软件学报 ISSN 1000-9825, CODEN RUXUEW
Journal of Software,2020,31(2):344−355 [doi: 10.13328/j.cnki.jos.005614]

E-mail: jos@iscas.ac.cn
http://www.jos.org.cn
Tel: +86-10-62562563

一种基于约束依赖性分析的 RDFS 模式抽取方法*

赵晓非[1,2], 史忠植[2], 田东平[2], 刘建伟[2]

[1](天津工业大学 计算机科学与技术学院,天津 300387)

[2](中国科学院 计算技术研究所 智能信息处理重点实验室,北京 100190)

通讯作者: 赵晓非, E-mail: zhaoxiaofei1978@hotmail.com

摘 要: 为了验证 RDFS(resource description framework schema)本体的正确性所执行的推理是一项计算开销很大的任务,该任务在附加约束存在的条件下变得更加复杂.提出了一种旨在不改变推理结果的前提下,对 RDFS 模式进行抽取的方法.该方法基于对约束间的依赖关系进行分析.为了获取 RDFS 模式的精确语义,首先,将模式元素和约束形式化为一阶谓词逻辑中的析取嵌入依赖;接着,根据约束间的相互影响建立约束依赖图,在此基础上,提出了删除与推理任务无关的边和节点的策略;最后,通过重构造过程获取 RDFS 子模式.该方法使得推理验证可以在抽取后的小规模本体上进行.实验结果显示,该方法可以显著地提高 RDFS 本体验证过程的效率,抽取过程的平均耗时为 0.60s,与推理检测时间相比几乎可以忽略,而获得的效率提升则为 2.00 倍~22.97 倍不等.

关键词: RDFS(resource description framework schema);约束;本体抽取;依赖性分析

中图法分类号: TP182

中文引用格式: 赵晓非,史忠植,田东平,刘建伟.一种基于约束依赖性分析的 RDFS 模式抽取方法.软件学报,2020,31(2):344−355. http://www.jos.org.cn/1000-9825/5614.htm

英文引用格式: Zhao XF, Shi ZZ, Tian DP, Liu JW. RDFS schema extracting method based on analysis of constraint dependency. Ruan Jian Xue Bao/Journal of Software, 2020,31(2):344−355 (in Chinese). http://www.jos.org.cn/1000-9825/5614.htm

RDFS Schema Extracting Method Based on Analysis of Constraint Dependency

ZHAO Xiao-Fei[1,2], SHI Zhong-Zhi[2], TIAN Dong-Ping[2], LIU Jian-Wei[2]

[1](School of Computer Science and Technology, Tianjin Polytechnic University, Tianjin 300387, China)

[2](Key Laboratory of Intelligent Information Processing, Institute of Computing Technology, Chinese Academy of Sciences, Beijing 100190, China)

Abstract: The reasoning performed to verify the correctness of the RDFS ontologies is the task with high computational cost. The task will become more complex in the scenario of existence of additional constraints. This paper presents an approach that can extract the RDFS schema without changing the reasoning results. This approach is based on the analysis of the dependency relationships between the constraints. To capture the precise semantics of the RDFS schema, firstly, the schema elements and the constraints are formalized into first-order formulas expressed as disjunctive embedded dependencies and the constraint depandency graph is established according to the interaction between the constraints. Then, the strategies for deleting the edges and the nodes that are irrelevant to the reasoning tasks are applied. Finally, the RDFS sub-schema is obtained through the reconstruction process. The proposed approach enables the reasoning to be carried out on the extracted small-scale ontologies. The experiment results show that the proposed approach can significantly improve the

* 基金项目: 国家重点基础研究发展计划(973)(2013CB329502); 国家自然科学基金(61035003); 江苏省计算机信息处理技术重点实验室开放基金(KJS1737); 陕西省科技厅工业攻关项目(2018GY-037)

Foundation item: National Basic Research Program of China (973) (2013CB329502); National Natural Science Foundation of China (61035003); Open Foundation of Jiangsu Provincial Key Laboratory for Computer Information Processing Technology (KJS1737); Industrial Research Project of Science and Technology Bureau, Shaanxi Province (2018GY-037)

收稿时间: 2018-01-31; 修改时间: 2018-05-01; 采用时间: 2018-06-21

efficiency of the RDFS ontology validation. Compared to the reasoning time, the average time (0.60s) consumed by the extraction process is almost negligible, while the efficiency increasement ranges from 2.00 times to 22.97 times.

Key words: RDFS (resource description framework schema); constraint; ontology extracting; dependency analysis

作为 W3C(world wide Web consortium)提出的对 Web 环境中信息资源进行统一描述的语义模型,RDFS (resource description framework schema)[1]使得通过一系列有确定语义的词汇来描述概念之间的层次结构及概念和属性的语义成为可能,因而成为构建语义 Web 本体的重要基础.

确保本体的正确性,是基于本体的建模过程中避免错误传播的关键.为了确保 RDFS 本体的正确性所执行的推理,通常称其为 RDFS 本体验证[2].该过程主要检测本体是否满足指定规约的集合,例如一个典型的规约是类的可满足性.如果存在一个满足所有约束且包含至少 1 个类 C 实例的 RDFS 例示,则称类 C 是可满足的.RDFS 本体验证涉及的其他规约还包括约束一致性、约束非冗余性等.

为了实现自动/半自动的 RDFS 本体验证,已有一些方法和推理工具被提出,例如 StardogICV[3]、RDD CHECKER[4]、ShEx[5]等.然而,考虑到目前已投入使用的本体动则包含几百个类以上,在大规模 RDFS 本体上执行推理是一项计算开销很大的任务,特别是由于 RDFS 描述语义约束能力的欠缺,在 RDFS 中引入非图形化的约束机制已成为当前的一个重要方向,例如文献[6−8]等.而这些约束的引入,使推理检测变得更加复杂.因此,提高 RDFS 本体验证的效率已成为本体建模领域一个亟待解决的问题.

本体抽取技术允许推理检测在规模较小的子本体上进行,因而可以显著地提高验证过程的效率,而模式抽取是本体抽取的先决条件.图 1 给出了一个 RDFS 模式.该模式描述了学校中的教师(*Teacher*)和课程(*Course*)之间的关系.教师分为全职教师(*Fulltime*)和兼职教师(*Parttime*),实验课(*ExperimentCourse*)是课程的一种且仅能由兼职教师来指导.附加的约束以 4 条描述逻辑公理的形式提供,其中,前 2 条约束规定教师只能属于全职教师和兼职教师两种情况之一,且全职教师类和兼职教师类是不相交的;后 2 条约束分别规定一名教师至少要教授 1 门课程以及一门课程至少要由 1 名教师来教授.

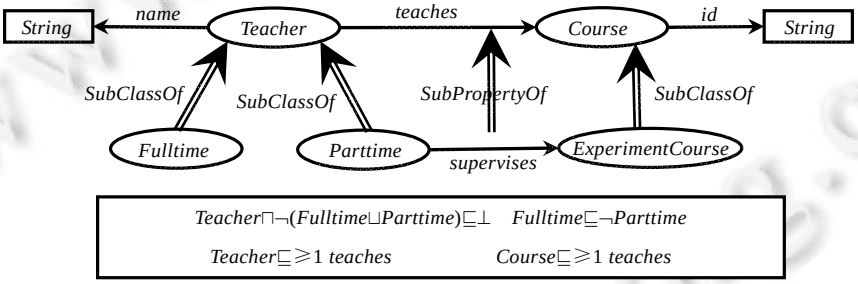

Fig.1 RDFS schema for the relation between teachers and courses

图 1 描述教师和课程之间关系的 RDFS 模式

为了提高 RDFS 本体验证的效率,本文提出了一种 RDFS 模式的抽取方法.该方法的基本思想是:给定待检测的规约,并不是所有的模式元素都与推导检测结果相关.事实上,只有可能与该规约相冲突的约束以及这些约束所涉及到的类、值属性和关联属性与之相关,因此,其他类、值属性、关联属性和约束就可以从模式中删除而不改变该规约的检测结果.由于推理仅在包含相关元素的简化模式上进行,显然,验证效率得以提高.本文的方法基于严格的形式化机制,对于 RDFS 模式的每一种约简情况都给出了严格的证明,这也是该方法优于其他方法的原因之一.

以图 1 中的 RDFS 模式为例,若检测的目标为关联属性 *teaches* 的可满足性,则抽取出的模式如图 2 所示.共计 1 个类、2 个值属性、1 个关联属性、1 个子类关系、1 个子属性关系和 2 个非图形约束被删除,抽取后的规模显著小于原始规模.

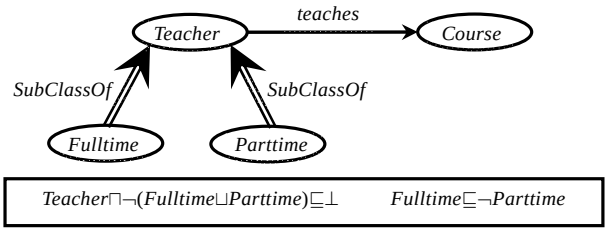

Fig.2 Extracted RDFS schema for checking satisfiability of teaches

图 2 检测 teaches 可满足性时抽取出的 RDFS 模式

为了进一步验证本方法的有效性,我们进行了超过 400 次的检测实验,结果显示,本方法在各种实验情况下均能显著地提高检测的效率,最多可以达到 22.97 倍的效率提升.

1 约束依赖性分析的逻辑基础及本方法的整体思路

1.1 约束依赖性分析的逻辑基础

限于篇幅,我们仅对与本文关系密切的逻辑机制进行简要介绍.

变元和常量统称为项,记为 t.原子 $p(t_1,...,t_n)$是作用于 n 个项 $t_1,...,t_n$ 的谓词,简记为 $p(\bar{t})$.如果一个原子中所有的项都是常量,则称该原子为闭原子.原子 $p(\bar{t})$ 及其否定 $\neg p(\bar{t})$ 统称为文字,记为ℓ.谓词 p 的一个闭原子称为 p 的一个实例.1 个或多个谓词的实例构成的有限集合称为一个例示.置换θ是形如$\{x_1/t_1,...,x_n/t_n\}$的一个集合,其中,每个变元 x_i 互不相同.所有 x_i 构成的集合被称为置换θ的域,记为 $domain(\theta)$.如果每个 t_i 均为常量,则称置换θ为闭置换.应用置换θ将文字ℓ中的每一个 x_i 同时替换为相应的 t_i 所获得的文字记为$\ell\theta$.类似地,在合取式φ的每一个文字之上,同时应用置换θ所得到的合取式记为$\varphi\theta$.

给定文字的合取式φ、谓词 p 和变元 x,用 $assignmentupper(\varphi,p,x)$/$assignmentlower(\varphi,p,x)$分别表示为了确保包含变元 x 的谓词 p 的每个文字都为真,我们需要赋值的 p 的闭原子的最大/最小个数.例如,假定$\varphi=p(x,u)\wedge p(x,v)\wedge p(x,w)\wedge u\neq v$,则 $assignmentupper(\varphi,p,x)=3$(此时 u,v,w 均不相等),而 $assignmentlower(\varphi,p,x)=2$(此时 $w=u$ 或 $w=v$).

一个否定约束是形如 $\forall\bar{t}.\varphi(\bar{t})\rightarrow\bot$ 的规则,其中,φ是文字的合取式而$\bot$是值为永假的原子.直观上看,否定约束的左边表示永远不会被任何例示所满足的条件.一个析取嵌入依赖(DED)是形如 $\forall\bar{t}.\varphi(\bar{t}_\varphi)\rightarrow\bigvee_{i=1..n}\exists\bar{y}_i.\phi_i(\bar{t}_i,\bar{y}_i)$ 的规则,其中,n 为非负整数.当 n 取值为 0 时,析取嵌入依赖右边的析取式为空,其取值为$\bot$,因此,否定约束实质上是析取嵌入依赖的一种特殊形式.

给定例示 I,如果 I 满足 DED 的左边但不满足该 DED 的右边,则称 I 违背该 DED,即存在某个闭置换θ,使得 $I\models\varphi(\bar{t}_\varphi)\theta$;但不存在任何闭置换$\sigma$,使得对于某个$\phi_i$,有 $I\models\phi_i(\bar{t}_i,\bar{y}_i)\theta\sigma$.若例示 I 违背某个 DED,修复该 DED 即建立 I 的超集 I',I'包含该 DED 右边谓词的必要实例,且满足:对于任意闭置换θ,如果 $I'\models\varphi(\bar{t}_\varphi)\theta$,则必存在闭置换$\sigma$,使得对于某个$\phi_i$,有 $I\models\phi_i(\bar{t}_i,\bar{y}_i)\theta\sigma$.

给定 DED 的集合$\mathcal{D}$及例示 I,若 I 不违背$\mathcal{D}$中的任意 DED,则称 I 关于$\mathcal{D}$是一致的,记作 $I\models\mathcal{D}$.若 DED 的集合$\mathcal{D}$至少存在一个一致的例示 I,则称$\mathcal{D}$是可满足的.

1.2 本方法的整体思路

给定 RDFS 模式和待检测的规约(简记为 SWC),RDFS 模式抽取的任务是删除模式中的类、属性和约束等而不改变 SWC 的可满足性.为了达到此目标,我们借助逻辑机制对 RDFS 模式和规约进行形式化描述,将二者指定为一组一阶约束,而规约的检测问题则转化为逻辑机制中的可满足性检测问题.例如,检测 *Course* 可满足性的任务转化为 $Course(c)$的可满足性检测,而检测是否存在授课教师为非全职教师的课程的任务则转化为 $Course(c)\wedge teaches(t,c)\wedge\neg Fulltime(t)$的可满足性检测.

把既不出现在任何约束中也不出现在SWC中的元素删除掉,是不影响可满足性检测结果的.我们将以此为出发点,但这还远远不够,我们力争尽可能多地删除对于推理结果无影响的约束,从而进一步删除更多的无关元素.由于SWC所对应的一阶公式的可满足性检测过程是搜索既满足约束也满足SWC的例示的过程,因此,如果能够识别哪些约束有可能使SWC不可满足,并把其他约束从模式中删除,是不改变SWC可满足性结果的,所以我们的目标就变成了如何识别可能使SWC不可满足的约束.

下面我们分析可能使SWC不可满足的约束属于哪些情况,从而为删除这些约束提供依据.检测SWC是否是可满足的,等价于判断是否可以修复一个满足SWC的例示.假定存在某个满足SWC的例示I_1,如果I_1违背某个约束,则通过I_1是无法判断SWC的可满足性的,然而在满足SWC的前提下,我们可以通过向I_1中添加有限数量的新实例来尝试修复被违背的约束.新实例的引入有可能导致其他约束被违背,因此该过程是一个迭代的过程,反复添加新实例修复被违背的约束,直到得到一个与所有约束集合一致的例示或者存在被违背的约束且无法再进行修复为止.前者意味着SWC是可满足的,而后者意味着SWC是不可满足的.

通过上述分析不难看出,可能使SWC不可满足的约束无外乎两种情况:(1) 试图满足SWC(或修复随后被违背的约束)时会被违背的约束;(2) 当被修复时将违背其他约束的约束.以图1为例,假定我们要找到一个例示以证明*Course*是可满足的.仅包含1个*Course*的实例的例示满足*Course*但却违背了一门课程至少要由1名教师来教授的约束.为了修复该约束,需要向例示中添加与*Course*相关联的一个*Teacher*的实例,然而该实例的添加进一步违背了一名教师只能属于全职教师和兼职教师两种情况之一的约束.因此,第1个约束有可能使*Course*不可满足.因为试图满足SWC时,它会被违背并且当修复它的时候也将违背其他约束.由于当第2个约束被违背时总是可以通过用*Fulltime*的实例代替*Teacher*的实例来进行修复,而这种修复不会违背其他任何约束,因此第2个约束不会使*Course*不可满足.我们将通过约束依赖图来反映约束之间的这种关系.需要特别说明的是,本文的目的并不是推断SWC的可满足性,而是在不改变SWC可满足性的前提下对RDFS模式进行约简,因此对上述迭代过程的分析仅是为了推断出可能使SWC不可满足的约束所属的情况,后文中,我们将针对每种情况给出模式的约简策略.

我们的RDFS模式抽取方法分3步进行:第1步,将RDFS模式形式化为一阶谓词逻辑中的DED集合,在此基础上建立约束依赖图;第2步,将SWC加入约束依赖图并对其进行约简;第3步,重新构建RDFS模式.第2节~第4节将对这3步分别进行详细介绍.

2 约束依赖图的建立

2.1 RDFS模式的形式化

为了对RDFS模式的语义进行精确的描述从而识别可被删除的约束,需借助一种良定义的形式化机制,我们的方法是将模式元素转换为DED的集合.

首先将RDFS模式中的每个类、值属性和关联属性形式化为一个单独的一阶谓词,以图1为例,可以得到如下谓词:*Teacher*(*t*),*Course*(*c*),*Fulltime*(*ft*),*Parttime*(*pt*),*ExperimentCourse*(*ec*),*TeacherName*(*t*,*n*),*CourseId*(*c*,*i*),*teaches*(*t*,*c*),*supervises*(*pt*,*ec*).

在此基础上,将模式中的约束转化为否定约束.例如图1中“值属性*id*的最大基数为1”的约束可被形式化为$CourseId(c,i_1)\wedge CourseId(c,i_2)\wedge i_1<>i_2\rightarrow\bot$.该式精确地描述了如果课程*c*存在课序号$i_1$和$i_2$而$i_1$和$i_2$不相同,则约束被违背.再如附加约束“教师只能属于全职教师和兼职教师2种情况之一”以及“全职教师类和兼职教师类是不相交的”可分别被形式化为$Teacher(t)\wedge\neg(Fulltime(t)\vee Parttime(t))\rightarrow\bot$和$Fulltime(t)\wedge Parttime(t)\rightarrow\bot$.通过上述形式化,可以对RDFS模式中的全部图形约束(包括值属性最小/最大基数、值属性完整性、关联属性完整性、类/属性层次)和一阶附加约束进行精确的语义描述.

随后,我们将所有的否定文字从否定约束的左边移至其右边并去掉否定符号,从而将每个否定约束等价转换为一个析取嵌入依赖.例如,否定约束$Teacher(t)\wedge\neg(Fulltime(t)\vee Parttime(t))\rightarrow\bot$被等价转换为$Teacher(t)\rightarrow Fulltime(t)\vee Parttime(t)$.

为了在后续建立约束依赖图的过程中对不同的 DED 进行匹配时能够应用置换操作,此步骤的最后需要将每个 DED 中的不同原子的每个变元更换为不同的名字.

经过上述形式化过程,图 1 中的全部约束所对应的 DED 集合列于表 1.

Table 1 DEDs of the constraints in the RDFS schema

表 1 RDFS 模式中约束对应的 DED

图形约束	值属性最小基数 1: (1) $Teacher(t)\rightarrow TeacherName(t,n)$ (2) $Course(c)\rightarrow CourseId(c,i)$
	值属性最大基数 1: (3) $TeacherName(t,n_1)\wedge TeacherName(t,n_2)\wedge n_1<>n_2\rightarrow\perp$ (4) $CourseId(c,i_1)\wedge CourseId(c,i_2)\wedge i_1<>i_2\rightarrow\perp$
	值属性完整性: (5) $TeacherName(t,n)\rightarrow Teacher(t)$ (6) $CourseId(c,i)\rightarrow Course(c)$
	关联属性完整性: (7) $teaches(t,c)\rightarrow Teacher(t)$ (8) $teaches(t,c)\rightarrow Course(c)$ (9) $supervises(pt,ec)\rightarrow Parttime(pt)$ (10) $supervises(pt,ec)\rightarrow ExperimentCourse(ec)$
	类/属性层次: (11) $Fulltime(ft)\rightarrow Teacher(ft)$ (12) $Parttime(pt)\rightarrow Teacher(pt)$ (13) $ExperimentCourse(ec)\rightarrow Course(ec)$ (14) $supervises(pt,ec)\rightarrow teaches(pt,ec)$
附加约束(非图形约束)	(15) $Teacher(t)\rightarrow Fulltime(t)\vee Parttime(t)$ (16) $Fulltime(t)\wedge Parttime(t)\rightarrow\perp$ (17) $Teacher(t)\rightarrow teaches(t,c)$ (18) $Course(c)\rightarrow teaches(t,c)$

2.2 基于DED集合建立约束依赖图

下面我们将建立(为了使某个约束可满足而应用的)修复和(应用该修复可能导致的其他约束的)违背之间的相互影响关系.如前文所述,如果例示满足 DED 的左边而不满足该 DED 的右边,则该 DED 被违背.因此,每个 DED 的左边是违背该 DED 的先决条件,而该 DED 右边析取式中的每个合取式是修改该 DED 的不同方法.进一步来说,如果某个 DED 的右边与另一个 DED 的左边存在相同的原子,通过添加该原子的新实例来修复前一个 DED,可能导致后一个 DED 被违背(因为有可能因为该添加而满足了后一个 DED 的左边),因此这两个 DED 之间存在修复-违背关系.我们通过建立约束依赖图来反映这种关系.

一个约束依赖图包含两种节点:约束节点和修复节点.每个 DED 的左边对应一个约束节点(以空心圆来表示);DED 右边的每个合取式对应一个修复节点(以实心圆来表示),用有向虚线边来表示 DED 之间的修复-违背关系.例如,图 3 展示了表 1 中 DED 7、DED 11 和 DED 15 之间的修复-违背关系.

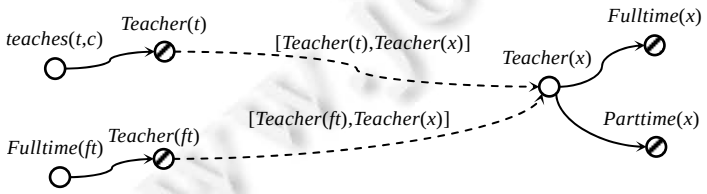

Fig.3 An example for repair-violation relation between DEDs

图 3 DED 之间的修复-违背关系的例子

由于添加 *Teacher* 的一个实例来修复 DED 7 可能导致 DED 15 的违背,我们在前者的修复节点 $Teacher(t)$

到后者的约束节点 $Teacher(x)$之间建立有向虚线边;同时,在边上标记[$Teacher(t)$,$Teacher(x)$]表示前者右边和后者左边共有的原子.

DED 11 和 DED 15 之间的关系与之类似.

下面给出约束依赖图的形式化定义.

定义 1. 一个约束依赖图 G 是一个 6 元组$\langle CN,RN,E_S,E_D,f,g\rangle$表示的有向二分图,且满足以下条件:

(1) CN 是约束节点的集合.

(2) RN 是修复节点的集合.

(3) E_S 是从 CN 到 RN 的有向边的集合,每个 $cn\in CN$ 可以有 0 个、1 个或多个到 RN 中节点的输出边,而每个 $rn\in RN$ 只能有 1 个输入边.对于每个 $e\in E_S$,用 $b(e)$和 $t(e)$分别表示 e 的始端和终端.令

$$repair(cn)=\{rn\in RN|\exists e\in E_S,b(e)=cn\wedge t(e)=rn\}.$$

(4) E_D 是从 RN 到 CN 的有向边的集合,每个 $rn\in RN$ 可以有 0 个、1 个或多个到 CN 中节点的输出边,每个 $cn\in CN$ 可以有 0 个、1 个或多个输入边.对每个 $e\in E_D$,用 $b(e)$和 $t(e)$分别表示 e 的始端和终端.令

$$violation(rn)=\{cn\in CN|\exists e\in E_D,b(e)=rn\wedge t(e)=cn\}.$$

(5) 函数 f 将 CN 和 RN 中的每个节点标记为相应的文字的合取式.

(6) 函数 g 将每个 $e\in E_D$ 标记为一个原子对$[g_{rn},g_{cn}]$,其中,$g_{rn}\in f(b(e))$和 $g_{cn}\in f(t(e))$是拥有相同谓词的原子.

下面给出约束依赖图的建立过程.

算法 1. 建立约束依赖图.

输入:DED 的集合$\mathcal{D}$.

输出:约束依赖图 $G=\langle CN,RN,ES,ED,f,g\rangle$.

(1) 初始化 CN,RN,E_S,E_D 为空;

(2) 对于每个 DED $d\in\mathcal{D}$,执行下述操作:

(3) 建立 d 的约束节点 cn 并令 $CN=CN\cup\{cn\}$;

(4) 利用函数 f 对 cn 进行标记;

(5) 对于 d 右边的每个合取式φ,执行下述操作:

(6) 建立φ的修复节点 rn 并令 $RN=RN\cup\{rn\}$;

(7) 利用函数 f 对 rn 进行标记;

(8) 建立从 cn 到 rn 的边 e,并令 $E_S=E_S\cup\{e\}$;

(9) 跳转到步骤(5);

(10) 跳转到步骤(2);

(11) 对于每个 $rn\in RN$ 和 $cn\in CN$,执行下述操作:

(12) 对于每个原子 $ar\in f(rn)$和 $ac\in f(cn)$,执行下述操作:

(13) 如果 ar 与 ac 的谓词名相同,则,

(14) {建立从 rn 到 cn 的边 e 并令 $E_D=E_D\cup\{e\}$;

(15) 利用函数 g 将 e 标记为$\langle ar,ac\rangle$;}

(16) 跳转到步骤(12);

(17) 跳转到步骤(11);

(18) 返回 G.

以表 1 列出的 DED 集合作为输入,算法 1 输出的约束依赖图如图 4 所示(为了便于理解,我们在每个约束节点旁标上了其对应的 DED 编号并省略了 f 函数和 g 函数的标记信息).

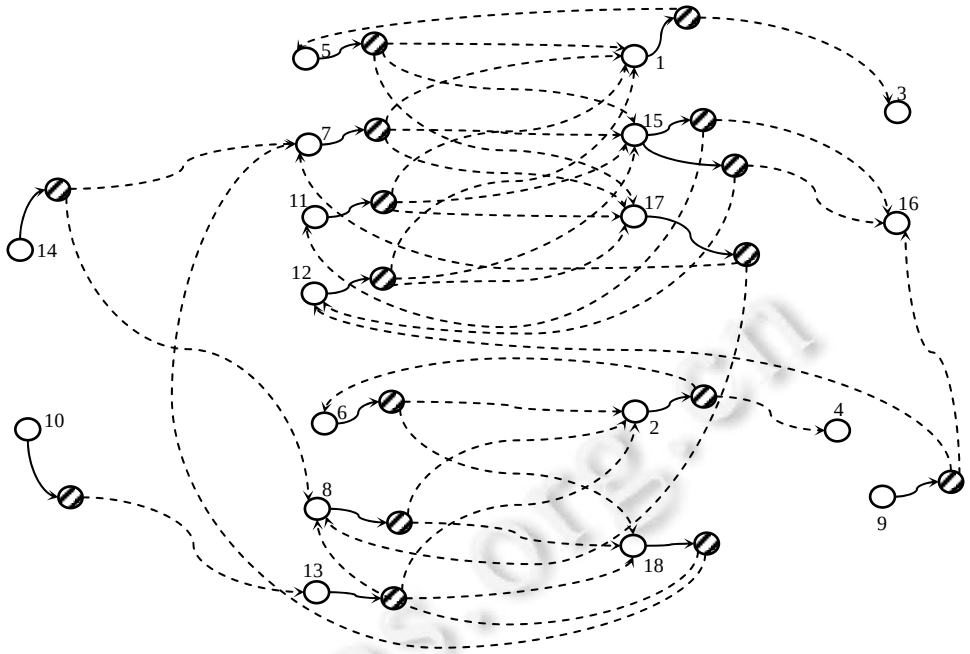

Fig.4 Constraint dependency graph established from the DEDs in Table 1

图 4 根据表 1 列出的 DED 集合建立的约束依赖图

3 约束依赖图的约简

下面的任务是在不改变 SWC 可满足性的前提下,消除尽可能多的 DED.我们首先将 SWC 对应的一阶公式转换为 DED 并加入约束依赖图;接着,迭代地执行消除非依赖边和消除 DED 的操作,直到无法继续消除为止,从而使图得到简化.以图 4 为例,假定 SWC 为检测 *teaches* 的可满足性,经过上述约简过程后得到的结果如图 5 所示.由于仅包含 4 个 DED 且其中的 1 个是待检测的目标,该结果的规模显著小于原始图.这意味着在检测 *teaches* 的可满足性时,我们只需在具有 3 个约束的 RDFS 模式上进行推理就可以了.

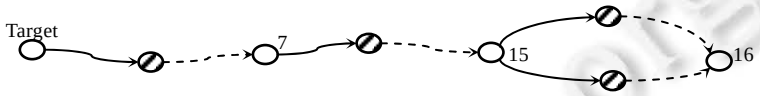

Fig.5 Reduced constraint dependency graph for the target of checking *teaches* satisfiability

图 5 以检测 *teaches* 可满足性为目标约简后的约束依赖图

3.1 SWC到DED的转换

通过将 SWC 对应的一阶公式φ取否定,我们将每个 SWC 转换为一个否定约束$\neg\varphi\rightarrow\bot$.该否定约束表示:如果 SWC 不被满足,则规约被违背.在此基础上,利用第 2.1 节所述的方法将其转换为 DED:$\top\rightarrow\varphi$.例如,检测 *Course* 的可满足性可转换为如下的 DED:$\top\rightarrow$*Course*(*c*).随后,将得到的 DED 添加进约束依赖图,并建立其与图中其他 DED 之间的修复-违背关系,即建立从 SWC DED 的修复节点到其他 DED 约束节点之间的E_D边(由于 SWC DED 的左边为真,故其对应的约束节点不存在输入边).根据原始 RDFS 模式对应的 DED 集合是可满足的可知,加入 SWC DED 后得到的新集合的可满足性与 SWC 的可满足性是等价的.

3.2 消除非依赖边

通过深入分析E_D集合中的边我们发现:当某些边的始端 *rn* 被添加时,并不会导致其终端 *violate*(*rn*)的违背,

我们称这种可消除的边为非依赖边.以图 4 中 DED 1 的修复节点为例,该节点指向 DED 3 和 DED 5 的约束节点.由于 DED 1/DED 3 分别描述每个 *Teacher* 至少/至多拥有 1 个名字,而最大基数约束并不会由于修复最小基数约束而被违背,因此 DED 1 的修复永远不会导致 DED 3 的违背.与之类似,DED 1 的修复也永远不会导致 DED 5 的违背(因为修复 DED 1 意味着它的左边 *Teacher*(*t*)为真,而 *Teacher*(*t*)恰好是 DED 5 的右边).所以说,从 DED 1 发出的两条边都是不影响 SWC 可满足性检测结果的非依赖边.

定义 2. 给定约束依赖图 G 及边 $e\in E_D$,对于任意例示 I,如果 $I\cup\{f(b(e))\}\models f(t(e))$,则必存在闭置换 θ,使得 $I\cup\{f(b(e))\}\models f(repair(t(e)))\theta$ 成立,则称 e 为非依赖边.

下面以定理的形式给出消除非依赖边的策略.以图 6 中的 2 个 DED d_1 和 d_2 为例来进行说明.

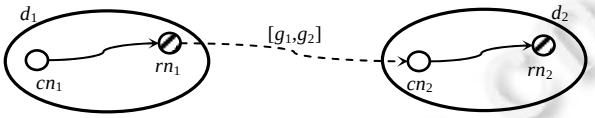

Fig.6 DEDs for deleting non-dependent edge

图 6 消除非依赖边用到的 DED

根据第 2.1 节的变元更名可知,g_1 和 g_2 是拥有相同谓词但变元名不同的原子,因此必然存在某个非闭置换 θ,满足 $g_1=g_2\theta$.给定修复节点 *rn*、*rn* 的一个原子 a 及 *rn* 的约束节点 *cn*,用 *increment*(*rn*)表示 *f*(*rn*)中所有增量变元的集合,即,在 *f*(*rn*)中出现但不在 *f*(*cn*)中出现的所有变元的集合,用 *increment*(*a*)表示出现在 a 中的 *increment*(*rn*)的子集.限于篇幅,定理 1~定理 3 的证明略.

定理 1. 若存在置换 θ_2 满足 $f(rn_2)\theta\theta_2\subseteq f(cn_1)\cup f(rn_1)$,其中,$domain(\theta_2)\subseteq increment(rn_2)$,则边 $[g_1,g_2]$ 是非依赖边.

粗略地说,每当 $f(cn_1)$ 成立,通过添加 $f(rn_1)$ 的实例进行修复时,$f(rn_2)$ 总是成立,因为它已包含于 $f(cn_1)\cup f(rn_1)$ 中.因此,尽管 $f(cn_2)$ 可能由于 $f(rn_1)$ 的添加而变为真,d_2 却不会被违背.表 1 中,DED 1 和 DED 5 之间的关系就属于这种情况,因此它们之间的边是可被删除的非依赖边.

定理 2. 若对于从 $f(rn_1)$ 到 $f(cn_2)$ 的所有置换 θ_2',均存在置换 θ_2 满足 $f(rn_1)\theta_2\subseteq f(cn_2)\theta\backslash f(rn_1)\theta_2'$,其中,$domain(\theta_2)\subseteq increment(rn_1)$,则边 $[g_1,g_2]$ 是非依赖边.

表 1 中,DED 1 和 DED 3 之间的关系就属于这种情况,如果 I' 违背 DED 3,则将 DED 3 左边的两个 *TeacherName* 实例中刚添加进来的(即为了修复 DED 1 而添加进来的)那一个去掉之后得到的 I 仍包含一个 *TeacherName* 的实例.该实例给出了 *Teacher*(*t*)的名字,因而 I 满足 DED 1,这与假设相悖.

定理 3. 如果变元 $x\in increment(rn_1)$ 的定义域是无穷的,且 g_1 包含 x,$f(cn_2)$ 中存在谓词 p,使得

$$assignmentupper(f(rn_1),p,x)<assignmentlower(f(cn_2),p,x),$$

则边 $[g_1,g_2]$ 是非依赖边.

粗略地说,定理 3 描述的是这样的情形:由于向 I 中添加 $f(rn_1)$ 修复 d_1 时为变元 x 赋予了新值,因此该修复将永远不会违背 d_2.假定存在某个左边包含 $TeacherName(t_1,n)\wedge TeacherName(t_2,n)$ 的 DED,由于修复 DED 1 的过程每次为 *Teacher* 赋予的名字不与任何已有的值重复,所以 DED 1 的修复永远不会导致该 DED 的违背.

3.3 消除DED

下面以定理的形式给出消除 DED 的策略.定理 4 和定理 5 对应于第 1.2 节中所述的第 1 种情况,而定理 6 则与第 2 种情况相对应.限于篇幅,定理 4 至定理 6 的证明略.

定理 4. 给定约束依赖图 G 及其对应的 DED 集合 $\mathcal{D}$,令 *cn* 是约束节点且 $f(cn)\neq\top$,对于 $f(cn_i)=\top$ 的任意约束节点 cn_i,若不存在从 cn_i 到 *cn* 的路径,则从 $\mathcal{D}$ 中消除 *cn* 所属 DED 后得到的集合 $\mathcal{D}'$ 与 $\mathcal{D}$ 的可满足性是等价的.

如前所述,SWC 与 $\mathcal{D}$ 的可满足性是等价的,因此应用定理 4 对约束依赖图进行的约简操作并不改变 SWC 的可满足性检测结果.

定理 5. 给定约束依赖图 G 及其对应的 DED 集合 $\mathcal{D}$,令 p 是出现在约束节点 *cn* 的标记 *f*(*cn*)中的谓词,如果对于所有修复节点 $rn\notin repair(cn)$,*f*(*rn*)中都不包含 p,则从 $\mathcal{D}$ 中消除 *cn* 所属 DED 后得到的集合 $\mathcal{D}'$ 与 $\mathcal{D}$ 的可满足

性是等价的.

定理 6. 给定约束依赖图 G 及其对应的 DED 集合$\mathcal{D}$,对于任意约束节点 cn,若存在修复节点 $rn \in repair(cn)$ 且 $violation(rn)$为空,则从$\mathcal{D}$中消除 cn 所属 DED 后得到的集合$\mathcal{D}'$与$\mathcal{D}$的可满足性是等价的.

4 RDFS 模式的重构造

重构造过程的核心思想是建立原始RDFS模式的子集.该子集应包含在简化的约束依赖图中出现的所有约束.首先,从一个空的 RDFS 模式出发,逐一添加与简化的约束依赖图中每个谓词对应的 RDFS 模式元素:对于表示类的每个谓词,将该类添加进新模式;对于表示类的值属性的每个谓词,将该值属性关联到该类(如果该类在新模式中尚不存在,则需要先将其加入);对于表示类之间的关联属性的每个谓词,直接建立对应的类之间的关联即可(如果这些类在新模式中尚不存在,则需先将它们加入).需要注意的是,通过上述添加过程,值属性最小/最大基数、值属性/关联属性完整性都已在图中得到反映,因此该过程实际上也将这部分图形约束添加进重新构造的 RDFS 模式中.随后,将简化的约束依赖图中剩余的图形约束和全部非图形约束添加进 RDFS 模式.对于图形约束来说,此时只有类/属性层次还未被添加.如果存在表示层次约束的 DED,根据前面的添加过程,子类/父类以及子属性/父属性均已被添加,因此可以直接建立它们之间的继承关系;对于非图形约束,由于涉及到的类和属性同样已被添加,因此直接把与 DED 对应的原始非图形约束包含进新模式即可.

以图 5 为例,重新构造的 RDFS 模式如图 2 所示.

5 实验过程及结果

为了验证本文方法的有效性,我们开发了原型系统并进行了超过 400 次的检测实验.所有实验是在 Intel Core i5-7400、4GB 内存和 Windows 10 操作系统的环境下执行的.实验选取 LUBM-50、LUBM-100 和 DBLP 作为数据集.我们分别使用两种推理工具 StardogICV 和 RDD CHECKER,并以逐步增加复杂度的方式执行可满足性检测.在实验过程中,对于每种 RDFS 本体我们分别随机选择 3 个元素并检测它们的可满足性.为了测试本方法在任务复杂度逐渐增加的情况下的表现,对于所选择的每个元素,我们依次检测了实例个数为 1,2,3,...,12 时的可满足性,因此每种实例个数和元素的组合构成了 1 个单独的检测目标.图 7 给出了检测的执行时间的部分结果.所有实验结果汇总于表 2~表 4 中.

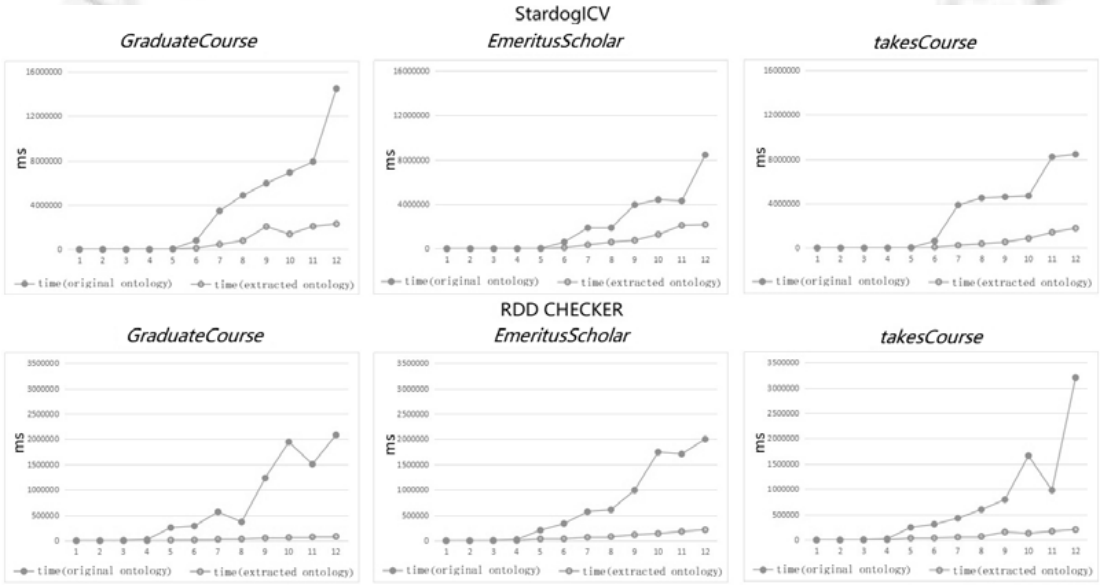

Fig.7 Checking times for LUBM-50 elements satisfiability

图 7 LUBM-50 元素的可满足性检测时间

Table 2 Summary of experimental results for LUBM-50

表 2 LUBM-50 实验结果汇总

	GraduateCourse	*EmeritusScholar*	*takesCourse*
平均抽取时间(ms)	503	498	491
DED(约简前)	276	276	276
DED(约简后)	36	57	42
删除的类	6	6	5
删除的值属性	13	9	12
删除的关联属性	9	12	9
删除的类/属性层次	18	14	13
StardogICV 平均时间(抽取前)(ms)	3 705 766	2 124 994	2 913 637
StardogICV 平均时间(抽取后)(ms)	766 483	613 410	440 126
平均效率提升	4.83	3.46	6.62
RDD CHECKER 平均时间(抽取前)(ms)	690 580	683 655	687 342
RDD CHECKER 平均时间(抽取后)(ms)	30 062	73 442	72 428
平均效率提升	22.97	9.31	9.49

Table 3 Summary of experimental results for LUBM-100

表 3 LUBM-100 实验结果汇总

	StaffMember	*SocialService*	*hasWeeklyBreakup*
平均抽取时间(ms)	948	918	983
DED(约简前)	513	513	513
DED(约简后)	96	112	87
删除的类	14	11	16
删除的值属性	37	34	41
删除的关联属性	19	16	14
删除的类/属性层次	26	30	26
StardogICV 平均时间(抽取前)(ms)	6 940 067	8 311 574	6 731 248
StardogICV 平均时间(抽取后)(ms)	613 623	424 468	333 726
平均效率提升	11.30	19.58	20.17
RDD CHECKER 平均时间(抽取前)(ms)	991 694	911 236	903 458
RDD CHECKER 平均时间(抽取后)(ms)	87 010	396 421	278 845
平均效率提升	11.40	2.30	3.24

Table 4 Summary of experimental results for DBLP

表 4 DBLP 实验结果汇总

	Edited publication	*Conference section*	*Conference date*
平均抽取时间(ms)	371	394	396
DED(约简前)	187	187	187
DED(约简后)	54	46	46
删除的类	2	3	2
删除的值属性	2	3	2
删除的关联属性	12	18	19
删除的类/属性层次	27	26	22
StardogICV 平均时间(抽取前)(ms)	1 286 413	1 135 423	1 448 786
StardogICV 平均时间(抽取后)(ms)	631 534	568 306	627 180
平均效率提升	2.04	2.00	2.31
RDD CHECKER 平均时间(抽取前)(ms)	360 410	753 335	807 792
RDD CHECKER 平均时间(抽取后)(ms)	19 633	61 836	142 217
平均效率提升	18.36	12.18	5.68

对于 LUBM-50 本体,我们随机选择了 3 个元素:类 *GraduateCourse*、类 *EmeritusScholar* 和关联属性 *takesCourse*.图 7 给出了检测上述元素可满足性的执行时间对比,实心圆点/空心圆点分别表示在原始本体/抽取后的本体上的执行时间.可以看出,无论对于 StardogICV 还是 RDD CHECKER,抽取后的检测时间均小于原始时间;且随着实例个数的增加,这种距离逐渐拉大.本方法在 LUBM-100 和 DBLP 上取得了与图 7 类似的执行时间结果,限于篇幅,此处略.表 2 是在 LUBM-50 上的实验结果汇总,包括原始的约束依赖图中 DED 的数目及约简后

图中 DED 的数目、被删除的各种 RDFS 模式元素的数目、抽取过程所耗费的平均时间以及抽取前后执行检测所耗费的平均时间.需要特别指出的是,由于消除非依赖边和 DED 的过程与实例个数无关,因此无论对于被选择的哪个元素,尽管实例个数从 1 逐渐增加到 12,但被消除的 DED 个数保持不变.通过第 2 节不难看出,被建立的约束依赖图中的节点/边的数目随着原始 RDFS 模式规模的增加,至多呈多项式级别增加,这也从平均耗时仅为 0.50s 的抽取时间中得到反映.实验结果也显示了在抽取后的本体上执行检测所需时间均显著小于原始时间,推理检测的效率提升从 3.46 倍到 22.97 倍不等.与之相比,抽取时间几乎可以忽略.表 3 和表 4 分别是针对 LUBM-100 和 DBLP 的实验结果,可以看出,我们的方法给二者带来的效率提升为 2.30~20.17 倍和 2.00~18.36 倍,相比之下,抽取过程的平均耗时仅为 0.95s 和 0.39s.

6 相关研究

与本文相关的研究主要集中在两个方面:RDFS 模式抽取和 RDFS 模式分区,下面分别进行讨论.

Georgia 等人[9]提出了一种 RDFS 模式的抽取方法,并开发了相应的原型工具 RDFDigest.在综合考虑模式语义、图形结构以及实例分布情况的基础上,该方法提出了相关性基数和集中度的概念,并用于描述模式元素之间的关联程度,从而可以识别出构成子模式的候选元素.基于对链接数据的解释进行排序,Hasan[10]提出了一种 RDFS 模式的抽取方法.该方法将模式元素分为 3 类:突出陈述、相似陈述和抽象陈述,针对每种情况定义了过滤准则,从而筛选出模式子集.Sejla 等人[11]提出了一种面向链接数据查询优化的 RDFS 模式抽取方法,针对不同程度的需求,他们提出了两种思路:基准抽取和精化抽取,并为其分别定义了一系列准则.前者可以高效地适用于大部分链接数据模式,但在某些情况下会导致关联属性的引用冲突;后者弥补了此缺陷,但具有较高的计算复杂度.文献[9–11]的不足之处在于:一方面都没有建立在严格的形式化基础之上,因此无法提供严格的证明,无法确保 SWC 在抽取前后得以保持;另一方面,上述方法在抽取过程中都没有考虑约束的相关性,因此无法处理存在非图形约束的情形.此外,我们的方法独立于任何推理工具的特点,使其可以与上述技术互为补充.可以首先通过上述技术对本体进行抽取,而后再应用本文的方法进一步提高检测的效率.

通过将本体元素划分到语义无关的闭包,RDFS 本体分区技术允许推理检测在不同的分区上并行执行,因而成为提高本体验证效率的另一种思路.Martin 等人[12]提出了一种基于可伸缩并行化规则的方法,可以将 RDFS 模式分区为语义无关的子集.如果至少有 1 个分区是可满足的,则原始本体是可满足的.该方法的主要缺陷在于:当某种特定约束存在的情况下,无法进行分区操作.假定某个约束规定类 C 至少存在 1 个实例,则 C 应该出现在每个分区中(因为如果 C 是不可满足的,则整个本体也是不可满足的,即每个分区都应是不可满足的).由于每个分区都是语义无关的,因此唯一的输出就是只有 1 个分区.Georgia 等人[13]提出的方法首先根据 RDF 数据的相关性选择出最重要的元素作为核心,而后根据数据之间的依赖性将剩余元素分配到适当的核心,从而构造出不同的分区.尽管可以获得显著小于原始规模的分区,然而由于同一数据会被分配到多个分区,该方法构造的分区冗余度较大,并且也没有考虑存在非图形约束的情形.文献[14]提出了一种基于语义感知的 RDFS 分区方法:首先,利用改进的页排序算法对模式元素进行排序;而后,根据其语义信息约简不同片段间的交叉边的数量.与文献[13]类似,该方法也没有考虑约束的语义.此外,文献[12–14]共同的缺陷还在于没有提供分区过程的正确性的形式化证明.我们认为这是一个重大的缺陷,因为本体验证均基于严格的数学基础,在无法确保推理结果不受影响的情况下,将分区技术集成进本体验证工具是不合适的.

7 结 论

为了提高本体验证的效率,本文提出了一种 RDFS 模式的抽取方法.给定 RDFS 模式和待检测的规约,该方法在不改变规约可满足性的前提下,可以返回显著小于原始规模的子模式.该方法首先将 RDFS 模式形式化为 DED 的集合,并据此建立约束依赖图;接着,迭代地执行一系列操作删除非依赖边和 DED;最后,通过重建 RDFS 模式实现模式的抽取.对于每一种约简操作,我们都给出了严格的证明.为了验证本方法的有效性,我们开发了原型系统并进行了大量实验.实验组合了多个数据集、推理工具和检测目标.结果显示,在各种情况下,该方法均

能显著地提高检测效率,且效率提升从 2.00 倍到 22.97 倍不等.

References:

[1] Polleres A, Hogan A, Delbru R, Umbrich J. RDFS and OWL reasoning for linked data. In: Proc. of the 9th Int'l Conf. on Reasoning Web: Semantic Technologies for Intelligent Data Access. LNCS 8067, Berlin: Springer-Verlag, 2013. 91−149.

[2] Bsch T, Eckert K. Requirements on RDF constraint formulation and validation. In: Proc. of the 2014 Int'l Conf. on Dublin Core and Metadata Applications. Dublin: Dublin Core Metadata Initiative, 2014. 95−108.

[3] Patel-Schneider PF. Using description logics for RDF constraint checking and closed-world recognition. In: Proc. of the 29th AAAI Conf. on Artificial Intelligence. AAAI, 2015. 247−253.

[4] Fischer PM, Lausen G, Schatzle A, Schmidt M. RDF constraint checking. In: Proc. of the Workshops of the EDBT/ICDT 2015 Joint Conf. New York: CEUR-WS.org, 2015. 205−212.

[5] Boneva I, Gayo JEL, Prudhommeaux EG. Semantics and validation of shapes schemas for RDF. In: Proc. of the 2017 Int'l Semantic Web Conf. LNCS 10587, Berlin: Springer-Verlag, 2017. 104−120.

[6] Halfeld-Ferrari M, Hara CS, Uber FR. RDF updates with constraints. In: Proc. of the 8th Int'l Conf. on Knowledge Engineering and the Semantic Web. CCIS 786, Berlin: Springer-Verlag, 2017. 229−245.

[7] Calvanese D, Fischl W, Pichler R, Sallinger E, Simkus M. Capturing relational schemas and functional dependencies in RDFS. In: Proc. of the 28th AAAI Conf. on Artificial Intelligence. AAAI, 2014. 1003−1011.

[8] Pichler R, Polleres A, Skritek S, Woltran S. Complexity of redundancy detection on RDF graphs in the presence of rules, constraints, and queries. Semantic Web, 2013,4(4):351−393.

[9] Troullinou G, Kondylakis H, Daskalaki E, Plexousakis D. RDF digest: Efficient summarization of RDF/S KBs. In: Proc. of the 12th European Semantic Web Conf. LNCS 9088, Berlin: Springer-Verlag, 2015. 119−134.

[10] Hasan R. Generating and summarizing explanations for linked data. In: Proc. of the 11th European Semantic Web Conf. LNCS 8465, 2014. 473−487.

[11] Cebiric S, Goasdoue F, Manolescu I. Query-oriented summarization of RDF graphs. In: Proc. of the 2015 British Int'l Conf. on Databases. LNCS 9147, Berlin: Springer-Verlag, 2015. 87−91.

[12] Peters M, Brink C, Sachweh S, Zundorf A. Scaling parallel rule-based reasoning. In: Proc. of the 11th European Semantic Web Conf. LNCS 8465, Berlin: Springer-Verlag, 2014. 270−285.

[13] Troullinou G, Kondylakis H, Plexousakis D. Semantic partitioning for RDF datasets. In: Proc. of the 11th Int'l Workshop on Information Search, Integration, and Personlization. CCIS 760, Berlin: Springer-Verlag, 2016. 99−115.

[14] Xu Q. Semantic-Aware partitioning on RDF graphs. In: Proc. of the 1st Int'l Joint Conf. on Asia-Pacific Web and Web-Age Information Management. LNCS 10366, Berlin: Springer-Verlag, 2017. 149−157.

赵晓非(1978－),男,辽宁黑山人,博士,副教授,主要研究领域为语义 Web,分布智能.

田东平(1981－),男,博士,副教授,主要研究领域为语义 Web,人工智能.

史忠植(1941－),男,研究员,博士生导师,CCF 会士,主要研究领域为人工智能,语义 Web,机器学习.

刘建伟(1979－),女,讲师,主要研究领域为本体工程,语义 Web.