面向 Java 的分布式程序测试系统*

顾 庆+, 陈道蓄, 谢 立, 孙钟秀

(南京大学 计算机软件新技术国家重点实验室,江苏 南京 210093)

A Java-Oriented Distributed Program Testing System

GU Qing+, CHEN Dao-Xu, XIE Li, SUN Zhong-Xiu

(State Key Laboratory for Novel Software Technology, Nanjing University, Nanjing 210093, China)

+ Corresponding author: Phn: 86-25-3592339, Fax: 86-25-3300710, E-mail: guq@nju.edu.cn

http://cs.nju.edu.cn

Received 2002-09-03; Accepted 2002-12-04

Gu Q, Chen DX, Xie L, Sun ZX. A Java-oriented distributed program testing system. *Journal of Software*, 2003,14(4):743~749.

Abstract: Because the program is running in a distributed way, both the concurrent features and the runtime environments should be taken into account when testing a distributed program. A Java oriented distributed program test system is put forward in this paper, which is called JDPT (Java-oriented distributed program testing system). JDPT defines events based on the runtime environments, records execution processes of the distributed program as event sequences, and defines event sequencing constraints to check the validity of those feasible set of event sequences. By the technology, the JDPT can effectively estimate the correctness of concurrent executions of the tested program within distributed environments, and can be used to test Java programs running upon multiple platforms.

Key words: software testing; distributed program testing; Java; concurrent program; event sequencing constraints

摘 要: 由于程序的分布运行,测试分布式程序必须同时考虑并发特性和运行环境.介绍了一个面向 Java 语言的分布式程序测试系统 JDPT(Java-oriented distributed program testing system).JDPT 基于运行环境定义事件,通过事件序列记录分布式程序的运行过程,并定义事件约束检测可行事件序列集的有效性.通过该技术,JDPT 可以有效地判断程序在运行环境中并发执行的正确性,适用于跨平台的 Java 程序测试.

关键词: 软件测试;分布式程序测试;Java;并发程序;事件约束

中图法分类号: TP311 文献标识码: A

随着 Internet 的普及和计算机软、硬件技术的发展,分布式程序和应用已成为今后软件开发的主要方向之一.与集中式的串行程序相比,分布式程序需要考虑与运行环境的合作以及分布单元间的并发问题,这为分布式

* Supported by the Key Science-Technology Project of the National 'Ninth Five-Year-Plan' of China under Grant No.98-780-01-07-03 (国家"九五"重点科技攻关项目); the National High-Tech Research and Development Program Plan of China under Grant No.2001AA113090 (国家高技术研究发展计划(863))

第一作者简介: 顾庆(1972－),男,江苏常州人,博士,副教授,主要研究领域为分布式语言和系统.

程序测试工具带来了新的挑战.

保障分布式程序的正确性需要从两方面入手:一是从分布式程序自身出发,侧重并发程序的形式化表示和验证;二是在分布运行环境中监控和记录分布式程序的执行过程,分析程序在各种运行环境中的执行状态和性能.测试需要结合这两个方面,从形式化表示中获得规约,在运行环境中检测程序实现与规约的一致性.

目前已有的分布式程序测试工具如 MAD(monitoring and debugging environment)[1],PSET(distributed program structure and event trace)[2],OrcShot[3],SPIN[4],LTSA[5]等都没有很好地考虑这两方面的结合.其中 MAD,PSET 和 OrcShot 偏重运行环境中的检测,缺乏判断程序正确性的依据;SPIN 和 LTSA 则偏重并发程序的验证,不能确保程序实现与运行环境的有效协作.本文提出的测试系统 JDPT(Java-oriented distributed program testing system)较为有效地解决了这两方面的结合问题.

1 分布式程序测试

1.1 分布式程序测试的主要问题

分布式程序由多个相互协作的分布单元构成.这些分布单元相对独立且位于网络中不同的机器上,单元间通过消息传递实现协作.在网络环境中,支撑分布式程序运行需要一个层次式的协议架构,包括通信平台(如 TCP/IP)和支撑系统(如 common object request broker architecture,简称 CORBA),两者构成分布式程序的运行环境.基于此,测试分布式程序主要需考虑以下两个方面的问题:

- 程序自身的问题.对于多个相对独立的分布单元,需要处理其间的通信、同步以及它们的并发操作.
- 分布运行环境的问题.包括程序与运行环境的合作、网络通信的完整性(通信信息无差错、无重复、不失序)、平台无关性以及异构系统间的互操作性等.

程序测试需要考虑测试的正确性和充分性.由于并发所导致的不确定性,以同一个测试用例运行同一个分布式程序,多次执行也会产生不同的运行结果.这时需要不同的测试手段,如以事件序列记录程序的运行过程、采用确定性测试以事件序列作为测试用例等.并发的多样性导致分布式程序难以得到充分的测试,这时需要定义新的充分性准则,采用新的测试用例选择方法等.

1.2 E-CSPE约束

E-CSPE(extended CSPE)[6]是在 CSPE(constraints on succeeding and proceeding events)基础上提出的一个事件约束描述规则.E-CSPE 约束是指采用 E-CSPE 描述的事件约束.E-CSPE 约束在给定状态谓词下定义前后两个事件间的依赖关系以及这种关系的或然性.E-CSPE 约束的基本形式是 $op[[evt_1;\rightarrow evt_2]]_U C$.其中,$U$ 是被测程序;evt_1 和 evt_2 是前后相关的两个运行事件,它们不需要紧密相连;C 是一个状态谓词,它决定 evt_1 和 evt_2 之间的依赖关系是否存在;op 规定依赖关系的或然性.op 有 3 种取值,对应 3 类不同的 E-CSPE 约束,具体如下:

- $a[[evt_1;\rightarrow evt_2]]_U C$:(always under condition),在程序 U 的运行过程中,若条件 C 成立,则 evt_2 发生于 evt_1 之后总有效.
- $m[[evt_1;\rightarrow evt_2]]_U C$:(must under condition),在程序 U 的运行过程中,若条件 C 成立,则 evt_1 发生以后 evt_2 必发生.
- $\sim[[evt_1;\rightarrow evt_2]]_U C$:(never under condition),在程序 U 的运行过程中,若条件 C 成立,则 evt_2 发生于 evt_1 之后总无效.

任意给定 U 的一个可行事件序列 s 和针对 U 的一个 E-CSPE 约束 r,s 与 r 之间可定义 3 种关系:(1) s 覆盖 r; (2) s 违反 r; (3) s 与 r 无关.针对约束 r,根据上述 3 种关系可以定义:

- *ConformSet*(r,U):所有覆盖 r 的可行事件序列集合;
- *ViolateSet*(r,U):所有违反 r 的可行事件序列集合;
- *NonsenseSet*(r,U):所有与 r 无关的可行事件序列组成的集合.

令 *Feasible*(U)为程序 U 所有可行事件序列的集合,显然有

$$ConformSet(r, U) \cup ViolateSet(r, U) \cup NonsenseSet(r, U) = Feasible(U).$$

上述集合定义可以作为测试时的依据,具体为(设 *CSET* 为针对 *U* 定义的 E-CSPE 约束集合):

- 若 $\bigcap_{\forall r\in CSET} NonsenseSet(r,U)=\{\}$,则针对任一可行事件序列 *s*,*CSET* 中总存在与 *s* 相关(覆盖或违反)的约束 *r*.可以认为约束集 *CSET* 相对于 *U* 的实现是充分的.
- 若 $\forall r\in CSET.ViolateSet(r,U)=\{\}$,则针对 *CSET* 中的任一约束 *r*,不存在违反 *r* 的可行事件序列.可以认为 *U* 的实现相对于约束集 *CSET* 是正确的.

1.3 并发程序FSP表示和E-CSPE约束推导

FSP(finite state processes)[5]是 LTSA(labeled transition system analyzer)中采用的一类进程代数记法.它可以描述一个并发程序并验证其行为(behavior)与规约的一致性.FSP 的优点是可以较方便地从代数记法转换成对应的 Java 程序.

在 FSP 中,分布单元被定义为进程(process),进程按递归的形式定义并抽象为动作序列,对应的状态机表示为 LTS(labeled transition system).对于生产者-消费者问题,可以采用 FSP 表示如下:

WAREHOUSE(*N*)=*GSET*[0],

GSET[*content*:0..*N*]=(when (*content*<*N*) *put*→*GSET*[*content*+1]

|when (*content*>0) *get*→*GSET*[*content*–1]),

PRODUCER=(*put*→*PRODUCER*),

CONSUMER=(*get*→*CONSUMER*),

||*PRO_CON*=(*pro*[1..3]:*PRODUCER* || *con*[1..3]:*CONSUMER* || {*pro*[1..3], *con*[1..3]}::*WAREHOUSE*(5)).

其中由大写字母表示的是进程,小写字母表示的是动作.FSP 中的基本操作符是→,表示动作间的顺序.其他操作符包括:选择“|”、条件控制“when”、进程前缀“:”、进程共享“::”、并发组合“||”等.在操作符的基础上,FSP 提供 3 种并发控制机制:共享动作、互斥及条件同步.在生产者-消费者的定义中即包括共享动作和条件同步机制.

在操作符和并发控制机制的基础上可以做 E-CSPE 约束的推导[7].例如,共享动作涉及安全性,可据以推导~类和 *m* 类约束;互斥和条件同步涉及安全性和进展性,可推导出 *a* 类、*m* 类和~类约束.

2 基于 Java 语言的测试系统 JDPT

2.1 JDPT的功能和特点

基于上述考虑,我们实现了一个针对 Java 语言的测试系统 JDPT.JDPT 的主要功能如下:

- 以图视的形式展示被测程序的控制结构和执行分布情况;
- 根据程序的形式化描述推导 E-CSPE 约束,并判断约束集的一致性;
- 与支撑系统协作,动态监控程序的分布运行;
- 利用 E-CSPE 约束检测收集到的可行事件序列;
- 计算程序在分布运行环境中运行时的性能指标.

JDPT 系统有如下一些特点:

- 以 Java 语言为基础;
- 强调形式化验证与运行监控分析的结合;
- 以事件序列的形式记录程序在运行环境中的执行过程;
- 强调对程序中分布特性的测试.

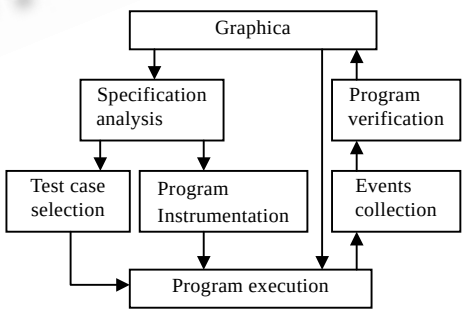

Fig.1 System structure of JDPT

图 1 JDPT 的系统结构

2.2 JDPT的构成

根据 TFDS(test system framework for distributed software system)[8]框架,JDPT 的系统结构如图 1 所示.其中,规约分析、程序装备和选择测试用例模块属于静态部分;程序执行、事件收集和程序验证属于动态部分;图视

界面是系统与用户的接口.

(1) 规约分析.包括两个方面.一方面是描述并分析被测程序的 FSP 表示,据以推导用于测试的 E-CSPE 约束.另一方面是分析被测程序的 Java 源码,获得程序的控制结构,主要是分布单元间的调用结构;并确定测试代码的装备位置.

测试代码的功能是生成和记录运行事件,主要针对的是 Java 中的同步语句和通信设施.例如考虑 Java 的 RMI(remote method invocation)设施:RMI 调用需要分两步,对应两段代码结构.第一步是 Bind,用于 Client 对象(如 Producer 或 Consumer)定位提供方法调用的服务对象(如 WareHouse),并在本地创建服务对象的 stub,代码结构如下:

```
try {①
    wareHouse=(WareHouse) Naming.lookup("//"+serverAddress+"/WareHouseServer");
    ② … /* other codes */
} catch (Exception e) {③/* code related to except manipulation */}
```

这段代码中可以生成 3 个事件:① Bind 开始;② Bind 成功;③ Bind 异常.

定位并创建本地 stub 以后,第 2 步是通过它进行 RMI 方法调用.RMI 调用的代码结构如下:

```
try {④
    wareHouse.put(parameters);
⑤} catch (Exception e) {⑥ /* code related to except manipulation */}
```

对应的服务对象类(WareHouse)需要实现被调用的方法,代码结构如下:

```
public class WareHouseImpl extends UnicastRemoteObjectserverClass
      implements WareHouse { …
      public synchronized void put(parameters) {⑦
          … /* implementation code */
          ⑧ return;
} …}
```

由 WareHouseImpl 实现的 WareHouse 是一个界面类,JavaRMI 需要用它来生成 stub 和 skeleton 代码.在上述两个代码段中需要生成 5 个事件:④ 本地 RMI 调用;⑤本地 RMI 成功返回;⑥ RMI 异常;⑦ 服务端 RMI 接收;⑧ 服务端 RMI 结束.

(2) 程序装备.在程序结构和规约分析的基础上,在源程序中装备测试代码.测试代码的主要目的是收集运行信息,并将这些信息记录成事件.在 JDPT 中,事件被定义为六元组$\langle T,S,D,M,I,O\rangle$.各项的含义分别是:

- 类型(type):事件的类型.
- 源对象(source):事件的发起者,由 Java 的对象名或线程标识表示.
- 目标对象(destination):事件针对的目标对象,由 Java 的对象名或线程标识表示.
- 对象方法名(function):指示事件对应的对象方法.
- 时间戳(timestamp):事件的发生时间,用于对事件的排序以及性能分析.
- 其他(other):与事件有关的其他信息,如产生事件的语句及变量值等.

JDPT 为每个对象增加一个 public 方法:CollectInfo,用于生成事件信息;同时定义监控对象 Monitor,其主要的方法是 RecordInfo,用于存储事件信息.装备的代码即由以下两条语句构成:

Self.CollectInfo(…); Monitor.RecordInfo(…).

(3) 选择测试用例.考虑到用户一般通过 Client 对象执行分布式程序,JDPT 依据 Client 对象的 FSM 表示选择测试用例,基于的覆盖准则是变迁覆盖.当存在多个 Client 对象时,取针对单个对象的测试用例集的并集.

JDPT 还根据 E-CSPE 约束选择测试用例.用例的形式一般是事件序列,所达到的目标是使生成的事件序列覆盖或违反某个 E-CSPE 约束,以满足测试的充分性要求.

(4) 程序执行.有两种执行被测程序的方式:一种是不加干预的正常执行,这种方式对应于不确定性测试,同

一测试用例需要执行多次;另一种是在 Monitor 的监控下干预程序的执行,这种方式对应于确定性测试,测试用例一般是事件序列.有两种方法用于确定性测试:

- 在程序的关键位置(如 RMI 调用、synchronized 语句、wait()-notify()函数等)插入交互式或延时语句,以控制程序运行时的竞争条件.
- 设定相邻两个事件间的最大发生间隔,超过这一间隔即判定两个事件不具备前后相邻的关系.

(5) 事件收集.Monitor 对象负责记录和汇总运行事件,当程序规模较大时将有多个 Monitor 实例在运行,此时需要指定一个 Monitor 为 Test Center 负责汇总来自各 Monitor 的事件.事件汇总有两种方式:一种是即时的,所有事件一旦记录下来立即传递给 Test Center,这可能会导致通信拥塞和瓶颈,干扰被测程序的运行;另一种是延迟的,事件先由各 Monitor 记录在本地的缓冲区或文件中,再按一定的周期或者应 Test Center 的请求将本地事件序列提交给 Test Center.

由于通信延迟等原因,Test Center 在汇总事件时需要为事件排序.可以采用时间戳定义逻辑时间为事件排序,通过这种方法,有因果关系的事件(如 send 和对应的 receive)可以得到准确的排序,但无此关系的事件其前后顺序就可能是随机的.另一种方案是使用绝对时间,它的好处是可以得到惟一排序,并为性能分析提供了依据;缺点是各台机器的本地时间难以保证完全同步.

(6) 程序验证.在 JDPT 中,程序验证包含两个方面,一个是检测程序运行的正确性,另一个是分析程序在运行环境中执行的性能指标.检测程序的正确性需要分两个阶段:一个是就运行事件本身检查是否出现了异常事件并检查事件间的匹配情况;二是根据 E-CSPE 约束判定事件序列的有效性.判定方法如下:(*CSET* 为 E-CSPE 约束集,*F* 为测试产生的事件序列集,*s* 为一事件序列)

- $\forall r \in CSET.\forall s \in F.s \notin ViolateSet(r)$,说明当前产生的事件序列集没有错误;
- $\forall r \in CSET.\exists s \in F.s \in ConformSet(r)$,说明当前产生的事件序列集针对 CSET 充分.

分析程序的性能时主要是计算各种性能指标,如各分布单元的执行时间、通信时间和等待时间.发现程序运行中的瓶颈,包括资源瓶颈、通信瓶颈等.

(7) 图视界面.图视界面是用户使用 JDPT 的接口.主要完成的功能包括:通过菜单界面调用其他模块;提供编辑界面,用于源程序编写、程序形式化描述以及 E-CSPE 约束的定义等;以视图的形式反馈程序的测试结果,包括规约分析产生的控制流图以及程序验证时的事件序列图(trace 图)等.

2.3 测试实例

图 2 显示了应用 JDPT 测试生产者-消费者问题的测试过程.如图 2 所示,测试一个程序分为两条线:一条是从程序实现出发,考虑结构分析、测试代码装备以及运行监控和事件收集;另一条是从程序形式化描述出发,根据 FSP 表示推导 E-CSPE 约束,再以 E-CSPE 约束检测事件序列集的有效性.通过这种方式,JDPT 系统较好地实现了形式化验证与运行监控分析的有机结合.

3 相关工作比较

保证分布式程序的正确性可以从两方面入手:一方面是将分布式程序视为并发程序,偏重于并发程序的形式化验证;另一方面是考虑程序实现,强调分布式程序运行过程的监控、记录和分析.

形式化验证主要是通过代数演算和逻辑推理证明程序并发过程的正确性,包括安全属性和进展属性,不考虑程序的具体实现和运行环境.以 SPIN 系统[4]为例,SPIN 的目的是证明进程间交互(interaction)的正确性.其中进程被定义为一个自动机,它以顺序、确定的方式运行.进程间交互有 3 种方式:会合元语(rendezvous primitives)、缓冲信道(buffered channel)通信和共享变量(shared variables)访问.多进程异步交互(asynchronous interleaving product of automata)构成系统行为,这个系统行为也被描述为自动机.SPIN 包含两种描述语言:用于描述程序规约要求的时序逻辑 LTL(standard linear temporal logic)和描述设计选择(design choices,即程序形式化描述)的语言 PROMELA(process meta language).程序正确性验证即要证明两种描述在逻辑上的一致性.验证时涉及 3 个基本实体:进程,包括有限个数的状态和谓词条件下的变迁;变量,有限个数的取值;缓冲信道,有限的

缓冲容量.

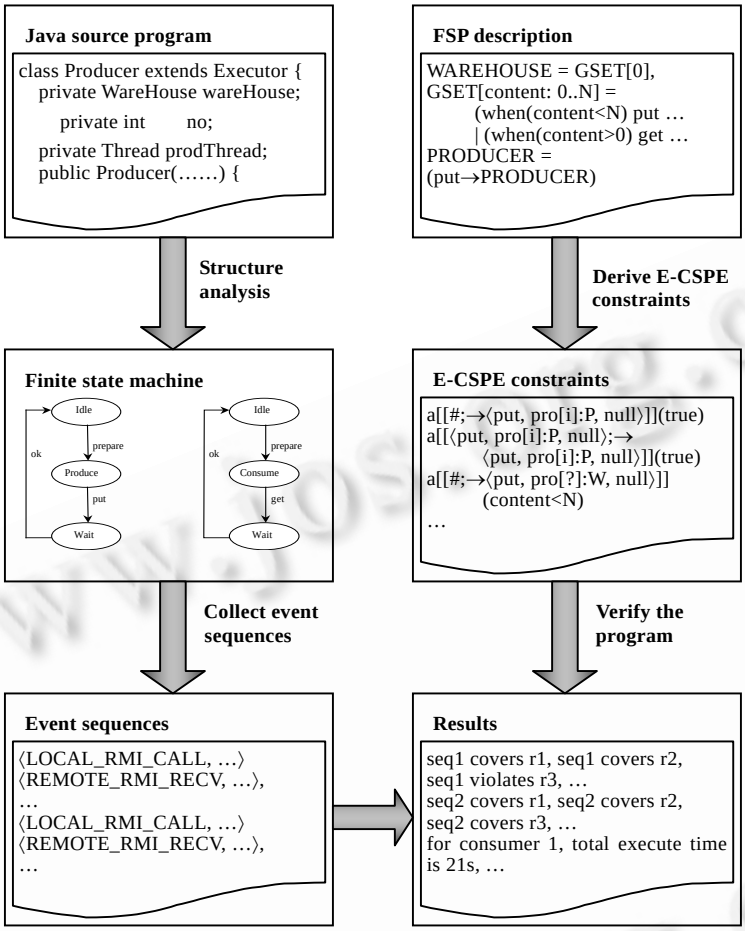

Fig.2 Test procedure of producer-consumer problem

图 2 生产者-消费者问题的测试过程

运行监控分析则将实现的分布式程序置于运行环境中执行,记录并控制其运行过程,对记录信息进行正确性和性能分析,然后把运行过程以图视的形式显示给用户.以 MAD(monitoring and debugging environment)系统为例,MAD 的主要目的是探测程序中的错误.它把程序错误分成两类:只影响一个进程的局部错误以及影响多个进程的全局错误.局部错误可以由串行程序测试工具检测出来,MAD 不予考虑.对于全局错误的检测,由于滚雪球效应(stampede effect),MAD 强调竞争条件(race condition)的探察.涉及竞争条件的事件被称为“race condition candidates”,这类事件的发生顺序是不确定的.MAD 包含 3 个基本工具:EMU(event monitoring utility),功能是装备测试代码,在程序运行时收集和记录运行事件并存入文件中;ATEMPT,功能是分析运行事件信息,检查通信中的错误,计算性能指标包括通信时间、执行时间和监控耗时,显示和操纵运行进程的 trace 图;PARASIT,功能是以事件序列(重排“race condition candidates”的顺序)为输入确定性地执行被测程序.

只考虑形式化验证不能保证程序实现与运行环境的有效协作;只考虑运行监控和分析又缺乏对程序行为的正确性约束.JDPT 实现了两者间的一个桥接:形式化验证可以保证程序形式化描述的正确性,据之推导的 E-CSPE 约束又可反映规约的要求,进而有效地检测程序实现行为与规约行为的一致性.

4 结束语

由于网络的普及和分布应用的拓展,用于检测分布式程序正确性和性能的测试工具有着广阔的应用前景.

目前针对分布式程序测试的研究尚处于初期阶段.就 JDPT 系统而言,下一步的工作包括:

- 如何根据程序实现推导描述可行事件序列特性的 E-CSPE 约束,并对比从规约导出的 E-CSPE 约束,验证程序实现的正确性.
- 如何重构 JDPT,使其支持其他类型的分布式编程语言并测试其他类型的分布式应用.

目前分布式支撑系统如 CORBA 等本身尚存在缺陷,需要增加对运行环境组成构件的测试.

致谢 感谢实现 JDPT 的诸位老师和同学,感谢评审专家的宝贵意见.

References:

[1] Kranzlmüller D, Grabner S, Volkert J. Debugging with the MAD environment. Parallel Computing, 1997,23(1):199~217.

[2] Gu Q, Chen DX, Xie L. PSET: A validation system for object-oriented distributed programming language named NC++. Journal of Software, 1997,8(6):352~356 (in Chinese with English abstract).

[3] Hofman R, Langendoen K, Bal H. Visualizing high_level communication and synchronization. In: Narasimhan I, Lakshmi V, eds. Proceedings of the 2nd IEEE International Conference on Algorithms and Architectures for Parallel Processing (ICA3PP'96). Singapore: Institute of Electrical and Electronics Engineers, Inc., 1996. 37~43.

[4] Holzmann GJ. The model checker SPIN. IEEE Transactions on Software Engineering, 1997,23(5):279~295.

[5] Magee J, Kramer J. Concurrency: State Models & Java Programs. Indianapolis: Wiley, 1999.

[6] Gu Q, Chen DX, Yu M, Xie L, Sun ZX. Validation test of distributed program based on events sequencing constraints. Journal of Software, 2000,11(8):1035~1040 (in Chinese with English abstract).

[7] Gu Q, Chen DX, Xie L, Han J, Sun ZX. Event constraints definition based on finite state process. Journal of Software, 2002,13(11): 2162~2168 (in Chinese with English abstract).

[8] Gu Q, Chen DX, Han J, Xie L, Sun ZX. A system framework for distributed programming test. Journal of Software, 2000,11(8): 1053~1059 (in Chinese with English abstract).

附中文参考文献:

[2] 顾庆,陈道蓄,谢立.基于面向对象的分布式程序设计语言 NC++的测试系统.软件学报,1997,8(6):352~356.

[6] 顾庆,陈道蓄,于勐,谢立,孙钟秀.基于事件约束的分布式程序正确性测试.软件学报,2000,11(8):1035~1040.

[7] 顾庆,陈道蓄,谢立,韩杰,孙钟秀.基于有限状态进程的事件约束定义.软件学报,2002,13(11):2162~2168.

[8] 顾庆,陈道蓄,韩杰,谢立,孙钟秀.一个面向分布式程序的测试系统框架.软件学报,2000,11(8):1053~1059.