文章编号： 1000-5013(2011)06-0641-05

一种本体协同冲突的检测方法

陈叶旺，李海波，余金山，陈维斌

（华侨大学 计算机科学与技术学院，福建 泉州 362021）

摘要： 针对农业领域分布式的迭代知识协同建构过程中存在的协同冲突及难于检测的问题，提出协同冲突划分、检测与消解方法. 把协同冲突与本体一致性问题区别开来，按本体知识编辑操作的影响范围、操作类型及启发式规则把协同冲突划分为硬冲突、软冲突. 依据这个划分，给出一个高效的检测算法和冲突消解方案.

关键词： 本体；协同冲突；硬冲突；软冲突

中图分类号： TP 301 **文献标志码：** A

本体已经为人们广泛地接受成为知识共享的基础设施，但本体建构却是一个令人头痛的问题[1]. 2002 年，Denny[2]分别介绍和比较了 56 种本体编辑工具. 借助 Protégé 2000[3]和 OilEd[4]等本体构建工具，可以把精力集中在本体内容的组织上，而不必了解本体描述语言的细节. 一些支持协同开发[5-7]工具提拱类似于数据库中的同步事务存储控制，包括锁定和回滚. 然而，本体与数据库有一点关键不同，本体实体之间存在着微妙的关系，改变一个实体就有可能引起连锁变化. 在本体协同建构中，冲突总是伴随着开发过程始终. 在传统的软件协同开发过程中，为了解决冲突问题，在版本管理系统中引入“锁”和“分枝/合并”技术. 这种技术的前提是以文档为基本单位，如一个开发人员在操作某个文档，而另一个开发人员也要对该文档修改，则冲突产生. 因此，其解决的办法有：(1) 锁定需要操作的文档；(2) 由版本管理系统自动合并两个开发人员对文档的修改. 然而，在本体开发过程中，情况则比较复杂，必然会在在协同冲突. 有些冲突本身不会对本体一致性造成损害，其错误的结果是隐性的，很难轻易地发现，各种本体推理机也无法检测出来，但其结果却不是知识操作员所期望的. 针对这些问题，本文提出分布式的迭代知识协同构建过程中协同冲突检测划分、检测与消解方法.

1 协同冲突划分与检测方法

在农业本体知识建构系统中，本体知识建构的基本单位是本体实体，包括概念、实例、关系、公理等，其粒度远小于物理文档. 为实现这些基本单位的操作，本体知识协同建构系统中定义一些基本的操作命令. 通过这些操作可计算用户对本体的编辑操作的影响范围，并以此为基础来划分和检测协同冲突.

1.1 OntoCommand

定义 1 OntoCommand(OC)：这是一个 3 元组，记为 oc=〈Name，E，V〉. 其中：Name 是操作名称；E 是其操作的本实体集合；V 为操作的参数.

表 1 列出了本体知识协同建构系统中的一些操作命令. 根据操作的类型，区分出 3 类不同的操作：ADD，DEL，MOD，分别表示增加、删除和修改 3 大类操作.

1.2 操作影响范围

本体中存在着丰富的语义关系，其中有些需要通过推理计算才能确定. 改变一个本体实体语义，有可能影响到与其逻辑上相关的其他实体. 这就意味着在本体的协同开发过程中，对于用户的任何一个对本体实体的操作，都存在一个受影响的实体集合的定义.

收稿日期： 2011-04-18

通信作者： 陈叶旺(1978-)，男，讲师，主要从事语义检索与数据挖掘的研究. E-mail：ywchen@hqu.edu.cn.

基金项目： 福建省自然科学基金资助项目(A0810013)；福建省农业科技重大项目(2010N5008)；华侨大学科研基金资助项目(09BS619)

表1 命令列表

Tab. 1 Commands list

Name	Parameter(E)
AddClass/DelClass	(aClass)
AddSubClass/DelSubClassRelation/DelEquClass/DelSameClass	(aClass,supClass)
AddDataProperty/AddObjProperty/DelObjProperty	(aProp)
AddDifferentClassRelation/AddEquivalentClass/AddSameClass	(aClass1,aClass2)
AddDifferentIndividual/AddEquivalentIndividual	(aInd1,aInd2)

定义2 影响范围(Impact Range,IR):给定一个命令操作 oc,IR(oc),计算 oc 的影响范围.

对于所有的原子命令操作,表2给出了每一个命令的影响计算值.

表2 部分操作命令影响范围表

Tab. 2 Affect range table in part of operation command

OCType(oc)	E	IR(oc)
DelInstance	(aIndividual)	Return $E\cup\{$ ins$|$ ins is the same as aIndividual $\}$
DelProperty	(aProperty)	Return $E\cup\{$r$|$ r.actOn(aProperty)= true, r is Restriction $\}$
AddEquClass	(aClass1,aClass2)	Return $E\cup\{$class$|$ (aClass. directSubclassOf (aClass1) $\}$ $\cup\{$class$|$ (aClass. directSubclassOf (aClass2)$\}$
AddObjProperty	(aClass1,aClass2)	Return $E\cup\{$class$|$ (aClass.directSubclassOf(aClass2)$\}$

1.3 协同冲突分析

在本体协同开发过程中,不同用户之间的操作冲突有些很容易判断;有些不那么明显,需要经过描述逻辑(DL)推理,才能判断冲突是否存在;还有一些很难得到冲突是否存在的确定判断,需要用到启发式规则,计算冲突是否有发生的可能性.根据检测冲突的容易程度,可以将协同冲突划分为硬冲突(hard conflicts)、软冲突(soft conflicts).

1.3.1 *硬冲突* 硬冲突是一种比较容易检测的冲突,可以根据操作命令的影响范围和操作类型来判断.给定两个操作命令 oc 和 oc′,OTC(oc,oc′)是冲突检测函数,返回真则表示 oc 有可能与 oc′发生冲突;否则不然.据此,判断两个操作命令是否有发生硬冲突的可能.表3为操作命令类型冲突函数表.其中:T 表示 OTC 值为真;F 表示为假.

表3 操作命令类型冲突函数表

Tab. 3 Conflict function table of the operation command

操作类别	OTC(oc,oc′)		
	MOD	ADD	DEL
MOD	OTC_MOD(oc,oc′)	F	T
ADD	F	F	T
DEL	T	T	F

从表3可知,OTC 值计算是根据操作命令类型来判断的.如两个操作都为 DEL 类型,则不视为冲突,即使两个操作要删除的本体实体是同一个.这是因为 ARGONT 会将相同操作合并.同理,对于同为 ADD 类型的操作也一样.如果两个操作同为 MOD 类型,则需通过 OTC_MOD 函数[8]作进一步判断.

定义3 硬冲突:给定两个由不同用户提交的操作命令,若满足(1) OTC(oc,oc′)为真;(2) DR(oc,co′)〈〉NULL,其中,DR(oc,co′)=IR(oc)∩IR(co′)两个条件,则 oc@oc′成立,表示 oc 与 oc′发生硬冲突.

1.3.2 *软冲突* 有些用户操作之间的冲突并不像硬冲突那样比较容易判断,因为有些操作间接地影响了某些内容的语义,而另一些操作仍旧使用它的原始语义.为了检测因为这种间接影响而造成的冲突,需要通过描述逻辑推理来完成.

为此,定义一个基于描述逻辑的语义规则(semantic rules)集合.以下列出其中几条,违反任何其中一条规则即视为软冲突.

(1) The Same VS Different:检测是否存在一个本体实体,使得它与 e_1,e_2 中的某一个存在 owl:sameAs 或 owl:Equivalent 关系,而与另一个存在 owl:differentFrom 关系.

(2) The Same VS DisjointWith:检测是否存在一个本体实体,使得它与 e_1,e_2 中的某一个存在 owl:sameAs 或 owl:Equivalent 关系,而与另一个存在 owl:disjointWith 关系.

(3) Functional Same VS Different:检测是否存在一个本体实体,使得用 owl:FunctionalObjecpro-

ty 关系可以推导出它与 e_1，e_2 中的某一个存在 owl:sameAs 或 owl:Equivalent 关系，而与另一个存在 owl:differentFrom 关系.

(4) Functional Same VSDisjointWith：检测是否存在一个本体实体，使得用 owl:FunctionalObjecproty 关系可以推导出它与 e_1，e_2 中的某一个存在 owl:sameAs 或 owl:Equivalent 关系，而与另一个存在 owl:disjointWith 关系.

定义 4　语义规则：给定一个本体实体 e 和两个操作命令 oc 和 oc′，语义规则函数 SEM(e, oc, oc′) 用于计算是否存在语义不一致. 返回值若为真，则表示 e_1 与 e_2 之间存在语义不一致，否则不然. 其中：(1) $e\in IR(oc)\cap IR(oc')$；(2) e_1 为独立执行 oc 后实体 e 演化的结果，即 $e\rightarrow e_1$；(3) e_2 为独立执行 oc′ 后实体 e 演化的结果，即 $e\rightarrow e_2$.

定义 5　软冲突：给定分别由不同用户提交的两个操作命令 oc 和 oc′. 若满足以下两个条件：(1) OTC(oc,oc′)为假；(2) $\exists$ SEM $\exists e\in IR(oc)\cap IR(oc')$, s. t. SEM($e$,oc,oc′)=0，则 oc$\sharp$oc′成立，表示 oc 与 oc′之间存在软冲突.

1.4　协同冲突检测算法

为了能高效、快速地进行冲突检测，定义一个数据结构 STRU_CON_SET，如图 1 所示. 其中，所有的 STRU_CON_SET 对象都存放在一个排序的 SortedList v 中. 每一个 STRU_CON_SET 对象，包括一个本体实体 e，一个操作命令集合 IRSet，$\forall$ oc$\in$ IRSet：$e\in IR(oc)$. 即 IRSet 包含的所有操作命令的影响范围中都包含了 e.

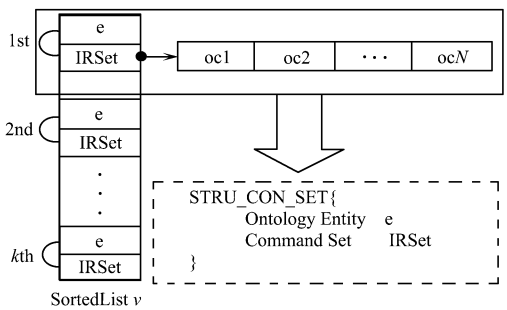


图 1　STRU_CON_SET 数据结构

Fig. 1　Data structure of STRU_CON_SET

那么，给定一个 STRU_CON_SET 对象，根据定义硬冲突、软冲突定义，则有

(1) $\forall$ oc,oc′$\in$scs. IRSet，如果 OTC(oc,oc′)成立，则 oc@oc′；

(2) $\forall$ oc,oc′$\in$scs. IRSet，如果 $\exists$ SEM，s. t. SEM(scs. e,oc,oc′)成立，且 OTC(oc,oc′)不成立，则 oc$\sharp$oc′.

依据上述定义好的 STRU_CON_SET 结构和 3 个定理，设计一个高效的协同冲突检测算法(Collaborative Conflicts Detector Algorithm). 输入的数据为所有用户及所有操作命令，初始化值均为空，即 hardConflicSet={}，softConflictSet={}，结果为产生两类冲突集合. 其伪代码：

```
初始化 Sorted-List〈STRU_CON_SET〉 v.
for each command c of all commands do
   for each entity e in IR(c) do
     STRU_CON_SET scs:=以 e 为关键值在 v 中按二叉排序方查找，若无结果则新
          STRU_CON_SET 对象，并插入 v 中
     for each command c′ in. IRSet of scs do
       if (OTC(c, c′)) do hardConflictSet:=hardConflicSet {〈c,c′〉} /* 硬冲突检测 */
       for each semantic rule SEM in Semantic-Rules-Set do
         if (SEM (e,c, c′)) softConflictSet:=softConflictSet {〈c,c′〉} /* 软冲突检测 */
       end.
     end
     scs. IRSet:=scs. IRSet {oc}
   end
end
```

该算法只对所得操作命令扫描一遍，就可以进行硬冲突、软冲突两重检测，但需要维护一个排序 SortedList v. 所有的冲突结果同样分别放入 hardConflicSet，softConflictSet 和 latentConflictSet 中. 算法主要步骤的复杂度分析：

(1) 第3步,二叉排序算法复杂度为 $\log n$;

(2) 第5步 $O(\mathrm{OTC})$为常量,即 $O(C)$;

(3) 第7步,因为 $O(\mathrm{SEM})=O(C)$,所以这一步复杂度也为常量,即 $O(C)$;

(4) 第9步,因为 $O(\mathrm{OntoSIM})=O(C)$,所以这一步复杂度也为常量,即 $O(C)$;

(5) 第6~7步循环,系统定义的 Semantic-Rules-Set 中 SEM 个数有限,所以这个循环复杂度为常量,即 $O(C)$;

(6) 第3~10步循环,对于一个 STRU_CON_SET 对象 scs 的 IRSet 而言,其所包含的操作命令个数远远小于 n,而根据实验统计结果,极少有超过100,故这一步复杂度为常量,即 $O(C)$;

(7) 第2~12步循环,对于一个操作命令 oc 而言,其影响范围 IR 值有限,因而交集 S 也是个有限集,这一步复杂度为常量,即 $O(C)$;

(8) 第1~13步循环,操作命令数为 n 个,因而复杂度为 $O(n)$.

综合可得,$O(\mathrm{Algorithm\ 2})=O(n\cdot\log n)$.

2 案例分析与实验

为进行有效测试,组织了多个懂得一些农业领域知识的人员,各自可以使用几个不同的帐号独立的完成一系列编辑操作.测试开始只有少数几人进行操作,慢慢地所有人逐步加入其中.在这个过程中没有任何交流,所有操作也均逐一记录下来.服务端配置为 Intel Centrino Duo T2400 1.83GHz PC+2 GB 内存+WindowsXP SP2,本体知识数据来源于自国际粮农组织(Food and Agriculture Organization).表4为测试的本体数据.

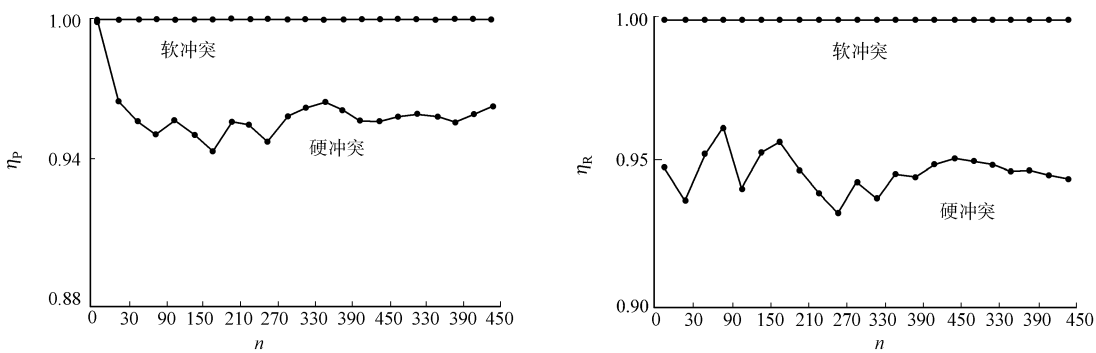
表4 测试本体数据

Tab.4 Body test data

类型	Class	Object Property	Data Property	Data Range	Functional Properties	Inverse Functional Properties	Symmetric Properties	Object-Property
数量	82	233	212	105	176	13	9	12

每隔一段时间,对所有操作做两次分析.第1次是根据操作记录做人工鉴定,找出认为真正的冲突(R);另一次根据的自动检测算法识别出来的冲突 P.将两者对比,可确认经算法检测出来的结果中真冲突的个数为 I,伪冲突为 $F=P-I$.

通过硬冲突和软冲突检测结果及比较,评判算法检测是否能有效检测硬冲突与软冲突.把所做操作中的硬冲突与软冲突检测结果做比较,两类冲突查准率(η_{P})和查全率(η_{R}),如图2所示.



(a) 查准率 (b) 查全率

图2 硬冲突与软冲突检测结果

Fig.2 Test results of hard conflicts and soft conflicts

由图2可知:由于查准率和查全率都比较高,故算法对软冲突检测比较有效.相对而言,对硬冲突检测不如软冲突检测,很难接近100%.究其原因,可能是对每一个命令的 IR 定义不够完备,造成查准率不能达到理想要求,但当扩充操作命令 IR 定义,使查准率接近100%时,查全率却急速下跌.

对于硬冲突检测结果,在 X 轴起始处,查准率查全率变化比较大.可以认为这是因为数据量还不

够，不能真正反映有效结果．因此，随着数据量的增大，其值也日趋平稳，查准率趋近于 95%，查全率也趋近于 93%.

3　结束语

文中提出协同冲突划分、检测与消解方法，将协同冲突与本体一致性问题区别开来，按本体知识编辑操作的影响范围、操作类型及启发式规则把协同冲突划分为硬冲突、软冲突．依据这个划分，给出一个高效的检测算法和冲突消解方案．然而，所提出的协同冲突划分、检测与消解方法仍然是在一个比较大的尺度上来划分冲突，造成有些冲突划分不明甚至错误.下一步的工作是逐步细划编辑操作的影响范围，进而更详细划分冲突类别，以提高冲突消解效率.

参考文献：

[1] NOY N F，CHUGH A，ALANI H. The CKC challenge：Exploring tools for collaborative knowledge construction [J]. IEEE Intelligent Systems，2008，23(1)：64-68.

[2] DENNY M. Ontology building：A survey of editing tools[EB/OL]. [2002-11-06]http://www. xml. com/pub/a/2002/11/06/ontologies. html.

[3] NOY N F，SINTEK M，DECKER S，et al. Creating semantic Web contents with protege-2000[J]. IEEE Intelligent Systems，2001，16(2)：60-71.

[4] BECHHOFER S，HORROCKS I，GOBLEN C，et al. OilEd：A reason-able ontology editor for the semantic Web[C]// Joint German/Austrian Conference on Artificial Intelligence (KI′01). Berlin：Springer，2001：396-408.

[5] BOZSAK E，ETC M. KAON-towards a large scale semantic Web[C]// The Third International Conference on EC-Web Aix-en-Provence. France：Springer，2002：304-313.

[6] DOMINGUE J. Tadzebao and webonto：Discussing，browsing and editing ontologies on the Web[C]// Proceedings of the Eleventh Knowledge Acquisition Workshop. Banff：[s. n.]，1998.

[7] TEMPICH C，PINTO H S，SURE Y，et al. An argumentation ontology for distributed[C]// Loosely Controlled and Evolving Engineering Processes of Ontologies. Crete：[s. n.]，2005.

[8] 陈叶旺. 国家农业本体协同建构与语义检索若干技术研究[D]. 上海：复旦大学，2009.

A Collaborative Conflicts Detection Method for Ontology Building

CHEN Ye-wang，LI Hai-bo，
YU Jin-shan，CHEN Wei-bin

(College of Computer Science and Technology，Huaqiao University，Quanzhou 362021，China)

Abstract：There exists conflicts in the process of iterative constructing agricultural ontology collaboritvelly，and they are difficult to detect. In this paper，based on the operation range，operation type and heuristic rulses，the collaborative conflicts was divided to hard conflicts and soft conflicts. We propose a method to classify，detect and digest collaborative conflicts，which differentiate the collaborative conficts from ontology consistency，finally we give a efficient solution to resolve conflits when they are detected.

Keywords：ontology；collaborative conflicts；hard conflicts；soft conflicts

（**责任编辑**：陈志贤　　**英文审校**：吴逢铁）