ISSN 1000-9825, CODEN RUXUEW
Journal of Software, Vol.21, No.11, November 2010, pp.2768−2781
doi: 10.3724/SP.J.1001.2010.03780

E-mail: jos@iscas.ac.cn
http://www.jos.org.cn
Tel/Fax: +86-10-62562563

基于行为自动机的构件可替换性分析与验证*

张敬周[1,2,4+], 任洪敏[3], 宗宇伟[1], 钱乐秋[2], 朱三元[1,2]

[1](上海计算机软件技术开发中心,上海 201114)

[2](复旦大学 计算机与信息技术系,上海 200433)

[3](上海海事大学 计算机科学与技术系,上海 201306)

[4](上海软件构件化服务中心,上海 201114)

Component Substitutability Analysis and Verification Based on Behavior Automata

ZHANG Jing-Zhou[1,2,4+], REN Hong-Min[3], ZONG Yu-Wei[1], QIAN Le-Qiu[2], ZHU San-Yuan[1,2]

[1](Shanghai Development Center of Computer Software Technology, Shanghai 201114, China)

[2](Department of Computer Science and Technology, Fudan University, Shanghai 200433, China)

[3](Department of Computer Science and Technology, Shanghai Maritime University, Shanghai 201306, China)

[4](Shanghai Service Center for Component-Based Software Development, Shanghai 201114, China)

+ Corresponding author: E-mail: zjz@ssc.stn.sh.cn

Zhang JZ, Ren HM, Zong YW, Qian LQ, Zhu SY. Component substitutability analysis and verification based on behavior automata. *Journal of Software*, 2010,21(11):2768−2781. http://www.jos.org.cn/1000-9825/3780.htm

Abstract: This paper discusses component substitutability at the protocol level. Component behavior is modeled by Component behavior automaton (CBA), which is a special kind of nondeterministic finite automata (NFA). Based on CBA, a component substitutability analysis model is presented, which contains four substitutability types partitioned by two dimensions: component environment transparency and interaction similarity. This model can better ensure interaction compatibility than a traditional model based on subtype, and related verification algorithms are developed to automatically analyze component substitutability. In order to make component substitution more precise and increase component reuse, this model makes the behavior of component substituted for the actual interactive behavior that is expressed in the component environment. The reference behavior is formally defined by analyzing the actions by which the component substituted for is bound within the environment.

Key words: component-based software engineering; component substitutability; interaction compatibility; component behavior automata; software evolving

摘 要: 在交互协议层面讨论构件的可替换性,采用非确定性有限状态自动机(nondeterministic finite automata,简称 NFA)来建模构件的交互行为,在保证交互兼容性的前提下,提出了按构件环境的透明度和构件交互的变化度两维划分的可替换性模型,给出了 4 类可替换性的形式化定义及其之间的关系,并基于 NFA 理论给出了相关的验证算法.另外,该模型以构件的替换行为而不是其全部行为作为构件替换的参照,从而使替换时有更多的候选构件可供使

* Supported by the National High-Tech Research and Development Plan of China under Grant No.2006AA012191 (国家高技术研究发展计划(863)); the Shanghai Municipal Science-Technology Program of China under Grant No.07DZ22924 (上海市科技计划)

Received 2008-12-01; Revised 2009-05-21; Accepted 2009-11-04

用,提高了构件复用的几率.

关键词: 基于构件的软件工程;构件可替换性;交互兼容性;构件行为自动机;软件演化

中图法分类号: TP311 **文献标识码**: A

基于构件的软件工程(component based software engineering,简称 CBSE)通过组装可复用、可插拔的软件构件构造软件系统,已逐渐成为开发大型复杂软件系统的主流范型,是目前实现软件工程化开发、工业化生产的现实可行途径[1].

构件的插拔与替换是 CBSE 中的基本手段之一.在基于构件的设计阶段,设计的逐步求精可视为抽象度高的构件不断被更细化的构件替换的过程;在构件的获取中,构件的检索可视为检索者所需的构件由构件库中的具体构件所替换的过程;在基于构件的软件组装阶段,通过构件的替换可支持系统的灵活定制;在系统运行阶段,构件替换是支持系统升级、维护和演化的主要手段[2].基于互联网的网构软件,通过服务构件的动态替换和编排机制实现网上服务的按需计算,呈现更强的动态性和演化性[3].构件的替换已成为 CBSE 领域近年来研究的核心和热点问题之一.

构件替换性的相关研究主要基于子类型(subtyping)替换分析,其思想是使构件替换后环境不被感知[4].传统的方法主要基于类型理论和面向对象中的对象子类型分析,可分两个层面:一是型构子类型分析,如果型构 A 和 B 在输入上存在逆变(contravariance)关系,在输出上存在协变(covariance)关系,则 A 为 B 的子类型,该方法相对简单但没有考虑构件接口之间的时序关系;二是接口前置约束/后置约束的分析,如果接口 B 的前置约束蕴涵了 A 的前置约束、A 的后置约束蕴涵了 B 的后置约束,则 A 为 B 的子类型,该方法难以描述构件的动态交互行为且算法实现复杂,从而限制了其应用[5−8].

近年来,基于动态交互协议的行为子类型分析方法成为构件可替换性研究的热点[9,10],其基本原则是,如果构件 A 的交互行为包含了 B 的交互行为,则 A 为 B 的行为子类型.行为子类型替换在型构子类型分析的基础上考虑了构件动态交互的时序约束,大大提高了构件替换的有效性.其主要不足有:① 行为子类型替换并不能保障替换后系统能正常运行(见本文第 3 节的分析);② 由于替换构件的行为不能少于被替换构件的行为,致使构件替换后整个系统的交互行为只能多不能少,不能有效满足系统演化的实际情况;③ 没有考虑被替换构件的具体环境,通常情况下被替换构件只有部分功能得到实际使用,替换时只要考虑被替换构件实际参与交互的部分即可[11].还有,即使替换构件与被替换构件不满足行为子类型关系,但若能通过与环境的交互屏蔽掉替换构件与被替换构件行为上的差异,则替换照样可行.因此,行为子类型替换过于严格[2],缩小了替换时构件的可选范围,减少了构件复用的机会[12−14].

本文在相关工作的基础上,主要研究构件交互协议层面的可替换性问题.针对行为子类型替换中存在的以上不足,用有限状态自动机(本文称为构件行为自动机)建模构件的交互行为.在保持交互兼容性的前提下,从构件替换前后与环境交互的变化度、构件替换环境的透明度两个维度,提出了一个两维 4 类的构件可替换性分析模型,并给出了相应的验证算法.同时,根据构件交互的特点、构件的绑定动作及其实际环境,给出了构件的替换行为的形式化定义.以被替换构件的替换行为而不是其全部行为做为构件替换的参照,从而扩大了替换时构件的可选范围,提高了构件复用的机会.

本文第 1 节给出构件行为自动机及其交互兼容性的形式化定义及可替换性的问题描述.第 2 节给出构件的替换行为的形式化定义.第 3 节给出两维 4 类的构件替换性模型及其之间的关系.第 4 节讨论构件替换性模型的验证算法.第 5 节用一个例子来说明本替换性模型的应用.第 6 节介绍相关研究工作.最后是结论和进一步工作的展望.

1 基本概念与问题描述

本节首先给出构件动态交互协议的基本概念,用有限状态自动机来形式化描述构件动态交互的时序约束,给出了构件交互兼容性的定义,然后给出在交互协议层面构件替换应考虑的问题.

1.1 构件行为协议及其形式化表示

一个基于构件的软件系统/复合构件 M_s 由若干构件组装而成,构件 M 通过动作绑定与 M_s 中的其他构件进行交互.我们将系统 M_s 中除去构件 M 的部分称为 M 在 M_s 中的环境,记作 E_M.从复用角度看,构件 M 可以运行在多个系统(或复合构件)中,用 $E_{\Sigma M}$ 表示 M 的所有可能的运行环境(简称外部环境).

定义 1(构件的动作及其表示). 构件 M 的一个动作 act 是 M 与 $E_{\Sigma M}$ 交互的基本单元,$act \in E_{prefix} \times ENS$.其中,$E_{Prefix}$,$ENS$ 分别表示前缀的集合、动作命名空间,$E_{prefix}=\{!,?,\tau\}$.前缀!表示该动作是由 M 发出、$E_{\Sigma M}$ 接收并处理的动作,它代表了 M 向 $E_{\Sigma M}$ 请求的服务,称输出动作;前缀?表示该动作是由 $E_{\Sigma M}$ 发出、M 接收并处理的动作,代表了 M 向 $E_{\Sigma M}$ 提供的服务,称输入动作;前缀 τ 表示该动作为内部动作,是在复合构件内部处理的动作,M 与 $E_{\Sigma M}$ 进行交互的动作称为 M 的外部动作.用 A_M 表示 M 的动作集合.

可根据构件建模的抽象层次定义相应的构件动作命名空间:在概念层,动作命名空间可为领域本体中的术语概念集合.此时,构件的一个动作是领域本体中的一个术语概念,它描述了该构件对外提供或请求的一个功能(或服务);在实现层,动作命名空间可为构件的接口集合.此时,构件的一个动作是该构件的一个接口,它描述了该接口的具体型构(signature).

定义 2(构件的迹(trace)和行为协议(behavior protocol)). M 为一个构件,$E_{\Sigma M}$ 为 M 的外部环境.

(1) M 在 $E_{\Sigma M}$ 下的一次执行所处理的若干个动作的顺序排列,称为 M 的一个迹(trace).若 t 为 M 的一个迹,若 t 的任何真子串都不是 M 的迹,则称 t 为 M 的基本迹.构件的基本迹相当于构件交互中的一个原子事务处理,构件与外部环境的任何交互都必须是 1 个或多个基本迹的串接(concatenation)或重复,否则,构件运行就出现异常.

(2) M 在 $E_{\Sigma M}$ 下所有可能的执行所对应的迹的集合称为 M 的交互行为协议(简称行为).M 的基本迹的集合称为 M 的基本行为.

M 的行为表示了 M 与 $E_{\Sigma M}$ 正常交互时的动作时序约束.一个构件被设计实现后,其行为通常是不变的.

定义 3(构件行为自动机(component behavior automata,简称 CBA)). 构件 M 的行为自动机 CBA_M 是一个特殊的有限状态自动机,$CBA_M=\langle V_M,A_M,r_M,F_M,\delta_M\rangle$.其中,$V_M$ 是状态的有穷集合;A_M 是构件 M 的动作集合,$A_M = A_M^! \cup A_M^? \cup A_M^\tau$;$r_M \in V_M$,是 CBA_M 的初始状态;F_M 是 CBA_M 的终止状态集合;δ_M 是状态之间的转化关系,$\delta_M \subseteq V_M \times A_M \times V_M$.

特别要说明的是,定义 3 中的构件行为自动机的状态与构件的内部状态没有直接关系.

在不引起歧义的前提下,本文中行为自动机 CBA_M 也表示 CBA_M 所接受的行为语言.

构件 M 的基本行为对应的自动机称为 M 的基本行为自动机,用 $CBBA_M$ 表示.容易看出,CBA_M 所接受的语言是由 $CBBA_M$ 所接受的语言通过串接和 Kleene star 运算形成的闭包,记作 $CBA_M=Reg(CBBA_M)$.

在对构件行为进行描述时,采用的动作命名空间不同,构件行为的含义也不同.若动作命名空间为领域本体中的术语概念集合,则构件行为可视为对构件功能语义的形式化描述;若动作命名空间为构件的接口集合,则构件行为即为该构件的接口行为协议.

在实际中,同一个构件的行为由不同的人来描述时,得出的具体构件行为自动机在形式上可能是不一样的.但只要正确、完整地描述了该构件在设计和实现时所要求的动作时序约束,这些具体的行为自动机所接受的语言就是相同的,其自动机就是等价的.

1.2 基于构件行为自动机的交互兼容性

要替换一个基于构件的系统/复合构件 M_s 中的构件 M,首要的前提是替换后 M_s 能够正常运行.从交互的层面上来说,这要求替换构件 N 与 M 的环境 E_M 保持交互兼容性.通常情况下,构件 M 被组装到系统/复合构件 M_s 中时,只有部分动作与环境 E_M 绑定,只要交互双方在绑定动作集合上保持同步即可.

下面定义构件在绑定动作集合上的行为约束.

定义 4(行为自动机的投影). M 为构件,$CBA_M=\langle V_M,A_M,r_M,F_M,\delta_M\rangle$,$I$ 为动作集合.CBA_M 在 I 上的投影也是一个

行为自动机,记作 $CBA_M{\downarrow}I=\langle V_M,A_{M\downarrow I},r_M,F_M,\delta_{M\downarrow I}\rangle$.其中,

- $A_{M\downarrow I}=\{\varepsilon\}\cup I$;
- $\delta_{M\downarrow I}=\{\langle u,act,u'\rangle|\langle u,act,u'\rangle\in\delta_M,act\in I\}\cup\{\langle u,\varepsilon,u'\rangle|\langle u,act,u'\rangle\in\delta_M,act\notin I\}$,$\varepsilon$为空跳转动作.

下面给出两个构件交互兼容性的定义.

设 CBA_P 为构件 P 的行为自动机,动作 $act\in A_P$,状态 $u\in V_P$,如果存在一个状态 $u'\in V_P$,且$(u,act,u')\in\delta_P$,则称构件 M 在状态 u 下可激活动作 act.我们用 $A_P^!(u),A_P^?(u)$ 分别表示在状态 u 下可激活的输出动作、输入动作集合,令 $A_P(u)=A_P^!(u)\cup A_P^?(u)$.用 $\overline{act}$ 表示可与 act 绑定的动作,用 $\overline{A_P}=\{\overline{act}\mid act\in A_P\}$ 表示可与 A_P 中动作绑定的动作集合.

定义 5(状态兼容性). 设 CBA_P,CBA_Q 为两个构件行为自动机,$A_P=\overline{A_Q}$(动作完全绑定),对任意 $u\in V_P,v\in V_Q$,称状态 u 与 v 兼容,记作 $u\sim v$,如果以下条件成立:

(1) $A_P^!(u)\subseteq\overline{A_Q^?(v)}$,且 $\forall(u',v').((u,act,u')\in\delta_P,(v,\overline{act},v')\in\delta_Q,act\in A_P^!(u))\rightarrow((u'\sim v')\vee(u'\in F_P\wedge v'\in F_Q))$;

(2) $A_Q^!(v)\subseteq\overline{A_P^?(u)}$,且 $\forall(u',v').((u,\overline{act},u')\in\delta_P,(v,act,v')\in\delta_Q,act\in A_Q^!(v))\rightarrow((u'\sim v')\vee(u'\in F_P\wedge v'\in F_Q))$.

定义 6(构件交互兼容性). M,E 为构件,M 通过在动作集合 I 上的绑定与 E 进行交互,M,E 的行为自动机为 CBA_M,CBA_E.若 $CBA_M\downarrow I\sim CBA_E\downarrow\overline{I}$,则 M 与 E 在 I 上交互兼容,记作 $M\sim^I E$(或 $CBA_M\sim^I CBA_E$).

1.3 基于构件行为自动机的可替换性问题描述

在实际应用中,要替换系统/复合构件 M_s 中的构件 M 有两种目的:一是要求替换后 M_s 的功能(或行为)保持不变,通常发生在构件 M 本身有质量问题,或出于商业目的需更换系统平台、改变构件供应商等情况下;二是要改变 M_s 的功能(或行为).从交互协议的角度来看,可从 E_M 与替换构件 N 的交互是否发生变化来判断 M_s 功能(行为)改变与否.

从构件替换的具体实施角度来看也有两种情况,即替换时环境 E_M 的具体行为是否已知.由于通常情况下 M 只有部分行为被使用,在环境已知的情况下,可精确地推知 M 在 E_M 中实际发生的行为,从而在替换时有更多候选构件可供使用.本文在交互协议的层面研究构件可替换性问题:给定被替换构件 M 的行为 CBA_M、构件 N 的行为 CBA_N 以及 M 与 E_M 的动作绑定集合 I,假定被替换构件 M 与 E_M 交互兼容,在保证 N 与 E_M 交互兼容的前提下,若要保持替换前后 M_s 的行为不变(即 E_M 与 N 的交互和替换前相比保持不变),N 需满足的行为最小约束是什么.上述问题又可分为替换时 E_M 的具体行为已知和未知两种情况.

分析构件 M 和 N 是否存在替换关系可分两个层面:首先是分析两个构件的相应动作是否匹配;在此基础上,分析两个构件在行为协议层面的替换关系.由于在构件动作的匹配判定方面已有大量的研究[15−17],本文主要研究构件行为协议层面的可替换性,力图为解决上述问题提供一个统一的模型,给出各种情况下构件替换的最小条件,以便在替换时有更多的候选构件可供使用,从而扩大构件复用的机会.

为便于表述和理解,本文下面的内容均以动作名称相同来表示两个构件的动作匹配关系.例如,若构件 M 有 4 个动作{!m1,?m2,?m3,!m4},构件 N 有 3 个动作{!n1,?n2,?n3},若!m1 与!n1 匹配、?m2 与?n2 匹配、!m4 可与?n3 绑定,则将 N 的动作表示为{!m1,?m2,?m4}.在实际应用中,可先对两个(或多个)构件的动作进行匹配判定,将相匹配的动作转化为相同的动作名称.判定动作匹配的主要思路是,基于动作逆变和逆变关系分析(即输入动作存在逆变关系,输出动作存在协变关系).判定步骤可分两步:首先,基于领域本体进行概念层面的判定;其次,基于接口型构进行子类型分析判定.在此不作具体阐述.

2 构件的替换行为

通常情况下,构件 M 在被组装到系统/复合构件 M_s 中时,M 只有部分功能被 M_s 使用.从交互的角度来看,M 只有部分行为在 M_s 下发生.因此,在替换 M 时无须以 M 的所有行为为替换参照,只需考虑 M 在 M_s 下可能发生的行为即可.这样可扩大替换时可选构件的范围,增大构件复用的机会.本文将构件 M 在 M_s 下可能发生的行为

称作 M 的替换行为(简记为 $BReplaced(M,M_s)$).

$BReplaced(M,M_s)$可由 M 的行为 CBA_M、M 与环境 E_M 的绑定动作集合 I 以及 E_M 的行为 CBA_{E_M} 来确定.下面分两种情况加以讨论.

2.1 环境行为不确定时构件的替换行为

很多情况下,在替换构件 M 时,环境 E_M 的具体行为是未知的,即 E_M 对替换者来说是个黑盒.此时,可根据 M 基本迹的事务性以及与 E_M 的动作绑定集合 I 来确定 $BReplaced(M,M_s)$.下面先给出相关定义.

定义 7(构件行为自动机在绑定动作集合上的限制). M 为构件,其行为自动机、基本行为自动机分别为 $CBA_M=\langle V_M,A_M,r_M,F_M,\delta_M\rangle$,$CBBA_M$,$I\in A_M$ 为绑定动作集合.CBA_M 在 I 上的限制也是一个行为自动机,记作 $CBA_M/I=\langle V_{M/I},A_{M/I},r_M,F_M,\delta_{M/I}\rangle$.其中,

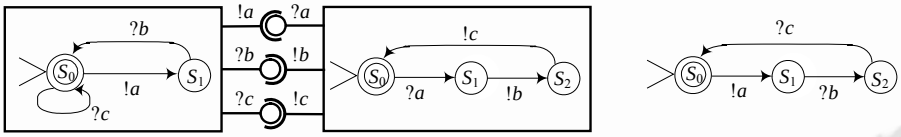
- $CBA_M/I=Reg(\{t|t\in CBBA_M,$且$\forall i\in\mathbf{N}\rightarrow t[i]\in I,\mathbf{N}$ 为自然数$\})$,其中,$t[i]$表示迹 t 中的第 i 个动作;
- $A_{M/I}\subseteq A_M$;
- $F_{M/I}\subseteq F_M$;
- $\delta_{M/I}\subseteq\delta_M$.

CBA_M/I 的含义是,由构件 M 的所有只包含 I 中动作的基本迹通过串接和 Kleene star 运算形成的闭包.在环境 E_M 具体行为不确定的情况下,由 M 基本迹的事务性(即交互必须为完整的基本迹的组合)和替换不被环境感知的原则可以看出,M 与 E_M 在 I 上绑定时,M 的替换行为 $BReplaced(M,M_s)=CBA_M/I$.根据以上定义,显然有

$$CBA_M/I\subseteq CBA_M.$$

2.2 环境行为确定时构件的替换行为

当构件 M 环境 E_M 的行为已知时(即环境为白盒),由于构件 M 与 E_M 在绑定动作集合 I 上的交互必须满足各自的行为约束,因此可根据 M 与 E_M 的交互推导出 M 在 M_s 下实际发生的行为.如图 1 所示,假定构件 M 与其环境 E_M 完全绑定,具体行为如图 1(a)、图 1(b)所示,此时 $BReplaced(M,M_s)$如图 1(c)所示,只要替换构件满足图 1(c)的要求,替换后与 E_M 的交互就不会发生变化.显然有 $BReplaced(M,M_s)\subseteq CBA_M/I$.



(a) Component M (b) Environment E_M (c) $Breplaced(M,M_s)$

(a) 构件 M (b) 环境 E_M (c) $Breplaced(M,M_s)$

Fig.1 Example for component's substitution behavior when its environment is known (binding action set is $\{!a,?b,?c\}$)

图 1 示例:环境行为确定时构件的替换行为(绑定动作集合为$\{!a,?b,?c\}$)

下面讨论 M 与 E_M 在绑定动作集合 I 上的交互的形式化推导方法.首先给出有关定义.

定义 8(行为自动机的对偶). 构件 M 的行为自动机为 $CBA_M=\langle V_M,A_M,r_M,F_M,\delta_M\rangle$,$I$ 为动作集合.CBA_M 在 I 上的对偶也是一个行为自动机,记作 $\overline{CBA_M}=\langle V_M,\overline{A_M},r_M,F_M,\overline{\delta_M}\rangle$.其中,

$$\overline{A_M}=\{\overline{act}\mid act\in A_M\},$$

$$\overline{\delta_M}=\{\langle u,\overline{act},u'\rangle\mid\langle u,act,u'\rangle\in\delta_M\}.$$

$\overline{CBA_M}$ 是与 CBA_M 同构的自动机,把 CBA_M 状态变迁中的输入/输出动作改为输出/输入动作,即得到 $\overline{CBA_M}$.

构件 M 与 E_M 通过在动作集合 I 上绑定进行交互时,M 在 I 上的动作时序约束可用 $CBA_M\downarrow I$ 表示,E_M 在 $\overline{I}$ 上的动作时序约束可用 $CBA_{E_M}\downarrow\overline{I}$ 表示.从 M 的角度来看,双方在 I 上的实际交互为 $(CBA_M\downarrow I)\cap(\overline{CBA_{E_M}}\downarrow I)$.

定义 9(构件行为自动机在特定交互协议下的限制). 构件 M,E 的行为自动机为 $CBA_M,CBA_E,\overline{A_E}\in A_M$. CBA_M 在 CBA_E 下的限制也是一个行为自动机,记作 $CBA_M//CBA_E=\langle V_{M//E},A_{M//E},r_{M//E},F_{M//E},\delta_{M//E}\rangle$,其中,

$$V_{M//E}=V_M\times V_E,$$

$$A_{M//E}=A_M,$$

$$r_{M//E}=r_M\times r_E,$$

$$F_{M//E}=F_M\times F_E,$$

$$\delta_{M//E}=\{((u,v),act,(u',v))\mid(u,act,u')\in\delta_M\wedge act\notin\overline{A_E}\wedge v\in V_E\}\cup$$
$$\{((u,v),act,(u',v'))\mid(u,act,u')\in\delta_M\wedge(v,\overline{act},v')\in\delta_E\wedge act\in\overline{A_E}\}.$$

根据定义 8 和定义 9,当 $\overline{A_E}\in A_M$ 时,显然有 $CBA_M//CBA_E\subseteq CBA_M/\overline{A_E}\subseteq CBA_M$.

根据定义 9,CBA_M 在 CBA_E 下的限制 $CBA_M//CBA_E$ 可分为两部分:CBA_M 中与动作集 $\overline{A_E}$ 无关的行为协议部分保持不变,CBA_M 中与动作集 $\overline{A_E}$ 相关的部分必须与 CBA_E 保持同步.当构件 M 的环境 E_M 行为确定时,令 $CBA_E=CBA_{E_M}\downarrow\overline{I}$,此时 $\overline{A_E}\in A_M$,M 与 M_s 的实际交互可用 $(CBA_M/I)//CBA_E$ 来形式化表示.其含义是,先去掉 M 行为协议中与绑定动作集 I 无关的部分,得到 CBA_M/I;再对 CBA_M/I 在 $CBA_{E_M}\downarrow\overline{I}$ 上进行限制操作.

根据以上定义和分析,可得出 $BReplaced(M,M_s)$ 的统一定义:

定义 10(构件的替换行为). M 为构件,E_M 为 M 在 M_s 中的环境,M 与 E_M 的绑定动作集合为 I,$BReplaced(M,M_s)$ 为 M 在 M_s 下的替换行为,则 $BReplaced(M,M_s)=(CBA_M/I)//(CBA_{E_M}\downarrow\overline{I})$.当 E_M 的具体行为未知时,

$$BReplaced(M,M_s)=CBA_M/I.$$

3 构件可替换性分析模型

本文基于构件行为自动机提出了一个两维 4 类构件替换性模型,如图 2 所示.图 2 的纵轴表示构件替换前后与环境交互的相似程度(interaction similarity),可分别对应于构件替换的两个不同目的.交互等价(interaction equivalence)表示构件 N 满足 $S_{i0}(i=0,1)$ 可替换性时,替换后 N 与环境的交互行为与替换前 M 与环境的交互等价;交互兼容(interaction compatibility)表示构件 N 满足 $S_{i1}(i=0,1)$ 可替换性时,若 M 与环境是交互兼容的,则在替换后,N 与环境的交互也是兼容的,但具体交互可能发生变化.

图 2 的横轴表示替换时构件环境的透明度(environment transparency),可对应于构件替换具体实施时的两种情况.在构件 M 的环境行为未知(environment unknown)的情况下,构件替换后与环境的绑定动作集应保持不变.若构件 N 满足 $S_{0i}(i=0,1)$ 可替换性,则在任何可与 M 正常交互的环境下,N 均可替换 M;在 M 的环境 E_M 为已知(environment known)的情况下,构件替换后构件与环境的交互保持兼容性,但绑定动作集可能发生变化.

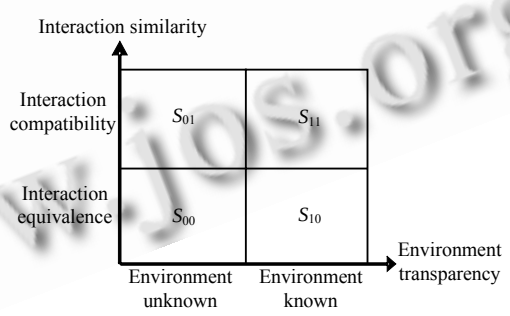


Fig.2 Component substitutability model based on behavior automata

图 2 基于行为自动机的构件替换性模型

下面讨论 4 类替换性的具体形式化定义.

3.1 环境行为未知情况下的构件可替换性

在讨论环境未知情况下的可替换性之前,先给出相关定义.

定义 11(状态相似性). 构件 P 和 Q 的行为自动机为 CBA_P,CBA_Q,对任意 $u\in V_P$,$v\in V_Q$,称状态 u 与状态 v 相似,记作 $u\mapsto v$,如果以下条件成立:

(1) $A_Q^!(v)\supseteq A_P^!(u)$,且$\forall(u',v').((u,act,u')\in\delta_P,(v,act,v')\in\delta_Q,act\in A_P^!(u))\rightarrow((u'\mapsto v')\vee(u'\in F_P\wedge v'\in F_Q))$;

(2) $A_Q^?(v)\subseteq A_P^?(u)$,且$\forall(u',v').((u,act,u')\in\delta_P,(v,act,v')\in\delta_Q,act\in A_Q^?(v))\rightarrow((u'\mapsto v')\vee(u'\in F_P\wedge v'\in F_Q))$.

若定义 11 中的条件(1) $A_P^!(u)=A_Q^!(v)$,则称状态 u 与状态 v 等价,记作 $u\rightrightarrows v$.

与定义 5 类似,定义 11 采用递归法,其含义是,状态 u 下可接受的输出动作是状态 v 下可接受的输出动作的子集,可接受的输入动作是 v 可接受的输入动作的超集,且由每个动作变迁到达的新状态也具有上述特性(即可接受的输出动作少、输入动作多)或者为各自的终止状态,则 u 与 v 是相似的.

定义 12(行为可替换性). CBA_P,CBA_Q 为两个构件行为自动机,若 $r_P\mapsto r_Q$,则 CBA_P 可替换 CBA_Q,记作 $CBA_P\mapsto CBA_Q$.若 $r_P\rightrightarrows r_Q$,则 CBA_P 可等价替换 CBA_Q,记作 $CBA_P\rightrightarrows CBA_Q$.

定义 13(环境未知情况下的构件可替换性). M,N 为构件,M 通过在接口集合 I 上的绑定与环境进行交互,M,N 的行为自动机为 CBA_M,CBA_N.

(1) 如果 $CBA_N/I\rightrightarrows CBA_M/I$,则称在 S_{00} 类型下 N 可替换 M,记作 $N\xrightarrow[I]{S_{00}}M$,简记为 $N\xrightarrow{S_{00}}M$.

(2) 如果 $CBA_N\downarrow I\mapsto CBA_M\downarrow I$,则称在 S_{01} 类型下 N 可替换 M,记作 $N\xrightarrow[I]{S_{01}}M$,简记为 $N\xrightarrow{S_{01}}M$.

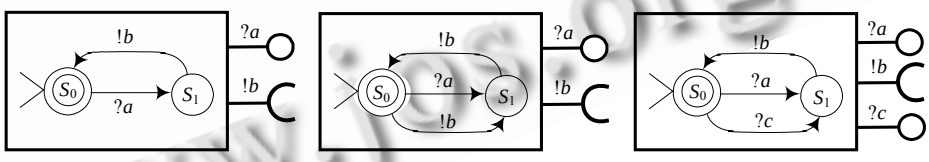
在上述定义中,S_{00} 类型替换后不仅要保持在绑定集合 I 上的交互不变,而且与 I 相关的其他动作(即由被替换构件提供的、复合构件对外的动作)的交互时序也要保持不变,因而此时 M 的替换行为

$$BReplaced(M,M_s)=CBA_M/I.$$

当 M_s 为复合构件时,满足 S_{00} 替换的构件 N 不仅可保持与 E_M 的交互不变,而且可使替换后复合构件 M_s 与其环境的交互保持不变.

在定义 13 中,如果 $I=A_M$,则满足 S_{00} 可替换性的构件 N 也满足传统意义上的行为子类型替换性要求,反之则不然.如图 3 所示,若绑定动作集合为{?a,!b},则构件 2 与构件 1、构件 3 与构件 1 均满足行为子类型替换关系,但构件 2 与构件 1 不满足 S_{00} 可替换性要求,构件 2 比构件构件 1 多出的行为容易引起替换后系统交互异常.因此,相对于行为子类型替换而言,本替换模型可更好地保障替换后系统的正常运行,从而提高了构件替换的有效性.

另外,即使两个构件不满足行为子类型关系,也可能满足替换关系.如在图 3 中,绑定动作集合为{?a,!b},则构件 3 可在 S_{01} 类型下替换构件 2,但构件 3 与构件 2 并不满足子类型替换关系.因此,本模型可更有效地指导构件的替换.



(a) Component 1 (b) Environment 2 (c) Environment 3

(a) 构件 1 (b) 环境 2 (c) 环境 3

Fig.3 Example for comparison with behavioral subtyping

图 3 与行为子类型替换的比较示例

定理 1(S_{00},S_{01} 替换的传递性). M,N,O 为构件,I 为动作集合:

(1) 若 $N\xrightarrow[I]{S_{00}}M$,$O\xrightarrow[I]{S_{00}}N$,则 $O\xrightarrow[I]{S_{00}}M$;

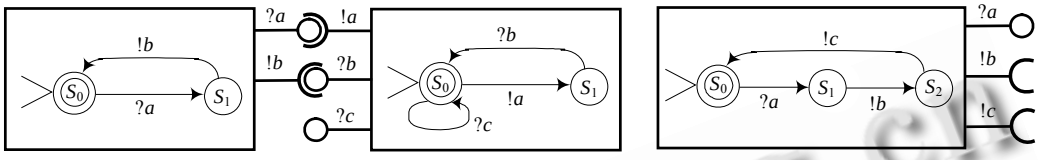
(2) 若 $N \xrightarrow[I]{S_{01}} M, O \xrightarrow[I]{S_{01}} N$, 则 $O \xrightarrow[I]{S_{01}} M$.

证明略.

3.2 环境行为已知情况下的构件可替换性

在环境 E_M 的具体行为已知的情况下,即使构件 N 与 M 不满足 S_{00} 或 S_{01} 替换性要求,但通过改变与 E_M 的绑定动作集,N 也有可替换 M 的可能.下面结合一个例子来说明绑定可变情况下的可替换性.

被替换构件 M、环境 E_M、替换构件 N 的行为如图 4 所示,M 与环境绑定动作集合 I={?a,!b}.N 若仅仅通过在 I 上与环境绑定来替换 M,替换后的系统则出现异常;N 若在集合{?a,!b,!c}上与环境绑定,则替换后的系统可正常运行,并且替换后 N 在动作集 I 上与环境的交互与替换前等价(此时,!c 绑定后成为 M_s 的内部动作).



(a) Component M (b) Environment E_M (c) Environment N

(a) 构件 M (b) 环境 E_M (c) 环境 N

Fig.4 Example for component substitutability when binding actions is changeable

图 4 绑定可变下的构件可替换性示例

定义 14(绑定可变情况下的可替换性). M,N 为构件,E_M 为 M 的环境,M 与 E_M 的绑定动作集合为 I:

(1) 称在 S_{10} 类型下 N 可替换 M,记作 $N \xrightarrow[I,E_M,I_1]{S_{10}} M$(简记为 $N \xrightarrow{S_{10}} M$),如果满足以下约束:

$$\exists I_1 \cdot (I \subseteq I_1 \subseteq A_N) \to (CBA_N \downarrow (A_N - (I_1 - I)) \xrightarrow[I]{S_{00}} BReplaced(M, M_s)) \wedge (CBA_N \sim^{I_1} CBA_{E_M}),$$

其中,$BReplaced(M, M_s) = (CBA_M / I) // CBA_{E_M}$.

(2) 称在 S_{11} 类型下 N 可替换 M,记作 $N \xrightarrow[I,E_M,I_1]{S_{11}} M$(简记为 $N \xrightarrow{S_{11}} M$),如果满足以下约束:

$$\exists I_1 \cdot (I^? \subseteq I_1^? \subseteq A_N) \to (CBA_N \sim^{I_1} CBA_{E_M}),$$

其中,$I^?$,$I_1^?$分别表示 I,I_1 中输入动作的集合.

在定义 14 的两类替换中,替换后 N 与 E_M 的绑定动作集为 I_1.

定义 14 中,S_{10} 替换类型的含义是:构件 N 在绑定动作集 I_1 上与环境 E_M 交互兼容,并且若 N 中额外的绑定动作(I_1–I)被视为内部动作屏蔽掉,则 N 涉及 I 的行为可等价替换(S_{00})M 在 M_s 下的替换行为 $BReplaced(M,M_s)$. S_{11} 类型的约束条件的含义是:N 的绑定动作集 I_1 包含了 I 中的输入动作,并且 N 可与 E_M 在 I_1 上交互兼容.在 S_{11} 类型替换中,由于替换的目的是使 M_s 的行为发生变化,因此只需考虑替换构件能够与 E_M 兼容即可.至于替换后 M_s 的具体行为,在此不再进一步讨论.

显而易见,在定义 14 的可替换性中,即使在 I=A_M 成立的情况下,替换构件和被替换构件已不再受行为子类型关系的限制.

定理 2(S_{10},S_{11} 替换的性质). M,N,O 为构件:

(1) 若 $N \xrightarrow[I,E_M,I_1]{S_{11}} M$, 且 $I^? = I_1^?$, 则 $M \xrightarrow[I_1,E_M,I]{S_{11}} N$;

(2) 若 $N \xrightarrow[I,E_M,I_1]{S_{11}} M, O \xrightarrow[I_1,E_M,I_2]{S_{11}} N$, 则 $O \xrightarrow[I,E_M,I_2]{S_{11}} M$;

(3) 若 $N \xrightarrow[I,E_M,I_1]{S_{10}} M, O \xrightarrow[I_1,E_M,I_2]{S_{10}} N$, 则 $O \xrightarrow[I,E_M,I_2]{S_{10}} M$.

证明略.

下面给出本模型中 4 类替换的一些性质.

定理 3(S_{00},S_{10} 替换下的交互等价性). M,N 为构件,E_M 为 M 的环境,M 与 E_M 的绑定动作集合为 I:

若 $N \xrightarrow[I]{S_{00},S_{10}} M$,则有 $(CBA_M \downarrow I) \cap (\overline{CBA_{E_M}} \downarrow I) = (CBA_N \downarrow I) \cap (\overline{CBA_{E_M}} \downarrow I)$.

证明略.

定理 4(4 类替换的交互兼容性). M,N 为构件,E_M 为 M 的环境,M 与 E_M 的绑定动作集合为 I, $CBA_M \sim^I CBA_{E_M}$:

(1) 若 $N \xrightarrow[I]{S_{0j}} M$ (j=0,1),则有 $CBA_N \sim^I CBA_{E_M}$;

(2) 若 $N \xrightarrow[I,E_M,I_1]{S_{1j}} M$ (j=0,1),则有 $CBA_N \sim^{I_1} CBA_{E_M}$.

证明略.

定理 4 说明,只要被替换构件与其环境是交互兼容的,则满足本替换模型中任一替换类型的构件与环境的交互仍保持兼容性.

定理 5(4 类替换之间的关系). M,N 为构件,E_M 为 M 的环境:

若 $N \xrightarrow{S_{ij}} M$ ($i,j \in \{0,1\}$)成立,则必有 $N \xrightarrow{S_{kl}} M$ ($i+j<k+l \leqslant 2, k,l \in \{0,1\}$)成立.

(证明略)

定理 5 说明,在图 2 的替换模型中,沿纵轴和横轴两个方向上,构件替换的条件逐步放宽,替换构件的可选择范围逐步扩大,即若构件 $N \xrightarrow{S_{00}} M$,则必有 $N \xrightarrow{S_{01},S_{10},S_{11}} M$.

4 构件替换性验证算法

本节讨论构件替换性验证的具体算法.即在构件替换的特定目的或环境下,验证替换构件是否满足第 3 节替换模型中的某类替换性要求.

若需通过构件替换改变系统(或复合构件)的功能,则只要验证替换构件 N 是否满足 S_{01}(对应于环境 E_M 行为未知)或 S_{11}(对应于环境 E_M 行为已知)即可.前者可通过计算 $CBA_N \downarrow I$ 和 $CBA_M \downarrow I$ 后,根据定义 13 来完成;后者可通过计算 $CBA_N \downarrow I$ 和 $CBA_{E_M} \downarrow \overline{I}$ 后,根据定义 6 来完成.行为自动机的投影操作(定义 4)的求解可在线性时间内完成,在此不再讨论.下面给出定义 12 的判定算法.

算法 1. 行为可替换性判定算法.

Input:行为自动机 P,Q.

Output:P 是否可替换 Q.

Initialization: $S1=S2=\{\langle r_P,r_Q\rangle\}$, $flag$=1; //r_P,r_Q 分别为 P,Q 的初始状态

//变量 $flag$ 为 1 表示等价替换,为 0 表示非等价的相似性替换

Repeat: **for each** $\langle u,v\rangle \in S2$, let $S2=S2-\{\langle u,v\rangle\}$, $S1=S1\cup\{\langle u,v\rangle\}$;

if Condition 1 and 2 of definition 11 are satisfied

then add all $\langle u',v'\rangle \notin S1$ to $S2$;

if $V_P^!(u) \neq V_Q^!(v)$ **then** $flag$=0;

else return; //P 不可替换 Q

Until: $S2=\varnothing$;

If $flag$=1 **return** $P \rightrightarrows Q$; //P 可等价替换 Q

Return: $P \mapsto Q$ //P 可替换 Q

算法 1 的思路是,从行为自动机 P 和 Q 初始状态对开始,按照定义 11 中的条件 1 和条件 2,采用广度优先的策略,逐步计算 P 和 Q 的可替换状态对,直至同时到达终止状态.容易看出,该算法的时间复杂度为

$$O(|V_P| \times |V_Q| \times (|\delta_P| + |\delta_Q|)),$$

其中,$|V_P|$,$|V_Q|$分别为自动机 P,Q 的状态数;$|\delta_P|$,$|\delta_Q|$为状态迁移数.

若需替换系统(或复合构件)的某个构件而保持系统功能不变,则根据定义 13,只要对行为自动机的限制操作进行求解,然后结合算法 1 即可得出 S_{00} 可替换性的验证算法;根据定义 14,只要给出行为自动机的限制操作(定义 7)、在特定交互行为上的限制(定义 9)的求解算法,再结合算法 1 及与算法 1 类似的兼容性判定算法,即可

得出 S_{10} 可替换性的验证算法.

对行为自动机 CBA_M/I 的求解比较简单,只要在 CBA_M 中裁掉那些非 I 中动作的迁移即可.行为自动机在特定交互行为上的限制(定义 9)的算法与一般自动机的交操作类似,因此,具体算法在此也不再给出.由定义可以看出,乘积操作的计算复杂度不超过 $O(|P|\times|Q|)$.在根据定义 14 进行 S_{10} 替换性验证时,可先将 $BReplaced(M,M_s)$ 中的非法状态在线形时间内滤掉,从而可大大简化 S_{10} 替换性验证的计算复杂度.

可以看出,对本文提出的 4 类可替换性的验证均可在多项式时间内完成.

另外,本文中涉及的交互兼容性及可替换性的定义与算法均基于非确定性有限状态自动机进行.因此,在描述一个构件时,其行为自动机的具体形式可以是不同的;但只要正确地描述了构件的行为协议,即可运用本文的模型和算法进行可替换性分析和验证.这样,在实际应用中,描述人员可以根据自己的思维方式给出特定构件的行为自动机,方便了对构件行为的建模,提高了本模型和算法的适用性.

5 实例分析

下面以上海构件库应用示范中的一个网上电子书籍销售构件为例来说明本替换模型的应用.如图 5(a)所示,该构件 C_1 由电子书籍管理 C_{11}(e-book management)、购物车 C_{12}(shopping cart)、用户管理 C_{13}(user management)、日志 C_{14}(logger)、通用服务 C_{15}(general services)这 5 个构件组装而成.

替换场景 1:由于性能和成本的原因,在保持 C_1 功能(行为)不变的前提下,替换购物车构件 C_{12}.C_{12} 有 8 个接口(本例中,构件的动作对应为构件接口):接口?*SCM*(提供购物车管理服务)、?*BUY*(提供购买服务和!*PAY*(请求费用支付服务)分别与构件 C_1 绑定而成为 C_1 的对外接口;接口!*GEB*(请求获取电子书籍)、!*LG*(请求日志服务)和!*SM*(请求电子邮件发送服务)分别与构件 C_{11} 的?*GEB*、C_{14} 的?*LG* 以及 C_{15} 的?*SM* 接口绑定;接口?*VIPS*(提供 VIP 购物服务)和?*VIPB*(提供对 VIP 用户的优惠服务)为闲置接口,说明 C_{12} 实现的功能或服务没有被充分利用(从而 C_1 的整体运行效率偏低,成本上升).

根据本文定义 7 可得出,在绑定集合为 I={?*SCM*,?*BUY*,!*GEB*,!*LG*,!*SM*,!*PAY*}的情况下,图 5(a)中构件 C_{12} 的替换行为 $BReplaced(C_{12},C_{12s})$与 C_2 构件的行为(图 5(b))相同,可用 C_2 等价替换 C_{12},即 $C_3 \xrightarrow[I]{S_{00}} C_{12}$,替换后对构件 C_1 的功能不产生任何影响.

由于使用了更简单的构件,可降低 C_1 的整体成本.另外,利用 C_{12} 的替换行为 $BReplaced(C_{12},C_{12s})$来选择构件,扩大了替换构件的选择范围.可以看出,$C_2$ 是 C_{12} 的父类型而非子类型,因此在传统的子类型替换方式下,C_2 是不能用来替换 C_{12} 的.

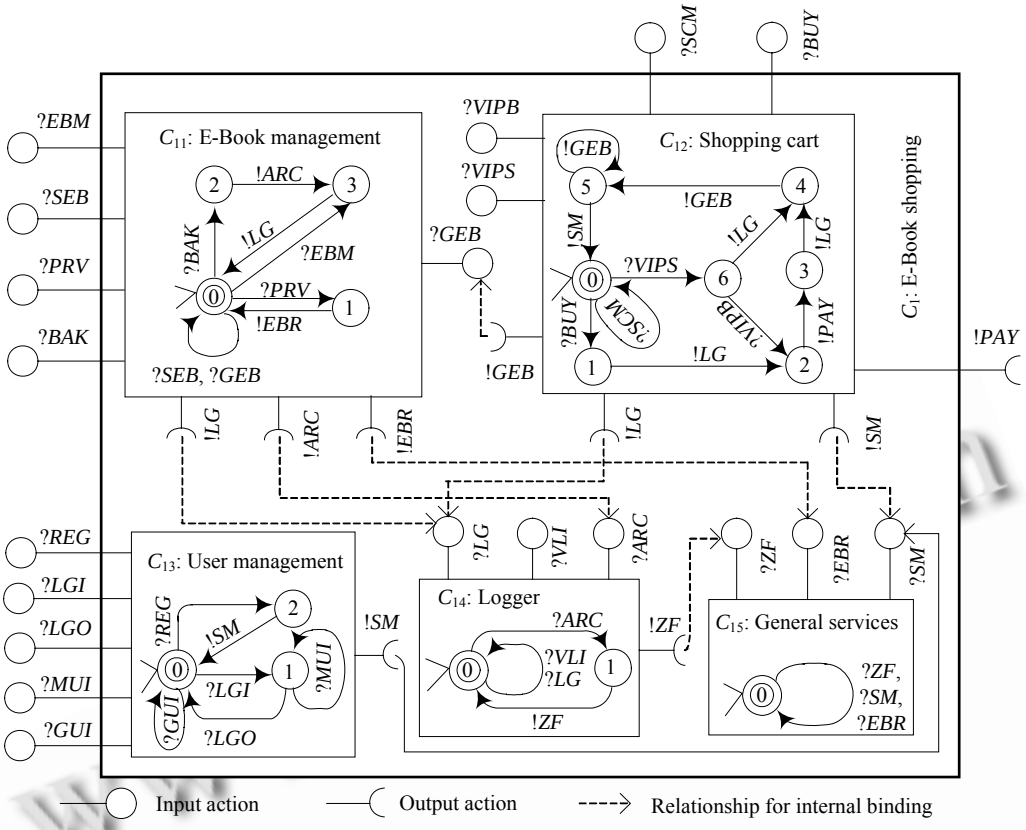
替换场景 2:在不影响 C_1 其他功能的前提下,增加新的功能或服务(以增加用户日志功能为例).在图 5(a)中,用户可注册帐户(?*REG*)、登录(?*LGI*)、退出登录(?*LGO*)、修改注册信息(?*MUI*)、获取注册信息(?*GUI*),但用户不能查看自己的访问历史记录.图 5(c)中的构件 C_3 具有查看用户的访问日志功能,但与 C_{13} 相比,C_3 多了两个服务请求动作:写日志(!*LG*)和查看日志(!*VLI*),因此在传统的构件替换方法下,C_3 是不能替换 C_{13} 的.根据本文的替换模型中的 S_{10} 替换类型,C_3 的两个服务请求动作!*LG* 和!*VLI* 正好可被 C_1 子构件 C_{14} 的提供服务所满足,也就是 C_3 多出来的两个服务请求正好可由被替换构件 C_{13} 的环境所满足.根据定义 9,C_{13} 的所有接口都参与交互(与其环境完全绑定),C_{13} 的替换行为 $BReplaced(C_{13},C_{13s})$即为 C_{13} 的行为.根据定义 14,可利用算法 1 得出

$$CBA_{C_3}\downarrow(A_{C_3}-!LG-!VLI)\xrightarrow[\{?REG,?LGI,?LGO,?MUI,?GUI,!SM\}]{S_{00}} BReplaced(C_{13},C_{13s}).$$

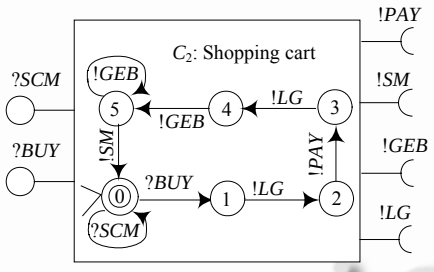
另外,替换后,C_3 的接口!*LG*,!*VLI* 分别与构件 C_{14} 接口?*LG* 和?*VLI* 绑定,C_3 与 C_{13} 的替换环境显然符合交互兼容性.因此,

$$C_3 \xrightarrow[\{?REG,?LGI,?LGO,?MUI,?GUI,!SM\},EM,\{?REG,?LGI,?LGO,?MUI,?GUI,!SM,?VAI,!LG,!VLI\}]{S_{10}} C_{13}.$$

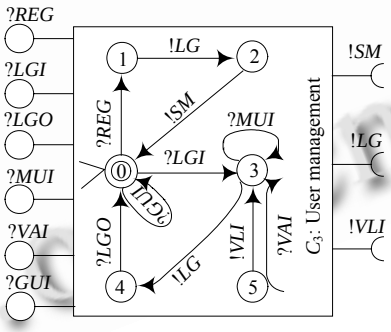
C_3 可等价替换 C_{13},替换后再将 C_3 的查看访问信息的服务(?*VAI*)绑定为 C_1 的对外服务.这样就实现了在不影响 C_1 已有服务的前提下,通过替换使 C_1 增加查看访问历史记录的功能.



(a) Online e-book shopping system
(a) 网上电子书籍购物系统



(b) Shopping cart component
(b) 购物车构件



(c) User management component
(c) 用户管理构件

EBM: E-Book management, *SEB*: Search e-book, *PRV*: Preview e-book, *GEB*: Get e-book, *EBR*: E-Book reader, *REG*: Register, *LGI*: Login, *LGO*: Logout, *MUI*: Modify UserInfo, *LG*: Logging, *GUI*: Get UserInfo, *VLI*: View LogInfo, *ARC*: Archive, *ZF*: Zip file, *SM*: Send message, *SCM*: Shopping cart management, *VIPS*: VIP services, *VIPB*: VIP bonus, *BUY*: Buy e-book, *PAY*: Pay, *VAI*: View AccessInfo.

图 5 Example for component substitution in e-book sales Web application

图 5 网上电子书籍销售构件的替换示例

6 相关研究工作

Wegner 和 Zdonik 提出了构件替换时的环境不能感知原则,强调替换构件和被替换构件具有子类型关系[4].其后的众多构件替换性研究工作皆基于该原则进行.但 Vallecillo[2]等人的研究和实践表明,基于纯粹的子类型关系决定构件的替换性通常过于严格,阻碍了软件构件的复用.Szyperski[18]指出,构件的替换既然环境不能感知,就应该考虑构件运行的环境,而不能仅仅考虑替换构件与被替换构件.

Wright[19],Darwin[20]等人运用 CSP、π演算等方法形式化地描述构件的行为交互协议,并显式地定义了系统的运行架构,但通常强调构件交互行为和系统架构的分析,缺少构件替换性的分析和验证算法.Canal 等人[14]运用π演算、Giese 等人[21]运用 Petri 网、Finkbeiner 等人[22]应用消息顺序图等描述构件的行为协议,侧重于行为协议的语法规约和兼容性的检测与分析.

Plasil 等人[12]基于 SOFA 架构定义了迹(trace)和迹的运算符,采用正则表达式描述构件的行为协议,基于子类型替换的原则,给出了构件行为的兼容性和替换性的形式化定义,可有效支持基于构件的设计求精,但未考虑构件运行环境的影响和复合构件中的构件替换问题.

Brada[11]分析了构件替换、运行环境与构件适配,把兼容性作为可替换性中的一个子问题来对待,基于 ENT 构件刻面描述元模型,从用户可理解的概念层次讨论构件的可替换性.基于子类型原则提出了严格替换性、环境替换性、完全替换性、部分替换性的概念,可有效应用于构件的分类组织与检索,没有给出在交互协议层面构件交互兼容性和可替换性验证的形式化分析和检测算法.

Beek[23],Brim 等人[24]提出用 team 自动机、交互自动机等描述构件的行为协议,注重系统的组装与交互特性,但未进行行为替换性分析.Attie 等人[25]提出了动态 I/O 自动机(dynamic I/O automata)模型描述和分析动态系统的行为,进行了构件替换性分析,但要求替换构件的迹包含于被替换构件.

De Alfaro 等人[26]提出了接口自动机(interface automata)来描述构件的行为.在此基础上提出了在构件环境行为已知的情况下“乐观型”的构件组装与兼容性验证方法,基于行为子类型原则给出了在基于构件的设计求精中保持兼容性的形式化定义和算法.其兼容性判定的原则是:两个构件组合后得到的接口自动机中若不出现非法状态,则该两个构件的组合是兼容的;若构件的环境可使组合后的接口自动机的非法状态是不可达的,则该两个构件在该环境下是兼容的.其兼容性的判定条件过于严格,限制了构件的组装.

Chaki 等人[27,28]基于 DLA(doubly labeled automaton)自动机描述构件的行为,基于子类型替换原则研究了在软件系统演化中构件的可替换性问题,给出了一个包括包含(containment)和兼容性判定两个步骤的构件可替换性分析框架,提出了基于假设-担保(assume-guarantee)形式的推理方法来适用于大型复杂系统,可支持同时替换系统中的多个构件,并开发了 COMFORT 验证工具.其不足主要在于,替换后系统的行为不可减少,且不支持在复合构件中的构件的替换性分析.

7 结束语

本文首先基于有限状态自动机理论,针对软件构件的动态交互,提出了构件行为自动机模型以描述构件的动态时序约束.在此基础上给出了构件交互兼容性的定义,讨论了被替换构件在各类情况下的替换行为,提出了按照构件替换后交互行为的相似度和环境透明度两维划分的 4 类可替换性形式化分析模型.该模型突破了传统行为子类型替换的限制,可有效保障替换后系统/复合构件的正常运行,扩大了替换时可选构件的范围.相关的验证算法具有多项式复杂度,具有实用性.

进一步的研究工作包括:对构件的动作进行语义上的描述,以支持在语义层面构件的可替换性分析;构件的迹在可并行执行情况下和具有条件判定情况下的替换性分析和验证;构件替换后系统(或复合构件)的行为推导,以更好地支持系统/或复合构件的演化分析;相关 CASE 工具的开发和实例建模与分析等.

References:

[1] Yang FQ, Mei H, Li KQ. Software reuse and software component technology. Acta Electronica Sinca, 1999,27(2):68−75 (in Chinese with English abstract).

[2] Vallecillo A, Hernández J, Troya JM. Component interoperability. Technical Report, ITI200037, Universidad de Málaga, 2000.

[3] Yang FQ. Thinking on the development of software engineering technology. Journal of Software, 2005,16(1):1−7 (in Chinese with English abstract). http://www.jos.org.cn/1000-9825/16/1.htm [doi: 10.1360/jos160001]

[4] Wegner P, Zdonik SB. Inheritance as an incremental modification mechanism or what like is and isn't like. In: Proc. of the European Conf. on Object Oriented Programming (ECOOP). Springer-Verlag, 1988. 55−77.

[5] Findler RB, Latendresse M, Felleisen M. Behavioral contracts and behavioral subtyping. SIGSOFT Software Engineering Notes, 2001,26(5):229−236. [doi: 10.1145/503271.503240]

[6] Beugnard A, Jézéquel JM, Plouzeau N, Watkins D. Making components contract aware. In: Proc. of the IEEE Software. 1999. 38−45.

[7] Kurtev S. Subtyping and inheritance in object-oriented programming. 2000. http://citeseer.ist.psu.edu/kurtev00subtyping.html

[8] Brada P, Valenta L. Practical verification of component substitutability using subtype relation. In: Proc. of the EUROMICRO Conf. on Software Engineering and Advanced Applications. IEEE, 2006. 38−45.

[9] Černá I, Vařeková P, Zimmerova B. Component substitutability via equivalencies of component-interaction automata. Electronic Notes in Theoretical Computer Science, 2007,182:39−55. [doi: 10.1016/j.entcs.2006.09.030]

[10] Hameurlain N. Flexible behavioural compatibility and substitutability for component protocols: A formal specification. In: Proc. of the 5th IEEE Int'l Conf. on Software Engineering and Formal Methods (SEFM 2007), Vol.00. 2007.

[11] Brada P. Specification-Based component substitutability and revision identification [Ph.D. Thesis]. Prague: Charles University, 2003.

[12] Plasil F, Visnovsky S. Behavior protocols for software components. IEEE Trans. on Software Engineering, 2002,28(11): 1056−1076. [doi: 10.1109/TSE.2002.1049404]

[13] Reussner RH. Enhanced component interfaces to support dynamic adaptation and extension. In: Proc. of the 34th Hawaii Int'l Conf. on System Sciences. IEEE Press, 2001. 76−86.

[14] Canal C, Fuentes L, Pimentel E, Troya JM, Vallecillo A. Extending CORBA interfaces with protocols. The Computer Journal, 2001,44(5):448−462. [doi: 10.1093/comjnl/44.5.448]

[15] Han J. A comprehensive interface definition framework for software components. In: Proc. of the 1998 Asia-Pacific Software Engineering Conf. (APSEC'98). IEEE Computer Society, 1998. 110−117.

[16] Zaremski AM, Wing JM. Specification matching of software components. ACM Trans. on Software Engineering and Methodology, 1997,6(4):333−369. [doi: 10.1145/261640.261641]

[17] Kratz BB, Reussner RH, van den Heuvel WJ. Empirical research on similarity metrics for software component interfaces. In: Proc. of the IOS 2004, Vol.8. IOS Press, 2004. 1−17.

[18] Szyperski C. Component Software: Beyond Object-Oriented Programming. Addison Wesley, ACM Press, 1998.

[19] Allen RJ. A formal approach to software architecture [Ph.D. Thesis]. Pittsburgh: Carnegie Mellon University, 1997.

[20] Magee J, Kramer J. Dynamic structure in software architectures. In: Kaise GE, ed. Proc. of the 4th Symp. on the Foundations of Software Engineering (ACM SIGSOFT'96). New York: ACM Press, 1996. 3−14.

[21] Giese H. Contract-Based component system design. In: Proc. of the 33rd Hawaii Int'l Conf. on Systems Sciences. IEEE Press, 2000. 125−134.

[22] Finkbeiner B, Krüger I. Using message sequence charts for component-based formal verification. In: Proc. of the OOPSLA Workshop on Specification and Verification of Component-Based Systems. ACM, 2001. 221−230.

[23] ter Beek MH, Ellis CA, Kleijn J, Rozenberg G. Synchronizations in team automata for groupware systems. The Journal of Collaborative Computing, 2003,12(1):21−69. [doi: 10.1023/A:1022407907596]

[24] Brim L, Černá I, Vařeková P, Zimmerova B. Component-Interaction automata as a verification-oriented component-based system specification. In: Proc. of the SAVCBS 2005. 2005. 31−38.

[25] Attie PC, Lynch NA. Dynamic input/output automata, a formal model for dynamic systems. In: Proc. of the 20th Annual ACM Symp. on Principles of Distributed Computing (PODC 2001). New York: ACM, 2001. 314−316. [doi: 10.1145/383962.384051]

[26] de Alfaro L, Henzinger TA. Interface automata. In: Proc. of the Foundations of Software Engineering. ACM, 2001. 109−120.

[27] Chaki S, Clarke E, Sharygina N, Sinha N. Dynamic component substitutability analysis. In: Proc. of the FM 2005. 2005. 512−528.

[28] Chaki S, Sharyina N, Sinha N. Verification of evolving software. In: Proc. of the 3rd Workshop on Specificaiton and Verificaiton of Component-Based Systems. ESEC/FSE, 2004.

附中文参考文献:

[1] 杨芙清,梅宏,李克勤.软件复用与软件构件技术.电子学报,1999,27(2):68−75.

[3] 杨芙清.软件工程技术发展思索.软件学报,2005,16(1):1−7. http://www.jos.org.cn/1000-9825/16/1.htm [doi: 10.1360/jos160001]

张敬周(1970－),男,山东寿光人,博士生,副研究员,主要研究领域为软件复用,基于构件的软件工程.

任洪敏(1969－),男,博士,副教授,主要研究领域为软件复用.

宗宇伟(1960－),男,教授级高工,主要研究领域为软件工程.

钱乐秋(1942－),男,教授,博士生导师,主要研究领域为软件工程, CASE 工具与环境.

朱三元(1936－),男,研究员,博士生导师,主要研究领域为软件工程.