基于描述逻辑的本体概念分类优化算法

龚 波 谢兴生 张一鸣
(中国科学技术大学自动化系 合肥 230026)

摘 要 本体采用形式化的方式表达和规范某一领域共享的知识。当前本体表达语言普遍将描述逻辑(Description Logic,DL)作为其逻辑基础。利用本体对领域知识进行表达,形成共享的领域知识库。知识库中的领域术语(概念)一般都是分类层次结构但并不完整,并且包含很多的隐藏信息。借助于描述逻辑的推理服务,可以将知识库中的隐藏信息挖掘出来,从而得到充分完备的概念层次结构。研究了基于描述逻辑的知识库概念分类优化算法,分析了影响算法性能的主要因素,并提出了一种逻辑独立的标记矩阵优化方法,可以在一定程度上提升分类算法的性能。

关键词 描述逻辑,知识库,分类,标记矩阵
中图法分类号 TP301 **文献标识码** A

Optimising Classification for Terminology of Ontology Based on Description Logic

GONG Bo XIE Xing-sheng ZHANG Yi-ming
(Department of Automation, University of Science and Technology of China, Hefei 230026, China)

Abstract Currently, ontologies are expressed by Description Logic(DL) based ontology languages in order to take advantage of reasoning service of DL. Generally, ontology is a taxonomic Knowledge Base(KB) to specify the terminologies of particular domain. However, these knowledge bases consist of a majority of visible classification, which is generally incomplete and needed to explore the unconspicuous taxonomy. We made a survey of optimising techniques associated with terminology classification algorithm and proposed a novel tagging optimization which can speed up classification to some extent.

Keywords Description logic, Knowledge base, Classification, Tagging matrix

1 引言

当前,越来越多的本体采用基于 DL 的本体语言,如 OWL-DL 语言(W3C 工作组指定的知识表达语言标准)进行表达[1]。这样,本体既能对领域知识进行合理的表达,形成领域本体,又能利用 DL 的推理服务支持有效地推理。领域本体或知识库(Knowledge Base,KB)中的术语(概念)集合一般具有分层结构,但分类信息并不完整,且含有大量的隐藏知识。因此,领域本体需要利用 DL 的推理服务重新计算知识库中的概念层次。KB 中的概念分类过程就是计算 KB 中每一个概念的完整确切的包含关系,包括父概念和子概念[2]。DL 的推理服务主要有概念包含测试(Subsumption Test)和知识库的一致性检测(Consistency Check)[3],本文所涉及的推理主要是包含检测。一般地,基于 DL 的知识库含有大量的公理(Axiom),这些公理用来标识概念之间包含关系。虽然概念之间的明显(已知)的包含可以通过公理集识别,但隐藏的概念包含需要通过包含测试推理得出。在 DL 中,概念之间的包含一般采用以下的方式转化为满足性问题:概念 C 包含概念 D ($D \sqsubseteq C$)成立,当且仅当概念 D 和概念 C 的补($\neg C$)的交($D \sqcap \neg C$)是不满足的[4](相关 DL 的预备知识将在后面给出)。当前一些非常优秀的描述逻辑推理机,如 Pellet,Racer 和 Fact++采用了约束传播的技术对领域本体中的概念集合进行分类,具体分类算法就是对概念执行上位(父)搜索(Top-phase Search)和下位(子)搜索(Bottom-phase Search),直到遍历知识库中的概念[2]。但这种上下位的遍历搜索将会执行大量的概念包含测试,并且很多的测试结果会没有价值,因为某些概念之间没有任何包含关系。与此同时,基于 Tableau 算法的包含测试推理的效率通常是不高的,尤其是面对基于表达能力很强的 DL 所构建的庞大且结构复杂的知识库[4]。一些优化技术如 Lazy Unfolding 和 Absorption 技术,已用来提升包含测试的性能,它们的基本出发点就是简化知识库的复杂结构,将复杂的公理简化为若干个形式简单的公理[5,6]。这些技术都是知识库分类前的一些预处理优化技术,简化知识库的结构并使包含测试的效率有所提升。

在本文中,相关 DL 的预备知识将在第 2 部分给出。第 3 部分将重点介绍概念分类算法的整体结构和其中所采用的一些优化方法。第 4 部分提出了一种标记矩阵的优化技术,可以进一步提升分类算法的性能。最后给出实验分析并且进行了总结。

到稿日期:2008-06-29 本文受国家自然科学基金(60401015)资助。
龚 波(1983-),男,硕士生,主要研究方向为智能信息处理等,E-mail:gongbo@mail.ustc.edu.cn;**谢兴生**(1965-),男,副教授,主要研究方向为智能信息集成与处理、人工智能与模式识别;**张一鸣**(1982-),男,硕士生,主要研究方向为人工智能与模式识别。

2 描述逻辑基础

1)描述逻辑 ALC 语法和语义：ALC 是最小的命题封闭的 DL[3]，NC 和 NR 是可数不相交的原子概念集和原子关系集。ALC 的概念描述递归定义如下：(1)任意原子概念 $A\in NC$ 是 ALC 概念；(2)C,D 是 ALC 概念，R 是原子关系 $R\in NR$，则表达式 $\neg C$(补)、$C\sqcup D$(并)、$C\sqcap D$(交)、$\exists R.C$(存在约束)和 $\forall R.C$(全称约束)是 ALC 概念，这些由操作子连接的概念也称为复杂概念。两个特殊记号：$\top$ 是指代论域全集的顶概念，是所有概念的父概念；$\bot$ 是指代空集的底概念，即 $\top=A\sqcup\neg A$，$\bot=A\sqcap\neg A$。ALC 的语义是一种 Tarksi 式的语义，概念被解释为论域上的一元谓词，关系被解释为论域上的二元谓词。ALC 的解释是一个二元对 $I=(\Delta^I,\cdot^I)$，其中 Δ^I 代表论域的非空集合；$\cdot^I$ 是解释函数，它将每个概念映射为 Δ^I 的子集，每个关系映射为 $\Delta^I\times\Delta^I$ 的子集，分别称作概念和关系的解释[9]。关于概念、关系的语义的细节在表 1 中给出。

表 1 描述逻辑 ALC 的语法和语义

	语法	语义
概念	A	A^I
	$\top$	Δ^I
	$\bot$	$\emptyset$
	$\neg C$	Δ^I-C^I
	$C\sqcap D$	$C^I\cap D^I$
	$C\sqcup D$	$C^I\cup D^I$
	$\exists R.C$	$\{d\in\Delta^I \mid R^I(d)\cap C^I\neq\emptyset\}$
	$\forall R.C$	$\{d\in\Delta^I \mid R^I(d)\subseteq C^I\}$
关系	R	$R^I\subseteq\Delta^I\times\Delta^I$
公理	$C\sqsubseteq D$	$C^I\subseteq D^I$

2)基于 DL 的知识库：DL 知识库包括表示术语公理的 TBox 和表示断言事实的 ABox。Tbox 表示知识库中概念间的包含和等价关系；ABox 陈述知识库中个体和概念以及个体对和关系之间的隶属关系。在 Tbox T 中，A,B 为原子概念，C 是概念，形如 $A\equiv C$(A 与 C 等价)的表达式称为概念定义。形如 $A\sqsubseteq C$ 的表达式称为简单或规则公理，若概念 C 不直接或间接引用 A(当 C 的表达式中含有 A 时，称为引用)。Tbox 中还有形如 $A\sqcup B\sqsubseteq\exists R.A\sqcup R.C$ 的概念公理，称为一般概念包含(复杂公理)，会大大增加对 TBox 进行推理的复杂性，应该采用上面提及的 Absorption 优化技术进行简化[5]。解释 I 满足 $A\equiv C(A\sqsubseteq C)$，当且仅当 $C^I=D^I(C^I\sqsubseteq D^I)$ 成立，如表 1 所列。解释 I 被称为 T 的模型，当且仅当 I 满足 T 中所有的公理[9]。而在 ABox 中，主要包括形如 a:C、(a,b):R 的概念断言和关系断言(a 为个体，$a^I\in\Delta^I$)，本文不做详细描述。

3)包含测试和 Tableau 算法：在 Tbox T 约束下，概念 C 包含于概念 D，当且仅当任意 T 的模型 I 满足 $C^I\sqsubseteq D^I$；概念 C 是可满足的，当且仅当存在 T 的模型 I，满足 $C^I\neq\emptyset$。而概念包含可转化为概念满足，即 C 包含于 $D(C\sqsubseteq D)$ 等价于概念 $C\sqcap\neg D$ 不可满足[4]。概念满足性问题在于找到一个解释 I，使得概念不映射为空集。但解释的数目是无限大的，不能遍历所有的解释来证明满足性。因此，Tableau 算法模拟一种普适的解释构造过程来判定概念的可满足性。若构造成功，则生成一个解释 I 使得概念满足；若构造失败，则表明不存在满足概念的解释。关于 Tableau 算法的细节参考文献[3,4]。

3 知识库概念分类优化算法

利用 DL 的推理服务对知识库进行概念分类，主要是依据包含关系计算知识库中每一个概念的上下位关系。但概念包含测试在整个分类过程中耗费较长时间，尤其是当采用表达能力强的 DL 来表达知识库时[4,5]。因此，分类算法要尽可能减少使用包含测试的次数。基于概念的上位或下位搜索的宽度优先遍历，能找出每一个概念的直接父或子概念，并且有助于减少包含测试的次数[2]。然而，通过利用知识库中概念间已知的包含关系，可以控制进行算法遍历的概念先后加入的顺序。这个顺序被称之为定义顺序(Definitional Order)，能很大程度地利用已知的概念包含并且传播已经执行的概念包含测试的结果，从而使部分后续的包含检测得以避免[2]。以下将给出优化概念分类算法的几个主要部分。

3.1 主要数据类型：*TaxonomyNode* 和 *Taxonomy*

TaxonomyNode{ String name; Set equivalents; Set instances; Set supers; Set subs}

Taxonomy { Map nodes; TaxonomyNode TOP_NODE, BOTTOM_NODE; …}

其中，分类结点 *TaxonomyNode* 和知识库中的概念(概念结点)一一对应，存放对应概念的名字、等价概念集、个体实例集、直接父概念集和直接子概念集。*Taxonomy* 则存储了知识库中所有概念的分类层次结构，二维 Map(nodes)的键存储每一个概念名，键值存储此概念对应的分类结点数据。

3.2 概念分类算法

分类算法通过计算知识库中每一个概念的直接父概念和直接子概念，可以确定每一个概念在知识库分类层次中的位置。但在分类之前还需要一些预处理技术，如通过已知的概念包含关系构建知识库预分类层次和控制概念进行上下位搜索并更新到最终分类层次的顺序。其目的就是减少概念包含测试调用的次数，从而在时间上提升算法的效率。

3.2.1 主体算法：计算知识库中完整的概念包含层次

```
Input: knowledge base-concepts, axioms, …
Output: complete subsumption hierarchy of all concepts
Begin
  Local Variable: defTaxonomy ∈ Taxonomy, G ∈ Taxonomy, Set
                  sortedClasses;
  defTaxonomy=compute_defTaxonomy(concepts, axioms);
  sortedClasses = topologocialSort (defTaxonomy);
  for all Ci∈sortedClasses do
    classify (Ci, G);
  end for;
  return G;
End;
```

在以上主体算法中，算法输入为知识库中所有的概念以及公理集(TBox)。但知识库或领域本体的解析需要第三方的工具支持，如惠普(HP)实验室开发的 Jena 工具包，可以对诸如 OWL 语言以及其它本体语言表达的领域本体进行处理。首先利用知识库中已知的概念包含关系计算所有概念的预分类层次(defTaxonomy)。例如，Tbox 中的显式公理 $A\sqsubseteq C,A\sqsubseteq C_1\sqcap\cdots C_n$，那么在预分类层次中，概念 $C,C_1,\cdots,C_n$ 处于概念 A 的上层。预分类之后采用了一种优化方法，对所有

的概念进行拓扑排序(topologocialSort)。排序的原理是基于知识库中每一个概念的父概念集合的长度,集合的长度越长,排序越靠后[1]。根据经验分析,如果一个概念的已知父概念数目越多,那么这个概念大致处于概念层次的底部;反之,处于概念层次的顶部。预分类层次的计算找出了所有已知的概念包含,从而这些概念间的包含测试可以避免[3]。而拓扑排序则初步明确了每一个概念在未来概念层次之中的位置,控制概念加入到完整概念层次(G)中的顺序 classify (C_i,G),使得预分类层次转换到最终分类层次的计算产生的代价最小,即减少了不必要的概念包含测试。

3.2.2 classify (C_i,G):计算每一个概念结点的包含关系

```
Input:原子概念 Ci, 完整分类层 G(包含已分类概念)
Output:更新概念结点 Ci 的完整的包含关系到 G
Function:classify (Ci, G)
Begin
  G.get (Ci).supers = Get_super (Ci, G);
  G.get(Ci).subs = Get_sub(Ci,G);
  G.get(Ci).equivalents = Get_equivalents (Ci, G);
  G.get(Ci).instances= Get_instance (Ci,G);
End
```

以上算法中的 4 个函数分别是为了计算每一个概念结点的完整分类信息,包括其直接父、子和等价概念集,以及属于此概念的个体集合,这些信息可以确定一个概念结点在知识库分类层次中的确切位置。计算一个概念的直接父概念集的算法细节将在 3.2.3 节给出,这个计算过程也被称作宽度优先的上位搜索,而下位(子概念集)搜索也采用了类似的方式。

3.2.3 宽度优先遍历计算一个概念的父概念集

为了详细地描述遍历算法的细节,我们采用一个具体的知识库概念层次结构来说明计算父概念集的过程。如图 1 所示,所有概念的拓扑排序为:[Top(或⊤),A,B,A_1,A_2,B_1,B_2,C_i]。假定除概念 C_i 之外,其它概念已经被正确分类,最后计算 C_i 在概念层次中的位置。通过已知的包含关系得出:概念 A 是 C_i 的直接父概念,但执行了包含测试之后,实际上 A_2 才是 C_i 的直接父概念,如图 2 所示。以下是宽度优先的上位搜索算法的详细流程。

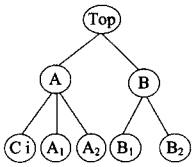

图 1 不完整的分类树结构

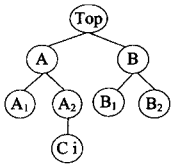

图 2 完整的分类树结构

1)以根节点(Top 或⊤对应的概念结点)作为遍历的初始结点,即令 start_searchNode=Top。

2) 取遍历起点 start_searchNode 直接子类概念集 *subs* 到当前处理队列 cList1(按图 1,start_searchNode 为已分类结点,其直接子类概念集 *subs* 已有正确值)。

3) 遍历检查当前处理队列 cList1 中的每个结点,将对 C_i 存在包含关系的概念结点加入到处理队列 cList2(根据图 1,开始只有概念 A 会被加入到 cList2 中)。

4) If cList2 为空 then 说明当前遍历的初始结点 start_searchNode (初始为 Top) 是 C_i 的直接父结点;将 start_searchNode 加入到 C_i 的直接父概念集 *supers* 中。

Else cList2 非空 (按图 1,cList2=$\{A\}$非空),cList2 中的每个结点都是 C_i 的直接或间接父概念结点。依次用队列 cList2 中的概念结点替换 start_searchNode,返回步骤 2),递归处理 cList2 中的每个结点。

End if

当对概念 C_i 进行分类的时候,通过传播概念包含测试的结果以及已知的概念包含关系,宽度优先的上位搜索算法充分利用了概念包含的传递性这一特点来减少使用包含测试的次数。如在算法的步骤 3)中,检查 cList1 =$\{A,B\}$中的概念和当前概念 C_i 的包含关系,A 和 C_i 的包含关系是已知的,而 B 和 C_i 的关系需要通过包含测试得出,测试的结果是 B 不包含 C_i($C_i \not\sqsubseteq B$),于是只有概念 A 加入到 cList2 中。后续的算法将概念 B 所属的概念层次分支剪掉,即概念 B 的子结点与 C_i 没有包含关系($C_i \not\sqsubseteq B$ 和 $B_1 \sqsubseteq B$ 隐含着 $C_i \not\sqsubseteq B_1$)。这种优化技术已经在诸如 Pellet 推理机中得以实现,并且每个概念结点采用了一个临时的二维 Map 来存储与其它概念的包含关系的标签值,如 *is_Supers*,*not_Supers*,*unknown*。当执行深度优先的上位或下位搜索时,只有当两个概念结点之间的包含关系未知(*unknown*),包含测试才被调用(如 C_i 和 B)。但这个 Map 只是一个临时的数据,当包含测试被用来计算 C_i 和 B 的包含关系后,需要更新 Map 中的代表概念结点 C_i 和 B 包含关系的标签值。但该数据的更新不能被其它的概念结点,特别是一些相关概念(如 C_i 和 B 的子、等价概念等)所充分利用,从而也可以更新各自 Map 中的标签值,以避免包含测试的调用。基于上述分析,概念分类算法,尤其是宽度优先的上下位搜索需要进一步优化。于是我们提出了一种基于动态标记矩阵的优化技术,用来最大程度地利用已执行的包含测试的结果,并利用概念包含的传递性,将此结果传递到更深的概念层次,以便最大数量地减少包含测试调用的次数。

4 一种基于标记(Tagging)矩阵的优化技术

这种优化技术的基本出发点是:在整个分类过程中,构建一个标记矩阵来存储知识库中所有概念之间的包含关系,并且动态更新所有相关概念的标记值。一旦找到已知的概念包含或执行了包含测试,力求把已确定的概念包含传播得最远。这就可能确定其它概念结点(不同于执行包含测试的概念)之间的包含关系,从而最大程度地减少时间耗费相对巨大的包含测试的调用次数。具体来说,每个概念结点 *Taxonomy-Node* 的数据结构扩充一个二维散列来存储标记值。这些标记值有:*is_super*(是父概念),*not_super*(非父概念),*is_sub*(是子概念),*not_sub*(非子概念),*equivalent*(等价),*disjoints*(不相交)和 *unknown*(未知),分别用来表示同其它概念结点的包含关系。所有概念结点的标记组合起来就构成了一个矩阵,而且每当矩阵里的一个标记值被更新后,会有其它足够多的相关标记值同样被更新。以下有两个依据经验分析而来的引理,给动态标记矩阵的优化方法提供理论上的正确性。

引理 1 C_1,C_2 是知识库中的两个原子概念,已知 $A_1 \equiv C_1$,$B_1 \sqsubseteq C_1 \sqsubseteq D_1$(概念 C_1 包含 B_1,D_1 包含 C_1),$A_2 \equiv C_2$,$B_2 \sqsubseteq C_2 \sqsubseteq D_2$ 成立。若由包含测试的结果得到 $C_1 \not\sqsubseteq C_2$,则有概念 C_2 不是 C_1 的等价概念和父概念的父概念。反之也成立(即 $A_1 \not\sqsubseteq C_2$,$A_1 \not\sqsubseteq A_2$,$A_1 \not\sqsubseteq B_2$,$D_1 \not\sqsubseteq C_2$,$D_1 \not\sqsubseteq A_2$,$D_1 \not\sqsubseteq B_2$)。

引理 2 C_1,C_2 是知识库中的两个原子概念,引理 1 的已

知条件同样在这里成立。若执行包含测试得到的结果是 $C_1 \sqsubseteq C_2$，则有 C_1 及其子概念和等价概念是 C_2 的等价概念和父概念的子概念。

以上引理充分利用了概念包含的传递性。将一次包含测试的结果最大限度地传播，使后续的包含测试得以避免。动态标记矩阵则结合了已知的概念包含和包含测试结果，通过所有相关概念标记值的更新，将已经确定的包含关系传播到所有相关概念，并且可能确定相关概念之间的包含关系。具体来说，在利用已知的概念包含构建知识库的概念预分类层次过程中，标记矩阵被初始化，其中的部分标记被更新。当深度优先的上位搜索开始时，标记矩阵根据包含测试的结果动态更新所有的关联概念的标记值。下面用一个具体的例子来描述动态标记矩阵的实现细节。

首先，引入符号 $[C\ D]$：*unknown*，表示概念结点 C 同其它概念（如 D）之间的包含关系的标记。如图 3 所示，知识库的预分类层次由已知的概念包含建立，并且其定义顺序为 $[C, E, D, D_1, D_2]$（省略顶概念⊤）。假设 C 为 D_2 的隐藏子概念，则概念层次需要通过上位搜索来修正。根据定义顺序和上位搜索算法，首先取出概念 C 和 E，D 执行包含测试，测试结果为 $C \not\sqsubseteq E$ 和 $C \sqsubseteq D$。接着需要更新标记矩阵中的标记值，即 $[C,E]$：*not_super*，$[C,D]$：*is_super*，$[D,C]$：*is_sub*，$[D,E]$：*not_super*，$[E,C]$：*not_sub*，$[E,D]$：*not_sub*。若概念 C，E 有等价或子概念，还需要修正标记矩阵。随后，依据宽度优先搜索算法，分别测试概念 C 和 D_1，D_2 的包含关系，结果是 $C \sqsubseteq D_1$，$C \sqsubseteq D_2$。标记矩阵需要更新：$[C, D_1]$：*is_super*，$[C, D_2]$：*is_super*，$[D_1, E]$：*not_super*，$[D_2, E]$：*not_super* 等等。根据搜索算法，需要测试概念 E 和 D 之间的包含关系，结果是 $E \not\sqsubseteq D$（$D \not\sqsubseteq E$ 已经有由先前更新的标记 $[D,E]$：*not_super* 得到）。执行完这个包含测试并且更新了相关标记值后，发现 $E \not\sqsubseteq D_2$，$E \not\sqsubseteq D_1$，$E \not\sqsubseteq C$，结合先前的标记值 $[C,E]$：*not_super*，$[D_1,E]$：*not_super*，$[D_2,E]$：*not_super*，概念 E, C, D_1, D_2 之间的包含关系完全确定。但是在原始搜索算法中，$E \not\sqsubseteq D_2$，$E \not\sqsubseteq D_1$，$E \not\sqsubseteq C$ 是需要通过包含测试来确定的。但在使用了动态标记矩阵技术后，这些包含测试得以避免，从而降低了算法的时间复杂性，提升了分类算法的性能。图 4 给出完整的概念分类树。

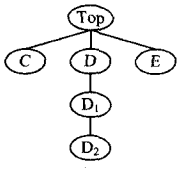

图 3　概念预分类树

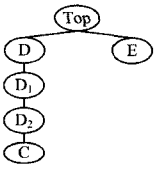

图 4　完整的概念分类树

事实上，如果宽度优先的搜索能够上下位同时遍历，也就是上位搜索和下位搜索同步进行，就可以避免更多的包含测试。在图 3 中，下位搜索首先会对概念结点 C 和 D_2 进行包含测试，其结果为 $C \sqsubseteq D_2$，则概念 C 与 D_1，D 的包含关系就完全确定了。而这在上位搜索中，还需执行包含测试才能确定其包含关系。所以，联合上位搜索和下位搜索，能够更进一步地减少包含测试的次数，提升分类算法的性能。由于知识库概念层次的差异性，上下位搜索的最优结合方式是很难确定的，但一个次优的融合方式是可行的。

5　实验与分析

在这一部分中，为了检验动态标记矩阵技术的优化效果，我们将动态标记矩阵技术灵活地融入到 Pellet 推理机的分类算法中。这两者完全兼容，因为标记矩阵优化技术是在逻辑上独立的。虽然标记矩阵的引入会增加额外的存储开销，但存储空间已经不是计算的瓶颈，重要的是能减少分类时间。通过对表 2 所列的有代表性的领域本体（知识库）进行概念分类测试，我们比较了原始的 Pellet 知识库概念分类算法和采用优化技术后的算法的分类性能。这些测试用的本体是源自一些著名的站点（如 Kona2，Mindswap 和 W3C），采用 OWL-DL 本体语言进行表达，完全支持描述逻辑推理。

表 2　测试本体

编号	本体名称	来源	NNC	NPA
1	Profile	kaon2	9	10
2	Concept	kaon2	17	17
3	Mindswappers	mindswap	71	48
4	Process	kaon2	85	40
5	Food	w3c	205	64
6	Dolce	kaon2	315	124
7	Government	kaon2	53	84

具体来说，我们比较了 Pellet1.4 版本的分类算法和优化后的加强版分类算法对表 2 中每个本体进行概念分类时所需要调用的包含测试的次数。之所以不以分类时间消耗作为比较标准，是因为所有包含测试的时间消耗占整个分类时间的绝大部分。而分类时间总是与计算环境的软硬件配置有关，所以采用包含测试的次数作为比较标准是很有代表性的。遗憾的是，我们在优化分类算法中并没有将上位搜索和下位搜索融合起来，只是采用了上位搜索，所以这种优化还没有达到最完美的效果。一些测试本体的细节在表 2 中给出，其中 NNC 表示本体知识库中原子概念的数目；NPA 是 Tbox 中简单公理的数目（上面提及的概念定义和概念特化）。概念包含测试次数的比较在图 5 中给出。通过对表 2 和图 5 的分析，我们可以得出以下结论。

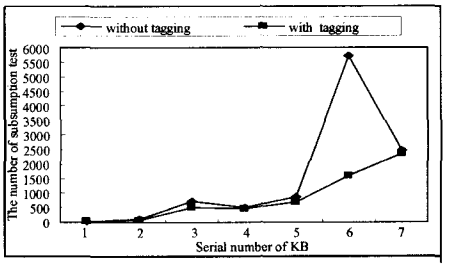

图 5　分类算法优化前后调用包含测试次数对比

1）本体的概念数目越多，公理数目越多，则知识库分类所需包含测试的次数就越多，分类时间就越长。因为知识库结构越复杂，DL 推理（包含测试）的复杂性增加。

2）采用优化技术后，分类算法的性能明显提升。对所有测试本体进行分类时，平均包含测试调用的次数减少了约 27%，分类时间也相应地减少。

3）知识库的原始概念层次结构与分类算法的优化效果有关联。知识库的概念越多，公理越多，原始概念层次越深且结构越平衡时，分类优化效果更加显著。进一步分析得出：当概

念层次的深宽比变大，且结构比较平衡对称时，标记矩阵会将已确定的概念包含在概念层次中，使其传播得更远，从而确定其它更多的概念包含，对一些概念层次分支进行有效的剪枝，从而减少包含测试调用的次数，提升分类性能。

结束语 通过上面的描述，我们给出了知识库术语分类算法的相关优化技术的轮廓。一些分类之前的预处理优化技术，如上面提及的 lazy unfolding，absorption 等优化技术，可以对知识库中的复杂公理进行处理，使得知识库中的公理结构上更加简单，从而使 DL 推理(包含测试)的效率大大提高，并为后续的概念分类提供有利的支持。事实上，这两种优化技术只是保证了 tableau 算法能快速地找出概念包含 $D \sqsubseteq C$ 或者概念 $D \sqcap \neg C$ 不满足。而检测概念不包含($D \not\sqsubseteq C$)或概念满足更加困难和复杂，另外已提出一些优化技术用来解决此类问题，如 semantic branching search，dependency-directed backtracking，caching[3,4,8]等。其中部分已经在诸如 Pellet，Fact 推理机中得以实现。本文随后给出了知识库概念分类算法的完整流程，在此基础上，提出了一种动态标记矩阵的优化技术，并在理论和实验上证明了它能提升分类算法的性能。该优化技术在逻辑上是独立的，因此能很灵活地同其它优化技术融合起来。当前，知识表达在人工智能的实际应用方面扮演了很重要的角色，但领域知识经过诸如本体这样通用知识表达规范表达之后，知识层次并不完整并且需要进一步挖掘知识中隐含的未知信息[10]。本文描述的相关分类算法和优化技术正是用来解决此类问题的。知识表达同其它的智能应用的融合是一大发展趋势，未来我们会重点关注如何将领域知识表应用于信息集成、信息检索。

参 考 文 献

[1] Tsarkov D, Horrocks I. Optimised Classification for Taxonomic Knowledge Bases // Proceedings of the 2005 International Description Logic Workshop (DL 2005). Edinburgh, Scotland, UK, 2005, 147

[2] Baader F, Franconi E, Hollunder B, et al. An empirical analysis of optimization techniques for terminological representation systems // Proc. of the 3rd Int. Conf. on the Principles of Knowledge Representation and Reasoning (KR92). Cambridge, MA: Morgan-Kaufmann, 1992: 270-281

[3] Baader F, Calvanese D, McGuinness D, et al. The Description Logic Handbook: Theory, Implementation and Applications. Cambridge. UK: Cambridge University Press, 2003: 47-100, 313-355

[4] Horrocks I, Patel - Schneider F. Optimizing Description Logic Subsumption. Journal of Logic and Computation, 1999, 9(3): 267-293

[5] Horrocks I, Tobies S. Reasoning with Axioms: Theory and Practice // Proc. of the 7th Int. Conf. on Principles of Knowledge Representation and Reasoning (KR 2000). Breckenridge, Colorado, USA, 2000: 285-296

[6] Zuo M, Haarslev V. High Performance Absorption Algorithms for Terminological Reasoning // Proc. of the 2006 Int. Description Logic Workshop (DL 2006). Lake District, UK, 2006: 159-166

[7] Horrocks I, Tobies S. Optimisation of Terminological Reasoning // Proc. of the 2000 Description Logic Workshop (DL 2000). Aachen, Germany, 2000: 183-192

[8] Horrocks I. Using an Expressive Description Logic: FaCT or Fiction? // Proc. of the 6th Int. Conf. on Principles of Knowledge Representation and Reasoning (KR 98). Trento, Italy, 1998: 636-647

[9] 陆建江，张亚非，苗壮，等. 语义网原理与技术. 科学出版社，2007

[10] 邓志鸿，唐世，张铭，等. Ontology 研究综述[J]. 北京大学学报：自然科学版，2002，38(5)：730-737

(上接第 118 页)

支持度最高的 n 个(n 值根据具体情况选取)频繁闭项集作为分析的基础，并将其转换为关联规则。表 2 中列举了其中部分关联规则及它们的支持度(数据仅为举例，不代表任何真实系统)。

通过对关联规则的分析，我们得出一系列感兴趣的知识。例如，雨天时突防成功的可能性更高、采用弹道 1 突防成功的可能性更高等。

结束语 数据流挖掘技术能够为用户及时地提供数据中感兴趣的模式。而大规模复杂仿真系统所产生的数据具有数据流的特点。本文提出了基于数据流挖掘技术的仿真应用框架，设计了通用数据流挖掘成员，并以具体实例介绍了所设计的通用数据流关联规则挖掘成员。

数据流技术主要包括数据流管理技术和数据流挖掘技术。我们认为都可以将它们运用到仿真中。利用数据流管理系统管理仿真数据，以便为用户提供方便快捷的查询，可以作为进一步的研究方向。

参 考 文 献

[1] 鞠儒生，黄柯棣. 基于数据挖掘的 HLA 仿真系统测试与评估[J]. 系统工程与电子技术，2006，28(10)：1599-1602

[2] 张文明，薛青. 粗糙集方法在作战仿真数据挖掘中的应用. 系统仿真学报，2006，18(2)：179-181

[3] Morbitzer C, Strachan P, Simpson C. Application of Data Mining Techniques for Building Simulation Performance Prediction Analysis [C] // Proc. of Eighth IBPSA. Eindhoven, Nehterlands, August 2003: 11-14

[4] Liu Y, Liao W K, et al. On-Line Processing Model for Data Mining in Large Scientific Simulations [C] // Proc. of 7th Workshop on Mining Scientific and Engineering Datasets in Conjunction with SDM. Lake Buena Vista, Florida, USA, 2004: 31-38

[5] Bachinsky S, Tarbox G, Powell E. Data Collection in an HLA Environment [C] // 97S-SIW-059. 1997

[6] Abdulla G, Critchlow T. Simulation Data as Data Streams [C]. ACM SIGMOD Record, 2004, 33(1): 89-94

[7] Gaber M, Zaslavsky A, et al. Mining data streams: A Review [C]. ACM SIGMOD Record, 2005, 34(2): 18-26

[8] 陈彬，张国强，黄柯棣. 一种基于 HLA 的通用仿真数据收集方法[J]. 计算机仿真，2006，23(5)：127-130

[9] Chi Y, Wang H, Yu P S, et al. Moment: maintaining closed frequent itemsets over a stream sliding window [C] // Proc. of ICDM'04. Brighton, 2004: 59-66

[10] Ao F J, Du J, Huang K D, et al. An Efficient Algorithm for Mining Closed Frequent Itemsets in Data Streams // Proc. of CIT'08. Sydney, Australia, July 2008