计算机科学2000Vol.27№.8

基于加权产生式规则知识库的不一致性和冗余性研究

Research on Nonconsistence and Redundancy of Knowledge-base Using Product Rules with Cost

陈朝东　黄国兴　鲍　钰
（华东师范大学计算机系　上海200062）

Abstract Knowledge-base maintenance is an important aspect in the construction of expert systems. In this paper, the authors introduce a new concept, i.e., "Product Rules with Cost". Then, the authors discuss the meaning of the nonconsistency and redundancy of knowledge-base using the concept above. To deal with the nonconsistency and redundancy, the authors give out the responding principles and algorithms, which are practical for knowledge-base maintenance in expert systems based on product rules with cost.

Keywords Product rule with cost, Nonconsistency, Redundancy

一、问题的提出

规则的产生式表示法是目前专家系统中最常用的一种方法，它易于表达浅层知识，并且具有模块性、清晰性、自然性等优点[1]。但是，由于产生式系统的知识库常常是不完备的，需要对规则进行增、删、改等维护操作，这往往会引起知识库中知识的不一致性和冗余性，如何处理这一问题，是专家系统维护中需要考虑的一个重要问题。

但是，现实世界中知识的不一致性和冗余性是一个很复杂的问题。以不一致性而言，人类在进行常识推理时，往往是在知识不完全或不一致的背景下进行的[4]，知识之间的不一致性随处可见，比如，一个人可能会对某人或某事有矛盾看法，但这并不影响其合理的推理行为。因此，专家系统中知识库往往允许存在局部的不一致性。再以冗余性而言，在专家系统中，有时候适当的冗余能够加快推理速度，从某种角度来说反而提高了系统的运行效率。

产生式知识库中的知识往往是启发式规则，规则前提中的各个子条件项对规则成立的贡献大小不同，因此，可以考虑对各子条件项进行加权，这就产生了本文提出的"加权产生式规则"的概念。基于加权产生式规则的系统的推理机制发生了变化，使得这类系统的知识库维护比相对于一般产生式知识库的维护要难。本文就是研究在这种基于加权产生式规则的知识库中，如何合理地处理知识的不一致性和冗余性。

二、加权产生式规则

1. 产生式规则的一般模式

产生式规则的基本形式是：IF P THEN Q，其中，P 代表一组前提或状态，Q 代表若干结论或动作，其含义是：若前提或状态 P 得以满足，则可得出结论 Q 或完成 Q 所规定的动作。这是一种理想情况下的知识表示形式，但在现实世界中，专家系统常常面对结构不良的复杂问题，这类问题的信息往往具有不确定性、模糊性、不完备性，甚至矛盾性；同时，人类专家在求解这类问题时所用的知识（尤其是启发式知识），也常常是不精确或不完备的。因此，在专家系统设计中，常常引入"可信度因子"（Certainty Fator，CF），并对规则形式进行改造[5]，以此反映出知识不精确性对求解结果的影响。改造后的规则形式为：

IF P THEN Q with CF＝m

其中，P、Q 的含义不变，CF 为规则的可信度因子，也称为规则强度，它表示前提 P 对结论 Q 的支持程度，m 为规则强度的值，一般来说，$0<m\leqslant 1$。

考察规则的前提和结论，我们得知，一条产生式规则 R 可以不失一般性地用公式表示为：

$$R:\bigcap_{i=1}^{n}E_i\rightarrow C(CF)$$

其中，"∩"为"合取"符号，E_i 称为规则 R 的前提的一个"合取子条件项"，简称为"子条件项"，C 称为规则 R 的结论，E_i 和 C 都不能再拆分为更小的表达式，CF 即

规则强度。这里之所以不必考虑"析取"，是因为，若一条规则的前提中含有析取项，则可以通过一定的方法将该规则拆分为两条或多条规则，最终还是转化为上述的公式形式。本文讨论的规则都具有这样的特点。

2. **加权产生式规则**[2]

经过改造后的规则反映了推理的不精确性，但规则的语义基本不变，各子条件项之间仍无轻重之分，地位完全相等，这与现实生活中的经验不甚吻合。例如，医学领域中诊断某种疾病时，主症的贡献就要比伴随症状的贡献大。因此，在设计专家系统的知识表示模式时，可以考虑对规则前提中的子条件项进行加权，以区别规则的各个子条件项，权是一个一般化的概念，在实际中可以是重要性、可能性、信息量等。对于一条规则，其子条件项的"权"定义为该子条件项对规则成立的贡献程度，权值由领域专家确定。这样，一条产生式规则 R_k 就可以用公式表示为：

$$R_k:\bigcap_{i=1}^{n_k}W_iE_i\rightarrow C_k(CF)(T)(k=1,2,\cdots,m)$$

其中，m 为知识库中规则的条数，E_i 代表规则 R_k 前提中的子条件项，W_i 称为子条件项 E_i 的"权"，C_k 代表规则 R_k 的结论，CF_k 代表规则 R_k 的规则强度，T_k 称为"规则阈值"，用于确定规则 R_k 是否可用。引入加权产生式规则后，系统控制方式发生了变化，即当遇到一个子条件项为假时，不能立即认定整个前提为假而放弃该规则，而是继续处理下一子条件，等全部子条件处理完毕，再计算出条件的总体可信度 T(R)，T(R)的计算公式如下。

$$T(R)=CF\times\sum_{i=1}^{n}(W_i\times CF_i)$$

其中 CF_i 为子条件项 E_i 的可信度。最后，系统将 T(R)与"规则阈值"T 相比较，以确定一条规则是否可用。确定的原则是：若 $T(R)\geqslant T$，则规则 R 可用，否则规则 R 不可用。

三、加权产生式规则库的不一致性和冗余性含义

对于一般的产生式知识库而言，知识的不一致性和冗余性表现在：循环规则链、矛盾规则链、等价规则、从属规则以及传递冗余等方面[3]。与之相对应，在基于加权产生式规则的知识库中，知识的不一致性也表现在上述几个方面，只不过含义有所不同。

设 R_k 代表一条规则，$R_k.E[n_k]$代表规则 R_k 前提子条件项集合，n_k 为子条件项个数。C 代表规则 R_k 的结论，又设一组规则 $R_1,R_2,\cdots,R_n(n)(n>1)$代表一条规则链，则可以给出如下几个定义。

定义1(循环规则链) 设有一条规则链 $R_1,R_2,\cdots,R_n(n>1)$，若它满足：(1)R_1 与 R_n 等价(两条规则等价的定义见以下定义3)，记为 $R_1\equiv R_n$(下同)；(2)对于$\forall R_K(1\leqslant k\leqslant n-1)$，$R_k.C\subseteq R_{k+1}.E[n_{k+1}]$，则称该规则链为循环的规则链。如，$A\rightarrow B,B\wedge C\rightarrow D,D\wedge E\rightarrow A$ 就是一条循环规则链。

定义2(矛盾规则链) 设有两条规则 R_1 和 R_2，若(1)$R_1.E[n_1]=R_2.E[n_2]$；(2)$R_1.C=\neg R_2.C$，其中"¬"代表"逻辑非"，则称规则 R_1 与 R_2 矛盾。又设有两条规则链 $R_{11},R_{12},\cdots,R_{1s}(s>1)$和 $R_{21},R_{22},\cdots,R_{2t}(t>1)$，若满足：(1)$R_{11}.E\equiv R_{21}.E$；(2)$R_{1s}.C=\neg R_{2t}.C$；(3)对于$\forall R_k,(1\leqslant k\leqslant s-1),R_{1k}.C\leqslant R_{1k+1}.E[n_{1k+1}]$；(4)对于$\forall R_k,(1\leqslant k\leqslant t-1),R_{2k}.C\subseteq R_{2k+1}.E[n_{2k+1}]$，则称这两条规则链为矛盾规则链。如规则链 $A\rightarrow B,B\wedge C\rightarrow D,D\wedge E\rightarrow F$ 和 $A\rightarrow G,G\wedge H\rightarrow I,I\wedge J\rightarrow F$ 是矛盾规则链。

定义3(等价规则) 对于规则 R_1 和 R_2，若：(1)$R_1.E[n_1]=R_2.E[n_2]$；(2)$R_1.C=R_2.C$，则称规则 R_1 与 R_2 等价，记为 $R_1\equiv R_2$。如 $A\wedge B\rightarrow C$ 与 $B\wedge A\rightarrow C$ 等价。

定义4(从属规则) 对于规则 R_1 和 R_2，若：(1)$R_1.E[n_1]\subseteq R_2.E[n_2]$；(2)$R_1.C=R_2.C$，则称规则 R_2 从属于规则 R_1。如规则 $A\wedge B\rightarrow C$ 从属于规则 $A\rightarrow C$。

定义5(传递冗余) 对于一条规则 $R_m(m\geqslant 1)$和一条规则链 $R_1,R_2,\cdots,R_n(n>1)$若：(1)$R_1.E[n_1]\subseteq R_m.E[n_m]$；(2)$R_n.C=R_m.C$，则称规则 R_m 对于规则链 $R_1,R_2,\cdots,R_n(n>1)$，是"传递冗余"的。如规则 $A\rightarrow D$ 对规则链 $A\rightarrow B,B\rightarrow C,C\rightarrow D$ 是传递冗余的。

定义1和2描述了知识的不一致性，其中循环规则链的存在可能导致系统推理陷入死循环，而矛盾规则(链)的存在则可能导致系统推出互相矛盾的结论；定义3、4和5描述了知识的冗余性，即某条规则对于推理而言是多余的，它们的存在占用了系统的知识库空间，影响了系统推理的效率。对于一般的产生式系统知识库，我们利用图论中的 Warshell 等算法，对各类不一致性和冗余性进行了深入的分析和研究，取得了令人满意的成果[2]。在基于加权产生式规则的知识库中，知识不一致性和冗余性的检测与一般产生式知识库类似，但对不一致性和冗余性的处理却有很大的不同。考虑到规则的加权特性，具体处理算法也将比一般产生式规则知识库复杂得多。

四、基于加权产生式规则知识库的不一致性和冗余性处理

根据加权产生式规则的公式表示法，定义其存储数据结构如下。

```
Structure R{
    Integer n;//n 为规则 R 的前提子条件项数目
```

```
Char E[n]; //E 为规则 R 的前提子条件项集合
Float CFE[n]; //CFE 为规则 R 前提子条件项的
    可信度集合，其每个元素初值均为1
Float WE[n]; //WE 为规则 R 前提子条件项的权
    值集合，每个元素的值均在[0,1]内
Char C; //规则 R 的结论
Float T; //规则 R 的规则阈值
Float CF; //规则 R 的规则强度
```

1. 循环规则链的处理

假设在基于加权产生式规则的知识库中，利用文[2]中的算法，可以检测出一条循环规则链 $R_1, R_2, \cdots, R_s, R_1 (s>1)$，它会引起潜在的推理死循环。对该循环规则链处理算法如下：

```
Function DealwithLoopLink(R)
//算法的输入 R 代表了上述循环规则链 R1,R2,…,Rs,R1,它
是一个数组，其每个元素都是一条规则，采用前述的数据结构
来存储。
Begin
    V←True; //布尔变量 V 用于判别该规则链是否需要提
    交
    for k=1 to s-1 do
    begin
    U←0;
    for j=1 to R[k].n do
//计算 ∑_{i=1}^{n_k} (CFi * Wi)
     U←U+ R[K].CFE[j]×WE[j];
   if U× R[k].CF≥R[k].T then//规则 R[k]可用
   begin
    M←Find(R,k); //查找规则 R[k]结论在规则 R[k+1]
    前提子条件项集合中的位置
    R[k+1].CF[M]←U× R[k].CF; //沿着循环规则链进
    行可信度传播
   end else
   begin
   V←False;
   Break; //跳出循环语句
  end
 end;
 if V=True then return R; //返回该循环规则链，提交各专
 家进行决策
End;
以下是函数 Find 的代码。
Function Find(R,k)
//查找并返回规则 R[k]结论在规则 R[k+1]前提子条件项
集合中的位置
  Begin
  for i=1 to R[k+1].n do
  if R[k+1] E[I]=R[k].C then return i;
End;
```

2. 矛盾规则链的处理

与循环规则链的处理类似，假设已经检测出两条相互矛盾的规则链，即 $R_{11}, R_{12}, \cdots, R_{1s} (s>1)$ 和 $R_{21}, R_{22}, \cdots, R_{2t} (t>1)$。对这两条规则链处理的具体算法如下：

```
Function DealWithConflictLink(R1,R2)
//R1 和 R2 代表上述两条规则链，它们都是数组，且每个元素
都是一条规则。
Begin
    V1←True; //V1 指示规则链 R1 是否可以推到终点。
    for k=1 to s-1 do
    begin
  U1←0;
  for j=1 to R1[k].n do //计算 ..
   U1←U1+R1[k].CFE[j]×WE[j];
   if U1×R1[K].CF≥R1[K].T then//规则 R1[k]可用
   begin
    M←Find(R1,k); //查找规则 R1[k]结论在规则 R1[k+1]
前提子条件项集中的位置
    R1[k+1].CF[M]←U1×R1[k].CF; //沿着规则链 R1进
行可信度传播
   end else
   begin
    V1←False;
    Break; //跳出循环语句
   end
 end;
 V2←True; //V2指示规则链 R2是否可以推到终点。
 for k=1 to t-1 do
 begin
  U2←0;
  for j=1 to R2[k].n do//计算 …
   U2←U2+R2[k].CFE[j]×WE[j];
  if U2×R2[k].CF≥R2[k].T then//规则 R2[k]可用
  begin
   M←Find(R2,k); //查找规则 R2[k]结论在规则 R2[k+1]
前提子条件项集中的位置
   R2[k+1].CF[M]←U2×R2[k].CF; //沿着规则链 R2进
行可信度传播
 end else
 begin
  V2←False;
  Break; //跳出循环语句
 end
end;
if V1=True and V2=True then//两条规则链均可推到终点
 if U1≥U2 then return R2; //应提交规则链 R2
    else return R1//应提交规则链 R1
End;
```

3. 等价规则的处理

假设有两条互相等价的规则 R_1 和 R_2，那么处理这两条规则的原则较简单，只需保留那条结论可信度值较大的规则，而舍弃结论可信度较小的那条规则，处理算法如下：

```
Function DealWithEqual(R[1],R[2])
Begin
 if R[1].CF≥R[2].CF then
  R[2].Remove(); //从知识库中删除规则 R[2]
 else
 R[1].Remove(); //从知识库中删除规则 R[1]
End;
```

4. 从属规则的处理

假设有规则 R_1 从属于规则 R_2，对这两条规则的处理原则是：删除 R_1，因为 R_1 对于推理而言是冗余的；调整 R_2 的前提子条件项的次序，这是因为，在实际系统推理机的实现过程中，有时只需计算出某规则前提中前几个子条件项的加权和，就可以得知该规则可用，而不必将结论的总体可信度 T(R)求出后再与规则阈值 T 比较。因此，在前提子条件项的可信度未知的情况下，一般可以将权值较大的子条件项排在前面，以加快系统的推理速度。具体的处理算法如下。

```
Function DealWithPertain(R[1],R[2])
Begin
 M←Sort(R[2]); //将规则 R[2]前提子条件项按权值从大
到小排序，结果放入数组 M
 R[0]←Copy(R[2]); //将规则 R[2]复制一份
 for i=1 to R[2].n do
  R[2].E[I]←R[0].E[M[i]];
 R[1].Romove(); //从知识库中删除规则 R[1]
End;
```

（下转第62页）

从而 N 也是可重复的。同样可以证明：

定理10 设 N_1，N_2是两个相容的 $Petri$ 网，若 $P_1 \cap P_2=\varnothing$，则 $N=N_1 \underline{\cup} N_2$也是相容的 Petri 网。

定理11 设 N_1，N_2是两个结构有界的 Petri 网，若 $P_1 \cap P_2=\varnothing$，则 $N=N_1 \underline{\cup} N_2$是结构有界的。

证明：若 N_1，N_2是两个结构有界的 Petri 网，A_1，A_2分别是 N_1，N_2的关联矩阵，则存在 $m(m=|P_1|+|P_2|)$维正整数向量 $Y_i=(y_{i1}y_{i2}\cdots y_{im})^T$（$P_i$中库所在 Y_i 中对应分量为0，P_i中库所在 Y_2中对应分量为0）使 $A_iY_i \leqslant 0$，$i=1,2$，令 $Y=Y_1+Y_2$，则：

$$(A_1 \otimes V_{n_2}^T)Y = A_1Y \otimes V_{n_2}^T = (A_1Y_1+A_1Y_2) \otimes V_{n_2}^T$$
$$= A_1Y_1 \otimes V_{n_2}^T \leqslant 0$$
$$(V_{n_1}^T \otimes A_2)Y = V_{n_1}^T \otimes A_2Y = V_{n_1}^T \otimes (A_2Y_1+A_2Y_2)$$
$$= V_{n_1}^T \otimes A_2Y_2 \leqslant 0$$

因 $P_1 \cap P_2=\varnothing$，所以 N 的关联矩阵 A 与 $A_1 \otimes V_{n_2}^T$ 在 P_1上对应的列向量完全相同，与 $V_{n_1}^T \otimes A_2$ 在 P_2上对应的列向量完全相同，故：

$$AY=(A_1 \otimes V_{n_2}^T)Y+(V_{n_1}^T \otimes A_2)Y \leqslant 0$$

从而 N 也是结构有界的。同理可证：

定理12 设 N_1，N_2是两个守恒的 Petri 网，若 $P_1 \cap P_2=\varnothing$，则 $N=N_1 \underline{\cup} N_2$是守恒的 Petri 网。

推论7 设 Y_i 是 Petri 网 $N_i(i=1,2)$的 S-不变量，若 $P_1 \cap P_2=\varnothing$，则 m 维向量 $Y=Y_1+Y_2$是 Petri 网 $N=N_1 \underline{\cup} N_2$的 S-不变量。

推论8 设 X_i 是 Petri 网 $N_i(i=1,2)$的 T-不变量，若 $P_1 \cap P_2=\varnothing$，则 n_1n_2维向量 $X=X_1 \otimes X_2$是 Petri 网 $N=N_1 \underline{\cup} N_2$的 T-不变量。

结束语 本文又给出三种网运算：Petri 网的笛并运算、Petri 网的Ⅰ型组合并运算和Ⅱ型组合并运算，它们分别是在库所集、变迁集上取笛积，库所集上取笛积、变迁集上取并，库所集上取并、变迁集上取笛积及权函数都取并得到，讨论了保持网的结构性质的条件。网运算共有十二种，到目前为止已讨论了九种网运算，其它三种网运算的性质及如何降低条件使已有网运算还能很好保持网的性质仍是要进一步讨论的内容。

参 考 文 献

1 Jiang Changjun, Wu Zhehui. Net operations. Journal of Computer Science and Technology, 1992, 7(4): 333～344
2 蒋昌俊. Petri 网的广义笛积运算. 自动化学报, 1993, 19(6): 745～748
3 王培良，蒋昌俊. Petri 网的并运算. 西北大学学报, 1997, 27(增刊): 111～114
4 杜玉越，李孝忠，等. 一种组合 Petri 网的性能分析. 西北大学学报, 1997, 27(增刊): 126～129
5 李孝忠，杜玉越. 两类组合 Petri 网与性能分析. 软件学报, 1998, 9(8): 619～621
6 李孝忠，杜玉越. Petri 网的笛加运算及性质研究. 小型微型计算机系统, 1998, 19(9): 50～54
7 李孝忠，曹德范，等. Petri 网的两类广义组合加网. 计算机科学, 1999, 26(6·增刊): 143～146
8 杜玉越，李孝忠. S-组合 Petri 网的活性分析与实现. 计算机学报, 1998, 21(8): 747～752
9 杜玉越，李孝忠，曹德范. 同步合成网的结构性质分析. 东南大学学报, 1999, 29(5)
10 Murata T. Petri nets: Properties, analysis and applications. Proceedings of the IEEE, 1989, 77(4): 541～580

（上接第69页）

5 传递冗余的处理

假设有规则 R_0相对于规则链 R_{d_1}，R_2，…，R_s（$s>1$）是传递冗余的，对这种冗余性具体的处理算法如下：

```
Function DealWithTransitSpith(R[0],R)
//规则 R[0]相对于规则链 R 是传递冗余的，
Begin
 V←True; //V 指示规则链 R 是否可以推到终点。
 for k=1 to s-1 do
 begin
  U←0;
  for j=1 to R[k].n do//计算
   U←U+R[k].CFE[j]×R[k].WE[j];
   if U>R[k].CF<R[k].T then//规则 R[k]不可用
   begin
      V1←False;
      Break; //跳出循环语句
  end
 end;
 if V=True and R[0].CF<R[s].CF then//应删除 R[0]
   R[0].Remove();
End;
```

结语 对于一般产生式系统知识库中知识的不一致性和冗余性，文[2]已经给出了较好的解决方法。本文提出了"加权产生式规则"的概念，它更加符合客观现实，许多系统都采用了基于加权产生式规则的知识表示方法。本文对基于加权产生式规则知识库不一致性和冗余性的研究有一定的学术意义和实际应用价值。

参 考 文 献

1 吴信东，邹燕. 专家系统技术. 电子工业出版社, 1988
2 陈朝东，黄国兴. 图论生产式知识库维护中的应用. 微型电脑应用, 2000, 16(4)
3 周爱武，汪海威. 基于"规则架＋规则体"知识库的一致性与冗余性检查. 合肥工业大学学报, 1998, 21(3)
4 王清毅，等. 处理知识库中不一致性的超决定逻辑研究. 软件学报. 1998, 9(4)
5 Graham Ian. Expert systems: Knowledge, uncertainty and decision. First published by Chapman and Hall Ltd. 1998