面向概念的通用知识获取系统

王申康

(浙江大学人工智能研究所, 杭州)

CONCEPTUALLY ORIENTED DESIGN GENERIC KNOWLEDGE ACQUISITION SYSTEM

Wang Shenkang

(*AI institute, Zhejiang University, Hangzhou*)

ABSTRACT

The conceptually oriented design knowledge acquisition system described in this paper is a very general knowledge acquisition enviroment which understands basic logic and language. The system contains ideas from artificial intellegence (ai), object—oriented programming (oop) and logic programming (lp). Knowledge is organized into small unit termed conceptual Descriptors (cds) in the system which are analogous to frames in ai or objects in oop. Cds are defined by a number of properties such as attributes, logic constraints and functional equations, and arranged in inheritance hierarchies. There are some semntic subsystems such as a first order logic subsystem and simple natural language subsystem which are used to perform various types of semantic checks and debug logically redundancy, contradication and non—consistence about knowledge. The system is intended to assist in the process of developing descriptions, definitions, or specifications of system under design, or to describe existing system for documentation or teaching purposes. Typical applications would include most kinds of expert system and software system.

摘 要

本文提出的面向概念的知识获取系统是一个能理解基本逻辑和自然语言的通用知识获取环境。系统结合了人工智能(ai)、面向对象的程序设计(oop)和逻辑程序设计(Lp)等技术。系统以概念为知识基元，由概念描述器(cd)予以描述。cd是一个类框架和类对象

1990年1月20日收到，1990年5月12日定稿。

的结构，它由一组概念特性和属性，逻辑约束和函数式等支持。这些特性可由具体的应用而赋于实际的含义。知识库中的概念集是一个层次式的继承网络。另外系统还附有一些语义子系统，如一阶逻辑系统和简单的自然语言系统，用以各种类型的语法检查及知识的冗余、互斥及非一致性检查。系统主要用于辅助专家开发新系统的描述、定义和说明。也可对已有的系统、文件文本进行描述并用于教育。最典型的应用是软件系统和专家系统的开发。

§1. 引言

许多系统的开发过程实质上就是对新开发系统中的一组概念、部件、元素及其相互之间关系予以清晰、精确及可机器处理的描述、说明及定义。软件系统的开发就是这种情况的一个很好例证。事实已证明，一个优质的软件产品的关键是如何做到这一点。不清晰的要求和说明将导致不清楚的，乃至错误的编码。其它情况，如对建筑物的设计说明、对军队规则、指令的解释等，也都要求对其概念集有尽可能的清晰的描述和理解。

知识表达的原始基础是一种不精确的方法。例如人们用自然语言来表达和组织他们的思想，这些思想又转换成非形式化的说明、文本等。然后，又被重新表达成较形式化的语言，如计算机代码等。在这个转换过程中就产生了很大的差异，甚至可能丢失乃至扭曲了原来的知识。为此，我们提出了一种命为“面向概念的设计、描述语言”(COD)调和在语言和概念上的问题。COD 是结合了人工智能、面向对象的程序设计、逻辑程序设计和自然语言处理等技术的一种知识获取语言。本文首先介绍COD 的思想，然后着重于COD 的实现环境CODE 系统。一般的知识类型都有其自己结构的、语言的和逻辑的特征。但我们也可看到在知识表达、分析和获取中存在着一些共性问题，这就导致了开发一个通用的系统，如CODE, 来解决这些问题。例如在软件工程、技术文本说明书、物理结构的设计说明、政府法规文本等状况中都有着表达、获取知识的共同问题。本文提出的CODE 并不针对上述任一状况的特殊要求，而是一个通用目的知识获取系统。

本文的 §2 简单介绍系统的概念基础，即COD 的基本思想。 §3 是CODE 的组成。§4 讨论一些有关的知识获取领域内的系统。最后讨论CODE 的进一步开发。

§2. 概念和特性

COD 的两个基本标记是“概念” (concept) 和“特性” (properties)。所谓“概念”就是对某一对象的概括的标记，它已无法再用某些更概括的标记来描述它。概念可以是我们欲讨论的任何对象的标记，也可以用某些自然语言或形式语言表达或定义。大多数概念可以用名词词组或动词词组来表达。COD 中概念有“类目” (class) 和“例证” (instance) 之分。“类目”描述一个完整的类别或是其所有可能的例证的聚集。例如“4”是类目“整数”的一个例证。“杭州”是类目“城市”的例证。在COD 中一般考虑“类目”概念，但有时考虑到某些需要也引入一些“例证”来检查和证实某些概念。

CODE 中把知识组织成一个基元，命名为概念描述器(Conceptual descriptor)。概念描述器是人工智能中框架和面向对象的程序设计中目标(object) 的模拟。一个cd 就是一个和某个对象有关的知识包。每个概念都有和它相关的cd 说明。在数学中这种说明可能

是完全的。Cds 根据其互相之间的关系被排列成层次式的多向继承网络。所以下层的概念可以继承比它更概括的上层概念的特性。一个特性就是一个信息元。它可以在一个cd上进行加入、删除或修改，也可传递到它的子概念或例证中去。

2.1 特性

特性分为系统特性和用户特性。本文只讨论一些主要特性，但系统内含丰富的特性类型。用户也可以根据自己的要求创立新的特性。

a. 系统特性

系统特性是每个概念所固有的、基本的特性。它控制了系统的基本操作，如命名cd、特性继承等。系统特性如下所示:

cdName: 概念名字

supers: 一个或几个上层概念

Kinds: 不相交的子概念集的划分

subConcept: 子概念表

instanceOf: 如cd 是例证时，它的“类目”概念

instances: 概念的例证集

b. 用户特性

最常用的一类用户特性是“属性” (attribute)，它相当于框架中的“槽” (slot) 或oop中的例证变量(instance variable)。属性是一个函数关系。属性是概念的必备特性。如概念“person”有如下属性:

(ATTRIBUTES
age: an itneger
sex: one of male, female
mother: a woman)

“person”是“人”这一类目，故“人”就有这三个属性，其值均在指定范围之内。ATTRIBUTE 是属性类特性的标题，下列的每一行就是一个特性。“: ”前为特性名，后面为特性体。概念“人”的例证可以有“人”的同样属性，但每个特性都赋予特定的值。例如，“Li Ming”是“person”的一个例证有如下属性:

(ATTRIBUTES
age: 30
sex: male
mother: Lu hung)

CODE 系统可以检查属性的任何组成是否满足概念所指定的约束。

用户特性的第二个类别是“约束” (constraints)。CODE 系统有一个相应的逻辑演绎系统来完成一些逻辑检测。如检查约束的一致性等。一个约束特性就是一个任意的逻辑表达式，它可以由用户自己建立，也可以由CODE 系统从一个cd 的其它表达式中自动生成。约束一般用来说明该概念的属性条件。例如一个人的年龄必定在0~120 之间，即有如下约束说明:

(CONSTRAINTS
agelimit: 0<age and age <120)

“agelimit”是特性名。“age”是该概念的一个属性。约束体是一个逻辑表达式。

第三个特性类别是“定义” (definitions)。它是说明一个概念的必要、充分条件。当然并非每个概念都具备有这一特性。例如“father”有如下“定义”特性:

(DEFINITIONS male person and parent)

这种表达不仅清晰地表达了一个概念的基本特性，而且也可使逻辑系统测出其可能潜在的矛盾性。

最后介绍一类特性“操作”(operations)。它又可分为“作用于”(operation-on) 和“被作用”(operation-by) 二种。即该概念对另一概念的行为或被另一概念行为所作用。例如

(OPERATIONS ON

marry: x marries #

divorse: x divorces #)

一般特性名是一个非限定性动词，特性体是一个语言模式，其中# 表示事件(event) 可作用的任一概念，x 则是概念本身。

2.2 特性组成:

特性一般有下列几部分组成:

name— — 字符串，一般是名词或动词词组;

flags— — 标记表，说明特性传递模式;

body— — 特性内容，其语法取决于特性类别;

comment— 注释，特性的说明。

另外每一个特性都有一个“source”，说明该特性从何处继承而来。当然特性也可以加上数字作有序排列。

2.3 特性标记

每个特性都有一组标记，它定义了特性传递中的继承行为。需指出的是，我们笼统地说继承是有二义性的。“继承一个特性”是继承特性名，还是连特性体也继承呢? CODE 对这些继承行为均予以不同的处理。flags 分成几个相关组。每个特性必有一个基本的flag。当然也可以有几个相容的flags。下列为几组基本的flags。

Basic flags	(互斥的，每个特性只能有其中之一)
privage(p)	该特性可传递到子概念上，但不能继承给例证，如一个类目的例证数目;
class(c)	可以继承给子概念和例证，如“person”中的age。
Body flags	(控制特性体的继承行为)
has(h)	特性只继承名字和一个空的特性体。然而继承这一特性的概念再给特性体定义。所以has 只说明这个特性存在。如“person”概念中的name;
modifiable(m)	特性体被继承，但每一子概念可作任何方式的修改，例如概念“person”中的age, 对不同子概念有不同的约束。
fixed(f)	继承过程中不能为子概念所修改，如概念“living thing”中的约束“is alive”。
Logic flags	
neccessary(n)	必要性特性，如概念person 中father 特性;
sufficient(s)	充分性特性，如概念“adult”是概念“person”的充分性特性是约束“age>17”。

flags 是用( ) 括起来的一组标记，( ) 中每一字母代表一种标记。它们是无序的。用户根据自己的需要建立特定的flags。

§3. 系统组成

图1 是整个系统的框图。用户界面(user interface) 由三部分组成，即“概念编辑图”(cdriew)、“继承网络图” (node grapher) 和“特性览目表” (properties browser)。用于语义检查的各子系统都可从用户界面上调用。

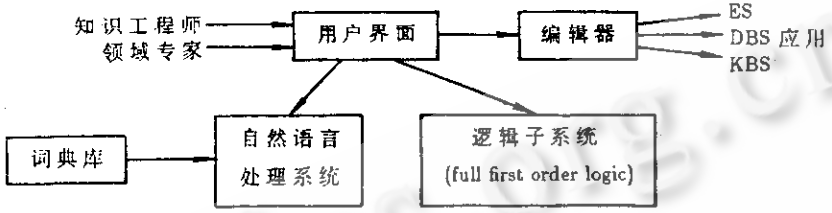

图1 CODE 系统总框图

3.1 用户界面组成

a. 概念编辑图

概念编辑图是展示一个概念描述器的最基本界面，用户用“鼠标”选择各种操作。CODE 提示用户给出概念名及super 名，则super 概念的全部特性根据flags 为新概念所继承。用户可在此基础上做各种特定的修改。每个概念编辑图分顶层和底层二部分。顶层又分为几条，最左边的为系统特性表，是每个概念所必备的。其它为用户自行定义的付特性表，主要用于用户的概念注释、建立日期、作者等等。下层为主特性表。概念除继承其上层概念的各用户特性类外，也可以增加、删减特性类。每一类特性如属性、约束等都有各自的子表，其所属的特性排列在标题之下。

b. 继承网络图

至今，大多数的知识获取系统都结合了图形窗口来辅助显示继承系统网络图。这是最基本的。CODE 系统中用户可以打开一个或几个网络图。如用一个来显示整个网络，而另一个显示用户感兴趣的某一局部。另外图形可以做纵向、横向显示切换。网络图反映了概念库中全部概念及其相互之间的关系。

c. 特性览目表

熟悉Smalltalk 的读者都了解browser 的作用。Browser 以十分方便的方法来编辑系统的各部分构成。特别是快速的“鼠标”选择及相关信息定位。为了增加和修改一个cd 的某个特性，用户可打开特性览目表。

3.2 子系统

a. 自然语言理解子系统

该子系统是一个简单的类英语系统。它可允许用户在概念描述器中使用实际的名词词组、动词词组或句子。在知识获取系统中，该子系统的无二义性语法及语义解释使 CODE 可以进行语法检查，可以用二种方法来使用它。

1) 如使用有限文法那样简单，用户可以输入一个类英语的句子，系统做语法和词汇的检查。这种方法限制用户只能使用简单的文法和有限的词汇。

2) 用一个相关的语法自动地映射到逻辑系统或概念描述器中。例如下列词组:

5 large person in a car

可以映射到一个cd 中来表达。假设“collection of person”概念已建立如下:

cdName: collection of persons
number: an integer
size: a size term
location: a place

则上述的词组被映射成一个该概念“collection of persons”的例证。

CdName: collections of persons
number: 5
size: large
location: carl

b. 逻辑子系统

逻辑子系统是一阶逻辑演绎系统，可用于检测(debug) 规则和事实。这些规则和事实来自用户界面上所建立的cds。生成方法有二种:

1) 显式的，由用户直接输入的wffs, 如同规则基专家系统的规则一样被输入。

2) 隐式的，不是由用户直接输入，而是因概念特性之间的各种隐含关系而潜在的逻辑表达式。例如，用户已输入“person”、“student”、“under graduate student”和“graduate student”概念。当我们检查“student”的逻辑特性时，下列的wffs 将自动地生成，并传递到逻辑子系统中去

person(x) if student(x);
under graduate Student(x) iff student(x) and ~graduate Student(x);
graduate Student(x)
iff student (x) and ~undergraduate Student,

这些表达式说明student 是person, 且undergraduate Student 和graduate student 是student 的一个不相交的划分。逻辑子系统获取表达式后，就可以搜索在特性中可能潜在的冗余、矛盾和不一致性的条件。

逻辑子系统有编译、冗余消除、矛盾检测、追踪及修改数据库等功能。但目前，系统尚无机制可自动地把知识中的逻辑条件修改后反馈到cd 中去。

§4. 有关的系统分析

4.1 知识获取系统

近年来知识获取系统的研究越来越活跃。在北美和欧州有6 个研究组专门从事此领域研究。J. Boose 等的一篇综述反映了当今的动向。Boose 根据已建立的系统及工具，做以下划分: I. 特定领域的任务— — 方法; II. 通用的任务— — 方法; III. 描述任务— — 方法的语言; IV. 通用的大型“world”知识库。

由Bolt、Beranek 和New man 开发的KREME 系统是在Lisp 机实现的。它同CODE 一样是通用性的，但它对知识工程师有较大的限制，另外系统缺乏自然语言和逻辑能力。其特性标记是传统的框架式的。它可以用图形来显示其网络、反映其继承性。

ONTOS 是由CMU 开发研究的知识获取系统，在功能上类同于KREME 系统。主要区别于该系统有较为灵活的框架表达语言，以致有众多的继承程式(scheme)。但系统要求用户有较高的框架语言和Lisp 知识。系统也有部分自然语言处理能力。

KEATS 系统由Open University 开发的，在Lisp 机上实现。它是直接的和启发式系统的结合。系统可辅助处理和专家面谈而得到的自然语言“抄本”，并协助知识工程师把这抄本知识抽样而构成一个专家系统所需的知识。

CYC 系统由MCC 开发。它是通用目的的知识获取系统，它企图获取真实世界范围内的知识。

SB-ONE 是集聚了专家系统和自然语言处理的大型理解系统。它采纳了KL-ONE 知识表达语言。

4.2 软件工程应用

人工智能和软件工程已建立了牢固的联系，而软件工程中一个主要领域——说明，CODE 系统将会找到其广泛的应用。

§5. 讨论与展望

CODE 系统是一个通用的知识获取系统，它有如下优点：有宽广的应用范围；小型化，可为微型计算机采用；有自然语言处理能力；有一阶逻辑分析能力；有灵活的特性继承行为，可多向继承。

CODE 可以在许多方面予以扩展。如语言处理和逻辑子系统的运行环境问题(目前Smalltalk 是用unix 中的sockets 来和prolog 通讯的)，以及在用户界面的进一步智能化问题。另外可望一个用户可以同时存取多个知识源。另外用CODE 来开发一个有广泛应用价值的ontology 来表达世界范围内更上层的常识也是十分有意义的工作。而让CODE 具有学习功能，从cds 中产生cd 的开发更是令人神往的。

本系统是作者在加拿大渥太华大学在导师D.Skuce 指导下完成，并得到该实验室其他同事Yves Beauville 的协助，在此表示感谢。

参考文献

[1] G.Abrelt and M.Burstein, “The KREME Knowledge Editing Enviroment” Int, J.Man—Machine Sendies, Vol. 27, 1987, pp 103–126.

[2] J.Boose and B.Gaines, Proc.3rd Knowledge Acquisition Workshop, Banff, Alberta, Nov. 1988.

[3] R.Brachman and J.Schmolse, “An Overview of the KL—ONE Knowledge Representation System” Cog.Sci.Vol. 9, No. 2, April, 1985, pp 171—216.

[4] L.Ford, “Artificial Intelligence And Software Enginerring: A Tutorial Introduction to Their Relationship”, AI Review, Vol. 1, 1987, pp 255—273.

[5] D.Lenat, M.Prakash and M.Shepered, “CYC: Using Commonsense Knowledge to Overcome Brittleness and Knowledge Aequisition Bottlenecks,” The AI Magazine, Vol. 6, No. 4, 1986, pp 65–85.

[6] S.Nirenburg, I.Monarch, T.Kaufman, I. Nirenberg and J.Caranell, “Acquisition of very Large Knowledge Bases: Methodolgy, Tools, and Application,” CMU Tech. Report CMO–CMT–88–108, 1988.

[7] D.Skuce, “The LESK Tutorial”, Dept, of Computer Science, University, of Ottawa. Dept of CS Teh. Report, No, TR–83–03, 1983.