第三次作业

5.6

$$O + O = R + 10 \cdot X_1$$
$$X_1 + W + W = U + 10 \cdot X_2$$
$$X_2 + T + T = O + 10 \cdot X_3$$
$$X_3 = F$$

$$\begin{array}{r} T\ W\ O \\ +\ T\ W\ O \\ \hline F\ O\ U\ R \end{array}$$

	X_3	X_2	X_1	F	T	W	O	U	R
初始域	{0,1}	{0,1}	{0,1}	{0,1⋯,9}	{0,1,..,9}	{0,1,..,9}	{0,1,..,9}	{0,1,..,9}	{0,1,..,9}
After $X_3 = 1$	1	{0,1}	{0,1}	{1}	{0,1,..,9}	{0,1,..,9}	{0,1,..,9}	{0,1,..,9}	{0,1,..,9}
After $F = 1$	1	{0,1}	{0,1}	1	{5,..,9}	{0,2,..,9}	{0,2,..,9}	{0,2,..,9}	{0,2,..,9}
After $X_2 = 0$	1	0	{0,1}	1	{5,..,9}	{0,2,3,4}	{0,2,4,6,8}	{0,2,..,9}	{0,2,..,9}

5.6

$$O + O = R + 10 \cdot X_1$$
$$X_1 + W + W = U + 10 \cdot X_2$$
$$X_2 + T + T = O + 10 \cdot X_3$$
$$X_3 = F$$

$$\begin{array}{r} T\ W\ O \\ +\ T\ W\ O \\ \hline F\ O\ U\ R \end{array}$$

	X_3	X_2	X_1	F	T	W	O	U	R
After $X_1 = 0$	1	0	0	1	{5,6,7}	{0,2,3,4}	{0,2,4}	{0,4,6,8}	{0,4,8}
After O=4	1	0	0	1	{7}	{0,3}	4	{0,6}	{8}
After T=7	1	0	0	1	7	{0,3}	4	{0,6}	{8}
After R=8	1	0	0	1	7	{0,3}	4	{0,6}	8
After W=3	1	0	0	1	7	3	4	{6}	8
After U=6	1	0	0	1	7	3	4	6	8

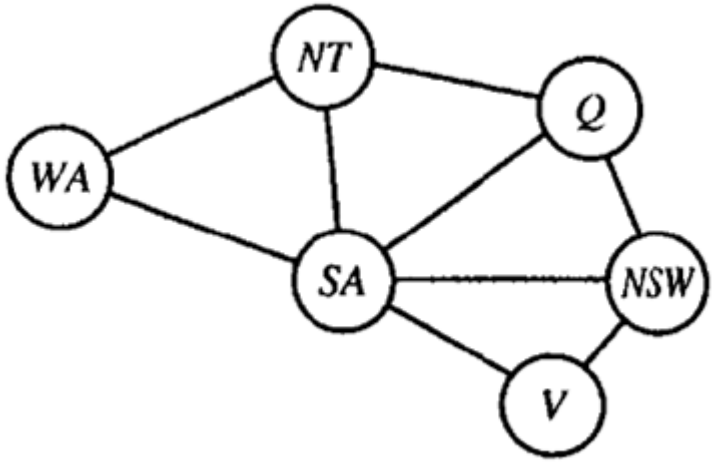
5.8

用 AC-3 算法说明弧相容对图 5.1 中所示问题能够检测出不完全赋值{*WA* = *red*, *V* = *blue*}的不相容。

- 证明：

  (SA,WA)消除不相容,SA={G,B}
  (SA,V)消除不相容,SA={G}
  同理，得到NT={B}, NSW={R}
  (Q,NT)消除不相容, Q={R,G}
  (Q,NSW)消除不相容, Q={G}
  (Q,SA)消除不相容, Q={}

  所以{WA=R, V=B}不相容



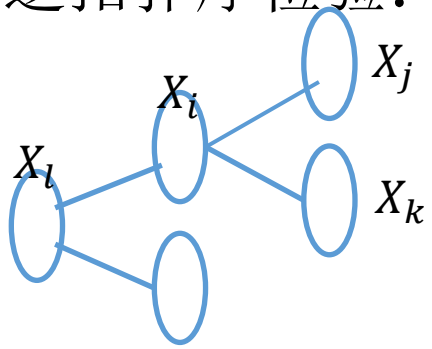
5.9

在树状结构 CSP 问题的最坏情况下运行 AC-3 算法的复杂度是多少？

- 假设有n个顶点，值域中最多有d个取值。树状结构下，弧的条数为$O(n)$，检验每条弧的复杂度为$O(d^2)$.
- 每条弧只需要检验一次，故总的复杂度为$O(nd^2)$.
- 如何做到每条弧只检验一次？

    1. 采用逆拓扑序进行检验

- 逆拓扑序检验：



当检验弧(X_i, X_j)时，X_i值域发生变化，需要将弧(X_l, X_i)重新检验。同理，当检验弧(X_i, X_k)时，需要重新检验弧(X_l, X_i)。

在逆拓扑序下，只有当X_i与其所有子节点组成的弧检验完后，(X_l, X_i)才会被检验，而(X_l, X_i)同时也是此前不断重新加入检验集合中的弧。这就保证了在树中，一条边被检验完毕后，不再会被加入待检验集合。

function AC-3(*csp*) **returns** the CSP, possibly with reduced domains
  inputs: *csp*, a binary CSP with variables $\{X_1, X_2, \ldots, X_n\}$
  local variables: *queue*, a queue of arcs, initially all the arcs in *csp*

  while *queue* is not empty **do**
    $(X_i, X_j) \leftarrow$ REMOVE-FIRST(*queue*)
    if REMOVE-INCONSISTENT-VALUES(X_i, X_j) **then**
      for each X_k **in** NEIGHBORS[X_i] **do**
        add (X_k, X_i) to *queue*

function REMOVE-INCONSISTENT-VALUES(X_i, X_j) **returns** true iff we remove a value
  removed $\leftarrow$ *false*
  for each x **in** DOMAIN[X_i] **do**
    if no value y in DOMAIN[X_j] allows (x,y) to satisfy the constraint between X_i and X_j
      then delete x from DOMAIN[X_i]; *removed* $\leftarrow$ *true*
  return *removed*

第七次作业

9.3 假定知识库中只包括一条语句：$\exists x\ AsHighAs(x, Everest)$。下列那个语句是应用存在量词实例化以后的合法结果？

a. $AsHighAs(Everest, Everest)$

b. $AsHighAs(Kilimanjaro, Everest)$

c. $AsHighAs(Kilimanjaro, Everest) \wedge AsHighAs(BenNevis, Everest)$ （在两次应用之后）

a. 错误，Everest为对象名，不能再属于另一个对象

b. 正确

c. 错误，Kilimanjaro & BenNevis 应用两次实例化

第三版 p.269

只要 C_1 不在知识库的别处出现。基本上，存在语句说明存在某些满足条件的对象，实例化过程仅仅是给该对象命名。自然地，该名字不能已经属于另一个对象。数学提供了一个很好的例子：假设我们发

存在实例化比全称实例化更加复杂，它在推理中也承担了稍微有些不同的角色。全称实例化可以多次应用从而获得许多不同的结果，而存在实例化只能应用一次，然后存在量化语句就可以被抛弃。

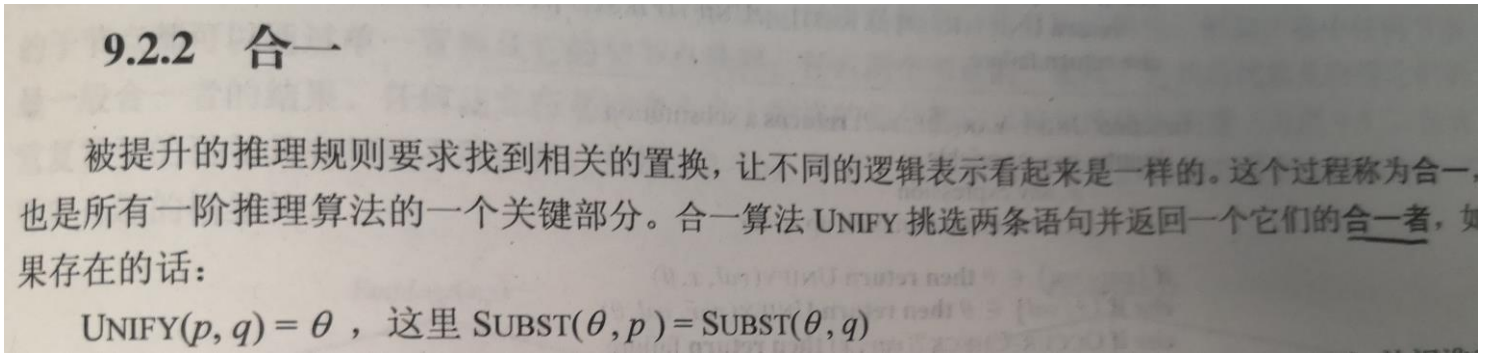
9.4 对于下列每对原子语句，如果存在，请给出最一般合一置换：

a. $P(A, B, B), P(x, y, z)$

b. $Q(y, G(A, B)), Q(G(x, x), y)$

c. $Older(Father(y), y), Older(Father(x), John)$

d. $Knows(Father(y), y), Knows(x, x)$



9.2.2 合一

被提升的推理规则要求找到相关的置换，让不同的逻辑表示看起来是一样的。这个过程称为合一，也是所有一阶推理算法的一个关键部分。合一算法 UNIFY 挑选两条语句并返回一个它们的合一者，如果存在的话：

UNIFY$(p, q) = \theta$ ，这里 SUBST(θ, p) = SUBST(θ, q)

a. {x/A, y/B, z/B}

b. 不存在，y = G(A, B), y = G(x, x)，所以需要x = A, x = B，矛盾。（x不能同时选取A和B二值）

c. {x/John, y/John}

d. 不存在，需要x = y, x = Father(y)，矛盾。

9.9 写出下列语句的逻辑表示，使得它们适合应用一般化分离规则：

a. 马、奶牛和猪都是哺乳动物。

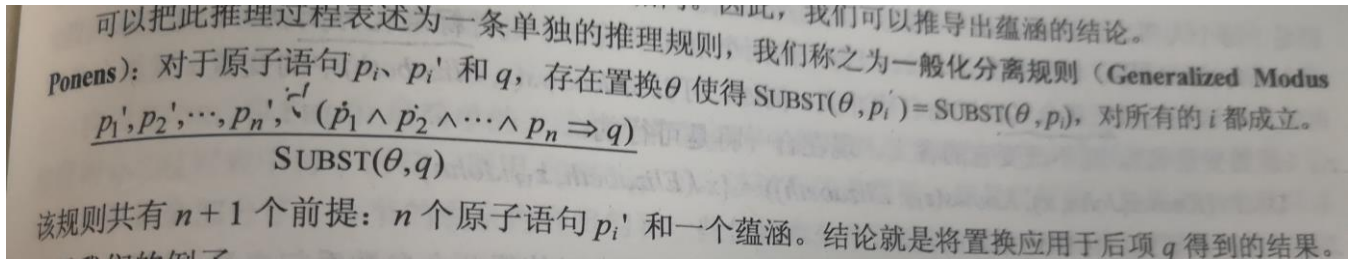
b. 一匹马的后代是马。

c. Bluebeard 是一匹马。

d. Bluebeard 是 Charlie 的父亲。

e. 后代和双亲是逆关系。

f. 每个哺乳动物都有一个双亲。



可以把此推理过程表述为一条单独的推理规则，我们称之为一般化分离规则（Generalized Modus Ponens）：对于原子语句 p_i、p_i' 和 q，存在置换 θ 使得 $\text{SUBST}(\theta, p_i') = \text{SUBST}(\theta, p_i)$，对所有的 i 都成立。

$$\frac{p_1', p_2', \cdots, p_n', (p_1 \wedge p_2 \wedge \cdots \wedge p_n \Rightarrow q)}{\text{SUBST}(\theta, q)}$$

该规则共有 $n+1$ 个前提：n 个原子语句 p_i' 和一个蕴涵。结论就是将置换应用于后项 q 得到的结果。

a. $Horse(x) \Rightarrow Mammal(x)$
$Cow(x) \Rightarrow Mammal(x)$
$Pig(x) \Rightarrow Mammal(x)$

b. $Offspring(x, y) \wedge Horse(y) \Rightarrow Horse(x)$

c. $Horse(Bluebeard)$

d. $Parent(Bluebeard, Charlie)$

e. $Offspring(x, y) \Rightarrow Parent(y, x)$
$Parent(x, y) \Rightarrow Offspring(y, x)$
(Note we couldn't do $Offspring(x, y) \Leftrightarrow Parent(y, x)$ because that is not in the form expected by Generalized Modus Ponens.)

f. $Mammal(x) \Rightarrow Parent(G(x), x)$ (here G is a Skolem function).

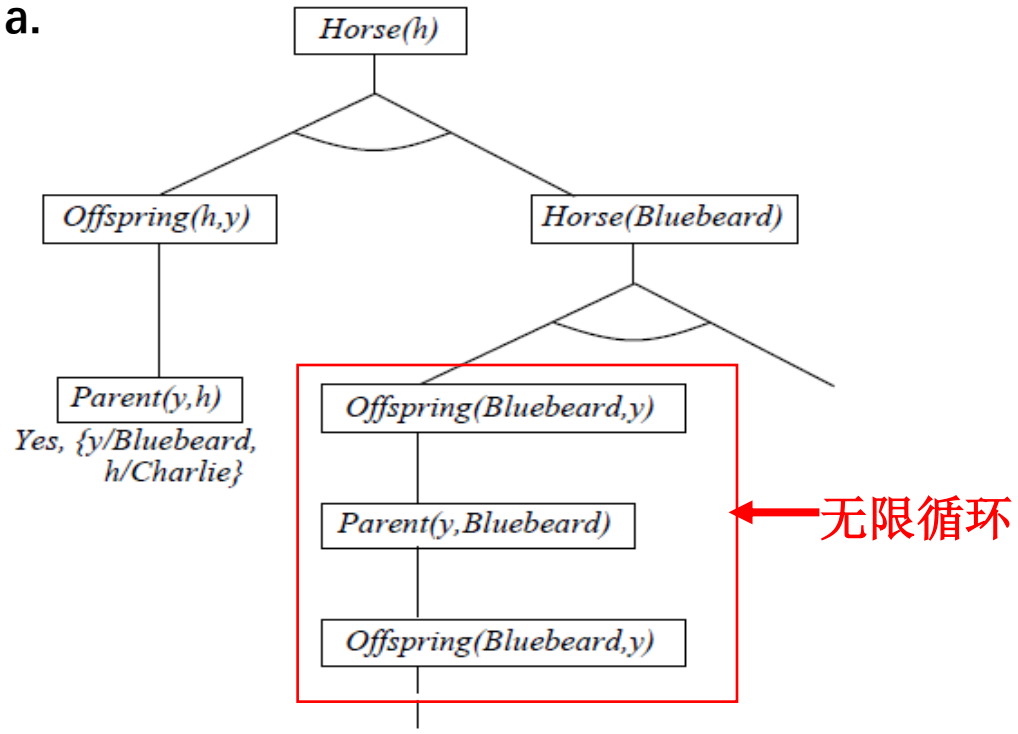
9.10 本习题中，我们将采用你在习题 9.9 中写出的语句，运用反向链接算法来回答一个问题。

a. 画出由一个穷举反向链接算法为查询 $\exists h$ *horse(h)*生成的证明树，其中子句按照给定的顺序进行匹配。

a. 你对于本领域注意到了什么？

b. 实际上从你的语句中得出了多少个 *h* 的解？

a.



a. $Horse(x) \Rightarrow Mammal(x)$
$Cow(x) \Rightarrow Mammal(x)$
$Pig(x) \Rightarrow Mammal(x)$

b. $Offspring(x, y) \wedge Horse(y) \Rightarrow Horse(x)$

c. $Horse(Bluebeard)$

d. $Parent(Bluebeard, Charlie)$

e. $Offspring(x, y) \Rightarrow Parent(y, x)$
$Parent(x, y) \Rightarrow Offspring(y, x)$
(Note we couldn't do $Offspring(x, y) \Leftrightarrow Parent(y, x)$ because that is not in the form expected by Generalized Modus Ponens.)

f. $Mammal(x) \Rightarrow Parent(G(x), x)$ (here G is a Skolem function).

a. 因为子句b：一匹马的后代是马

$$Offspring(x, y) \wedge Horse(y) \Rightarrow Horse(x)$$

出现在子句c的前面，得到无限循环

b. 容易证明：Bluebeard 和 Charlie 都是马

图 9.6 显示了一个简单的反向链接算法，FOL-BC-ASK。它用只包含单个元素（即原始查询）的目标列表来调用，并返回满足查询的所有置换的集合。目标列表可以认为是一个等待处理的“栈”；如果所有的栈内目标都可以得到满足，则当前的证明分支是成功的。算法选取列表中的第一个目标，在知识库中寻找正文字（或称为**头**）能与该目标合一的每个子句。每个这样的子句创建一个新的递归调用，在该递归过程中，子句的前提（或称为**体**）都被加入到目标栈内。记住事实就是只有头没有体的子句，因此当目标和某个已知事实合一时，不会有新的子目标添加到栈里，目标也就得到了解决。图 9.7 是

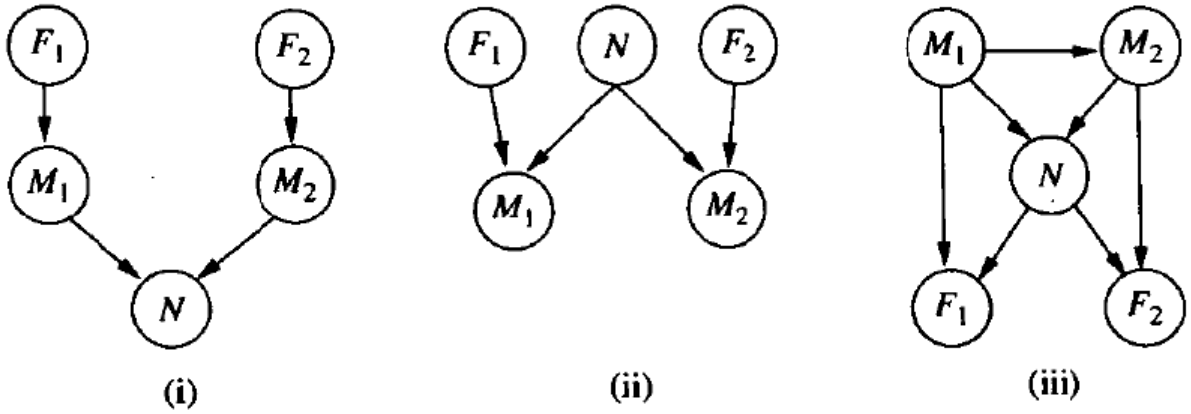
第九次作业

14.3 两个来自世界上不同地方的宇航员同时用他们自己的望远镜观测了太空中某个小区域内恒星的数目 N。记这两个宇航员的测量结果分别为 M_1 和 M_2。通常，测量中会有不超过 1 颗恒星的误差，发生错误的概率 e 很小。而每台望远镜可能发生（发生的概率 f 更小一些）对焦不准确（分别记作 F_1 和 F_2），在这种情况下科学家会少数 3 颗甚至更多的恒星（或者说，当 N 小于 3 时，连一颗恒星都观测不到）。考虑图 14.19 所示的 3 种贝叶斯网络结构。

a. 这 3 种网络结构哪些是对上述信息的正确（但不一定是高效的）表示？

b. 哪一种网络结构是最好的？请解释。

c. 当 $N \in \{1, 2, 3\}$，$M_1 \in \{0, 1, 2, 3, 4\}$时，写出一个关于 $\mathbf{P}(M_1 \mid N)$的条件概率表。概率分布表里的每个条目都应该表达为参数 e 和/或 f 的一个函数。



(ii) F1, F2, N, M1, M2

(iii) M1, M2, N, F1, F2

a.写出联合概率的的公式，以（ii）为例：

$$P(F_1, F_2, N, M_1, M_2) = P(F_1)P(F_2)P(N)P(M_1|F_1, N)P(M_2|F_2, N)$$

（ii）和（iii）正确；（i）错误；

b.（ii）更好

c. 当 $N \in \{1, 2, 3\}$，$M_1 \in \{0, 1, 2, 3, 4\}$时，写出一个关于 $\mathbf{P}(M_1 | N)$的条件概率表。
概率分布表里的每个条目都应该表达为参数 e 和/或 f 的一个函数。

$$P(M_1|N) = \frac{P(M_1, N)}{P(N)} = \frac{\sum_{f_1} P(M_1, N, f_1)}{P(N)} = \frac{P(M_1, N, F_1) + P(M_1, N, \neg \mathrm{F}_1)}{P(N)}$$
$$= \frac{P(M_1|N, F_1)P(N)\mathrm{P}(\mathrm{F}_1) + \mathrm{P}(\mathrm{M}_1|\mathrm{N}, \neg F_1)\mathrm{P}(\mathrm{N})\mathrm{P}(\neg F_1)}{P(N)}$$
$$= P(M_1|N, F_1)P(F_1) + \mathrm{P}(\mathrm{M}_1|\mathrm{N}, \neg F_1)\mathrm{P}(\neg F_1)$$

	$N=1$	$N=2$	$N=3$
$M_1=0$	f + e(1-f)	f	f
$M_1=1$	(1-2e)(1-f)	e(1-f)	0.0
$M_1=2$	e(1-f)	(1-2e)(1-f)	e(1-f)
$M_1=3$	0.0	e(1-f)	(1-2e)(1-f)
$M_1=4$	0.0	0.0	e(1-f)

（1）分为对焦准确，对焦不准两种情况；

（2）多数一颗和少数一颗的概率都是e，正确的概率是（1-2e）；

注：另一种理解为多数一颗或少数一颗的概率是e/2，正确概率是（1-e），均可。

14.13 考虑图 14.22(ii)的网络，假设两个望远镜完全相同。$N \in \{1,2,3\}$，$M_1, M_2 \in \{0,1,2,3,4\}$，CPT 表和习题 14.12 所描述的一样。使用枚举算法（图 14.9）计算概率分布 $\mathbf{P}(N \mid M_1=2, M_2=2)$。

$$P(N|M1=2,M2=2)=\frac{1}{P(M1=2,M2=2)}P(N,M1=2,M2=2)$$
$$=\frac{1}{\mathrm{P}(M1=2,M2=2)}\sum_{F1,F2}P(N,M1=2,M2=2,F1,F2)$$
$$=\frac{1}{\mathrm{P}(M1=2,M2=2)}\sum_{F1,F2}P(N)P(F1)F(F2)P(M1=2|F1,N)P(M2=2|F2,N)$$

由于N取值不大于3，F1=true 或 F2=true时，相应的概率$P(M1=2|F1,N)$或$(M2=2|F2,N)$ 为零，因此上式总只有一项（F1=false, 且F2=false时) 非零

$$P(N|M1=2,M2=2)=\frac{1}{P(M1=2,M2=2)}P(N)P(F1=false)F(F2=false)P(M1=2|F1=false,N)P(M2=2|F2=false,N)$$
$$=\frac{1}{P(M1=2,M2=2)}P(N)(1-f)^2P(M1=2|F1=false,N)P(M2=2|F2=false,N)$$

$N=1$时，$P(N=1|M1=2,M2=2)=\frac{1}{P(M1=2,M2=2)}P(N=1)(1-f)^2e^2$

同理，

$$P(N=2|M1=2,M2=2)=\frac{1}{P(M1=2,M2=2)}P(N=2)(1-f)^2(1-2e)^2$$

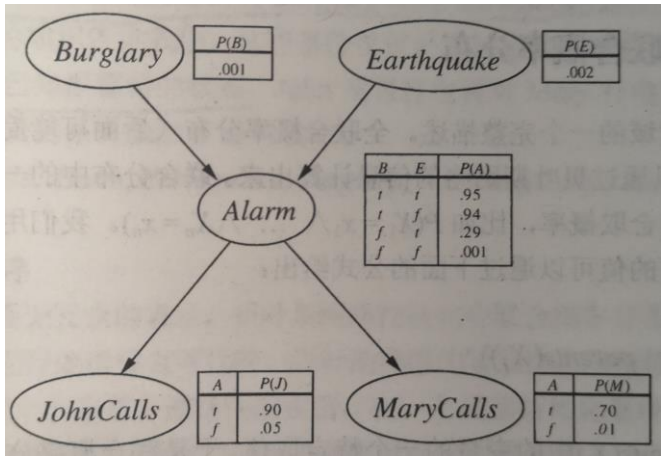
$$P(N=3|M1=2,M2=2)=\frac{1}{P(M1=2,M2=2)}P(N=3)(1-f)^2e^2$$

14.7 这道习题是关于图 14.10 所示的变量消元算法的：

a. 第 14.4 节对如下查询应用了变量消元算法

$$\mathbf{P}(Burglary \mid JohnCalls = true, MaryCalls = true)$$

执行必要的计算，并检验计算结果的正确性。



$$
\begin{aligned}
&P(B|j,m) \\
&= \alpha P(B) \sum_e P(e) \sum_a P(a|b,e) P(j|a) P(m|a) \\
&= \alpha P(B) \sum_e P(e) \left[.9 \times .7 \times \begin{pmatrix} .95 & .29 \\ .94 & .001 \end{pmatrix} + .05 \times .01 \times \begin{pmatrix} .05 & .71 \\ .06 & .999 \end{pmatrix} \right] \\
&= \alpha P(B) \sum_e P(e) \begin{pmatrix} .598525 & .183055 \\ .59223 & .0011295 \end{pmatrix} \\
&= \alpha P(B) \left[.002 \times \begin{pmatrix} .598525 \\ .183055 \end{pmatrix} + .998 \times \begin{pmatrix} .59223 \\ .0011295 \end{pmatrix} \right] \\
&= \alpha \begin{pmatrix} .001 \\ .999 \end{pmatrix} \times \begin{pmatrix} .59224259 \\ .001493351 \end{pmatrix} \\
&= \alpha \begin{pmatrix} .00059224259 \\ .0014918576 \end{pmatrix} \\
&\approx \langle .284, .716 \rangle
\end{aligned}
$$

10次乘法，4次加法

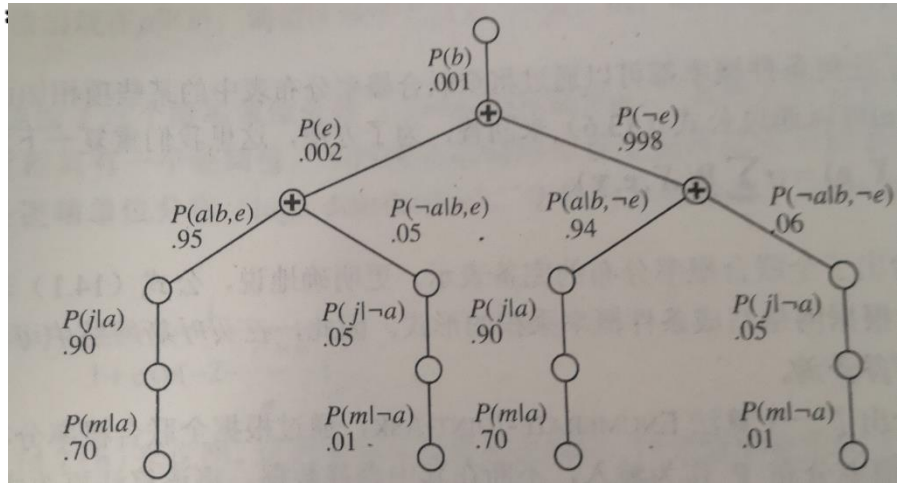
4次乘法，2次加法

2次乘法

1次加法，2次除法

b. 统计计算中算术运算的次数，将其与枚举算法完成的运算次数进行比较。

c. 假设贝叶斯网络具有*链式*结构，即由一个布尔随机变量序列 $X_1, \ldots, X_n$ 构成，其中 $Parents(X_i) = \{X_{i-1}\}$，对于 $i = 2, \ldots, n$。请问通过枚举算法计算 $\mathbf{P}(X_1 \mid X_n = true)$ 的复杂度是什么？用变量消元算法呢？



b. 共有7次加法，16次乘法，2次除法
而枚举运算需要7次加法，22次乘法，2次除法

c.枚举：两个深度（n-2）完全二叉树，$O(2^n)$

变量消元：

$$
\begin{aligned}
&\mathbf{P}(X_1|X_n = true) \\
&= \alpha\mathbf{P}(X_1)\ldots\sum_{x_{n-2}} P(x_{n-2}|x_{n-3})\sum_{x_{n-1}} P(x_{n-1}|x_{n-2})P(X_n = true|x_{n-1}) \\
&= \alpha\mathbf{P}(X_1)\ldots\sum_{x_{n-2}} P(x_{n-2}|x_{n-3})\sum_{x_{n-1}} \mathbf{f}_{X_{n-1}}(x_{n-1}, x_{n-2})\mathbf{f}_{X_n}(x_{n-1}) \\
&= \alpha\mathbf{P}(X_1)\ldots\sum_{x_{n-2}} P(x_{n-2}|x_{n-3})\mathbf{f}_{\overline{X_{n-1}X_n}}(x_{n-2})
\end{aligned}
$$

最后一行中最后一项的计算只与x_{n-1}, x_{n-2}相关，而该过程的代价独立于n，为一常量，
故复杂度为O(n)