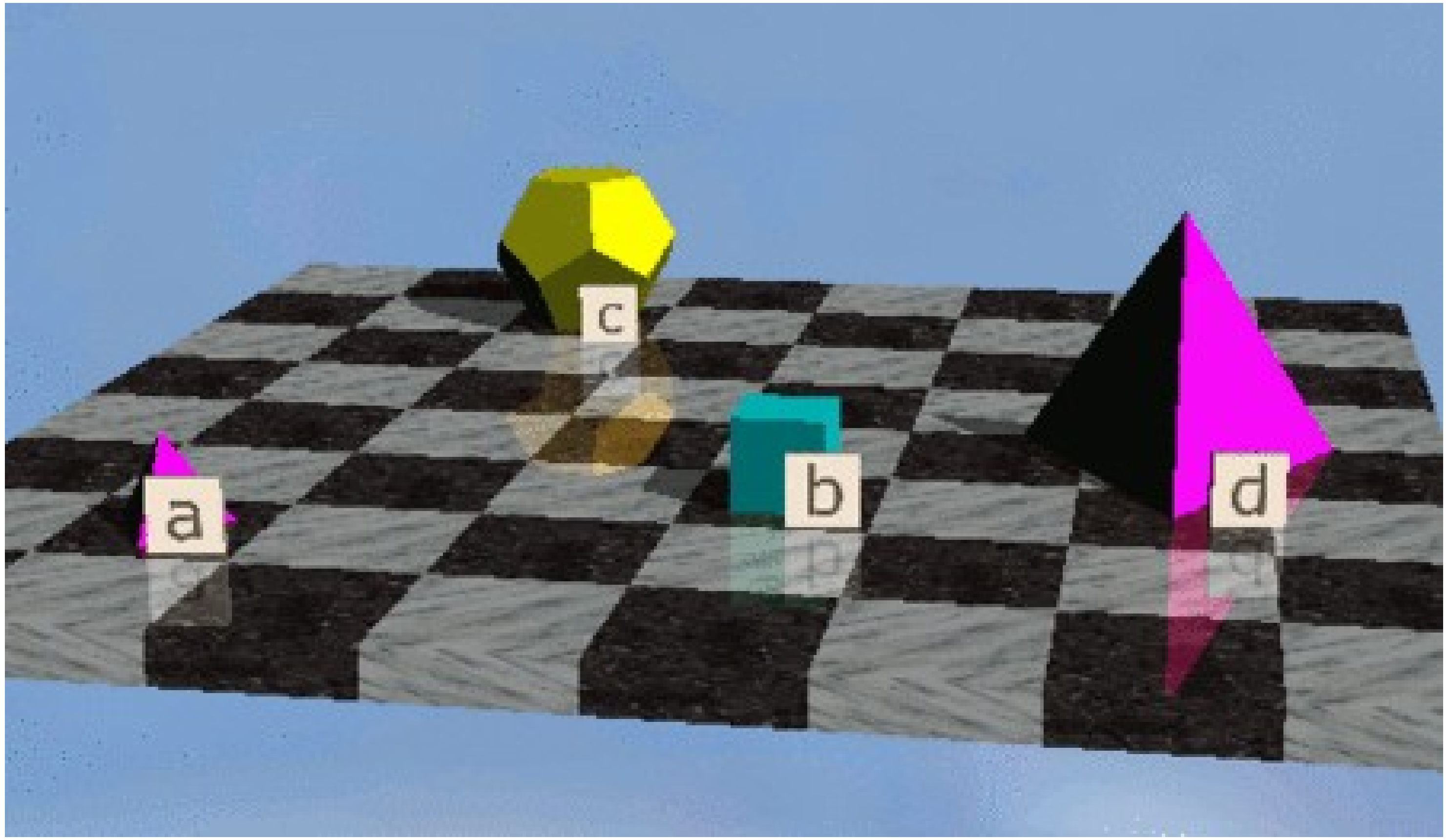
Artificial Intelligence

第 8 章 一阶逻辑推理



表示能力

- 命题逻辑不考虑命题之间的内在联系和数量关系。
- 一阶（谓词）逻辑可通过个体词、谓词和量词、以及它们之间的逻辑关系，反映内在联系。
  - 例如， Rules of chess:
    - 100,000 pages in propositional logic
    - 1 page in first-order logic
  - Rules of Wumpus World:
    - $\forall r\ Pit(r) \Rightarrow [\forall s\ Adjacent(r,s) \Rightarrow Breezy(s)]$

$$R_4 : B_{1,1} \Leftrightarrow (P_{1,2} \vee P_{2,1})$$
$$R_5 : B_{2,1} \Leftrightarrow (P_{1,1} \vee P_{2,2} \vee P_{3,1})$$

提纲

一阶逻辑基本概念

一阶逻辑化为子句

基于归结原理的推理算法

一阶逻辑中的基本概念

FOL的语法

语句 *Sentence* → *AtomicSentence*
| (*Sentence Connective Sentence*)
| *Quantifier Variable,... Sentence*
| ¬ *Sentence*

原子语句 *AtomicSentence* → *Predicate*(*Term*,...) | *Term* = *Term*

对象 *Term* → *Function*(*Term*,...)
| *Constant*
| *Variable*

FOL 的语法：基本元素

连接符 *Connective* → $\Rightarrow \mid \wedge \mid \vee \mid \Leftrightarrow$

量词 *Quantifier* → $\forall \mid \exists$

常量 *Constant* → $A \mid X_1 \mid John \mid \ldots$

变量 *Variable* → $a \mid x \mid s \mid \ldots$

谓词 *Predicate* → *Before* | *HasColor* | *Raining* | ...

函数 *Function* → *Mother* | *LeftLeg* | ...

谓词和函数的区别：

谓词： *HasFather*(x,y)

函数： *Father*(x)

知识的一阶逻辑表达方法

怪兽世界的知识库

$\forall x,y,a,b\ Adjacent([x,y],[a,b]) \Leftrightarrow Belong([a,b], \{[x+1,y], [x-1,y],[x,y+1],[x,y-1]\})$

在无底黑洞旁边的方块有风：

− Diagnostic 规则—由现象推测原因

$\forall s\ Breezy(s) \Rightarrow \exists\ r\ Adjacent(r,s) \wedge Pit(r)$

Causal 规则—由原因推测现象

$\forall r\ Pit(r) \Rightarrow [\forall s\ Adjacent(r,s) \Rightarrow Breezy(s)]$

提纲

一阶逻辑基本概念

一阶逻辑化为子句

基于归结原理的推理算法

一阶逻辑化为子句

- 子句
  - 谓词逻辑中，把原子公式及原子公式的否定统称为文字
  - 任何文字的析取式称为子句
    - $P \vee Q$、 $\square P(x,f(x),y) \vee Q(y) \vee R(f(x))$ 都是子句
  - 不包含任何文字的子句称为空子句
    - 空子句不能被任何解释满足，所以空子句是永假的，不可满足的
  - 一阶逻辑中，任何一个一阶逻辑公式都可以化成一个子句集

一阶逻辑化为子句

将一阶逻辑公式化为子句集的步骤：

(1) 消蕴含和等价

(2) 否定内移

(3) 变量标准化

(4) 消去存在量词

(5) 将公式化为前束形

(6) 化为合取范式

(7) 略去全称量词

(8) 消去合取符号∧

(9) 子句变量标准化

如果存在量词不在任何一个全称量词的辖域内，则该存在量词不依赖于任何其它的变量，因此可用一个新的个体常量代替 如 $\exists x\ P(x)$ 化为 $P(A)$

如果存在量词是在全称量词的辖域内

如：$\forall y(\exists x\ P(x,y))$，x 的取值依赖于 y 的取值

由 **Skolem** 函数 **f(y)** 表示依赖关系化为 $\forall y(P(f(y), y))$

一阶逻辑化为子句

【例】化成子句集

$(\forall x)\{[\neg P(x)\vee\neg Q(x)]\Rightarrow(\exists y)[S(x,y)\wedge Q(x)]\}\wedge(\forall x)[P(x)\vee B(x)]$

转换过程遵照上述 9 个步骤：

(1) 消蕴含“□”： $(\forall x)\{\neg[\neg P(x)\vee\neg Q(x)]\vee(\exists y)[S(x,y)\wedge Q(x)]\}\wedge(\forall x)[P(x)\vee B(x)]$

(2) 否定内移：

$(\forall x)\{[P(x)\wedge Q(x)]\vee(\exists y)[S(x,y)\wedge Q(x)]\}\wedge(\forall x)[P(x)\vee B(x)]$

一阶逻辑化为子句

(1) 消蕴含“”： $(\forall x)\{\neg[\neg P(x)\vee\neg Q(x)]\vee(\exists y)[S(x,y)\wedge Q(x)]\}\wedge(\forall x)[P(x)\vee B(x)]$

(2) 否定内移：

$(\forall x)\{[P(x)\wedge Q(x)]\vee(\exists y)[S(x,y)\wedge Q(x)]\}\wedge(\forall x)[P(x)\vee B(x)]$

(3) 变量标准化：

$(\forall x)\{[P(x)\wedge Q(x)]\vee(\exists y)[S(x,y)\wedge Q(x)]\}\wedge(\forall w)[P(w)\vee B(w)]$

(4) 消去存在量词：

$(\forall x)\{[P(x)\wedge Q(x)]\vee[S(x,\mathbf{f(x)})\wedge Q(x)]\}\wedge(\forall w)[P(w)\vee B(w)]$

一阶逻辑化为子句

(5) 将公式化为前束形

$(\forall x)(\forall w)\{[P(x)\wedge Q(x)]\vee[S(x,f(x))\wedge Q(x)]\}\wedge[P(w)\vee B(w)]$

(6) 化为合取范式：

$(\forall x)(\forall w)\{[P(x)\vee S(x，f(x))]\wedge Q(x)\wedge[P(w)\vee B(w)]\}$

(7) 略去全称量词：

$[P(x)\vee S(x，f(x))]\wedge Q(x)\wedge[P(w)\vee B(w)]$

(8) 消去合取符号**∧：**

子句集为：$P（x）\vee S(x，f(x)）$；$Q(x)$；$P（w）\vee B（w）$

(9) 子句变量标准化：

最终的子句集为：$P（x）\vee S（x，f(x)）$；$Q（x）$；

五、归结原理

归结原理

归结原理

归结原理又称为消解原理，它是定理证明基础

命题逻辑归结

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

其中 l_i 和 m_j 是互补文字（一个是另一个的否定式）

归结原理

- 一阶逻辑归结

在一阶逻辑中，子句中含有变量

为将归结原理应用于含有变量的子句，应找出一个置换，作用于给定的两个子句，使它们包括互补的文字，然后才能进行归结

例 1: 子句集 S={P(x)∨Q(x)，□ P(A)∨R(y)}，两个子句不能直接归结，但若对子句集置换 s={A/x}，则两个子句分别为 P(A)∨Q(A) 和 □ P(A)∨R(y)，归结为：Q(A)∨R(y)

置换和合一

为了使用推理规则，如假言推理、假言三段论等，一个推理系统必须决定两个表达式是否相同或匹配：

两个表达式匹配当且仅当其语法是等价的

一个表达式的项可以是常量、变量或函数，合一就是寻找项对变量的置换而使表达式一致的过程，合一是一阶推理算法的关键

例：公式 P(x,f(y),B) 与 P(x,f(B),B），用常量 B 代替变量 y ，从而使两个公式一致。

称为：通过置换 {B/y} 就可使上述公式集合一

置换和合一

置换用有序对的集合 $s=\{t_1/v_1, t_2/v_2, \ldots, t_n/v_n\}$ 表示。

其中 t_i/v_i 表示用项 t_i 置换变量 v_i ，t_i 可以是变量、常量或函数

例：表达式 P(x,f(y),A) 通过不同的置换

- 变量 / 变量：置换为 $s_1=\{z/x, w/y\}$, P(z,f(w),A)
- 常量 / 变量：置换为 $s_2=\{B/y\}$, P(x,f(B),A)
- 函数 / 变量：置换为 $s_3=\{g(z)/x, B/y\}$, P(g(z),f(B),A)

置换和合一

```
function UNIFY(x, y, θ) returns a substitution to make x and y identical
   inputs: x, a variable, constant, list, or compound expression
           y, a variable, constant, list, or compound expression
           θ, the substitution built up so far (optional, defaults to empty)

   if θ = failure then return failure
   else if x = y then return θ
   else if VARIABLE?(x) then return UNIFY-VAR(x, y, θ)
   else if VARIABLE?(y) then return UNIFY-VAR(y, x, θ)
   else if COMPOUND?(x) and COMPOUND?(y) then
       return UNIFY(x.ARGS, y.ARGS, UNIFY(x.OP, y.OP, θ))
   else if LIST?(x) and LIST?(y) then
       return UNIFY(x.REST, y.REST, UNIFY(x.FIRST, y.FIRST, θ))
   else return failure
```

变量

复合语句

列表

```
function UNIFY-VAR(var, x, θ) returns a substitution

   if {var/val} ∈ θ then return UNIFY(val, x, θ)
   else if {x/val} ∈ θ then return UNIFY(var, val, θ)
   else if OCCUR-CHECK?(var, x) then return failure
   else return add {var/x} to θ
```

图 9.1 合一算法

该算法以元素为单位对输入的结构进行比较。置换 θ 作为 UNIFY 的参数是在运算的过程中逐渐建立起来，用于确保此后的比较与先前建立的约束一致。在复合表达式如 $F(A, B)$中，函数 OP 提取函词 F，函数 ARGS 提取参数表(A, B)

合一算法 UNIFY 以两条语句 x ，y 为输入，如果合一置换存在，则返回它们的合一置换 θ

P(u ，f(v) ，A)

P（ z ，f （ w ），A ）

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

命题逻辑归结

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

  $clauses \leftarrow$ the set of clauses in the CNF representation of $KB \wedge \neg\alpha$
  $new \leftarrow \{\}$
  loop do
    for each pair of clauses C_i, C_j **in** $clauses$ **do**
      $resolvents \leftarrow$ PL-RESOLVE(C_i, C_j)
      if $resolvents$ contains the empty clause **then return** *true*
      $new \leftarrow new \cup resolvents$
    if $new \subseteq clauses$ **then return** *false*
    $clauses \leftarrow clauses \cup new$

转化为
CNF

置换合一

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

归结原理

例 2: 设有两个子句 P(A)∨¬Q(x)∨R(x) 和□ P(y)∨Q(B) ， 则有两种归结方法：

① 取置换为 s={A/y} ，归结为□ Q(x)∨R(x)∨Q(B)

② 取置换为 s={B/x} ，归结为 P(A)∨R(B)∨¬P(y)

P(A) ∨¬ Q(x)∨R(x)　　¬ P(y) ∨Q(B)

s={A/ y }

¬ Q(x)∨R(x) ∨Q(B)

P(A) ∨¬ Q(x)∨R(x)　　¬ P(y) ∨Q(B)

s={B/ x }

P(A) ∨R(B) ∨¬ P(y)

- 注意：在求归结式时，不能同时消去两个互补文字对，消去两个互补文字对

六、归结原理的应用

归结原理的应用

- <u>应用归结原理进行定理证明</u>

设要被证明的定理可用谓词公式表示为如下的形式：

$$A_1 \land A_2 \land \ldots \land A_n \Rightarrow B$$

(1) 否定结论 B ，并将□ B 与前提公式集组成谓词公式：

$$G = A_1 \land A_2 \land \ldots \land A_n \land \neg B$$

(2) 求谓词公式 G 的子句集 S

(3) 应用归结原理，证明子句集 S 的不可满足性

归结原理的应用

例 3:

已知前提为 F:$(\forall x)\{(\exists y)[P(x,y)\wedge Q(y)]\rightarrow(\exists u)[R(u)\wedge S(x,u)]\}$

求证结论 G:$\neg(\exists x)R(x)\rightarrow(\forall x)(\forall y)[P(x,y)\rightarrow\neg Q(y)]$

- 证明：1）前提和结论化为子句集

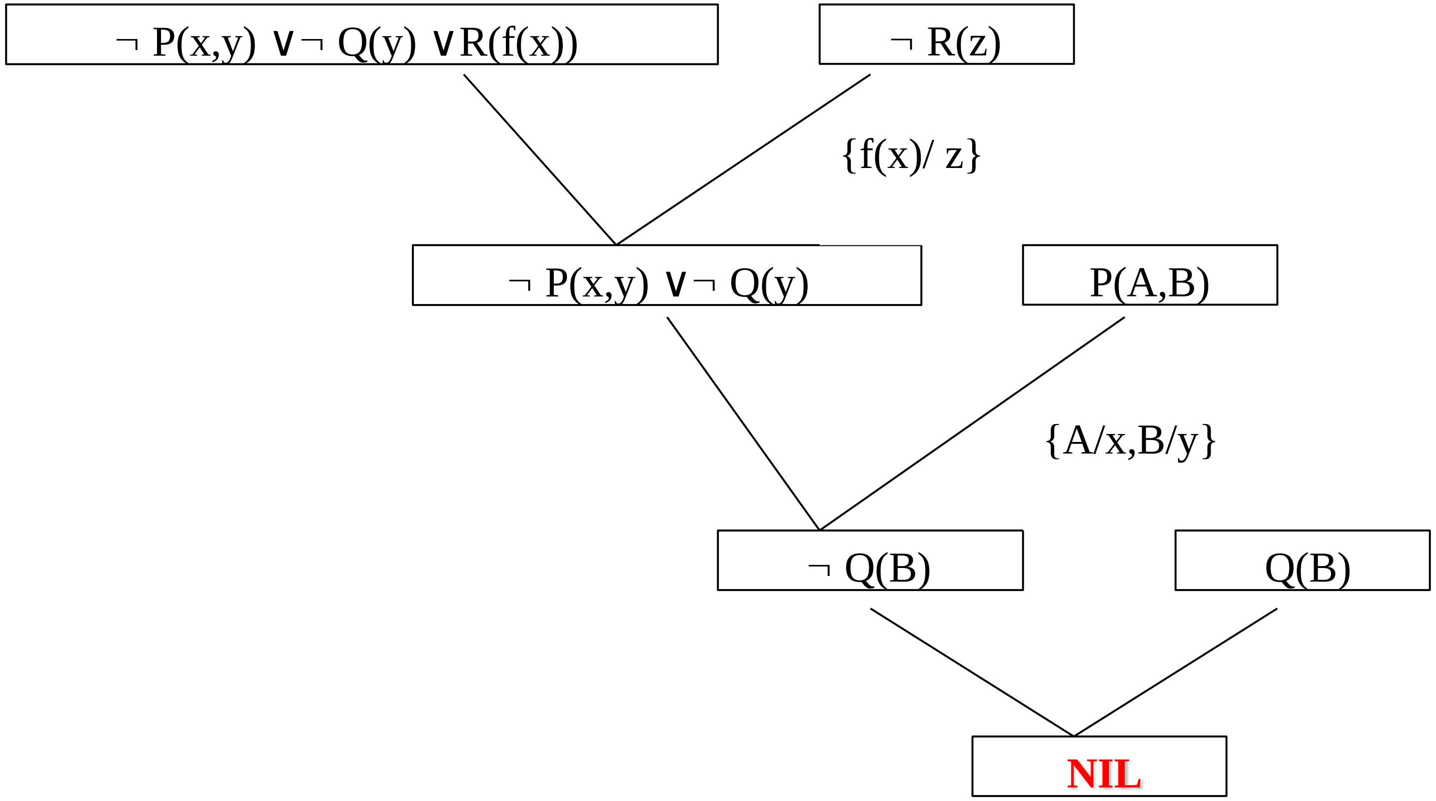
前提 F 所对应的子句集为：$\neg P(x,y)\vee\neg Q(y)\vee R(f(x))$

$\neg P(x,y)\vee\neg Q(y)\vee S(x,f(x))$

结论 G 的否定对应子句集为：$\neg R(z)$ ；$P(A,B)$ ；$Q(B)$

归结原理的应用

- 归结过程如下



归结原理的应用

- 利用归结原理求取问题答案

步骤：

1 已知前提条件用谓词公式表示，并化成相应的子句集 S_1。

2 待求解的问题也用谓词公式表示，然后将其否定，并与谓词 ANSWER 构成析取式。谓词 ANSWER 是一个专为求解问题而设置的谓词

3 把（2）中的析取式化为子句集，并把该子句集与 S_1 合并构成子句集 S

4 对子句集 S 应用归结原理进行归结，在归结的过程中，通过合一，改变 ANSWER 中的变元。

5 如果得到归结式 ANSWER ，则问题的答案即在 ANSWER 谓词中。

归结原理的应用

例：任何兄弟都有同一个父亲

John 和 Peter 是兄弟，且 John 的父亲是 David ，问 :Peter 的父亲是谁？

第一步：将已知条件用谓词公式表示出来，并化成子句集，那么要先定义谓词。

（1）定义谓词：

设 Father(x,y) 表示 x 是 y 的父亲。

Brother(x,y) 表示 x 和 y 是兄弟。

归结原理的应用

（２）将已知事实用谓词公式表示出来。

F_1 ：任何兄弟都有同一个父亲。

$\forall x \forall y \forall z\ (Brother(x,y) \wedge Father(z,x) \rightarrow Father(z,y))$

F_2 ： John 和 Peter 是兄弟。

Brother(John,Peter)

F_3 ： John 的父亲是 David 。 Father(David, John)

（３）将它们化成子句集得：

$S_1=\{\neg Brother(x,y) \vee \neg Father(z,x) \vee Father(z,y),$
$Brother(John,Peter), Father(David,John)\}$

归结原理的应用

- 第二步：把问题用谓词公式表示出来
    - 设 Peter 的父亲是 u ，则有： Father(u,Peter)
    - 将其否定与 ANSWER 作析取，得：

  G ： ¬ Father(u,Peter)∨ANSWER(u)

- 第三步：将上述公式 G 化为子句集 S_2, 并将 S_1 和 S_2 合并到 S 。

(1) ¬ Brother(x,y)∨¬ Father(z,x)∨Father(z,y)

(2) Brother(John,Peter)

(3) Father(David,John)

(4) ¬ Father(u,Peter)∨ANSWER(u)

归结原理的应用

(1)¬ Brother(x,y)∨ ¬ Father(z,x)∨Father(z,y)

(2) Brother(John,Peter)

(3) Father(David,John)

(4) ¬ Father(u,Peter)∨ANSWER(u)

第四步：应用归结原理进行归结

(5) ¬ Brother(John,y)∨Father(David,y)

（1）与（3）归结 σ={David/z,John/x}

(6) ¬ Brother(John,Peter)∨ANSWER(David)

（4）与（5）归结 σ={David/u,Peter/y}

(7) ANSWER(David) （2）与（6）归结

第五步：得到了归结式 ANSWER(David)

答案即在其中，所以 u=David。即 Peter 的父亲是 David

思考题

破案问题：在一栋房子里发生了一件神秘的谋杀案，现在可以肯定以下几点事实：

(1) 在这栋房子里仅住有 A,B,C 三人； (2) 是住在这栋房子里的人杀了 A ；

(3) 谋杀者非常恨受害者； (4)A 所恨的人，C 一定不恨；

(5) 除了 B 以外， A 恨所有的人； (6)B 恨所有不比 A 富有的人； (7)A 所恨的人，B 也恨； (8) 没有一个人恨所有的人； (9) 杀人嫌疑犯一定不会比受害者富有。

为了推理需要，增加如下常识：(10)A 不等于 B 。

问：谋杀者是谁？