基于 π 演算的 BPEL 组合服务的形式化描述及验证

夏红星

（江苏靖江教师进修学校，江苏 泰州 214500）

摘 要：BPEL 是实现 SOA 组合服务和服务编制的重要技术。重点论述了 π 演算的语法定义和 Π 演算建模 Web 服务的算法，然后以一个典型的银行借贷服务系统为例，利用 π 演算进行了形式化描述和验证。

关键词：BPEL 服务；π 演算；形式化描述；验证

中图分类号：TP393.02 **文献标识码**：A **文章编号**：1674-7720(2010)22-0062-03

Formal description and verification of π-calculus based BPEL service composition

XIA Hong Xing

(Jingjiang Teachers' College for Vocational Studies, Taizhou 214500, China)

Abstract: BPEL is an important technology to achieve SOA service composition and service orchestration. This paper mainly discussed the syntax definition of π-calculus and the algorithm using π-calculus to model web service, then took a typical bank loan service system for example, used π-calculus for formal description and verification.

Key words: BPEL service; π-calculus; formal description; verification

在 SOA 系统中，通常根据企业的业务需求，利用 BPEL 技术将各个独立的 Web 服务进行组装[1]。如何判断组合后的服务能够满足企业的需求，如何判断组合服务不会在运行中出现问题，这是软件开发人员交付服务前必须考虑的问题。事实上，人工设计或自动构建的服务组合可能存在死锁、活锁、状态不可达等众多问题，所以必须在部署前进行严格的验证，以及时发现服务组合中存在的问题。

所谓的 Web 组合服务验证就是指在实际部署组合服务系统前，通过某些理论或工具检查该服务组合流程逻辑的合理性、服务间的兼容性，从而及时发现和修正存在的问题，以免日后运行过程中给企业和用户造成损失。本文的重点是验证组合服务的内部流程逻辑[2]。

当前主要是通过基于状态转换模型的 Petri 网理论、自动机理论和基于进程代数模型的 π 演算理论对 Web 组合服务进行验证[1]。采用 Petri 网或者自动机对服务组合进行描述较直观简洁，但是当业务流程复杂、牵扯到的子服务众多且服务间的交互频繁的时候，往往会引起状态空间急剧增加，因此导致验证复杂度剧增。而 π 演算正是表示这种复杂行为的有效方式，可以清楚表示系统的并发交互行为。π 演算采用文本的进程表达式描述系统，表达能力强且形式简洁，加之 π 演算中的行为理论非常适合用来描述动态并发的 Web 服务，因此本文采用 π 演算理论对服务组合进行建模和验证。

1 基于 π 演算的 BPEL 服务组合建模

1.1 π 演算的语法定义

π 演算是由 MILNER R 等人在通信系统演算 CCS 的基础上提出来的描述和分析通信拓扑结构动态变化的计算模型。设 N 为无限名字集，x、y 等小写字母为名字集上的名字；A、B 等表示进程；P、Q 等表示进程表达式，进程表达式包括以下几种形式：

(1)求和：$\sum_{i \in S} P_i$，其中 $i \in S$，S 为索引有限集。

0 表示空进程；$P+Q$ 表示选择执行 P 或者 Q，只执行其中一个。

(2)并发表达式：$P|Q$ 表示并发执行进程 P、Q。

(3)前缀表达式 $y(x).P$、$\bar{y}x.P$、$\tau.P$：

正前缀 $y(x).P$ 表示在端口 y 上输入名字 x，然后执行进程 P。

负前缀 $\bar{y}x.P$ 表示在端口 y 上输出 x 后，再执行进程 P；

$\tau.P$ 称为哑前缀，表示进程外部不可见的动作，执行完进程后，再执行 τP。

(4)循环表达式：$!P$ 表示无穷复制进程 P。

(5)限制表达式：$(x)P$ 表示进程 P 在通道 x 上的外部动作被禁止，但是可以进行在通道 x 上的内部通信。

(6)匹配表达式：$[x=y].P$ 表示当条件 $x=y$ 成立时才执行进程 P。

(7)进程标识符：$A(x,y,\cdots,z)$ 对每个进程来说，必须有其定义 $A(x,y,\cdots,z)::=P$，其中 $x,y,\cdots,z$ 表示进程 P 中的自由名。

至此，将 π 演算定义如下：

$$P::=0|P+Q|(P|Q)|y(x).P|\bar{y}x.P|\tau.P|!P|(x)P|[x=y]P|A(x,y,\cdots,z)$$

1.2 BPEL 描述的 Web 组合服务的 π 演算建模

1.2.1 建模算法

为了利用 π 演算验证 BPEL 服务组合，首先要将 Web 服务的逻辑关系映射为 π 演算的表达式，形式化描述 Web 服务组合，这个过程就是建模，算法描述如下：

(1)将 BPEL 服务组合中的每个 Web 服务看作一个 π 演算的进程。如果该服务本身又是一个组合服务那么将其递归细分为一系列的子服务，然后将每个子服务看作一个 π 演算进程。

(2)服务之间的调用关系抽象成 π 演算进程之间的通道上的消息交互。两个相关进程之间至少有一条通道，如果两进程间有多条顺序消息，则抽象成在一条通道上传递。

(3)根据 Web 组合服务业务流程图，按照下表所示的 Web 服务元素与 π 演算的元素对应关系，将业务流程图转换成 π 演算流图。对应关系如表 1 所示。

表 1 BPEL 中 Web 服务与 π 演算元素对应关系

Service	process
Operation	Action
PortType	Sort
Port	Port
Binding	Interaction
Message	message
Types	Sort

(4)根据上一步骤产生的 π 演算流图，写出建模表达式，将 Web 服务及其组合用 π 演算形式化描述出来。

1.2.2 建模实例

下面以基于 BPEL 的信贷服务系统（由客户、银行信贷受理服务、客户信用评估服务、信贷审批服务组成）为例，进行组合服务的 π 演算建模。

信贷服务系统的流程是：首先客户向银行信贷受理部门提出贷款请求，银行收到客户的贷款请求后，询问客户的具体贷款信息（比如客户姓名、贷款数额等），客户将这些信息反馈给银行信贷部门；银行信贷部门将客户贷款信息提交给客户信用评价部门，要求对该客户的信用状况进行评价，然后信用评价部门将评价信息反馈给信贷受理部门；信贷受理部门将客户申请信息和信用评价信息汇总初审。如果初审不通过，则直接拒绝客户的借贷活动；如果初审通过，则通知客户借贷请求已受理；然后将所有信息提交给审批部门，审批部门将审批结果反馈给信贷受理部门；信贷受理部门最后将审批结果反馈给客户，通知是否可以对客户发放贷款。具体业务流程如图 1 所示。

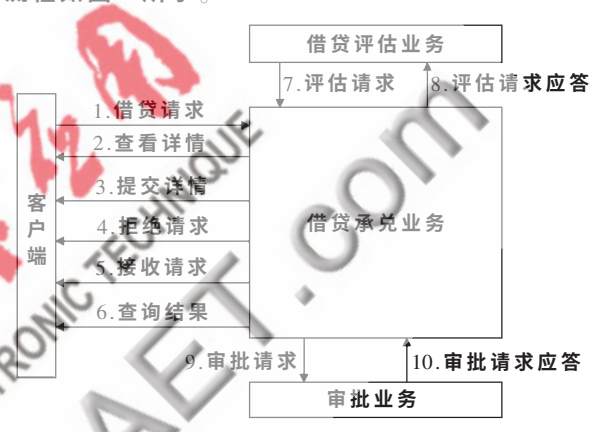

图 1 借贷系统业务流程图

根据建模算法，将业务流程图抽象出如下的 π 演算流图，如图 2 所示。

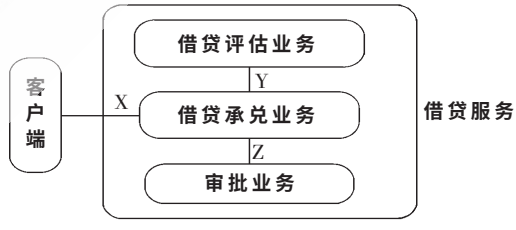

图 2 借贷系统 π 演算流图

根据 π 演算流图，写出借贷系统服务的建模表达式如下：

设客户－Client、信贷受理服务(Client Accepting Service)－CAS、信用评估服务 Credit Evaluation Service－CES、审批服务－Approving Service－AS。

客户服务 Client 在 X 通道上与信贷受理服务通信：

设 $\bar{u}=\{X,ReqLoan,AskDetails,ProvideDetails,RefuseReq,AcceptReq,InformResult\}$，则：

$$Client(\bar{u})=\bar{X}<ReqLoan>.X(msg).([msg=AskDetails](\bar{X}(ProvideDetails).X(msg1).([msg1=RefuseReq]Client(\bar{u})+[msg1=AcceptReq].X(msg2).[msg2=InformResult]Client(\bar{u}))))$$

信贷受理服务 CAS 在 Y 通道上与信用评估服务

通信：

设 $\bar{b}$={X,Y,Z,ReqLoan,AskDetails,ProvideDetails,RefuseReq,AcceptReq,InformResult,ReqEvaluation, ReplyEvaluation, ReqApprove,ReplyApprove}，则：

CAS($\bar{b}$)=X(msg).[msg=ReqLoan]$\overline{X}$<AskDetails>.X(msg1).[msg1=ProvideDetails]$\overline{Y}$<ReqEvaluation>.Y(msg2).[msg2=ReplyEvaluation]($\overline{X}$<RefuseReq>CAS($\bar{b}$)+$\overline{X}$<AcceptReq>.$\overline{Z}$<ReqApprove>.Z(msg3).[msg3=ReplyApprove]$\overline{X}$<InformResult>.CAS($\bar{b}$))

信用评估服务 CES 在通道 Y 上与信贷服务通信：

设 $\bar{c}$={Y,ReqEvaluation, ReplyEvaluation}，则：

CES($\bar{c}$)=Y(msg).[msg=ReqEvaluation]$\overline{Y}$<ReplyEvaluation>.CES($\bar{c}$)

审批服务 AS 在通道 Z 上与信贷服务通信：

设 $\bar{a}$={Z,ReqApprove, ReplyApprove}，则：

AS ($\bar{a}$)=Z (msg).[msg=ReqApprove].$\overline{Z}$<ReplyApprove>.AS($\bar{a}$)

整个信贷服务系统 LoanService 由信贷受理服务(Client Accepting Service)CAS、信用评估服务(Credit Evaluation Service)CES、审批服务(Approving Service)AS 组合而成。{Y、Z}属于 LoanService 的内部通道，作为受限名字出现，于是信贷服务系统的定义如下：

LoanService (X,ReqLoan,AskDetails,ProvideDetails,RefuseReq,AcceptReq,InformResult)=(Y、Z)(CAS($\bar{b}$)|CES($\bar{c}$)|AS($\bar{a}$))。

2 基于 π 演算的 BPEL 组合服务的验证

2.1 Web 服务组合内部逻辑验证的内容

Web 服务组合内部流程逻辑的验证主要包括流程的可达性验证、流程的正确完成性验证、流程的活锁验证、死锁验证、观察等价性验证等，本文只验证观察等价性。验证观察等价性也就是验证两个进程是否是弱互模拟的。弱互模拟的定义：如果进程 P、Q 的外部行为是一致的，即模拟进程和被模拟进程在外部观察者看来具备完全相同的行为能力，其中一个进程能执行的动作另一个进程也能模拟，则称进程 P、Q 弱互模拟。

2.2 观察等价性和死锁的验证

本文的验证除了手工推演，还借助了自动化 π 演算工具 MWB。MWB 采用基于 New Jersey SML 语言编译器，适用于操作和分析动态并发系统。

理论推演：在前面建模时，将 Client 建模为 π 演算进程，这样做是出于下面验证观察等价性的需要。根据 π 演算相关理论和参考文献[3]提出的反转证明法，要判断系统中 π 演算描述的 Client 的逆进程 ReverseClient 与 LoanService 是否观察等价，只需要判断 ReverseClient 与 LoanService 是否是弱互模拟的。

Client 的逆进程为：

设 $\bar{u}$={X,ReqLoan,AskDetails,ProvideDetails,RefuseReq,AcceptReq,InformResult}，则：

ReverseClient($\bar{u}$)=X(msg).[msg=ReqLoan]$\overline{X}$<AskDetails>.X(msg1).[msg1=ProvideDetails]($\overline{X}$<RefuseReq>.ReverseClient($\bar{u}$)+$\overline{X}$<AcceptReq>.$\overline{X}$<InformResult>.ReverseClient($\bar{u}$))

在组合服务 LoanService 中，Y、Z 是服务内部的私有通道，在这两个通道上的动作集{ReqEvaluation, ReplyEvaluation,ReqApprove,ReplyApprove}是对外不可见的 动作。

记消除 τ 动作后的服务组合 LoanService 为 LoanServiceOut，则：

LoanServiceOut(X,ReqLoan,AskDetails,ProvideDetails,RefuseReq,AcceptReq,InformResult) $\overset{(op+)}{=}$ X(msg).[msg=ReqLoan]$\overline{X}$<AskDetails>.X(msg1).[msg1=ProvideDetails]($\overline{X}$<RefuseReq>.0+$\overline{X}$<AcceptReq>.$\overline{X}$<InformResult>.0)，其中操作符 op+表示对系统进行一个或多个消除 τ 动作运算。

对比可见，ReverseClient 与 LoanServiceOut 的外部可观察动作集完全一致，即 ReverseClient 与 LoanServiceOut 弱互模拟，所以 ReverseClient 与 LoanServiceOut 观察等价，又 LoanServiceOut 与 LoanService 观察等价，从而 ReverseClient 与 LoanService 是弱互模拟的，即观察等价性成立。

利用 MWB 验证工具证明：

在 MWB 目录下创建 LoanService.ag 文件，将 Client、CAS、CES、AS、ReverseClient、LoanService 的进程表达式写入文件中。然后在命令行输入"weq ReverseClient LoanService"，运行结果提示两个进程等价，证毕。

本文简单介绍了 π 演算的语法定义，给出了 π 演算理论形式化描述 BPEL 服务组合的建模算法，最后结合一个银行信贷系统的服务组合实例，进行建模和观察等价性验证。π 演算是进行 Web 服务建模验证的有效理论工具，相信利用 MWB 等工具对进程表达式进行自动求逆和验证是未来的研究热点。

参考文献

[1] MILNER R.Communicating and mobile systems:the π-calculus[M].Cambridge University Press,1999.

[2] DENG Shui Guang.Research on automatic service composition and formal verication.Zhejiang University,2007.

(收稿日期：2010-08-04)

作者简介：

夏红星，男，1977 年生，讲师，主要研究方向：计算机应用。