软件学报 ISSN 1000-9825, CODEN RUXUEW
Journal of Software,2018,29(10):2948−2965 [doi: 10.13328/j.cnki.jos.005550]

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

不协调本体调试与修复的冲突路径优化策略*

张 瑜[1,2], 欧阳丹彤[1,2], 叶育鑫[1,2]

[1](吉林大学 计算机科学与技术学院,吉林 长春 130012)

[2](符号计算与知识工程教育部重点实验室(吉林大学),吉林 长春 130012)

通讯作者: 叶育鑫, E-mail: yeyx@jlu.edu.cn

摘 要: 以基于黑盒法的不协调本体调试与修复技术为研究对象,分析了黑盒法及其优化方法在调试过程中所存在的问题.针对这一问题,提出了一种基于冲突路径的调试与修复策略,证明了该策略能够正确构造出与基本冲突模式相对应的冲突路径.将黑盒法调试目标限定在与该冲突路径相关的冲突集上,以此降低调试目标的规模从而提高调试的效率.进而根据构造出的冲突路径,获得不可满足依赖路径并基于该路径制定出不可满足概念的修复策略.理论证明与实验结果均证实了所提出的调试与修复策略的正确性与有效性.

关键词: 不协调本体;本体调试;本体修复;冲突路径;不可满足依赖路径

中图法分类号: TP18

中文引用格式: 张瑜,欧阳丹彤,叶育鑫.不协调本体调试与修复的冲突路径优化策略.软件学报,2018,29(10):2948−2965. http//www.jos.org.cn/1000-9825/5550.htm

英文引用格式: Zhang Y, Ouyang DT, Ye YX. Debugging and repairing incoherent ontologies based on the clash path. Ruan Jian Xue Bao/Journal of Software, 2018,29(10):2948−2965 (in Chinese). http.//www.jos.org.cn/1000-9825/5550.htm

Debugging and Repairing Incoherent Ontologies Based on the Clash Path

ZHANG Yu[1,2], OUYANG Dan-Tong[1,2], YE Yu-Xin[1,2]

[1](College of Computer Science and Technology, Jilin University, Changchun 130012, China)

[2](Key Laboratory of Symbolic Computation and Knowledge Engineering of Ministry of Education (Jilin University), Changchun 130012, China)

Abstract: This study focuses on the debugging and repairing techniques for incoherent ontology based on the black-box method and discusses the limitations of the existing black-box methods and their optimizations. To solve this problem, the study proposes a new strategy called clash path for debugging and repairing incoherent ontology. This strategy can construct the clash path related to the basis clash models and then identify the clash set based on the clash path. In this case, deubgging can be rapidly performed based on the clash set because the clash set is smaller than the original ontology. In addition, the unsatisfiable dependent path can be identified from the clash path and the repair set can be easily obtained on the basis of the unsatisfiable dependent path. The theoretic proofs and experimental evaluation demonstrate that the presented debugging and repairing strategies are correct and efficiency.

Key words: incoherent ontology; ontology debugging; ontology repairing; clash path; unsatisfiable dependent path

描述逻辑(description logics,简称 DLs)是知识表示的常用形式化语言,用描述逻辑表示的本体知识库由术语集(TBox)和断言集(ABox)两部分组成[1].本体一般由领域专家手工构建或计算机程序半自动构建,建成的本

* 基金项目: 国家自然科学基金(61672261, 61502199)
Foundation item: National Natural Science Foundation of China (61672261, 61502199)
本文由“本体工程与知识图谱”专题特约编辑漆桂林教授推荐.
收稿时间: 2017-07-20; 修改时间: 2017-11-08; 采用时间: 2018-01-24; jos 在线出版时间: 2018-02-08
CNKI 网络优先出版: 2018-02-08 11:55:43, http://kns.cnki.net/kcms/detail/11.2560.TP.20180208.1155.006.html

体被广泛应用于本体工程的各个方面[2−4].若术语集中概念的定义出现了逻辑冲突,则称该概念是不可满足的,该术语集是不协调的(incoherent)[5].本体调试的大多数工作都集中在如何解决本体术语集部分的不协调问题上[6],研究目标是期望提供有效的方法辅助本体建模人员实现逻辑冲突的快速定位和冲突解决方案的自动生成[7].

不协调本体调试的方法包括基于 Tableau 扩展策略的白盒法[8−11]与基于“扩张-收缩”策略的黑盒法[12−16]两种.Schlobach 等人首先提出了极小不可满足保持子术语集(minimal unsatisfiability-preserving sub-TBoxes,简称 MUPS)[8]并提供了白盒求解方法,接着又在文献[5]中将其扩展到黑盒求解方法.Baader 等人[9]则提出了类似 MUPS 的极小公理集合 MinA,并提供了求解 MinA 的白盒方法.Kalyanpur[12]提出了一个适用于解释任何逻辑蕴涵式的最小公理集合——辨解(justification),并提供了黑盒求解方法,而且证明了在不协调问题中,不可满足概念的 MUPS 与不可满足概念的辨解是相同的并可以相互转化.欧阳丹彤等人[7]提出的基于有序标签演算的概念 R- MUPS 白盒法能够有效确定不可满足概念的 MUPS 之间的覆盖特征,从而加快 MUPS 的求解.

白盒法虽然效率较高,但受制于本体语言类型并需要修改推理机内部的推理机制,不具有可移植性,因而缺乏通用性.黑盒法则独立于具体的推理机与本体语言类型,可移植并易于实施,因而应用比较广泛[12].但黑盒法需要多次调用推理机进行可满足性检测,当本体规模较大或不可满足概念较多时,效率较低.因此,针对黑盒法的“扩张”与“收缩”两个阶段,一系列优化工作相继展开.在扩张阶段,选择函数法[5,13,15]与模块化方法[14,16−18]用于选择一个与不可满足概念有直接或间接语法相关的一个候选子集.Ji 等人[15]提出了类似于选择函数的定向相关的方法求解辩解,并在文献[19]中提出了基于基本冲突模式的本体调试方法.Grau 等人[14]定义了一种局部模块用来获取与蕴涵有关的公理子集,Suntisrivaraporn 等人[16]证明了该模块能够涵盖所有的辨解.在收缩阶段,滑动窗口[12,20]与分治法[13,20]用于提高精确定位较小不可满足子集的效率.Horridge[19]采用组合优化的方式,同时将模块化方法与分治法用于辩解求解问题.Shchekotykhin 等人[21]经过对比实验,认为分治法比滑动窗口方法更具有优势.此外,Ji 在文献[22]中将现有的本体调试工具进行了测试.

本体调试的结果能够提供导致本体不协调的极小冲突集,删除或者修改冲突集中的部分公理就能使得不可满足概念变得可满足.那么,选择哪部分公理进行修复,以及采取何种策略确定修复的优先级,则是本体修复工作的主要研究目标.本体修复代表性的工作是 Kalyanpur[12]提出的根不可满足概念以及采取的公理分割的细粒度方法,细粒度方法修复的主要目的是为了尽可能地保留有用信息.类似地,Lam 等人[23]和 Du 等人[24]提出的细粒度的公理定位方法,也都是将调试问题深入到公理内部,从而能够定位导致概念不可满足的真正原因,为本体修复提供精确目标.

关于轻量级的描述逻辑 DL-Lite 本体的调试和修复两方面的工作,漆桂林小组[25−29]做出了较多较为重要的贡献,该小组利用 DL-Lite 本体的特殊结构特征,构造肯定与否定逻辑闭包形成图结构,基于 DL-Lite 逻辑闭包图,不但能够较快地完成本体调试,而且可以高效地实现本体修复.

1 基础理论与研究动机

1.1 描述逻辑ALCOI

描述逻辑提供了一系列构造算子用于构建复杂的概念表达式.描述逻辑 ALCOI 由γ(全集)、$\bot$(空集)、A(原子概念)、$\neg A$(原子否定概念)、$\cap$(概念合取)、$\cup$(概念析取)、$\{o_1,o_2,\ldots\}$(枚举概念)、$\exists r.A$(存在限定)、$\forall r.A$(全称限定)构成.其中,r 是原子角色或逆角色.ALCOI 复杂概念按如下方式构成.

$C,D\leftarrow\gamma|\bot|A|\neg A|\exists r.A|\forall r.A|\{o_1,o_2,\ldots\}|C\cap D|C\cup D|$.

ALCOI 术语集由概念包含公理($C\subseteq D$)或概念等价公理($C\equiv D$)组成,其中,C 和 D 都是 ALCOI 概念.

描述逻辑本体的语义是通过解释 I 体现的,I 是一个二元组$(\Delta^I,\bullet^I)$,其中,Δ^I 是一个非空集合,代表该领域所有个体,称为论域.映射$\bullet^I$将每个概念 A 映射为Δ^I的一个子集 $A^I\subseteq\Delta^I$,将每个角色 r 映射为Δ^I上的二元关系 $A^I\subseteq\Delta^I\times\Delta^I$.例如,$(\gamma)^I=\Delta^I,(\bot)^I=\varnothing,(C\subseteq D)^I=C^I\subseteq D^I,(C\equiv D)^I=(C^I=D^I)$.

根据描述逻辑解释,不可满足概念和不协调(incoherent)术语集的形式化定义如下[5].

定义 1(不可满足概念). 若概念 C 关于 TBox T 不可满足,当且仅当对于 T 的所有解释 I,都有$(C)^I=\varnothing$.

定义 2(不协调 TBox). 若 TBox T 中出现了不可满足概念,则 T 是不协调 TBox.

为了找到不协调产生的原因,Schlobach 等人[5]定义了极小不可满足保持子术语集(minimal unsatisfiability-preserving sub-TBoxes,简称 MUPS)来描述 TBox 中的不协调问题.

定义 3(MUPS). 设 C 是关于 T 的不可满足概念且 $T'\subseteq T$ 是 T 的一个子集,如果 C 关于 T'不可满足但是对于任意 T'的真子集 $T''\subset T'$,C 关于 T''都可满足,则 T'是 C 的 MUPS.

C 的 MUPS 集合记作 MUPS(T,C),T 中所有概念的 MUPS 集合记作 MUPS(T).

1.2 黑盒法

设 C 为 T 的某个不可满足概念,则求解 C 关于 T 的一个 MUPS 的黑盒法由“扩张”阶段与“收缩”阶段组成.在扩张阶段,算法首先生成一个初始为空的集合 M,然后从 T 中随机选择公理添加进 M 中,每次添加都要判断 C 是否关于 M 可满足,若可满足,则继续添加.若不可满足,则说明 C 的一个 MUPS 已经被包括进 M 中,此时扩张阶段结束.接下来进入收缩阶段,从 M 中随机选择公理删除,每次删除都要判断 C 是否关于 M 可满足,若仍然不可满足,则该删除的公理与 C 的不可满足性无关,删除即可.否则,该删除的公理与 C 的不可满足性有关,需要将其放回 M 中.遍历完全部的公理,则得到 C 的一个 MUPS.

一般情况下,不可满足概念会有多个 MUPS,因此需要使用 Kalyanpur 的碰集树方法[12]求解出所有的 MUPS.方法是,从刚求得的 MUPS 中随机选择一个公理α_i,将α_i从 T 中删除得到 $T'=T-\{\alpha_i\}$,然后采用上述方法再求得一个新的 MUPS,此时,新的 MUPS 一定与刚得到的 MUPS 不同,以此类推,则可求得全部的 MUPS.

为了刻画 MUPS 结构的复杂程度,将 MUPS 集合中元素的个数定义为 MUPS 的基数,每个 MUPS 元素中公理的个数定义为该 MUPS 的长度.

1.3 研究动机

选择函数法很容易选出与不可满足无关的冗余公理且需要反复调用外部推理机,而模块化方法抽取的模块过大会导致计算效率提升不明显[20].例如若有 T 如下:

α_1: $B_1\subseteq B_2$,α_2:$B_2\subseteq B_3$,…,α_{10}:$B_{10}\subseteq B_{11}$,α_{11}:$D_1\subseteq D_2$,α_{12}:$D_2\subseteq D_3$,…,α_{19}:$D_9\subseteq D_{10}$,α_{20}:$D_{10}\subseteq A$,

α_{21}:$C_1\subseteq B_1\cap D_1\cap\neg A$,$\alpha_{22}$:$C_2\subseteq C_1$,$\alpha_{23}$:$C_3\subseteq C_2$.

调用推理机进行可满足性检测,易得 C_1、C_2、C_3 为不可满足概念.下面使用选择函数方法求解 C_1 的 MUPS,在扩张过程中,由于 C_1 是由公理α_{21} 定义的,因此,$S_1=\{\alpha_{21}\}$,由于 B_1 和 D_1 与 C_1 语法相关,它们又分别由α_1 和α_{11}所定义,因而 $S_1=\{\alpha_{21},\alpha_1,\alpha_{11}\}$,可满足性检测发现 C_1 关于 S_1 可满足,因此扩张继续进行,直到 $S_1=\{\alpha_{21},\alpha_1,\ldots,\alpha_{20}\}$为止,此时 C_1 关于 S_1 不可满足.再经过收缩过程后得到 MUPS(C_1)={{$\alpha_{11},\ldots,\alpha_{21}$}},不难发现,{$\alpha_1,\ldots,\alpha_{10}$}是与 MUPS 无关的冗余公理,却被选择进来了.

Parsia 等人[30]总结了 3 种类型的基本冲突:原子类型、基数类型与数据类型.原子类型的冲突是指一个实例 a 同时属于某个概念θ与该概念的补集$\neg\theta$,即 $a\in\theta^I$ 且 $a\in(\neg\theta^I)$,则 $a\in\varnothing$.描述逻辑 ALCOI 的冲突属于原子类型的冲突,本文基于文献[19]给出的不可满足概念的模式提取出适合标准化的 ALCOI-TBox 本体调试的 4 种基本冲突模式.

(1) $\theta\subseteq\neg\theta$;

(2) $\neg\theta\subseteq\theta$;

(3) $C\subseteq\phi(\theta)$,$C\subseteq\phi(\neg\theta)$,其中,$\phi(\theta)\cap\phi(\neg\theta)\subseteq\bot$;

(4) $\phi(\theta)\subseteq\neg C$,$\phi(\neg\theta)\subseteq\neg C$,其中,$\gamma\subseteq\phi(\theta)\cup\phi(\neg\theta)$.

这里,$\phi(\theta)$与$\phi(\neg\theta)$分别表示含有概念θ与其互补的否定概念$\neg\theta$的复杂概念.前两类情形所体现的是否定概念$(\neg\theta)$的存在导致了概念的不可满足,后两类情形所体现的是一对互补概念$(\theta,\neg\theta)$的存在导致了概念的不可满足.倘若从否定概念与一对互补概念入手,获取与之有直接或间接关系的一个公理集,此时与不可满足概念有关的公理都将包括在该公理集中,那么求解 MUPS 就可以在该公理集上进行.

上例中,很容易发现一对互补概念$(A,\neg A)$,分别为 A 与$\neg A$ 构造出与它们有依赖关系的两条依赖关系路径:$A\rightarrow D_{10}\rightarrow D_9\rightarrow\ldots\rightarrow D_1\rightarrow C_1\rightarrow C_2\rightarrow C_3$ 和$\neg A\rightarrow C_1\rightarrow C_2\rightarrow C_3$.

获取关系路径上的公理得到 $Q=\{\alpha_{11},\ldots,\alpha_{23}\}$,这样可以有效避免冗余公理集$\{\alpha_1,\ldots,\alpha_{10}\}$被选择进来.如果 Q 是不协调的,则它是一个冲突集,那么 MUPS 的求解过程就基于冲突集 Q 而不是 T 之上进行,以此可以提高求解效率.若采取这种方法,从一个很大规模的 T 中获得一个较小规模的冲突集 Q,则求解效率的提高很是可观.

2 基于冲突路径的不协调本体调试

2.1 相关路径

设 $T=\{\alpha_1,\ldots,\alpha_n\}$表示有 n 个公理的 ALCOI-TBox.根据等价转换规则,将所有概念都化为否定范式,例如,$\neg(A\cup B)\Leftrightarrow\neg A\cap\neg B$.将所有概念等价公理转换为概念包含公理,例如,$C\equiv D\Leftrightarrow C\sqsubseteq D,D\sqsubseteq C$.设$\Sigma_l$ 与Σ_r 分别表示出现在 T 中公理左侧与右侧的概念和否定概念集合,则通过公理索引可以将每个公理与其两侧的概念和否定概念联系起来.

定义 4(公理索引). 公理α的索引形如 $I_\alpha=(\xi_\alpha^l,\alpha,\xi_\alpha^r)$,其中,$\xi_\alpha^l$ 与 ξ_α^r 分别表示出现在公理左边与右边的概念和否定概念集合.

基于公理索引,采用序列⟨…⟩的表示方式定义相关路径来建立公理之间的彼此联系.

定义 5(左相关路径). 设 x 为出现在公理α_0 左侧的概念或否定概念,当满足如下两个条件时,$P_l(x)=\langle I_{\alpha_0},I_{\alpha_1},\ldots,I_{\alpha_m}\rangle$是 x 的一个左相关路径.

(1) $\xi_{\alpha_{i-1}}^r\cap\xi_{\alpha_i}^l\neq\varnothing$;

(2) 不存在其他的 $P_l'(x)=\langle I_{\alpha_0},I_{\alpha_1},\ldots,I_{\alpha_n}\rangle$满足 $n>m$.

x 的 k 个左相关路径表示为 $G_l(x)=\bigcup_1^k P_l(x)_i$.

定义 6(右相关路径). 设 x 为出现在公理α_0 右侧的概念或否定概念,当满足如下两个条件时,$P_r(x)=\langle I_{\alpha_0},I_{\alpha_1},\ldots,I_{\alpha_m}\rangle$是 x 的一个右相关路径.

(1) $\xi_{\alpha_{i-1}}^l\cap\xi_{\alpha_i}^r\neq\varnothing$;

(2) 不存在其他的 $P_r'(x)=\langle I_{\alpha_0},I_{\alpha_1},\ldots,I_{\alpha_m}\rangle$满足 $n>m$.

x 的 k 个右相关路径表示为 $G_r(x)=\bigcup_1^k P_r(x)_i$.

定理 1. 设Ω是 $G_l(\neg\theta)$中的公理集合,φ是复杂概念,如果 $T|=\neg\theta\sqsubseteq\varphi,T|=\varphi\sqsubseteq\theta$,则一定有$\Omega|=\neg\theta\sqsubseteq\varphi,\Omega|=\varphi\sqsubseteq\theta$.

证明:由 $T|=\neg\theta\sqsubseteq\varphi,T|=\varphi\sqsubseteq\theta$可知,否定概念$\neg\theta$出现在 T 的某个公理的左侧,设该公理为α_0,根据定义 5 可以构造出$\neg\theta$的左相关路径 $G_l(\neg\theta)=\bigcup_1^k P_l(\neg\theta)_j=\bigcup_1^k\langle I_{\alpha_0},I_{\alpha_1},\ldots,I_{\alpha_m}\rangle_j$, 在 $P_l(\neg\theta)_j$ 中,$\neg\theta\in\xi_{\alpha_0}^l,\theta\in\xi_{\alpha_m}^r$.对于 $P_l(\neg\theta)_j$ 中每两个相邻的公理索引 $I_{\alpha_{i-1}}=(\xi_{\alpha_{i-1}}^l,\alpha_{i-1},\xi_{\alpha_{i-1}}^r)$与 $I_{\alpha_i}=(\xi_{\alpha_i}^l,\alpha_i,\xi_{\alpha_i}^r)$,由定义 5 可知,$\xi_{\alpha_{i-1}}^r\cap\xi_{\alpha_i}^l\neq\varnothing$,则必定存在一个概念或否定概念$\varepsilon$满足$\varepsilon\in\xi_{\alpha_{i-1}}^r$且$\varepsilon\in\xi_{\alpha_i}^l$, 因此,$\varepsilon$能够将 $I_{\alpha_{i-1}}$ 与 I_{α_i} 连接起来.此时从 $I_{\alpha_{i-1}}$ 与 I_{α_i} 中可以获得两个相邻公理α_{i-1} 与α_i 并存入集合Ω中.类似地,遍历所有相邻的公理索引便得到$\Omega=\{\alpha_0,\ldots,\alpha_m\}$,从而有$\Omega|=\neg\theta\sqsubseteq\varphi,\Omega|=\varphi\sqsubseteq\theta$.
证毕. □

定理 2. 设Ω是 $G_r(\neg\theta)$中的公理集合,φ是复杂概念,如果 $T|=\theta\sqsubseteq\varphi,T|=\varphi\sqsubseteq\neg\theta$,则一定有$\Omega|=\theta\sqsubseteq\varphi,\Omega|=\varphi\sqsubseteq\neg\theta$.

证明:由 $T|=\theta\sqsubseteq\varphi,T|=\varphi\sqsubseteq\neg\theta$可知,$\neg\theta$出现在 T 的某个公理的右侧,设该公理为α_0,根据定义 6 可以构造出$\neg\theta$的右相关路径 $G_r(\neg\theta)=\bigcup_1^k P_r(\neg\theta)_j=\bigcup_1^k\langle I_{\alpha_0},I_{\alpha_1},\ldots,I_{\alpha_m}\rangle_j$. 在 $P_r(\neg\theta)_j$ 中,$\neg\theta\in\xi_{\alpha_0}^r,\theta\in\xi_{\alpha_m}^l$.对于 $P_r(\neg\theta)_j$ 中每两个相邻的公理标签 $I_{\alpha_{i-1}}$ 与 I_{α_i},由定义 6 可知 $\xi_{\alpha_{i-1}}^l\cap\xi_{\alpha_i}^r\neq\varnothing$,则必定存在$\varepsilon$使得$\varepsilon\in\xi_{\alpha_{i-1}}^l$且$\varepsilon\in\xi_{\alpha_i}^r$, ε的存在使得从 $I_{\alpha_{i-1}}$ 与 I_{α_i} 中得到的两个相邻公理α_{i-1} 与α_i 具有依赖关系,将它们存入集合Ω中.类似地,遍历所有相邻的公理索引,得到$\Omega=\{\alpha_0,\ldots,\alpha_m\}$,从而有$\Omega|=\theta\sqsubseteq\varphi,\Omega|=\varphi\sqsubseteq\neg\theta$. □

定理 1 与定理 2 确保了冲突路径的方法能够找到前两种基本冲突模式的相关路径.

2.2 互补路径

定义 7(右互补路径). 设$(\theta,\neg\theta)$是出现在公理右侧的一对互补概念,$P_r(\theta)=\langle I^+_{\alpha_0}, I^+_{\alpha_1},\ldots, I^+_{\alpha_m}\rangle$, $P_r(\neg\theta)=\langle I^-_{\alpha'_0}, I^-_{\alpha'_1},\ldots, I^-_{\alpha'_n}\rangle$,其中,$I^+_{\alpha_m}=(\xi^{l+}_{\alpha_m},\alpha_m,\xi^{r+}_{\alpha_m})$, $I^-_{\alpha'_n}=(\xi^{l-}_{\alpha'_n},\alpha'_n,\xi^{r-}_{\alpha'_n})$.若$\xi^{l+}_{\alpha_m}\cap\xi^{l-}_{\alpha'_n}\neq\varnothing$,则$f_r(\theta,\neg\theta)=\{P_r(\theta),P_r(\neg\theta)\}$是$(\theta,\neg\theta)$的右互补路径.

$(\theta,\neg\theta)$的 k 个右互补路径表示为$F_r(\theta,\neg\theta)=\bigcup_1^k f_r(\theta,\neg\theta)$.

定义 8(左互补路径). 设$(\theta,\neg\theta)$是出现在公理左侧的一对互补概念,$P_l(\theta)=\langle I^+_{\alpha_0}, I^+_{\alpha_1},\ldots, I^+_{\alpha_m}\rangle$,$P_l(\neg\theta)=\langle I^-_{\alpha'_0}, I^-_{\alpha'_1},\ldots, I^-_{\alpha'_n}\rangle$,其中,$I^+_{\alpha_m}=(\xi^{l+}_{\alpha_m},\alpha_m,\xi^{r+}_{\alpha_m})$, $I^-_{\alpha'_n}=(\xi^{l-}_{\alpha'_n},\alpha'_n,\xi^{r-}_{\alpha'_n})$.若$\xi^{r+}_{\alpha_m}\cap\xi^{r-}_{\alpha'_n}\neq\varnothing$,则$F_l(\theta,\neg\theta)=\{P_l(\theta),P_l(\neg\theta)\}$是$(\theta,\neg\theta)$的左互补路径.

$(\theta,\neg\theta)$的 k 个左互补路径表示为$F_l(\theta,\neg\theta)=\bigcup_1^k f_l(\theta,\neg\theta)$.

定理 3. 设Ω_r是$F_l(\theta,\neg\theta)$中的公理集,如果$T|=C\sqsubseteq\phi(\theta)$,$T|=C\sqsubseteq\phi(\neg\theta)$,则一定有$\Omega_r|=C\sqsubseteq\phi(\theta)$,$\Omega_r|=C\sqsubseteq\phi(\neg\theta)$.

证明:由$T|=C\sqsubseteq\phi(\theta)$,$T|=C\sqsubseteq\phi(\neg\theta)$可知,$(\theta,\neg\theta)$出现在 T 中公理的右侧.根据定义 7,可得$F_r(\theta,\neg\theta)$.对于每一个$f_r(\theta,\neg\theta)\in F_r(\theta,\neg\theta)$,可以得到 $P_r(\theta)=\langle I^+_{\alpha_0}, I^+_{\alpha_1},\ldots, I^+_{\alpha_m}\rangle$,$P_r(\neg\theta)=\langle I^-_{\alpha'_0}, I^-_{\alpha'_1},\ldots, I^-_{\alpha'_n}\rangle$,且有$\theta\in\xi^{r+}_{\alpha_0}$,$\neg\theta\in\xi^{r-}_{\alpha'_0}$,$C\in\xi^{l+}_{\alpha_m}$,$C\in\xi^{l-}_{\alpha'_n}$.分别从$P_r(\theta)$与$P_r(\neg\theta)$获取相邻的公理索引并将其中的相邻公理存入$\Omega_r(\theta)$与$\Omega_r(\neg\theta)$中,从而得到$\Omega_r(\theta)=\{\alpha_0,\alpha_1,\ldots,\alpha_m\}$与$\Omega_r(\neg\theta)=\{\alpha'_0,\alpha'_1,\ldots,\alpha'_n\}$,则有$\Omega_r=\Omega_r(\theta)\cup\Omega_r(\neg\theta)$满足$\Omega_r|=C\sqsubseteq\phi(\theta)$,$\Omega_r|C\sqsubseteq\phi(\neg\theta)$. □

定理 4. 设Ω_l是$F_l(\theta,\neg\theta)$中的公理集,若$T|=\phi(\theta)\sqsubseteq\neg C$,$T|=\phi(\neg\theta)\sqsubseteq\neg C$,则一定有$\Omega_l|=\phi(\theta)\sqsubseteq\neg C$,$\Omega_l|=\phi(\neg\theta)\sqsubseteq\neg C$.

证明:由$T|=\phi(\theta)\sqsubseteq\neg C$,$T|=\phi(\neg\theta)\sqsubseteq\neg C$ 可知,$(\theta,\neg\theta)$出现在 T 中公理的左侧,根据定义 8,可得$F_l(\theta,\neg\theta)$.对于每一个 $f_l(\theta,\neg\theta)\in F_l(\theta,\neg\theta)$,可以得到 $P_l(\theta)=\langle I^+_{\alpha_0}, I^+_{\alpha_1},\ldots, I^+_{\alpha_m}\rangle$与 $P_l(\neg\theta)\langle I^-_{\alpha'_0}, I^-_{\alpha'_1},\ldots, I^-_{\alpha'_n}\rangle$,且有$\theta\in\xi^{l+}_{\alpha_0}$,$\neg\theta\in\xi^{l-}_{\alpha'_0}$,$\neg C\in\xi^{r+}_{\alpha_m}$,$\neg C\in\xi^{r-}_{\alpha'_n}$.分别从$P_l(\theta)$与$P_l(\neg\theta)$获取相邻的公理索引并将其中的相邻公理存入$\Omega_l(\theta)$与$\Omega_l(\neg\theta)$中,从而得到$\Omega_l(\theta)=\{\alpha_0,\alpha_1,\ldots,\alpha_m\}$与$\Omega_l(\neg\theta)=\{\alpha'_0,\alpha'_1,\ldots,\alpha'_n\}$,则有$\Omega_l=\Omega_l(\theta)\cup$与$\Omega_l(\neg\theta)$满足$\Omega_l|=\phi(\theta)\sqsubseteq\neg C$,$\Omega_l|=\phi(\neg\theta)\sqsubseteq\neg C$. □

定理 3 与定理 4 确保了冲突路径的方法能够找到后两种基本冲突模式的相关路径.

2.3 特殊情形的处理

(1) 特殊概念形式的处理.若公理中出现了枚举类型的概念,则根据文献[31]中的转换规则,将枚举类型概念转换为概念的析取形式.例如:{blue,green,red}转换为 blue∪green∪red.若公理中出现了全集符号γ与空集符号$\bot$,则将γ与$\bot$分别转换为两个互补概念的析取和合取形式.例如:$C\sqsubseteq\gamma\cap A$ 转换为 $C\sqsubseteq(X_1\cup\neg X_1)\cap A$,$C\sqsubseteq\bot\cap A$ 转换为 $C\sqsubseteq(X_2\cap\neg X_2)\cap A$.其中,$X_1$和 X_2为引入的与 TBox 中现有概念均不相同的新概念.

(2) 当 TBox 中出现循环公理时,解决方法是:在路径构造过程中,一旦相同的公理索引又一次出现时,则终止该路径的构造.

2.4 冲突路径

构造与 4 种基本冲突模式对应的否定路径与互补路径,是为了从中筛选与概念不可满足有关的冲突路径.

2.4.1 右冲突路径

定义 9(右冲突路径). 设Ω_r是来自$G_r(\neg\theta)$或$F_r(\theta,\neg\theta)$的公理集,若Ω_r是不协调的,则$G_r(\neg\theta)$与$F_r(\theta,\neg\theta)$分别是$\neg\theta$与$(\theta,\neg\theta)$的右冲突路径,并且Ω_r是$\neg\theta$或$(\theta,\neg\theta)$的冲突集.

算法 1 实现了获取不协调术语集 T 的右冲突路径的冲突集.

算法 1. GetRightClashSet(T,Σ_l,Σ_r).

输入:T,Σ_l,Σ_r //不协调术语集 T 及其公理左侧与右侧的概念与否定概念集;

输出:Q_r //T 的右冲突路径的冲突集.

1. $Q_r\neq\varnothing$
2. **for each** $\neg\theta$ **in** Σ_r //对于Σ_r中的每一个否定概念$\neg\theta$
3. $G_r(\neg\theta)\leftarrow T$ //根据定义 6 构造出$\neg\theta$的右相关路径 $G_r(\neg\theta)$

4. $\Omega_r \leftarrow G_r(\neg\theta)$ //获得 $G_r(\neg\theta)$中的公理,存入Ω_r
5. **if** Ω_r 不协调 **then** //如果Ω_r 是不协调的,那么
6. $Q_r=Q_r\cup\Omega_r$ //将Ω_r 中的公理存入 Q_r
7. **if** $\theta\in\Sigma_r$ **then** //若θ也出现在某个公理的右侧
8. $F_r(\theta,\neg\theta_r)\leftarrow T$ //根据定义 7 构造出$(\theta,\neg\theta)$的右互补路径
9. $\Omega_r \leftarrow F_r(\theta,\neg\theta_r)$ //获得 $F_r(\theta,\neg\theta_r)$中的公理,存入Ω_r
10. **if** Ω_r 不协调 **then** //如果Ω_r 是不协调的,则
11. $Q_r=Q_r\cup\Omega_r$ //将Ω_r 中的公理存入 Q
12. **return** Q_r //输出 Q_r

定理 5. 设 Q_r 是从 T 的右冲突路径获得的冲突集,C 是 T 的一个不可满足概念,当 $T|=\theta\sqsubseteq\varphi,T|=\varphi\sqsubseteq\neg\theta$,且$T|=C\sqsubseteq\theta$时,概念 C 关于 Q_r 是不可满足的.

证明:考虑 $C=\theta$与 $C\neq\theta$两种情况.

a) $C=\theta$.此时有 $T|=C\sqsubseteq\varphi,T|=\varphi\sqsubseteq\neg C$,由定义 6 构造出$\neg C$ 的右相关路径 $G_r(\neg C)$.设Ω_r 为 $G_r(\neg C)$中的公理集合,根据定理 2 可得$\Omega_r|=C\sqsubseteq\varphi,\Omega_r|=\varphi\sqsubseteq\neg C$,因为$\{C\sqsubseteq\varphi,\varphi\sqsubseteq\neg C\}|=C\sqsubseteq\neg C|=C\sqsubseteq\bot$,而$\Omega_r\subseteq Q_r$,所以 $Q_r|=C\sqsubseteq\bot$.

b) $C\neq\theta$.由定义 6 构造出$\neg\theta$的右相关路径 $G_r(\neg\theta)=\bigcup_1^k P_r(\neg\theta)_j=\bigcup_1^k\langle I_{\alpha_0},I_{\alpha_1},...,I_{\alpha_m}\rangle_j$, 其中,$\neg\theta\in\xi^r_{\alpha_0}$,$\theta\in\xi^l_{\alpha_m}$. 设$\Omega_\theta$为 $G_r(\neg\theta)$中的公理集合,根据定理 2 可得$\Omega_\theta|=\theta\sqsubseteq\varphi,\Omega_\theta|=\varphi\sqsubseteq\neg\theta$,因为$\{\theta\sqsubseteq\varphi,\varphi\sqsubseteq\neg\theta\}|=\theta\sqsubseteq\bot$,因此,$\Omega_\theta|=\theta\sqsubseteq\bot$.又因为$T|=C\sqsubseteq\theta$,将 $G_r(\neg\theta)$扩展为 $G_r^*(\neg\theta)=\bigcup_1^k P_r(\neg\theta)_j=\bigcup_1^k\langle I_{\alpha_0},I_{\alpha_1},...,I_{\alpha_m},...,I_{\alpha_n}\rangle_j$, 此时有 $\theta\in\xi^l_{\alpha_m}$, $C\in\xi^l_{\alpha_n}$. 设Ω_r 为 $G_r^*(\neg\theta)$ 中的公理集合,则有$\Omega_r=\Omega_\theta\cup\bigcup_1^k\{\alpha_{m+1},...,\alpha_n\}_j$, 因为$\Omega_r\subseteq Q_r$,所以 $Q_r|=C\sqsubseteq\bot$. □

定理 5 证明了从第 1 种基本冲突模式的冲突路径中获得的公理集确实是一个冲突集.

定理 6. 设 Q_r 是从 T 的右冲突路径获得的冲突集,C 是 T 的一个不可满足概念,当 $T|=C\sqsubseteq\phi(\theta),T|=C\sqsubseteq\phi(\neg\theta)$时,其中,$\phi(\theta)\cap\phi(\neg\theta)\sqsubseteq\bot$,则概念 C 关于 Q_r 是不可满足的.

证明:

由定义 7 构造出$(\theta,\neg\theta)$的右互补路径 $F_r(\theta,\neg\theta)$,设Ω_r 是 $F_r(\theta,\neg\theta)$中的公理集,根据定理 3 可以得到$\Omega_r|=C\sqsubseteq\phi(\theta),\Omega_r|=C\sqsubseteq\phi(\neg\theta)$.又因为$\phi(\theta)\cap\phi(\neg\theta)\sqsubseteq\bot$,则有$\Omega_r|=C\sqsubseteq\bot$.

因为$\Omega_r\subseteq Q_r$,所以 $Q_r|=C\sqsubseteq\bot$.即概念 C 关于 Q_r 是不可满足的. □

定理 6 证明了从第 3 种基本冲突模式的冲突路径中获得的公理集确实是一个冲突集.

2.4.2 左冲突路径

定义 10(否定型左冲突路径). 设$\Omega_{\neg\theta}$是 $G_l(\neg\theta)$中的公理集,若$\theta\in\Sigma_r$,设Ω_θ是 $G_r(\theta)$中的公理集,则当$\Omega_{\neg\theta}\cup\Omega_\theta$不协调时,$G_l(\neg\theta)\cup G_r(\theta)$是$\neg\theta$的否定型左冲突路径.

算法 2 实现了获取不协调术语集 T 的否定型左冲突路径的冲突集.

算法 2. GetNegLeftClashSet(T,Σ_l,Σ_r).

输入:T,Σ_l,Σ_r //不协调术语集 T 及其公理左侧与右侧的概念与否定概念集;

输出:$Q_{\neg\theta}$ //T 的否定型左冲突路径的冲突集.

1. $Q_{\neg\theta}=\varnothing$
2. **for each** $\neg\theta$ **in** Σ_l //对于Σ_l 中的每一个否定概念$\neg\theta$
3. $G_l(\neg\theta)\leftarrow T$ //根据定义 5 构造出$\neg\theta$的左相关路径 $G_l(\neg\theta)$
4. $\Omega_{\neg\theta}\leftarrow G_l(\neg\theta)$ //获得 $G_l(\neg\theta)$中的公理,存入$\Omega_{\neg\theta}$
5. **if** $\theta\in\Sigma_r(\Omega_{\neg\theta})$ **then** //若θ也出现在$\Omega_{\neg\theta}$中某个公理的右侧
6. $G_r(\theta)\leftarrow T$ //根据定义 6 构造出θ的右相关路径 $G_r(\theta)$
7. $\Omega_\theta\leftarrow G_r(\theta)$ //获得 $G_r(\theta)$中的公理,存入Ω_θ
8. **if** $\Omega_{\neg\theta}\cup\Omega_\theta$不协调 **then** //如果$\Omega_{\neg\theta}\cup\Omega_\theta$是不协调的,则
9. $Q_{\neg\theta}\leftarrow(\Omega_{\neg\theta}\cup\Omega_\theta)$ //将$\Omega_{\neg\theta}$和Ω_θ中的公理存入 $Q_{\neg\theta}$

10. **else if** $Q_{\neg\theta}$ 不协调 **then** //如果$\Omega_{\neg\theta}$是不协调的,那么

11. $Q_{\neg\theta}\leftarrow\Omega_{\neg\theta}$ //将$\Omega_{\neg\theta}$中的公理存入 $Q_{\neg\theta}$

12. **return** $Q_{\neg\theta}$ //输出 $Q_{\neg\theta}$

定理 7. 设 $Q_{\neg\theta}$是从 T 的左冲突路径获得的冲突集,C 是 T 的一个不可满足概念,当 $T|=\neg\theta\sqsubseteq\varphi,T|=\varphi\sqsubseteq\theta,$且$T|=C\sqsubseteq\theta$时,概念 C 关于 $Q_{\neg\theta}$是不可满足的.

证明:考虑 $C=\theta$与 $C\neq\theta$两种情况.

a) $C=\theta$.此时有 $T|=\neg C\sqsubseteq\varphi,T|=\varphi\sqsubseteq C,$由定义 5 构造出$\neg C$ 的左相关路径 $G_l(\neg C)$.设Ω_l 为 $G_l(\neg C)$中的公理集合,根据定理 1 可得$\Omega_l|=\neg C\sqsubseteq\varphi,\Omega_l|=\varphi\sqsubseteq C,$因为$\{\neg C\sqsubseteq\varphi,\varphi\sqsubseteq C\}|=\neg C\sqsubseteq C|=C\sqsubseteq\bot,$因此,$\Omega_l|=C\sqsubseteq\bot$.

b) $C\neq\theta$.由定义 5 构造出$\neg\theta$的左相关路径 $G_l(\neg\theta)$,根据定理 1 可得$\Omega_{\neg\theta}|=\neg\theta\sqsubseteq\varphi,\Omega_{\neg\theta}|=\varphi\sqsubseteq\theta,$因为$\{\neg\theta\sqsubseteq\varphi,\varphi\sqsubseteq\theta\}|=\theta\sqsubseteq\bot,$因此$\Omega_{\neg\theta}|=\theta\sqsubseteq\bot$.

又因为 $T|=C\sqsubseteq\theta,$则由定义 6 可以构造出 $G_r(\theta)$.设Ω_θ为 $G_r(\theta)$中的公理集,则有$\Omega_\theta|=C\sqsubseteq\theta$.

设$\Omega_l=\Omega_{\neg\theta}\cup\Omega_\theta,$则有$\Omega_l|=\theta\sqsubseteq\bot$且$\Omega_l|=C\sqsubseteq\theta,$因此$\Omega_l|=C\sqsubseteq\bot$.

在两种情况下,都有$\Omega_l|=C\sqsubseteq\bot$.因为$\Omega_l\sqsubseteq Q_{\neg\theta},$所以 $Q_{\neg\theta}|=C\sqsubseteq\bot$.即概念 C 关于 $Q_{\neg\theta}$是不可满足的. □

定理 7 证明了从第 2 种基本冲突模式的冲突路径中获得的公理集确实是一个冲突集.

定义 11(互补型左冲突路径). 设$\Omega_{(\theta,\neg\theta)}$是 $F_{l(\theta,\neg\theta)}$中的公理集,再设δ为$\Omega_{(\theta,\neg\theta)}$中公理右侧的概念与否定概念.对于每一个$\upsilon\in\delta,$若$\upsilon$的补集$\upsilon^*\in\Sigma_r,$设$\Omega_{\upsilon^*}$是 $G_r(\upsilon^*)$中的公理集,则当$\Omega_{(\theta,\neg\theta)}\cup\Omega_{\upsilon^*}$不协调时,$F_{l(\theta,\neg\theta)}$和 $G_r(\upsilon^*)$是$(\theta,\neg\theta)$的互补型左冲突路径.

算法 3 实现了获取不协调术语集 T 的互补型左冲突路径的冲突集.

算法 3. GetCompLeftClashSet(T,Σ_l,Σ_r).

输入:T,Σ_l,Σ_r //不协调术语集 T 及其公理左侧与右侧的概念与否定概念集;

输出:$Q_{(\theta,\neg\theta)}$ //T 的互补型左冲突路径的冲突集.

1. $Q_{(\theta,\neg\theta)}=\varnothing$
2. **for each** $(\theta,\neg\theta)$ **in** Σ_l //对于Σ_l中的每一对互补概念$(\theta,\neg\theta)$
3. $F_{l(\theta,\neg\theta)}\leftarrow T$ //根据定义 8 构造出$(\theta,\neg\theta)$的左互补路径 $F_{l(\theta,\neg\theta)}$
4. $\Omega_{(\theta,\neg\theta)}\leftarrow F_{l(\theta,\neg\theta)}$ //获得 $F_{l(\theta,\neg\theta)}$中的公理,存入$\Omega_{(\theta,\neg\theta)}$
5. $\delta\leftarrow\Omega_{(\theta,\neg\theta)}$ //获得$\Omega_{(\theta,\neg\theta)}$中公理右侧的概念与否定概念
6. **for each** $\upsilon\in\delta$ //对于δ中的每一个概念或否定概念
7. **if** $\upsilon^*\in\Sigma_r$ **then** //若υ的补集υ^*也出现在Σ_r中
8. $G_r(\upsilon^*)\leftarrow T$ //根据定义 6 构造出υ^*的右相关路径 $G_r(\upsilon^*)$
9. $\Omega_{\upsilon^*}\leftarrow G_r(\upsilon^*)$ //获得 $G_r(\upsilon^*)$中的公理,存入Ω_{υ^*}
10. **if** $\Omega_{(\theta,\neg\theta)}\cup\Omega_{\upsilon^*}$不协调 **then** //如果$\Omega_{(\theta,\neg\theta)}\cup\Omega_{\upsilon^*}$是不协调的,则
11. $Q_{(\theta,\neg\theta)}\leftarrow(\Omega_{(\theta,\neg\theta)}\cup\Omega_{\upsilon^*})$ //将$\Omega_{(\theta,\neg\theta)}$和$\Omega_{\upsilon^*}$中的公理存入 $Q_{(\theta,\neg\theta)}$
12. **else if** $\upsilon^*\notin\Sigma_r$ 且$\Omega_{(\theta,\neg\theta)}$不协调 **then** //如果$\upsilon^*\notin\Sigma_r$且只有$\Omega_{(\theta,\neg\theta)}$不协调,那么
13. $Q_{(\theta,\neg\theta)}\leftarrow\Omega_{(\theta,\neg\theta)}$ //将$\Omega_{(\theta,\neg\theta)}$中的公理存入 $Q_{(\theta,\neg\theta)}$
14. **return** $Q_{(\theta,\neg\theta)}$ //输出 $Q_{(\theta,\neg\theta)}$

定理 8. 设 $Q_{(\theta,\neg\theta)}$是算法 3 求出的冲突集,C 是 T 的一个不可满足概念,当 $T|=\phi(\theta)\sqsubseteq\upsilon,T|=\phi(\neg\theta)\sqsubseteq\upsilon,T|=C\sqsubseteq\upsilon^*$且$\upsilon\cap\upsilon^*\sqsubseteq\bot$时,其中,$\gamma\sqsubseteq\phi(\theta)\cup\phi(\neg\theta),$则概念 C 关于 $Q_{(\theta,\neg\theta)}$是不可满足的.

证明:由定义 8 构造出$(\theta,\neg\theta)$的左互补路径 $F_l(\theta,\neg\theta),$设Ω_l 是 $F_l(\theta,\neg\theta)$中的公理集,根据定理 4 可以得到$\Omega_{(\theta,\neg\theta)}|=\phi(x)\sqsubseteq\upsilon,\Omega_{(\theta,\neg\theta)}|=\neg\phi(x)\sqsubseteq\upsilon$.又因为$\gamma\sqsubseteq\phi(x)\cup\neg\phi(x),$则有$\Omega_{(\theta,\neg\theta)}|=\gamma\sqsubseteq\upsilon,$即$\Omega_{(\theta,\neg\theta)}|=\neg\upsilon\sqsubseteq\bot$.

又因为 $T|=C\sqsubseteq\upsilon^*,$则由定义 6 可以构造出 $G_r(\upsilon^*),$设Ω_{υ^*}为 $G_r(\upsilon^*)$中的公理集,则有$\Omega_{\upsilon^*}|=C\sqsubseteq\upsilon^*$.

设$\Omega_l=\Omega_{(\theta,\neg\theta)}\cup\Omega_{\upsilon^*},$则有$\Omega_l|=\neg\upsilon\sqsubseteq\bot$且$\Omega_l|=C\sqsubseteq\upsilon^*$.因为$\upsilon\cap\upsilon^*\sqsubseteq\bot,$则有$\upsilon^*\sqsubseteq\neg\upsilon,$于是$\Omega_l|=\upsilon^*\sqsubseteq\bot$且$\Omega_l|=C\sqsubseteq\upsilon^*,$因此,$\Omega_l|=C\sqsubseteq\bot$.因为$\Omega_l\sqsubseteq Q_{(\theta,\neg\theta)},$所以 $Q_{(\theta,\neg\theta)}|=C\sqsubseteq\bot$.即概念 C 关于 $Q_{(\theta,\neg\theta)}$是不可满足的. □

定理 8 证明了从第 4 种基本冲突模式的冲突路径中获得的公理集确实是一个冲突集.

定理 9. 设 Q 是从 T 中获得的冲突集,对于 T 中的任意不可满足概念 C,则 C 关于 T 不可满足当且仅当 C 关于 Q 不可满足.

证明:

(1) C 关于 Q 不可满足$\Rightarrow C$ 关于 T 不可满足.

因为 Q 是从 T 中获得的冲突集,因而 $Q\subseteq T$,所以 C 关于 Q 不可满足$\Rightarrow C$ 关于 T 不可满足.

(2) C 关于 T 不可满足$\Rightarrow C$ 关于 Q 不可满足.

使用反证法证明,假设存在一个概念 C_k 关于 T 不可满足,但是 C_k 关于 Q 是可满足的.考虑 4 种情况.

a) $T|=\theta\sqsubseteq\varphi, T|=\varphi\sqsubseteq\neg\theta$,且 $T|=C_k\sqsubseteq\theta$;

b) $T|=C_k\sqsubseteq\phi(\theta), T|=C_k\sqsubseteq\phi(\neg\theta)$,其中,$\phi(\theta)\cap\phi(\neg\theta)\sqsubseteq\bot$;

对于 a)与 b),可由算法 1 求得冲突集 Q,由定理 5 与定理 6 可知 C_k 关于 Q 是不可满足的.

c) $T|=\neg\theta\sqsubseteq\varphi, T|=\varphi\sqsubseteq\theta$,且 $T|=C_k\sqsubseteq\theta$;

对于 c)可由算法 2 求得冲突集 Q,由定理 7 可知 C_k 关于 Q 是不可满足的.

d)$T|=\phi(\theta)\sqsubseteq\upsilon, T|=\phi(\neg\theta)\sqsubseteq\upsilon, T|=C_k\sqsubseteq\upsilon^*$且$\upsilon\cap\upsilon^*\sqsubseteq\bot$,其中,$\gamma\sqsubseteq\phi(\theta)\cup\phi(\neg\theta)$.

对于 d)可由算法 3 求得冲突集 Q,由定理 8 可知 C_k 关于 Q 是不可满足的.

无论哪种情况,C_k 关于 Q 都是不可满足的,与假设相矛盾.

因此 C 关于 T 不可满足$\Rightarrow C$ 关于 Q 不可满足. □

定理 10. 设 Q 是从 T 中获得的冲突集,对于 T 中的任意不可满足概念 C,设 $M_T(C)$为 C 关于 T 的某一个 MUPS,则一定存在 C 关于 Q 的一个 MUPS 为 $M_Q(C)$使得 $M_T(C)=M_Q(C)$.

证明:根据黑盒法求解 MUPS 的“扩-缩”过程进行证明.

设 $T=P\cup Q$,且 $P\cap Q=\varnothing$.根据定理 9 可得 $T|=C\sqsubseteq\bot\Leftrightarrow Q|=C\sqsubseteq\bot$,因此 $P|\neq C\sqsubseteq\bot$.

(1) 扩张阶段

设$\Sigma_T=\varnothing$和$\Sigma_Q=\varnothing$分别为 T 和 Q 的扩张初始集合,当第 1 个公理α_1 添加进Σ_T时,考虑两种情形.

(1) $\alpha_1\in Q$.此时,也将α_1 同时添加进Σ_Q 中,从而得到$\Sigma_T=\{\alpha_1\}$与$\Sigma_Q=\{\alpha_1\}$.

(2) $\alpha_1\in P$.此时,则有$\Sigma_T=\{\alpha_1\}$与$\Sigma_Q=\varnothing$.

设第 k 个公理α_k 添加进Σ_T 后,C_k 关于Σ_T 变得不可满足,即$\Sigma_T|=C\sqsubseteq\bot$.此时将$\alpha_k$ 也添加进Σ_Q 中,则有$\Sigma_Q|=C\sqsubseteq\bot$,这时扩张阶段结束.此时得到的$\Sigma_T$ 和Σ_Q 满足$\Sigma_T|=C\sqsubseteq\bot\Leftrightarrow\Sigma_Q|=C\sqsubseteq\bot$,且有$\Sigma_Q\subseteq\Sigma_T$.

(2) 收缩阶段

设$\Sigma_T=\Sigma_P\cup\Sigma_Q$,且$\Sigma_P\cap\Sigma_Q=\varnothing$.由于$\Sigma_T|=C\sqsubseteq\bot\Leftrightarrow\Sigma_Q|=C\sqsubseteq\bot$,则有$\Sigma_P|\neq C\sqsubseteq\bot$.

在收缩阶段,设 Σ'_T 为从Σ_T 中删除某个公理α_x 后得到的公理集,考虑两种情形.

(1) $\alpha_x\in\Sigma_Q$.此时,也将α_x 从Σ_Q 删除,设 Σ'_Q 为从Σ_Q 中删除α_x 后得到的公理集,这时出现两种情况.

(a) $\Sigma'_T|=C\sqsubseteq\bot$.则 $\Sigma'_Q|=C\sqsubseteq\bot$,这种情况表明$\alpha_x$ 与 C 的不可满足性无关,因为删除α_x 后 C 关于 Σ'_T 或 Σ'_Q 仍然是不可满足的.

(b) $\Sigma'_T|\neq C\sqsubseteq\bot$,则 $\Sigma'_Q|\neq C\sqsubseteq\bot$,这种情况表明$\alpha_x$ 与 C 的不可满足性有关,因为删除α_x 后 C 由不可满足概念变成可满足概念了.那么,α_x 不可删除,因而将α_x 再次添加进 Σ'_T 和 Σ'_Q 中,则有 $\Sigma'_T=\Sigma'_T\cup\{\alpha_x\}$和 $\Sigma'_Q=\Sigma'_Q\cup\{\alpha_x\}$.

(2) $\alpha_x\in\Sigma_P$.此时,Σ_Q 没有变化.由于$\Sigma_P|\neq C\sqsubseteq\bot$且$\Sigma_P\cap\Sigma_Q=\varnothing$,则有 $\Sigma'_T|=C\sqsubseteq\bot$和$\Sigma_Q|=C\sqsubseteq\bot$.

Σ'_T 和 Σ'_Q 中的所有公理都检测完毕后,则有 $\Sigma'_T=\Sigma'_Q$.

黑盒法的“扩-缩”两个阶段完成后,可以得到 $M_T(C)=\Sigma'_T$,$M_Q(C)=\Sigma'_Q$, 因而有 $M_T(C)=M_Q(C)$. □

定理 11. 设 Q 是从 T 中获得的冲突集,对于 T 中的任意不可满足概念 C,则有 MUPS(Q,C)=MUPS(T,C).

证明:设 MUPS(T,C)=$\{M_T(C)_1,\ldots,M_T(C)_m\}$是由 MUPS_HST 算法[12](关于 MUPS_HST 算法的介绍请参考第 1.2 节)求得的 C 关于 T 的 MUPS,m 是 MUPS 的个数.对于任意一个 $M_T(C)_i$,根据定理 10,可求得关于 Q 的一个

MUPS 为 $M_Q(C)_i$ 满足 $M_T(C)_i=M_Q(C)_i$.因此,MUPS(Q,C)=$\bigcup_1^m M_Q(C)_i$ 满足 MUPS(Q,C)=MUPS(T,C). □

算法 4 实现了从获取的冲突集上使用基于碰集树的黑盒法求解所有不可满足概念的所有 MUPS.

算法 4. CSMUPS(T).

输入:T //不协调术语集 T;

输出:MUPS(T) //T 的所有不可满足概念的所有 MUPS.

1. 标准化 T
2. **for each** α_i **in** T //对于 T 中的每一个公理α_i
3. $\Sigma_l,\Sigma_r\leftarrow\alpha_i$ //将α_i的左(右)侧的概念与否定概念分别存入Σ_l,Σ_r
4. Q_r=*GetRightClashSet*(T,Σ_l,Σ_r) //求出右冲突路径的冲突集 Q_r
5. $Q_{\neg\theta}$=*GetNegLeftClashSet*(T,Σ_l,Σ_r) //求出否定型左冲突路径的冲突集 $Q_{\neg\theta}$
6. $Q_{(\theta,\neg\theta)}$=*GetCompLeftClashSet*(T,Σ_l,Σ_r) //求出互补型左冲突路径的冲突集 $Q_{(\theta,\neg\theta)}$
7. $Q=Q_r\cup Q_{\neg\theta}\cup Q_{(\theta,\neg\theta)}$ //获得 T 的冲突集 Q
8. $U\leftarrow$reasoner(Q) //调用推理机求出 Q 中的不可满足概念集
9. **for each** C_i **in** U //对于 U 中的每一个不可满足概念 C_i
10. MUPS(Q,C_i)=Black-box(Q) //使用基于碰集树的黑盒法求解 C_i 的所有 MUPS
11. MUPS(T)$\leftarrow$MUPS(Q,C_i) //将求得的 MUPS(Q,C_i)存入 MUPS(T)
12. **return** MUPS(T) //输出 MUPS(T)

定理 12. 设 Q 是从 T 中获得的冲突集,则有 MUPS(Q)=MUPS(T).

证明:设 U 是 T 中的不可满足概念集,设$|U|=n$,对于 U 中的每一个不可满足概念 C_i,根据定理 11 可知,C_i 关于 T 不可满足当且仅当 C_i 关于 Q 不可满足.因此,对于任意不可满足概念 C_i,则有 MUPS(Q,C_i)=MUPS(T,C_i).那么,$\bigcup_1^n$ MUPS(Q,C_i)=$\bigcup_1^n$ MUPS(T,C_i),即 MUPS(Q)=MUPS(T). □

定理 1~定理 4 证明了能够找到 4 种基本冲突模式的相关路径;定理 5~定理 8 基于定理 1~定理 4 证明了从 4 种基本冲突模式的冲突路径中获得的公理集确实是一个冲突集;定理 9 基于定理 5~定理 8 证明了在冲突集 Q 上,不可满足概念 C 在 T 中不可满足当且仅当 C 在 Q 上不可满足;定理 10 基于定理 9 证明了对于 C 的其中一个 MUPS,一定可以在 Q 中求出这个 MUPS;定理 11 基于定理 10 证明了对于 C 的所有 MUPS,一定可以在 Q 中求出 C 的所有 MUPS;定理 12 基于定理 11 证明了对于 T 中所有的不可满足概念的所有 MUPS,一定可以在 Q 求出所有不可满足概念的所有 MUPS.

3 基于冲突路径的不协调本体修复

第 2 节中,我们阐述了如何采用冲突路径的方法从不协调 TBox 中获取冲突集,进而使得 MUPS 的求解过程限定在冲突集上从而有效减少本体的规模.本节我们继续使用冲突路径的方法进一步从求得的 MUPS 中获取精确的修复集提供给专家进行修复,同时尽可能避免本体信息内容的损失.

引例:若有 T 包括公理:α_1:$C\subseteq A$,α_2:$A\subseteq E_1\cap D_1$,α_3:$D_1\cup E_2\subseteq D_2$,α_4:$D_2\subseteq E_3\cap\neg A$,$\alpha_5$:$E_1\subseteq E_3$.

易知 C 和 A 是不协调 T 的两个不可满足概念,C 的 MUPS 为 MUPS(T,C)={{α_1,α_2,α_3,α_4}},A 的 MUPS 为 MUPS(T,A)={{α_2,α_3,α_4}}.删除{α_1,α_2,α_3,α_4}中的任意一个公理,则 C 变得可满足,删除{α_2,α_3,α_4}中的任意一个公理,则 C 和 A 都变得可满足.进一步分析发现:

(1) 由 $C\subseteq A$ 可知,C 的不可满足性是由 A 造成的,完成了对 A 的修复,也就完成了对 C 的修复,因而无需再专门对 C 进行修复.(2) A 不可满足的原因是由 $S1$={$A\subseteq D_1$,$D_1\subseteq D_2$,$D_2\subseteq\neg A$}这 3 部分公理造成的,而另外的部分公理 $S2$={$A\subseteq E_1$,$E_2\subseteq D_2$,$D_2\subseteq E_3$}与 A 的不可满足性无关,删除它们会造成本体信息的损失.第 1 种情况对应于 Kalyanpur[12]所定义的根不可满足概念与派生不可满足概念.如果能够从所有不可满足概念中筛选出根不可满足概念,将修复目标定位于根不可满足概念,则可以大大提高修复效率.对于第 2 种情况,为了尽可能避免本体信息内容的损失,将 MUPS 中的公理分割成与不可满足性有关和无关的两部分,修复操作仅针对有关的部分,无关

的部分则予以保留.

为了实现以上目标,借助于第 2 节所描述的冲突路径的良好结构,很容易筛选出根不可满足概念,同时获取“$\neg A\rightarrow D_2\rightarrow D_1\rightarrow A$”这种形式的不可满足依赖关系,基于这一依赖关系,采用公理分割的方式将 MUPS 中的公理分割成与不可满足性有关的 $S1$ 部分以及与不可满足性无关的 $S2$ 部分,本体修复仅针对 $S1$ 部分实施,而 $S2$ 部分则可以保留,从而避免本体信息内容的损失.

3.1 不可满足概念的根依赖路径

对于 T 中的任意不可满足概念 C,设 $M_T(C)$为 C 关于 T 的某一个 MUPS,使用第 2 节所描述的方法,构造出 $M_T(C)$的冲突路径,借助于冲突路径获取概念依赖路径.

定义 12(概念右依赖路径). 设 $P_r(x)$为 $M_T(C)$的一条右冲突路径,令 $P_r(x)=\langle(\xi^l_{\alpha_0},\alpha_0,\xi^r_{\alpha_0}),\ldots,(\xi^l_{\alpha_m},\alpha_m,\xi^r_{\alpha_m})\rangle$,设 $\varepsilon_i=\xi^r_{\alpha_{i-1}}\cap\xi^l_{\alpha_i}$, 则 $R_r(x)=\langle x,\varepsilon_1,\ldots,\varepsilon_m,C\rangle$为 $P_r(x)$的概念右依赖路径.

令 $R_r(C)=\{R_r(x)_1,\ldots,R_r(x)_n\}$为 $M_T(C)$所有的概念右依赖路径.

定义 13(概念左依赖路径). 设 $P_l(x)$与 $P_r(y)$分别为 $M_T(C)$的左冲突路径(左冲突路径包括否定型左冲突路径与互补型左冲突路径两种)中的一条左相关路径和右相关路径,令

$$P_l(x)=\langle(\xi^l_{\alpha_0},\alpha_0,\xi^r_{\alpha_0}),\ldots,(\xi^l_{\alpha_m},\alpha_m,\xi^r_{\alpha_m})\rangle,$$

$$P_r(y)=\langle(\xi^l_{\alpha_0*},\alpha^*_0,\xi^r_{\alpha_0*}),\ldots,(\xi^l_{\alpha_k*},\alpha^*_k,\xi^r_{\alpha_k*})\rangle.$$

设$\varepsilon_i=\xi^l_{\alpha_{i-1}}\cap\xi^r_{\alpha_i}$, $\varepsilon_{i^*}=\xi^r_{\alpha_{i-1}*}\cap\xi^l_{\alpha_i*}$, 则 $R_l(x)=\langle x,\varepsilon_1,\ldots,\varepsilon_m,\varepsilon_{1^*},\ldots,\varepsilon_{m^*},C\rangle$为 $P_l(x)$的概念左依赖路径.

令 $R_l(C)=\{R_l(x)_1,\ldots,R_l(x)_n\}$为 $M_T(C)$所有的概念左依赖路径.

定义 14(互补概念右依赖路径). 设 $R_r(x)$、$R_r(\neg x)$为 $M_T(C)$的两条概念右依赖路径,则 $R_r(x,\neg x)=\{R_r(x),R_r(\neg x)\}$为 $M_T(C)$关于$(x,\neg x)$的一条互补概念右依赖路径.

定义 15(互补概念左依赖路径). 设 $R_l(x)$、$R_l(\neg x)$为 $M_T(C)$的两条概念左依赖路径,$R_r(y)$为 $M_T(C)$的一条概念右依赖路径,则 $R_l(x,\neg x)=\{R_l(x),R_l(\neg x),R_r(y)\}$为 $M_T(C)$关于$(x,\neg x)$的一条互补概念左依赖路径.

$M_T(C)$的概念依赖路径可能有多条,定义 16 能够从这些概念依赖路径中筛选出与不可满足有关的概念依赖路径.

定义 16(不可满足依赖路径). 设 $R(x)$为 $M_T(C)$的一条概念依赖路径(概念依赖路径 $R(x)$指的是概念左依赖路径 $R_l(x)$或概念右依赖路径 $R_r(x)$),其中,x 表示否定概念$\neg\theta$或互补概念$(\theta,\neg\theta)$,若将$\neg\theta$替换成θ后 C 变得可满足,则 $R(x)$为 $M_T(C)$的一条不可满足依赖路径.

令 $R(C)=\{R(x)_1,\ldots,R(x)_n\}$为 $M_T(C)$所有的不可满足依赖路径.

不可满足依赖路径的提出是为了筛选出导致 C 不可满足的否定概念$\neg\theta$或互补概念$(\theta,\neg\theta)$,从而将否定概念或互补概念通过不可满足依赖路径与不可满足概念联系起来.

获得了不可满足依赖路径之后,则很容易从中提取出不可满足概念路径.

定义 17(不可满足概念路径). 设 U 为 $M_T(C)$的不可满足概念集,对于 $M_T(C)$中的某一条不可满足依赖路径 $R(x)$,设ε为 $R(x)$中的概念,若$\varepsilon\notin U$,则从 $R(x)$中删除ε后得到的 $R(x)'$为 $M_T(C)$的不可满足概念路径.

不可满足概念路径的求解由算法 5 实现.

算法 5. UnConceptPath($M_T(C)$).

输入:$M_T(C)$ //C 关于 T 的某一个 MUPS;

输出:$R(C)'$ //$M_T(C)$的不可满足概念路径.

1. 求出 $M_T(C)$中的不可满足概念集合 U
2. 获取 $M_T(C)$的不可满足依赖路径 $R(C)$
3. $R(C)'=\varnothing$
4. **for each** $R(x)$ **in** $R(C)$
5. **for each** ε **in** $R(x)$ //对于 $R(x)$路径中的每个概念ε

6. **if** $\varepsilon \notin U$ **then** //如果该概念ε不属于不可满足概念集合
7. $R(x)'=R(x).remove(\varepsilon)$ //则将该概念从 $R(x)$路径中删除而得到 $R(x)'$
8. $R(x)'$.add($R(x)'$) //将 $R(x)'$添加进 $R(C)'$里
9. **return** $R(C)'$ //输出 $R(C)'$

Kalyanpur[12]定义了根与派生两种不可满足概念类型.

定义 18(根与派生不可满足概念). 设 C 是 T 中的不可满足概念,若 C 的不可满足性是由其自身定义造成的,则 C 是根不可满足概念,若 C 的不可满足性是由其他不可满足概念引起的,则 C 是派生不可满足概念.

推论 1. 设 $R(x)'$为 $M_T(C)$的一条不可满足概念路径.若 C 是根不可满足概念,则 $R(x)'$的长度$|R(x)'|=1$.

证明:使用反证法证明.假设$|R(x)'|>1$,考虑两种情形.

(1) 若不可满足概念来自概念右依赖路径,根据定义 12,设 $R(x)=\langle x,\varepsilon_1,...,\varepsilon_m,C\rangle$,如果$|R(x)'|>1$,由定义 17 可知,必有$\varepsilon_k \in R(x)$满足 $R(x)'=\langle \varepsilon_k,\ldots,C\rangle$,此时,$C$ 的不可满足性是由ε_k 造成的,根据定义 18,则 C 是派生不可满足概念.与条件矛盾.

(2) 若不可满足概念来自概念左依赖路径,根据定义 13,设 $R_l(x)=\langle x,\varepsilon_1,...,\varepsilon_m, \varepsilon_{1^*},...,\varepsilon_{m^*},C\rangle$,如果$|R(x)'|>1$,必有 $\varepsilon_{k^*} \in R_l(x)$满足 $R(x)'=\langle \varepsilon_{k^*},\ldots,C\rangle$,此时,$C$ 的不可满足性是由 ε_{k^*} 造成的,根据定义 15,则 C 是派生不可满足概念.与条件矛盾.

无论哪种情况,若$|R(x)'|>1$,则 C 都是派生不可满足概念.与条件矛盾. □

如果某个不可满足概念的不可满足依赖路径长度为 1,则它出现在路径的最前面,这说明它不依赖于任何其他不可满足概念.因此,在本体修复过程中,不可满足依赖路径长度为 1 的根不可满足概念是本体修复的目标,因为它是造成本体不协调的根本原因.完成了对根不可满足概念的修复,就完成了对不协调本体的修复.

定义 19(根不可满足路径). 设 $R(x)'$为 $M_T(C)$的一条不可满足概念路径且 $R(x)'$的长度$|R(x)'|=1$,则 $R(x)'$为 $M_T(C)$的一条根不可满足路径.

定义 20(根依赖路径). 设 $R(x)'$为 $M_T(C)$的一条根不可满足路径,则 $R(x)$为 $M_T(C)$关于 $R(x)'$的根依赖路径.

Kalyanpur 定义的根与派生两种不可满足概念类型存在的一个问题是:某些不可满足概念,同时具备根与派生两种属性,例如 $T=\{\alpha_1:B_1\subseteq\neg A_1,\alpha_2:B_2\subseteq A_1,\alpha_3:B\subseteq B_1\cap B_2,\alpha_4:C_1\subseteq B\cap\neg A_2\cap A_2\}$.

易知 B 是根不可满足概念,然而,完成对 B 的修复后,T 仍然是不协调的,因为 C_1 仍然不可满足.这种现象产生的原因是:C_1 的不可满足性一方面是由 $C_1\subseteq\neg A_2\cap A_2$ 导致的,因而具备根不可满足这一属性.另一方面,C_1 的不可满足性又是由 $C_1\subseteq B$ 导致的,此时 C_1 又具备派生不可满足这一属性.完成对 B 的修复只解决了 C_1 的派生不可满足这一属性而无法解决 C_1 的根不可满足这一属性.所以,基于根不可满足概念的修复策略在某些情况下是不适用的.而采用根依赖路径的方法则可以解决这一问题.

3.2 基于根依赖路径的公理分割

定义 21(不可满足依赖标签). 设 $S(x)=\langle(\xi^l_{\alpha_0},\alpha_0,\xi^r_{\alpha_0}),\ldots,(\xi^l_{\alpha_n},\alpha_n,\xi^r_{\alpha_n})\rangle$为 $M_T(C)$的一条冲突路径.令 $R(x)=\langle x,\varepsilon_1,...,\varepsilon_n,C\rangle$为 $S(x)$的根依赖路径.对于与ε_k 相关的任意公理α_k,其不可满足依赖标签为 $L(\alpha_i)=(\varepsilon_k,\alpha_k,\varepsilon_{k+1})$.令 $L(C)=\{L(\alpha_0),\ldots,L(\alpha_n)\}$为 $M_T(C)$的所有不可满足依赖标签.

接下来,根据 $M_T(C)$中公理的不可满足依赖标签,就可以对这些公理进行分割.算法 6 实现了基于根依赖路径进行公理分割获取修复集这一目标.

算法 6. *GetRepairSet*.

输入:MUPS(T) //调试得到的 MUPS 集合;

输出:$S_{\min}$ //修复集.

1. **for each** $M_T(C)$ **in** MUPS(T,C)
2. 求得 $M_T(C)$的根依赖路径 $R(x)$
3. **for each** $\alpha_k=Y\subseteq X$ **in** $M_T(C)$

4. 从 $R(x)$中获得α_k的不可满足依赖标签 $L(\alpha_k)=(\varepsilon_k,\alpha_k,\varepsilon_{k+1})$
5. **if** Y 形如 $Y_1\cup\ldots\cup Y_m$ **then**
6. **for each** Y_j **in** Y
7. P_Y=***Partition*1**(Y_j,$L(\alpha_k)$)
8. **if** X 形如 $X_1\cap\ldots\cap X_n$ **then**
9. **for each** X_i **in** X
10. P_X =***Partition*1**(X_i,$L(\alpha_k)$)
11. **else** $P_Y=\{Y\}$, $P_X=\{X\}$
12. **for each** y **in** P_Y and **each** x **in** P_X
13. $S.add(y\subseteq x)$
14. **end for**
15. **end for**
16. $S_{\min}$=**Minimality**(S)
17. **return** $S_{\min}$ //输出修复集

在算法 6 中,当待分割公理左侧是析取式时,则对该析取式的每一部分,调用函数 *Partition*1(Y_j,$L(\alpha_k)$)进行分割(第 5 行~第 7 行).当待分割公理右侧是合取式时,则对该合取式的每一部分,调用函数 *Partition*1(X_i,$L(\alpha_k)$)进行分割(第 8 行~第 10 行).其他情况则保留原表达式不变(第 11 行).分割完成后对于获取的分割集 P_Y 与 P_X 中的每个元素 y 与 x,重新组合成新的公理(第 12 行~第 13 行).最后,再对获取的分割集进行一次极小化操作(第 16 行).

需特别说明的是,对于公理右边的析取形式与公理左边的合取形式没有进行分割操作.这是因为分割的目的是排除与不可满足性无关的那些部分.从不可满足性的根源考虑,公理右边的析取式的各部分必须同时与不可满足性有关,以及公理左边的合取式的各部分必须同时与不可满足性有关,这样的公理才能出现于 MUPS 中.

函数.*Partition*1(S).

输入:S;

输出:$P_{X/Y}$.

1. **if** $\varepsilon_k\subseteq sig(Y_j)$且$\varepsilon_{k+1}\subseteq sig(X_i)$,或$\varepsilon_{k+1}\subseteq sig(Y_j)$且$\varepsilon_k\subseteq sig(X_i)$ **then**
2. **if** X_i 或 Y_j 形如 C,$\neg$或$\exists R.C$ **then**
3. $P_X=\{X_i\}$或 $P_Y=\{Y_j\}$
4. **if** X_i 或 Y_j 形如$\forall R.C$ **then**
5. P_X 或 P_Y=***Partition*2**($\forall R.C$,$L(\alpha_i)$) //求出$\forall R.C$ 的分割集
6. **return** P_X 或 P_Y

在分割函数 *Partition*1 中,首先判断待分割的 Y_j 或 X_i 是否存在 $L(\alpha_k)$中的依赖概念,若存在,则进行分割,在分割过程中,保留形如 C,$\neg C$ 或$\exists R.C$ 的 X_i 或 Y_j,而对形如$\forall R.C$ 的 X_i 与 Y_j,则进一步调用函数 *Partition*2 进行分割.在分割函数 *Partition*2 中,首先存储角色 R 于 *prop* 中,然后针对合取式 C 递归调用 *Partition*2,继续进行分割.

函数.*Partition*2($\forall R.C$,$L(\alpha_i)$).

输入:$\forall R.C$,$L(\alpha_i)$;

输出:P.

1. *prop*=$\{R\}$
2. **if** C 形如 C_k,$\neg C_k$ 或$\exists R.C_k$ 且$\varepsilon_k\subseteq sig(C_k)$或$\varepsilon_{k+1}\subseteq sig(C_k)$, **then**
3. $P=\{\forall R.C\}$
4. **if** C 形如 $C_1\cap\ldots\cap C_n$, **then**
5. P=*Partition*2($\forall R.C_k$,$L(\alpha_i)$)
6. **return** P

极小化函数 Minimality(S)是为了排除那些存在多个相同否定概念的公理.例如公理 $B\subseteq A\cap\neg A\cap\exists R.\neg A$,对其分割后得到 $S=\{B\subseteq A,B\subseteq\neg A,B\subseteq\exists R.\neg A\}$,$B\subseteq\exists R.\neg A$ 之所以能存在于 S 中,是因为只有当发现某个公理存在否定概念时,才开始构造与该否定概念有关的冲突路径,但是无法区分该否定概念究竟是 $B\subseteq\neg A$ 还是 $B\subseteq\exists R.\neg A$ 中的否定概念.这一现象不会影响本体调试结果,但会造成修复集存在冗余的公理.因此需要进行一次极小化操作.具体方法是对每一个存在否定概念的公理(第 1 行~第 3 行),将它从修复集中删除(第 4 行),然后检测该修复集中的不可满足概念集是否有变化(第 5 行),若无变化,则该公理是冗余公理,删除即可(第 6 行).

函数.Minimality(S).

输入:S;

输出:$S_{\min}$.

1. **for each** $\neg\theta$ **in** $sig(S)$
2. **for each** α=$y\subseteq x$ **in** S
3. **if** $\neg\theta\in sig(y)$或 $sig(x)$, **then**
4. $S'=S.remove(\alpha)$
5. **if** $U_S=U_{S'}$ **then** //若 S 与 S'中的不可概念集 U_S 与 $U_{S'}$相同
6. $S_{\min}=S.remove(\alpha)$ //删除冗余公理α
7. **return** $S_{\min}$

考虑本节开始的引例 $T=\{\alpha_1:C\subseteq A,\alpha_2:A\subseteq E_1\cap D_1,\alpha_3:D_1\cup E_2\subseteq D_2,\alpha_4:D_2\subseteq E_3\cap\neg A,\alpha_5:E_1\subseteq E_3.\}$通过上述方法得到的修复集 $S=\{A\subseteq D_1,D_1\subseteq D_2,D_2\subseteq\neg A\}$,极小化 S 后仍是$\{A\subseteq D_1,D_1\subseteq D_2,D_2\subseteq\neg A\}$.

设 $M_T(C)$是一个有 m 个公理的 MUPS,求 $M_T(C)$的冲突路径的复杂度是 km,其中,k 是冲突路径的条数.这是因为在使用本文的方法构造 $M_T(C)$的冲突路径时,有效地避免了循环操作,从而大大降低了求解冲突路径的复杂度.接下来,从冲突路径上出现的所有概念中筛选出不可满足概念,从而将冲突路径变为不可满足概念依赖路径,对于出现 n 个概念的冲突路径,判断每个概念是否属于不可满足概念集合(不可满足概念集合在调试阶段已经得到),一共需要 n 次判断.所以,算法 5 的时间复杂度为 $k(m+n)$.在获取修复集的过程中,对于有 m 个公理的 $M_T(C)$,需要从不可满足概念依赖路径中获取每个公理的不可满足概念的依赖标签,该过程的时间复杂度为 m.这是因为,在构造依赖路径时,路径上的每一个节点就是一个依赖标签,一旦匹配成功,就直接可以读取该标签.然后根据依赖标签对公理 $Y\subseteq X$ 进行分割,在此过程中,需要进行公理类型的判断:如果 Y 形如 $Y_1\cup\ldots\cup Y_p$,则需要 p 次分割操作,如果 X 形如 $X_1\cap\ldots\cap X_q$,则需要 q 次分割操作,而每次操作都需要调用分割函数 1 和分割函数 2,每个分割函数都需要进行 2 次判断.最后是极小化过程,设修复集的公理个数为 s,修复集中否定概念的个数为 t,此时需要进行 st 次可满足性检测.所以,算法 6 的时间复杂度为 $2mpq+st$.

4 实验与分析

本文实验是在 PC 机 Windows 10 操作系统(Intel(R) Core(TM) i7-6700 CPU @ 3.40GHZ,16.0G 内存)下运行的.对本体的解析和操作采用了 OWLAPI(http://owlapi.sourceforge.net/),使用推理机 Pellet(https://github.com/stardog-union/pellet)对概念进行可满足性检测,最大 JVM 堆设置为 3.0G.超时界限设定为 60 分钟.实验数据、CP 算法源码与结果由 http://www.zhyweb.cn/cpdebrep/index.php 下载.

本文实验的数据来源自本体库(http://www.cs.ox.ac.uk/isg/ontologies/lib/),鉴于所提出的调试与修复策略是基于不协调的 ALCOI-TBox 本体的,本文使用文献[26]所采取的处理方式,将不符合 ALCOI-TBox 的公理删去,并以随机的方式从本体中选择两个概念构成不相交公理添加进原本体中,从而生成不协调本体.例如,对于协调的本体 $T=\{A\subseteq B,B\subseteq C,B\subseteq D\}$,通过添加公理 $C\subseteq\neg D$ 从而导致 A 和 B 不可满足.表 1 列出了实验数据的主要属性.$|T|$是本体规模,$|U|$是不可满足概念个数,$MN_{\max}$ 与 ML_{avg} 分别是 MUPS 最大基数与最大长度,MN_{avg} 与 ML_{avg} 分别是 MUPS 的平均基数与平均长度.

Table 1 Properties of incoherent TBoxes used in the experiments

表 1 实验中不协调 TBox 的主要属性

ID	TBox	$\|T\|$	$\|U\|$	MN_{max}	ML_{max}	MN_{avg}	ML_{avg}
T_1	OBI10	3 968	103	8	19	2.330	5.167
T_2	PATO	3 489	158	4	8	1.177	3.409
T_3	NIF	3 460	133	10	12	3.519	5.447
T_4	OBO09	34 367	59	4	10	1.085	2.953
T_5	OBO12	51 782	24	7	8	2.375	3.018
T_6	Cellular	6 190	64	10	7	2.766	4.305
T_7	OBI09	2 017	22	3	6	1.182	3.423
T_8	SAO	1 112	241	3	9	1.075	6.139
T_9	SWO	2 624	166	4	4	1.560	2.807
T_{10}	OBOGO	37 615	40	3	7	1.750	3.700
T_{11}	Anatomy	36 157	35	4	15	1.257	5.045
T_{12}	MAO	1 619	20	1	5	1.000	2.200
T_{13}	FLU	1 326	65	4	33	1.600	11.692
T_{14}	Bockman	3 043	25	2	4	1.160	2.138
T_{15}	Cavender	1 623	23	2	7	1.130	3.731

4.1 基于冲突路径的本体调试实验

实验针对黑盒法的 4 种优化算法和 3 个调试工具展开.黑盒优化算法包括扩张阶段的选择函数方法(记作 *S*)与模块化方法(记作 *M*)以及收缩阶段的滑动窗口方法(记作 *W*)与分治方法(记作 *D*).对比实验中将两个阶段的优化方法组合起来构成 4 种算法(例如选择函数与滑动窗口组合算法记作 *SW*).针对每一种组合算法,采用本文的基于冲突路径(记作 *CP*)的策略进行优化,优化后的算法记作 CPSW.3 个调试工具包括 Radon[19]、Swoop[24]和 Protégé(http://www.cs.ox.ac.uk/isg/ontologies/lib/).针对每一个调试工具,采用 CP 策略获取冲突集,将冲突集作为调试工具的输入.图 1 显示的是 14 种算法求解所有不可满足概念的所有 MUPS 的时间的对比情况.

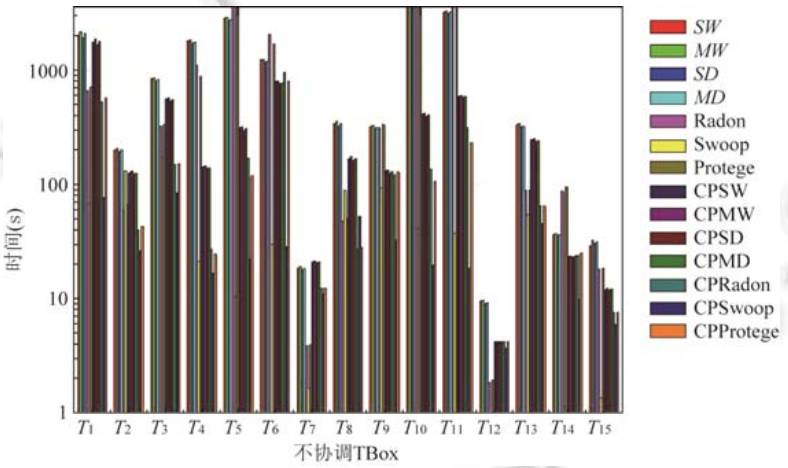

Fig.1 Compared run time of fourteen algorithms for computing MUPS

图 1 14 种算法求解 MUPS 的时间对比

图 1 中,除了 T_7 之外,基于 CP 优化策略的组合算法均优于未使用 CP 优化策略的组合算法.除了 T_7 和 T_{12} 之外,基于 CP 优化策略的调试工具均优于未使用 CP 优化策略的调试工具.这是因为,相对于黑盒法求解 MUPS 频繁地调用推理机进行可满足性检测,基于冲突路径获取冲突集的方法仅仅在判定获取的否定路径和互补路径的公理集是否协调时才调用一次推理机,因而所需时间较少,这是冲突路径优化策略性能较好的主要原因.表 2 对 CP 优化策略获取冲突集的属性进行了统计,$|CP|$是互补概念对的个数,$|Pn|$与$|Pp|$分别是否定路径与肯定路径中公理的数目,$|Q|$是冲突集里公理的数目,$|Q/T|$是冲突集占原本体的百分比,*Time* 是获取冲突集的时间(单位:s).

Table 2 Properties of obtaining clash set based on CP approach

表 2 CP 优化策略获取冲突集的属性统计

TBox	$\|T\|$	$\|CP\|$	$\|U\|$	$\|Pn\|$	$\|Pp\|$	$\|Q\|$	$\|Q/T\|$ (%)	*Time*(s)
T_1	3 968	274	103	3 941	3 666	3 670	92.49	22.001
T_2	3 489	269	158	3 465	3 196	3 176	91.03	6.937
T_3	3 460	261	133	3 413	3 152	2 211	63.90	5.311
T_4	34 367	78	59	10 436	6 158	5 290	15.39	4.520
T_5	51 782	99	24	19 003	25 387	12 466	24.07	18.674
T_6	6 190	120	64	5 726	4 658	4 439	71.71	10.917
T_7	2 017	160	22	1 816	1 656	1 658	82.20	9.490
T_8	1 112	206	241	1 024	832	816	73.38	3.338
T_9	2 624	159	166	1 508	1 224	1 555	59.26	1.420
T_{10}	37 615	210	40	22 902	15 824	6 418	17.06	13.63
T_{11}	36 157	272	35	19 775	15 426	13 030	36.04	12.626
T_{12}	1 619	247	20	1 567	1 317	758	46.82	3.267
T_{13}	1 326	275	65	1 301	1 026	1 047	78.96	5.717
T_{14}	3 043	94	25	3 014	2 911	1 102	36.21	8.804
T_{15}	1 623	264	23	1 595	1 332	509	31.37	4.89

综合分析图 1 与表 2 的实验结果,可以得出如下结论.

(1) 在本体规模较大、不可满足概念较多以及 MUPS 结构较复杂的情况下(体现为表 1 的后 4 个属性),黑盒法求解 MUPS 需要耗费大量的时间.例如使用 *SW* 方法调试 T_{12} 本体的时间为 9.531s,而调试 T_5 本体的时间则高达 2 862.556s,而 T_{10} 本体的调试时间则超过 60 分钟.

(2) 当使用 CP 优化策略时,如果 MUPS 求解过程是基于一个较小规模的冲突集之上,则效率较高.例如,对于规模较大的本体(例如 T_4,T_5 与 T_{10}),所得到的冲突集相对于原本体较小(分别是 15.39%、24.07%与 17.06%),则 CP 效率较高,例如 T_4 本体,*SW*=1818.296s,而 *CPSW*=142.778s.如果冲突集接近于原本体,则 CP 方法效率就会受到一定的影响,例如,T_1 本体的求解时间分别为 *SW*=2034.249s 和 *CPSW*=1755.991s,这是因为冲突集占原本体的比例高达 92.49%.

(3) 当获取冲突集所耗费的时间较多时,则 CP 优化策略的性能就无法有效发挥.例如,对于 T_{77},*SW* 的求解时间是 18.407s,而 *CPSW* 的求解时间是 20.961s,这是由于 CP 方法获取冲突集消耗了 9.490s 的时间,而基于冲突集求解 MUPS 的时间是 11.471s.相对于求解 MUPS 所需要的时间,获取冲突集的时间较多,因而影响了 CP 优化算法的性能.

(4) 在影响黑盒法求解效率的众多因素中,本体规模的影响相对较大,而 CP 优化策略所获取的冲突集仅由冲突路径所决定,因而基本上不受本体规模的影响.例如 T_4,本体规模为 34 367,而冲突集规模仅为 5 290,而获取冲突集的时间仅为 4.520s,因而基于 CP 优化的黑盒法在 5 290 规模量级的冲突集上求解 MUPS 的效率较高.

本文提出的基于冲突路径的本体调试方法可以说是一个基于模块的本体调试方法,如下实验比较了基于语法局部(syntactic locality module)的模块化方法[14]和本文方法抽取的模块中公理的数量.

设 T 有 n 个不可满足概念,对于不可满足概念 u_i,在采用基于模块的黑盒调试方法求解 MUPS(T,u_i)时,在扩张阶段,抽取与 u_i 有关的基于语法局部的模块,记为 $M(u_i)$,再在收缩阶段将 $M(u_i)$削减成极小集合.

设 $M(u_i)$的公理数量为 $S_{M(u_i)}$, 那么 n 个不可满足概念需要抽取的模块大小则为 $M=\sum_1^n S_{M(u_i)}$.

表 3 中第 2 行统计的是基于原本体所抽取的所有不可满足概念的模块中公理的总数量,第 3 行统计的是:基于各个本体的冲突集所抽取的所有不可满足概念的模块中公理的总数量.

Table 3 Modules extracted by Modularity and CPModularity methods for all unsatisfiable concepts

表 3 Modularity 与 CPModularity 方法抽取所有不可满足概念的模块统计结果

方法	T_1	T_2	T_3	T_4	T_5	T_6	T_7	T_8	T_9	T_{10}	T_{11}	T_{12}	T_{13}	T_{14}	T_{15}
Modularity	13 625	11 878	11 074	2 298	1 198	4 128	1 857	16 798	3 623	7 038	5 896	116	4 604	230	194
CPModularity	11 992	9 035	8 557	518	146	2 567	1 744	16 432	1 194	305	341	45	4 390	113	182

表 3 显示,基于冲突集所抽取的模块中公理的数量确实少于基于原本体所抽取的模块中公理的数量.

需要说明的是:本文获取冲突集这一过程不是取代基于“扩-缩”策略的黑盒法的扩张阶段,而是尽可能地为黑盒调试算法和调试工具提供一个较小规模的输入.这是因为:黑盒法对本体规模非常敏感,一个较小规模的输入本体对黑盒法的效率提升是有很大帮助的.

4.2 基于冲突路径的本体修复实验

虽然基于DL-Lite描述逻辑的本体修复方法较多,但这些方法均利用DL-Lite特殊的语法特征构造DL-Lite逻辑闭包图完成本体修复,这类方法无法处理任何非 DL-Lite 的本体.因而本文基于 ALCOI 描述逻辑的本体修复实验无法与 DL-Lite 本体的修复方法进行对比.而对于非 DL-Lite 本体的修复方法,最具代表性的是基于根不可满足概念实现细粒度本体修复的方法(记作 RSRepair[12]),与此对应的一套算法包括:基于依赖跟踪的方法获取根不可满足概念,基于公理分割的方法对 MUPS 中的所有公理进行分割获取 MUPS 的分割集,以及基于分割集再次求解MUPS的细粒度修复.这一套算法与本文提出的基于冲突路径的修复方法(记作CPRepair)的出发点和目标都是相同的,差别在于如下 3 点:一是获取根不可满足概念的方式,二是公理分割的对象,三是公理分割的方式,因而具有较强的可比性.表 4 列出了 RSRepair 与 CPRepair 实验对比结果.

Table 4 Compared results of RSRepair and CSRepair for ontology repairing (ms)

表 4 RSRepair 与 CPRepair 本体修复对比实验结果 (毫秒)

方法	T_1	T_2	T_3	T_4	T_5	T_6	T_7	T_8	T_9	T_{10}	T_{11}	T_{12}	T_{13}	T_{14}	T_{15}
RSRepair	1 844	91	129	177	51	27	53	55	32	68	722	175	3579	122	870
CPRepair	174	83	**693**	76	45	**105**	47	**135**	**151**	53	71	68	263	57	56

表 4 显示,除了 T_3,T_6,T_8,T_9 这 4 个本体外,CPRepair 均优于 RSRepair.为了深入 RSRepair 方法内部探查其各个环节对修复效率的影响,表 5 统计了基于依赖跟踪方法获取根不可满足概念的时间($Time_{root}$),基于公理分割方法获取 MUPS 分割集的时间($Time_{splitting}$)以及基于分割集再次求解 MUPS 的时间($Time_{MUPS}$).由表 5 可知,前两个环节所花费的时间较少,时间基本上都耗费在第 3 个环节,这是由于,基于分割集再次求解 MUPS 的过程需要多次调用推理机进行可满足性检测.本文CPRepair方法的不同之处在于:一是从MUPS中获取冲突路径,再从冲突路径中筛选出根依赖路径从而确定根不可满足概念;二是公理分割仅仅针对具有依赖概念的那部分公理.整个过程只有在筛选根依赖路径以及对修复集进行极小化操作时才调用推理机,因而效率较高.特别是 T_1,T_{11},T_{13},T_{15} 这 4 个本体,修复效率比 RSRepair 高达 10 多倍.对于 T_6 与 T_9,一方面是分割集较小,另一方面是分割集基本接近最终的MUPS求解结果,因而RSRepair所耗费的时间很少.需要补充说明的是,虽然某些不可满足概念存在着根与派生两种属性(见第 3.1 节),但是 RSRepair 在对 MUPS 中的所有公理进行分割之后,就可以将这两种属性分割开来,从而可以将该不可满足概念的两种属性同时显示出来.

Table 5 Time of three algorithms for RSRepair (ms)

表 5 RSRepair 方法的 3 种算法运行时间 (毫秒)

时间	T_1	T_2	T_3	T_4	T_5	T_6	T_7	T_8	T_9	T_{10}	T_{11}	T_{12}	T_{13}	T_{14}	T_{15}
$Time_{root}$	6	5	7	5	3	3	9	5	8	3	3	6	5	13	6
$Time_{splitting}$	9	5	4	8	5	4	5	4	4	5	7	6	6	6	5
$Time_{MUPS}$	1 829	81	118	164	43	20	39	46	20	60	712	163	3 568	103	859
All time	1 844	91	129	177	51	27	53	55	32	68	722	175	3 579	122	870

获取了根不可满足概念的修复集之后,就可以提交给领域专家进行修复,只要完成了根不可满足概念的修复,就完成了整个不协调本体的修复.

5 结束语

本文对不协调本体调试与修复的黑盒法进行了研究,基于“扩张-收缩”策略的黑盒法及其优化方法独立于特定的推理机而应用十分广泛.然而,它们在求解 MUPS 过程中很容易选出与不协调无关的冗余公理而影响本体调试的效率.针对这一问题,提出了一种基于冲突路径的调试与修复策略,该策略通过构造与 4 种基本冲突模

式所对应的冲突路径,能够获取与不协调密切相关的冲突集.在此过程中,由于推理机的调用次数很少,所以获取冲突集的时间耗费较少.接下来,将 MUPS 的求解过程限定在该冲突集上能够较大幅度地减少参与黑盒法求解的术语集规模,这是效率得以提高的主要原因.进一步地,借助于冲突路径的良好结构,获得不可满足概念依赖路径并基于该路径制定出不可满足概念的修复策略,该策略能够获取精确的修复集,并且能够避免本体信息内容的损失.需要特别说明的是,本体的描述逻辑语言不同,冲突路径的构造方式也随之会变化,下一步的工作将针对更高表达能力的描述逻辑本体构造其对应的冲突路径进行本体调试与修复.

References:

[1] Baader F, Calvanese D, McGuinness D, Nardiand D, Patel-Schncider PF. Basic Description Logics. 2nd ed., Cambridge: Cambridge University Press, 2007. 47–100.

[2] Li P, Jiang YC, Wang J. Modular ontology reuse based on conservative extension theory. Ruan Jian Xue Bao/ Journal of Software, 2016,27(11):2777–2795 (in Chinese with English abstract). http://www.jos.org.cn/1000-9825/4920.htm [doi: 10.13328/j.cnki.jos.004920]

[3] Cui XJ, Ouyang DT, Ye YX. Integrity constraint validation for ontology using modularization. Journal of Information & Computational Science, 2014,11(9):2705–2713. [doi: 10.12733/jics20103476]

[4] Ouyang DT, Qu JF, Ye YX. Extending training set in distant supervision by ontology for relation extraction. Ruan Jian Xue Bao/Journal of Software, 2014,25(9):2088–2101 (in Chinese with English abstract). http://www.jos.org.cn/1000-9825/4638.htm [doi: 10.13328/j.cnki.jos.004638]

[5] Schlobach S, Huang Z, Cornet R, Van Harmelen F. Debugging incoherent terminologies. Journal of Automated Reasoning, 2007,39(3):317–349. [doi: 10.1007/s10817-007-9076-z]

[6] Meyer T, Lee K, Booth R, Pan J.Z. Finding maximally satisfiable terminologies for the description logic ALC. In: Proc. of the 21st AAAI. American Association for Artificial Intelligence, 2006. 269–274.

[7] Ouyang DT, Su J, Ye YX, Cui XJ. The ontology debugging method based on concept R-MUPS. Ruan Jian Xue Bao/Journal of Software, 2015,26(9):2231–2249 (in Chinese with English abstract). http://www.jos.org.cn/1000-9825/4735.htm [doi: 10.13328/j.cnki.jos.004735]

[8] Schlobach S, Cornet R. Non-Standard reasoning services for the debugging of description logic terminologies. In: Proc. of the 18th Int'l Joint Conf. on Artificial Intelligence. 2003. 355–360.

[9] Baader F, Penaloza R. Automata-Based axiom pinpointing. In: Proc. of the 4th IJCAR. Heidelberg: Springer-Verlag, 2008. 226–241.

[10] Baader F, Penaloza R, Suntisrivaraporn B. Pinpointing in the description logic EL+. In: Proc. of the 20th DL. Tilburg: Sun SITE Central Europe CEUR-WS, 2007. 171–178.

[11] Baader F, Suntisrivaraporn B. Debugging snomed CT using axiom pinpointing in the description logic EL+. In: Proc. of the 3rd KR-MED. Tilburg:Sun SITE Central Europe CEUR-WS, 2008. 1–7.

[12] Kalyanpur A. Debugging and repair of OWL ontologies [Ph.D. Thesis]. Maryland: University of Maryland, 2006.

[13] Kalyanpur A, Parsia B, Horridge M, Sirin E. Finding all justifications of OWL DL entailments. In: Proc. of the 6th ISWC. Heidelberg: Springer-Verlag, 2007. 267–280. [doi: 10.1007/978-3-540-76298-0_20]

[14] Cuencagrau B, Ian H, Yevgeny K, Ulrike S. Just the right amount: Extracting modules from ontologies. In: Proc. of the 16th Int'l World Wide Web Conf. New York: Association for Computing Machinery, 2007. 717–726. [doi: 10.1145/1242572.1242669]

[15] Ji Q, Qi GL, Haase P. A relevance-directed algorithm for finding justifications of DL entailments. In: Proc. of the 4th ASWC. Heidelberg: Springer-Verlag, 2009. 306–320. [doi: 10.1007/978-3-642-10871-6_21]

[16] Suntisrivaraporn B, Ji Q, Haase P. A modularization-based approach to finding all justifications for OWL DL entailments. In: Proc. of the 3rd ASWC. Heidelberg: Springer-Verlag, 2008. 1–15. [doi: 10.1007/978-3-540-89704-0_1]

[17] Del Vescovo C, Parsia B, Sattler U, Schneider T. The modular structure of an ontology: Atomic decomposition. In: Proc. of the 22nd Int'l Joint Conf. on Artificial Intelligence. 2011. 2232–2237. [doi: 10.5591/978-1-57735-516-8/IJCAI11-372]

[18] Grau BC, Halaschek-Wiener C. Incremental classification of description logics ontologies. Journal of Automated Reasoning, 2010,44:337–369. [doi: 10.1007/s10817-009-9159-0]

[19] Ji Q, Gao ZQ, Huang ZS, Zhu M. An efficient approach to debugging ontologies based on patterns. In: Proc. of the Joint Int'l Semantic Technology Conf. Heidelberg: Springer-Verlag, 2011. 425–433. [doi: 10.1007/978-3-642-29923-0_33]

[20] Horridge M. Justification based explanation in ontologies [Ph.D. Thesis]. Manchester: University of Manchester, 2011.

[21] Shchekotykhin K, Friedrich G, Dietmar J. On computing minimal conflicts for ontology debugging. In: Proc. of the ECAI 2008 Workshop on Model-Based Systems. 2008. 7–11.

[22] Ji Q, Gao ZQ, Huang ZS, Zhu M. Measuring effectiveness of ontology debugging systems. Knowledge-Based Systems, 2014,71: 169–186. [doi: 10.1016/j.knosys.2014.07.023]

[23] Lam SC, Pan JZ, Sleeman D, Vasconcelos W. A fine-grained approach to resolving unsatisfiable ontologies. Journal on Data Semantics X, 2800,62–95. [doi: 10.1007/978-3-540-77688-8_3]

[24] Du JF. Qi GL, Fu XF. A practical fine-grained approach to resolving incoherent owl 2 DL terminologies. In: Proc. of the 23rd ACM-CIKM. 2014. 919–928. [doi: 10.1145/2661829.2662046]

[25] Fu XF, Qi GL, Zhang Y. A graph-based approach for calculating minimal unsatisfiability-preserving subsets of ontology in DL-Lite. Chinese Journal of Electronics, 2016,44(9):2040–2045 (in Chinese with English abstract). [doi:10.3969/j.issn.0372-2112.2016.09.002]

[26] Fu XF, Qi GL, Zhang Y, Zhou, Z. Graph-Based approaches to debugging and revision of terminologies in DL-Lite. Knowledge-Based Systems, 2016,100:1–12. [doi: 10.1016/j.knosys.2016.01.039]

[27] Qi GL, Wang Z, Wang KW, *et al*. Approximating model-based ABox revision in DL-Lite: Theory and practice. In: Proc. of the 29th AAAI. AI Access Foundation. 2015. 254–260.

[28] Gao SB, Qi GL, Wang HF. A new operator for ABox revision in DL-Lite. In: Proc. of the 26th AAAI Conf. on Artificial Intelligence. El Segundo: AI Access Foundation, 2012. 2423–2424.

[29] Zhuang ZQ, Wang Z, Wang KW, Qi GL. Contraction and revision over DL-Lite TBoxes. In: Proc. of the 28th AAAI Conf. on Artificial Intelligence. El Segundo: AI Access Foundation, 2014. 1149–1155.

[30] Parsia B, Sirin E, Kalyanpur A. Debugging owl ontologies. In: Proc of the WWW-2005. 2005,1:633–640.

[31] Krötzsch M, Simancik F, Horrocks I. A description logic primer. Perspectives on Ontology Learning, 2014,18:3–20.

附中文参考文献:

[2] 李璞,蒋运承,王驹.基于保守扩充理论的模块化本体重用.软件学报,2016,27(11):2777–2795. http://www.jos.org.cn/1000-9825/4920.htm [doi: 10.13328/j.cnki.jos.004920]

[4] 欧阳丹彤,瞿剑峰,叶育鑫.关系抽取中基于本体的远监督样本扩充.软件学报,2014,25(9):2088–2101. http://www.jos.org.cn/1000-9825/4638.htm [doi: 10.13328/j.cnki.jos.004638]

[7] 欧阳丹彤,苏静,叶育鑫,崔仙姬.基于概念 R-MUPS 的本体调试方法.软件学报,2015,26(9):2231–2249. http://www.jos.org.cn/1000-9825/4735.htm [doi: 10.13328/j.cnki.jos.004735]

[25] 付雪峰,漆桂林,张勇.一种基于图的 DL-Lite 本体最小不可满足保持子集的计算方法.电子学报,2016,44(9):2040–2045.

张瑜(1982－),男,河南信阳人,博士生,主要研究领域为语义 Web,自动推理.

叶育鑫(1981－),男,博士,副教授,CCF 专业会员,主要研究领域为语义 Web,自动推理.

欧阳丹彤(1968－),女,博士,教授,博士生导师,CCF 高级会员,主要研究领域为模型诊断,语义 Web,自动推理.