

基于扩展 CHAM 的 SA 级路径覆盖方法

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: 化学抽象机形式化语言(CHAM)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,以此来指导分析和测试. 针对软件体系结构(SA)中构件的概念,采用 CHAM 对 SA 中的构件进行建模,并据此导出该构件的标号迁移系统(LTS),以表示该构件与其它构件之间的行为交互,并根据选取的测试覆盖准则,生成基于此构件的 LTS 中的所有测试路径. 最后以 B/S 体系结构为例,通过实验验证了该方法在生成 DB 构件的测试路径上是可行的.

关键词: 测试路径; 标号迁移系统; 测试覆盖准则; 化学抽象机

中图分类号: TP311.5

文献标识码: A

文章编号: 1000-7180(2011)09-0022-05

Method of Paths Coverage at SA Level Based on E-CHAM

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: Chemical abstract machine (CHAM) language describes system states diversification by combining chemical reaction and concept of abstract machine. For this reason, description of CHAM can be employed in guiding analysis and testing. This paper uses CHAM to model components of software architecture (SA), and then derives labelled transition system (LTS) of this component to show interactive of component's behaviors. According to the selected coverage criteria, we derive all testing paths based on the LTS of the component. Finally, regarding Browser/Server(B/S) architecture as an example, the experiments on generating testing paths of DB component is shown feasible.

Key words: testing path; LTS; testing criteria; chemical abstract machine

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. 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.

在传统测试技术的基础上,结合软件体系结构开发的特点,基于软件体系结构的形式化测试技术

的研究,取得了一定进展. 文献[1]提出利用化学抽象机来测试软件体系结构这一思想,指出从体系结构的描述中导出测试计划^[2]. 文献[3]从可测试角度出发,思考了将软件体系结构规格说明作为软件测试的基础这一问题,认为在软件体系结构层次考虑软件测试是可行的. 文献[4]引入体系结构描述语言 Wright,为描述和分析软件体系结构和结构类型提供一种有效方式^[5]. 文献[6-7]提出了基于软件体系结构的一系列测试覆盖准则,并设计了一套针对

收稿日期: 2010-12-28; 修回日期: 2011-02-03

基金项目: 教育部项目(10YJC870025)

这些覆盖准则的计算规则。

文中提出一种基于扩展 CHAM 的 SA 中的构件测试技术:首先利用化学抽象机 CHAM 对 SA 中需要测试的构件进行建模,接着利用 LTS 状态图生成算法导出该构件的 LTS 图,然后根据测试需求定义 LTS 的测试覆盖准则,最后根据覆盖准则生成该构件的测试路径,从而完成该构件在 SA 中的测试工作。

2 基本概念

化学抽象机 CHAM 是一种对动态 SA 的分析、测试非常有用的形式化描述技术。文献[7]首先利用抽象机这一形式化系统描述了软件体系结构中的各种概念,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 $S_1, S_2 \dots$ 和迁移规则 $T_1, T_2 \dots$ 组成。分子构成了 CHAM 的基本构件;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 来标识迁移规则集合和相应的标号集合。

定义 1 化学抽象机 CHAM:一个化学抽象机 CHAM 是一个 4 元组 (S, T, S_0, S_f) ,在此 S 是溶液的集合, $T \subseteq S \times S$ 是迁移规则的集合,其中迁移关系 $S_i \rightarrow S_j$ 提供了溶液变化的依据, $S_0 \in S$ 是初始溶液, $S_f \subseteq S$ 是终止溶液的集合。

通过 CHAM 可以对 SA 进行动态描述,然后导出整个系统的 LTS,以表示系统构件之间进行的信息交互,因此,生成该系统的 LTS 是决定测试能否成功的关键。

定义 2 标号迁移系统 LTS:一个标号迁移系统 LTS 是一个五元组 (S, S_0, L, S_f, T) , S 是状态的集合, L 是标号的集合, $S_0 \in S$ 是初始状态, $S_f \subseteq S$ 是终止状态的集合, $T = \{\xrightarrow{l} \subseteq S \times S \mid l \in L\}$ 是迁移关系。

通过 CHAM 描述的 SA 可以导出该系统的 LTS 图,其中,CHAM 中的每个溶液对应于 LTS 中的一个状态,CHAM 中迁移规则的集合对应于 LTS 中标号的集合,CHAM 中初始溶液和终止溶液集分别对应于 LTS 中初始状态和终止状态集。在得到构件的 LTS 后,需要定义一条 LTS 中的路径,以指导测试。

定义 3 在 LTS 中, $p = S_0 \xrightarrow{l_1} S_1 \xrightarrow{l_2} S_2 \xrightarrow{l_3} \dots \xrightarrow{l_n} S_n$ 就是一条路径,其中 S_i 表示状态, $l_i (i =$

$1 \dots n)$ 表示标号, $\xrightarrow{l_1}, \xrightarrow{l_2} \dots \xrightarrow{l_n}$ 表示标有标号 l 的迁移关系。如果 S_0 是初始状态, S_n 是终止状态,则 p 就是一条完全路径。为了简洁,路径也可用标号的序列表示: $p = l_1, l_2, l_3 \dots l_n$ 。

3 扩展的 CHAM 及相应的 LTS

通过 CHAM 可以对整个 SA 进行建模,从而导出整个 SA 的 LTS,并将其作为参考模型生成测试方案。但得到的整体 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,有必要根据测试需求,利用 CHAM 对 SA 中某个需要测试的构件进行建模,选取体系结构中与该构件相关的所有交互行为,并生成该构件的 LTS 模型,然后根据选取的测试覆盖准则生成基于该构件的 LTS 中的测试路径。

具体的实现过程包括:将 CHAM 描述的 SA 中的迁移规则 T 根据需要进行测试的构件分为两个组:与该构件相关的交互集 R 和与该构件无关的交互集 N 。因此,有 $T = R \cup N$,且 $R \cap N = \emptyset$ 。同时,定义一个函数 FUN ,它包含两个域:交互集 R 对应的域 W ,交互集 N 对应的域中的标号都统一用元素 τ 表示。具体表示如下:

$$FUN: T \rightarrow T'; (r \in R) = d \in W, \\ (n \in N) = \tau, \quad T' = W \cup \tau$$

得到 FUN 后,根据测试需求,通过重新构造溶液中的分子 M ,添加新的初始溶液 S_0 ,并将交互集 R 对应的域 W 作为新的迁移规则的集合,同时构造新的溶液集 S ,由此可以得到一个基于该测试构件的新的 CHAM,即 $(Extend\ CHAM, E-CHAM)$,然后根据 $E-CHAM$ 导出一个基于该测试构件的 LTS。

定义 4 给定一个 CHAM 描述的 SA,即 (S, T, S_0, S_f) , $E-CHAM$ 是一个四元组,即 (S', W, S'_0, S'_f) ,其中 S' 是 $E-CHAM$ 中溶液的集合, $W \subseteq T$ 是迁移规则集合, $S'_0 \in S'$ 是初始状态, $S'_f \subseteq S'$ 是终止溶液的集合。

得到 $E-CHAM$ 后,就可以根据 LTS 状态生成算法导出与 $E-CHAM$ 相对应的 LTS 中的所有溶液,具体算法如下:

输入:初始溶液 S'_0 ,迁移规则 $W_k (k=0,1 \dots)$

输出:LTS 中的溶液集

Step1: initialize SET1, initialize SET2,使之成为

空;并将 S'_0 , 存入数组 SET1 中, 数组 SET2 中存放最终的溶液集.

Step2: while(数组 SET1 不为空)

{从 SET1 中取出一个溶液 S_i , 并将 S_i 从 SET1 中删除;

For(S_i 中的每一个分子 C_i)

{If (C_i 属于自反应型分子且可以应用迁移规则 W_m)

{根据迁移规则 W_m , 对 C_i 进行重写操作, 同时生成新的溶液 S_p }

If (S_p 不包含在数组 SET2 中)

{将 S_p 存入 SET1 中}}

For(S_i 中的每一对分子 C_i 和 C_j)

{If (Match(C_i, C_j, W_k)) 的匹配计算结果中的 W_k 不为空)

{则将 C_i, C_j 根据迁移规则 W_k 进行重写操作, 生成新溶液 S_q }

If (S_q 不包含在数组 SET2 中)

{将 S_q 存入 SET1 中}}

If (SET2 中不存在溶液 S_i)

{将 S_i 存入 SET2 中}}

Step3: 算法结束.

根据以上算法, 可以得到在数组 SET2 中包含了 LTS 中的所有溶液, 最终可以根据 E-CHAM 以及 LTS 中的所有溶液得到 LTS 图. 在 LTS 中, 圆圈表示溶液状态, 弧表示引起的迁移, 弧上的标号表示引起该迁移的迁移规则.

4 基于 LTS 的测试路径生成

4.1 基于 LTS 的测试覆盖准则

得到测试构件的 LTS 后, 就要考虑在此 LTS 上应用什么覆盖准则来生成测试路径. 由于生成的 LTS 属于状态图, 图中只有迁移和溶液的状态, 且根据 LTS 的生成过程, 可以定义基于 LTS 的测试覆盖准则, 且以下覆盖准则的能力依次减弱:

①全路径覆盖: 覆盖 LTS 中所有路径至少一次.

②迁移覆盖: 覆盖 LTS 的所有迁移至少一次.

③状态覆盖: 覆盖 LTS 的所有状态至少一次.

在以上定义的基于 LTS 的测试覆盖准则中, 其中, 迁移覆盖、状态覆盖都不能做到对 LTS 进行完全测试, 生成的测试路径集都不完整, 因此, 要想对 LTS 进行完全测试, 生成 LTS 中的所有测试路径, 就需要采用基于 LTS 的全路径覆盖准则, 来生成

LTS 中的所有测试路径. 基于此, 提出了基于 LTS 的全路径生成算法的基本思想.

4.2 基于 LTS 的全路径生成算法

首先定义两个数组 PATH1 和 PATH2, 这两个数组中的每个元素用来存储每条路径, 最后得到的数组 PATH2 中包含了 LTS 中的所有路径.

输入: LTS

输出: LTS 中的所有路径集

Step1: Initialize PATH1, Initialize PATH2, Initialize S_0 , 使之为初始节点, Initialize S_t , 使之为终止节点.

Step2: 从初始节点 S_0 向下搜索, 找到所有与 S_0 直接相连的边(($S_0 \rightarrow S_i$), ($S_0 \rightarrow S_j$)...), 同时修改数组 PATH1, 将每条路径分别存入数组 PATH1 中 (即将路径($S_0 \rightarrow S_i$), ($S_0 \rightarrow S_j$)...分别存入数组中的每个元素).

Step3:

While(PATH1! = NULL)

{For (数组 PATH1 中的每条路径 C_i)

{If (路径 C_i 的最后一个节点为 S_t)

{则将 C_i 存入数组 PATH2 中, 同时将其从数组 PATH1 中删除}

Else {沿路径 C_i 中的最后一个节点 S_k 继续向下搜索, 找出与 S_k 节点直接相连的路径(($S_k \rightarrow S_p$), ($S_k \rightarrow S_q$)...), 并将其与路径 C_i 进行连接, 即将与 S_k 节点直接相连的路径连接到 C_i 的后面(($S_0 \cdots \rightarrow S_k \rightarrow S_p$)($S_0 \cdots \rightarrow S_k \rightarrow S_q$)...), 并分别存入数组 PATH1 中, 同时将 C_i 从 PATH1 中删除. }

}}

生成的数组 PATH2 中的所有路径就是 LTS 的所有测试路径. 根据 PATH2 中的测试路径, 设计测试数据的输入和预期输出.

5 实例

5.1 浏览器/服务器(B/S)实例

B/S 体系结构风格具体结构为: 浏览器/Web 服务器/数据库服务器^[8]. 图 1 给出了基于构件和连接件的静态 B/S 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通过连接件进行交互.

5.2 B/S 结构的 CHAM 描述

CHAM 描述的 B/S 结构的规格说明包括分子的语法描述、迁移规则的集合和初始溶液 S_0 , 其中分子的语法描述包括:

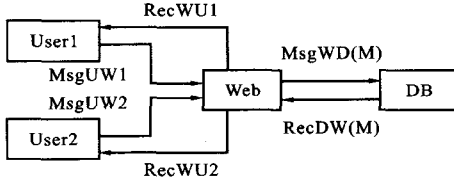


图1 静态 B/S 的 SA 描述

$$M_i = P | C | D | M. M$$

$$P_i = \text{User} | \text{Web} | \text{DB}$$

$$D_i = \text{MsgUW} | \text{MsgWD} | \text{RecDW} | \text{RecWU}$$

$$C_i = i(D) | o(D)$$

在此, M 表示 B/S 结构的构件集合, P 为处理元素的集合, 包含 User、Web 和 DB 三个构件. C 为连接元素的集合, 指出系统构件之间可能的数据通信, 包含 i 和 o 两个操作, 表示数据元素的交互端口, 且 i 为输入端口, o 为输出端口. D 为数据元素的集合, 表示在不同系统构件之间进行的信息交换, 包含 MsgUW、MsgWD、RecDW、RecWU 四个分子, 分别表示用户向 Web 服务器发送的信息, Web 服务器向 DB 服务器发送的信息, Web 服务器接收到 DB 服务器的反馈信息, 用户接收到 Web 的反馈信息. 而符号“.”表示了处理元素输入输出行为的状态.

在用 CHAM 定义系统体系结构时, 必须定义 CHAM 的初始溶液, 即明确系统的最初状态. 对于 B/S 结构, 其初始溶液 S_0 为 $\text{User}, i(\text{MsgUW}), o(\text{MsgWD}), i(\text{RecDW}), o(\text{RecWU}), \text{Web}, i(\text{MsgWD}), o(\text{RecDW}), \text{DB}$.

以 CHAM 规格说明为基础, 并根据 B/S 结构的工作原理, 给出了基于 B/S 结构的迁移规则. 由于终止溶液集 S_f 中的终止溶液不唯一, 且存在循环, 作如下规定: User1 和 User2 每次只发送一次数据请求, 当 User1 和 User2 都发送数据请求时, 则应用反应规则 T_7 和 T_8 后, 反应终止. B/S 结构的迁移规则:

$$T_0: \text{User} = o(\text{MsgUW1}), i(\text{RecWU1}), \text{User1}, o(\text{MsgUW2}), i(\text{RecWU2}), \text{User2}$$

$$T_1: o(\text{MsgUW1}), i(\text{RecWU1}), \text{User1}, i(\text{MsgUW}), o(\text{MsgWD}), i(\text{RecDW}), o(\text{RecWU}), \text{Web} = i(\text{MsgUW}), o(\text{MsgWD}), i(\text{RecDW}), o(\text{RecWU}), \text{Web}, o(\text{MsgWD1}), i(\text{RecDW1}), o(\text{RecWU1}), \text{Web}, i(\text{RecWU1}), \text{User1}, o(\text{MsgUW1})$$

$$T_2: o(\text{MsgUW2}), i(\text{RecWU2}), \text{User2}, i(\text{MsgUW}), o(\text{MsgWD}), i(\text{RecDW}), o(\text{RecWU}), \text{Web}$$

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

$$T_3: o(\text{MsgWD1}), i(\text{RecDW1}), o(\text{RecWU1}), \text{Web}, i(\text{MsgWD}), o(\text{RecDW}), \text{DB} = i(\text{RecDW1}), o(\text{RecWU1}), \text{Web}, i(\text{MsgWD}), o(\text{RecDW}), \text{DB}, o(\text{RecDW1}), \text{DB}$$

$$T_4: o(\text{MsgWD2}), i(\text{RecDW2}), o(\text{RecWU2}), \text{Web}, i(\text{MsgWD}), o(\text{RecDW}), \text{DB} = i(\text{RecDW2}), o(\text{RecWU2}), \text{Web}, i(\text{MsgWD}), o(\text{RecDW}), \text{DB}, o(\text{RecDW2}), \text{DB}$$

$$T_5: o(\text{RecDW1}), \text{DB}, i(\text{RecDW1}), o(\text{RecWU1}), \text{Web} = o(\text{RecWU1}), \text{Web}$$

$$T_6: o(\text{RecDW2}), \text{DB}, i(\text{RecDW2}), o(\text{RecWU2}), \text{Web} = o(\text{RecWU2}), \text{Web}$$

$$T_7: o(\text{RecWU1}), \text{Web}, i(\text{RecWU1}), \text{User1}, o(\text{MsgUW1}) = \text{User1}, o(\text{MsgUW1}), i(\text{RecWU1})$$

$$T_8: o(\text{RecWU2}), \text{Web}, i(\text{RecWU2}), \text{User2}, o(\text{MsgUW2}) = \text{User2}, o(\text{MsgUW2}), i(\text{RecWU2})$$

根据 SA 的 CHAM 描述, 可以导出整个 SA 的所有状态, 及所有的测试路径, 通过实验一共生成了 25 个 LTS 状态, 70 条测试路径.

5.3 DB 构件的溶液集和测试路径

根据以上的实验结果, 可以看出得到的整体 LTS 图非常庞大, 同时包含有太多的测试路径, 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. 例如, 根据测试需求, 需要对 B/S 结构中的 DB 构件进行测试, 则只需选取 SA 中与 DB 构件相关的所有交互行为, 然后根据测试覆盖准则生成该构件在 SA 中的测试路径, 从而完成对这一构件在 SA 中的测试.

在 B/S 结构的迁移规则中, 与 DB 构件相关的迁移规则集 W 包括 T_3, T_4, T_5, T_6 , 同时定义 DB 构件的初始溶液 S_0 为: $o(\text{MsgWD1}), i(\text{RecDW1}), o(\text{RecWU1}), \text{Web}, o(\text{MsgWD2}), i(\text{RecDW2}), o(\text{RecWU2}), \text{Web}, i(\text{MsgWD}), o(\text{RecDW}), \text{DB}$. 根据迁移规则集 W 及初始溶液 S_0 就可以构造 DB 构件的 E-CHAM.

根据 DB 构件的 E-CHAM, 可以导出 DB 构件的 LTS, 然后采用基于 LTS 的全路径覆盖准则及相应的全路径生成算法, 通过实验一共生成了 8 个 LTS 中的状态, 及 6 条相应的测试路径, 如图 2 所示. 从实验结果可以看出, 根据 E-CHAM 生成的

