslant n\rangle$, where $TC_{i,0}$ is the initial state of test component i; (3) the set of input actions of this *TS* is $\{Ext\cup Tim\}\times N\times L_{TS,i}$, and the set of output actions is $N\times\{Ext\cup Tim\}\times L_{TS,o}$; (4) identifier *Ext* represents the outside environment of this test system; (5) *Tim* represents the timing system; (6) N is the set of natural number $\{1,2,\ldots,n\}$ which is used to identify each test component and τ denotes unobservable actions; (7) *Tran* is the set of transitions that are defined by the following axioms and reference rules:

A1: $s_{TS}\xrightarrow{(Ext,i,a)}s'_{TS}$, where $s_{TS}=\{TCi|1\leqslant i\leqslant n\}$,
$s'_{TS}=\{TC_1,\ldots,TC'_i=[B,sto]\ll\sigma_i\cdot(Ext,i,a),\ldots,TC_n\}$;

A2: $s_{TS}\xrightarrow{(Time,i,e\_tid)}s'_{TS}$, where $s_{TS}=\{TCi|1\leqslant i\leqslant n\}$,
$s'_{TS}=\{TC_1,\ldots,TC'_i=[B,sto]\ll\sigma_i\cdot(Tim,i,e\_tid),\ldots,TC_n\}$, *e_tid* represents the timer expired;

I1: if $TC_1=[create\ id\ B1;B2,sto]\ll\sigma_1$ then $s_{TS}\xrightarrow{\tau}s'_{TS}$ where $s_{TS}=\{TC_i|1\leqslant i\leqslant n\}$,
$s'_{TS}=\{TC'_1=[B2,sto]\ll\sigma_i,TC_2,\ldots,TC_n,TC_{n+1}\}=[B1,sto0]\ll\varepsilon$;

I2: if $TC_1=[done\ i;B1,sto]\ll\sigma_i$ and $TC_i=[stop;B2,sto]\ll\sigma_i$ then $s_{TS}\xrightarrow{\tau}s'_{TS}$ where
$s_{TS}=\{TCi|1\leqslant i\leqslant n\}$, $s'_{TS}=\{TC'_1=[B1,sto]\ll\sigma_i,\ldots,TC_{i-1},TC_{i+1},\ldots,TC_n\}$;

I3: if $TC_i=[start\ tid;B1,sto]\ll\sigma_i$ then $s_{TS}\xrightarrow{(i,Tim,s\_tid)}s'_{TS}$ where
$s_{TS}=\{TCi|1\leqslant i\leqslant n\}$, $s'_{TS}=\{TC_1,\ldots,TC'_i=[B1,sto]\ll\sigma_i,\ldots,TC_n\}$;
s_tid represents the timer being started;

I4: if $TC_i=[cancel\ tid;B1,sto]\ll\sigma_i$ then $s_{TS}\xrightarrow{(i,Tim,c\_tid)}s'_{TS}$ where
$s_{TS}=\{TCi|1\leqslant i\leqslant n\}$, $s'_{TS}=\{TC_1,\ldots,TC'_i=[B1,sto]\ll\sigma_i,\ldots,TC_n\}$;
c_tid represents the timer being canceled;

I5: if $TC_i=[timeout\ tid;B1,sto]\ll(Tim,i,e\_tid)\cdot\sigma_i$ then $s_{TS}\xrightarrow{\tau}s'_{TS}$ where $s_{TS}=\{TCi|1\leqslant i\leqslant n\}$,
$s'_{TS}=\{TC_1,\ldots,TC'_i=[B1,sto]\ll\sigma_i,\ldots,TC_n\}$;

I6: if $TC_i=[id?\ a;B1,sto]\ll(id,i,a)\cdot\sigma_i$ then $s_{TS}\xrightarrow{\tau}s'_{TS}$ where $s_{TS}=\{TC_i|1\leqslant i\leqslant n\}$,
$s'_{TS}=\{TC_1,\ldots,TC'_i=[B1,sto]\ll\sigma_i,\ldots,TC_n\}$;

I7: if $TC_i=[id!\ y;B1,sto]\ll\sigma_i$ and *id* is a *CP* then $s_{TS}\xrightarrow{\tau}s'_{TS}$ where $s_{TS}=\{TC_i|1\leqslant i\leqslant n\}$,

$s'_{TS}=\{TC_1,\ldots,TC'_i=[B1,sto]\ll\sigma_i,\ldots,TC'_{id}=[B,sto]\ll\sigma_i\cdot(i,id,y),\ldots,TC_n\}$;

I8: if $TC_i=[id!\ y;B1,sto]\ll\sigma_i$ and *id* is a *PCO* then $s_{TS}\xrightarrow{(i,Ext,y)}s'_{TS}$ where

$s_{TS}=\{TC_i\mid 1\leqslant i\leqslant n\}$, $s'_{TS}=\{TC_1,\ldots,TC'_i=[B1,sto]\ll\sigma_i,\ldots,TC_n\}$;

I9: if $TC_i=[[q];B1,sto]\ll\sigma_i$ and *q*'s value is *TRUE* then $s_{TS}\xrightarrow{\tau}s'_{TS}$ where

$s_{TS}=\{TC_i\mid 1\leqslant i\leqslant n\}$, $s'_{TS}=\{TC_1,\ldots,TC'_i=[B1,sto]\ll\sigma_i,\ldots,TC_n\}$;

I10: if $TC_i=[[q];B1,sto]\ll\sigma_i$ and *q*'s value is *FALSE* then $s_{TS}\xrightarrow{\tau}s'_{TS}$ where

$s_{TS}=\{TC_i\mid 1\leqslant i\leqslant n\}$, $s'_{TS}=\{TC_1,\ldots,TC'_i=[stop,sto]\ll\sigma_i,\ldots,TC_n\}$;

I11: if $TC_i=[(I:=val);B1,sto]\ll\sigma_i$ then $s_{TS}\xrightarrow{\tau}s'_{TS}$ where $s_{TS}=\{TC_i\mid 1\leqslant i\leqslant n\}$,

$s'_{TS}=\{TC_1,\ldots,TC'_i=[B1,sto']\ll\sigma_i,\ldots,TC_n\}$, $sto'=(sto-\{(I,x)\})\cup\{(I,val)\}$;

I12: if $TC_i=[B1[]B2,sto]\ll\sigma_i$ and $[B1,sto]\ll\sigma_i\xrightarrow{u}[B1',sto']\ll\sigma'_i$ then $s_{TS}\xrightarrow{u}s'_{TS}$ where

$s_{TS}=\{TC_i\mid 1\leqslant i\leqslant n\}$, $s'_{TS}=\{TC_1,\ldots,TC'_i=[B1',sto']\ll\sigma'_i,\ldots,TC_n\}$;

I13: if $TC_i=[B1[]B2,sto]\ll\sigma_i$ and $[B2,sto]\ll\sigma_i\xrightarrow{u}[B2',sto']\ll\sigma'_i$, then $s_{TS}\xrightarrow{u}s'_{TS}$ where

$s_{TS}=\{TC_i\mid 1\leqslant i\leqslant n\}$, $s'_{TS}=\{TC_1,\ldots,TC'_i=[B2',sto']\ll\sigma'_i,\ldots,TC_n\}$;

I14: if $TC_i=[B1\gg B2,sto]\ll\sigma_i$ and $B1=exit$, then $s_{TS}\xrightarrow{\tau}s'_{TS}$ where

$s_{TS}=\{TC_i\mid 1\leqslant i\leqslant n\}$, $s'_{TS}=\{TC_1,\ldots,TC'_i=[B2,sto]\ll\sigma_i,\ldots,TC_n\}$;

I15: if $TC_i=[B1\gg B2,sto]\ll\sigma_i$ and $[B1,sto]\ll\sigma_i\xrightarrow{u}[B1',sto']\ll\sigma'_i$ then $s_{TS}\xrightarrow{\tau}s'_{TS}$ where

$s_{TS}=\{TC_i\mid 1\leqslant i\leqslant n\}$, $s'_{TS}=\{TC_1,\ldots,TC'_i=[B1'\gg B2,sto']\ll\sigma'_i,\ldots,TC_n\}$.

For the convenience of discussion, we suppose that systems under test can also be modeled by a communication system. When testing, it consists of many interfaces connected to the test system.

Definition 2.9. The system under test (SUT) is a communication system Σ that consists of lots of Input-Buffered Transition Systems. Σ is denoted by $\langle S_{SIUT},\{Ext\}\times N\times L_{TS,o}\cup N\times\{Ext\}\times L_{TS,i},Tran,s_{SUT,0}\rangle$, where: (1) the state of Σ is obtained from the states of two IBTSs, $s_{SUT}=(s_{IUT1},s_{IUT2},\ldots,s_{IUTn})$, where s_{IUT1}, $s_{IUT2},\ldots,s_{IUTn}$ are the states of the behaviors at each interaction point, respectively; (2) the initial state of Σ is $s_{SUT,0}=(s_{IUT1,0},s_{IUT2,0},\ldots,s_{IUTn,0})$, where $s_{IUT1,0},s_{IUT2,0},\ldots,s_{IUTn,0}$ are the initial states of the behaviors at each interaction point, respectively; (3) the set of input actions to Σ is $\{Ext\}\times\{1,\ldots,n\}\times L_{TS,o}$, corresponding to the output actions from TS; (4) the set of output actions from Σ is $\{1,\ldots,n\}\times\{Ext\}\times L_{TS,i}$, corresponding to the input action to TS, τ is an unobservable action; (4)Tran is the same as in definition 2.9.

The testing process can be described by synchronous communications between TS and SUT. We use a synchronous operator '$\|$' to express that when TS (or SUT) sends out an action, SUT (or TS) will receive it at the same time (they are synchronous at this action). We can use $s_{TS,0}\|s_{SUT,0}$ to describe the whole test execution process. The synchronous operation can be represented as follows:

$s_{TS}\|s_{SUT}\xrightarrow{?a}s'_{TS}\|s'_{SUT}$, if $s_{TS}\xrightarrow{(Ext,i,a)}s'_{TS}$ and $s_{SUT}\xrightarrow{(j,Ext,a)}s'_{SUT}$;

$s_{TS}\|s_{SUT}\xrightarrow{!y}s'_{TS}\|s'_{SUT}$, if $s_{TS}\xrightarrow{(i,Ext,y)}s'_{TS}$ and $s_{SUT}\xrightarrow{(Ext,j,y)}s'_{SUT}$;

$s_{TS}\|s_{SUT}\xrightarrow{\tau}s'_{TS}\|s_{SUT}$, if $s_{TS}\xrightarrow{\tau}s'_{TS}$.

The result of test execution is a trace σ of $s_{TS,0}\|s_{SUT,0}$ that satisfies the above operation rules. Given a test case t, the execution result of t in a TS defined above is a trace σ: $Run(t,SUTs)=\sigma$, where

$s_{TS,0}\|s_{SUT,0}\xRightarrow{\sigma}s'_{TS}\|s'_{SUT}$ and $\neg\exists\ a\in\{Ext\cup Tim\}\times N\times L_{TTS,i}\cup N\times\{Ext\cup Tim\}\times L_{TS,o}\cup\{\tau\}$ $s'_{TS}\|s'_{SUT}\xrightarrow{a}$.

The final test verdict of this test case t is based on the corresponding verdict of this trace σ.

3 C-TTCN Test Suite Generation

3.1 Basic definitions of CEBE

In this section, we propose an entity specification model CEBE. It is an extension of EBE presented in

Ref. [3]. The essence of both EBE and CEBE is to describe the data in addition to the control behavior from the viewpoint outside the system. By eliminating the internal actions, the specification of CEBE could be obtained from ETS (the extension of ETS presented in Ref. [4]), which could be translated from LOTOS and Estelle. Therefore, the CEBE may be translated from the standard FDTs.

Definition 3.1. A CEBE is a 7-tuple $\langle S, G, \Sigma, A, R, s_0, v_0\rangle$, where: (1) S is a finite set of external states, whose elements are pause states of interactions exchanged between the entity and its external environment; (2) G is a set of gates over which a CEBE can communicate; (3) Σ is the set of data on CEBE (see below); (4) A is a set of external actions on CEBE (see below); (5) R is a set of logic relations of transitions between external states (see below); (6) $s_0 \in S$ is the initial state of the entity, which represents the initial external state; (7) v_0 is the set of initial assignments (initial values of all variables and timers) of Σ.

Definition 3.2. $\Sigma = IO \cup VAR \cup CONS \cup TIMER \cup PARA$ is a set of data on CEBE, where: (1) IO is the set of input or output events (i.e., the ASPs and PDUs); (2) VAR is the set of variables (the type of variables includes integer, boolean, octetstring, bitstring and so on); (3) $CONS$ is the set of constants (the type of constants includes integer, boolean, octetstring, bitstring and so on); (4) $TIMER$ is the set of timers (second and microsecond); (5) $PARA$ is the set of parameters of events in IO. For an event $e \in IO$, the parameters $p_1, p_2, \ldots, p_n$ are denoted as $e.p_1, e.p_2, \ldots, e.p_n$, e is denoted as $e(e.p_1, e.p_2, \ldots, e.p_n)$.

Definition 3.3. A behavior b is elements g of G with a list of input and output events and their value declaration. Then the set of b is denoted as: $B = \{(g_i, ?\, r_i(r_i.p_1, r_i.p_2, \ldots, r_i.p_n), !\, s_i(s_i.p_1, s_i.p_2, \ldots, s_i.p_n)) \mid g_i \in G, r_i, s_i \in IO, r_i.p_i, s_i.p_i \in PARA\}$, where symbol "!" indicates the output of an event s_i to external environment and symbol "?" indicates the input of an event r_i. The absence of r_i or s_i could be denoted as symbol "—", for example, $b = (g_i, ?\, r_i(r_i.p_1, r_i.p_2, \ldots, r_i.p_n), !\, -)$. Here we give some notations: (1) $b_1 \gg b_2 \gg \ldots \gg b_n$, the behaviors are sequential; (2) $b_1 [] b_2 [] \ldots [] b_n$, the behaviors are equally valid alternatives (choice); (3) $b_1 \parallel b_2 \parallel \ldots \parallel b_n$, the behaviors are parallel.

Definition 3.4. Actions A are the set of concurrent behaviors. $A = \{a_i \mid a_i \in A\}$, $a_i =_{def} b_j \in B \mid a_i [] a_i \mid a_i \gg a_i \mid a_i \parallel a_i \mid (a_i)$, where the operator precedence is '()' > '[]' > '≫' > '≫':

Definition 3.5. Transition relation $R \subseteq S \times A \times S \times P \times PowerSet(O) \times PowerSet(F)$ is the set of transition relations where (1) A is the set of actions; (2) S is the set of states of the CEBE; (3) $P(\Sigma)$: $PowerSet(\Sigma) \to \{TRUE, FALSE\}$ is the set of predicate functions ($=, \geqslant, \leqslant, >, <, \wedge, \vee, \neg$, Timcout, etc.); (4) $O(\Sigma)$: $PowerSet(\Sigma) \to PowerSet(\Sigma)$ is the set of operation functions (:=, +, −, *, /, StartTimer, CancelTimer, etc.); (5) $F(\Sigma)$: $PowerSet(\Sigma) \to PowerSet(\Sigma)$ is the set of functions (i.e. max(), min(), etc.).

The intuitive meaning of a transition $r \in R$ is that if CEBE is in state s and the enabling action a is offered, then the enabling predicate p is evaluated on the current assignment of variables. When p is true, CEBE will go into a new state s' and the environment is updated by the operation o and function f. The absence of predicate could be denoted as $TRUE$, absence of operation or function could be denoted as NIL.

3.2 CEBE based test sequence derivation

The test sequence derivation strategy is defined as two steps:

Step 1. Identify all possible valid IPs (interaction path). An IP is the externally observable track on which a sequence of interactions between the entity and its external environment occurs, starting from the initial external state s_0 and ending at the same state. Any interaction loop (or cycle) in an IP is traveled only once. Interaction path with loop is an interaction path in which one or more loop interactions exist.

Step 2. Generate $SIPs$ (I/O Subpath) from the above IPs by checking the dependencies between transitions in each IP. An SIP is the externally observable track $r_1, \ldots, r_k$ in an IP, where:

1. $r_1=\langle s_{r1},a_{r1},s'_{r1},p_{r1},O_{r1},F_{r1}\rangle$ is the first transition relation which satisfies:

a) a predicate $p_{r1}(\Sigma_{pr1})$, and/or

b) a set of operations $\{\ldots,o_{r1i}(\Sigma_{or1i}),\ldots\}$, and/or

c) a set of functions $\{\ldots,f_{r1i}(\Sigma_{fr1i}),\ldots\}$;

2. $r_k=\langle s_{rk},a_{rk},s'_{rk},p_{rk},O_{rk},F_{rk}\rangle$ is obtained when logical relations satisfy:

a) $(\forall i,(\Sigma_{prk}\bigcap o_{r1i}(\Sigma_{or1i}))\neq\varphi)\vee(\forall i,(\Sigma_{brk}\bigcap f_{r1i}(\Sigma_{fr1i}))\neq\varnothing)\vee(\forall i,\forall j,(\Sigma_{orkj}\bigcap O_{r1i}(\Sigma_{or1i}))\neq\varnothing)\vee(\forall i,\forall j,(\Sigma_{orkj}\bigcap f_{r1i}(\Sigma_{fr1i}))\neq\varnothing)\vee(\forall i,\forall j,(\Sigma_{frkj}\bigcap o_{r1i}(\Sigma_{or1i}))\neq\varnothing)\vee(\forall i,\forall j,(\Sigma_{frkj}\bigcap f_{r1i}(\Sigma_{fr1i}))\neq\varnothing)$, and

b) a predicate $p_k(\Sigma_{prk})$, and/or

c) a set of operations $\{\ldots,o_{kj}(\Sigma_{orkj}),\ldots\}$ and/or

d) a set of functions $\{\ldots,f_{kj}(\Sigma_{frkj}),\ldots\}$.

3.3 The C-TTCN test suite generation

Based on the SIP, a test tree with the correctness, nondeterminstic and defense branches can be derived, and three kinds of verdicts (PASS, FAIL and INCONC) are assigned to each leaf of the test tree. The test tree can be mapped to a test case in C-TTCN. Finally, these test cases are grouped according to some rules, and are mapped to a complete test suite in C-TTCN. To understand the algorithm, here we use TBE to describe the rules for C-TTCN generation.

Rule 1. Each SIP is mapped to one C-TTCN test case.

Rule 2. In one SIP, if there are several relations $r_1,r_2,\ldots,r_n$, then attach tree $i+1$ to tree i.

TBE_M: $r_1\gg r_2\gg\ldots\gg r_n$

Rule 3. In action a of relation r, if $b_1\gg b_2\gg\ldots\gg b_n$, then $b_1,b_2,\ldots,b_n$ are behaviors and it is successfully completed if b_{i+1} happens immediately after the successful completion of $b_i(i=1,2,\ldots,n-1)$.

TBE_M: create $P1$ TBE1;start $t1$; (CP1? CM_1; down P1; stop) [] (timeout $t1$; stop)

TBE_1: $b_1\gg b_2\gg\ldots\gg b_n\gg$ (CPi! CM_i; stop)

Rule 4. In action a of relation r, if $b_1[]b_2[]\ldots[]b_n$, then an alternative is matched if the behavior b_i corresponding to the alternative happens first.

TBE_M: create $P1$ TBE_1; start $t1$; (CP1? CM_1; down $P1$; stop) [] (timeout $t1$; stop)

TBE_1: ($b_1\gg$ (CPi! CM_i; stop)) [] ($b_2\gg$ (CPi! CM_i; stop))[]...[]($b_n\gg$(CPi! CM_i; stop)

Rule 5. In action a of relation r, if $b_1\parallel b_2\parallel\ldots\parallel b_n$, then $b_1,b_2,\ldots,b_n$ are started in parallel and the parallel tree completes only if $b_1,b_2,\ldots,b_n$ complete.

TBE_M: create $P1$ TBE_1; ... ; create P_n TBE_n; start $t1$; (CP1? CM_1; down $P1$; start $t2$;(...; (CPn? CM_n; donePn; stop) [] (timeout tn; stop));...) [] (timeout t2; stop) []...[] (CPn? CM_n; down Pn; start t2;(.....) [] (timeout t2; stop)) [] (timeout t1;stop)

TBE_i: $b_i\gg$(CPi! CM_i;stop)

Rule 6. In each behavior b of action a in a relation r, there exists $\langle g_i,?\ r_i(r_i.p_1,r_i.p_2,\ldots,r_i.p_n),!\ s_i(s_i.p_1,s_i.p_2,\ldots,s_i.p_n)\rangle$, then

TBE_i: $g_i!\ r_i;g_i?\ s_i$; CP1! CM_1; stop

Rule 7. In the declaration part, the timer, variable and constant could be mapped from TIMER, VAR, and CONS with the initial value from v_0.

Rule 8. In declaration part, *ASP*, *PDU* could be mapped from *IO* with parameters *PARA* for the events.

Rule 9. In constraint part, the timer, variable, *ASP* and *PDU* could be mapped from each event $e_i(e_i.p_1,e_i.p_2,\ldots,e_i.p_n)$ in b with the values of parameters.

4 Conclusions

In this paper, we use C-TTCN to meet the needs of OSPF routing protocol testing. The real testing is accomplished by an integrated test system PITS over Sun workstation and with Solaris operating system. PITS is a general system with C-TTCN operational schematics based test execution. In future, we would perform more real tests to verify our methods, test generation tool and test system. The CEBE and C-TTCN based concepts and method in this paper may be further used in the area of conformance testing in distributed systems. For example, CEBE could be used to model most entities in OSI, TCP/IP and ODP.

References

1 Pickin S *et al*. An approach to the validation of open object-based distributed applications. In: Baumgarten B ed. Testing of Communicating Systems. London: Kluwer Academic Publisher, 1996,9:115～121

2 Wong A C Y, Chanson S T, Cheung S C *et al*. A framework for distributed object-oriented testing. In: Higashino T ed. Formal Description Techniques. London: Kluwer Academic Publisher, 1997,10:39～56

3 Wu J, Chanson S T. Testing sequence derivation based on external behavior expression. In: Chanson S T ed. IFIP International Workshop on Protocol Test Systems. Norh-Hollan: Chap & Hall, 1989

4 Wu J, Chanson S T. Translation from LOTOS and Estelle specifications to extended transition system and its verification. In: Hogrefe D ed. IFIP International Conference on Formal Description Techniques for Distributed Systems. North-Holland: Chap & Hall, 1989

基于形式化方法的因特网路由协议的一致性测试

毕军 吴建平

(清华大学计算机科学与技术系 北京 100084)

摘要 并发数表组合表述法(并发 TTCN)是可以描述并发测试行为的测试表述法.该文提出一种基于并发 TTCN 的分布式路由协议的测试方法.首先讨论路由协议实体的测试结构,然后给出基于并发 TTCN 测试系统的设计.最后介绍了测试集的设计.

关键词 一致行测试,分布式系统,路由协议,因特网,并发数表组合表述法.

中图法分类号 TP393