语义 Web 服务的自动化组合方法：研究综述*)

王杰生[1] 李舟军[2] 李梦君[1]

(国防科技大学计算机学院 长沙 410073)[1] (北京航空航天大学计算机学院 北京 100083)[2]

摘 要 语义 Web 服务的提出是为了解决 Web 服务资源在语义 Web 中的智能化整合问题，而语义 Web 服务的自动化组合技术作为这个整合过程中的一个关键技术正在蓬勃发展。本文考察了来自人工智能领域、形式化方法和自动推理等领域的众多服务组合方案，讨论了各种方法的原理和优劣之处，并探索了将来的研究工作和趋势。

关键词 语义网，Web 服务，服务组合

Solutions Towards Automated Composition of Semantic Web Services: A Survey

WANG Jie-Sheng[1] LI Zhou-Jun[2] LI Meng-Jun[1]

(School of Computer, National University of Defence Technology, Changsha 410073)[1]

(School of Computer Science & Engineering, Beihang University, Beijing 100083)[2]

Abstract Promoted by the ideal of intelligent integration of Web services on semantic Web, automated composition of semantic Web services as one of the keys to achieve that ideal is under great development. By examining a great variety of solutions with their pros and cons for service composition, which are proposed by researcher from fields in AI, formal method and automatic reasoning etc, a discussion of possible directions in future research on this subject is delivered.

Keywords Service composition, Web service, Semantic Web

1 引言

随着语义 Web 的各项技术，特别是语义 Web 本体标记语言 OWL[1]的迅猛发展和逐步完善，人们对利用语义 Web 技术来智能化地整合各种 Web 服务资源抱有越来越大的兴趣和期望，于是语义 Web 服务就应运而生。而语义 Web 服务的自动化组合作为这个整合过程中的关键技术之一，从一开始就受到众多研究人员的广泛关注。

语义 Web 服务的自动化组合工作主要来自两个领域：一个是人工智能领域，另一个是形式化方法和自动推理领域。当然，还有一小部分工作来自数据库领域，这一小部分工作主要是从 Web 服务组合方法的效率的角度来研究 Web 服务组合的[2]。人工智能以及形式化方法和自动推理这两个领域的工作既互相交叉，又互为补充。来自人工智能领域中的工作包括构建用于 Web 服务描述的本体，以便更好地支持 Web 服务资源的智能化整合。人工智能领域中的研究人员也从人工智能规划(AI planning)的角度提出了一系列的面向 Web 服务功能的 Web 服务组合(规划)方案[3~5]。来自形式化方法和自动推理领域的工作包括面向 Web 服务行为的服务组合(验证)方法[6,7]，也不乏借鉴自动化程序综合(program synthesis)[8]和模型检验(model checking)的方法[9]。

本文以自动化和智能化为基本出发点，讨论了多种 Web 服务组合方法。本文第 2 节考察了任何一个语义 Web 服务自动化组合方案都无法脱离的服务描述问题，并对现有的几个影响比较广泛的语义 Web 服务描述框架进行了比较。第 3 节系统而深入地讨论了各种 Web 服务组合方案的原理和各自的优劣之处，叙述的重点偏向于面向服务功能的那些方法；因为通常我们需要一个 Web 服务时，并不从 Web 服务的行为去指定它，关注的是这个 Web 服务能够达成什么样的功能。第 4 节对 Web 服务组合方案的发展趋势和研究前景进行了探讨和展望。

2 语义 Web 服务的服务描述

Web 服务组合的方法在一定程度上依赖于具体的 Web 服务描述框架。在本节中，我们将考察经常与语义 Web 服务组合纠缠在一起的服务描述问题，并对几个被学术界以及业界广泛讨论和研究的语义 Web 服务描述框架进行比较。

Web 服务组合(services composition)所采用的具体方法和技术依赖于 Web 服务描述(services description)所采用的方法和方式。更详细地说，Web 服务描述能够提供什么样的信息，乃至于提供多少信息都会对 Web 服务组合方法的选择做出限制。甚至，Web 服务描述提供信息的方式也会影响到 Web 服务组合方法的选取。因为，同样的信息其呈现形式不同，它为一个 Web 服务组合方法所利用这些信息的难易程度也有所不同。虽然呈现形式的差别并不影响 Web 服务组合算法计算的结果，但会影响到 Web 服务组合算法的计算效率。

从服务组合的角度来看，服务描述的主要工作在于选择

*)本文受到国家自然科学基金项目(60473057,60573057,90604007)的资助。**王杰生** 硕士研究生，主要研究领域为语义网、Web 服务组合；**李舟军** 博士，教授，CCF 高级会员，博士生导师，主要研究方向为进程代数理论、安全协议的形式化验证、语义网、Web 服务与 P2P 计算、数据挖掘与生物信息学；**李梦君** 博士，讲师，主要研究方向为安全协议的形式化验证技术。

合适的Web服务描述语言和Web服务模型，有效地为Web服务组合提供必要的信息。另外，由于语义Web服务是以语义Web为目标平台的，所以语义Web服务的描述框架也必须能够融入到语义Web的体系结构之中。

从Web服务的功能来看，Web服务是一个黑盒。我们可以不管Web服务内部的执行路径，而从Web服务的输入(input)、输出(output)、Web服务执行的前提条件(precondition)和Web服务执行之后的结果(effect)等四个方面来进行刻画。在OWL-S[10]中，这四者统称为IOPE，通常由Web服务的服务轮廓(service profile)来描述。在SWSL[11]中，也有与之对应的元素。SWMO[12]是面向UPML (Unified Problem Solving Method Development Language)[13]的，我们可以用该本体中"目标(goal)"的概念近似地来刻画Web服务的执行结果，借助UPML中的"目标-手段分析方法"进行Web服务组合。为了更好地描述Web服务的前提条件和结果，这些Web服务描述框架都具备了描述产生式系统中的规则的能力。与OWL-S一起使用的SWRL[14]是Horn逻辑和描述逻辑的一个超集，与SWMO一起使用的SWML[15]也融合了Datalog和F-逻辑[16]的一个交集。

从Web服务的结构来看，Web服务是一个白盒。我们可以把Web服务看作一个进程(process)，这个进程又由许多子进程组成，这些子进程构成了该Web服务可能的执行路径(通常执行路径不会是唯一的)。在OWL-S中，这些信息由Web服务的服务模型(service model)给出。而在SWSL中，服务模型的描述借助了一个已经比较完善的本体PSL[17]。

从Web服务的外部行为看，Web服务既是一个黑盒，也是一个白盒。在这样的模型下，Web服务只有在与外界进行消息交互的情况下才是可见的，在其余的情况下都是透明的。BPEL4WS[18]和WS Choreography[19]都是专门为此设计的描述语言，这些语言中的大部分元素与进程代数中的各种构造有着直接的对应关系。这些语言的目标与语义Web的设计思想却不甚相关，这些语言旨在描述Web服务是"怎样的"，而语义Web更关心Web服务是"什么样的"。我们通常不把在Web服务交互消息的层次上进行的程序综合称为"服务组合"，而称为"服务交响(services orchestration)"。

总的来说，我们很少去详细地描述一个Web服务的内部结构，总是从Web服务的外部行为的角度尽可能地提高Web服务结构的描述粒度。在服务组合中，我们更多地把Web服务看成为一个黑盒，因为Web服务组合不被Web服务验证，并不需要了解Web服务在行为方面太多的细节。这与我们在软件工程中对系统进行抽象的做法是一致的。

3 语义Web服务的自动化组合方法

语义Web服务组合和语义Web服务匹配的联系是非常密切的，本节的第一部分所讨论的一类服务组合算法就是构建于服务匹配之上的。虽然目前的语义Web服务匹配算法都局限于服务参数类型的匹配，但是这不等于说不能从服务的功能上和从服务的外部行为上来进行服务匹配。实际上，在这一类服务组合算法中，如果把服务组合当作一个状态搜索过程的话，那么服务匹配就相当于这个搜索过程之中筛选合适的后继状态的步骤。

第二部分所讨论的方案主要是从服务的功能的角度进行服务组合的。但是基于服务功能的组合算法在语义Web的环境中或多或少都有推理上的诟病(最显著的如计算的不可判定性)，难以获得普遍的适应性，这也是目前的语义Web服务匹配算法仍停留于参数类型的匹配的原因之一。

第三部分所讨论的一类方案是从服务的外部行为的角度进行服务组合的。需要指出的是，查询基本上不会指定人们所需的组合服务的行为，所以这一类方法一般都不会被单独使用。使用这一类方法的服务组合方案一般是两段式的，即先使用基于服务输入、输出参数的类型匹配或基于服务功能的人工智能规划的服务组合方法，寻找满足查询的组合服务，然后针对这个组合服务，使用本类方法找出与该组合服务交互的其他Web服务。

3.1 基于服务输入、输出参数的类型匹配的组合方法

在文[20]中Pao阐述了基于DAML-S[1)]的输入、输出参数类型的上下位匹配的思想。文[21]进一步发掘了参数类型之间的重叠关系，提出了部分匹配的算法；文[2]针对Web服务数量巨大的特点，在文[21]的基础上设计了基于矩阵[2)]的高效匹配算法以及前向搜索的服务组合算法。下面我们先简单地介绍一下基于Pao方法的一类服务组合方案的原理。

从输入、输出的角度来看待Web服务和查询，一个Web服务S能够满足一个查询Q意味着：对于查询Q提供的所有输入，Web服务S必须都能够接受；对于查询Q所要求的所有输出，Web服务S必须至少满足其中之一。根据类型之间是否存在互相包含或相交的关系，我们可以定义一个Web服务S能够满足一个查询Q的程度，它们分别是(按从高到低的顺序)：exact，plugIn，subsume和overlap[20, 21, 2]。

假设有四个在线销售水果的Web服务$S1$，$S2$，$S3$和$S4$，$S1$接受中国银行和花旗银行的人民币信用卡，$S2$接受所有的人民币信用卡，$S3$接受中国银行的人民币信用卡，$S4$接受花旗银行的多币种信用卡。如果你希望使用手中的中国银行的人民币信用卡和花旗银行的人民币信用卡来购得一些水果，那么从输入的角度看，服务$S1$满足你的查询的程度是exact；服务$S2$满足你的查询的程度是plugIn，因为中国银行和花旗银行的人民币信用卡都属于人民币信用卡；服务$S3$满足你的查询的程度是subsume，因为你还期望花旗银行的人民币信用卡也能够购买水果；服务$S4$满足你的查询的程度是overlap，因为你手中的信用卡与服务$S4$所接受的信用卡不存在简单的包含与被包含的关系，但你确实可以使用花旗银行的人民币信用卡购买水果。更进一步，如果从输出的角度看，假定Web服务$S1$只出售香蕉，那么服务$S1$满足你的查询的程度是plugIn。也就是说，在确定Web服务满足查询的程度时，输入参数类型的上下位匹配方向和输出参数类型的上下位匹配方向正好是相反的。

根据上面的分析，我们知道，仅仅在subsume或overlap程度上满足查询的Web服务是无法单独满足我们的确切需要的。为此，它必须与其他Web服务"相加"，并且这些相加

1) 已演变成为现在的OWL-S。

2) 作者在原文中使用的术语是table-base，但是这样的说法容易使没有数据库背景的读者产生误会。

的 Web 服务必须能够囊括查询提供的输入。这正是文[2]中的服务组合算法的基本出发点。该算法通过一个矩阵对服务进行初步的组合，以得到一个在 exact 或 plugIn 程度上满足查询的组合 Web 服务。该矩阵的维度由需要匹配的输入参数的个数决定，矩阵中的元素是一组 Web 服务，它们在矩阵的各维上的投影是这些 Web 服务中对应于该维的输入参数所能接受的类型。在上述的例子中，需要匹配的输入参数只有一个，即购买水果所使用的信用卡，我们用图 1 的坐标轴来示意这个一维矩阵(向量)。由图 1 可以看出，将 $S3$ 和 $S4$ 组合在一起，也能够在 plugIn 程度上满足查询。

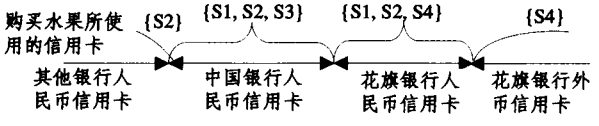

图 1　一个基于输入参数匹配的 Web 服务矩阵的图示

很明显，这样简单的 Web 服务组合(匹配)算法即使通过深度搜索也不能尽最大可能找到满足查询的组合 Web 服务。下面介绍的 Rao 方法[8]仍然依赖于输入、输出参数类型的上下位匹配，只不过该方法借鉴了自动化程序综合的思想，提出了采用线性逻辑推理进行 Web 服务组合，因而具备对 Web 服务进行参数个数的匹配的能力。但是该方法仍然没有充分地利用语义 Web 所提供的强大推理能力，最大限度地提取类型之间的各种关系，所以它也不能尽最大可能找到满足查询的组合 Web 服务。另外，该方法所采用的线性逻辑具有不可判定性，这从一定程度上削弱了它能够对 Web 服务进行参数个数的匹配的优势。

在文[8]中，Rao 用线性逻辑来刻画 Web 服务。譬如一个具有 I_1 类型和 I_2 类型输入参数和 O 类型输出参数的 Web 服务可表示为 $I_1 \otimes I_2 \multimap O$。线性逻辑不同于经典逻辑，它以资源的观点来看待命题，$\otimes$表示两个资源都存在，$\oplus$表示两个资源中必有一个，$\multimap$表示消耗前面的资源可以产生后面的资源。例如，分别用 D 和 C 表示一美元和一包烟，那么“两美元能购买一包烟”可以表示为 $D \otimes D \multimap C$。

Rao 利用了进程代数和线性逻辑之间紧密的理论联系来进行 Web 服务的组合，因此在 Rao 的方案中，组合 Web 服务的进程构造子实际上就是进程代数中的顺序运算符“.”、不确定选择“+”和并发运算符“|”。例如 Web 服务 exchange 可以让客户用一张礼券换取一支铅笔，即 $Coupon \multimap_{exchange} Pencil$；Web 服务 buy 可以让客户支付一美元购买一支铅笔，即 $Dollar \multimap_{buy} Pencil$；那么我们可以得出结论：无论客户是选择 Web 服务 exchange 还是选择 Web 服务 buy，都可以获得一支铅笔，即 $Coupon \oplus Dollar \multimap_{exchange+buy} Pencil$，尽管用礼券换来的铅笔可能质量上不如买来的铅笔。在 Rao 的方案中，与这对应的推理步骤是 $\dfrac{\Gamma \vdash Coupon \multimap_{exchange} Pencil, \Gamma \vdash Dollar \multimap_{buy} Pencil}{\Gamma \vdash Coupon \oplus Dollar \multimap_{exchange+buy} Pencil}$。通过对每个推理步赋予一定的操作语义(即产生对应的 Web 服务的进程代数表达式)，我们可以从推理序列获得组合 Web 服务的进程代数表达式，这个表达式可以直接翻译成组合 Web 服务的服务模型。

3.2　基于人工智能规划的组合方法

这类方法的基本思想是使用人工智能规划中的动作来对 Web 服务进行建模，利用人工智能中的规划算法来进行 Web 服务组合。这类方法所找到的组合服务通常比基于服务输入、输出参数的类型匹配的方法要来得准确，但是这类方法所能适用的 Web 服务范围也是有限的。

人工智能规划的理论基础是情景演算，但是具体的人工智能规划算法则是多种多样的，有采用逻辑程序等自动推理工具的规划算法[3~5]，有采用模型检验的规划算法[9]，还有利用图的各种搜索算法的规划算法[22]，也不乏利用机器学习来加速图搜索的规划算法。我们只对前面两类规划算法进行分析，其余的规划算法涉及太多的优化技巧，并不适合在此讨论。

文[3]是这类方法的先行者，它把 Web 服务看作 AI 规划中的动作，因而 Web 服务组合的过程就是产生计划的过程。文[3]所采用的规划算法是在逻辑程序 Golog 上改进而来的，而 Golog 是采用逻辑编程语言 prolog 来实现情景演算的。文[4]提出的基于 HTN(Hierarchical Task Network)规划算法的服务组合的方案以及文[5]采用的基于描述逻辑的服务组合方法也是沿用这样的思路。

在经典规划问题中，动作由动作的前提条件和效应所刻画，而动作的前提条件和效应是参与动作的个体的一组状态构成的，动作的执行将使得某些个体处于新的状态之中。例如在经典的积木世界里面，使用手臂举起物体的动作 *pickup* 可用 PDDL(Planning Domain Definition Language)描述如下：

```
(:action pickup
    :parameters (? ob)
    :precondition (and (clear ? ob)(arm-empty))
    :effect (and (holding ? ob)(not (arm-empty))(when (on-
    table ? ob)(not (on-table ? ob))))
)
```

情景演算最基本的思想就是通过把动作和情境(situation)具体化(reify)，以方便进行一阶逻辑推理。所谓情景，形式上就是参与规划的个体所处的状态。在情景演算中，我们用流(fluent)来抽象个体的某一特性随情景变化的过程，而个体的状态则就是个体在特定情境下所具有的特性。假设初始情景 S_0 恰好满足动作 *pickup*(a)的前提，也即是说 S_0 为{*clear*(a, S_0), arm-empty(S_0), …}；那么我们只要通过一步的推理就可得出执行动作 *pickup*(a)后的情景是{holding(a, *do*(*pickup*(a), S_0)), …}，其中 *do*(*pickup*(a))是情景集合上的一函词，指定了执行动作 *pickup*(a)之后相应的情景迁移。

情景演算通常包含两类公理：一类是动作的前提公理，用于指定各个动作能够被触发的条件；另一类是流的后继公理，用于指定各个状态在每个动作执行之后的变化情况。就上例来说，动作 *pickup* 的前提公理是 Poss(*pickup*(ob), S) = $\forall$ ob. clear(ob) $\wedge$ arm-empty。假设执行另外一个动作 *putdown*，将使得物体从手臂中脱离，则流 holding 的后继公理为 holding(x, *do*($action$, S)) = $action$ = *pickup*(x) $\vee$ $action \neq$ *putdown*(x)；而流 on-table 的后继公理 on-table(x, *do*($action$, S)) = on-table(x, S) $\wedge$ $action \neq$ *pickup*(x) $\vee$ …则显得更为复杂些，因为它出现在动作 *pickup* 的条件效应 when (…)(…)中。

但是针对经典规划问题提出的情景演算仅能产生由一组顺序动作构成的计划，因而 Golog(及其变种 ConGolog)在处理循环、非不确定性和并发性时的行为更像是一个解释器，而不是一个规划产生器。而且 Web 服务的行为特性和经典规划中的动作的行为特性是非常不一样的，这使得规划算法难以在 Web 服务组合中得到实用。因为 Web 服务的执行通常会导致新的个体产生，这些个体通常作为 Web 服务执行的结果返回

给用户；而在经典规划中，参与规划的个体不会在规划的过程中产生或消失，动作执行只是导致个体的状态发生变化。

如果把 Web 服务进一步细分为感知动作（sensing action）和实效动作（effect action）两类，则 Web 服务组合问题可以转化为不确定领域中的条件规划问题。下面我们将要介绍的用于服务交响自动化的规划算法正是反映了上述思想，这个思想对服务组合的自动化来说也具有非常大的借鉴意义。文[9]在 BPEL4WS 上使用不确定领域中的规划算法[23]，利用规划中的动作来刻画 Web 服务的交互信息，能够较好地处理 Web 服务的非确定性，产生非常健壮的 Web 服务组合方案。虽然这种方法避免了对 Web 服务产生的新个体进行处理，但从 Web 服务的交互信息的层面上进行显然违背了语义 Web 服务组合的初衷。

文[23]中的规划算法的基本思想是先用迁移系统（transition system）来刻画初始状态在各个动作执行后的迁移过程，然后用模型检验检查目标状态的可达性。目标状态的可达性蕴含了规划算法对付不确定性的健壮性。文[23]中定义了多种可达性，每种可达性的迁移过程不一样，但是每个迁移过程都可以通过 OBDD 进行编码。下面我们简单地介绍一下 OBDD 和刻画这个迁移过程的程序框架。

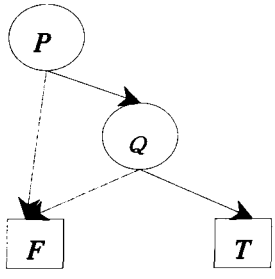

图 2　命题逻辑表达式 PQ 的一个 OBDD 表示

文[24]表明，OBDD（Ordered Binary Decision Diagram）对解决规模巨大的问题非常有效。在一个由 n 个命题变量构成的向量空间$\langle P_1, P_2, \cdots, P_n\rangle$中，任意一组状态集合可以由 OBDD 以紧致的形式表示出来。例如对于有三个命题变量$\langle P, Q, V\rangle$的系统，状态集合$\{\langle T, T, F\rangle, \langle T, T, T\rangle\}$可以用命题逻辑的表达式 PQ 表示，它的一个 OBDD 表示如图 2 所示，其中实边表示把源节点的变量赋为真 T，虚边表示把源节点的变量赋为假 F。

在规划中，我们习惯于用一阶逻辑谓词来刻画规划中的动作和状态。但是注意到参与规划的个体的数量 n 是有限的，我们可以把一阶逻辑谓词转化为命题词。特别地，一个 m 元谓词最多可以对应 n^m 个命题词。由于任意的动作都可以看作从一组状态集合到另一组状态集合的迁移，我们令 rn（Old）为状态集合上的一个函数，用于计算这样的状态集合 New，使得 New 中的元素不包含于状态集合 Old 中但却能够迁移至状态集合 Old 中的状态。设规划问题的初始状态集合为 I，目标状态集合为 G，我们可以使用下面程序刻画与这个迁移过程相对应的迁移系统：

```
function GENERATEPLAN(I,G)
    X:=G;
    while (rn(X)≠∅ 并且 I⊈X)
        X:=X∪rn(X);
    if (I⊆X)
        return X;
    else
        return ∅;
end function.
```

对上面这个程序进行模型检验，若 GENERATEPLAN（I,G）=∅，则不存在满足要求的规划。至于把上面的程序用 OBDD 编码的过程，请参考与模型检验相关的文[24]。

3.3　基于自动机和进程代数的组合方法

用自动机和进程代数可以非常自然地刻画 Web 服务的行为以及相应的状态变化。例如一个接收 *search* 消息然后发送 *result* 消息的 Web 服务 S，可以用图 3(a)中的进程代数公式来刻画：

$S=S_0$;
$S_0=?search.S_1$;
$S_1=!result.S_0$.

(a) 刻画 Web 服务 S 的行为的进程代数公式

$R=R_0$;
$R_0=?filter.R_1$;
$R_1=!result.R_0$.

(b) 刻画 Web 服务 R 的行为的进程代数公式

$Q=Q_0$;
$Q_0=?search.Q_1$;
$Q_0=?filterQ_1$;
$Q_1=!result.Q_0$.

(c) 刻画组合服务 Q 的行为的进程代数公式

图 3　一组刻画 Web 服务行为的进程代数公式

如果不区分消息前缀“?”和“!”在意义上的不同，那么图 3(a)中的各式同时也刻画了一个自动机，它的字母表为{? *search*, ! *result*}，状态集为$\{S_0, S_1\}$，并且 S_0 既是初始状态也是终止状态。由此可见，我们可以把进程代数看作自动机这个普遍的理论中的一个特例。

文[6]讨论了如何使用进程代数进行 Web 服务验证，文[7]给出了使用自动机从服务行为的角度对 Web 服务进行组合的方案。文[7]还利用了动态命题逻辑 PDL（propositional dynamic logics）对刻画 Web 服务行为的自动机进行编码。从这个编码的过程中，我们可以看到，只要增添一组刻画 τ 动作的 PDL 公式，该方案同样适用于进程代数。下面我们着重介绍文[7]中的 Web 服务组合方案。

为方便下文叙述，我们在图 3(b)和图 3(c)中给出了另外一个 Web 服务 R 和我们所要获取的组合服务 Q。现在让我们在自动机 S,R 和 Q 的基础上构造一个“组合”自动机，并使用 *PDL* 对这个“组合”自动机进行编码。设 u 是由所有原子程序的并构成的程序，即 $u=?\ search \cup ?\ filter \cup !\ result$。由于“组合”自动机中所有的动作都来自于自动机 S 和 R，我们用命题变量 $Moved_S$ 和 $Moved_R$ 分别来模拟自动机 S 和 R 在“组合”自动机中的动作，即

$[u](\langle ?\ search\rangle true? \rightarrow [?\ search](Moved_S \vee Moved_R))$;

$[u](\langle ?\ filter\rangle true? \rightarrow [?\ filter](Moved_S \vee Moved_R))$;

$[u](\langle !\ result\rangle true? \rightarrow [!\ result](Moved_S \vee Moved_R))$.

自动机 S,R 和 Q 在“组合”自动机中的动作遵循图 4 中的 *PDL* 公式，例如图 4(c)中的$[u](Q_0 \rightarrow (\langle ?\ search\rangle true \wedge [?\ search]Q_1))$的意义为：如果“组合”自动机当前状态满足 Q_0 的话，那么“组合”自动机可以接受输入？*search*，并且只要接收到输入？*search*，则“组合”自动机的下一个状态必定满足 Q_1。

$[u](S_0 \rightarrow [?search]Moved_S \wedge S_0 \vee \neg Moved_S \wedge S_0)$;
$[u](S_0 \rightarrow [?filter] \neg Moved_S \wedge S_0)$;
$[u](S_0 \rightarrow [!result] \neg Moved_S \wedge S_0)$.

$[u](S_1 \rightarrow [!result]Moved_S \wedge S_0 \vee \neg Moved_S \wedge S_1)$;
$[u](S_1 \rightarrow [?filter] \neg Moved_S \wedge S_1)$;
$[u](S_1 \rightarrow [?search] \neg Moved_S \wedge S_1)$.

(a) 刻画自动机 S 的各个状态在每种输入下的动作的 *PDL* 公式

[u] (R0→ [?filter] MovedR∧R1∨¬MovedR∧R0);
[u] (R0→ [?search]¬MovedR∧R0);
[u] (R0→ [!result]¬MovedR∧R0).

[u] (R1→ [!result]MovedR∧R0∨¬MovedR∧R1);
[u] (R1→ [?filter]¬MovedR∧R1);
[u] (R1→ [?search]¬MovedR∧R1).

(b) 刻画自动机 R 各个状态在每种输入下的动作的 PDL 公式

$[u](Q_0 \rightarrow (<?search>true \wedge [?search]Q_1))$;
$[u](Q_0 \rightarrow (<?filter>true \wedge [?filter]Q_1))$;
$[u](Q_0 \rightarrow [!result]false)$.

$[u](Q_1 \rightarrow (<!result>true \wedge [!result]Q_0))$;
$[u](Q_1 \rightarrow [?search]false)$;
$[u](Q_1 \rightarrow [?filter]false)$.

(c) 刻画自动机 Q 各个状态在每种输入下的动作的 PDL 公式

图 4 一组刻画 Web 服务的自动机的 *PDL* 编码

显然，这个“组合”自动机还须满足初始状态和接受状态等其它一系列的要求。具体地说，这个“组合”自动机的初始状态必须满足 $Q_0 \wedge S_0 \wedge R_0$，并且“组合”自动机的接受状态必须满足 $[u](Q_0 \rightarrow S_0 \wedge R_0)$。即是说，如果自动机 Q 处于接受状态中，那么此时的 R 和 S 也必须处于接受状态之中。最后，我们须指明各个自动机中的每个状态都是不同的，故在自动机 Q 中，我们有 $[u](Q_0 \rightarrow \neg Q_1)$；在自动机 R 中，我们有 $[u](R_0 \rightarrow \neg R_1)$；在自动机 S 中，我们有 $[u](S_0 \rightarrow \neg S_1)$。

从上面构造“组合”自动机的过程中，我们可以看出文[7]的基本思想与模型检验的原理如出一辙。这不是偶然的，因为从理论上讲，一个迁移系统与一组动态逻辑公式是对等的。所以，不论这个迁移系统是用于刻画动态逻辑中的 Kripke 语义结构，或是用于刻画自动机，还是用于刻画进程代数的操作语义，它从形式上总是与一组动态逻辑公式相对应。

4 语义 Web 服务自动化组合的难点及解决途径

4.1 Web 服务的消息、参数类型、执行的前提条件及结果

根据上面的分析我们可以看到，Web 服务的输入、输出参数类型信息在 Web 服务组合方法中有着重要地位。那么，为什么要使用类型呢？仅仅是因为现行的业界标准采用了类型来规范 Web 服务的参数吗？事实上，对纷繁复杂的事物进行分类，一直是人类认识世界的一个重要方法和工具，通过分类人类可以辨识出事物的共同点和不同点。语义 Web 就是采用了这样的世界观。撇开深奥的哲学问题，我们可以看到类型给 Web 服务带来的一个实际的好处就是：我们能够更加有效地从数目众多的 Web 服务中进行匹配和组合。值得注意的是，迄今为止，还没有一个 Web 服务组合算法能够充分地利用语义 Web 所提供的丰富的类型信息。然而，通过进一步改进文[2]中的服务组合方法，我们可以获得这样的组合算法。

当然，仅仅依赖 Web 服务的输入、输出参数的类型似乎无法给出一个正确的 Web 服务组合。例如只根据参数的类型，我们将难以区别整数的加法和乘法。虽然 Web 服务的名称可以为我们提供一些区别它们的线索，但是这样近乎自然语言的非结构化信息非常难以利用。通过对 Web 服务功能的结构化描述（即 Web 服务的前提条件及执行结果），我们能够保证获得的组合服务是能够满足查询的要求的。对 Web 服务功能进行结构化描述的需求也是导致本体和规则的整合问题成为当前语义 Web 中的研究热点的一个原因。

目前已经有很多研究探索了语义 Web 上本体和规则的整合工作。其中非单调逻辑编程方法 Answer Set 非常引人注目。Answer Set 是一个前向演绎系统，它具备了扩张析取逻辑程序（extended disjunctive logic program）和 Datalog 的特性。与经典逻辑的 Horn 逻辑程序相比，Answer Set 允许在规则中出现析取和取非运算[3]，但是限制了函词的使用，所以 Answer Set 的推理具有可判定性。文[25～27]尝试了用 Answer Set 模拟描述逻辑的推理，文[28]则提出基于 Answer Set 非单调性推理的人工智能规划方法。由此可见，Answer Set 在语义 Web 服务组合中的应用将会是十分引人注目的。当然，还有另外的一些整合工作试图通过某种折衷方法以保留 Horn 规则和描述逻辑的可判定性[29]，还有一些工作借助 F-逻辑来完成本体和规则的整合。

正如上一节所提到的，在 Web 服务消息层面进行服务组合不是我们的初衷，但是把这种方法作为服务组合的补充，将能够产生行为更为健壮的组合 Web 服务。我们知道，Web 服务在消息中传递参数，而且这些参数是带有类型的，加之 Web 服务的运行环境是一个典型的多 Agent 系统，Web 服务可以并发执行，所以带类型的进程代数[30]将是进一步深化该工作的有力工具。

4.2 语义 Web 服务组合语言的计算复杂度和表达能力

由上面的分析我们可以看到，用于 Web 服务组合的形式化描述和推理方法多种多样，有命题逻辑、Horn 逻辑、非单调逻辑，有描述逻辑、动态逻辑，有自动机和进程代数，还有线性逻辑、并发事务逻辑 CTR(concurrent transaction logics)[31]等亚逻辑结构。

从可计算性和计算复杂度的角度来看，在上述的形式系统中，有一些是可判定的，如描述逻辑和 Horn 逻辑都是一阶逻辑的可判定子集；还有一些是不可判定的，如一般意义上的线性逻辑。

从语言的表达能力的角度来看，上述的各个形式系统所能刻画的领域都存在一定程度的交叉，比如描述逻辑和动态逻辑也有着直接的对应关系，而动态逻辑的描述能力与一类特殊的自动机存在等价关系[32]；又比如进程代数能够方便地刻画自动机之间的交互，而且进程代数与线性逻辑也有着密切的联系[33]；另一方面，线性逻辑也不是与经典逻辑毫无联系[34]。

总之，语义 Web 服务组合的自动化离不开自动推理，而各种各样的形式系统都有其本身在计算复杂度和表达能力之

（下转第 29 页）

3) Answer Set 中的析取和取非运算的含义是不同于经典逻辑的含义的。

WebEye[9]中，进一步提高了大型分层视频组播系统的整体效能。

参考文献

1 Amir E, Mccanne S, Katz R H. An active service framework and its application to real-time multimedia transcoding. In: Proceedings of SIGCOMM '98 , Vancouver, BC CANADA, Sept. 1998
2 Li X, Paul S, Pancha P, Ammar M H. Layered multicast with retransmission (LVMR): Evaluation of error recovery schemes. In: Proceedings of NOSSDAV '97, May 1997
3 王晖. 异构环境下大型自适应视频组播方法优化研究[D]:[博士学位论文]. 国防科学技术大学, 2005
4 Tao Jun, Wang Hui, et al. Rate-distortion Optimized Rate Allocation Scheme for MC-EZBC based Video Streaming. In: 8th IASTED International Conference on Internet and Multimedia systems and Applications, 2004
5 Pan Yantao, Wang Hui, Wang Hongxia , et al. HSRM: A Hierarchical Scalable Reliable Multicast Model For Conferencing Control. In: 5th IEEE International Conference on High-Speed Networks and Multimedia Communications, 2002
6 Kouvelas I, Hardman V, Crowcroft J. Network Adaptive Continuous-Media Applications Through Self Organised Transcoding. In: Proc. Network and Operating System Support for Digital Audio and Video (NOSSDAV 98), 1998
7 Ratasamy S, Mccanne S. Inference of multicast routing trees and bottleneck bandwidths using end-to-end measurements. In: Proceedings IEEE Infocom '99 , New York, NY, Mar. 1999
8 王晖, 姜志宏, 张军, 等. 一个基于 IP 组播的协同学习环境[J]. 计算机工程, 2002, 28(3): 230～233
9 王晖, 黄英君, 姜志宏, 等. 基于 IP 组播的多媒体远程监控系统 WebEye 的设计与实现[J]. 中国图象图形学报, 2001, 6(A): 1235～1239

(上接第 23 页)

间的权衡和考量。如何选择和设计一个适合语义 Web 服务组合的形式系统，是语义 Web 服务组合的自动化中最核心、最根本的问题。目前的研究工作表明，对现有的形式系统进行剪裁和整合是解决该问题的一个有效途径[25～27,29]。

结束语 语义 Web 服务组合的粒度可参照语义 Web 服务描述分为三个层次。第一个层次是在 Web 服务输入、输出参数的类型上进行服务组合，该层次中的推理对象只涉及到 OWL 中的类(class)，语义 Web 刚好能满足它对推理能力的要求。第二个层次是在 Web 服务执行的前提条件和结果上进行服务组合，该层次通常要借助于 OWL 之外的推理机制，比如演绎规则系统，但是 OWL 和演绎规则系统的结合常常导致服务组合算法不可判定。第三个层次是在 Web 服务外部行为上进行服务组合，它所采用的方法或是借鉴或是直接取自服务验证领域的工作，但是通常我们并不会单独地在这个层次上进行服务组合，而是把它当作前面的两个层次上的服务组合方法的有效补充。

现有的形式系统各自有在计算复杂度和表达能力之间的权衡和考量，并不完全适合语义 Web 服务组合自动化的需要。不管是从服务功能的角度，还是从服务行为的角度，对现有的形式系统进行剪裁和整合将是解决语义 Web 服务组合自动化的一个重要途径。

参考文献

1 McGuinness D L, van Harmelen F. OWL Web Ontology Language Overview. World Wide Web Consortium (W3C) Recommendation. February 2004. http://www.w3.org/TR/owl-features/
2 Constantinescu I, Faltings B, Binder W. Large Scale, Type-Compatible Service Composition. ICWS, 2004. 506～513
3 McIlraith S A, Son T C. Adapting Golog for Composition of Semantic Web Services. KR, 2002. 482～496
4 Sirin E, Parsia B, Wu Dan, et al. HTN planning for Web Service composition using SHOP2. J Web Sem. , 2004, 1(4): 377～396
5 Baader F, Lutz C, Milicic M, et al. A Description Logic-based Approach to Reasoning About Web Services. WWW 2005 Workshop on Web Service Semantics
6 Salaün G, Bordeaux L, Schaerf M. Describing and Reasoning on Web Services Using Process Algebra. ICWS, 2004. 43
7 Berardi D, Calvanese D, De Giacomo G, et al. Automatic Service Composition Based on Behavioral Descriptions. IJCIS, 2005, 14(4): 333～376
8 Rao Jinghai, Küngas P, Matskin M. Logic-based Web Services Composition: From Service Description to Process Model. ICWS, 2004. 446～453
9 Pistore M, Traverso P, Bertoli P, et al. Automated synthesis of executable web service compositions from BPEL4WS processes. WWW (Special interest tracks and posters), 2005. 1186～1187
10 Martin D, et al. OWL-S: Semantic Markup for Web Services. November 2004. http://www.w3.org/Submission/OWL-S
11 Battle S, Bernstein A, et al. Semantic Web Services Language. In: September 2005. http://www.w3.org/Submission/SWSF-SWSL/
12 de Bruijn J, et al. Web Service Modeling Ontology (WSMO). June 2005. http://www.w3.org/Submission/WSMO/
13 Fensel D, Motta E, van Harmelen F, et al. The Unified Problem-Solving Method Development Language UPML. Knowl Inf Syst, 2003, 5(1): 83～131
14 Horrocks I, et al. SWRL: A Semantic Web Rule Language Combining OWL and RuleML. November 2003. http://www.daml.org/2003/11/swrl/
15 de Bruijn J, et al. Web Service Modeling Language (WSML). June 2005. http://www.w3.org/Submission/WSML/
16 Kifer M, Lausen G, Wu J. Logical Foundations of Object-oriented and Frame-Based Languages. J ACM, 1995, 42(4): 741～843
17 Grüninger M. A Guide to the Ontology of the Process Specification Language. R Studer, Staab S, eds. Handbook on Ontologies in Information Systems. Springer Verlag, 2003
18 Andrews T, et al. Business Process Execution Language for Web Services. May 2003. http://www-128.ibm.com/developerworks/library/specification/ws-bpel/
19 Kavantzas N, et al. Web Services Choreography Description Language. November 2005. http://www.w3.org/TR/ws-cdl-10/
20 Paolucci M, Kawamura T, Payne T R, et al. Semantic Matching of Web Services Capabilities. In: International Semantic Web Conference, 2002. 333～347
21 Li Lei, Horrocks I. A software framework for matchmaking based on semantic web technology. WWW, 2003. 331～339
22 Oh Seog-Chan, Lee Dongwon, Kumara S. A Comparative Illustration of AI Planning-based Web Services Composition . ACM SIGecom Exchanges, 2005, 5(5): 1～10
23 Cimatti A, Pistore M, Roveri M, et al. Weak, strong, and strong cyclic planning via symbolic model checking. Artif. Intell. , 2003, 147(1-2): 35～84
24 Burch J R, Clarke E M, McMillan K L, et al. Symbolic Model Checking: 10^{20} States and Beyond. In: LICS, 1990. 428～439
25 Motik B, Sattler U, Studer R. Query Answering for OWL-DL with Rules. In: International Semantic Web Conference, 2004. 549～563
26 Heymans S, Vermeir D. Integrating Description Logics and Answer Set Programming. In: PPSWR, 2003. 146～159
27 Heymans S, Van Nieuwenborgh D, Vermeir D. Extending Conceptual Logic Programs with Arbitrary Rules. Answer Set Programming, 2005
28 Lifschitz V. Answer set programming and plan generation. Artif Intell. , 2002, 138(1-2): 39～54
29 Levy A Y, Rousset M C. Combining Horn rules and description logics in CARIN. Artificial Intelligence, 1998, 104(1-2): 165～209
30 Igarashi A, Kobayashi N. A generic type system for the Pi-calculus. In: POPL, 2001. 128～141
31 Davulcu H, Kifer M, Ramakrishnan I V. CTR-S: a logic for specifying contracts in semantic web services. In: WWW (Alternate Track Papers & Posters), 2004. 144～153
32 Grädel E, Thomas W, Wilke W. Automata, Logics, and Infinite Games: A Guide to Current Research [outcome of a Dagstuhl seminar, February 2001]. Springer, 2002
33 Bellin G, Scott P J. On the Pi-calculus and Linear Logic. Theoretical Computer Science, 1994, 135(1): 11. 65
34 Avron A. The Semantics and Proof Theory of Linear Logic. Theoretical Computer Science, 1988, 57 : 161～184