软件复合连接件的形式化研究*)

任洪敏[1] 张敬周[2] 钱乐秋[2]

(上海海事大学计算机系 上海200135)[1] (复旦大学计算机科学系软件工程实验室 上海200433)[2]

摘 要 基于构件的软件开发是软件开发的主流范型。构件通过连接件进行连接和交互。连接件是软件系统设计的一阶实体(First-class Entities),是决定软件系统功能和质量的重要因素。伴随软件系统和构件的规模及复杂程度日益增加,连接件变得愈加复杂和多样。因此,需要运用增量、合成的方法开发和支持复杂的连接件。本文提出一种根据已有连接件合成新连接件的形式化方法。该方法的特点是定义连接件合成的基本方式,即连接件运算符,组合运用连接件运算符合成复杂的连接件。并基于Wright连接件的形式化规约,给出了该复合连接件形式化规约的生成算法。

关键词 连接件合成,连接件,连接件运算符,高阶连接件

Research on a Formal Method for Software Composite Connectors

REN Hong-Min[1] ZHANG Jing-Zhou[2] QIAN Le-Qiu[2]

(Department of Computer Science and Technology, Shanghai Maritime University, Shanghai 200135)[1]

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

Abstract Composing complex software systems from coarse-grained components is the mainstream of software development paradigms. Components interact and communicate each other by connectors. Connectors are first-class entities in software system design and development. They are key determinants of system functions and properties. As software systems and components become more complex and larger increasingly, connectors become more complicated and diversiform. Hence it is very important to design and implement connectors in an incremental and compositional way. In this paper, a mechanism for producing new kinds of connectors systematically and compositionally from existing connectors is proposed. The mechanism is characterized by defining and combining the primitive composing methods of connectors, i.e. connector operators, to construct complex connectors. Moreover, based on architectural description language Wright, algorithms for generating formal specification of composite connectors are developed.

Keywords Connector composition, Connectors, Connector operators, Higher-order connectors

1 引言

软件构架技术(Software Architecture)[1]、中间件平台技术(Middleware Platform)[2]、基于构件开发技术(Component-based Software Development,简称CBSD)[3]用于开发大型、复杂软件系统,其核心都是运用大粒度构件合成软件系统。构件通过连接件进行连接和交互。连接件是软件系统设计的一阶实体(First-class Entities),是决定系统功能和质量属性,如性能、安全性、可靠性等的重要因素,应显式对待,独立存在[1,4]。

伴随软件系统和构件的规模及复杂程度日益增加,连接件变得愈加复杂和多样。但现在的CBSD技术、中间件平台技术、软件构架技术提供种类固定的基本连接件,不能满足构架设计和构件合成的需要。因此,需要运用增量、合成方法开发和支持复杂连接件[5~7]。

本文基于Wright连接件形式化规约方法,提出一种增量的、合成的设计和开发复杂连接件的方法。该方法的特点是定义连接件合成的基本方式,即连接件运算符,组合运用连接件运算符和已有连接件的形式规约,合成复杂的连接件及其形式规约。它简单易用,具有强的连接件合成功能。

运用本文提出的连接件合成方法,在软件构架形式规约的时候,能够带来三方面的好处:

(1) 复用已有连接件的形式规约,减少复杂连接件形式规约的开发成本,降低复杂连接件形式规约的理解难度。

(2) 提高构件连接的抽象级别和粒度。大粒度复杂构件的连接需要相应复杂的连接件,以克服运用过程调用等基本、低级连接件进行构件连接带来的诸多问题[1,8]。

(3) 帮助解决构架失配(Architectural Mismatch)[9]问题。复杂连接件的形式规约通过结构化方法复合而成,不是单体,奠定了高效分析构架失配的基础。

同时,该复杂连接件和相应形式规约合成方法,对代码级别可执行连接件的合成和相关代码生成具有指导和借鉴作用,能够作为代码级别连接件合成的语义定义。

本文第2节是相关研究工作的介绍,第3节给出连接件运算符的定义。第4节给出基本连接件的定义。第5节给出合成复杂连接件的方法。第6节举例说明连接件合成的方法。最后是全文总结和进一步的研究工作。

2 相关研究工作

文[1,8]指出支持合成连接件、用户自己定义连接件具有

*)基金项目:国家863高科技发展计划资助项目(2001AA1100241)和上海市教育委员会青年基金项目(2004166)。任洪敏 博士,讲师,主要研究领域为软件复用、形式化方法。张敬周 博士生,主要研究领域为软件复用、基于构件的软件工程。钱乐秋 教授,主要研究领域为软件工程。

重要意义，但未给出相应方法。David Garlan[7]1998年提出高阶连接件(Higher-order Connectors)，基本思想是合成连接件和服务功能，本质是单个连接件的调整变换。2001年，Bridget Spitznagel 和 David Garlan[5]提出了一个连接件合成方法。该方法给出5个连接件变换操作，其重在复合连接件的代码生成和变换。M. Wermelinger、A. Lopes[6,10]等亦展开了连接件合成的研究工作，并用范畴论给出了连接件合成的语义。

本文给出了一个不同的连接件增量合成的方法，即定义连接件的基本合成方式，即连接件运算符，组合运用连接件运算符和现有连接件，生成复杂连接件，并给出了其形式化规约的推导算法。

3 连接件运算符

连接件运算符是连接件的基本合成方式，它作用于现有的连接件，产生新的连接件。应该定义哪些连接件运算符，它们是否完备和最小，它们的精确语义定义，都值得探索和研究。

连接件是构件之间交互和通信的协议，它协调构件之间的交互，因此，具有一定的行为和时序特性。本文针对连接件的这种协调行为和时序特性，提出了五个连接件的基本运算符，包括一个单目运算符和四个二目运算符。这五个连接件运算符皆具有宏观的时序控制特性，运用它们作用于基本的连接件，能够生成具有复杂时序控制特性的软件连接件。

下面给出它们的基本定义。

设 P、Q 是两个连接件

定义1(连接件循环运算符) 循环运算符记为"@"，是一元运算符，$@P$ 是一新的连接件，它表示的连接交互机制是：根据构件的需要，能够多次执行连接件 P 的连接交互机制。

定义2(连接件顺序运算符) 顺序运算符记为";"，是二元运算符。$P;Q$ 是一新的连接件，它表示的连接交互机制是：其先执行连接件 P 的连接交互机制，然后执行连接件 Q 的交互机制。

定义3(连接件选择运算符) 选择运算符记为"□"，是二元运算符。$P□Q$ 是一新的连接件，它表示的连接交互机制是：其根据构件的需要，选择执行连接件 P 的连接交互机制或连接件 Q 的连接交互机制。

定义4(连接件并发运算符) 并发运算符记为"‖"，是二元运算符，$P‖Q$ 是一新的连接件，它表示的连接交互机制是：其并发执行连接件 P 的连接交互机制和连接件 Q 的连接交互机制。

定义5(连接件中断运算符) 中断运算符记为"^"，是二元运算符，P^Q 是一新的连接件，它表示的连接交互机制是：其先执行连接件 P 的连接交互机制，当连接件 Q 开始活动时，中断 P 的执行，执行连接件 Q 的连接交互机制。

五个运算符中，循环运算符@是一元运算符，优先级最高，并发运算符、选择运算符优先级次之，顺序运算符、中断运算符优先级最低。四个二元运算符都是左结合性，自左向右运算。

本文借鉴 CSP[11]中的运算符号作为连接件运算符号。一是因为它们都有相似的语义，二是方便第5节合成算法中角色规约、粘接规约 CSP 进程的生成。但连接件运算符独立于 CSP，其作用对象是连接件，CSP 运算符的作用对象是 CSP 事件和进程。

4 基本连接件

连接件运算符是构造连接件的基本方法。而基本连接件是构件交互的基本方式，是连接件构造的基本单元。具体选择哪些连接件为基本连接件，它们是否完备和最小，本文对此不予探讨。基本连接件与连接件运算符结合，能够构成一个完整的连接件合成体系，合成复杂的软件连接件。

根据基本连接件的形式化规约和在其上所做的连接件运算，能够推导复合连接件的形式化规约。基本连接件的形式化规约是复合连接件形式化规约推导的基础。常见的基本连接件包括过程调用、数据共享、事件机制和管道机制等。下面基于 Wright 体系结构描述语言，给出过程调用和事件机制两类连接件的形式化规约。

4.1 过程调用

文[12]给出了多次过程调用的形式化定义。考虑基本连接件的粒度问题和连接件循环运算符，选择一次过程调用作为基本连接件。下面三种基本连接件的定义遵循同样方式。过程调用形式化定义如下：

Connector Procedurecall
Role Caller=call→return→ §
Role Definer=call→return→ §
Glue=Caller. call→Definer. call→Definer. returen→Caller. return→ §

4.2 事件机制

事件交互机制是一个构件发出一个事件，其它多个构件对此事件进行响应的交互机制。其形式化定义如下：

Connector event (n:1..)
Role Announcer=announce→ §
Role Responder1..n=respond→ §
Glue=annlouncer. announce→(;i:1..n•Responderi. respond)→ §
Capped=Source. write?X→Capped□Source. close→ §

5 连接件合成方法

本节详细阐述论文提出的连接件合成方法、推导算法和合成约束。基本思想是：连接件包括粘接规约(Glue Specification)和角色两个部分，复杂连接件的粘接规约由子连接件的粘接规约按照一定方式复合而成，它的角色及其规约由子连接件的角色和规约按照相应方式复合而成。连接合成共分为四个步骤。

为精确定义软件连接的合成方法和奠定 CASE 工具研究的理论基础，先定义软件连接件的数学模型。

定义6 设所有合法的标识符集合是 *IDset*，所有合法 *CSP* 进程的集合是 *CSPset*。

定义7 所有已经定义的软件连接件类型的集合是 *Connectorset*，其定义如下：

$$Connectorset=\{\langle Conid, Roleset, Glue\rangle \mid Conid\in IDset, Roleset\subset P(IDset\times CSPset)\wedge(\forall\langle m_1,n_1\rangle,\langle m_2,n_2\rangle\in Roleset\ \langle m_1,n_1\rangle\neq\langle m_2,n_2\rangle\Rightarrow m_1\neq m_2), Glue\in CSPset, \forall\ a,b\in Connectorset\ a\neq b\Rightarrow a.Conid\neq b.Conid\}$$

即一个软件连接件类型由三个部分组成，连接件标识符，角色集合，表示交互协议的 CSP 进程。每个角色包含两个部分，即角色标识符和代表角色行为的 CSP 进程。连接件类型标识符和角色标识符皆不能同名。

5.1 定义合成需要的子连接件

合成连接件的第一步是根据待合成的复合连接件的意义和合成的需要，定义子连接件，子连接件是已有连接件类型的实例。为方便合成方法的阐述，设待合成的复合连接件命名为

Comconnect，定义的子连接件实例的集合是 *Instanceset*。连接件实例的定义方法和语义具体参见 *Wright* 语言。

5.2 定义合成表达式，给出其语法树

合成连接件的第二步是：根据合成的复合连接件的交互连接协议，确定复合连接件中子连接件间的关系，运用第4节定义的连接件运算符，定义相应的合成表达式。合成表达式是整个连接件合成的骨架。

定义8 合法的合成表达式定义：

(1)P 是连接件实例，则 P 是合法的表达式。

(2)P、Q 是合法的表达式，则@P、$P\|Q$、$P;Q$、$P\hat{}Q$、$P\square Q$ 是合法的表达式。

(3)P、Q 是合法的表达式，$(P\|Q)$、$(P;Q)$、$(P\hat{}Q)$、$(P\square Q)$是合法的表达式。括号运算优先级最高。

合成表达式的语法树，是其二叉树表示形式，用于复合连接件角色定义的生成算法。

定义9 合成表达式语法树的递归定义：若合成表达式是单个连接件实例，则相应的二叉树中仅有一个根结点，其数据域存放该连接件实例。若合成表达式＝(第一操作数)(连接件运算符)(第二操作数)，则相应的二叉树中以左子树表示第一操作数，右子树表示第二操作数，根结点存放连接件运算符(若为一元运算符，则左子树为空)。操作数本身又是合成表达式。

5.3 复合连接件角色合成和推导算法

合成连接件的第三步是复合连接件角色定义。复合连接件的每个角色由一个或多个子连接件的角色复合而成，其每个角色的行为规约由相应的子连接件角色的行为规约复合而成。一个复合连接件究竟应该包括几个角色，其每个角色包括哪些子连接件的角色，由复合连接件定义和设计人员决定，但须遵循一定的原则。为阐述相关原则，先定义两个函数。

定义10 定义函数 $InstancedRole: Instanceset \rightarrow \mathbb{P}(\{\langle p, id, csp\rangle \mid p \in Instanceset, id \in IDset, csp \in CSPset\})$，$p \in Instanceset$，则 $InstancedRole(p) = \{\langle p, id, csp\rangle \mid \langle id, csp\rangle \in p.Roleset\}$。$InstancedRole(p)$的结果是一集合，集合的成员是实例 p、p 的一个角色名字、该角色的 *CSP* 进程构成的元组。函数 *Instanceset* 的目的是给实例 p 的每个角色增加实例名限定，区分不同实例的角色，防止角色同名。

定义11 定义函数 $SubRole: Comconnect.Roleset \rightarrow \mathbb{P}(\{\langle p, id, csp\rangle \mid p \in Instanceset, \langle id, csp\rangle \in p.Roleset\})$，$r \in Comconnect.Roleset$，则 $SubRole(r) = \{\langle p, id, csp\rangle \mid$ 角色 r 由实例 p 的角色$\langle id, csp\rangle$合成$\}$。函数 $SubRole(r)$由用户定义，其定义了复合连接件角色 r 由哪些子连接件的角色复合而成。

复合连接件角色合成的原则如下：

(1)$\forall r \in Comconnect.Roleset, SubRole(r) \neq \phi$。即任何复合连接件的角色必须经由子连接件的角色复合而成。

(2)$\bigcup_{r \in Comconnect.Roleset} SubRole(r) = \bigcup_{p \in Instaceset} InstancedRole(p)$。即所有并且只有子连接件的角色参与合成复合连接件的角色。

(3)$\forall m, n \in Comconnect.Roleset\ m \neq n \Rightarrow SubRole(m) \cap SubRole(n) = \phi$。即任何子连接件的角色只能参与合成一个复合连接件的角色。

(4)$\forall r \in Comconnect.Roleset \forall m, n \in SubRole(r) m \neq n \Rightarrow m.p \neq n.p$。即复合连接件的一个角色只能由不同子连接件的角色合成。

已经确定了复合连接件的角色及其由哪些子连接件的角色合成之后，我们确定如何根据该信息和合成表达式，自动生成该角色的行为，即其 *CSP* 进程。设 $r \in Comconnect.Roleset$，是一复合连接件的角色，则 $SubRole(r)$记录了合成该角色的子连接件角色，生成该角色的 *CSP* 进程算法如下。

算法1 复合连接件角色的 *CSP* 进程生成算法

第一步：标记。在合成表达式语法树中对下述集合中的所有结点进行标记：

$\{p \mid p \in Instancese \wedge \exists\ id \in IDset \exists\ csp \in CSPset(p, id, csp) \in SubRole(r)\}$，即标记所有参与合成角色 r 的叶子结点。

第二步：裁剪。删除语法树中所有未被标记的叶子结点。对语法树中的内部结点，如果其左、右子树中都无被标记的结点，则删除该结点和其左、右子树；如果其左、右子树中有一方存在被标记的结点，则删除该结点和其不存在标记结点的子树，并用其另一存在标记结点的子树代替其作为其父结点的子树。

第三步：生成。根据裁剪过后剩余的语法树，生成相应的表达式。对该表达式中的每一个连接件实例 k，有$\langle k, kid, kcsp\rangle \in SubRole(r)$，用 $k.kid$ 代替。最后所得表达式即代表角色 r 的行为。

该算法具体可用二叉树的遍历算法实现，故其时间复杂度和空间复杂度同于二查树遍历算法，都为 $O(n)$，n 为合成表达式语法树结点个数。

5.4 复合连接件粘接规约的推导算法

算法2 复合连接件粘接规约生成算法

复合连接件粘接规约的生成相当简单。把合成表达式中每个实例 $p \in Instanceset$，换成 $p.Glue$，保持整个表达式的结构不便，即得到复合连接件的粘接规约。该算法通过对合成表达式顺序扫描、替换即可实现，故其时间复杂度和空间复杂度为 $O(n)$，n 为合成表达式中连接件实例出现次数。

6 连接件合成举例

本节通过一个例子，具体阐述上述连接件合成方法。表明运用该连接件合成方法，能够简单地生成复杂的连接件和其形式规约，并使复杂连接件的理解变得简单。如果运用手工，从头开始生成这样复杂的连接件及其 CSP 形式规约，则是个繁琐的任务。

举例：设一电影点播系统，其客户构件和服务构件的交互机制是：首先客户构件请求电影服务，然后服务构件要求客户构件提供帐号，得到帐号后，服务构件向客户构件提供电影服务。同时，在上述过程中，服务构件能够向客户构件多次发送广告信息。命名该连接交互机制为 *Watchmovie*。*Watchmovie* 涉及两个过程调用，一个管道机制和一个事件机制，所需连接件实例定义如下：

Pmovie, *Paccount*: *Procedurecall*; //分别用于电影服务请求和帐号请求。

aMovie: *Pipe*; //用于发送电影数据流。

aEvent: *Event*(1); //用于发布广告，只有一个事件响应角色。

根据前述 *Watchmovie* 的交互方式和连接件运算符、合成表达式定义，得出复合连接件 *Watchmovie* 的合成表达式：$(Pmovie; Paccount; amovie) \| @\ aEvent$。其语法树如图1。

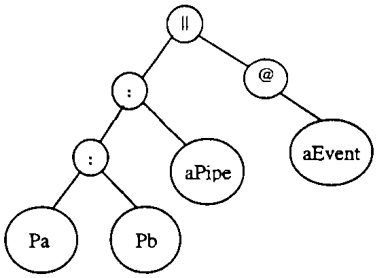

图1 举例复合连接件的语法树

Watchmovie 具有角色 *Viewer* 和 *Cinema*，代表电影点播的客户端构件和服务端构件。根据前述，*Viewer* 由 *Pmovie* 的角色 *Caller*，*Paccount* 的角色 *Definer*，*amovie* 的角色 *Sink*，*aEvent* 的角色 *Responder* 合成，即 $SubRole(Viewer) = \{\langle Pmovie, Caller\rangle, \langle Paccount, Definer\rangle, \langle aMovie, Sink\rangle, \langle aEvent, Announcer\rangle\}$。

能够推导得出 *Watchmovie* 的角色 *Viewer* 的行为规约，即：

$$Viewer = ((Pmovie.Caller; Paccount.Definer; aMovie.Sink) \| @\ aEvent.Responder_1$$

其直观语义是：角色 *Viewer* 先按照 *Pmovie* 的角色 *Caller* 规定的方式进行行为，再按照 *Paccount* 的角色 *Definer* 规定的方式进行行为，然后按照 *aMovie* 的角色 *Sink* 规定的方式进行行为，与此同时，其按照 *aEvent* 的角色 *Reponder*1进行若干次行为。

推导得出复合连接件 *Watchmovie* 的粘接规约如下：

$$Watchmovie.Glue = (Pmovie.glue; Paccount.glue; aMovie.glue) \| @\ aEvent.glue$$

结论 连接件是软件系统设计的一阶实体，如同构件合成，连接件合成的研究具有重要意义。本文提出了一种连接件合成的形式化方法。该方法的特点在于分析和得出连接件合成的基本方式，即连接件运算符，组合运用连接件运算符和连接件实例，构造复杂的连接件。并基于 Wright 的连接件形式规约方法，给出了复合连接件形式化规约的推导算法和合成约束。

进一步的研究工作中，我们将致力运用该合成方法对实际系统的连接件进行建模，加深对连接件合成的理解。同时，遵循本文提出的复合连接件和其形式规约生成方法，探索可执行的、代码级别的连接件的合成和相关代码的生成。

参 考 文 献

1 Shaw M, Garlan D. Software architecture: perspectives on an emerging discipline. Prentice Hall, Inc. 1996. 165～172
2 Object Management Group. CORBA, OMG Website. 2000. http://www.omg.org
3 Szyperski C. Components Software—Beyond Object-Oriented Programming. Cambrige, MA: Addison-Wesley Publishing Company, 1997. 3～13
4 Shaw M, DeLine R, Klen D V, et al. Abstractions for software architecture and tools to support them. IEEE Trans. on Software *Engineering*, *1995*, *21(4)*: *314～355*
5 Spitznagel B, Garlan D. A Compositional Approach for Constructing Connectors. In: Proc. of the 2nd IEEE/IFIP Working Conf. on Software Architecture, Los Alamitos: IEEE Press, 2001. 148～157
6 Lopes A, Wermelinger M, Fideiro J. Higher-Order Architectural Connectors. 2001. http:// www-ctp. di. fct. unl. pt/ ～mw/pubs/2001/bighocs. pdf
7 Garlan D. Higher-order Connectors. In: Proc. of Workshop on Compositional Software Architectures, California, 1998. 3～12
8 Shaw M. Procedure Calls are the Assembly Language of Software Interconnections: Connectors Deserve First-Class Status: [Technical Report, CMU//CS-94-107]. Carnegie Mellon University, 1994
9 Gacek C. Detecting Architectural Mismatches During Systems Composition: [Technical Report, USC-CSE-97-506]. University of Southern California, 1997
10 Wermelinger M, Lopes A, Fiadeiro J L. Superposing Connectors. In: Proc. of 10th Intl. Workshop on Software Specification and Design, IEEE Computer Society Press, 2000. 87～94
11 Hoare C A R. Communicating Sequential Processes. Prentice Hall, 1985
12 Allen R J. A Formal Approach to Software Architecture: [Ph. D. Thesis]. Carnegie Mellon University, Pittsburgh, 1997

（上接第120页）

据处理的能力。另外，联机分析、数据挖掘技术以及信息可视化处理技术等也都是提供智能化决策支持系统的必备手段。

构建电子校务应用体系需要很强的技术保证，除了上面提及的之外，还包括其它许多保障可靠性和高可用性的相关技术，如网络存储与容灾技术等。

结束语 电子校务工作是一个不断探索、逐步完善的过程，可以开拓出一系列全新的研究、发展与应用领域，它将实现学校资源数字化、通信网络化、教学现代化、学习个性化、办公自动化、管理科学化，由此带来教学管理、课堂教学组织、教育评价、德育工作等等从观念到实施的深刻转变。虽然本文给出了目前的电子校务建设框架，但随着人们对电子校务的认识不断深入，随着计算机软、硬件技术创新的不断涌现，在不久的将来，电子校务的建设还会向更高的层次发展。

参 考 文 献

1 Gómez A F, Martínez G, Cánovas óscar. New security services based on PKI [J]. Future Generation Computer Systems, 2003, 19(2): 251～262
2 Hays J D, et al. Earth science instruction with digital data [J]. Systems and Software, 2000, 26(6): 657～668
3 Yang Yanyan, Rana O F. Agent based data management in digital libraries [J]. Parallel Computing, 2002, 28(5): 773～792
4 Yan Hongfei, Wang Jianyong. Architectural design and evaluation of an efficient Web-crawling system [J]. Systems and Software, 2002, 60(3): 185～193
5 Cecchet E, Marguerite J. Performance and Scalability of EJB Applications [J]. ACM Sigplan Notices, 2002, 37(11): 246～261
6 李庭晏，王倩宜，宋式斌. 新形势下电子校务的建设与发展[J]. 实验技术与管理，2004，21(2)
7 教育部办公厅关于教育电子政务建设的指导意见[J]. 中国现代教育装备，2004. 2
8 曾月，范玉顺. 基于 COM 和 ASP 技术的工作流管理系统的设计与实现[J]. 计算机工程与应用，2002(1)：38～38
9 张传武，等. 细胞自动机制换群加密技术研究[J]. 计算机科学，2003，30(3)：171～174