计算机科学 1996 Vol. 23 No. 3

用面向对象方法建立框架库的研究*)

王万森
(河南师范大学　河南新乡453002)

摘　要　The foundation of object-oriented knowledge base is an important technology in the development of object-oriented expert system. This paper discusses the problem on the object-oriented design and the realization of the knowledge base with the frame representation of knowledge.

关键词　Object-oriented, Frame base, Frame representation, Knowledge base, Expert system.

随着面向对象技术及其应用的不断发展，越来越多的人工智能工作者开始使用这种技术来建造面向对象的专家系统。在这种专家系统中，当采用框架表示法时，知识库实际上是一个由一系列框架所构成的框架库。如何用面向对象技术建立这种框架库，是面向对象专家系统开发的一个重要问题。这个问题的研究可分为两个方面，一是框架表示法的面向对象描述，二是框架库的面向对象设计和实现。

1　框架表示法及其面向对象描述

框架表示法是一种比较接近于人类思维形式的知识表示方法，常被用来表示那些比较复杂且具有一定结构性的领域知识。采用这种方法，知识表示的基本单位是框架，领域知识是通过框架的层次结构来描述的。框架的可嵌套性是构造框架层次结构的基础，同时也体现了框架之间的继承关系。

当然，框架仅仅是一种抽象结构，是对概念或事物的抽象描述。要用框架表示某一个具体事物或概念，还需要对框架进行实例化，这种实例化的框架常被称为实体(entity)。框架与其实体之间的关系是一种一对多的关系，即一个框架可以对应多个实体。

面向对象既是一种程序设计方法学，同时也是一种认知方法学。从程序设计角度看，面向对象的核心概念包括数据抽象、信息隐蔽、继承和消息传递等。从认知角度看，面向对象既提供了从一般到特殊的演绎(如继承)手段，又提供了从特殊到一般的归纳(如类)方法。

当用面向对象方法实现框架知识表示时，框架用类来描述，实体用类的对象来描述，框架的槽用相应类的属性来描述。但是，由于框架的一个槽可含有槽名、槽值、槽缺省值、槽置信度等，故类的相应属性应该是某种结构类型，而不能是简单数据类型。特别情况下，当框架的一个槽为另外一个子框架时，其属性则是一个派生类。

在上述描述方式下，框架方面的四个基本要素：框架、实体、槽及继承，将分别对应着面向对象方面的四个基本要素：类、对象、属性及继承。此外，框架与其子框架之间的关系，可通过类的派生来实现；一个框架的诸槽之间的联系，以及一个框架与其实体之间的联系，都可通过建立相应的链来实现。

2　框架库的面向对象设计与实现

2.1　知识库的设计途径和步骤

面向对象作为一种程序设计方法，并非一定需要面向对象语言的支持。面向对象框架库的设计可以采用面向对象语言，也可以采用非面向对象语言。究竟选择哪种途径，应根据具体问题和实际情况而定。为简化讨论，本文采用了 C++ 面向对象程序设计语言。

C++ 具有支持面向对象设计的语言设施，但它并不支持运行期间类的动态定义和类层次之间的柔性连接。C++ 的这种限制，使得用户必须在设计知识库之前就知道问题的完整结构。显然，这是一种比较苛刻的要求，不符合人们对事物的认识过程。幸好，C++ 支持原型方法，这就为解决上述问题提供了一个很大的缓冲。尽管如此，面向对象设计还应该从问题的基本结构开始。

利用 C++ 语言建立框架库，一般需要经过以下

*)河南省科委资助。王万森　副教授，人工智能研究所所长，研究方向：人工智能，专家系统，图像理解。

四步：第一步，确定问题的类层次结构；第二步，对类层次结构中的每个类，定义其框架和槽的数据结构，并且对最低层的每一个类，还需要定义其框架的实体和实体链；第三步，根据类层次结构中各个类之间的联系进行类定义；第四步，对框架中的有关槽，定义操作其槽值的成员函数。

2.2 问题的类层次结构的确定

问题的类层次结构是根据问题的基本结构来确定的。由于问题基本结构中的非叶结点都对应着知识表示中的一个框架或子框架，因此在把问题的基本结构转化为其类层次结构时，需要用类或子类来描述这些框架或子框架。为了既便于讨论又不失一般性，图1给出了一个简化后的肝脏疾病诊断问题的基本结构，它所对应的类层次结构如图2所示。

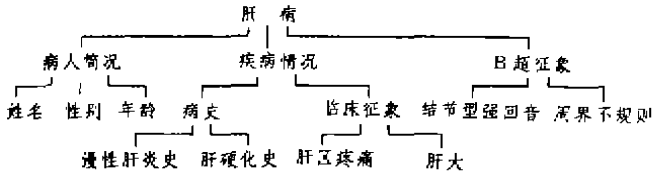


图1 一个问题基本结构的例子

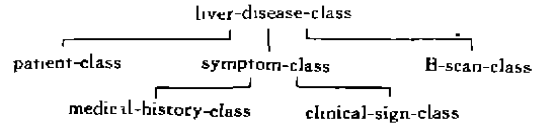


图2 图1所对应的类层次结构

2.3 框架和实体的数据结构的定义

在类层次结构中，除最低层类的框架一定有实体对应外，其余类的框架则不一定有对应的实体。因此，对框架的数据结构的定义应分开讨论。

2.3.1 无对应实体的框架的数据结构。这种框架的数据结构是一个由该框架的所有槽结点构成的一个链，简称槽链。其链头结构包括该框架的名字和一个指向槽链的指针。即：

```
struct frame{
  char *frame-name;
  slot-node *slot-list;
}
```

槽链中每个槽结点的结构由该槽的槽名和指向下一个槽结点的指针所组成。即：

```
struct slot-node{
  char *slot-name;
  slot-node *slot-next;
}
```

2.3.2 有对应实体的框架的数据结构。这种框架的数据结构除包含上述槽链外，还应包含一个由该框架的诸实体所构成的一个实体链。它的链头结构包括该框架的名字、指向槽链的指针和指向实体链的指针。以图2的 patient 类为例，其链头定义为：

```
struct patient-frame{
  char *frame-name;
  slot-node *slot-list;
  patient-entity *entity-list;
}
```

2.3.3 实体的数据结构。实体的数据结构是指实体链中每个实体结点的结构，它包括该实体的名字、指向该实体所对应框架的指针和指向下一个实体结点的指针。仍以图2中的 patient 类为例，其实体结点的定义为：

```
struct patient-entity{
  char *entity-name;
  patient *frame-of-entity;
  patient-entity *next-entity;
}
```

2.4 类层次结构中类的定义

在对类层次结构进行类定义时，类中的一个属性描述框架中的一个槽。如果框架的某个槽是另外一个子框架，则应将此子框架的类定义为其父框架的类的友类，并按公有方式派生。以 patient 类为例，它是 liver-disease 类的友类，并从该类公有派生。patient 类的定义为：

```
clase patient:public liver-disease{
    char-slot *name;
    char-slot *sex;
    int-slot *age;
  pubilc:
    patient();
    ~patient();
    void set-slot-value();
    void show-slot-value();
    patient *get-slot-value();
}
```

其中，set-slot-value()、show-slot-value()和 get-slot-value()分别是建立、显示和得到槽值的三个公有成员函数；name、sex 和 age 分别对应病人简况框架的三个槽。若假设每个槽仅包括槽名和槽值，则槽值为字符型或整型时的两种不同槽结构可分别定义为：

```
struct char-slot{
    char *slot-name;
    char *slot-value;
  }
struct int-slot{
    char *slot-name;
    int *slot-value;
  }
```

2.5 类中成员函数的定义

类的成员函数是对类的属性进行操作的方法，或者说是对框架中的槽值进行操作的函数。例如前面所提到的 set-slot-value()，show-slot-value()和 get-slot-value()等。类的成员函数可根据需要定义，由于此问题较为简单，故本文从略。

2.6 框架和实体的建立

框架和实体的建立过程是在已经定义的框架和实体的数据结构上进行的。

2.6.1 建立框架和槽链。建立一个框架的主要任务是根据该框架所包含的诸槽的槽名，建立各个槽结点和由这些槽结点所形成的槽链。例如，对图1中的病人简况框架，它所包含的诸槽的槽名可用如下字符数组说明：

```
char * patient-slot [ ] = {"name", "sex", "age"};
```

根据此数组，建立一个框架的函数为：

```
frame * set-up-class-slot (char * frame-name, char *slot[]){
  frame * temp=new frame;
  int i=0;
  temp->frame-name=frame-name;
  temp->slot-list=NULL;
  while(strcmp(slot[i],SLOT-END)){
    slot-node * temp1=new slot-node;
    temp1->slot-name=slot[i];
    temp1->slot-next=temp->slot-list;
    temp->slot-list=temp1;
    i++;
  }
  return temp;
}
```

用相应的实参调用此函数，就可建立起一个框架结构，并返回指向该框架的指针。此指针实际上是指向该框架的槽链的指针。

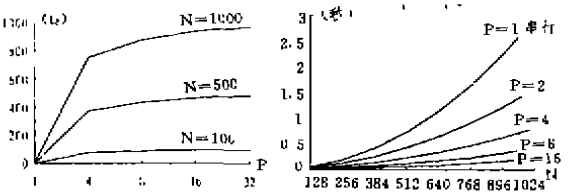
对上述框架，建立其槽链的函数为：

```
add-slot-list(frame * f,patient-frame * p){
  p->frame-name=f->frame-name;
  p->slot-list=f->slot-list;
}
```

2.6.2 建立实体和实体链。建立一个实体的主要任务是对该实体的框架的各个槽赋值，这是由相应类的成员函数来完成的。一实体被建立后，还应将其加入到相应的实体链中。下面是将一实体加入相应实体链的函数：

```
void add-entity-list (char * entity-name, patient * s, patient-frame * sl){
  patient-entity * temp=new * patient-entity;
  temp->entity-name=entity-name;
  temp->frame-of-entity=s;
  temp->next-entity=sl->entity-list;
  sl->entity-list=temp;
};
```

（上接第79页）



图二 通信开销与 P 的关系　　图三 同代过程串行与并行执行的响应时间

从(4)，(5)二式可得加速比：

$$S_p=\frac{T_1}{T_{p0}+T_{com}}$$

$$=\frac{P(1+1/N)}{1+(p\delta-1-\delta)/N+(1-1/P)t_s/N}$$

当 N 足够大，且 P 具有一定规模时 $S_p\approx P/(1+t_s/N)$，t_s 基本上为一常数，单位是与执行一次乘除法时间的比。在我们的实验中，构成虚拟机节点的主机都是 SUN SparcStaion 1，除首次起动时的同步接收等待较长，以后 t_s 的开消是一常数。我们对以下假定的上三角阵进行了求解。

$a_{ij}=1, b_i=(n+i-1)(n-i)/2, i,j=0,1,\cdots,n$ $\delta=32$ 并对不同的 N，P 及串行算法进行了比较，结果见图三。从图三可以看出，随 N 的增大，加速也更明显，这与加速比模型分析是一致的。

参 考 文 献

[1]Al Geist, et al., PVM3 User's Guide And Reference Manual, May, 1993

[2]V. S. Sunderam, PVM: A framework for parallel distributed computing, J. Concurrency: Practice and Experience, 2(4)1990

[3]陈景良，《并行算法引论》