基于uMSD的Web服务组合验证方法研究

王志坚[1] 李雯睿[2,3] 杨种学[2] 张鹏程[1,3]
(河海大学计算机与信息学院 南京 210098)[1] (南京晓庄学院数学与信息技术学院 南京 211171)[2]
(武汉大学软件工程国家重点实验室 武汉 430072)[3]

摘 要 手工分析组合服务相当困难和耗时,为此提出了一种基于uMSD的Web服务组合的模型检验方法。如何简单和直观地表示Web服务组合的时态性质是该方法的关键问题。鉴于uMSD在简单性和表达力之间找到了一个平衡点,定义了uMSD的形式语法和语义。以Web服务组合OJA为实例,使用uMSD来图形化地表示组合服务的时态性质,展示了uMSD的可行性。实验分析表明,该验证方法能够有效地检测组合服务中的逻辑错误。

关键词 模态顺序图,Web服务组合,模型检验
中图法分类号 TP311 **文献标识码** A

Research on Verification of Web Service Composition Based on uMSD

WANG Zhi-jian[1] LI Wen-rui[2] YANG Zhong-xue[2] ZHANG Peng-cheng[1]
(College of Computer and Information, Hohai University, Nanjing 210098, China)[1]
(School of Mathematics & Information Technology, Nanjing Xiaozhuang University, Nanjing 211171, China)[2]
(State Key Laboratory of Software Engineering, Wuhan University, Wuhan 430072, China)[3]

Abstract In order to solve the problem that analyzing the composite services by manual is rather difficult and time-consuming, an approach was proposed to verify composite services by model checking based on uMSD. How to represent the temporal properties of the composite service easily and intuitively is a critical issue of the approach. Because uMSD finds a balance between simplicity of use and expressiveness, the paper defined the formal syntax and semantics of uMSD. In the paper, uMSD was used to graphically represent the temporal properties of a composite service On-the-Job Assistant as a case study, presenting the feasibility of uMSD. A series of experiments show the approach can effectively detect the logical errors in the composite service.

Keywords Modal sequence diagram, Web service composition, Model checking

1 引言

Web服务是一种新型分布式计算范型——面向服务的体系结构的具体实现之一[1]。单个Web服务的功能是有限的,为了快速地构建分布式企业应用,可以将多个基本服务组合成一个增值服务来满足用户不断变化的需求。Web服务组合的本质是多个服务之间进行交互。通常服务是由不同的组织开发和管理的,在网络环境中就要保证这些服务之间互操作的正确性。

为了支持形式化分析和验证Web服务组合的正确性,应考虑Web服务组合验证的3个重要方面:Web服务组合过程的性质表示、建模和采用的验证技术。当前,需从以下3个方面考虑Web服务组合方法所存在的局限。

时态逻辑通常用来推理并发系统,由于其内在结构的复杂性,使得难以正确地表示性质。顺序图表示性质易于理解,但表达力不够。这时需要考虑如何在简单性和表达力之间找到一个平衡点,以便以一种直观的具有足够表达力的图形化性质规约语言,来表示Web服务组合的性质。Modal Sequence Diagram(MSD,模态顺序图)[2]具有图形化的优点,并且表达力较强,适合用来表示业务过程的时态性质。

Web服务组合建模语言WS-BPEL[3],WS-CDL和BPMN等都是基于XML的建模语言,这些语言本身是无形式化语义的,这会导致规约的不一致、歧义和不完整等问题,同时在使用时还需要考虑将这些建模语言转换为何种形式化模型以用于验证目的。

服务构件之间的交互行为非常复杂,使用已有的Web服务组合过程建模语言对其进行建模是非常繁琐的,所以手工分析业务过程是不现实的。由于模型检验技术具有高度自动化的特点[4],因此采用模型检验技术来检测组合服务的模型是否满足给定的性质,即分析组合服务的行为规约的正确性。

到稿日期:2010-10-13 返修日期:2011-03-02 本文受国家高技术研究发展计划(2007AA01Z178),武汉大学软件工程国家重点实验室开放基金项目(2010-08-01),河海大学中央高校基本科研业务经费(2009B04314),江苏省高校自然科学基金(11KJD520010)资助。
王志坚(1958—),男,博士,教授,博士生导师,CCF高级会员,主要研究方向为软件复用技术、构件技术、分布式计算,E-mail:w51178@sina.com;李雯睿(1981—),女,博士,讲师,CCF会员,主要研究方向为形式化方法、Web服务建模、分析和验证;杨种学(1974—),男,副教授,CCF会员,主要研究方向为分布式计算;张鹏程(1981—),男,博士,讲师,CCF会员,主要研究方向为软件建模、分析和验证。

2 模态顺序图 uMSD

2.1 uMSD 基本概念

UML 2.0 顺序图已广泛应用在业界[5]，但其语义模糊以至于不能被有效地使用。MSD 是一种基于场景的图形化语言，是利用 universal/existential 模态语义扩展 UML2.0 顺序图而来的。MSD 区分了可能的场景和强制场景，eMSD(existential MSD)描述可能的场景，uMSD(universal MSD)描述强制场景。由于 eMSD 是对需求不充分的描述，其表达能力相当弱，只定义了一组样例运行集，因此适用于系统生命周期的早期。uMSD 具有较强的表达力，本文利用 uMSD 来表示业务过程的时态性质。

一个 uMSD 图是由生命线(lifelines)和事件构成的，每条垂线表示构件实例的生命线，连接生命线的箭头表示事件，且事件沿着生命线自顶向下是有序的。事件包括发送消息、接收消息或状态不变式(StateInvariant)。状态不变式即条件规定了运行时一个或多个构件实例的约束，覆盖一条或多条生命线。条件是构件实例性质的布尔表达式，共享条件的实例需同步执行。

uMSD 规定所有可能的系统运行的限制。在 uMSD 中，消息分为两类：cold(existential)消息表示可能发生的消息，hot(universal)消息表示必须发生的消息。状态不变式也分为 hot 状态不变式，表示条件必须成立；cold 状态不变式，表示不强制条件成立。如图 1 所示，uMSD 的边框是实线，图的左上角标为 usd，其中 cold 条件用蓝色虚线菱形表示，而 hot 消息用红色实线箭头表示，该图含义是如果条件 cond 成立，则必须顺序执行 m_1 和 m_2。

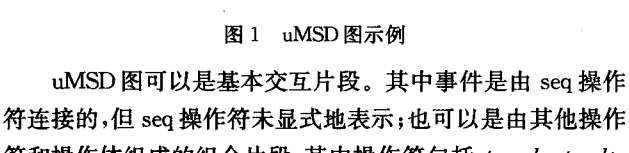

图 1 uMSD 图示例

uMSD 图可以是基本交互片段。其中事件是由 seq 操作符连接的，但 seq 操作符未显式地表示；也可以是由其他操作符和操作体组成的组合片段，其中操作符包括 *par*、*loop*、*alt*、*consider/ignore*、*critical* 和 *negate*，操作体又可能是组合片段。为了充分理解 uMSD，我们给出了 uMSD 的元模型，如图 2 所示。

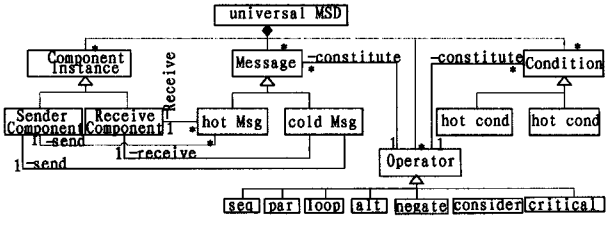

图 2 uMSD 元模型

2.2 uMSD 转换规则

为了使 uMSD 能够用于形式化分析、验证和监控，我们已在文献[6]给出了基于自动机理论的 uMSD 的形式化语义。弱交换 Büchi 自动机(Weak Alternating Büchi Automaton，WABA)[7]具有 existential 和 universal 结构，适合用来解释 uMSD 的语义规则。

uMSD 的语义规则分为基本转换规则和组合转换规则。基本转换规则讨论了如何将一个 uMSD U 中的单个消息或条件转换成 WABA。基本交互片段只能描述简单的场景，而复杂的场景需要用操作符将基本交互片段构成组合片段。组合操作符包括 *seq*、*alt*、*par*、*loop*、*consider/ignore*、*negate* 和 *critical*，增强了 uMSD 的表达能力。由于篇幅限制，这里仅给出以上操作符转换成 WABA 算法的主要思想。首先是两个基本的算法：merge 和 compose，操作符算法是建立在这两个算法的基础之上的。

merge 算法展示了如何将两个 WABAs A_1 和 A_2 顺序组合。该算法添加了从 A_1 的粘合状态 $s_{glue} \in G_1$ 到 A_2 的初始状态 s_0^2 的 ε 转换；A_2 中所有状态 s 到接受状态 s_{acc}^2 或拒绝状态 s_{rej}^2 的转换都替换为 A_2 中所有状态 s 到 A_1 的接受状态 s_{acc}^1 或拒绝状态 s_{rej}^1 的转换，并删除 A_2 中的接受和拒绝状态。

compose 算法是求多个 WABAs 的并。compose 算法用 ε 标签将新的初始状态 s_0、新的接受状态 s_{acc} 和新的拒绝状态 s_{rej} 分别与 n 个 WABAs 的初始状态 s_0^i、接受状态 s_{acc}^i 和拒绝状态 s_{rej}^i 连接。

seq 操作符的含义是通过基本转换规则生成相应的 WABA，然后用 merge 算法将从 r 到 u 个事件相应的 WABAs 两两顺序组合。

alt 操作符的含义是将 $w^{r,u}$ 分成 n 个操作体 $alt_1, alt_2, \cdots, alt_n$，对每个操作体应用 *seq* 算法生成相应的 WABA，然后用 compose 算法将这 n 个操作体的 WABAs 组合成一个 WABA。

par 操作符的含义是将 $w^{r,u}$ 上划分成 n 个操作体 $par_1(w^{l1,u1}), par_2(w^{l2,u2}), \cdots, par_n(w^{ln,un})$，在保证每个操作体中事件顺序不变的前提下，每个操作体中事件与其他操作体中事件以任意顺序交错执行，得到含有 h 个元素的 *par* 集$\{par_1(w^{l1,b1}), par_2(w^{l2,b2}), \cdots, par_h(w^{lh,bh})\}$。对于每个并行组合后的 $par_i(1 \leqslant i \leqslant h)$，应用 seq 算法生成相应的 WABA，然后用 compose 算法将这些 WABAs 组合成一个 WABA。

loop 操作符规定了 *loop* 循环的下界为 h、上界为 $p(1 \leqslant h \leqslant p)$，其含义是反复执行操作体至少 h 次和至多 p 次，用 $loop^*$ 表示无限次循环。*loop* 集由 $loop_1 = (w^{r,u})^h$，$loop_2 = (w^{r,u})^{h+1}, \cdots, loop_{p-h+1} = (w^{r,u})^p$ 组成，其中 $(w^{r,u})^l$ 表示自身连接 $l(h \leqslant l \leqslant p)$ 次，转换成相应的自动机，然后组合成一个 WABA。

critical 操作符用来规定 *critical* 操作体中事件执行的原子性，只要 *critical* 操作体中第一个事件发生，就期望其余事件也按顺序紧接着发生，而不期望 *critical* 操作体中事件之间发生其他事件，从而保证该操作体执行的原子性。*critical* 操作符的语义是指该操作体的迹不能被其他事件交错执行。例如，当封装组合片段中含有 *par* 操作符时，可根据需求在 *par* 操作体中嵌套 *critical* 操作符来阻止事件之间交错执行。先用 seq 算法求得事件序列 $w^{r,u}$ 的 WABA，然后修改该 WABA 的转换标签，除了 *critical* 操作体中第一个事件的状态自转换标签为 $\Sigma \backslash M$，其余事件的相应状态上源自转换标签 $\Sigma \backslash M$ 均去除。当第一个事件发生后，如果其余事件 $e_i(i = r+1, \cdots, u)$ 没有发生，则需根据事件的模态修改其余事件的相应状态到接受状态或拒绝状态的转换标签。当事件 e_i 的模态为 cold 时，转换 $t = \langle s_i, M \backslash e_i, s_{acc} \rangle$ 为 $t' = \langle s_i, \Sigma \backslash e_i, s_{acc} \rangle$；或者当事

件 e_i 的模态为 hot 时，转换 $t=\langle s_i, M\backslash e_i, s_{rej}\rangle$ 为 $t'=\langle s_i, \Sigma\backslash e_i, s_{rej}\rangle$。

negate 操作符规定了不期望发生的场景。*negate* 操作体隐含了值为 FALSE 的 hot 条件作为该操作体的最大要素，即不期望的事件发生后不会再有其他事件发生，所以不允许 *negate* 操作符自嵌套，也不允许其他操作符嵌套 *negate* 操作符。

uMSD 考虑了未出现在交互片段中的事件，这样的事件被认为是违反了已规定事件的偏序关系，从而影响该交互片段的迹语言。*consider/ignore* 操作符是用来规定在交互片段中应考虑/忽略的事件。*consider* 操作符应用到 $w^{r,u}$，且 $w^{r,u}$ 分为主操作体 $cons_1(w^{1,u1})$ 和 *considered* 操作体集 $\{cons_2(w^{2,u2}),\cdots,cons_n(w^{n,un})\}$。*considered* 操作体集作为额外需考虑的操作体出现在主操作体之后，规定每个 *considered* 操作体的模态是 hot 或 cold。每个 *considered* 操作体与主操作体中事件交错执行，至多选择一个分支执行，就会影响封装组合片段的迹语言：选择一个模态为 cold 的 *considered* 操作体执行，则发生 cold 违反，使得该迹包含在封装组合片段的迹语言中；选择一个模态为 hot 的 *considered* 操作体执行，则发生 hot 违反，使得该迹不能包含在封装组合片段的迹语言中。另外，将 *considered* 操作体集中所有事件 E 从主操作体中事件相应状态上的自转换标签 $\Sigma\backslash M$ 中移除。而对于 *ignore* 操作符，则是将 *ignored* 操作体集中所有事件 E 添加到主操作体中事件相应状态上的自转换标签 $\Sigma\backslash M$ 中。由于篇幅限制，*ignore* 操作符的算法可以类似地给出。

3 实例分析

本节以一个工作助理(On-the-Job-Assistant，OJA)为实例贯穿于本文。OJA 是一个 Web 服务组合，以提供给专业人员各种视频帮助。OJA 包含以下 4 个服务：(1)知识库(Knowledge Base，KB)存储各种专业的视频信息，并提供查询结果；(2)虚拟助理(Virtual Assistant，VA)向 KB 提交用户的查询请求，是用户和 KB 的中介；(3)银行(Bank)负责验证用户的信用卡帐户是否有效，并从用户的信用卡中扣除观看视频的费用；(4)其他服务提供者(Other Providers)是指当 KB 中没有满足用户需求的视频信息时，VA 则向其他服务提供者提交用户的查询请求。用户通过手机上网请求视频帮助，并提出需求：高质量的视频流(即在线实时观看视频，而不是将视频完全下载后再观看)和中等质量的音频，预定支付金额不超过 20 元，且用信用卡支付。

uMSD 是一种基于场景的语言，其图形化的优点表示性质直观且易于理解，具有足够的表达力，可以表示 Web 服务组合过程中的性质。下面考虑了 VA 业务过程的两个性质，将 uMSD 描述的场景转换成 WABA，可用于后续分析和验证组合服务行为的正确性。

性质 1 如果用户的信用卡帐户没有经过确认，则用户就不能获取视频。该性质是一个简单的安全性性质，其 uMSD 如图 3 所示，用 *negate* 操作符来规定不期望发生的场景。根据定义的基本规则和 *negate* 操作符算法，生成的 WABA 如图 4 所示，每个状态上的自转换标签 $\Sigma\backslash M$ 未显式地标出。值为 FALSE 的 hot 条件作为该交互片段的最大要素并未显式地表示在 uMSD 图中。若用户的信用卡帐户没有经过银行确认，而 KB 向用户提供视频，这时触发了值为 FALSE 的 hot 条件的执行，则迁移到拒绝状态 s_{rej}，这条执行轨迹是无效的轨迹。

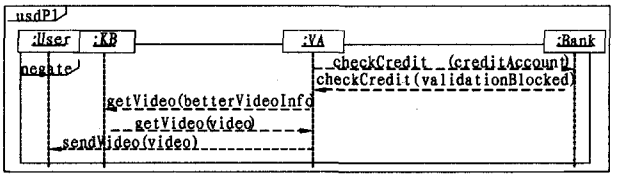

图 3 性质 1 的 uMSD 表示

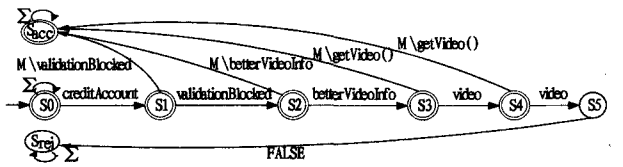

图 4 性质 1 的 WABA

性质 2 如果检测用户的信用卡是有效的，并且假设用户的信用卡帐户内有足够的金额，则最终用户能够获取视频。该性质是一个简单的活性，图 5 显示了性质 2 相应的 uMSD 表示。根据定义的基本规则、*seq* 操作符和 *loop* 操作符算法，生成的 WABA 如图 6 所示，每个状态上的自转换标签 $\Sigma\backslash M$ 未显式地标出。

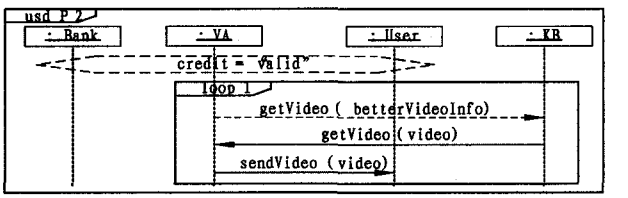

图 5 性质 2 的 uMSD 表示

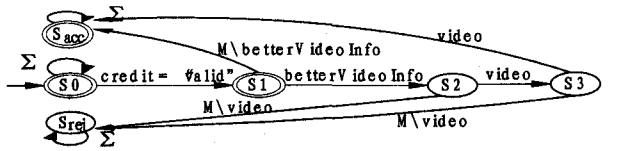

图 6 性质 2 的 WABA

表 1 将各个性质的类型、构件实例个数、事件个数、生成的 WABA 的状态个数和转换个数作为参数，比较了 2 个性质及其相应的 WABA。比如说，对于性质 2 需要 4 个参与者、4 个事件，生成的 WABA 有 6 个状态和 13 个转换。

表 1 比较 OJA 实例的 4 个性质和相应的 WABA

性质	性质类型	参与者	事件个数	状态个数	转换个数
P1	安全性	4	5	8	18
P2	活性	4	4	6	13

4 基于 uMSD 的 Web 服务组合的验证方法

针对已有 Web 服务组合的静态验证方法表示待验证的性质不够清晰和直观，并已引入图形化 uMSD 来表示 Web 服务组合的性质，我们提出了一种基于 uMSD 的 Web 服务组合的模型检验方法。结合 Web 服务组合的特点，为了描述业务过程之间的并发交互过程，采用 SPIN 模型检验工具[8]作为验证工具。本方法的流程分为以下 5 个步骤，如图 7 所示。

(1) 首先根据 uMSD 的基本转换规则和组合转换规则，

将 uMSD 规约转换为相应的 WABA；

(2) 由于 SPIN 模型检验工具不能直接处理 WABA，因此需进一步将 WABA 转换为 SPIN 模型检验工具可接受的性质规约语言 Büchi 自动机；

(3) EFA(Extended Finite-state Automaton，扩展有限状态机)作为 Web 服务组合过程的形式化模型，即将 Web 服务组合描述语言 BPEL 映射为 EFA 模型；

(4) 把 EFA 模型转换为 Promela 代码，因为 Promela 代码是 SPIN 模型检验工具的输入语言；

(5) 最后，性质的 Büchi 自动机和系统模型的 Promela 代码作为 SPIN 模型检验工具的输入，来验证系统模型是否满足待验证的性质。从理论上讲，验证过程是完全自动进行的，而在实际操作中，需要人工干预。这样可能产生两种结果：如果正确，则是 OK；如果错误，则显示出一条错误的轨迹，用来帮助组合服务的设计者来分析错误的原因。当修正了错误后，重新进行验证。在验证过程中，如果不能正常中止验证过程，比如系统规模太大，则需要调整模型检验工具的一些参数或对模型进一步抽象。

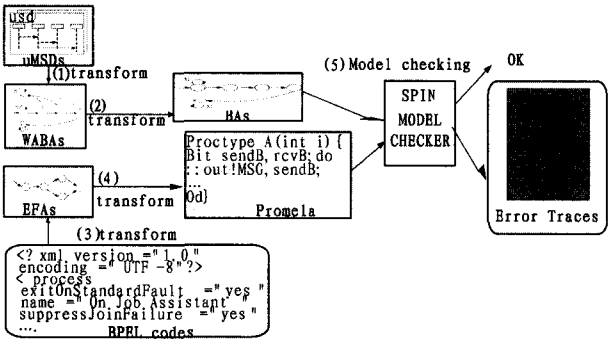

图 7　基于 uMSD 的 Web 服务组合模型检验方法流程

4.1　WABA 转换为 Büchi 自动机

模型检验问题假设 A 是系统 M 的自动机，那么 M 的行为是 $L(A)$ 语言。S 是系统性质的自动机，则 $L(S)$ 包含了允许的行为集合。当 $L(A)\subseteq L(S)$ 时，则称系统模型 A 满足性质 S，也就是说 $L(A)\cap\overline{L(S)}$ 包含的每条行为轨迹对应于一个反例。所以，模型检验工具需要对性质的自动机 S 求补，才能执行分析。

SPIN 模型检验工具的性质规约语言是 LTL 或 Büchi 自动机，为了使得 SPIN 接受 uMSD 为其性质规约语言，如果直接将 WABA 转换为期望性质的 Büchi 自动机，则还需要对 Büchi 自动机求反。然而，求一个 Büchi 自动机的补自动机相当复杂，所以一个替代办法就是将 WABA 直接转换为代表不期望行为的 Büchi 自动机。当执行 Büchi 自动机到达接受状态，则表示性质违背。

由于 WABA 具有 existential 和 universal 结构，因此很容易将其转换为表示行为补的 Büchi 自动机。将 WABA 转换为 Büchi 自动机的具体步骤如下。

1) 遍历 WABA，消除所有状态和转换标签的模态；

2) 原始 WABA 的中间状态和相关的转换标签在生成的 Büchi 自动机中依然保留为中间状态和原转换标签；

3) 把所有到 WABA 接受状态 s_{acc} 的转换标签删除，并将接受状态 s_{acc} 和自转换标签删除；

4) 所有到拒绝状态 s_{rej} 的转换标签保留，原始 WABA 的拒绝状态 s_{rej} 替换为生成 Büchi 自动机的接受状态。

根据以上步骤，我们可以得到 OJA 实例的两个性质的 WABA 相应的 Büchi 自动机。性质 1 和性质 2 的 WABA 转换为 Büchi 自动机如图 8 和图 9 所示。

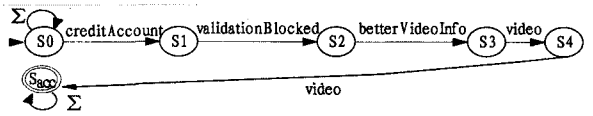

图 8　性质 1 的 WABA 转换为 Büchi 自动机

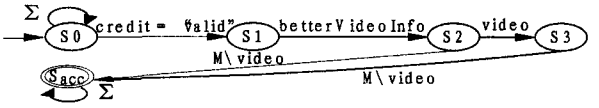

图 9　性质 2 的 WABA 转换为 Büchi 自动机

4.2　WS-BPEL 转换为 Promela 代码

WS-BPEL 的语义是非形式化的，不能提供严格的自动分析和验证，给系统分析工作带来了困难，难以保证系统建模的正确性。为了能够形式化分析和验证 WS-BPEL，有必要将 WS-BPEL 转换为 SPIN 模型检验工具的输入语言 Promela。这里采用扩展的有限状态机(Extended Finite-state Automaton，EFA)作为 WS-BPEL 和 Promela 之间的中间模型。

首先给出 EFA 的形式化定义。

定义 1(扩展的有限状态机)　一个扩展的有限状态机是一个六元组 $M=\langle Q,V,L,\delta,q_0,F\rangle$，其中

- Q 是有限的状态集；
- V 是有限的数据集，主要包括交互过程中用到的消息变量和控制变量；
- $L=$ [guardCondition]portype. action 是有限的转换标签集，其中 guardCondition 是一个有关变量集合的布尔表达式，作为转换的前置条件，包含了转换过程中数据流限制。也就是说，当前置条件 guardCondition 成立，那么动作发生。portype 指发送或接收消息的端口；action = ? / ! operation (parameter)表示交互动作，其中“?”和“!”分别表示接收消息和发送消息；operation(parameter)是操作格式，其中 operation 是操作的名称；parameter 是操作中使用的参数。! $Y(a)$ 和 ? $Y(b)$ 表示与外界环境或自动机进行通信。从操作的角度讲，输出动作 ! $Y(a)$ 是指发送消息，即操作 Y 将消息 a 发送到通信信道中。输入动作 ? $Y(b)$ 是指接收消息，即操作 Y 接收来自通信信道中的消息 b。标签 ε 表示内部转换，含义是没有动作发生。
- $\delta: Q\times L\to Q$ 是一个转换函数，表示状态 q 经转换标签 l 迁移到状态 q'，其中 $q,q'\in Q, l\in L$；
- $q_0\in Q$ 是初始状态；
- $F\subseteq Q$ 是终止状态集。

EFA 描述了组合服务的动态行为，从初始状态执行一系列的动作迁移到新的状态。这种迁移是用转换关系定义的，转换关系定义了在什么条件下执行动作和执行结果规定了服务状态的修改。

4.2.1　WS-BPEL 基本活动转换为 Promela 代码

WS-BPEL 的基本活动和 Promela 的映射关系如图 10 所示，以 invoke 和 receive 活动为例来简单阐述其含义。对于

invoke 活动，调用模式分为单向(one-way)调用和请求-响应模式。在单向调用模式中，invoke 活动定义了消息发送，而无需等待响应；定义了被调用伙伴的操作、端口和变量值。在请求-响应模式中，invoke 活动定义了伙伴的响应，当活动结束时输出变量存储了被接收的消息值。其中 pt! op(inVar)表示组合服务过程通过端口 pt、操作 op 和参数 inVar 调用另一个服务，是一个异步的过程，不需要参数返回；如果需要参数返回，则通过 pt? op(outVar)返回参数。

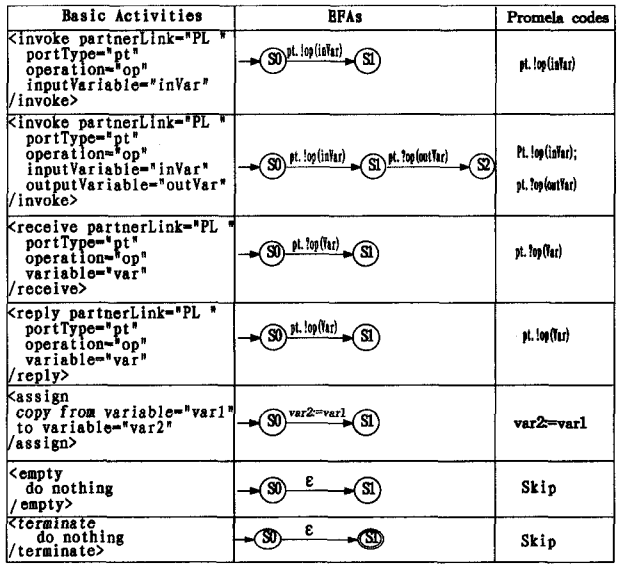

图 10　WS-BPEL 的基本活动和 Promela 的映射关系

类似地，receive 活动规定了接收消息，定义了操作、端口和接收来自伙伴消息的变量值。对于 receive 活动，pt? op(Var)则表示组合服务过程接受一个来自端口 pt、操作 op、参数为 Var 的请求。针对其它的基本服务操作，可类似解释。

4.2.2　WS-BPEL 结构化活动转换为 Promela 代码

结构化活动转化为 Promela 代码如图 11 所示。sequence 结构定义了多个活动顺序地执行、相应的自动机顺序组合。针对顺序活动(sequence)，可生成顺序的 Promela 代码 activity1 和 activity2。

if-else 结构定义了条件选择，由有序的选择分支组成，按照分支出现的顺序执行分支。针对 if…else 条件活动，可根据多个条件(cond1，cond2，…，cond$n$)生成可选择的多个活动。

flow 结构规定了并行执行的活动，即指所有活动交错执行。所有活动相应的 EFA 中状态交错迁移，而保持每个活动相应 EFA 状态顺序。根据生成的 EFA，在 Promela 代码中同样可以生成多个无条件可选择的活动，保持各个活动在原来每个 flow 活动中的顺序。

while 和 repeatUntil 结构规定了反复执行一个具体的活动。当 cond 条件满足时，重复执行活动对应的 EFA；当条件不满足时，则中止 while 循环。而在 Promela 代码中，通过 do 语句就可生成相应的表示循环的语句。

pick 结构规定了在一组互斥的事件集合中等待一个事件发生，并执行相关的活动。pick 是事件/活动的分支集合，根据发生的事件，选择一个分支执行。在 Promela 代码中，可以生成多个无条件的分支语言，无条件地选择其中的某一个执行。

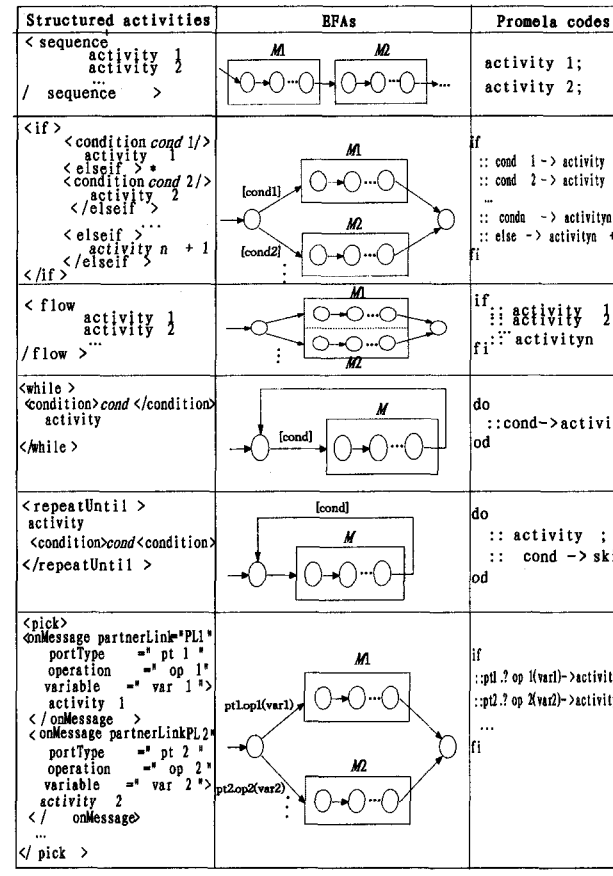

图 11　WS-BPEL 的结构化活动和 Promela 的映射关系

5　实验评估

5.1　实验环境和目的

实验环境设定：Windows XP 操作系统，Intel P8700 2.53 GHzCPU 和 1.98GB 内存。

将转换得到的 Promela 代码和性质的 Büchi 自动机的文本表示作为 SPIN 模型检验工具的输入，用于验证 BPEL 业务过程的系统模型是否满足 uMSD 表示的性质。实验目的有以下两个：(1) 对于转换为 Promela 的进程，这些进程之间进行并发交互，最终能够生成的状态数。(2) 对于用 uMSD 表示的正确或错误的行为，SPIN 能否验证它是正确的，或找到反例来证明行为是错误的。

5.2　验证结果

在实验中，调用 SPIN 模型检验工具的仿真器来解释 OJA 组合服务实例的 Promela 程序。仿真器可以帮助设计者获取有关系统模型的早期反馈。在设计者形式化分析系统模型之前，早期反馈有助于设计者理解系统模型。

首先调用 SPIN 的仿真器对系统进行模拟，OJA 实例的验证结果如图 12 所示。在本例中使用一个穷举状态空间搜索算法，而对于较大规模的系统来说，则采用比特状态哈希(Bit State hashing)算法来有选择地搜索部分状态空间。

为了分析仿真结果，仿真器模拟 VA 业务过程和其它 4 个服务之间的并发交互过程，使用顺序图(Sequence Chart)为设计者提供了不同视角，按照时间序列提供了进程之间的通信。

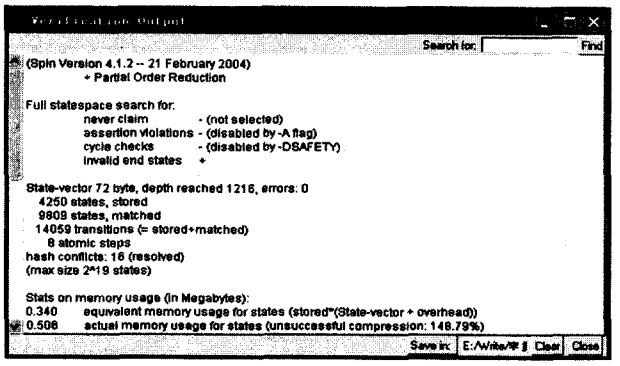

图 12 OJA 实例的验证结果

接下来验证 uMSD 表示的性质。OJA 实例的 Promela 代码和性质的 Büchi 自动机表示作为 SPIN 模型检验工具的输入，来验证组合服务的系统模型是否满足规定的性质。

性质 1 的验证结果如图 13 所示。对于性质 1，系统模型存在错误行为。在验证过程中，生成 171 个状态和 188 个转换时产生了一个反例。反例的结果说明，当检测信用卡不成功时，用户就不能够获得视频，应到达中止状态，从而验证了性质 1 表示的是错误行为。对于最终的验证结果和需求表示的行为是一致的。而性质 2 是正确行为，最终的验证结果是正确的，如图 14 所示。

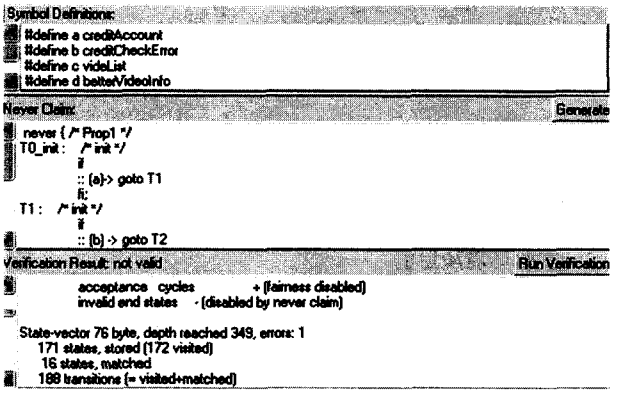

图 13 性质 1 的验证结果

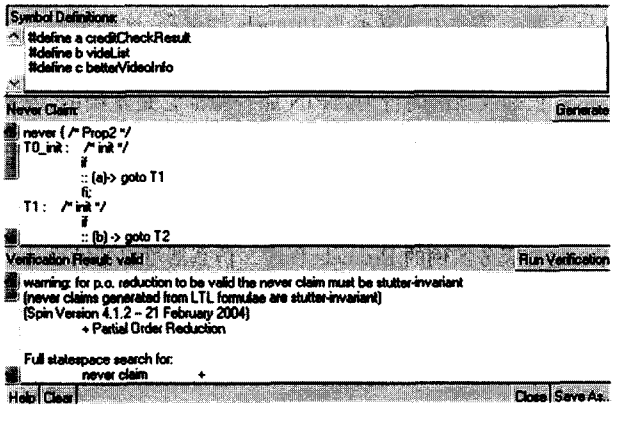

图 14 性质 2 的验证结果

6 相关工作

通过调研发现，Web 服务组合的静态验证方法多数采用了模型检验技术，将我们所提出的方法和一些典型的 Web 服务组合验证方法进行比较，如表 2 所列。

表 2 典型的 Web 服务组合验证方法比较

提出者	协作模型	形式化模型	性质规约语言	验证性质	模型检验工具
范等	抽象的服务组合模型	Petri 网	CTL	事务性质和可靠性	原型系统
Lohmann 等	BPEL	oWFN	PNML	不可达活动和可控制性	Fiona
Dijkman 等	BPMN	Petri 网	PNML	死锁和活锁	ProM
侯等	OWL-S	π 演算	进程表达式	死锁、状态空间的各种互模拟关系	MWB
Foster 等	WS-CDL 和 BPEL	FSP	MSC	安全性和有无死锁	LTSA-WS
Abouzaid 等	BPEL	π 演算	π 逻辑	功能性质	HA
雷等	抽象的服务组合模型	EDFA	LTL	安全性和活性	SPIN
Diaz	WS-CDL	TA	TCTL	可达性	UPPAAL
李等	BPEL	EFA	uMSD	正确性	SPIN

Petri 网作为一种基于状态的形式化建模方法，具有直观的图形化标识，能够对并发系统中控制流进行建模。一些基于 Petri 网的典型方法[9-11]的本质是将业务过程转换为 Petri 网模型，并判断是否满足期望的性质。然而，基于 Petri 网的方法侧重于描述服务内部的行为流，无法表示服务与外界的复杂消息交互。

进程代数是一种描述并发和动态变化的系统的建模语言，具有严格的理论基础，能够自动验证系统行为的性质。从组合的角度来看，进程代数提供了各种组合构造子，且在自动验证方面，进程代数优于 Petri 网。由于进程的迁移可以刻画服务的动态行为，一些基于进程代数的典型方法[12-14]利用进程的互模拟性来验证服务的行为能力是否等价，因此进程代数适合描述 Web 服务组合的动态行为，及验证业务过程的可靠性。然而，进程代数比较抽象，难以广泛推广。

自动机是用于形式化规约系统的基本模型，用状态集或程序位置来描述服务，服务的行为是用一组转换给出的。自动机可以作为更为复杂的形式化模型(如进程代数)的低级语义模型。使用自动机能够自动化分析过程，并提供简化的和严格的形式化标识。一些基于自动机的典型方法[15,16]通常是将 Web 服务组合规约转换成基于 XML 格式的自动机，并采用某种模型检验工具来验证规定的性质。使用自动机能够有效地实现分析技术，然而难以表示组合的高级方面，如事务性、移动性和复杂的数据结构。

结束语 网络环境是开放的、动态的，如何保证 Web 服务组合的行为正确性是亟待解决的问题之一。本文提出一种基于 uMSD 的 Web 服务组合模型检验方法。为了对服务构件的行为进行有效的验证，本方法鉴于 uMSD 图形化的优点，采用 uMSD 来表示 Web 服务组合的性质。uMSD 具有图形化的优点，在简单性和表达力之间找到了一个平衡点，用以简单和正确地表示 Web 服务组合过程中的时态性质，从而解决时态逻辑表示时态属性复杂和不够直观的问题。

当前，本方法所应用的实例仍然较为简单，仅用于展示所提验证方法的可行性和有效性。在下一阶段，我们将把验证方法应用到更大规模的系统中，并进一步考察方法的有效性。对于以时间为关键性质的系统，将进一步考虑对 uMSD 添加时间约束来表示实时需求，并验证实时性质的正确性。

参 考 文 献

[1] Papazoglou M P, Heuvel W J. Service oriented architectures: ap-

proaches, technologies and research issues[J]. The VLDB Journal - The International Journal on Very Large Data Bases, 2007, 16(3): 389-415

[2] Harel D, Maoz S. Assert and Negate Revisited: Modal Semantics for UML Sequence Diagrams[J]. Software and Systems Modeling, 2008, 7(2): 237-252

[3] OASIS. Web Service Business Process Execution Language Version 2.0 Specification[EB/OL]. http://docs.oasis-open.org/wsbpel/2.0/wsbpel-v2.0.pdf. 2007

[4] Jhala R, Majumdar R. Software model checking[J]. ACM Comput Surv, 2009, 41: 1-54

[5] Object Management Group(OMG). UML: Superstructure version 2.0 [S]. 2005

[6] 李雯睿. 基于 uMSD 的 Web 服务组合验证技术研究[D]. 南京: 河海大学, 2010

[7] Kupferman O, Vardi M. Weak alternating automata are not that weak[J]. ACM Trans. Comput. Log, 2001, 2(3): 408-429

[8] Holzmann G J. The SPIN Model Checker: Primer and Reference Manual[M]. Addison-Wesley, 2004

[9] 范贵生, 虞慧群, 陈丽琼, 等. 基于 Petri 网的服务组合故障诊断与处理[J]. 软件学报, 2010, 21(2): 231-247

[10] Lohmann N, Massuthe P, Stahl C, et al. Analyzing interacting WS-BPEL processes using flexible model generation[J]. Data & Knowledge Engineering, 2008, 64(1): 38-54

[11] Dijkman R M, Dumas M, Ouyang C. Semantics and analysis of business process models in BPMN[J]. Information and Software Technology, 2008, 50(12): 1281-1294

[12] 侯丽珊, 金芝, 吴步丹. 需求驱动的 Web 服务建模及其验证: 一个基于本体的方法[J]. 中国科学 E 辑, 2006, 36(10): 1189-1219

[13] Foster H, Uchitel S, Magee J, et al. WS-Engineer: A Tool for Model-based Verification of Web Service Compositions and Choreography[C]// IEEE International Conference on Software Engineering(ICSE 2006). Shanghai, China, 2006

[14] Abouzaid F, Mullins J. A calculus for generation, verification and refinement of bpel specifications[J]. Electronic Notes in Theoretical Computer Science, 2008, 200(3): 43-65

[15] 雷丽晖, 段振华. 一种基于扩展有限自动机验证组合 Web 服务的方法[J]. 软件学报, 2007, 18(12): 2980-2990

[16] Diaz G, Cambronero M, Tobarra M L, et al. Analysis and Verification of Timed Properties Applied to Web Service Compositions [C]// Proc. WS-FM. 2006: 178-192

(上接第 115 页)

通过以上步骤，可获得在模型 C 的状态空间中公式 $P_{\infty p}(\wp^{\leq t})$ 的可满足集合。其中，对于验证步骤的第一步，设某信息系统的抽象模型如图 1 所示。该模型具有 4 个状态，用圆圈表示，状态的标记分别为 $\{init\}$、$\{try\}$、$\{succ\}$、$\{false\}$。转移用箭头表示，箭头旁的组合符号分别表示该转移的触发动作和转移率。

由图 1 可知，在 $\{try\}$ 状态中存在不确定的两个动作 E、D。该模型经过均匀化处理后，结果如图 2 所示。

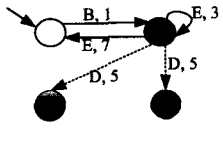

图 1　原始模型

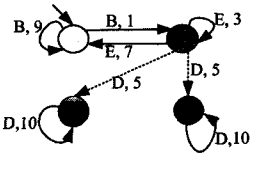

图 2　均匀处理后的模型

模型检测器 MRMC 为荷兰 Twente 大学、德国亚琛大学等合作开发的针对随机、概率系统的模型检测工具[11]，目前可免费使用。它支持的模型有 Markov 链、Markov 过程以及各种带有回报结构(Reward Structure)的变体[12]。特别地，它支持具有内部不确定性的连续时间 Markov 决策过程对于 CSL 公式的模型检测，正好满足本文的验证需要。在积模型中的可达概率求解，可直接用 MRMC 得出结果，并进一步得到可满足集合。限于篇幅，图 1、2 所示模型的具体验证过程不再给出。上述模型检测算法的复杂度分析过程可参考相关文献。

结束语　本文从理论上分析了对 CTMDP 模型进行功能性能验证的可行性，并给出了验证方法。验证过程分为如下步骤：首先，建立系统的 CTMDP 模型；其次，将待验证的功能和性能的统一性质用 pathCSL 表示；然后，将表示功能属性的正则式转换成有穷自动机，与 CTMDP 模型进行求积；最后，运用模型检测器 MRMC 获得满足验证属性的状态空间子集。如果关注的状态不满足期望属性，则改进 CTMDP 模型，从而提高设计的有效性。上述子过程均可通过已有的成熟方法进行。分析表明，本文提出的基于 CTMDP 模型的验证方法和步骤是可行的。下一步将研究带有回报结构的 Markov 过程的功能性能验证方法。

参 考 文 献

[1] Clarke E M, Grumberg O, Peled D A. Model Checking [M]. Cambridge: MIT Press, 2000

[2] Baier C, Haverkort B, Hermanns H, et al. Model-checking algorithms for continuous time Markov chains[J]. IEEE Transaction on Software Engineering, 2003, 29(6): 524-541

[3] 郭兵, 沈艳, 邵子立. 绿色计算的重定义与若干探讨[J]. 计算机学报, 2009, 32(12): 2311-2319

[4] 董威, 王戟, 齐治昌. UML Statecharts 的模型检验方法[J]. 软件学报, 2003, 14(4): 750-756

[5] Puterman M L. Markov Decision Processes: Discrete Stochastic Dynamic Programming[M]. New Jersey: John Wiley & Sons, 1994

[6] Baier C, Hermanns H, Katoen J-P, et al. Efficient computation of time-bounded reachability probabilities in uniform continuous-time Markov decision processes[J]. Theoretical Computer Science, 2005, 345(1): 2-26

[7] Hermanns H. Interactive Markov Chains[D]. Friedrich-Alexander-University. Erlangen-Nurnberg, 1998

[8] Baier C, Katoen J-P. Principles of Model Checking[M]. Massachusetts: The MIT Press, 2008

[9] Cloth L, Haverkort B, Hermanns H, et al. Model Checking path CSL[C]// Proc. of PMCCS-6. Illinois USA: September 2003: 19-22

[10] Baier C, Cloth L, Haverkort B, et al. Model Checking Markov Chains with Actions and State Labels[J]. IEEE Transactions on Software Engineering, 2007, 33(4): 209-224

[11] Markov Reward Model Checker Version 1.4.1 Manual[OL]. http://www.mrmc-tool.org/downloads/MRMC/Specs/. 2009

[12] 钮俊, 曾国荪, 陈波. 一种刻画功能和时间空间性能的同意验证模型 atsFPM[J]. 计算机学报, 2009, 32(4): 740-750