命题逻辑

张文生

中国科学院自动化研究所
中科院大学人工智能学院

2020年11月13日

人类似乎是认识世界的；而且他们对世界的认识能够帮助他们做事。这些不是空洞的陈述。人类一直强调人的智能是如何获得的——不是靠反射机制而是对知识的内部**表示**进行操作的**推理**过程。在人工智能界，这种智能方法体现在**基于知识的 Agent** 上。

讲课内容

一、 命题逻辑

二、 定理证明

三、 模型检验

特殊的智能体

- 人类一直强调人的智能是如何获得——不是靠反射机制而是**对知识**的内部表示进行操作的**推理**过程；人工智能界，这种智能方法体现在**基于知识的Agent**上
- 基于知识的Agent能够以显式描述目标的形式接受新任务；通过被告知或者主动学习环境的新知识从而快速获得能力；通过更新相关知识适应环境的变化

- 基于知识的Agent的核心部件是知识库，或称KB
- 知识库是一个语句集合，这些语句表示关于世界的某些断言
- 公理：直接给定而不是推导得到
- TELL（告诉）和Ask（询问）：将新语句添加到知识库以及查询目前所知内容的方法

- 基于知识的Agent程序轮廓
  - 输入：感知信息
  - 输出：一个行动
  - Agent维护一个知识库*KB*，初始化时包括背景知识

```
function KB-AGENT(percept) returns an action
  persistent: KB, a knowledge base
              t, a counter, initially 0, indicating time

  TELL(KB, MAKE-PERCEPT-SENTENCE(percept, t))
  action ← ASK(KB, MAKE-ACTION-QUERY(t))
  TELL(KB, MAKE-ACTION-SENTENCE(action, t))
  t ← t + 1
  return action
```

构建语句断言在给定时刻感知到的感知信息

构建语句询问当前应该执行什么行动

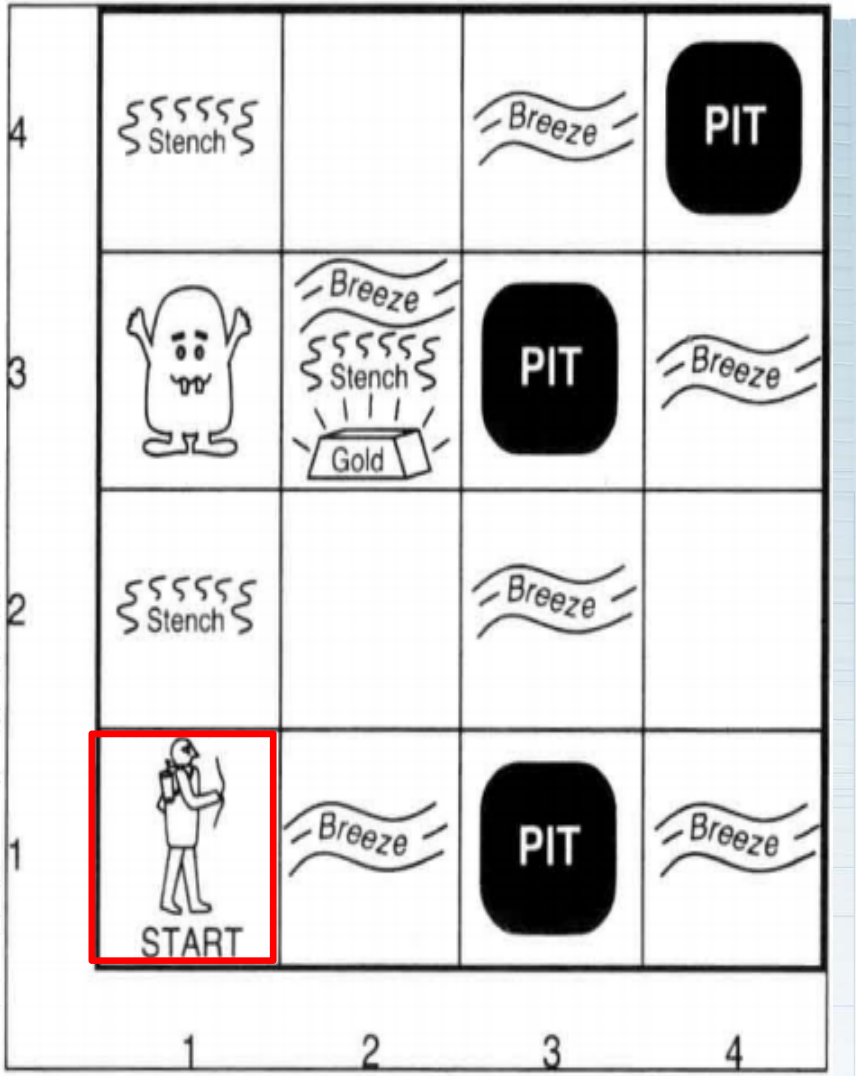
构建语句断言选择的行动已执行

- Agent要服从知识层的描述，在知识层只需描述Agent知道的内容和它的目标
- 例如：自动出租车的目标是将乘客从旧金山送到马林郡，它知道这两个地点通过金门大桥连接，那么可以期望它会穿过金门桥，因为它知道这可以完成目标，独立于实现层

连接旧金山与马林郡的金门大桥

Wumpus世界

- Wumpus世界由多个房间组成并相连接起来的山洞。在洞穴某处隐藏着一只Wumpus（怪兽），它会吃掉进入它房间的任何人。Agent可以射杀Wumpus，但是Agent只有一枚箭。某些房间是无底洞，任何人漫游到这些房间都会被无底洞吞噬。生活在该环境下的唯一希望是存在发现一堆金子的可能性



- 假设移动到标有OK的方格[2,1]，感知到微风，基于前述环境规则，判断[2,2]、[3,1]可能有无底洞，返回[1,1]，前进到[1,2]

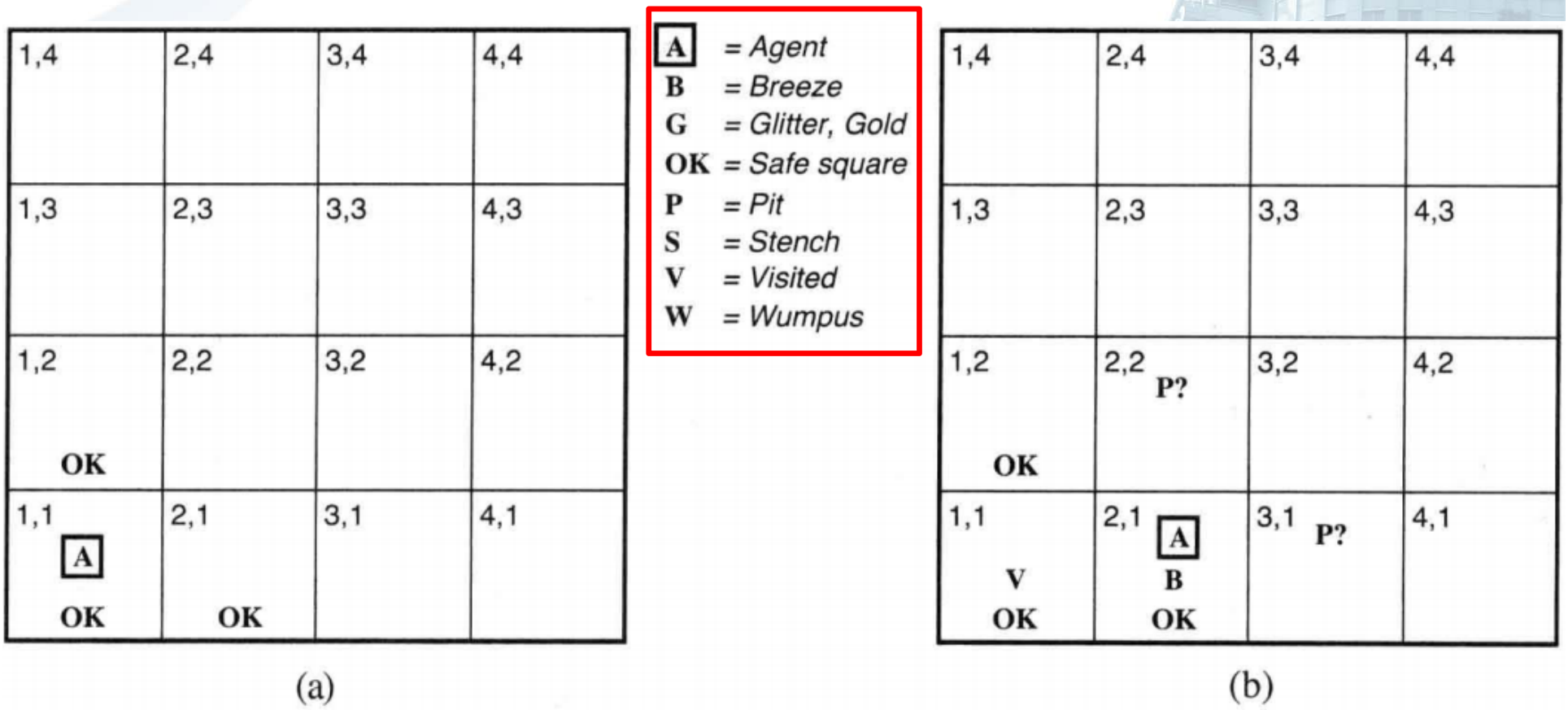


图 7.3　Wumpus 世界中 Agent 采取的第一步行动

- 在[1,2]中感知到臭气，根据前述环境规则及上一步推理结果，判断Wumpus在[1,3]，[2,2] 是安全的，[3,1]是无底洞

1,4	2,4	3,4	4,4
1,3 W!	2,3	3,3	4,3
1,2 A S OK	2,2 OK	3,2	4,2
1,1 V OK	2,1 B V OK	3,1 P!	4,1

(a)

A = *Agent*
B = *Breeze*
G = *Glitter, Gold*
OK = *Safe square*
P = *Pit*
S = *Stench*
V = *Visited*
W = *Wumpus*

1,4	2,4 P?	3,4	4,4
1,3 W!	2,3 A S G B	3,3 P?	4,3
1,2 S V OK	2,2 V OK	3,2	4,2
1,1 V OK	2,1 B V OK	3,1 P!	4,1

(b)

图 7.4　Agent 取得进展的两个后续阶段

- 逻辑推理的本质：在每种情况下，Agent根据可用信息得出结论，如果可用信息正确，那么该结论能确保是正确的
- 接下来描述如何建造可以表示必要信息并推理得出结论的逻辑Agent

逻辑的概念

- 知识库由语句构成。根据表示语言的语法来表达这些语句，语法为合法语句给出规范
- 逻辑定义语言的语义，语义定义了每个语句在每个可能世界的真值
  - 算术的语义规范了语句“$x+y=4$”，在x等于2，y也等于2的世界中为真；在x等于1，y等于1的世界中为假
- 标准逻辑中，每个语句在每个可能世界中非真即假——不存在“中间状态”

- 模型是数学抽象，每个模型只是简单地关注于每个相关语句的真或假
  - 例如可能世界中x个男性和y个女性正坐在桌子旁玩桥牌，当总共有四个人的时候，语句$x+y=4$为真。可能模型就是对变量x和y的所有可能赋值
- 语句α在模型m中为真，称m满足α，有时也称m是α的一个模型，使用表示$M(\alpha)$表示所有模型

- 语句间的逻辑蕴涵（entailment）关系——某语句逻辑上跟随另一个语句，数学符号表示为

$$\alpha \models \beta$$

  - 意为语句α蕴涵语句β。蕴涵的形式化定义：当且仅当在使α为真的每个模型中，β也为真。可以记为

$$\alpha \models \beta \text{当且仅当} M(\alpha) \subseteq M(\beta)$$

    - α比β更强的断言：排除了更多的可能世界

- 蕴涵关系与算术类似：语句x=0蕴涵了语句xy=0。显然，在任何x=0的模型中，xy的值都是0，不管y的值是多少
- 蕴涵就像是干草堆里的一根针；推理就像寻找它的过程，形式表示：如果推理算法i根据KB导出α，记为 $KB|=_i \alpha$
  - 读为“α通过i从KB导出”或者“i从KB导出α”

逻辑

- 原子语句由单个命题词组成。每个命题词代表一个或为真或为假的命题，采用大写字母表示
  - 可能包括其他符号或下标：P、Q、R、$W_{1,3}$和$North$等
  - 有两个命题词有固定含义：True是永真命题，False为永假命题
- 复合句由简单语句用括号和逻辑连接词构成

语法

- 常用的五种逻辑连接词：
  - ¬（非），如 $\neg W_{1,3}$被称为$W_{1,3}$的否定式。文字指原子语句（正文字）或原子语句的否定式（负文字）
  - $\wedge$（与），如$W_{1,3} \wedge P_{1,3}$，被称为合取式，各部分称为合取子句
  - $\vee$（或），如（$W_{1,3} \wedge P_{1,3}$）$\vee W_{2,2}$，是析取子句$W_{1,3} \wedge P_{1,3}$和$W_{2,2}$的析取式
  - =>（蕴含） $W_{1,3} \wedge P_{3,1} \Rightarrow \neg W_{2,2}$的语句称为蕴含式（implication）（或条件式）

- =>（蕴含）它的前提或称前项是$W_{1,3} \wedge P_{3,1}$，结论或称后项为$\neg W_{2,2}$。蕴含式同时也称为规则或if-then语句，有些书中记为⊃或→
- <=>（当且仅当）：语句$W_{1,3}$ <=> $\neg W_{2,2}$是双向蕴含式，有些书记为≡
- 运算符优先级：“否定”运算符¬具有最高优先级
  - $\neg A \wedge B$等价于$(\neg A) \wedge B$而不是$\neg (A \wedge B)$

语义

- 命题逻辑固定了每个命题词的真值——*true*或*false*
  - 例如，知识库中的语句采用命题词$P_{1,2}$、$P_{2,2}$和$P_{1,3}$，那么一个模型可能是

$$m_1=\{P_{1,2}=false, P_{2,2}=false, P_{1,3}=false\}$$

- 命题逻辑的语义需要规范在已知模型下如何计算任一语句的真值，通过递归来实现
  - 所有语句都是由原子语句和5种连接词构成

- 复合句有5条规则，其中P和Q为任意子句，m为任一模型，iff 表示当且仅当
  - 在模型m中$\neg P$为真 iff P在m中为假
  - 在模型m中$P \wedge Q$为真 iff P和Q在m中都为真
  - 在模型m中$P \vee Q$为真 iff P或Q在m中为真
  - 在模型m中$P => Q$为真 除非 P在m中为真且Q在m中为假
  - 在模型m中$P <=> Q$为真 iff P和Q在m中都为真或者都为假

知识库

- 基于命题逻辑的语义定义，为Wumpus世界构建一个知识库，对于每个位置$[x,y]$需要如下命题词
  - 如果$[x,y]$中有无底洞，则$P_{x,y}$为真
  - 如果$[x,y]$中有怪兽，则$W_{x,y}$为真，不管是死是活
  - 如果在$[x,y]$中感知到微风，则$B_{x,y}$为真
  - 如果在$[x,y]$中感知到臭气，则$S_{x,y}$为真

推理过程

- 目标是判断对于某些语句α，$KB \models \alpha$是否成立
  - 例如，$\neg P_{1,2}$是否可从现有KB得出
- 推理算法模型检验是对蕴涵定义的直接实现：枚举所有模型，验证α在KB为真的每个模型中为真
- Wumpus世界中相关命题词有7个，其中的三个模型，KB为真

$B_{1,1}$	$B_{2,1}$	$P_{1,1}$	$P_{1,2}$	$P_{2,1}$	$P_{2,2}$	$P_{3,1}$	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

- 判定命题逻辑的蕴涵的一个通用算法
- 该算法是可靠的，因为直接实现了蕴涵的定义
- 完备的，因为可以用于任意KB和α，而且总能够终止

完成对变量赋值有限空间的递归枚举

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

  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 // when KB is false, always return true
  else do
    P ← FIRST(symbols)
    rest ← REST(symbols)
    return (TT-CHECK-ALL(KB,α, rest,model ∪ {P = true})
            and
            TT-CHECK-ALL(KB,α, rest, model ∪ {P = false }))
```

讲课内容

一、 命题逻辑

二、 定理证明

三、 模型检验

命题逻辑定理证明

- 通过**模型检验**判断蕴涵：枚举所有模型，并验证语句在所有模型中为真
- 如何通过**定理证明**判断蕴涵——在知识库的语句上直接应用推理规则以构建目标语句的证明，而无须关注模型
- 如果模型数目庞大而证明很短，那么定理证明就比模型检验更有效

- 逻辑等价：如果两语句α和β在同样的模型集中为真，则它们是逻辑等价的，记为$\alpha \equiv \beta$
  - 例如，用真值表很容易证明$P \wedge Q$和$P \wedge Q$逻辑等价
- 任意两语句α和β是等价的当且仅当它们互相蕴涵时，$\alpha \equiv \beta$ 当且仅当 $\alpha \models \beta$且$\beta \models \alpha$

$$
\begin{aligned}
(\alpha \wedge \beta) &\equiv (\beta \wedge \alpha) && \text{commutativity of } \wedge \\
(\alpha \vee \beta) &\equiv (\beta \vee \alpha) && \text{commutativity of } \vee \\
((\alpha \wedge \beta) \wedge \gamma) &\equiv (\alpha \wedge (\beta \wedge \gamma)) && \text{associativity of } \wedge \\
((\alpha \vee \beta) \vee \gamma) &\equiv (\alpha \vee (\beta \vee \gamma)) && \text{associativity of } \vee \\
\neg(\neg\alpha) &\equiv \alpha && \text{double-negation elimination} \\
(\alpha \Rightarrow \beta) &\equiv (\neg\beta \Rightarrow \neg\alpha) && \text{contraposition} \\
(\alpha \Rightarrow \beta) &\equiv (\neg\alpha \vee \beta) && \text{implication elimination} \\
(\alpha \Leftrightarrow \beta) &\equiv ((\alpha \Rightarrow \beta) \wedge (\beta \Rightarrow \alpha)) && \text{biconditional elimination} \\
\neg(\alpha \wedge \beta) &\equiv (\neg\alpha \vee \neg\beta) && \text{De Morgan} \\
\neg(\alpha \vee \beta) &\equiv (\neg\alpha \wedge \neg\beta) && \text{De Morgan} \\
(\alpha \wedge (\beta \vee \gamma)) &\equiv ((\alpha \wedge \beta) \vee (\alpha \wedge \gamma)) && \text{distributivity of } \wedge \text{ over } \vee \\
(\alpha \vee (\beta \wedge \gamma)) &\equiv ((\alpha \vee \beta) \wedge (\alpha \vee \gamma)) && \text{distributivity of } \vee \text{ over } \wedge
\end{aligned}
$$

- 有效性：一个语句是有效的，如果在所有模型中它都为真
  - 例如，语句$P \vee \neg P$为有效的
- 有效语句也被称为重言式，必定为真
- 演绎定理
  - 任意语句α和β，$\alpha \models \beta$当且仅当语句（$\alpha \Rightarrow \beta$）是有效的
  - 说明每个有效的蕴含语句都描述了一个合法的推理
- 可满足性：如果一个语句在某些模型中为真，那么这个句子是可满足的

推导和证明

- 应用推理规则得到一个证明——一系列结论直到目标语句，最著名的规则是假言推理规则，记为

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

  - 只要给定任何形式为$\alpha=>\beta$和α的语句，就可以推导出语句β
  - 例如，如果已知($WumpusAhead \wedge WumpusAlive$)$=>Shoot$和($WumpusAhead \wedge WumpusAlive$)，那么可

- 所有逻辑等价都可以作为推理规则
  - 例：用于双向蕴含消去的等价给出两条推理规则

$$\frac{\alpha \Leftrightarrow \beta}{(\alpha \Rightarrow \beta) \wedge (\beta \Rightarrow \alpha)} \quad \text{以及} \quad \frac{(\alpha \Rightarrow \beta) \wedge (\beta \Rightarrow \alpha)}{\alpha \Leftrightarrow \beta}$$

- 不是所有的推理规则都是两个方向都生效
  - 例：无法按相反方向运用假言推理规则，无法从β得到

    $\alpha => \beta$和α

例子：Wumpus世界

- 从包含R_1到R_5的知识库，如何证明$\neg P_{1,2}$，即证明[1,2]中没有无底洞

已知R_1到R_5：

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

R_2: $B_{1,1} \Leftrightarrow (P_{1,2} \vee P_{2,1})$

R_3: $B_{2,1} \Leftrightarrow (P_{1,1} \vee P_{2,2} \vee P_{3,1})$

R_4: $\neg B_{1,1}$

R_5: $B_{2,1}$

将双向蕴含消去应用于R_2，得到：

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

接着对R_6消去合取词得到：

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

其逆否命题的逻辑等价给出：

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

对R_8和感知信息R_4
运用假言推理规则，得到：

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

最后，应用De Morgan定律，给出结论：

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

即[1,2]和[2,1]都不包含无底洞

搜索证明序列

- 上述证明过程也可以采用搜索算法来找出证明序列，证明问题可以的定义为：
  - 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：包含要证明语句的状态

- **单调性**：意味着逻辑蕴涵语句集会随着添加到知识库的信息增长而增长，对于任意语句α和β
  - 如果KB$\models \alpha$，那么KB$\wedge \beta \models \alpha$
  - 例如，假设知识库包含附加断言β， β宣称世界中正好有8个无底洞，这条知识可能有助于Agent推导出附加结论，但无法推翻任意已经推导出的结论α
- 单调性意味着只要在知识库中发现了合适的前提，就可以应用推理规则——规则的结论“与知识库中的其余内容无关”

归结证明

- **归结**，一个推理规则，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
- 以Wumpus世界为例，讲解归结规则的一个简单版本，考虑导出下图的步骤：

1,4	2,4	3,4	4,4
1,3 W!	2,3	3,3	4,3
1,2 A S OK	2,2 OK	3,2	4,2
1,1 V OK	2,1 B V OK	3,1 P!	4,1

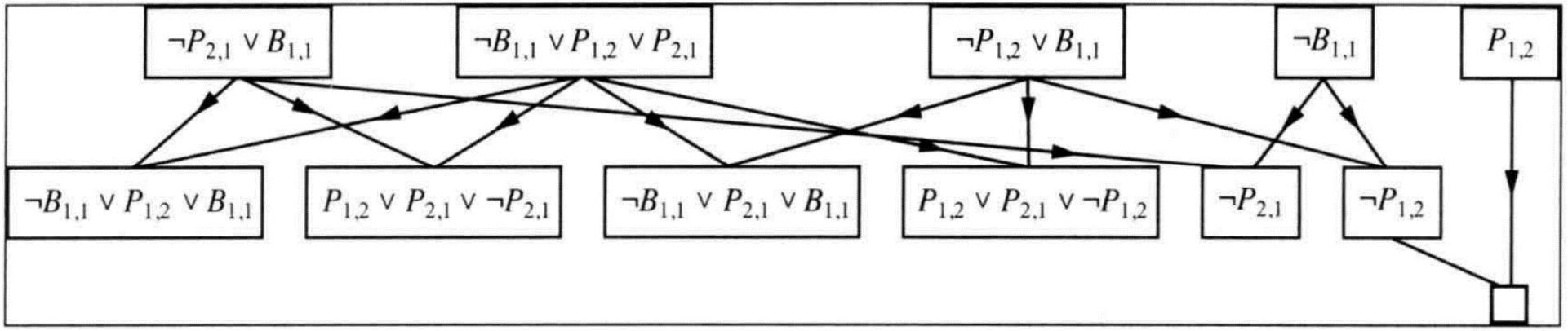
- ✓ Agent从[2,1]返回[1,1]，接着走到[1,2]
- ✓ 在此地感知到臭气，但没有微风

- 基于归结的推理过程使用的是反证法证明原理
  - 例：若要证明$KB|=\alpha$，则证$KB/\backslash\neg\alpha$是不可满足的

返回对两个输入子句进行归结得到的所有结果子句集合

```
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 C_i, C_j in clauses do
          resolvents ← PL-RESOLVE(C_i, C_j)
          if resolvents contains the empty clause then return true
          new ← new ∪ resolvents
      if new ⊆ clauses then return false
      clauses ← clauses ∪ new
```



- 第二行给出了对第一行进行归结得到的所有子句
- 当$P_{1,2}$与$\neg P_{1,2}$进行归结时得到空子句，用小方框表示
- 注意：可以删除同时出现两个互补文字的任何子句

归结的完备性

- 子句集S的**归结闭包**$RC(S)$，通过对S中的子句或其派生子句反复应用归结规则而生成的所有子句的集合
- PL-RESOLUTION计算的归结闭包是变量*clauses*的值
  - $RC(S)$一定是有限的，因为S中出现的符号P_1，…，P_k只能构成有限多个不同的子句，所以PL-RESOLUTION可终止

- 命题逻辑中归结的完备性定理被称为**基本归结定理**

  —如果子句集是不可满足的，那么这些子句的归结闭包包含空子句

- 证明其逆否命题，如果闭包$RC(S)$不包含空子句，那么S是可满足的。可以用P_1，…，P_k的适当真值构造S的模型，构造过程如下：

i从1到k重复执行以下操作：

- 如果$RC(S)$中有包含文字$\neg P_i$的子句，该子句所有的其他文字在P_1，…，P_{i-1}选择的赋值下都为假，那么对P_i赋值*false*。
- 否则，对P_i赋值*true*。

Horn子句和限定子句

- **限定子句**是指恰好只含一个正文字的析取式
  - 例如，子句（$\neg L_{1,1} \vee \neg \text{Breeze} \vee B_{1,1}$）是限定子句，而($\neg B_{1,1} \vee P_{1,2} \vee P_{2,1}$)不是
- 更一般的形式有**Horn子句**，是指至多只有一个正文字的析取式
  - 所有限定子句都是Horn子句
  - 没有正文字的析取式也是Horn子句：称为目标子句

Horn子句在归结下是封闭的，如果对两个Horn子句

前向和反向链接

```
function PL-FC-ENTAILS?(KB, q) returns true or false
  inputs: KB, the knowledge base, a set of propositional definite clauses
          q, the query, a proposition symbol
  count ← a table, where count[c] is the number of symbols in c's premise
  inferred ← a table, where inferred[s] is initially false for all symbols
  agenda ← a queue of symbols, initially symbols known to be true in KB

  while agenda is not empty do
    p ← POP(agenda)
    if p = q then return true
    if inferred[p] = false then
      inferred[p] ← true
      for each clause c in KB where p is in c.PREMISE do
        decrement count[c]
        if count[c] = 0 then add c.CONCLUSION to agenda
  return false
```

运行时间是线性的

图 7.15　命题逻辑的前向链接算法

讲课内容

一、命题逻辑

二、定理证明

三、模型检验

有效的命题逻辑模型检验

- 基于模型检验的命题推理有两类有效算法
  - 回溯搜索
  - 爬山法

完备的回溯算法

- Davis-Putnam算法，以MartinDavis和HilaryPutnam (1960)的开创性论文命名。实际上，该算法是Davis、Logemann和Loveland(1962)描述的版本，因此按所有四个作者的首字母缩写将其命名为DPLL
- 本质上是可能模型的递归深度优先枚举算法，相对于TT-ENTAILS的简单方法，改进如下：
  - 及早终止：可以用部分完成的模型来判断语句是否一定为真或为假，避免了搜索空间的全部子树
    - 如果A为真，那么语句$(A \vee B) \wedge (A \vee C)$为真，无论$B$和$C$取何值

- DPLL算法基本框架

```
function DPLL-SATISFIABLE?(s) returns true or false
  inputs: s, a sentence in propositional logic

  clauses ← the set of clauses in the CNF representation of s
  symbols ← a list of the proposition symbols in s
  return DPLL(clauses, symbols, { })

---

function DPLL(clauses, symbols, model) returns true or false

  if every clause in clauses is true in model then return true
  if some clause in clauses is false in model then return false
  P, value ← FIND-PURE-SYMBOL(symbols, clauses, model)
  if P is non-null then return DPLL(clauses, symbols – P, model ∪ {P=value})
  P, value ← FIND-UNIT-CLAUSE(clauses, model)
  if P is non-null then return DPLL(clauses, symbols – P, model ∪ {P=value})
  P ← FIRST(symbols); rest ← REST(symbols)
  return DPLL(clauses, rest, model ∪ {P=true}) or
         DPLL(clauses, rest, model ∪ {P=false}))
```

图 7.17　检验命题逻辑语句可满足性的 DPLL 算法

局部搜索算法

- 目标是找出满足每个子句的变量赋值，评价函数可以选择未满足子句的数量。算法涉及完全赋值空间，该空间包括很多局部极小值点
  - 试图找出贪婪性和随机性之间的平衡点
- WALKSAT是最简洁有效的算法
  - 每次迭代中选择一个未得到满足的子句
  - 从该子句中选择一个命题符号进行翻转操作
    - “最小冲突”，最小化新状态下未得到满足语句数量
    - “随机游走”来随机挑选符号

- WALKSAT算法框架：

```
function WALKSAT(clauses, p, max_flips) returns a satisfying model or failure
  inputs: clauses, a set of clauses in propositional logic
          p, the probability of choosing to do a "random walk" move, typically around 0.5
          max_flips, number of flips allowed before giving up

  model ← a random assignment of true/false to the symbols in clauses
  for i = 1 to max_flips do
      if model satisfies clauses then return model
      clause ← a randomly selected clause from clauses that is false in model
      with probability p flip the value in model of a randomly selected symbol from clause
      else flip whichever symbol in clause maximizes the number of satisfied clauses
  return failure
```

图 7.18　通过随机翻转变量的值检验可满足性的 WALKSAT 算法。算法存在多种版本

中国科学院
自动化研究所
INSTITUTE OF AUTOMATION
CHINESE ACADEMY OF SCIENCES

感谢同学们听课

欢迎讨论与交流