软件学报 ISSN 1000-9825, CODEN RUXUEW
Journal of Software,2018,29(10):2995−3008 [doi: 10.13328/j.cnki.jos.005553]

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

基于互补概念和搜索图的 MUPS 求解优化方法*

崔仙姬[1], 何加亮[1], 张俊星[1], 高 健[2]

[1](大连民族大学 信息与通信工程学院,辽宁 大连 116600)

[2](大连海事大学 信息科学技术学院,辽宁 大连 116026)

通讯作者: 张俊星, E-mail: zhangjunxing@dlnu.edu.cn

摘 要: 本体调试是人工智能中非标准推理任务之一,对于本体工程具有很重要的意义.结合互补概念与基于术语集的搜索图提出极小不可满足子术语集求解的优化方法.首先,通过判断扩展的术语集是否包含互补概念,确定该子术语集是否需要进行概念可满足性检测,可以有效减少推理机的调用次数.接着,根据术语集扩展过程构造一个术语集搜索图,分别采用宽度优先搜索和深度优先搜索策略快速查找不可满足子术语集.该优化方法一方面减少了待测子术语集的规模,另一方面提高了查找不可满足子术语集对应的节点的查找效率.最后,实现了所给出的各类优化算法并与现有的黑盒优化算法进行了比较.实验结果表明,该方法从推理机调用次数和待测术语集规模方面均优于现有的 MUPS 求解方法,能够有效提高求解术语集 MUPS 的效率.

关键词: 描述逻辑;本体调试;MUPS;互补概念;选择函数

中图法分类号: TP18

中文引用格式: 崔仙姬,何加亮,张俊星,高健.基于互补概念和搜索图的 MUPS 求解优化方法.软件学报,2018,29(10): 2995−3008. http.//www.jos.org.cn/1000-9825/5553.htm

英文引用格式: Cui XJ, He JL, Zhang JX, Gao J. Optimization of MUPS calculation based on complementary concepts and search graph. Ruan Jian Xue Bao/Journal of Software, 2018,29(10):2995−3008 (in Chinese). http.//www.jos.org.cn/1000-9825/5553.htm

Optimization of MUPS Calculation Based on Complementary Concepts and Search Graph

CUI Xian-Ji[1,2], HE Jia-Liang[1], ZHANG Jun-Xing[1], GAO Jian[2]

[1](College of Information and Communication Engineering, Dalian Minzu University, Dalian 116600, China)

[2](College of Information Science and Technology, Dalian Maritime University, Dalian 116026, China)

Abstract: Ontology debugging is one of the non-standard reasoning tasks in artificial intelligence, and is important for ontology engineering. In this work, the complementary concepts and search graph are combined to optimize the calculation of minimal unsatisfiability preserving sub-TBox (MUPS) for the unsatisfiable concepts. Firstly, the necessity of checking the satisfaction of the concept for the expanded terminology is determined by whether it contains the complementary concepts to reduce the number of calling reasoners to some extent. Then, a search graph is constructed according to the terminology expanding process to quick search the node which is corresponding to the unsatisfiable sub-terminologies by breadth-first-search and depth-first-search strategies. This optimization reduces the number of axioms in terminologies to be checked, and it also improves the searching efficiency of the nodes corresponding to the unsatisfiable sub-terminologies. Finally, the optimized algorithms are realized and compared with existing black box algorithm. The

* 基金项目: 国家自然科学基金(61402070); 辽宁省自然科学基金(2015020023)

Foundation item: National Natural Science Foundation of China (61402070); Natural Science Foundation of Liaoning Province of China (2015020023)

本文由“本体工程与知识图谱”专题特约编辑欧阳丹彤教授推荐.

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 11:55:59, http://kns.cnki.net/kcms/detail/11.2560.TP.20180208.1155.009.html

experimental results show that the proposed method is superior to existing MUPS calculation in the calling number of reasoners and the number of axioms in the terminologies, which may effectively improve the efficiency of MUPS calculation.

Key words: description logics; ontology debugging; MUPS; complementary concepts; selection function

1 引 言

本体作为语义 Web 的核心,允许语义 Web 中的数据以语义明确的方式进行共享并重用领域知识,在语义 Web 应用中起着至关重要的作用[1].本体构建在语义 Web 相关的各个领域具有广泛的应用[2−4].而语义 Web 技术高度依赖于这些本体的质量与正确性.然而,由于本体构建过程中出现的建模错误及本体融合过程中不可避免的逻辑冲突,需要对本体进行一致性验证[5].保证本体质量涉及两种关键性策略,一种是开发更为复杂且成熟的本体建模工具,另一种是基于逻辑推理进行验证与修补,大多数的语义 Web 相关研究更关注于后者.随着表达能力强的本体语言(Web ontology language,简称 OWL)及其与描述逻辑之间的紧密关系,目前最先进的本体推理机即使在非常大的本体上也可以有效地监测逻辑一致性.

事实上,面对不一致的本体更为实际的问题是:“如果本体被检测为不一致,应该做什么”.有两种方式可以处理不一致本体.第 1 种是“忍受”不一致,应用非标准推理方法在存在不一致的情况下获得有意义的答案.另一种可行的方法是:当本体出现不一致时,解决或调试错误.本体调试作为一种非标准推理任务,可以识别并消除本体知识库中出现的逻辑错误[6].本文关注于本体术语集上的调试过程.

本体调试用于找出导致逻辑冲突的解释并进行修改.本体调试任务的主要思想在于求解极小不可满足子集.Schlobach 和 Cornet 首次将调试不一致术语集看作是非标准推理服务,并通过极小不可满足保持子术语集(minimal unsatisfiability-preserving sub-TBoxes,简称 MUPS)来解释术语集不一致的原因[7].这种方法主要针对概念关于术语集的可满足性问题,求解导致一个概念不可满足的解释.这是本体调试中的最基本问题.针对整体术语集的一致性问题,可以求解导致一个术语集不可满足的极小不一致保持子术语集(minimal incoherence preserving sub-TBox,简称 MIPS).该调试任务可以通过不可满足概念的 MUPS 集合覆盖的方式或直接调用术语集一致性验证方法实现[8].查找本体逻辑蕴涵的理由(justification)是 OWL 中的另一个关键推理服务,主要用于求解本体蕴涵一个公理集合过程中所涉及的极小公理集合.查找逻辑蕴涵的理由对于调试不可满足概念和冲突也是必要的.Horridge 提出了求解术语集所有理由的几种算法[9].另一方面,针对较为复杂的公理集合,还可以在找出极小不可满足子术语集之后,将解释进一步定位到公理的内部,从而简化导致冲突的解释[10−12].本体诊断任务作为本体调试的进一步工作,将基于模型诊断的思想引入到本体调试过程中[13].Schlobach 等人通过查找极小不可满足子术语集的极小碰集的方式构造诊断系统[14,15],并根据这些诊断系统研究删除冲突公理的调试策略.进一步地,Janach 等人针对当今的多核计算机提出了并行诊断算法,在碰集树算法中同时求全部极小不可满足子集[16,17].

然而,本体调试任务中最基本的问题还是求解一个概念的 MUPS 的方法.求解本体中不可满足概念的 MUPS 的现有工作分为黑盒技术与白盒技术.白盒技术为表达能力强的描述逻辑给出一个基于表演算的可判定过程.这个过程通过对表演算加入跟踪技术实现,需要修改推理机内部[18].由于不同本体对应的描述逻辑语言不同,需要针对不同的描述逻辑语言给出相应的修改机制.而黑盒技术使用推理机作为子程序,通过调用现有推理机进行,其实现过程无需修改推理机内部[19].黑盒技术的关注点在于如何减少推理机的调用次数及减少待测术语集中的公理个数,而不关注具体的描述逻辑语言,应用更为广泛.黑盒方法一般会采用结构相关性确定待测子术语集从而减少待测本体的规模,但因未关注直接导致概念不可满足的原因也会将一些不相关的公理加入到待测术语集中.大多数情况下,原子概念与原子概念否定的合取式才是导致一个概念不可满足的直接原因.

本文基于这一点,在传统黑盒技术的基础上,结合互补概念与术语集搜索图优化不可满足子术语集的扩展过程,主要工作如下.

1) 在采用概念相关选择函数的基础上,通过构造互补概念术语集确定是否进行概念可满足性检测,一定程度上减少了推理机的调用次数;

2) 采用概念相关选择函数扩展不可满足子术语集的过程中,构造术语集搜索图,分别采用宽度优先搜索和深度优先搜索方法查找不可满足子术语集对应的节点,一方面减少了不可满足子术语集的规模,另一方面提高了搜索不可满足子术语集对应节点的查找效率;

3) 应用不同数据集的实验结果表明,本文提出的不可满足子术语集扩展优化方法可以有效地提高 MUPS 求解的效率.

2 基于黑盒技术的 MUPS 求解

本节主要介绍 MUPS 的概念及使用黑盒技术求解术语集 T 关于概念 C 的 MUPS 的方法.由于黑盒技术不受具体描述逻辑语言的限制,对于任意的描述逻辑语言都适用.现在主流的本体语言 OWL 2 对应的逻辑基础为 DL SROIQ,且现在所有的描述逻辑语言都是 DL SROIQ 的子集.

下面介绍 DL SROIQ 中的语法定义.SROIQ 概念描述通过如下语法规则递归定义(因角色描述不会直接导致概念不可满足,本文不考虑角色描述):

$$C,D \leftarrow \top \mid \bot \mid A \mid \neg C \mid C \sqcap D \mid C \sqcup D \mid \exists R.C \mid \forall R.C \mid \geqslant nR \mid \leqslant nR \mid \geqslant nR.C \mid \leqslant nR.C \mid \exists R.\mathrm{Self} \mid \{a_1,...,a_n\},$$

其中,$\top$表示顶层概念,$\bot$表示底层概念,A 表示原子概念名,R_a 表示原子角色名,C、D 是概念,a_i 是个体($1\leqslant i\leqslant n$),n 表示非负整数.以 SROIQ 表示的描述逻辑知识库中的术语集是形如 $C\sqsubseteq D$ 的一般概念包含公理的有限子集,其中 C 和 D 是上面描述的 DL SROIQ 概念,以概念主合取范式的形式表示.关于 DL SROIQ 知识库详细的语法与语义描述可参见文献[20].

下面给出一个概念 C 关于术语集 T 的 MUPS 的概念.

定义 1(MUPS)[8]. 称术语集 $T'\subseteq T$ 是 T 关于概念 C 的一个极小不可满足保持子术语集(minimal unsatisfiability preserving sub-TBox,简称 MUPS),如果 C 关于 T'不可满足且对于任意的子术语集 $T''\subset T'$,C 关于 T''都可满足.将 T 关于 C 的所有 MUPS 记作 $mups(T,C)$.

例 1: $T_1=\{ax_1,ax_2,ax_3,ax_4,ax_5,ax_6\}$是如下所示的一个不一致术语集,其中,$A,B,C,A_1,\ldots,A_6$是概念,$r$ 和 s 是角色.概念可满足性检测结果显示$\{A_1,A_2,A_3,A_6\}$是不可满足概念的集合.

ax_1: $A_1\sqsubseteq A_2\sqcap A_3$ ax_2: $A_2\sqsubseteq A\sqcap A_4$ ax_3: $A_3\sqsubseteq A_4\sqcap A_5$

ax_4: $A_4\sqsubseteq \neg A\sqcap \exists s.\neg B$ ax_5: $A_5\sqsubseteq \forall s.B\sqcap C$ ax_6: $A_6\sqsubseteq A_1\sqcap \exists r.(A_3\sqcap \neg C\sqcap A_5)$

根据MUPS的定义,可以得出 $mups(T_1,A_1)$:$\{\{ax_1,ax_2,ax_4\},\{ax_1,ax_3,ax_4,ax_5\}\}$,$mups(T_1,A_2)$:$\{\{ax_2,ax_4\}\}$,$mups(T_1,A_3)$:$\{\{ax_3,ax_4,ax_5\}\}$,$mups(T_1,A_6)$:$\{\{ax_1,ax_2,ax_4,ax_6\},\{ax_5,ax_6\}\}$.

下面介绍使用黑盒技术求解一个术语集关于不可满足概念的 MUPS 的算法,输入一个不可满足概念 C 和术语集 T,算法返回 T 关于 C 的一个 MUPS.算法分为两个部分:(1) 不可满足子术语集扩展阶段(第 1 行~第 4 行),算法从一个空的候选术语集 T'开始,每次循环将 T 中的公理添加到 T'中,直到概念 C 关于 T'不可满足,通过不断添加公理扩展 T'从而获得关于 C 不可满足的子术语集;(2) 不可满足子术语集收缩阶段(第 5 行~第 8 行),因无法保证扩展阶段得到的子术语集是极小的,需要在此基础上进行公理的删减.算法在每次循环中都从 T'中删除一个公理并检测 C 关于 T'是否变为可满足,若是,将这个公理重新加入到 T'中,直到 T'中所有的公理都被检测完毕,得到一个 MUPS.具体算法如算法 1 所示.

算法 1. 求解 MUPS 的黑盒算法.

输入:不可满足概念 C,术语集 T;

输出:一个 MUPS T'.

1. $T'\leftarrow\varnothing$
2. **while** C is satisfiable w.r.t. T'
3. select a set of axioms $S\subseteq T/T'$
4. $T'\leftarrow T'\cup S$
5. **for** each axiom $Ax\in T'$

6. $T'\leftarrow T'/\{Ax\}$
7. **if** C is satisfiable w.r.t.T'
8. $T'\leftarrow T'\cup\{Ax\}$

对于给定的一个术语集 T 和不可满足概念 C,T 关于 C 的 MUPS 不一定唯一,需要求出其所有的 MUPS.给出术语集 T 关于概念 C 的一个 MUPS,可以通过 Reiter 碰集树的方法求解所有的 MUPS[14].其具体思想如下:从算法 1 得到的一个 MUPS 中随机选择一条公理 ax_i,将 ax_i 从术语集 T 中删除,以新的术语集及概念 C 为输入求一个新的 MUPS,以此类推,直到求出术语集 T 关于概念 C 的所有 MUPS.这个过程中可以使用提前剪枝和节点重用方法适当减少调用算法 1 的次数,具体优化方法详见参考文献[8].

下面介绍求 T_1 关于概念 A_1 的所有 MUPS 的过程.假设在算法 1 的扩展阶段依次加入 ax_1,ax_2,ax_3,ax_4,得到了一个不可满足子术语集 $T_1'=\{ax_1,ax_2,ax_3,ax_4\}$,在收缩阶段对于 T_1' 中的每个元素需要判断其是否可以从不可满足子术语集中删除.通过判断删除该公理之后得到的术语集是否满足概念 A_1 来判定.选择删除 ax_3 时,子术语集依然关于概念 A_1 不可满足,可以删除.接下来选择 ax_4,删除 ax_4 得到的术语集关于概念 A_1 可满足,因此 ax_4 不能删除,将其放回 T_1' 中,得到 T_1 中关于概念 A_1 的一个 MUPS=$\{ax_1,ax_2,ax_4\}$.进一步使用碰集树方法求 T_1 关于 A_1 的所有 MUPS.假设从 T_1 中选择 ax_1,得到的子术语集关于概念 A_1 可满足,找不到新的 MUPS.接下来选择 ax_2,得到$\{ax_1,ax_3,ax_4,ax_5\}$,将该术语集与 A_1 作为输入运行算法 1,得到关于概念 A_1 的一个新的 MUPS=$\{ax_1,ax_3,ax_4,ax_5\}$.

在使用黑盒技术求解 MUPS 的过程中,可以通过直接调用推理机的方式判断概念 C 关于 T 的子术语集是否可满足.这种方式完备且容易实现,但因其外部推理机调用次数过多,在实际应用中并不适用.分析算法 1 可知,求解 MUPS 的过程中推理机调用主要应用在两个方面:不可满足子术语集扩展阶段每次选择的公理添加到待测术语集都需要调用推理机判断可满足性;不可满足子术语集收缩过程中对于不可满足子术语集扩展阶段得到的子术语集中每个元素都需要调用推理机判断该元素是否可以从待测术语集中删除.收缩阶段虽然能够使用滑动窗口等方法来减少推理机的次数,但最终确定 MUPS 时仍需要一个个公理进行删减.因此,若在不可满足子术语集的扩展阶段得到一个元素相对较少的不可满足子术语集,则可以大大减少推理机的调用次数,一方面减少公理个数,另一方面可以尽快找到不可满足子术语集.本文主要关注于求解一种 MUPS 时的优化方法,这种优化方法对求解所有 MUPS 也能有所优化.

3 结合互补概念与搜索图的术语集扩展优化

本节在概念相关选择函数的基础上,结合互补概念优化计算术语集 T 关于概念 C 的一个 MUPS,主要考虑不可满足子术语集扩展阶段如何使用选择函数优化其扩展.

3.1 基于互补概念的术语集扩展

不可满足子术语集扩展过程的关键在于如何选择合适的公理加入到不可满足子术语集 T'中(对应于算法 1 的第 3 行).不同的公理选择方式会影响推理机的调用次数.例如,在例 1 中求 $mups(T_1,A_3)$时,在选择公理 ax_3 的基础上,选择 ax_4 和 ax_5 可直接导致不可满足,而若先选择 ax_1 和 ax_2,则需要再加入 ax_4 和 ax_5 才能导致概念 A_3 不可满足.一般使用选择函数来控制待测子术语集 T 中的元素.选择函数以启发式添加相关公理的方式进行术语集的扩展.给出一个术语集 T 和一条公理 ax,选择函数 s 将返回 T 的子集的一个线性有序集合,其形式化定义如下.

定义 2(选择函数)[8]. 一个本体语言 L,选择函数 s 是一个映射 $s:P(L)\times L\times N\rightarrow P(L)$使得 $s(T,\varphi,k)\subseteq T$.

在求解术语集 T 关于概念 C 的 MUPS 过程中,一种比较简单且高效的选择函数是基于概念相关性的选择函数,其基本思想为:一个公理 ax 与一个概念名 A 相关当且仅当 A 出现在 ax 的左侧.使用 $V_c(ax)(V_c(C))$表示公理 ax(概念 C)中出现的概念名的集合,概念相关和概念相关选择函数的定义如下.

定义 3(概念相关)[8]. 称一个公理 ax 与概念(或公理)ϕ是概念相关的,当且仅当

(1) $V_c(C_1)\cap V_c(\phi)\neq\varnothing$,若 ax 形如 $C_1\sqsubseteq C_2$;

(2) $V_c(C_1)\cap V_c(\phi)\neq\varnothing$或 $V_c(C_2)\cap V_c(\phi)\neq\varnothing$,若 ax 形如 $C_1\equiv C_2$ 或 $disjoint(C_1,C_2)$.

定义 4(概念相关选择函数)[8]. T 是一个术语集,C 是一个概念,关于 T 和 C 的概念相关选择函数定义如下:

(1) $s(T,C,0)=\varnothing$;

(2) $s(T,C,1)=\{ax|ax\in T$ 且 ax 与 C 概念相关$\}$;

(3) $s(T,C,k)=\{ax|ax\in T$ 且 ax 与 $s(T,C,k-1)$中的一个元素概念相关$\}$,其中,$k>1$.

概念相关选择函数针对术语集 T 和不可满足概念 C,可递归生成 T 的一个子集 $s(T,C,k)$.该子术语集可作为算法 1 第 3 行中选择的公理集合进行概念 C 的可满足性检测,直到 C 关于 $s(T,C,k)$不可满足.用 $cus(T_1,A_i)$表示使用概念相关选择函数时,输入术语集 T_1 和概念 $A_i(1\leqslant i\leqslant 6)$返回的结果.可以得出,$cus(T_1,A_1)$:$\{ax_1,ax_2,ax_3,ax_4,ax_5\}$,$cus(T_1,A_2)$:$\{ax_2,ax_4\}$,$cus(T_1,A_3)$:$\{ax_3,ax_4,ax_5\}$,$cus(T_1,A_6)$:$\{ax_1,ax_3,ax_5,ax_6\}$.可以看出,使用概念相关选择函数可以减少部分不相关公理加入到 T'中.

在不可满足子术语集扩展阶段,黑盒技术求解 MUPS 的算法中使用概念相关选择函数需要进行 k 次概念可满足性检测即可得到不可满足子术语集.然而,不可满足子术语集扩展过程中并不需要针对每一次使用选择函数获得的子术语集都判断其是否满足概念 C.事实上,概念可满足性并不依赖于选择函数的使用次数 k,而是依赖于导致该概念不可满足的一些限定,如互补概念、数量限定、枚举类等[5].大多数情况下的概念不可满足通常都是以互补概念合取式的形式产生冲突.基于这一点,在概念相关选择函数的基础上,我们考虑基于互补概念的术语集扩展方法,下面给出互补相关概念及互补相关公理的定义.

定义 5(互补相关概念). C_1,C_2 是概念,称概念 C_2 是 C_1 关于原子概念 A 的互补相关概念,若存在一个原子概念名 A 使得 A 出现在 C_1 的概念描述而$\neg A$ 出现在 C_2 的概念描述.

定义 6(互补相关公理). 称公理α和β是互补相关公理,若α中存在一个概念 C_1,β中存在一个概念 C_2,使得 C_1 是 C_2 关于一个原子概念的互补相关概念.称一个术语集 T 为互补概念术语集,若存在两个公理α,β,使得$\{\alpha,\beta\}\subseteq T$ 且这两个公理为互补相关公理.

以例 1 中的术语集 T_1 为例,概念 $\neg A\sqcap\exists s.\neg B$ 是概念 $A\sqcap A_4$ 关于原子概念 A 的互补相关概念,公理 ax_2: $A_2\sqsubseteq A\sqcap A_4$ 和 ax_4: $A_4\sqsubseteq\neg A\sqcap\exists s.\neg B$ 是互补相关公理.术语集$\{ax_2,ax_4\}$是互补概念术语集.

基于互补概念的不可满足子术语集扩展算法基本思想如下:在不可满足子术语集扩展阶段,针对每一次使用选择函数获得的术语集不直接进行概念可满足性检测,而是先判断该术语集是否为互补概念术语集.只有在选择函数生成的术语集包含互补概念时才进行概念可满足性检测,也就是调用推理机.这种情况下,若得到的术语集关于概念 C 不可满足,则只需调用一次推理机即可得到不可满足子术语集 T',否则,进一步使用选择函数扩展术语集并检测概念可满足性.具体算法如下所示,其中,$Compl(T)$用于判断术语集 T 是否为互补概念术语集,若是,返回 true,否则,返回 false.

算法 2. 基于互补概念的不可满足子术语集扩展算法.

输入:不可满足概念 C,术语集 T;

输出:一个不可满足子术语集 T'.

```
1.  k←0,T'←∅
2.  while Compl(T') is false
3.      k←k+1
4.      T'←s(T,C,k)
5.  while C is satisfiable w.r.t. s(T,C,k)
6.      k←k+1
7.  T'←s(T,C,k)
8.  return T'
```

算法 2 包括两个循环,第 1 个循环(第 2 行~第 4 行)中算法从一个空的候选术语集 T'开始,每次循环根据选择函数将 T 中的概念相关公理集合添加到 T'中,直到扩展的术语集为互补概念术语集.第 2 个循环(第 5 行~第 7 行)中,同样根据选择函数扩展 T'直到概念 C 关于 T'不可满足.若第 1 个循环中找出的 T'为互补概念术语集,该公理集很可能关于概念 C 不可满足,需要进行概念可满足性检测.若概念 C 关于该术语集不可满足,则术语集扩展

过程终止,返回 T',否则,继续使用选择函数扩展待测术语集.以例 1 中的术语集 T_1 和概念 A_1 为例,可以得出,$s(T_1,A_1,1)=\{ax_1\}$,$s(T_1,A_1,2)=\{ax_1,ax_2,ax_3\}$,$s(T_1,A_1,3)=\{ax_1,ax_2,ax_3,ax_4,ax_5\}$.因为只有待测术语集为互补概念术语集时才进行概念可满足检测,这里仅需要调用 1 次.而未引入互补概念的选择函数扩展策略,需要在每使用一次选择函数扩展时调用一次推理机,共调用 3 次才找到一个不可满足子术语集.然而,并不是所有情况下上述算法都只会调用推理机 1 次.下面考虑两种需要调用多次推理机得到不可满足子术语集的情况.

(1) 本文中的优化方法在出现互补概念术语集时进行概念可满足性验证,也就是只要出现原子概念及原子概念的否定就进行概念可满足性的验证.但事实上,只有原子概念及原子概念的否定的合取式才会导致概念的不可满足.因此,第 5 行进行概念可满足性验证时可能会出现可满足的情况.例如,将例 1 中公理 ax_6 中的合取算子改为析取算子,有: $A_6 \sqsubseteq A_1 \sqcap \exists r.(A_3 \sqcap \neg C \sqcap A_5)$, 则使用概念相关选择函数得到的子术语集是 $s(T_1,A_6,1)=\{ax_5,ax_6\}$,$s(T_1,A_6,2)=\{ax_1,ax_3,ax_5,ax_6\}$,可以看出,这个术语集关于概念 A_6 是可满足的.这时需要进一步扩展术语集,得到 $s(T_1,A_6,3)=\{ax_1,ax_2,ax_3,ax_4,ax_5,ax_6\}$.这种情况下,尽管未能直接获得不可满足子术语集,但因之前使用概念相关选择函数进行扩展时,对应的术语集中无关于原子概念的互补相关概念不会导致冲突,不需要进行概念可满足性检测,可以适当减少推理机的调用次数.

(2) 本文中的优化方法针对的是大多数情况下出现的互补概念合取式导致的不可满足问题.事实上,数量限定也可能导致复杂概念不可满足,在这种情况下算法 2 并未能优化扩展过程.例如,术语集 $T_2=\{C_1 \sqsubseteq \leqslant 2R.D, C_1 \sqsubseteq C_2, C_2 \sqsubseteq \geqslant 3R.D\}$, 可以看出,概念 C_2 关于术语集 T_2 不可满足.由于术语集 T_2 不包括关于原子概念的互补相关概念,算法在不可满足子术语集扩展阶段会得到整个 T_2,并未减少不相关公理.基于数量限定的冲突导致的原因在于同时出现 $\leqslant n$ 量词和 $\geqslant m$ 量词($m>n$).针对这种情况,可以在不可满足子术语集扩展阶段判断所扩展的公理是否包括≤量词,对于包含≤量词的公理的扩展不进行优化,每一次使用概念相关选择函数时都需要进行概念可满足性检测.

3.2 基于宽度优先搜索的公理集选择

上述基于互补概念的术语集扩展方法仅在找到互补概念术语集的情况下才会进行概念的可满足性检测,其优势在于,可以一定程度地减少选择函数调用推理机的次数.事实上,互补概念可以进一步应用于选择函数扩展的术语集元素的选择过程中.也就是针对每一次使用选择函数生成的子术语集,只选择可能包含互补概念的公理进行术语集扩展,从而减少选择函数涉及的公理个数.对于不可满足概念 C 和术语集 T,选择函数应用到术语集 T,会不断地将与 C 概念相关的公理添加到待测公理集中.在这一过程中,可以同时构造一个术语集搜索图.

定义 7(术语集搜索图). 给出一个关于概念 C 不可满足的术语集 T,若一个有向无环图 $G(V,E,r)$满足下面的条件:(1) 包含唯一汇节点 r,对应的公理形如 $C \sqsubseteq \top$;(2) $\forall v \in V \backslash r$ 是其父节点 $p \in V$ 的第 m 个子节点,假设节点 p 对应的术语集为 $T_p=\{ax_1,\ldots,ax_n\}$,那么节点 v 对应的术语集为 $T_m=T_p \cup \{ax_m \in V_m\}$,其中,$V_m=\{ax|ax$ 与 ax_i 概念相关的公理,$ax_i \in T_p\}$;而边 $e_{pv} \in E$ 表示从父节点 p 指向子节点 v,则称 $G(V,E,r)$为术语集 T 关于概念 C 的搜索图.

图 1 表示例 1 中 T_1 关于概念 A_1 的搜索图.图中每个节点上的标签即为对应的子术语集.

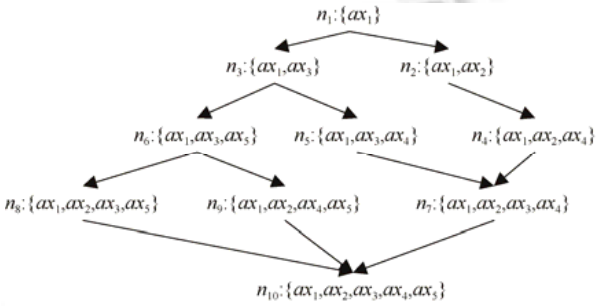

Fig.1 Search graph for terminologies T_1 w.r.t.A_1

图 1 术语集 T_1 关于概念 A_1 的搜索图

为了方便检测节点对应的术语集关于概念的可满足性,将术语集 T 关于概念 C 的搜索图 $G(V,E,r)$的节点集合 V 分为 3 类:live 节点、dead 节点和 pending 节点.其定义分别为:live 节点表示其对应的术语集中并未包含互补相关公理,即该术语集关于概念 C 是可满足的,并且搜索过程正运行于当前节点的分支上,该类节点可以继续向下扩展;dead 节点表示其对应的公理集合为互补概念术语集,这类节点不能继续向下扩展,需要进行概念可满足性检测;pending 节点表示当前图中未被搜索过的节点.搜索图中 3 类节点之间的转换关系如下:当一个 pending 节点所对应的术语集为互补概念术语集时,该节点就转化为 dead 节点;当算法的搜索过程到达一个 pending 节点,并且其对应的术语集不包括互补相关公理时,就将该节点转化为 live 节点.

假设一个关于概念 C 不可满足的术语集 T,基于宽度优先搜索的子术语集扩展算法为术语集 T 构造一个关于 C 的搜索图 $G(V,E,r)$,形如 $C\sqsubseteq\top$的公理对应的节点为汇节点 r,对于搜索图 G 中的每个节点 $v\in V$,v 都对应术语集 T 的一个子集.基于宽度优先搜索的子术语集扩展算法主要思想如下:通过为输入的术语集 T 构造一个搜索图 $G(V,E,s)$,采用宽度优先搜索查找图中的 dead 节点.因为 dead 节点包含互补相关公理,需要判断其对应的术语集 T'是否满足概念 C.若 T'不满足 C,T'为不可满足子术语集,返回 T',否则,需要回溯到父节点,将使用选择函数生成的术语集作为待测术语集进行概念 C 可满足性检测.具体算法如下所示.

算法 3. 基于宽度优先搜索的子术语集扩展算法.

输入:不可满足概念 C,术语集 T;

输出:一个不可满足子术语集 T'.

```
1.  k←1,T'←∅,W=∅
2.  while W≠T do
3.    S←s(T,C,k)
4.    Σ←s(T,C,k+1)\s(T,C,k)
5.    if W=∅
6.      W←{S}
7.    for each axiom a_x∈Σ do
8.      for S'∈W do
9.        if Compl(S'∪{ax}) is false and S'∪{ax}∉W
10.         W←W{S'∪{ax}}
11.       if Compl(S'∪{ax}) is true
12.         if C is not satisfiable w.r.t. S'∪{ax}
13.           T'←S'∪{ax}
14.         else
15.           while C is satisfiable w.r.t. s(T,C,k+1)
16.                 k←k+1
17.           T'←s(T,C,k)
18.         return T'
19.   k←k+1
20. return T
```

算法 3 通过 3 层循环构造术语集 T 关于概念 C 的搜索图 $G(V,E,r)$,并以宽度优先搜索策略查找 $G(V,E,r)$中的 dead 节点.k 从 1 开始递归地产生与不可满足概念 C 直接或间接相关的公理集合,加入到 $s(T,C,k)$中.S 为第 k 次使用选择函数获得的公理集合,而Σ为第 k+1 次使用选择函数获得的新的公理集合.根据这两个变量生成搜索图 $G(V,E,r)$中节点对应的公理集合 S':初始化 S'为 $s(T,C,k)$,对于Σ中的任何一个公理 ax,如果 $S'\cup\{ax\}$是互补概念术语集,则表明该术语集对应的节点为 dead 节点,判断该术语集是否满足概念 C.若 $S'\cup\{ax\}$不满足 C,则 $S'\cup\{ax\}$作为结果返回,否则,将当前的 $s(T,C,k)$作为子术语集进行概念 C 的可满足性检测,这部分与算法 2 一致;S'

$\cup\{ax\}$中未包含互补相关公理,说明该节点为 live 节点,需要进一步扩展,将 ax 加入到当前节点所在的术语集中作为下一轮循环中的父节点.算法 3 使用了如下启发式来优化搜索过程:(1) 第 8 行 W 中选择术语集 S'的过程中,以公理元素个数对 W 中的公理集合进行排序,从元素个数最少的术语集开始选择;(2) 第 7 行选择公理 ax 的过程中,对Σ中的公理进行排序,将包含原子概念否定的公理排在其他公理之前.

图 2 表示 T_1 中关于概念 A_1 的宽度优先搜索子术语集扩展对应的搜索图.从 n_1:$\{ax_1\}$开始递归地添加与该公理集概念相关的公理集合,找到公理集$\{ax_2,ax_3\}$,生成节点 n_2:$\{ax_1,ax_2\}$和 n_3:$\{ax_1,ax_3\}$,因这两个节点对应的术语集均不包含互补相关公理,需要进一步扩展.找到公理集$\{ax_3,ax_4,ax_5\}$,对公理集进行排序.得到公理序列 ax_4-ax_5-ax_3,将其顺序应用到节点 n_2 和 n_3,生成节点 n_4:$\{ax_1,ax_2,ax_4\}$和 n_5:$\{ax_1,ax_3,ax_4\}$.n_4 为互补概念术语集,判断 n_4 对应的术语集关于概念 A_1 的可满足性,因 n_4 对应的术语集$\{ax_1,ax_2,ax_4\}$关于 A_1 不可满足,该术语集即为不可满足子术语集 T'.

可以看出,使用如上启发式方式进行搜索一方面可以提高查找 dead 节点的速度,另一方面使得找到的术语集是极小的.换句话说,若算法 3 第 13 行得到的 dead 节点对应的子术语集 T'关于概念 C 不可满足,则 T'是一个 MUPS 且是所有 MUPS 中元素最少的不可满足子术语集.显然,如果宽度优先搜索方法针对搜索图的每一层遍历术语集 T 中所有公理时,该结论成立.下面证明基于概念相关选择函数和互补概念进行公理选择,该结论依然成立.由定义 4 可知,通过概念相关选择函数仅扩展与待扩展元素概念相关的公理集合,对于不相关的公理不予考虑.这是因为概念不相关公理无法直接应用到概念可满足性检测过程中.例如,对于 T_1 中考虑概念 A_1 的可满足性时,第 1 次扩展生成$\{ax_1\}$,第 2 次扩展时仅需将 ax_2 和 ax_3 加入到术语集中,这是因为只有 A_2 和 A_3 出现在 ax_1: $A_1\sqsubseteq A_2\sqcap A_3$ 中.而像 ax_6: $A_6\sqsubseteq A_1\sqcap\exists r.(A_3\sqcap\neg C\sqcap A_5)$ 与 ax_1 不直接相关不需要扩展.因此,使用概念相关选择函数扩展术语集搜索图中的节点只是将不相关的公理从待测术语集中删除.这样能够在保证正确性的前提下减少待测公理个数.另一方面,算法 2 中的启发式优化策略只是对于同一层(属于同一个 $s(T,C,k)$中的元素)扩展的公理进行排序,每次只扩展 $s(T,C,k)$中的一个元素,对应的待测子术语集规模会保持一致.因此,在宽度优先搜索策略下,得到的不可满足子术语集中的元素是最少的,同时也是极小的,亦即它是关于概念 C 的一个 MUPS.

若将例 1 中公理 ax_1 改为如下形式: $A_1\sqsubseteq A_2\sqcup A_3$, 则 n_4 对应的节点关于概念 A_1 是可满足的,需要将公理序列中其余公理也加入到待测术语集中,得到节点 n_6:$\{ax_1,ax_2,ax_3,ax_4,ax_5\}$,判断 n_6 对应的术语集关于 A_1 的可满足性,若 n_6 对应的术语集关于 A_1 仍是可满足的,需要进一步利用概念选择函数扩展术语集以及其关于概念 A_1 的可满足性,直到待测术语集关于 A_1 不可满足或达到 T,否则,n_6 对应的术语集即为不可满足子术语集 T',如图 2 中虚框内所示术语集$\{ax_1,ax_2,ax_3,ax_4,ax_5\}$.

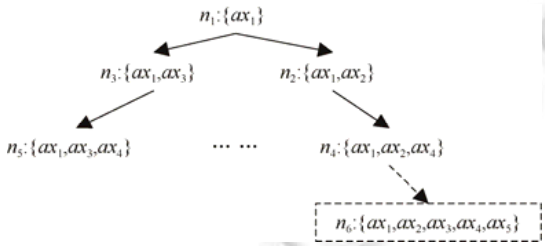

Fig.2 Terminology expansion process based on breadth-first-search w.r.t. concept A_1

图 2 关于概念 A_1 的宽度优先搜索的子术语集扩展过程

3.3 基于深度优先搜索的公理集快速选择

上一节算法的优点在于若 dead 节点对应的术语集关于概念 C 不可满足,则一定能找出 T 中关于 C 的一个 MUPS.但是由于宽度优先搜索需要考虑的节点个数较多,搜索 dead 节点的速度较慢.下面给出一种基于深度优先搜索策略的术语集快速选择算法.

假设一个关于概念 C 不可满足的术语集为 T,基于深度优先搜索的公理集子术语集快速扩展算法同样为术语集 T 构造一个关于 C 的搜索图 $G(V,E,r)$,汇节点 r 对应 $C\sqsubseteq D$ 形式的公理,对于搜索图 G 中的每个节点 $v\in V$,v

都对应术语集 T 的一个子集.算法主要思想如下:通过为输入的术语集 T 构造一个搜索图 $G(V,E,s)$,采用深度优先搜索策略查找图中的dead节点.因为dead节点对应的术语集包含互补相关公理,需要判断其对应的不可满足子术语集 T'是否满足概念 C.若 T'不满足 C,则返回不可满足子术语集为 T',否则,需要回溯到父节点,将当前使用选择函数生成的术语集作为待测术语集进行概念 C 可满足性检测.具体算法如下所示.

算法 4. 基于深度优先搜索的子术语集快速扩展算法.

输入:不可满足概念 C,术语集 T;

输出:一个不可满足子术语集 T'.

```
1.  k←1,T'←∅,W=∅
2.  while W≠T do
3.    S←s(T,C,k)
4.    Σ←s(T,C,k+1)\s(T,C,k)
5.    if W=∅
6.      W←S
7.    select an axiom ax∈Σ
8.    while ax∈W
9.      Σ←Σ\(ax)
10.   if Compl(W∪{ax}) is false
11.     W←W∪{ax}
12.   if Compl(S'∪{ax}) is true
13.     if C is not satisfiable w.r.t. W∪{ax}
14.       T'←W∪{ax}
15.     else
16.       while C is satisfiable w.r.t.s(T,C,k+1)
17.         k←k+1
18.         T'←s(T,C,k)
19.     return T'
20.   k←k+1
21. return T
```

与上一小节介绍的基于宽度优先搜索策略查找 dead 节点的方法相比,算法通过单层循环即可构造术语集 T 关于概念 C 的搜索图 $G(V,E,r)$,从而 dead 节点.同样地,k 从 1 开始递归产生待测术语集加入到 $s(T,C,k)$中.S 为第 k 次使用选择函数获得的公理集合,而Σ为第 k+1 次使用选择函数获得的新的公理集合.初始化待测术语集 W 为 $s(T,C,k)$,对于Σ中的任何一个公理 ax,如果 $W\cup\{ax\}$是互补概念术语集,那么表明该术语集对应的节点为 dead 节点,判断该术语集是否满足概念 C.若 $W\cup\{ax\}$不满足 C,则 $W\cup\{ax\}$作为结果返回,否则,将当前的 $s(T,C,k)$作为子术语集进行概念 C 可满足性检测,这部分与算法 2 一致;$W\cup\{ax\}$中未包含互补相关公理,说明该节点为 live 节点,需要进一步扩展,将 ax 作为元素加入到当前节点所在的术语集中作为下一轮循环中的父节点.

算法使用了如下启发式来优化搜索过程:(1) 第 7 行选择公理 ax 的过程中,对Σ中的公理进行排序,按“包含原子概念否定的公理-与包含原子概念否定的公理概念相关的公理-其他公理”顺序进行排序,可以提高查找 dead 节点的速度;(2) 算法每一次循环时,将本次迭代中用到的公理从待测术语集中删除,减少下次循环时的待测公理个数;(3) 将搜索过的 dead 节点对应的子公式进行标记,当搜索到的 pending 节点与之前已标记的子公式相同时,直接将 pending 节点转换为 dead 节点.

采用基于深度优先搜索方法计算 T_1 关于概念 A_1 的不可满足子术语集对应的搜索图仅生成 3 个节点.从 n_1:$\{ax_1\}$开始递归地添加与该公理集概念相关的公理集$\{ax_2,ax_3\}$.因 ax_2 与 ax_4 概念相关且 ax_4 含有原子概念的

否定,优先选择 ax_2.生成节点 n_2:{ax_1,ax_2},因 n_2 对应的术语集不包含互补相关公理,需要进一步扩展.找到公理集 ax_4,生成节点 n_3:{ax_1,ax_2,ax_4}.n_3 为互补概念术语集,判断 n_3 对应的术语集关于概念 A_1 的可满足性,因 n_3 对应的术语集{ax_1,ax_2,ax_4}关于 A_1 不可满足,该术语集即为不可满足子术语集 T'.可以看出,使用深度优先搜索方法可以减少生成节点的个数.针对 dead 节点对应的术语集关于概念 A_1 可满足的情况,处理方式与算法 2 一致,这里不再赘述.

4 实验结果

本文实验在如下环境中进行:Intel(R) core(TM) i7-6700 3.4GHz CPU;8.0GB RAM,Windows 7 OS,而本体解析与概念的可满足性检测均通过调用 OWLAPI(http://owlapi.sourceforge.net/)实现.

4.1 实例评估

本文实验数据采用的是本体调试中的标准测试集.表 1 给出了这些术语集的相关信息,包括本体的名字、表达能力,本体中公理、类、属性、实例(分别对应描述逻辑中的概念、角色、个体)的个数(*C*/*P*/*I*),不可满足概念的个数以及该本体对应的 MUPS 中元素个数的平均值.

Table 1 The description of ontologies

表 1 待测本体的描述

ID	Ontology	Expressivity	Axioms	*C*/*P*/*I*	UnsConcept	$\overline{MUPS}$
1	Economy	*ACH*(*D*)	2 332	339/46/482	51	3.6
2	MadCow	*ALCHOIN*(*D*)	161	54/16/13	1	4
3	Chemical	*ALCH*(*D*)	192	48/9/0	37	7.6
4	Transportation	*ALCH*(*D*)	2 178	445/89/183	62	4.4
5	Koala	*ALCHOIN*(*D*)	69	21/4/6	3	4
6	Pizza	*SHOIN*(*D*)	916	98/8/5	2	3

下面以 Economy 本体(用 *Te* 表示)中的一个概念 Clover 为例,介绍本文中提到的不可满足子术语集扩展优化方法的实现过程,分别为现有黑盒技术(主要思想是基于概念之间的结构相关性选择公理,ConceptRel)、基于互补概念的术语集扩展方法(Comp)、基于宽度优先搜索的公理集选择方法(BFSminPath)以及基于深度优先搜索的公理集快速选择方法(DFSQuick).

ConceptRel 与 Comp 方法均通过概念相关选择函数来确定待扩展术语集.对术语集 *Te* 和不可满足概念 Clover,s(*Te*,Clover,1)={axe_1,axe_3},s(*Te*,Clover,2)={axe_1,axe_2,axe_3,axe_4,axe_5,axe_6}.可以看出,使用两次选择函数即可得到不可满足子术语集{axe_1,axe_2,axe_3,axe_4,axe_5,axe_6},其中,

axe_1: Clover⊑ Fodder　　axe_2: PlantAgriculturalProduct⊑ AgriculturalProduct

axe_3: Clover⊑ PlantAgriculturalProduct　　axe_4: AgriculturalProduct≠ disjointWith Fodder

axe_5: AnimalAgriculturalProduct≠ PlantAgriculturalProduct　　axe_6: Fodder⊑ OrganicObject

BFSminPath 方法需要以宽度优先方式扩展节点生成术语集搜索图,扩展过程从 Clover⊑⊤开始,使用与当前节点对应的公理概念相关的公理集合进行扩展.可以看出,与 Clover⊑⊤概念相关的公理为{axe_1,axe_3},该术语集不包括互补相关公理,需要继续扩展.进一步地,需要将{axe_1,axe_3}作为搜索图的一个节点.这是因为本文第 3 节研究术语集扩展方法时默认术语集中只包括一个公理与不可满足概念 *C* 相关,也就是说,对于本例中的两个公理 axe_1 与 axe_3 会整合为一条公理 Clover⊑ Fodder⊓PlantAgriculturalProduct 来表示.因此需要将两条公理都作为待测子术语集加入到术语集搜索图对应的节点中.与 axe_1 概念相关的公理为 axe_4 与 axe_6,与 axe_3 概念相关的公理为 axe_2 与 axe_5.那么得到子术语集{{axe_1,axe_4},{axe_1,axe_6},{axe_3,axe_5},{axe_2,axe_3}},这些公理集合都不包括互补相关公理,需要继续扩展.由{axe_1,axe_3}扩展得到{axe_1,axe_3,axe_5}和{axe_1,axe_2,axe_3},由{axe_1,axe_3}扩展得到{axe_2,axe_3,axe_4},进一步扩展{axe_1,axe_2,axe_3}获得{axe_1,axe_2,axe_3,axe_4}.{axe_1,axe_2,axe_3,axe_4}中包括互补相关公理,需要判断概念可满足性检测.由于 Clover 关于术语集{axe_1,axe_2,axe_3,axe_4}不可满足,该术语集即为 *Te* 关于 Clover 的一个 MUPS.

DFSQuick 方法需要以深度优先方式扩展节点生成术语集搜索图,扩展过程同样从 Clover⊑⊤开始,对与当前节点对应的公理概念相关的术语集进行扩展.但这个过程中每一次选择函数得到的公理集合中仅有一个公理参与扩展,这时扩展的术语集不包含互补相关概念才会进一步回溯扩展其他公理.可以看出,与 Clover⊑⊤概念相关的公理为{axe_1,axe_3},选择 axe_1 进行扩展(选择 axe_3 也可以).由 axe_1 可以扩展 axe_4 与 axe_6,由于 axe_4 中包括否定操作算子($C \not\equiv D$ 等价于 $C \sqsubseteq \neg D$ 且 $D \sqsubseteq \neg C$),选择 axe_4,{axe_1,axe_4}不包括互补相关概念而 axe_4 无法继续扩展,需要回溯将 axe_6 也加到待测术语集中.{axe_1,axe_4,axe_6}不包括互补相关概念且 axe_6 无法继续扩展,需要回溯将 axe_3 也加到待测术语集中.由 axe_3 可以扩展 axe_2 与 axe_5.同样地,可以先选择 axe_2,发现得到的术语集{axe_1,axe_2,axe_3,axe_4,axe_6}包含互补相关公理,需要判断其概念可满足性.由于概念 Clover 关于术语集{axe_1,axe_2,axe_3,axe_4,axe_6}不可满足,该术语集即为不可满足扩展阶段得到的结果.

4.2 术语集扩展优化方法比较

在本节中,我们实现了本文中涉及的关于 MUPS 求解的各类优化算法.在表 1 中的标准本体调试数据集上对这些优化方法进行了测试,并将现有的黑盒技术方法与本文中的各类优化方法进行了比较.表 2 给出了分别使用 ConceptRel、Comp、BFSminPath 以及 DFSQuick 这 4 种优化方法时,不可满足子术语集扩展阶段生成同一个 MUPS 的不可满足子术语集时的推理机调用次数(第 4,6,8,10 列)和所生成的术语集元素个数(第 5,7,9,12 列),其中表格第 2 列给出 MUPS 中的元素个数,第 11 列给出了使用 DFSQuick 方法时的回溯次数.值得注意的是,传统黑盒技术(算法 1)在不可满足子术语集扩展阶段选择的公理集合随机性高,无法给出其具体的推理机调用次数及扩展生成的术语集中元素个数.而且传统黑盒技术在每次加入术语集时都需要进行概念可满足性检测,使其推理机调用次数远远超过其他 4 类方法,这里不作比较.

Table 2 The comparison of teminology expansion optimization algorithms

表 2 子术语集扩展优化算法比较

ID	Ontology	MUPS size	ConceptRel		Comp		BFSminPath		DFSQuick		
			Reasoner	size	Reasoner	size	Reasoner	size	Reasoner	BackTrack	size
1	Economy	3	2	14	1	14	1	**3**	1	1	**3**
2	MadCow	4	3	10	1	10	1	**4**	1	1	**4**
3	Chemical	9	8	83	1	83	1	**9**	1	7	19
4	Transportation	4	3	81	1	81	1	**4**	1	3	7
5	Koala	4	3	11	1	11	1	**4**	1	2	6
6	Pizza	3	2	17	2	17	2	**3**	2	5	7

从表 2 可以看出,本文给出的 3 种优化方法从推理机调用次数和待测术语集规模方面均优于现有的 MUPS 求解方法.

1) 针对标准术语集在不可满足子术语集扩展阶段,本文中的优化方法在大多数情况下仅需调用一次推理机即可.本文中基于互补概念的方法仅在出现互补公理时判断其概念可满足性.本文的本体测试集中公理多以互补概念合取范式形式导致概念可满足.这使得仅调用一次推理机即可得到不可满足子术语集.对于 Pizza 本体,因包含数量限定量词导致的概念不可满足使得 3 种优化方法调用次数与 ConceptRel 方法相同.

2) 在不可满足子术语集扩展阶段得到的术语集元素个数方面,由于 Comp 方法只是考虑使用概念相关选择函数进行扩展,生成公理个数与使用 ConceptRel 方法相同.而在 BFSminPath 以及 DFSQuick 方法中由于优化了每次选择函数扩展时的公理集选择方式,所生成的术语集规模相对较少.这使得在术语集收缩阶段其推理机调用次数也相对减少,可以适当减少求解 MUPS 的执行时间.而 BFSminPath 方法由于选用宽度优先搜索策略,在不可满足子术语集扩展阶段即可得到一个 MUPS 且得到的 MUPS 个数小于等于 MUPS 个数的平均值,省去了不可满足子术语集收缩阶段的工作.这部分也可通过图 3 中的运行时间看出结果.

图 3 给出了分别使用现有黑盒技术(CR)、基于互补概念的术语集扩展方法(Comp)、基于宽度优先搜索的公理集选择方法(BFS)以及基于深度优先搜索的公理集快速搜索方法(DFS)时,不可满足子术语集扩展阶段运行时间(加后缀_E 表示)和求一个 MUPS 的总体运行时间(加后缀_A 表示).图 3 中横轴表示表 1 中的本体术语集,纵轴表示的是运行时间,单位为毫秒(ms).可以看出,本文给出的 3 种优化方法均优于现有的黑盒技术方法,且

针对规模较大且复杂的本体,其效率更为明显,如 *T*3.

从图 3 可以看出,使用 BFSminPath 与 DFSQuick 时求解 MUPS 的运行时间优于 ConceptRel 与 Comp 方法,而 BFSminPath 与 DFSQuick 针对不同的本体其效率差异并不明显(*T*1 与 *T*2 是 DFSQuick 效率高,其他本体是 BFSminPath 效率高).BFSminPath 的优点在于能够在不可满足子术语集扩展阶段直接得到一个 MUPS,减少了收缩阶段的运行时间,而缺点在于需要考虑术语集中所有公理.DFSQuick 的优点在于每次选择函数生成的公理集合中只需选择一个即可,不需要遍历所有节点,可以快速查找不可满足子术语集.缺点在于如果选择节点不合适,需要多次回溯才能找到不可满足子术语集.分析数据集 *T*1~*T*6 发现,当通过一次概念相关选择函数得到的公理集合中的多个公理属于同一个 MUPS 时,DFSQuick 需要完成多次回溯,降低运行效率.因此,可以通过本体术语集中的公理之间逻辑关系选择优化策略:对于一个不可满足概念 *C*,若使用选择函数得到的子术语集 *T* 中公理之间包含相同概念,则建议使用 BFSminPath;而当 *T* 中公理元素过多时,建议使用 DFSQuick.

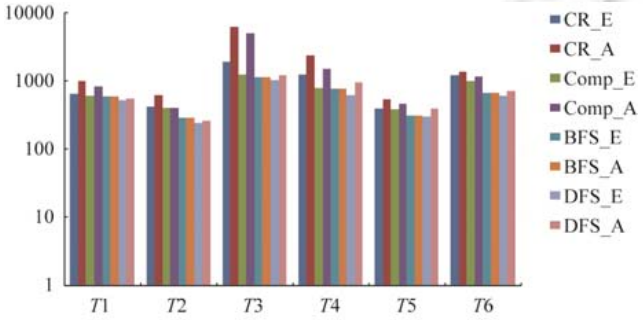

Fig.3 The comparison of runtime for terminology expansion and computing MUPS

图 3 不同优化算法的术语集扩展与 MUPS 求解时间对比

4.3 求解所有 MUPS 优化

前面介绍了求解术语集关于不可满足概念的一个 MUPS 的优化方法.而现有的多数本体测试工具更多地关注于求解一个不可满足概念的所有 MUPS 的方法.如本文第 3.1 节中介绍,现有的求解所有 MUPS 的方法主要是采用碰集树的方法,其优化技术主要在于剪枝操作与节点重用,相关工作已应用到 Pellet 推理机中.本文中的术语集扩展优化方法是针对求解单个 MUPS 的优化方法,可以直接应用于求解所有 MUPS 方法之中,可以有效地提高运行效率.图 4 给出了分别将现有黑盒技术(CR)、基于互补概念的术语集扩展方法(Comp)、基于宽度优先搜索的公理集选择方法(BFS)以及基于深度优先搜索的公理集快速搜索方法(DFS)应用到求解所有 MUPS 的方法时的运行时间(加后缀_All 表示求解全部 MUPS).图 4 中横轴表示表 1 中的本体术语集,纵轴表示运行时间,单位为毫秒(ms).可以看出,将术语集扩展优化的方法也可以优化求解所有 MUPS 的任务.同样,针对规模较大且复杂的本体,其效率更为明显,如 *T*3 和 *T*4.这是因为针对复杂本体,选择公理的启发式策略更有优势.

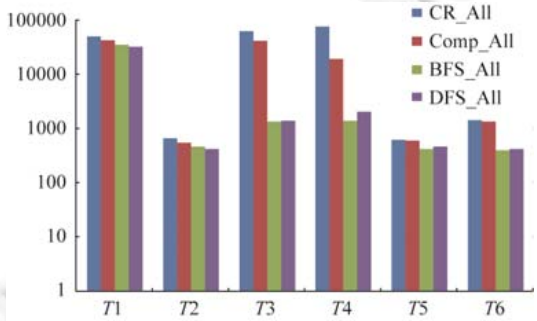

Fig.4 The comparison of runtime for computing all MUPS

图 4 不同优化算法求解所有 MUPS 的时间对比情况

5 总 结

本文结合互补概念和术语集的搜索图提出了一种不可满足子术语集扩展优化方法,构造与术语集扩展对应的搜索图,并分别采用宽度优先搜索和深度优先搜索策略查找不可满足子术语集.基于互补概念的术语集扩展方法仅在待测术语集包含互补概念时才进行概念可满足性检测,大大减少了推理机的调用次数.基于宽度优先搜索的公理集选择方法通过宽度优先搜索方法查找不可满足子术语集扩展对应的搜索图中的 dead 节点,该节点对应的子术语集在大多数情况下是一个极小不可满足子术语集,减少了术语集收缩阶段的操作.而深度优先搜索策略通过启发式方法从选择函数获得的公理集合中优先选择公理进行术语集扩展,大大减少了搜索图中节点的生成个数及待测术语集的规模.实验结果表明,本文给出的术语集扩展优化方法在一定程度上提高了求解 MUPS 的效率.

References:

[1] Staab S, Studer R. Handbook on ontologies. Int'l Journal of Information Management, 2009.

[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] Doms A, Schroeder M. GoPubMed: Exploring PubMed with the gene ontology. Nucleic Acids Research, 2005,33(Suppl. 2): 783−786.

[4] Yang YH, Du JP, Ping Y, Wang J. Ontology-Based intelligent information retrieval system. Ruan Jian Xue Bao/Journal of Software, 2015,26(7):1675−1687 (in Chinese with English abstract). http://www.jos.org.cn/1000-9825/4622.htm [doi: 10.13328/j.cnki.jos.004622]

[5] Baader F, Calvanese D. The Description Logic Handbook: Theory, Implementation, and Applications. 2nd ed., Cambridge University Press, 2007.

[6] Kalyanpur A, Parsia B, Sirin E, Hendler J. Debugging unsatisfiable classes in OWL ontologies. Journal of Web Semantics, 2005,3(4):268−293.

[7] 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. 355−362.

[8] Schlobach A, Huang ZS, Cornet R, Harmelen F. Debugging incoherent terminologies. Journal of Automated Reasoning, 2007,39(3): 317−349.

[9] Kalyanpur A, Parsia B, Horridge M, Sirin E. Finding all justifications of OWL DL entailments. In: Proc. of the Semantic Web, the 6th Int'l Semantic Web Conf., the 2nd Asian Semantic Web Conf., ISWC/ASWC. 2007. 267−280.

[10] Baader F, Pealoza R. Automata-Based axiom pinpointing. Journal of Automated Reasoning, 2010,45(2):91−129.

[11] Ye YX, Ouyang DT, Su J. Entailment-Based axiom pinpointing in debugging incoherent terminologies. In: Proc. of the 2nd Int'l Workshop on Semantic Technologies. 2015. 105−115.

[12] Baader F, Suntisrivaraporn B. Debugging SNOMED CT using axiom pinpointing in the description logic EL+. In: Proc. of the 3rd Int'l Conf. on Knowledge Representation in Medicine. 2008.

[13] Reiter R. A theory of diagnosis from first principles. Artificial Intelligence, 1987,32:57−95.

[14] Schlobach S. Diagnosing terminologies. In: Proc. of the 20th National Conf. on Artificial Intelligence. 2005. 670−675.

[15] Friedrich G, Shchekotykhin K. A general diagnosis method for ontologies. In: Proc. of the Int'l Semantic Web Conf. 2005. 232−246.

[16] Shchekotykhin K, Friedrich G, Fleiss P, Rodler P. Interactive ontology debugging: Two query strategies for efficient fault localization. Journal of Web Semantics, 2012,12:88−103.

[17] Jannach D, Schmitz T, Shchekotykhin K. Parallel model-based diagnosis on multi-core computers. Journal of Artificial Intelligence Research, 2016,55:835−887.

[18] Parsia B, Sirin E, Kalyanpur A. Debugging OWL ontologies. In: Proc. of the 14th Int'l World Wide Web Conf. 2005. 633−640.

[19] Kalyanpur A, Parsia B, Sirin E. Black box techniques for debugging unsatisfiable concepts. In: Proc. of the 2005 Int'l Workshop on Description Logics. 2005.

[20] Horrocks I, Kutz O, Sattler U. The even more irresistible SROIQ. In: Proc. of the 10th Int'l Conf. on Principles of Knowledge Representation and Reasoning. 2006,6:57–67.

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] 杨月华,杜军平,平源.基于本体的智能信息检索系统.软件学报,2015,26(7):1675–1687. http://www.jos.org.cn/1000-9825/4622.htm [doi: 10.13328/j.cnki.jos.004622]

崔仙姬(1986－),女,吉林图们人,博士,讲师,CCF 专业会员,主要研究领域为语义Web,自动推理.

张俊星(1969－),男,博士,教授,主要研究领域为民族信息处理.

何加亮(1977－),男,博士,讲师,主要研究领域为物联网,移动互联网应用.

高健(1983－),男,博士,副教授,CCF 专业会员,主要研究领域为人工智能,约束求解.