文章编号：1000-8608(2009)04-0598-07

一种从企业遗留智能系统学习OWL本体方法研究

高俊杰， 邓贵仕*

（大连理工大学 系统工程研究所，辽宁 大连 116024）

摘要： 为规避基于文本的本体学习中尚未解决的中文分词歧义问题，实现企业遗留智能系统中拥有的大量领域知识的复用，提出一种从遗留智能系统学习OWL本体的方法. 在分析关系数据库模式、元组集与OWL本体之间元素对应关系的基础上，详述了该方法的具体步骤. 与现有相关方法相比，本方法针对的数据源蕴含更丰富的领域知识，更加适合实际的工程应用；通过一个简单、低时间复杂度的转换算法，而非中间模型或大量抽象规则，从关系数据库模式中自动获取相应的OWL本体部分；并按照一定的先后顺序将遗留系统中范例、规则知识项（元组集）移植为OWL本体中对应的个体. 一个面向企业工装工时定额的遗留智能系统应用实例证实了该方法的有效性.

关键词： 遗留智能系统；本体学习；工时定额；OWL

中图分类号： TP182 **文献标志码：** A

0 引 言

本体是共享概念模型的形式化规范说明，已经日渐成为企业建模、智能信息集成、语义Web、知识系统等众多领域的重要组成部分. 但是，完全手工构建本体十分耗时、费力，易出现倾向性错误、动态更新困难等问题. 于是自动或半自动构建本体的方法——本体学习（ontology learning）应运而生，并成为国内外研究热点. 其中，基于文本的本体学习方法是本体学习的主要方法和研究重点；然而由于中文分词的歧义问题始终未能很好地解决，此方法处理和分析的语料多是基于西文的. 为了满足本体应用发展的需要，以非文本的其他形式为数据源的本体学习已经引起越来越多的关注.

随着信息化进程的加快，企业拥有大量的遗留智能系统，并且在使用过程中，系统中的领域知识也在不断地增加、更新、优化. 由于关系数据库技术的成熟性和应用的广泛性，企业遗留智能系统中的规则、范例等知识大都以关系表的形式存储在关系数据库中. 然而，关系数据库模式仅能完整描述数据库的结构，而不能为其数据提供清晰的语义信息，使得这些知识不能在领域内共享、复用. 因此，从企业遗留智能系统学习本体的研究显得十分重要并拥有广泛的应用前景. 考虑到在种类众多的本体描述语言中，W3C提出的最新标准OWL无疑是最有前景的，本文将学习目标设定为OWL本体，提出一种从企业遗留智能系统学习OWL本体的方法.

1 相关研究现状

目前，遗留系统的定义仍不统一，大致可理解为一个已经运行了很长时间的，人们不知道如何处理但对其自身的组织又是至关重要的软件系统[1]. 这里讨论的遗留智能系统定义为一个已经运行了一段时间，能管理、运用特定知识或通过智能推理完成企业中一定业务管理和决策工作，而性能已落后的智能软件系统. 这些遗留智能系统中大都拥有用于智能推理以解决领域问题的规则、范例知识.

本文研究的数据源设定为遗留智能系统中存储于关系数据库的规则和范例知识，学习目标是OWL本体；数据源基于关系模型，关系模式是型，元组集是值；而OWL本体是一种具有更多语

收稿日期： 2008-02-11； **修回日期：** 2009-06-01.
基金项目： 国家自然科学基金资助项目（70671016）.
作者简介： 高俊杰（1978-），男，博士；邓贵仕*（1946-），男，博士，教授，博士生导师，E-mail：denggs@dlut.edu.cn.

义、结构更为复杂的模型.所以,从企业遗留智能系统学习OWL本体的主要任务可以设定为分析相应关系数据库模式和元组集中蕴含的语义信息,将其映射到OWL本体中的相应部分.

与此相关的研究——从关系数据库学习本体,已有一些研究者提出了自己的解决方案.文献[2,3]提出先将关系数据库模式翻译为一个中间模型,再由中间模型翻译成本体模型;文献[4,5]为实现P2P关系数据库之间的无缝互操作,提出先定义一种关系数据库本体OWL-RDBO,再将数据库模式元数据和结构限制转换为本体实例的方法,但这种方法没能很好地考虑领域内概念及其层次关系等语义信息的提取;文献[6]通过对主键、数据、属性的相关性分析,采用一定的映射规则完成向flogic本体的转换,理论上可获取更多的语义信息,但是,其依据属性间甚至数据间的包含依赖关系的判断,实现难度大,不利于实际工程应用;文献[7~12]提出通过相应的一组学习规则从关系数据库模式信息提取OWL本体的方法,此类方法与本文采用的方法最为接近,然而,它们要么仅在理论层研究相应的学习规则,过分强调表间较为复杂的主、外键关系(实际数据库中很少出现)向本体转换的规则,要么粗糙地分析关系数据库模式和本体间的对应关系,仅生成轻量级本体,使得获取的本体很像关系型的.

更重要的是已有方法基本都以一般的关系数据库为研究对象,仅考虑关系模式中蕴含的领域知识的提取,而本文以遗留智能系统为研究对象,其蕴含更加丰富的可重用的领域知识,相应地除了考虑关系模式信息的转换问题,还需要考虑存储于元组集内范例、规则知识项的移植问题.并且,现有相关研究大都处于研究的初始阶段,相应的方法更多地停留在理论层面,基本没有具体的示范性应用研究.

2　数据源和学习目标建模及对应关系分析

为了便于分析和讨论,首先对关系数据库模式和OWL本体进行形式化建模,然后分析关系数据库模式、元组集与OWL本体的对应关系.

2.1　关系数据库模式的形式化建模

基表结构设计和完整性约束申明两部分构成了关系数据库模式的主要内容.基表结构定义了关系(表)的结构、属性(列)及其数据类型与长度等;完整性约束定义了语义施加在数据上的约束,包括关系层的全局约束、元组层的表约束以及属性层的列约束.这两部分信息作为元数据存储于数据库的数据字典中.基于本文研究的问题,定义1给出了一个关系数据库模式的形式化定义.

定义1(关系数据库模式的形式化定义)　$S=(T,D,A,PK,FK,DOM,U,N)$.其中T代表表名的一个有限集合,由实体表(entity table)名的集合ET和联系表(relationship table)名的集合RT组成,且ET和RT不相交(ET拥有单主键,RT拥有复合主键);D代表一个数据类型(data type)名的集合,每个数据类型名是DBMS预定义的类型名;A代表列名的一个有限集合,$a_j(T_i)$代表表T_i的第j列的列名;PK代表主键约束;FK代表外键约束;DOM代表一个特定列$a_j(T_i)$到其取值范围(数据类型)的映射;U代表一个特定列到其值是否惟一约束的映射,取值惟一或不惟一;N代表一个特定列到其值是否可取空值约束的映射.

2.2　OWL本体的形式化建模

一个OWL本体中的大部分元素是与类(class)、属性(property)、类的实例(instance)以及这些实例间的关系有关的[13].其中,领域内的基本概念被表示为拥有不同层次关系的各个类;属性(properties)可用以断言关于类成员的一般事实以及关于个体的具体事实,其又被分为数据类型属性(datatype property)和对象属性(object property)两种.数据类型属性表示类的实例与RDF文字或XML Schema数据类型之间的关系;对象属性表示两个类的实例之间的关系.类的一个实例表示相应概念的一个具体成员.此外,为了更加详尽地说明属性,属性特性(characteristic)提供了一种强有力的机制以增强对于一个属性的推理,属性限制(restriction)可以进一步在一个明确的上下文中限制属性的值域,例如基数限制(cardinality)等.关于OWL构造子的抽象语法及语义等内容见文献[14].

形式上,一个用OWL表示的本体简单地包含一个可选的本体标识符、一组可选的标注(annotation)以及一组公理(axiom).标注用于记录本体的作者身份以及对其他本体的输入引用等信息;本体的主要内容由公理提供,包括类公理、属性公理以及个体公理(即事实).为了方便讨论,这里暂时不考虑与关系模型无关的标注等内容.

以下给出一个简化的 OWL 本体的形式化定义.

定义 2(OWL 本体的形式化定义) $OWL\_O=(CIDo, OPIDo, DPIDo, DTIDo, ITIDo, Caxiom, Paxiom, Iaxiom)$,其中 $CIDo$ 为一个类标识符的集合;$OPIDo$ 为一个对象属性标识符的集合;$DPIDo$ 为一个数据类型属性标识符的集合;$DTIDo$ 为一个数据类型标识符的集合(每个数据类型标识符是 OWL 本体中使用的预定义的 XML Schema 数据类型标识符);$ITIDo$ 为一个个体标识符的集合;$Caxiom$ 为一个类公理集合;$Paxiom$ 为一个属性公理集合;$Iaxiom$ 为一个个体公理集合.

2.3 对应关系分析

根据以上形式化建模分析,发现关系数据库模式、元组集与 OWL 本体元素的对应关系大体如表 1 所示.

表 1 关系数据库模式、元组集与 OWL 本体元素对应关系

Tab. 1 Corresponding relationship among elements of relational database schema, tuple set and OWL ontology

关系数据库模式	OWL 本体
T(ET 型关系表)	$CIDo$ and $Caxiom$ (owl: class)
($a_j(T_i)$ and ! $FK(T_i)$) dom($a_j(T_i)$)	$DPIDo$ and $Paxiom$ (owl:datatypeproperty)
fkey(T) or T(RC 型关系表)	$OPIDo$ and $Paxiom$ (owl:objectproperty)
D (INT, $CHAR$, $FLOAT$…)	$DTID$ (xsd:integer, xsd:string, xsd:float …)
notnull() unique() PK()	$Paxiom$ (restriction such as: cardinality)
$a_j(T_i)=PK(T_i)$ and $a_j(T_i)\in FK(T_i)$referring T_k	$Caxiom$ (class(T_i) subclassof class(T_k))
元组集(范例和规则知识表中的知识项)	$ITIDo$, $Iaxiom$

3 从遗留智能系统学习 OWL 本体

在以上分析的基础上,本文提出的从遗留智能系统中学习 OWL 本体的方法大体分为关系数据库模式信息的提取以及模式信息、知识项向 OWL 本体的转换两步.以下分别阐述.

3.1 关系数据库模式信息提取

一般而言,数据库模式信息既可以直接从数据库的数据字典中提取,也可以通过分析 SQL 数据定义语言(DDL)语句来获得.数据字典中保存的模式信息反映的是数据库当前"最终"模式状态,只要利用能屏蔽各 RDBMS 异构性的中间层 API(如 ODBC API、JDBC API)来设计提取处理逻辑,就可以实现直接、通用的模式提取,因此,本文通过读取数据字典来获取数据库模式信息.

具体方法如下:通过 JDBC API 与数据库建立连接,调用 Connection 对象的 getMetaData () 方法返回一个 DatabaseMetadata 对象;进而调用 java. sql 包中 DatabaseMetadata 接口提供的众多方法获取数据库模式信息.其中主要用到的方法包括 getTables()、getColumns()、getPrimaryKey()、getImportedKey().相应地,模式信息的数据结构如下:

```
class DBMetaData{
  String dbName; //数据库名
  RelationMetaData [ ] allRelations; //所有关系}
class RelationMetaData{
  String relationName; //关系名
  AttrMetaData [ ] attrInfo; //所有属性
  KeyMetaData [ ] keyInfo ; // 所有键
  FKMetaData [ ] fkeyInfo ; // 所有外键}
class FKMetaData {
  String fkeyName ; // 外键名
  String [ ] attrNames ; // 外键属性(组)
  String [ ] refedTable ; // 外键引用的表名(组)
  String [ ] refedAttributes ; // 引用的属性(组)}
class AttrMetaData {
  int attrID; // 属性 ID
  String attrName ; // 属性名
  String attrType ; // 数据类型
  Boolean attrUnique ; // 惟一性约束
  Boolean attrNotNull ; // 非空值约束}
class KeyMetaData {
  String keyName ; // 键名
  String [ ] attrNames ; // 属性(组)}
```

此部分技术相对成熟,可见文献[15].

3.2 模式信息、知识项向 OWL 本体的转换

模式信息、知识项向 OWL 本体的转换从逻辑上可以分为模式转换和知识项移植两部分.

3.2.1 *模式转换* 模式转换完成将关系数据库模式映射成 OWL 本体的过程.基于前述的形式化建模和对应关系分析,本文提出如下一个简单而拥有较好时间性能的模式转换算法:

输入：关系数据库模式 $S=(T,D,A,PK,FK,DOM,U,N)$
输出：OWL 本体 $O=(CIDo,DRIDo,OPIDo,DPIDo,Caxiom,Paxiom)$
for ($i=0$, $i<|T|$, $i++$)
{ if ($|PK(T_i)|=1$)//T_i 为 ET
　{创建一个类标识符 $\varphi(T_i)\in CIDo$;
　创建类公理：〈owl:Class rdf:ID="$\varphi(T_i)$" 〉;}
}
for ($i=0$, $i<|T|$, $i++$)
{ if ($|PK(T_i)|=1$)　//T_i 为 ET
　{ for ($j=0$,$j<|A(T_i)|$,$j++$)
　{ if ($a_j(T_i)\in FK(T_i)$ where $a_j(T_i)$ referring to T_k)
　　{创建对象属性标识符：$\varphi(a_j(T_i))\in OPIDo$ 和 $\varphi(\text{inv\_}a_j(T_i))\in OPIDo$;
　　创建属性公理：ObjectProperty($\varphi(a_j(T_i))$ domain(class(T_i)) range(class(T_k)));
　　if (notnull($a_j(T_i)$)=true) { minCardinality($\varphi(a_j(T_i))$)=1}
　　if (unique($a_j(T_i)$)=true) { maxCardinality($\varphi(a_j(T_i))$)=1}
　　if ($a_j(T_i)=PK(T_i)$) {创建类公理：class(T_i) subclassof class(T_k) }
　　}
　　else　{创建一个数据类型属性标识符 $\varphi(a_j(T_i))\in DPIDo$;
　　创建属性公理：DatatypeProperty($\varphi(a_j(T_i))$ domain(class(T_i)) range(dom(a_j)))
　　if ($a_j(T_i)\in PK(T_i)$) then {Cardinality($\varphi(a_j(T_i))$)=1}
　　else {if (notnull($a_j(T_i)$)=true) {minCardinality($\varphi(a_j(T_i))$)=1}
　　if (unique($a_j(T_i)$)=true){maxCardinality($\varphi(a_j(T_i))$)=1}
　　}
　　}
　}
　}
　else if ($|PK(T_i)|=2$ and $|FK(T_i)|=2$ and pkey1(T_i)$\in FK(T_i)$ where pkey1(T_i) referring to T_k and pkey2(T_i)$\in FK(T_i)$ where pkey1(T_i) referring to T_m)
　//（即 T_i 为 RT，由 T_m 与 T_k 所表示的实体之间 $m:n$ 关系转化而来）
　{创建对象属性标识符：$\varphi(T_i)\in OPIDo$ 和 $\varphi(\text{inv\_}T_i)\in OPIDo$;
　创建属性公理：ObjectProperty($\varphi(T_i)$ domain(class(T_m)) range(class(T_k)));
　ObjectProperty($\varphi(\text{inv\_}T_i)$ domain(class(T_k)) range(class(T_m)) inverseof$\varphi(T_i)$);
　}
　else
　{//T_i 为复杂语义结构表，可通过交互，由领域专家确定处理方案;
　}
}

其中在从字母表到标识符集的转换方面，$\varphi(T_i)$ 的片断标识符和表名 T_i 同名，$\varphi(a_j(T_i))$ 的片断标识符为 has_列名 $a_j(T_i)$，并且各标识符中的片断标识符可通过交互，由领域专家重命名；||操作为取模，例如 $|T|$ 为取得所有关系表的数量. 需要说明的是本算法不过多考虑在一般数据库中极少出现的语义结构过于复杂的表结构信息，仅做出提示，以便由领域专家根据更多的遗留设计文档信息来进行处理.

此算法的时间性能可以做以下一般性分析. 由于全部标识符创建可以在公理创建中直接进行，可认为算法的基本操作为公理的创建. 设一个关系数据库模式 S 的规模 $N=N_T+N_A$，其中 N_T 是表名集 T 的基数，N_A 是列名集 A 的基数. 分析此算法：极端的情况下，数据库中全为实体表，则第 1 个 for 循环创建类标识符和类公理次数最多为 N_T 次；第 2 个大 for 循环分为 E_T 和 R_T 两个部分，对于 E_T 部分创建次数最多为 $4N_A$，对于 R_T 部分创建次数不多于 N_T. 所以，最坏情况下，算法的基本操作总执行次数 $T=N_T+4N_A+N_T<4N$. 故算法的时间复杂度低于 $O(N)$.

3.2.2　*知识项移植*　知识项移植过程是将遗留系统中存储在关系数据库的范例、规则知识项移植到 OWL 本体中的过程. 方法是：将关系数据库中相应实体表的一个元组映射成 OWL 本体的一个个体；把关系数据库元组的属性值（除外键）转换成本体个体相应 owl:DatatypeProperty 的值；根据关系数据库元组的外键创建本体个体之间的关系，把

元组中外键的值映射到一个对应个体，用这个个体作为由外键生成的 owl:ObjectProperty 的值.

根据上述方法，在具体的知识项移植过程中，存在一个移植先后问题. 移植流程如下：先移植无外键实体表的元组集，再移植有外键实体表的元组集；在移植有外键实体表 T_i 的元组集时，先判断其外键引用的实体表 T_j 的元组集是否已经被移植，如果未被移植则先移植 T_j 的元组集，然后再移植 T_i 的元组集.

3.3 本方法与已有相关方法的对比

与已有相关方法相比，本方法针对的数据源蕴含更加丰富的领域知识，并拥有自身的优势. 以下从数据源、提取内容、实现难易等方面进行对比，如表 2 所示.

表 2 本方法与已有相关方法性能对比

Tab. 2 Performance comparison between this approach and existing approaches

方法	中间模型	语义信息提取	数据源	知识项移植	实现难易
文献[2,3]	需要	较多	关系模式	无	较难
文献[4,5]	不需	较少	关系数据库	无	一般
文献[6]	不需	较多	关系数据库	无	较难
文献[7～12]	不需	较多	关系数据库或关系模式	无	一般
本文	不需	较多	特定的关系数据库(遗留智能系统)	有	较易

4 应用实例

现以企业遗留的工装工时定额智能系统为例来评估、检验本文提出的方法.

本遗留智能系统是面向企业级工装工时定额领域，主要任务是在编制好详尽的工艺规程前，辅助工时定额工程师利用总结的个人经验，通过混合推理(CBR 和 RBR 相结合)的方式快速定额出结构复杂的整套装备的工时值，具有很强的预测性、经验性. 系统中以关系表形式存储有大量由工时定额工程师们总结的规则、范例知识.

现以简化的冲孔模对应的范例知识和规则知识片断为例展开讨论. 用 SQL Server 2000 设计的关系数据库模式定义 knowledge_punching_die. sql，见图 1.

```
CREATE DATABASE Knowledge_punching_die;
CREATE TABLE Case_punching_die (
  case_punching_ID char (8) PRMARY KEY,
  punching_die_stru_type char (8) NOT NULL UNIQUE,
  punching_die_surface_roughness char (8) NOT NULL UNIQUE,
  Man_hour float,
  FOREIGN KEY (punching_die_stru_type) REFERENCES
Rule_punching_die_stru_type (punching_die_stru_type_ID
  FOREIGN KEY (punching_die_surface_roughness) REFERENCES
Rule_punching_die_surface_roughness
(punching_die_surface_roughness_ID) );
CREATE TABLE Rule_punching_die_surface_roughness (
  punching_die_surface_roughness_ID char (8) PRMARY KEY,
  punching_die_surface_roughness char (10) NOT NULL UNIQUE,
  punching_die_surface_roughness_pama float NOT NULL UNIQUE);
CREATE TABLE Rule_punching_die_stru_type(
  punching_die_stru_type_ID char (8) PRMARY KEY,
  punching_die_stru_type_name char (10) NOT NULL UNIQUE,
  punching_die_stru_type_pama float NOT NULL UNIQUE);
```

图 1 关系数据库模式定义实例片断

Fig. 1 Instance fraction of relational database schema definition

冲孔模范例表中存储的是每个冲孔模范例对应的各个特征信息值，以及工时定额结果信息. 而冲孔模的众多规则表(例如冲孔模表面粗糙度规则表、冲孔模结构类型规则表等)则存储影响工时定额结果的各个特征信息以及与标准描述相比的影响系数.

显然，可得到此关系数据库模式的形式化定义，其中域符号集合 D={char(8)，char(10)，float}；表名集合 T={Case _punching_die，Rule_punching_die_surface_roughness，Rule_punching_die_stru_type}；列名集合 A={case_punching_ID，punching_die_stru_type，punching_die_surface_roughness，Man_hour，punching_die_surface_roughness_ID，punching_die_surface_roughness，punching_die_surface_roughness_pama，punching_die_stru_type_ID，punching_die_stru_type_name，punching_die_stru_type_pama}.

经过模式信息提取和模式转换，可学习得到 OWL 本体片断如图 2 所示. 以下对输出进行简单说明.

此 OWL 本体片断的公理序列组成如下：3 条类公理 Class(…)；2 条对象属性公理 ObjectProperty(…)；8 条数据类型属性公理 DatatypeProperty(…).

冲孔模范例、规则表内存储的元组信息片断如图 3 所示.

对于以上元组信息片断，通过知识项移植得到冲孔模规则、范例知识项移植结果 OWL 本体片断，如图 4 所示.

```
</owl:Ontology>
<owl:Class rdf:ID="rule_punching_die_stru_type" />
<owl:Class rdf:ID="rule_punching_die_surface_roughness" />
<owl:Class rdf:ID="case_punching_die" />
- <owl:ObjectProperty rdf:ID="has_punching_die_stru_type">
    <rdfs:domain rdf:resource="#case_punching_die" />
    <rdfs:range rdf:resource="#rule_punching_die_stru_type" />
  </owl:ObjectProperty>
- <owl:ObjectProperty rdf:ID="has_punching_die_surface_roughness">
    <rdfs:domain rdf:resource="#case_punching_die" />
    <rdfs:range rdf:resource="#rule_punching_die_surface_roughness" />
  </owl:ObjectProperty>
- <owl:DatatypeProperty rdf:ID="has_case_punching_ID">
    <rdfs:domain rdf:resource="#case_punching_die" />
    <rdfs:range rdf:resource="http://www.w3.org/2001/XMLSchema# Integer" />
  </owl:DatatypeProperty>
- <owl:DatatypeProperty rdf:ID="has_punching_die_surface_roughness_ID">
    <rdfs:domain rdf:resource="#rule_punching_die_surface_roughness" />
    <rdfs:range rdf:resource="http://www.w3.org/2001/XMLSchema# Integer" />
  </owl:DatatypeProperty>
- <owl:DatatypeProperty rdf:ID="has_punching_die_surface_roughness">
    <rdfs:domain rdf:resource="#rule_punching_die_surface_roughness" />
    <rdfs:range rdf:resource="http://www.w3.org/2001/XMLSchema# String" />
  </owl:DatatypeProperty>
```

图 2　关系数据库模式转换结果片断

Fig. 2　Result fraction of relational database schema mapping

case_punching_die

case_punching_ID	punching_die_stru_type	punching_die_surface_roughness	Man_hour
12	18	34	23.4

rule_punching_die_stru_type

punching_die_stru_type_ID	punching_die_stru_type_name	punching_die_stru_type_pama
18	Punching_die_1	0.7

rule_punching_die_surface_roughness

punching_die_surface_roughness_ID	punching_die_surface_roughness	punching_die_surface_roughness_pama
34	3.2	0.65

图 3　冲孔模范例、规则实例的元组信息片断

Fig. 3　Tuple set fraction of punching dies cases and instances of rules

```
- <rule_punching_die_stru_type rdf:ID="rule_punching_die_stru_type_18">
    <has_punching_die_stru_type_ID>18</has_punching_die_stru_type_ID>
    <has_punching_die_stru_type_name>punching_die_1</has_punching_die_stru_type_name>
    <has_punching_die_stru_type_pama>0.7</has_punching_die_stru_type_pama>
  </rule_punching_die_stru_type>
- <rule_punching_die_surface_roughness rdf:ID="rule_surface_roughness_34">
    <has_punching_die_surface_roughness_ID>34</has_punching_die_surface_roughness_ID>
    <has_punching_die_surface_roughness>3.2</has_punching_die_surface_roughness>
    <has_punching_die_surface_roughness_pama>0.65</has_punching_die_surface_roughness_pama>
  </rule_punching_die_surface_roughness>
- <case_punching_die rdf:ID="case_punching_die_12">
    <has_case_punching_ID>12</has_case_punching_ID>
    <has_punching_die_stru_type rdf:resource="#rule_punching_die_stru_type_18" />
    <has_punching_die_surface_roughness rdf:resource="#rule_surface_roughness_34" />
    <has_man_hour>23.4</has_man_hour>
  </case_punching_die>
```

图 4　知识项移植结果片断

Fig. 4　Result fraction of knowledge items transform

对照算法可以看出，以上输出就是算法理论上应有的结果，这说明了算法的实现是正确的. 通过分析发现，采用本文提出的方法学习得到的 OWL 本体与利用本体建模工具 Protégé 2000 手工构建的工装工时定额领域知识本体(TDO)[16]的对应部分基本类似，很好地证明了本方法的有效性.

5　结　语

本文提出的从遗留智能系统学习 OWL 本体的方法具有以下优点：结构清晰、步骤简单，数据源蕴含更丰富的领域知识，更加适合实际的工程应用；通过一个简单、低时间复杂度的转换算法，而非中间模型或大量抽象规则，完成从关系数据库模式中自动获取 OWL 本体，易于实现；不但将关系模式信息映射为 OWL 本体的相应部分，而且将遗留系统数据库中大量的规则、范例知识项直接转化为相应的本体个体，最大限度地重用已有的领域知识，便于实现知识共享.

在后续研究工作中，如何实现从领域内多个不同遗留智能系统中学习得到的本体间的集成将成为研究的重点.

参考文献：

[1] 张友生. 遗留系统的评价方法和进化策略[J]. 计算机工程与应用，2003，**38**(13)：29-35

[2] UPADHYAYA S R，KUMAR P S. ERONTO：A tool for extracting ontologies from extended E/R diagrams [C] // **Proceedings of the 2005 ACM Symposium Applied Computing**. New York：ACM，2005

[3] KASHYAP V. Design and creation of ontologies for environmental information retrieval [C] // **Proceedings of the 12th Workshop on Knowledge Acquisition，Modeling and Management**. Canada：Springer-Verlag，1999

[4] TRINH Q，BARKER K，ALHAJJ R. RDB2ONT：A tool for generating OWL ontologies from relational database systems [C] // **Proceedings of the Advanced International Conference on Telecommunications and International Conference on Internet and Web Applications and Services**. Washington D C：IEEE Computer Society，2006

[5] TRINH Q，BARKER K，ALHAJJ R. Semantic interoperability between relational database systems [C] // **11th International Database Engineering and Applications Symposium**. Canada：IEEE Press，2007

[6] 曹泽文，张维明. 一种从关系数据库向 Flogic 本体转换的方法[J]. 计算机科学，2007，**34**(4)：149-155

[7] LI M，DU X Y. Learning ontology from relational database [C] // **Proceedings of the Fourth International Conference on Machine Learning and**

Cybernetics. Guangzhou:IEEE Press, 2005

[8] 许卓明,王 琦. 一种从关系数据库学习 OWL 本体的方法[J]. 河海大学学报, 2006, **34**(2):208-211

[9] 王洪伟,伊 磊. 本体模型的逆向获取研究[J]. 计算机科学, 2007, **34**(2):166-170

[10] DOGAN G, ISLAMAJ R. Importing relational databases into the semantic Web [EB/OL]. [2008-03-12]. http://www.mindswap.org/webai/2002/ fall/Importing_20Relational_20Databases_20into_20the_20Semantic_20Web.html

[11] ASTROVA I. Reverse engineering of relational databases to ontologies [M] // **Lecture Notes in Computer Science**. Berlin:Springer, 2004:327-341

[12] ASTROVA I. Extracting ontologies from relational databases [C] // **Proceedings of the 22nd IASTED International Conference on Databases and Applications**. Austria:IASTED, 2004

[13] SMITH M K, WELTY C, MCGUINNESS D L. OWL Web ontology language guide [M/OL]. (2004-02-10). http://www.w3.org/TR/owl-guide/

[14] HAYES P, HORROCKS I. OWL Web ontology language semantics and abstract syntax [M/OL]. (2004-02-10). http://www.w3.org/TR/owl-semantics/

[15] 许卓明, 苏文萍. 关系数据库模式信息的提取[J]. 河海大学学报, 2005, **33**(2):202-206

[16] GAO Jun-jie, DENG Gui-shi. The research of applying domain ontology to case-based reasoning system [C] // **Proceedings of International Conference on Services Systems and Services Management**. Chongqing:IEEE Press, 2005

An approach to learning OWL ontology from enterprise legacy intelligent systems

GAO Jun-jie, DENG Gui-shi*

(Institute of Systems Engineering, Dalian University of Technology, Dalian 116024, China)

Abstract: In order to get around ambiguities in Chinese word segmentation and reuse a lot of domain knowledge in enterprise legacy intelligent system, a novel approach to learning OWL ontology from legacy intelligent systems is proposed. On the basis of analyzing the element correspondence between relational database schema, tuple set and OWL ontology, the approach is specified in detail. Compared with existing methods, the approach is more appropriate for actual engineering application and its data source implies more domain knowledge. OWL ontology from legacy intelligent systems can be acquired automatically via a simple translation algorithm with low time-complexity instead of using a middle model or a lot of abstract learning rules, and numerous knowledge items about rules and cases (tuple set) can be reused as instances of OWL ontology according to certain priority. Validation of the approach is done by an application instance learning OWL ontology from a legacy intelligent system in the wide range of tooling man-hour rationing.

Key words: legacy intelligent system; ontology learning; man-hour rationing; OWL