一种面向语义 Web 浏览器的基于规则的推理方法

龚赛赛,葛唯益

(东南大学 计算机科学与工程学院,江苏 南京 210096)

[摘要] 讨论了语义 Web 浏览器环境下 RDF 数据推理与查询面临的若干挑战:动态性、可伸缩性、及时性和可信性. 提出了一种基于前向链的规则推理方法,通过动态提取与查询相关的规则和事实来处理动态性和可伸缩性挑战. 该方法通过 Magic Sets 优化技术提高推理性能来满足及时性要求. 提取的规则包含了本体公理转化而来的规则. 为了提高推理结果的可信性,转化本体公理时考虑了数据源的权威性,并提供推理结论的证据解释. 实验评估显示了该方法在语义 Web 浏览器中的有效性.

[关键词] 基于规则的本体推理,语义 Web 浏览器,权威文档

[中图分类号] TP311 [文献标志码] A [文章编号] 1672-1292(2011)04-0040-07

A Reasoning Approach to Rule Based Reasoning for Semantic Web Browsers

Gong Saisai, Ge Weiyi

(School of Computer Science and Engineering, Southeast University, Nanjing 210096, China)

Abstract: This paper discusses the challenges of performing reasoning and query answering over RDF data in the context of a Semantic Web browser, including dynamics, scalability, prompt response and trustworthiness. We propose a rule based reasoning mechanism by means of forward chaining, which addresses the challenges of dynamics and scalability by dynamic extraction of rules and facts that are relevant for query answering. To meet prompt response, the mechanism uses Magic Sets optimization to improve reasoning performance. Those extracted rules contain ones translated from ontologies. To improve the trustworthiness of reasoning results, authoritative analysis is performed during ontology translation, and proof explanations for each consequence are provided. An experimental evaluation shows the effectiveness of the approach in the context of a Semantic Web browser.

Key words: rule based ontology reasoning, semantic web browser, authoritative document

目前,语义 Web 已经汇集了超过几十亿条的 RDF 三元组,其范围覆盖了地理信息、社交网络、政府数据和生物医学等众多领域[1],它们构成了所谓的数据 Web(Web of Data). 通过 RDFS 和 OWL 表示的本体,这些 RDF 数据被赋予了良好定义的语义,因而应用程序有机会对它们进行推理来提供智能的服务,例如回答具有精确信息需求的形式化查询(如 SPARQL 查询). 本文的工作是出于 MyView(http://ws.nju.edu.cn/explorer/myview/)项目的需求. 作为一个语义 Web 浏览器,MyView 的一个基本功能是允许用户通过链接(以 RDF 三元组表示)导航的方式来探索语义 Web. 特别地,MyView 能够让用户查询语义 Web. 也即用户向 MyView 提交形式化查询,MyView 根据用户的查询在语义 Web 上进行推理,完成查询应答. 为了实现后者,本文提出一种面向语义 Web 浏览器的基于规则的推理方法应用于 MyView,同时该方法也适用于类似的语义 Web 浏览器.

语义 Web 存在两种主流的推理机制:一种是将 RDF 数据转化成描述逻辑中的公理和事实,采用 tableau 算法[2] 进行推理;另一种是基于逻辑编程[3] 的思想,将 RDF 数据等价转换成规则,通过传统的规则引擎完成推理. 语义 Web 上存在海量的实例数据(ABox),而术语数据(TBox)所占的比例相当小,用户浏览时更加关心这些实例数据. 相对于 tableau 算法而言,规则在大规模实例数据上的推理效率更高,可伸缩性更好,更加适合查询应答[4]. 鉴于此,本文采用基于规则的推理方法.

收稿日期:2011-08-10.
基金项目:高等学校博士点基金(20090092110023).
通讯联系人:瞿裕忠,教授,博士生导师,研究方向:Web 科学与 Web. E-mail:yzqu@nju.edu.cn

然而,在语义 Web 浏览器上下文中进行 RDF 数据推理面临如下的挑战:

(1) 动态性: 首先,语义 Web 浏览器下载的是 Web 上的 RDF 数据,它们会经常更新; 其次,Web 是个开放的数据空间,在其中进行推理所需的数据源无法预先知道,需要动态获取[5]; 再者,用户浏览时的交互操作如自定义规则,会引起浏览器知识库的变化. 也即,推理是在一个动态变化的知识库上进行.

(2) 可伸缩性: 语义 Web 浏览器中的推理是在计算能力和存储能力都受限的客户端进行,这给推理的可伸缩性提出了新的要求. 例如,大规模的 RDF 数据无法完全载入内存,需要借助外存存储部分数据,此时推理需考虑 I/O 优化.

(3) 及时性: 和传统的浏览器一样,语义 Web 浏览器也要满足及时响应的要求. 因此,RDF 数据的查询与推理必须在足够短的时间内完成,才能保证浏览器的可用性.

(4) 可信性: 一方面是数据的可信性,语义 Web 浏览器中下载的 Web 数据呈现低质量性,只有可信度高的数据才能用于推理. 另一方面是推理过程的可信性,为了让用户信任浏览器的推理结果,需要向用户解释每个推论,使得背后的推理过程透明化.

迄今为止,现有的研究工作并没有很好地应对这些问题,导致现有的基于规则的推理方法不能直接用在语义 Web 浏览器中. 为应对上述 4 个挑战,本文动态提取与查询相关的规则和事实用于求值,减少与查询应答不相关的计算,同时一定程度上减少内存开销. 本方法确保正确性,尽可能提高完整性.

1 基本概念

在本文中,U 表示 URI 集合,B 表示空白节点集合,L 表示文字集合,V 表示变量集合,$C = U \cup B \cup L$ 表示资源集合.

在语义 Web 中,现实世界的对象称为实例,实例的集合称为类,实例之间的关系称为属性. RDFS 和本体语言如 OWL 2[6] 用来显示描述类和属性之间的关系,从而能够形式化表示论域中的知识. 为方便起见,本文使用 rdf: 表示 RDF 的命名空间,rdfs: 表示 RDFS 的命名空间,owl: 表示 OWL 的命名空间,foaf: 表示 FOAF 的命名空间.

一个 RDF 三元组 $t = (s,p,o) \in (U \cup B) \times U \times C$,其中 s,p,o 分别表示该三元组的主语、谓语和宾语.

一个原子公式(atom) 具有以下的两种形式: $C(x)$ 或者 $P(x,y)$,其中 C, $P \in U \cup B$, x, $y \in C \cup V$. C 和 P 称为谓词,x 和 y 称为词项. 为方便起见,对于一个原子公式 atomf,在后续的章节中使用 pred(atomf) 表示 atomf 的谓词,terms(atomf) 表示 atomf 中词项的集合. 在本文中,一个查询是一个原子公式. 一个事实 f 是一个原子公式并且 $\forall x \in \text{terms}(f)$, $x \notin V$.

本文采用 DLP[7] 的思想,将类看作一元谓词,属性看作二元谓词. RDF 三元组 $(a, \text{rdf:type}, c)$ 可以转化成事实 $c('a')$,RDF 三元组 (a, p, b) (其中 $p \neq \text{rdf:type}$) 可以转化为事实 $p('a','b')$,反方向亦然.

本文中每个规则均具有如下的形式: $H: -B_1,\cdots,B_n$. 其中,H 称为规则的头(也称为后件),$B_1,\cdots,B_n$ 称为规则的体(也称为前件),H,B_i 是原子公式. 整个规则读作"如果'体',那么'头'". 值得注意的是,本文中规则的表达能力在 definite logic programming[7] 之内,满足 range restricted 条件即规则头中包含的变量必须在体中出现.

一个知识库 $KB = (RS,FS)$ 由规则集合 RS 和事实集合 FS 组成.

对于原始的事实,其证据(proof) 就是包含它的一个数据源; 对于推理得到的事实,其证据 proof = (r, seq),其中 r 是推导出这个结论使用的规则,seq 是对应的事实序列. 例如,通过规则 $p(?x,?z) :- q(?x,?y)$, $t(?y,?z)$ (记为 r) 和事实序列 $\langle q('a','b'), t('b','c')\rangle$(记为 seq) 可以推导出事实 $p('a','c')$,那么 $p('a','c')$ 的证据是 (r,seq).

2 方法概述

本文假定存在一个知识库,它所包含的事实和规则在动态变化且部分数据存储在外存上. 给定这样一个知识库,本文基于前向链的推理方法,以一个查询作为输入,将满足该查询的事实集合(包括原始的和推理的) 及其相关的证据作为输出. 整个方法分为 5 个步骤,其体系结构如图 1 所示.

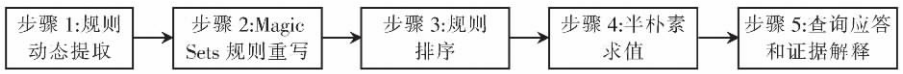

图1　推理方法的体系结构

Fig.1　Architecture of the reasoning method

2.1　规则动态提取

知识库中的规则包含两部分:用户制定的规则和本体公理转化而来的规则.规则动态提取意味着每次推理时根据查询从最新版本的知识库中提取规则,这减少了使用失效的规则进行推理的可能性,有利于应对动态性挑战.随着浏览器长时间使用,尤其是遇到包含大量公理的本体(如基因本体)后,其知识库规则数目会越来越大.由于客户端计算资源的局限性,可能无法将知识库所有规则载入内存.因此,每次推理时必须动态提取与查询相关的规则,减少内存消耗,提高可伸缩性.此外,语义Web是个开放的、不一致的知识库,为了提高数据的质量,在将本体公理转成规则时,本文基于权威文档的概念进行了数据可信度分析.

2.2　*Magic Sets* 规则重写

得到相关规则后,直接使用前向链进行推理效率将很低,因为它需要推导出所有的结论,然后才去回答查询.鉴于此,为了满足及时性要求,本文在前向链基础上使用了Magic Sets[8]规则重写优化技术,减少与查询应答无关的计算.

2.3　规则排序

虽然逻辑编程是声明式的即规则之间没有顺序,但若在规则求值之前按照规则之间的依赖关系对规则进行排序,则会减少求值时的迭代次数,缩短推理时间[9].

2.4　半朴素求值

前向链采用半朴素算法[10](semi-naïve evaluation)进行规则求值.如同规则一样,事实难以全部载入内存.因此,在半朴素求值过程中,事实是根据当前规则的求值情况动态装入内存.

2.5　查询应答和证据解释

语义Web是一个开放的、异构的、存在冲突的知识库,一个智能的浏览器有必要向用户提供相关的证据解释,让背后的推理过程透明化.文献[11]指出,一个好的证据解释应该是可理解的和可跟踪的.这里,由于使用了Magic Sets重写技术,直接的论证跟踪得到的证据很难让普通用户理解.因此必须对原始的证据进行解释,还原到重写前的规则,并提供每个规则和事实的来源.

3　规则动态提取

目前,本文在进行本体推理时只考虑RDFS以及owl:equivalentProperty和owl:equivalentClass的单方向语义,如表1所示.值得注意的是,(P,owl:equivalentProperty,Q)根据其语义可以转成两个规则:$Q(?x,?y):-P(?x,?y)$和$P(?x,?y):-Q(?x,?y)$.而后者会引起本体劫持问题[12],即在非权威文档中重新定义类或属性的含义,因此本文只支持前者,也即只考虑owl:equivalentProperty的单方向语义.对于owl:equivalentClass同样如此.rdfs:subClassOf和rdfs:subPropertyOf的传递性在前向链推理时自然得到保证,故不再引入其传递性规则.

表1　支持的本体推理规则

Table 1　Ontology translation rules supported

RDF三元组	规则
(C, rdfs:subClassOf, D)	$D(?x):-C(?x)$
(P, rdfs:subPropertyOf, Q)	$Q(?x,?y):-P(?x,?y)$
(P, rdfs:domain, C)	$C(?x):-P(?x,?y)$
(P, rdfs:range, C)	$C(?y):-P(?x,?y)$
(P, owl:equivalentProperty, Q)	$Q(?x,?y):-P(?x,?y)$
(C, owl:equivalentClass, D)	$D(?x):-C(?x)$

类似传统Web,语义Web中许多数据是不可信的.为了提高数据的质量,在将本体公理转化成规则时需要考虑该公理是否可信.

定义1　一个空白节点或URI的权威文档如下定义:

(1)对于一个URI,通过HTTP协议的get方法,将请求头中的accept域设定为"application/rdf+xml"用引(dereference)该URI,如果能够得到一个包含该URI的RDF/XML文档,即该URI出现在该文档的一个三元组中,则称此文档为该URI的权威文档.

(2) 对于一个空白节点,它的权威文档就是包含它的 RDF 文档.

基于权威文档的定义,一个表示本体公理的三元组(s,p,o) 是可信的当且仅当这条三元组出现在 s 的权威文档中. 在进行本体推理时,只考虑这些可信的公理转化而来的规则.

令 r 表示一个规则,hPred(r) 表示 r 的头谓词(头中原子公式的谓词),BPred(r) 表示出现在 r 的体中的谓词的集合,RS 表示知识库中的规则集合.

定义 2 一个谓词依赖图 PDG = (V,E,l_E) 是一个边带标记的有向图,其中:

$V = \{p \mid \exists r \in \mathrm{RS}, p \in \{\mathrm{hPred}(r)\} \cup \mathrm{BPred}(r)\}$;

$E = \{\langle p,q\rangle \mid \exists r \in \mathrm{RS}, p = \mathrm{hPred}(r), q \in \mathrm{BPred}(r)\}$;

$l_E(\langle p,q\rangle) = \{r \mid r \in \mathrm{RS}, p = \mathrm{hPred}(r), q \in \mathrm{BPred}(r)\}$.

规则 r_a 依赖于规则 r_b 当且仅当在谓词依赖图中存在边 $\langle \mathrm{hPred}(r_a), \mathrm{hPred}(r_b)\rangle$. 根据谓词依赖图,可以对规则进行排序. 算法 1 给出了基于查询,动态提取规则,完成 Magic Sets 重写和排序的过程.

算法 1 动态提取规则

输入: 一个知识库 KB,一个查询 q.

输出: 用于查询应答的规则序列.

```
1  创建集合 visited,初始时包含元素 pred(q)
2  创建集合 todo,初始时包含元素 pred(q)
3  创建集合 rules,初始为空
4  while todo 非空
5      p ← todo 中的一个元素
6      将 p 从 todo 删除
7      set_p ← 知识库中头谓词为 p 的规则集合
8      for set_p 中每个规则 r
9          rules ← rules ∪ {r}
10         for 每个谓词 predicate ∈ BPred(r)
11             if predicate 不在 visited 中
12                 visited ← visited ∪ {predicate}
13                 todo ← todo ∪ {predicate}
14             end if
15         end for
16     end for
17 end while
18 /* Magic Sets 规则重写 */
19 rulesRewriting ← rewrite(q, rules)
20 /* 构建谓词依赖图 */
21 PDG ← bulid (rulesRewriting)
22 /* 规则排序 */
23 rulesReorder ← sort(q, PDG, rulesRewriting)
24 返回 rulesReorder
```

算法 1 的 4 到 17 行是每次推理时从知识库动态提取规则的过程. 这种动态提取方法不要求将所有规则全部载入内存,这意味着可以在外存存储规则,推理时只提取与查询相关的规则,从而减少内存消耗.

值得注意的是算法的第 7 行是向知识库请求一个谓词关联的规则. 如果知识库将部分规则存储在外存上,那么额外的 I/O 开销会影响整个推理所需的时间. 再从知识库动态提取事实时,I/O 操作的影响会更大.

算法 1 第 7 行在提取本体公理转化而来的规则时,需要首先确定那些本体公理对应三元组的范围. 对于本文中的 RDFS 推理,每个公理对应一条三元组. 鉴于此,对于一个查询,只需要检查哪些三元组对应的规则头谓词与查询相同. 例如,查询以一个类 C 为头谓词的规则,需要考察的三元组模式如下所示:

(1) (?D, rdfs: subClassOf, C),

(2) (?D, owl: equivalentClass, C),

(3) (?P, rdfs: domain, C),

(4) (?P, rdfs: range, C).

算法 1 第 19 行 Magic Sets 规则重写技术的具体实现参考文献[8],此处不再详述,仅以例 1 说明其原理,其中 myView: 是一个命名空间.

例 1 假定有规则 myView: friendEmail(?x,?y) :- foaf: knows(?x,?z), foaf: mbox(?z,?y) 和查询 myView: friendEmail('myView: me',?x),最终的规则重写结果包含: 事实(Seed) myView: magicfriendEmail('myView: me') 和规则 myView: friendEmail(?x,?y) :- myView: magicfriendEmail(?x), foaf: knows(?x,?z), foaf: mbox(?z,?y).

Magic Sets 重写规则时,会在原始的规则体中加入一个新的原子公式,它的谓词是所谓的 Magic Predicate[8](如例 1 中的 myView: magicfriendEmail),它起到了"门卫"的作用,减少与查询回答无关的计

算. 例如,例1重写后的规则求值时变量 x 的取值只能是 myView: me.

算法1第23行首先对谓词依赖图进行强连通分支压缩,然后根据新图包含的偏序关系对谓词构成的强连通分支进行拓扑排序,最后计算以这些谓词为头谓词的规则的排序.

4 查询应答和证据解释

在逻辑编程中,前向链推理算法基于不动点语义反复应用规则进行求值,直至没有新的事实可以推导出来. 半朴素算法要求第 i 轮求值时必须用到第 $i-1$ 轮的结论[13].

如上所述,在回答每个查询时,从知识库中动态地提取相关的规则集合. 在半朴素求值过程中,采取类似的做法,即根据规则的求值情况动态地从知识库中提取事实. 在例1中,通过对第一个原子公式 myView: magicfriendEmail(?x) 求值,得到变量 x 取值为 myView: me. 当考察满足第二个条件即原子公式 foaf: knows(?x,?z) 的事实时,只需要向知识库提取与模式 foaf: knows(´myView: me´,?z) 匹配的事实即可,而不是装载所有以 foaf: knows 作为谓词的事实. 这种做法在一定程度上可以节省内存消耗,减少I/O开销,尤其是在某些谓词很流行的情况下(比如社交网络中存在很大规模的RDF数据描述 foaf: knows 关系). 总之,这种动态提取规则和事实的推理方式满足及时响应的要求,有利于应对推理的可伸缩性问题.

由于浏览器有必要让推理过程透明化,而直接的论证追踪得到的证据并不能让普通用户理解,需要对其进行解释. 在这样的原则下,本文首先在提取规则时记录每个规则的来源,在半朴素求值过程中对从知识库提取的原始事实记录其所在数据源. 每当推导出一个新的结论时,记录这个推导过程使用的规则及其相关的事实序列. 推理完成后,对这些原始的证据进行修正. 如例1中,Magic Sets 重写后的规则并非用户原始定义的. 修正一个推论的原始证据的做法如下: 对于规则,删除规则体中以 Magic Predicate 作为谓词的原子公式; 对于事实序列,将以 Magic Predicate 作为谓词的事实从序列中删除. 因为这些删除操作并没有影响规则中变量的取值,所以正确性能够得到保证.

5 实验评估

为验证语义Web浏览器环境下本文所述方法的有效性及性能,本文基于Java在MyView中实现该方法来对语义Web上的RDF数据进行查询应答,所有的评估工作在2.66 GHz,Intel Core 2 Quad Q8400,Java虚拟机最大可用内存为1.4 G的Win7上完成.

5.1 实例

本文通过执行以下4个查询来评估方法的可行性:

Q1: 找到"阿凡达"的导演的所有作品的年份;

Q2: 找到 Tim Berners-Lee 的合作者(co-author);

Q3: 找到 Tim Berners-Lee 认识的人的兴趣;

Q4: 找到所有 ISWC2010 会议论文的作者.

其结果如表2所示. 表2给出了每个查询的结果数目,在查询期间需要从Web上下载解析的RDF文档数目,执行查询的总时间(包括文档下载和解析以及查询应答时间)和查询应答时间(不计文档下载和解析时间).

表2 查询结果统计

Table 2 Statistics of the query results

查询	Q1	Q2	Q3	Q4
结果数目	15	48	25	225
下载解析的RDF文档数目	2	90	80	89
总时间/s	15.38	25.71	29.35	34.00
查询应答时间/ms	501	516	508	530

实验结果表明,在执行查询的过程中,文档下载和解析即获取数据源的时间远大于查询应答的时间. 换而言之,由于RDF文档没有下载和解析完成,一次具体的推理结果可能不完整. 因此需要每隔一段时间在数据不断增大的知识库上重新计算查询,随着时间的推移,推理结果会更加完整.

5.2 可伸缩性

为验证本文提出的推理方法的可伸缩性,本文基于LUBM[14]生成随机数据,在这些随机数据上完成性能测试. LUBM是用来评估语义Web推理与查询方法性能的基准测试集,可通过设置参数大学数的取值来生成不同规模的测试集.

首先,本文通过设置LUBM大学数的取值为1到10来生成10份不同大小的测试集,每份测试集包含

的三元组数目如表 3 所示. 对于每份测试集,将相应的数据全部装入内存. 通过 LUBM 提供的 3 个查询即查询 5、6、7(查询复杂度依次升高) 来评估方法的性能:

query5(?x) : - takesCourse(?x,'graduateCourse0') , graduateStudent(?x) ;

query6(?x,?y) : - graduateStudent(?x) , memberOf(?x,?z) , undergraduateDegreeFrom(?x,?y) , university(?y) , department(?z) , subOrganizationOf(?z,?y) ;

query7(?x,?y) : - advisor(?x,?y) , teacherOf(?y,?z) , takesCourse(?x,?z) , student(?x) , faculty(?y) , course(?z) .

表 3 LUBM 测试集的规模

Table 3 Size of test collections

测试集	大学数	三元组数目	测试集	大学数	三元组数目
D1	1	100 545	D6	6	722 955
D2	2	230 063	D7	7	884 145
D3	3	337 129	D8	8	1 001 420
D4	4	477 786	D9	9	1 124 616
D5	5	624 534	D10	10	1 272 577

图 2 的实验结果显示,给定规则后本方法的推理时间随数据集的规模呈线性增长. 本文观察到测试集 D10 全部载入内存并且生成内存对象后,需要消耗约 800M 的内存,这大大超出了浏览器程序正常运行时的内存消耗. 而本文的方法只考虑与查询相关的规则和事实,这意味着可以使用外存存储事实和规则,根据查询动态装载相关的事实和规则以减少内存消耗. 为了验证该想法,本文实现了外存版的知识库,并考察测试集 D10 在回答查询 5、6 和 7 时的内存消耗,结果如表 4 所示. 表 4 给出了基于从外存动态提取规则和事实的方法,测试集 D10 推理时针对不同查询的内存消耗及其占 D10 数据全部载入时消耗内存的比例. 表 4 的结果显示,本文的方法在可伸缩性上有一定的优势.

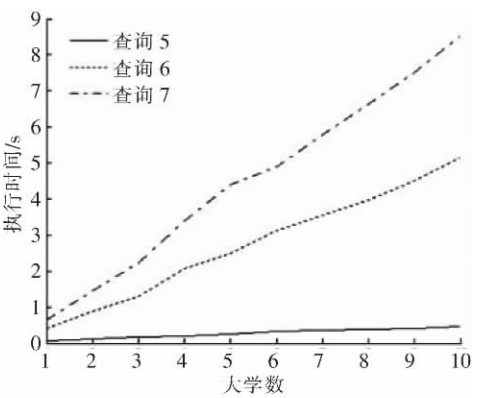

图 2 不同测试集上的查询执行时间

Fig.2 Query execution time for different test collections

6 结语

表 4 D10 针对不同查询的内存消耗

Table 4 Main memory used by D10 for different queries

	Q5	Q6	Q7
内存消耗/M	123	350	574
所占比例/%	15.4	43.8	71.8

文献 [6,7,15-19] 从理论上研究将 RDF 数据转换成逻辑程序对应的规则,使用规则引擎完成推理与查询. 本文将 RDF 数据表示成逻辑程序的方法与文献 [7] 类似,然而本文还考虑了基于查询的 RDF 数据转化问题.

Oracle 使用前向链方法对数据库中的大规模 RDF 数据进行推理[20] ,然而其目标是闭包推理即推导出所有的结论,而不是回答特定的查询. 此外,该推理是基于静态数据集的,而本文处理的是浏览器环境中的动态数据集. 类似本文的方法,文献 [12] 推理时也考虑数据源的权威性,然而该推理是在拥有可观计算资源的服务器上完成静态数据集的闭包推理,同时没有考虑推理结论的证据解释问题.

语义 Web 浏览器中的推理受到 4 个方面的影响: 知识库是动态变化的; 客户端计算资源和存储资源受限; 及时响应; 数据和推理的可信性. 本文提出的规则推理方法,通过动态提取与查询相关的规则和事实,来处理动态性和可伸缩性问题,通过 Magic Sets 优化技术缩短推理时间来满足及时响应要求. 为了提高数据和推理的可信性,本文的方法提供了推理结论的证据解释,在转化本体公理时考虑了数据源的权威性.

本文的下一步工作将在本体推理上支持更多 OWL 2 RL[6] 的公理,同时研究推理时内存和外存 RDF 数据的调度问题来优化推理的性能.

[参考文献](References)

[1] Bizer C, Heath T, Berners-Lee T. Linked data-the story so far [J]. International Journal on Semantic Web and Information Systems, 2009, 5(3): 1-22.

[2] Baader F, Sattler U. An overview of tableau algorithms for description logics [J]. Studia Logica, 2001, 69(1): 5-40.

[3] Lloyd J. Foundations of Logic Programming [M]. 2nd ed. Berlin: Springer-Verlag, 1987.

[4] Meditskos G, Bassiliades N. Combining a dl reasoner and a rule engine for improving entailment-based owl reasoning [C]//Proc of the 7th International Semantic Web Conference. Berlin: Springer, 2008: 277-292.

[5] Hartig O, Bizer C, Freytag J C. Executing SPARQL queries over the web of linked data [C]//Proc of the 8th International Semantic Web Conference. Berlin: Springer, 2009: 293-309.

[6] Motik B, Grau B C, Horrocks I, et al. OWL 2 Web ontology language profiles [EB/OL]. W3C Recommendation [2009-10-27]. http://www.w3.org/TR/owl2-profiles/.

[7] Grosof B N, Horrocks I, Volz R, et al. Description logic programs: combining logic programs with description logic [C]//Proc of the 12th International Conference on World Wide Web. New York: ACM Press, 2003: 48-57.

[8] Beeri C, Ramakrishnan R. On the power of magic [J]. The Journal of Logic Programming, 1991, 10(3/4): 255-299.

[9] Ramakrishnan R, Srivastava D, Sudarshan S. Rule ordering in bottom-up fixpoint evaluation of logic programs [C]//Proc of the 16th International Conference on Very Large Data Bases. San Francisco: Morgan Kaufmann, 1990: 359-371.

[10] Ullman J D. Principles of Database and Knowledge-base Systems [M]. New York: Computer Science Press, 1989.

[11] McGuinness D L, Pinheiro da Silva P. Explaining answers from the semantic Web: the inference Web approach [J]. Web Semantics: Science, Services and Agents on the World Wide Web, 2004, 1(4): 397-413.

[12] Hogan A, Harth A, Polleres A. SAOR: authoritative reasoning for the Web [C]//Proc of the 3rd Asian Semantic Web Conference. Berlin: Springer, 2008: 76-90.

[13] Bry F, Eisinger N, Eiter T, et al. Foundations of rule based query answering [C]//Proc of the 3rd Reasoning Web Summer School. Berlin: Springer, 2007: 1-153.

[14] Guo Y, Pan Z, Heflin J. LUBM: a benchmark for OWL knowledge base systems [J]. Web Semantics: Science, Services and Agents on the World Wide Web, 2005, 3(2/3): 158-182.

[15] Boley H, Hallmark G, Kifer M, et al, eds. RIF core dialect [EB/OL]. W3C Recommendation [2010-06-22]. http://www.w3.org/TR/2010/REC-rif-core-20100622/.

[16] Motik B, Sattler U, Studer R. Query answering for OWL-DL with rules [J]. Web Semantics: Science, Services and Agents on the World Wide Web, 2005, 3(1): 41-60.

[17] ter Horst H J. Completeness, decidability and complexity of entailment for RDF schema and a semantic extension involving the OWL vocabulary [J]. Web Semantics: Science, Services and Agents on the World Wide Web, 2005, 3(2/3): 79-115.

[18] Krötzsch M, Rudolph S, Hitzler P. ELP: tractable rules for OWL 2 [C]//Proc of the 7th International Semantic Web Conference. Berlin: Springer, 2008: 649-664.

[19] Horrocks I, Patel-Schneider P F, Bechhofer S, et al. OWL rules: a proposal and prototype implementation [J]. Web Semantics: Science, Services and Agents on the World Wide Web, 2005, 3(1): 23-40.

[20] Kolovski V, Wu Z, Eadon G. Optimizing enterprise-scale OWL 2 RL reasoning in a relational database system [C]//Proc of the 9th International Semantic Web Conference. Berlin: Springer, 2010: 436-452.

[责任编辑: 严海琳]