维普资讯 http://www.cqvip.com

用于测试的SA动态模型生成方法

顾燕萍,高建华
(上海师范大学 数理信息学院 计算机科学与工程系,上海 200234)

摘 要:SA(软件体系结构)的动态性描述在基于软件体系结构的一致性测试中是非常关键的一步。许多软件体系结构描述语言是利用带标号的转换系统(LTS)来模拟软件体系结构动态性的,利用LTS作为软件体系结构动态性模型并从中选取测试序列。通过实例研究了两种体系结构描述语言及其分别向动态模型LTS转变的过程。

关键词:带标号的转换系统;有限状态过程;化学抽象机;软件体系结构动态性描述;测试序列
中图分类号:TP31 **文献标识码**:A **文章编号**:1673-629X(2006)08-0100-03

SA Dynamics Description and Its Application in Testing

GU Yan-ping, GAO Jian-hua
(Dept. of Computer Sci. and Eng., Sch. of Math., Physics and Info., Shanghai Normal Univ., Shanghai 200234, China)

Abstract: The SA (software architecture) dynamics description is a critical step in the conformance testing based on SA. Many architectural description languages (ADLS) rely on labelled transition systems (LTS) to model the SA dynamics, and LTS graph could be used as the reference model for deriving the test sequences. This article introduces two ADLS which can derive LTS by a case study.

Key words: LTS; FSP; CHAM; SA dynamics description; test sequences

0 引 言

利用软件体系结构模型来导出测试是目前软件测试研究的热点问题之一。文献[1~3]都提到了基于软件体系结构的测试方法。软件体系结构的动态性是利用带标号的转换系统(LTS)来描述的,它可以模拟软件体系结构动态性规格说明,然后把LTS作为一个参考模型用作测试。所以能否得到一个比较完整准确的LTS,对于测试用例的选取及有效的测试系统都起着至关重要的作用。

目前有两种LTS的描述语言(FSP和CHAM)及其向LTS的转换[1,4,5]。有限状态过程(FSP,Finite State Process)是基于过程的定义,系统中的每个构件都通过一个过程来表示,每个过程中的行为通过LTS模型来表示,同时许多过程可以组合来描述不同过程间的交互。而化学抽象机(CHAM,Chemical Abstract Machine)形式化语言主要用于异步并行计算模型的建模,通过把化学反应和抽象机的概念有机地结合来描述系统状态的变化,从而生成LTS。文章没有直接从FSP,CHAM规格说明中导出测试序列,而是先转换为LTS模型,再从LTS模型中选取测试,这样就使选取变得更直观更方便。文章在介绍FSP和CHAM的基础上,转变为LTS模型。

收稿日期:2005-12-05

作者简介:顾燕萍(1982-),女,上海人,硕士研究生,主要研究方向为软件可靠性设计;高建华,博士,教授,主要从事软件可靠性设计研究。

1 带标号的转换系统(LTS)定义

LTS(Labelled Transition System)是一个五元组,即(S, L, S_0, S_F, T),S是状态的集合,L是特异标号的集合(即actions的集合),$S_0 \in S$是初始状态,$S_F \square S$是终止状态的集合,而$T = \{\rightarrow \square S \times S \mid l \in L\}$是标有元素$L$的转换关系。

一个LTS可以描述一个软件系统,其状态之间的转换是通过某些操作或影响程序运行的因素。

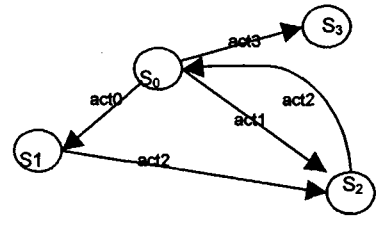

图1 LTS图

图1给出了一个LTS图,其中S_0既是初始状态,同时又是终止状态S_F,act0,act1,act2,act3组成了特异标号集。

其中路径$p = S_0 \xrightarrow{act0} S_1 \xrightarrow{act2} S_2 \xrightarrow{act2} S_0$,就是一条完全路径。为了简洁,LTS路径也可用标号的序列表示:p = act0.act2.act2。

2 LTS导出过程

2.1 NetMeeting实例

以网络视频会议为例,可根据构件和连接器的形式,得到一个静态的软件体系结构描述。在这里,特别识别了

维普资讯 http://www.cqvip.com

3 种构件:

(1)User:送开会请求信息给 Router 处理,在送出信息后等待从 Router 来的确认。

(2)Router:等待 User 来的信号,解析寻找目的地址,向目的服务器提交信息;在收到目的服务器答复后,向 User 发送确认信息。

(3)Server(目的服务器):分派请求信息,并给出答复信息。

系统除结构化的、静态的规格说明外,还需要有构件和连接件间交互的动态描述,这里用 LTS 作为软件体系结构动态性模型。

2.2 FSP 规格说明

在软件体系结构规格说明中,用 FSP 模型可表达规格说明中的动态性部分。FSP 规格说明是基于过程的定义,在上节提到的每种构件,都是通过一个过程来表示的,而每个过程中的行为是通过 LTS 模型来表示的,它包含了一个过程可能到达的所有状态及它可能执行的所有转换。同时许多过程可以组合来描述不同过程间的交互。

使用这一语言的好处在于有 LTSA[4] 工具支持 FSP 语言,可以用此工具自动地从 FSP 规格说明中生成 LTS,用它图形化地描述这个过程的行为。由此可知,每一个过程所对应的 LTS 都能通过此工具,从相应的 FSP 规格说明开始自动建立。

根据上述概念,经分析,得到网络视频会议系统的 FSP 规格说明。前面系统描述时分成的 3 个构件(User, Router, Server)分别用 3 个 FSP 过程表示,分别为 User 过程、Router 过程、Server 过程,这些过程又分别由几个子过程并行而来,例如,User 过程的规格说明表示为:

USER_REQUEST = (sendRequest_To_Router → receiveAck_From_Router→USER_REQUEST)

USER_CHECK = (sendCheck_To_Router → USER_SEND-CHECK)

‖USER=(USER_REQUEST ‖ USER_CHECK)

User 过程是 USER_REQUEST 和 USER_CHECK 过程的并行,所以它们两个过程中的行为是随意穿插进行的,顺序可以是:(sendCheck_To_Route, SendRequest_To_Router, ReceiveAck_From_Router),也可以是:(SendRequest_To_Router, sendCheck_To_Route, ReceiveAck_From_Router)。

其 LTS 图如图 2 所示。

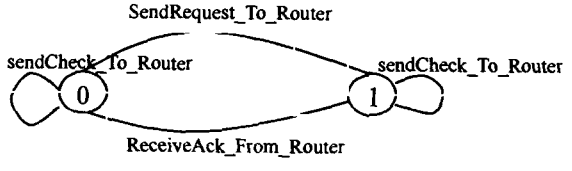

图 2　USER 过程对应的 LTS 图

规格说明最后一部分用来并行系统中的各种过程,即实例中它是并行 User, Router, Server3 个过程的,详细说明各个 LTS 是如何协作的。它确定了系统是如何运转的,也就是表示了 User, Router, Server3 个过程是如何并行来描述整个系统行为的。

由于 LTSA 工具的存在,可以自动化这个转换过程,在每个 FSP 规格说明上利用此工具就可得到不同的 LTS,这些 LTS 可以进行组合,最后产生一个包含有 256 个状态的全局 LTS 模型,它指定了所有系统过程组合而来的行为。

2.3 CHAM 规格说明

2.3.1 CHAM 形式化语义

一个化学抽象机由一组分子 m, m'…、溶液 s, s'…和变换规则 T, T'…组成。分子是由一个常数集和操作符集派生而来的句法代数定义;溶液是有限多个分子的集合,它反应了系统的某种状态;溶液中的分子根据变换规则进行反应,以指示溶液演化的方式 $S \rightarrow S'$。这里用 R 来标识规则集合和相应的标号集合。

用 CHAM 描述的软件体系结构规格说明包括 4 个部分:

* 体系结构组成构件(即分子)的语法描述,构件被分为 3 类:数据元素、处理元素和连接元素。

* 系统的初始状态,用溶液 S_0 表示。初始溶液是所有按照分子语法可能产生的分子的一个子集,它实际上是系统的一个初始的、静态的配置。

* 一套反应规则,通过描述构件间的交互活动以实现系统动态行为的描述。

* 一组表示系统状态变化的溶液集。

我们要求初始溶液包含模拟每一个构件初始状态的所有分子。被应用到初始溶液的变换规则定义了系统是如何从初始配置进行动态进化的。利用这样一个操作上的特点,可以从一个 CHAM 描述中导出 LTS。

2.3.2 CHAM 规格说明导出 LTS

下面的定义给出了导出机制:

定义一:(由 R 引导的操作语义)让 R 作为一个 CHAM 的反应规则集合,那么 R 定义了一个关系 $D \square M \times M$(M 为分子),这个关系是满足规则的最小关系。

定义二:(派生)给定一个反应规则集 R,从溶液 S_0 到 S_n 的 R 派生是一个序列 $\{S_i, 0 \leqslant i \leqslant n\}$, $n > 0$,所以,对于任何 $0 \leqslant i \leqslant n-1$, $S_i \rightarrow_R S_{i+1}$。如果存在从 S_i 到 S_j 的 R 派生,则 S_j 被称为 S_i 的一个 R 派生。S_0 的所有派生的集合表示为 $D_R(S_0)$。

在这里 LTS 中的每个状态对应一种溶液,因此每个状态是由一系列描述构件状态的分子组成的。在 LTS 弧上的标号表示了让系统从一个尾结点状态到头结点状态的变换规则。

下面列出了实例 Cham 的一些反应规则,T_0 代表系统启动,T_1 代表用户发送开会请求信号,T_2 表示处理 check 信息,T_3 表示处理请求信息。

Reaction Rules

维普资讯 http://www.cqvip.com

T0: User = User1, User2

T1: User1. o(requestUR1). i(ackRU1) = o(requestUR1). i(ackRU1). User1

T2: o(check1). User1, i(check). Router, NoSent = User1. o(check1), i(check). Router, Sent

T3: o(requestUR1). i(ackRU1). User1, i(requestUR). o(requestRS). i(ackSR). o(ackRU). Router = i(requestUR). o(requestRS). i(ackSR). o(ackRU). Router, o(requestRS1). i(ackSR1). o(ackRU1). Router, i(ackRU1). User1. o(requestUR1)

经过转换机制,得到了有 500 个状态左右的 LTS, T_i 就是 LTS 弧上的标号。

2.4 FSP 和 CHAM 的比较说明

FSP 在体系结构动态性描述方面,其简洁性显而易见,用过程来表示各个构件和连接器,用 FSP 语言模拟每个过程的行为,更重要的是,这里有一个开发成熟的工具 LTSA,使每个 FSP 规格说明到 LTS 状态图间的转化过程变得更简单易行。但 FSP 不提供体系结构静态特征描述,所以不可能用它进行完整的综合的体系结构规格说明。

CHAM 语言是通过把化学反应和抽象机的概念有机地结合来描述系统状态的变化,所以它能从系统操作的动态性方面来进行描述,使人们了解系统功能和行为。但从动态性规格说明转换到 LTS 的过程要比 FSP 方法复杂,从最后得到的全局 LTS 状态图中的状态数(FSP:256 个状态;CHAM:500 多个状态)也可以看出,从 CHAM 导出的 LTS 要比由 FSP 导出的 LTS 图复杂得多。但由 CHAM 导出的 LTS 更完整,上面的信息量更大,可能更能够全面测试整个系统。

另外,与 FSP 不同,CHAM 可以用分子的代数结构模拟系统静态的结构,因此能得到一个描述软件体系结构静态和动态特征的综合框架。同时 CHAM 是一种形式化表示方法,而形式化方法是提高软件质量的重要途径,在从高层规范到最终实现的过程中,选用适当的、以形式化方法为基础的工具进行辅助设计和验证对于提高系统安全性、可靠性是非常有帮助的。

3 相关问题讨论

FSP 和 CHAM 语言都能进行体系结构的动态描述,并各有各的特点,但不一定能从它们的描述中得到一个全局的体系结构模型,原因有两个:

(1)描述各个体系结构构件的模型都比较复杂,在最后合并模型的过程中可能引起状态爆炸的问题;

(2)由于一些未知的行为或未指定的构件,体系结构模型可能是不完整的。

所以,现在有人提出把 UML(统一建模语言)应用到软件体系结构的一致性测试中,用标准的 UML 表示法来模拟软件体系结构的行为,这在两方面是有用的:

a. 推动严格的基于软件体系结构测试方法在工业生产中的应用;

b. 利用基于 UML 的丰富的测试技巧。如在文献[6]中,作者就用了 UML 的状态图来指定行为的信息,为软件体系结构建模。

参考文献:

[1] Bertolino A, Corradini F, Inverardi P, et al. Deriving Test Plans from Architectural Descriptions[A]. Jazayeri M, Wolf A. Proceedings of ICSE 2000, International Conference on Software Engineering[C]. Limerik, Ireland: [s. n.], 2002. 220 - 229.

[2] Harrold M J. Architecture - Based Regression Testing of Evolving Systems[A]. Proceedings of the International Workshop on the Role of Software Architecture in Testing and Analysis (ROSATEA)[C]. Marsala, Sicily, Italy: [s. n.], 1998. 73 - 77.

[3] Harrold M J. Testing: A Roadmap[A]. Finkelstein A. The Future of Software Engineering[C]. New York: ACM Press, 2000. 61 - 72.

[4] Tretmans J. Testing Concurrent Systems: A Formal Approach [A]. In Proceedings of the 10th International Conference on Concurrency Theory (CONCUR '99)[C]. London, UK: Springer - Verlag, 1999. 46 - 65.

[5] Compare D, Inverardi P, Wolf A L. Uncovering Architectural Mismatch in Component Behavior[R]. Technical Report CU - CS - 828 - 97, 1997.

[6] Medvidovic N, Rosenblum D S, Redmiles D F, et al. Modeling software architectures in the Unified Modeling Language[J]. ACM Transactions on Software Engineering and Methodology (TOSEM), 2002, 11(1): 13 - 15.

(上接第 99 页)

3 结束语

在低端服务器上,通过对 Oracle 数据库的初始参数配置进行调整并优化相关参数,如:块大小、多块连续读取、SGA 等。然后对数据表结构进行分区和优化索引,并优化查询 SQL 语句,使查询 Clob 字段速度得到了极为明显的提高,最终满足了用户的需要。

参考文献:

[1] 程春玲,李玲娟,周宁宁. Oracle 的大对象(LOBs)及其在企业 MIS 中的应用[J]. 微机发展,2002,12(6):28 - 29.

[2] 张燕平,王向阳,张力平. ORACLE 数据库的优化问题[J]. 微机发展,1998,8(6):46 - 47.

[3] 谈竹贤,王 毅,赵景亮. Oracle 9i PL/SQL 从入门到精通[M]. 北京:中国水利水电出版社,2002.

[4] Whalen E, Schroeter M. Oracle 性能调整和优化[M]. 高艳春,周 确,唐艳军译. 北京:人民邮电出版社,2002.

[5] Casteel J. Oracle 9i 开发指南:PL/SQL 程序设计[M]. 天宏工作室译. 北京:清华大学出版社,2004.