

基于过程分析的服务发现

张大陆,张君婕,荆利强

(同济大学 计算机科学与技术系,上海 201804)

(daluz@ieee.org)

摘 要:在分析若干现有服务匹配算法的基础上,提出了基于过程分析的服务发现算法,利用服务描述中的 Process 信息,进行更加细化的 IO 匹配,处理部分 Output 匹配的情况,提高了服务的查全率。在原型系统的基础上对新的服务发现算法和现有其他算法进行了比较实验,对实验结果的分析证明了新算法的有效性。

关键词:电子商务;语义 Web;Web 服务;服务发现

中图分类号: TP393;TP391 **文献标识码:** A

Service discovery based on process analysis

ZHANG Da-lu, ZHANG Jun-jie, JING Li-qiang

(Department of Computer Science and Technology, Tongji University, Shanghai 201804, China)

Abstract: After analyzing some recent service matching algorithms, a new service matching algorithm based on process analysis was proposed. Using the information in service process during matching, more sophisticated IO matching could be done, and incomplete output matching could be dealt with. By using process-based service discovery algorithm, recall ratio could be improved. The results from comparison experiments verify the efficiency of the new algorithm.

Key words: e-commerce; semantic Web; Web service; service discovery

0 引言

Web 服务^[1]是一种新的 Web 应用模式,其中存在两个需要解决的问题:Web 服务应该能找到它需要的其他 Web 服务;Web 服务应该能交互组装成更复杂的服务。

本文集中解决前面的问题,即根据提供的服务找到所需的 Web 服务。要解决这个问题需要一种服务描述语言和一种现有服务与服务需求间的匹配算法。本文采用 OWL-s^[4]来描述服务,OWL-s 从语义的角度描述服务,将服务的抽象描述与服务交互协议以及实际的服务交互信息都联系在一起。

OWL-s 本体定义了三种基本元素:服务 profile、process model 和服务 grounding。本文主要工作在于发现和选择 Web 服务,只讨论匹配算法主要用到的服务 profile。服务 profile 由三个部分组成:1) 主要面向人的文字描述和联系方式;2) 服务的功能描述,规范该服务的输入、输出、服务正确执行前的先决条件以及服务成功执行后的结果,简称为 IOPE (Input, Output, Precondition, Effects),由于先决条件和执行结果还没有标准化,本文的匹配算法中只会涉及到输入和输出;3) 一系列描述服务特质的附加属性,通过这些属性将服务对应到本体上或把服务按某种分类法进行分类,还可以包括其他一些可选属性如元素质量排序、用户定义的服务执行时间等。

服务发现是通过在服务需求和现有服务间进行匹配来实现的。匹配算法根据领域本体的知识,在服务需求和现有服务描述间进行推理匹配。匹配结果并不是简单的“是”或者“否”两种结果,而是一个描述匹配程度的数值。

匹配算法对服务描述文档的生成方式,以及领域本体的

获得方式都没有特别的要求。只要服务描述文档是正确的 OWL-s 描述文档,领域本体是采用正确的 OWL 格式,匹配算法就可以通过这三项输入,获得匹配的结果。

1 相关工作

1.1 服务匹配

现有的服务匹配方法主要有两类:一类是 UDDI (Universal Description Discovery and Integration)^[5]等使用关键字进行比较的方法;另一类是基于 OWL-s 等具有语义的服务描述,利用领域本体的知识,对描述中的 Input 和 Output 等进行语义匹配的方法。

UDDI 只提供了很简单的查询机制:对商业机构、服务以及 TModule 名称进行基于关键字的查找。比如说,查找所有具有 WSDL 描述的服务,或者符合某 TModel 特定值的服务。UDDI 仅限于关键字匹配,没有提供任何形式的关键字方面的推理,查找也不具有灵活性。

1.2 具备语义的服务匹配

另一类方法利用了 SW (Semantic Web)^[2,3]提供的语义描述信息以及蕴涵推理能力,进行服务匹配,称为具备语义的服务发现。这方面的代表有 CMU (Carnegie Mellon University)^[6,7]、TUB (Technische Universität Berlin)^[8]和 NYU (New York University)^[9]等。

具备语义的服务发现,根据服务匹配所考虑的语义描述信息不同,主要分为两类:IO 匹配和 Profile 整体匹配。

IO 匹配,就是在匹配中主要考虑服务描述中的 Input 和 Output 信息,将服务需求和现存服务的 Input 和 Output 分别进行语义匹配,以此为根据来获得整个服务的匹配结果。IO 匹

配的缺点是只考虑了接口,查准率不高。两个 IO 信息相同的,可能在功能上截然不同。

实际上,OWL-s 等服务描述信息中不仅包括 IO 信息,还有 Precondition 信息,Effect 信息,以及关于服务详细功能描述的 Process 信息,这些信息通过一个 Service 概念联系在一起,服务 Profile 的整体匹配会把整个 Profile 作为一个概念进行匹配并把 Input、Output 和 Process 等 Profile 属性全部考虑进来。Profile 整体匹配的缺点是将整个服务描述作为一个概念进行蕴涵等语义匹配操作,复杂度高,服务描述中的任何不完整或部分不匹配信息,都会导致整个服务匹配失败,所以这种方法的查全率不高。

1.3 现有服务匹配方法的比较

表 1 介绍了若干现有服务匹配方法的特征,并进行比较。

表 1 相关工作比较

相关工作	匹配信息	语义推理	匹配过程	查全率	查准率
UDDI	关键字	无	无	低	低
CMU	语义 IO	蕴涵	分步骤	高	中
TUB	语义 IO	蕴涵	分步骤	高	中
NYU	语义 Profile	蕴涵	整体	中	高

其中,“匹配信息”指的是进行服务匹配所用到的服务描述信息;“语义推理”指的是在服务匹配过程中使用的语义推理方法,UDDI 没有任何的语义推理能力,CMU、TUB 以及 NYU 使用的是基本的语义蕴涵推理,也就是判断服务需求和现存服务的 Input 或 Output 对应的概念间是否存在语义蕴涵的关系;CMU 和 TUB 的“匹配过程”是分步骤的,首先进行 Output 匹配,在 Output 匹配成功的情况下,再进行 Input 匹配,最后综合获得整个服务的匹配结果,NYU 将整个 Profile 进行语义蕴涵推理,匹配过程是一步完成的。

“查全率”和“查准率”是比较服务匹配算法的两个重要指标。查全率指的是算法输出匹配成功的服务数量占事实上应该匹配的服务数量的比率;查准率是算法输出匹配成功的服务中正确的比率。UDDI 在这两项上都是最低的。CMU 和 TUB 的算法采用了 IO 分步骤的匹配方法,不会因为服务描述中的任何不完整的信息或者任何部分不匹配信息而导致整个服务匹配失败,所以查全率比 NYU 的方法高。NYU 匹配过程中不仅使用了 IO 描述信息,还使用了其他的语义信息,所以查准率比 CMU 和 TUB 高。

2 基于过程分析的服务发现

与 CMU 以及 TUB 的分步骤的 IO 匹配方法相似,我们的方法也采用了分步骤的 IO 匹配方法,但对前两者的算法进行了改进,以实现更加细化的 IO 匹配。

CMU 和 TUB 的算法中的 Output 匹配,只有当服务需求的所有 Output 都能在现存服务的 Output 中找到匹配项时,才算 Output 匹配成功,才能够接下来进行 Input 的匹配。但我们认为这样会造成遗漏,影响查全率。例如,服务需求希望输入一段文字,能够输出这段文字的作者及出自哪本书,而现有服务是人名言查询服务,能够根据输入查询并输出这段话的作者。这样,现有服务的输出只能满足服务需求的一部分。按照 CMU 和 TUB 的算法,这个服务匹配到此就应该返回匹配失败。但是,我们的方法这时并不返回失败,而返回一个较低的匹配分值,允许进行接下来的 Input 匹配。因为如果找不到其他更匹配的服务,这个现有服务还是能满足一部分需求的。最后根据 Output 和 Input 的匹配度,计算出两个服务的

总匹配度,按匹配度从高到低提供给用户一个可选服务列表。

进行更加细化的 IO 匹配,就不能在匹配过程中仅仅考虑 OWL-S Profile 提供的信息,还要通过 OWL-S Process 信息进行基于过程分析的服务发现。这里的过程分析,指的就是将 OWL-S Process 信息的分析纳入到服务匹配过程中。

假设现有服务 SA 的 Inputs 是集合 $IA = \{IA_k | 1 \leq k \leq p\}$, Outputs 是集合 $OA = \{OA_i | 1 \leq i \leq m\}$;而服务需求 SR 的 Inputs 是集合 $IR = \{IR_t | 1 \leq t \leq q\}$, Outputs 是集合 $OR = \{OR_j | 1 \leq j \leq n\}$,要将 SA 与 SR 进行匹配,计算出最后的匹配度 $M \in [0,1]$ 。

1) 对 SR 与 SA 进行 Output 匹配,计算出两者的 Output 匹配度 $MO \in [0,1]$ 。

这一步又分为:

(a) 依次匹配 SR 的 OR 与 SA 的 OA 中的每个元素,得出 $MO_{ij} \in [0,1] (1 \leq i \leq n, 1 \leq j \leq m)$ 。匹配评分规则如表 2 所示。

表 2 每个 OR_i 与 OA_j 匹配的评分规则

$OR_i ? OA_j$	MO_{ij}
exact	1
trans_exact	0.9
subsume	0.6
trans_subsume	0.5
invert	0.3
trans_invert	0.2
fail	0

(b) 依次求得 SR 每个 OR_i 所对应的匹配度 MO_i :

$$MO_i = \max \{MO_{i,j} | 1 \leq j \leq m\}$$

这里同时可以求得 SR 中的每个 OR_i 对应于 SA 中的哪个 Output XOR_i :

$$\exists j' (MO_{i,j'} = MO_i \wedge 1 \leq j' \leq m) \Rightarrow XOR_i = OA_{j'}$$

如果 $MO_i = 0$,也就是说服务需求 SR 的 Output OR_i 与现有服务 SA 中的所有 Output 的匹配度都是 0,那么 $XOR_i = \emptyset$,即空元素。

(c) 按照 SR 每个 OR_i 的权值 c_i 及其 MO_i , 求出总的 Output 匹配度 MO :

$$MO = \sum_{1 \leq i \leq n} (c_i * MO_i)$$

2) 找出现有服务的有效 Input 集合 U_{SAI} 。

这一步分为:

(a) 求出 SA 中的有效 Output 集合 U_{SAO} ,也就是说找出所有能对应到服务需求 Output 的现有服务的 Output:

$$U_{SAO} = \cup \{XOR_i | 1 \leq i \leq n\}$$

(b) 求出 SA 中的有效 Input 集合 U_{SAI} :

经过 Process 分析可知 SA 中的一个 Output OA_j 来自于 SA 的 Input IA_{k1}, IA_{k2}, \dots ,记做 OA_j 的来源 $Y_{OA_j} = \{IA_{k1}, IA_{k2}, \dots\}$,那么:

$$U_{SAI} = \cup \{Y_{OA_j} | OA_j \in U_{SAO}\}$$

3) 对 SA 和 SR 进行 Input 匹配,计算出 U_{SAI} 与 SR 的 Input 的匹配度 $MI \in [0,1]$ 。

这一步分为:

(a) 将 SA 有效输入集合 U_{SAI} 中的每个 Input $IA_{k'}$ 与 SR 中的每个 Input IR_t 进行匹配,计算出两者的匹配度:

$$MI_{i,k'} \in [0,1], 1 \leq t \leq 1, IA_{k'} \in U_{SAI}$$

(b) 对于 SA 有效输入集合 U_{SAI} 中的每个 $IA_{k'}$, 计算出其与 SR 的 Input 的总的匹配度 $MI_{k'}$:

$$MI_{k'} = \max \{ MI_{k',t} \mid 1 \leq t \leq q \}$$

(c) 根据 U_{SAI} 中每个 $IA_{k'}$ 的权值 $d_{k'}$, 计算出 U_{SAI} 与 SR 的 input 的匹配度 MI :

$$MI = \sum_{IA_{k'} \in U_{SAI}} (d_{k'} * MI_{k'})$$

4) 计算出 SA 与 SR 的匹配度 M :

$$M = MO * MI$$

3 有效 Input 查找算法

下面介绍对于服务描述,如何通过 Process 分析查找它的某个指定 Output 对应的 Input 集合。

在组合服务的 OWL-S Process 描述中,会有服务是如何由若干 Process 组件通过 Sequence、Split、Iterator 等结构组合而成的说明。同时也会指定一个 Process 组件 Input 的来源:或者是从此 Process 组件的前驱组件的 Output 得到,或者是从父 Process(即此 Process 组件所在的组合服务)的 Input 得到。而父 Process 的 Output 是从它的 Process 组件的 Output 得到的。所有这些指定 Input 或 Output 来源的描述,称为 Parameter Binding。我们查找有效 Input,也正是利用了这些 Binding 信息。

图 1 是一个 Process 中 Binding 信息的实例,表示一个组合服务 S 的 Output O.1 的 Output Binding 及相关信息。S 由两个服务 S1 和 S2 用 Sequence 结构组合而成。S1 由 S11 和 S12 用 Split 结构组合而成。S11、S12 和 S2 是原子服务,S1 和 S 是组合服务。图 1 中的 I.1、I.2、I.3 是 S 的三个 Input,O.1、O.2 是 S 的 Output,类似的,I1.1 表示 S1 的 Input,O12.2 表示 S12 的 Output。图中的实线箭头表示 Binding,指出 O.1 由 O2.1 得到,I2.1 由 O1.1 和 O1.2 得到等信息。

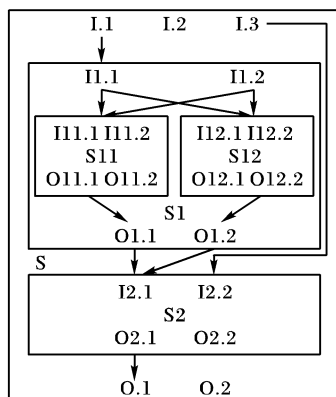


图 1 组合服务 Binding 实例

表 3 实验 economy/ car_price_service. owls 部分结果

Relevance set	Cos	FJ	JS	TUB Degree	Process Oriented Degree
Car_price_service. owls	0.98	1.0	1.0	Exact	1.0
...					
Car_recommendedprice_service. owls	0.94	0.89	0.94	Plugin	0.8
...					
Car_recommendedpriceindollar_service. owls	0.92	0.84	0.9	Subsumes	0.8
...					
4wheeledcar_price_service. owls	0.97	0.94	0.97	Failed	0.3
...					

从图 1 中可以发现,这些 Binding 关系组成一个以 O.1 为根的树型结构,这样求某个 Output 的有效 Input 集合,就成为通过遍历求所有叶节点的问题。在图 1 中,叶节点是 I.1 和 I.3,因此 O.1 的有效 Input 集合就是 I.1 和 I.3。

有效 Input 查找的核心算法为:

```

leaves = { All Inputs of S } { consts}
resultInputs ValidInput( Process S, Output O)
{
  InitQueue( que)
  Enqueue( O)
  while not Empty( que)
  {
    p = DeQueue( que)
    if p in leaves
    {
      if p is Input of S
        add p to resultInputs list
    }
    else
    {
      forall parameter p' that p BindingFrom
        EnQueue( p')
    }
  }
  return resultInputs list
}

```

4 实验分析

我们实现了一个原型系统,系统的组件结构如图 2 所示。其中服务需求与现有服务都是使用 OWL-s 描述。

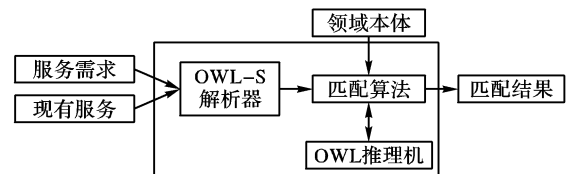


图 2 系统组件结构

在原型系统的基础上,我们使用 OWLS-TC 标准数据集来进行实验,比较新算法和 TUB 等原有算法的查全率。OWLS-TC 数据集提供了 28 个服务需求,对于每个服务需求,给出了 10 到 15 个不等的服务,这些服务在 TREC (http://trec.nist.gov/data/reljudge_eng.html) 中是判定为相关的。

按照 TUB 的算法和改进后的基于过程分析的算法,对 28 个服务需求和相应领域的服务进行匹配,获得了实验结果。表 3 是实验结果的一部分。

表 3 中, Cos(Cosine)、EJ(Extended Jaccard)和 JS(Jensen Shannon)等信息检索匹配测量值数据直接从 OWLS-TC 数据集中获得,TUB Match Degree 列的数据是根据 TUB 的算法实现的程序获得,Process Oriented Match Degree 列是从基于过程分析的服务匹配算法原型系统中得到的。

为了比较 TUB 的算法和本文提出的算法的查全率,将

TUB 的匹配结果进行标准化处理,得到两种匹配算法在不同

的匹配阈值下返回的匹配结果数目,证明了新算法的查全率得到了提高。

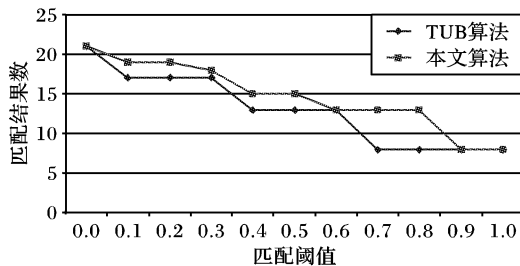


图3 查全率比较

从图3中可以看到,使用基于过程分析的服务发现算法,查全率有一定的提高,而且随着匹配阈值的降低,返回的服务数目增长曲线比较平滑,这说明用户改变匹配阈值时,可以更好地控制返回服务的数目。

对于查准率,由于目前的服务发现测试数据集中没有包含转换本体等更丰富的语义信息,仅是一些 Input 和 Output 信息,所以没有在实验部分表现出算法查准率的变化,但上面通过实例验证了算法的正确性,从理论上分析了如何保证查准率。

5 结语

在分布式电子商务环境中,如何更有效地发现服务供应者与服务需求者,也就是说如何具有更高查全率与查准率的 Web 服务发现是研究的中心问题。虽然 TUB, CMU 等机构都在这方面进行了研究工作,但仍存在着一些不足。基于过程分析的服务发现,在确保查准率的前提下,利用部分 Output 匹配情况下对过程进行分析的方法,提高了查全率。本文还

对新的服务发现算法与现有其他算法进行了比较实验,对实验结果的分析证明了新算法的有效性。

参考文献:

- [1] HAAS H. Web Services activity statement[R]. Technical report, W3C, <http://www.w3.org/2002/ws/Activity>, 2001.
- [2] MILLER E. Semantic Web activity statement[R]. Technical report, W3C, <http://www.w3.org/2001/sw/Activity>, 2003.
- [3] ANTONIOU G. A Semantic Web Primer (Cooperative Information Systems) [M]. MIT Press, 2003.
- [4] MARTIN D. OWL-S: Semantic Markup for Web Services[R]. Technical report, W3C, <http://www.w3.org/Submission/OWL-S/>, 2004.
- [4] BELLWOOD T. UDDI Version 2.04 API Specification[R]. Technical report, Oasis, <http://uddi.org/pubs/ProgrammersAPI-V2.04-Published-20020719.htm>, 2002.
- [5] PAOLUCCI M, KAWAMURA T, PAYNE TR. Semantic matching of Web services capabilities[A]. Proceedings of the 1st International Semantic Web Conference (ISWC) [C]. Sardinia, Italia, 2002. 333 - 348.
- [6] SRINIVASAN N, PAOLUCCI M, SYCARA K. Adding OWL S to UDDI: implementation and throughput [A]. First International Workshop on Semantic Web Services and Web Process Composition [C]. San Diego, USA, 2004.
- [7] JAEGER MC, TANG S. Ranked Matching for Service Descriptions using DAML-S [A]. CAISE Workshops (3) [C]. 2004. 217 - 228.
- [8] KLEIN M, BERNSTEIN A. Searching for services on the semantic Web using process ontologies [A]. The Emerging Semantic Web - Selected papers from the first Semantic Web Working Symposium [C]. Amsterdam, 2002. 159 - 172.

(上接第 2788 页)

算法,节点从第 2020s 开始死亡,1100 多秒后,所有节点死亡。采用本算法,第一个死亡节点出现的时间推迟到了第 3750s,到所有节点死亡仅用了 240s。本算法将网络 FND 向后推迟的同时,所有节点接近同时死亡。这是因为簇头的均匀分布可以避免各节点与簇头之间距离差异而引起的耗能的差距。选取簇头时,依据节点的剩余能量水平,避免了簇头节点由于耗能较大而能量消耗过快,可能过早死亡的情况,同时对于每轮中剩余能量最小的节点,由于会距离簇头节点较近,传输耗能受到了“保护”,从而也利于节点能耗均衡。

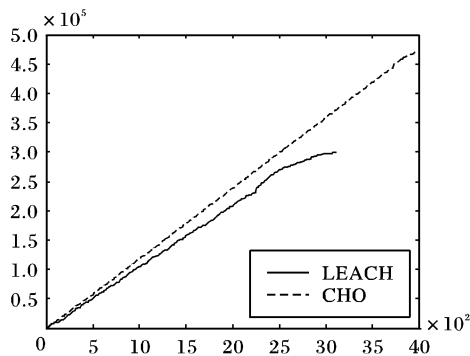


图5 运行时间与接收数据量

仿真时间与基站收到的总数据量之间的关系如图5所示。从图中可以看出,本算法基站接收的数据量要比 LEACH 多。虽然在每一轮基站不会收到更多的数据,但是因为本算法延长了网络生存时间,所以基站就可以有更多的时间去接收数据。从而提高了整个网络传输的数据量和网络的效率。

本算法在分簇时避免了簇头过于集中,同时在选取簇头时“保护”了能量最低点,从而保证了节点能耗更为均衡,延长了网络生存寿命,同时也提高了数据传输量。

参考文献:

- [1] HEINZELMAN WB, CHANDRAKASAN AP, BALAKRISHNAN H. An application - specific protocol architecture for wireless microsensor networks Wireless Communications [J]. IEEE Transactions, 2002, 1(4): 660 - 670.
- [2] HEINZELMAN WR, CHANDRAKASAN A, BALAKRISHNAN H. Energy-Efficient Communication Protocol for Wireless Microsensor [A]. Proceeding of the 33rd Hawaii International Conference on System Sciences [C]. 2000.
- [3] MHATRE V, ROSENBERG C. Homogeneous vs heterogeneous clustered sensor networks: a comparative study [A]. 2004 IEEE International Conference on Communications [C]. 2004, Vol 6: 3646 - 3651.
- [4] YONIS O. HEED: A Hybrid, energy-efficient, distributed clustering approach for ad - hoc sensor network [J]. IEEE Trans on Mobile Computing, 2004, 3(4): 366 - 379.
- [5] LINDSEY S, RAGHAVENDRA CS. Pegasus: Power efficient gathering in sensor information systems [A]. Proceedings of IEEE Aerospace Conference [C]. Montana, USA: IEEE Computer Society, 2002. 23 - 29.
- [6] 龚海刚, 刘明, 陈力军, 等. DEED: 一种无线传感器网络中高效节能的数据通信协议 [J]. 电子学报, 2005, 8(33): 1391 - 1396.
- [7] 孙利民. 无线传感器网络 [M]. 北京, 清华大学出版社, 2005.