第33卷 第19期 计 算 机 工 程 2007年10月
Vol.33 No.19 Computer Engineering October 2007
·博士论文· 文章编号：1000—3428(2007)19—0019—04 文献标识码：A 中图分类号：TP311

基于B方法的Web Service组合描述和验证方法

王帅强，万建成，侯金奎，冯仕红
(山东大学计算机科学与技术学院，济南 250061)

摘 要：对于Web Service及其组合来说，保证其组合的正确性是十分必要的。B方法是一个基于模型的形式化方法，有很强的结构化机制和很好的工具支持，对于建模和软件验证是一种有效的方法。该文基于B方法对Web服务及其组合进行了形式化建模，并能够利用B方法相对成熟和完善的模型检查工具，来完成模型的正确性验证。

关键词：Web Service；B方法；服务形式化；服务组合

Approach of Describing and Verifying Web Service Composition Using B-method

WANG Shuai-qiang, WAN Jian-cheng, HOU Jin-kui, FENG Shi-hong
(School of Computer Science and Technology, Shandong University, Jinan 250061)

【Abstract】It is necessary to guarantee the validity of Web services and their composition. Since B-method is a model-based formal method, which has strong structuring mechanisms and good tool support, it is an effective method for modeling and verifying. Therefore, this paper models for Web Services and their composition and verifys the B model with the help of the tools of B.

【Key words】Web Service; B-method; formal service; service composition

1 概述

近年来，Web服务的理论和技术取得了长足的发展。Web服务采用以XML为规范的Web服务描述语言(WSDL)来描述，从而可以用真正与平台无关的方式来描述任何(所有)数据，以跨系统交换数据，因此在互操作方面实现了调用远程网络的应用程序和远程网络之间的松耦合。而且，Web服务可以在较抽象的层面上工作，较抽象层面可以按照需要动态地重新评估、修改或处理数据类型。所以，从技术层面上讲，Web服务技术促进了服务的重用。

但是，如何使得Web服务实现跨组织、跨管理域的系统集成和自动交互，还面临着很多问题。其中一些问题是在传统的中间件应用中已经解决了的，而另外一些则是新问题，比如Web服务如何组合、能否自动组合以及组合的正确性验证等问题。对于解决上述问题，学术界和工业界提出的解决方法主要有两类：一类是利用工作流的方法，例如基于WSDL的、同样以XML为规范的Web服务的业务流程执行语言(BPEL4WL)，它的建模和验证常采用工作流和进程代数等形式化方法；另一类是人工智能的思想，如语义Web，常基于时序逻辑的方法。

Foster, Magee等人利用LTSA工具把BPEL4WS文档转换成FSP进程，并利用LTSA工具来对FSP进行模型验证，但是LTSA工具仅支持模型检查。Koshkina和van Breugel引入基于BPEL4WS的BPE演算，使用PAC工具和CWB-NC工具来建模和验证。此外，基于Petri-Net的Web服务组合建模方法也在广泛开展。

20世纪80年代初有关规格说明语言Z的研究形成了B方法[1]的背景。B方法可以使程序和程序规格说明处于一个统一的数学框架下，以一种基于集合论的符号表示法来书写。长期以来，B方法大都应用在安全攸关的领域，因此其证明检查机制相对严格和完善。B有两种商用的证明工具集，Atelier-B[2]和B-Toolkit[3]，而南安普顿大学也开发了一个用于科研的模型检查工具Prob[2~4]。

由此，本文提出一种建模和验证方法，利用B较为成熟和严格的验证证明机制，将基于WSDL的Web服务和基于BPEL4WS的业务流程采用B方法建模，并且采用B的模型检查工具对其进行正确性验证。

2 B方法

2.1 抽象机

抽象机是构成B方法系统的基本单位，类似于UML模型中的类、抽象数据类型、模块、包等，其形式化定义如下：

定义1 抽象机

S 和 T 是抽象机中用到的集合，c 是抽象机内部的常量，v 是变量，U 是抽象机初始化的初始代换，$u \leftarrow O(w)$是一个返回值为 u、参数为 w 的操作，V 是一个或者一系列代换，C, P, I, J, Q, Hc, Hr 是一些谓词，则抽象机包含 X 和 x 两个参数的抽象机 M 的形式定义如清单1所示。

清单1 B抽象机语法

```
MACHINE   M (X, x)
CONSTRAINTS
    C
SETS
    S；T = {a, b}
(ABSTRACT_)CONSTANTS
```

基金项目：山东省科技发展项目(20051014)

作者简介：王帅强(1981－)，男，博士研究生，主研方向：形式化方法，Web Service；万建成，教授、博士生导师；侯金奎、冯仕红，博士研究生

收稿日期：2006-10-08 **E-mail：**wangshuaiqiang@mail.sdu.edu.cn

c

PROPERTIES

P

(CONCRETE_)VARIABLES

v

INVARIANT

I

ASSERTIONS

J

INITIALIZATION

U

OPERATIONS

$u \leftarrow O(w)$ =PRE Q THEN V END;

…

END

2.2 精化

精化是将软件系统的抽象模型(规范)变换到较为具体的模型(精化结果)的一种技术。精化模型的具体性表现在：(1)它可能包含了更多的细节；(2)它可能更接近于实现。具体说来，它不但减少了非确定性，弱化了前条件，还有可能使得变量的空间发生改变。其形式定义如下：

定义 2 代换的精化

假设 S 和 T 为两个作用在集合 s 上的代换，那么，

$$\frac{S}{T} \Leftrightarrow \forall a \cdot (a \subseteq s \Rightarrow str(S)(a) \subseteq str(T)(a))$$

表示代换 T 是代换 S 的精化。

定义 3 抽象机的精化

假设 M 和 N 是两个抽象机，并且它们的变量分别是集合 b 和 c 的成员，则

$$\frac{M}{N} \Leftrightarrow \frac{@y \cdot (y \in b \Rightarrow T)}{@z \cdot (z \in c \Rightarrow U)}$$

表示抽象机 N 是抽象机 M 的精化。

如图 1 所示，一个抽象机经过一步或多步的精化操作，最终得到一个实现(Implementation)，从而可在计算机上执行。

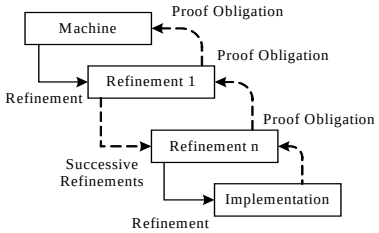

图 1　B 方法的精化步骤

2.3 证明义务

B 方法的证明义务保证了其严格的正确性。它包含两部分，即抽象机的证明义务和精化的证明义务。抽象机的证明义务保证了一个抽象机的正确性，而对于抽象机的每一步精化而言，它又保证了精化结果是原抽象机的具体化，从而保证了最终得到的代码的正确性。B 方法的证明义务的严格性使得它在安全攸关的领域得到了广泛的应用。

定义 4 抽象机的证明义务

首先定义如表 1 所示的简写。

表 1　简写

简写	定义
A	$X \in POW_1(INT)$
B	$S \in POW_1(INT) \wedge T \in POW_1(INT) \wedge T = \{a, b\} \wedge a \neq b$

那么，抽象机的证明义务为

$A \wedge B \wedge C \wedge P \Rightarrow [U]I$

$A \wedge B \wedge C \wedge P \wedge I \Rightarrow J$

$A \wedge B \wedge C \wedge P \wedge I \wedge J \wedge Q \Rightarrow [V]I$

其他符号见定义 1 所示。

3　用 B 方法对 Web 服务组合的建模

目前在工业界，Web Service 的描述语言是基于流程的，其协议栈如图 2 所示。在网络层之上是基于 XML 的消息传递，它表示使用 XML 作为消息传递协议的基础；WSDL 定义了服务交互的接口和结构 通过 UDDI 协议发现 Web 服务，并且加以组合，使之符合 BPEL4WS 的规范。

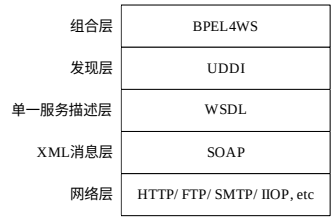

图 2　Web Service 的协议栈

由此，必须将 WSDL 描述的单一服务以及 BPEL4WS 描述的服务组合转化为 B 抽象机，从而能够利用 B 方法的验证机制来验证服务组合的正确性。

3.1 从 WSDL 到 B 抽象机的转换

一个 WSDL 文档通常由数据类型定义(types)、消息定义(message)、接口类型定义(port type)、绑定(binding)以及服务(service)组成。由于 bpel4ws 只是在抽象层次上对接口类型(port type)的引用，因此在这里的 wsdl 文档不包括 binding 和 service。

3.1.1　WSDL 文档对应的 B 抽象机——DTM & PTM

每一个接口类型都定义了一系列的操作(operation)，表示一类相对独立的服务功能，因此作为一个单独的 B 抽象机来重点描述功能，称之为接口类型抽象机(port type machine, PTM)；而对于 WSDL 定义的复杂数据类型以及消息格式(在这里也看作数据类型)，可看作是被服务接口抽象机使用的抽象机，它重点描述数据类型，称之为数据类型抽象机(data type machine, DTM)，其名字为 message。这样，一个 WSDL 描述事实上被拆为两类、多个 B 抽象机，单独的数据类型抽象机定义了一些作为数据类型的集合和函数映射关系，为服务接口抽象机的进一步描述奠定了基础；而一个或多个服务接口抽象机均使用数据类型抽象机，在它定义的集合和函数映射关系的基础上完成对服务的完整描述。

3.1.2　WSDL 和 B 抽象机的映射关系

在数据类型抽象机中，主要定义了 SETS, DEFINATIONS (实际上也是对集合的定义)，VARIABLES 和它们之间要满足的不变式。集合主要由简单、复杂数据类型，数据类型的元素类型以及消息元素的数据类型构成。

接口类型抽象机必须为 BPEL4WS 引用，因此这些抽象机的名字为相应的 BPEL4WS 中使用的接口链接的类型(在 WSDL 文档中相应的 <partnerLinkType> 元素中定义，BPEL4WS 文档的引用参见第 3.2 节)。每一个接口类型抽象机都使用数据类型抽象机。因为在 BPEL4WS 文档转化的抽象机中，包含数据类型抽象机和接口类型抽象机，为了维持接口类型抽象机和数据类型抽象机“粘接”起来的不变式，在接口类型抽象机中应该使用，而不是包含数据类型抽象机。

WSDL 和 B 抽象机的对应关系如表 2 所示。

表 2　WSDL 和 B 抽象机的映射关系

	WSDL	B-Machine
Data Type Machine		
1	Types & Messages	MACHINE message
2	Types	elements of SETS
3	Complex Type & Simple Type	element of SETS
4	element of Complex Type or Simple Type	element of SETS & DEFINATIONS
5	message-name	element of SETS & VARIABLES & INVARIANTS
6	message-part-name	element of VARIABLES
7	message-part-type	element of SETS & INVARIANTS
Port Type Machine		
8	Port Type	MACHINE(USES message)
9	operation	element of OPERATIONS
10	operation-input	arguments & pre-condition
11	operation-output	return value
12	Partner Link Type	the name of corresponding portType machine

(1)数据类型抽象机 DTM

如表 2 所示，前 7 条规则是关于数据类型抽象机的。将每一个复杂或者简单数据类型都建立一个名字为其本身名字的集合(规则 3)；将每一个复杂或者简单数据类型的元素都建立一个名字为其本身的名字的集合，和规则 3 建立的集合不同，数据类型元素的集合是函数映射的形式，映射的源是这些元素隶属于的规则 3 建立的复杂或简单数据类型集合，映射的像是这些元素，映射关系定义为全函数(规则 4)；对每个消息 message，都要根据其名字构造一个集合和一个变量，并且构造相应的变量满足的不变式。构造不变式说明这个变量是同它一起构造的集合的子集，形式为 variable : POW(SET)(规则 5)；对构成消息的每个元素，都建立一个变量，其名字和这个元素的名字相同(规则 6)；对消息的每一个元素的数据类型，如果不是 B 的基本数据类型，并且前面定义的集合里没有包含这些元素的数据类型，则需要在集合里加上相应的声明；而不变式则是为了说明规则 6 中构造的变量的数据类型，其形式为一个函数映射。它的源是规则 5 中构造的相应的变量，像为集合部分中相应元素的集合，形式为 v : variable --> SET，映射关系定义为全函数(规则 7)。如清单 2 所示的 WSDL 片段，转化为清单 3 所示的 B 抽象机。

(2)接口类型抽象机 PTM

清单 2 WSDL 文档片断

```
<types>
        …
    <sns:complexType name="CustomerInfo">
      <sns:sequence>
        <sns:element name="OriginFrom" type="sns:string" />
        <sns:element name="Birthday" type="sns:date" />
      </sns:sequence>
    </sns:complexType>
</types>
        …
<message name="POMessage">
<part name="customerInfo"
type="sns: CustomerInfo "/>
<part name="purchaseOrder"
      type="sns:purchaseOrder"/>
</message>
        …
```

清单 3 B 抽象机片断

```
SETS
CustomerInfo, PurchaseOrder, POMessage …
DEFINITIONS
OriginFrom = CustomerInfo --> string;
Birthday = CustomerInfo --> date;
            …
VARIABLES
pomessage, customerInfo, purchaseOrder, …
INVARIANT
pomessage <: POMessage &
customerInfo : pomessage --> CustomeInfo &
purchaseOrder : pomessage --> PurchaseOrder
            …
```

如表 2 所示，后 5 条规则是关于接口类型抽象机的。每一个 Port Type 都构成一个抽象机，并且这些抽象机都使用抽象机 message；抽象机的名称不是 Port Type 的名称，而是为了和 BPEL4WS 对应起来，应该到下面的< partnerLinkType> 元素组里寻找对应的链接类型，并将其 name 作为抽象机的名称(规则 8, 12)；将 WSDL 文档的每一个操作都转化为 B 抽象机的对应的操作；以 WSDL 文档中每一个操作的输入的消息对应的变量作为相应的 B 抽象机操作的参数，并将变量的数据类型作为前条件作为操作必须满足的前条件；将 WSDL 文档的每个操作的输出消息对应的变量作为相应的 B 抽象机操作的返回值(规则 9~规则 11)。例如，关于接口类型定义的 WSDL 片断如清单 4 所示，转换的 B 接口类型抽象机如清单 5 所示。

清单 4 WSDL 文档片断

```
<portType name="purchaseOrderPT">
<operation name="sendPurchaseOrder">
<input message="pos:POMessage"/>
<output message="pos:InvMessage"/>
</operation>
   …
</portType>
        …
<plnk:partnerLinkType name="purchasingLT">
<plnk:role name="purchaseService">
<plnk:portType name="pos:purchaseOrderPT"/>
</plnk:role>
</plnk:partnerLinkType>
        …
```

清单 5 B 抽象机片断

```
MACHINE  purchasingLT
USES Message
        …
OPERATIONS
        invm <-- purchaseOrderPT(pom)
        PRE   pom : pomessage
        THEN        inv := inv \/{invm}
        END;
END
```

3.2　BPEL4WS 和 B 抽象机的映射关系

BPEL4WS 指定了基于 Web 服务的业务流程行为，使用合作伙伴(在 BPEL4WS 文档的<partnerLinks>元素中定义)的交互方式，定义了 BPEL4WS 的流程，因此只需要一个 B 抽象机描述即可，称之为业务过程抽象机(business process

machine, BPM)。

BPEL4WS 和 B 抽象机的对应关系如表 3 所示。业务过程抽象机的名字由业务过程的名字决定，并且它包含 BPEL4WS 文档中在<partnerLinks>元素中链接的所有由 WSDL 文档定义的接口类型抽象机，并将它重命名，新名字为<partnerLink>的 name 属性的值，还包含 WSDL 定义的数据类型抽象机 message(规则 2)；所有操作序列均作为 B 抽象机中 OPERATIONS 部分的唯一的一个名字为 Main 的无参数无返回值的操作(规则 4)。Main 操作没有参数，也没有返回值，只叙述 Web 服务的过程逻辑。BPEL4WS 主要有 5 种结构化，即 sequence, flow, pick, switch 和 while。对应于这 5 种活动，B 抽象机均有相应的符号表示，如表 3 所示，并且有相应的证明义务来保证过程序列的正确执行。例如，BPEL4WS 片断如清单 6 所示，转化的 B 抽象机片断如清单 7 所示。

表 3　BPEL4WS 和 B 抽象机的映射关系

	BPEL4WS	B- Machine
1	Process	MACHINE process
2	Partner Link	EXTENDS
3	Variables	VARIABLES
4	operations sequence	OPERATIONS
5	Sequence	Each action step separated by “;”
6	Flow	Each action step separated by “‖”
7	Switch	Case clause
8	Pick	Choice clause
9	While	While clause
10	Receive & Invoke & Reply	action step
11	Assign	assign operation

清单 6 BPEL4WS 文档片断

```
<process name="purchaseOrderProcess" …>
      …
  <partnerLinks>
      <partnerLink name="purchasing"
      partnerLinkType="lns:purchasingLT"
      myRole="purchaseService"/>
  </partnerLinks>
      …
  <variables>
    <variable name="PO" messageType="lns:POMessage"/>
  </variables>
      …
  <sequence>
     <receive partnerLink="purchasing"
  portType="lns:purchaseOrderPT"
  operation="sendPurchaseOrder"
  variable="PO">
</receive>
<flow>
<sequence>
<assign>
<copy>
<from variable="PO" part="customerInfo"/>
<to variable="shippingRequest"
  part="customerInfo"/>
</copy>
  </assign>
      …
</sequence>
<sequence>
<invoke partnerLink="invoicing"
portType="lns:computePricePT"
operation="initiatePriceCalculation"
inputVariable="PO">
</invoke>
…
  </sequence>
…
  </flow>
<reply partnerLink="purchasing"
portType="lns:purchaseOrderPT"
operation="sendPurchaseOrder"
variable="Invoice"/>
</sequence>
</process>
```

清单 7 B 抽象机片断

```
MACHINE Process
EXTENDS  purchasing.purchasingLT …
VARIABLES
  po …
INVARIANT
  po : POMessage …
OPERATIONS
Main =
BEGIN
Invoice:=purchasing.sendPurchaseOrder(po) ;
shippingRequest.customerInfo:=po.customerInfo
||invoicing.initiatePriceCalculation(po);
Purchasing.sendPurchaseOrder(Invoice);
END
END
```

4　B 抽象机模型检查和精化

现在有两个 B 方法的商业证明工具，Atelier-B 和 B-Toolkit。这里主要使用南安普顿大学开发的 ProB 来完成检查工作。

ProB可以通过计算每个操作的前条件从而查找出一系列可以执行的操作，通过执行操作来计算状态的改变，并且计算出每个状态是否满足不变式的要求，从而实现对B模型的检查，并给出状态图以便确认和查阅。不但支持模型检查和验证，而且支持抽象机的精化检查；为了支持大规模的开发过程，ProB还可以对描述进行分解并对分解的子结构进行并发精化，且对每一步的精化操作通过“迹”的概念来保持与原抽象机的一致性[4]，见精化的概念(定义 3)。

在 ProB 中检查得到的 BPM，得到的状态图如图 3 所示。此状态图说明，笔者的服务组合通过了 ProB 的检查，是正确的。

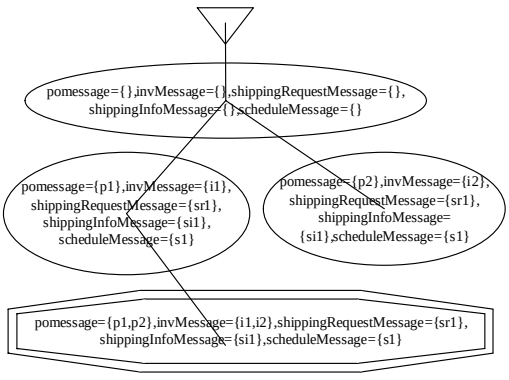

图 3　Web 服务组合的 B 方法模型状态

（下转第 41 页）