软件学报 ISSN 1000-9825, CODEN RUXUEW
Journal of Software, [doi: 10.13328/j.cnki.jos.006870]

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

强表达描述逻辑本体的后继式公理定位研究*

李静[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: 公理定位能够挖掘描述逻辑中可解释的缺陷,并为逻辑蕴含结果寻找隐藏的理由,因此在描述逻辑研究中引起了广泛的关注.平衡描述逻辑表达能力和推理机求解效率问题一直是公理定位研究的重点内容.本文基于一种后继式判定算法,从白盒和黑盒两个角度将其用于公理定位.白盒方法利用修正的后继式规则(即定位规则)来追踪推理的具体过程,并引入定位公式的概念建立子句的布尔公式标签与逻辑蕴含的所有极小公理集之间的对应关系.黑盒方法则直接调用基于未修正的后继式规则的推理机,并进一步利用碰集树方法求得逻辑蕴含的所有理由.最后,基于这样的两种面向强表达描述逻辑本体的公理定位算法设计推理工具,从理论和实验两方面验证其可行性,并与已有的推理工具比较求解性能.

关键词: 公理定位;后继式判定算法;定位规则;白盒与黑盒

中图法分类号: TP311

中文引用格式: 李静,欧阳丹彤,叶育鑫.强表达描述逻辑本体的后继式公理定位研究.软件学报. http://www.jos.org.cn/1000-9825/6870.htm

英文引用格式: LI J, Ouyang DT, Ye YX. Consequence-based Axiom Pinpointing for Expressive Description Logic Ontoligies. Ruan Jian Xue Bao/Journal of Software (in Chinese). http://www.jos.org.cn/1000-9825/6870.htm

Consequence-based Axiom Pinpointing for Expressive Description Logic Ontologies

LI Jing[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 120012, China)

Abstract: Axiom pinpointing has attracted extensive interest for Description Logics due to its effect of exploring explicable defects in the ontology and searching hidden justifications for the logic consequence. Balancing the expressive power of description logics and the solving efficiency of reasoners has always been the focus of axiom pinpointing research. In this paper, from both glass-box and black-box perspectives proposed a consequence-based method to computing justifications. The glass-box method uses modified reasoning rules to trace the specific process of inference, and introduces the concept of pinpointing formula to establish the correspondence between the label of Boolean formula and all the minimal axioms sets. The black-box method directly calls the inference engine based on the unmodified reasoning rules, and further uses the HST to compute all justifications for the inference. Two reasoning tools have been designed based on the two axiom pinpointing algorithms for expressive description logics ontologies. Its feasibility is verified theoretically and experimentally, and its solving efficiency is compared with that of existing axiom pinpointing tools.

Key words: axiom pinpointing; consequence-based method; pinpointing rules; glass-box and black-box

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包括术语集(TBox)和断言集(ABox)两部分[1].描述逻辑中所包含的构造算子不同,则其所表示的本体表达能力强弱

* 基金项目: 国家自然科学基金(42050103, 62076108)
收稿时间: 2022-09-06; 修改时间: 2022-10-13; 采用时间: 2022-12-14; jos 在线出版时间: 2022-12-30

也不同.随着本体的应用范围不断扩大,所需要构建的本体越来越复杂,因此,从对弱表达描述逻辑本体的研究发展到强表达描述逻辑本体的研究具有必然趋势.推理是本体领域的重要研究内容,可以从明确给出的知识中得出隐含的知识.公理定位(axiom pinpointing)是 Baader 等人[1]提出来的一种特殊的推理任务,它的主要目标是在本体中找到对某一给定逻辑蕴含负责的极小公理集(minimal axiom sets,简称 MinAs),常用来对本体进行缺陷查找和逻辑错误隐藏原因的研究.例如在生物医学本体 Snomed-CT[2]中,根据对本体的推理可以得出“断指即断臂”这样的关系,这就意味着如果病人接受手指截肢手术,那么他的手臂也必须被截肢.这样的推理结论是极其荒谬的,因此要求我们能够从本体中计算出给定公理的最小逻辑原因.在本体的构建、更新、合并等过程中难免发生错误,然而对本体的开发人员或用户来说,他们通常很难理解某个结论为什么成立,而要在不需要某个结论的情况决定如何修改本体就变得更加困难.因此对逻辑蕴含的公理定位研究具有重要的实际意义,引起了研究者们的广泛关注.

Schlobach 等人[3]提出了基于 Tableau 的白盒(glass-box)定位方法,修改了已有的判定算法,使得一次程序的运行过程就可以计算出逻辑蕴含的理由(justification).他们引入了极小不可满足保持子集(minimal unsatisfiability-preserving sub-TBoxes,简称 MUPS)的概念,但只适用于非循环的 ALC 不一致本体.为了能够对表达能力更强的本体计算理由,Parsia 等人[4]提出了适用于 SHIF (D) 本体的白盒方法,Kalyanpur 等人[5]进一步将其扩展至 SHOIN (D) 本体.欧阳丹彤等人[6]提出基于有序标签演算的概念 R-MUPS,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.以上方法都是基于 Tableau 的白盒定位算法,Schlobach 等人[7]还提出了一种黑盒(black-box)定位方法,通过重复调用一个未修改的推理机来计算理由.Ji 等人[8]采用基于相关度的方法计算蕴含的理由,张瑜等人[9]提出了一种基于冲突路径的调试与修复策略,降低调试目标规模以提高效率,他们[10]还提出了增量本体定义序列的定义,改进黑盒方法的扩张和收缩阶段提高求解效率.Gao 等人[11]探讨了 MUPS 和 MCPS 之间的对偶性,设计了定位方法并对其应用了并行策略同样提高了效率.

为了解决推理能力和描述逻辑表达能力之间的平衡问题,Baader 等人[12]对 EL 系列语言的推理问题进行研究,通过减弱本体的表达能力来提高推理效率.他们对 EL 系列本体的分类过程进行命题逻辑编码,通过求解极小不可满足子式(MUS)来得到逻辑蕴含理由.考虑到轻量级描述逻辑本体往往规模较大,Gao 等人[13]提出一种基于近似核方法的本体理由求解策略,使局部搜索算法体现出更好的性能.Ye 等人[14]则利用关键公理的概念,通过对缩放阶段的修改使得选择函数能够避免不必要的检测,提高了求解一个理由的效率.以上的研究实际上是借用了高性能的 SAT 求解器的黑盒方法,在大规模的生物医学本体中体现出了极高的求解效率,但无法适用于表达能力更强的本体.

对强表达描述逻辑本体进行公理定位,由于本体的复杂度高难于处理往往会极大地影响求解效率,为了提高公理定位求解效率,本文提出了面向强表达描述逻辑本体的后继式公理定位方法.该方法结合了基于 Tableau 的方法和 EL 本体定位方法中采用的编码方式,通过一系列转换规则将描述逻辑表达式转换为多类型的子句等式逻辑表达式,能够适用于表达能力更强的本体.它所利用的后继式判定算法最早适用于 EL 本体[15],接着扩展到了 ALCH 本体[16]中.后来,Horrocks 等人[17]为了使得这样的后继式方法能够用于表达能力更强的 ALCHI 本体,引入了上下文的概念并开发了一个统一的推理框架.目前该推理框架已经由 Horrocks 等人[18]采用现收现付的方式被扩展到除了数据类型外的所有描述逻辑语言中.与现有工作相比,本文创新如下: (1) 利用多类型的子句等式逻辑编码,将 Rafael 等人[19]提出的面向 ALC 本体的后继式公理定位扩展到 ALCHIQ 的强表达本体后继式公理定位; (2) 与现有的后继式公理定位的理论证明工作相比[19],本文首次实现了后继式白盒公理定位推理机,完成从理论到实践、和实验测试的全流程工作; (3) 参照基于 Tableau 和基于 SAT 的公理定位工作,本文提出基于后继式的黑盒测试方法,完善了后继式公理定位的方法体系,框架图如图 1 所示.

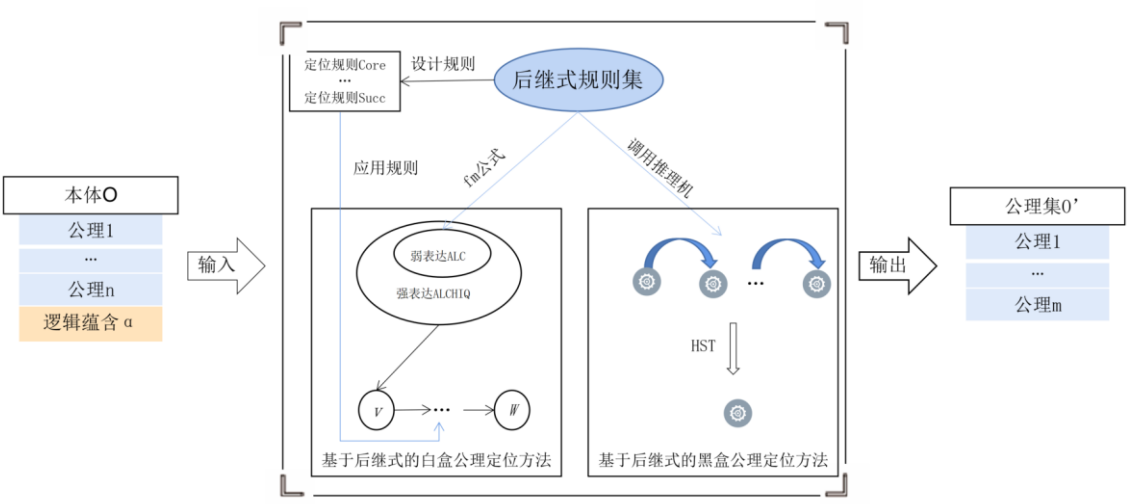

图 1 后继式公理定位方法体系框架图

本文第 1 节介绍所需的基础知识,首先介绍了描述逻辑和 DL 表达式以及描述逻辑公理和 DL 子句之间的转换规则,同时给出了后继式判定算法的相关概念.第 2 节介绍本文所采用的基于后继式的白盒定位算法.第 3 节中介绍了基于后继式的黑盒定位算法.第 4 节中通过与其他推理机的对比实验验证了两个定位算法的可行性和有效性.最后总结全文.

1 基础知识.

1.1 DL表达式

描述逻辑由概念、角色、个体和构造算子这四个基本元素组成,构造算子的多少决定了描述逻辑的语言表达能力的强弱,构造算子越多则表达能力越强.描述逻辑的语法是通过逻辑运算符号来定义的,强表达描述逻辑 $\mathcal{ALCHIQ}$ 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)、n(数量限制)、$\exists r.A$ (存在限定)和 $\forall r.A$ (全称限定)构成,并允许角色之间存在层次关系和逆角色.描述逻辑的语义则是通过解释 I 来定义的, $\mathrm{I}=(\Delta^{\mathrm{I}},\cdot^{\mathrm{I}})$ 是由解释论域 Δ^{I} 和解释函数 $\cdot^{\mathrm{I}}$ 组成的解释,其中 Δ^{I} 是一个非空集合,表示某个领域内的所有对象,解释函数 $\cdot^{\mathrm{I}}$ 将概念 A 映射为 Δ^{I} 的一个子集 $A^{\mathrm{I}}\subseteq\Delta^{\mathrm{I}}$, 将角色 R 映射为 Δ^{I} 上的一个二元关系 $R^{\mathrm{I}}\subseteq\Delta^{\mathrm{I}}\times\Delta^{\mathrm{I}}$.

描述逻辑表达式可以通过一种保留了可满足性和蕴含关系的方式转换为多类型的子句等式逻辑表达式,其中多类型签名为 $\langle\Sigma^S,\Sigma^F\rangle$, 它的 Σ^S 是一组类型的非空集合,而 Σ^F 是由函数符号组成的一个可数集合,等式(equality)是由两个类型相同的项 s 和 t 组成的表达式 $s\approx t$. 多类型的子句等式逻辑中,文字(literal)是一个等式或不等式,子句(clause)为形如 $\forall\vec{x}(\Gamma\to\Delta)$ 的语句,其中主体(body) Γ 是由等式构成的合取式,头部(head) Δ 是由文字构成的析取式, $\forall\vec{x}$ 表示出现在这个子句中的所有变量,这里的全称量词往往被省略.基于非空集合 S 的严格的(不完全)顺序 $\succ$ 是一个 S 中元素间的非自反并且可传递的二元关系,严格顺序 $\succ$ 通过它的自反闭包包含了不严格顺序 $\pm$.

DL 表达式使用一个双类型的 DL 签名 $\langle$a,p$\rangle$,其中类型 a 代表标准一阶逻辑项,类型 p 代表标准一阶逻辑原子.形如 $f_j(t)$ 的项是 t 的 f_j 后继,而 t 是它的前驱.DL 签名中的变量是类型为 a 的 $\{x\}\cup\{z_i\}_{i\geq 1}$, 其中 x 称作中心变量, z_i 称作相邻变量.DL-a-term 是形如 z_i, x 或 $f_i(x)$ 这样的项,DL-p-term 是形如 $B_i(z_j)$, $B_i(x)$, $B_i(f_j(x))$, $S_i(z_j,x)$, $S_i(x,z_j)$, $S_i(x,f_j(x))$ 或 $S_i(f_j(x),x)$ 这样的项.DL 文字是形如 $A\approx true$ 这样的等式或 A ,其中 A 是一个 DL-p-term,也可以是两个 DL-a-term 构成的等式或不等式.DL 子句 $\forall\vec{x}(\Gamma\to\Delta)$ 的主体 Δ 只能是形如 $B_i(x)$, $S_i(z_j,x)$ 或 $S_i(x,z_j)$ 这样的项,并且 DL 子句的头部 Γ 只能是 DL 文字.

O是一个ALCHIQ本体,对其标准化后O的公理如表1左侧所示,按照上述多类型的子句等式逻辑将本体的公理转换为DL子句[20],如表1右侧所示.

表1 ALCHIQ本体公理到DL子句的转换规则

序号	本体公理转换为DL子句			
DL1	$\mathop{\text{簧}}_{1\le i\le n} B_i \quad ? \quad \mathop{}_{n+1\le i\le m} B_i$	I	$\bigwedge_{1\le i\le n} B_i(x) \to \bigvee_{n+1\le i\le m} B_i(x)$	
		I	$B_1(x) \to S(x, f_i(x))$	for $1\le i\le n$
DL2	$B_1 ? \ge nS.B_2$	I	$B_1(x) \to B_2(f_i(x))$	for $1\le i\le n$
		I	$B_1(x) \to f_i(x) \not\approx f_j(x)$	for $1\le i<j\le n$
DL3	$\exists S.B_1 ? \ B_2$	I	$S(z_1,x)\wedge B_1(x) \to B_2(z_1)$	
DL4	$B_1 ? \le nS.B_2$	I	$S(z_1,x)\wedge B_2(x) \to S_{B_2}(z_1,x)$	for fresh S_{B_2}
		I	$B_1(x)\wedge \bigwedge_{1\le i\le n+1} S_{B_2}(x,z_i) \to \bigvee_{1\le i<j\le n+1} z_i \approx z_j$	
DL5	$S_1 ? \ S_2$	I	$S_1(z_1,x) \to S_2(z_1,x)$	
DL6	$S_1 ? \ S_2^-$	I	$S_1(z_1,x) \to S_2(x,z_1)$	

1.2 后继式判定算法

给定一个任意ALCHIQ本体O,判断O‘ Q是否成立,其中$Q=\Gamma_Q\to\Delta_Q$是一条查询子句.根据Bate等人[20]提出的后继式判定算法将本体派生出的逻辑蕴含表示为上下文(*context*)子句.不同于上文中DL子句的定义,上下文子句只允许具有特殊意义的变量x和y,其中变量y对应于x的前驱.上下文结构(context structure)是一个有向图,它的顶点(也称上下文)代表输入本体模型中的域元素集合,边由后继函数符号标记.上下文结构分配给每个顶点v以下信息:

(1) 上下文v的核心$core_v$决定了用v表示的域元素的集合,只包含了两个变量x和y;

(2) 一组不含常量的上下文子句S_v,只允许形如x,y以及$f(x)$这样的项.每一个上下文子句$\Gamma\to\Delta$表示本体的一个逻辑蕴含,并且$core_v\subseteq\Gamma$.简而言之,上下文子句可以写作$\Gamma'\to\Delta$, 其中$\Gamma=core_v\wedge\Gamma'$;

(3) 上下文v利用推理规则[20]选择文字和上下文子句所依赖的顺序$\succ_v$进行参数化.

上下文结构中的边用后继函数符号来标记,从上下文v到w的用函数符号f所标记的边可表示若$core_w\neq\{\}$, 则v中的每个元素都有一个w中的元素为它的f后继.要判断O‘ Q是否成立,后继式判定算法需要构造一个上下文结构,它的上下文表示在本体O的一个模型中Γ_Q的所有实例,并对这个上下文结构应用推理规则直到规则应用达到饱和,即无法再应用推理规则时,查询是否存在子句$\bot\to\Delta_Q$.

2 基于后继式的白盒定位算法

2.1 公理定位问题

给定本体O和一条公理α,若O‘ α,则称公理α是本体O的一条逻辑蕴含.公理定位的主要目标是找到与一个给定逻辑蕴含有关的极小公理集,即找到这样的公理集O'满足$O'\subseteq O$,使得O'‘ α成立.

若本体中的每一条公理都与一个唯一的命题变量$lab(\alpha)$相关联, $lab(O)$表示与本体O中公理对应的所有命题变量的集合.以本体O_1为例,O_1共有四条公理: C 趁$\exists r.C,\exists r.C \quad D,C$ 趔$\forall r.D,C \quad D$? $\bot$. 分别用四个不同的命题变量ax_1,ax_2,ax_3,ax_4作为这四条公理的标签,则$lab(O_1)=\{ax_1,ax_2,ax_3,ax_4\}$.命题赋值用来表示一组为真的变量集合,对于一个赋值V和公理集O,O的V-投影为集合$O_V:=\{\alpha\in O\mid lab(\alpha)\in V\}$. 如本体$O_1$的一个赋值$V_1=\{ax_1,ax_3\}$,它的$V_1$-投影为集合$O_{V_1}:=\{C$ 趁$\exists r.C,C \quad \forall r.D\}$. 基于$lab(O)$的单调布尔公式$\varphi$是只使用$lab(O)$中的变量、析取$(\wedge)$和合取$(\vee)$的布尔公式.给定公理集$O$和一条公理$\alpha$,基于$lab(O)$的单调布尔公式$\varphi$被称作定位公式,若对于所有赋值$V\subseteq lab(O)$, O‘ α当且仅当V满足φ. 定位公式φ所对应的公理集即为与O‘ α有关的所有极小公理集.例如O_1‘ C ô $\bot$的定位公式$\varphi=ax_1\wedge ax_4\wedge(ax_2\vee ax_3)$, 对于所有的赋值$V_1\subseteq lab(O_1)$, O_1‘ C ô $\bot$

成立当且仅当 V_1 满足 φ_1. 定位公式 φ_1 所对应的公理集{ax$_1$,ax$_2$,ax$_4$},{ax$_1$,ax$_3$,ax$_4$}就是 O_1 ‘ C ô ⊥ 的所有极小公理集.

2.2 后继式定位规则

Rafael 等人[19]提出弱表达的描述逻辑本体的后继式推理规则,当描述逻辑本体从弱表达的AΛX本体扩展到强表达的AΛXHIΘ时,这样简单的规则集无法完成推理任务.而在[18]中研究者针对AΛXHIΘ本体的后继式判定算法根据多类型的子句等式逻辑对描述逻辑本体进行编码,并引入上下文的概念开发了一个统一的推理框架.[19]中提到的扩展方法只需要将规则分解为条件部分和结果部分,对应的判定算法中每运用一条推理规则,则定位过程中将这条规则的条件部分所包含的公理添加到结果部分得到的公理的标签中,再利用 fm 公式来确保最终所求的逻辑蕴含对应的标签为其定位公式.当描述逻辑本体的表达能力增强时, AΛXHIΘ本体的判定算法中利用的推理规则除了根据规则的条件部分新增结果部分对应的公理,还有避免冗余重复的删除规则和扩展有向图中上下文结构时用到的 Succ 规则和 Pred 规则,要求我们在推理框架下单独考虑不同规则的扩展问题,不能如[19]一样直接利用 fm 公式.如何设计扩展的定位规则集并证明是本文所提出的白盒定位算法的核心内容.

[18]中针对AΛXHIΘ本体的后继式判定算法根据多类型的子句等式逻辑对描述逻辑本体进行编码,并引入上下文的概念开发了一个统一的推理框架.首先用一个唯一的命题变量标记每一条由公理转换规则得到的 DL 子句,本体的每一条蕴含 α 对应的上下文子句都可以用一个单调布尔公式 φ_α 标记.用 $\alpha:\varphi_\alpha$ 来表示蕴含 α, 前半部分是蕴含 α 所对应的上下文子句,后半部分是该子句的布尔公式标签.接着给出以下七条由后继式推理规则作修改得到的定位规则来追踪推理的具体过程:

定位规则 Core:对于 $core_v$ 中的每一个原子 A 在 v 的子句集 S_v 中添加一条上下文子句 $\bot\to A:\varphi_\alpha,\varphi_\alpha:=\bot$..

定位规则 Hyper:若 $\bigwedge_{i=1}^n A_i\to\Delta:\varphi_\alpha\in O,\Gamma_i\to\Delta_i\vee A_i\sigma:\varphi_{\beta_i}\in S_v$, 替换 σ 满足 $\sigma(x)=x$, Δ_i ½$_v$ $A_i\sigma$, $1\le i\le n$, 则向 S_v 中新增子句 $\bigwedge_{i=1}^n\Gamma_i\to\bigvee_{i=1}^n\Delta_i\vee\Delta\sigma:\varphi_\gamma$, $\varphi_\gamma:=\varphi_\alpha\wedge\bigwedge_{i=1}^n\varphi_{\beta_i}$.

定位规则 Eq:若 $\Gamma_1\to\Delta_1\vee s_1\approx t_1:\varphi_\alpha\in S_v,t_1$ 浇 s_1,Δ_1 $_v$ $s_1\approx t_1$, 且 $\Gamma_2\to\Delta_2\vee s_2\otimes t_2:\varphi_\beta\in S_v,\otimes\in\{\approx,\not\approx\},s_2\succ_v t_2$, Δ_2 ½$_v$ $s_2\otimes t_2,s_2|_p=s_1$, 则向 S_v 中新增子句 $\Gamma_1\wedge\Gamma_2\to\Delta_1\vee\Delta_2\vee s_2[t_1]_p\otimes t_2:\varphi_\gamma$, $\varphi_\gamma:=\varphi_\alpha\wedge\varphi_\beta$.

定位规则 Ineq:若 $\Gamma\to\Delta\vee t\not\approx t:\varphi_\alpha\in S_v$, 则向 S_v 中新增子句 $\Gamma\to\Delta:\varphi_\beta$, $\varphi_\beta:=\varphi_\alpha$.

定位规则 Factor: 若 $\Gamma\to\Delta\vee s\approx t\vee s\approx t':\varphi_\alpha\in S_v,\Delta\cup\{s\approx t\}$ ½$_v$ $s\approx t',s\succ_v t'$, 则向 S_v 中新增子句 $\Gamma\to\Delta\vee t\not\approx t'\vee s\approx t':\varphi_\beta$, $\varphi_\beta:=\varphi_\alpha$.

定位规则 Elim:若 $\Gamma\to\Delta:\varphi_\alpha\in S_v,\Gamma\to\Delta:\varphi_\beta\in S_v\backslash(\Gamma\to\Delta)$, 则从 S_v 中删除子句 $\Gamma\to\Delta,\varphi_\beta:=\varphi_\alpha\vee\varphi_\beta$.

定位规则 Pred:若 $\langle u,v,f\rangle\in\varepsilon,\bigwedge_{i=1}^n A_i\to\bigvee_{i=l+1}^{l+n}A_i:\varphi_\alpha\in S_v$, 并且 $\Gamma_i\to\Delta_i\vee A_i\sigma:\varphi_{\beta_i}\in S_u$, Δ_i½ $_u$ $A_i\sigma,1\le i\le l$,其中 $A_i\in\mathrm{Pr}(O)$[19],$l+1\le i\le l+n$, 则向 u 的子句集 S_u 中新增子句 $\bigwedge_{i=1}^l\Gamma_i\to\bigvee_{i=}\Delta_i\vee\bigvee_{i=l+}^{l+n}A_i\sigma:\varphi_\gamma$, $\sigma=\{x\mapsto f(x),y\mapsto x\}$, $\varphi_\gamma:=\varphi_\alpha\wedge\bigwedge_{i=1}^{l+n}\varphi_{\beta_i}$.

定位规则 Succ:若 $\Gamma\to\Delta\vee A:\varphi_\alpha\in S_u,\Delta$ ½$_u$ A, A 中包含 $f(x)$,则令 $\langle v,core',\succ'\rangle=strategy($ f, K, D),若 $v\notin V$, 令 $\succ_v:=\succ_v\cap\succ'$; 若 $v\notin V$, 令 $V:=V\cup\{v\},core_v:=core',\succ_v:=\succ'$,$S_v:=\varnothing$. 往 ε 中新增一边 $\langle u,v,f\rangle$, 若 $A'\in K_2\backslash core'$, 向 S_v 中新增子句 $A'\to A':\varphi_{\beta_i}$. 其中 $\sigma=\{y\mapsto x,x\mapsto x$ $(f_1=)x\}\in,K$ [$\{A\to 9$]$'\sigma\notin u$)O $K_2=\{A'\in Su(O)\mid\Gamma'\to\Delta'\vee A'\sigma\in Su(O)\},\Delta'$ ½$_u$ $A'\sigma,\varphi_{\beta_i}:=\varphi_\alpha$.

不同于[19]中的简单扩展,定位规则 Core 在利用推理规则 Core 向子句集中添加上下文子句时,为该子句添加标签 $\varphi_\alpha:=\bot$. 定位规则 Hyper 和 Eq 以及规则 Pred 在利用对应的推理规则由多条公理得到一条新增的公理时,用那几条公理标签的析取式来表示新增公理的标签.定位规则 Ineq 在利用推理规则 Ineq 由公理 α 得到新增的公理 β 时,用 α 的标签来表示 β 的标签.比较特殊的两条定位规则是规则 Elim 和 Succ,其中 Elim 推理规则会根据公理 α 和 β 删去其中冗余的公理 α, 那么此时则需要修改公理 β 的标签公式, $\varphi_\beta:=\varphi_\alpha\vee\varphi_\beta$ 用析取式表示被保留下的公理 β 既可以如原本的推理过程得到,也可以由得到公理 α 的过程得到.定位规则 Succ 通过扩展策略构造新的上下文结构,往该上下文结构对应的子句集中新增子句,新增的子句对应的标签公式来自于被扩展的原有的上下文结构中符合推理规则的子句.

按照如上七条定位规则构成的规则集能够处理强表达的描述逻辑本体的公理定位问题,同样也适用于弱表达的描述逻辑本体.以AΛX本体为例,[19]中提出的定位方法不需要对本体进行编码,直接在定位状态下对本

体应用七条后继式推理规则就可以求得理由.而本文的方法用于AΛX本体时,由于本体中所包含的构造算子比较简单,所以在编码后对子句集匹配定位规则条件时更加容易,算法终止时同样能够得到逻辑蕴含的定位公式.

2.3 后继式定位算法及证明

基于后继式的白盒定位算法为由本体 O 通过编码得到的 DL 子句集中的每一条子句都添加标记得到定位子句集 O^{Pin},上下文 v 的子句集中所有子句添加标记得到定位子句集 S_v^{Pin},为上下文结构 D 中的所有子句添加标记得到上下文定位结构 D^{Pin}.对 O^{Pin} 和 D^{Pin} 应用上述定位规则,当没有定位规则可以被应用的时候,称当前的上下文定位结构 D^{Pin} 达到饱和.算法的具体过程见算法 1.

算法 1. 求给定逻辑蕴含的所有理由的基于后继式的白盒定位算法.

输入:公理集 O,公理 Q;

输出:O ‘ Q 相关的所有极小公理集.

Step 1. 构造一个空的上下文结构 D 并选择一个扩展策略[20]K;

Step 2. 在 D 中引入一个上下文 q,其中 $core_q = \Gamma_Q$,为所有的公理和逻辑蕴含添加标记;

Step 3. 对 D^{Pin} 和 O^{Pin} 应用上述定位规则直到没有规则可以被应用,达到饱和;

Step 4. 当且仅当 $\Gamma_Q \to \Delta_Q : \varphi_Q \in S_v^{Pin}$ 时,O ‘ Q 成立,这条子句对应的布尔公式标记 φ_Q 恰为它的定位公式.

Step 5. 返回 φ_Q 对应的所有极小公理集;

在后继式判定算法中若某一状态下某一条推理规则可以被应用,则对应到定位算法中此条规则对应的定位规则仍可被应用.也就是说,判定过程中的规则应用情况与公理定位算法中的规则应用情况是一样的.不同的是判定算法改变上下文结构,删除或新增子句集中的子句,而定位算法同时利用定位规则追踪推理的具体过程,修改推理过程中所删除或增加的子句对应的单调布尔公式.同样的,应用定位规则产生的所有逻辑蕴含也可以通过推理规则的应用产生.

用 A 表示判定算法当前状态,A^{Pin} 则表示定位算法中当前定位状态,A_0 表示算法开始时的初始状态,B 表示算法结束时的终止状态,为了方便理解,这里的状态也可简单地看作逻辑蕴含集合.将 V-投影的概念扩展到 A^{Pin} 中,则 $A_V := \{\alpha \mid \alpha : \varphi_\alpha \in A^{Pin}, V \text{ ‘ } \varphi\}$.

引理 1. 令 A^{Pin}、B^{Pin} 为定位状态,V 是一组赋值.若 $A^{Pin} \to^* B^{Pin}$,则有 $A_V \to^* B_V$

证明: 如果当前状态可应用的是推理规则 Elim,则 A 状态下子句集中删除某一条重复子句得到 B 状态,A^{Pin} 与 B^{Pin} 中所包含的逻辑蕴含是一致的,只是蕴含的标记不一致,即 $A_V = B_V$.如果可应用的是其他的规则 R,则应用规则 R 后,由 A 中新增的逻辑蕴含得到 B,即 $A_V \to_R B_V$.因此,若定位状态 A^{Pin} 下能够应用规则 R,即 $A^{Pin} \to_R B^{Pin}$,则 $A_V \to_R B_V$ 或 $A_V = B_V$. 得证.

因为所有的子句标记都是单调的布尔公式,$V_\cdot = lab(O)$ 这样的使得每一个命题变量都为真的赋值满足所有标签,因此对于所有的定位状态 A^{Pin} 有 $A_{V_\cdot} = A$.因此引理 1 就意味着定位算法并没有构造新的逻辑蕴含,而只是给这些蕴含添加了布尔公式标记.后继式判定算法终止时,后继式定位算法也随之终止.

引理 2. 令 A^{Pin} 为定位状态,V 是一组赋值.若 A^{Pin} 是饱和的,即无法再对 A^{Pin} 应用任何定位规则,则 A_V 也是饱和的,无法对 A_V 应用任何推理规则.

证明:如果当前状态可应用的是 Elim 规则,即 A_V 中存在重复的子句 α 和 β,其中 $A_V \subseteq A$.那么 A^{Pin} 中必然也有这样的两条定位子句 $\alpha : \varphi_\alpha$ 和 $\beta : \varphi_\beta$,则 A^{Pin} 也能够应用 Elim 定位规则.如果可应用的是其他规则,即 A_V 中包含了应用规则 R 所需的条件子句,其中 $A_V \subseteq A$. 那么 A^{Pin} 中必然也包含定位规则 R^{Pin} 的条件子句,即 A^{Pin} 能够应用定位规则 R^{Pin}. 也就是说,假设存在一条规则 R 能够应用于 A_V,那么这个规则对应的定位规则 R^{Pin} 也必然能应用于 A^{Pin}. 得证.

至此我们可以证明上述的后继式定位算法确实是一个能够完成推理任务的公理定位算法,对 O^{Pin} 和 D^{Pin} 应用定位规则直到无法再应用定位规则达到饱和,饱和时的定位状态为 A^{Pin},则查询子句 $\alpha : \varphi_\alpha \in A^{Pin}$,$\varphi_\alpha$ 即为 α 的定位公式.

定理 3(基于后继式的白盒定位算法的正确性) 给定本体 O 和一条逻辑蕴含 α,其中 $O \not\models \alpha$. 当对 O^{Pin} 和 D^{Pin} 应用定位规则达到饱和时, φ_α 为 α 关于 O 的定位公式,对应得到的公理集是 $O \not\models \alpha$ 的极小公理集.其中, O^{Pin} 为由本体 O 转换成的带有标记的定位子句集, D^{Pin} 为由上下文结构 D 给所有子句添加标记后得到的定位结构, φ_α 为算法终止时子句 α 的标记.

证明:要证明 φ_α 为 α 关于 O 的定位公式,即对所有赋值 $V \subseteq lab(O)$, 有 $(O_v,\alpha) \in P$ 当且仅当 $V \not\models \varphi_\alpha$.

假设 $(O_v,\alpha) \in P$,也就是 $O_v \not\models \alpha$.因为后继式定位算法对于每一条查询公理都必然终止且后继式判定算法具有完备性,因此存在这样的一个饱和状态 B,满足 $A_0 \to^* B$,其中 A_0 为初始状态, $\alpha \in B$. 由条件可知, $A_0^{Pin} \to^* A^{Pin}$, A^{Pin} 是定位规则达到饱和时的状态.由引理 1 可知 $A_{0_V} \to^* A_V$,由引理 2 可知 A_V 是饱和的.因此由后继式判定算法的正确性可得, $A_V = B$.因为 $\alpha \in A_V$,因此有 $V \not\models \varphi_\alpha$.相反,假设 $V \not\models \varphi_\alpha$, 则有 $\alpha : \varphi_\alpha \in A^{Pin}$. 由条件可知 $A_0^{Pin} \to^* A^{Pin}$,其中 A^{Pin} 是饱和的.由引理 1 可知, $A_{0_V} \to^* A_V$. 因为 $V \not\models \varphi_\alpha$, 有 $\alpha \in A_V$, 又因为后继式判定算法的有效性,因此有 $A_{0_V} \not\models \alpha$, 即 $O_v \not\models \alpha$. 因此, φ_α 即为 α 关于 O 的定位公式,对应得到的公理集是 $O \not\models \alpha$ 的极小公理集.得证.

2.4 后继式定位算法示例

以本体 O_2 为例,表 2 给出了按照表 1 中的转换规则由 O_2 的公理集所得到的 DL 子句集,其中每一条子句用唯一的命题变量 $ax1$、$ax2$ 等标记,子句序号用(1)、(2)等表示.图 2 给出了按照后继式定位算法构建上下文定位结构探求公理 $O_2 \not\models A \sqsubseteq D$ 的所有理由的具体过程.

表 2 本体 O_2 根据转换规则对应的 DL 子句集

$A \sqsubseteq \exists S.B$	⇔	$ax1: A(x) \to S(f_0(x), x)$	(1)	
		$ax2: A(x) \to B(f_0(x))$	(2)	
$C_1 \sqcap C_2 \sqsubseteq \bot$	⇔	$ax3: C_1(x) \wedge C_2(x) \to \bot$	(3)	
$B \sqsubseteq \exists S.C_i$	⇔	$ax4: B_1(x) \to S(x, f_i(x))$	(4)	for $1 \le i \le 2$
		$ax5: B_1(x) \to B_2(f_i(x))$	(5)	
$C_i \sqsubseteq D$	⇔	$ax6: C_i(x) \to D(x)$	(6)	for $1 \le i \le 2$
$B \sqsubseteq \le 2.S$	⇔	$ax7: B(x) \wedge \bigwedge_{1 \le i \le 3} S(x, z_i) \to \bigvee_{1 \le j < k \le 3} z_j \approx z_k$	(7)	

首先构建一个上下文 v_0 并且对其初始化产生子句(8),它的定位公式为 $\bot$, 如图 2 所示.这样确保了算法产生的整个上下文定位结构中所表示的每一个解释都包含一个 A 表示的基项.接着,应用 Hyper 定位规则生成子句(9)和(10),这两个子句的定位公式分别为 $ax1$ 和 $ax2$. 此时,可以由(4)和(10)得到子句 $\bullet \to S(f_0(x), f_1(f_0(x)))$,但是由于嵌套的增加,这样很容易导致算法无法终止.因此,在应用 Hyper 定位规则时要求将 DL 子句中的变量 x 映射到上下文子句中的变量 x 中,这样在每个上下文中应用 Hyper 定位规则只会得到关于 x 的逻辑蕴含,从而避免了冗余推导.

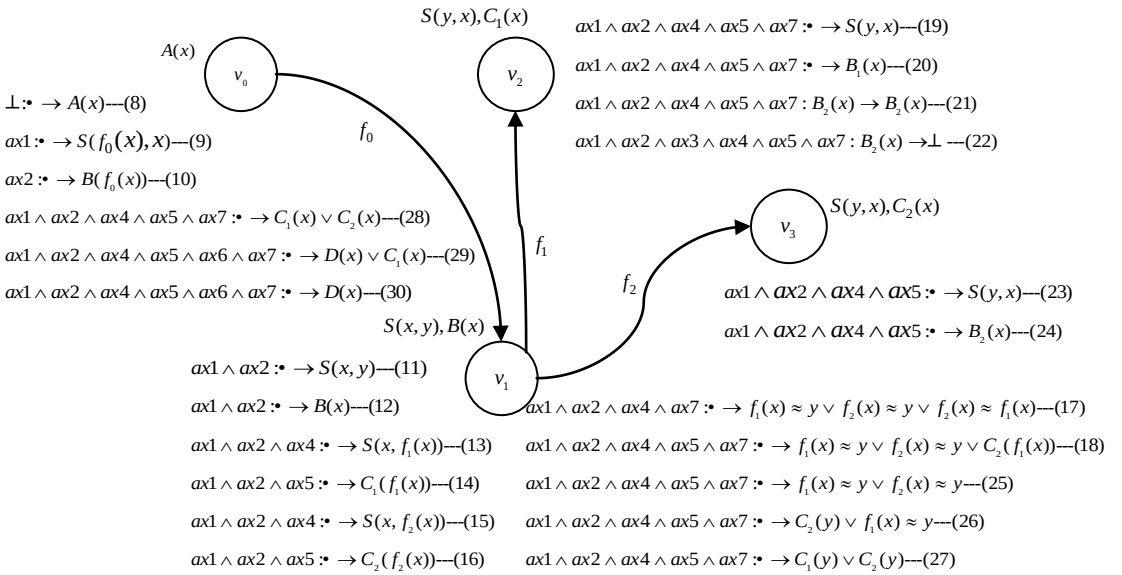

图 2 后继式定位算法判断 O_2 ‘ $A\hat{o}$ D 是否成立并计算其理由的具体过程

接下来应用 Succ 定位规则处理子句(9)和(10)中的函数符号 f_1,为了确定将哪些信息传递给后继 successor,后继式定位算法中引入了一组后继触发器 Su(O)[20].在此例中,DL 子句(7)的主体 body 中包含了原子 $B(x)$ 和 $S(x,z_i)$,并且 z_i 可以映射到 x 的前继或后继中,对子句(7)应用定位规则的上下文将对这个子句的前继信息感兴趣,因此在 Su(O)中添加 $B(x)$ 和 $S(x,y)$.在产生新的上下文时采用的是急切策略(eager strategy),所以 Succ 规则引入了上下文 v_1,它的核心设为 $B(x)$ 和 $S(x,y)$.根据 Core 定位规则产生子句(11)和(12),它们的定位公式都为 $ax1\wedge ax2$.接下来应用 Hyper 定位规则产生子句(13)-(16),它们的定位公式如图 2 所示.此时已有的定位子句集中有足够的信息由(1)、(2)、(4)、(7)和(8)推出子句(17).对子句中的文字进行排序,并且只对排序最大的文字应用规则,由此得出子句(18),它的定位公式为 $ax1\wedge ax2\wedge ax4\wedge ax5\wedge ax7$.

子句(13)、(14)和(18)中包含函数符号 f_2,因此由 Succ 定位规则引入上下文 v_2.根据子句(14)可知, $C_1(x)$ 对上下文 v_2 所代表的所有基项都成立,因此将 $C_1(x)$ 添加到 v_2 的核心中.相反地,原子 $C_2(f_1(x))$ 出现在子句(18)的析取子句中,这就意味着它在上下文 v_2 中可能不成立,因此将 $C_2(x)$ 添加到子句(21)的主体部分.接着由子句(3),(20)和(21)应用 Hyper 定位规则产生新的子句(22),定位公式为 $ax1\wedge ax2\wedge ax3\wedge ax4\wedge ax5\wedge ax7$.

子句(22)本质上表达的是 $C_2(f_1(x))$ 不应该在前继中成立,这由 Pred 定位规则以子句(25)的形式传播到上下文 v_1 中.这个逻辑蕴含(25)被理解为由子句(18)和(22)应用 Hyper 定位规则得到的,同时观察到在上下文 v_1 中的项 $f_1(x)$在上下文 v_2 中被表示为变量 x.

由子句(14)和(26)应用 Eq 定位规则产生子句(27),这个子句本质上表达的是当前上下文的前继必须满足 $C_1(x)$ 或 $C_2(x)$.前继触发器的集合 Pr(O)[20]中包含 $C_1(y)$ 和 $C_2(y)$, 因此可以由子句(27)应用 Pred 定位规则得到子句(28),定位公式为 $ax1\wedge ax2\wedge ax4\wedge ax5\wedge ax7$.最后再通过两次 Hyper 定位规则的应用终于得到了子句(30),定位公式为 $ax1\wedge ax2\wedge ax4\wedge ax5\wedge ax6\wedge ax7$,算法终止. O_2 ‘ $A\hat{o}$ D 成立并且推出这个逻辑蕴含的极小公理集是{(1),(2),(4),(5),(6),(7)}.

3 基于后继式的黑盒定位算法

3.1 逻辑蕴含的单一理由求解过程

白盒定位算法的核心是基于后继式判定算法的推理规则设计出能够计算公理对应标签的新的定位规则,

它的优势是当定位规则应用达到饱和时就可以得到蕴含的定位公式,并且可以直接求出蕴含的所有理由,而不需要重复调用推理机来判断蕴含能否成立,也不需要在求出一个理由后再利用别的如碰集树这样的方法,来求解所有理由.但是白盒方法往往要求我们针对不同的推理规则设计对应的定位规则,且随着本体表达能力增强还需要重新设计新的定位规则.但黑盒定位算法则由于它不局限于具体的描述逻辑和推理机制,不需要设计对应的定位规则,可以直接调用后继式判定算法作为推理机.并且,当面向表达能力更强的描述逻辑本体进行公理定位时,黑盒方法也不需要考虑新增的构造算子带来的定位规则的变化仍可以处理公理定位问题.因此,本文提出基于后继式的黑盒定位算法来求解逻辑蕴含的理由.

要计算给定逻辑蕴含 α 的一个极小公理集,主要分为扩张和收缩两个阶段[5].在扩张阶段,随机从本体 O 中选择公理,每选择出一条公理则调用后继式判定算法判断一个新构造的本体 O' 能否推导出 α.将 α 添加到 O' 中,直到 O' ‘ α, 则扩张阶段完成.在收缩阶段,对扩张阶段得到的本体 O' 中的公理进行逐条删除,每删除一条公理则调用后继式判定算法判断 O' 能否推导出 α,若 O' ‘ α 则继续删除,直到删除当前公理后得到的新公理集不能推出 α,将此条公理重新加入公理集中,将 O' 遍历完成逐条删除公理后所得的公理集则为 α 的一个理由.

由于算法在扩张阶段和收缩阶段进行公理选择时是随机的,没有针对性,导致算法的效率较低,因此往往考虑在扩张阶段利用相关性来选择特定的公理,使算法选择公理时更具有针对性.一般的黑盒方法根据概念相关、公理直接相关和间接相关的概念来建立相关公理图,并基于深度优先或宽度优先提出选择函数的搜索策略.而当黑盒定位方法的求解过程调用的是基于后继式判定规则的推理机来判断本体能否得到某给定逻辑蕴含时,与基于 Tableau 的黑盒定位方法不同,要判断蕴含是否成立,需要对本体的子句集应用一系列对应的判定规则[20].黑盒定位方法并不依赖于具体的推理机,一般黑盒定位方法的选择函数优化策略同样也适用于本文基于后继式的黑盒定位算法的优化.考虑到每一次调用基于后继式判定规则的推理机时,要判断 $\Gamma_Q \to \Delta_Q$ 能否成立都需要首先构建一个以 Γ_Q 为核的根上下文,再按照算法 1 的具体过程建立有向图.因此,我们的选择函数构建 Γ_Q 的相关公理图,并采用基于宽度优先的搜索策略来决定扩张阶段选择的公理.

以 2.4 节中的本体 O_2 为例,按照如上求解过程探求公理 O_2 ‘ $A\hat{o}\ D$ 的理由.O_2 中一共有五条公理如表 2 左侧一栏所示,将这五条公理从上到下分别标记为 $\alpha_1,\alpha_2,\alpha_3,\alpha_4,\alpha_5$. 构建概念 A 的相关公理图,按照宽度优先的策略首先选择公理 α_1 添加到一个空的公理集 O' 中,调用[20]中的推理算法判断逻辑蕴含 O' ‘ $A\hat{o}\ D$ 不成立.继续选择公理 α_3 添加到 O' 中,调用推理算法判断蕴含仍不成立.按照宽度优先的策略继续选择公理 α_5 添加到 O' 中,调用推理算法判断蕴含仍不成立.继续依次选择公理 α_2 和 α_4 添加到 O' 中直到 $O'=\{\alpha_1,\alpha_2,\alpha_3,\alpha_4,\alpha_5\}$时蕴含成立,扩张阶段完成.接着为了确保得到的公理集是给定逻辑蕴含的极小公理集,开始收缩阶段.逐条删除 O' 中的公理,首先遍历公理集 O' 选择一条公理 α_1, 删除后得到新的 O', 调用推理机得出 O' ‘ $A\hat{o}\ D$ 不成立,将删除的公理 α_1 重新加入到公理集中.再继续遍历公理集选择一条公理 α_2,删除这条公理后得到新的 O', 调用推理机目标公理仍不成立,将删除的公理 α_2 也重新加入到公理集中.重复上述操作,当选择公理 α_3 时,得到的公理集为集合 $O'=\{\alpha_1,\alpha_2,\alpha_4,\alpha_5\}$, 调用推理机发现 O' ‘ $A\hat{o}\ D$ 成立,则重复上述操作继续遍历这样的新的 O' 直到遍历结束.最终得到的公理集为 $\{\alpha_1,\alpha_2,\alpha_4,\alpha_5\}$,这就是所求的 O_2 ‘ $A\hat{o}\ D$ 的理由.

与第 2.2 节中给出的定位规则不同,判定规则只会在子句集中新增或删除所需的子句,而不追踪推理的过程.与利用 SAT 求解器的黑盒方法对轻量级描述逻辑本体进行编码不同,该求解过程虽然也需要对描述逻辑本体进行编码,但是为了能够处理表达能力更强的本体,它将描述逻辑表达式编码为了多类型的子句等式逻辑表达式.不仅能够处理强表达的描述逻辑本体,还由于后继式判定推理机的优越性也能一定程度提高求解效率.

3.2 逻辑蕴含的所有理由求解过程

计算给定逻辑蕴含的所有理由的黑盒算法利用 Reiter 的碰集树(Hitting Set Tree)[21]方法,通过如上理由单一求解过程中得到公理的一个理由后,利用碰集树的方法再得到其他的理由.

定义 4(碰集)[21] 设 F 是一个集合簇,F 的碰集(hitting set)是一个集合 $H\subseteq\bigcup_{S\in F}S$,使得对所有 $S\in F$, $H\bigcap S\neq\varnothing$. 称 F 的一个碰集为极小的当且仅当它的任何一个真子集都不是 F 的碰集.

碰集树(HS-Tree)[21]方法定义了一个集合簇 F 的结点和边都有标记的碰集树 T_r ,其中树的结点通过 F 中的元素来标记,树的边通过该边的前继结点的标记中的元素来标记.以 F 中的某一个元素作为树的根结点,再依次

处理树中的结点,为该结点添加标记并构造新的结点来完成碰集树的构造.

定义 $P(n)$为从根结点到结点 n 的路径上边标记的集合,若存在集合簇 F 中的元素与 $P(n)$交集为空集,则用该元素作为结点 n 的标记,否则,结点 n 标记为'√' .如果 n 是一个标记为'√' 的结点,那么 $P(n)$则是集合簇 F 的一个碰集.F 的每一个极小碰集都是它所对应的碰集树中某个标记为'√' 的结点 n 的 $P(n)$.标记为'√' 的结点 n 对应的 $P(n)$并不包含 F 的所有碰集,但包含了 F 的所有极小碰集.

Reiter 的碰集树方法通常用于在给定一组冲突集的情况下找到所有极小碰集,但基于算法的对偶性,它也可以用于动态地找到所有冲突集(本文中的理由).在本节的所有理由求解过程中,调用单一理由求解过程计算得到给定蕴含的一个理由并将其设为碰集树(HS-Tree)的根结点,接着分别移除这个理由中的每个公理从而创建碰集树的新的分支,并在删除该条公理后的修改了的本体中沿着这些分支利用单一理由求解过程动态地寻找新的理由.为了计算出所有的理由,这个过程需要被彻底地完成.与 Reiter 的算法进行类比的好处是,我们可以利用后者提供的所有优化来加快搜索速度.

F 是给定逻辑蕴含的所有理由,一次访问需要调用一次单一理由求解过程来计算给定蕴含的一个理由.为了产生一棵尽可能小、只给出 F 的极小碰集并且访问次数最少的 HS-Tree,所有理由求解过程有早期路径终止和重用解释两条用于减少单一理由求解过程调用次数的优化方法.

4 实验分析

4.1 实验数据

本节中实验的主要目的是验证本文提出来的两种白盒和黑盒定位算法的可行性与有效性,根据这两个方法设计得到推理工具 CBPin_Glass_Box 和 CBPin_Black_Box 与其他已有的推理工具 Pellet[22]、FaCT++[23]和 EL2SAT[24]进行比较,分析该系统在表达能力不同的描述逻辑本体中的优劣性.此实验在 PC 机 VMware 虚拟机 64 位 Fedora 系统(2GB 的内存)下运行,实验测试所采用的数据来自一个标准本体库,主要来自 Open Biological ontology(OBO) Foundry、Gardiner 本体套件、Phenoscape 项目以及 GALEN 本体的几个变体.对所有本体进行预处理来解析本体导入,这样每个测试本体都能够包含在单个文件中,可以通过 OWL API 加载.由于我们的实验是基于一致本体计算给定逻辑蕴含的理由,因此不考虑不一致本体,也不考虑因为过于简单而不能提供任何有用的性能测试的本体(即,包含很少的公理或类的本体).又由于我们的定位算法主要面向的是强表达描述逻辑本体,因此从http://www.cs.ox.ac.uk/isg/ontologies中选取了表 3 所示的十个本体,并给出了这些本体的 ID、名称、描述逻辑表达能力、类的数量、属性的数量和 TBox 中公理的数量.所选取的这十个本体中既有如 00511 本体这样表达能力较强的,也有如 00365 本体这样表达能力较弱的,这样可以对比本文提出来的推理工具在表达能力强弱不同的本体中进行公理定位的效率.

表 3 实验数据集

ID	名称	描述逻辑表达能力	类的数量	属性的数量	TBox 公理数量
00004	BAMS-simplified	SHIF	1110	12	18813
00011	Biopax-level2	ALCHN(D)	41	33	333
00013	CommonSenseMapping	SHIN(D)	126	273	981
00015	DOLCE-Lite	SHI	37	70	279
00031	Galen-ians-full-	EL++	2749	413	4205
00040	GO_extensions-anatomy	SRIQ	58882	220	130376
00365	OBO-bfo	ALC	39	0	95
00511	OBO-kisao	ALCHIQ(D)	225	9	698
00512	OBO-lipid	ALCHIN	716	46	2349
00582	OBO-poro	SRQ	642	17	798

表 3 给出的十个本体表达能力有强有弱,其中本体 00031 和本体 00365 表达能力最弱,这几个推理工具都可以对其进行公理定位,而剩下的八个强表达能力的本体只有 EL2SAT 推理机无法处理.EL2SAT 推理机虽然在轻量级描述逻辑本体中表现出较高效率,但它只能用于弱表达描述逻辑本体.因此在下文的实验分析中不对它进行过多分析.

4.2 实验结果与分析

对于表 3 中给出的十个本体随机选择五十条查询公理,分别用 CBPin_Glass_Box、CBPin_Black_Box、Pellet 和 FaCT++计算这些公理的所有理由并记录平均求解时间.图 3 是 CBPin_Glass_Box 算法和 CBPin_Black_Box 算法对表 3 中的十个本体分别计算它的随机公理的所有理由的平均时间.

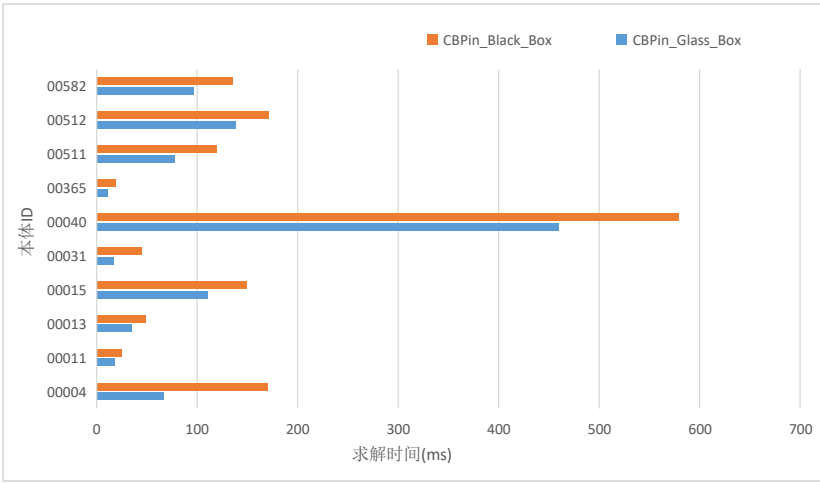

图 3 两个后继式公理定位算法对不同本体计算所有理由的时间对比

从图 3 可以看出白盒定位算法的效率要高于黑盒定位算法,这是因为白盒的定位算法修改判定算法的一次执行过程,利用定位规则建立子句的布尔公式标记与逻辑蕴含的所有极小公理集之间的联系.而黑盒算法则需要重复调用后继式判定算法,所以耗时高,效率较低.再用表现较好的白盒定位算法 CBPin_Glass_Box 与已有的 pellet 和 FaCT++推理机进行对比,如图 4 所示.

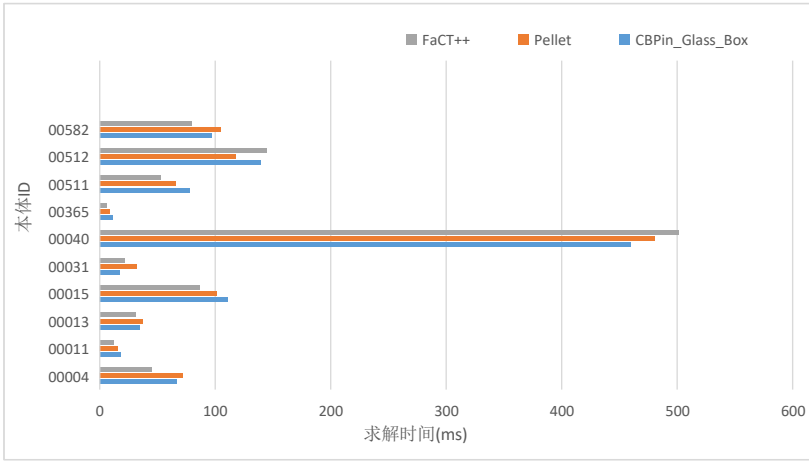

图 4 后继式定位算与其他推理工具对不同本体计算所有理由的时间对比

后继式白盒定位算法的优势在于它通过定位规则直接追踪推理的具体过程,从而使得判定给定逻辑蕴含能否成立时就能够得出该蕴含的所有极小公理集,不同于基于 Tableau 的定位方法还需要利用解释来追踪冲突再回溯求解.但是由于 pellet 和 FaCT++都是经过研究者们高度优化后所实现的求解工具,本文提出的白盒定位工具还只是对基于后继式的白盒定位算法的一个基本实现,还未做过多效率上的优化.所以从图 4 实验结果的整体情况来看,Pellet 推理机和 FaCT++推理机的求解速度是优于 CBPin_Glass_Box 的,但是在本体 00040 上

CBPin_Glass_Box 算法也展现出它的优势.同样的,在本体 00004、00013、00031、00582 中,白盒算法的求解时间虽然比不上 FaCT++但是都比 Pellet 要略快,因此可以看出这样的基于后继式的白盒定位算法具有一定的有效性,对它进行研究是有一定意义的.

5 总 结

本文给出了面向强表达描述逻辑本体的后继式白盒和黑盒两种公理定位方法.在白盒方法中,通过修正后继式判定规则得到公理定位规则,并引入定位公式的概念建立了子句的布尔公式标签与逻辑蕴含的所有极小公理集之间的对应关系,同时证明了该方法的正确性.考虑到当描述逻辑本体表达能力继续增强时,白盒定位算法需要针对构造算子重新设计,本文进一步给出了直接调用后继式判定推理机的黑盒公理定位方法,通过单一理由求解和所有理由求解过程完成公理定位.本文最后基于黑盒和白盒定位算法实现推理工具,并在多个表达能力不同的描述逻辑本体上进行实验,结果验证了后继式定位算法在强表达描述逻辑本体上具有一定的优势.

References:

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

[2] Suntisrivaraporn B, Baader F, Schulz S, Spackman K A. Replacing SEP-Triplets in SNOMED CT Using Tractable Description Logic Operators. In: Proc. of the 11th Conference on Artificial Intelligence in Medicine. Amsterdam: Springer, 2007. 287-291.

[3] Schlobach S, Cornet R. Non-standard Reasoning Services for the Debugging of Description Logic Terminologies. In: Proc. of the 18th International Joint Conference on Artificial Intelligence. Acapulco, 2003. 355-362.

[4] Kalyanpur A, Parsia B, Sirin E, Hendler J A. Debugging Unsatisfiable Classes in OWL Ontologies. Journal of Web Semantics, 2005,3(4):268-293.

[5] Kalyanpur A, Parsia B, Horridge M, Sirin E. Finding All Justifications of OWL DL Entailments. In: Proc. of the 16th International Semantic Web Conference. Heidelberg: Springer-Verlag, 2007. 267-280.

[6] 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). http://www.jos.org.cn/1000-9825/4735.htm

[7] Schlobach S, Huang Z, Cornet R, Van Harmelen F. Debugging Incoherent Terminologies. Journal of Automated Reasoning, 2007,39(3):317-349.

[8] Ji Q, Qi GL, Haase P. A Relevance-Directed Algorithm for Finding Justifications of DL Entailments. In: Proc. of the 4th Asian Conference on the Semantic Web. Shanghai: Springer, 2009. 306-320.

[9] 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

[10] Zhang Y, Ouyang DT, Cui XJ, Ye YX. Semi-Models Based Justifications Detection for OWL Ontologies. Chinese Journal of Computers, 2018,41(12):2710-1733 (in Chinese). https://kns.cnki.net/kcms/detail/11.1826.TP.20180512.2149.004.html

[11] Gao J, Ouyang DT, Ye YX. Exploring Duality on Ontology Debugging. Applied Intelligence, 2020,50(2):620-633.

[12] Baader F, Peñaloza R, Suntisrivaraporn B. Pinpointing in the Description Logic EL. In: Proc. of the 2007 International Workshop on Description Logics. Brixen-Bressanone, 2007. 52-67.

[13] Gao MY, Ye YX, Ouyang DT, Wang B. Finding Justifications by Approximating Core for Large-scale Ontologies. In: Proc. of the 28th International Joint Conference on Artificial Intelligence. Macao: ijcai.org. 2019. 6432-6433

[14] Ye YX, Cui XJ, Ouyang DT. Extracting a justification for OWL ontologies by critical axioms. Frontiers of Computer Science, 2020,14(4):144305.

[15] Baader F, Brandt S, Lutz C. Pushing the EL envelope. In: Proc. of the 19th International Joint Conference on Artificial Intelligence. Edinburgh, 2005. 364-369.

[16] Simančík F, Kazakov Y, Horrocks I. Consequence-based reasoning beyond Horn ontologies. In: Proc. of the 22nd International Joint Conference on Artificial Intelligence. Barcelona, 2011. 1093-1098.

[17] Simančík F, Motik B, Horrocks I. Consequence-based and fixed-parameter tractable reasoning in description logics. Artificial Intelligence, 2014,209:29-77.

[18] Cucala D T, Grau B C, Horrocks I. Pay-as-you-go consequence-based reasoning for the description logic SROIQ. Artificial Intelligence, 2021,298:103518.

[19] Ozaki A, Peñaloza R. Consequence-based axiom pinpointing. In: Proc. of 12th International Conference on Scalable Uncertainty Management. Milan, 2018. 181-195.

[20] Bate A, Motik B, Grau B C, et al. Extending consequence-based reasoning to SRIQ. In: Proc. of the 15th International Conference on the Principles of Knowledge Representation and Reasoning. Cape Town, 2016. 187-196.

[21] Reiter R. A Theory of Diagnosis from First Principles. Artificial Intelligence, 1987,32(1):57-95.

[22] Sirin E, Parsia B, Grau B C, et al. Pellet: A practical OWL-DL reasoner. Journal of Web Semantics, 2007,5(2):51-53.

[23] Tsarkov D, Horrocks I. FaCT++ description logic reasoner: System description. In: Porc. of the 3rd International Joint Conference on Automated Reasoning. Seattle: Springer-Verlag, 2006. 292-297.

[24] Sebastiani R, Vescovi M. Axiom pinpointing in large EL+ ontologies via SAT and SMT techniques. Technical Report DISI-15-010, DISI, University of Trento, Italy. DISI-15-010, 2015.

附中文参考文献:

[6] 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

[9] 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

[10] 张瑜,欧阳丹彤,崔仙姬,叶育鑫.基于半模型的 OWL 本体理由探求方法研究.计算机学报,2018,41(12):2720-2733. https://kns.cnki.net/kcms/detail/11.1826.TP.20180512.2149.004.html