doi:10.3969/j.issn.1000-436x.2012.z2.001

基于 Alloy 的服务组合验证

曹玖新[1,2]，吴江林[1,2]，王国进[3]，刘波[1,2]，杨鹏伟[1,2]，董丹[1,2]

(1. 东南大学 计算机科学与工程学院，江苏 南京 210096；
2. 东南大学 计算机网络与信息集成教育部重点实验室，江苏 南京 210096；3. 华为技术有限公司，江苏 南京 210012）

摘 要：服务组合是服务计算的核心问题，而服务组合的正确性与可算性则是服务正确执行的前提保证。首先提出一种基于 Alloy 的服务组合验证方法，采用有限的状态机建模 WS-BPEL 业务流程的状态变迁，利用 Alloy 语言对待验证的属性进行描述，通过 Alloy 模型完成有限状态机的形式化，最后使用 Alloy Analyzer 分析组合服务是否满足验证属性要求。实验研究表明，所提出的基于 Alloy 的服务组合验证方法具有较好的可行性。

关键词：Web 服务；服务组合；有限状态机；Alloy；验证

中图分类号：TP311 **文献标识码：**A **文章编号：**1000-436X(2012)Z2-0001-08

Alloy-based verification of Web service composition

CAO Jiu-xin[1,2], WU Jiang-lin[1,2], WANG Guo-jin[3], LIU Bo[1,2], YANG Peng-wei[1,2], DONG Dan[1,2]

(1. School of Computer Science and Engineering, Southeast University, Nanjing 210096, China;
2. Key Laboratory of Computer Network and Information Integration, Ministry of Education, Southeast University, Nanjing 210096, China;
3. Huawei Technologies Co., Nanjing 210012, China)

Abstract: Service composition was the core problem of service computing, the validity and reliability of service composition had become the premise of service execution. A method was presented which utilizes the finite state machine (FSM) to model the business process's state transitions, and described the required properties with Alloy language. Then, Alloy model was used to formalize the service FSM and the required properties of the system. Finally, Alloy Analyzer was used to verify the model that whether the required properties were satisfied. It is shown that the method of Alloy-based verification of the service composition is of good feasibility.

Key words: Web service; service composition; finite state machine; Alloy; verification

1 引言

Web 服务作为实现面向服务软件开发方法的核心技术得到了众多标准和规范的支持[1]。不同的标准和语言被用于 Web 服务组合，WS-BPEL (Web services business process execution language)用于描

收稿日期：2012-10-23

基金项目：国家自然科学基金资助项目(61070161, 61070158, 61003257, 90912002)；国家重点基础研究发展计划（“973”计划）基金资助项目（2010CB328104）；国家科技支撑计划课题基金资助项目（2010BAI88B03）；教育部博士点基金课题基金资助项目（200802860031）；江苏省自然科学基金资助项目（BK2008030）；国家科技重大专项课题基金资助项目（2009ZX03004-004-04）；江苏省“网络与信息安全”重点实验室基金资助项目（BM2003201）

Foundation Items: The National Natural Science Foundation of China (61070161, 61070158, 61003257, 90912002); The National Basic Research Program of China (973 Program) (2010CB328104); The National Key Technology R&D Program(2010BAI88B03); The Ph.D. Programs Foundation of Ministry of Education of China(200802860031); The Natural Science Foundation of Jiangsu Province (BK2008030); The National Science and Technology Major Project of China(2009ZX03004-004-04); The Program of Jiangsu Province Key Laboratory of Network and Infermation Security (BM2003201)

述服务组合业务流程，许多企业的产品已经实现了WS-BPEL的执行引擎，而且已经存在很多实例使用基于WS-BPEL的应用，因而WS-BPEL作为服务组合语言已经成为事实上的工业标准。但是WS-BPEL缺乏服务组合的正确性、有效性验证机制组合而成的服务系统，可能存在执行失败的风险，从而产生与预期相违背的结果。因此，需要采用形式化方法对WS-BPEL业务流程进行形式化建模与分析，以检验组合服务流程是否存在死锁、各个服务节点的可达性等属性。这样在业务流程执行之前就可以发现错误和漏洞，规避组合服务执行中的失败风险。

本文提出了一种形式化建模和分析方法，采用Alloy语言对基于WS-BPEL的组合服务流程进行建模，并在所建模型的基础上验证业务流程是否满足属性，最后使用Alloy Analyzer分析该模型对应业务流程的结构化属性。

2 相关工作

随着服务计算领域的迅速发展，业务流程的验证已经是面向服务计算领域一个重要的研究方向。目前国内外已经有相关的研究工作，主要有3种方法用于服务组合验证：基于Petri网、基于自动机和进程代数的方法。文献[2]使用了有色Petri网(colored Petri nets)的方法对服务组合进行建模、分析和验证，提出了WS-BPEL到CPN模型的映射机制。这样就可以使用CPN tools对该网系统的可达树进行分析验证；Deepak等在文献[3]中使用FSM(finite state machines)模型开发出工具HUMSAT生成业务流程的可执行代码，并验证可能产生的不可达、死锁、临时不一致等缺陷；文献[4] 首先将WS-BPEL语言转化为SPIN检验工具的输入—Promela语言，并使用模型检验工具SPIN来分析该模型的可达性、安全性、死锁和活性等属性；在文献[5]和文献[6]中，首先用消息序列图(message sequence chart)对应用进行设计建模，并使用LTSA (labeled transition system analyzer)将每个建模的场景转换成有限状态过程(finite state process)，另外使用WS-BPEL服务业务流程，并将服务业务流程转换成有限状态的变迁，最后根据2个有限状态过程验证是否出现对方没有的变迁序列，并可以验证每个有限状态过程的可达性和死锁等属性。这种方法将需求和具体实现的服务流程属性进行对比，发现不同的变迁序列能很好地发现异常的状态，从而可以不断修改实现的服务流程或消息序列图以达到需要的服务流程，这种迭代的方法也符合软件开发的过程，但需要2次转换成有限状态过程，2次验证变迁序列，并且目前只能手工验证；文献[7]采用通信顺序进程(CSP, communicating sequential processes)模型描述Web服务之间的竞争关系，利用CSP的跟踪语义(trace semantics)解决死锁问题，并使用CSP模型检测工具FDR验证模型的正确性；Zhao等在文献[8]中利用LOTOS建模语言及其支撑工具集CADP将BPEL映射到LOTOS进行验证；文献[9]使用模型检验方法，利用Boger模型检验工具，WS-CoL和LTL对WS-BPEL进行建模验证。

由上述现状分析可见，基于状态机的方法是近期使用最多的方法之一，尤其是随着模型检验理论的发展和完善以及大量模型检验工具的支持，这种方法得到更广泛的运用。基于状态机方法的最重要优点是能够利用工具自动化地分析系统模型，给出分析结果，同时，根据模型检验理论还可以给出错误模型的反例，这个特点在分析系统模型中具有重要作用。SPIN、Boger、SMV、NuSMV、BLAST等工具的支持也很好地推动了模型检验的发展。目前服务组合正确性验证的主流思想是在形式化模型的基础上，采用验证工具对Web服务组合进行静态验证。其中，模型的选择对于验证的性能至关重要。

基于此，本文提出使用Alloy语言对WS-BPEL业务流程建模；一个好的模型最重要的就是抽象：抓住系统最有价值、最重要、存在最大风险的方面，而忽略其他不重要的方面。使用Alloy描述系统模型具有以下几个特点。

1) Alloy模型是微模型。使用Alloy建模语言描述的模型一般比实际系统要小几个数量级。

2) Alloy模型是可分析的。使用Alloy语言描述的模型可以使用一个自动化的工具来模拟和检验系统属性，模拟不需要提供输入样例或测试用例。检验可以产生反例，通过反例可以进一步检查模型出错的地方。

3) Alloy模型是说明性的。一个说明性的模型通过列出属性和约束来描述一个系统的状态和行为。与一个操作性的程序不同，它不是通过如何构建状态以及状态之间如何执行转换的细节。这个特点使得Alloy非常适合用来进行增量式的建模，而且不需要另外一种语言来描述需要验证的属性。

4) Alloy使用的模型是结构化的。Alloy语言是

用来描述、分析系统结构化复杂度。

上面 4 个特点中的任何一个特点都不是 Alloy 所特有的，但是，Alloy 综合了上面的 4 个特点，特别是说明性和可分析这 2 个特点的结合使得 Alloy 非常适合用于软件系统的分析。

3　WS-BPEL 和 Alloy

3.1　WS-BPEL 业务流程执行语言

WS-BPEL 是用于 Web 服务的组合、编排以及协作，以实现业务流程自动化的语言规范。WS-BPEL 基于 XML 语言的规范，它提供了一系列具有丰富语义的关键字用于描述业务流程行为，使得业务流程的执行能够自动化。一个 WS-BPEL 业务过程的核心部分在<sequence></sequence>标签中定义，在这个标签中将使用<invoke>、<receive>、<reply>、<assign>、<throw>、<wait>、<terminate>、<copy>等结构对业务过程进行描述。

一个关于网上书店购书的 WS-BPEL 业务流程如下：

```
<sequence>
    <!-- Receive the initial request from client -->
    <receive partnerLink="Client"
             portType="buy:BuyBookPT"
             operation="BuyBook"
             variable="BookPurchase"
             createInstance="yes" />
    <!-- Prepare the input for the Book Rating -->
    <assign>
    <copy>
  <from variable="BookPurchase" part="book"/>
  <to variable="BookRatingRequest"
part="book"/>
          </copy>
        </assign>
  <!-- Synchronously invoke the Book Rating Web
Service -->
      <invoke partnerLink="BookRating"
              portType="bkr:BookRatingPT"
              operation="BookRating"
inputVariable="BookRatingRequest"
outputVariable="BookRatingResponse"
      />
    …//省略
</sequence>
```

3.2　Alloy 建模语言

Alloy 是由 MIT 软件设计工作组于 1997 年设计出原型[10]的，已经在多个领域中应用。Alloy 是基于关系的一阶逻辑语言，同时它也是结构化的建模语言。根据文献[11]可知，Alloy 语言在语言表述能力、与用户的交互以及提供信息量方面都具有优秀的性能。而 Alloy Analyzer 能够为使用 Alloy 语言建模的模型提供自动化的分析功能。Alloy Analyzer 首先将模型转化为布尔表达式，然后使用满足性理论对模型进行分析。Alloy Analyzer 按使用技术来说其实是一个“模型查找”工具，只要给出一个 Alloy 语言的逻辑表达式，Alloy Analyzer 将会试图查找符合该逻辑表达式的模型，而这种逻辑表达式就是软件系统需要满足的属性。

Alloy 是为了分析系统的结构化复杂度而设计的一种模型语言。一个典型的 Alloy 模型是由一系列 signature、field、fact、predicate、assert 的声明组成。其中，signature 是 Alloy 的基本单位，代表了一个原子的结合，可能会包含 field 的声明；而 field 的声明则代表了 signature 之间的关系；fact 则是对现有 signature 的限制性声明，明确规定 signature 之间必须满足的关系；predicate 是可以拥有参数的限制声明，它可以被用于 fact 中；assert 就是断言，是笔者认定系统需要满足的系统属性，利用 Alloy Analyzer 就可以针对这种断言进行分析。如果系统不满足这个属性，Alloy Analyzer 就会产生不满足该属性的反例。

4　Alloy 建模与分析

为了利用 Alloy 对 WS-BPEL 业务流程进行分析，需要将 WS-BPEL 转化为 Alloy，并使用 Alloy Analyzer 进行分析。在这过程中，本文提出使用有限状态机（finite state machine）对 WS-BPEL 的业务过程进行建模，即使用有限状态机来描述 WS- BPEL 业务过程随着接收和发送消息而导致的状态转换过程，然后使用 Alloy 对该有限状态机和 WS-BPEL 进行描述，接着需要将需求或需要验证的属性用 Alloy 进行形式化描述，最后使用 Alloy Analyzer 对 Alloy 模型进行分析，得到分析结果，由此可以判断 WS-BPEL 描述的业务流程是否满足业务需求。

4.1　从 WS-BPEL 业务流程到有限状态机

WS-BPEL 实质上是使用发送和接受消息（message）

的方式来实现已有服务之间的交互。当客户向 WS-BPEL 发送服务请求消息时，WS-BPEL 业务过程通过相关的赋值等操作，准备好相关的请求消息，并向外部服务节点发送服务请求消息；外部服务执行后得到最后的结果，并向 WS-BPEL 发送其服务响应消息，而 WS-BPEL 正是通过接收响应消息从而完成服务的调用。利用同样的方法，WS-BPEL 业务流程也可以调用多个服务，实现服务功能的组合。WS-BPEL 在服务调用的过程中其状态在不断地变化，而 WS-BPEL 状态的变化正是由这些请求和响应消息交互的顺序所决定的。

基于以上分析，本文提出使用有限状态机方法来为 WS-BPEL 进行建模，具体步骤如下。

1) 消息变量获取

在每个 WS-BPEL 业务流程的定义中都有一个定义变量的部分，这些变量定义都包含在标签<variables> </variables>之中。这些变量定义了消息的类型，却没有指出是接收消息类型还是发送消息的类型。这就需要从具体的 WS-BPEL 业务流程定义中解析 XML 来明确消息类型。这些消息的类型其实就是有限状态机中所有接收和发送的消息。在关键词<reply>、<invoke>和<receive>中，指明了调用的是什么服务、什么接口、什么操作以及输入的变量。通过这个变量就可以了解在上面的变量定义中各个消息是接收还是发送消息的类型。

<variable *name*="BookRatingRequest"
messageType="bkr:BookRatingRequestMessage"
/>

上面的变量定义在<variable></variable>标签之中，该变量名为 BookRatingRequest，变量类型即消息类型为 BookRatingRequestMessage。所以在有限状态机符号集中可以加入 bookRatingRequestMessage 的消息类型。同样，根据其他变量定义可以将所有消息类型加入符号集中。

2) 状态转换关系

由于 WS-BPEL 业务流程的状态是通过接收和发送消息的方式来转变的，所以，业务流程与子服务的交互将改变业务流程的状态。在 WS-BPEL 规范中存在一些关键词是用来调用子服务（通过发送请求消息完成），也还有一些关键词是用来接收子服务执行结果的（通过接收子服务的响应消息完成）。这样根据不同的关键词，可以定义不同的状态转换关系。WS-BPEL 的关键词<reply>、<invoke>和<receive>实现了业务流程与子服务的消息交互。在有限状态机中，状态转换关系定义如表 1 所示。

4.2 从有限状态机到 Alloy 建模语言

本文以一个上文提到的购书业务流程为例，分析如何将有限状态机转化为 Alloy 建模语言。

1) 定义消息 Message，并分为输入消息以及输出消息。

```
abstract sig Message{}
abstract sig InputMessage extends Message {}
abstract sig OutputMessage extends Message {}
// Input Messages Definition
one sig ClientBookPurchaseMessage extends InputMessage{}
one sig BookRatingResponseMessage extends InputMessage{}
```

表 1 状态转换关系

WS-BPEL 关键词	有限状态机
Receive *partnerLink*=“client” *portType*=“buy:BuybookPT” *operation*=“BuyBook” *variable*=“BookPurchase”	
Invoke *partnerLink*=“BookRating” *portType*=“bkr:BookRatingPT” *operation*=“BookRating” *inputVariable*=“BookRatingRequest”	
Reply *partnerLink*=“client” *portType*=“ReceiveResultPT” *operation*=“ReceiveResult” *variable*=“BookResult”	

```
one sig BookResponseMessage extends InputMessage{}
// Output Messages Definition
one sig BookRatingRequestMessage extends OutputMessage{}
one sig BookRequestMessage extends OutputMessage{}
one sig WS-BPELBookPurchaseMessage extends OutputMessage{}
```

2) 定义状态 State，每个状态由输入和输出消息集合组合，接收或发送消息将导致状态的变化，而初始状态的输入和输出消息集合为空。

```
// State Definition
sig State {
    input: set InputMessage,
    output: set OutputMessage
}
fact initialState {
    let s0 = ord/first | {
      no s0.input && no s0.output
    }
}
```

3) 在定义好状态后，必须规定状态转化的条件，即接收和发送消息。使用 Alloy 建模语言定义的状态转换关系也需要根据有限状态机的状态转换关系有不同的定义，如表 2 所示。

表 2 描述了有限状态机的状态转换关系，而在 Alloy 的定义中使用了关键词 Pred 来定义一个约束，其中的参数如：*m*:BookPurchase 表示接收了类型为 BookPurchase 的 *m* 消息，而 *pre* 和 *post* 代表了在接收消息前后的不同状态，*pre* 是接收消息前的状态，*post* 代表了接收消息后的状态。下面是从一个实例中得到的接收和发送消息的完整定义。

```
//接收消息
pred ReceiveClientBookPurchaseMessage[m: ClientBookPurchaseMessage, pre, post: State] {
    pre = ord/first
        post.input = m
    }
//发送消息
pred SendBookRatingRequestMessage[m: BookRatingRequestMessage, pre, post: State] {
        one cbpm: ClientBookPurchaseMessage | cbpm in pre.input
        post.input = pre.input
        post.output = pre.output + m
    }
```

4) 必须定义状态转换的总体条件，用 fact 明确所有实例必须遵守的条件，即只能是通过输入和输出条件，并且不存在相同的 2 个状态。

```
//Transition condition
fact LegalTrans{
        all pre: State - ord/last, post : ord/next[pre] | one m:Message {
            (m in InputMessage) =>{
            ReceiveMessage[m, pre, post]
        }else{
            SendMessage[m, pre, post]
        }
        }
}
// not exist two equivalent states
fact NoEquivalentStates{
        no pre: State - ord/last |
```

表 2　　状态转换关系的 Alloy 定义

有限状态机	Alloy 定义
	Pred ReceiveBookPurchase[*m*:BookPurchase, *pre*, *post*: State]
	Pred SendBookRating [*m*:BookRating, *pre*, *post*: State]
	Pred SendResult [*m*: BookResult, *pre*, *post*: State]

let *post* = ord/next[*pre*] | {
post.input = *pre*.input &&
post.output = *pre*.output
}
}

4.3 Alloy Analyzer 分析业务流程

WS-BPEL 业务流程在经过转化之后已经成为使用 Alloy 语言描述的模型，可以使用 Alloy Analyzer 针对需要验证的属性进行分析。所以必须明确需求，并将需求使用 Alloy 进行形式化。

本文考虑了下面的属性进行分析：WS-BPEL 执行后必须能够达到最终状态。针对这个属性，使用 Alloy 可以表述为

assert FinallyEnd{
ord/last.input = InputMessage && ord/last.output = OutputMessage
}
check FinallyEnd for 9

其中，check 命令是用来验证一个断言在规定范围（scope）内的所有实例是否满足，如果该断言是正确的，则不会产生反例。反之，如果该断言是不成立的，则会产生不成立的反例。所以利用该命令，就可以检查需要验证的属性是否成立，即使不成立也可以利用生成的反例来检查出错的地方，并以此来修正模型的定义。另外一个需要注意的是，对于每个属性的验证是在一定范围内所有实例进行穷举式的检查。所以当没有出现反例时，就可以判断在该范围内所有实例的这个属性是成立的。但是，被验证的属性也有可能不成立，因为 Alloy Analyzer 并没有验证所有范围内的实例。根据文献[12]中针对“小范围假设”所做出的实验结果可以得到结论：绝大部分的 bug 在小范围内的实例中就可以检查出来。所以，经过验证后的属性基本是成立的。

而在购书业务流程这个例子中，使用 SAT4J 解算器，check 命令经过 125ms 的执行后，发现没有反例产生。可以基本确定该属性是成立的，而如果有反例产生，将 assert 改为 pred，并且执行 run FinallyEnd for 9 这个命令，经过 79ms 的执行后，可以得到一个实例，如图 1 所示。经过 9 个状态后，到达了最终状态。

根据需求，还可以检验其他的状态是否可达。下面的约束用来检验能否到达接收 4 种消息的状态，run 命令用来查找是否存在这样的实例。

pred StateReachable[S: State]{
S.input = ClientBookPurchase + BookRatingResponse&& S.output = BookRatingRequest + BookRequest1
}
run StateReachable for 5

运行后发现的实例如图 2 所示，经过 5 个状态，发现“State4”就是符合要求的状态。此外，利用 check 命令可以检查需要验证的属性是否成立，如果不成立将给出反例。在服务组合中，业务流程运行后不能进入死锁状态。在 Alloy 模型中，这个属性可以表达为

assert Deadlock{
no *pre*: State - ord/last |
let *post* = ord/next[*pre*] |
pre.input = *post*.input && *pre*.output = *post*.output
}
check Deadlock for 9

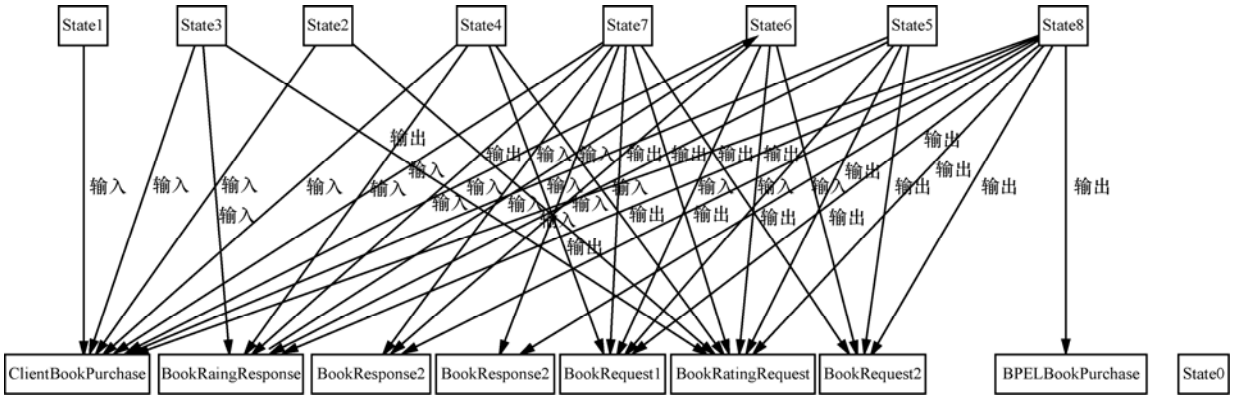


图 1 模型实例

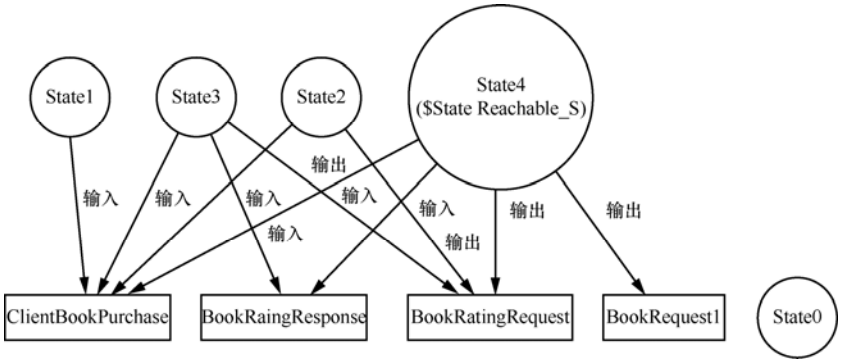


图 2　可达性实例

Executing “Check Deadlock for 9”

Solver=sat4j *Bitwidth*=4 *MaxSeq*=7　*SkolemDepth*=1　*Symmetry*=20

295 vars.81 primary vars. 653 clauses. 110ms.

Counterexample found. Assertion is invalid. 50ms.

图 3　Alloy Analyzer 死锁运行结果

上述 Alloy 断言描述了这样一种情况：在 Alloy 模型中不能出现反复进入同一种状态的情况，即前后状态相同。同样使用 SAT4J 解算器，check 命令经过 50ms 的运行后，发现反例，这就说明 Alloy 模型在范围 9 内存在死锁，运行结果如图 3 所示。

5　结束语

服务组合是服务计算领域最重要、最核心的问题，而服务组合的验证则尤为重要。本文提出了一种新的方法来对服务组合流程进行验证。这个方法首先将 WS-BPEL 转换为有限状态机，用有限状态机来描述业务流程在发送和接收消息过程中的状态转换；并在此基础上，使用 Alloy 建模语言来描述 WS-BPEL 业务流程；最后，使用 Alloy 描述需要验证的属性，然后使用 Alloy Analyzer 分析该模型是否满足系统应该满足的属性。

在将来的工作中，将进一步优化有限状态机的模型，使得模型更加精确地符合服务组合；另外，在使用 Alloy 描述服务组合时可以加入异常处理造成的状态变迁。

参考文献：

[1] AI-MASRI E, MAHMOUD Q H. Investigating Web services on the world wide web[A]. WWW 2008[C]. Beijing China, 2008. 795-804.

[2] WANG Y B, PAN S L. CPN-based verification of web service composition model[A]. ICEIT[C]. Chongqing, China, 2010. 795-804.

[3] DEEPAK C, SUPRIYA V, HRUSHIKESHA M. Verification of Web services modeled as finite state machines[A]. Fourth Asia International Conference on Digital Object Identifier[C]. Kota Kinabalu, Malysia, 2010. 526-531.

[4] TODICA V, VAIDA M F, CREMENE M. Formal verification in Web services composition[A]. 2012 IEEE International Conference on Digital Object Identifier[C]. Cluj-Napoca, Romania, 2012. 195-200.

[5] FOSTER H, UCHITEL S, KRAMER J, *et al*. Model-based verification of Web service compositions[A]. Proceedings of the Automated Software Engineering (ASE) Conference 2003[C]. Montreal, Canada, 2003. 152-163.

[6] FOSTER H. A Rigorous Approach To Engineering Web Service Compositions[D]. London: Imperial College London, 2006.

[7] DING J Q, ZHU H, ZHU H B. Formal modeling and verifications of deadlock prevention solutions in Web service oriented system[A]. Engineering of Computer Based Systems (ECBS)[C]. Oxford, England, 2010. 335-343.

[8] ZHAO H Q, WANG W W, SUN J. Research on formal modeling and verification of BPEL-based Web service composition[A]. Computer and Information Science (ICIS)[C]. Shanghai, China, 2012. 631-636.

[9] BIANCULLI D, GHEZZI C, SPOLETINI P. A model checking approach to verify BPEL4WS workflows[A]. IEEE International Conference on Service-Oriented Computing and Applications (SOCA'07)[C]. California, US, 2007. 13-20.

[10] Alloy: a language & tool for relational models[EB/OL]. http://alloy.mit.edu/alloy/, 2012.

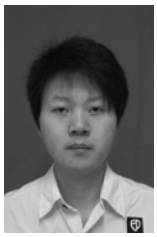
[11] ANDONI A, DANILIUC D, KHURSHID S, *et al*. Evaluating the Small Scope Hypothesis[R]. MIT Laboratory for Computer Science, 2002.

[12] AYDAL E G, UTTING M, WOODCOCK J. A comparison of state-based modelling tools for model validation[A]. TOOLS'08[C]. 2008.

278-296.

作者简介：

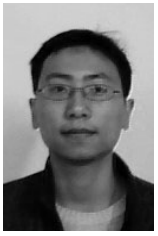
曹玖新（1967-），男，河南商丘人，东南大学教授、博士生导师，主要研究方向为移动互联网、服务与社会计算。



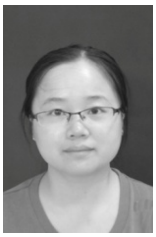
吴江林（1988-），男，江苏南通人，东南大学硕士生，主要研究方向为服务计算。

王国进（1983-），男，江苏高邮人，硕士，华为技术有限公司软件工程师，主要研究方向为服务计算。

刘波（1975-），女，河南南阳人，博士，东南大学副教授，主要研究方向为社会计算。



杨鹏伟（1989-），男，山东临沂人，东南大学硕士生，主要研究方向为服务计算。



董丹（1989-），女，江苏南通人，东南大学硕士生，主要研究方向为服务计算。