模糊 Petri 网模型的反向推理算法

鲍培明

(南京师范大学数学与计算机科学学院, 210097, 南京)

[摘要]　提出了一种建立在模糊 Petri 网的基本结构上的反向推理算法. 通过建立模糊 Petri 网模型的关联矩阵、库所向量和变迁向量, 运用矩阵运算的基本方法实现. 通过该算法的运行, 可以在模糊 Petri 网模型中抽取出一个子模型, 从而把一个大的、复杂的系统转化为一个只与问题相关的小的系统来处理. 采用数学运算的方法实现的反向推理算法简单, 具有通用性, 它适用于各种类型的模糊 Petri 网结构. 对于其它的大系统生成子系统的问题, 这种矩阵运算的方法也可以借鉴. 同时对该算法中的矩阵运算和模型中的图形结构之间的关系进行了分析, 讨论了算法的复杂性.

[关键词]　模糊, Petri 网, 关联矩阵, 反向推理, 知识库

[中图分类号] O242,　[文献标识码] A,　[文章编号] 1672－1292(2003)03－0021－05

0　引言

petri 网是一个图形化的数学建模工具, 是信息处理系统描述和研究的一个强有力的工具. 模糊集合理论已经广泛地应用在模糊的、不精确的信息的表示和推理中. 模糊 Petri 网(Fuzzy Petri Net, 简称 FPN)是基于模糊产生式规则的知识库系统的良好建模工具, 它结合了 Petri 网的图形描述能力, 使得知识的表示简单、清晰, 而又表现出了知识库系统中规则之间的结构化特性; 它又具有模糊系统的模糊推理能力, 便于知识的分析、推理、测试, 以及决策支持等. 由于模糊 petri 网对知识表示和推理的独特优点, 近年来, 提出了多种模糊 petri 网模型和相应的模糊推理算法[1,3,4,6], 应用到了知识库系统的多个方面, 例如: 决策支持、数据一致性检查、知识库系统维护等.

模糊 petri 网中的推理算法是: 当一个命题在输入条件已知的情况下, 利用模糊 petri 网模型描述的知识库系统进行推理, 求得这个命题的真实程度. 在通常的推理算法中存在两个问题: (1) 对结果命题来讲, 它所需要的输入命题有哪些? (2) 在实际情况中, 对结果命题的推理, 只涉及到基于规则的知识库系统中的一小部分规则. 而在一般的推理算法中, 推理建立在整个知识库系统上进行, 相对算法的复杂度就加大了. 反向推理的目的就是要解决正向推理中的上述问题: 对于需要求解的结果命题, 如何从整个知识库系统中抽取出与结果命题有关的规则, 同时寻找到与结果命题相关的输入条件. 反向推理是逆向于一般的正向推理过程, 又是正向推理算法的一个初始工作.

反向推理算法在[6]中已经提出. [6]中的反向推理建立在模糊 Petri 网的图形结构上, 从结果库所, 沿着弧反向寻找信息, 因此, [6]是一种基于图形的搜索算法, 它的数据结构和算法的复杂性都较大. 本文提出了一种反向推理算法, 它建立在数学中的矩阵运算基础上, 实现知识的反向搜寻. 采用数学运算方法实现的反向推理算法简单、通用, 而且具有并行处理能力.

1　模糊 Petri 网的基本构成

近年来, 提出了许多支持模糊推理和决策支持的模糊 Petri 网模型(Bugarn & Barro, 1994; Cao & Sanderson, 1995; Chen, Ke & Chang, 1990; Looney, 1994; Scarpelli & Gomide, 1993; Scarpelli, Gomide & Yager,

收稿日期: 2003－01－09.

作者简介: 鲍培明, 女, 1966－, 南京师范大学数学与计算机科学学院副教授. 主要从事计算机数据处理等方面的教学与研究.

1996; Yeung & Tsang, 1994, 1998). 模糊 Petri 网的类型大致可以分为: 模糊 Petri 网(FPN)、高级模糊 Petri 网(HLFPN)、有色模糊 Petri 网(CFPN)、自适应模糊 Petri 网(AFPN)等.

反向推理算法是寻找与结果命题相关的输入条件和相关规则, 并不求解结果命题的值. 算法仅涉及知识库系统中的命题、规则, 并不涉及命题和规则中的具体参数. 因此, 反向推理算法建立在模糊 petri 网的基本结构上, 与模糊 petri 网模型的细节属性无关. 本文从上述各种模糊 Petri 网模型中, 抽取出模糊 Petri 网模型的基本结构, 在这个基础上, 建立反向推理算法.

一个模糊 Petri 网的基本结构是一个五元组(P, T, D, I, O), 其中:

$P = \{p_1, p_2, \cdots, p_n\}$, 表示库所结点的有限集合.

$T = \{t_1, t_2, \cdots, t_m\}$, 表示变迁结点的有限集合.

$D = \{d_1, d_2, \cdots, d_n\}$, 表示命题的有限集合, $|P| = |D|$. $P \cap T \cap D = \Phi$.

$I(O): T \to P^{\infty}$, 是输入(输出)函数, 反映变迁到库所的映射.

若$p_i \in I(t)$, $t \in T$, 则从库所 p_i 到变迁 t 之间有一条有向弧, 是变迁 t 的输入弧. p_i 是变迁 t 的输入库所, t 是库所 p_j 的输出变迁. 若 $p_j \in O(t)$, 则从变迁 t 到库所 p_j 之间有一条有向弧, 是变迁 t 的输出弧, p_j 是变迁 t 的输出库所, t 是库所 p_j 的输入变迁.

知识库系统中的每一条模糊产生式规则对应 FPN 中的一个或一组变迁, 规则中的命题一一对应 FPN 中的库所.

例 1: 一个基于规则的知识库系统有如下的模糊产生式规则:

R_1: IF P_1 and P_2 THEN P_4

R_2: IF P_2 THEN P_5

R_3: IF P_2 and P_3 THEN P_9

R_4: IF P_4 THEN P_6

R_5: IF P_5 THEN P_6

R_6: IF P_5 THEN P_8

R_7: IF P_6 THEN P_7

R_8: IF P_9 THEN P_{10} and P_{11}

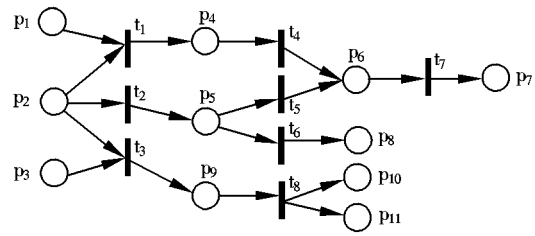

图 1 例 1 的模糊 Petri 网基本结构

根据模糊产生式规则建立的 FPN 模型的基本结构如图 1 所示.

2 反向推理算法

在基于模糊产生式规则的知识库系统中执行知识分析、推理、决策支持等相关的推理时, 从已知的前提命题的值推导出结果命题的值, 相应的推理算法一般都建立在整个知识库系统上[4, 5, 6, 7, 8]. 知识库系统可能是很复杂的规则库系统, 而与已知命题和结果命题相关的规则可能只是规则库系统中很小的一部分. 因此, 有必要建立一个算法, 把与问题相关的部分规则从整个知识库系统中分离出来, 建立在这一小部分规则集上的知识推理要简单得多, 也更容易满足实时性的要求. 另外, 在一个较大的知识库系统上进行决策支持时, 可能需要寻找出与结果命题相关的已知命题应该是哪些. 解决上述问题的推理算法不同于通常意义上的推理算法, 它是从问题的结果出发, 在知识库系统中抽取出一个子系统, 或者寻找出与结果命题相关的前提命题. 因此: 把这种算法称为反向推理算法.

本文提出的反向推理算法建立在矩阵运算的基础上, 算法简单、正确, 但缺乏图形的直观性.

2.1 关联矩阵和向量

(1) 对一个有 n 个库所和 m 个变迁的模糊 Petri 网模型建立一个 $n * m$ 维的关联矩阵 $\boldsymbol{H} = \{h_{ij}\}$, $i = 1, \cdots, n, j = 1, \cdots, m$.

$$h_{ij}=\begin{cases}1 & \text{若 } p_i\in \boldsymbol{I}(t_j),\ p_i\in \boldsymbol{P},\ t_j\in \boldsymbol{T}\\ -1 & \text{若 } p_i\in \boldsymbol{O}(t_j),\ p_i\in \boldsymbol{P},\ t_j\in \boldsymbol{T}\\ 0 & \text{否则}\end{cases}$$

(2)库所向量 $\boldsymbol{A}=(a_0, a_1, \cdots, a_n)^{\mathrm{T}}$. $|\boldsymbol{A}|=|\boldsymbol{P}|$

$$a_i=\begin{cases}1 & \text{若 } p_i \text{ 是结果库所或相关的输入库所, } p_i\in \boldsymbol{P}\\ 0 & \text{否则}\end{cases}$$

(3)变迁向量 $\boldsymbol{B}=(b_0, b_1, \cdots, b_m)^{\mathrm{T}}$. $|\boldsymbol{B}|=|\boldsymbol{T}|$

$$b_i=\begin{cases}1 & \text{若 } t_i \text{ 是与结果库所相关的变迁, } t_i\in \boldsymbol{T}\\ 0 & \text{否则}\end{cases}$$

2.2 矩阵运算符

矩阵运算中使用最大代数中的两个运算符 $\oplus$和 $\otimes$.

$\oplus$ $\boldsymbol{X}\oplus\boldsymbol{Y}=\boldsymbol{Z}$, $\boldsymbol{X}$、$\boldsymbol{Y}$ 和 $\boldsymbol{Z}$ 都是 $r*q$ 维矩阵, $z_{ij}=\max(x_{ij}, y_{ij})$.

$\otimes$ $\boldsymbol{X}\otimes\boldsymbol{Y}=\boldsymbol{Z}$, $\boldsymbol{X}$、$\boldsymbol{Y}$ 和 $\boldsymbol{Z}$ 分别是 $r*s$ 维、$s*q$ 维和 $r*q$ 维矩阵, $z_{ij}=\max\limits_{1\leqslant k\leqslant s}(x_{ik}*y_{kj})$.

2.3 反向推理算法

如果结果命题是 $d_j, \cdots, d_k$, 现在希望从知识库系统中寻找到为推导 $d_j, \cdots, d_k$ 而相关的规则, 以及相应的前提命题, 那么在模糊 Petri 网模型上的反向推理算法为:

步骤 1: $i=1$, 建立初始库所向量 $\boldsymbol{A}_0=(a_0, a_1, \cdots, a_n)^{\mathrm{T}}$ 和初始变迁向量 $\boldsymbol{B}_0=(0_0, 0_1, \cdots, 0_m)^{\mathrm{T}}$, 以及关联矩阵 $\boldsymbol{H}$. 其中: $a_j=\begin{cases}1 & \text{若 } p_j \text{ 是结果命题, } p_j\in \boldsymbol{P}\\ 0 & \text{否则}\end{cases}$.

步骤 2: 计算: $\boldsymbol{B}_i=(-\boldsymbol{H}^{\mathrm{T}})\otimes\boldsymbol{A}_{i-1}$, $\boldsymbol{A}_i=\boldsymbol{H}\otimes\boldsymbol{B}_i\oplus\boldsymbol{A}_{i-1}$.

步骤 3: 若 $B_i\neq B_{i-1}$, 则 $i=i+1$, 返回步骤 2, 否则算法结束.

算法结束时, 向量 $\boldsymbol{A}_i$ 和 $\boldsymbol{B}_i$ 中为"1"的元素所代表的库所和变迁对应于知识库系统中与结果命题 $d_j, \cdots, d_k$ 而相关的前提命题和规则. 实际上, 在模糊 Petri 网模型中删除向量 $\boldsymbol{A}_i$ 和 $\boldsymbol{B}_i$ 中为"0"的元素所代表的库所和变迁, 以及相关的弧, 得到的一个子模型就是与结果命题 $d_j, \cdots, d_k$ 相关的知识库系统的一个子系统模型.

2.4 举例

在例 1 的知识库系统中, 如果结果命题是 P_7 和 P_8, 那么如何在图 1 的模糊 Petri 网模型中抽取出一个与结果命题 P_7 和 P_8 关联的子模型.

根据图 1 的模糊 Petri 网模型建立 $n*m$ 维的关联矩阵 $\boldsymbol{H}$:

	t_1	t_2	t_3	t_4	t_5	t_6	t_7	t_8
p_1	1	0	0	0	0	0	0	0
p_2	1	1	1	0	0	0	0	0
p_3	0	0	1	0	0	0	0	0
p_4	-1	0	0	1	0	0	0	0
p_5	0	-1	0	0	1	1	0	0
p_6	0	0	0	-1	-1	0	1	0
p_7	0	0	0	0	0	0	-1	0
p_8	0	0	0	0	0	-1	0	0
p_9	0	0	-1	0	0	0	0	1
p_{10}	0	0	0	0	0	0	0	-1
p_{11}	0	0	0	0	0	0	0	-1

初始库所向量 $\boldsymbol{A}_0=(0, 0, 0, 0, 0, 0, 1, 1, 0, 0, 0)^{\mathrm{T}}$

和初始变迁向量 $\boldsymbol{B}_0=(0, 0, 0, 0, 0, 0, 0, 0)^{\mathrm{T}}$

$i=1$:

$\boldsymbol{B}_1=(-\boldsymbol{H}^{\mathrm{T}})\otimes\boldsymbol{A}_0=(0, 0, 0, 0, 0, 1, 1, 0)^{\mathrm{T}}$

$\boldsymbol{A}_1=\boldsymbol{H}\otimes\boldsymbol{B}_1\oplus\boldsymbol{A}_0=(0, 0, 0, 0, 1, 1, 1, 1, 0, 0, 0)^{\mathrm{T}}$

$i=2$:

$\boldsymbol{B}_2=(-\boldsymbol{H}^{\mathrm{T}})\otimes\boldsymbol{A}_1=(0, 1, 0, 1, 1, 1, 1, 0)^{\mathrm{T}}$

$\boldsymbol{A}_2=\boldsymbol{H}\otimes\boldsymbol{B}_2\oplus\boldsymbol{A}_1=(0, 1, 0, 1, 1, 1, 1, 1, 0, 0, 0)^{\mathrm{T}}$

$i=3$:

$\boldsymbol{B}_3=(-\boldsymbol{H}^{\mathrm{T}})\otimes\boldsymbol{A}_2=(1,1,0,1,1,1,1,0)^{\mathrm{T}}$

$\boldsymbol{A}_3=\boldsymbol{H}\otimes\boldsymbol{B}_3\oplus\boldsymbol{A}_2=(1,1,0,1,1,1,1,1,1,0,0,0)^{\mathrm{T}}$

$i=4$:

$\boldsymbol{B}_4=(-\boldsymbol{H}^{\mathrm{T}})\otimes\boldsymbol{A}_3=(1,1,0,1,1,1,1,1,1,0,0,0)^{\mathrm{T}}$

$\boldsymbol{A}_4=\boldsymbol{H}\otimes\boldsymbol{B}_4\oplus\boldsymbol{A}_3=(1,1,0,1,1,1,1,1,1,0,0,0)^{\mathrm{T}}$

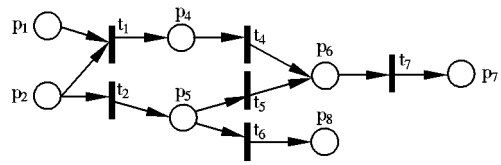

图 2　例 1 的模糊 Petri 网模型的子模型

此时, $\boldsymbol{B}_3=\boldsymbol{B}_4$, 算法结束. 在图 1 的模糊 Petri 网模型中删除向量 $\boldsymbol{A}_4$ 和 $\boldsymbol{B}_4$ 中为"0"的元素所对应的库所 p_3、p_9、p_{10}、p_{11}和变迁 t_3、t_8, 以及相关的弧, 得到的子模型为图 2 所示. 它是知识库系统中抽取的一个子系统模型, 在这个子系统模型上去处理命题 P_7 和 P_8 的有关内容, 而不必在整个系统模型上进行.

3　反向推理算法的讨论

3.1　算法中矩阵运算的理解

对于初始库所向量 $\boldsymbol{A}_0$, 若 $a_k=1$, 则 p_k 是结果命题, $p_k\in\boldsymbol{P}$.

当 $i=1$ 时: $\boldsymbol{B}_1=(-\boldsymbol{H}^{\mathrm{T}})\otimes\boldsymbol{A}_0$, 即 $b_j=\max\limits_{1\leqslant k\leqslant n}(-h_{kj}*a_k)$, $j=1,\cdots,m$. 若存在 $h_{kj}=-1$, 同时 $a_k=1$, 则就有 $b_j=1$, $k=1,\cdots,n$. $h_{kj}=-1$, 说明 $p_k\in\boldsymbol{O}(t_j)$. 因此, $\boldsymbol{B}_1=(-\boldsymbol{H}^{\mathrm{T}})\otimes\boldsymbol{A}_0$ 将结果命题 p_k 的输入变迁 t_j"加入到"变迁向量 $\boldsymbol{B}_1$ 中.

设 $\boldsymbol{G}=\boldsymbol{H}\otimes\boldsymbol{B}_1$, 即 $g_s=\max\limits_{1\leqslant j\leqslant m}(h_{sj}*b_j)$, $s=1,\cdots,n$. 若存在 $h_{sj}=1$, 同时 $b_j=1$, 则就有 $g_s=1$, $j=1,\cdots,m$. $h_{sj}=1$, 说明 $p_s\in\boldsymbol{I}(t_j)$, 因此, $\boldsymbol{G}=\boldsymbol{H}\otimes\boldsymbol{B}$ 将结果命题 p_k 的输入变迁的输入库所 p_s"加入到"向量 $\boldsymbol{G}$ 中. $\boldsymbol{A}_1=\boldsymbol{H}\otimes\boldsymbol{B}_1\oplus\boldsymbol{A}_0=\boldsymbol{G}\oplus\boldsymbol{A}_0$ 反映了结果命题和它们的前提命题. 它们之间的图形关系如图 3 所示.

图 3　$\boldsymbol{B}_1$ 和 $\boldsymbol{A}_1$ 两个矩阵运算的理解

同理, 可以理解 $\boldsymbol{B}_i=(-\boldsymbol{H}^{\mathrm{T}})\otimes\boldsymbol{A}_{i-1}$ 和 $\boldsymbol{A}_i=\boldsymbol{H}\otimes\boldsymbol{B}_i\oplus\boldsymbol{A}_{i-1}$ 的含义.

理解了反向推理算法中矩阵运算的含义, 该算法的正确性也就不难理解了.

3.2　算法的复杂性

以最慢的情况考虑, 每次 $\boldsymbol{B}_i=(-\boldsymbol{H}^{\mathrm{T}})\otimes\boldsymbol{A}_{i-1}$ 运算, 只有一个变迁而且一定有一个变迁"加入"变迁向量 $\boldsymbol{B}_i$ 中, 经过 m 次的循环, 模糊 Petri 网模型中的所有变迁都"加入"变迁向量 $\boldsymbol{B}_i$ 中了. 在第 $m+1$ 次循环中, 没有不同的变迁可"加入"了, 必然 $\boldsymbol{B}_{m+1}=\boldsymbol{B}_m$. 因此算法中的循环次数最多为 $m+1$ 次. 矩阵运算具有并行处理能力, 因此, 该反向推理算法的复杂性为 $\boldsymbol{O}(m*n)$ 或 $\boldsymbol{O}(m*m)$

4　总结

本文提出了一种在模糊 Petri 网模型的基本结构上生成子模型的反向推理算法. 通过该算法的应用, 可以把一个大的、复杂的系统转化为一个只与问题相关的小的系统来处理. 该算法具有通用性, 它适用于各种类型的模糊 Petri 网结构. 对于其它的大系统生成子系统的问题, 这种矩阵运算的方法也是可以借鉴的.

[参考文献]

[1] Looney C G. Fuzzy petri nets and application. In: Fuzzy Reasoning in Information, Decision and Control Systems[M]. S. G. Tzafestas and A, N. Venetsanopoulos, Eds. Norwell, MA: Kluwer, 1994, 511~527.

[2] Xiaoou Li, Wen Yu, Lara-Rosano F. Dynamic Knowledge Inference and Learning under Adaptive Fuzzy Petri Net Framework[J]. IEEE Transactions on Systems, Man, and Cybernetie-Part C: Applications and Reviews, November 2000, 30(4): 442~449.

[3] Xiaoou Li, Lara-Rosano F. Adaptive Fuzzy Petri Nets for Dynamic Knowledge Representation and Inference[J]. Expert Systems with Applications, 2000, 19(3): 235~241.

[4] Shyi-ming Chen, Jyh-sheng Ke, Jin-fu Chang. knowledge Representation Using Fuzzy Petri Nets[J]. IEEE Transactions on Knowledge and Data Engineering, September 1990, 2(3): 311~319.

[5] Koriem S M. A Fuzzy Petri Net Tool for Modeling and Verification of Knowledge-Based Systems[J]. The Computer Journal, 2000, 43(3): 206~223.

[6] Scarpelli H, Gomide F, Yager R. A Reasoning Algorithm for High-Level Fuzzy Petri Nets[J]. IEEE Transactions on Fuzzy Systems, August 1996, 4(3): 282~294.

[7] Fay A. A Fuzzy Knowledge-Based System for Railway Traffic Control[J]. Engineering Applications of Artificial Intelligence, 2000, 13: 719~729.

[8] Shyue-Liang Wang, Yi-huey Wu. Reasoning in Fuzzy Production Systems When Input Information is Incomplete[A]. In: 1999 IEEE International Fuzzy Systems Conference Proceedings. Seoul, Korea: August 22~25, 1999, 1557~1561.

Reverse Reasoning Algorithm Based on the Fuzzy Petri Net Model

Bao Peiming

(College of Mathematics and Computer Science, Nanjing Normal University, 210097, Nanjing, PRC)

Abstract: This paper proposes a reverse reasoning algorithm on the basic struture of Fuzzy Petri Net model. Based on the model, an incidience matrix, a palce vector and a transition vector are built, with the method of matrix operation used in the algorithm. A subnet from the FPN can be extracted by operating the algorithm. Therefore a large and complex system can be transformed into a small system relating to the problems. Being simple and universal, this algorithm can bc applied to the majority of FPN model. For other problems involving a large system producing sub_systems, the matrix expression method can be used for reference. The relation is analyzed between the matrix expression in the algorithm and the graphics stuctrue in the model, and its complexity is also discussed in this paper.

Key words: fuzzy, Petri net, incidence matrix, reverse reasoning, knowledge base

[责任编辑: 刘健]