

一种软件体系结构级测试序列生成技术

徐士华¹, 吕莉媛², 赵 磊³

(1 黑龙江司法警官职业学院, 黑龙江 哈尔滨 150060;

2 哈尔滨师范大学 图书馆, 黑龙江 哈尔滨 150025;

3 哈尔滨师范大学 现代实验中心, 黑龙江 哈尔滨 150025)

摘 要: 软件体系结构(software architecture, SA)通过对系统构件及其交互的抽象, 提供了一个描述大型、复杂系统的高层次模型, 软件体系结构的动态描述常被用来指导分析和测试. 本文通过 CHAM(chemical abstract machine, CHAM)描述的 SA 规格说明生成 LTS, 并根据测试需求进行测试功能的选取, 提出了基于功能的最小 LTS 图(M-LTS)生成方法, 根据 McCabe 覆盖方法生成 M-LTS 图的测试路径. 最后以 B/S 结构为例, 验证了该方法在生成 SA 级的测试路径上是可行的.

关键词: 软件体系结构; 软件测试; 标号迁移系统; 测试序列

中图分类号: TP311.5

文献标识码: A

文章编号: 1000-7180(2012)01-0117-04

Technique of Testing Sequences Generation on SA Level

XU Shi-hua¹, LV Li-yuan², ZHAO Lei³

(1 The Professional Judicial Police College of Heilongjiang, Harbin 150060, China;

2 Harbin Normal University Library, Harbin 150025, China;

3 Modern experiment Center of Harbin Normal University, Harbin 150025, China)

Abstract: Software architecture provide a high level model to describe large, complex systems by abstracting system components and their interactions. Dynamic descriptions of SA can be employed in guiding test and analysis. This paper proposes an approach to generate LTS described by CHAM. According to the testing requirement, we select testing function and provide method of minimized labelled transition system (M-LTS) graph generation. Then, we generate testing paths of M-LTS graph according McCabe's covering method. Finally, Regarding Browser/Server architecture as an example, the method on generating testing path of SA is shown feasible.

Key words: software architecture; software testing; labelled transition system; testing sequences

1 引言

利用 SA 模型来指导测试是目前软件测试的一个重要研究领域. 软件体系结构描述语言 ADLs 从不同方面对软件体系结构进行规约和建模分析, 是对 SA 的形式化描述. 典型的 ADL 包括 UniCon、Rapide、Darwin、Wright、C2 SADL、Acme、xADL、XYZ/ADL 和 ABC/ADL 等. 这些 ADLs 虽然都能够对 SA 进行建模, 但在描述 SA 的动态性方面表

现不足. 为描述 SA 的动态规格说明, 文中引入 CHAM.

化学抽象机 CHAM 能有效描述 SA 的动态性, 并导出标号迁移系统 LTS, LTS 可以模拟的 SA 的动态规格说明, 对系统动态行为配置提供了一个通用的、整体的描述. LTS 的生成路径能够直接被用作测试序列, 以指导测试. 但 LTS 图包含了大量的相关信息, 生成基于 LTS 的覆盖准则和测试路径变得相当困难. 因此, 有必要对 LTS 图进行

简化,以得到相应的测试覆盖准则,从而生成测试路径。

文献[1]提出利用化学抽象机形式化描述 SA,定义了基于化学抽象机的模型和测试覆盖准则,由此导出基于结构描述的测试计划。文献[2]针对软件体系结构描述,提出了通用的体系结构测试覆盖准则。文献[3]提出了如何将体系结构级的测试转化为代码级的测试。文献[4]提出基于一组基于软件体系结构配置的测试标准及其计算规则。实现了基于软件体系结构配置的测试用例生成算法并分析了其时间复杂性。并对人机接口进行了实验。

本文提出一种基于 CHAM 的新技术:针对化学抽象机 CHAM 可以描述 SA 的动态规格说明,可以根据 CHAM 的 SA 描述来建立 LTS 模型,再根据 SA 的功能特征,对 LTS 模型最小化,得到 M-LTS。然后根据 McCabe 的基本路径测试方法^[5]生成基于 M-LTS 的测试路径,以指导测试。

2 基本概念

2.1 化学抽象机和标号迁移系统

定义 1 化学抽象机 CHAM 这一形式化系统是由 Berry 和 Boudol 共同开发出来的^[6]。CHAM 主要使用了化学隐喻,即化学抽象机由一组分子 $m_1, m_2 \dots$ 、溶液 $s_1, s_2 \dots$ 和变换规则 $T_1, T_2 \dots$ 组成。分子构成了 CHAM 的基本元素(构件);溶液是有限多个分子的集合,它解释成一个 CHAM 状态;迁移规则指出了在 CHAM 中溶液状态改变的依据。这里用 T 来标识规则集合和相应的标号集合。

CHAM 描述的 SA 规格说明包括 4 个部分:

(1) 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、过程元素、连接元素。

(2) 表示系统初始状态的溶液,用溶液 S_0 表示。

(3) 一组描述构件间交互活动以实现系统动态行为的反应规则。

(4) 一组表示系统预期终止状态的溶液集。

定义 2 一个标号迁移系统 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, $S_0 \in S$ 是初始状态, $S_F \in S$ 是终止状态的集合,而 $T = \{ \xrightarrow{l} \subseteq S \times S \mid l \in L \}$ 是标有元素 l 的迁移关系。

定义 3 路径 $p = S_0 \xrightarrow{l_1} S_1 \xrightarrow{l_2} S_2 \xrightarrow{l_3} \dots \xrightarrow{l_n} S_n$ 就是一条 LTS 路径。为了简洁, LTS 路径也可用

标号的序列表示: $p = l_1, l_2, l_3, \dots, l_n$ 。

CHAM 能够有效的描述 SA 的动态特性,并且其动态特性能够用 LTS 表示出来,利用这样一个操作上的特点,可以从一个 CHAM 描述中导出 LTS。文献[7]中给出了 CHAM 到 LTS 的详细导出方法。

2.2 标号迁移系统最小化

原则上,最后得到的 LTS 可以直接作为参考模型生成测试方案,但得到的 LTS 图包含通用的、整体的描述,图中含有大量信息,从这个整体模型中提取跟验证有关的系统行为就变得相当困难。因此,就需要对 LTS 模型根据需要进行分解,得到多个分解了的 LTS 模块,其中每个 LTS 模块表示某个功能涉及到的所有行为的交互。然后再对其进行最小化,得到最终的 M-LTS。

基于此,本文构造一个功能化的 LTS,即 FUN-LTS。首先将 LTS 的行为集合 l 根据需要进行的功能分为两个组:与实现此功能相关的交互关系 R 和实现此功能无关的交互关系 N 。因此,有 $l = R \cup N$,且 $R \cap N = \emptyset$ 。同时,将 LTS 的标号系统与上面两个组相对应的分为两个域:交互关系 R 对应的域 D ,交互关系 N 对应的域中的标号都统一用元素 τ 表示。具体表示如下:

FUN-LTS: $l \rightarrow l' : (r \in R) = d \in D, (n \in N) = \tau, l' = D \cup \tau$

生成 FUN-LTS 后,需要做以下两个工作:(1) 根据生成的 FUN-LTS,对 LTS 重新标号每一个迁移。(2) 对重新标号的 LTS 进行化简,只保留与此功能相关的行为和迁移,去除掉与此功能无关的行为和迁移,得到最小化的 M-LTS。

FC2Tool 工具^[8]提供了由 LTS 向 M-LTS 的转换,通过 FC2Tool 工具的 fc2explicit 功能可以得到最终的 M-LTS。

3 实例

3.1 浏览器/服务器(B/S)实例

B/S 具体结构为:浏览器/Web 服务器/数据库服务器。当用户需要访问数据库服务器中的数据时,只需向 Web 服务器发送数据访问请求,Web 服务器接到数据访问请求后,从数据库服务器中调取相关的数据,然后将结果通过浏览器显示给用户。图 1 给出了基于构件和连接件的静态 B/S 体系结构的 SA 描述,方框代表构件,箭头代表连接件,箭头上的标签表示数据元素的交互。

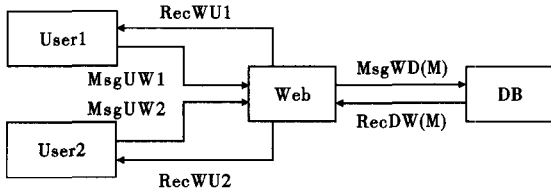


图 1 静态 B/S 的 SA 描述

以 Cham 规格说明为基础,给出了基于 B/S 结构的反应规则:

$T_0: \text{User} = \text{User1}, \text{User1}$

$T_1: \text{User1} = \text{User1}. o(\text{MsgUW1}). i(\text{RecWU1})$

$T_2: \text{User2} = \text{User2}. o(\text{MsgUW2}). i(\text{RecWU2})$

$T_3: \text{User1}. o(\text{MsgUW1}). i(\text{RecWU1}) = o(\text{MsgUW1}). i(\text{RecWU1}). \text{User1}$

$T_4: \text{User2}. o(\text{MsgUW2}). i(\text{RecWU2}) = o(\text{MsgUW2}). i(\text{RecWU2}). \text{User2}$

$T_5: o(\text{MsgUW1}). i(\text{RecWU1}). \text{User1}. i(\text{MsgUW}). o(\text{MsgWD}). i(\text{RecDW}). o(\text{RecWU}). \text{Web} = i(\text{MsgUW}). o(\text{MsgWD}). i(\text{RecDW}). o(\text{RecWU}). \text{Web}, o(\text{MsgWD1}). i(\text{RecDW1}). o(\text{RecWU1}). \text{Web}, i(\text{RecWU1}). \text{User1}. o(\text{MsgUW1})$

$T_6: o(\text{MsgUW2}). i(\text{RecWU2}). \text{User2}. i(\text{MsgUW}). o(\text{MsgWD}). i(\text{RecDW}). o(\text{RecWU}). \text{Web} = i(\text{MsgUW}). o(\text{MsgWD}). i(\text{RecDW}). o(\text{RecWU}). \text{Web}, o(\text{MsgWD2}). i(\text{RecDW2}). o(\text{RecWU2}). \text{Web}, i(\text{RecWU2}). \text{User2}. o(\text{MsgUW2})$

$T_7: o(\text{MsgWD1}). i(\text{RecDW1}). o(\text{RecWU1}). \text{Web}, i(\text{MsgWD}). o(\text{RecDW}). \text{DB} = i(\text{RecDW1}). o(\text{RecWU1}). \text{Web}, i(\text{MsgWD}). o(\text{RecDW}). \text{DB}, o(\text{RecDW1}). \text{DB}$

$T_8: o(\text{MsgWD2}). i(\text{RecDW2}). o(\text{RecWU2}). \text{Web}, i(\text{MsgWD}). o(\text{RecDW}). \text{DB} = i(\text{RecDW2}). o(\text{RecWU2}). \text{Web}, i(\text{MsgWD}). o(\text{RecDW}). \text{DB}, o(\text{RecDW2}). \text{DB}$

$T_9: o(\text{RecDW1}). \text{DB}, i(\text{RecDW1}). o(\text{RecWU1}). \text{Web} = o(\text{RecWU1}). \text{Web}$

$T_{10}: o(\text{RecDW2}). \text{DB}, i(\text{RecDW2}). o(\text{RecWU2}). \text{Web} = o(\text{RecWU2}). \text{Web}$

$T_{11}: o(\text{RecWU1}). \text{Web}, i(\text{RecWU1}). \text{User1}. o(\text{MsgUW1}) = \text{User1}. o(\text{MsgUW1}). i(\text{RecWU1})$

$T_{12}: o(\text{RecWU2}). \text{Web}, i(\text{RecWU2}). \text{User2}. o(\text{MsgUW2}) = \text{User2}. o(\text{MsgUW2}). i(\text{RecWU2})$

3.2 B/S 结构的 LTS

由于整体 B/S 的 LTS 包含了太多的状态,图 2 只给出了 B/S 结构的部分 LTS. 其中,箭头上的标号 0,1,...,12 对应反应规则标号 $T_0, T_1, \dots, T_{12}$, 圆圈表示状态,其中状态 0 表示初始状态,双箭头表示还可继续向下反应。

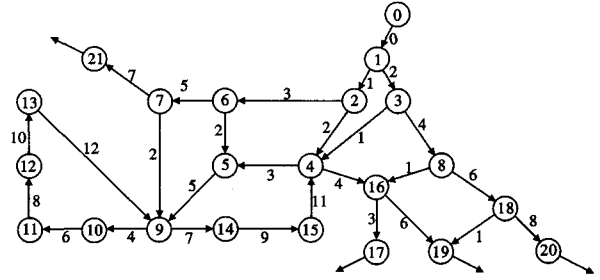


图 2 B/S 结构的部分 LTS

3.3 基于 M-LTS 的测试路径生成

B/S 结构的 LTS 包含了大量的信息,很难生成基于 LTS 的测试规则和测试路径. 因此,有必要根据测试需求选取感兴趣的动态行为,以指导测试. 实例中,假设要测试数据库服务器这一构件的功能,那么所关心的动态行为就是数据库服务器与 Web 服务器之间的交互行为 D,其他行为都可被表示为 τ ,得到功能化的 FUN-LTS. 即:

$D = \{T_7, T_8, T_9, T_{10}\}$

$\text{FUN}(T_7) = \text{MsgWD1}; \text{Msg1 From Web To DB}$

$\text{FUN}(T_8) = \text{MsgWD2}; \text{Msg2 From Web To DB}$

$\text{FUN}(T_9) = \text{RecDW1}; \text{Rec1 From DB To Web}$

$\text{FUN}(T_{10}) = \text{RecDW2}; \text{Rec2 From DB To Web}$

$\text{FUN}(T_i) = \tau$; 其他行为都用 τ 来表示。

根据 FUN-LTS,重新标号每一个迁移,并应用最小化算法,生成了最终的 M-LTS,如图 3 所示,其中状态 A 既是初始状态又是终止状态. 可以看出,生成的 M-LTS 比全局 LTS 小得多,但包含了测试需求的所有动态行为. 因此,就可以定义基于 M-LTS 的测试覆盖准则,以生成测试序列. 由于生成的 M-LTS 图是强联通图,可以使用 McCabe 路径覆盖准则选取 $N = m - n + 1$ 条独立的路径作为测试路径,其中 m 为图中弧的数目, n 为图中节点的数目, N 为能够生成所有可能路径的最小数目. McCabe 路径覆盖准则为弧和路径覆盖之间提供了一个很好的折衷方案。

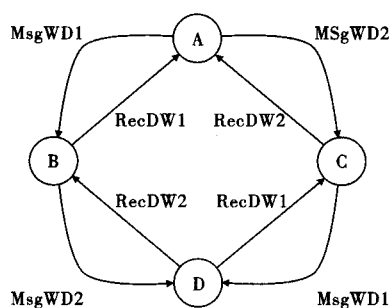


图3 DB构件的 M-LTS

根据图3,利用 McCabe 覆盖方法,得到如下的测试路径:

Path1: A $\xrightarrow{\text{MsgWD1}}$ B $\xrightarrow{\text{RecDW1}}$ A
 Path2: A $\xrightarrow{\text{MsgWD1}}$ B $\xrightarrow{\text{MsgWD2}}$ D $\xrightarrow{\text{RecDW2}}$ B
 $\xrightarrow{\text{RecDW1}}$ A
 Path3: A $\xrightarrow{\text{MsgWD1}}$ B $\xrightarrow{\text{RecDW1}}$ A $\xrightarrow{\text{MsgWD2}}$ C
 $\xrightarrow{\text{RecDW2}}$ A
 Path4: A $\xrightarrow{\text{MsgWD1}}$ B $\xrightarrow{\text{MsgWD2}}$ D $\xrightarrow{\text{RecDW1}}$ C
 $\xrightarrow{\text{RecDW2}}$ A
 Path5: A $\xrightarrow{\text{MsgWD1}}$ B $\xrightarrow{\text{MsgWD2}}$ D $\xrightarrow{\text{RecDW1}}$ C
 $\xrightarrow{\text{MsgWD1}}$ D $\xrightarrow{\text{RecDW1}}$ C $\xrightarrow{\text{RecDW2}}$ A

根据以上生成的测试路径,就可设计出基于此路径的测试数据。

4 结束语

本文提出一套基于 CHAM 的 SA 级测试路径生成方法. 为描述 SA 的动态性引入了 CHAM, 提出了基于功能的最小化的 LTS 测试路径生成方法. 首先根据测试需求, 对 LTS 功能化, 然后重新标号 LTS 的每一个迁移, 最终生成最小化的 M-LTS. 最后根据 McCabe 路径覆盖准则生成基于 M-LTS 的测试路径。

以 B/S 结构为例, 验证了该方法在生成 SA 级

的测试路径上是可行的. 此方法同时也存在应用上的困难, 例如, 在选取测试覆盖准则时, 采用 McCabe 路径覆盖准则, 很难说该方法生成的测试路径就是最优的, 同时, SA 级测试向代码级测试的转换也是下一步研究的重点。

参考文献:

- [1] Bertolino A. Deriving test plans from architectural descriptions[C]//Proc Int Conf on Software Engineering (ICSE2001), Limerick, Ireland: IEEE, 2000: 220 - 229.
- [2] Abdurazik A, Zhenyi Jin, Offutt A J, et al. Analyzing software architecture descriptions to generate system—level tests[EB/OL]. [2011-04-14]. <http://www.ics.uci.edu/~djr/rosatea/attendees.html>.
- [3] Muccini H, Bertolino A, Inverardi P. Using software architecture for code testing[EB/OL]. [2011-04-15]. <http://www.HenryMuccini.com/publications.htm>
- [4] 叶俊民, 赵恒, 黄萍, 等. 一种基于软件体系结构配置的测试用例生成算法研究[J]. 计算机科学, 2006, 33(2): 266-268.
- [5] McCabe T J. Structured testing: a testing methodology using the cyclomatic complexity metric[R]. NIST Special Publication 500-235, 1996.
- [6] Berry G, Boudol G. The chemical abstract machine [J]. Theoretical Computer Science, 1992(6): 217-248.
- [7] 赵磊, 徐士华. 基于 CHAM 的 SA 级全路径覆盖方法[J]. 微电子学与计算机, 2010; 27(4): 140-144.
- [8] Bertolino A, Inverardi P, Muccini H. Formal methods in testing software architectures[M]. SA Lectures, SFM-03; LNCS 2804, Springer, 2003: 124-149.

作者简介:

徐士华 女, (1978—), 硕士研究生. 研究方向为软件测试、软件体系结构。

吕莉媛 女, (1977—), 硕士研究生. 研究方向为软件工程。

赵磊 男, (1979—), 硕士研究生. 研究方向为构件测试。