

基于配置前置约束算法的 NP 测试系统的研究

李德明, 叶 进

(桂林电子科技大学 信息与通信学院, 广西 桂林 541004)

摘 要: 本文在介绍了现有主流自动测试系统的相关技术基础之上, 针对网络处理器的工作特点, 设计了一款具有基于配置前置约束的用例自动生成算法的关键词驱动的 NP 自动化测试系统. 实现了从测试用例自动生成、测试序列自动生成、自动执行测试到输出测试结果的全自动 NP 测试系统框架, 在实际对比应用中, 由于采用了用例自动生成技术, 该测试系统在测试效率和检出问题方面均比原有半自动测试系统有大幅度的提升, 并取得了良好的应用效果.

关键词: 网络处理器; 自动化测试; 用例自动生成; 关键词驱动; 系统架构

中图分类号: TP393

文献标识码: A

文章编号: 1000-7180(2013)05-0063-04

Research of NP Auto-testing System Based on Pre-configuration Condition Constrains Algorithm

LI De-ming, YIE Jin

(School of Information and Communication, Guilin University of Electronic Technology, Guilin 541004, China)

Abstract: This paper describes related technologies used by the existing mainstream automated test systems, and proposed and designed a keyword-driven NP automated testing system with pre-configuration constraints testing case automatic generation algorithm. The system realized all procedure from the automatic generation of case, automatically generate the test sequence, and automatically performs the test to the output test results automatically. By actually contrast applications, due to the technology of test case automatic generation, the system significantly improved test efficiency and ability of problem-detection than the original test system.

Key words: network processor; automated testing; test case automatic generation; keyword-driven; system architecture

1 引言

作为网络设备的核心部件, 网络处理器(Network Processor, NP)是一种针对网络传输的专门 CPU, 不仅有着复杂的内部逻辑结构, 还拥有一套完整的专用指令系统用于加速其网络处理能力; 除此之外, 相对于通用的 CPU, NP 还拥有更多、更为复杂的外部接口. 因此, 在设计开发一款新的网络处理器产品时, 不仅需要在设计及实现阶段进行系统周密的单元测试(Unit Test), 更需要在系统集成状态下对其各个逻辑模块及整体功能和性能进行全面

细致的系统集成测试(System Integrated Test). 而对于如此复杂的对象, 仅使用传统手工测试将难以保证测试效率及质量. 因此, 本文提出了一款具有配置前置约束(pre-configuration condition constrains, PCC)算法的关键词驱动(Keyword Driven)的自动化测试系统(ATS)^[1]来完成对网络处理器复杂的集成测试.

2 平台的总体架构

鉴于 NP 处理器结构复杂模块众多, 且各个模块均可配置成不同的工作模式, 并影响自身及其它

的模块处理结果的特点,同时针对后继不同型号处理器的测试工作,系统采用了易于升级扩展的层次架构(Layered Architecture^[2])的应用模型,依据系统功能将系统逻辑分为:用例层、解析层、测试层、适配层等四个层次.每个层次完成相对独立的功能,并设计成相应的系统构件,构件之间以 XML 及 SOAP^[3]标准协议进行通信.系统架构如图 1 所示.

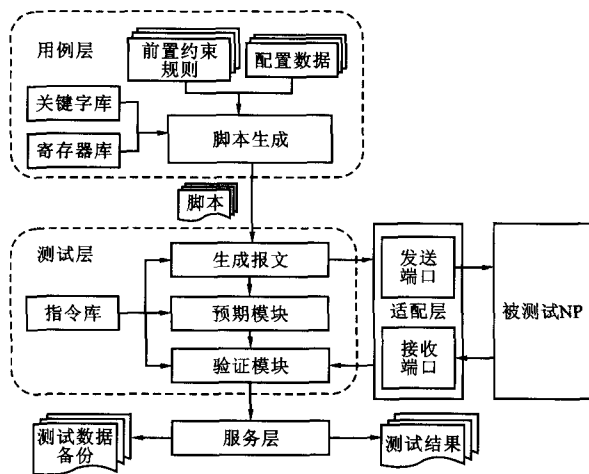


图 1 自动测试系统的系统架构

(1)用例层(Test case Layer)功能是根据用户设置的被测对象的配置信息及其前置约束条件等相关信息,按配置前置约束算法自动生成测试用例,并按测试层的需求输出相应的测试脚本。

(2)测试层(Testing Layer)是功能是运行测试脚本并自动完成相关测试.它主要由生成报文、预期及验证三个模块构成.生成报文模块可按测试脚本的设计自动生成相应顺序及内容的输入协议报文(测试数据);预期模块可根据系统的配置信息及生成的输入报文的顺序及内容自动预期 NP 处理后的预期结果;验证模块可将预期模块所预期的结果与被测试对象返回的实际输出结果进行比对验证,完成包括报文输出顺序、指令处理验证、报文内容验证及寄存器结果验证等验证过程。

(3)服务层(Service Layer)的功能是能测试层的测试结果按指定的形式生成测试结果分析报告,并自动将每个测试用例运行生成的输入数据、实际输出结果及测试日志按需求进行相应备份,以便对疑难问题的诊断提供必要的信息。

(4)适配层(Adapter Layer)主要提供了测试系统与测试对象的数据交互的软、硬件接口,通过此接口可实现设置及访问被测 NP 相关寄存器;实现测试层与被测对象间测试报文的发送及接收功能;并可通过对接口的重定义以适应 NP 的测试。

3 配置前置约束用例自动生成算法

测试用例是自动测试系统的基础,也决定着测试系统功能实现.目前,测试用例自动生成技术已开始应用在各种测试系统之中.但这些用例自动生成技术大多是针对软件的结构测试中的测试数据自动生成问题(TDGP),即按指定的程序中的节点,分支、控制流构成的路径生成相应的测试数据进行覆盖.如 M. Dorigo 提出的基于蚁群算法^[4],J. Kennedy 与 R. Eberhart 提出的粒子群优化算法(Particle Swarm Optimization)^[5],以及 Sthamer H 的遗传算法^[6]等.而针对复杂系统的功能测试的用例自动生成技术及相应工具及理论的研究却相对较少。

目前,针对功能测试的测试用例自动生成技术主要有:Tsai 等^[7]提出的从关系代数查询表示的规格说明中生成测试用例的方法,其关系代数查询表示的规格适合于数据库及其数据处理过程领域的描述,不适合描述大型复杂系统以及交互式系统测试;Weyuker 等^[8]提出了基于布尔规格说明的测试用例生成方法,这种布尔规格说明方法使用 AND/OR 来描述测试规格说明中的条件,这种方法更适用于过程控制系统的测试;Hall^[9]提出的利用 z 规格说明生成测试用例方法.它是基于模式的形式化规格语言,以谓词逻辑和集合论作为形式语言的基础,通过构建形式化的软件需求规约,产生测试用例,这种方法构建的形式化规约过于繁琐,而且,没有考虑实时数据交互的因素;李翔等提出了一种基于事件关系的交互式软件测试方法^[10],从交互式软件具有的事件驱动运行特征出发,根据事件关系设计测试用例;谢棠棠等提出了基于 UML 模型来构造测试用例^[11].这些功能测试方法通过描述对象的动态交互来描述软件功能的实现,但以上两种方法生成的测试用例未考虑软件的具体需求和逻辑流程,因此,产生测试用例的随意性较大,无法判定系统运行中的实际路径集合,难以判断测试用例的覆盖程度,也不利于有序地测试业务流程并定位故障。

为此,本文专门针对 NP 处理器这种内部处理过程明确且各模块均可实时配置,但又不需关心其实现过程的复杂系统的功能测试,提出了基于配置项前置约束(CPC)用例自动生成算法.在介绍具体算法之前,先给出一组相关定义。

定义 1 如果在测试配置项 A 之前,必须将配置项 $\{PreA_1, PreA_2 \dots, PreA_i\}$, $(i \geq 1)$ 配置成 $\{vA_1,$

$vA_2 \cdots vA_i \} (i > 1)$,那么就称配置项序列 $\{ \text{Pre}A_1, \text{Pre}A_2 \cdots, \text{Pre}A_i \} (i > 1)$ 是配置项 A 的前置约束,逻辑表达式为 $CA \{ \text{Pre}A_1, \text{Pre}A_2 \cdots, \text{Pre}A_i \}, i \geq 1$,配置项值序列 $\{ vA_1, vA_2 \cdots vA_i \} (i > 1)$ 为配置项 A 的前置约束值,记为 vCA .

定义 2 若在配置项 A 的非空前置约束 $CA = \{ \text{Pre}A_1, \text{Pre}A_2 \cdots, \text{Pre}A_n \}$,则称 $\text{Pre}A_i (1 \leq i \leq n)$ 为配置项 A 为子配置项 SCI (Sub Configuration Item);若配置项 A 不是任何配置项的子配置项,则称配置项 A 为根配置项 RCI (Root Configuration Item).

定义 3 若在配置项 A 的前置约束 $CA \{ \}$ 为空,则称 A 为叶子配置项 LCI (Leaf Configuration Item).

如图 2 所示. $\{ b= vb, e= ve1, d= vd \}$ 即为配置项 a 的前置约束,配置项 b, e, d 亦为配置项 a 的子配置项;图中配置项 a, f 分别为根配置项和叶子配置项.

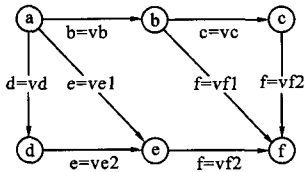


图 2 配置项前置约束示意图

由图 2 及上述定义说明,可以看出,测试配置项 a 时,必须先满足其前置约束条件,即配置项 b, e, d 的值分别为 vb, ve1, vd, 否则配置项 a 的测试条件将不能满足,无法进行测试.

用例自动生成算法是按系统的前置约束条件图,按配置项所需前置约束条件的先后秩序计算出正确的系统测试序列,并按测试序列自动生成用例并依次执行测试的过程. 算法具体步骤如下:

Step1: 遍历所有配置项集合中的结点,按当前结点是否存在于其它结点的子配置中 (SCI), 找到系统的根配置项,并以此作为将要生成的测试生成树的根结点 (RCI), 初始化生成树,并将此结点从配置项集中移除.

Step2: 如果根配置项的前置约束配置项集不为空,则在配置项集中查找所有相应的约束配置项,并依次作为该结点的子节点加入到生成树后从此配置项集中移除.

Step3: 依次以 Step2 中的子节点做为根节点,重复 Step2,即以递归形式得到测试生成树 (TGT), 如图 3 所示.

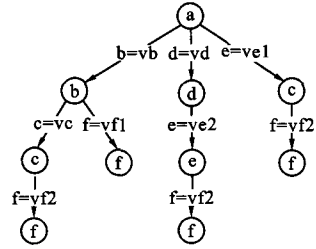


图 3 系统测试生成树

Step4: 将 Step3 得到的 TGT, 以左子树表示孩子结点, 以右子树表示兄结点, 将 TGT 转化成二叉测试生成树 (BTGT), 如图 4 所示.

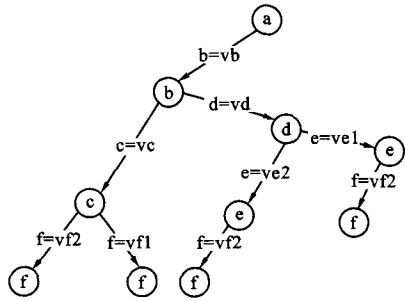


图 4 二叉测试生成树

Step5: 以后序遍历 BTGT, 即可得到系统前置约束算法生成的系统完整测试序列. 如图 5 所示.

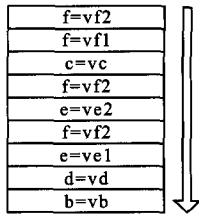


图 5 系统测试序列

Step6: 按系统测试序列及各配置项的配置信息, 按预设的数据选取方式自动生成测试用例集, 从而为后继的测试层等完成进行相应的测试做好准备.

4 测试系统的实现及验证

4.1 测试系统的实现

测试系统在基于现有主流测试工具——NUNIT 进行二次开发即可方便实现. 这样即可充分利用 NUNIT 的强大多语言支持及并行分布式执行测试的特性, 也可降低系统的实现成本, 提高测试工作效率.

用于自动生成测试用例的被测试系统的配置前置约束规则及相关数据可通过 XML 或数据库表进行表示, 其主要结构由表 1 所示.

表 1 配置前置约束条件表主要组成

FieldName	Description
moduleId	配置模块 ID
preModuleId	前置约束模块 ID
preKeyword	前置约束关键字
parameters	前置约束参数
...	...

4.2 新系统的测试对比验证

在测试对比实验中,测试系统分别为华为公司某 NP 先前测试系统,即人工按 FS 进行设计编写用例集,再自动执行测试的测试平台和改进后实现有配置项前置约束用例自动生成算法的测试系统.对比结果如表 2 所示.

表 2 新旧平台测试对比结果

	脚本的数量	验出问题数量	测试周期
原有平台	1261	153	260 天
改进平台	1336	205	100 天

从生成脚本数量上对比来看,两系统的脚本数量基本相当,这是因为:①原来系统存在测试重复问题既冗余脚本;②人工编写脚本含有对系统的异常测试,而自动生成目前还暂未考虑异常测试.

从脚本的测试覆盖全面性上对比,可以看出改进后的测试系统可大大减少人为测试疏漏,对被测对象具有更全面的测试覆盖率.

从测试周期对比结果来看,人工编写测试脚本是提高测试进度的主要瓶颈,而拥有用例自动生成算法的测试平台,可使测试人员只专注于系统的测试设计,而脚本的实现则由系统自动完成,从而大大缩短了测试周期.

综上对比,具有配置项前置约束算法的测试平台可有效减少测试人员的工作量,而且通过对前置约束等相关定义,使得系统测试覆盖率和测试效率大大提高.

5 结束语

本文针对网络处理器的工作及配置的特点,提出了基于配置项前置约束的测试用例自动生成算法的测试系统,从而使测试系统可自动根据测试人员设置的配置项约束规则自动提取测试路径生成测试数据,并自动生成相应的测试用例及脚本,从而大幅度提高了 NP 测试效率及测试质量,也极大地减小了人工编写脚本所引入的人为错误发生的可能,降低了测试人员的工作强度,提高测试工作效率.并在

实际对比应用中取得很好的效果.

参考文献:

[1] 李宝安,李行善. 自动测试系统(ATS)软件的发展及关键技术[J]. 测控技术,2004,22(1):124-127.

[2] Shaw M,Clements P. The golden age of software architecture[J]. IEEE Software,2006,23(2):1129-1139.

[3] SOAP Version 1.2 specification[EB/OL]. [2010-10-15]. <http://www.w3.org/TR/soap/>.

[4] 李士勇,陈永强,李研. 蚁群算法及其应用[M]. 哈尔滨:哈尔滨工业大学出版社,2004.

[5] Kennedy J,Eberhart R. Particle swarm optimization[C]//Proc IEEE Conf on Neural Networks. Sydney: Austrldia. 1998.1942-1948.

[6] Sthamer H,Jones B F. Automatic structural testing using genetic algorithms[J]. Software Engineering Journal,1996,11(5):299-306.

[7] Tsai W T. Volovik D. Automated test case generation for programs specified by relational algebra queries[J]. IEEE Transactions on Software Engineering, 1995,16(3):316-324.

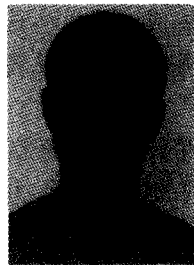
[8] Elaine Weyuker, Tarak Goradia. Automatically generating test data from a Boolean specification[J]. IEEE Transactions on Software Engineering, 1996,20(5):353-363.

[9] Hall P A. Relationship between specifications and testing[J]. Information and software Technology, 1997,33(1):47-55.

[10] 李翔,高建华. 一种基于事件关系的图形界面程序测试方法[J]. 小型微型计算机系统,2007,26(4):671-675.

[11] 谢棠棠,张为群. 一种基于 UML 模型的系统测试方法[J]. 西南师范大学学报:自然科学版,2010,30(2):259-263.

作者简介:



李德明 男,(1973-),硕士,讲师.研究方向为自动测试及网络应用等.

叶进 女,(1971-),博士,教授.研究方向为网络协议、嵌入式系统、分布式及网络应用.