一种基于代数语义的软件体系结构求精方法

林雷蕾[1] 周 华[1] 代 飞[1,2] 何臻力[1] 沈 勇[1] 康洪炜[1]
(云南大学软件学院 昆明 650091)[1] (云南省软件工程重点实验室(云南大学) 昆明 650091)[2]

摘 要 软件体系结构是引导需求到实现的桥梁,目前在软件体系结构建模方法中主要分为形式化和非形式化两种。针对大型分布式系统的体系结构采用Petri网进行建模,兼顾了可视化操作和形式化的准确性,同时利用细化求精操作建立体系结构的层次模型,有效解决了状态空间爆炸问题。此外,在求精过程中为了保证用于下层求精的子网能准确表达上层行为规约,引入了进程代数来刻画Petri网的行为语义。最后,给出了进程项构造子网的算法及案例研究,并通过开源工具验证上述内容的正确性。

关键词 体系结构,进程代数,构件重用,ACP,Petri网
中图法分类号 TP311 **文献标识码** A **DOI** 10.11896/j.issn.1002-137X.2017.07.026

Method of Software Architecture Refinement Based on Algebraic Semantics

LIN Lei-lei[1] ZHOU Hua[1] DAI Fei[1,2] HE Zhen-li[1] SHEN Yong[1] KANG Hong-wei[1]
(School of Software, Yunnan University, Kunming 650091, China)[1]
(Key Laboratory for Software Engineering of Yunnan Province, Yunnan University, Kunming 650091, China)[2]

Abstract Software architecture is a bridge to connect the requirements and achievements. At present, there are two approaches to model architecture, which are formal and informal. In order to strike a balance between convenient operation and accuracy, we applied Petri nets to model the architecture of large and complex systems, especially in distributed system. First of all, this paper proposed a top-down approach for refining transitions by subnets in Petri nets. This method can establish hierarchical structure model of the complex system. In addition, the method effectively solves the problem of state space explosion. Moreover, we used process algebra to define the algebraic semantics of the behavior of Petri nets, which can guarantee the consistency of behavior between the subnets and the specification. Besides, we gave an algorithm for transforming a process term into a subnet. Finally, a case study was given to verify the correctness of above ideas through open source tools.

Keywords Architecture, Process algebra, Component reuse, ACP, Petri nets

1 引言

软件体系结构(Software Architecture)自20世纪90年代开始就是软件工程领域的研究热点。在软件开发过程中,体系结构的好坏直接关系到软件项目的成败。然而随着社会进步,系统的复杂程度不断增加,规模也在扩大,导致建模系统的体系结构变得异常困难。尤其是分布式系统的出现,使得建模软件体系结构的难度更大。但是,针对分布式系统的研究具有一定理论意义和实践价值。C. A. R. Hoare曾指出:"分布式计算是一个具有重大理论与实践意义的迷人课题,其迷人之处在于理论与实践的同步发展,一方面实践推动了理论,另一方面理论又指导着实践。"因此,分布式系统的体系结构建模研究将是一个持续讨论的话题。

软件体系结构研究主要集中在如何对软件体系结构进行描述,即体系结构描述语言(Architecture Description Language, ADL)。文献[1]采用数据互操作标准XML作为元语言对体系结构进行描述,并支持软件系统演化。朱雪阳等人运用时序逻辑描述软件体系结构,不仅能实现静态语义表示,而且能刻画其动态语义[2]。此外,较为熟知的体系结构描述语言还有Wright, Rapid等[3-4]。在体系结构建模中,主流方法是采用从上往下细化求精的方法。以上ADL虽然有些支持,但在求精过程中需要坚实的形式化基础。文献[5-6]都是

到稿日期:2016-06-04 返修日期:2016-10-18 本文受国家自然科学基金项目(61462095),云南省高新技术产业发展项目(云发改高技[2012]1956号),云南省中青年学术和技术带头人后备人才培养(C6143002),云南省应用基础研究计划项目:过程编排定义和可实现性研究(2016FB102),云南省教育厅科学研究基金研究生项目(2017YJS107),云南省软件工程重点实验室专项研究项目(2016SE202)资助。

林雷蕾(1989—),男,博士,主要研究方向为软件体系结构、软件工程,E-mail:leilei_lin@126.com;**周 华**(1963—),男,博士,研究员,主要研究方向为软件体系结构(通信作者);**代 飞**(1982—),男,博士,副教授,主要研究方向为软件工程、业务建模;**何臻力**(1987—),男,博士,讲师,主要研究方向为软件体系结构、云计算;**沈 勇**(1970—),男,硕士,副研究员,主要研究方向为软件体系结构;**康洪炜**(1976—),男,博士,副研究员,主要研究方向为软件体系结构。

在UML的基础上对体系结构进行建模求精，虽然具备可视化表示，但缺少形式化支持。

分布式系统通常由分布在不同地理位置上的节点通过局域网或广域网连接在一起组成。每个节点都可以是一个独立的系统单元，各自处理相应的业务。这些特性决定了分布式系统体系结构中每个节点的细化求精需通过消息传递的方式进行。进一步讲，传统的体系结构建模求精方式是基于上下文无关的。而分布式系统的体系结构在细化求精过程中需考虑如何保证不同层次体系结构之间的语义一致性问题。文献[7]提出基于上下文有关文法对构件进行描述，进一步从结构上对构件进行求精建模。但是该方法缺少图形化表示，晦涩的形式化逻辑推理让人难以把握其正确性。

针对上述问题，本文提出了基于代数语义的体系结构求精方法。其核心思想是用Petri网来建模体系结构模型，同时引入进程代数(Algebra of Communicating Processes，ACP)来刻画Petri网的行为语义，并通过进程代数在Petri网点火执行过程中的传递来进一步求精体系结构模型。

本文第1节总结了当前在分布式系统的体系结构建模研究过程中的不足之处；第2节引入了Petri网等基本概念术语；第3节采用Petri网建模体系结构内容；第4节扩展了Petri网的语义，引入进程代数；第5节对案例进行研究；最后对相关工作进行比较。

2 基本概念

2.1 Petri网

Petri网是公认的比较好的描述和分析并发系统的模型。它不仅可以描述系统的静态结构，也能描述系统的动态行为；不仅具有坚实的数学理论基础和成熟的分析技术，也具有直观的图形化表示。本文的重点是讨论如何运用Petri网建模软件体系结构，因此为了突出重点，这里只引入与本文相关的少数几个概念。为了简便，本文没有涉及权重和容量等概念，更多内容可以参见文献[8]。

定义1(Petri网) Petri网是一个四元组 $\Sigma=(P,T;F,M_0)$，其中：

(1)$P\cup T\neq\emptyset$ 且 $P\cap T=\emptyset$，习惯称 P 为库所集，T 为变迁集；

(2)$F\subseteq(P\times T)\cup(T\times P)$，称 F 为流关系；

(3)$dom(F)\cup cod(F)=P\cup T$，其中 $dom(F)=\{x\in P\cup T\mid \exists y\in P\cup T:(x,y)\in F\}$，$cod(F)=\{x\in P\cup T\mid \exists y\in P\cup T:(y,x)\in F\}$；

(4)$M:P\rightarrow\{0,1,2,\cdots\}$，称 M 为 Σ 的一个标识，习惯用 M_0 表示 Σ 的初始标识。

定义2(前集和后集) 设 $\Sigma=(P,T;F,M_0)$ 是一个Petri网，$x\in P\cup T$，${}^{\cdot}x=\{y\mid(y,x)\in F$，则称 ${}^{\cdot}x$ 为 x 的前集；$x^{\cdot}=\{y\mid(x,y)\in F\}$，则称 $x^{\cdot}$ 为 x 的后集。

定义3(Petri网的变迁发生规则) 设 $\Sigma=(P,T;F,M_0)$ 是一个Petri网，并满足下述变迁发生规则(transition firing rules)：

(1)对于变迁 $t\in T$，如果 $\forall p\in P:p\in{}^{\cdot}t\rightarrow M(p)\geqslant 1$，则称变迁 t 在标识 M 下有发生权，记为 $M[t\rangle$；

(2)若 $M[t\rangle$，在标识 M 下，变迁 t 发生点火，则标识将由 M 演变为一个新的标识 M'，记为 $M[t\rangle M'$，对 $\forall p\in P$，满足：

1)$M'(p)=M(p)-1$，若 $p\in{}^{\cdot}t-t^{\cdot}$；

2)$M'(p)=M(p)+1$，若 $p\in t^{\cdot}-{}^{\cdot}t$；

3)$M'(p)=M(p)$，其他。

2.2 进程代数ACP

通信进程代数(Algebra of Communicating Processes，ACP)由J. A. Bergstra等提出[9]，其目的是通过代数演算的方式来研究系统的行为。一般认为，进程代数中“进程”是刻画系统行为或动作的，这里的系统是广义上的系统，一个计算系统或一个人等。“代数”表明了这类研究的基础是代数方法。具体讲，进程代数的思想是将一个有机整体抽象为系统，用严格的语义描述系统行为或动作，并通过语法规则来演算系统的动态行为。ACP的内容较为丰富，以下只引入与本文相关的少数几个概念，其他内容可参见文献[10]。

定义4(顺序复合操作) 两个行为特征 a 和 b，通过顺序复合算子·组成一个复合的行为特征，表示为 $a\cdot b$。其语义就是行为 a 执行完之后，b 才开始执行。

定义5(选择复合操作) 两个行为特征 a 和 b，通过选择复合算子+，组成一个复合的行为特征，表示为 $a+b$。其语义就是在行为特征 a 和 b 之间选择其中之一，并按照选择的行为特征操作。

定义6(并发复合操作) 两个行为特征 a 和 b，通过并行复合算子‖组成一个复合的行为特征，表示为 $a\parallel b$。其语义就是在行为特征 a 和 b 之间没有执行的先后关系，二者可以并发执行。

定义7(递归复合操作) 对于行为特征 a，通过递归复合算子*组成一个复合的行为特征，表示为 $(a)^*$。其语义就是重复 a 规定的行为操作 n 次，然后终止，n 是大于或等于0的自然数。

定义8(进程项) 由行为特征集合中的元素通过算子有限次(含0次)复合而成的代数项称为进程项(process term)。

注：在文献[10]中，作者将执行的动作称为活动，本文为了保持文章一致性，将其称为行为特征。此外，复合算子不仅仅只有顺序复合算子·、选择复合算子+、并发复合算子‖和递归复合算子*这4种，还有封装算子、左并发算子等。但为了突出研究重点，本文没有涉及。

3 体系结构建模

传统的描述体系结构方法可以分为两种：可视化和形式化。前者多数集中在采用UML图形加自然语言描述，后者主要采用复杂的符号形式进行刻画，如时序逻辑、Z语言和PI演算等。但是这两种方法都各有缺点，可视化缺乏准确的描述而形式化又不够直观且难以掌握。权衡二者的利弊，本文最终选定Petri网作为建模软件体系结构的工具。其主要原因是Petri网既有直观的图形化表示，又兼顾了形式化语义的准确性。

虽然软件体系结构的研究一直是研究热点，但是目前还没有一个标准化的、统一的体系结构定义。但是，从当前对体系结构的定义描述中可以抽取出两个重要元素：构件和连接件[2]。

软件体系结构站在全局的角度去描述软件项目每个模块之间的协作关系，整体上对各个模块进行管理，以达到高效地完成目标需求。因此，构件可以理解为一个功能模块，而模块之间的协作则通过连接件进行。同时，每个功能模块又可以划分为更小的单位，因此在细化求精过程中其针对构件进行细化。

Petri 网一般用矩形变迁，用圆圈表示库所，用带有箭头的直线表示流关系，用黑点表示托肯。为了区分构件和连接件，本文用实线矩形表示构件，虚线矩形表示连接件，如图 1 所示。

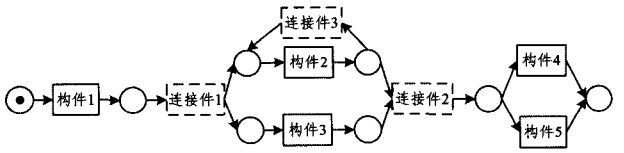

图 1　体系结构 Petri 网模型

如图 1 所示，该体系结构中只有 5 个构件和 3 个连接件，但是在针对分布式系统这种大型的复杂系统时，构件和连接件的个数规模是相当大的。这种情况不仅会让整个 Petri 网模型变得异常复杂，更会导致状态空间爆炸。解决这类问题行之有效的方法是对 Petri 网进行分层。具体讲，就是从上往下逐步求精建立体系结构模型。细化求精的过程实质就是采用一个 Petri 网的子网替代一个构件的过程。为此，借鉴文献[11]的思想，定义了子网及求精操作。

定义 9(子网)　子网是一个五元组 $S=(P_0,T_0;F_0,t_s,t_e)$，其中：

(1) ${}^{\cdot}P_0 \cup P_0{}^{\cdot} \subseteq T_0$；

(2) $t_s,t_e \in T_0$ 分别为子网 S 的入口及出口，且满足 ${}^{\cdot}t_s=\emptyset, t_s{}^{\cdot} \in P_0, {}^{\cdot}t_e \in P_0, t_e{}^{\cdot}=\emptyset$；

(3) 令 $T'=T_0-\{t_s\}-\{t_e\}$，${}^{\cdot}T' \cup T'{}^{\cdot} \subseteq P_0$。

定义 10(求精操作)　设 $\Sigma=(P,T;F,M_0)$ 是一个 Petri 网，$t \in T$，$S=(P_0,T_0;F_0,t_s,t_e)$ 是一个子网。细化操作 $Refine(t,S)$ 将 Petri 网中的变迁 t 细化为子网 S，得到一个新的 Petri 网 $\Sigma'=(P',T';F',M_0')$，其中：

(1) $P'=P \cup P_0$；

(2) $T'=T-\{t\} \cup T_0$；

(3) $F'=F-({}^{\cdot}t \times \{t\})-(\{t\} \times t^{\cdot}) \cup F_0 \cup ({}^{\cdot}t \times t_s) \cup (t_e \times t^{\cdot})$。

需要指出的是，定义 10 中的变迁 t 是上层模型中的构件，而 t_s,t_e 是下层模型中子网 S 的变迁，同时它们又是子网 S 的入口和出口。当执行细化操作时，变迁 t 便由子网 S 替换，从而形成更为复杂的体系结构模型。

求精操作有效避免了分布式系统的体系结构模型状态空间爆炸问题。但是，随之而来的问题是如何保证不同层次体系结构之间的语义一致性。在建模一个系统的体系结构时，上层体系结构中的一个构件可以看作是下层子网的规约。下层子网是对上层构件的具体实现。因此，充分准确地理解分布式系统中构件之间的行为语义是保证模型正确性的前提。如图 2 所示，通过求精操作对构件进行细化。

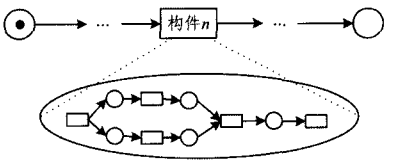

图 2　体系结构层次模型

在图 2 中，构件 n 进行求精的子网为椭圆形中展示的 Petri 网子网。也就是说，子网最后表现的行为必须与构件 n 在上层中体现出的功能一致。

4　代数语义

分布式系统的一个主要应用特征就是消息传递。具体讲，分布式应用的软件系统运行环境是由分布在不同地理位置的节点通过网络连接组成的。单个节点可以看作是一个独立单元。这些单元要在外部环境输入消息的情况下，具体执行功能模块产生输出。软件系统的结构应该是由一组通过消息传递的构件组成的，因此在建立分布式系统体系结构时，可以考虑如何通过消息传递来构建整个结构。

4.1　语义等价

Petri 网是通过变迁的执行来描述系统的行为语义，进程代数 ACP 采用代数演算的方式来刻画系统的行为，二者之间具有天然互补性。进程代数提出的背景是希望通过严谨代数演算的方式来表示行为，这种行为含义包括了不同系统之间的交互，但是其缺点是需要使用者具备很强的数据基础以及缺乏直观的图形化操作。相反，Petri 网虽然在描述不同系统交互行为方面偏弱，尤其是信息传递方面，但是它具有很好的图形化界面，给使用者以直观的判断。同时，Petri 网又具备了精确的形式化语义，而其他图形化操作语言(如 UML)是半形式化的描述系统，存在二义性。

本文在建模复杂系统时考虑到了 Petri 网的优势，又因为大型分布式系统之间的行为语义具备了分布式系统的消息传递的属性，因此采用进程代数来描述。二者的结合不仅直观地建模了复杂系统的层次结构模型，还很好地解决了不同层次模型之间的行为一致问题。为了分析二者之间的关系，首先给出 Petri 网结构分析以及 Petri 网的代数语义定义。

Petri 网的基本结构可以分为 4 种：顺序结构、选择结构、循环结构和并发结构。通过分析可以发现，以上 4 种 Petri 网结构在刻画构件执行行为语义时，与之前分析的 ACP 4 种复合操作是相似的。此外，无论多复杂的 Petri 网模型都可以通过这 4 种基本结构组合或嵌套形成。因此，通过 4 种基本复合操作形成的进程项也可以准确刻画一个 Petri 网的行为语义内容。为了证明进程项在刻画行为语义方面与 Petri 网结构的变迁执行语义是等价的，给出以下证明。

证明：参见文献[10]，主要思想是采用互模拟来证明语义等价，此处不再赘述。

定义 11(Petri 网的代数语义)　由进程代数 ACP 中的进程项刻画 Petri 网构件之间的行为关系，称为 Petri 网的代数语义。

为了说明语义之间的等价关系，图 3 示出了 Petri 网 4 个基本结构的对应代数语义，即进程项内容。其他无论多复杂

的 Petri 网结构都可以通过这 4 个基本进程项的复合操作来进行刻画。需要注意的是，进程项中的行为特征元素刻画的是构件的动作，而在 Petri 网建模体系结构模型中，除了构件外，还存在连接件(用虚线矩形表示)。因此，为了保证进程项能准确表达构件之间的行为关系，在并发结构和循环结构中加入了连接件，使得最小单元的行为整体与进程项是一致的。同时，该 Petri 网结构与实际建模情况也是相吻合的。

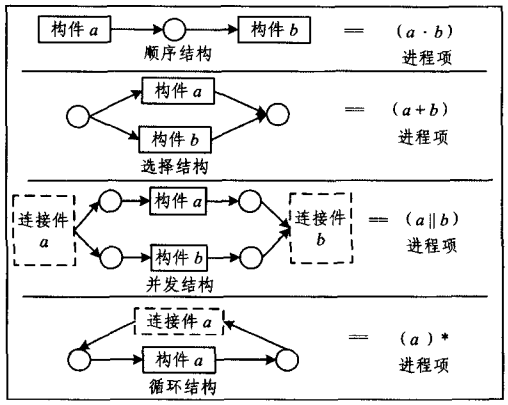

图 3 Petri 网模型的等价代数语义

4.2 引入进程代数

信息传递包括两个方面：传递方式及传递内容。传统的 Petri 网只能刻画传递方式，不能表示传递内容。在建模该类系统的体系结构时，为了不破坏原有特征的一致性，同时保证体系结构的求精方法能准确理解构件间的语义，本文在 Petri 网的基础上扩展了进程代数内容。其主要思想是信息的传递过程通过原有的 Petri 网点火规则进行传递，而信息的内容则用进程代数 ACP 的进程项来刻画。

由于要通过消息传递来建立具有层次的分布式系统体系结构模型，因此在细化构件过程中，要准确无误地指导构件细化的行为内容，即 Petri 网的结构模型。为此，先给出每个构件的行为规约，在细化过程中，子网的结构要与行为规约保持一致。

定义 12(行为规约) 行为规约 B 是行为空间 g 的集合。$g=\langle v,id,c\rangle$ 是一个三元组，其中 v 的取值范围为自然数，代表体系结构的层次；id 代表每个构件的唯一标识；c 是一个进程项，用于刻画构件的子网行为。

例子：假设体系结构模型中构件 i 细化的子网要求是开始执行构件 a，随后执行构件 b，最后并发执行构件 c 和构件 d。根据行为规约定义，使用进程项，行为规约可以描述为：$B=\{\langle 0,i,(a\cdot b\cdot(c\parallel d))\rangle\}$，其中，数字 0 代表最顶层模型；$i$ 是构件的唯一标识；进程项 $(a\cdot b\cdot(c\parallel d))$ 是构件 i 要细化的子网行为内容，· 代表顺序复合，‖ 代表并发复合。

Petri 网用托肯表示变迁执行的条件，由于在变迁细化时要考虑到行为规约，因此可以用托肯来装载行为规约，当构件满足执行条件后，可以从托肯中获取规约资源，从而指导构件求精。扩展后，托肯不仅代表点火执行条件，更富有资源内容。本文将这种 Petri 网称为 APN(Algebra Petri Net)。由于 APN 在结构及元素上未进行任何改动，只是丰富了托肯内容，即增加了 Petri 网求精操作的判断。为此，定义 APN 点火执行中的求精判断规则如下。

定义 13(APN 的变迁求精规则) 设 $\Sigma=(P,T;F,M_0)$ 是一个 APN 网，并满足下述变迁发生规则：

(1)对于变迁 $t\in T$，如果 $\forall p\in P: p\in {}^{\cdot}t\rightarrow M(p)\geqslant 1$，则称变迁 t 在标识 M 下有发生权，记为 $M[t\rangle$；

(2)若 $M[t\rangle$，t 为构件变迁且 B 中存在与 t 对应的行为空间 g，则执行 $Refine(t,S)$。其中，S 是子网，且 S 等价于 $g.c$。

4.3 构造算法

从 APN 求精规则的定义中可以发现，第一条规则与 Petri 网的变迁发生规则是一致的，即没有破坏 Petri 网的原有执行条件。第二条规则判断求精主要是细化的子网要等价于行为空间中的进程项。因此，要证明是否任给一个进程项都能转化为一个 Petri 网子网。以下给出证明(定理 1)及转化算法。

定理 1 任给一个行为规约 B，则可以构造一个唯一的具有层次结构的体系结构 Petri 网模型。

证明的主要思路是将进程项转化为 Petri 网子网。因为行为规约是行为空间的集合，每个行为空间对应一个需要进行求精操作的构件，而求精的内容用进程项刻画。采用反证法求证。

证明：假设子网 S 不能用进程项 $g.c$ 刻画，则必存在一种结构不能用定义 11 中的 4 种基本结构嵌套或组合形成。为此分两种情况考虑：

(1)存在第 5 种基本结构描述两个构件之间的行为关系。即两个构件不是顺序关系、选择关系和循环关系，也不是并发关系。这与 Petri 网定义的内容是互相矛盾的，因此不成立。

(2)存在一种 Petri 网嵌套结构无法用进程项明确刻画构件之间的关系。如图 4 所示，是先刻画构件 a 与构件 b 之间的选择关系 $(a+b)$，还是先刻画构件 a 与构件 c 之间的顺序关系 $(a\cdot c)$。如果存在这种情况，很容易证明该子网具有 PT 型结构或者 TP 型结构，即会造成模型的死锁和无界现象。证明思路可参见文献[12]。所以这种结构是不正确的，也是建模者要避免的，因此不成立。

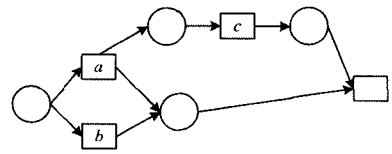

图 4 证明例子

因此，综上所述，任给一个行为规约是可以构造成对应的层次 Petri 网。证毕。

算法 1 进程项转为子网

输入：任意进程项 c

输出：Petri 子网 S

BEGIN

1. 定义堆栈元素 stack 和 temp；
2. 初始化变量：整型 iterator=0，布尔型 loop=false，整型 tempIter=−1；
3. stack. push('#')，并且在 c 后面加入字符'#'；
4. WHILE(iterator 小于 c 的长度) //循环遍历 c
   4.1 IF(c 当前字符不等于')'或'#')//压栈
      4.1.1 stack. push(当前字符)；

4.2 ELSE IF(c 当前字符等于')')//出栈

4.2.1 tempIter=iterator+1;

4.2.2 IF(c[tempIter]==' * ')

4.2.2.1 loop=true，并且 iterator 下移；

4.2.3 将 stack 中的字符执行出栈操作，同时压栈进入 temp 中，直到碰到'('停止。

4.2.4 Char T′=createComponent(temp,loop);// 以括号为单元将子进程项转化为新构件

4.2.5 stack.push(T′)，同时 temp 清空，loop=false;

4.3 ELSE IF(c 当前字符等于'#') //处理完成

4.3.1 stack.clear();

4.4 iterator 下移;

END

从算法 1 中可以看出，由于进程项是由多个基本的复合算子组成的，每种基本复合操作可以对应为一种基本的 Petri 网结构，因此算法主要是将每个括号中的子进程项抽取出来，优先转化为一个基本结构，再将这个基本结构看作一个新的构件 T'，与其他构件继续复合操作，逐步形成一个子网。算法 2 的主要内容就是将一个括号中的子进程项转化为一个新构件。

算法 2　子进程项转为新构件

输入：任给一个括号中的子进程项 temp 和循环变量 loop

输出：新的构件 T′

BEGIN

1. 定义新的构件字符为 T′;

2. WHILE(temp≠Ø&& 指针 i 不越界)

2.1 IF(temp.size()==1)

2.1.1 IF(loop)

2.1.1.1 构造循环结构，并赋值给 T′，跳出循环；

2.1.2 ELSE 直接跳出循环；

2.2 IF(temp[i]==' + ') //构造选择结构

2.2.1 temp 弹出第 i+1 和第 i−1 两个构件进行选择复合操作，并赋值给 T′;

2.2.2 temp 进栈 T′，同时栈顶位置赋值给 i;

2.3 按照步骤 2.2 同理进行顺序和并发两种结构复合操作；

2.4 指针 i 下移一位；

3. RETURN T′;

END

5　案例研究

当前网上购物已经成为日常生活中的一部分内容，各种购物节的兴起更是创造了各大商业银行单位时间内交易量峰值的奇迹。大多数网上银行的体系架构都采用分布式，相对于其他架构其特点主要是支持高访问量、可扩展性强和较好的风险控制。在此背景下，给出网上银行的体系结构求精过程。如图 5 所示，银行体系结构最初的构件系统模型主要包括 3 个：登录构件 $t0$、业务处理构件 $t1$ 和安全退出构件 $t2$。

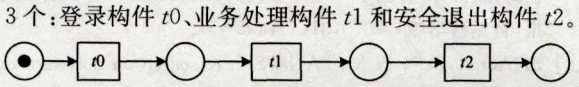

图 5　网上银行核心构件

现在需要对整个体系结构进行求精，根据实际分析可以得到业务处理过程中涉及到安全构件 $t1.1$、信息查询构件 $t1.2$ 和网上支付构件 $t1.3$，并且查询和支付是选择关系，安全检测是在二者之前进行的。因此，三者之间的代数语义为：$(t1.1 \cdot (t1.2+t1.3))$。

网上支付过程又包含支付口令构件 $t1.3.1$、转账构件 $t1.3.2$、在线付款构件 $t1.3.3$、短信验证构件 $t1.3.4$。其中转账和付款是选择进行，短信验证可以重复发送接收。因此其代数语义是：$(t1.3.1 \cdot (t1.3.2+t1.3.3) \cdot (t1.3.4)^*)$。此外，安全构件中主要包含加密构件 $t1.1.1$ 和访问控制构件 $t1.1.2$，二者可以并发执行，代数语义是：$(t1.1.1 \parallel t1.1.2)$。

根据上以实际分析内容，给出网上银行体系结构的行为规约 $B=\{\langle 0,t1,(t1.1 \cdot (t1.2+t1.3))\rangle,\langle 1,t1.3,(t1.3.1 \cdot (t1.3.2+t1.3.3) \cdot (t1.3.4)^*)\rangle,\langle 1,t1.1,(t1.1.1 \parallel t1.1.2)\rangle\}$。根据当前行为规约，可以将网上银行的体系结构建模出来，并且该模型具有 3 层结构，如图 6 所示。

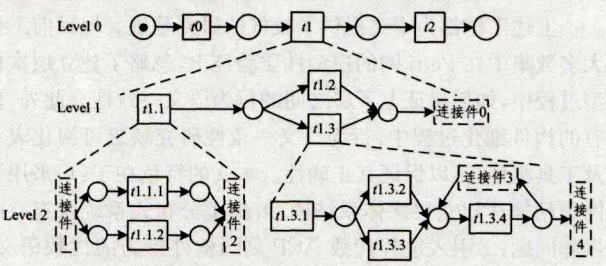

图 6　网上银行体系结构层次模型

需要说明的是，本文在求精操作时定义的子网是单入口、单出口的，因此在求精过程中会加入连接件以保证求精操作的正确性，如图 6 中的连接件 0 和连接件 4。其中连接件 1、连接件 2 和连接件 3 是为了保证并行结构和循环结构的完整性。

为了验证通过行为规约进行求精的分布式系统体系结构模型的正确性，采用 Petri 网的开源工具（Platform Independent Petri Net Editor，PIPE）[13] 对模型进行验证。需要注意的是，一个 Petri 网模型的正确性主要从活性（安全且不死锁）和有界性两方面考虑[8]。如图 7 所示，通过 PIPE 分析证明了该模型是有界且可达的。

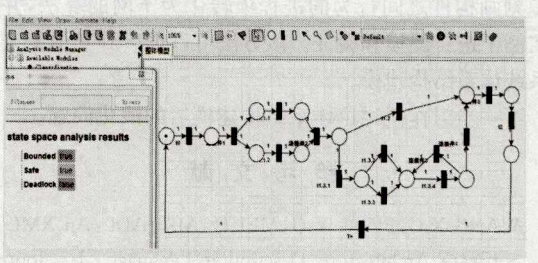

图 7　网上银行的整体体系结构模型

在图 7 中，将图 6 的层次结构模型作为一个整体模型绘制，从而可以验证整个网上银行的体系结构模型是否是正确的。此外，为了准确刻画模型验证结果，在模型的结束库所和开始库所之间添加一个变迁 $t+$，使得模型形成一回路整体，表示可以正常执行回到初始状态。具体讲，如验证结果所示，整个模型是有界的、安全的，并且不存在死锁情况。

6　相关工作

在体系结构建模方面，Petri 网作为一种形式化工具，兼

顾了系统的静态结构及动态执行特征，因此被广泛运用在建模中。文献[14]通过研究 Petri 网的性质来评估体系结构模型，但没有考虑系统复杂性引起的模型状态空间爆炸问题。文献[15]提出通过 Petri 网的可达图来标识构件之间的波及效应，但是对于复杂的分布式系统，其 Petri 网模型的可达图是难以计算的，即状态空间爆炸。

在构件求精过程中，为了保证其求精之后的行为语义与原来保持一致。王建龙等人[7]提出了上下文有关的构件求精方法，但上下文有关文法本身晦涩，难以掌握理解。此外，他们还提出理论缺乏算法实现及工具验证，难以保证其正确性。文献[16]根据分布式系统中消息传递的应用特征提出基于信息网络图和行为状态机的方法来建立构件模型。但是该方法对于构件外部及内部采用了不同的形式建模，无法运用统一工具验证二者的正确性。

上述工作都为本文提供了较好的研究基础。但目前工作大多数集中在 Petri 网的模型性质验证上，忽略了建立层次模型过程中，如何保证上下层之间的行为语义一致性。此外，现有的构件细化过程中，行为语义一致性研究缺乏可视化表示及工具验证，难以保证其正确性。本文的特色在于：1)采用子网求精操作建立层次体系结构，解决了分布式系统状态空间爆炸问题；2)引入进程代数 ACP 刻画构件行为的代数语义，解决了分布式系统求精过程中不同层次之间的行为一致性问题；3)托肯不仅代表控制执行信息，更携带了求精内容，克服了传统 Petri 网只能从控制流角度刻画行为的问题；4)在实现方面给出实例验证，表明了方法的正确性。

结束语 针对目前在分布式系统体系结构建模方面的研究存在的不足之处，本文主要工作如下：首先，为了兼顾建模的准确性及可视化，选用 Petri 网作为形式化工具进行分布式系统的体系结构建模。同时，为了避免状态空间爆炸问题引入子网求精操作，建立模型的层次结构。接着，为了保证构件在求精过程中保持前后行为语义的一致性，引入了进程代数 ACP 刻画构件之间的行为语义。此外，将行为规约作为信息放入托肯中，通过 Petri 网点火规则进行求精操作。最后，给出了刻画构件之间行为的进程项转化为子网的算法，并针对现实例子进行建模和验证。该方法可以推广到其他复杂类型的系统体系结构建模。

下一步工作拟讨论体系结构建模后的演化问题。

参考文献

[1] WANG X G, FENG Y G, MEI H. ABC/ADL: An XML-Based Software Architecture Description Language[J]. Journal of Computer Research and Development, 2004, 41(9): 1521-1531. (in Chinese)
王晓光，冯耀东，梅宏. ABC/ADL：一种基于 XML 的软件体系结构描述语言[J]. 计算机研究与发展，2004，41(9)：1521-1531.

[2] ZHU X Y, TANG Z S. A temporal logic-based software architecture description language XYZ/ADL[J]. Journal of Software, 2003, 14(4): 713-712. (in Chinese)
朱雪阳，唐稚松. 基于时序逻辑的软件体系结构描述语言 XYZ/ADL[J]. 软件学报，2003，14(4)：713-720.

[3] ALLEN R, GARLAN D. A Formal Basis for Architectural Connection[J]. Acm Transactions on Software Engineering & Methodology, 1997, 7(2): 66-90.

[4] LUCKHAM D C, VERA J. An Event-Based Architecture Definition Language[J]. IEEE Transactions on Software Engineering, 1995, 21(9): 717-734.

[5] RONG M, ZHANG G Q. Software Architecture Description Approach Integrating Formal Methods and Visual Methods[J]. Computer Science, 2005, 32(4): 205-208. (in Chinese)
戎玫，张广泉. 形式化与可视化相结合的软件体系结构描述方法研究[J]. 计算机科学，2005，32(4)：205-208.

[6] GARLAN D. Style-Based Refinement for Software Architecture [C]// Proceedings of the Second International Software Architecture Workshop(ISAW-2). 1998: 72-75.

[7] WANG J L, SHEN L M, NIU J C. Context sensitive-based refinement of components[J]. Journal of Computer Applications, 2008, 28(4): 1058-1060. (in Chinese)
王建龙，申利民，牛景春. 基于上下文有关的构件求精方法[J]. 计算机应用，2008，28(4)：1058-1060.

[8] 吴哲辉. Petri 网导论[M]. 北京：机械工业出版社，2006.

[9] BERGSTRA J A, KLOP J W. Process algebra for aynchronous communication[J]. Information and Control, 1984, 60(1/3): 109-137.

[10] DAI F, LI T, XIE Z W, et al. Towards an Algebraic Semantics of Software Evolution Process Models[J]. Journal of Software, 2012, 23(4): 846-863. (in Chinese)
代飞，李彤，谢仲文，等. 一种软件演化过程模型的代数语义[J]. 软件学报，2012，23(4)：846-863.

[11] XIA C L. The Analysis and Application of a PP Type Petri Net's Subnet Refinement[J]. Computer Science, 2006, 33(9): 241-244. (in Chinese)
夏传良. Petri 网的 TT 型子网精细化操作性质分析及其应用[J]. 计算机科学，2006，33(9)：241-244.

[12] LIN L L, DAI F, LI T, et al. Research on State Consistency Based on Hierarchical Petri Nets[J]. Journal of Chinese Computer Systems, 2015, 36(7): 1481-1486. (in Chinese)
林雷蕾，代飞，李彤，等. Petri 网层次结构中的状态一致性研究[J]. 小型微型计算机系统，2015，36(7)：1481-1486.

[13] DINGLE N J, KNOTTENBELT W J, SUTO T. PIPE2: a tool for the performance evaluation of generalised stochastic Petri Nets[J]. Acm Sigmetrics Performance Evaluation Review, 2009, 36(4): 34-39.

[14] LUO A M. Research on Information System Architecture Description Method based on Multi-view[J]. Computer Science, 2010, 37(8): 294-297. (in Chinese)
罗爱民. 基于可执行模型的体系结构验证评估方法[J]. 计算机科学，2010，37(8)：294-297.

[15] YANG H B, QIAO Y M. Component assembly model based on Petri net[J]. Journal of Jiangsu University(Natural Science Edition), 2010, 31(4): 452-457. (in Chinese)
杨鹤标，乔亦民. 基于 Petri 网的构件组装模型[J]. 江苏大学学报(自然科学版)，2010，31(4)：452-457.

[16] ZHAO H, SUN W M, TANG S F. Research on Software Component Refinement[J]. Journal of Chinese Computer Systems, 2010, 31(3): 418-422. (in Chinese)
赵恒，孙为民，唐素芬. 软件构件求精方法研究[J]. 小型微型计算机系统，2010，31(3)：418-422.