构件组装及其形式化推导研究*

任洪敏+, 钱乐秋

(复旦大学 计算机科学与技术系,上海 200433)

Research on Component Composition and Its Formal Reasoning

REN Hong-Min+, QIAN Le-Qiu

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

+Corresponding author: Phn: 86-21-65642826, E-mail: HongminRen@163.com; HongminRen@sohu.com

http://www.fudan.edu.cn

Received 2002-07-04; Accepted 2002-12-23

Ren HM, Qian LQ. Research on component composition and its formal reasoning. *Journal of Software*, 2003,14(6):1066~1074.

http://www.jos.org.cn/1000-9825/14/1066.htm

Abstract: Component based software engineering (CBSE) is one of the most effective solutions to improve software development quality and productivity. Component composition and compositional reasoning are the core technologies and frontier research areas in CBSE. Based on the characteristics of software components and inspired by process construction methods in process algebra, in this paper, six component composition mechanisms are proposed to integrate software components simply and conveniently. And it is argued to compose interfaces at the same time of component composition, consequently to generate more powerful and more abstract interfaces to support integration of coarse-grained components and raise the abstract level of component composition. Moreover, based on the Wright's research on formal specification of software architecture, compositional reasoning algorithms about the behaviors of composite component as well as the protocols of composite interfaces are developed in this paper, which establish a foundation to analyze, validate, simulate composite systems.

Key words: component based software engineering; component composition; compositional reasoning; software architecture; process calculus

摘 要: 基于构件的软件工程(component based software engineering,简称 CBSE)能够有效地提高软件开发的质量和效率.构件组装和组装推导(compositional reasoning)是 CBSE 的关键技术.基于软件构件的特点,借鉴进程代数中进程构造的方法,提出 6 种构件组装机制,能够灵活、简便地集成软件构件,并主张在构件组装的同时进行接口组装,通过生成功能更强、抽象级别更高的复合接口,提高构件组装的抽象级别和粒度.同时,基于 Wright 的形式化规约软件体系结构的研究,给出了复合构件和复合接口的组装推导算法,为系统行为的形式化分析、验

* Supported by the National High-Tech Research and Development Plan of China under Grant No.2001AA1100241 (国家高技术研究发展计划(863))

第一作者简介: 任洪敏(1969－),男,四川阆中人,博士生,讲师,主要研究领域为软件体系结构,基于构件的软件工程,形式化方法.

证和仿真奠定了基础.

关键词: 基于构件的软件工程;构件组装;组装推导;软件体系结构;进程演算

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)逐渐成为软件开发的主流范型,是软件开发工程化的现实可行途径.构件组装(component composition)和组装推导(compositional reasoning)是CBSE 的关键技术[1].构件组装机制是运用多个构件构造软件系统的方法,而组装推导则是指根据构件的功能行为、质量属性来推断、预测软件系统的功能行为和质量属性.

本文基于软件构件的特点,借鉴进程代数中进程构造的方法,提出了 6 种构件组装机制,运用这 6 种构件组装机制,能够灵活、简便地集成软件构件.秉承和深化插头插座式体系结构(plug and socket architecture)的思想[2,3],主张构件组装的同时进行接口组装,生成功能更强和抽象级别更高的接口,从而能够提高构件组装的抽象级别和粒度.

同时,基于 Wright 运用通信顺序进程[4](communicating sequential process,简称 CSP)规约软件体系结构的研究,结合提出的构件组装机制,给出了复合构件和复合接口行为的推导算法.该推导算法奠定了系统行为分析、验证和仿真的基础,同时能够作为构件组装机制的语义,指导粘接代码(glue code)的生成.系统、科学地建模、推导和评估是成熟工程学科的标志.

本文首先介绍接口组装和行为组装推导问题,然后定义构件组装机制,并对其进行详细阐述,接着给出构件行为组装推导算法,最后是相关研究工作介绍和总结.

1 接口组装和行为组装推导

本节首先叙述接口绑定,然后提出接口组装,并从相关方面阐述行为组装推导问题.

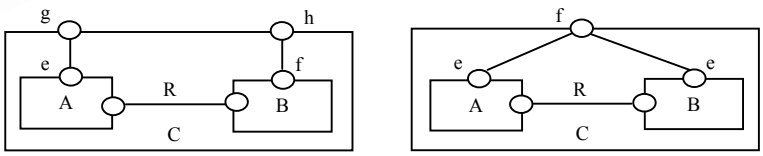
1.1 接口绑定和行为组装推导问题

构件的接口是构件与外界交互的端口,包含一组逻辑内聚的功能元素.接口的型构(signature)和交互协议(protocol)是接口规约的两个重要方面[5].接口型构描述接口元素的语法,如接口元素的名字、参数、返回值类型.接口的交互协议描述接口元素活动的时序.

接口绑定[6](interface binding)建立复合构件的外部接口和内部接口的对应关系.如图 1(a)所示,构件 A,B 通过连接件 R 连接,得到复合构件 C,C 的接口 g 绑定到 A 的接口 e,表示复合构件 C 的接口 g 就是内部构件 A 的接口 e.如果一个复合构件的外部接口只与一个内部接口进行绑定,则它的型构、交互协议与该内部接口的相同.

如果多个内部接口型构相同,并且包含的元素都是构件要求的外部功能,则它们能够绑定到一个复合构件的外部接口[7],表示它们都需要该功能服务.如图 1(b)所示,A 和 B 具有两个型构相同的请求外部功能的接口 e,它们绑定到复合构件的接口 f.f 的型构与 A.e 和 B.e 的相同,但端口 f 的交互协议却既不同于 A.e,又不同于 B.e,因为 A.e 中元素在活动的时候,可能穿插 B.e 中元素的活动,它们都反映到 f 中,反之亦然.但是,A.e 中元素的活动和 B.e 中元素的活动不是任意穿插的,它受到 A,B 连接成为一个有机整体的制约.

故组装推导机制必须能够根据绑定到同一外部接口的多个内部接口和它们的交互协议,推导该外部接口的交互协议.



(a) One-to-One interface binding (b) Many-to-One interface binding

(a) 一一对应的接口绑定 (b) 多到一的接口绑定

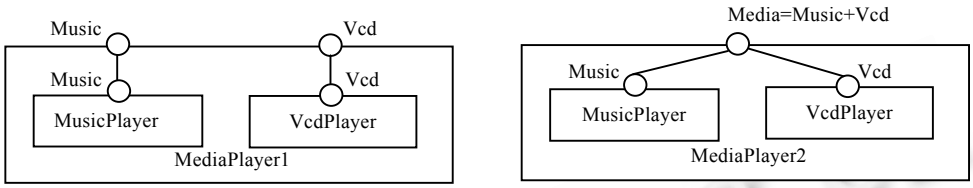
Fig.1 Interface binding

图 1 接口绑定

1.2 接口组装和行为组装推导

本节通过一个简单例子进行对比,提出并阐述接口组装.

例:有音乐播放器构件 MusicPlayer 和 VCD 播放器构件 VcdPlayer,分别通过 Music,Vcd 两个接口,输入音乐播放信息和 VCD 播放信息,提供音乐和 VCD 的播放服务.现在组装 MusicPlayer 和 VcdPlayer,生成一个媒体播放器,既能播放音乐,又能播放 VCD(如图 2 所示).



(a) Interface binding (a) 接口绑定 (b) Interface composition (b) 接口组装

Fig2. Composing media player

图 2 组装媒体播放器

如图 2(a)所示,在构件组装的时候,两个内部接口绑定到复合构件 Mediaplayer1 的两个外部接口上.运用 Mediaplayer1 构件与其他构件进行组装的时候,组装人员必须区分和掌握 Music,Vcd 两个接口,根据不同情况加以运用.

如图 2(b)所示,在构件组装的同时,进行接口组装.两个接口 Music,Vcd 合成一个媒体接口 Media,Media 接口既能接收音乐播放信息,又能接收 VCD 播放信息,相应地进行音乐或 VCD 的播放.构件组装的时候,Media 接口既能与音乐输出接口连接从而播放音乐,又能与 VCD 输出接口连接从而播放 VCD,或者与一个既能输出音乐,又能输出 VCD 的接口进行连接,根据输入的信息,决定播放音乐或 VCD.构件组装人员无须对这 3 种情况进行区分,故能方便构件组装,提高组装的抽象级别.

接口组装的思想源于插头插座式体系结构(plug and socket architecture)[2,3].插头插座式体系结构把接口中关系紧密的功能元素组织成组,并封装成为服务,使得接口中直接包含的内容减少,降低接口连接的规模.而接口组装把多个功能紧密耦合的内部接口组织为一个外部接口,封装为功能更加复杂和抽象的服务.从而减少了复合构件接口的数目,同样能够降低接口连接的规模,并且提高构件组装的抽象级别,方便大粒度构件的组装.

复合接口的元素是内部接口元素的并集,但复合接口的交互协议因为内部构件之间存在的联系而变得复杂.故组装推导机制必须能够:根据组装同一复合接口的多个内部接口及其交互协议,推导该复合接口的交互协议.

1.3 复合构件和行为组装推导

图 2 组装 MusicPlayer 和 VcdPlayer 两个构件,得到 Mediaplayer 复合构件,能够直观地得到 Mediaplayer 构件的功能,但却不能严密地得到和分析 Mediaplayer 的行为,即使已经有了 MusicPlayer 和 VcdPlayer 行为的形式规约.因为没有相应的复合构件行为组装推导机制,故组装推导机制必须能够:根据内部构件的行为规约,推导复合构件的行为规约.

本文根据构件的形式规约,结合构件组装的机制,推导复合构件行为的形式规约和复合接口交互协议的形式规约.复合接口交互协议的形式规约用于构件组装时接口匹配的检测,复合构件行为的形式规约用于系统行为的分析、验证和仿真,从而提高构件组装系统质量,帮助 CBSE 实现它的目标,即第三方独立进行可预测构件组装.另一方面,行为组装推导能够作为构件组装的语义,指导和规范构件组装时粘接代码的生成.

2 构件组装机制

通常的软件构件组装,遵循计算机硬件的组装方式,即构件通过对偶的接口进行连接,或者遵循软件体系结构的思想,通过连接件进行连接.但在如图 2 所示的媒体播放器的构造中,MusicPlayer 和 VcdPlayer 两个构件之

间既无接口连接,也无连接件连接,而是按照功能选择关系,即二者不能同时活动,两个构件组合成为一个有机整体,构成一个功能复杂的复合构件.

构件组装的本质是,在构件之间建立关联,根据这种关联,协调它们的行为,把它们组织成为一个有机的整体.构件通过接口、或者连接件进行连接,其实质是通过接口、或者连接件在它们之间建立关联,协调它们的行为.而功能选择同样是两个或多个构件行为的关联、协调的方式,即只能执行其中一个,不能同时执行.连接件建立了构件行为之间一种强的交互、耦合关系,而功能选择建立了构件行为之间一种弱的交互、耦合关系.因此,可以把功能选择看做是一种广义的连接件,作为一种构件组装的机制.

基于构件的软件开发是集成多个构件来构造软件系统.应该支持多种构件集成方式,方便构件的集成.而软件构件不同于硬件构件,是逻辑产品,易于自动生成粘接代码.如存在两个构件,给它们各自生成一个线程,让它们并行执行,从而得到一个有机整体,构成软件系统或者能够主动执行的复合构件.这同样是一种构件集成的方式,称为并行组装.

基于上述讨论,同时吸取进程代数中进程构造的思想,下面我们提出 6 种构件组装机制,并阐述它们的直观语义.

2.1 6种构件组装机制

设 P,Q 是两个构件.

定义 1(构件并行组装). 构件 P,Q 的并行组装记为 $P\|Q$,$P\|Q$ 是一个新的构件.$P\|Q$ 的复合接口的交互协议是,P 和 Q 的接口交互协议并行执行.$P\|Q$ 的功能行为是,P 和 Q 的功能行为并行执行.并行组装得到一个功能更加强大的复合构件,该复合构件同时提供这两个构件的功能.

定义 2(构件选择组装). 构件 P,Q 的选择组装记为 $P\square Q$,$P\square Q$ 是一个新的构件.$P\square Q$ 的复合接口的交互协议是,根据外部环境需求,执行 P 或 Q 的接口交互协议.$P\square Q$ 的功能行为是,根据外部环境需求,执行 P 或 Q 的功能.选择组装得到一个功能更加强大的复合构件,该复合构件能够提供这两个构件的功能,但在一定时间内,只能执行其中一个.

定义 3(构件复制组装). 构件 P 的复制组装记为 $P@n(n\in N)$,$P@n$ 是一个新的构件.$P@n$ 的复合接口的交互协议是,并行执行 n 个 P 的接口交互协议.$P@n$ 的功能行为是,并行执行 n 个 P 的功能行为.复制组装得到一个功能更加强大的复合构件,它提供同一构件类型的多个实例同时并行执行,用于提高性能或可靠性.

定义 4(构件顺序组装). 构件 P,Q 的顺序组装记为 $P;Q$,$P;Q$ 是一个新的构件.$P;Q$ 的复合接口的交互协议是,先执行 P 的接口交互协议,然后执行 Q 的接口交互协议.$P;Q$ 的功能行为是,先执行 P 的功能行为,然后执行 Q 的功能行为.

定义 5(构件中断组装). 构件 P,Q 的中断组装记为 $P\wedge Q$,$P\wedge Q$ 是一个新的构件.$P\wedge Q$ 的复合接口的交互协议是,首先执行 P 的接口交互协议,一旦 Q 的接口交互协议开始执行,则终止 P 的接口交互协议,执行 Q 的接口交互协议.$P\wedge Q$ 的功能行为是,首先执行 P 的功能行为,一旦 Q 开始活动,则终止 P 的功能行为,执行 Q 的功能行为.中断组装用于故障处理和恢复.

定义 6(构件连接组装). 构件连接组装是构件通过匹配、对偶的接口直接连接,或者通过连接件进行连接的组装机制.接口直接连接,能够生成相应的连接件,它直接把请求的服务和对应提供的服务进行连接,能够统一成为连接件连接,方便组装推导.连接组装的复合构件的功能行为是各个构件功能行为的并行执行,但同时受到连接件的协调和制约.

本文借鉴 CSP 的进程构造方法作为构件组装方法,6 种构件组装机制组成一个构件集成体系.CSP 的灵活和强大建模能力,表明了该体系相应的构件组装建模能力.将 CSP 的运算符号作为构件组装机制的符号表示,是因为它们在语义方面存在相通的地方,并且方便行为组装推导.

2.2 构件组装机制和程序控制结构

系统开发时,通常运用程序语言或脚本(script)语言定义一段“主程序”,按照顺序、选择、循环这 3 种方式调用并执行构件的功能.它们与构件的顺序组装、复制组装、选择组装具有相似的控制思想,但它们并不一样,其

本质差异在于,程序控制、结构控制的对象是程序语句,而顺序组装、选择组装等组装机制控制的对象是构件和构件接口.

首先,它们存在 3 方面的差异:

(1) “主程序”按照顺序、选择、循环这 3 种方式调用构件功能,主要目的是按照相应控制方式驱动、执行构件的功能.而顺序组装、选择组装、复制组装是指按照相应控制、协调方式把多个构件组织、包装成为一个整体,其主要目的是封装,形成复合构件,便于构件的组装和应用.

(2) “主程序”按照顺序、选择、循环这 3 种方式进行构件调用,仅指按照相应控制方式执行构件的功能.而在进行顺序组装、复制组装、循环组装的时候,不仅按照相应方式控制构件功能的执行,通常还要进行接口组装,从而形成功能更强、抽象级别更高的接口,提高构件组装的抽象级别.

(3) 构件不全是函数,通常具有复杂的内部结构、单独的执行进程和多个接口,每个接口具有多个功能元素.在“主程序”中,程序语言级别的顺序、选择、循环控制结构,控制程序语句的执行,抽象级别低,难于表达构件之间以及它们的多个接口之间总体的控制结构.而顺序组装、选择组装能够简单地表达这种关系.

其次,顺序、选择、复制等组装机制与程序控制结构具有各自的作用:

程序控制结构,控制的对象是程序语句,控制粒度小,因此较为灵活,并且能够运用程序语言提供的功能,在各个语句之间建立关系,表达复杂的控制关系.而构件顺序、选择、复制组装等组装机制,它们控制的对象是构件和接口,抽象程序高,表达构件之间的宏观控制关系,便于系统理解和组装.

最后,显式地支持构件顺序、选择、复制、并行、中断组装机制,能够带来两方面的好处:

(1) 实现上,支持和方便代码自动生成.构件可能具有不同的语言,支持不同的平台,分布在不同的网络站点.它们不能简单地按照顺序、选择或循环方式直接调用,必须考虑它们的异构特性.提供顺序、选择、并行等组装机制,便于在构件组装的 CASE 环境中自动生成相关封装代码和粘接代码,把异构和分布的构件组织成为一个逻辑上的本地构件,方便构件的组装和调用.

(2) 设计上,便于形式化分析和仿真.把构件之间的这种控制关系显式地、抽象地表达出来,而不是分散于程序语句中,便于进行组装推导分析.运用相关推导算法,得出复合构件功能行为和复合接口的交互协议,用于系统行为分析和接口兼容性的检测,从而提高组装软件系统的质量.

3 构件组装的行为推导

本文基于 Wright 规约和 CSP,结合第 2 节提出的 6 种构件组装机制进行构件行为组装推导,并分 3 个步骤对其进行阐述.首先简介 Wright 形式规约,然后定义相关概念和函数,最后给出构件行为组装推导算法.

3.1 Wright软件体系结构描述语言

Wright 定义和描述方法参见文献[8,9].在 Wright 中,构件定义由 3 部分组成:构件标识符、端口集合、表示构件计算行为的 CSP 进程(Computation 进程).在下面的叙述中,符合 Wright 的习惯,称接口为端口(port).

例:一个简单构件,它有两个端口,一个端口反复读入数据,直到遇到结束信号,另一个端口反复输出数据,或通知结束.其计算功能是读入的数据乘 2.

Component Double //构件类型标识符定义

Port Out= $\overline{\text{Write!}x}$ →Out∏ $\overline{\text{Close}}$ → §

//构件计算进程定义

Computation=(In.read?x→ $\overline{\text{Out.write!}(2*x)}$ →Computation)□(In.close→ $\overline{\text{Out.close}}$ → §)

Adouble:Double; //Double 构件的实例定义

Adouble 实例的语义是:运用符号 Adouble 标记构件类型 Double 的计算进程,即:

Adouble:Computation

具有上划线的事件表示由该进程控制启动的事件(initiated event),不带上划线的事件表示由其他进程控制启动并由该进程观察到的事件(observed event).

从该例中能够得到:端口进程中的事件具有一级结构,即只有事件名称.计算进程中的事件具有二级结构:端口名和事件名,如 In.read?*x*.而构件实例进程的事件具有三级结构:实例名.端口名.事件名,如 Adouble.In.close,表示:Adouble 实例的 In 端口观察到事件 close.

在 Wright 中,连接件由 3 部分组成:连接件标识符、角色集合、表示交互连接协议的 CSP 进程(Glue 进程).Wright 对连接件的描述和实例处理方法与构件相同.

3.2 CSP进程变换

CSP 提供简单但强大的并行合成机制,方便构件组装推导.两个 CSP 进程并行合成,它们字母表中共同的事件需要同步执行,其他事件各自独立执行[4].因此,在行为组装推导的时候,要对构件和连接件进程的相关事件进行变换、重新命名,从而调整、安排各个进程之间的同步,使整个系统按照其语义进行执行.下面,我们来提供符号变换和 3 个符号变换函数的定义.

定义 7(字母表). CSP 运用进程描述客体的行为模型.事件是客体行为的基本元素.进程的字母表是进程事件的集合,进程 P 的字母表记为 αP.

定义 8(符号变换). 设 f 是一个单射函数,将进程 P 的字母表映射到符号集合 A 上,即 f: $\alpha P \rightarrow A$,则 $f(P)$表示如下进程:当 P 执行事件 c 时,$f(P)$执行事件 $f(c)$.$\alpha f(P)=f(\alpha P)$,f 称为符号变换函数.

定义 9(进程标记). 进程 P 用标记符号 m 进行标记,得到一个新的进程 $m:P$.定义函数 $f_m(x)=m.x, x\in\alpha P$,则 $m:P=f_m(P)$.

定义 10(进程限定). 给定进程 P 和事件集合 E,$P\uparrow E$ 表示进程 P 中除 E 之外的事件都被隐藏、屏蔽而得到的进程.

下面定义 3 个用于行为组装推导的符号变换函数.

定义 11(端口角色变换函数). 构件实例 N 的端口 M 和连接件实例 I 的角色 J 连接,则有一个相应的端口角色变换函数 $R_{(I,J)}^{(N,M)}(e)$,把 I.Glue 进程中的事件 $I.J.o$ 变换为 $N.M.o$,表示因为端口角色的连接,角色 $I.J$ 中事件 o 实为端口 $N.M$ 中的事件 o. $R_{(I,J)}^{(N,M)}(e)$ 是一个符号变换全函数,定义如下:

$$R_{(I,J)}^{(N,M)}(e)=\begin{cases} N.M.o, & \text{if } e=I.J.o \\ e, & \text{otherwise} \end{cases}.$$

设一个复合构件有 V 个端口和角色的连接,则相应 V 个端口角色变换函数的合成记为 $\overline{R}$,即 $\overline{R}=R_1\circ\cdots\circ R_v$.

定义 12(端口绑定、组装变换函数).内部构件实例 N 的端口 M 绑定到复合构件的端口 K,或者参与组装复合端口 K,则有一个相应的端口绑定、组装变换函数 $S_{N,M}^{K}(e)$,把 N 的 Computation 进程中三级结构的事件 $N.M.o$ 变换为 $K.o$,表示因为端口绑定或组装,端口 $N.M$ 中的事件 o 实为外部端口 K 中的事件 o. $S_{N,M}^{K}(e)$ 是一个符号变换全函数,定义如下:

$$S_{N,M}^{K}(e)=\begin{cases} K.o, & \text{if } e=N.M.o \\ e, & \text{otherwise} \end{cases}.$$

设一复合构件共有 V 个端口绑定、组装变换函数,则它们的合成记为 $\overline{S}$,即 $\overline{S}=S_1\circ\ldots\circ S_v$.

定义 13(端口实例变换函数). 内部构件实例 N 的端口 M 参与组装一个外部端口 K,则有一个相应的端口实例变换函数 $T_{N,M}(e)$,把三级结构的事件 $N.M.e$ 变换为一级结构的事件 e.因为在推导过程中,复合接口交互协议进程的事件具有三级结构,需要进行符号变换,具备一级结构,从而与原子构件端口 CSP 进程中的事件一样,实现透明组装,即从外部不能判定构件是一个复合构件或原子构件. $T_{N,M}(e)$ 是一个符号变换全函数,定义如下:

$$T_{N,M}(e)=\begin{cases} o, & \text{if } e=N.M.o \\ e, & \text{otherwise} \end{cases}.$$

设共有 V 个端口绑定到端口 K 或者组装成端口 K,则相应 V 个端口实例变换函数的合成记为 $\overline{T}$,即 $\overline{T}=T_1\circ\ldots\circ T_v$.

3.3 构件行为组装推导算法

定义下列符号,方便构件行为组装推导,使表达简洁、灵活.

定义 14(进程构造简记). 设 $OP\in\{\Box,;,\wedge,\|\}$,则$(OP\ i{:}1..n.P_i)=P_1\ OP\ P_2\ldots OP\ P_n$.

定义 15(构件元素简记). 设 Q 为构件类型 COM 的实例,即有 Q:COM 实例定义,则 Q^{csp},COM^{csp} 表示 COM 构件的计算进程,$Q.P$ 表示构件 Q 的端口 P,$Q.P^{csp}$ 表示 COM 的端口 P 的 CSP 进程.连接件遵循同样的简记方法.

软件系统是复合构件的特例,接口绑定是接口组装的特例,故只需考虑复合构件和复合接口的组装推导.下面分几种情况对其进行阐述.

(1) 选择、顺序、中断组装行为推导

设组装机制符号为 OP,$OP\in\{\Box,;,\wedge\}$,$Q_1,Q_2,\ldots,Q_n,n$ 个构件实例参与组装复合构件 COM,则 COM 的计算进程是 $\overline{S}(OP\ i{:}1\cdots n\cdot Q_i:Q_i^{csp})$.

设 $Q_1.P_1,Q_2.P_2,\ldots,Q_m.P_m,m$ 个内部构件端口绑定到复合构件的端口 K,或者组装成端口 K,该端口的行为交互进程是 $(OP\ i{:}1\cdots m\cdot Q_i.P_i^{csp})$.

即组装机制是选择、顺序、中断,则复合构件的计算进程是:各个内部构件实例的进程按照选择、顺序或者中断关系执行,并考虑端口绑定、组装,用 $\overline{S}$ 函数进行符号变换而得到.其复合接口的交互协议进程是:相关内部端口进程运用选择、顺序或中断运算符连接构造而成.

(2) 复制组装、并行组装行为推导

复制组装是并行组装的特例,故与并行组装具有统一的行为组装推导机制.设 $Q_1,Q_2,\ldots,Q_n,n$ 个构件实例按照并行方式,组装得到复合构件 COM,则 COM 的计算进程是 $\overline{S}(\|\ i{:}1\cdots n\cdot Q_i:Q_i^{csp})$.

设 $Q_1.P_1,Q_2.P_2,\ldots,Q_m.P_m,m$ 个端口绑定到端口 K,或者组装成端口 K,则端口 K 的进程是 $\overline{T}(\|\ i{:}1\cdots m\cdot Q_i\cdot P_i:Q_i.P_i^{csp})$.

即组装机制是复制、并行,则复合构件的计算进程是:各个内部构件实例的进程并行执行,并考虑端口绑定、组装,用 $\overline{S}$ 函数进行符号变换而得到.

其复合接口的交互协议进程同样应是相关内部接口交互协议的并行合成.但 CSP 并行合成机制规定:两个进程字母表中相同事件同步执行.而 $Q_i.P_i^{csp}\ (1\le i\le m)$的字母表是其接口功能元素的名字,因此,根据接口组装规则,$Q_u.P_u^{csp}$,$Q_v.P_v^{csp}\ (u\ne v\wedge 1\le u\le m\wedge 1\le v\le m)$中共同的事件是 $Q_u.P_u^{csp}$,$Q_v.P_v^{csp}$ 两个接口共同的功能元素,表示 $Q_u.P_u^{csp}$,$Q_v.P_v^{csp}$ 能够并行地执行该功能元素,如果直接合成,则成为二者同步执行,违反接口组装的语义.故对接口进程 $Q_i.P_i^{csp}\ (1\le i\le m)$运用符号 $Q_i.P_i$ 进行标记,即得到 $Q_i.P_i:Q_i.P_i^{csp}$,然后进行并行合成.为实现组装透明,运用函数 $\overline{T}$ 进行符号变换,把 $(\|\ i{:}1\cdots m\cdot Q_i.P_i:Q_i.P_i^{csp})$ 的三级结构变换成为一级结构,从而得到该复合接口的交互协议进程.

(3) 连接组装的行为推导

设 $Q_1,Q_2,\ldots,Q_n,n$ 个构件实例,$C_1,C_2,\ldots,C_m,m$ 个连接件实例按照连接组装机制,组装成为复合构件 COM,$P=(\|\ i{:}1\cdots n\cdot Q_i:Q_i^{csp})\|(\|\ j{:}1\cdots m\cdot\overline{R}(C_m:C_m^{csp}))$,则 COM 计算进程即 $COM^{csp}=\overline{S}(P)$.

设 $Q_1.P_1,Q_2.P_2,\ldots,Q_u.P_u,u$ 个端口绑定到接口 K,或者组装成为接口 K,则接口 K 的进程是 $\overline{T}((\|\ i{:}1\cdots u\cdot Q_i.P_i:Q_i.P_i^{csp})\|(P\uparrow(\bigcup_{i=1}^{u}\alpha Q_i.P_i:Q_i.P_i^{csp})))$.

运用连接组装机制构造复合构件,复合构件功能行为的主体由各个内部构件的功能行为组成,故第 1 步得到 $(\|\ i{:}1\cdots n\cdot Q_i:Q_i^{csp})$,但各个内部构件不能完全独立并行执行,它们的执行受到连接件的制约和协调,即它们的执行必须遵循连接件 Glue 进程执行的时序,同时,连接件自身 Glue 进程的功能行为是复合构件整体功能行为的一个组成部分,根据 CSP 进程并行合成的语义,并行合成的进程满足其各个部分的规约,故第 2 步得到 $(\|\ i{:}1\cdots n\cdot Q_i:Q_i^{csp})\|(\|\ j{:}1\cdots m\cdot\overline{R}(C_m:C_m^{csp}))$,该进程满足连接件的制约和限制.其中,运用 $\overline{R}$ 进行符号变换,它是根据接口、角色的连接关系,修改连接件实例中事件为构件实例进程中的事件,从而建立它们之间的同步、协调关系.对上述得到的进程,运用 $\overline{S}$ 进行符号变换,即把相关事件转化为复合构件接口的事件,使其具备二级结构,实现组装透明,从而得到复合构件的计算进程 $COM^{csp}=\overline{S}(P)$.

$Q_i.P_i\ (1\le i\le u)$组装生成端口 K,故其行为主体应是 $Q_i.P_i^{csp}\ (1\le i\le u)$的并行合成.与并行组装时接口交互协议进程的推导相同,运用 $Q_i.P_i$ 进行进程限定,第 1 步得到 $(\|\ i{:}1\cdots u\cdot Q_i.P_i:Q_i.P_i^{csp})$.但各个接口 $Q_i.P_i\ (1\le i\le u)$不能

完全独立并行执行,它们的事件不能任意穿插执行,$Q_i.P_i$ 的并行执行受到系统整体的制约,该制约可用进程限定得到:$P\uparrow(\bigcup_{i=1}^{u}\alpha Q_i.P_i : Q_i.P_i^{csp})$),它与第 1 步得到的进程并行合成,施加它的限制,并运用 $\overline{T}$ 进行符号变换,事件由三级结构调整变为一级结构,从而得到该复合接口的交互协议进程.

4 相关研究工作

4.1 软件体系结构领域的相关研究工作

由北京大学主持、研制的集成环境"青鸟Ⅲ系统"[10]对构件获取、构件制作、构件组装进行了系统而深入的研究,提出了基于体系结构的构件组装技术和相应的体系结构描述语言 ABC (architecture based composition) /ADL.分布式系统开发环境 Darwin[7],通过服务提供接口和服务请求接口的直接连接、内部构件接口到复合构件接口的绑定,层次化地构造软件系统.但 Darwin 并未对构件的计算行为进行建模,从而不能支持复合构件行为的推导.Wright[8,9]是体系结构的形式化描述语言,通过连接件进行构件组装.Wright 能够层次化地构造构件和连接件,但复合构件接口和内部接口一一对应绑定.Unicon[11],ACME[6]及其他体系结构描述语言都能够提供构件连接组装机制,但不支持多种构件组装机制,并且通常缺乏形式语义支持和组装推导机制[12].

4.2 CBSE领域的相关研究工作

CBSE 领域对构件组装推导进行了大量的研究工作.如文献[13]运用 Petri 网对实时系统建模,强调时间规约,对系统的行为和死锁、活性(liveness)等非功能属性进行推导.文献[14]对组装性能推导(compositional performance reasoning)及其相关问题进行了系统的探索.文献[15]采用假定-承诺范型(assumption commitment paradigm),扩充时段演算(duration calculus),对系统行为进行建模和推导.但 CBSE 领域的组装推导研究侧重于形式化建模和相关推导理论的研究,不关注软件体系结构和组装机制对推导算法的简化作用.

5 结 论

构件组装和组装推导是 CBSE 的核心技术和前沿研究领域.本文基于软件构件的特点,借鉴进程代数中进程构造的思想,提出了 6 种构件组装机制,灵活运用 6 种组装机制,能构造复杂的软件系统.深化插头插座式体系结构的思想,主张构件组装的同时进行接口组装,把紧密相关的内部接口组织成为一个外部接口,封装成为功能更加复杂和抽象的服务,从而更好地支持大粒度构件组装,并提高构件组装的抽象级别.

同时,基于 Wright 运用 CSP 形式化规约软件体系结构的研究,本文给出了复合构件的功能行为和复合接口的交互协议的推导算法,从而奠定了软件系统行为分析、仿真、验证的基础,提升复合软件系统的质量,向 CBSE 进行可预测构件组装的目标迈进了一步.

进一步的研究工作包括在行为组装推导的基础上进行非功能属性的组装推导和相关 CASE 工具的开发,并进行更多的实例建模和分析.

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] Luckham D,Vera J, Meldal S. Three concepts of system architecture. Technical Report, CSL-TR-95-674, Stanford University, 1995.

[3] Zhang SK, Zhang WJ, Chang X, Wang LF, Yang FQ. Building and assembling reusable components based on software architecture. Journal of Software, 2001,12(9):1351~1359 (in Chinese with English abstract).

[4] Hoare CAR. Communicating Sequential Processes. Prentice Hall, 1985.

[5] Canal C, Fuentes L, Pimentel E, Troya JM, Vallecillo A. Extending CORBA interfaces with protocols. The Computer Journal, 2001,44(5):448~462.

[6] Garlan D, Monroe R, Wile D. ACME: An architectural interconnection language. Technical Report, CMU-CS-95-219, Carnegie Mellon University, 1995.

[7] Magee J, Kramer J. Dynamic structure in software architectures. In: Kaise GE, ed. Proceedings of the ACM SIGSOFT'96: the 4th Symposium on the Foundations of Software Engineering. New York: ACM Press, 1996. 3~14.

[8] Robert J, Allen R. A formal approach to software architecture [Ph.D. Thesis]. Pittsburgh: Carnegie Mellon University, 1997.

[9] Allen R, Garlan D. A formal basis for architectural connectors. ACM TOSEM, 1997,6(3):213~249.

[10] Yang FQ, Mei H, Li KQ, Yuan WH, Wu Q. An introduction to JB3 system supporting component reuse. Computer Science, 1999,26(5):50~55 (in Chinese with English abstract).

[11] Shaw M, DeLine R, Klen DV. Abstractions for software architecture and tools to support them. IEEE Transactions on Software Engineering, 1995,21(4):314~355.

[12] Nenad M, Richard NT. A classification and comparison framework for software architecture description languages. IEEE Transactions on Software Engineering, 2000,26(1):70~93.

[13] Ling S, Schmidt HW, Fletcher R. Constructing interoperable components in distributed systems. In: Meyer B, Mingins C, eds. Proceedings of the TOOLS Pacific'99. Melbourne: IEEE, 1999. 274~284.

[14] Murali S. Compositional performance reasoning. In: Harris CC, ed. Proceedings of the ICSE CBSE4. IEEE, 2001. 98~101.

[15] Xu QW, Swarup M. Compositional reasoning using assumption-commitment paradigm. Technical Report, UNU/IIST-136, the United National University, 1998.

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] 张世琨,张文娟,常欣,王立福,杨芙清.基于软件体系结构的可复用构件制作和组装.软件学报,2000,12(9):1351~1359.

[10] 杨芙清,梅宏,李克勤,袁望洪,吴穹.支持构件复用的青鸟 III 型系统概述.计算机科学,1999,26(5):50～55.