第一章 简介

1. **人工智能是一门什么样的学科:** 一门由计算机科学、控制论、信息论、语言学、神经生理学、心理学、数学、哲学等多种学科相互渗透而发展起来的综合性新学科
2. **获得图领奖的人工智能学者:**

   Marvin Minsky(1969 年), 美国, 知识的框架理论（Frame Theory）创立者
   John McCarthy(1971 年),美国人工智能之父
   Herbert Simon 和 Allen Newell(1975 年),美国,人工智能符号主义学派的创始人
   Edward Feigenbaum 和 Raj Reddy (1994 年), 美国,大型人工智能系统的开拓者
   Leslie Gabriel Valiant(莱斯利·瓦里安特) (2010 年),英国
   姚期智 首位华裔 (2000 年) 在计算理论方面的贡献
   Judea Pearl (犹大·伯尔)(2011 年), 美国 最早提出概率和因果性推理演算法
3. **重要国际会议:**
   a) IJCAI（International Joint Conference on AI）国际人工智能联合会议,1969 年开始两年一次,2016 年开始一年一次.
   b) AAAI (AAAI Conference on Artificial Intelligence) AAAI 是国际人工智能促进协会
4. **重要国际期刊**
   a) AI (Artificial Intelligence) Elsevier 出版
   b) AI Magazine AAAI 出版
   c) IEEE Transactions on Pattern Analysis and Machine Intelligence
   d) Journal of Machine Learning Research
5. **国内会议:**全国人工智能学术年会（CAAI）, 中国人工智能联合会议（CJCAI）
6. **人工智能的研究目标:**要求计算机不仅能模拟而且可以延伸、扩展人的智能,达到甚至超过人类智能的水平。
7. **人工智能的学派**
   a) **符号主义/逻辑主义学派,(符号智能)**又称“功能模拟”学派，主张从功能方面模拟、延伸、扩展人的智能；以研究符号为基础。代表性成果：启发式程序、专家系统、知识工程等。
   b) **连接主义(计算智能),** “结构模拟”学派，主张从结构方面模拟、延伸、扩展人的智能，用“电脑”模拟“人脑”神经系统的联结机制；以研究人脑为物质基础。代表成果是各种神经网络.
   c) **行为主义(低级智能),** “行为模拟”学派，主张从行为方面模拟、延伸、扩展人的智能，认为：“智能”可以不需要“知识”，认为智能只是在与环境的交互作用中表现出来。该学派认为人工智能源于控制论。早期研究工作重点是模拟人在控制过程中的智能行为和作用。代表性成果：MIT 的 Brooks 研制的智能机器人。
8. **人工智能的主要研究领域**

   自动推理,专家系统,机器学习,自然语言理解,机器人学和智能控制,模式识别,基于模型的诊断,智能规划,智能 Agent,神经网络,智能信息检索,自动程序设计,博弈
9. **人工智能与其它领域的交叉**

   数据挖掘--AI 与数据库的交叉领域
   基于知识的软件工程-- AI 与软件工程的结合
   基于 Agent 的软件工程-- AI 中多 Agent 技术与软件工程的结合
   网络智能-- AI 与因特网技术的结合
   计算机动画自动生成-- AI 技术与计算机图形学及电影艺术结合的产物

第二章 产生式系统

1. 概述

a) 产生式系统的三大组成部分:综合数据库,产生式规则,控制系统;
b) 综合数据库存储问题的状态描述
c) 产生式规则描述状态之间的转换,形如 if 条件满足 then 操作;
d) 控制系统决定在多个适用的产生式规则中选择哪个来执行,并决定系统何时停机;

2. 控制策略

a) 不可撤回策略:爬山法
   i. 目标状态的爬山函数值最大
   ii. 每一次选择规则都是选择爬山函数值最大的
   iii. 爬山函数值不降且尽量没有局部最优解
b) 回溯策略
   i. 一种可用的回溯策略:
      1. 产生了一个上溯至初始状态的路径上已经出现过的状态时回溯
      2. 搜索深度达到深度限制而没有结束时回溯;
      3. 没有可用的产生式规则时回溯
      4. 可以按照某种方式对可用的产生式规则进行排序
c) 图搜索策略
   i. 图搜索只是用图记录规则的应用和所产生的状态,
   ii. 图搜索并不是一种和回溯或不可撤回策略平行的第三种策略
d) 产生式系统的工作方式
   i. 正向和反向,顾名思义.
   ii. 双向,控制系统决定使用正向规则还是反向规则,并判断已经产生的状态和目标状态能否以某种方式匹配,从而终止搜索.

3. 特殊的产生式系统

a) 可交换产生式系统
   i. 每一条对状态 D 适用的规则对 D 的后继状态仍然适用
   ii. 如果状态 D 满足目标条件,则 D 的后继状态仍满足条件
   iii. 对 D 应用一个规则序列,最终结果与序列的顺序无关
b) 可分解产生式系统
   i. 状态是可以分解的
   ii. 产生式规则可以应用到每个组成部分上
   iii. 整个系统的终止条件可以用分解出的各个组成部分的终止条件描述

第三章 产生式系统的搜索策略

1. 回溯策略

a) 回溯算法(BACKTRACK)

i. 从开始状态出发,找出所有可用的产生式规则,

ii. 按照某种策略对这些可用的产生式规则进行排序,然后选出一个执行,产生一个新的状态;

iii. 从新的状态出发,递归调用,直到满足终止规则,如果新的状态没有任何可用的规则且不是终止状态,则回溯到前一个状态,从之前排列好的规则序列中选出未被尝试过的一个,进行递归调用;

iv. 程序只会记录从初始状态到目前正在探索的这个状态这条路经,其他的一律忘掉.并不生成树等数据结构.

v. 改进的方式:如果在之后的探索过程中发现生成了当前路径的前面存在的状态,就立刻终止继续探索并回溯.设定最大探索深度等

b) **例子**

i. 八皇后问题的解法

1. 第 i 层节点探索棋盘的第 i 行的摆法
2. 排序策略: 如果 diag(i, j) < diag(i, k), 则在排序中把 Rij 放在 Rik 的前面；如果 diag(i, j)=diag(i, k), j<k,则把 Rij 放在 Rik 的前面。其中 diag(i, j)定义为通过单元(i, j)的最长对角线的长度 .

2. 图搜索策略

a) 使用有向图记录全部的搜索过程.根节点的深度为 0.关于有向树和有向图的细节没有什么特殊的规定

b) 图搜索是一个大略的框架,并不是指某种特定的算法,只要满足 a,都是图搜索.

3. 无信息图搜索过程

a) 深度优先

i. 注意不带回溯策略的深度优先会生成整个决策树,对于有些问题例如 8 数码谜题,回溯策略可能不好设计或者不怎么有效;

ii. 可以设置深度限制

b) 广度优先

i. 如果存在解路径,广度优先搜素一定能找到最优路径,如果不存在解路径,可能会失败,也可能会无法停机.

4. 启发式图搜索

a) 启发式图搜索的极限情况是广度优先搜索.

b) f*(n)=g*(n)+h*(n), f*(n)是从起点到终点,经过节点 n 的最短路径的费用, g*(n)表示从初始节点到 n 的最佳解路径的费用, h*(n)表示通过节点 n, 从起点到目标节点的最佳解路径的费用.估价函数 f(n)=g(n)+h(n)是对 f*(n)的估计.h(n)称为启发函数.

c) 根据估价函数对所有节点进行估值,在当前所有开放节点中选择函数值最小的进行下一步扩展,如果选择扩展某个节点,就扩展这个节点的全部子节点,并计算估价函数值.

d) 完成一次扩展后,参与排序的节点包括刚生成的本层节点和尚未被扩展过的前面层的节点.

e) 发现解路径后,搜索终止.

f) 使用估价函数的图搜索称为 A 算法;

g) 对任何节点 n 都有 h(n)≤h*(n)(低估)的 A 算法称为 A*算法.

5. **A*算法的可采纳性**
   a) 如果一个搜索算法对于任何有解路径的图都能找到一条最佳解路径,则是可采纳的.
   b) 图搜索对于有限图必然终止;在 A*算法终止前的任何时刻,OPEN 表中总是存在 n',n'在最佳路径上气人 f(n')≤f*(s);如果存在解路径,A*算法必然终止.
   c) A*算法是可采纳的.
6. **A*算法的比较**
   a) 解决同一个问题的 A*算法,启发函数 h(n)更接近 h*(n)的算法具有较多的信息.
   b) 如果两个 A*算法 A1 和 A2,其中 A2 具有的信息多,那么 A2 扩展的节点比 A1 少(A1 至少扩展了与 A2 相同多的节点),而且所有被 A2 扩展的节点也一定被 A1 扩展.
   c) 信息更多的 A*算法不一定效率更高.
   d) 在满足 h (n) ≤h*(n)的前提下，启发函数越大，其所包含的启发信息越多，所扩展的节点越少.
7. **单调限制**
   a) 如果启发函数 h 对任何节点 n_i和 n_j，只要 n_j是 n_i的后继，都有 $h(n_i)\leq c(n_i, n_j)+h(n_j)$, $h(t)=0$ (t 是目标节点),则称启发函数 h 满足单调限制.即一个节点到终点的代价估计小于等于他子节点的代价估计加上他和这个子节点之间的有向弧的代价.
   b) 如果 A*算法启发式函数 h 满足单调限制,则 A*所扩展的节点序列的估价函数值是非递减的.
   c) 满足单调限制的情况下,每走一步都在最佳解路径上.
8. **A*算法的启发能力**
   a) 设 A_1和 A_2是两个启发式算法，它们分别使用估价函数 f_1和 f_2，如果在寻找解路径的过程中，A_1所用的计算费用比 A_2少，则称 A_1比 A_2有较强的启发能力，也可以称估价函数 f_1比 f_2有较强的启发能力.
   b) 影响算法 A 启发能力的三个重要因素：
      i. 算法 A 所找到的解路径的费用。(找到的解路径越长,过程中的计算开销越大)
      ii. 算法 A 在寻找这条解路径的过程中所需要扩展的节点数。
      iii. 计算启发函数所需要的计算量。
   c) 为了增加启发能力可以牺牲可采纳性,这样也许能在较短的时间内找到一个次优解.
   d) 在八数码问题中一个可用的启发函数(注意它破坏了可采纳性)
      i. h(n)=P(n)+3S(n)
      ii. P(n)：每个数码离“家”距离的和。
      iii. S(n)：记分函数：对于中心方格，若有数码，记 1 分，否则记 0 分。
      iv. 对非中心的外围上的数码，沿顺时针方向依次检查每个数码：若此数码与其后面的数码与目标中顺序不同(若此数码后面的数码不是它在目标状态的后继)，则记 2 分，否则记 0 分。
9. **启发能力的度量**
   a) **渗透度**是对一个搜索算法的搜索性能的度量，表示搜索集中指向某个目标的程度，而不是在无关的方向上徘徊。
      i. P = L / T
      ii. L 是算法发现的解路径的长度, T 是算法在寻找这条解路径期间所产生的节点数（不包括初始节点，包括目标节点）
      iii. 若一个算法每次选取的节点都在解路径上，则 L= T ，P=1；一般搜索的渗透度 P<1 ；无信息的搜索 P<<1；

b) **有效分枝系数**就是一棵搜索树的平均分枝数．
   i. 设搜索树的深度是 L，算法所产生的总节点数为 T，有效分枝系数是 B，则有 B＋B2 十…＋BL=T 或 B（BL-1）/（B-1）=T

第四章 可分解产生式系统的搜索策略

1. 与或图

a) 父节点与一组子节点用超弧连接,超弧又称 k-连接符.
b) 当 K>1 的时候,用角度弧线标记这个连接符.
c) 若所有的 k-连接符都是 1-连接符,得到与或图的特例:普通有向图.
d) 与或树:每个节点最多有一个父节点的与或图.

2. 与或图搜索

a) 可分解产生式系统实际上是从起始符号出发的一套推导规则,这些规则中隐含了一个与或图,图的根节点是产生式系统的初始状态描述，连接符表示对一状态描述应用产生式规则或把这一状态描述分解成若干组成部分．
b) 可分解产生式系统的任务:从隐含的与/或图出发找出一个从根节点出发到终止节点集的解图。这个解图描述了如何从初始符出发以最小的代价推导出指定的解状态.
c) 代价标在路径上.
d) 假定 h*(n)是从 n 出发的最佳解图的费用,h(n)是 h*(n)的估计值.
e) 能解节点（SOLVED）:
   i. 终止节点是能解节点；
   ii. 若非终止节点有“或”子节点时，其子节点有一能解，该非终止节点是能解节点；
   iii. 若非终止节点有“与”子节点时，若其子节点均能解，该非终止节点是能解节点。

3. AO*算法

a) 定义变量:集合 G 为已经探明的节点集合,G'为目前要扩展的节点集合,h(n_i)为节点n_i的启发函数值,$q_j(n_i)$为从n_i发出的第 j 条连接符的代价. q(n_i)为节点n_i的代价.

b) 定义代价计算法:

i. 如果q(n_i)未曾被计算过且他的子节点未探明,则q(n_i) = h(n_i),如果之前已经计算过q(n_i),那么就取之前算好的值;

ii. 如果n_i发出两条连接符 1 和 2,则q(n_i) = min($q_1(n_i), q_2(n_i)$);

iii. $q_j(n_i)$=第 j 条连接符的固有代价+这条连接符指向的所有子节点的代价

c) 流程:

i. 外层循环:直到初始节点n_0被标记为 SOLVED

1. 将指针指向的连接符所指向的节点加入到 G',第一次循环直接将n_0加入 G';
2. 从 G'中选一个节点,标记为 n,扩展 n
3. 将扩展出来的节点加入 G,并计算他们的q(n_i)初值;
4. 初始化 S 集合={n};

a) 内层循环:直到 S 集合为空

i. m=从集合中取出的一个节点,保证 m 的子节点不在 S 中;

ii. 计算$q_1(m), \dots, q_n(m)$,即 m 发出的全部连接符的代价;

iii. q(m)= min($q_1(m), \dots, q_n(m)$),设$q_k(m)$是其中最小的;

iv. 将指针指向 m 的第 k 条连接符

v. 如果这条连接符所指向的所有子节点都是 SOLVED,则 m 也是 SOLVED;

vi. 如果被标记了 SOLVED 或者q(m)发生了更新,则将 m 的所有曾被指针标记过的连接符所通向的父节点加入 S

b) 内层循环结束

ii. 外层循环结束

d) 解释:

i. 从 G'中选一个节点的策略是未知的,可以是选 G'中 h 值最大的,或者 q 值最大的,或者随机选择.

ii. 用来标记连接符的指针全局上只存在一个,也就是说每次执行到"1"步骤的时候,最多只有一条被指针标记的连接符;

iii. 第一次执行的时候,由于 G 中只有一个初始节点n_0,而且指针还没有标记任何连接符,所以直接扩展n_0;

iv. 被角度弧标记的数条连线属于一条连接符.

4. 博弈树搜索

a) 极大极小过程

i. 静态估值函数 e(p),对一个局面打分,对我方有利打正分,对我方不利打负分.我方必胜打正无穷,我方必败打负无穷;

ii. 顶点深度为 0,Max 代表我方,Min 代表地方,我方执 Max 先行.

iii. Max 一方在所有的选择中走 e 最大的那个,min 在所有的选择中走 e 最小的那个.

iv. 评价函数在搜索树的最底层开始行动,

v. 过程:

1. 根据最大深度,广度优先,生成搜索树;
2. 对树的最底层计算估值函数值,并标记出每层节点是 Min 还是 Max;
3. 从最后一层倒着算到定点,
4. 根据每层取到最大最小值的路径找出最终的决策方向.

b) α-β剪枝规则

i. 在计算过程中,对于每个节点,存储α值,β值和确定值, α≤确定值≤β;

ii. 如果在某一时刻α≥β,则发生剪枝,不再继续探索此节点的其他子节点;

iii. 如果剪枝发生在 Min 节点上,称为α剪枝;如果发生在 Max 节点上,称为β剪枝;

1. 过程:
   a) 从根节点出发,深度优先的,按照最大深度,生成最底层的第一个叶节点,所有非叶节点在生成时,初始化α=-∞,β=+∞,确定值为未定义;
   b) 叶节点调估值函数,计算评价值;
   c) 逐个生成第一个叶子节点的全部兄弟节点,每生成一个,就更新他父节点的(β/α)值,直到生成完毕,
   d) 当一个节点的全部子节点探索完毕,或已经确定进行剪枝时,为该节点的确定值赋值,如果他是一个 Min 节点,赋值为β值,如果为 Max 节点,赋值为α值;
   e) 当一个节点获得了自己的确定值时,回溯到上一层,如果是从 min 节点发生的回溯,则用自己的β值更新给父节点的α值,如果是从 Max 节点发生的回溯,则用自己的α更新给父节点的β值,(更新的时候注意α值不能变小,β值不能变大,即范围不能变大)并探索兄弟节点;
   f) 如果本节点是 max 节点,回溯后向下探索时,带上α值,如果是 min 节点,回溯后向下探索时带上β值;
   g) 无论何时,只要一个节点的α≥β,就不在探索这个节点的其他子节点.
   h) 当根节点获得自己的确定值时搜索结束,提供这个值的子树方向就是决策结果的方向.

iv. 注意:

1. Min 节点无论是向上返回还是向下探索,总是带着他的β值,Max 节点总是带着他的α值,
2. 向上返回的时候,会发生α和β的交叉赋值,向下探索的时候,会带着原值.
3. 如果没有其他值来源,节点的α=-∞,β=+∞
4. 在使用相同存储空间的条件下, α-β 过程能把搜索深度扩大一倍 .

第五章 谓词演算

1. 引言

a) 逻辑连接词
   i. ~**非**,∨**或(析取)**,∧**且(合取)**.$P \to Q$**若 P 则 Q**;P↔Q **等值**
b) 原子:命题变元,用大写字母 P,Q 等表示;
c) 公式:由原子和逻辑连词组成的有限长度的串;
d) 解释:$T_I(G)$为公式 G 在解释 I 下的真值;
e) 等价关系
   i. (G↔H)=(G→H)∧(H→G)
   ii. (G→H)=(~ G∨H)
   iii. G∨G=G，G∧G=G； (等幂律)
   iv. G∨H=H∨G，G∧H=H∧G； (交换律)
   v. G∨(H∨S)=(G∨H)∨S,
      G∧(H∧S)=(G∧H)∧S； (结合律)
   vi. G∨(G∧H)=G，G∧(G∨H)=G； (吸收律)
   vii. G∨(H∧S)=(G∨H)∧(G∨S),
      G∧(H∨S)=(G∧H)∨(G∧S)； (分配律)
   viii. G∨F=G，G∧T=G； (同一律)
   ix. G∧F=F，G∨T=T； (零一律)
   x. ~(G∨H)= ~G∧ ~H,
      ~(G∧H)= ~G∨ ~H。 (De Morgan 律)
   xi. G∨ ~G=T；G∧ ~G=F （互补律）
   xii. ~ ~G=G （双重否定律）
f) 蕴含
   i. G→H 恒为真,记作$G \Longrightarrow H$,称 G 蕴含 H,或 H 是 G 的逻辑结果;
g) 范式
   i. 短语:原子或原子的非的合取;
   ii. 子句:原子或原子的非的析取;
   iii. 短语的析取称为析取范式,子句的合取称为合取范式;
   iv. 化范式:
      1. 使用 i 和 ii 等价关系去掉↔和→
      2. 使用摩根定律和双重否定率把所有的~放在原子之前
      3. 使用分配率转化为范式.

2. 谓词与量词

a) 设 D 是非空个体名称集合，定义在 D^n 上取值于{T，F}上的 n 元函数，称为 n 元命题函数或 n 元谓词。其中 D^n 表示集合 D 的 n 次笛卡尔乘积。
b) 零元谓词就是命题,一元谓词描述个体的关系,多元谓词描述多个个体的关系;对谓词指定元素就得到命题;
c) 语句 “对任意 x”称为全称量词,记以∀x,语句 “存在一个 x”称为存在量词,记以∃x;
d) 被量词作用的变量是约束的,否则是自由的.变量的出现要么是约束的要么是自由的,变量可以既是约束的又是自由的.
e) **约束变量的改名规则:**可将公式中出现的**约束变量**改为**另一个约束变量**，这种改名必须在量词作用区域内各处以及该量词符号中实行, 并且改成的新约束变量要有别于改名区域中的所有其它变量,改名不改变真值.

3. 公式的解释

a) 常量符号一般用小写英文字母 a,b,c∈D 代表
b) 变量符号一般有 u,v,w,x,y,z 代表,可以带入 D 中的元素.
c) 函数符号一般是 f(…),g(…)之类,可以是 D^n 到 D 的任意一个映射
d) 用大写英文字母 P, Q, R, …表示, 当个体名称集合 D 给出时, n 元谓词符号 $P(x_1, \cdots, x_n)$可以是 D^n 上的任意一个谓词。
e) 谓词逻辑中的项:
   i. 常量符号是项
   ii. 变量符号是项
   iii. 函数的变量符号用项替换,得到的还是项
   iv. 有限使用上述三条得到的串是项
f) 谓词逻辑中的原子:若 $P(x_1,\cdots,x_n)$是 n 元谓词符号，$t_1,\cdots,t_n$ 是项，则 $P(t_1,\cdots,t_n)$是原子。
g) 谓词逻辑中的公式:
   i. 原子是公式;
   ii. 公式应用逻辑连接词得到的还是公式;
   iii. 对公式中的变量添加量词约束,得到的还是公式;
   iv. 有限使用上述三条得到的串是公式;
h) 公式的解释:
   i. **对每个常量符号，指定 D 中一个元素 ；**
   ii. **对每个 n 元函数符号，指定一个函数，即指定 D^n 到 D 的一个映射 ；**
   iii. **对每个 n 元谓词符号，指定一个谓词，即指定 D^n 到{T，F}的一个映射。**
   iv. 给常量以具体元素,给抽象函数以具体函数,给谓词符号以真假.
i) 谓词逻辑是半可判定的;恒真性可在有限步骤内验证,非恒真不可在有限步骤内验证.
j) 等价: G=H **等价于** G↔H 恒真**等价于** G，H 在 I 下的真值相同
k) G 蕴涵 H 的充要条件是：对任意解释 I，若 I 满足 G，则 I 必满足 H;
l) 基本蕴含式:
   i. $(P \vee P) \Rightarrow P$
   ii. $P \Rightarrow (P \vee Q)$
   iii. $(P \vee Q) \Rightarrow (Q \vee P)$
   iv. $(Q \rightarrow R) \Rightarrow ((P \vee Q) \rightarrow (P \vee R))$
   v. $\forall x P(x) \Rightarrow P(y)$
   vi. $P(y) \Rightarrow \exists x P(x)$
   vii. $\forall x\, P(x) \Rightarrow \exists x P(x)$
   viii. $\forall x\, P(x) \vee \forall x\, Q(x) \Rightarrow \forall x\, (P(x) \vee Q(x))$
   ix. $\exists x (P(x) \wedge Q(x)) \Rightarrow \exists x P(x) \wedge \exists x Q(x)$
   x. $\forall x\, (P(x) \rightarrow Q(x)) \Rightarrow \forall x\, P(x) \rightarrow \forall x\, Q(x)$
   xi. $\exists x\, (P(x) \rightarrow Q(x)) \Rightarrow \exists x\, P(x) \rightarrow \exists x\, Q(x)$
m) 前束范式像这样的: **$\forall x \forall y \exists z (P(x, y) \rightarrow Q(x, z))$, $\forall x \forall y \exists z$ 称为首标,后边的称为母式**
n) 设 G 是仅含有自由变量 x 的公式，记以 G(x)，H 是不含变量 x 的公式，于是有
   i. (1) $\forall x(G(x) \vee H) = \forall x G(x) \vee H$
   ii. (1)' $\exists x(G(x) \vee H) = \exists x G(x) \vee H$
   iii. (2) $\forall x(G(x) \wedge H) = \forall x G(x) \wedge H$
   iv. (2)' $\exists x(G(x) \wedge H) = \exists x G(x) \wedge H$
   v. (3) $\sim(\forall x G(x)) = \exists x (\sim G(x))$

vi. (4) ~ (∃xG(x))= ∀x (~ G(x))

o) 设 H，G 是两个仅含有自由变量 x 的公式，分别记以 H(x)，G(x)，于是有：

i. (1) ∀xG(x)∧∀x H(x)= ∀x(G(x)∧H(x))

ii. (2) ∃xG(x)∨∃x H(x)= ∃x(G(x)∨H(x))

iii. (3) ∀xG(x)∨∀x H(x)= ∀x∀y (G(x)∨H(y))

iv. (4) ∃xG(x)∧∃x H(x)= ∃x∃y (G(x)∧H(y))

p) 任意公式都能转化成前束范式,步骤如下

i. 使用基本等价式 F↔H=(F→H) ∧(H→F) 和 F→H= ~ F∨H 将↔和→删掉

ii. 使用否定之否定和摩根率将所有的~放在原子之前

iii. 如果必要的话,将约束变量改名

iv. 使用上述两个引理将所有的量词提到公式的前面

q) **将前束范式转化为 Skolem 范式**

i. 设 G 是一个公式，$Q_1x_1\cdots Q_nx_nM$ 是与 G 等价的前束范式，其中 M 为合取范式形式。

ii. 若 Q_r 是存在量词，并且它左边没有全称量词，则取异于出现在 M 中所有常量符号的常量符号 c，并用 c 代替 M 中所有的 x_r，然后在首标中删除 Q_rx_r.

iii. 若 $Q_{s1}, \cdots, Q_{sm}$ 是所有出现在 Q_rx_r 左边的全称量词(m≥1,1≤s1<s2<…<sm<r),则取异于出现在 M 中所有函数符号的 m 元函数符号 $f(x_{s1},\cdots,x_{sm})$，用 $f(x_{s1},\cdots,x_{sm})$代替出现在 M 中的所有 x_r，然后在首标中删除 Q_rx_r.

iv. 对首标中的所有存在量词做上述处理后，得到一个在首标中没有存在量词的前束范式，这个前束范式就称为公式 G 的 Skolem 范式。其中用来代替 x_r 的那些常量符号和函数符号称为公式 G 的 Skolem 函数.

v. 例如 G=∃x∀y∀z∃u∀v∃wP(x,y,z,u,v,w)，用 a 代替 x，用 f(y，z)代替 u，用 g(y，z，v)代替 w，得公式 G 的 Skolem 范式：∀y∀z∀vP(a,y,z,f(y,z),v,g(y,z,v))

vi. 前束范式与原式等价，**Skolem 范式与原式不一定等价.**

r) 子句集

i. 设 G 是一阶逻辑中的公式，将其化为 Skolem 标准型，设其母式为 M，M 给出的子句集 S 称为公式 G 的子句集．

ii. 将子句集 S 理解为如下一个谓词逻辑中的公式：S 中的所有子句合取起来，并将 S 中的每个变量都看做具有全称量词约束的。

iii. 设 S 是公式 G 的子句集．于是，G 是不可满足的，当且仅当 S 是不可满足的．

iv. 设 G = G1∧…∧Gn ，Si 是 Gi 的 Skolem 标准型，i=1,…,n。令 S = S1U…USn .S 是 G 的子句集

v. 原公式是子句集的逻辑结果.

第六章 归结原理

1. 子句集的 Herbrand(埃尔布朗)域

a) Herbrand 域:

i. 设 S 为子句集，令 H_0 是出现于子句集 S 的常量符号集。如果 S 中无常量符号出现，则 H_0 由一个常量符号 a 组成。

ii. $H_i = H_{i-1} \cup$ {所有形如 $f(t_1, \cdots, t_n)$ 的项}，$f(t_1, \cdots, t_n)$ 是出现在 S 中的所有 n 元函数符号，$t_j \in H_{i-1}$，$j = 1, \cdots, n, i = 1, 2, \cdots$,

iii. 称 H_i 为 S 的 i 级常量集，H_∞ 称为 S 的 Herbrand 域，简称 S 的 H 域。

iv. 0 级就是常量,往后就是前一级并上把前一级带入到函数中得到的全部东西.

v. Herbrand 域中只存在常量和带入了常量的函数,没有别的东西.

b) 基：把对象中的变量用常量代替后得到的无变量符号出现的对象称为相应的基项、基项集、基原子、基原子集合、基文字、基子句、基子句集 (无变量的加个"基")

c) 原子集、Herbrand 底

i. 设 S 是子句集，形如 $P(t_1, \cdots, t_n)$ 的基原子集合，称为 S 的 Herbrand 底或 S 的原子集．

ii. 其中 $P(x_1, \cdots, x_n)$ 是出现于 S 的所有 n 元谓词符号，$t_1, \cdots, t_n$ 是 S 的 H 域中的元素．

iii. 例如: S={P(x)∨Q(x), R(f(y)) }， S 的 H 域={a, f(a), f(f(a)), … } S 的原子集：A={P(a), Q(a), R(a), P(f(a)), Q(f(a)), R(f(a)), … }

d) 基例

i. 设 S 是子句集，C 是 S 中的一个子句．用 S 的 H 域中元素代替 C 中所有变量所得到的基子句称为子句 C 的基例。

ii. 例如 S={P(x),Q(f(y))∨R(y)},C=P(x),S 的 H 域是{a,f(a),f(f(a))…},C 的基例为 P(a),P(f(a)),P(f(f(a))),…

e) 子句集的 H 解释

i. 例如: S={P(x) Q(x), R(f(y)) }

ii. S 的 H 域={a, f(a), f(f(a)), … }

iii. S 的原子集 A={P(a), Q(a), R(a), P(f(a)), Q(f(a)), R(f(a)), … }

iv. S 的 H 解释：

v. I_1 *={ P(a), Q(a), R(a), P(f(a)), Q(f(a)), R(f(a)), … }

vi. I_2 *={ ~P(a), ~Q(a), R(a), P(f(a)), ~Q(f(a)), ~R(f(a)), … }

vii. 可以看出由于原子集是个无穷集合,H 解释也是个无穷集合

viii. H 解释是个普通解释,但不是所有的普通解释都是 H 解释.

2. Herbrand(埃尔布朗)定理

a) 设 A 是原子，两个文字 A 和 ~A 都是另一个的补，集合{A, ~A}称为一个互补对．

b) 子句中含有一个互补对,就说这个子句是 **Tautology/重言式/恒真式,因为子句是一个析取式,是或的关系.**

c) 语义树

i. 设 S 是子句集，A 是 S 的原子集．关于 S 的语义树是一棵向下生长的树 T．在树的每一节上都以如下方式附着 A 中有限个原子或原子的否定：

ii. 对于树中每一个节点 N，只能向下引出有限的直接的节 $L_1, \cdots, L_n$．设 Q_i 是附着在 L_i 上所有文字的合取，i=1, … ,n，则 $Q_1 \vee \cdots \vee Q_n$ 是一个恒真的命题公式．

iii. 对树中每一个节点 N，设 I（N）是树 T 由根向下到节点 N 的所有节上附着文字的并集，则 I（N）不含任何互补对．

d) 完全语义树
   i. **设 $A=\{A_1,\cdots,A_n,\cdots\}$ 是子句集 S 的原子集．**
   ii. **如果一颗语义树中每一个尖端节点 N（即叶节点），都有 A_i 或 $\sim A_i$ 有且仅有一个属于 $I(N),i=1,\cdots,k,\cdots$ 则他是完全语义树.**
e) 正规语义树是满二叉树
f) **失效点**:N 为语义树 T 中的节点,S 为对应的子句集:
   i. I(N)弄假 S 中某个字句的某个基例;
   ii. I(N')不弄假 S 中任意子句的任意基例,其中 N'是 N 的任意祖先节点
g) 封闭语义树
   i. 语义树 T 是封闭的，当且仅当 T 的每—个分枝的终点都是失效点.
h) 推理点
   i. 在封闭语义树中,如果 N 的**所有**直接下降节点(所有儿子节点)都是失效点,则称 N 为推理点.
i) Herbrand 定理 I . 子句集 S 是不可满足的，当且仅当对应于 S 的每一个完全语义树都存在一个有限的封闭语义树 .
j) Herbrand 定理 II 子句集 S 是不可满足的，当且仅当存在 S 的一个有限不可满足的 S 的基例集 S'。
k) D-P 方法
   i. 恒真式删除规则: 从子句集 S 中删除所有的恒真式,如果剩下的是恒假的,那么 S 恒假
   ii. 单文字规则:删除 S 中的所有单元基子句 L,若剩下空集,则 S 为可满足的,否则,继续删除所有文字~L,如果剩下的子句集恒假,则 S 恒假
   iii. **纯文字规则:**称 S 的基子句中的文字 L 是纯的，如果 ~ L 不出现在 S 中.从 S 中删除所有包含纯文字的基子句,若剩下的是空集,则 S 可满足,否则若剩下的是恒假的,则 S 恒假.
   iv. **分裂规则:**若 $S=(A_1 \vee L) \wedge\cdots\wedge (A_m \vee L) \wedge(B_1 \vee \sim L) \wedge \cdots\wedge (B_n \vee \sim L)\wedge R$ 其中 A_i , B_i ,R 都不含 L 或 ~ L，令 $S_1 =A_1\wedge\cdots\wedge A_m\wedge R$,$S_2= B_1\wedge\cdots \wedge B_n\wedge R$,则当 S_1 和 S_2 同时恒假时 S 恒假.

3. 归结原理

a) **归结原理基本思想:**要检查一个子句集的可满足性,转而检查子句集 S 是否包含一个空子句·,如果包含空子句,则 S 是不可满足的,如果不包含空子句,但是能由 S 中推导出空子句,S 也是不可满足的.

b) **归结式:**对任意两个基子句 C_1 和 C_2。如果 C_1 中存在文字 L_1，C_2 中存在文字 L_2，且 $L_1 = \sim L_2$，则从 C_1 和 C_2 中分别删除 L_1 和 L_2，将 C_1 和 C_2 的剩余部分**析取**起来构成的子句，称为 C_1 和 C_2 的归结式，记为 $R(C_1, C_2)$.C_1 和 C_2 称为亲本子句.

c) **~P 和 P 可以归结为空子句.**

d) **归结演绎:** 设 S 是子句集。从 S 推出子句 C 的一个归结演绎是如下一个有限子句序列：C_1，C_2，…，C_k 其中 C_i 或者是 S 中子句，或者是 C_j 和 C_r 的归结式 $(j<i, r< i)$；并且 $C_k = C$。称从子句集 S 演绎出子句 C，是指存在一个从 S 推出 C 的演绎.且 C 是 S 的逻辑结果.如果 S 是可满足的,则从 S 推出的任意子句是可满足的.

e) 从 S 推出空子句的演绎称为反驳,存在反驳的子句集是不可满足的.

f) **演绎树:**这是一颗倒着的树(上大下小),描述了归结的过程,上层节点是亲本子句,对应的下层节点是归结式.

g) **例如: S={P∨Q, ~P∨Q, P∨~Q, ~P∨~Q }存在归结演义:**

h) **(1) P∨Q**

i) **(2) ~P∨Q**

j) **(3) P∨~Q**

k) **(4) ~P∨~Q**

l) **(5) Q 由(1)、(2)**

m) **(6) ~Q 由(3)、(4)**

n) **(7) · 由(5)、(6)**

o) **所以 S 是不可满足的.**

p) **用归结原理证明某式成立: 证明(P →Q) ∧~Q ⇒ ~p**

i. **首先建立子句集:S={~P∨Q, ~Q , P}(因为 P →Q=~P∨Q,且子句集是个合取式)**

ii. **进行归结演义:**

iii. **(1) ~P∨ Q**

iv. **(2) ~ Q**

v. **(3) P**

vi. **(4) ~P (1)(2)归结**

vii. **(5)Q (1)(3)归结**

viii. **S 是不可满足的, (P →Q) ∧~Q∧P ⇒ F,所以原式成立.**

4. 合一算法

a) 替换: 一个替换是形如{$t_1/v_1, \cdots, t_n/v_n$ }的一个有限集合，其中 v_i 是变量符号，t_i 是不同于 v_i 的项。并且在此集合中没有在斜线符号后面有相同变量符号的两个元素，称 t_i 为替换的分子，v_i 为替换的分母。

b) 基替换:当 t1,⋯,tn 是基项时，称此替换为基替换

c) **空替换：**没有元素的替换称为空替换，记为ε。

d) 改名:设替换σ ={ $t_1/x_1, \cdots, t_n/x_n$ },如果 $t_1, \cdots, t_n$ 是不同的变量符号，则称σ为一个改名替换，简称改名。

5. 表推演

a) 符号公式:在公式 X 前面加上 T 和 F,TX 的真假性与 X 相同,FX 的真假性与 X 相反;;

b) 符号公式的分类:

α	α_1	α_2
T(X∧Y)	TX	TY
F(X∨Y)	FX	FY
F(X⟶Y)	TX	FY
T(~x)	FX	FX

β	β_1	β_2
T(X∨Y)	TX	TY
F(X∧Y)	FX	FY
T(X⟶Y)	FX	TY
F(~x)	TX	TX

c) 命题逻辑表推演的扩展规则:

i. α规则:$\frac{\alpha}{\begin{matrix}\alpha_1\\ \alpha_2\end{matrix}}$

ii. β规则$\frac{\beta}{\beta_1 \quad \beta_2}$

d) 封闭树:所有的原子都有 T 和 F 两种符号公式.

e) 构造过程:

f) 例如证明(p→(q→r))→((p→q)→(p→r))的有效性:

i. ①F(p→(q→r))→((p→q)→(p→r))

ii. ②T(p→(q→r))) ①α

iii. ③F((p→q)→(p→r)) ①α

iv. ④T(p→q) ③α

v. ⑤F(p→r) ③α

vi. ⑥Tp ⑤α

vii. ⑦Fr ⑤α

viii. ⑧Fp ⑨T(q→r) ②β

ix. ⑩Fp ⑪Tq ④β

x. ⑫Fq ⑬Tr ⑨β

xi. 由于 p,q,r 均有 F 和 T 两种符号公式,表推演树封闭,所以原公式有效

g) 证明公式有效:从 FS 推演出封闭树;证明公式 S 不可满足:从 TS 推演出封闭树

6. 扩展规则推理方法

a) 扩展规则:给定一个子句 C 和一个原子的集合 AT:D={C∨a, C∨¬a|a ∈ AT且 a 和¬a都不在 C 中出现},把从 C 到 D 中元素的推导过程叫做扩展规则,D 中的元素叫做应用扩展规则的结果,C 和他扩展的结果 D 是等价的.

b) 极大项:一个非重言式(恒真式)子句是集合 AT 上的极大项当且仅当他包含集合 AT 上的所有原子或其否定.

c) 给定一个子句集φ,他的原子集是 AT,(|AT|=m),若φ中的子句都是 AT 上的极大项,且 AT 中含有 2^m 个互相不同的子句,**则φ是不可满足的**.

d) 子句集φ={C_1,C_2,⋯,C_n},AT 是他的原子集,|AT|=m,P_i 是 C_i 扩展出的所有极大项的集合,S 为φ能扩展出的所有极大项的个数,

$$S=\left|\bigcup_{i=1}^{n}P_i\right|=\sum_{i=1}^{n}|P_i|-\sum_{1\le i\le j\le n}|P_i\cap P_j|+\sum_{1\le i\le j<\le n}|P_i\cap P_j\cap P_l|-\cdots+(-1)^{n+1}|P_1\cap P_2\cap\ldots\cap P_n|$$

$|P_i|=2^{m-|C_i|}$　　$|P_i\cap P_j|=\begin{cases}0, 在C_i\cup C_j 中含有互补对\\ 2^{m-|C_i\cup C_j|}, 否则\end{cases}$　　|Ci|是 Ci 子句含有的原子的数量

S 的计算是这样的:对所有的子句,计算$|P_i|$;

对所有子句的 2 次交叉,计算$|P_i\cap P_j|$;

…

对所有子句的 n 次交叉,计算$(-1)^{n+1}|P_1\cap P_2\cap\ldots\cap P_n|$

对上述结果求和.

e) 例如:判定子句集φ={¬A ∨ B ∨ ¬C, A ∨ C, ¬A}的可满足性.

i. 原子集 AT={A,B,C},m=|AT|=3

ii. ¬A ∨ B ∨ ¬C,扩展不出任何极大项

iii. A ∨ C,缺一个 B,能扩展出 2 个极大项

iv. ¬A,缺一个 B 一个 C,能扩展出 4 个极大项

v. 一共能扩展出 6 个极大项,2^M=8>6,所以φ可满足.

f) 例如:证明 S={P∨Q，Q→R，P→M, ~ M} ⇒R

i. 要证 S⇒R,只需证明 S∧ ~R是不可满足的;S'={P∨Q，~Q∨R，~P∨M, ~ M,~R}

ii. $AT_{S'}$={P,Q,R,M},m=|AT|=4,2^4=16