基于关系数据库的OWL本体构建方法的研究

吕艳辉[1,2] 马宗民[2] 王玉喜[2]

(沈阳理工大学信息科学与工程学院 沈阳 110168)[1] (东北大学信息科学与工程学院 沈阳 110004)[2]

摘 要 利用已有的数据资源以自动或半自动方式构建本体是实现语义Web的任务之一。在分析了现有研究成果及不足的基础上,给出了一个比较系统的基于关系数据库的OWL本体构建方法。介绍了如何从关系模式中识别实体、联系、继承关系、聚类关系及基数约束等语义,完成了从关系数据库语义到本体相应部分的转换,通过原型系统的实现验证了该方法的有效性。

关键词 关系数据库,Web本体语言,语义Web,本体构建

Study on the OWL Ontology Construction Approach Based on Relational Databases

LU Yan-hui[1,2] MA Zong-min[2] WANG Yu-xi[2]

(College of Information Science and Engineering, Shenyang Ligong University, Shenyang 110168, China)[1]

(College of Information Science and Engineering, Northeastern University, Shenyang 110004, China)[2]

Abstract How to make use of the existing data resources to construct ontology(semi)automatically is one of the tasks of achieving Semantic Web. By analyzing the current research results and deficiency, a systematic approach to OWL ontology construction based on RDB was presented. Described the processing steps of extracting the semantic information from RDB including entity, relation, inheritance, aggregation and cardinality ratio constraint. Accomplished the transformation from RDB semantic information to corresponding elements of ontology. Prototype system implementation demonstrates the effectiveness of the approach.

Keywords RDB, OWL, Semantic Web, Ontology construction

1 引言

现在,互联网已成为人们获取信息最重要的途径,其规模也以惊人的速度增长着。网页的内容从原来的以超文本为主,发展到后来的以动态页面数据为主,动态页面的数据量约为直接可见页面数据量的400~500倍,它们都存储在数据库中。动态页面中的数据以及互联网上的绝大多数信息是以人类能够理解的格式来表示的,而作为智能程序的软件代理却不能理解和处理这些信息。为解决这个问题,T. Berners-Lee提出语义Web(Semantic Web)[1]作为万维网(World Wide Web)的扩展。在语义Web上,信息是以结构化的形式表示的,而本体(ontology)则描述了其中的语义。当信息用本体标记后,软件代理就能理解其意义,也就可以自动完成互联网上的信息收集和集成。语义Web的实现很大程度上依赖于本体的建立,同时本体也越来越广泛地应用到很多领域,如信息检索、机器翻译、知识管理、电子商务和信息集成等。

近年来,计算机科学中关于本体的研究越来越多。所谓本体,被广泛引用的定义是:"本体是一个共享概念化的形式化、显式的说明"[2]。它通常表示为一组对象(概念)、关系、函数、定理和实例。目前已经出现的一些本体构建工具能够提供比较友好的图形化界面和一致性检查机制,使得用户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,避免了很多错误的发生,方便了本体的构建。但是,这些工具仅仅能提供本体编辑功能,支持的仍然是手工构建本体的方式,用户依然需要逐个地输入和编辑每个概念的名字、约束、属性等内容。由于手工方法费时、费力,使得本体的构建成为一项艰巨的任务。因此,从现有数据资源获取领域知识、以(半)自动方式构造本体,是开发本体的有效途径。

近几年来,利用数据库这种结构化数据源生成语义Web本体的技术得到了广泛的研究,主要集中在对关系模式进行语义分析,获取构建本体所需的概念和关系。例如,文献[3]提出基于领域数据库的关系模式设计本体的方法,文献[4]在Wonder Web项目中开发了从关系数据库模式资源中抽取轻量级本体的功能,文献[5]提出了从SQL关系数据库到本体的转换规则,文献[6,7]对关系数据库到本体的语义映射及推理进行了研究,文献[8,9]给出了一组从关系数据库模式到OWL本体的通用映射规则以及从ER模式到OWL DL本体语义保持的翻译方法,文献[10]设计了一种从关系数据库自动提取本体的方法,文献[11]提出了从关系数据库向Flogic本体转换的方法。现有方法丰富了利用数据库语义构建本体的技术,但也存在一些不足,主要表现在:不能充分表达关系模式所蕴涵的语义信息;仅在数据模式与本体之间建立映射关系,没有考虑到元组转换过程中的语义保持问题。

本文借鉴现有研究成果以及不足之处,提出一个比较系

到稿日期:2008-06-22 返修日期:2009-04-29 本文受教育部新世纪优秀人才支持计划(NCET20520288)资助。

吕艳辉 女,博士研究生,副教授;马宗民 男,教授,博士生导师,CCF高级会员;王玉喜 硕士研究生。

统的从关系数据库数据获取 OWL 本体的方法。与现有方法相比，不但对关系模式中隐含的继承与聚类语义信息进行了处理，同时还对元组到 OWL 本体个体转换的一些问题给出了解决方法，并通过原型系统的实现验证了该方法的有效性。

2 Web 本体语言 OWL

本体作为语义 Web 的核心技术是解决语义层次上信息共享和交换的基础。Web 本体语言 OWL[12] 作为 W3C 的推荐标准，旨在提供一种语言来描述网络文档和应用中固有的类和类之间的关系。它通过定义类和类的属性来形式化一个领域，声明和定义对象和对象的属性，以及在 OWL 形式化语义允许的程度上对类和个体进行推理。

OWL 语言提供了 3 种表达能力依次增强的子语言：OWL-Lite，OWL-DL 和 OWL-Full，其中 OWL DL 支持那些需要最强表达能力的推理系统的用户，且这个推理系统能够保证计算的完全性和可判定性。

2.1 OWL DL 的基本元素

(1)类 Class。类定义了一组因共有某些属性而同属于一个集合的个体，它提供了一个将具有相似特点的资源聚集在一起的机制。

(2)属性 Property。一个属性是一个二元关系，有数据类型属性与对象属性 2 种。属性具有层次关系，有定义域和值域，属性约束有取值约束与属性基数限制 2 种。

(3)个体 Individual。个体是用个体公理来定义的，一种是通过个体的类成员关系及属性值来定义个体，另一种是通过个体身份。

(4)基本数据类型 Datatypes。OWL 使用 XML Schema datatype 作为自己的基本数据类型。

3 基于关系数据库的 OWL 本体的构建

与关系数据模型相比，本体是一种具有更多语义、结构更为复杂的模型，基于关系数据库的本体构建任务就是分析关系数据模型中蕴涵的语义信息，将其映射到本体中的相应部分。在关系数据模型中，实体以及实体间的联系都是用表来表示的，所以，无论是概念的获取还是概念间联系的获取，首先要区分出哪些表是用来描述实体的，哪些表是用来描述实体间联系的，然后才能将实体信息映射为本体中的概念，将联系信息映射为本体中的关系。

3.1 识别关系数据库语义

关系数据库由两部分数据组成，一部分是元数据，也称为模式，如关系(表)名、关系的属性名、数据类型名、主键和外键等完整性约束名；另一部分是真正的数据，从数据粒度上又可以分为元组和属性值。识别关系数据库的语义包括识别出其包含的实体、联系及联系类型等内容。

为方便介绍，这里设计了一个关于银行存贷款业务的数据库 bankbusiness。表 1 给出了它的关系模式，分别表示关系：账号、客户、银行、贷款、储蓄账户、支票账户、开户及支付。

(1) 识别实体

一种情况，如果关系的主键仅有一个属性，那么这个关系对应着一个实体。另外一种情况，如果关系的主键有 1 个以上的属性，且其中至少有一个属性不是外键，那么这个关系也对应着一个实体。

表 1 中，除 Establish 关系外，其它关系都是实体。

表 1 关系模示例表

关系名	属性	主键	外键及参照关系
Account	AccID, CusID, balance	AccID	CusID(Customer)
Customer	CusID, name, city	CusID	no
Subbank	BankID, bankname	BankID	no
Loan	LoaID, CusID, L-amount	LoaID	CusID(Customer)
Saving-Acc	AccID, inter-rate	AccID	AccID(Account)
Checking-Acc	AccID, overdraft-num	AccID	AccID{Account
Establish	BankID, CusID, D-date	BankID, CusID	BankID(Subbank), CusID(Customer)
Payment	LoaID, PayID, date, amount	LoaID, PayID	LoaID(Loan)

(2) 识别联系

一种情况，关系的主键包含一个以上的属性，所有主属性又都是关系的外键，这样的关系不表示实体，而是对应着 2 个实体间的 1 个联系，其中两个实体分别是关系中外键指向的实体(对于 3 个以上实体间的联系，可以简化成 2 个实体间的联系)。另外一种情况，一个实体与其对应关系外键所指向的实体之间存在联系，但是如果这个实体关系的外键同时又是主键的情况除外，事实上这种实体联系属于继承关系。

表 1 中的 Establish 关系是一个联系，表示 Customer 和 Subbank 2 个实体之间的关联关系。Account 关系的外键 CusID 表示 Account 和 Customer 存在引用关系，同样 Loan 和 Customer 也有引用关系。

(3) 识别继承关系

如果实体关系 ET2 的主键属性集合包含实体关系 ET1 主键属性集合，并且 ET1 的主键同时又是自己的外键，所有外键都指向 ET2，那么这两个实体之间存在着继承关系，即 ET1 IS-A ET2。

表 1 中 Account 和 Saving-Acc 以及 Account 和 Checking-Acc 是继承关系，Saving-Acc 和 Checking-Acc 是 Account 的子类。

(4) 识别聚类关系

如果实体关系 ET1 有 1 个以上的主键属性，这些主键属性中至少有 1 个不是外键，ET1 所有外键都是实体关系 ET2 的主键属性，那么这 2 个实体之间存在着聚类关系，即 ET1 IS-PART-OF ET2。这种聚类关系是由弱实体引起的，这里 ET1 是一个依赖于 ET2 的弱实体。

上例中 Loan 和 Payment 之间是聚类关系，Payment 是一个弱实体。

(5) 描述基数约束

一种情况，实体 ET1 和 ET2 之间有联系，ET1 的主键是 ET2 的外键，如果 ET2 的外键值可以重复，说明每个 ET1 实例可以对应多个 ET2 实例，ET1 与 ET2 是一对多(1:n)联系；如果 ET2 的外键值不能重复，说明每个 ET1 实例正好对应一个 ET2 实例，则 ET1 与 ET2 是一对一(1:1)联系。若一个关系不对应一个实体，而是表示 2 个实体间的联系，则这个联系是多对多(m:n)联系。

表 1 中 Account 中的外键 CusID 值如果可以重复，表明一个客户可以办理多个账号，即 Customer 与 Account 之间是一对多的联系。如果 Loan 中的外键 CusID 值不可以重复，表明 Loan 与 Customer 之间是一对一的联系，即一个客户在一个银行内最多有一笔贷款(只有还清上次贷款，才能再贷

款)。Loan 与 Payment 是一对多的联系,表明客户可以分多次支付一笔贷款。关系 Establish 对应实体 Subbank 和 Customer 多对多的联系,表明一个客户可以在多个银行开户,一个银行可以有多个客户开户。

3.2 从关系数据库语义到本体元素的转换

识别关系数据库的语义是为了使之更接近于本体的表达习惯,进而与 Web 本体语言 OWL DL 具体语法建立映射关系,实现向 OWL 本体的转换。

(1) 从实体到类的转换

关系模式中的实体,对应于 OWL DL 中的类,用 owl:class 来声明。下面代码段声明实体 Account 为类。

```
〈owl:class rdf:ID="Account"/〉
```

(2) 从多对多联系到类的转换

对两个实体间的多对多联系,若这个联系除了表达两个实体的属性外,还有联系自身的属性,则需要在 OWL DL 中创建一个类;若这个联系没有自身属性,则不需要单独创建类,可以通过建立对象属性表示联系语义。

(3) 从外键属性到对象属性的转换

外键属性表示两个实体间的关联关系,这种语义与 OWL DL 中的对象属性相同,用 owl:objectProperty 来声明。在 OWL DL 中,要为除子类以外的所有实体关系中的所有外键属性建立对象属性。在 OWL DL 中属性有定义域和值域,对象属性的定义域就是外键所在关系对应的类,值域是外键所指向的参考关系对应的类。下面代码段描述了实体 Account 的外键 customerID 是对象属性。

```
〈owl:objectProperty rdf:ID="Account. customerID"〉
 〈rdfs:domain rdf:resource="#Account"/〉
 〈rdfs:range rdf:resource="#Customer"/〉
〈/owl:ObjectProperty〉
```

(4) 从实体的非外键属性和子类、有自身属性的联系实体的所有属性到数据类型属性的转换

a. 对于实体对应的关系中的所有非外键属性,它们的取值不受其它实体的限制,在 OWL DL 中与数据类型属性类似,用 owl:dtatTypeProperty 声明,其定义域为非外键属性所在关系对应的类,值域为相应的数据类型。下面代码段描述了实体 Payment 的属性 paymentID 是数据类型属性。

```
〈owl:DatatypeProperty rdf:ID="Payment. paymentID"〉
 〈rdfs:domain rdf:resource="#Payment"/〉
 〈rdfs:range rdf:resource="&xsd;int"/〉
〈/owl:DatatypeProperty〉
```

b. 子类表中的外键表达的是与超类之间的继承关系,在 OWL DL 中用 owl:subclass 声明,不需要通过建立对象属性来表达这种语义关系。子类表的所有属性在 OWL DL 中对应数据类型属性。

c. 把有自身属性的多对多联系的所有属性转换成 OWL DL 的数据类型属性。也就是说,对有自身属性的联系,不但要将外键属性转换成对象属性,还要将所有的属性转换换成数据类型属性。

(5) 从继承关系到子类关系的转换

把识别出的继承关系用 owl:subclass 来声明,在 OWL DL 中表示类之间的层次关系。下面代码段描述了实体 Saving-Acc 是实体 Account 的子类。

```
〈owl:class rdf:ID="Saving-Acc"〉
 〈owl:subclass rdf:about="#Account"〉
〈/owl:class〉
```

(6) 从聚类关系到自定义属性的转换

实体间的聚类关系,体现了实体间的依赖关系,依赖实体是一个弱实体,必须依赖主实体才能存在。在 OWL DL 中没有内置词汇表示类之间的这种关系,但 OWL DL 支持用户自定义属性。这里定义一个属性 depend,因为要表达类之间的关系,它应该是一个对象属性。定义如下:

```
〈owl:ObjectProperty rdf:about="#depend"〉
  〈rdfs:range rdf:resource="&owl;Thing"/〉
  〈rdfs:domain rdf:resource="&owl;Thing"/〉
〈/owl:ObjectProperty〉
```

用自定义属性 depend 声明特定的属性具有依赖特性,即声明特定属性为 depend 的子属性。这种用户自定义属性,可以为将来的本体构建工具提供必要的语义线索,使得本体构建工具能够把握领域语义,从而进行准确的推理判断。下面代码段声明了实体 Payment 的属性 loanID 为 depend 的子属性。

```
〈owl:ObjectProperty rdf:about="#Payment. loanID"〉
 〈rdfs:domain rdf:resource="#Payment"/〉
 〈rdfs:range rdf:resource="#Loan"/〉
  〈rdfs:subPropertyOf rdf:resource="#depend"/〉
〈/owl:ObjectProperty〉
```

(7) 基数约束的转换

实体之间的联系可分为一对一,一对多和多对多三种类型,其中多对多联系已转换成 OWL DL 中的类,所以只处理前两种情形。

在 OWL DL 中,有 3 种构造子可以用来表达基数限制,分别为 owl:minCardinality,owl:maxCardinality 与 owl:cardinality,这 3 个语义元素可以表达实体间的一对一和一对多联系。其中 owl:cardinality 允许对一个关系中的元素数目作出精确的限制;而 owl:minCardinality 能够用来指定一个下界,owl:maxCardinality 能够用来指定一个上界,二者的组合就能够将一个属性的基数限制为一个数值区间。

(8) 基本数据类型的转换

将所使用的数据库系统的数据类型转换成 OWL DL 的基本数据类型。

(9) 从元组到个体的转换

数据库元组转换成 OWL DL 个体后,通常生成的个体数量非常多,因此如何为个体命名,是元组到个体的转换过程中首先要考虑的问题。一般可以考虑把元组的主键属性值转换成字符串格式,作为该个体的个体名。

a. 个体的命名

如果关系中只有一个主键,那么该元组在主键属性字段的值转换成字符串类型后,作为在 OWL DL 中生成的个体名;如果关系中有两个或以上的主键,就把所有主键属性值转换成字符串类型,这些字符串合起来组成的新字符串,作为 OWL DL 中该元组对应的个体名。

b. 实体元组到个体的转换

将数据库的元组转换成 OWL DL 的个体,就是为每个元组创建一个对应类的实例。因为每个实体的所有属性列,都

对应 OWL DL 类的相应属性(包括对象属性和数据类型属性),所以类的每个个体的所有属性与相应元组的属性列一一对应。个体对象属性的取值就是该元组外键指向的参考关系对应类的个体;个体数据类型属性的取值就是元组在相应属性字段的值转换成 OWL DL 基本数据类型后的值,但是有些情况可能需要根据个体的属性类型作相应调整。

4 实现

原型系统以 MySql 数据库为数据源,选用 JAVA 编程语言,使用 OWL API[13],采用 Pellet[14] 推理机对所生成的 OWL 文档进行一致性检查。开发原型系统包括以下工作:

(1)创建 MySql 数据库,录入实验数据;

(2)建立 MySql 与 OWL API 常用数据类型对应关系;

(3)连接数据库,获取关系数据模式元数据;

(4)分析关系模式,识别语义元素;

(5)通过转换方法,实现关系模式和元组到 OWL 本体的转换;

(6)OWL 本体的生成和存储;

(7)设计图形界面。

为节省篇幅,这里只给出原型系统的操作界面,如图 1 所示。操作界面分菜单栏、按钮栏、数据库信息显示区、转换过程中词汇更改区、OWL 本体显示区及 OWL 本体一致性检查区。

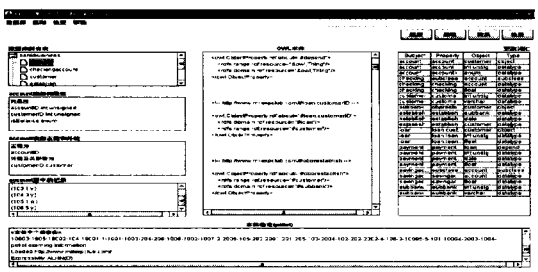

图 1 原型系统操作界面图

其中"连接"操作是与所选数据库建立连接并获取关系模式数据。"提取"操作是将前面获取的关系模式数据进行转换,识别其包含的所有语义,显示在更改词汇面板上,同时,用户可以在更改词汇面板上进行修改词汇的操作。"转换"操作是将关系模式的语义进行相应的转换,生成 OWL 本体。"检查"操作是使用 Pellet 对新生成的 OWL 本体进行一致性检查。通过本文介绍的方法生成的 OWL 本体,一方面可以在本体显示区内显示,另一方面可以 OWL 文档的格式存储在文件系统中。

数据在数据库里是严格按照结构格式存储的,同一个信息可能被分散存储在许多关系(表)中,结构比较复杂。在 OWL 本体中信息被存放在一个文档里,呈线性结构,相对比较简单。图 2 和图 3 分别显示了对类 subbank 和类 checkingaccount 的一个个体进行描述的代码段。

图 2 中的个体 10002 是类 subbank 的一个实例,在关系数据库中这个实体在关系(表)subbank 中对应一个元组,元组的结构是:bankID 值为 10002,bankname 值为 beishi。这个实体在关系(表)establish 中对应于 3 个元组,表达的语义是,这个银行中的 3 名客户,分别是"1"、"3"和"4"。

```
587 〈www:subbank rdf:about="#10002"〉
588 〈www:subbank.bankID rdf:datatype="&xsd;int"〉10002
〈/www:subbank.bankID〉
589 〈www:obprestablish rdf:resource="#1"〉
590 〈www:subbank.bankname
591 rdf:datatype="&xsd;string"〉beishi〈/www:subbank.bankname〉
592 〈www:obprestablish rdf:resource="#3"/〉
593 〈www:obprestablish rdf:resource="#4"/〉
594 〈/www:subbank〉
```

图 2 类 subbank 的一个个体描述

从图 2 可以看出,原本在数据库两个表中关于实体 10002 的信息,在 OWL 本体中,以线性的结构表示出来了。代码中两个数据类型属性 subbank.bankID 和 subbankname,是由表 subbank 中的属性转换而来,而对象属性 obprestablish 是由关联关系(表)establish 转换而来。

```
765 〈wwww:account rdf:about="#106"〉
765 〈rdf:type rdf:resource="#checkingaccount"/〉
766 〈www:checkingaccount,overdraftNumber
767 rdf:datatype="&xsd;float"〉22.8〈/www:checkingaccount.overdraftNumber〉
768 〈www:account,isBalance rdf:datatype="&xsd;boolean"〉true〈/www:account,isBalance〉
769 〈www:account,accountID rdf:datatype="&xsd;int"〉106〈/www:account,account ID〉
770 〈www:account,customerID rdf:resource="#5"/〉
771 〈www:checkingaccount,accountID
772 rdf:datatype="&xsd;int"〉106〈/www:checkingaccount,accountID〉
773 〈/www:account〉
```

图 3 类 checkingaccount 的一个个体描述

图 3 中的个体 106 是类 checkingaccount 的一个实例,在数据库中对应于关系(表)checkingaccount 的一个元组,这个元组仅有 accountID 和 overdraftNumber 两个属性。由于类 checkingaccount 是类 account 的子类,父类中的所有属性同样也是子类的属性,因此关系(表)account 中实体 106 对应的元组的信息,由属性 accountID,customerID,isBalance 表达,在 OWL 本体中也被描述出来了。

结束语 本文给出了一个比较系统的利用关系数据库的数据资源来构建本体的方法。该方法首先识别出关系模式所包含的语义,即实体、联系、继承关系、聚类关系以及基数约束,然后将其转换为本体的语义要素。在处理元组数据的转换过程中,针对个体命名及个体对象属性的取值、数据类型属性的取值给出了相应的解决方法。原型系统的实现证明了基于本文介绍的方法,可以完成从关系数据库到相应 OWL 本体的转换,并能够在语义 Web 上进行简单的操作。

OWL 本体在关系数据库领域的研究和应用还处于起步阶段,许多问题还需要进一步的研究。例如,有关数据库中大量元组数据转换时的临时存储问题以及最后生成的 OWL 本体的存储问题。此外,通过使用关系数据库转换的 OWL 本体来解决网络环境下的数据库语义冲突也是一个未来的研究方向。

参 考 文 献

[1] Berners-Lee T, Handler J, Lassila O. The Semantic Web[J]. Scientific American, 2001, 284(5): 34-43

(下转第 226 页)

铃声	64和弦	支持MP3铃声	支持MIDI	支持AAC
信息功能	SMS短信	EMS短信	MMS短信	
E-mail	POP3	IMAP4	SMTP	
输入法	T9中文	T9英文	中文	

表4 手机功能设计的Pareto优化解

功能元素	方案1	方案2	方案3
主观评定	好	较好	优
蓝牙耳机	230	529	385
数据线	58	139	100
手机耳机	80	159	170
内存扩展卡	130	200	260
电子地图	249	249	249

结束语 概念设计中经常存在着多目标的需求和显性、隐性相混合的性能指标。针对这一问题，本文将协同进化算法和交互式遗传算法应用到手机概念设计方案求解中。协同进化算法可在较大范围内搜索解空间，并能以较快的收敛速度提供解空间内分布均匀的Pareto最优解集。利用交互式遗传算法提供的对隐性性能指标的定性评价较好地解决了复杂产品概念设计中的人机结合问题，为概念设计中的方案求解带来了新的思路。本文提出的方法也同样适用于产品概念设计中的布局设计、外形设计等。

参考文献

[1] Pahl G, Beitz W. Engineering. Design-A Systematic Approach [J]. Springer, 1996

[2] 刘希玉，王文平，刘弘，等. 动态小生境微粒群优化技术在概念设计中的应用[J]. 计算机科学，2006，33(10)：163-168，209

[3] 崔逊学. 多目标进化算法及其应用[M]. 北京：国防工业出版社，2006

[4] 黄洪钟，赵正佳，等. 基于遗传算法的方案智能优化设计[J]. 计算机辅助设计与图形学学报，2002，14(5)：437-441

[5] 吴子燕，张智，胡秦. 基于多目标遗传算法的高层建筑概念设计优化[J]. 系统工程理论与实践，2005，10：120-124

[6] Liu Xiyu, Liu Hong, Duan Huichuan. Particle swarm optimization based on dynamic niche technology with applications to conceptual design [J]. Advances in Engineering Software, 2007, 38：668-676

[7] Hillis W D. Co-evolving parasites improve simulated evolution as an optimization procedure [D]. Physica D, 1990, 42：228-234

[8] Potter M A, De Jong K A. A Cooperative Coevolutionary Approach to Function Optimization[C]// Proceeding of the Third Parallel Problem Solving from Nature. Springer-Verlag, 1994：249-257

[9] Potter M A, De Jong K A. Cooperative Coevolution：An Architecture for Evolving Coadapted Subcomponents [J]. Evolutionary Computation, 2000, 8(1)：1-29

[10] 刘建成，蒋新华，吴今培. 广义模糊模型的协同进化方法研究[J]. 计算机学报，2006，29(3)：423-430

[11] 肖人彬，刘勇，梅顺齐，等. 基于多粒度共进化功能推理的机械运动方案设计新方法[J]. 机械工程学报，2005，41(12)：108-117

[12] Jin Y, Li W, Lu S C-Y. A Hierarchical Co-evolutionary Approach to Conceptual Design [J]. CIRP Annals-Manufacturing Technology, 2005, 54(1)：155-158

[13] 巩敦卫，郝国生，周勇，等. 交互式遗传算法原理及其应用[M]. 北京：国防工业出版社，2007

[14] Goldberg D E, Korb B, Deb K. Messy genetic algorithms：motivation, analysis, and first results [J]. Complex Systems, 1989, 3：493-530

[15] 张文志，吕恬生. 基于改进的遗传算法和模糊逻辑控制的移动机器人导航[J]. 机器人，2003，25(1)：1-6

[16] Van Veldhuizen D A, Lamont G B. Multiobjective Optimization with Messy Genetic Algorithms[C]// Proceedings of the 2000 ACM Symposium on Applied Computing. Como, Italy, 2000

[17] Walters G A, Halhal D, et al. Improved design of "Anytown" distribution network using structured messy genetic algorithms [J]. Urban Water, 1999, 1：23-38

(上接第156页)

[2] Gruber T R. A Translation Approach to Portable Ontology Specifications[R]. Knowledge System Laboratory KSL 92-71. 1993

[3] Kashyap V. Design and Creation of Ontologies for Environmental Information Retrieval[C]// Proc. Workshop on Knowledge Acquisition, Modeling and Management. Alberta Canada, 1999

[4] Stojanovic L, Stojanovic N, Volz R. Migrating Data-Intensive Web Sites into the Semantic Web[C]// Proc. ACM Symposium on Applied Computing. Madrid Spain, 2002

[5] Astrova I. Reverse Engineering of Relational Database to Ontologies[C]// Proc. European Semantic Web Conference. Heraklion, Greece, 2004

[6] An Y, Borgida A, Mylopoulos J. Inferring Complex Semantic Mappings between Relational Tables and Ontologies from Simple Correspondences[C]// Proc. OTM Confederated International Conferences CoopIS, DOA, and ODBASE. Agia Napa, Cyprus, 2005

[7] An Y, Borgida A, Mylopoulos J. Refining Semantic Mappings from Relational Tables to Ontologies[C]// Proc. Semantic Web and Databases. Toronto, Canada, 2004

[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] 许卓明，董逸生，陆阳. 从ER模式到OWL DL本体的语义保持的翻译[J]. 计算机学报，2006，29(10)：1786-1796

[10] 赵荣娟，王丹. 一种从关系数据库提取本体的方法[J]. 微电子学与计算机，2006，23(增刊)：116-118

[11] 曹泽文，张维明，邓苏，等. 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)：149-153

[12] Bechhofer S, van Harmelen F, Hendler J. OWL Web Ontology Language Reference[R]. http://www.w3.org/TR/owl-ref/, 2004

[13] OWL API[OL]. http://owlapi.sourceforge.net/index.html

[14] Pellet[OL]. http://pellet.owldl.com