第 35 卷 第 17 期 计 算 机 工 程 2009 年 9 月
Vol.35 No.17 Computer Engineering September 2009
·软件技术与数据库· 文章编号：1000—3428(2009)17—0058—02 文献标识码：A 中图分类号：TP312

一种交互式本体除错方法

吕 律
(广东商学院数学与计算科学学院，广州 510320)

摘 要：在本体设计过程中容易出现逻辑错误，利用现有本体除错工具难以诊断并修正此类错误。针对该问题提出一种新的交互式本体除错方法。通过分析 unsatisfiable concept 自动生成相关问题，基于用户对问题的回答自动修复错误本体。实验结果表明，该方法能利用少量问题实现修复目的。

关键词：网络本体语言；本体工程；除错

Interactive Method for Ontology Debugging

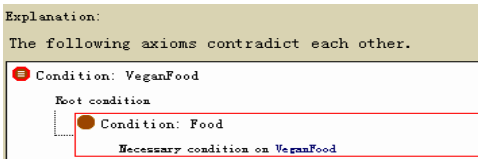
LV Lv
(School of Mathematics & Computing Science, Guangdong University of Business Studies, Guangzhou 510320)

【Abstract】 Logical errors is easy to appear in the process of Ontology designing. It is hard to diagnose and amend these errors by using existing tools for Ontology debugging. Aiming at this problem, this paper proposes an interactive method for Ontology debugging. By analyzing the unsatisfiable concept, it generates correlative questions and automatically repairs the Ontology based on users' reply. Experimental results show that this method can achieve the repair purpose by a few questions.

【Key words】 Web Ontology Language(OWL); Ontology engineering; debugging

1 概述

Protégé[1], Racer 能通过 unsatisfiable concept 发现本体中的错误，但需要本体设计人员手工完成对此类错误的修改。图 1 是现有本体除错工具对 badfood.owl[2]进行分析的结果。“VeganFood”是 badfood.owl 中的 unsatisfiable concept。图 1(a)是 Protégé 中 OWL Debugger 给出的结果，它指出 VeganFood 错误的原因是“Food”。图 1(b)是 SWOOP[3]给出的解释，它指出 VeganFood 错误的原因是“Meat”。



(a)Protégé的结果

(b)SWOOP的结果

图 1 关于“VeganFood”错误的解释

基于图 1 的解释，本体设计人员需要查找本体中的相关概念，分析错误原因(图 1(a)与图 1(b)的解释不一致)并进行除错工作。上述过程的实现难度较大，因此，本文提出一种交互式除错方法。在该方法中，本体设计人员只要对关于本体错误的问题回答“yes”，“no”或“unknown”，而无须主动分析并修正本体错误。

2 本体除错

本节简单介绍本文用到的描述逻辑和本体除错领域中较重要的概念。

本文使用描述逻辑中的 SHOIQ[4]，它由角色和概念 2 个部分组成。角色包括包含、层次、传递和逆关系。概念包括概念名 C，$C\cap D$，$C\cup D$，$\neg C$，$\forall R.C$，$\exists R.C$，$\leqslant nR.C$，$\geqslant nR.C$，其中，R 是角色。基于上述定义可以找到 3 种关系：(1)概念之间的关系，记为 $T1$(如 $C1\subseteq C2$)；(2)角色和概念之间的领域/范围关系，记为 $T2$(如“isHardWorking domain Person”)；(3)角色之间的关系，记为 $T3$(如 $R1\subseteq R2$)。

介绍 unsatisfiable concept 前先介绍冲突，文献[4]定义了以下 2 种冲突：(1) $\{C(x),\neg C(x)\}\subseteq ABox$，其中，$x$ 是概念 C 的一个实例；C 和 $\neg C$ 称为 conflict concept[4]。(2) $\{(\leqslant nR)(x),(>mR)(x)\}\subseteq ABox$，且 $n<m$。当一个概念包含冲突时，此概念即 unsatisfiable concept。

定义 1(MUPS)[5] $TBox\ T'\subseteq T$，C 是 T 中的一个概念，当 C 在 T'中是不可满足的(unsatisfiable)，而在 T'的任意一个子 $TBox$ 中都是可满足的，则称 T'是概念 C 在 $TBox$ 中的 MUPS。

定义 2(Root Class)[5] 当一个 unsatisfiable concept 不依赖(不包含)另一个 unsatisfiable concept 时，称此概念为 Root Class。

定义 3(Derived Class)[5] 当一个 unsatisfiable concept 依赖(包含)另一个 unsatisfiable concept 时，称此概念为 Derived Class。

3 交互式的本体除错

交互式本体除错的一个示例如下：

MUPS1: ax-1: $Koala\subseteq\exists isHardWorking.false$; ax-2: isHard Working domain Person; ax-3: $Person\subseteq\neg Marsupials$; ax-4: $Koala\subseteq Marsupials$。

作者简介：吕 律(1980－)，男，助教、硕士，主研方向：本体，知识工程

收稿日期：2009-04-07 **E-mail：**lulu@gdcc.edu.cn

MUPS2: bx-1: *Quokka* ⊆ *Marsupials*; bx-2: *Quokka* ⊆ ∃*isHardWorking.true*; bx-3: isHardWorking domain Person; bx-4: *Person* ⊆ ¬*Marsupials* 。

MUPS1 和 MUPS2 是本体 koala.owl[2]中的 2 个 MUPS，根据 MUPS 的定义，*Koala* 是 MUPS1 的 unsatisfiable concept，*Quokka* 是 MUPS2 的 unsatisfiable concept。因为 *Koala* 和 *Quokka* 都不依赖其他 unsatisfiable concept，所以 *Koala* 和 *Quokka* 都是 Root Class。本文方法先从错误本体中选择 Root Class 的 MUPS 进行分析，当有多个 Root Class 时，选择包含较多 Derived Class 的 Root Class 进行分析。在上述示例中，KoalaWithPhD 是 *Koala* 的 Derived Class，根据定义 2 和定义 3，当 Root Class 消失，Derived Class 也消失。下文讨论如何消除 Root Class。

3.1 MUPS 树

在本体除错领域，如何根据 unsatisfiable concept 计算其相关的 MUPS 已被广泛研究，本文采用文献[5]的方法。要找出错误必须分析 MUPS，为了方便分析，本文提出一种树结构(MUPS 树)。

定义 4(根节点) 设 UC 是 MUPS($m1$)的 unsatisfiable concept，$GC1$ 和 $GC2$ 是一般性概念[4]，则根节点(ax(UC))是满足以下条件的公理[4]:

(1)若 $UC \subseteq GC1 \in m1$, $GC2 \subseteq \neg UC \in m1$ ，则 $UC \subseteq GC1 \cap \neg GC2$ 。

(2)若 $UC \subseteq GC1 \in m1$, $GC2 \equiv \neg UC \in m1$ ，则把“≡”作为“⊆”处理，$UC \subseteq GC1 \cap \neg GC2$ 。

定义 5 当满足以下条件时，称公理 ax1 依赖公理 ax2，记为 AD(ax1, ax2):

(1)ax1 和 ax2 都是 $T1$，$GC1$ 是 ax1 的右边且 $GC1$ 是 ax2 的左边。

(2)ax1 是 $T1$，ax2 是 $T2$，R 是 ax1 和 ax2 中的角色。

(3)ax1 是 $T2$，ax2 是 $T1$，C 是 ax1 和 ax2 中的概念。

定义 6 MUPS 树以 ax(UC)为根节点，子节点(child)和父节点(parent)满足 AD(parent, child)，叶子节点包含 conflict concept 的树。

根据定义 4~定义 6，*Koala* 的 MUPS 树如图 2 所示。根节点(1)由 ax-1 和 ax-2 合并而得。ax-4 中的 *Marsupial* 即 conflict concept。

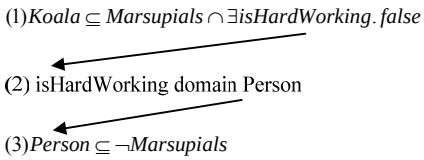


图 2 *Koala* 的 MUPS 树

3.2 问题的自动生成和本体的修复

根据 MUPS 生成问题的直接方法是对 MUPS 的每条公理都向用户询问是否正确，该方法将产生很多无关问题。

要生成问题就要考虑 SHOIQ 中的 2 种冲突。第(2)种冲突 $\{(\leqslant nR)(x),(>mR)(x)\} \subseteq ABox$ 较容易解决。例如，若用户对问题(is C1 the domain of “<n.R”)的回答是“yes”，则删除 conflict concept(“>n.R”)；如果用户回答“no”，则删除“<n.R”；若用户回答“unknown”，则把 2 个 conflict concepts 都删除。

下文主要讨论第(1)种冲突 $\{C(x),\neg C(x)\} \subseteq ABox$ 。对本文方法，即基于 MUPS 树生成问题的规则描述如下:

Rule-1：如果(1)$C_1 \subseteq R.C_2,\cdots, C_{n-1} \subseteq R.C_n$, $C_1 \neq$ UC; $C \subseteq R_1.C_1$ 或 C_1 is the domain/range of R_1 ; $R_1 \neq R$， R 具有传递性，或(2)$C_1 \subseteq R.C_n$, C_1 是 UC，则问题为“is R the role of C_1 and C_n?”。如果回答是“yes“，则新的公理是 $C_1 \subseteq R.C_n$。如果回答是“no”，则新的公理是 $C_1 \subseteq \neg R.C_n$ 如果回答是“unknown”且 $C_1 \subseteq R.C_n \in$KB，则删除此公理。

Rule-2：如果将 R 从 Rule-1 中删除，则问题是“is C_1 the subset of C_n?”。

Rule-3：如果(1)C_1 is the domain/range of R_1, $C_1 \subseteq C_2,\cdots, C_{n-1} \subseteq C_n$, $C_1 \neq$UC，或(2)C_n is the domain/range of R_1 , C_n is UC，则问题为“is C_n the domain/range of R_1?”。如果回答是“yes”，则新的公理是“C_n the domain./range of R_1”。如果回答是“no”，则新的公理是“C_n the domain./range of $\neg R_1$”。如果回答是“unknown”且“C_n the domain./range of R_1” ∈KB，则删除此公理。

Rule-4：如果 $GC_1 \equiv GC_2$ 且该公理紧跟“GC_1”，或 GC_1 是 UC，则把“$GC_1 \equiv GC_2$”作为“$GC_1 \subseteq GC_2$”处理。如果 $GC_1 \equiv GC_2$, $GC_1 \neq$ UC，且该公理紧跟“$\neg GC_1$”，则把“$GC_1 \equiv GC_2$”作为“$\neg GC_1 \subseteq \neg GC_2$”处理。

Rule-1 利用角色的传递性生成问题。Rule-2 考虑 $C1 \subseteq C2$ 形式的公理。虽然公理中不含角色，但“⊆”本身具备传递性。Rule-3 考虑 Domain/Range 公理。Rule-4 处理 MUPS 树中的“≡”，在问题生成过程中，它被作为“⊆”处理。公理 ax1 ∈ KB 表示该公理存在于本体中。

从 MUPS 树的叶子开始，通过上述规则生成问题。基于用户的回复生成新的公理并更新本体。例如，基于 Rule-3 和图 2，将生成以下的问题：“is “not Marsupials” the domain of “isHardWorking”?”。如果用户回答“yes”，则用新生成的公理更新本体，从而使本体中 Marsupials 的子概念或等价概念都不包含角色 isHardWorking。本体更新的基本思想是找出 Marsupial 的子概念或等价概念中，与新公理冲突的概念。在上述示例中，ax-1 和 bx-2 与新的公理冲突。从本体中删除与新公理冲突的概念前，需要了解 *Koala* 和 *Quakka* 是否是 Marsupials 的子概念，因此，系统生成第 2 个问题“is *Quakka* the subset of Marsupials?”和第 3 个问题“is *Koala* the subset of Marsupials?”，如图 3 所示。在示例中，当 *Koala* 的错误修正后，*Quokka* 的错误也被修正。

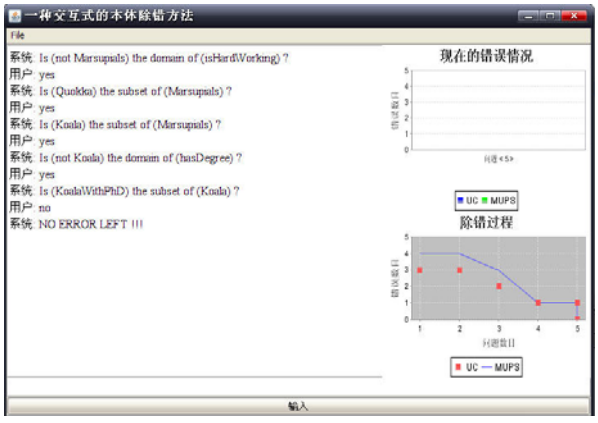


图 3 本体 *Koala* 的除错过程

如果用户回答“no”，在本体被更新后，若错误依然存在，则可以把错误定位到用于生成问题的路径。用于查找错误公理的方法与二分法类似。设该路径的最顶公理为 AXt，中间公理为 AXm，最底公理为 AXb。以 AXm 和 AXb 作为起始点生成问题。如果用户回答“yes”，则 AXm 成为新的 AXb，

(下转第 63 页)