Artificial Intelligence

Section 7 : **Logic Agent**

推理：判断逻辑蕴涵

- Method 1: ***model-checking*** （模型检验）
  - 一种简单的枚举推理
  - 枚举所有的模型(可能世界)，并验证语句在所有模型中为真
- Method 2: Application of inference rules （逻辑规则）
- Method 3: ***theorem-proving*** （定理证明）

Wumpus world sentences

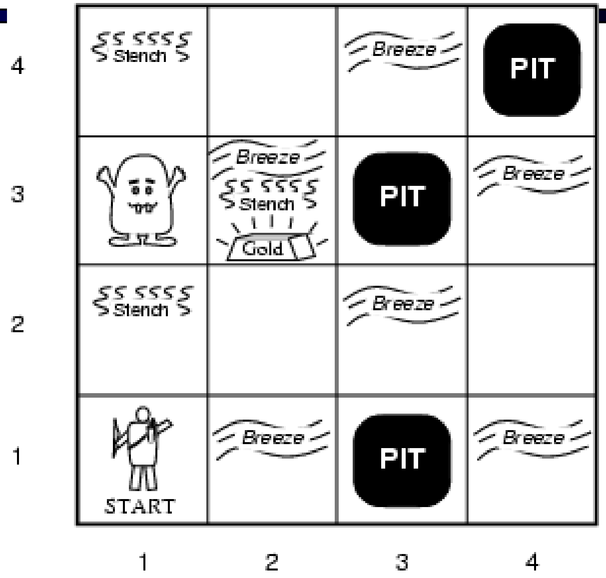
Let $P_{i,j}$ be true if there is a pit in [i, j].

Let $B_{i,j}$ be true if there is a breeze in [i, j].

$R_1 : \neg P_{1,1}$

$R_2 : \neg B_{1,1}$

$R_3 : B_{2,1}$



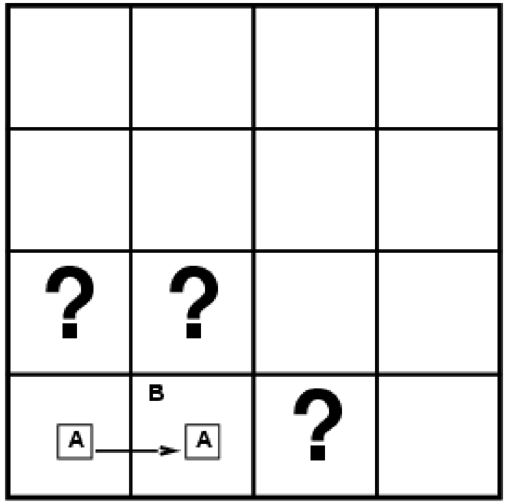
- "Pits cause breezes in adjacent squares"

$R_4 : B_{1,1} \Leftrightarrow (P_{1,2} \vee P_{2,1})$

$R_5 : B_{2,1} \Leftrightarrow (P_{1,1} \vee P_{2,2} \vee P_{3,1})$

此时的知识库 KB 由以上五条语句组成

KB=$R_1 \wedge R_2 \wedge R_3 \wedge R_4 \wedge R_5$



Truth tables for inference

$B_{1,1}$	$B_{2,1}$	$P_{1,1}$	$P_{1,2}$	$P_{2,1}$	$P_{2,2}$	$P_{3,1}$	R_1	R_2	R_3	R_4	R_5	KB
false	*false*	*false*	*false*	*false*	*false*	*false*	*true*	*true*	*true*	*true*	*false*	*false*
false	*false*	*false*	*false*	*false*	*false*	*true*	*true*	*true*	*false*	*true*	*false*	*false*
⋮	⋮	⋮	⋮	⋮	⋮	⋮	⋮	⋮	⋮	⋮	⋮	⋮
false	*true*	*false*	*false*	*false*	*false*	*false*	*true*	*true*	*false*	*true*	*true*	*false*
false	*true*	*false*	*false*	*false*	*false*	*true*	*true*	*true*	*true*	*true*	*true*	*true*
false	*true*	*false*	*false*	*false*	*true*	*false*	*true*	*true*	*true*	*true*	*true*	*true*
false	*true*	*false*	*false*	*false*	*true*	*true*	*true*	*true*	*true*	*true*	*true*	*true*
false	*true*	*false*	*false*	*true*	*false*	*false*	*true*	*false*	*false*	*true*	*true*	*false*
⋮	⋮	⋮	⋮	⋮	⋮	⋮	⋮	⋮	⋮	⋮	⋮	⋮
true	*true*	*true*	*true*	*true*	*true*	*true*	*false*	*true*	*true*	*false*	*true*	*false*

Figure 7.9 A truth table constructed for the knowledge base given in the text. KB is true if R_1 through R_5 are true, which occurs in just 3 of the 128 rows (the ones underlined in the right-hand column). In all 3 rows, $P_{1,2}$ is false, so there is no pit in [1,2]. On the other hand, there might (or might not) be a pit in [2,2].

KB $\models \neg P_{1,2}$

R_6 : $\neg$ $P_{1,2}$

With seven symbols, there are 2^7 =128 possible models; in three of these, KB is true

通过枚举推理

- 用于判断蕴涵 KB ⊨ α 的真值表枚举推理算法

```
function TT-ENTAILS?(KB, α) returns true or false
    symbols ← a list of the proposition symbols in KB and α
    return TT-CHECK-ALL(KB, α, symbols, [ ])

function TT-CHECK-ALL(KB, α, symbols, model) returns true or false
    if EMPTY?(symbols) then
        if PL-TRUE?(KB, model) then return PL-TRUE?(α, model)
        else return true
    else do
        P ← FIRST(symbols); rest ← REST(symbols)
        return TT-CHECK-ALL(KB, α, rest, EXTEND(P, true, model)) and
                  TT-CHECK-ALL(KB, α, rest, EXTEND(P, false, model))
```

如果 KB 在模型 model 中为真， PL-Ture?(KB, model) 返回真

- 对所有模型来说，深度优先的枚举（递归枚举）是完备的

- 对于 n 个符号，时间复杂度为 $O(2^n)$，空间复杂度为 $O(n)$

推理：判断逻辑蕴涵

- Method 1: ***model-checking*** （模型检验）
  - 枚举所有的模型（可能世界），并验证语句在所有模型中为真
- Method 2: Application of inference rules （推理规则）
  - 在知识库的语句上，直接应用推理规则，构建目标语句的证明
- Method 3: ***theorem-proving*** （定理证明）

7.5.1 推理规则

- **Modus Ponens:**

  假言推理规则

$$\frac{\alpha \Rightarrow \beta, \qquad \alpha}{\beta}$$

- **And-Elimination:**

  消去合取词

$$\frac{\alpha \wedge \beta}{\alpha}$$

- **Other rules:**

  逻辑等价

例如：双向蕴涵 $$\frac{\alpha \Leftrightarrow \beta}{(\alpha \Rightarrow \beta) \wedge (\beta \Rightarrow \alpha)} \quad \text{and} \quad \frac{(\alpha \Rightarrow \beta) \wedge (\beta \Rightarrow \alpha)}{\alpha \Leftrightarrow \beta}$$

逻辑等价性

- 任意两个语句是逻辑等价的 iff 它们在相同模型中互相蕴涵：

$\alpha \equiv \beta$ iff $\alpha \models \beta$ and $\beta \models \alpha$

交换律

$(\alpha \wedge \beta) \equiv (\beta \wedge \alpha)$ commutativity of $\wedge$

$(\alpha \vee \beta) \equiv (\beta \vee \alpha)$ commutativity of $\vee$

结合律

$((\alpha \wedge \beta) \wedge \gamma) \equiv (\alpha \wedge (\beta \wedge \gamma))$ associativity of $\wedge$

$((\alpha \vee \beta) \vee \gamma) \equiv (\alpha \vee (\beta \vee \gamma))$ associativity of $\vee$

$\neg(\neg\alpha) \equiv \alpha$ double-negation elimination

$(\alpha \Rightarrow \beta) \equiv (\neg\beta \Rightarrow \neg\alpha)$ contraposition

假言易位 逆否命题的逻辑等价

$(\alpha \Rightarrow \beta) \equiv (\neg\alpha \vee \beta)$ implication elimination

消去蕴含词

$(\alpha \Leftrightarrow \beta) \equiv ((\alpha \Rightarrow \beta) \wedge (\beta \Rightarrow \alpha))$ biconditional elimination

$\neg(\alpha \wedge \beta) \equiv (\neg\alpha \vee \neg\beta)$ de Morgan

De Morgan 定律

$\neg(\alpha \vee \beta) \equiv (\neg\alpha \wedge \neg\beta)$ de Morgan

分配律

$(\alpha \wedge (\beta \vee \gamma)) \equiv ((\alpha \wedge \beta) \vee (\alpha \wedge \gamma))$ distributivity of $\wedge$ over $\vee$

$(\alpha \vee (\beta \wedge \gamma)) \equiv ((\alpha \vee \beta) \wedge (\alpha \vee \gamma))$ distributivity of $\vee$ over $\wedge$

Wumpus world sentences

R_1 ： $\neg P_{1,1}$

R_2 ： $\neg B_{1,1}$

R_3 ： $B_{2,1}$

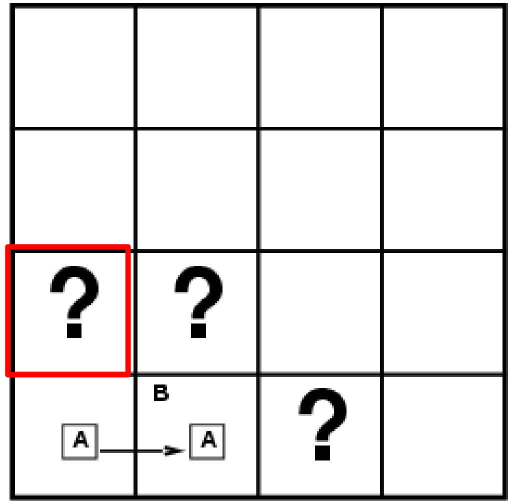
R_4 ： $B_{1,1} \Leftrightarrow (P_{1,2} \vee P_{2,1})$

R_5 ： $B_{2,1} \Leftrightarrow (P_{1,1} \vee P_{2,2} \vee P_{3,1})$

证明： KB $\models$ α

KB ： $R_1 \wedge R_2 \wedge R_3 \wedge R_4 \wedge R_5$

α： $\neg P_{1,2}$



消去双向蕴涵 to R_4

$$R_6 : (B_{1,1} \Rightarrow (P_{1,2} \vee P_{2,1})) \wedge ((P_{1,2} \vee P_{2,1}) \Rightarrow B_{1,1})$$

Wumpus world sentences

R_1 ： $\neg P_{1,1}$

R_2 ： $\neg B_{1,1}$

R_3 ： $B_{2,1}$

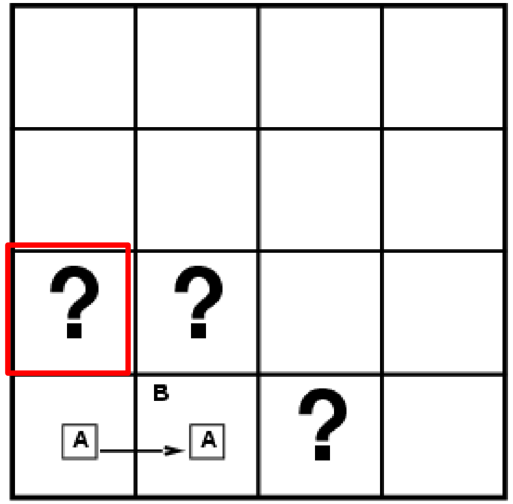
R_4 ： $B_{1,1} \Leftrightarrow (P_{1,2} \vee P_{2,1})$

R_5 ： $B_{2,1} \Leftrightarrow (P_{1,1} \vee P_{2,2} \vee P_{3,1})$

证明： KB $\models \alpha$

KB ： $R_1 \wedge R_2 \wedge R_3 \wedge R_4 \wedge R_5 \wedge R_6$

α ： $\neg P_{1,2}$



$R_6 : (B_{1,1} \Rightarrow (P_{1,2} \vee P_{2,1})) \wedge ((P_{1,2} \vee P_{2,1}) \Rightarrow B_{1,1})$

消去合取词 to R_6

$R_7 : (P_{1,2} \vee P_{2,1}) \Rightarrow B_{1,1}$

Wumpus world sentences

R_1 : $\square\ P_{1,1}$

R_2 : $\square\ B_{1,1}$

R_3 : $B_{2,1}$

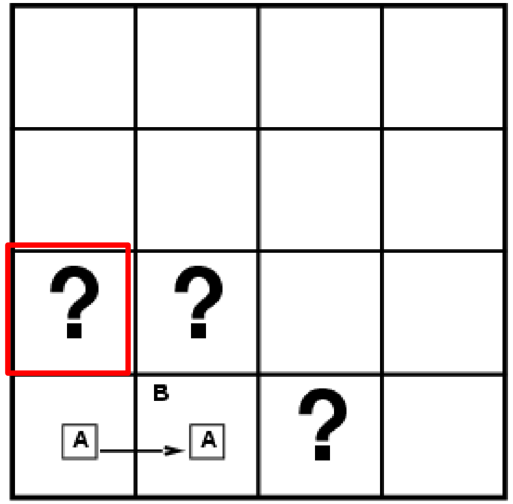
R_4 : $B_{1,1} \Leftrightarrow (P_{1,2} \vee P_{2,1})$

R_5 : $B_{2,1} \Leftrightarrow (P_{1,1} \vee P_{2,2} \vee P_{3,1})$

证明：KB ⊨ α

KB : $R_1 \wedge R_2 \wedge R_3 \wedge R_4 \wedge R_5 \wedge R_6 \wedge R_7$

α : $\neg P_{1,2}$



$R_6 : (B_{1,1} \Rightarrow (P_{1,2} \vee P_{2,1})) \wedge ((P_{1,2} \vee P_{2,1}) \Rightarrow B_{1,1})$

$R_7 : (P_{1,2} \vee P_{2,1}) \Rightarrow B_{1,1}$

逆否命题的逻辑等价 of R_7

$R_8 : \neg B_{1,1} \Rightarrow \neg (P_{1,2} \vee P_{2,1})$

Wumpus world sentences

R_1 ： □ $P_{1,1}$

R_2 ： □ $B_{1,1}$

R_3 ： $B_{2,1}$

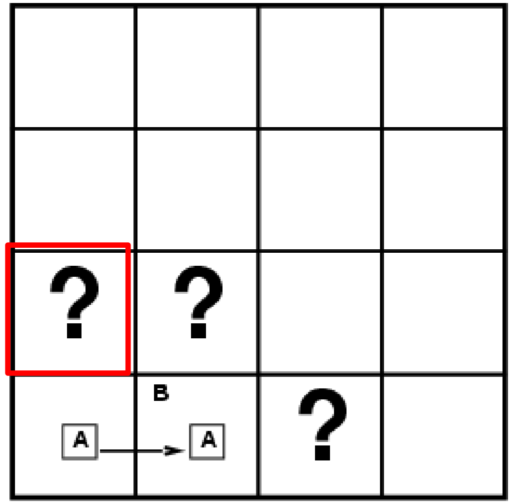
R_4 ： $B_{1,1} \Leftrightarrow (P_{1,2} \vee P_{2,1})$

R_5 ： $B_{2,1} \Leftrightarrow (P_{1,1} \vee P_{2,2} \vee P_{3,1})$

证明： KB $\models$ α

KB ： $\boldsymbol{R_1 \wedge R_2 \wedge R_3 \wedge R_4 \wedge R_5 \wedge R_6 \wedge R_7 \wedge R_8}$

α : $\neg \mathbf{P_{1,2}}$



$R_6 :(B_{1,1} \Rightarrow (P_{1,2} \vee P_{2,1})) \wedge ((P_{1,2} \vee P_{2,1}) \Rightarrow B_{1,1})$

$R_7 : (P_{1,2} \vee P_{2,1}) \Rightarrow B_{1,1}$

$R_8 : \neg B_{1,1} \Rightarrow \neg (P_{1,2} \vee P_{2,1})$

假言推理 with R_8 and R_2 (i.e., $\neg B_{1,1}$)

$\boldsymbol{R_9 : \neg(P_{1,2} \vee P_{2,1})}$

Wumpus world sentences

R_1 : $\neg P_{1,1}$

R_2 : $\neg B_{1,1}$

R_3 : $B_{2,1}$

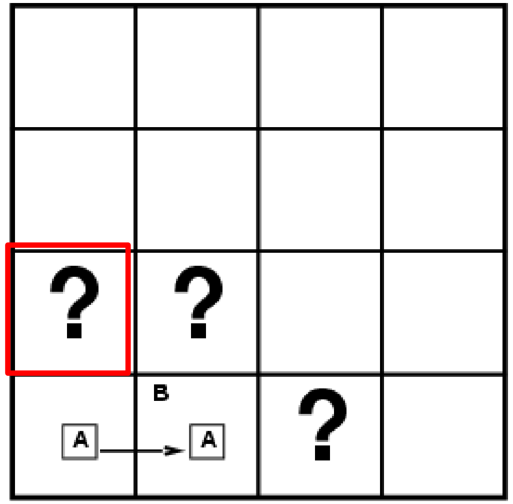
R_4 : $B_{1,1} \Leftrightarrow (P_{1,2} \vee P_{2,1})$

R_5 : $B_{2,1} \Leftrightarrow (P_{1,1} \vee P_{2,2} \vee P_{3,1})$

证明： KB $\models$ α

KB ： $R_1 \wedge R_2 \wedge R_3 \wedge R_4 \wedge R_5 \wedge R_6 \wedge R_7 \wedge R_8 \wedge R_9$

α ： $\neg P_{1,2}$



$R_6 :(B_{1,1} \Rightarrow (P_{1,2} \vee P_{2,1})) \wedge ((P_{1,2} \vee P_{2,1}) \Rightarrow B_{1,1})$

$R_7 : (P_{1,2} \vee P_{2,1}) \Rightarrow B_{1,1}$

$R_8 : \neg B_{1,1} \Rightarrow \neg (P_{1,2} \vee P_{2,1})$

$R_9 : \neg(P_{1,2} \vee P_{2,1})$

De Morgan 定律 to R_9

$R_{10} : \neg P_{1,2} \wedge \neg P_{2,1}$

消去合取词 to R_{10}

$R_{11} : \neg P_{1,2}$

结论： KB $\models$ α

上述过程是手工给出的，如何利用搜索算法来找出证明序列？

推理：判断逻辑蕴涵

- Method 2: Application of inference rules （推理规则）
  - 在知识库的语句上，直接应用推理规则，构建目标语句的证明
  - 找出证明序列的搜索算法：
    - 问题描述：
      - 初始状态：初始知识库
      - 行动：行动集合由应用于语句的所有推理规则组成，要匹配推理规则的上半部分
      - 结果：行动的结果是将推理规则的下半部分的语句实例加入知识库
      - 目标：包含要证明的语句的状态
    - 解序列：找出一条证明序列（动作序列），每个动作是在 * 语句上应用 * 规则

推理：判断逻辑蕴涵

- Method 1: ***model-checking*** （模型检验）
  - 枚举所有的模型(可能世界)，并验证语句在所有模型中为真
- Method 2: Application of inference rules （推理规则）
  - 在知识库的语句上，直接应用推理规则，构建目标语句的证明
- Method 3: ***theorem-proving*** （定理证明）
  - 归结证明
  - 当它和任何一个完备的搜索算法相结合时，可以得到完备的推理算法

Outline

- 7.1 基于知识的智能体
- 7.2 Wumpus world
- 7.3 知识的逻辑表示和推理
- 7.4 命题逻辑：一种简单的逻辑
- 7.5 命题逻辑定理证明
  - 归结原理

7.5.2 归结证明

- 归结 推理规则
  - 单元归结规则：选取一个子句（文字的析取式）和一个文字，生成一个新的子句

$$\frac{l_1 \vee \dots \vee l_k, \qquad m}{l_1 \vee \dots \vee l_{i-1} \vee l_{i+1} \vee \dots \vee l_k}$$

其中，每个 l 都是一个文字， l_i 和 m 是互补文字（一个是另一个的否定式）

E.g., $\dfrac{P_{1,1} \vee P_{3,1}, \qquad \neg P_{1,1}}{P_{3,1}}$

文字是指原子语句（正文字）或原子语句的否定式（负文字）

7.5.2 归结 Resolution

- 归结 推理规则
  - 全归结：选取两个子句（文字的析取式），生成一个新的子句（包含除了两个互补文字之外的原始子句中的所有文字）

$$\frac{l_1 \vee \dots \vee l_k, \qquad m_1 \vee \dots \vee m_n}{l_1 \vee \dots \vee l_{i-1} \vee l_{i+1} \vee \dots \vee l_k \vee m_1 \vee \dots \vee m_{j-1} \vee m_{j+1} \vee \dots \vee m_n}$$

其中 l_i 和 m_j 是互补文字（一个是另一个的否定式）

注意：结果子句中每个文字只出现一次，多余副本将被归并。

如， $A \vee B$ 与 $A \vee \neg B$ 归结，得到 $A \vee A$ ，简化为 A

7.5.2 Resolution 归结

Resolution inference rule 只应用于子句（文字的析取式），

那么对于所有的命题逻辑，如何实现完备推理呢？

1 ）语句转换为合取范式

2 ）归结算法

合取范式 CNF

以子句的合取式表达的语句被称为合取范式

Conjunctive Normal Form (CNF)

conjunction of disjunctions of literals clauses

E.g., **$(A \lor \neg B) \land (B \lor \neg C \lor \neg D)$**

<u>定理：命题逻辑的每个语句逻辑上都等价于某个子句的合取式</u>

语句转换成 **CNF** 的过程：

1. 消去等价词□
2. 消去蕴含词□
3. 否定词□内移
4. 使用分配率，将 □对 □进行分配

转换成 CNF

Conjunctive Normal Form (CNF)

conjunction of disjunctions of literals clauses

$B_{1,1} \Leftrightarrow (P_{1,2} \vee P_{2,1})$

1. 消除 □，用 $(\alpha \Rightarrow \beta) \wedge (\beta \Rightarrow \alpha)$ 代替 $\alpha \Leftrightarrow \beta$:

   $(B_{1,1} \Rightarrow (P_{1,2} \vee P_{2,1})) \wedge ((P_{1,2} \vee P_{2,1}) \Rightarrow B_{1,1})$

2. 消除 □，用 □ $\alpha \vee \beta$ 代替 $\alpha \Rightarrow \beta$:

   $(\neg B_{1,1} \vee P_{1,2} \vee P_{2,1}) \wedge (\neg(P_{1,2} \vee P_{2,1}) \vee B_{1,1})$

3. 将 □ 移进去，利用 de Morgan's 规则 and 双重否定：

   $(\neg B_{1,1} \vee P_{1,2} \vee P_{2,1}) \wedge ((\neg P_{1,2} \wedge \neg P_{2,1}) \vee B_{1,1})$

4. 使用分配率 ($\wedge$ over $\vee$) 和结合律：

$(\neg B_{1,1} \vee P_{1,2} \vee P_{2,1}) \wedge (\neg P_{1,2} \vee B_{1,1}) \wedge (\neg P_{2,1} \vee B_{1,1})$

归结算法 Resolution algorithm

- 证明 KB |= α
- 归结证明步骤：

  1 ）反证法，为了证明 KB |= α ，需要证明 KB $\wedge \neg$ α 是不可满足的

  2 ） $KB \wedge \neg \alpha$ 转化为合取范式

  3 ）利用归结规则证明 $KB \wedge \neg \alpha$ 不可满足

For example: KB |= α

$$KB = (B_{1,1} \Leftrightarrow (P_{1,2} \vee P_{2,1})) \wedge \neg B_{1,1}$$

$$\alpha = \neg P_{1,2}$$

1 ）反证法

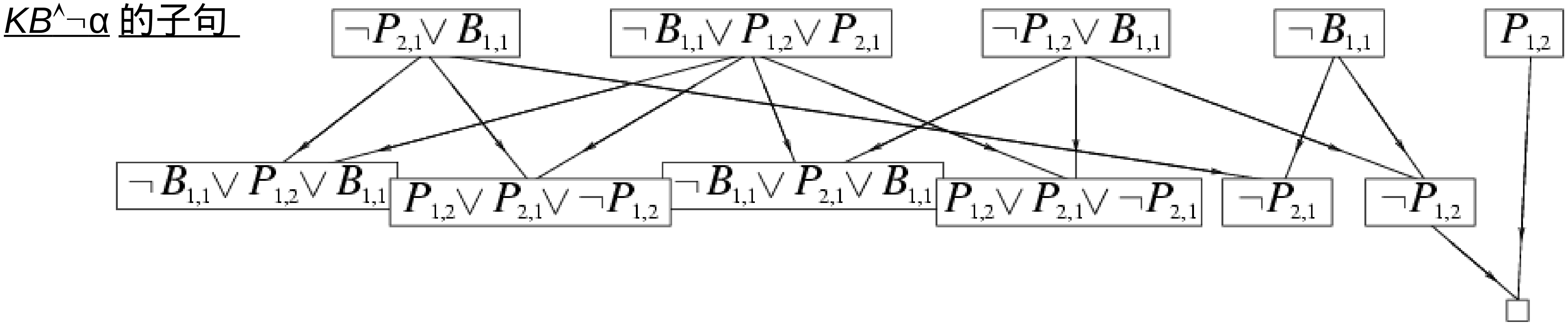
$$KB \wedge \neg \alpha \equiv (B_{1,1} \Leftrightarrow (P_{1,2} \vee P_{2,1})) \wedge \neg B_{1,1} \wedge P_{1,2}$$

Resolution example

1 ） $KB \wedge \neg\alpha \equiv (B_{1,1} \Leftrightarrow (P_{1,2} \vee P_{2,1})) \wedge \neg B_{1,1} \wedge P_{1,2}$

2 ）转为合取范式 $(\neg P_{2,1} \vee B_{1,1}) \wedge (\neg B_{1,1} \vee P_{1,2} \vee P_{2,1}) \wedge (\neg P_{1,2} \vee B_{1,1}) \wedge \neg B_{1,1} \wedge P_{1,2}$

3 ）两子句运用归结规则



If a set of clauses is unsatisfiable, then the resolution closure of those clauses contains the empty clause.

空子句不包含任何文字的子句，是永假的

归结算法

- 用反证法证明
- 为了证明 KB |= α ，需要证明 KB ^ ¬ α 是不可满足的

```
function PL-RESOLUTION(KB, α) returns true or false
   inputs: KB, the knowledge base, a sentence in propositional logic
           α, the query, a sentence in propositional logic

   clauses ← the set of clauses in the CNF representation of KB ∧ ¬α
   new ← { }
   loop do
      for each pair of clauses Ci, Cj in clauses do
         resolvents ← PL-RESOLVE(Ci, Cj)
         if resolvents contains the empty clause then return true
         new ← new ∪ resolvents
      if new ⊆ clauses then return false
      clauses ← clauses ∪ new
```

转化为 CNF

任意两个子句，运用归结规则

归结出空子句

没有新的语句

函数 PL-RESOLVE 返回对两个输入子句进行归结得到的所有结果子句的集合

练习题：合取范式与归结证明

1、将语句转换为合取范式（ CNF ）

7.20 Convert the following set of sentences to clausal form.

S1: $A \Leftrightarrow (B \vee E)$.
S2: $E \Rightarrow D$.
S3: $C \wedge F \Rightarrow \neg B$.
S4: $E \Rightarrow B$.
S5: $B \Rightarrow F$.
S6: $B \Rightarrow C$

2、归结证明

7.12 Use resolution to prove the sentence $\neg A \wedge \neg B$ from the clauses in Exercise 7.20.