知识表示与推理

人工智能的任务是用机器模拟人类的智能，推理与问题求解是人类智能的两个主要功能。计算机要像人一样解决问题，首先需要将问题表示给计算机，即知识表示；然后计算机要有推理和问题求解的能力，即推理和问题求解。传统人工智能分为三个部分：确定性知识表示与推理、不确定性知识表示与推理、问题求解。

1. 确定性知识表示

知识表示是人工智能最基本的技术之一，其基本任务就是用一组符号将知识编码成计算机可以接受的数据结构，即通过知识表示可以让计算机存储知识，并在解决问题时使用知识。所谓知识表示过程就是把知识编码成某种数据结构的过程。一般来说，同一知识可以有多种不同的表示形式，而不同表示形式所产生的效果又可能不一样。确定性知识，是其结果只能为“真”或“假”的知识，这些知识是可以精确表示的。

1.1 命题与谓词

1.1.1 命题

对确定的对象做出判断的陈述句称为命题，一般用大写字母 P、Q 等表示。

例如“雪是白的”，“齐次线性方程组无解”，“20 是 5 和 10 的最小公倍数”。

命题的判断结果称为命题的真值，般使用 T(真)、F(假)表示。

命题的真值有以下特点：

- 只能有一个取值，要么为T(真)、要么为F(假)，不能同时既为真又为假；
- 在一定条件下命题为真，而在另一条件下命题为假。

不能再分解的陈述句称为简单命题，又称为原子命题

可以分解为几个原子命题的命题称为复合命题。

例 “2 是偶数而且 3 是奇数”是两个原子命题“2 是偶数”和“3 是奇数”组合而成的复合命题，该复合命题是由一个连接词“而且”连接而成的。

在命题中可以使用逻辑连接词将原子命题连接组成复合命题(命题公式)。连接词有如下五个：

- “¬”称为“非”，表示对后面的命题的否定，使该命题的真值与原命题相反；
- “∨”称为“析取”，$P \vee Q$读作P与Q的析取，表示“或”的关系；
- “∧”称为”合取”，$P \wedge Q$读作 P 与 Q 的合取，表示“与”的关系；

- “→”称为“蕴含”，表示”若…则…“的语义，$P \to Q$读作P蕴含Q，一般称P为前件，Q为后件；
- “↔”称为“等价”，表示“P 当且仅当 Q”的语义，$P \leftrightarrow Q$读作 P 等价于 Q。

由原子命题和逻辑连接词组成的命题称为命题公式或复合命题，其语法如下：

- 单个原子命题是命题公式；
- 若A是命题公式，则$\neg A$也是命题公式；
- 若A、B都是命题公式，则$A \wedge B$、$A \vee B$、$A \to B$、$A \leftrightarrow B$也都是命题公式。

复合命题的真值是通过下表进行运算的：

P	Q	¬P	P ∨ Q	P ∧ Q	P → Q	P ↔ Q
F	F	T	F	F	T	T
F	T	T	T	F	T	F
T	F	F	T	F	F	F
T	T	F	T	T	T	T

例 用命题逻辑的方式表示下面问题：若厂方拒绝增加工资，则罢工不会停止，除非罢工超过一年且工厂经理辞职。问：如果厂方拒绝增加工资，而罢工刚刚开始，罢工能否停止?

定义命题

P：厂方拒绝增加工资

Q：罢工停止

R：工厂经理辞职

S：罢工超过一年

构造命题公式

1. 若厂方拒绝增加工资，则罢工不会停止：$P \to \neg Q$
2. 除非罢工超过一年且工厂经理辞职：$\neg Q \leftrightarrow (S \wedge R)$
3. 当前情景：$P \wedge \neg Q$

真值计算

$P \to \neg Q$为真

当$\neg S$时(罢工刚开始)：$(S \wedge R) =$ 假

$\neg Q \leftrightarrow$ 真 $\Rightarrow Q$必须为假

结论：罢工不能停止

这样通过命题逻辑给出了一个用计算机推理的框架，通过程序计算机可以完成这个问题的推理过程。

命题逻辑推理有以下缺陷:

1. 使用命题讨论问题时，原子命题是最小单元，即原子命题是一个不可分的整体；
2. 命题无法表示不同事物的共性。

一个陈述句用一个符号表示，无法细分，即知识的颗粒太大，有些问题在命题逻辑的表示下推理困难。

P：小李是老李的儿子;

Q：张三是学生;

R：李四是学生。

用符号P表示命题“小李是老李的儿子”，那么命题“小张是老张的儿子”则需要用另个符号表示，而两个命题都表示两个人(小李，老李；小张，老张)之间的关系是“父子”那么是否可以用一个符号表示出“父子”关系？

还有一些问题用命题逻辑推理比较困难如：

所有科学都是有用的:

数理逻辑是科学:

数理逻辑是有用的。

人类进行以上问题推理时非常简单，但用命题表示问题时，三个命题间的关系无法显现，给计算机推理带来闲难。

1.1.2 一阶谓词

谓词逻辑是在命题逻辑的基础上发展起来的，其基本想法是把命题分解为两部分：

- 谓词名：表示个体的属性、状态、动作或个体间的关系；
- 个体：命题的主语，用来表示客观世界存在的事物或者某个抽象概念。

下面给出谓词、函数两个概念：

- 谓词：如果D是个体域，P：$D^n \to \{T, F\}$是一个映射，其中$D^n = \{(x_1, x_2, \ldots, x_n) | x_1, x_2, \ldots, x_n \in D\}$，则称$P$是一个$n$元谓词，记为$P(x_1, x_2, \ldots, x_n)$。
- 函数：设D是个体域，$f: D^n \to D$是一个映射，其中$D^n = \{(x_1, x_2, \ldots, x_n) | x_1, x_2, \ldots, x_n \in D\}$，则称$f$是一个$n$元函数，记为$f(x_1, x_2, \ldots, x_n)$。

例如：$P(x, y)$表示“x是y的父亲”，$Q(x, y)$表示“x是y的儿子”，$R(x, y)$表示“x和y是夫妻”，本身取值是 T(真)或者 F(假)，因此都属于谓词，而不是函数。

前面定义的谓词一般称为一阶谓词。当谓词中的某个变元x_i也是谓词时则称之为二阶谓词。

- 项满足如下规则：

单独的一个个体是项;

若$t_1, t_2, \ldots, t_n$是项，f是n元函数，则$f(t_1, t_2, \ldots, t_n)$是项。

- 即项是个体常量、个体变量和函数的统称。
- 原子谓词：若$t_1, t_2, \ldots, t_n$是项，P是谓词名，称$P(t_1, t_2, \ldots, t_n)$为原子谓词。
- 通过连接词将原子谓词进行连接形成的谓词被成为符合谓词。
- 复合谓词中的连接词与命题中的连接词相同。

与命题逻辑相比，由于谓词带有变元，其需要量词的约束。

量词是由量词符号和被其量化的变元所组成的表达式，用来对谓词中的个体做出量的规定。

- **全称量词符号**$\forall$：语义是”个体域中的所有或任意一个x“，谓词公式$(\forall x)P(x)$为真的含义是对个体域中所有的x，$P(x)$都为真；
- **存在量词符号**$\exists$：语义是”个体域中至少存在一个x“，谓词公式$(\exists x)P(x)$为真的含义是个体域至少有一个x，使$P(x)$为真。

有了前面的准备，下面给出谓词公式的概念：满足如下规则的谓词演算可得到谓词公式：

- 单个原子谓词公式是谓词公式；
- 若A是谓词公式，则$\neg A$也是谓词公式；
- 若A、B都是谓词公式，则$A \wedge B$、$A \vee B$、$A \rightarrow B$也都是谓词公式；
- 若A是谓词公式，x是项，则$(\forall x)A$、$(\exists x)A$也都是谓词公式。

谓词逻辑是一种形式语言，也是能够表达人类思维活动规律的一种精确语言。其适合表示事物的状态、属性、概念等事实性知识，同时也可以用来表示事物间的因果关系。

1.1.3 基于谓词逻辑的知识表示

使用谓词逻辑表示知识的一般步骤为：

(1) 根据所表示的知识定义谓词

(2) 使用连接词和量词将谓词连接成公式

例 “快乐学生”问题：假设任何通过计算机考试并获得奖励的人都是快乐的，任何肯学习或幸运的人都可以通过所有考试，李不肯学习但他是幸运的，任何幸运的人都能获奖。求证：李是快乐的。

第一步：定义谓词

(1) Happy(x)：x 快乐；

(2) Study(x)：x 肯学习；

(3) Lucky(x)：x 幸运；

(4) Win(x)：x 获奖；

(5) Pass(x, y)：x 通过考试 y；

第二步：用谓词表示问题

(1) 任何通过计算机考试并获奖的人都是快乐的：

$$(\forall x)[\text{Pass}(x, \text{Computer}) \wedge \text{Win}(x) \rightarrow \text{Happy}(x)]$$

(3) 任何肯学习或幸运的人都可以通过所有考试：

$$(\forall x)(\forall y)[\text{Study}(x) \vee \text{Lucky}(x) \rightarrow \text{Pass}(x, y)]$$

(4) 李不肯学习但他是幸运的：

$$\neg\text{Study}(\text{Li}) \wedge \text{Lucky}(\text{Li})$$

(5) 任何幸运的人都能获奖：

$$(\forall x)[\text{Lucky}(x) \rightarrow \text{Win}(x)]$$

结论：Happy(Li)

例 机器人移盒子问题：设在一个房间里，c 处有一个机器人，a处和b处各有一张桌子，a桌上有一个盒子，要求机器人从 c 处出发把盒子从a桌移到b桌上，然后回到 c 处。用谓词逻辑描述机器人的行动过程。

第一步：定义谓词

(1) 描述状态：

$\text{TABLE}(x)$：x是桌子

$\text{EMPTY}(y)$：y手中是空的

$\text{AT}(y, z)$：y在z处

$\text{HOLDS}(y, w)$：y拿着w

$\text{ON}(w, x)$：w在x上

(2) 描述操作：

$\text{GOTO}(x, y)$：从x处走到y处

$\text{PICKUP}(x)$：在x处拿起盒子

$\text{SETDOWN}(x)$：在x处放下盒子

(3) 操作规则：

$\text{GOTO}(x, y)$条件：$\text{AT}(robot, c)$，结果：$\text{AT}(robot, y)$

$\text{PICKUP}(x)$条件：$\text{ON}(box, x) \wedge \text{TABLE}(x) \wedge \text{AT}(robot, x) \wedge \text{EMPTY}(robot)$，结果：$\text{HOLDS}(robot, box)$

$\text{SETDOWN}(x)$条件：$\text{AT}(robot, x) \wedge \text{TABLE}(x) \wedge \text{HOLDS}(robot, box)$，结果：$\text{EMPTY}(robot) \wedge \text{ON}(box, x)$

第二步：问题描述

(1) 初始状态：

AT$(robot, c)$

EMPTY$(robot)$

ON(box, a)

TABLE(a)

TABLE(b)

(2) 目标状态：

AT$(robot, c)$

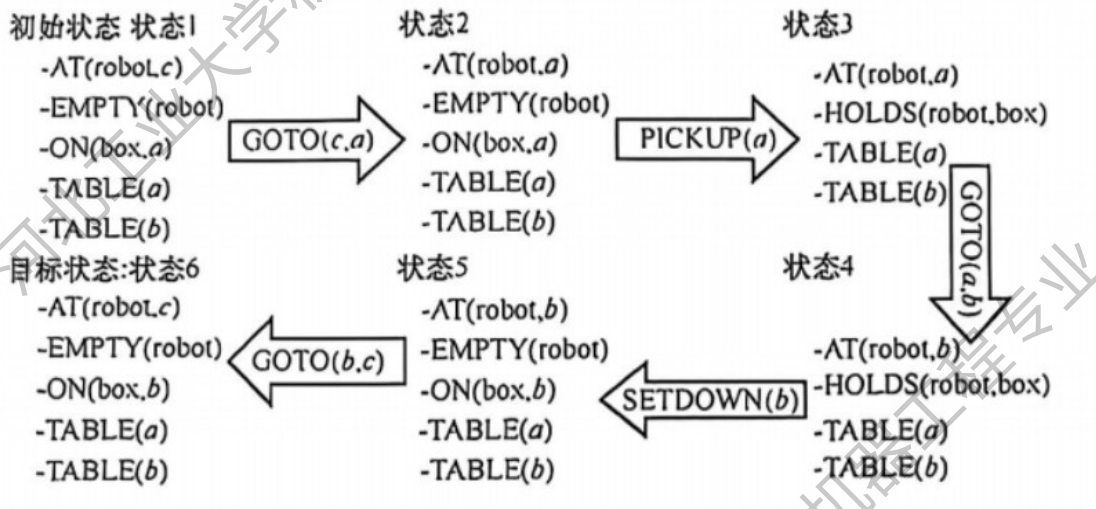
EMPTY$(robot)$

ON(box, b)

TABLE(a)

TABLE(b)

具体求解过程如图所示：



1.1.4 谓词范式和置换

范式是指谓词公式的标准形式，常用的谓词公式范式有析取范式、合取范式、前束范式和Skolem 范式。

- 定义原子谓词或原子谓词的否定称为文字，有限个文字的析取式称为一个子句。
- 有限个文字的合取式称为短语。
- 有限个短语的析取式称为析取范式。
- 有限个子句的合取式称为合取范式。

有这样的结论：对于任意谓词公式，都存在与之等价的析取范式和合取范式。

谓词公式G称为**前束范式**，如果G有如下形状：

$$(Q_1x_1)(Q_2x_2)\cdots(Q_nx_n)M$$

其中，$Q_i x_i$为$\forall x_i$或$\exists x_i$，$i = 1,2,\cdots,n$；M是不含量词的谓词公式；$(Q_1x_1)(Q_2x_2)\cdots(Q_nx_n)$称为**首标**；$M$称为**母式**；

存在性定理：∀谓词公式G，∃与之等价的前束范式

Skolem 范式定义：设谓词公式G的前束范式为$(Q_1x_1)(Q_2x_2)\cdots(Q_nx_n)M$，若首标中$\forall Q_i$ $(i = 1,\cdots,k \leq n)$都是全称量词，则称该范式为 **Skolem 范式**。

不可满足性等价定理：设S是G的 Skolem 范式，则G不可满足⇔S不可满足

置换与合一的定义：设$\theta = \{t_1/x_1, t_2/x_2, \cdots, t_n/x_n\}$为有限集合，其中，$t_i$为项，$x_i$为互不相同的变元，$t_i/x_i$表示用$t_i$置换$x_i$，并且要求$t_i$与$x_i$不能相同，$x_i$不能循环出现在另一个$t_j$中。一般用希腊字母$\theta$和$\lambda$表示。称这样的集合为谓词公式的置换。

对谓词公式F应用置换θ，记作$G = F\theta$，即将F中所有x_i替换为t_i $(i = 1,2,\cdots,n)$，所得新公式 G 称为F在θ下的**例示**。

基本性质：

∀谓词公式 F，其例示 G 都是 F 的逻辑结论，即

$$F \Rightarrow G$$

设公式集 $F = \{F_1, F_2, \cdots, F_n\}$，若存在置换 θ 使得：

$$F_1\theta = F_2\theta = \cdots = F_n\theta$$

则称θ是F的**合一**(unifier)，并称F是**可合一的**(unifiable)。

1.1.5 一阶谓词逻辑表示的特性

▌优点

✓ 自然性强：接近自然语言的表达方式

✓ 结构明确：

- 原子谓词构造合式公式有严格定义
- 语法规则清晰

✓ 语义精确：

- 限定二值逻辑(真/假)
- 消除自然语言歧义

✓ 灵活独立：

- 知识与处理程序分离
- 表示不依赖具体推理机制

✓ 模块化设计：

- 知识条目相互独立

• 便于知识库维护更新

▌缺点

✕ 表达能力受限：

• 难以表示不确定知识

• 时序性知识表达困难

✕ 管理复杂度高：

• 知识库规模膨胀快

• 一致性维护困难

✕ 组合爆炸问题：

• 谓词组合呈指数级增长

• 搜索空间快速扩大

✕ 执行效率低下：

• 推理过程计算密集

• 实时性较差

1.2 知识的产生式表示

“产生式”是由美国数学家波斯特(E. POST)于 1934 年首先提出的，它是根据串代替规则提出的一种称为波斯特机的计算模型，模型中的每条规则称为产生式。1972 年，纽厄尔和西蒙在研究人类的认知模型中开发了基于规则的产生式系统。产生式表示法已经成为人工智能中应用最多的一种知识表示模式，尤其是在专家系统方面，许多成功的专家系统都采用产生式知识表示方法。

1.2.1 事实表示

事实表示是指把事实看作是断言一个语言变量的值或多个语言变量间的关系的陈述句。对确定性知识的表示为一个三元组：

$$(\text{对象，属性，值})\text{或}(\text{关系，对象 1，对象 2})$$

例如：① 雪是白的；② 王峰热爱祖国。

$$(\text{雪，颜色，白})\text{；}(\text{热爱，王峰，祖国})$$

1.2.2 规则的表示

规则一般描述事物间的因果关系，规则的产生式表示形式称为产生式规则，简称为产生式。

$$P \to Q \text{ 或 IF } P \text{ THEN } Q\ [\text{置信度}]$$

P 是产生式的前提或前件，一般由事实的逻辑组合构成；Q 是产生式的后件或结论，一般是结论或操作。

产生式的含义：如果前件 P 满足，则可推出结论 Q，或执行 Q 所规定的操作。

1.2.3 产生式与蕴含式的区别

蕴含式只能表示确定性知识，产生式不仅可以表示确定性知识，也可表示不确定性知识。使用过程中对前件的匹配，蕴含式要精确匹配，而产生式可以相似匹配，产生式规则的不确定性也可以有规则的置信度表示。

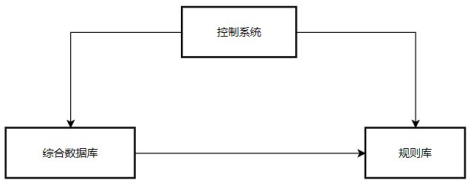
1.2.4 产生式语法规范

采用巴科斯范式(BNF)定义：

$$
\begin{aligned}
&\langle\text{产生式}\rangle ::= \langle\text{前提}\rangle \rightarrow \langle\text{结论}\rangle \\
&\langle\text{前提}\rangle ::= \langle\text{简单条件}\rangle \mid \langle\text{复合条件}\rangle \\
&\langle\text{结论}\rangle ::= \langle\text{事实}\rangle \mid \langle\text{操作}\rangle \\
&\langle\text{复合条件}\rangle ::= \langle\text{简单条件}\rangle \wedge \langle\text{简单条件}\rangle \\
&\langle\text{操作}\rangle ::= \langle\text{操作名}\rangle[\langle\text{变元}\rangle,\ldots]
\end{aligned}
$$

1.2.5 产生式系统

通常将使用产生式表示方法构造的系统称为产生式系统，其是专家系统的基础框架，产生式系统的基本结构如图所示。



综合数据库：又称为事实库、工作内存，用来存放问题求解过程中信息的数据结构，包含初始状态、原始证据、推理得到的中间结论以及最终结论。

规则库：用于存放系统相关领域的所有知识的产生式。其对知识进行合理的组织与管理，如将规则分成无关联的子集。

控制系统：由一组程序组成的推理机，主要任务包括：

- 按一定的策略从规则库中选择规则，与综合数据库中的已知事实进行匹配，若匹配成功则启用规则，否则不使用此规则；
- 当匹配成功的规则多于一条时，使用冲突消解机制，选出一条规则执行；
- 执行规则后，将结果添加到综合数据库中，若后件是操作时执行操作；
- 确定系统执行停止的条件是否满足。

1.2.6 产生式表示法的特点

▌优点

✓ 自然性

符合人类因果表达习惯

支持直观推理过程

✓ 模块性

规则库、综合数据库、推理机，三分离架构

✓ 有效性

既可以表达确定知识，也可以表达不确定性知识

✓ 清晰性

固定知识格式规范

▌缺点

✕ 效率限制

规则匹配时间复杂度高

存在组合爆炸风险

✕ 结构性缺陷

难以表达层次化知识

关系型知识表示受限

1.3 知识的结构化表示

客观世界的问题一般由世界中的客体及其之间的关系构成。传统的程序语言把被动的数据或数据结构作为解空间的对象，程序设计人员需借助很复杂的算法或过程才能操纵解空间对象，从而求得问题的解，这导致基于知识的复杂软件的构造异常困难，且难以理解和维护。结构化的知识表示方法，能够表示和处理解空间的对象及其之间的关系。

1.3.1 框架表示的基本概念

1975 年明斯基(Minsky)提出框架理论，并将其应用于理解视觉和自然语言对话等方面的研究。框架表示的基本思路：当一个人遇到新的情况(或看待问题的观点发生实质变化)时，他会从记忆中选择一种结构，即“框架”。这是一种记忆下来的轮廓，按照需要改变其细节就可以用其拟合真实情况。

例如

框架名<硕士生>

姓名：单位(姓，名)

性别：范围(男，女)，默认：男

年龄：单位(岁)，条件：岁>16

学籍：(硕士学籍)

实例：

框架名<硕士生-1>

姓名：杨叶

性别：女

年龄：23 岁

学籍：<硕士学籍-1>

框架表示的优点有以下几方面：

(1) **结构性**：善于表示结构性知识；

(2) **深层性**：框架表示不仅可以从多个方面、多重属性表示知识，而且可以通过预定义槽表示知识的结构层次和因果关系，因此能表达事物间复杂深层联系；

(3) **继承性**：下层框架可以继承上层框架的槽值；

(4) **自然性**：模拟人类对实体的多方面、多层次的存储结构，直观自然，易于理解。

框架表示的缺点：

(1) **形式理论缺失**：缺乏框架的形式理论；

(2) **过程性知识不足**：缺乏过程性知识表示；

(3) **清晰性问题**：系统清晰性难以保证。

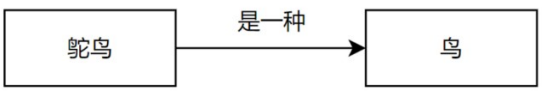
1.3.2 语义网络

语义网络是 1968 年奎廉(J.R.Quilian)在研究人类联想记忆时提出的一种心理学模型，认为记忆是由概念间的联系实现的。1972 年西蒙(H.A.Simon)将其用于自然语言理解系统。语义网络把知识表示为一种图，结点表示事实或概念，弧对应于概念间的关系和关联。其中，结点和弧必须带有标识，用来说明它所代表的实体或语义。

1. 语义单元

语义单元一般用三元组(结点 1，弧，结点 2)表示。

例如，三元组表示“驼鸟是一种鸟”如图所示。



2. 语义网络

把多个语义单元用相应的语义关联在一起时，就形成了一个语义网络。

<语义网络>::=<语义基元>|Merge(<语义基元>...)

<语义基元>::=<结点><语义联系><结点>

<结点>::=(<属性-值对>,...)

<属性-值对>::=<属性名>:<属性值>

<语义联系>::=<系统预定义语义联系>|<用户自定义语义联系>

3. 基本语义关系

(1) 类属关系

指具有共同属性的不同事物间的分类关系、成员关系或实例关系，体现具体与抽象、个体与集体的概念。

例：

A kind of：“是一种”，如石头是一种物质；

A member of：“是一员”，如张强是协会成员；

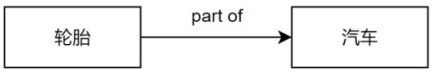
Is a：“是一个”，如刘翔是一个运动员。

(2) 包含关系

指具有组织或结构特征的部分与整体之间的关系。

例：

Part of：“是一部分”，如轮胎是汽车的一部分。



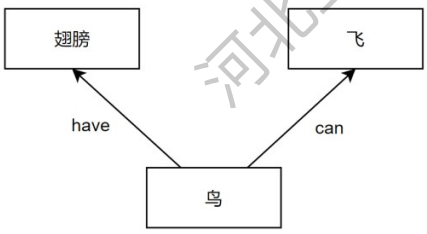
(3) 属性关系

指事物及其属性间的关系。

例：

Have：“有”，如鸟有翅膀；

Can：“会/能”，如鸟会飞。



(4) 相近关系

指不同事物在形状、内容等方面相似或接近。

(5) 推论关系

指从一个概念推出另一个概念的语义关系(类似产生式规则)。

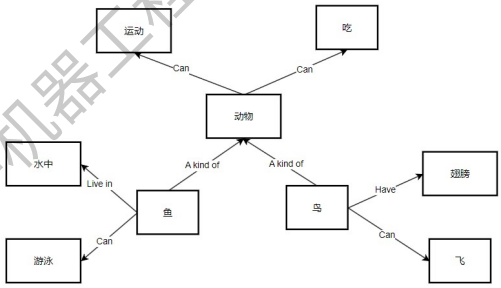
用语义网络表示情况、事件和动作时，通过增加情况、事件和动作结点来表示复杂语义。

(6) 逻辑关系表示：

通过增加合取(∧)和析取(∨)结点表示逻辑关系。

4. 语义关系示例

用语义网络表示：动物能运动，会吃；鸟是一种动物，有翅膀，会飞；鱼是一种动物，生活在水中，会游泳。语义网络如图所示：



5. 语义网络表示法的特征

语义网络表示法的优点有以下几方面：

(1) 结构性：能把事物的属性及其之间的各种语义联系显式表达出来；

(2) 联想性：根据人类记忆的心理模型提出；

(3) 自然性：自然语言与语义网络的转换易实现。

语义网络表示法的缺点有以下几方面：

(1) 非严格性：没有公认的形式表示体系；

(2) 复杂性：手段多、表示灵活，但由于形式的不一致增加了处理问题的复杂度。

1.4 状态空间表示法

状态空间表示法的基本思想是将待求解的问题通过状态和操作表示出来：

状态：表示问题过程中每一步问题状况的数据结构；

操作：也称为运算符，是将问题从一种状态变换为另一种状态的手段，它包括操作和操作的条件

状态空间是状态和操作的总称，一般表示为(S, O)，其中：

S 表示问题求解过程中所有可能的合法状态构成的集合

O 表示所有有效操作算子集合及操作的前提条件

求解问题的表达一般采用如下的三元组：$W = (SP, I, G)$，其中：

SP表示(S, O)问题的状态空间；

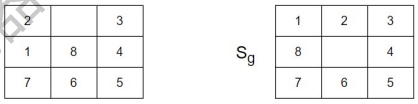
I表示问题的初始状态集合；

G表示问题的目标状态集合。

如果将问题的状态集合看作一个图中的结点，将操作集合看作是结点间的有向连线，则当用状态空间表示出一个问题后，就可以用图来表示，此图则称为状态空间图。因此状态空间搜索有时也称为图搜索。

例 4 八数码问题。

在 3×3 的方格棋盘上，分别放置了标有数字 1,2,3,4,5,6,7,8 的八张牌，初始状态为S_0，目标状态为S_g，如图所示。通过数码牌的移动找到一条从初始状态S_0到目标状态S_g的路径。



问题的表示：

状态$S[i][j]$是一个二维数组，i、j表示空格所处的行、列，其中 $1 \le i, j \le 3$，除i、j位置的元素取值集合为$\{1,2,\ldots,8\}$，问题的所有可能的状态集合S有 9!个元素。

问题的操作算子定义如下：

空格操作：i、j是空格所在的行、列

空格左移操作：IF $j-1 \ge 1$ THEN $S[i][j] = S[i][j-1], S[i][j-1] = 0$

空格上移操作：IF $i-1 \ge 1$ THEN $S[i][j] = S[i-1][j], S[i-1][j] = 0$

空格右移操作：IF $j+1 \le 3$ THEN $S[i][j] = S[i][j+1], S[i][j+1] = 0$

空格下移操作：IF $i+1 \le 3$ THEN $S[i][j] = S[i+1][j], S[i+1][j] = 0$

所以操作集合O有 4 个元素：问题的初始状态集合$I = \{S_0\}$，目标状态集合$G = \{S_g\}$，问题的状态空间$SP = (S, O)$，如此问题可以用$W = (SP, I, G)$表示给计算机。

至此，介绍了用符号将问题表达为计算机能够理解的符号系统，并利用介绍的表示方法将本章开始提出的几个问题表示出来。下面将讨论如何用计算机解决问题。

2. 确定性推理方法

在人工智能中，利用知识表示方法表达一个待求解的问题后，还需要利用推理和搜索技术来求解问题。问题的求解方法一般可以分为两类，即搜索和推理。确定性推理是指推理时所用的知识与证据都是确定的，推出的结论也是确定的，其值要么为真、要么为假，这里的证据是指已知事实或在推理过程中得到的结论。

本节主要介绍一般演绎推理和归结演绎推理这两种确定性推理方法。

2.1 一般演绎推理

推理是按某种策略由已知事实出发推出结论的思维过程，也常把这个过程称为证明过程。人工智能中的推理是由程序完成的，即计算机通过已知事实和知识推出结论。

2.1.1 演绎推理

演绎推理是指由一般到具体(个别)的推理，演绎推理一般是三段论：

- 大前提：已知的一般性知识或假设；
- 小前提：关于所研究的具体情况或个别事实的判断；
- 结论：由大前提推出的适合小前提所示情况的新判断。

例：

大前提：足球运动员身体是强壮的；

小前提：高波是足球运动员；

结论：高波的身体是强壮的。

通俗地讲，演绎推理就是由一组已知事实出发，直接使用经典逻辑的推理规则推出结论的过程。比如将已知事实表示为$S = F_1 \wedge F_2 \wedge \ldots \wedge F_n$，结论表示为$C_{12}$。$C_{12}$是公式，如果证明了谓词公式(命题公式)$S \to G$是永真的(真值为$T$)，则称$G$是$S$的逻辑结论，同时也完成了由已知证明结论的过程。

2.1.2 演绎推理的基础

1) 推理规则

前提引入(P)规则：可以随便使用前提。

结论引入(T)规则：可以随便使用前面演绎出的某些公式的逻辑结果。

CP规则：如需要由前提集合演绎出公式$G \to H$，则可将G作为附加前提使用，演绎出H。

2) 推理中常用的等价式

- **交换律**：$P \vee Q \Leftrightarrow Q \vee P$，$P \wedge Q \Leftrightarrow Q \wedge P$
- **结合律**：$(P \vee Q) \vee R \Leftrightarrow P \vee (Q \vee R)$，$(P \wedge Q) \wedge R \Leftrightarrow P \wedge (Q \wedge R)$
- **分配律**：$P \vee (Q \wedge R) \Leftrightarrow (P \vee Q) \wedge (P \vee R)$，$P \wedge (Q \vee R) \Leftrightarrow (P \wedge Q) \vee (P \wedge R)$
- **摩根定律**：$\neg(P \vee Q) \Leftrightarrow \neg P \wedge \neg Q$，$\neg(P \wedge Q) \Leftrightarrow \neg P \vee \neg Q$
- **双重否定律**：$\neg\neg P \Leftrightarrow P$
- **吸收律**：$P \vee (P \wedge Q) \Leftrightarrow P$，$P \wedge (P \vee Q) \Leftrightarrow P$
- **补余律**：$P \vee \neg P \Leftrightarrow T$，$P \wedge \neg P \Leftrightarrow F$
- **连词化归律**：$P \to Q \Leftrightarrow \neg P \vee Q$，$P \leftrightarrow Q \Leftrightarrow (P \wedge Q) \vee (\neg P \wedge \neg Q)$

- **量词转换律**：$\neg(\exists x)P \Leftrightarrow (\forall x)\neg P$，$\neg(\forall x)P \Leftrightarrow (\exists x)\neg P$
- **量词分配律**：$(\forall x)(P \wedge Q) \Leftrightarrow (\forall x)P \wedge (\forall x)Q$，$(\exists x)(P \vee Q) \Leftrightarrow (\exists x)P \vee (\exists x)Q$

3) 推理中常用的蕴含式

- **化简式**：$P \wedge Q \Rightarrow P$，$P \wedge Q \Rightarrow Q$
- **附加式**：$P \Rightarrow P \vee Q$，$Q \Rightarrow P \vee Q$
- **析取三段论**：$\neg P$，$P \vee Q \Rightarrow Q$
- **假言推理**：P，$P \rightarrow Q \Rightarrow Q$
- **拒取式**：$\neg Q$，$P \rightarrow Q \Rightarrow \neg P$
- **假言三段论**：$P \rightarrow Q$，$Q \rightarrow R \Rightarrow P \rightarrow R$
- **两难推论**：$P \vee Q$，$P \rightarrow R$，$Q \rightarrow R \Rightarrow R$
- **全称固化**：$(\forall x)P(x) \Rightarrow P(y)$，其中$y$是个体域中的任一个体
- **存在固化**：$(\exists x)P(x) \Rightarrow P(y)$，其中$y$是个体域中使$P(x)$为真的个体

例 若厂方拒绝增加工资，则罢工不会停止，除非罢工超过一年且工厂经理辞职。问：如果厂方拒绝增加工资，而罢工刚刚开始，罢工能否停止？

解：

设命题：

P：厂方拒绝增加工资

Q：罢工停止

R：工厂经理辞职

S：罢工超过一年

用以上定义的命题和逻辑连接符表示出文字中的逻辑语义和已知条件。

已知条件：P，$\neg S$

逻辑语义：$P \wedge \neg(S \wedge R) \rightarrow \neg Q$

推理过程：

$\neg(S \wedge R)$：使用了前提引入规则、德·摩根定律；

$P \wedge \neg(S \wedge R)$：使用了前提引入规则、结论引入规则；

$P \wedge \neg(S \wedge R)$，$P \wedge \neg(S \wedge R) \rightarrow \neg Q$：使用了前提引入规则、结论引入规则和假言推理

结论：罢工没有停止

例 设已知如下事实：

① 只要是需要编程的课，王程都喜欢；

② 所有程序设计语言课都是需要编程的课；

③ C 语言是一门程序设计语言课。

求证：王程喜欢 C 语言这门课。

解：

第一步：定义谓词。

Prog(x)：x 是需要编程的课

Like(x, y)：x 喜欢 y

Lang(x)：x 是一门程序设计语言课

第二步：把事实及待求解问题用谓词公式表示。

只要是需要编程的课，王程都喜欢

$$(\forall x)\text{Prog}(x) \rightarrow \text{Like}(\text{Wang},x) \tag{1}$$

所有程序设计语言课都是需要编程的课

$$(\forall \text{x})\big(\text{Lang}(\text{x}) \rightarrow \text{Prog}(\text{x})\big) \tag{2}$$

C 语言是一门程序设计语言课

$$\text{Lang}(\text{C}) \tag{3}$$

第三步：应用规则推理。

由式(2)，利用置换{C/x}，可以得到下式：

$$\text{Lang}(\text{C}) \rightarrow \text{Prog}(\text{C}) \tag{4}$$

由式(3)和式(4)利用假言推理规则 Lang(C)，Lang(C) → Prog(C)，可以推得 Prog(C)为真。式 (1) 经过置换 $\{C/x\}$ 得 Prog(C) → Like(Wang,C)，结合 Prog(C) 为真和假言推理得 Like(Wang,C)为真。

结论：此谓词公式的语义为王程喜欢 C 语言这门课，所以王程喜欢 C 语言这门课。

2.2 归结演绎推理

人工智能中的问题求解几乎都可以转化为定理证明问题。定理证明的实质是从公式集$S = \{P_1, P_2, \ldots, P_n\}$出发推出结论$G$，即需要验证蕴含式永真$(P_1 \wedge P_2 \wedge \ldots \wedge P_n) \rightarrow G$。蕴含式永真不容易验证，可将其转化为不可满足性，即将验证$(P_1 \wedge P_2 \wedge \ldots \wedge P_n) \rightarrow G$永真转化为验证$(P_1 \wedge P_2 \wedge \ldots \wedge P_n) \wedge \neg G$不可满足。引入两个新的概念——子句、子句集，并定义不包含任意文字的子句称为空子句。空子句是永假的、不可满足的，一般记为 NIL 或□。

原子或原子的否定称为文字，有限个文字的析取式称为一个子句，由子句和空子句组成的集合称为子句集。在谓词逻辑中，任何一个谓词公式都可以通过应用等价关系及推理规则化成相应的子句集。子句集的特点：无量词约束；每个元素(子句)只是文字的析取；否定符只作用于单个文字；子句间默认为合取。下面介绍将一个谓词公式化成相应的子句集的过程。

2.2.1 化子句集的步骤

步骤 1：消去连接词“→”和“↔”

反复使用如下等价式：$P \to Q \Leftrightarrow \neg P \wedge Q$，$P \leftrightarrow Q \Leftrightarrow (\neg P \wedge \neg Q) \vee (P \wedge Q)$，即可消去连接词“→”和“↔”。

例如：

$$(\forall x)\Big((\forall y)(P(x,y) \to \neg(\forall y)\big(Q(x,y) \to R(x,y)\big)\Big)$$

$$\neg(\forall y)\big(Q(x,y) \to R(x,y)\big) \Leftrightarrow \neg(\forall y)\big(\neg Q(x,y) \vee R(x,y)\big)$$

$$(\forall x)\Big((\forall y)(P(x,y) \to \neg(\forall y)\big(Q(x,y) \to R(x,y)\big)\Big)$$
$$\Leftrightarrow (\forall x)\Big(\neg(\forall y)P(x,y) \vee \neg(\forall y)\big(\neg Q(x,y) \vee R(x,y)\big)\Big)$$

步骤 2：减少否定符号的辖域

反复使用：德·摩根定律：$\neg(P \vee Q) \Leftrightarrow \neg P \wedge \neg Q$，$\neg(P \wedge Q) \Leftrightarrow \neg P \vee \neg Q$；双重否定律：$\neg\neg P \Leftrightarrow P$；量词转换律：$\neg(\exists x)P \Leftrightarrow (\forall x)\neg P$，$\neg(\forall x)P \Leftrightarrow (\exists)x\neg P$；

接上例：

$$(\forall x)\Big((\exists y)\neg P(x,y) \vee (\exists y)\big(Q(x,z) \wedge \neg R(x,z)\big)\Big)$$

步骤 3：变元标准化

在一个量词的辖域内，把谓词公式中受该量词约束的变元全部用另外一个没有出现过的变元代替，使不同量词约束的变元的名称不同。

接上例：

$$(\forall x)\Big((\exists y)\neg P(x,y) \vee (\exists z)\big(Q(x,z) \wedge \neg R(x,z)\big)\Big)$$

步骤 4：化为前束范式

将所有量词移至谓词公式的最前面。

接上例：

$$(\forall x)(\exists y)(\exists z)\Big(\neg P(x,y) \vee \big(Q(x,z) \wedge \neg R(x,z)\big)\Big)$$

步骤 5：消去存在量词

用一个 Skolem 函数代替每一个出现存在量词的约束变量。

接上例：

$$\forall x\Big(\neg P\big(x,f(x)\big) \vee \big(Q\big(x,g(x)\big) \wedge \neg R\big(x,g(x)\big)\big)\Big)$$

步骤 6：化 Skolem 标准形

接上例：

$$\forall x\Big(\big(\neg P\big(x,f(x)\big) \vee Q\big(x,g(x)\big)\big) \wedge \big(\neg P\big(x,f(x)\big) \vee \neg R\big(x,g(x)\big)\big)\Big)$$

步骤 7：消去全称量词

接上例：

$\left(\neg P\left(x,f(x)\right)\vee Q\left(x,g(x)\right)\right)\wedge\left(\neg P\left(x,f(x)\right)\vee\neg R\left(x,g(x)\right)\right)$

步骤 8：消去合取词

接上例：

$\neg P\left(x,f(x)\right)\vee Q\left(x,g(x)\right),\neg P\left(x,f(x)\right)\vee\neg R\left(x,g(x)\right)$

步骤 9：更换变元名称，得到谓词公式的标准子句集

接上例：

$\{\neg P\left(x,f(x)\right)\vee Q\left(x,g(x)\right),\neg P\left(f(y),f(y)\right)\vee\neg R\left(y,g(y)\right)\}$

2.2.2 鲁滨逊归结原理

1. 基本思想

要证明$(P_1\wedge P_2\wedge\ldots\wedge P_n)\wedge\neg G$不可满足，首先将谓词公式$(P_1\wedge P_2\wedge\ldots\wedge P_n)\wedge\neg G$转化为子句集$S$，然后证明子句集$S$不可满足。子句集$S$不可满足可以通过后面定义的归结运算找出空子句，而空子句是不可满足的。

2. 命题逻辑归结

定义归结运算：设C_1和C_2是子句集 S 中的任意两个子句，如果子句C_1中的文字L_1与C_2中的文字L_2互补，则从C_1和C_2中分别消去L_1和L_2，并将两个子句余下的部分析取构成一个新子句C_{12}，这一过程称为归结，C_{12}为C_1和C_2的归结式，C_1和C_2为C_{12}的亲本子句。可以证明归结式C_{12}是其亲本子句C_1和C_2的逻辑结论。

下面不加证明地给出在归结演绎中要用的几个结论：

结论 1：设C_1和C_2是子句集S中的两个子句，C_{12}是C_1和C_2的归结式，用C_{12}代替C_1和C_2后构成新子句集S_1，则由S_1的不可满足性可以推出S的不可满足性，即S_1不可满足$\Rightarrow S$不可满足。

结论 2：设C_1和C_2是子句集S中的两个子句，C_{12}是C_1和C_2的归结式，将C_{12}添加到S中构成新子句集S_2，则S_2的不可满足性与S的不可满足性等价，即S_2不可满足$\Leftrightarrow S$不可满足。

结论 3：子句集S是不可满足的，充要条件是存在一个从S到空子句的归结过程。(归结的完备性)

应用归结原理证明定理的过程称为归结反演。已知 F，证明 G 的归结反演过程如下：

① 否定目标公式G，得$\neg G$；

② 把$\neg G$并入到公式集F中，得$\{F,\neg G\}$；

③ 把$\{F,\neg G\}$化为子句集S；

④ 应用归结原理对S中的子句进行归结并将归结式并入S，反复进行，直到出现空子句。

例 设已知公式集为$\{P,(P\wedge Q)\rightarrow R,(S\vee T)\rightarrow Q,T\}$，求证$R$。

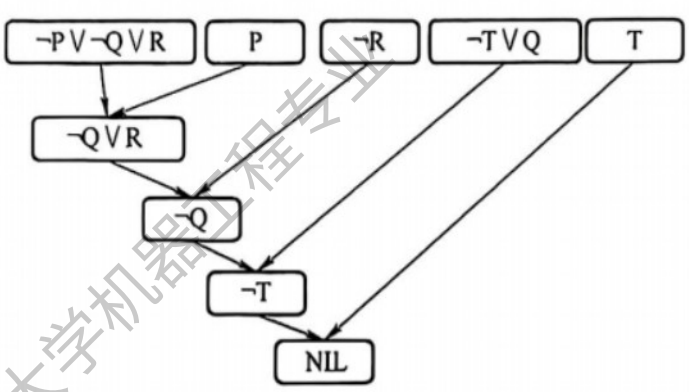
解：设 R 为假并加入已知公式集得

$$\{P,(P\wedge Q)\rightarrow R,(S\vee T)\rightarrow Q,T,\neg R\}$$

化子句集得子句集 F

$$F=\{P,\neg P\vee\neg Q\vee R,\neg S\vee Q,\neg T\vee Q,T,\neg R\}$$

对子句集 F 归结，归结演绎树过程如图所示。



3. 谓词逻辑归结

设C_1和C_2是两个没有公共变元的子句，L_1和L_2分别是C_1和C_2中的文字，如果L_1和$\neg L_2$存在最一般合一σ，则称$C_{12}=(C_1\sigma-\{L_1\sigma\})\cup(C_2\sigma-\{L_2\sigma\})$为$C_1$和$C_2$的归结式。

例 $C_1=P(a)\vee R(x)$，$C_2=\neg P(y)\vee Q(b)$，求C_{12}。

解：取$L_1=P(a)$，$L_2=\neg P(y)$

最一般合一 $\sigma=\{a/y\}$

$$\begin{aligned}C_{12}&=(\{P(a),R(x)\}-\{P(a)\})\cup(\{\neg P(a),Q(b)\}-\{\neg P(a)\})\\&=\{R(x),Q(b)\}\\&=R(x)\vee Q(b)\end{aligned}$$

谓词的归结与命题的归结相比多了置换和合一，所以谓词的归结多了以下内容。设C_1和C_2是两个没有公共变元的子句，谓词的归结包括如下几种情况：

① C_1和C_2的二元归结式；

② C_1和C_2的因子$C_2\sigma$的二元归结式；

③ C_1的因子$C_1\sigma$和C_2的二元归结式；

④ C_1的因子$C_1\sigma$和C_2的因子$C_2\sigma$的二元归结式。

以上四种归结式都是C_1和C_2的归结式，记为C_{12}。

谓词逻辑归结反演与命题逻辑归结反演过程基本相同，具体有：

① 否定目标公式G，得$\neg G$；

② 把$\neg G$并入到公式集F中，得$\{F,\neg G\}$；

③ 把$\{F,\neg G\}$化为子句集S；

④ 应用归结原理对S中的子句进行归结并将归结式并入S，反复进行，直到出现空子句。

例 “快乐学生”问题。

假设：任何通过计算机考试并获奖的人都是快乐的；任何肯学习或幸运的人都可以通过所有考试；李不肯学习但他是幸运的；任何幸运的人都能获奖。

求证：李是快乐的。

解：

第一步：定义谓词。

$Happy(x)$：x快乐

$Study(x)$：x肯学习

$Lucky(x)$：x幸运

$Win(x)$：x获奖

$Pass(x,y)$：x通过考试y

第二步：用谓词表示问题。

任何通过计算机考试并获奖的人都是快乐的：

$$(\forall x)\big(Pass(x,Computer)\wedge Win(x)\rightarrow Happy(x)\big) \tag{5}$$

任何肯学习或幸运的人都可以通过所有考试：

$$(\forall x)(\forall y)\big(Study(x)\vee Lucky(x)\rightarrow Pass(x,y)\big) \tag{6}$$

李不肯学习但他是幸运的：

$$\neg Study(Li)\wedge Lucky(Li) \tag{7}$$

任何幸运的人都能获奖：

$$(\forall x)\big(Lucky(x)\rightarrow Win(x)\big) \tag{8}$$

结论“李是快乐的”之否定：

$$\neg Happy(Li) \tag{9}$$

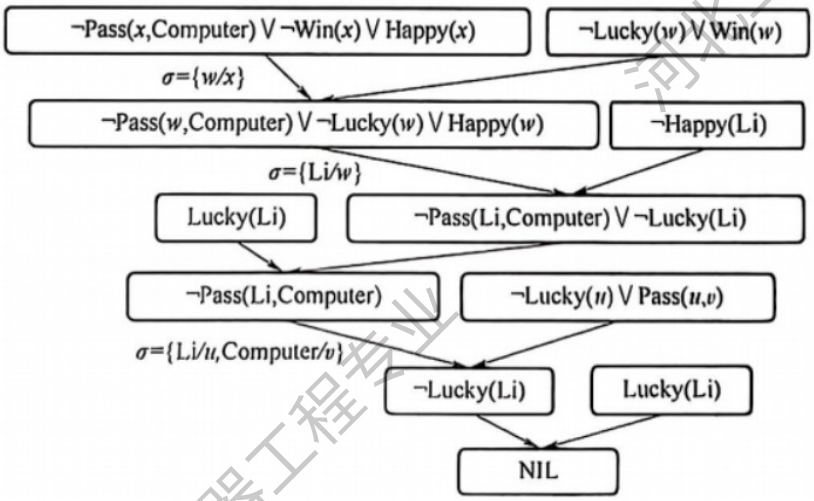
第三步：将式(5)～式(9)化为子句集，应用连词化归律去掉式(5)中的蕴含符号，得到谓词公式 $\neg Pass(x,Computer)\vee\neg Win(x)\vee Happy(x)$，得到一个子句。类似地，式(6)有：

$$(\forall x)(\forall y)\big(Study(x)\vee Lucky(x)\rightarrow Pass(x,y)\big)\Leftrightarrow(\forall x)(\forall y)\Big(\big(\neg Study(x)\wedge\neg Lucky(x)\big)\vee Pass(x,y)\Big) \tag{10}$$

得到两个子句$\{Study(y)\vee Pass(y,z),Lucky(u)\vee Pass(u,v)\}$。照此处理得到问题的子句集为：

$$\{\neg Pass(x,Computer)\vee\neg Win(x)\vee Happy(x),Study(y)\vee Pass(y,z),Lucky(u)\vee Pass(u,v),\neg Study(Li),Lucky(Li),Lucky(w)\vee Win(w),\neg Happy(Li)\}$$

归结过程见“快乐学生”问题的归结反演树，如图所示。



3. 不确定性知识表示方法与推理

本章的 4.1 节与 4.2 节介绍了确定性知识表示和推理的方法，本节将简单介绍不确定性知识如何表示以及建立在不确定性知识和证据基础上的“不确定性推理”的几种方法。

不确定性知识的表示方法主要有概率方法(也叫作概率模型)以及模糊集方法两大类。

3.1 为何要讨论不确定性推理？

人工智能主要是模拟人类的智能行为，而人类推理主要的形式是不确定性推理，因为人们在做决策时大部分是在信息不完全或者证据和结果不确定现象的情况下进行的，如掷骰子。因此，要很好地模拟人类智能，就一定要能够处理这些不确定信息的情形。

本节将介绍：

(1) 如何使用概率方法表示不确定性知识；

(2) 概率推理基本框架；

(3) 一种类概率的表示和推理方法——证据理论

3.2 概率表示及推理方法

概率的基本概念大部分人在中学时已经初步接触过，其是一种具有严格数学基础、处理具有统计规律的不确定现象的数学工具。

概率是用数学的方法描述一个可观察结果的人工或自然的过程，其产生的结果可能不止一个，且不能事先确定会产生什么结果，将这个过程称为**随机试验**。

随机试验的全部可能结果的集合称为**样本空间** Ω，其中的点称为**样本点**或**原子事件**。

样本空间有时也称为原子事件的集合，其有两个性质：(1) **互斥性**：同时成立的原子事件只能有一个；(2) **真实性**：至少有一个原子事件发生。

例如掷硬币，其样本空间$\Omega = \{$正面,反面$\}$，每一次出现的结果只能是“正面”或“反面”，而且每次试验必有一个结果出现。

通过一个简单的例子来介绍概率论中的基本概念。假设掷两个骰子，其编号为 1#和 2#，每个骰子的可能取值为集合 $\Omega = \{1,2,3,4,5,6\}$，定义两个随机变量 X、Y 分别表示 1#和 2#骰子的投掷结果。这两个随机变量的样本空间是集合 Ω，即变量所有的可能取值。表 2 和表 3 分别表示 X、Y 这两个变量的取值规律。

表 2 随机变量X的概率分布

X =	1	2	3	4	5	6
p	1/6	1/6	1/6	1/6	1/6	1/6

表 3 随机变量Y的概率分布

Y =	1	2	3	4	5	6
p	1/12	1/12	1/6	1/6	1/6	1/3

这两个表称为随机变量X、Y的概率分布，表示了随机试验后随机变量X、Y取某个值的可能性。第一个随机变量表示 1#骰子，其概率分布是正常的，第二个随机变量表示 2#骰子是异常的。两个随机变量的联合概率分布记为$P(X,Y)$，它是 36 个样本点的概率，下面介绍几个基本概念。

随机事件A：随机试验的一些可能结果的集合是样本空间的一个子集，子集中任何一个样本发生都称为随机事件A发生。

随机事件的概率$P(A)$：随机事件A出现的可能性，集合中所有样本点的概率之和，称之为A的无条件概率或先验概率。

例如，随机事件A是随机变量X为奇数的集合，随机事件B是随机变量Y为偶数的集合，则随机事件A的概率为$P(A) = 0.5$，随机事件B的概率为$P(B) = \frac{7}{12}$。

条件概率$P(A|B)$：随机事件B出现的条件下事件A出现的可能性，称之为A在条件B下的条件概率或后验概率。

随机变量按样本空间可以分成如下类型：

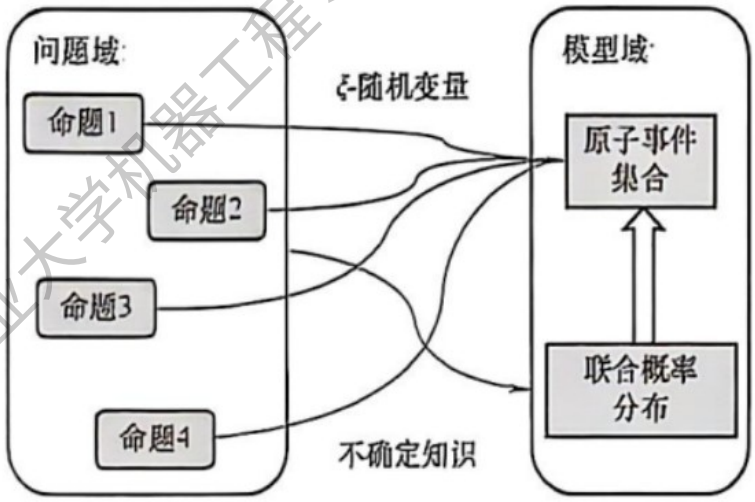
(1) **布尔型随机变量**：样本空间仅有两个样本，一般样本空间表示为$\{T,F\}$；

(2) **离散型随机变量**：取值于一个有限或可数集合；

(3) **连续型随机变量**：取值于一个区间或连续空间。

由多个随机变量组成的向量称为**随机向量**，如二维随机向量 (X,Y)。随机变量的特征是由联合概率分布描述的，如随机向量 (X,Y) 的样本空间 $\Omega = \{(1,1),(1,2),...,(1,6),(2,1),...,(2,6),...,(6,6)\}$，共有 36 个原子样本，联合概率分布类似表 2、表 3，由每个原子样本的概率给出。

基于概率的不确定表示可以视为将确定性推理中的命题(一阶谓词)看成随机变量(概率论中的基本单元)，问题中的所有随机变量构成的随机向量看成问题描述的基础，随机向量的联合概率分布是推理的基础知识。基于概率的不确定知识表示如图所示。

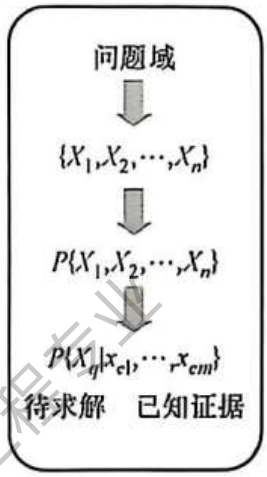


推理的基本思路：

(1) 把随机变量根据问题的需要分成三部分：查询变量、证据变量、隐变量；

(2) 根据已经观察到的证据计算查询命题的后验概率，而联合概率分布作为“知识库”。

基本思路如图所示。



在使用概率方法表示不确定性知识的基础上，使用联合概率分布进行推理一般来说可以使用以下推理过程：

(1) X：查询变量；

(2) E：证据变量集合，e表示其观察值；

(3) Y：未观测变量集合，表示其一个具体的赋值。

即：

$$P(x|E=e)=\sum_{y} P(x, E=e, y)$$

但是，在使用联合概率分布进行推理时需要注意的是，如果问题域由n个布尔型随机变量描述，则问题的原子样本集合有2^n个样本，需要2^n个数据构成联合概率分布，作为知识库。在实际问题中实现较为困难，为解决计算复杂度高的问题，贝叶斯网络(Bayesian Network)、条件随机场等技术被提出。

3.3 证据理论

证据理论(Evidential Theory)的名称有很多，如 Dempster-Shafer 理论、Dempster-Shafer 证据理论和 DS(或 D-S)理论。证据理论诞生于 20 世纪 60 年代，由美国哈佛大学数学家 A.P.Dempster 在利用上、下限概率来解决多值映射问题方面的研究工作中提出。

之后，Dempster 的学生 G.Shafer 对证据理论做了进一步的发展，引入信任函数概念，形成了一套基于”证据”和”组合”来处理不确定性推理问题的数学方法，并于 1976 年出版了《证据的数学理论》(*A Mathematical Theory of Evidence*)，这标志着证据理论正式成为一种处理不确定性问题的完整理论。

由于在证据理论中需要的先验数据比概率推理理论中的更为直观、更容易获得，再加上Dempster合成公式可以综合不同专家或数据源的知识或数据，这使得证据理论在专家系统、信息融合等领域中得到了广泛应用。

1. 概率分配函数

DS理论处理的是集合上的不确定性问题，为此需要先建立命题与集合之间的一一对应关系。概率分配函数是一种把一个有限集合的幂集映射到[0,1]区间的函数，以把命题的不确定性问题转化为集合的不确定性问题。

(1) **幂集**

设Ω为样本空间，且Ω中的每个元素都相互独立，则由Ω的所有子集构成的幂集记为2^{Ω}。当Ω中的元素个数为N时，则其幂集2^{Ω}的元素个数为2^{N}，且其中的每一个元素都对应于一个关于取值情况的命题。

(2) **概率分配函数(mass函数)**

设函数$m: 2^{\Omega} \to [0,1]$，且满足：

$$m(\emptyset) = 0, \quad \sum_{A \subseteq \Omega} m(A) = 1$$

则称m是2^{Ω}上的概率分配函数，$m(A)$称为A的基本概率数。

需要注意的是：

概率分配函数的作用是把Ω的任一子集映射为[0,1]上的一个数$m(A)$。

当$A \subset \Omega$，且A由单个元素组成时，则$m(A)$表示对A的精确信任度。

当$A \subset \Omega$、$A \neq \Omega$，且A由多个元素组成时，$m(A)$也表示对A的精确信任度，但却不知这部分信任度该分给A中的哪些元素。

当$A = \Omega$时，则$m(A)$也表示不知该如何分配。

注意：概率分配函数不是概率。

(3) **一个特殊的概率分配函数**

设$\Omega = \{s_1, s_2, ..., s_n\}$，$m$为定义在$2^{\Omega}$上的概率分配函数，且$m$满足：

1) $m(\{s_i\}) \geq 0$，对任何$s_i \in \Omega$；
2) $\sum_{i=1}^{n} m(|s_i|) \leq 1$；
3) $m(\Omega) = 1 - \sum_{i=1}^{n} m(\{s_i\})$；
4) 当$A \subset \Omega$且$|A| > 1$或$|A| = 0$时，$m(A) = 0$。

其中，$|A|$表示命题A所对应的集合中的元素个数。

该概率分配函数的特殊性体现在：

只有当子集中的元素个数为 1 时，其概率分配数才有可能大于 0；

当子集中有多个或 0 个元素，且不等于全集时，其概率分配数均为 0；

全集的概率分配数按条件 3 计算。

(4) **概率分配函数的合成**

设m和n是两个不同的概率分配函数，其正交和$m \oplus n$满足：

1) $m \oplus n(\emptyset) = 0$；

2) $m \oplus n(A) = K \sum_{X \cap Y = A} m(X) \cdot n(Y)$

其中，$K = 1 - \sum_{X \cap Y = \emptyset} m(X) \cdot n(Y)$。

2. 信任函数和似然函数

根据上述特殊的概率分配函数定义，对任何命题$A \subseteq \Omega$，其信任函数为：

$$Bel(A) = \sum_{s_i \in A} m(\{s_i\})$$

特别地，对于全集Ω：

$$Bel(\Omega) = \sum_{B \subseteq \Omega} m(B) = \sum_{i=1}^{n} m(\{s_i\}) + m(\Omega) = 1$$

信任函数也称为下限函数，表示对A的总体信任度。

对任何命题$A \subseteq \Omega$，其似然函数为：

$$\begin{aligned} Pl(A) &= 1 - Bel(\neg A) \\ &= 1 - \sum_{s_i \notin A} m(\{s_i\}) \\ &= 1 - [1 - m(\Omega) - Bel(A)] \\ &= m(\Omega) + Bel(A) \end{aligned}$$

特别地，对于全集Ω：

$$Pl(\Omega) = 1 - Bel(\neg\Omega) = 1 - Bel(\emptyset) = 1$$

似然函数也称为上限函数，表示对A的非假信任度。

3. 类概率函数

设Ω为有限域，对任何命题$A \subseteq \Omega$，命题A的类概率函数为：

$$f(A) = Bel(A) + \frac{|A|}{|\Omega|}[Pl(A) - Bel(A)]$$

其中，$|A|$和$|\Omega|$分别是A及Ω中元素的个数。类概率函数$f(A)$具有如下性质：

1) $\sum_{i=1}^{n} f(\{s_i\}) = 1$

2) $\forall A \subseteq \Omega$，有 $Bel(A) \leq f(A) \leq Pl(A)$

3) $\forall A \subseteq \Omega$，有 $f(\neg A) = 1 - f(A)$

3.4 证据理论的推理模型

(1) 知识不确定性的表示

$$\text{IF } E \text{ THEN } H = \{h_1, h_2, \ldots, h_n\} \text{ CF} = \{c_1, c_2, \ldots, c_n\}$$

其中：

E为前提条件，它既可以是简单条件，也可以是用合取或析取词连接起来的复合条件

H是结论，用样本空间中的子集表示，$h_1, h_2, \ldots, h_n$是该子集中的元素

CF 是可信度因子，用集合形式表示，该集合中的元素$c_1, c_2, \ldots, c_n$用来指出$h_1, h_2, \ldots, h_n$的可信度c_i与h_i一一对应，并且c_i应满足如下条件：

$$c_i \geq 0, \quad i = 1,2, \ldots, n; \quad \sum_{i=1}^{n} c_i \leq 1$$

(2) 证据不确定性的表示

设A是规则条件部分的命题，E'是外部输入的证据和已证实的命题，在证据E'的条件下，命题A与证据E'的匹配程度为：

$$\text{MD}(A|E') = \begin{cases} 1, & \text{如果}A\text{的所有元素都出现在}E\text{中} \\ 0, & \text{否则} \end{cases}$$

条件部分命题A的确定性为：

$$\text{CER}(A) = \text{MD}(A|E') \times f(A)$$

其中，$f(A)$为类概率函数。由于$f(A) \in [0,1]$，因此 $\text{CER}(A) \in [0,1]$。

(3) 组合证据不确定性的表示

当组合证据是多个证据的合取时，即$E = E_1 \text{ AND } E_2 \text{ AND} \ldots \text{ AND } E_n$时，则：

$$\text{CER}(E) = \min\{\text{CER}(E_1), \text{CER}(E_2), \ldots, \text{CER}(E_n)\}$$

当组合证据是多个证据的析取时，即$E = E_1 \text{ OR } E_2 \text{ OR} \ldots \text{ OR } E_n$时，则：

$$\text{CER}(E) = \max\{\text{CER}(E_1), \text{CER}(E_2), \ldots, \text{CER}(E_n)\}$$

(4) 不确定性的更新

设有知识 $\text{IF } E \text{ THEN } H = \{h_1, h_2, \ldots, h_n\}$ $\text{CF} = \{c_1, c_2, \ldots, c_n\}$，则求结论$H$的确定性 $\text{CER}(H)$的方法如下：

1) 求H的概率分配函数：

$$m(\{h_1\}, \{h_2\}, \ldots, \{h_n\}) = (\text{CER}(E) \times c_1, \text{CER}(E) \times c_2, \ldots, \text{CER}(E) \times c_n)$$

$$m(\Omega) = 1 - \sum_{i=1}^{n} \mathrm{CER}\,(E) \times c_i$$

2) 如果有两条或多条知识支持同一结论H，例如：

$$\text{IF } E_1 \text{ THEN } H = \{h_1, h_2, ..., h_n\} \text{ CF} = \{c_{11}, c_{12}, ..., c_{1n}\}$$

$$\text{IF } E_2 \text{ THEN } H = \{h_1, h_2, ..., h_n\} \text{ CF} = \{c_{21}, c_{22}, ..., c_{2n}\}$$

则按正交和求$\mathrm{CER}(H)$，即先求出：

$$m_1 = m_1(\{h_1\}, \{h_2\}, ..., \{h_n\})$$

$$m_2 = m_2(\{h_1\}, \{h_2\}, ..., \{h_n\})$$

然后再用公式$m = m_1 \oplus m_2$求m_1和m_2的正交和，最后求得H的m。

3) 求$Bel(H)$、$Pl(H)$及$f(H)$：

$$Bel(H) = \sum_{i=1}^{n} m\,(\{h_i\})$$

$$Pl(H) = 1 - Bel(\neg H)$$

$$f(H) = Bel(H) + \frac{|H|}{|\Omega|}[Pl(H) - Bel(H)] = Bel(H) + \frac{|H|}{|\Omega|} \times m(\Omega)$$

4) 求H的确定性$\mathrm{CER}(H)$：

$$\mathrm{CER}(H) = \mathrm{MD}\left(H|E'\right) \times f(H)$$

可以看出，证据理论的优点在于它能够满足比概率更弱的公理系统并能够处理由“不知道”引起的不确定性。但同时证据理论要求Ω中的元素是互斥的，这一点在许多应用领域中难以做到，并且证据理论需要给出的概率分配数过多。

4. 问题求解

4.1 一般图搜索

问题求解是人类智能的一个基本功能，而一般问题求解都可以转化为一个以状态空间表示的搜索过程。搜索是人工智能中的一个基本问题，也是推理不可分割的一部分，因此尼尔逊将其列为人工智能研究的核心问题之一。

所谓搜索就是找到能够达到所希望目标状态的动作序列的过程。在状态空间描述下，搜索就是通过搜索引擎寻找一个操作算子的调用序列，使问题从初始状态变迁到目标状态之一，而状态变迁序列及相应的操作算子调用序列称为从初始状态到目标状态的解。

若要把状态集合中的所有元素都画出，计算机的资源需求过大，所以搜索的执行思路是从初始状态出发，每次只将当前状态可以到达的状态画出，随着搜索过程的进展逐步画出搜索图。

4.1.1. 状态空间图搜索的术语

State(状态)：搜索树/图中与该结点相对应的状态。

Parent-Node(父结点)：搜索树/图中产生该结点的结点。

Action(操作)：由父结点产生该结点的操作，结点扩展是指应用操作算子将上一状态结点n_i变迁到下一状态结点n_j，n_i是被扩展结点，n_j是n_i的子结点。

Depth(结点深度)：搜索图是一个有根图，根结点是初始状态，记其结点深度为 0，其他结点深度定义为其父结点深度d_n加 1：$d_n = d_{n-1} + 1$。

Path(路径)：从结点n_i到结点n_{i+1}的路径是由相邻结点间的弧线构成的折线，通常要求路径是无环的。

Path-Cost(路径代价)：相邻结点间路径代价为$C(n_i, n_{i+1})$，路径(n_0, n_k)的代价为$\sum C(n_i, n_{i+1})$。

CLOSED 表：用于存放已经扩展或将要扩展的结点。CLOSED 表的一般形式如图所示。

编号	状态结点	父结点

OPEN 表：用于存放刚生成的结点，由于这些结点还没有扩展，也称为未扩展结点表。OPEN 表的一般形式如图所示。

状态结点	父结点

4.1.2 状态空间一般图搜索过程

步骤 1： 初始化：

产生一个仅有初始结点S_0的 OPEN 表；

建立一个仅有初始结点S_0的图G。

步骤 2： 创建空表：

产生一个空的 CLOSED 表。

步骤 3： 检查终止条件：

如果 OPEN 表为空，则失败退出。

步骤 4： 选择结点：

在 OPEN 表的首部取一个结点 n；

将其放入 CLOSED 表；

在 OPEN 表删除结点 n。

步骤 5： 目标检测：

若 n ∈ Goal，则成功退出；

解为图 G 中沿指针从 n 到S_0的路径。

步骤 6： 扩展结点：

扩展结点 n 生成一切后继结点；

将不是结点 n 的先辈结点集合记成 M；

将 M 作为结点 n 的后继结点装入图 G 中。

步骤 7： 处理新结点：

对没有在图 G 中出现的新结点 p；

设置一个指向父结点的指针；

放入 OPEN 表中；

若 p ∈ G(即结点 p 出现在 OPEN 表或 CLOSED 表中)；

根据具体情况决定是否改变图 G 中相关结点的指针。

步骤 8： 重新排序：

对 OPEN 表中的结点按某种原则重新排序。

步骤 9： 循环：

转步骤 3。

关于步骤 7 的说明

1. 结点*p*在 OPEN 表中：

说明结点 p 在结点 n 之前已经是某一个结点 m 的子结点，但本身并没有被考察(即没有生成 p 的后继结点)。这表明从S_0到结点 p 至少存在两条路径，指向父结点的指针按路径代价较小确定。

2. 结点 p 在 CLOSED 表中：

说明结点 p 在结点 n 之前已经是某一个结点 m 的子结点，指向父结点的指针按路径代价较小确定。结点 p 已经生成后继结点：对后继结点指向父结点的指针按路径代价确定。

一般图搜索算法的伪代码如图所示。

① G：=G_0(G_0=s)，OPEN：=(s)；
② CLOSED：=()；设置 CLOSED 表，起始设置为空表
③ LOOP：IF OPEN=()，THEN EXIT(FAIL)；
④ n：=FIRST(OPEN)，REMOVE(n，OPEN)，ADD(n，CLOSED)；称 n 为当前结点
⑤ IF GOAL(n)，THEN EXIT(SUCCESS)；由 n 返回到 s 路径上的指针，可给出解路径
⑥ EXPAND(n)→{m_e}，G：=ADD(m_k，G)；子结点集{m_k}中不包含 n 的父辈结点
⑦ 标记和修改指针：
ADD(m_n，OPEN)；并标记 m_n 连接到 n 的指针，m_n 为 OPEN 和 CLOSED 中未出现过的子结点，$\{m_k\}=\{m_n\}\cup\{m_o\}\cup\{m_c\}$
计算是否要修改 m_o、m_c 到 n 的指针；m_o 为已出现在 OPEN 中的子结点，m_c 为已出现在 CLOSED 中的子结点
计算是否要修改 m_c 到其后继结点的指针；
⑧ 对 OPEN 中的结点按某种原则重新排序；
⑨ GO LOOP；

4.2 盲目搜索

一般图搜索提高效率的关键是优化 OPEN 表中结点的排序方式。若每次排在表首的结点都在最终搜索到的解路径上，则算法不会扩展任何多余结点就可以快速结束搜索。因此，不同的排序方式形成多种搜索策略。

4.2.1 广度优先搜索

广度优先搜索(Breadth-First Search，BFS)也称为宽度优先搜索。这种搜索策略的搜索过程是从初始结点S_0开始逐层向下扩展，在第n层没有扩展完成之前，不进行下一层结点的搜索。OPEN 表中的结点总是按产生的先后排序，先产生的结点排在 OPEN 表的前面，后产生的结点排在 OPEN 表的后面。

例 八数码问题：

在 3×3 的方格棋盘上，分别放置了标有数字 1,2,3,4,5,6,7,8 的八张牌，初始状态为S_0，目标状态为S_g。如下图所示，可以使用的操作有：

- 空格左移
- 空格右移
- 空格上移
- 空格下移

应用广度优先搜索策略寻找从初始状态到目标状态的路径。

S_0

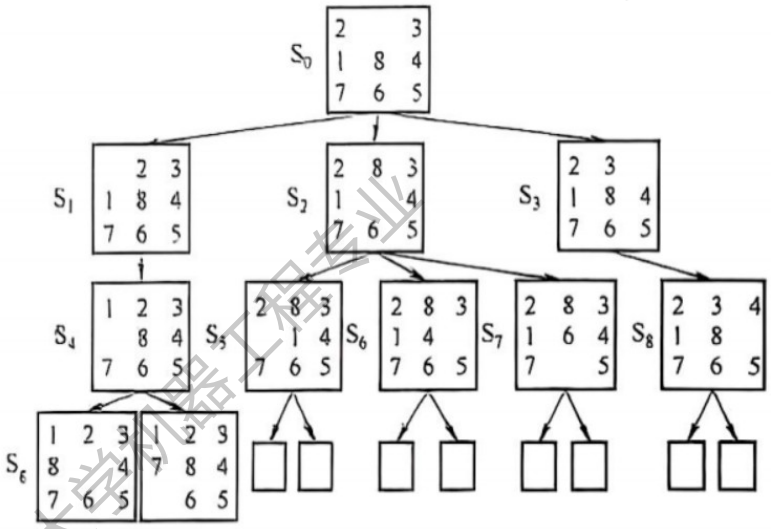
2		3
1	8	4
7	6	5

S_g

1	2	3
8		4
7	6	5

解：

按照一般图搜索算法，OPEN 表中的元素采用先进先出(FIFO)的排序方式，得到的广度优先搜索图如图所示。



广度优先搜索的优缺点如下：

优点：

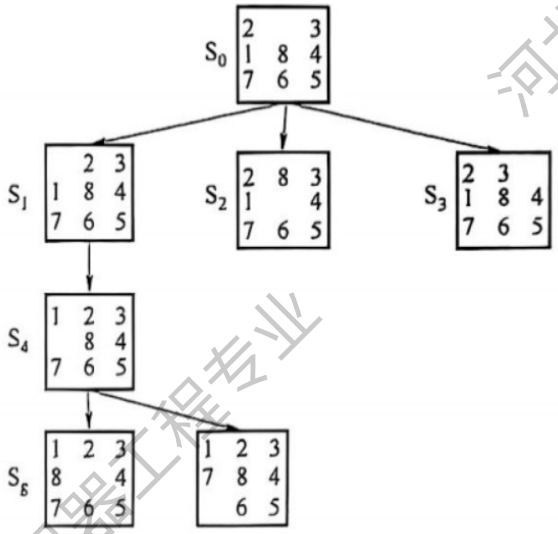
- 完备性：广度优先搜索是一种完备的搜索策略，即只要问题有解，就一定可以找到解路径。
- 最优性：广度优先搜索得到的解一定是路径最短的解。

缺点：

- 盲目性较大，搜索过程会产生很多无用的结点，因此搜索效率较低。

4.2.2 深度优先搜索

深度优先搜索(Depth-First Search，DFS)过程是从初始结点S_0开始，选择最新产生的结点进行考察和扩展，直到找到目标结点为止。OPEN 表中的结点总是将新产生的结点放在 OPEN 表的前面。上例的深度优先搜索图如图所示。



深度优先搜索的优缺点如下：

优点：

- 虽然解一般不是最短路径，但如果目标恰好在搜索分支，则可以很快找到解

缺点：

- 深度优先搜索是一种不完备的搜索策略，即问题有解但不一定能搜索到问题的解

4.3 启发式搜索

前面讨论的搜索策略中没有考虑问题本身的信息，而只是按事先规定的路线进行搜索。如果在搜索过程中使用在搜索过程中获得的问题自身的一些特性信息来指导搜索显然会有利于搜索。将这种利用问题自身特性信息来引导搜索过程的搜索方法称为**启发式搜索**。

启发信息在搜索过程中的主要作用是对结点的重要性进行评估，通过评估来实现 OPEN 表中结点的排序。启发信息越强，则生成的结点越少。评估一般是通过估价函数实现的。

4.3.1 估价函数

估价函数是以结点为自变量的函数，其一般形式定义为：

$$f(n) = g(n) + h(n)$$

其中：

- n是搜索图中当前被扩展的结点
- $f(n)$是从初始状态经由结点n到达目标结点的所有路径中最小路径代价$f^*(n)$的估计值
- $g(n)$是从初始结点到结点n的最小代价$g^*(n)$的估计值
- $h(n)$是从结点n到达目标结点的最优路径代价$h^*(n)$的估计代价，也称为启发函数

在一般图搜索算法中，如果每一步都利用估价函数$f(n)=g(n)+h(n)$对 OPEN 表中的结点进行排序，则称该搜索算法为**A 算法**。由于估价函数带有问题自身的启发性信息，所以 A 算法是启发式搜索算法。

A 算法伪代码

1. 初始化：
   - $G:=G_0$ $(G_0=s)$
   - $OPEN:=\{s\}$
   - $CLOSED:=(\ \ )$
2. 主循环： **A 算法伪代码**
3. $G:=G_0$ $(G_0=s)$, $OPEN:=(s)$
4. $CLOSED:=(\ \ )$
5. LOOP: IF $OPEN=(\ \ )$ THEN EXIT(FAIL);
6. $n:=FIRST(OPEN)$, REMOVE(n, $OPEN$), ADD(n, $CLOSED$);
7. IF GOAL(n) THEN EXIT(SUCCESS);
8. EXPAND(n) → $\{m_e\}$, $G:=ADD(m_k,G)$;
   $f(n,m_k)=g(n,m_k)+h(m_k)$;
   $ADD(m_n,OPEN)$; 标记 m_n 到 n 的指针;
   IF $f(n,m_o)<f(m_o)$ THEN $f(m_o):=f(n,m_o)$; 并标记 m_o 连接到 n 的指针;
   IF $f(n,m_c)<f(m_c)$ THEN $f(m_c):=f(n,m_c)$; 并标记 m_c 连接到 n 的指针;
   $ADD(m_c,OPEN)$;
9. $OPEN$ 中的结点按 f 值从大到小排序;
10. GO LOOP;

例 八数码问题

设初始状态和目标状态如图所示，且估价函数为：

$$f(n)=d(n)+w(n)$$

其中：

$d(n)$ 表示结点n的深度

$w(n)$ 表示结点n中”不在位”的数码个数

解：A算法搜索图如图所示，每个状态的估价函数值在其左上角。

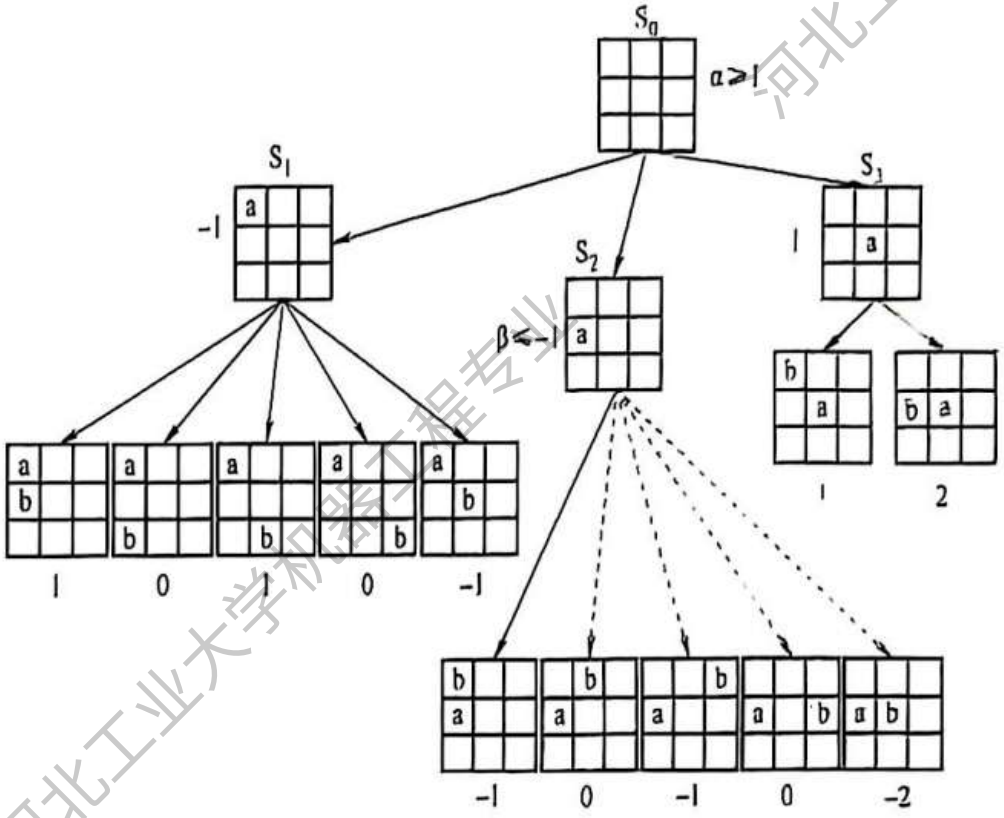


image.png

2. A*算法

对 A 算法增加如下限制： 1. $g(n)$ 是对 $g^*(n)$ 的估计，并且 $c(n, n_j) > \epsilon > 0$ 2. $h(n)$ 是 $h^*(n)$ 的下界，即对任意结点都有 $h(n) \leq h^*(n)$

满足以上条件的算法称为**A*算法**。

续例：估价函数为

$$f(n) = d(n) + w(n)$$

其中：

$d(n)$ 表示结点n的深度

$w(n)$ 表示结点n中”不在位”的数码个数

A*算法搜索图如图所示，每个状态的估价函数值在其左上角。

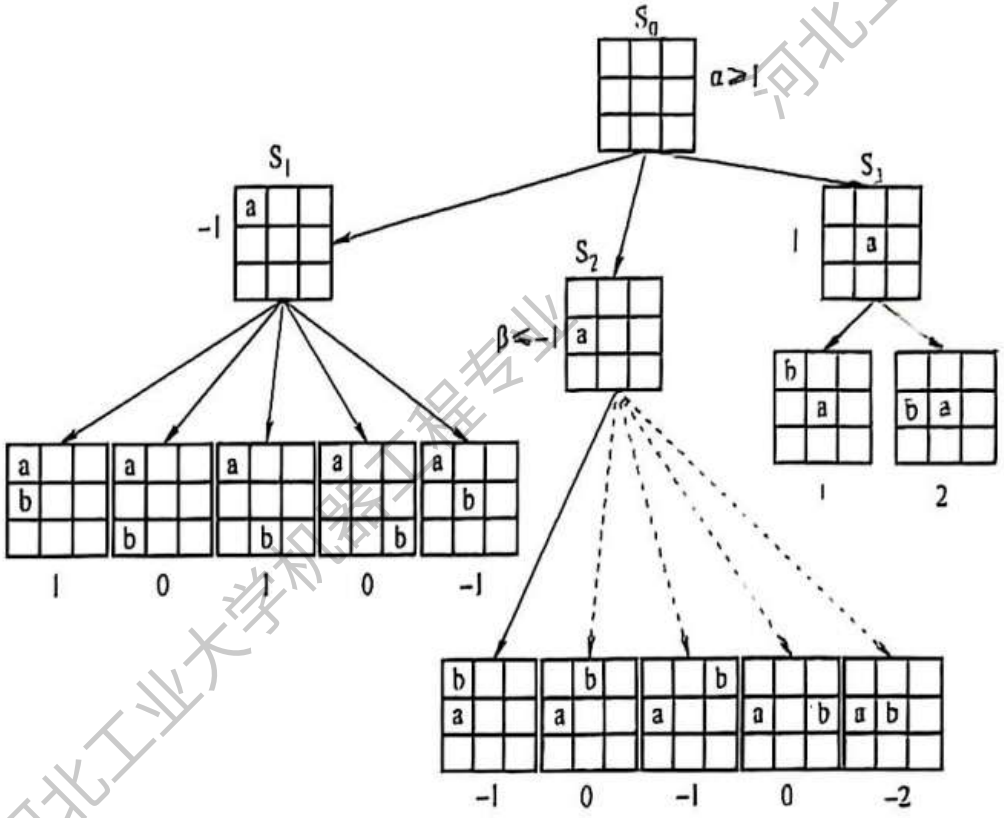


image.png

4.4.4 对抗搜索

对抗搜索一般解决的是博弈问题，机器博弈是假设”两人零和、全信息、非偶然”情况下的博弈，即：

1. 对垒双方轮流采取行动，结局 A 胜、B 胜或和局；
2. 对垒过程中，任一方均了解当前的格局和历史；
3. 双方均理智地决定对策。

示例：Grundy 博弈

Grundy 博弈是一个分钱币的游戏，规则如下：

有一堆数目为N的钱币

由两位选手轮流进行分堆

每个选手每次只能把其中某一堆分成数目不等的两小堆

例如：选手甲将一堆钱币分成两堆后，轮到选手乙就可以挑其中一堆来分，如此进行下去直到有一位选手无法将钱币再分成不相等的两堆时就得认输。

假设共有 7 枚钱币，则整个博弈过程的所有可能情况如图所示，其形成了一颗博弈树。

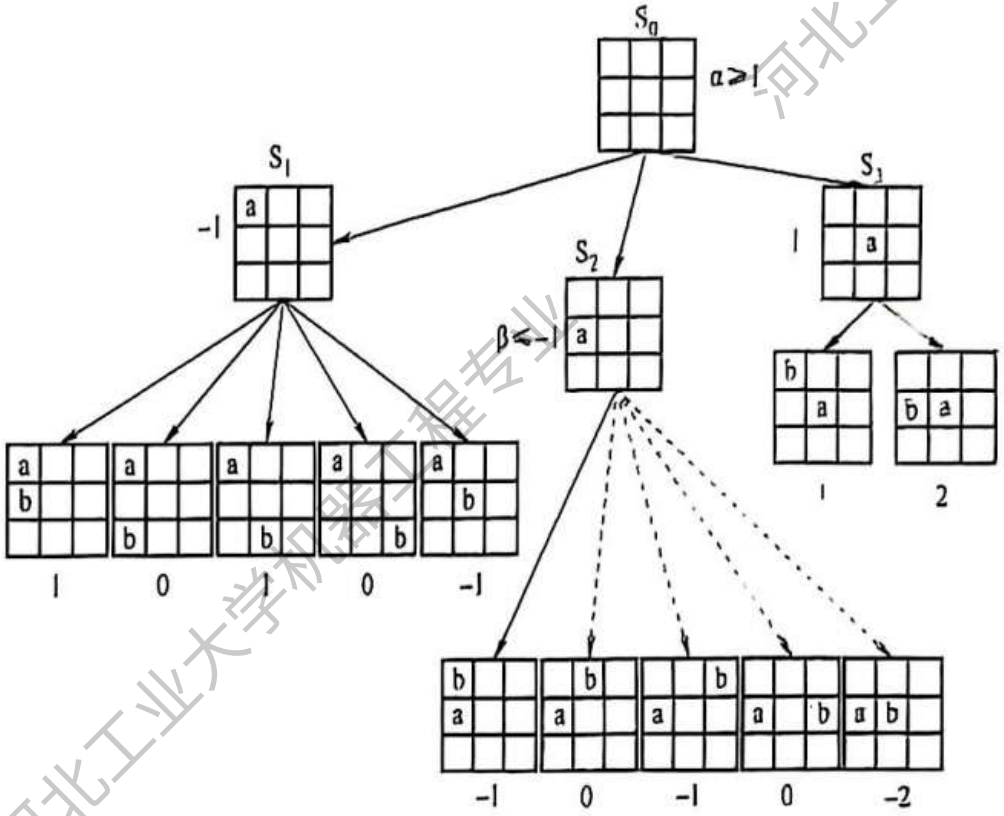


image.png

1. 博弈树是一个”与或”树

博弈树的特点有以下几方面：

1. 博弈的初始状态是初始结点；
2. 博弈树中的”或”结点和”与”结点逐层交替出现；
3. 整个博弈过程始终站在某一方的立场，所有使己方获胜的结点都是本原问题、是可解结点，使对方获胜的终局都是不可解结点。

2. 寻找一个最优行动方案

博弈树搜索是为其中一方寻找一个最优行动方案，极大极小过程如图所示，具体包括以下过程：

1. 对各个方案可能产生的结果进行比较分析并计算可能的得分；
2. 定义估价函数；
3. 端结点的估值计算后，倒推父结点的得分，“或”结点取最大，“与”结点取最小；
4. 取估值最大的方案(树)为行动方案(希望树)。

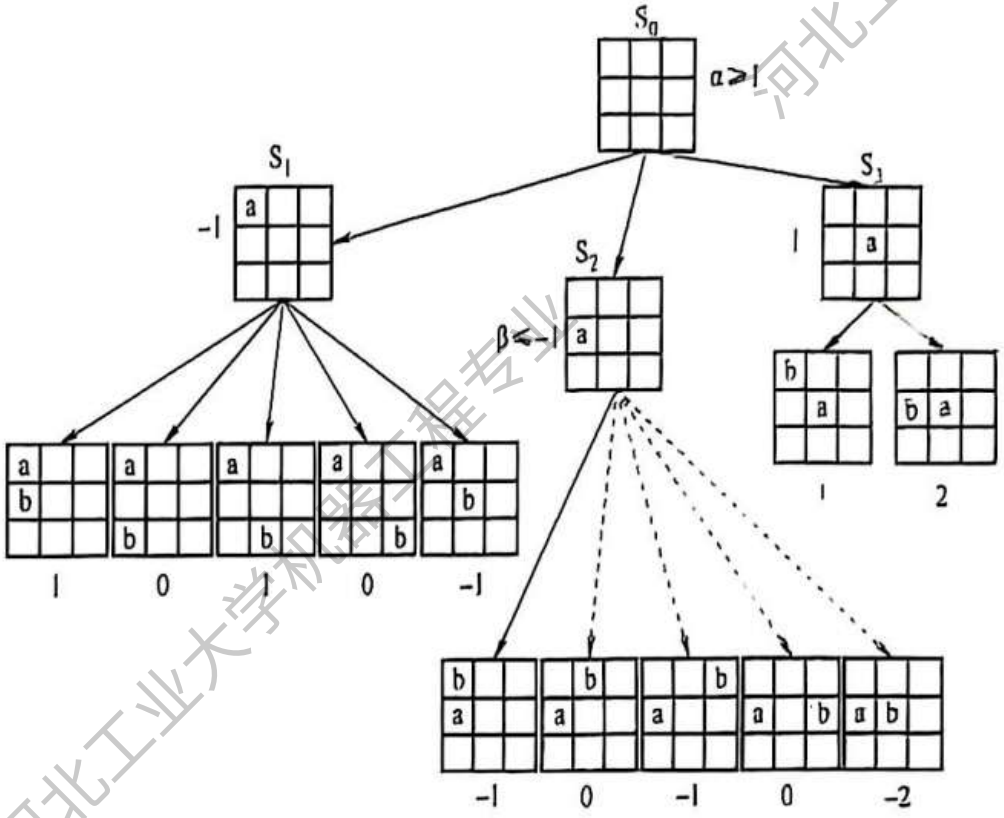


image.png

3. 一个简单的博弈例子(一字棋游戏)

有一个正方形九宫格，A、B 两人对弈，轮到谁走棋谁就在空格上放自己的一个棋子，谁先使棋子构成三子一线，谁就取得胜利。假设如下：

1. A 的棋子为a，B 的棋子为b；
2. 每次扩展两层；
3. 估价函数定义为：

- 若P是 A 必胜的棋局，则$e(P) = +\infty$；
- 若P是 B 必胜的棋局，则$e(P) = -\infty$；
- 若P是胜负未定的棋局，则$e(P) = e(+P) - e(-P)$，其中：
  - $e(+P)$是使得a成为三子一线的数目
  - $e(-P)$是使得b成为三子一线的数目

得到部分博弈树如图所示。

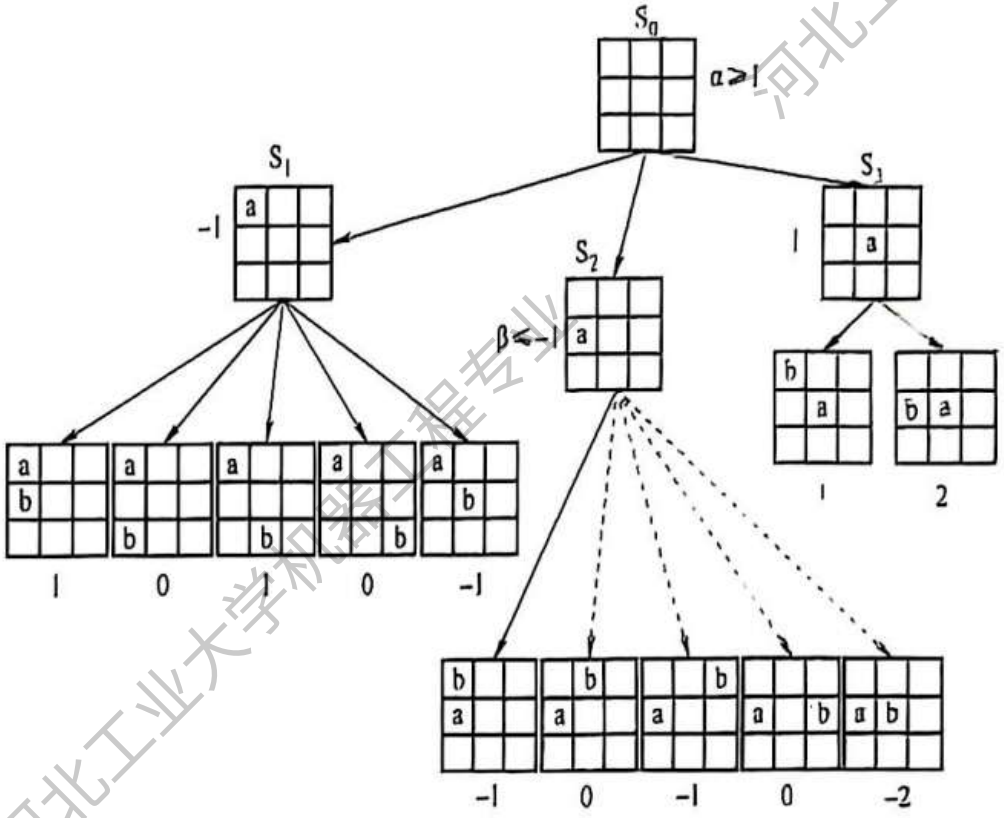


image.png

4. α - β 剪枝技术

在极大极小搜索方法中，由于要先生成指定深度以内的所有结点，其结点数将随着搜索深度的增加呈指数增长，这极大地限制了极大极小搜索方法的使用。那么能否在搜索深度不变的情况下，利用已有的搜索信息减少生成的结点数？

α - β 剪枝技术把生成博弈树的过程与倒推值估计过程结合起来，及时剪掉一些无用的分枝，提高了搜索效率。具体剪枝方法是边生成边估值，剪掉一些没用的分枝。具体做法是：

- 对于一个”与”结点，其倒推值的上界为β值
- 对于一个”或”结点，其倒推值的下界为α值
- 任何”与”结点的β值如果不能提升其父结点的α值，则对结点以下的分枝可以停止搜索，并使x的倒推值为β，此即α剪枝
- 任何”或”结点的α值如果不能降低其父结点的β值，则对结点n以下的分枝可以停止搜索，并使n的倒推值为α，此即β剪枝

对上面提到的一字棋游戏博弈树应用α-β剪枝技术过程如下：

1. 从最左路分枝的叶子结点开始，1、0、1、0、-1 取最小值，则S_1点确定为-1

2. S_1点的最左端-1 和S_2点-1 大小进行比较，假设-1 是S_2点的最小值，由于S_2的父结点是取最大值，$-1 = -1$，无法升高S_0的值，所以S_2点的-1 可以停止搜索，进行α剪枝，如图所示。

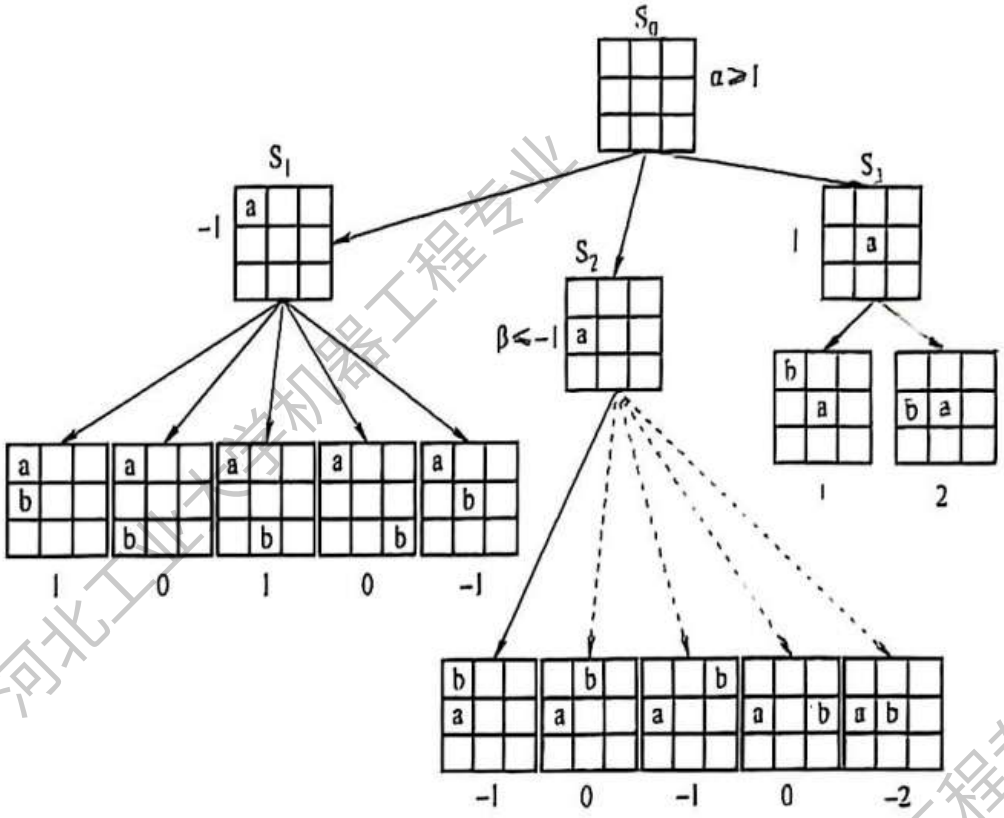


image.png