

一种基于约束条件的服务能力匹配方法

白东伟^{1,2}, 余永林², 陈俊亮¹

(1. 北京邮电大学网络与交换技术国家重点实验室, 北京 100876; 2. 空军工程大学电讯工程学院, 陕西西安 710077)

摘 要: 结合产生式系统中冲突消解和约束满足问题的方法, 提出了一种新的基于约束条件的服务能力匹配方法。该方法把服务的约束条件分成两大类, 一类是状态关系条件, 一类是数值关系条件, 根据这两类约束条件分别提出了相应的语义匹配方法, 给出了实现原型的软件系统结构和例子, 说明了该方法的有效性。

关键词: 约束条件; 服务能力匹配; 约束满足问题

中图分类号: TP301 **文献标识码:** A **文章编号:** 1009-3516(2007)02-0071-04

文献[1]提出了一种基于 OWL-S 的 Web 服务能力的匹配方法, 主要是针对服务的输入和输出进行服务能力的匹配, 也涉及到了服务能力约束条件的匹配, 并利用 θ 包含的方法进行约束条件包含关系的推理, 但是该方法的约束条件主要是针对输入和输出参数的, 而不是服务的前置条件和效果。文献[2]主要是针对语义层次上 Web 服务之间的可组合性问题, 提出了一个多层次的语义 Web 服务组合模型, 并把 Web 服务的前置条件和效果的匹配作为其中的一个动态语义描述层次, 但文章对该问题进行了简化, 把前置条件和效果的描述局限在数值大小比较的范围内, 通过定义有限的数值比较规则来判断两个服务描述之间的兼容性。文献[3]提出了一种基于软件组件描述规范的匹配方法, 文章采用定理证明的方法进行条件之间 θ 包含关系的推理, 但是主要针对的是组件领域的组件重用、检索和替换等问题。文献[4]针对产生式规则系统中的规则冲突消解问题, 提出了一种框架逻辑和规则系统相结合的推理方法, 来进行语义模式的匹配和分类。

本文针对基于 OWL-S 的前置条件和效果匹配问题, 把前置条件和效果的约束条件分成两大类: 一类是数值约束条件, 一类是状态约束条件。根据这两类约束条件分别提出了匹配方法, 即把数值条件的匹配转化为规则推理, 而把状态条件匹配转化为规则系统中的冲突消解和约束满足问题。在匹配的过程中, 充分利用了领域本体的知识, 来进行约束条件之间的语义匹配, 可以增强服务能力匹配和选择的准确性与灵活性。

1 约束条件匹配方法

根据 OWL-S 1.1 规范, 前置条件和效果的约束条件由一些合取形式的 Atoms 组成, 并采用 SWRL 语言进行描述。这些 Atoms 都是来自于 OWL 本体中定义类或属性。虽然 OWL 善于描述具有分类层次结构的概念和属性, 但是在服务能力的约束条件中经常会涉及到时间、价格、距离等约束条件, 需要采取数值约束条件的描述, 但是这类数值约束条件很难利用 OWL 的包含关系推理来进行数值约束条件的一致性判断。我们根据 WS 服务能力约束条件描述的具体需求, 把前置条件和效果中的约束条件分成两类, 一类是利用本体层次概念描述的状态约束条件, 主要利用领域本体中的概念和属性来描述服务的外部状态和关系约束; 另一类是数值约束条件, 主要对外部数值条件或者输入输出参数的数值进行约束。

1.1 数值约束条件的判定

收稿日期: 2006-06-20

基金项目: 国家自然科学基金资助项目(60432010)

作者简介: 白东伟(1974-), 男, 河南长葛人, 博士生, 主要从事语义 Web 服务技术研究;

陈俊亮(1933-), 男, 浙江宁波人, 中国科学院、中国工程院院士, 主要从事通信网、网络智能等研究。

定义数值约束条件的形式如下:

$$\text{NumberCondition} = \{\text{NumberTerm1}, \text{NumberTerm2}, \dots, \text{NumberTerm } i\}$$

其中 NumberTerm 为 $r(x, v)$ 形式的二元谓词逻辑形式, x 是参数变量的名字, 这个名字可以是 OWL 本体中定义类或者变量; v 是具体的数值; r 是相等 (Equal)、大于等于 (LargeOrEqual)、小于等于 (LessOrEqual)、范围内 (RangeIn) 4 种数值约束关系谓词, 当然可以根据需要进行描述谓词的扩充。这 4 种数值约束条件描述谓词对应的推理规则的定义分别如下:

Equal - Consistent : $\text{Equal}(\text{Provider}, ?x), \text{Equal}(\text{Requester}, ?y), \text{Equal}(?x, ?y) \Rightarrow \text{Equal - Consistent}$

LessOrEqual - Consistent : $\text{LessOrEqual}(\text{Provider}, ?x), \text{LessOrEqual}(\text{Requester}, ?y), \text{LessOrEqual}(?y, ?x) \Rightarrow \text{LessOrEqual - Consistent}$

? LargeOrEqual - Consistent : $\text{LargeOrEqual}(\text{Provider}, ?x), \text{LargeOrEqual}(\text{Requester}, ?y), \text{LargeOrEqual}(?y, ?x) \Rightarrow \text{LargeOrEqual - Consistent}$

? RangeIn - Consistent : $\text{RangeIn}(\text{Provider}, ?x, ?y), \text{RangeIn}(\text{Requester}, ?z, ?w), \text{LessOrEqual}(?z, ?x), \text{LargeOrEqual}(?w, ?y) \Rightarrow \text{RangeIn - Consistent}$

数值条件的兼容性可以通过上述的规则进行推理判断。具体过程如下: 首先根据 NumberCondition - Provider 条件中的每个 NumberTerm 找出 NumberCondition - Request 中具有相同 x 和 r 的 numberTerm, 然后利用上面定义的判断规则进行推理, 得出数值条件是否兼容的结论。

1.2 状态约束条件的判定

Web 服务能力状态约束条件的匹配就是合取形式的子句之间蕴涵关系的判定, 目前大多采用 θ 包含方法来进行条件子句之间的蕴涵判断, 但是 θ 包含不能直接进行语义层次的约束条件匹配, 如状态条件 NormalMember(? Provider) 和 GoldMember(? Requester), 采用 $\theta = (? \text{Provider}/? \text{Requester})$ 的变量置换后, 并不能判断这两个条件的蕴涵关系, 只有利用 OWL 本体中的领域知识 GoldMember 是 NormalMember 的子类, 才能判断出这两个条件之间的蕴涵关系, 得到服务请求者可以满足服务提供者状态条件要求的结论。

为了判定两个状态约束条件子句之间是否存在蕴涵关系, 我们主要借鉴了文献[4]中关于软件组件间匹配的一些定义和方法。

定义 1: 假定 P_1 和 P_2 是两个合取形式的子句, 子句中的文字来自本体库 T 中的类或者属性, 则 P_1 包含 P_2 , 记为 $P_1 \supseteq P_2$, 当且仅当存在一个变量置换 θ , 即把 P_1 中的变量置换成 P_2 中的变量或者常量, 使得在本体 T 中, P_2 蕴涵 $P_1\theta$, 如下:

$$P_1 \supseteq P_2 \text{ if } \exists \theta \text{ such that } P_1 \Rightarrow P_2\theta$$

这个定义和归纳逻辑程序设计中的 θ 包含的定义类似, 不同的是我们不是采用集合包含的方法, 而是利用了 OWL 的术语层次知识, 进行子句中各文字的蕴涵判断, 也就是语义层次的判断, 而不只是语法层次的比较。状态约束条件匹配算法主要分 3 步:

第 1 步: 进行约束条件子句的规格化, 就是把条件表达式中的蕴涵语义信息挖掘出来。

第 2 步: 降低约束条件中合取子句谓词间变量置换和匹配的复杂性, 把这个问题转化成约束满足问题, 并利用领域本体中术语分类层次知识, 减少变量置换的搜索范围。

第 3 步: 采用约束满足算法, 在一元谓词形成的值域内, 搜索满足所有二元谓词形成的约束的变量置换对, 然后再利用本体推理机进行子句文字之间的包含推理。

针对匹配算法的第 1 步, 采用了如下的规格化定义:

定义 2 说条件模式 P 是规格化的了, 当且仅当:

$$\forall l_i \text{ if } \xrightarrow{T} l_i \text{ then } \exists l' \text{ in } P \text{ such that } l' \xrightarrow{T} l \text{ 其中 } l \text{ 和 } l' \text{ 是具有相同数量参数的文字。}$$

目前, 主要针对 OWL - DL 语言, 制订了如下 3 种具体的规格化类型:

1) 二元属性的域和范围推理

利用二元属性的域和范围的定义推导出隐含的一元条件类, 例如根据图 1 的 CreditCard 本体, 从 Has - Gold - Card(? x , ? y), 应该可以推出 Person(? x) 和 Gold - CreditCard(? y) 的隐含信息。

2) 一元条件的规格化

把具有变量名相同的一元谓词通过采用合取的形式,形成一个新的谓词,如 $\text{Femal}(\ ?\ x)$ 和 $\text{Student}(\ ?\ x)$ 可以形成 $\text{Femal} - \text{Student}(\ ?\ x)$, 这样条件子句中的每个变量只有一个谓词,可简化匹配搜索。

3) 二元条件的规格化

把具有相同参数的二元属性条件进行组合形成一个复合属性,并同时

考虑其域和范围的一元条件。这样确保子句条件中相同的一对变量之间只有一种属性。例如根据图1的 CreditCard 本体,从 $\text{Has} - \text{Card}(\ ?\ z, \ ?\ w) \wedge \text{Gold} - \text{CreditCard}(\ ?\ w)$, 应该能得到 $\text{Has} - \text{Gold} - \text{Card}(\ ?\ z, \ ?\ w) \wedge \text{Gold} - \text{CreditCard}(\ ?\ w)$ 。

采用如下定理1作为规格化后的约束条件子句的蕴含关系判定准则。

定理1 假定 P_1 和 P_2 是两个规格化后的合取子句模式: $p_1 = l_1^1 \wedge l_2^1 \wedge \dots \wedge l_n^1$; $p_2 = l_1^2 \wedge l_2^2 \wedge \dots \wedge l_m^2$

则 P_1 包含 P_2 , 当且仅当存在一个置换 θ , 使得 P_1 中的每一个文字 l_i^1 都包含至少一个 P_2 中具有相同变量的文字, 即: $P_1 \supseteq P_2 \Leftrightarrow \exists \theta [\forall l_i^1 \text{ in } p_1; \exists l_j^2 \text{ in } p_2; \text{such that } l_j^2 \xrightarrow{\theta} l_i^1]$, l_i^1 和 l_j^2 是具有相同变量的文字。

为降低变量置换搜索的复杂性,我们再次利用 OWL 本体的术语层次知识,把两条件子句中具有包含关系的一元谓词的变量对作为置换的值域,而把具有包含关系的二元谓词的变量对作为置换的约束。这样变量置换的过程可以转化成满足所有二元谓词形成的置换约束的一元谓词之间的变量置换。例如,有条件子句 $\text{Person}(\ ?\ x) \wedge \text{Has} - \text{Gold} - \text{Card}(\ ?\ x, \ ?\ y)$ 和条件子句 $\text{Beijing} - \text{Person}(\ ?\ z) \wedge \text{Has} - \text{Gold} - \text{Card}(\ ?\ z, \ ?\ w) \wedge \text{Shanghai} - \text{Person}(\ ?\ c)$, 因为 Person 包含 $\text{Beijing} - \text{Person}$ 和 $\text{Shanghai} - \text{Person}$, 因此潜在的对 x 的变量置换可以是 z 和 c , 但是 $\text{Has} - \text{Gold} - \text{Card}$ 二元属性的存在,变量置换 $(\ ?\ x / \ ?\ z, \ ?\ y / \ ?\ w)$ 作为约束,则变量 x 应置换为变量 z 。经过变量置换后,就可以根据两条件子句中变量相同的一元和二元谓词之间的包含关系来判定条件子句的包含关系了。

2 原型系统结构和例子

根据文中提出的方法思想,进行了系统原型的实现,并进行了相关例子的测试。系统原型结构如图2所示。Matchmaking Engine 是基于 CommonRuler 和 Racer 推理机的,其中 CommonRuler 负责数值约束条件的匹配, Racer 负责进行 OWL 本体中类和属性层次关系的推理。OWL-S API 1.1 工具包对服务描述进行处理,主要是获取服务描述中的约束条件描述,并把数值条件和状态条件分别送给相应的推理机。Protégé-OWL API 主要负责对 OWL 本体知识库进行处理,其中包含领域术语层次知识和 SWRL 语言描述的数值约束条件推理规则。Protégé-OWL API 把数值条件推理规则送给 CommonRuler 进行数值条件的推理,把领域术语知识送给 Racer 进行语义推理。

图3是原型系统实现的进行服务能力约束条件匹配的一个例子。图3(a)为服务请求者和提供者的约束条件描述。其中 $\text{LessOrEqual}(\text{Rank}, 5)$ 和 $\text{LessOrEqual}(\text{Rank}, 4)$ 分别为服务双方的数值条件约束,通过定义的数值条件推理规则第2条, CommonRuler 可以直接判定双方的数值条件兼容。而对于其它的状态约束条件,直接使用 θ 包含的方法是无法判断出两个条件子句之间的包含关系的;图3(b)为进行规格化后的状态条件描述。其中 ServiceRequester 的条件描述 $\text{Person}(\ ?\ x)$ 是通过规格化从属性 $\text{Has} - \text{Gold} - \text{Card}$ 的 Domain 限制中得出的隐含信息,而 ServiceProvider 的条件 $\text{Has} - \text{Card}$ 也由于 $\text{Gold} - \text{CreditCard}(\ ?\ w)$ 的限制而

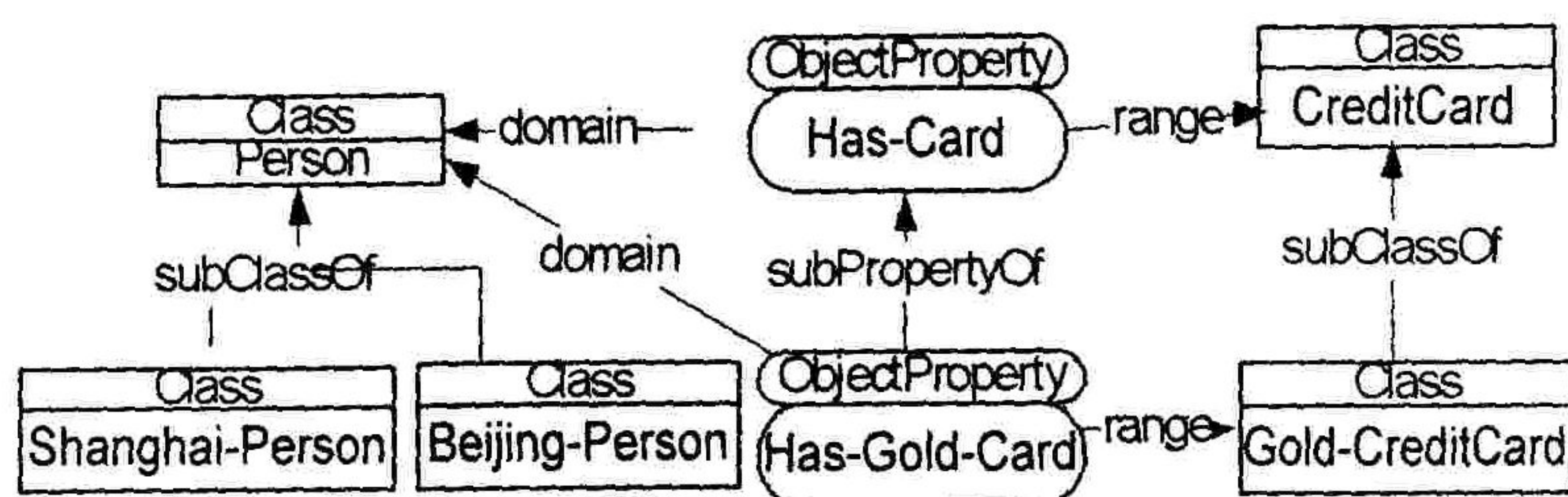


图1 CreditCard 本体图

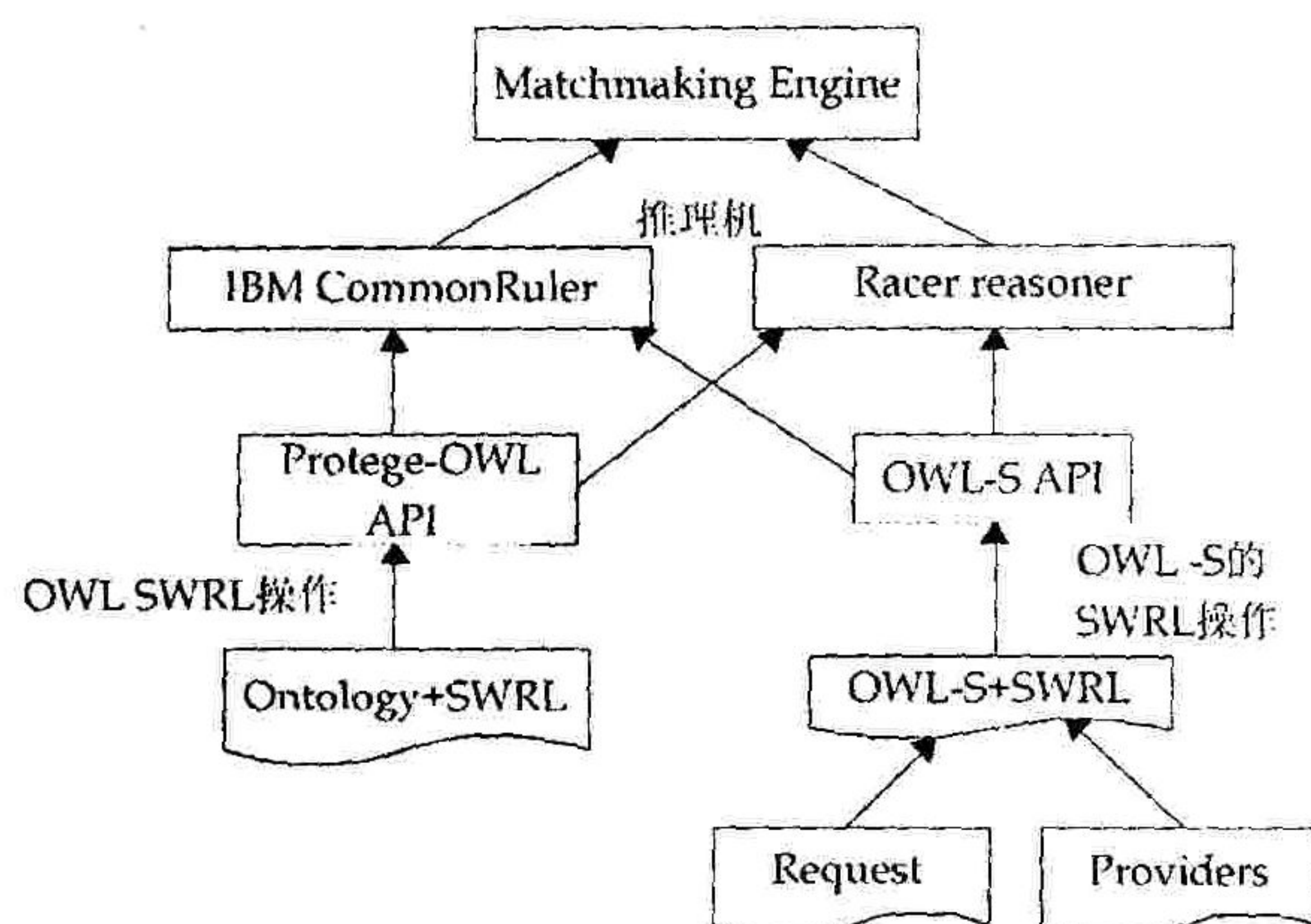


图2 系统原型结构图

转化成了 Has - Gold - Card; 图 3(c) 则是进行了变量置换后的状态条件描述, 这种形式就很容易判断出服务提供者的条件可以被服务请求者所满足。

Service Provider: LessOrEqual(Rank, 5) Has - Gold - Card(? x, ? y) Gold - CreditCard(? y) Service Requester: LessOrEqual(Rank, 4) Beijing - Person(? z) Has - Card(? z, ? w) Gold - CreditCard(? w) ShangHai - Person(? c)	Service Provider: Person(? x) Has - Gold - Card(? x, ? y) Gold - CreditCard(? y) Service Requester: Beijing - Person(? z) Has - Gold - Card(? z, ? w) Gold - CreditCard(? w) ShangHai - Person(? c)	Service Provider: Person(? z) Has - Gold - Card(? z, ? w) Gold - CreditCard(? w) Service Requester: Beijing - Person(? z) Has - Gold - Card(? z, ? w) Gold - CreditCard(? w) ShangHai - Person(? c)
(a)	(b)	(c)

图 3 基于约束条件的 Web 服务匹配例子

3 结束语

本文通过对 OWL 和 SWRL 语言的分析, 把服务能力约束条件分成了两类, 一类是数值关系约束条件, 一类是状态关系约束条件, 并提出了一种新的基于约束条件的服务能力匹配方法。实现的原型系统和例子, 说明了该方法的有效性。下一步的工作主要是对利用 OWL 本体描述的约束条件所隐含信息的发现进一步研究, 将主要针对 Property 具有的特点, 如传递性、对称性、函数性、自反性等, 从中发现隐含的语义信息。

参考文献:

- [1] Kawamura T, De Blasio J A, Hasegawa T, et al. Preliminary Report of Public Experiment of Semantic Service Matchmaker with UDDI Business Registry[A]. ICSOC 2003[C]. 2003, 208 - 224.
- [2] Medjahed B, Bouguettaya A. A multilevel composability model for semantic Web services[J]. Knowledge and Data Engineering, IEEE Transactions, 2005, 17(7): 954 - 968.
- [3] Zaremski A M, Wing J M. Specification Matching of Software Components[J]. ACM Transactions on Software Engineering and Methodology, 1997, 6(4): 333 - 369.
- [4] Yen J, Neches R, MacGregor R. CLASP: Integrating Term Subsumption Systems and Production Systems. Knowledge and Data Engineering[J]. IEEE Transactions, 1991, 3(1): 25 - 32.
- [5] Jun - jang J, Betty C. Using Automated Reasoning Techniques to Determine Software Reuse[J]. International Journal of Software Engineering and Knowledge Engineering, 1992, 2(4): 523 - 546.
- [6] Martin O C, Holger K, Samson T, et al. Supporting Rule System Interoperability on the Semantic Web with SWRL[A]. Fourth International Semantic Web Conference[C]. Galway: 2005.

(编辑: 门向生)

A Web Services Capability Matchmaking Method Based on Constraint Conditions

BAI Dong - wei^{1,2}, YU Yong - lin², CHEN Jun - liang¹

(1. State Key Laboratory of Networking and Switching Technology, Beijing University of Posts and Telecommunications, Beijing 100876, China; 2. The Telecommunication Engineering Institute, Air Force Engineering University, Xi'an 710077, Shaanxi, China)

Abstract : This paper presents a new web services capability matchmaking method based on constraint conditions, which integrates rule system and semantic matchmaking. The constraint conditions are categorized into two kinds: status - conditions and number - conditions. Two matchmaking methods are proposed separately according to these two kinds constraint conditions. This paper also proposes the software architecture of the prototype and examples to show that the proposed approach is practical and effective.

Key words: constraint condition; service capability matchmaking; constraint satisfaction problem