一种基于知识规则推理的C++实现方法

徐　彤，　雷英杰
(空军工程大学 导弹学院，陕西 三原　713800)

摘　要:提出了一种决策支持系统中基于产生式规则的知识表示和推理的C++语言实现方法。该方法把规则的结构和推理定义成规则类,把具体的规则定义成规则类的实体,用这些规则实体构建动态链表从而组成知识库,推理的过程由知识库中各规则实体提供的方法完成。通过设计的实例表明,该方法是简单高效的。

关键词:决策支持系统;知识表示;产生式规则;推理
中图分类号:TP393　　**文献标识码**:A　　**文章编号**:1009-3516(2002)06-0075-04

决策支持系统(DSS)是一门新兴边缘学科,它不但能够有效支持结构化问题的求解,而且通过把人工智能关于知识表示和处理的技术应用到DSS,与专家系统相结合,从而形成了智能决策支持系统(IDSS),可以有效地支持半结构化和非结构化问题的求解。与单纯利用模型库的定量方法相比,IDSS的优越性突出的表现在其建立了知识库,存放有各种知识规则、因果关系和决策人员经过知识化抽象的经验,并且具有综合利用知识库、数据库和定量计算结果进行推理和问题求解的推理机。由此可见,一个组织恰当的知识库对于IDSS是至关重要的。

通常在IDSS知识库中进行知识表示的方法有:一阶谓词逻辑、语义网络、产生式规则、框架理论、面向对象表示等。其中产生式规则由于Post严格地理论证明,具有完备地理论基础;各产生式之间相互较为独立,修改与扩充都较容易,所以应用较广。本文提出了产生式知识规则的C++语言实现方法,并将推理机与知识表示相结合,构造出了一个简单高效的知识库。

1　设计思想

1.1　知识表示

在IDSS中建立基本事实数据库和规则库。基本事实数据库,简称事实库,其结构如图1所示,主要包括事实号和事实内容等。规则库中每个记录包含一条规则,其结构如图2所示。

事实号	事实内容

图1　事实库记录结构

规则号	条件事实号	条件事实号		结论事实号

图2　规则库记录结构

我们定义三个类,且构建这三个类对象的动态链表。

1)定义条件事实类

条件事实对象结构如图3所示。

2)定义规则类

将规则的结构和推理关系定义成规则类,每一条规则都是根据规则库中的一个记录生成的规则类对象,

收稿日期:2001-09-05
基金项目:军队科研基金资助项目
作者简介:徐　彤(1978-),男,山东菏泽人,硕士生,主要从事智能决策支持系统研究;
雷英杰(1956-),男,陕西渭南人,教授,博士生导师,主要从事人工智能、决策支持、网络与信息安全研究.

把这些规则对象构建成动态链表,我们称为规则链。规则对象的结构如图 4 所示。

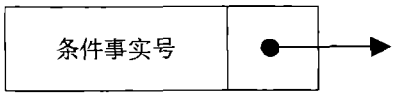

图 3 条件事实对象

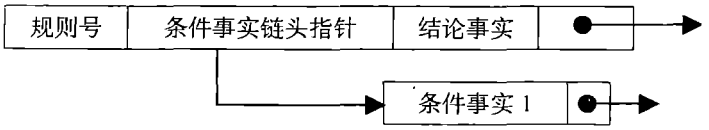

图 4 规则对象的结构

3)定义基本事实类。

基本事实库中的每条记录生成一个基本事实对象,再把这些对象建立成动态链表。结构如图 5 所示。

规则链、每条规则的条件事实链和基本事实链共同构成了知识库,对知识库的操作方法由规则类的操作方法提供。

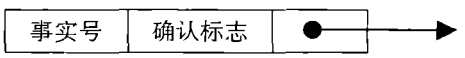

图 5 基本事实对象

现举例说明如何由产生式规则建立数据库和知识库。

假定有如下规则:

规则 1:若是中年人,则老练。

规则 2:若是老练、细心并有驾驶经验,则不会出交通事故。

对这两条规则在数据库中建立的事实库和规则库如图 6 所示,所建立的知识库如图 7 所示。

事 实 库

基本事实号	事实内容
1	中年人
2	老练
4	有驾驶经验
5	不会出交通事故

规 则 库

规则号	条件事实	条件事实	条件事实	结论事实
1	1	0	0	2
2	2	3	4	5

图 6 事实库和规则库结构

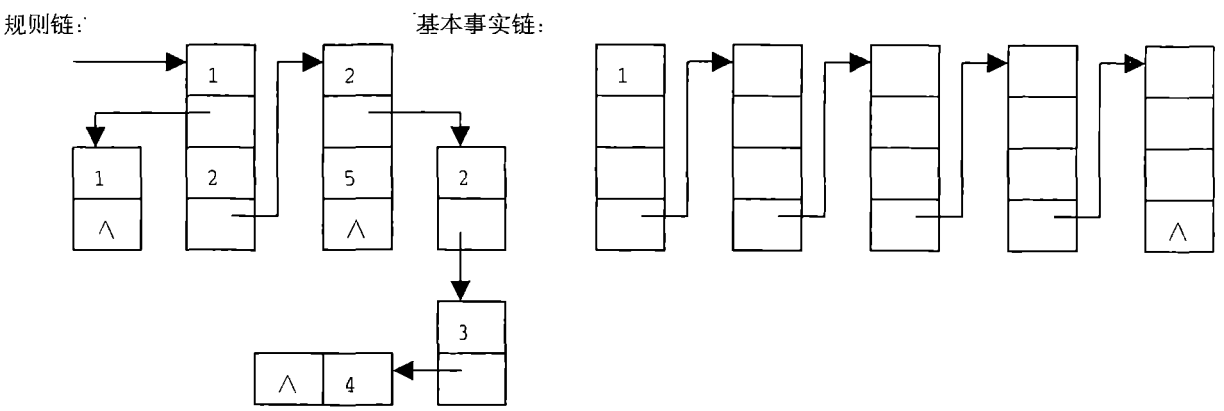

图 7 知识库结构

1.2 推理机制

每次推理,DSS 都依次调用规则链中每个规则对象的 Query 方法(此即推理函数),直至推理结束,得到

结论。在 Query 方法中,DSS 从条件事实链中读取每一个条件事实,并在基本事实链中找相同事实,找到后,考察该事实对象的 TorF 字段。若为 True,表明条件事实链中的事实已经获得用户确认,则读取下一个条件事实;若为 False,表明条件事实链中的事实已被用户否认,则换下一条规则;若为 Unknown,则与用户交互,由用户确认或否认该事实。若用户确认,则置 TorF 为 True;若用户否认,则置 TorF 为 False,并换下一规则。

这样,只有当某规则上的条件事实全部是被用户确认过的,才能得到结论,从而保证了推理的正确性。

2　代码实现

1) 定义基本事实类

```
class Fact     // 基本事实类{   int Number;    int TorF;  // 基本事实号,确认标志
public: Fact *Next;          // 后继链接指针
    Fact (int number) { Number = number; TorF = Unknown; Next = NULL; } // 构造函数
    int GetNumber( ){return Number;}
    int GetTorF( ){return TorF;}
    void SetTorF(int torf){TorF = torf; }
};
```

2) 定义条件事实类

```
class List     // 条件事实类
{   int Number;      // 基本事实号
public: List *Next;      // 后继链接指针
    List (int number){Number = number; Next = NULL;}// 构造函数
    int GetNumber( ){return Number;}
};
```

3) 定义规则类

```
class Rule     // 规则类
{   int Number;List *pCause;int result;// 规则号,该规则的条件事实链头指针,结论号
public: Rule *Next;      // 后继链接指针
    Rule(int number, int a[])     // 构造函数
    {   int i; List *l; pCause = NULL; Next = NULL; Number = number;i = 0;
        while (a[i]! = -1 && i<3){l = new List (a[i++]); l->Next =pCause; pCause = l;} result = a[3]; }
    ~Rule( )     // 析构函数
    {   List *l;
        while (pCause){ l = pCause->Next; delete pCause; pCause = l;}}
    int Query( )     // 规则查询推理函数
    {    char c;List *l; Fact *f = fact;l = pCause;
        for ( ; l! = NULL; )
        {     f = fact; for ( ; ; ){if (l->GetNumber( ) = = f->GetNumber ( )) break; f = f->Next;}
            if (f->GetTorF ( ) = = True) {l = l->Next; continue;}
            if (f->GetTorF ( ) = = False) {return False;}
            if (f->GetTorF( ) = = Unknown)
            {   cout << endl << str[f->GetNumber ( )] << "(Y/N)?"; c = getchar( );
                if ((c = = 'Y') || (c = = 'y')){f->SetTorF (True); l = l->Next;} else
                    if ((c = = 'N') || (c = = 'n')) {f->SetTorF (False); return False;}}}
        if (result<4){ for( ; ; ){ if (result = = (f->GetNumber( ))) break; f = f->Next;}
```

```
        f - >SetTorF(True); return False;} else
            { cout < <"\n" < <str[result]; return True;} }
};
```

3 应用实例

产生式规则:中年人是老练的,中年人是细心的。老练、细心且有驾驶经验是不会出交通事故的。由于规则较少,这里用数组代替数据库实现基本事实库和规则库。其实现代码如下:

```
char *str[ ] = {"中年人","老练","细心","有驾驶经验","不出交通事故","\0"};
int a[ ][4] = {{1,2,3,4},{0, -1, -1,1},{0, -1, -1,2}};
int main( )
{Fact *f, *fact;Rule *rule, *r; int i =0; fact = NULL;
while(str[i]! ="\0"){f = new Fact(i + +); f - >Next =fact; fact = f;} rule = NULL;
for (i =0;i <3;i + +){ r = new Rule(i,a[i]); r - >Next = rule; rule = r;}
for (;;){ i = rule - >Query ( ); if ( i = =True) break; rule = rule - >Next ;
if (rule = =NULL) { cout < <endl < <"可能会出交通事故!"; break;}}
return True;
}
```

4 结论

本文所提供的 IDSS 中产生式规则知识表示、推理的实现方法是与 DSS 数据库相独立的,即用这样的方法产生的知识库是不依赖于数据库的。只要在 DSS 数据库中构造不同的基本事实库和规则库,系统就可以自动生成相应的知识库,因此具有很高的灵活性。

参考文献:

[1] 陈文伟 决策支持系统及开发(第二版)[M]. 北京:清华大学出版社,2000.
[2] Shim J P. Past, Present and future of Decision Support Technology [J]. Decision Support Systems, 2000,33 (3): 111 - 126.
[3] Silvano Mussi. Sequential Decision - theoretic and Expert Systems[J]. Expert systems, 2002,19(2): 99 - 108.
[4] Bruce Eckel. Thinking in C ++ [M]. America: Prentice Hall,1999.
[5] 张水平,郑飞雁. 数据仓库技术研究[J]. 空军工程大学学报(自然科学版),2000,1(3):68 - 71.

(编辑:田新华)

A Rule - based Approach to Implementing Inference Using C ++

XU Tong, LEI Ying - jie
(The Missile Institute, Air Force Engineering University, Sanyuan, Shaanxi 713800, China)

Abstract: This paper presents a new method of implement inference and knowledge representation in IDSS based on the product - rule by using C ++ language. In this method, we define the structure and inference of the rule as a rule class and the specific rule as an object of the rule class to construct the knowledge base. The inference process is accomplished by using the method provided by each object of the rule class in knowledge base. The designed example shows that the new method is simple and effective.

Key Words: decision support system; knowledge representation; product rule; inference