基于 $ALCQ(D)$ 的 CBR 事例修正算法研究

黄金龙 古天龙 孙晋永 徐周波
(桂林电子科技大学广西可信软件重点实验室 桂林 541004)

摘 要 CBR(基于事例推理)是人工智能领域的一个分支,它克服了知识获取的瓶颈问题,事例修正是CBR的关键步骤。以ALC为代表的描述逻辑已被充分应用到CBR中,但目前在基于描述逻辑的CBR中还没有比较有效的算法来判断检索到的相似事例是否需要修正和如何进行修正。$ALCQ(D)$是在ALC的基础上引入定性数量约束Q和有型域D得到的。提出的算法用$ALCQ(D)$概念来描述CBR源事例和目标事例,先假定检索到的相似事例能够解决目标问题,即假定目标事例和相似事例同时满足知识库,但这样可能会与知识库产生冲突;接着使用冲突检测机制来查找相似事例概念描述中导致冲突的概念;最后使用概念替换规则在TBox本体库中检索该概念的最相似概念去替换它自己。研究表明,该算法具有界限性、可靠性和完备性。通过一个实例对其进行检验,结果表明,该算法可以准确修正检索到的相似事例,解决目标问题。

关键词 基于事例推理,描述逻辑,事例修正,定性数量约束,有型域
中图法分类号 TP301 **文献标识码** A **DOI** 10.11896/j.issn.1002-137X.2014.11.046

Research on CBR Case Adaptation Based on $ALCQ(D)$

HUANG Jin-long GU Tian-long SUN Jin-yong XU Zhou-bo
(Guangxi Key Laboratory of Trusted Software, Guilin University of Electronic Technology, Guilin 541004, China)

Abstract CBR(Case-based Reasoning) is a branch of artificial intelligence, which overcomes the bottleneck of knowledge acquisition. Case adaptation is a key step in CBR. Description logic ALC has been fully applied to the CBR, but now no alogirhtm is more effective to determine whether a retrieved similar case needs to be modified and how to fix based on description logic. ALC becomes $ALCQ(D)$ as it introduces a qualitative quantity Q and a type constraint domain D. The algorithm in this paper uses $ALCQ(D)$ concept to represent the source case and target case. Firstly, it presupposes that the retrieved source example can solve the problem of target, which means the target and source case examples both satisfy KB(knowledge base), but this may lead to inconsistent with KB. Then according to the conflict detection in the algorithm, it finds the concepts which lead to inconsistent in source concept instances and finally uses concept of replacement rules defined in this article to retrieve the most similar concepts to the inconsistent concept in ontology repository for replacing itself. Studies show that this algorithm has boundaries, reliability and completeness. This paper used an example to illustrate this algorithm. The results show that it can revise the similar case to solve the target problem.

Keywords Case-based reasoning, Description logic, Case adaptation, Qualified number restrictions, Concrete domain

1 引言

CBR(Case-Based Reasoning)是近来人工智能领域研究的一个热点。与专家系统相比,CBR克服了规则框架的限制[1],充分利用了过去处理问题的经验。它模拟人类普遍的思维模式[2]——遇到问题首先回忆以前自己或别人解决类似问题的方法。CBR的研究可以追溯到20世纪70年代末,Roger Schank在动态存储器技术[3]的研究中发现,过去的一些经验对问题的解决和学习起着重要的作用,并根据这个思想做了大量研究,最后提出了CBR方法。Kolodner领导的小组于1983年在耶鲁大学开发出第一个CBR系统CYRUS[4]。Aamodt和Plaza将CBR工作的过程归纳为4R[5]模型,即retrieve(事例检索)、reuse(事例重用)、revise(事例修正)、retain(事例保存)。事例检索是从事例库中检索与当前问题事例相似的事例。事例重用是将事例检索得到的结果用来解决当前问题。事例修正是在事例重用失败时,需要对事例检索得到的结果进行修改的过程。目前事例检索的研究有很多,并且已经取得了很多成就,已有大量的成熟算法被提出。而事例

到稿日期:2014-01-16 返修日期:2014-04-11 本文受国家自然科学基金(61100025,61363030),桂林电子科技大学研究生教育创新计划资助项目(XJYC2012016)资助。

黄金龙(1988—),男,硕士生,主要研究方向为知识表示与推理、形式化方法、CBR推理;古天龙(1964—),男,教授,博士生导师,CCF高级会员,主要研究方向为符号计算、形式化方法等;孙晋永(1978—),男,博士生,CCF会员,主要研究方向为知识表示与推理、CBR推理等;徐周波(1976—),女,博士,副教授,主要研究方向为符号计算、智能规划、约束满足问题求解等。

修正是CBR研究以来的一大难点，一直没有形成真正有效且系统的方法，所以很多商用CBR系统缺少事例修正这一功能。

文献[6]中提出了一种领域独立的算法，其重点是源、目标事例的问题特征的差异性描述。通过准确的差异性描述，以及源事例问题和它的解答之间的关联性，计算出解答目标事例问题的各个属性值。文献[7]中充分利用归纳技术，由两个事例为一组进行比较得出修正知识，然后应用这些修正知识去修正。文献[8]介绍了一种混合策略的修正方法，即把基于规则推理和基于事例推理结合，当需要进行事例修正时首先检索以前修正成功的事例，如果没有检索到合适的修正成功的事例，再用规则推理去修改。文献[9]提出了一种自省的、从事例库获得修正知识的方法。首先随机从事例库选择一个事例A，然后从事例库其他事例中选择一定数目并且与A相似的事例，将其与A进行比较，充分描述出因问题差异而导致结果不同的函数性变化，多次迭代形成一个修正知识库CBA。在事例修正时，把源事例和目标事例的问题的差异作为索引来检索CBA库，获得修正方法。文献[10]提出了一种基于FCA(形式概念分析)的半自动获取修正知识的方法，该方法以逻辑概念分析的形式提取修正条件，增强了事例修正功能，它已经在COBRA系统中实现。文献[11]中将CBR方法用于可满足性问题(SAT)，在修正阶段，文中将已存在的适合的修正方法都拿来作比较，并经过具体的实例验证评价。文献[12]将CBR用到环境紧急情况预报系统中，为了克服事例修正的困难，作者在修正过程中采用了遗传算法，首先用框架法，根据属性来表示一个事例，然后用改进的遗传算法来完成修正步骤。文献[13]提出了一种贝叶斯修正指导检索的方法，具体做法是用贝叶斯网络来搭建事例库，并且将事例检索与事例修正联系在一起，在检索步骤就要考虑是否易于修正。文献[14]提出了一种用于数值问题的修正方法，新问题的解决方法依赖于相似事例与目标事例之间的关联，事例修正就好像微积分一样，整体的不同是由细微的不同堆积而成的，所以要完成整个修正，首先需要对每个细小的地方进行修正。

本文的事例修正方法建立在由描述逻辑$ALCQ(D)$表示事例和领域知识的前提下。在众多知识表示的形式化方法中，描述逻辑在十多年来受到人们的特别关注，主要原因在于：i)它们有清晰的模型-理论机制；ii)很适合于通过概念分类学来表示应用领域；iii)提供了很有用的推理服务。$ALCQ(D)$也是一种描述逻辑，具有很强的表达能力和可判定性。CBR的事例用$ALCQ(D)$-概念表示，与以前的属性-值对的事例表示方法相比较，它能表示的事例范围更广，表达的事例更细腻，所以适用的领域更广泛。而$ALCQ(D)$的概念和公式的可满足性判定定理可以用于CBR的事例重用。本文第2节详细介绍$ALCQ(D)$，包括它的语法、语义以及推理；第3节重点介绍基于$ALCQ(D)$的事例修正算法，详细地给出算法是如何判断事例是否需要修正以及如何修正的；最后是结论和以后研究工作方向的预测。

2 描述逻辑 $ALCQ(D)$

描述逻辑(DL)是一族用于知识表示的逻辑语言[15]，主要用于描述概念分类和概念间的关系，通过定义应用领域的概念及其结构关系，刻画领域内的个体信息。它的框架一般由一组概念名和角色名组成，借助概念构子递归定义概念描述，构子决定了DL的表达能力。最基础的DL是ALC[16](attributive concept description language with complements)，由合取、析取、否定、存在性限定和值限定[17]这5种构子构成。在ALC的基础上可以添加一些构子使描述概念的能力更强，比如函数性约束、定性数量约束，可以增强比如传递性[18]角色、逆角色等角色的表达能力。ALC扩展之后就有了SHIN，SHIF，SHIQ[19-22]。

Tableau方法是一种一阶逻辑的证明论。一般地，描述逻辑能够构造出可靠完全的Tableau算法，用于判定系统的推理问题。Tableau算法最早由Schmidt-Schau和Smolka为了检验ALC概念的可满足性而提出，被广泛应用于各种描述逻辑中，判定概念的可满足性或概念间的包含关系[23]。

本文用描述逻辑$ALCQ(D)$来刻画事例库中的事例概念和构造其相应的Tableau算法进行推理。$ALCQ(D)$是在ALC的基础上添加定性数量构子Q和有型域构子D得到的。添加Q构子之后，描述逻辑的概念就多了两种形式：$\geqslant nR.C$，$\leqslant nR.C$，这两种概念用来约束角色成员的数量和所属概念；添加D算子使得ALC包括了数值、字符串、时间等这类有型对象[24,25]。如下例所示：

$\mathrm{Woman}=\mathrm{Person}\sqcap\mathrm{Female}$

$\mathrm{Man}=\mathrm{Person}\sqcap\neg\mathrm{Woman}$

$\mathrm{Mother}=\mathrm{Woman}\sqcap\exists\,\mathrm{hasChild}.\mathrm{Person}$

$\mathrm{Father}=\mathrm{Man}\sqcap\exists\,\mathrm{hasChild}.\mathrm{Person}$

$\mathrm{Parent}=\mathrm{Mother}\sqcup\mathrm{Father}$

$\mathrm{motherWithoutDaughter}=\mathrm{Mother}\sqcap\forall\,\mathrm{hasChild}.\neg\mathrm{Woman}$

$\mathrm{motherWithManyDaughter}=\mathrm{Mother}\sqcap\geqslant 3\mathrm{hasChild}.\mathrm{Person}$

$\mathrm{happyMother}=\mathrm{Mother}\sqcap\exists\,\mathrm{hasChild}.\mathrm{Woman}\sqcap\exists\,\mathrm{monthWage}.\geqslant 5000$

$\mathrm{Woman(Allen)}$，$\mathrm{Man(Paul)}$，$\mathrm{hasChild(Allen,Paul)}$

一个概念可以看成解释域的一个子集。上面等式左边是一些被定义的概念名，等式右边是原子概念或者定义概念，如Woman、Father、Person、Female等都是概念。一个角色可以看成解释域上的二元关系，是指两个概念之间的关系，上式中的hasChild即为一个角色。个体是解释域中的所有元素，概念断言是满足该概念的所有个体集合，如Woman(Allen)。角色断言是满足该角色的个体对的集合，如hasChild(Allen，Paul)。其中，$\geqslant$3hasChild.Person是一个定性数量约束，它定义概念“有很多孩子的妈妈”是至少有3个孩子的妈妈，而这个概念仅通过ALC是不能够准确表达的。$\exists$monthWage.$\geqslant$5000表达的是月工资超过5000元，这是概念happy-Mother的一个典型的有型域约束。抽象个体通过特征式映射为有型域上的值，这些值再被有型域上的谓词公式约束。上式中，monthWage是特征式，$\geqslant$5000是一个一元谓词。

以下是它的具体语法和语义。

2.1 语法

定义1 有型域D是一个二元组$(\Delta^D, pred(D))$，其中Δ^D是有型论域，$pred(D)$是谓词集合。任意n元谓词$P\in pred(D)$是论域上的n元关系，即$P^D\in(\Delta^D)^n$。

定义2 设N_c是概念名的集合，N_R是简单角色名集合，N_f为特征式集合，且彼此不相交。$ALCQ(D)$的角色集合是$N_R\cup N_f$，特征链是指特征式的合成$f_1\cdot f_2\cdot\cdots\cdot f_k$。

定义$ALCQ(D)$概念集合为满足下列条件的最小集合[26,27]：

(1)若概念名$C\in N_C$，则C是$ALCQ(D)$的概念。

(2)若C,D是$ALCQ(D)$概念，$R\in N_R$，则$C\sqcap D$，$C\sqcup D$，

$\neg C$，$\exists R.C$，$\forall R.C$ 都是 $ALCQ(D)$ 的概念。

(3)若 $R\in N_R$，$C\in N_C$，则 $\geqslant nR.C$，$\leqslant nR.C$ 是 $ALCQ(D)$ 的概念。

(4)若 $u_1\cdot u_2\cdot\cdots\cdot u_n$ 是特征链，$P\in pred(D)$ 是 n 元谓词，则 $\exists u_1\cdot u_2\cdot\cdots\cdot u_n.P$ 和 $\forall u_1\cdot u_2\cdot\cdots\cdot u_n.P$ 是 $ALCQ(D)$ 的概念。

如果 C、D 是两个概念，R 是简单角色，a、b 是两个个体，在 $ALCQ(D)$ 中会出现 4 类公式：1)$C\sqsubseteq D$，是一般概念包含公理；2)$C\equiv D$，称为概念定义式；3)$C(a)$，称为概念断言；4)$R(a,b)$，称为角色断言。其中 1)和 2)两类公式用来描述知识库的一个框架，所有概念包含公理和概念定义组成的集合称为 TBox；3)和 4)都是断言公式，称为个体断言，所有个体断言的集合称为 ABox。

2.2 语义

定义 3 $ALCQ(D)$ 的解释形如 $I=(\Delta^I,\Delta^D,\cdot^I)$，其中，$\Delta^I$ 是由个体组成的解释域，Δ^D 是有型论域，$\cdot^I$ 是一个解释函数。集合 Δ^D 与 Δ^I 不相交，函数 $\cdot^I$ 将每个概念映射为 Δ^I 的一个子集，如 C^I 表示满足概念 C 的所有个体集合。将每个角色映射为 $\Delta^I\times\Delta^I$ 的一个子集，如 R^I 表示满足关系 R 的所有个体对的集合。将每个特征式映射为 $\Delta^I\times\Delta^D$ 的一个子集，如 $\langle x,a\rangle\in u^I$ 表示抽象个体 x 与有型域个体 a 满足特征式 u 的约束。$ALCQ(D)$ 的概念和公式满足下列语义：

(1)$(C\sqcap D)^I=C^I\cap D^I$

(2)$(C\sqcup D)^I=C^I\cup D^I$

(3)$(\neg C)^I=\Delta^I\backslash C^I$

(4)$(\forall R.C)^I=\{x\in\Delta^I\mid$对任意的 $y\in\Delta^I$，如果 $\langle x,y\rangle\in R^I$，那么 $y\in C^I\}$

(5)$(\exists R.C)^I=\{x\in\Delta^I\mid$存在一 $y\in\Delta^I$，满足 $\langle x,y\rangle\in R^I$，而且 $y\in C^I\}$

(6)$(\geqslant nR.C)^I=\{x\in\Delta^I\mid\#\{y\in\Delta^I\mid\langle x,y\rangle\in R^I$ 且 $y\in C^I\}\geqslant n\}$

(7)$(\leqslant nR.C)^I=\{x\in\Delta^I\mid\#\{y\in\Delta^I\mid\langle x,y\rangle\in R^I$ 且 $y\in C^I\}\leqslant n\}$

(8)$(\exists u_1\cdot u_2\cdot\cdots\cdot u_n.P)^I=\{x\in\Delta^I\mid$存在 $a_1,\cdots,a_n\in\Delta^D$，满足 $\langle x,a_1\rangle\in u_1^I\wedge\cdots\wedge\langle x,a_n\rangle\in u_n^I$，且 $(a_1,\cdots,a_n)\in P^D\}$

(9)$(\forall u_1\cdot u_2\cdot\cdots\cdot u_n.P)^I=\{x\in\Delta^I\mid$对于所有的 $a_1,\cdots,a_n\in\Delta^D$，如果满足 $\langle x,a_1\rangle\in u_1^I\wedge\cdots\wedge\langle x,a_n\rangle\in u_n^I$，那么 $(a_1,\cdots,a_n)\in P^D\}$

(10)$I\models C(p)$ 当且仅当 $p^I\in C^I$

(11)$I\models R(p,q)$ 当且仅当 $(p^I,q^I)\in R^I$

(12)$I\models\neg\phi$ 当且仅当 $I\not\models\phi$

(13)$I\models\phi\wedge\psi$ 当且仅当 $I\models\phi$ 并且 $I\models\psi$

由于不用表达太复杂的概念，因此对有型域 D 作了简化，只考虑一个特征。这个特征可称为有型角色，对应的谓词公式是一元谓词，按语义 Web 的惯例，称 D 为数据类型。考虑的对象为：存在性数据类型 $\exists F.d$ 和任意性数据类型 $\forall F.d$，例如描述概念“主频超过 3.0 的电脑”为 Computer $\sqcap\exists$ hasBasicFreq.$\geqslant$3.0，其中 basicfreq 是有型角色，也即是特征式。

定义 4 (1)称概念 C 是可满足的当且仅当存在解释 I 使得 $C^I\neq\emptyset$，I 称为 C 的模型。(2)称概念 D 包含概念 C 当且仅当对于所有解释 I 都有 $C^I\subseteq D^I$ 成立，记作 $C\sqsubseteq D$。(3)称概念 C、D 是等价的当且仅当 $C\sqsubseteq D$，$D\sqsubseteq C$，记作 $C\equiv D$。(4)对于一个解释 I，称个体 $x(x\in\Delta^I)$ 是概念 C 的实例当且仅当 $x\in C^I$。

2.3 推理

除了很好的知识描述能力外，描述逻辑具还有良好的推理能力。$ALCQ(D)$ 的主要推理问题可分为 3 类：1)概念的可满足性；2)知识库的一致性(ABox 相对于 TBox 的一致性)；3)公式的可满足性。

定义 5 令 T 为一般 TBox，C 为概念，则称概念 C 相对于 T 是可满足的，当且仅当存在某个语义解释 I 使得 $I\models T$ 且 $C^I\neq\emptyset$。

定义 6 令 T 为一般 TBox，A 为 ABox，则称知识库 $K=(T,A)$ 是一致的，当且仅当存在某个语义解释 I 使得 $I\models T$ 且 $I\models A$。

定义 7 令 T 为一般 TBox，φ 为一公式，则称公式 φ 相对于 T 是可满足的，当且仅当存在某个语义解释 I 使得 $I\models T$ 且 $I\models\varphi$。

$ALCQ(D)$ 中的典型公式有两类，即概念包含和个体断言，对应的公式可满足性判定可分为概念包含检测和 ABox 一致性检测。概念包含检测是在给定的知识库中，判断某个概念是否是另一个概念的子集，这一思想对组织概念的分层非常有帮助。ABox 的一致性检测是判断 ABox 中的所有个体断言在 TBox 下是否仍成立，它可用于矛盾的检测。本文就是根据 ABox 的一致性检测来检验目标事例中存在的矛盾。其中概念包含的检测也可转换成 ABox 的一致性检测。即 $KB\models C\sqsubseteq D$ iff $\{(C\sqcap\neg D)(a)\}$ 是不满足的。

3 基于 $ALCQ(D)$ 的事例修正算法

CBR 的工作流程一般可分为事例检索、事例复用、事例修正和事例保存 4 个阶段。其中事例复用是检查事例检索阶段检索到的相似事例能否解决当前新问题，事例修正是在复用结果不满意的前提下对事例进行的部分或完全修改。这两个过程可能重复多次，因此事例复用和事例修正的关系非常密切。本文综合考虑这两个过程，提出了事例修正算法。该算法首先进行事例重用工作，假设相似事例的解决方法和目标事例问题匹配[28]，那么它们应该同时满足给定的知识库；然后通过 $ALCQ(D)$ 的概念可满足性判定定理来检测相似事例和目标事例是否同时匹配知识库，如果不匹配，则可以检测出相似事例概念中引起冲突的概念。接着在本体库检索该概念的最相近概念进行替换操作，完成修改。

3.1 概念图

算法的前提是给定领域知识库 DK，本文的 DK 完全是由 TBox 构成，主要包括概念定义式和概念包含式，其中概念定义式定义了各种复杂概念，概念包含式通过包含关系构筑了一个领域本体库。相似事例被描述成概念 C_s，目标事例被描述成概念 C_t。

一般的 Tableau 算法是为了判定概念的可满足性，最后的结果是满足或者不满足这两种结论。而本文算法的目的是不仅要判断概念是否满足的问题，而且要在概念不满足时，找出产生不满足的子概念；而且如果 $\alpha\models\beta$，且 β 需要被移除或修改，那么 α 也要移除或修改。所以本文算法在 Tableau 算法的基础上做了改进，不仅能判断 β 概念是否可满足，而且在 β 概念不满足时，能够找出 α 概念。

称改进后的结构为概念图 G，G 中每个结点标记里有且

仅有一个概念，结点与结点之间是有向边，方向是指向概念扩展的方向，$Nodes(G)$表示G中的结点集合，$Edges(G)$表示G中结点之间边的集合，$L(x)$表示结点x的所属概念的标注集合，$F(x,y)$表示由x到y用到的扩展规则。设G为概念图，x为根结点，具体规则如下。

1)⊓规则：如果结点x中的概念出现$C_1 \sqcap \cdots \sqcap C_P$的形式，那么施展⊓规则。在$G$中加入$p$个结点$y_1 \cdots y_p$，每个结点与$x$结点引入一条有向边$\langle x, y_k \rangle (1 \leqslant k \leqslant p)$，$F$是该有向边的标记函数，函数值为有向边的名称，$L$是结点的标记函数，函数值为该结点满足的概念，扩展后的概念图为G'，G'满足如下内容：

$Nodes(G') = Nodes(G) \cup \{y_k \mid 1 \leqslant k \leqslant p\}$

$L(y_k) = C_k (1 \leqslant k \leqslant p)$

$Edges(G') = Edges(G) \cup \{\langle x, y_k \rangle \mid 1 \leqslant k \leqslant p\}$

$F(\langle x, y_k \rangle) = partOf$

2)⊔规则：如果结点x的概念出现$C_1 \sqcup \cdots \sqcup C_P$的形式，那么施展⊔规则。添加$p$个新概念图$G_1 \cdots G_p$，每个新概念图都是原概念图$G$的复制，并且对每个概念图$G_k (1 \leqslant k \leqslant p)$加入对应结点$y_k (1 \leqslant k \leqslant p)$，结点$x$与结点$y_k$引入一条有向边$\langle x, y_k \rangle$，$F$是该有向边的标记函数，函数值为有向边的名称，$L$是结点的标记函数，函数值为该结点满足的概念，那么对于G_k都满足：

$Nodes(G_k) = Nodes(G) \cup \{y_k\}$

$L(y_k) = C_k (1 \leqslant k \leqslant p)$

$Edges(G_k) = Edges(G) \cup \{\langle x, y_k \rangle\}$

$F(\langle x, y_k \rangle) = partOf$

3)$\exists R$规则：如果结点x的概念出现$\exists R.C$的形式，那么施展$\exists R$规则。在G中加入一个y，并且在结点x与y之间引入一条有向边$\langle x, y \rangle$，在G中加入另一个结点z，并且在结点x与z之间引入一条有向边$\langle x, z \rangle$，F是该有向边的标记函数，函数值为有向边的名称，L是结点的标记函数，函数值为该结点满足的概念，扩展后的概念图为G'，满足以下内容：

$Nodes(G') = Nodes(G) \cup \{y, z\}$

$L(y) = C, L(z) = 1$

$Edges(G') = Edges(G) \cup \{\langle x, y \rangle, \langle x, z \rangle\}$

$F(\langle x, y \rangle) = \exists R, F(\langle x, z \rangle) = \geqslant R$

4)$\forall R$规则：如果结点x的概念出现$\forall R.C$的形式，那么施展$\forall R$规则。

如果x不存在一条$F = \forall R$的有向边，那么在G中加入一个结点y，并且在结点x与y之间引入一条有向边$\langle x, y \rangle$，F是该有向边的标记函数，函数值为有向边的名称，L是结点的标记函数，函数值为该结点满足的概念，扩展后的概念图为G'，满足以下内容：

$Nodes(G') = Nodes(G) \cup \{y\}$

$L(y) = C$

$Edges(G') = Edges(G) \cup \{\langle x, y \rangle\}$

$F(\langle x, y \rangle) = \forall R$

如果x与某个结点y之间存在一条$F = \forall R$的有向边，那么直接在结点y的标记函数L中加入概念C，扩展后的概念图为G'，y结点的标记函数为L'，满足以下内容：

$Nodes(G') = Nodes(G)$

$L'(y) = L(y) \cup \{C\}$

$Edges(G') = Edges(G)$

$F(\langle x, y \rangle) = \forall R$

5)$\exists F$规则：如果结点x的概念是$\exists F.d$的形式，那么施展$\exists F$规则。在G中加入一个结点y，并且在结点x与y之间引入一条有向边$\langle x, y \rangle$，F是该有向边的标记函数，函数值为有向边的名称，L是结点的标记函数，函数值为该结点满足的概念，扩展后的概念图为G'，满足以下内容：

$Nodes(G') = Nodes(G) \cup \{y\}$

$L(y) = d$

$Edges(G') = Edges(G) \cup \{\langle x, y \rangle\}$

$F(\langle x, y \rangle) = \exists F$

6)$\forall F$规则：如果结点x的概念出现$\forall F.d$的形式，那么施展$\forall F$规则。

如果x不存在一条$F = \forall F$的有向边，那么在G中加入一个结点y，并且在结点x与y之间引入一条有向边$\langle x, y \rangle$，F是该有向边的标记函数，函数值为有向边的名称，L是结点的标记函数，函数值为该结点满足的概念，扩展后的概念图为G'，满足以下内容：

$Nodes(G') = Nodes(G) \cup \{y\}$

$L(y) = d$

$Edges(G') = Edges(G) \cup \{\langle x, y \rangle\}$

$F(\langle x, y \rangle) = \forall F$

如果x与某个结点y之间存在一条$F = \forall F$的有向边，那么直接在结点y的标记函数L中加入概念d，扩展后的概念图为G'，y结点的标记函数为L'，满足以下内容：

$Nodes(G') = Nodes(G)$

$L'(y) = L(y) \cup \{d\}$

$Edges(G') = Edges(G)$

$F(\langle x, y \rangle) = \forall F$

7)$\geqslant n$规则：如果结点x的概念包含$(\geqslant nR.C)$的形式，那么施展$\geqslant n$规则。在G中加入一个结点y，并且在结点x与y之间引入一条有向边$\langle x, y \rangle$，如果$n \geqslant 1$，则在G中加入另一个结点z，并且在结点x与z之间引入一条有向边$\langle x, z \rangle$，F是该有向边的标记函数，函数值为有向边的名称，L是结点的标记函数，函数值为该结点满足的概念，扩展后的概念图为G'，满足以下内容：

如果$n < 1$，则

$Nodes(G') = Nodes(G) \cup \{y\}$

$L(y) = \geqslant nC$

$Edges(G') = Edges(G) \cup \{\langle x, y \rangle\}$

$F(\langle x, y \rangle) = \geqslant R$

如果$n \geqslant 1$，则

$Nodes(G') = Nodes(G) \cup \{y, z\}$

$L(y) = \geqslant nC, L(z) = C$

$Edges(G') = Edges(G) \cup \{\langle x, y \rangle, \langle x, z \rangle\}$

$F(\langle x, y \rangle) = \geqslant R, F(\langle x, z \rangle) = \exists R$

8)$\leqslant n$规则：如果结点x的概念包含$(\leqslant nR.C)$的形式，那么施展$\leqslant n$规则。在G中加入一个结点y，并且在结点x与y之间引入一条有向边$\langle x, y \rangle$，F是该有向边的标记函数，函数值为有向边的名称，L是结点的标记函数，函数值为该结点满足的概念，扩展后的概念图为G'，满足以下内容：

$Nodes(G') = Nodes(G) \cup \{y\}$

$L(y) = \leqslant nC$

$Edges(G') = Edges(G) \cup \{\langle x, y \rangle\}$

$F(\langle x, y \rangle) = \leqslant R$

定义 8(完全概念图)　概念图中的每个叶子结点的标记

都有且只有一个原子概念或者数学表达式。

3.2 算法与实例

一个顾客计划购置一台电脑。他的要求是：独立显卡，主频≥2.5GHz，内存条数目≥2 根，并且预算≤3000 元。现在把他的需求用概念 Tgt 表示，还有一些根据他的需求检索到的一些以前配置电脑成功的方案，用概念 $Source_i$ 将电脑领域的知识表示成本体库。要求参照这些配置成功的案例和顾客提的要求，完成一个合适的配置方案。其中整个事例中用到的概念名集合为{Computer, independentCard, Mainboard, ASUS, ASUS1, ASUS2, Gigabyte, CPU, AMD1, AMD2, AMD3, Intel1, Intel2, Intel3, M1, M2, M3, M4}，角色名集合为{hasCard, hasBaiscFreq, hasMemory, spend, hasMainboard, hasCpu}，完整的概念表示如下：

目标事例：

Tgt=Computer⊓∀ hasMainboard.（Mainboard⊓∀ hasCard. independentCard)⊓∀ hasCpu.（Cpu⊓∀ hasBasicFreq.≥2.5G⊓∀ spend.≤3000Y)⊓⊓≥2hasMemory[1)]

源事例库：

$Source_1$ =Computer⊓∃ hasMainboard. Gigabyte⊓∃ hasCpu. AMD⊓M1

$Source_2$ =Computer⊓∃ hasMainboard. Gigabyte⊓∃ hasCpu. AMD⊓∃ spend. =4500Y⊓=4hasMemory

$Source_3$ =Computer⊓∃ hasMainboard. Gigabyte⊓∃ hasCpu. Intel⊓∃ spend. =3500Y⊓=4hasMemory

$Source_4$ = Computer ⊓ hasMainboard. ASUS ⊓ hasCpu. AMD⊓ hasCard. →independentCard

本体库：

Mainboard=ASUS⊔ Gigabyte[2)]

ASUS=ASUS1⊔ ASUS2[3)]

ASUS1=∃ hasCard. →independentCard[4)]

ASUS2=∃ hasCard. independentCard[5)]

Gigabyte=∃ hasCard. →independentCard

Cpu=Intel⊔ AMD

AMD=AMD1⊔ AMD2⊔ AMD3

AMD1=∃ spend. =2000Y⊓∃ hasBasicFreq. =2.0G[6)]

AMD2=∃ spend. =3000Y⊓∃ hasBasicFreq. =2.5G

AMD3=∃ spend. =5000Y⊓∃ hasBasicFreq. =3.0G

Intel=Intel1⊔ Intel2⊔ Intel3

Intel1=∃ spend. =1000Y⊓∃ hasBasicFreq. =2.0G

Intel2=∃ spend. =2000Y⊓∃ hasBasicFreq. =2.5G

Intel3=∃ spend. =3000Y⊓∃ hasBasicFreq. =3.0G

M1=Computer⊓=1hasMemory. Meomry[7)]

M2=Computer⊓=2hasMemory. Meomry

M3=Computer⊓=3hasMemory. Meomry

M4=Computer⊓=4hasMemory. Meomry

经过事例检索，$Source_2$ 和 $Source_3$ 的配置花费为 4500 元和 3500 元，超过顾客的预算上限 3000 元，$Source_4$ 中的显卡为集成显卡，不符合顾客独立显卡的需求。所以与目标事例最相似的源事例为 $Source_1$。下面的事例修正算法在相似事例 $Source_1$ 的基础上进行。

1. 概念扩展

· 对于相似事例 C_s（对应上例 $Source_1$）和目标事例 C_t（对应上例 Tgt）这两个复杂的概念，为了简化构造，先把这两个概念都转化为否定范式(NNF)，即否定只出现在概念名前。

· 设 $C_{s,t}=C_s⊓C_t$，即 $C_{s,t}=Tgt⊓Source_1$，用 $ALCQ(D)$ 的 Tableau 算法判断概念 $C_{s,t}$ 相对于知识库的可满足性。如果概念 $C_{s,t}$ 是可满足的，说明相似事例和目标事例不存在冲突，即事例复用成功，所以不需要进行后面的修正步骤；如果产生矛盾，那么需要进行下面的修正步骤。

· 使用 3.1 节提出的 8 条扩展规则把概念 $C_{s,t}$ 扩展成完全概念图 G_i（$1≤i≤m$），如下图所示。在扩展过程中，由于概念 $C_{s,t}$ 中有析取符号⊔存在，因此根据⊔规则会生成如下的 3 个图 $G1$、$G2$ 和 $G3$，分别如图 1—图 3 所示，这 3 个图完全独立。

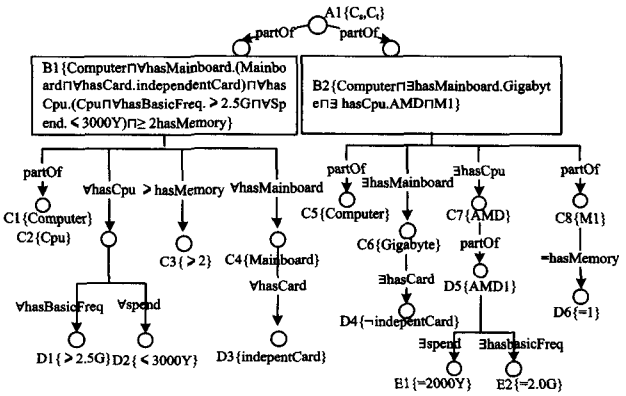

图 1 概念图 $G1$

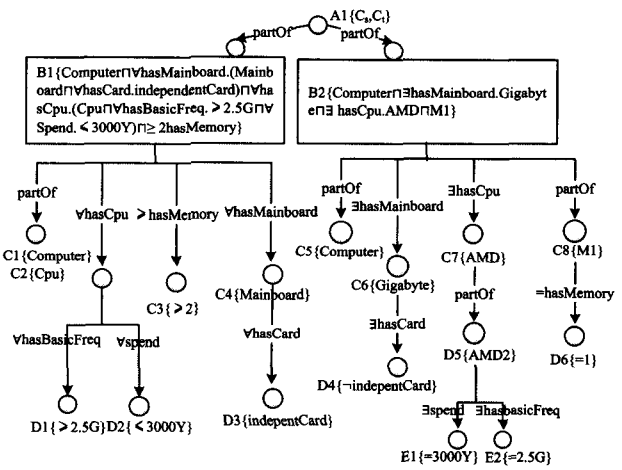

图 2 概念图 $G2$

---

1) 电脑配置必须是独立的显卡，主频要不小于 2.5GHz，内存条数目不小于 2 根

2) 有华硕和技嘉两种主板

3) 华硕主板有两种型号

4) 集成显卡的华硕主板

5) 独立显卡的华硕主板

6) 主频为 2.0GHz，花费为 2000 元的 AMD1 芯片

7) 有一根内存条的电脑 M1

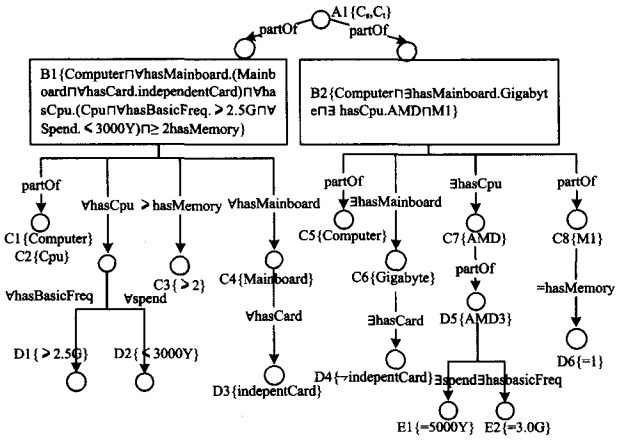

图 3　概念图 $G3$

2. 冲突检测

定义 9(结点有向边，VEV)　结点之间有向边的集合。设共有 n 个结点 $x_1,\cdots,x_n$，根结点到 $x_i(1\leqslant i\leqslant n)$结点的路径上所有的有向边集合为 E_{xi}，则 n 个结点 $x_1,\cdots,x_n$ 的$VEV=E_{x1}\cup\cdots\cup E_{xn}-E_{x1}\cap\cdots\cap E_{xn}$。

定义 10(结点距离)　结点的距离之和为结点之间 VEV 集合里所有有向边的距离之和。设 x、y 是存在有向边的相邻结点，有向边距离规定如下：如果 $F(\langle x,y\rangle)=$partOf，则 x、y 之间有向边 F 的距离为 0。设整个概念图中共有 p 个简单角色 $R_1,\cdots,R_p$，如果 $F(\langle x,y\rangle)=\forall R_i$，则 x、y 之间有向边 F 的距离为 $-r_i$。如果 $F(\langle x,y\rangle)=\exists R_i$，则 x、y 之间有向边 F 的距离为 r_i，其中 r_i、$r_j(i\neq j)$是未知变量，它们之间不能运算。设整个概念图中共有 q 个有型角色 $F_1,\cdots,F_q$，如果 $F(\langle x,y\rangle)=\forall F_i$，则 x、y 之间有向边 F 的距离为 $-f_i$。如果 $F(\langle x,y\rangle)=\exists F_i$，则 x、y 之间有向边 F 的距离为 f_i，其中 f_i、$f_j(i\neq j)$是未知变量，它们之间不能运算。如果 $F(x,y)=\geqslant R_i$，则 x、y 之间有向边 F 的距离为 n_i。如果 $F(x,y)=\leqslant R_i$，则 x、y 之间有向边 F 的距离为 $-n_i$，其中 n_i、$n_j(i\neq j)$是未知变量，它们之间不能运算。

下面对概念中的冲突进行检测的方法称为冲突检测机制。

对上一步扩展后的完全概念图中叶子结点进行分类检测，分类依据是有向边的标记函数 F 的值和结点距离之和，设定任意 n 个结点 $x_1\cdots x_n$，这 n 个结点的结点距离为 S_n，x_0 为根节点，则：

A 类结点$=\{(x_1,\cdots,x_n)\mid S_n=0$，并且存在结点 x_i 使得 $F(\langle x_0,x_i\rangle)=$partOf$\}$。

B 类结点$=\{(x_1,\cdots,x_n)\mid S_n=0$，并且存在两个结点 x_i，x_j，使得 $F(\langle x_i,x_j\rangle)=\exists R(R$ 为任意的简单角色)，且对于任意两个结点 x_i，x_j，$F(\langle x_0,x_i\rangle)\neq\exists F(F$ 为任意的有型角色)$\}$。

C 类结点$=\{(x_1,\cdots,x_n)\mid S_n=0$，并且存在两个结点 x_i，x_j，使得 $F(\langle x_i,x_j\rangle)=\geqslant R(R$ 为任意的简单角色)$\}$。

D 类结点$=\{(x_1,\cdots,x_n)\mid S_n=0$，并且存在两个结点 x_i，x_j，使得 $F(\langle x_i,x_j\rangle)=\exists F(F$ 为任意的有型角色)。

1)对于 A 类结点标记的概念，出现下面两种情况之一，就说明产生了冲突的情况，记下这组结点和冲突的个数：①如果 $x_1,x_2\in$A 类结点，并且 $L(x_1)=C,L(x_2)=\neg C$。②如果 $x_1,x_2\in$A 类结点，并且 $L(x_1)=C_1,L(x_2)=\neg C_2$，$C_1$、$C_2$ 是不同的概念，但是满足 $C_1\subseteq C_2$。

2)对于 B 类结点标记的概念，出现下面两种情况之一，就说明产生了冲突的情况，记下这组结点和冲突的个数：①如果 $x_1,x_2\in$A 类结点，并且 $L(x_1)=C,L(x_2)=\neg C$。②如果 $x_1,x_2\in$A 类结点，并且 $L(x_1)=C_1,L(x_2)=\neg C_2$，$C_1$、$C_2$ 是不同的概念，但是满足 $C_1\subseteq C_2$。

3)对于 C 类结点标记的表达式，出现下面两种情况之一，就说明产生了冲突的情况，记下这组结点和冲突的个数：①表达式中的概念为 C_1、C_2，P_1、P_2 分别为概念前的谓词约束，C_1、C_2 相同，且 $P_1\cap P_2=\emptyset$。②表达式中的概念为 C_1、C_2，P_1、P_2 分别为概念前的谓词约束，其中 $P1=\geqslant n_1$，$P2=\leqslant n_2$，C_1、C_2 是不同的概念，但是满足 $C_1\subseteq C_2$，且 $P_1\cap P_2=\emptyset$。

4)对于 D 类结点标记的概念，设这些结点标记内容分别为 $d_1,\cdots,d_n$，如果 $d_1{}^D\cap\cdots\cap d_2{}^D=\emptyset$，说明这组结点产生了冲突，记下这组结点和冲突的个数。

例如图 $G3$ 中，A 类结点集合为$\{(C1,C5),(C1,C8),(C5,C8),(C1,C5,C8)\}$，这 4 组结点中的概念均没有产生冲突。B 类结点集合为$\{(D3,D4)\}$，$(D3,D4)$这组结点标记的概念分别为 independentCard 和¬independentCard，出现了冲突情况，所以记下这组结点。C 类结点集合为$\{(C3,D6)\}$，$(C3,D6)$这组结点标记的表达式分为“$\geqslant 2$”和“$=1$”，谓词约束对应为$(\geqslant 2)$和$(=1)$，$(\geqslant 2)\cap(=1)=\emptyset$，所以这组结点产生了冲突，记下这组节点。D 类结点集合为$\{(D1,E2),(D2,E1)\}$，$(D1,E2)$这组结点标记的内容分别为$(\geqslant 2.5G)$和$(=3.0G)$，而$(\geqslant 2.5G)\cap(=3.0G)\neq\emptyset$，所以这组结点没产生冲突。$(D2,E1)$这组结点标记的内容分别为$(\leqslant 3000Y)$和$(=5000Y)$，$(\leqslant 3000Y)\cap(=5000Y)=\emptyset$，所以这组结点产生了冲突，记下这组结点。

3. 冲突消除

定义 11(概念替换规则)　在领域知识本体库中，查找产生冲突的概念 C 的兄弟概念或者兄弟概念的子概念 P 来替换 C 概念。如果 P 概念替换 C 后能消除冲突，则结束概念替换。如果 P 概念没有消除概念 C 产生的冲突，就在本体库中往上回溯查找 C 的父概念。重复这个过程，如果回溯到根结点还没有找到合适的替换概念，就结束概念替换。

通过检测阶段，得到了存在冲突的概念图，优先选择概念图中冲突概念数量最少的概念图进行修正。如果修正成功，则不需要继续修正剩余的概念图。

那么对于概念图中产生矛盾的源事例概念 C_1，C 是根据概念替换规则查找到的概念，具体步骤如下：

1)如果 C 是不存在的，即找不到合适的替换概念，则考虑删除概念，达到消除矛盾的目的。由于目标事例概念 C_t 是不能改变的，因此只有删除 C_s 中的概念 C_1。

2)如果 C 是存在的，就用 C 概念去替换 C_1，消除冲突。

上一阶段检测 $G3$ 图中出现冲突情况的有：B 类结点 $(D3,D4)$，$D3$ 结点由 target 概念扩展而来，$D4$ 结点由 Source 概念扩展而来，所以考虑替换 $D4$ 结点标记中产生冲突的概念¬independentCard，因为 Gigabyte⊨¬independentCard，所以替换 Gigabyte 概念。根据定义 11 的概念替规则在本体库中找到 Gigabyte 相似的概念 ASUS2 进行替换，经过本文算法检测，替换之后没有产生冲突，这表明$(D3,D4)$这组结点产

生的冲突修正成功。同理可以修正概念图中其他冲突情况。那么消除所有冲突后的解答为：

R=Computer⊓∃ hasMainboard. ASUS2⊓∃ hasCpu. Intel2⊓M2

3.3 算法性质

定义 12($ALCQ(D)$范式)[29] 设 D 是 $ALCQ(D)$-概念，当且仅当 D 形如⊤或⊥或 $D_1 \sqcup D_2 \sqcup \cdots \sqcup D_n$，且 D_i 具有下述形式，则 D 是 $ALCQ(D)$-概念范式。

$$D_i = \sqcap_{A \in prim(D_i)} A \sqcap \sqcap_{R \in N_R} [\forall R.\, val_R(D_i) \sqcap \sqcap_{E \in ex_R(D_i)} \exists R.\, E] \sqcap \sqcap_{\geqslant mR.C \in Lbqual(D_i)} \geqslant mR.\, C \sqcap \sqcap_{\leqslant nR.C \in Gbqual(D_i)} \leqslant nR.\, C \sqcap \sqcap_{F \in N_f, P \in ex_F(D_i)} [\exists F.\, P \sqcap \forall F.\, val_T(D_i)] \quad (1)$$

其中，$prim(D_i)$表示 D_i 中顶层(即最外层，没有任何约束)的概念名及概念名的否定形式(形如 A 或$\neg A$)的集合；$val_R(D_i)$表示 D_i 中顶层的、角色 R 值限定约束概念的后缀概念的合取，如有 $\forall R.\, D \sqcap \forall R.\, C$，则 $val_R(D_i) = C \sqcap D$，这里的 C,D 如果是复杂概念，仍是 $ALCQ(D)$-概念范式；$ex_R(D_i)$表示 D_i 中顶层的、角色 R 存在限定约束概念的后缀概念组成的集合，如有 $\exists R.\, C \sqcap \exists R.\, D$，则 $ex_R(D_i) = \{C, D\}$，这里的 C,D 如果是复杂概念，仍是 $ALCQ(D)$-概念范式；$Lbqual(D_i)$表示 D_i 中顶层的、所有的最小定性数量约束概念的集合，且在集合中，如果存在形如$\geqslant nR.\, C$，$\geqslant mR.\, C$(只有自然数 n,m 不同)，则简写为$\geqslant \max(n,m)R.\, C$；$Gbqual(D_i)$表示 D_i 中顶层的、所有的最大定性数量约束概念的集合，且在集合中，如果存在形如$\leqslant nR.\, C$，$\leqslant mR.\, C$(只有自然数 n,m 不同)，则简写为$\leqslant \min(n,m)R.\, C$；$ex_F(D_i)$表示 D_i 中顶层的、有型角色 F 存在限定约束概念的后缀有型谓词组成的集合，如有 $\exists F.\, P \sqcap \exists F.\, d$，则 $ex_F(D_i) = \{P, d\}$；$val_F(D_i)$表示 D_i 中顶层的、有型角色 F 值限定约束概念的后缀有型谓词的合取，如有 $\forall F.\, d \sqcap \forall F.\, P$，则 $val_F(D_i) = P \sqcap d$。

定理 1 所有 $ALCQ(D)$-概念都可以等价转换为 $ALCQ(D)$-概念范式。

证明：因为 $ALCQ(D)$-概念满足蕴涵定律，即$\sqsubseteq$可等价转换为⊔,⊓,¬表示，如 $A \sqsubseteq B$ 等价于$\neg A \sqcup B$；满足德摩根定律，可将否定符号¬深入到原子公式；且满足分配律、结合律及交换律。通过上述定律可将 $ALCQ(D)$概念等价转换为对应的概念范式。

定理 2(界限性) 对任一 $ALCQ(D)$的概念，算法都停机。因为：1)在扩展阶段，概念树中的结点个数取决于 $C_{s,t}$中原子概念个数，且每个原子概念最多只能产生一个结点，而沿着概念树的路径，结点标注概念越来越简单，因此概念树的深度与输入概念 $C_{s,t}$正相关。在检测阶段，因为概念树的有限性，所以完全针对概念树的检测工作也是有限的。在修正阶段，需要修正的矛盾个数依赖于检测工作，检测工作有限，所以修正的矛盾个数有限。

定理 3(可靠性) 如果对于 $ALCQ(D)$概念 $C_{s,t}$，用基于 $ALCQ(D)$的事例修正算法检测出某种矛盾，那么 $C_{s,t}$确实存在这种矛盾，并且经过算法修正一定能消除矛盾。因为：1)如果 $C_{s,t}$不存在矛盾，那么根本不会进入算法的检测阶段。2)如果 $C_{s,t}$存在矛盾，那么矛盾有如下 3 种形式：

(1)存在概念 C，有$\{C, \neg C\} \subseteq C_{s,t}$；

(2)存在概念 C、角色 R 和自然数 n，有$\{(\geqslant(n+1)R.\, C), (\leqslant nR.\, C)\} \subseteq C_{s,t}$；

(3)对于 $C_{s,t}$的数据类型(或否定式)$d_1 \cdots d_n$，有 $d_1^D \cap \cdots \cap d_n^D = \emptyset$。

而基于 $ALCQ(D)$的事例修正算法正是基于这 3 种形式的矛盾进行检测，检测出的矛盾概念用概念替换方法替换之后，还要经过算法的检测，并且重复这个过程，直至找到合适的替换概念。而且因为本体是有限的，可替换概念也是有限的，经过有限次替换找不到合适的替换概念，就直接删除矛盾概念，消除矛盾。

定理 4(完备性) 如果概念 $C_{s,t}$确实存在矛盾，即概念 $C_{s,t}$相对于知识库不可满足，那么经过上述算法一定能检测出矛盾，并且经过算法完成修正之后，$C_{s,t}$一定没有矛盾。

证明：设 T 为 TBox，如果 $C_{s,t}$是不可满足的概念，那么根据概念的可满足性判定定理可以推出不存在某个解释 I，使得 I 是 T 的一个模型并且 $C_{s,t}^I \neq \emptyset$。因为 $C_{s,t}$是一个 $ALCQ(D)$的概念，根据定理 1 可知 $C_{s,t}$可以等价转换为 $ALCQ(D)$范式，那么 $C_{s,t} = D_1 \sqcup D_2 \sqcup \cdots \sqcup D_n$，其中 D_i 与式(1)相同，则 $[D_1 \sqcup D_2 \sqcup \cdots \sqcup D_n]^I = \emptyset$。由 $ALCQ(D)$的语义可知 $D_1^I \cup D_2^I \cup \cdots \cup D_n^I = \emptyset$，那么必有 $D_1^I = \emptyset, \cdots, D_n^I = \emptyset$ 都成立。而 $D_i^I = \emptyset$ 成立，所以有：

$[\sqcap_{A \in prim(D_i)} A]^I = \emptyset$ 或者 $[\sqcap_{R \in N_R} [\forall R.\, val_R(D_i) \sqcap \sqcap_{E \in ex_R(D_i)} \exists R.\, E]]^I = \emptyset$ 或者 $[\sqcap_{F \in N_f, P \in ex_F(D_i)} [\exists F.\, P \sqcap \forall F.\, val_T(D_i)]]^I = \emptyset$ 或者 $[\sqcap_{\geqslant mR.C \in Lbqual(D_i)} \geqslant mR.\, C \sqcap \sqcap_{\leqslant nR.C \in Gbqual(D_i)} \leqslant nR.\, C]^I = \emptyset$。

那么如果$[\sqcap_{A \in prim(D_i)} A]^I = \emptyset$ 成立，则有$(A_1 \sqcap A_2 \sqcap \cdots \sqcap A_m)^I = \emptyset (A_j \in prim(D_i))$，那么 $A_1^I \cap \cdots \cap A_m^I = \emptyset$。因为 A_i 是原子概念，所以 A_i^I 不为空集，那么必然有这样的情况，即：$A_i^I \cap A_k^I = \emptyset (1 \leqslant i,k \leqslant m)$，则$(A_i \sqcap A_k)^I = \emptyset$，那么就有 $A_i \sqcap A_k = \perp$，则 $A_i = \neg A_k$，说明存在概念 C，使得$\{C, \neg C\} \subseteq C_{s,t}$。

如果$[\sqcap_{\geqslant mR.C \in Lbqual(D_i)} \geqslant mR.\, C \sqcap \sqcap_{\leqslant nR.C \in Gbqual(D_i)} \leqslant nR.\, C]^I = \emptyset$ 成立，则有$[\sqcap_{\geqslant mR.C \in Lbqual(D_i)} \geqslant mR.\, C]^I \cap [\sqcap_{\leqslant nR.C \in Gbqual(D_i)} \leqslant nR.\, C]^I = \emptyset$，因为$(\geqslant mR.\, C)^I$、$(\leqslant nR.\, C)^I$ 单独不为空，所以$\geqslant mR.\, C$，$\leqslant nR.\, C$ 同时存在，并且 $m > n$，即存在概念 C、角色 R 和自然数 n，有$\{(\geqslant(n+1)R.\, C), (\leqslant nR.\, C)\} \subseteq C_{s,t}$。

如果$[\sqcap_{F \in N_f, P \in ex_F(D_i)} [\exists F.\, P \sqcap \forall F.\, val_T(D_i)]]^I = \emptyset$，则有$(\exists F.\, P_1 \sqcap \exists F.\, P_2 \sqcap \cdots \sqcap \exists F.\, P_n)^I \cap (\forall F.\, d_1 \sqcap \forall F.\, d_2 \cdots \sqcap \forall F.\, d_m)^I = \emptyset$，$(\exists F.\, P_1 \sqcap \exists F.\, P_2 \sqcap \cdots \sqcap \exists F.\, P_n)^I$，$(\forall F.\, d_1 \sqcap \forall F.\, d_2 \cdots \sqcap \forall F.\, d_m)^I$ 单独均不为空，所以必有$(\exists F.\, P_1 \sqcap \cdots \sqcap \exists F.\, P_k \sqcap \forall F.\, d_1 \sqcap \cdots \sqcap \forall F.\, d_j) = \perp$，则 $P_1 \sqcap \cdots \sqcap P_k \sqcap d_1 \sqcap \cdots \sqcap d_j = \perp$，所以 $P_1^D \cap \cdots \cap d_j^D = \emptyset$，即对于 $C_{s,t}$的数据类型(或否定式)$d_1 \cdots d_n$，有 $d_1^D \cap \cdots \cap d_n^D = \emptyset$。

事实上，$C_{s,t}$经过规则扩展后，由定理 2 可知最终能扩展成一个完全概念树，如果 $C_{s,t}$确实存在矛盾，那么根据上面的推断，可能是下列 3 种矛盾的一种或几种：

(1)存在概念 C，有$\{C, \neg C\} \subseteq C_{s,t}$；

(2)存在概念 C、角色 R 和自然数 n，有$\{(\geqslant nR.\, C), (\leqslant nR.\, C)\} \subseteq C_{s,t}$；

(3)对于 $C_{s,t}$的结点标记包含的数据类型(或否定式)$d_1 \cdots d_n$，有 $d_1^D \cap \cdots \cap d_n^D = \emptyset$。

第一种矛盾在算法检测过程中，在 A 类和 B 类的结点中能够完全检测出来；第二种矛盾在算法检测过程中，在 C 类的结点中能够完全检测出来；第三类矛盾在 D 类的结点中完全能检测出来。所以对于概念 $C_{s,t}$存在的 3 种矛盾，算法能

够全部检测出来，并且对于每一个矛盾概念都会进行以下过程：即先用本体中矛盾概念的相似概念替换，再用本文算法验证这个矛盾是否还存在。替换达不到消除矛盾的目的，就删除矛盾概念，最终一定能消除这个矛盾，所以 $C_{s,t}$ 一定不存在矛盾。

研究表明，该事例修正算法具有界限性、可靠性和完备性。

结束语 本文首先介绍了描述逻辑 $ALCQ(D)$ 的语法和语义以及相关的推理问题，接着在 $ALCQ(D)$ 表示事例的基础上提出了事例修正的算法。事例修正算法引入了概念图，定义了基于概念图的概念扩展规则。把源事例和目标事例作为两个复杂的概念，通过概念图以及概念扩展规则，将这两个复杂概念完全展开。然后将概念图中的叶子结点分成 A、B、C、D 类来检测产生冲突的结点。最后在本体库中用概念替换规则消除冲突完成修正。通过整个算法，能够尽可能地减少人工干预带来的偏颇，达到自动检错、自动修正的目的。不足的是该算法为了展开概念，需要很大空间。

下一步的工作是：1）对本文提出算法，改进优化，编程实现，开发一个完整的事例修正系统原型；2）研究更复杂事例的表示方法，如带模糊语义和时态特性的事例，研究其在事例修正中的算法；3）考虑知识库中存在 ABox 时，算法怎么处理。

参 考 文 献

[1] De Mántaras, R L, Plaza E. Case-based reasoning: An overview [J]. AI communications, 1997, 10(1): 21-29

[2] 杜云艳，周成虎，邵全琴，等. 地理案例推理及其应用[J]. 地理学报，2002，57(2)：151-158

[3] Schank R C. Dynamic memory-a theory of reminding and learning in computers and people[M]. Cambridge University Press, 1983: 1-234

[4] Kolodner J L. Reconstructive memory: A computer model[J]. Cognitive Science, 1983, 7(4): 281-328

[5] Aamodt A, Plaza E. Case-based reasoning: Foundational issues, methodological variations, and system approaches[J]. AI Communications, 1994, 7(1): 39-59

[6] Fuchs B, Lieber J, Mille A, et al. An algorithm for adaptation in case-based reasoning[C]// ECAI. 2000: 45-49

[7] D'Aquin M, Badra F, Lafrogne S, et al. Adaptation knowledge discovery from a case base[OL]. http://www.fadi.lautre.net/publis/ecaiob.pdf

[8] Leake D B, Kinley A, Wilson D C. Acquiring Case Adaptation Knowledge: A Hy rid Approae[C]// National Conference on Artifical Intellgence—AAAI. 1996: 684-689

[9] Crawa S, Wiratunga N, Rowe R C. Learning adapting knowledge to improve case-base reasoning[J]. Artifical Intelligence, 2006, 170(16/17): 1175-192

[10] Assali A A, Lenne D, Debray B. Adaptation Knowledge Acquisition in a CBR System[J]. International Journal on Artificial Intelligence Tools, 2013, 22(1): 1-13

[11] Hurley B, O'Sullivan B. Adaptation in a CBR-Based solver portfolio for the satisfiability problem[M]// Case-Based Reasoning Research and Development. Springer Berlin Heidelberg, 2012: 152-166

[12] Liao Z, Mao X, Hannam P M, et al. Adaptation methodology of CBR for environmental emergency preparedness system based on an Improved Genetic Algorithm[J]. Expert Systems with Applications, 2012, 39(8): 7029-7040

[13] Djebbar A, Merouani H F. Applying BN in CBR Adaptation-Guided Retrieval for Medical Diagnosis[J]. International Journal of Hybrid Information Technology, 2012, 5(2): 41-56

[14] Fuchs B, Lieber J, Mille A, et al. Differential Adaptation: an Operational Approach to Adaptation for Solving Numerical Problems with CBR[J]. Knowledge-Based Systems, 2014, 68: 103-114

[15] Baader F. The description logic handbook: theory, implementation, and applications[M]. Cambridge: Cambridge University Press, 2003

[16] Sattler U. A concept language extended with different kinds of transitive roles[M]// KI-96: Advances in Artificial Intelligence. Springer Berlin Heidelberg, 1996: 333-345

[17] Schmidt-Schauβ M, Smolka G. Attributive concept descriptions with complements[J]. Artificial intelligence, 1991, 48(1): 1-26

[18] Horrocks I, Gough G. Description logics with transitive roles[C]// Proceedings of the International Workshop on Description Logics(DL'97). 1997: 1-4

[19] Horrocks I, Sattler U, Tobies S. Practical reasoning for expressive description logics[C]// Logic for Programming and Automated Reasoning. Springer Berlin Heidelberg, 1999: 161-180

[20] Horrocks I, Sattler U. A description logic with transitive and inverse roles and role hierarchies[J]. Journal of logic and computation, 1999, 9(3): 385-410

[21] Horrocks I, Sattler U, Tobies S. Practical reasoning for very expressive description logics[J]. Logic Journal of IGPL, 2000, 8(3): 239-263

[22] Baader F, Sattler U. An overview of tableau algorithms for description logics[J]. Studia Logica, 2001, 69(1): 5-40

[23] Horrocks I, Sattler U, Tobies S. A PSpace algorithm for deciding ALCNIR satisfiability[OL]. http: academic. reasearch. mecrosoft. com/paptr/290356. pdf

[24] Lutz C. Description logics with concrete domains-A survey[J]. Advances in Modal Logic, 2003, 4: 265-296

[25] Hustadt U, Motik B, Sattler U. Reasoning in description logics with a concrete domain in the framework of resolution[C]// ECAI. 2004, 16: 353

[26] Stanchev L. On Efficient Access to Knowledge bases[C]// Proceedings of The 20th Midwest Artificial Intelligence and Cognitive Science Conference. Indiana University-Purdue University Fort Wayne, Fort Wayne, 2009

[27] 梅婧，林作铨. 从 ALC 到 SHOQ(D)：描述逻辑及其 Tableau 算法[J]. 计算机科学，2005，32(3)：24-35

[28] Cojan J, Lieber J. An algorithm for adapting cases represented in an expressive description logic[M]// Case-Based Reasoning. Research and Development. Springer Berlin Heidelberg, 2010: 51-65

[29] d'Amato C, Fanizzi N, Esposito F. A dissimilarity measure for ALC concept descriptions[C]// Proceedings of the 2006 ACM symposium on Applied computing. ACM, 2006: 1695-1699