

人工智能 (Artificial Intelligence)

AilanXier

19 级应该是最后一届人工智能考试课，之后变为考查了，我觉得是有道理的。课程主要内容是符号主义，更接近数理逻辑，而目前主流的联结主义只讲了一节课。考试更偏向于应用，需要背的不多，及格非常简单。考试基本是每个部分出一道题目，基本上作业都练过，就是不确定推理部分计算量偏大。

笔记完全复制于 PPT，没有什么营养，实在是 PPT 上的公式字母过于丑陋，影响观感。但是不能仅靠该整理笔记复习，因为 PPT 不同老师会有修改，没有包含全部重点，及格应该没啥问题。

1 人工智能概论

1.1 人工智能的定义

💡 智能定义：

⚡ 自然智能：

指人类和一些动物所具有的智力和行为能力

⚡ 人类的自然智能（简称智能）：

指人类在认识客观世界中，由思维过程和脑力活动所表现出的综合能力

⚡ 智能（自然智能）现象：

- 1、人是怎样思考问题的？例如：树上还有几只鸟？（常识推理）
- 2、人是怎样横穿马路的？（常识推理和逻辑问题的形象处理）
- 3、人是怎样识别景物的？例如：小孩的妈妈是谁？（形象思维）
- 4、人是怎样实现感知、学习、思维等的？（神经系统的心智活动）
- 5、人是怎样产生情绪、情感的？（神经系统的心理过程）

⚡ 定义智能的困难：

- ⌚ 从结构上，人脑有 10^{11-12} 量级的神经元，是广泛分布、并行的、巨复杂系统
- ⌚ 从功能上，人脑具有记忆、思维、观察、分析等能力
- ⌚ 有待于人脑奥秘的揭示，进一步认识

⚡ 内涵：「知识+思维」

⚡ 外延：获取知识、运用知识的能力；分析问题、解决问题的能力等

⚡ 思维理论：

智能来源于思维活动，智能的核心是思维，人的一切知识都是思维的产物。可望通过对思维规律和思维方法的研究，来揭示智能的本质。

知识阈值理论：

智能取决于知识的数量及其可运用程度。一个系统所具有的可运用知识越多，其智能就会越高。

进化理论：

是美国 MIT 的 Brooks 在对人造机器虫研究的基础上提出来的。智能取决于感知和行为，取决于对外界复杂环境的适应，智能不需要知识、不需要表示、不需要推理，智能可由逐步进化来实现。

智能的层次结构：

高层智能：

以大脑皮层（抑制中枢）为主，主要完成记忆、思维等活动。

中层智能：

以丘脑（或称间脑，感觉中枢）为主，主要完成感知活动。

低层智能：

以小脑、脊髓为主，主要完成动作反应活动。

思维理论和知识阈值理论 → 高层智能

进化理论 → 中层智能和低层智能

智能包含的能力：

感知能力：

通过感知器官感知外界的能力。是人类获得外界信息的基本途径，其处理方式有两种：

 感知 - 动作方式：对简单、紧急信息（如膝跳反射）

 感知 - 思维 - 动作方式：对复杂信息

记忆和思维能力：

 记忆：

对感知到的外界信息和由思维产生的内部知识的存储过程

 思维：

对已存储信息或知识的本质属性、内部规律的认识过程

 思维方式：

 抽象思维（逻辑思维）：

根据逻辑规则对信息和知识进行处理的理性思维方式。例如，逻辑推理等

 形象思维（直感思维）：

基于形象概念，根据感性形象认识材料对客观现象进行处理的一种思维方式。

例如，图像、景物识别等

 灵感思维（顿悟思维）：

是一种显意识和潜意识相互作用的思维方式。例如，因灵感而顿时开窍

／＼ 学习和自适应能力：

／＼ 学习：

是一个具有特定目的的知识获取过程，是人的一种本能。不同人的学习方法、能力不同

／＼ 自适应：

是一种通过自我调节适应外界环境的过程，是人的一种本能。不同人的适应能力不同

／＼ 行为能力：

／＼ 含义：

是人们对感知到的外界信息作出动作反应的能力

／＼ 信息来源：

由感知直接获得的外界信息或经过思维加工后的信息

／＼ 实现过程：

通过脊髓来控制由语言、表情、体姿等来实现

／＼ 什么是人工智能：

／＼ 人工方法实现的智能

／＼ 目前的「人工智能」一词是指用计算机模拟实现的智能，同时，人工智能又是一个学科名称

／＼ 人造的智能机器或系统

／＼ 模仿、延伸以及扩展人的智能

如何衡量机器是否具有智能：

／＼ 图灵测试：

／＼ 把一个等待测试的计算机和一个思维正常的人分别关在两间屋子里，然后提问题

／＼ 通过提问，分析机器和人对你的回答来想办法区分哪一个是机器，哪一个是人

／＼ 如果你无法区分，那么，这台机器就通过了测试，就证明这台机器和人一样具有思维，这是一台会思考的机器

／＼ 中文屋子（中文房间，华语房间，Chinese room，the Chinese room argument）：

／＼ 由美国哲学家约翰·希尔勒（John Searle）在1980年设计的一个思维试验以推翻强人工智能（机能主义）提出的过强主张：只要计算机拥有了适当的程序，理论上就可以说计算机拥有它的认知状态以及可以像人一样地进行理解活动。

这个实验要求你想象一位只说英语的人身处一个房间之中，这间房间除了门上有一个小窗口以外，全部都是封闭的。他随身带着一本写有中文翻译程序的书。房间里还有足够的稿纸、铅笔和橱柜。写着中文的纸片通过小窗口被送入房间中。

房间中的人可以使用他的书来翻译这些文字并用中文回复。虽然他完全不会中文，Searle 认为通过这个过程，房间里的人可以让任何房间外的人以为他会说流利的中文。

2 定义方式：



2.1 类人思维方法（认知模型方法）：

- 一种基于人类思维工作原理的可检测理论来定义智能的方法。
- 典型代表是贝尔曼（Bellman）于 1978 年提出的定义：
人工智能是那些与人的思维、决策、问题求解和学习等有关活动的自动化。
- 认知科学，研究人类感知和思维信息处理过程，它把来自人工智能的计算机模型和来自心理学的实验技术结合起来，试图创立一种精确而且可检测的人脑思维过程的工作模型。
- 如果能把上述所得到关于思维的足够精确的模型用计算机程序表示出来，并且该程序的输入/输出和实时行为能够与人类相一致，那就说明该程序的某些机制可能是按照人脑思维模式运行的。
- 这方面的典型例子是艾伦纽厄尔（Allen Newell）和赫伯特·西蒙（Herbert Simon）等人于 1960 年研制了通用问题求解（general problem solving, GPS）程序。

2.2 类人行为方法（图灵测试方法）：

- 一种基于人类自身的智能去定义一个机器或系统是否具有智能的方法。
- 典型代表是库兹韦尔（Kurzweil）于 1990 年提出的定义：
人工智能是一种创建机器的技艺，这种机器能够执行需要人的智能才能完成的功能。
- 按照图灵测试要求，一台计算机要能够通过，至少应该具有以下能力：
 - 自然语言处理（含语音技术），实现用自然语言与计算机的交流。
 - 知识表示，存储它所知道的或听到的知识或信息。
 - 自动推理，运用存储的知识或信息来回答问题，并提取新的结论。

4. 机器学习，能适应新的环境，并能自我获取新的知识。

如果还需要测试被测对象利用视频信号的感知能力和传递接受物体的行为能力，即所谓的完全图灵测试，则计算机还应该具有如下能力：

- 计算机视觉，可以感知物体。
- 机器人技术，可以操纵和移动物体。

2 理性思维方法（思维法则方法）：

- 一种基于逻辑推理定义智能的方法。
- 典型代表是查尼艾克（E.Charniak）和麦克德莫特（D.McDermott）于 1985 年提出的定义：
 - 人工智能是通过计算模型来进行心智能力研究的。
 - 计算模型主要是指能「正确思维」的逻辑学模型。古希腊哲学家亚里士多德（Aristotle）是首先严格定义「正确思维」的人之一，他将其定义为「不能辩驳的推理过程」，例如三段论推理方法。
 - 理性思维方法正是人工智能领域中所谓的逻辑主义观点，他们希望通过编制逻辑程序来建造智能系统。
 - 这种方法存在两个主要问题：
 - 非形式的知识用形式的逻辑符号表示不易实现，尤其是对不确定的知识
 - 原则上可以解决的问题与实际解决问题之间存在较大差异，需要考虑推理过程的控制。

2 理性行为方法（理性智能体方法）：

- 一种基于智能体定义智能的方法。
- 典型代表是尼尔森（N.J.Nilsson）于 1998 年提出的定义：
 - 人工智能关心的是人工制品中的智能行为。
 - 这里的人工制品主要是指能够感知环境、适应变化、自主操作、执行动作的理性智能体（Agent）。
 - 按照这种方法，可以认为人工智能就是研究和建造理性智能体。
- 理性行为方法与理性思维方法的关系：
 - 理性行为和理性思维强调的重点不同。理性思维方法强调的是正确思维，而理性行为方法强调的则是合理行动。
 - 理性行为可以依据理性思维进行。例如，对一些能够通过理性思维做出正确结论的事情，实现理性行为的途径往往是先通过逻辑推理得出该行为能达到的目标和结论，然后再付诸实施。
 - 理性行为不一定要依据理性思维进行。例如，对有些事情，即使理性思维无法证明哪些行为是正确的，而其它行为是错误的，理性行为也必须有所行动。

上述四种方法都有人做了工作，在以人为中心的方法和以理性为中心的方法之间存在着一定的争议。例如：

- ／＼ 以人为中心的方法是一种经验科学，它需要涉及到很多假设和实验证实。
- ／＼ 以理性为中心的方法则涉及到把数学与工程相结合

2 人工智能的一般解释：

／＼ 从能力的角度：

- ／＼ 指用人工的方法在机器（计算机）上实现的智能。
- ／＼ 智能机器所执行的通常与人类智能有关的功能，如判断、推理、证明、识别、感知、理解、设计、思考、规划、学习和问题求解等思维活动。
- ／＼ 从学科的角度：
- ／＼ 人工智能是一门研究如何构造智能机器或智能系统，去模拟、延伸和扩展人类智能的学科。
- ／＼ 计算机科学中涉及研究、设计和应用智能机器的一个分支。

2 人工智能的研究意义

- ／＼ 研究人工智能是当前信息化社会的迫切要求。
- ／＼ 智能化也是自动化发展的必然趋势。
- ／＼ 探索人类自身智能的奥秘，发现自然智能的渊源。

2 人工智能的研究目标：

- ／＼ 目前没有一个统一的说法。
- ／＼ 1978 年，斯洛曼对人工智能给出了三个主要目标：

- ／＼ 对智能行为有效解释的理论分析
- ／＼ 解释人类智能
- ／＼ 构造具有智能的人工制品

／＼ 远期目标：

- ／＼ 揭示人类智能的根本机理，用智能机器去模拟、延伸和扩展人类的智能
- ／＼ 涉及到脑科学、认知科学、计算机科学、系统科学、控制论等多种学科，并依赖于它们的共同发展
- ／＼ 用自动机重现人类的思维过程和智能行为

／＼ 近期目标：

- ／＼ 研究如何使现有的计算机更聪明，即使它能够运用知识去处理问题，能够模拟人类的智能行为。
- ／＼ 建造智能计算机代替人类的部分智力劳动

远期目标为近期目标指明了方向

近期目标则为远期目标奠定了理论和技术基础

2 AI 的学科位置：

★ 一门新兴的交叉学科

● 一门新兴的学科，是自然科学与社会科学的交叉学科

● 交叉包括：

逻辑、思维、生理、心理、计算机、电子、语言、自动化、光、声等

● AI 的基础学科包括：

数学（离散、模糊）、思维科学（认知心理、逻辑思维学、形象思维学）和计算机（硬件、软件）等

★ 信息时代的核心技术：

在数学机械化领域，吴文俊提出的几何定理的机器证明被国际数学界称为「吴方法」。

2 与脑科学和认知科学的交叉研究：

★ 脑科学与神经科学：

● 脑科学：

一门研究脑与心智现象及规律的科学，其主要目标就是要揭示脑功能的本质，认识脑与智能的规律，保护脑和创造脑。

● 神经科学：

一门研究神经系统内分子水平、细胞水平及细胞间的变化过程，及这些过程在中枢的功能、控制系统内的整合作用所进行的学科。

● 脑的涵义：

从狭义方面，脑是指中枢神经系统，有时特指大脑

从广义方面，脑可泛指整个神经系统。人工智能是从广义角度来理解脑科学的

● 脑的复杂度：

人脑是由巨量神经元经其突触的广泛并行互联所形成的一个巨复杂系统，是自然界中最复杂、最高级的智能系统。

● 现代脑科学的基本问题：

● 揭示神经元之间的连接形式，奠定行为的脑机制的结构基础

● 阐明神经活动的基本过程，说明在分子、细胞到行为等不同层次上神经信号的产生、传递、调制等基本过程

● 认识实现各种功能的神经回路基础

● 解释脑的高级功能机制等

● 脑科学是 AI 的基础：其任何研究进展都将对人工智能的研究起到积极的推动作用。

★ 认知科学和思维科学：

● 认知：

一般地认为是和情感、动机、意志相对应的理智或认识过程，或者是为了一定的目的，在一定的心理结构中进行的信息加工过程。美国心理学家浩斯顿（Houston）等人把认知归纳为以下 5 种主要类型：

- 📎 信息的处理过程
- 📎 心理上的符号运算
- 📎 问题求解
- 📎 思维
- 📎 一组相关的活动，如知觉、记忆、思维、判断、推理、问题求解、学习、想象、概念形成及语言使用等。

💡 认知科学：

- 📎 认知科学（思维科学）是研究人类感知和思维信息处理过程的一门学科，其主要研究目的就是要说明和解释人类在完成认知活动时是如何进行信息加工的认知科学也是人工智能的重要理论基础，对人工智能发展起着根本性的作用。
- 📎 认知科学涉及的问题非常广泛，除了像浩斯顿提出的相关联活动外，还会受到环境、社会、文化背景等方面的影响。
- 📎 从认知观点看，AI 应同时开展对逻辑思维、形象思维和灵感思维的研究

💡 智能模拟的方法和技术研究：

💡 机器感知：

- 💡 就是要让计算机具有类似于人的感知能力，如视觉、听觉、触觉、嗅觉、味觉
- 💡 机器视觉（或叫计算机视觉）：就是给计算机配上能看的视觉器官，如摄像机等，使它可以识别并理解文字、图像、景物等
- 💡 机器听觉（或叫计算机听觉）：就是给计算配上能听的听觉器官，如话筒等，使计算机能够识别并理解语言、声音等。
- 💡 机器感知相当于智能系统的输入部分。
- 💡 机器感知的专门的研究领域：计算机视觉、模式识别、自然语言理解

💡 机器思维：

让计算机能够对感知到的外界信息和自己产生的内部信息进行思维性加工：

- 💡 逻辑思维
- 💡 形象思维
- 💡 灵感思维

💡 机器学习：

- 💡 让计算机能够像人那样自动地获取新知识，并在实践中不断地完善自我和增强能力。
- 💡 机器学习方法：机械学习、类比学习、归纳学习、发现学习、遗传学习和连接学习等

💡 机器行为：

- 💡 让计算机能够具有像人那样地行动和表达能力，如走、跑、拿、说、唱、写画等。
- 💡 相当于智能系统的输出部分。

💡 智能系统与智能机器：

无论是人工智能的近期目标还是远期目标，都需要建立智能系统或构造智能机器，需要开展对系统模型、构造技术、构造工具及语言环境等研究

💡 人工智能的研究策略：

先部分地或某种程度地实现机器的智能，并运用智能技术解决各种实际问题特别是工程问题，从而使现有的计算机更灵活、更好用和更有用，成为人类的智能化信息处理工具，从而逐步扩展和不断延伸人的智能，逐步实现智能化。

1.2 人工智能的发展简史

- ∅ 第1阶段（1956年以前）孕育期
- ∅ 第2阶段（1956 - 1970年）形成期
- ∅ 第3阶段（1970 - 20世纪80年代末）知识应用期
- ∅ 第4阶段（20世纪80年代到本世纪初）从学派分立到综合：
- ∅ 第5阶段（本世纪初以来）智能科学技术的兴起
- ∅ 历史上的人工智能大师：
 - ／ 阿伦·图灵（1912~1954）
 - ／ 马文·明斯基
 - ／ 约翰·麦卡锡
 - ／ 赫伯特·西蒙，符号主义学派创始人
 - ／ 艾伦·纽厄尔，符号主义学派创始人之一
 - ／ 爱德华·费根鲍姆，知识工程的提出者
 - ／ 劳伊·雷迪，大型人工智能系统的开拓者

1.3 人工智能研究内容

- ∅ 如何获取知识
- ∅ 如何将获取的知识以计算机内部代码形式加以合理表示
- ∅ 如何运用知识进行推理，解决实际问题

1.4 人工智能研究的方法与途径（三大学派）

- ∅ 符号主义（Symbolicism）：
 - ／ 运用计算机科学的方法（逻辑演绎）
 - ／ 核心是知识表示和知识推理
 - ／ 功能模拟
- ∅ 联结主义（Connectionism）：
 - ／ 运用仿生学的方法（网络连接机制）
 - ／ 提出了智能行为的基元是神经元，而不是符号；思维过程是神经元的联结活动过程，而不是符号运算过程
 - ／ 结构模拟
- ∅ 行为主义（Behaviorism）：
 - ／ 运用进化论的思想（控制论和机器学习算法）
 - ／ 智能取决于感知和行动，AI可以像人类智能那样逐步进化
 - ／ 行为模拟

1.5 人工智能研究领域

机器思维、机器感知、机器行为、机器学习、计算智能、分布智能、智能系统

目前，人工智能能够完成的任务（大多是封闭环境下的任务）：

- ∅ 打乒乓球 ✓
- ∅ 在崎岖的山路上安全驾驶 ✓
- ∅ 在深圳安全驾驶 ?
- ∅ 从网上买一周的货物 ✓
- ∅ 从美康百货超市买一周的货物 ✗
- ∅ 发现并证明一个新的数学定理 ?
- ∅ 成功的同一个人聊天一小时 ✗
- ∅ 实施医学手术 ?
- ∅ 实时的将汉语翻译为英语 ✓
- ∅ 有意识写一个有趣的故事 ✗
- ∅ 收拾碗筷和叠衣服 ✓

1.5.1 机器思维

∅ 推理的概念：

推理是指按照某种策略从已知事实出发利用知识推出所需结论的过程。

∅ 推理的类型（根据所用知识的确定性）：

★ 确定性推理：

- ∅ 指推理所使用的知识和推出的结论都是可以精确表示的，其真值要么为真、要么为假。
- ∅ 主要是基于一阶经典逻辑。它能解决的问题很有限。

★ 不确定性推理：

- ∅ 指推理所使用的知识和推出的结论可以是不确定的。
- ∅ 所谓不确定性是对非精确性、模糊型和非完备性的统称。
- ∅ 主要基于非经典逻辑和概率等。

∅ 最常用的不确定性推理方法：

- ∅ 基于可信度的确定性理论
- ∅ 基于 Bayes 公式的主观 Bayes 方法
- ∅ 基于贝叶斯网络的概率推理和基于模糊逻辑的可能性理论等。

∅ 推理的理论基础：

逻辑是一门研究人们思维规律的学科，数理逻辑则是用数学的方法去研究逻辑问题。

∅ 搜索的概念：

是指为了达到某一目标，不断寻找推理线路，以引导和控制推理，使问题得以解决的过程。

2 搜索的类型（根据问题的表示方式）：

- ／＊ 状态空间搜索是一种用状态空间法求解问题时的搜索方法
- ／＊ 与/或树搜索是一种用问题规约法求解问题时的搜索方法

3 博弈树搜索：

博弈是一个典型的搜索问题。到目前为止，人们对博弈的研究还主要是以下棋为对象。

1.5.2 机器感知

1 计算机视觉：

- ／＊ 概念：

用计算机来实现或模拟人类的视觉功能，其主要研究目标是使计算机具有通过二维图像认知三维环境信息的能力。

- ／＊ 重要性：

在人类感知到的外界信息中，有 80% 以上是通过视觉得到的。

- ／＊ 不仅仅指对光信号的感受，它包括了对视觉信息的获取、传输、处理、存储与理解的全过程。

2 模式识别：

是指让计算机能够对给定的事物进行鉴别，并把它归入与其相同或相似的模式中。被鉴别的事物可以是物理的、化学的、生理的，也可以是文字、图像、声音等。

3 自然语言处理：

自然语言处理就是要研究人类与计算机之间进行有效交流的各种理论和方法。

1.5.3 机器行为

1 智能控制/制造：

- ／＊ 是指那种无需或需要尽可能少的人工干预就能独立的驱动智能机器实现其目标的控制过程。

- ／＊ 是人工智能技术与传统自动控制技术相结合的产物。

2 主要应用领域：

智能机器人系统、计算机集成制造系统（CIMS）、复杂工业过程的控制系统、航空航天控制系统、社会经济管理系统、交通运输系统、环保及能源系统等。

1.5.4 机器学习

1 让计算机能够像人那样自动地获取新知识，并在实践中不断地完善自我和增强能力。

2 机器获取知识的根本途径，同时也是机器具有智能的重要标志。

3 机器学习有多种不同的分类方法，如果按照对人类学习的模拟方式，机器学习可分为：

- ／＊ 符号学习：

- ／＊ 概念：

是指从功能上模拟人类学习能力的机器学习方法，它是一种基于符号主义学派的机器学习观点。

类型：

可根据学习策略，即学习中所使用的推理方法，将其分为记忆学习、归纳学习、演绎学习等。

神经学习：

概念：

神经学习也称为连接学习，它是一种基于人工神经网络的学习方法。现有研究表明，人脑的学习和记忆过程都是通过神经系统来完成的。在神经系统中，神经元既是学习的基本单位，同时也是记忆的基本单位。

类型：

(多层) 感知器学习、BP 网络学习、Hopfield 网络

1.5.5 计算智能

计算智能 (Computational Intelligence, CI) 是借鉴仿生学的思想，基于人们对生物体智能机理的认识，采用数值计算的方法去模拟和实现人类的智能。

计算智能的三大基本领域包括神经计算、进化计算、模糊计算。

神经计算：

亦称神经网络 (Neural Network, NN)，它是通过对大量人工神经元的广泛并行互联所形成的一种人工网络系统，用于模拟生物神经系统的结构和功能。

主要研究内容：

包括人工神经元的结构和模型，人工神经网络的互连结构和系统模型，基于神经网络的联结学习机制等

神经网络具有自学习、自组织、自适应、联想、模糊推理等能力，在模仿生物神经计算方面有一定优势。目前，神经计算的研究和应用已渗透到许多领域，如机器学习、专家系统、智能控制、模式识别等。

进化计算：

是一种模拟自然界生物进化过程与机制，进行问题求解的自组织、自适应的随机搜索技术。它以达尔文进化论的「物竟天择、适者生存」作为算法的进化规则，并结合孟德尔的遗传变异理论，将生物进化过程中的繁殖、变异、竞争和选择引入到了算法中，是一种对人类智能的演化模拟方法。

进化计算的主要分支：

遗传算法、进化策略、进化规划和遗传规划四大分支。其中，遗传算法是进化计算中最先形成的一种具有普遍影响的模拟进化优化算法。

模糊计算 (模糊系统)：

通过对人类处理模糊现象的认知能力的认识，用模糊集合和模糊逻辑去模拟人类的智能行为的。模糊集合与模糊逻辑是美国加州大学扎德 (Zadeh) 教授 1965 年提出来的一种处理因模糊而引起的不确定性的有效方法。

模糊概念的定义：

通常，人们把那种因没有严格边界划分而无法精确刻画的现象称为模糊现象，并把反映模糊现象的各种概念称为模糊概念。例如，「大，小，多，少」等。

模糊概念的表示：

通常是用模糊集合来表示的，而模糊集合又是用隶属函数来刻画的。一个隶属函数描述一个模糊概念，其函数值为 $[0, 1]$ 区间的实数，用来描述函数自变量所代表的模糊事件隶属于该模糊概念的程度。

1.5.6 分布智能

分布智能的概念

分布智能主要研究在逻辑上或物理上分布的智能系统之间如何相互协调各自的智能行为，实现问题的并行求解。

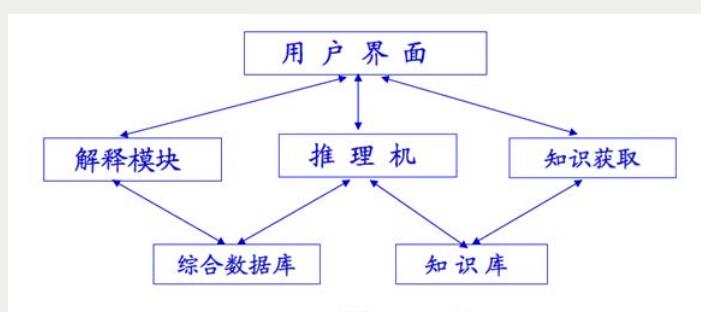
1.5.7 智能系统

智能系统可以泛指各种具有智能特征和功能的软硬件系统。这里主要介绍除前述研究内容以外的专家系统和智能决策支持系统。

1.5.8 专家系统

专家系统是一种基于知识的智能系统，它将领域专家的经验用知识表示方法表示出来，并放入知识库中，供推理机使用。

随着计算网络、多 Agent、计算智能等技术的发展，出现了模糊专家系统、神经网络专家系统、基于 Web 的专家系统、协同式专家系统和分布式专家系统等。



1.6 人工智能的应用、现状与思考

应用：

智能机器人

智能网络

智能游戏

现状：

❖ 多学科交叉研究：

人工智能理论基础研究应强调与脑科学、认知科学、心理学、信息科学、生物学、逻辑学、物理学和数学等学科的交叉研究。

❖ 多学派和多技术融合研究：

- ❖ 多学派融合是一种必然趋势
- ❖ 符号主义：高层、宏观联结主义：底层、微观
- ❖ 行为主义：感知反应、进化
- ❖ 三大学派各有所长、各有所短，它们各自经过一段时间的分立研究之后，正逐步开始走向融合。多学派融合是人工智能发展的一种必然趋势。

❖ 分布智能研究：

主要研究在逻辑上或物理上分布的智能系统之间如何相互协调各自的智能行为，实现问题的并行求解。

❖ 群体智能研究：

是指无智能的或具有简单智能的个体通过群体协作和组织所表现出来的智能。主要研究单个个体，如何通过群内个体的连接、信息交流、沟通、组织和自组织去产生群体的智能。

❖ 社会智能研究、集成智能研究、认知计算与情感计算研究、智能系统与智能服务

2 发展方向：

张钹院士提出了第三代人工智能的四要素：知识、数据、算法、算力

1.7 人工智能对人类的影响

2 经济：

- ❖ 直接经济效益
- ❖ 推动计算机技术发展

2 社会：

- ❖ 劳务就业问题
- ❖ 社会结构变化
- ❖ 思维方式与观念的变化
- ❖ 心理上的威胁
- ❖ 技术失控的危险
- ❖ 引起的法律问题

2 文化：

- ❖ 改善人类语言
- ❖ 改善文化生活

2 知识表示

2.1 知识与知识表示

- ∅ 知识是人类智能的基础（符号主义学派）。
- ∅ 智能活动过程主要是一个获取知识并运用知识的过程。
- ∅ 人工智能问题的求解也是以知识为基础的，知识的获取、知识的表示和运用知识进行推理是人工智能学科研究的 3 个主要问题。

2.1.1 知识的含义和结构

- ∅ 知识的金字塔结构，从上到下是：元知识、知识、信息、数据、噪声
- ∅ 数据：

是记录信息的符号，是信息的载体和表示。

- ∅ 信息：
- 是对数据的解释，是数据在具体的场合下具体的含义。

「100」是数据，它可能表示「100 元钱」、「100 个人」、「100 分」等信息。

- ∅ 知识：
- 一般把有关信息关联在一起所形成的信息结构称为知识。
- ∅ 知识、信息、数据是 3 个层次的概念：

抽象方向：有格式的数据 → 信息 → 知识

知识形成例子：

1. 数据 137179766832525156430015
2. 数据加工：

加工规则：

- (a) 将每两位数字分为一组；
- (b) 忽略那些小于 32 的两位数；
- (c) 把余下的每组两位数用 ASCII 字符代替

得到加工后的信息 GOLD 438+

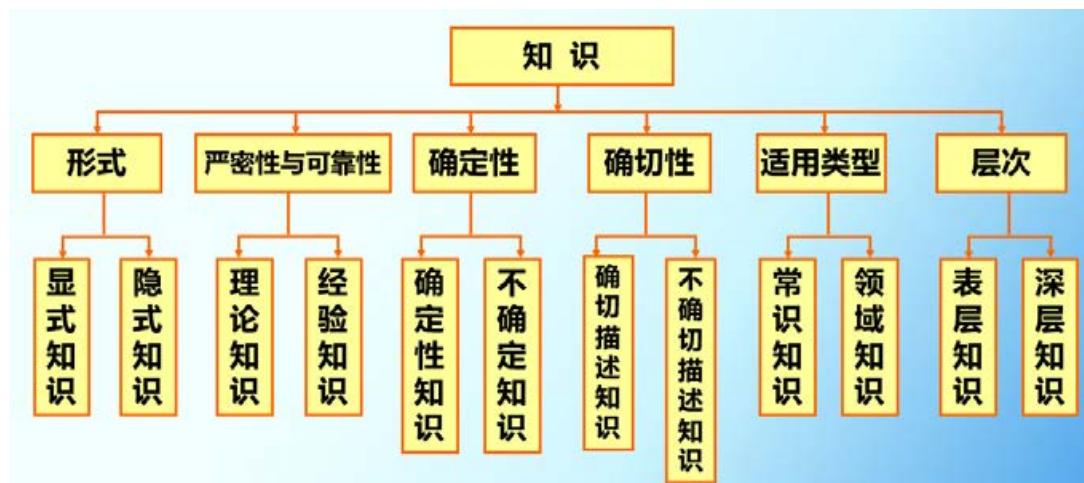
3. 信息所表示的知识：
- 黄金价格为 438，并且在升值（+）

- ∅ 元知识：
- 所谓元知识，就是指使用知识的知识。

如果：黄金价格低于 500 且价格正在上涨 (+)

那么：购买黄金

2.1.2 知识的种类



2.1.3 知识的特性

ℳ 相对正确性：

例如： $1+1=10$ 在不同的进制下有不同的正确性。

ℳ 不确定性：

知识并不总是只有 True 和 False 两种状态。引起知识不确定性的原因有：

ℳ 随机性（概率）：我有八成的把握打中目标。

ℳ 模糊性：高个子适合于打篮球。

ℳ 不完全性：这种药可能会治疗 SARS。

ℳ 经验性：土干了就给花浇水。

ℳ 可表示性

ℳ 可利用性

2.1.4 知识表示

ℳ 什么是知识表示：

ℳ 面向计算机的知识描述或表达的形式和方法。

ℳ 知识表示的过程就是把知识编码成某种数据结构的过程。

ℳ 研究的主要内容：

表示方法

ℳ 知识表示方法：

ℳ 谓词逻辑表示

ℳ 产生式表示

ℳ 语义网络表示

ℳ 框架表示

2.2 一阶谓词逻辑表示法

一阶谓词逻辑表示法是一种重要的知识表示方法，它以数理逻辑为基础，是到目前为止能够表达人类思维活动规律的一种最精确的形式语言。

个体域：

个体变元的变化范围称为个体域（或论域）。

包揽一切事物的集合称为全总个体域。

谓词：

谓词逻辑中的 n 元谓词：

$$P(x_1, x_2, \dots, x_n)$$

x_i 是参量（项/个体）

项包含变量、常量、以及函数符号

P 是谓词符号（大写字母）

函数：

为了表达个体之间的对应关系，引入 n 元个体函数，简称函数：

$$f(x_1, x_2, \dots, x_n)$$

x_i 是个体变元

P 是函数符号（小写字母）

量词：

全称量词：所有，一切，任一，全体，凡是

存在量词：存在，有些，至少有一个，有的

谓词公式：

谓词连接符（优先级从大到小）：

\neg ：否定（非）

\wedge ：合取（与）

\vee ：析取（或）

\rightarrow ：蕴含（IF … THEN）

\leftrightarrow ：等价（当且仅当）

辖域：

紧接于量词之后被量词作用（即说明）的谓词公式称为该量词的辖域。

全称量词辖域 $\forall x(P(x) \rightarrow G(x, y))$

存在量词辖域 $\exists xP(x) \wedge B(x)$ ，只有 $P(x)$ 在辖域中

指导变元：

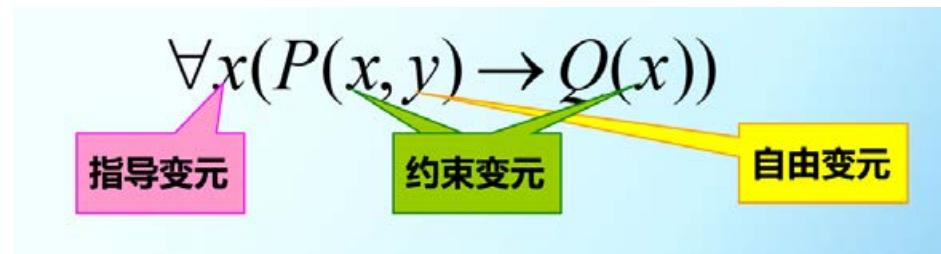
量词后面的变元称为量词的指导变元：

约束变元：

在一个量词的辖域中的与该量词的指导变元相同的变元称为约束变元；

自由变元：

其它的变元称为自由变元



改名规则：

一个变元在一个谓词公式中既可约束出现，又可自由出现，为了避免混淆，通常通过改名规则，使得一个谓词公式中一个变元仅以一种形式出现。

换名规则：

在谓词公式中，将某量词辖域中出现的某个约束变元以及对应的指导变元更改为本辖域中没有出现过的个体变元符号，公式其它部分不变，谓词公式的等价性不变。

代替规则：

在谓词公式中，将某量词辖域中出现的某个自由变元的所有出现用本辖域中未曾出现过的某个个体变元符号代替，谓词公式的等价性不变。

$$\forall x P(x, y, z) \rightarrow \exists y Q(x, y, z)$$

利用换名规则：

$$\begin{aligned} &= \forall u P(u, y, z) \rightarrow \exists y Q(x, y, z) \\ &= \forall u P(u, y, z) \rightarrow \exists v Q(x, v, z) \end{aligned}$$

利用代替规则：

$$\begin{aligned} &= \forall x P(x, u, z) \rightarrow \exists y Q(x, y, z) \\ &= \forall x P(x, u, z) \rightarrow \exists y Q(v, y, z) \end{aligned}$$

谓词公式表示知识的步骤：

1. 定义谓词及个体，确定每个谓词及个体的确切含义
2. 根据所要表达的事物或概念，为每个谓词中的变元赋以特定的值
3. 根据所要表达的知识的语义，用适当的联接符号将各个谓词联接起来，形成谓词公式。

例子：

1. 有的人既喜欢梅花又喜欢菊花。

定义谓词 $LIKE(x, y)$: x 喜欢 y 。

定义个体 x : 人

$y : \{(meihua\ 梅花), (juhua\ 菊花)\}$

$\exists x(\text{LIKE}(x, meihua) \wedge \text{LIKE}(x, juhua))$

2. 所有整数不是偶数就是奇数。

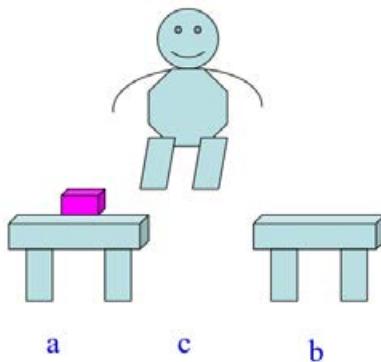
$\forall x(I(x) \rightarrow E(x) \vee O(x))$

3. 并不是所有的学生选修了历史和生物。

$\neg(\forall x)(\text{STUDENT}(x) \rightarrow \text{TAKES}(x, H) \wedge \text{TAKES}(x, B))$

机器人大移盒子例子：

假设在房间里， c 处有一个机器人， a 和 b 处各有一张桌子， a 桌上有一盒子。要求机器人从 c 处出发把盒子从 a 桌上拿到 b 桌上，然后再回到 c 处。



描述状态的谓词：

- TABLE(x)： x 是桌子
- EMPTY(y)： y 手中是空的
- AT(y, z)： y 在 z 处
- HOLDS(y, w)： y 拿着 w
- ON(w, x)： w 在 x 桌面上

变元的个体域：

- x 的个体域是 $\{a, b\}$
- y 的个体域是 $\{\text{robot}\}$
- z 的个体域是 $\{a, b, c\}$
- w 的个体域是 $\{\text{box}\}$

问题的初始状态：

- AT(robot, c)
- EMPTY(robot)
- ON(box, a)
- TABLE(a)
- TABLE(b)



问题的目标状态：

- AT(robot, c)
- EMPTY(robot)
- ON(box, b)
- TABLE(a)
- TABLE(b)

描述操作的谓词:

条件部分:

用来说明执行该操作必须具备的先决条件，用谓词公式来表示。

动作部分:

给出了该操作对问题状态的改变情况，通过在执行该操作前的问题状态中删去和增加相应的谓词来实现：

Goto(x, y):

从 x 处走到 y 处。

条件: AT(robot, x)

动作:

删除表: AT(robot, x)

添加表: AT(robot, y)

Pickup(x):

在 x 处拿起盒子。

条件: ON(box, x), TABLE(x), AT(robot, x), EMPTY(robot)

动作:

删除表: EMPTY(robot), ON(box, x)

添加表: HOLDS(robot, box)

Setdown(y):

在 x 处放下盒子。

条件: AT(robot, x), TABLE(x), HOLDS(robot, box)

动作:

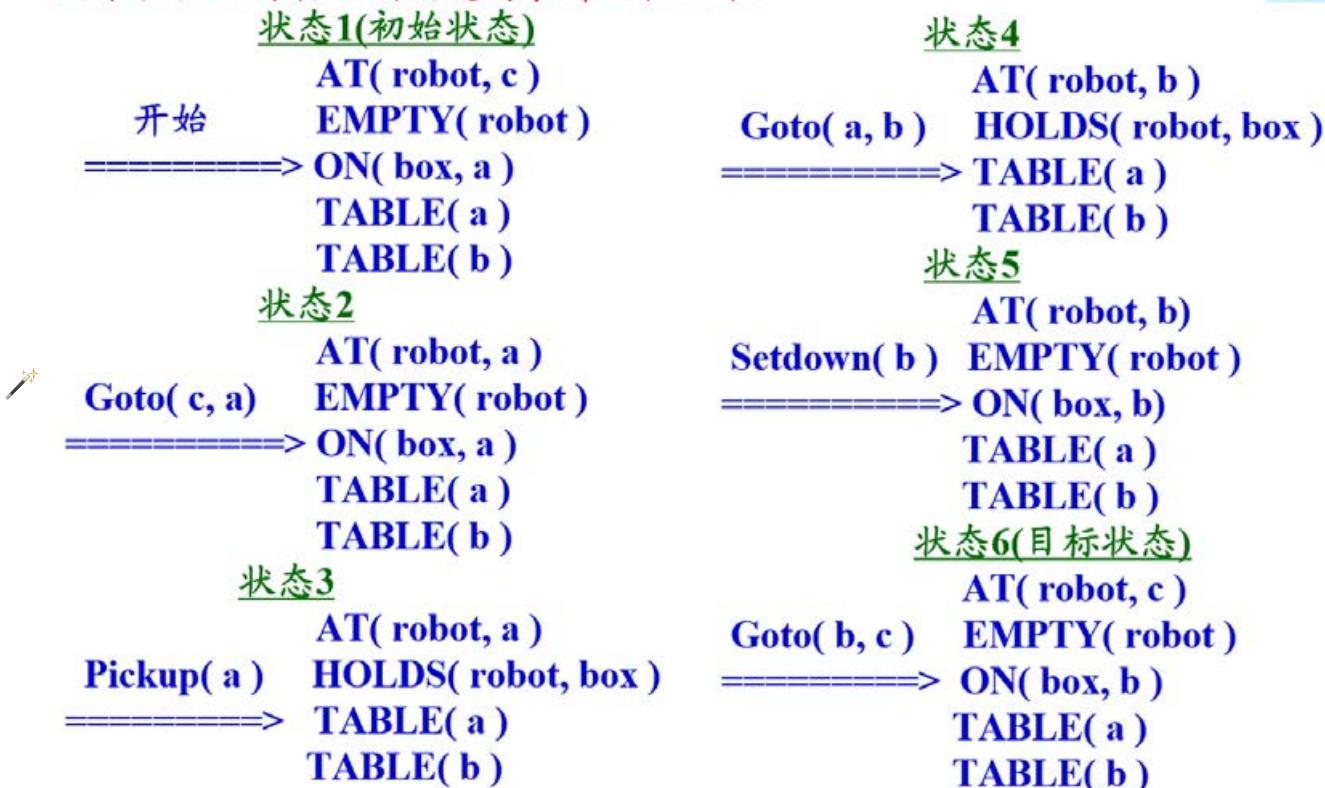
删除表: HOLDS(robot, box)

添加表: EMPTY(robot), ON(box, x)

各操作的执行方法:

机器人每执行一操作前，都要检查该操作的先决条件是否可以满足。如果满足，就执行相应的操作；否则再检查下一个操作。

这个机器人行动规划问题的求解过程如下：



2 谓词逻辑表示法的特点：

↗ 优点：

- ↗ 严密性：可以保证其演绎推理结果的正确性，可以较精确的表达知识。
- ↗ 自然性：谓词逻辑是一种接近于自然语言的形式语言。
- ↗ 通用性：拥有通用的逻辑演算方法和推理的规则。
- ↗ 易于实现：用它表示的知识易于模块化，便于知识的增删及修改，便于在计算机上实现。

↗ 局限性：

- ↗ 知识表示能力差：不便于表达和加入非确定性、启发性知识等。
- ↗ 知识库管理困难：缺乏知识的组织原则，形成的知识库不易管理。
- ↗ 组合爆炸：在其推理过程中，随着事实数目的增大及盲目的使用推例规则，有可能形成组合爆炸。
- ↗ 效率低：由于推理是根据形式逻辑进行的，把推理演算与知识含义截然分开，抛弃了表达内容中所含有的语义信息，往往使推理过程太冗长，降低了系统的效率。

2.3 产生式表示法

1943 年，Post 首先在一种计算形式体系中提出。形式上很简单，但在一定意义上模仿了人类思考的过程。60 年代开始，成为专家系统基本的知识表示方法。

2 优点：

- ↗ 适合表示事实性知识和规则性知识；
- ↗ 容易描述事实、规则以及它们的不确定性度量。

2 事实的表示：

2.1 事实的概念：

事实是断言一个语言变量的值或断言多个语言变量之间关系的陈述句。

语言变量的值：例如，「雪是白的」

语言变量之间的关系：例如，「王峰热爱祖国」

2.2 事实的表示：

确定性知识：

(对象, 属性, 值)

例 (snow, color, white) 或 (雪, 颜色, 白)。其中, 对象就是语言变量。

(关系, 对象1, 对象2)

例 (love, Wang Feng, country) 或 (热爱, 王峰, 祖国)

非确定性知识：

(对象, 属性, 值, 可信度因子)

其中, 「可信度因子」是指该事实为真的相信程度, 可用 [0, 1] 之间的一个实数来表示。

3 规则的表示：

3.1 规则的产生式表示形式常称为产生式规则。

产生式的基本形式：



P 是产生式的前提（前件），它给出了该产生式可否使用的先决条件。

Q 是一组结论或操作（后件），它指出当 P 满足时，应该推出的结论或应该执行的动作。

3.2 产生式的简例：

「如果王宏是计算机系学生，则王宏会编程序」

3.3 可用产生式表示为：

IF 该学生是计算机专业 THEN 该学生会编程序

也分为确定性规则和不确定性规则

产生式系统的基本结构：

综合数据库 DB (Data Base) :

存放推理过程的各种当前信息。如：问题的初始状态；输入的事实；中间结论及最终结论

作为推理过程选择可用规则的依据。

推理过程中某条规则是否可用，是通过该规则的前提与 DB 中的已知事实的匹配来确定的。

可匹配的规则称为可用规则。利用可用规则进行推理，将会得到一个结论。

该结论若不是目标，将作为新的事实放入 DB，成为以后推理的已知事实。

规则库 RB (Rule Base, 或知识库 KB, Knowledge Base)

作用：

用于存放推理所需要的所有规则，是整个产生式系统的知识集。是产生式系统能够进行推理的根本。

要求：

知识的完整性、一致性、准确性、灵活性和可组织性

控制系统 (Control system)，亦称推理机：

控制系统的主要作用：

用于控制整个产生式系统的运行，决定问题求解过程的推理线路。

控制系统的主要任务：

选择匹配：按一定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。匹配是指把所选规则的前提与综合数据库中的已知事实进行比较，若事实库中存的事实与所选规则前提一致，则称匹配成功，该规则为可用；否则，称匹配失败，该规则不可用。

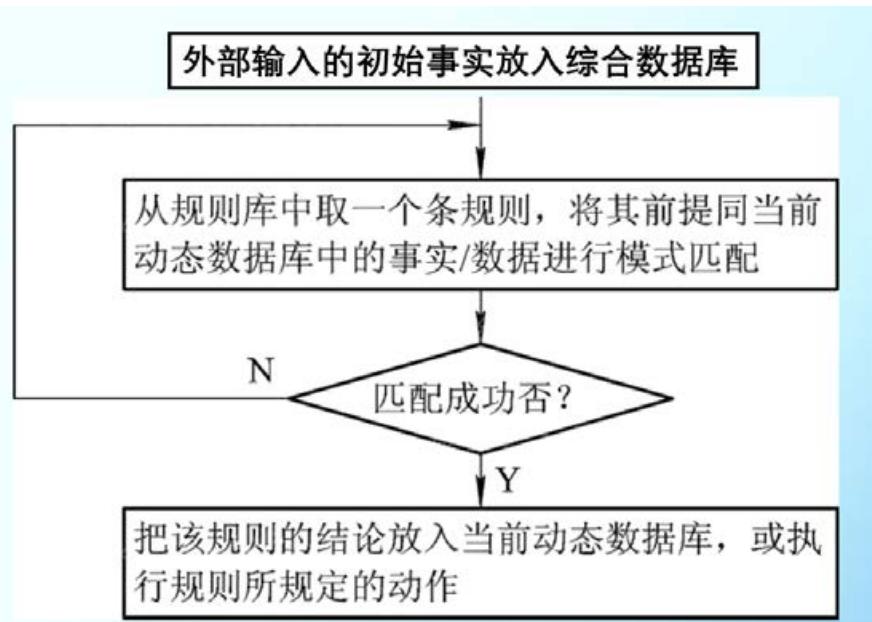
冲突消解：对匹配成功的规则，按照某种策略从中选出一条规则执行。

执行操作：对所执行的规则，若其后件为一个或多个结论，则把这些结论加入综合数据库；若其后件为一个或多个操作时，执行这些操作。

终止推理：检查综合数据库中是否包含有目标，若有，则停止推理。

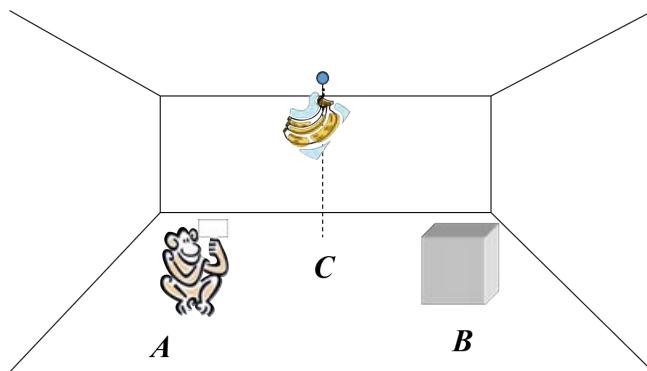
路径解释：在问题求解过程中，记住应用过的规则序列，以便最终能够给出问题的解的路径。

产生式系统的运行过程：



2 猴子摘香蕉例子：

假设房间里面有一只猴子（即机器人），位于 a 处。在 b 处上方的天花板上有一串香蕉，猴子想吃，但摘不到。房间 c 处还有一个箱子，如果猴子站在箱子上，就可以摸着天花板。



综合数据库: (M, B, Box, On, H)

M : 猴子的位置

B : 香蕉的位置

Box : 箱子的位置

$On = 0$: 猴子在地板上

$On = 1$: 猴子在箱子上

$H = 0$: 猴子没有抓到香蕉

$H = 1$: 猴子抓到了香蕉

初始状态: $(a, c, b, 0, 0)$

结束状态: $(c, c, c, 1, 1)$

规则集:

$r_1 : \text{IF}(x, y, z, 0, 0) \text{ THEN}(w, y, z, 0, 0)$ (猴子移动)

$r_2 : \text{IF}(x, y, x, 0, 0) \text{ THEN}(z, y, z, 0, 0)$ (猴子推箱子)

$r_3 : \text{IF}(x, y, x, 0, 0) \text{ THEN}(x, y, x, 1, 0)$ (猴子爬箱子)

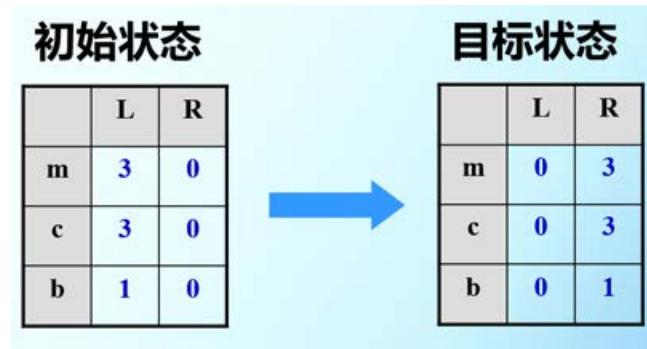
$r_4 : \text{IF}(x, y, x, 1, 0) \text{ THEN}(x, y, x, 0, 0)$ (猴子下箱子)

$r_5 : \text{IF}(x, x, x, 1, 0) \text{ THEN}(x, x, x, 1, 1)$ (猴子抓香蕉)

其中， x, y, z, w 为变量。

2 传教士与野人问题：

★ N 个传教士 (missionary, m) , N 个野人 (crumpler, c) , 一条船, 可同时乘坐 k 个人; 要求在任何时刻, 在河的两岸, 传教士和野人同时存在时, 传教士的人数不能少于野人的人数。问: 如何过河? (以 $N = 3$, $k = 2$ 为例求解)



★ 以为左右两岸状态可以互推, 所以可以只记左岸的状态

★ 综合数据库: $(m, c, b), 0 \leq m \leq 3, 0 \leq c \leq 3, b \in \{0, 1\}$

★ 初始状态: $(3, 3, 1)$

★ 目标状态: $(0, 0, 0)$

2.4 语义网络表示法

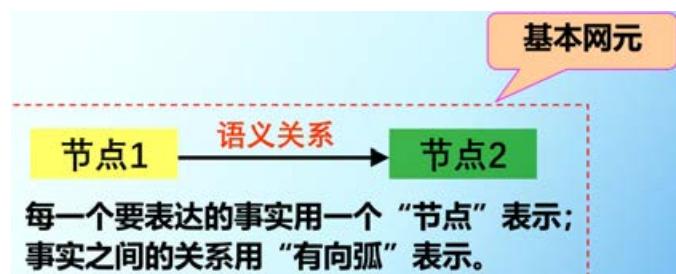
2.4.1 语义网络定义:

★ 通过概念及语义关系来表示知识的一种网络图, 它是一个带标注的有向图。

★ 图中的各个节点表示各种概念、事物、对象、行为、状态等

★ 图中的有向弧表示节点间的联系或关系。

★ 一般由一些最基本的语义单元 (语义基元) 组成, 可用三元组来表示: (节点 1, 弧, 节点 2), 也可用有向图表示:



例子: 小李和小王是朋友

语义网络表示:



一阶谓词逻辑表示:

定义谓词: $\text{Friend}(x, y)$ 。定义个体词: Li : 小李; $Wang$: 小王表示为= $\text{Friend}(Li, Wang)$

产生式表示: (Friend , Li , $Wang$)

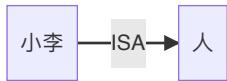
把多个基本网元用相应的语义联系关联在一起时, 就可得到一个语义网络。

语义网络中的节点还可以是一个语义子网络, 所以, 语义网络实质上是一种多层次的嵌套结构。

基本语义关系:

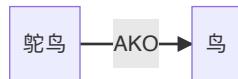
实例关系 ISA:

体现的是「具体与抽象」的概念, 含义为「是一个」, 表示一个事物是另一个事物的一个实例。



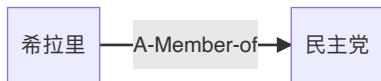
分类关系 AKO:

亦称泛化关系, 体现的是「子类与超类」的概念, 含义为「是一种」, 表示一个事物是另一个事物的一种类型。



成员关系 A - Member - of:

体现的是「个体与集体」的关系, 含义为「是一员」, 表示一个事物是另一个事物的一个成员。



以上关系一个最主要的特征是属性的继承性, 处在具体层的节点可以继承所有抽象层节点的所有属性。

聚类关系 (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。常用的包含关系是:

Part-of: 含义为「是一部分」, 表示一个事物是另一个事物的一部分。

聚类关系与实例、分类、成员关系的主要区别:

聚类关系一般不具备属性的继承性。如上例, 手不一定具有人的各种属性。

属性关系:

指事物和其属性之间的关系。常用的有：

- Have: 含义为「有」，表示一个结点具有另一个结点所描述的属性。
- Can: 含义为「能」、「会」，表示一个结点能做另一个结点的事情。

时间关系：

指不同事件在其发生时间方面的先后次序关系，常用的时间关系有：

- Before: 含义为「在 … 前」
- After: 含义为「在 … 后」

位置关系：

指不同事物在位置方面的关系，常用的有：

- Located-on: 含义为「在 … 上面」
- Located-under: 含义为「在 … 下面」
- Located-at: 含义为「在 … 」
- Located-inside: 含义为「在 … 内」
- Located-outside: 含义为「在 … 外」

相近关系：

指不同事物在形状、内容等方面相似或接近。常用的相近关系有：

- Similar-to: 含义为「相似」
- Near-to: 含义为「接近」

2 事物和概念的表示：

表示一元关系：

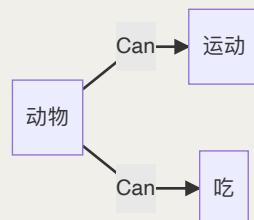
一元关系：

- 指可以用一元谓词 $P(x)$ 表示的关系。谓词 P 说明实体的性质、属性等。
- 描述的是一些最简单、最直观的事物或概念
- 常用：「是」、「有」、「会」、「能」等语义关系来说明。如，「雪是白的」。

一元关系的描述：

- 语义网络一般表示的是二元关系。
- 结点 1 表示实体，结点 2 表示实体的性质或属性等，弧表示语义关系。

用语义网络表示「动物能运动、会吃」。



表示二元关系：

二元关系：

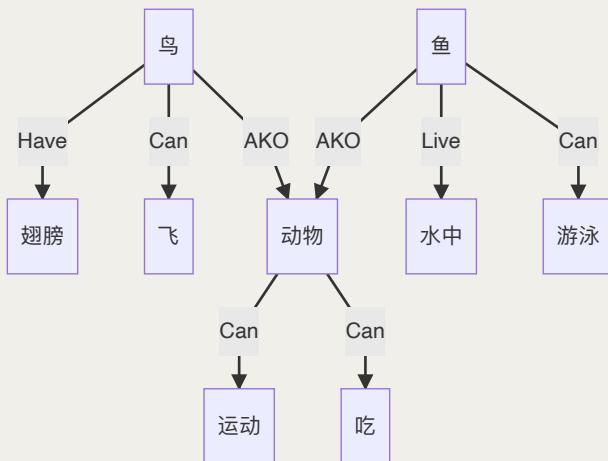
指可用二元谓词 $P(x, y)$ 表示的关系。其中， x, y 为实体， P 为实体之间的关系。

💡 二元关系的表示：

- 📎 单个二元关系可直接用一个基本网元来表示。
- 📎 复杂关系，可通过一些相对独立的二元或一元关系的组合来实现。

用语义网络表示：

- 📎 动物能运动、会吃。
- 📎 鸟是一种动物，鸟有翅膀、会飞。
- 📎 鱼是一种动物，鱼生活在水中、会游泳。



⚡ 表示多元关系：

💡 多元关系：

可用多元谓词 $P(x_1, x_2, \dots)$ 表示的关系。其中， x_1, x_2, \dots 为实体，谓词 P 说明这些实体之间的关系。

💡 多元关系的表示：

用语义网络表示多元关系时，可把它转化为一个或多个二元关系的组合，把这种多元关系表示出来。

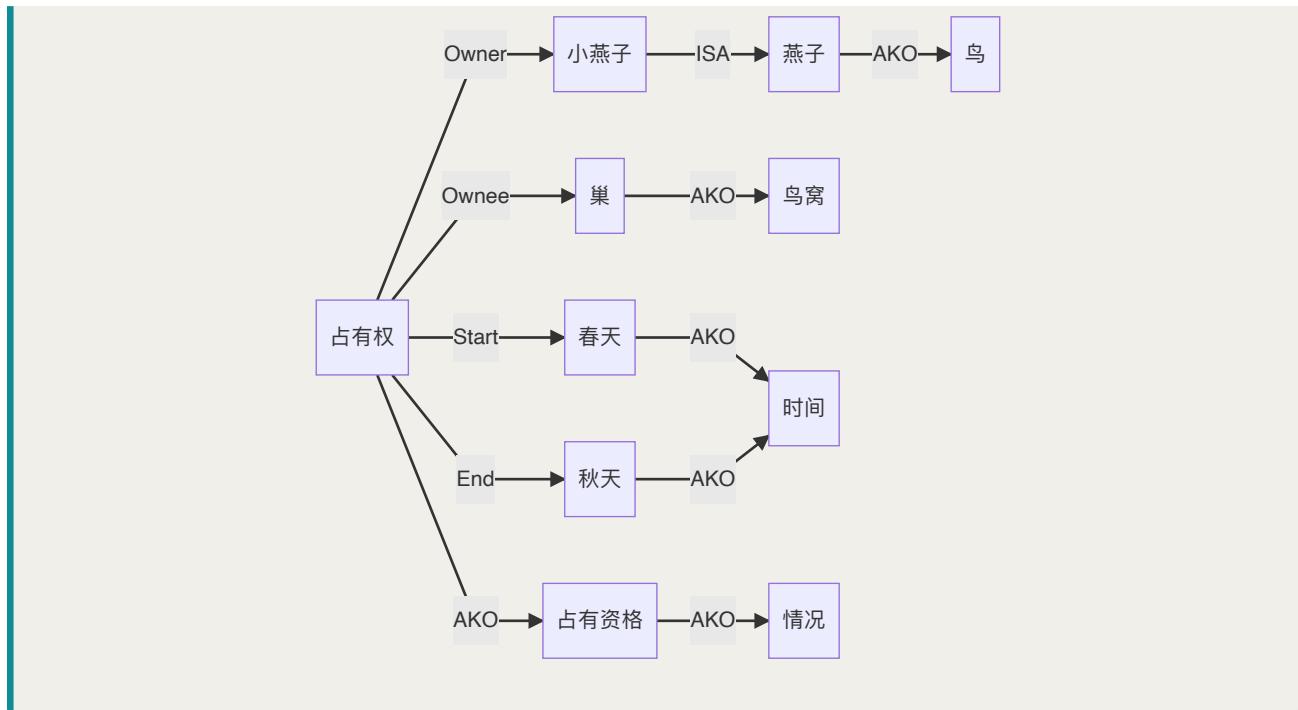
💡 情况的表示：

⚡ 表示方法：西蒙提出了增加情况和动作结点的描述方法。

用语义网络表示 「小燕子这只燕子从春天到秋天占有一个巢」

✗ 可以把占有用一条弧来表示，但在这种表示方法下，占有关系就无法表示了（时间）。

✓ 需要设立一个占有权结点，表示占有物和占有时间等。

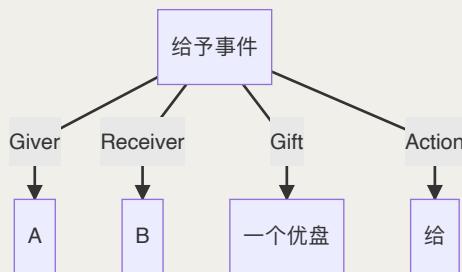


事件和动作的表示：

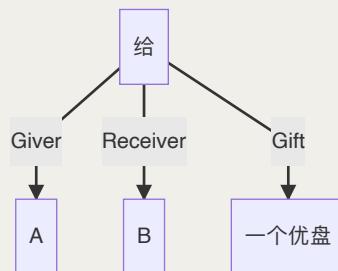
- 用这种方法表示事件或动作时，需要设立一个事件节点或动作结点。
- 事件节点由一些向外引出的弧来指出事件行为及发出者与接受者。
- 动作结点由一些向外引出的弧来指出动作的主体与客体。

用于语义网络表示「A 给 B 一个优盘」：

用事件节点表示：



用动作节点表示：



基于语义网络的推理：

语义网络的推理过程主要有：

💡 继承:

💡 是指把对事物的描述从抽象结点传递到实例结点。通过继承可以得到所需结点的一些属性值，它通常是沿着 ISA、AKO 等继承弧进行的。

💡 继承过程:

1. 建立一个结点表（队列），用来存放待求解结点和所有以 ISA、AKO 等继承弧与此结点相连的那些结点。初始情况下，表中只有待求解结点。
2. 检查表中的第一个结点是否是有继承弧。如果有，就把该弧所指的所有结点放入结点表的末尾，记录这些结点的所有属性，并从结点表中删除第一个结点。如果没有继承弧，仅从结点表中删除第一个结点。
3. 重复 2，直到结点表为空。此时，记录下来的所有属性都是待求解结点继承来的属性。

💡 匹配:

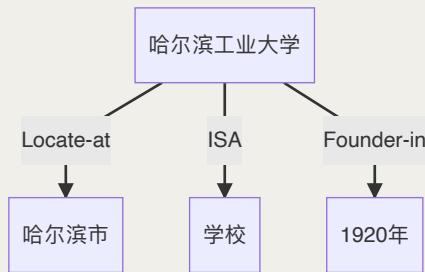
💡 是指在知识库的语义网络中寻找与待求解问题相符的语义网络模式。

💡 匹配的过程:

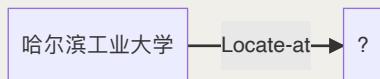
1. 根据待求解问题的要求构造一个网络片断，该网络片断中有些结点或弧的标识是空的，称为询问处，它反映的是待求解的问题。
2. 根据该语义片断到知识库中去寻找所需要的信息。
3. 当待求解问题的网络片断与知识库中的某语义网络片断相匹配时，则与询问处相匹配的事实就是问题的解。

设在语义网络系统的知识库中存在以下事实的语义网络 「哈尔滨工业大学是一所学校，位于哈尔滨市，成立于1920年」，假若要求解的问题是：哈尔滨工业大学位于哪个城市？

知识库中关于哈尔滨工业大学的语义网络：



待求解问题的语义网络片段：



2.5 框架表示法

💡 1975 年，Minsky 提出了框架理论。他根据人们在理解情景、故事时提出的心理学模型，认为人的知识以框架结构存在人脑中。

人们对现实世界中各种事物的认识都是以一种类似于框架的结构存储在记忆中的，当遇到一个新事物时，就从记忆中找出一个合适的框架，并根据新的情况对其细节加以修改、补充，从而形成对这个新事物的认识。例如，对饭店、教室等的认识。

2 框架表示法概述：

2.1 框架：

- 人们认识事物的一种通用的数据结构形式，由框架名、槽名、侧面名、值组成。
- 当新情况发生时，只要把新数据加入到该通用数据结构中，便可形成一个具体的实体（类）
- 框架由若干个「槽」组成，每个「槽」又划分为若干个「侧面」
- 一个「槽」描述对象的一个方面属性
- 一个「侧面」描述相应属性的一个方面。

2.2 实例框架：

对于一个框架，当人们把观察或认识到的具体细节填入后，就得到了该框架的一个具体实例，框架的这种具体实例被称为实例框架。

2.3 框架系统：

在框架理论中，框架是知识的基本单位，把一组有关的框架连结起来使可形成一个框架系统。

2.4 框架系统推理：

由框架之间的协调来完成。

3 横向联系：

是指那种以另外一个框架名作为一个槽的槽值或侧面值所建立起来的框架之间的联系。如 Student 框架与 S-Address 框架之间就是横向联系。

```
Frame < Student >
  Name: Unit (Last-name, First-name)
  Sex: Area (male, female)
    Default: male          // 缺省
  Age: Unit (Years)
    If-Needed: Ask-Age     // 询问赋值
  Address: < S-Address >
```

4 纵向联系：

是指那种具有继承关系的上下层框架之间的联系。如：学生可按照接受教育的层次分为本、硕和博。每类学生又可按照所学专业的不同划分。纵向联系通过预定义槽名 AKO 和 ISA 等来实现。

硕士生框架

Frame < Master >

AKO: < Student >	// 预定义槽名
Major: Unit (Major)	// 专业
If-Needed: Ask - Major	// 询问赋值
If-Added: Check-Major	// 后继处理

2 特性继承过程：

- ／＼ 通过 ISA、AKO 链来实现。
- ／＼ 当需要查询某一事物的某个属性，且描述该事物的框架未提供其属性值时，系统就沿 ISA 和 AKO 链追溯到具有相同槽的类或超类框架。
- ／＼ 如果该槽提供有 Default 侧面值，就继承该默认值作为查询结果返回。
- ／＼ 如果该槽提供有 If-Needed 侧面供继承，则执行 If-Needed 操作，去产生一个值作为查询结果。
- ／＼ 如果对某个事物的某一属性进行了赋值或修改操作，则系统会自动沿 ISA 和 AKO 链追溯到具有相应的类或超类框架，去执行 If-Added 操作，作相应的后继处理。

If-Needed 与 If-Added 过程的区别（激活时机和操作目的不同）：

- ／＼ If-Needed 操作是在系统试图查询某个事物框架中未记载的属性值时激活，并根据查询需求，被动地即时产生所需要的属性值。
- ／＼ If-Added 操作是在系统对某个框架的属性作赋值或修改工作后激活，目的在于通过这些后继处理，主动做好配套操作，以消除可能存在的不一致。

3 匹配和填槽：

框架的匹配实际上是通过对相应槽的槽名和槽值逐个进行比较，并利用继承关系来实现

4 优点：

- ／＼ 结构性：最突出特点是善于表示结构性知识，把知识的内部结构关系以及知识间的特殊联系表示出来
- ／＼ 深层性：框架表示法不仅可以从多个方面、多重属性表示知识，而且还可以通过 ISA、AKO 等槽以嵌套结构分层地对知识进行表示，因此能用来表达事物间复杂的深层联系。
- ／＼ 继承性：在框架系统中，下层框架可以继承上层框架的槽值，也可以进行补充和修改，这样既减少知识冗余，又较好地保证了知识的一致性
- ／＼ 自然性：框架能把与某个实体或实体集相关特性都集中在一起，从而高度模拟了人脑对实体多方面、多层次的存储结构，直观，自然，易于理解

5 缺点：

- ／＼ 缺乏框架的形式理论
- ／＼ 缺乏过程性知识表示：

框架系统不便于表示过程性知识，缺乏如何使用框架中知识的描述能力。框架推理过程需要用到一些与领域无关的推理规则，而这些规则在框架系统中又很难表达。

／＼ 清晰性难以保证（各框架本身的数据结构不一定相同）

3 确定性推理

3.1 推理概述

／＼ 推理的定义：

推理就是按照某种策略从已有事实和知识推出结论的过程。

／＼ 心理学对推理有两种解释：

／＼ 从结构的角度：

／＼ 判断是在概念的基础上进行的，所揭示的是概念之间联系和关系

／＼ 推理由两个以上的判断所组成，把判断定义为对客观事物做出肯定或否定的思维活动

／＼ 从过程的角度：

认为推理是在给定信息和已有知识的基础上的一系列加工操作，提出了如下人类推理的公式：

$$y = F(x, k)$$

其中， x 为推理时给出的信息， k 为推理时可用的领域知识和特殊事例， F 为可用的一系列操作， y 为推理过程所得到的结论。

／＼ 推理的机器实现：

／＼ 人工智能中的推理是由推理机完成的。所谓推理机，是指系统中用来实现推理的那段程序。

／＼ 根据推理所用知识的不同，推理方式和推理方法的不同，推理机的构造也有所不同。

／＼ 按推理的逻辑基础分类

／＼ 演绎推理：

／＼ 从已知的一般性知识出发，推理出适合于某种个别情况的结论过程。即从一般到个别的推理。

／＼ 常用形式：三段论法（大前提、小前提、结论）

／＼ 大前提：是已知的一般性知识或推理过程得到的判断

／＼ 小前提：是关于某种具体情况或某个具体实例的判断

／＼ 结论：是由大前提推出的，并且适合于小前提的判断

／＼ 是在已知领域内的一般性知识的前提下，通过演绎求解一个具体问题或者证明一个结论的正确性。它所得出的结论实际上早已蕴含在一般性知识的前提中，演绎推理只不过是将已有事实揭露出来，因此它不能增殖新知识。

／＼ 归纳推理：

／＼ 从大量特殊事例出发，归纳出一般性结论的推理过程。即从个别到一般的推理过程。

／＼ 所推出的结论是没有包含在前提内容中的。这种由个别事物或现象推出一般性知识的过程，是增殖新知识的过程。

／＼ 完全归纳推理：

对公司生产的每台计算机进行质量检查，并且都合格，则推理出结论「该公司生产的计算机质量合格」。

💡 不完全归纳推理：

随机抽查检查合格。

💡 枚举归纳推理：

如果已知某类事物的有限可数个具体事务都具有某种属性，则推出该类事物都具有此种属性。

💡 类比归纳推理：

在两个或两类事物有许多相同的或相似的基础上，推出他们在其他属性上也有相同或相似。

一位计算机维修员，从书本知识，到通过大量实例积累经验，是一种归纳推理方式。运用这些一般性知识去维修计算机的过程则是演绎推理。

💡 按所用知识的确定性分类：

💡 确定性推理：

推理时所有用的知识和证据都是确定的，推出的结论也是确定的，其真值或者为真或者为假，没有第三种情况出现。

💡 不确定性推理：

推理时所用的知识或证据不是确定的，推出的结论也不确定的。

💡 按推理中所用知识是否具有启发性分类：

💡 启发式推理：

推理过程中应用与问题有关的启发性知识，即解决问题的策略、技巧及经验，以加快推理过程，提高搜索效率。

💡 非启发式推理：

在推理过程中，不运用启发性知识，只按照一般的控制逻辑进行推理。这种方法缺乏对求解问题的针对性，所以推理效率较低，容易出现「组合爆炸」问题。

💡 推理的控制策略：

推理过程不仅依赖于所用的推理方法，同时也依赖于推理的控制策略。推理的控制策略是指如何使用领域知识使推理过程尽快达到目标的策略。

💡 控制策略的分类：

由于智能系统的推理过程一般表现为一种搜索过程，推理的控制策略又可分为推理策略和搜索策略：

💡 推理策略：

💡 推理方向控制策略用于确定推理的控制方向，可分为正向推理、逆向推理、混合推理及双向推理。

💡 求解策略是指仅求一个解，还是求所有解或最优解等。

- 限制策略是指对推理的深度、宽度、时间、空间等进行的限制。
- 冲突消解策略是指当推理过程有多条知识可用时，如何从这多条可用知识中选出一条最佳知识用于推理的策略。

搜索策略：

主要解决推理线路、推理效果、推理效率等问题。

3.2 产生式系统

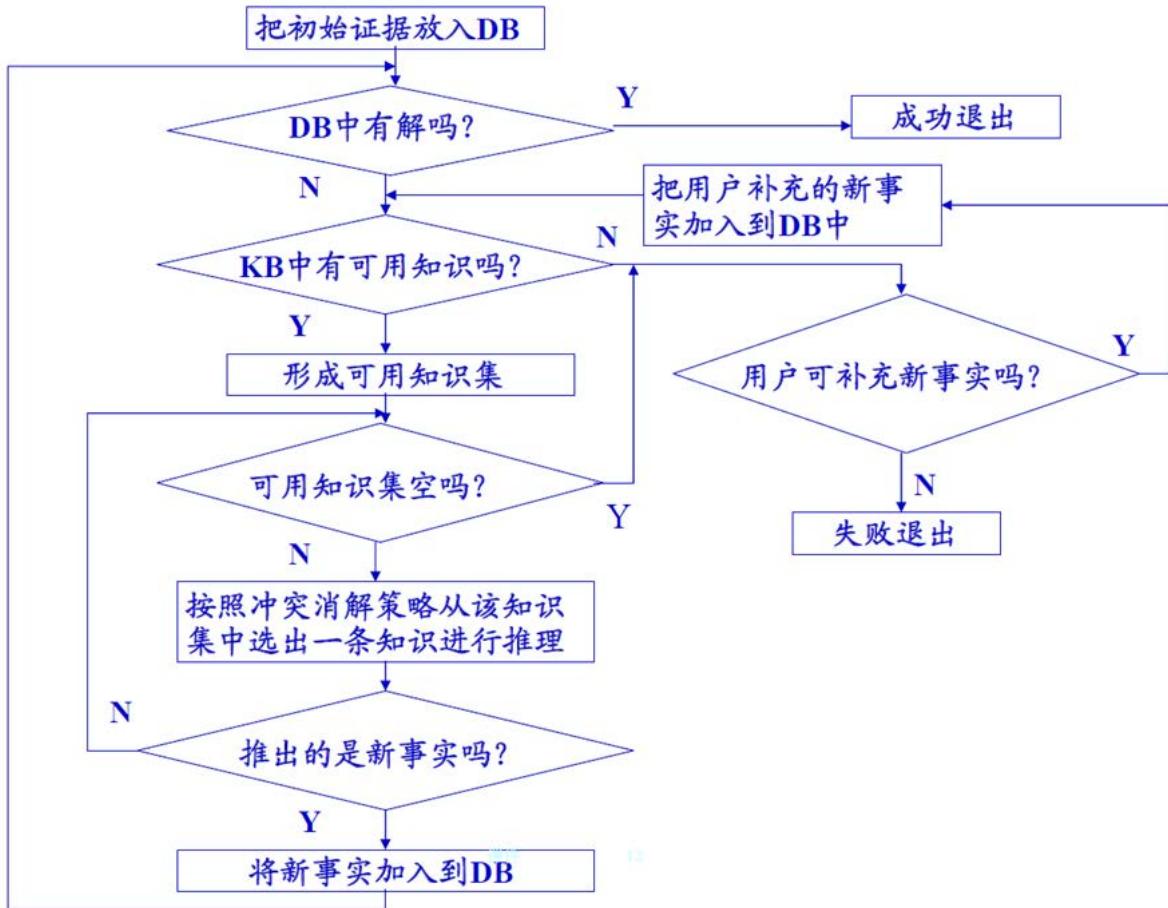
正向推理：

从已知事实出发、正向使用规则，亦称为数据驱动推理或前向链推理。算法描述：

- 把用户提供的初始证据放入综合数据库
- 检查综合数据库中是否包含了问题的解，若已包含，则求解结束，并成功推出；否则执行下一步
- 检查知识库中是否有可用知识，若有，形成当前可用知识集，执行下一步；否则转 5
- 按照某种冲突消解策略，从当前可用知识集中选出一条规则进行推理，并将推出的新事实加入综合数据库中，然后转 2
- 询问用户是否可以进一步补充新的事实，若可补充，则将补充的新事实加入综合数据库中，然后转 3；否则表示无解，失败退出

知识库 KB 放规则（知识）

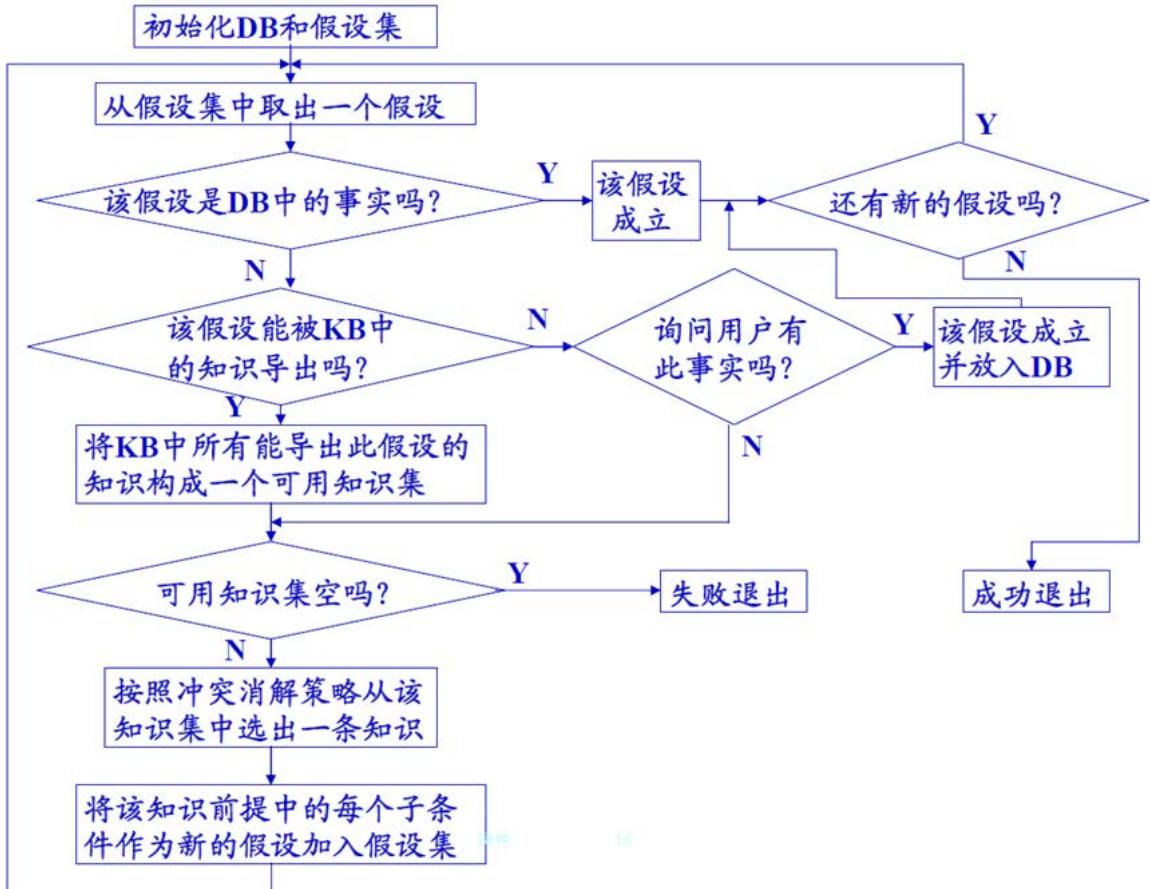
综合数据库 DB 放事实



逆向推理:

从某个假设目标出发，逆向使用规则，亦称为目标驱动推理或逆向链推理。算法描述：

1. 将要求证的目标（称为假设）构成一个假设集
2. 从假设集中选出一个假设，检查该假设是否在综合数据库中，若不在则执行下一步；否则则该假设成立，此时，若假设集为空，则成功退出，否则仍执行 2
3. 检查该假设是否可由知识库的某个知识导出，若能由某个知识导出，则执行下一步；否则先则询问用户该假设是否为可由用户证实的原始事实，若是，该假设成立，并将其放入综合数据库，转 2，若不是，则转 5
4. 将知识库中可以导出该假设的所有知识构成一个可用知识集
5. 检查可用知识集是否为空，若是，失败退出；否则执行下一步
6. 按冲突消解策略从可用知识集中取出一个知识，继续
7. 将该知识的前提中的每个子条件都作为新的假设放入假设集，然后转 2



假设知识库中包含以下 2 条规则：

$r_1 : \text{IF } B \text{ THEN } C$

$r_2 : \text{IF } A \text{ THEN } B$

已知初始证据 A , 求证目标 C 。

推理过程如下：

1. 推理开始前，综合数据库和假设集均为空。
2. 推理开始后，先将初始证据 A 和目标 C 分别放入综合数据库和假设集，然后从假设集中取出一个假设 C ，查找 C 是否为综合数据库中的已知事实，回答为 NO。
3. 再检查 C 是否能被知识库中的知识所导出，发现 C 可由 r_1 导出，于是 r_1 被放入可用知识集。由于知识库中只有 r_1 可用，故可用知识集中仅含 r_1 。
4. 接着从可用知识集中取出 r_1 ，将其前提条件 B 作为新的假设放入假设集。
5. 从假设集中取出 B ，检查 B 是否为综合数据库中的事实，回答为 NO。再检查 B 是否能被知识库中的知识所导出，发现 B 可由 r_2 导出，于是 r_2 被放入可用知识集。由于知识库中只有 r_2 可用，故可用知识集中仅含 r_2 。
6. 从可用知识集中取出 r_2 ，将其前提条件 A 作为新的假设放入假设集。然后从假设集中取出 A ，检查 A 是否为综合数据库中的实事，回答为 YES。
7. 说明该假设成立，由于无新的假设，故推理过程成功结束，于是目标 C 得证。

正向推理的特性：

主要优点是比较直观

主要缺点是推理无明确的目标，求解问题时可能会执行许多与解无关的操作，导致推理效率较低

逆向推理的特性：

主要优点是不必寻找和使用那些与假设目标无关的信息和规则，推理过程的目标明确

主要缺点是当用户对解的情况认识不清时，由系统自主选择假设目标的盲目性比较大，若选择不好，会影响系统效率

3.3 自然演绎推理

3.3.1 一阶谓词逻辑基础

谓词公式的解释：

设 D 是谓词公式 P 的非空个体域，若对 P 中的常量，函数和谓词按如下规定赋值：

为每个个体常量指派 D 中的一个元素

为每个 n 元函数指派一个从 D^n 到 D 的一个映射，其中

$$D^n = \{(x_1, x_2, \dots, x_n) \mid x_1, x_2, \dots, x_n \in D\}$$

为每个 n 元谓词指派一个从 D^n 到 $\{F, T\}$ 的映射，则称这些指派为 P 在 D 上的一个解释。

谓词公式的永真性：

如果谓词公式 P 对非空个体域 D 上的任一解释都取得真值 T ，则称 P 在 D 上是永真的

如果 P 在任何非空个体域上都是永真的，则称 P 是永真

谓词公式的可满足性：

对于谓词公式 P ，如果至少存在 D 上的一个解释，使公式 P 在此解释下的真值为 T ，则称公式 P 在 D 上是可满足的。

谓词公式的永假性：

如果谓词公式 P 对非空个体域 D 上的任一解释都取真值 F ，则称 P 在 D 上是永假的

如果 P 在任何非空个体域上均是永假的，则称 P 永假

谓词公式的等价性：

设 P 与 Q 是 D 上的两个谓词公式，若对 D 上的任意解释， P 与 Q 都有相同的真值，则称 P 与 Q 在 D 上是等价的。如果 D 是任意非空个体域，则称 P 与 Q 是等价的，记作 $P \Leftrightarrow Q$ 。

常用的等价式有：

双重否定律： $\neg\neg P \Leftrightarrow P$

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： $P \vee (Q \wedge R) \Leftrightarrow (P \vee Q) \wedge (P \vee R), P \wedge (Q \vee R) \Leftrightarrow (P \wedge Q) \vee (P \wedge R)$

狄摩根定律： $\neg(P \vee Q) \Leftrightarrow \neg P \wedge \neg Q, \neg(P \wedge Q) \Leftrightarrow \neg P \vee \neg Q$

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:

$$P \rightarrow Q \Leftrightarrow \neg P \vee Q \quad P \leftrightarrow Q \Leftrightarrow (P \rightarrow Q) \wedge (Q \rightarrow P) \quad P \leftrightarrow Q \Leftrightarrow (P \wedge Q) \vee (\neg Q \wedge \neg P)$$

★ 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: $(\forall x)(P \wedge Q) \Leftrightarrow (\forall x)P \wedge (\forall x)Q, \quad (\exists x)(P \vee Q) \Leftrightarrow (\exists x)P \vee (\exists x)Q$

2 永真蕴含式:

对谓词公式 P 和 Q , 如果 $P \rightarrow Q$ 永真, 则称 P 永真蕴含 Q , 且称 Q 为 P 的逻辑结论, P 为 Q 的前提, 记作 $P \Rightarrow Q$ 。

常用的永真蕴含式如下:

化简式 $Q \wedge P \Rightarrow P, \quad P \wedge Q \Rightarrow Q$

附加式 $P \Rightarrow P \vee Q, \quad Q \Rightarrow P \vee Q$

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 $P \rightarrow Q, Q \rightarrow R \Rightarrow P \rightarrow R$

二难推理 $P \vee Q, P \rightarrow R, Q \rightarrow R \Rightarrow R$

全称固化 $(\forall x)P(x) \Rightarrow P(y)$

其中, y 是个体域中的任一个体, 依此可消去谓词公式中的全称量词。

存在固化 $(\exists x)P(x) \Rightarrow P(y)$

其中, y 是个体域中某一个可以使 $P(y)$ 为真的个体, 依此可消去谓词公式中的存在量词。

3.3.2 置换和合一

2 在不同谓词公式中, 往往会出现谓词名相同但其个体不同的情况, 此时推理过程是不能直接进行匹配的, 需要先进行置换:

$$W(a) \text{ 和 } W(x) \rightarrow Q(x)$$

对谓词 $W(a)$ 与 $W(x)$ 谓词名相同, 但个体不同, 不能直接进行推理。

2 要使用假言推理, 首先需要找到项 a 对变元 x 的置换, 使 $W(a)$ 与 $W(x)$ 一致。这种寻找项对变元的置换, 使谓词一致的过程叫做合一的过程。

2 置换:

★ 在一个谓词公式中用置换项去替换变量, 形如 $\{t_1/x_1, t_2/x_2, \dots, t_n/x_n\}$ 的有限集合。

★ 其中, t_1, t_2, \dots, t_n 是项, x_1, x_2, \dots, x_n 是互不相同的变元, t_i/x_i 表示用 t_i 替换 x_i 。

★ 要求 t_i 与 x_i 不能相同, x_i 不能循环地出现在另一个 t_i 中。

★ 通常, 置换是用希腊字母 $\theta, \sigma, \alpha, \lambda$ 等来表示的。

$\{a/x, c/y, f(b)/z\}$ 是一个置换。

但 $\{g(z)/x, f(x)/z\}$ 不是一个置换。原因是它在 x 与 z 之间出现了循环置换现象。即当用 $g(z)$ 置换 x , 用 $f(x)$ 置换 z 时, 既没有消去 x , 也没有消去 z 。

若改为 $\{g(a)/x, f(x)/z\}$ 即可, 原因是用 $g(a)$ 置换 x , 用 $f(x)$ 置换 z , 若再用一次置换, 用 $g(a)$ 置换 x , 最终原 x 和 z 被 $g(a)$ 和 $f(g(a))$ 置换, 则经过有限次置换消去 x 和 z 。

设 $\theta = \{t_1/x_1, t_2/x_2, \dots, t_n/x_n\}$ 是一个置换, F 是一个谓词公式, 把公式 F 中出现的所有 x_i 换成 $t_i (i = 1, 2, \dots, n)$, 得到一个新的公式 G , 称 G 为 F 在置换 θ 下的例示, 记作 $G = F\theta$ 。

置换的合成: 设

$$\begin{aligned}\theta &= \{t_1/x_1, t_2/x_2, \dots, t_n/x_n\} \\ \lambda &= \{u_1/y_1, u_2/y_2, \dots, u_n/y_n\}\end{aligned}$$

是两个置换, 则将集合

$$\{t_1\lambda/x_1, t_2\lambda/x_2, \dots, t_n\lambda/x_n, u_1/y_1, u_2/y_2, \dots, u_n/y_n\}$$

中符合下列条件的元素删除:

先将 x_i 置换为 t_i , 再通过 λ 置换, 表示为 $t_i\lambda/x_i$

如果最终置换单元 x_i , 相当于没有置换

先将 x_i 置换为 t_i , 再通过 λ 置换, 表示为 $t_i\lambda/x_i$

如果最终置换单元 x_i , 相当于没有置换

如此得到的集合仍然是一个置换, 该置换称为 θ 与 λ 的合成, 记作 $\theta \cdot \lambda$ 。

设 $\theta = \{f(y)/x, z/y\}, \lambda = \{a/x, b/y, y/z\}$, 求 θ 与 λ 的合成。

解: 先求出集合

$$\begin{aligned}&\{f(y)\lambda/x, z\lambda/y, a/x, b/y, y/z\} \\ &= \{f(b)/x, y/y, a/x, b/y, y/z\}\end{aligned}$$

删除无效置换得:

$$\theta \cdot \lambda = \{f(b)/x, y/z\}$$

合一:

可理解为是寻找相对应变量的置换, 使两个或多个谓词公式一致。

设有公式集 $F = \{F_1, F_2, \dots, F_n\}$, 若存在一个置换 θ , 可使 $F_1\theta = F_2\theta = \dots = F_n\theta$,

$F_1\theta = F_2\theta = \dots = F_n\theta$, 则称 θ 是 F 的一个合一, 称 F_1, F_2, \dots, F_n 是可合一的。

一般情况下, 一个公式集的合一不是唯一的。

例如, 设有公式集 $F = \{P(x, y, f(y)), P(a, g(x), z)\}$, 则 $\lambda = \{a/x, g(a)/y, f(g(a))/z\}$ 是它的一个合一。

最一般合一:

设 σ 是谓词公式集 F 的一个合一置换, 如果对 F 的任意一个合一置换 θ , 都存在一个置换 λ , 使得 $\theta = \sigma \cdot \lambda$, 则称 σ 是一个最一般 (或最简单) 合一 (most general unifier, 简记为mgu) 置换。

一个公式集的最一般合一是唯一的。

差异集 (Disagreement Set) 定义:

设 $F = \{F_1, F_2, \dots, F_n\}$ 是一个非空有限的公式集

从 F 中每个公式的第一个符号同时向右比较, 直到发现第一个不相同的符号为止

从 F 的各个公式中取出那些以第一个不一致符号开始的最大子表达式, 并以这些子表达式为元素组成一个集合 D , 称 D 为 F 的一个差异集, 也称分歧集合。

设 $S = \{P(x, f(y), z), P(x, f(a), h(b))\}$, S 有差异集 $D = \{y, a\}$

最一般合一求法:

设 S 为非空有限公式集合, 求 S 的最一般合一的算法如下:

1. 置 $k = 0, S_0 = S, \sigma_k = \varepsilon$

ε 和任何置换单元都得到该置换单元

2. 若 S_k 只含有一个谓词公式, 则算法停止, σ_k 就是要求的最一般合一置换
3. 求 S_k 的差异集 D_k
4. 若 D_k 中存在元素 x_k 和 t_k , 其中 x_k 是变元, t_k 是项且 x_k 不在 t_k 中出现, 则置 $S_{k+1} = S_k \{t_k/x_k\}, \sigma_{k+1} = \sigma_k \cdot \{t_k/x_k\}, k = k + 1$, 然后转 2;
5. 如果不存在, 算法停止, S 的最一般合一不存在。

求公式集 $S = \{P(a, x, f(g(y))), P(z, h(z, u), f(u))\}$ 的最一般合一及对应的置换:

1. $k = 0$:

$S_0 = S, \sigma_0 = \varepsilon, S_0$ 不是单元素集, 求得 $D_0 = \{a, z\}$, 其中 z 是变元, 且不在 a 中出现, 所以有

$$\sigma_1 = \sigma_0 \cdot \{a/z\} = \varepsilon \cdot \{a/z\} = \{a/z\}$$

$$S_1 = S_0 \{a/z\} = \{P(a, x, f(g(y))), P(a, h(a, u), f(u))\}$$

2. $k = 1$:

$$\begin{aligned} S_1 &\text{ 不是单元素集, 求得 } D_1 = \{x, h(a, u)\}, \\ \sigma_2 &= \sigma_1 \cdot \{h(a, u)/x\} = \{a/z\} \cdot \{h(a, u)/x\} = \{a/z, h(a, u)/x\} \\ S_2 &= S_1 \{h(a, u)/x\} = \{P(a, h(a, u), f(g(y))), P(a, h(a, u), f(u))\} \end{aligned}$$

3. $k = 2$:

$$\begin{aligned} S_2 &\text{ 不是单元素集, } D_2 = \{g(y), u\}, \\ \sigma_3 &= \sigma_2 \cdot \{g(y)/u\} = \{a/z, h(a, g(y))/x, g(y)/u\} \\ S_3 &= S_2 \{g(y)/u\} = \{P(a, h(a, g(y), f(g(y))), P(a, h(a, g(y)), f(g(y)))\} \\ &= \{P(a, h(a, g(y)), f(g(y)))\} \end{aligned}$$

4. $k = 3$:

S_3 已是单元素集, 所以 σ_3 就是 S 的最一般合一置换。

例子: 判定 $S = \{P(x, x), P(y, f(y))\}$ 是否可合一?

解:

$k = 0$:

$$S_0 = S^2 \sigma_0 = \varepsilon$$

S_0 不是单元素集, $D_0 = \{x, y\}$

$$\sigma_1 = \sigma_0 \cdot \{y/x\} = \{y/x\}$$

$$S_1 = S_0 \{y/x\} = \{P(y, y), P(y, f(y))\}$$

$k = 1$:

S_1 不是单元素集, $D_1 = \{y, f(y)\}$, 由于变元 y 在项 $f(y)$ 中出现, 所以算法停止, S 不存在最一般合一。

3.3.3 自然演绎推理方法

从一组已知为真的事实出发, 直接运用命题逻辑或谓词逻辑中的推理规则推出结论的过程。

最基本的推理规则有:

假言三段论: $P \rightarrow Q, Q \rightarrow R \Rightarrow P \rightarrow R$

假言推理: $P, P \rightarrow Q \Rightarrow Q$

拒取式: $\neg Q, P \rightarrow Q \Rightarrow \neg P$

T 规则: 在推理时, 如果前面步骤有一个或多个永真蕴涵公式 S , 则可把 S 引入推理过程。

P 规则: 在推理过程的任何步骤上都可以引入前提。

例子 1:

$$R, S, R \rightarrow T, S \wedge T \rightarrow P, P \rightarrow Q$$

求证: Q 为真。

证明:

1. $R, R \rightarrow T \Rightarrow T$ P 规则及假言推理
2. $S, T \Rightarrow S \wedge T$ 引入合取词
3. $S \wedge T, S \wedge T \rightarrow P \Rightarrow P$ T 规则及假言推理
4. $P, P \rightarrow Q \Rightarrow Q$ T 规则及假言推理

所以 Q 为真。

例子 2:

设已知如下事实：

- ↗ 如果是需要编程序的课，王程都喜欢。
- ↗ 所有的程序设计语言课都是需要编程序的课。
- ↗ C 是一门程序设计语言课。

求证：王程喜欢 C 这门课。

证明：首先定义谓词：

- $N(x)$ x 是需要编程序的课。
 $L(x, y)$ x 喜欢 y 。
 $P(x)$ x 是一门程序设计语言课

把已知事实及待求解问题用谓词公式表示如下：

- $N(x) \rightarrow L(\text{Wangcheng}, x)$
 $(\forall x)(P(x) \rightarrow N(x))$
 $P(C)$

应用推理规则进行推理：

1. $P(y) \rightarrow N(y)$ 全称固化
2. $P(C), P(y) \rightarrow N(y) \Rightarrow N(C)$ 假言推理 $\{C/y\}$
3. $N(C), N(x) \rightarrow L(\text{Wangcheng}, x) \Rightarrow L(\text{Wangcheng}, C)$ 假言推理 $\{C/x\}$

因此，王程喜欢 C 这门课。

3.4 归结演绎推理

- ℳ 归结演绎推理是一种基于鲁宾逊 (Robinson) 归结原理的机器推理技术。鲁宾逊归结原理亦称为消解原理，是鲁宾逊于 1965 年在海伯伦 (Herbrand) 理论的基础上提出的一种基于逻辑的「反证法」，它使定理证明的机械化成为现实。
- ℳ 定理证明的实质，就是要对前提 P 和结论 Q ，证明 $P \rightarrow Q$ 永真。
- ℳ 要证明 $P \rightarrow Q$ 永真，就是要证明 $P \rightarrow Q$ 在任何一个非空的个体域上都是永真的。
- ℳ 要把关于永真性的证明转化为关于不可满足性的证明。即要证明 $P \rightarrow Q$ 永真，只要能够证明 $P \wedge \neg Q$ 是不可满足的就可以了。

3.4.1 谓词公式的范式

范式是谓词公式的标准形式。

2 前束范式：

- ／＼ 设 F 是一个谓词公式，如果其中的所有量词均非否定出现在公式的最前面，而它们的辖域为整个公式，则称 F 为前束范式。任一谓词公式均可化为与其对应的前束范式。
- ／＼ 前束范式一般可写成 $(Q_1 x_1)(Q_2 x_2) \cdots (Q_n x_n)M(x_1, x_2, \dots, x_n)$ ，其中 $Q_i \in \{\forall, \exists\}$, $(i = 1, 2, \dots, n)$, $M(x_1, x_2, \dots, x_n)$ 中不含有任何量词。

例如， $(\forall x)(\forall y)(\exists z)(P(x) \wedge Q(y, z) \vee R(x, z))$ 是前束范式。

2 Skolem 标准型：

- ／＼ 从前束范式中消去全部存在量词所得到的公式，并将谓词公式化为合取范式，这时所得的式子称为原公式的 Skolem 标准型。
- ／＼ Skolem 标准型的一般形式为：

$$(\forall x_1)(\forall x_2) \cdots (\forall x_n)M(x_1, x_2, \dots, x_n)$$

其中 $M(x_1, x_2, \dots, x_n)$ 中不含有任何量词，且为合取范式。

2 子句和子句集：

原子谓词公式及其否定统称为文字：

例如， $P(x)$ 、 $Q(x)$ 、 $\neg P(x)$ 、 $\neg Q(x)$ 等都是文字。

任何文字的析取式称为子句：

例如， $P(x) \vee Q(x)$, $P(x, f(x)) \vee Q(x, g(x))$ 都是子句。

不含任何文字的子句称为空子句：

／＼ 由于空子句不含有任何文字，也就不能被任何解释所满足，因此空子句是永假的，不可满足的。

／＼ 空子句一般被记为 NIL。

／＼ 由子句或空子句所构成的集合称为子句集。

3.4.2 子句集化简

例如公式 $(\forall x)((\forall y)P(x, y) \rightarrow \neg(\forall y)(Q(x, y) \rightarrow R(x, y)))$ 其化简步骤如下：

1. 消去连接词 \rightarrow 和 \leftrightarrow ：

反复使用如下等价公式：

$$\begin{aligned} P \rightarrow Q &\Leftrightarrow \neg P \vee Q \\ P \leftrightarrow Q &\Leftrightarrow (P \wedge Q) \vee (\neg P \wedge \neg Q) \end{aligned}$$

例子经等价变化后为

$$(\forall x)(\neg(\forall y)P(x, y) \vee \neg(\forall y)(\neg Q(x, y) \vee R(x, y)))$$

2. 减少否定符号的辖域:

反复使用双重否定率: $\neg(\neg P) \Leftrightarrow P$

摩根定律: $\neg(P \wedge Q) \Leftrightarrow \neg P \vee \neg Q, \quad \neg(P \vee Q) \Leftrightarrow \neg P \wedge \neg Q$

量词转换率: $\neg(\forall x)P(x) \Leftrightarrow (\exists x)\neg P(x), \quad \neg(\exists x)P(x) \Leftrightarrow (\forall x)\neg P(x)$

将每个否定符号移到仅靠谓词的位置, 使得每个否定符号最多只作用于一个谓词上。例子经等价变换后为

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

3. 对变元标准化:

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任意变元代替, 使不同量词约束的变元有不同的名字, 消除了任何由变元引起冲突的可能。例子中两个 $\exists y$ 是不同的, 经变换后为

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

4. 化为前束范式:

把所有量词都移到公式的左边, 并且在移动时不能改变其相对顺序。例子化为前束范式后为

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

5. 消去存在量词:

消去存在量词时, 需要区分以下两种情况:

- \mathcal{D} 若存在量词不出现在全称量词的辖域内 (即它的左边没有全称量词), 只要用一个新的个体常量替换受该存在量词约束的变元, 就可消去该存在量词
- \mathcal{D} 若存在量词位于一个或多个全称量词的辖域内, 例如 $(\forall x_1) \dots (\forall x_n)(\exists y)P(x_1, x_2, \dots, x_n, y)$ 则需要用 Skolem 函数 $f(x_1, x_2, \dots, x_n)$ 替换受该存在量词约束的变元 y , 然后再消去该存在量词。

例子中存在量词 $(\exists y)$ 和 $(\exists z)$ 都位于 $(\forall x)$ 的辖域内, 因此都需要用 Skolem 函数来替换。设替换 y 和 z 的 Skolem 函数分别是 $f(x)$ 和 $g(x)$, 则替换后的式子为

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

6. 化为 Skolem 标准形:

对上述前束范式的母式应用以下等价关系 $P \vee (Q \wedge R) \Leftrightarrow (P \vee Q) \wedge (P \vee R)$, 例子化为 Skolem 标准形后为

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

7. 消去全称量词:

由于母式中的全部变元均受全称量词约束, 并且与全称量词的次序无关, 因此可省掉全称量词。但剩下的母式, 仍假设其变元是被全称量词量化的。例子消去全称量词后为:

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

8. 消去合取词:

在母式中消去所有合取词, 把母式用子句集的形式表示出来。其中, 子句集中的每一个元素都是一个子句。例子的子句集中包含以下两个子句:

$$\begin{aligned} &\neg P(x, f(x)) \vee Q(x, g(x)) \\ &\neg P(x, f(x)) \vee \neg R(x, g(x)) \end{aligned}$$

9. 更换变量名称:

对子句集中的某些变量重新命名, 使任意两个子句中不出现相同的变量名。由于任意两个不同子句的变量之间实际上不存在任何关系, 更换变量名是不会影响公式的真值的。

例子可把第二个子句集中的变元 x 更换为 y , 得到如下子句集:

$$\begin{aligned} &\neg P(x, f(x)) \vee Q(x, g(x)) \\ &\neg P(y, f(y)) \vee \neg R(y, g(y)) \end{aligned}$$

3.4.3 子句集定理

由于子句集化简过程在消去存在量词时所用的 Skolem 函数可以不同, 因此所得到的标准子句集不唯一。

重要结论:

设有谓词公式 F , 其标准子句集为 S , 则 F 为不可满足的充要条件是 S 为不可满足的。

由于连词化归律、双重否定律、狄摩根律、量词转化律、变元标准化得到的均为等价式, 因此, F 化为前束形之后与谓词公式 F 是等价的。另外, 消去存在量词以后的各个步骤也均使用等价式, 所以语义不会发生变化。不等价只出现在消去存在量词那个步骤。

子句集中的子句之间是合取关系。因此, 子句集中只要有一个子句为不可满足, 则整个子句集就是不可满足的;

空子句是不可满足的。因此, 一个子句集中如果包含有空子句, 则此子句集就一定是不可满足的。

3.4.4 鲁滨逊归结原理

基本思想:

1. 首先把欲证明问题的结论否定, 并加入子句集, 得到一个扩充的子句集 S' 。
2. 然后设法检验子句集 S' 是否含有空子句:

- 若含有空子句，则表明 S' 是不可满足的
- 若不含有空子句，则继续使用归结法，在子句集中选择合适的子句进行归结，直至导出空子句或不能继续归结为止。

2 命题逻辑的归结：

1 归结式的定义及性质：

- 若 P 是原子谓词公式，则称 P 与 $\neg P$ 为互补文字。
- 设 C_1 和 C_2 是子句集中的任意两个子句，如果 C_1 中的文字 L_1 与 C_2 中的文字 L_2 互补，那么可从 C_1 和 C_2 中分别消去 L_1 和 L_2 ，并将 C_1 和 C_2 中余下的部分按析取关系构成一个新的子句 C_{12} ，则称这一过程为归结，称 C_{12} 为 C_1 和 C_2 的归结式，称 C_1 和 C_2 为 C_{12} 的亲本子句。

例：设 $C_1 = \neg P \vee Q, C_2 = \neg Q, C_3 = P$ ，求 C_1, C_2, C_3 的归结式 C_{123} 。

解：

若先对 C_1, C_2 归结，可得到： $C_{12} = \neg P$

再对 C_{12} 和 C_3 归结，得到 $C_{123} = \text{NIL}$

如果改变归结顺序，同样可以得到相同的结果。归结过程可以表示为归结树。

2 定理：

归结式 C_{12} 是其亲本子句 C_1 和 C_2 的逻辑结论。

证明：

设 $C_1 = L \vee C'_1, C_2 = \neg L \vee C'_2$ 关于解释 I 为真，则只需证明 $C_{12} = C'_1 \vee C'_2$ 关于解释 I 也为真。

对于解释 I ， L 和 $\neg L$ 中必有一个为假。

若 L 为假，则必有 C'_1 为真，不然就会使 C_1 为假，这将与前提假设矛盾，因此只能有 C'_1 为真。

同理，若 $\neg L$ 为假，则必有 C'_2 为真。

因此，必有 $C_{12} = C'_1 \vee C'_2$ 关于解释 I 为真。即 C_{12} 是 C_1 和 C_2 的逻辑结论。

3 推论 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，若用 C_{12} 代替 C_1 和 C_2 后得到新的子句集 S_1 ，则由 S_1 的不可满足性可以推出原子句集 S 的不可满足性。

4 推论 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，若把 C_{12} 加入 S 中得到新的子句集 S_2 ，则 S 与 S_2 的不可满足性是等价的。

5 定理：

子句集 S 是不可满足的，当且仅当存在一个从 S 到空子句的归结过程。

2 谓词逻辑的归结：

在谓词逻辑中，由于子句集中的谓词一般都含有变元，不能像命题逻辑那样直接消去互补文字，需要先用变元置换对谓词进行合一，才能进行归结。谓词逻辑的归结要比命题逻辑的归结麻烦。

谓词逻辑中的归结式可用如下定义来描述：

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 $\neg L_2$ 存在一个最一般合一置换 σ ，则称

$$C_{12} = (C_1\sigma - L_1\sigma) \cup (C_2\sigma - L_2\sigma)$$

为 C_1 和 C_2 的二元归结式，而 L_1 和 L_2 为归结式上的文字。

这里使用集合符号和集合的运算，即先将子句 $C_i\sigma$ 和 $L_i\sigma$ 写成集合的形式，在集合表示下做减法和并集运算，然后再写成子句集的形式。

定义中还要求 C_1 和 C_2 无公共变元，这也是合理的。例如

$C_1 = P(x), C_2 = \neg P(f(x))$ ，而 $S = \{C_1, C_2\}$ 是不可满足的。但由于 C_1 和 C_2 的变元相同，就无法合一了。没有归结式，就不能用归结法证明 S 的不可满足性，这就限制了归结法的使用范围。

如果对 C_1 或 C_2 的变元进行换名，便可通过合一，对 C_1 和 C_2 进行归结。如上例，若先对 C_2 进行换名，即 $C_2 = \neg P(f(y))$ ，则可对 C_1 和 C_2 进行归结，得到一个空子句，从而证明了 S 是不可满足的。事实上，在由公式集化为子句集的过程中，其最后一步就是做换名处理。因此，定义中假设 C_1 和 C_2 没有相同变元是可以的。

例 1 设 $C_1 = P(a) \vee R(x), C_2 = \neg P(y) \vee Q(b)$ ，求 C_{12}

解：

取 $L_1 = P(a), L_2 = \neg P(y)$ ，则 L_1 和 L_2 的合一置换是 $\sigma = \{a/y\}$ 。根据定义可得

$$\begin{aligned} C_{12} &= (\{C_1\sigma\} - \{L_1\sigma\}) \cup (\{C_2\sigma\} - \{L_2\sigma\}) \\ &= (\{P(a), R(x)\} - \{P(a)\}) \cup (\{\neg P(a), Q(b)\} - \{\neg P(a)\}) \\ &= (\{R(x)\}) \cup (\{Q(b)\}) = \{R(x), Q(b)\} \\ &= R(x) \vee Q(b) \end{aligned}$$

例 2 设 $C_1 = P(x) \vee Q(a), C_2 = \neg P(b) \vee R(x)$ ，求 C_{12}

解：

由于 C_1 和 C_2 有相同的变元 x ，不符合谓词逻辑归结定义的要求。为了进行归结，需要修改 C_2 中变元 x 的名字为 y ，令 $C_2 = \neg P(b) \vee R(y)$ 。此时 $L_1 = P(x), L_2 = \neg P(b), L_1$ 和 L_2 的合一置换是 $\sigma = \{b/x\}$ 。则有

$$\begin{aligned} C_{12} &= (\{C_1\sigma\} - \{L_1\sigma\}) \cup (\{C_2\sigma\} - \{L_2\sigma\}) \\ &= (\{P(b), Q(a)\} - \{P(b)\}) \cup (\{\neg P(b), R(y)\} - \{\neg P(b)\}) \\ &= (\{Q(a)\}) \cup (\{R(y)\}) = \{Q(a), R(y)\} \\ &= Q(a) \vee R(y) \end{aligned}$$

例 3 设 $C_1 = P(x) \vee \neg Q(b), C_2 = \neg P(a) \vee Q(y) \vee R(z)$

解：

对 C_1 和 C_2 通过合一置换 $\{a/x, b/y\}$ 的作用，可以得到两个互补对。

注意：求归结式不能同时消去两个互补对，其结果不是二元归结式。如在 $\sigma = \{a/x, b/y\}$ ，若同时消去两个互补对所得 $R(z)$ 不是 C_1 和 C_2 的二元归结式。

例 4 设 $C_1 = P(x) \vee P(f(a)) \vee Q(x), C_2 = \neg P(y) \vee R(b)$, 求 C_{12}

解：

对参加归结的某个子句，若其内部有可合一的文字，则在进行归结之前应先进行合一。本例 C_1 中有 $P(x)$ 与 $P(f(a))$ ，若它用们的合一置换 $\sigma = \{f(a)/x\}$ 进行代换，可得到：

$$C_1\sigma = P(f(a)) \vee Q(f(a))$$

此时可对 $C_1\sigma$ 与 C_2 进行归结。选 $L_1 = P(f(a)), L_2 = \neg P(y), L_1$ 和 L_2 的合一是 $\sigma' = \{f(a)/y\}$ ，则可得到 C_1 和 C_2 的二元归结式为：

$$C_{12} = R(b) \vee Q(f(a))$$

其中， $C_1\sigma$ 为 C_1 的因子。

若 C 中有两个或两个以上的文字具有合一置换 σ ，则称 $C\sigma$ 为子句 C 的因子。

若 $C\sigma$ 是一个单文字，则称它为 C 的单元因子。

应用因子概念，可对谓词逻辑中的归结原理给出如下定义：

若 C_1 和 C_2 是无公共变元的子句，则

1. C_1 和 C_2 的二元归结式
2. C_1 和 C_2 的因子 $C_2\sigma_2$ 的二元归结式
3. C_1 的因子 $C_1\sigma_1$ 和 C_2 的二元归结式
4. C_1 的因子 $C_1\sigma_1$ 和 C_2 的因子 $C_2\sigma_2$ 的二元归结式

这四种二元归结式都是子句 C_1 和 C_2 的二元归结式，记为 C_{12}

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

解：对 C_1 ，取合一 $\sigma = \{f(x)/y\}$ ，得 C_1 的因子 $C_1\sigma = P(f(x)) \vee Q(g(x))$

对 C_1 的因子和 C_2 归结 ($\sigma = \{g(a)/x\}$)，可得到 C_1 和 C_2 的二元归结式 $C_{12} = Q(g(g(a))) \vee Q(b)$

对谓词逻辑，即「归结式 C_{12} 是其亲本子句 C_1 和 C_2 的逻辑结论」仍然适用。用归结式取代它在子句集 S 中的亲本子句，所得到的子句集仍然保持着原子句集 S 的不可满足性。此外，对谓词逻辑「子句集 S 是不可满足的，当且仅当存在一个从 S 到空子句的归结过程」也仍然适用，即从不可满足的意义上说，一阶谓词逻辑的归结原理也是完备的

3.4.5 归结演绎推理的方法

2 命题逻辑的归结反演：

归结原理：

假设 F 为已知前提， G 为欲证明的结论，归结原理把证明 G 为 F 的逻辑结论转化为证明 $F \wedge \neg G$ 为不可满足。

再根据定理，在不可满足的意义上，公式集 $F \wedge \neg G$ 与其子句集是等价的，即把公式集上的不可满足转化为子句集上的不可满足。

归结反演：

应用归结原理证明定理的过程称为归结反演。

归结反演过程：

在命题逻辑中，已知 F ，证明 G 为真的归结反演过程如下：

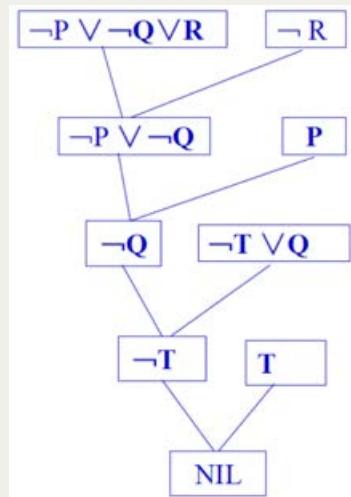
1. 否定目标公式 G ，得 $\neg G$
2. 把 $\neg G$ 并入到公式集 F 中，得到 $\{F, \neg G\}$
3. 把 $\{F, \neg G\}$ 化为子句集 S
4. 应用归结原理对子句集 S 中的子句进行归结，并把每次得到的归结式并入 S 中。如此反复进行，若出现空子句，则停止归结，此时就证明了 G 为真。

例 设已知的公式集为 $P, (P \wedge Q) \rightarrow R, (S \vee T) \rightarrow Q, T$ ，求证结论 R

解：假设结论 R 为假，将 $\neg R$ 加入公式集，并化为子句集

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

归结过程如下：



其含义为：

利用归结原理，对子句集进行归结，并把所得的归结式并入子句集中。重复这一过程，最后归结出了空子句。根据归结原理的完备性，可知子句集 S 是不可满足的，即 $\neg R$ 为假，这就证明了 R 为真。

2 谓词逻辑的归结演绎推理：

谓词逻辑的归结反演过程与命题逻辑的归结反演过程相比，其步骤基本相同，但每步的处理对象不同：

- ／ 在步骤 3 化简子句集时，谓词逻辑需要把由谓词构成的公式集化为子句集
- ／ 在步骤 4 按归结原理进行归结时，谓词逻辑的归结原理需要考虑两个亲本子句的合一

已知：

$$\begin{aligned} F &: (\forall x)((\exists y)(A(x, y) \wedge B(y)) \rightarrow (\exists y)(C(y) \wedge D(x, y))) \\ G &: \neg(\exists x)C(x) \rightarrow (\forall x)(\forall y)(A(x, y) \rightarrow \neg B(y)) \end{aligned}$$

求证 G 是 F 的逻辑结论。

证明：

先把 G 否定，并放入 F 中，得到的 $\{F, \neg G\}$ 为

$$\begin{aligned} &\{(\forall x)((\exists y)(A(x, y) \wedge B(y)) \rightarrow (\exists y)(C(y) \wedge D(x, y))), \\ &\quad \neg(\neg(\exists x)C(x) \rightarrow (\forall x)(\forall y)(A(x, y) \rightarrow \neg B(y)))\} \end{aligned}$$

转化为子句集：

1. 消去连接词 \rightarrow 和 \leftrightarrow ：

$$F : (\forall x)(\neg((\exists y)(A(x, y) \wedge B(y)) \vee (\exists y)(C(y) \wedge D(x, y))))$$

$$G : \neg((\exists x)C(x) \vee ((\forall x)(\forall y)(\neg A(x, y) \vee \neg B(y))))$$

2. 减少否定符号的辖域：

$$F : (\forall x)((\forall y)(\neg A(x, y) \vee \neg B(y)) \vee (\exists y)(C(y) \wedge D(x, y)))$$

$$G : ((\forall x)\neg C(x) \wedge ((\exists x)(\exists y)(A(x, V) \wedge B(y))))$$

3. 对变元标准化：

$$F : (\forall x)((\forall y)(\neg A(x, y) \vee \neg B(y)) \vee (\exists z)(C(z) \wedge D(x, z)))$$

$$G : ((\forall x)\neg C(x) \wedge ((\exists z)(\exists y)(A(z, y) \wedge B(y))))$$

4. 化为前束范式：

$$F : (\forall x)(\forall y)(\exists z)((\neg A(x, y) \vee \neg B(y)) \vee (C(z) \wedge D(x, z)))$$

$$G : (\forall x)(\exists z)(\exists y)(\neg C(x) \wedge A(z, y) \wedge B(y))$$

5. 消去存在量词：

$$F : (\forall x)(\forall y)((\neg A(x, y) \vee \neg B(y)) \vee (C(f(x)) \wedge D(x, f(x))))$$

$$G : (\forall x)(\neg C(x) \wedge A(m, n) \wedge B(n))$$

F 没有化成 $f(x, y)$ 的原因:

因为那部分在原式不在 y 的辖域, 所以可以不用 $f(x, y)$, 相当于 $f(x, y) = 2x$, 与 y 无关

G 化成 m, n 的原因:

因为那部分在原式不在 y 的辖域

6. 化为 Skolem 标准形 (合取范式):

$$F : (\forall x)(\forall y)((\neg A(x, y) \vee \neg B(y) \vee C(f(x))) \wedge (\neg A(x, y) \vee \neg B(y) \vee D(x, f(x))))$$

$$G : (\forall x)(\neg C(x) \wedge A(m, n) \wedge B(n))$$

7. 消去全称量词:

$$F : (\neg A(x, y) \vee \neg B(y) \vee C(f(x))) \wedge (\neg A(x, y) \vee \neg B(y) \vee D(x, f(x)))$$

$$G : \neg C(x) \wedge A(m, n) \wedge B(n)$$

8. 消去合取词:

$$F : \neg A(x, y) \vee \neg B(y) \vee C(f(x)), \neg A(x, y) \vee \neg B(y) \vee D(x, f(x))$$

$$G : \neg C(x), A(m, n), B(n)$$

9. 更换变量名称:

子句集

$$\{\neg A(x, y) \vee \neg B(y) \vee C(f(x)), \neg A(u, v) \vee \neg B(v) \vee D(u, f(u)), \neg C(z), A(m, n), B(n)\}$$

再把 $\{F, \neg G\}$ 化成子句集, 得到:

1. $\neg A(x, y) \vee \neg B(y) \vee C(f(x))$

2. $\neg A(u, v) \vee \neg B(v) \vee D(u, f(u))$

3. $\neg C(z)$

4. $A(m, n)$

5. $B(n)$

其中, 1、2 是由 F 化出的两个子句, 3、4、5 是由 $\neg G$ 化出的 3 个子句。

最后应用谓词逻辑的归结原理对上述子句集进行归结, 其过程为:

6. $\neg A(x, y) \vee \neg B(y)$ 由 1 和 3 归结, 取 $\sigma = \{f(x)/z\}$

7. $\neg B(n)$ 由 4 和 6 归结, 取 $\sigma = \{m/x, n/y\}$

8. NIL 由 5 和 7 归结

因此, G 是 F 的逻辑结论。



「激动人心的生活」问题:

假设：所有不贫穷并且聪明的人都是快乐的，那些看书的人是聪明的。李明能看书且不贫穷，快乐的人过着激动人心的生活。

求证：李明过着激动人心的生活。

先定义谓词：

$\text{Poor}(x)$ x 是贫穷的

$\text{Smart}(x)$ x 是聪明的

$\text{Happy}(x)$ x 是快乐的

$\text{Read}(x)$ x 能看书

$\text{Exciting}(x)$ x 过着激动人心的生活

再将问题用谓词表示如下：

所有不贫穷并且聪明的人都是快乐的

$$(\forall x)((\neg \text{Poor}(x) \wedge \text{Smart}(x)) \rightarrow \text{Happy}(x))$$

那些看书的人是聪明的

$$(\forall y)(\text{Read}(y) \rightarrow \text{Smart}(y))$$

李明能看书且不贫穷

$$\text{Read}(\text{Liming}) \wedge \neg \text{Poor}(\text{Liming})$$

快乐的人过着激动人心的生活

$$(\forall z)(\text{Happy}(z) \rightarrow \text{Exciting}(z))$$

目标：「李明过着激动人心的生活」的否定

$$\neg \text{Exciting}(\text{Liming})$$

将上述谓词公式转化为子句集如下：

• $\text{Poor}(x) \vee \neg \text{Smart}(x) \vee \text{Happy}(x)$

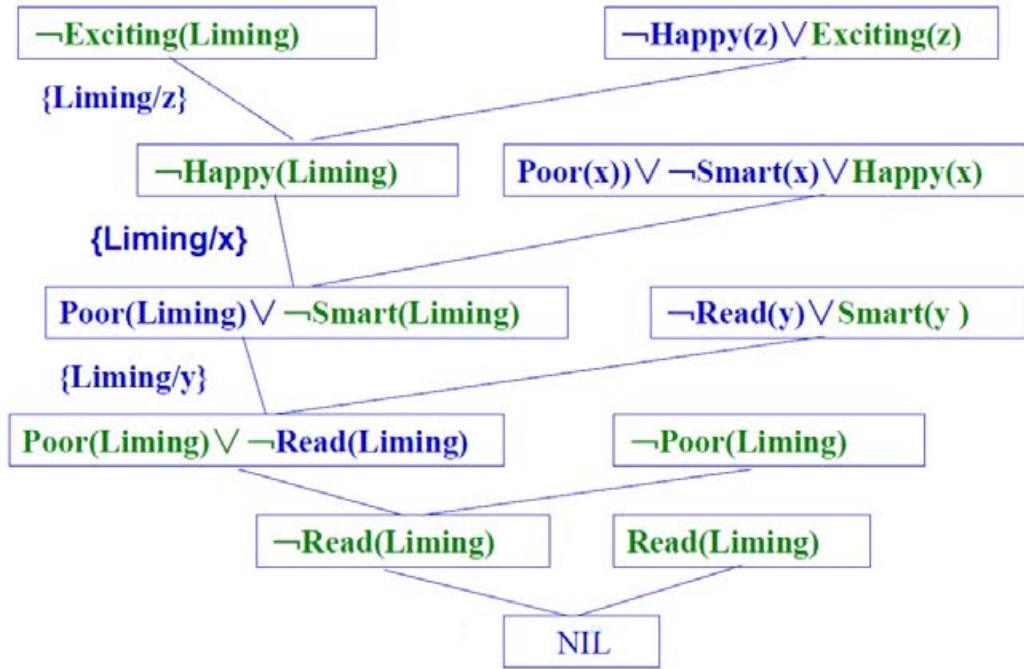
• $\neg \text{Read}(y) \vee \text{Smart}(y)$

• $\text{Read}(\text{Liming})$

• $\neg \text{Poor}(\text{Liming})$

• $\neg \text{Happy}(z) \vee \text{Exciting}(z)$

• $\neg \text{Exciting}(\text{Liming})$ (结论的否定)



3.4.6 归结策略

⟨ 排序策略（广度优先搜索）：

一种穷尽子句比较的复杂搜索方法。设初始子句集为 S_0 ，广度优先策略的归结过程可描述如下：

1. 从 S_0 出发，对 S_0 中的全部子句作所有可能的归结，得到第一层归结式，把这些归结式的集合记为 S_1
2. 用 S_0, S_1 中的子句与 S_1 中的子句进行所有可能的归结，得到第二层归结式，把这些归结式的集合记为 S_2
3. 用 S_0, S_1 和 S_2 中的子句与 S_2 中的子句进行所有可能的归结，得到第三层归结式，把这些归结式的集合记为 S_3

如此继续，直到得出空子句或不能再继续归结为止。

⟨ 删除策略：

基本思想：

归结过程在寻找可归结子句时，子句集中的子句越多，需要付出的代价就会越大。如果在归结时能把子句集中无用的子句删除掉，这就会缩小搜索范围，减少比较次数，从而提高归结效率。

纯文字删除法：

- 如果某文字 L 在子句集中不存在可与其互补的文字 $\neg L$ ，则称该文字为纯文字。
- 在归结过程中，纯文字不可能被消除，用包含纯文字的子句进行归结也不可能得到空子句，因此对包含纯文字的子句进行归结是没有意义的，应该把它从子句集中删除。
- 对子句集而言，删除包含纯文字的子句，是不影响其不可满足性的。

例如，子句集 $S = P \vee Q \vee R, \neg Q \vee R, Q, \neg R$

其中 P 是纯文字，因此可以将子句 $P \vee Q \vee R$ 从子句集 S 中删除。

／＼重言式删除法：

- 如果一个子句中含有互补的文字对，则称其为重言式。如 $P(x) \vee \neg P(x)$, $P(x) \vee Q(x) \vee \neg P(x)$ 都是重言式。
- 重言式是真值为真的子句
- 对一个子句集来说，不管是增加还是删除一个真值为真的子句，都不会影响该子句集的不可满足性。

／＼包孕（含）删除法：

- 设有子句 C_1 和 C_2 ，如果存在一个置换 σ ，使得 $C_1\sigma \subseteq C_2$ ，则称 C_1 包孕于 C_2 。
- 对子句集来说，把其中包孕的子句删去后，不会影响该子句集的不可满足性。因此，可从子句集中删除那些包孕的子句。

例如：

$P(x)$ 包孕于 $P(a), \sigma = a/x$

$P(x)$ 包孕于 $P(a) \vee Q(z), \sigma = a/x$

$P(x) \vee Q(a)$ 包孕于 $P(f(a)) \vee Q(a) \vee R(y), \sigma = f(a)/x$

$P(x) \vee Q(y)$ 包孕于 $P(a) \vee Q(u) \vee R(w), \sigma = a/x, u/y$

2 限制策略：

／＼支持集策略：

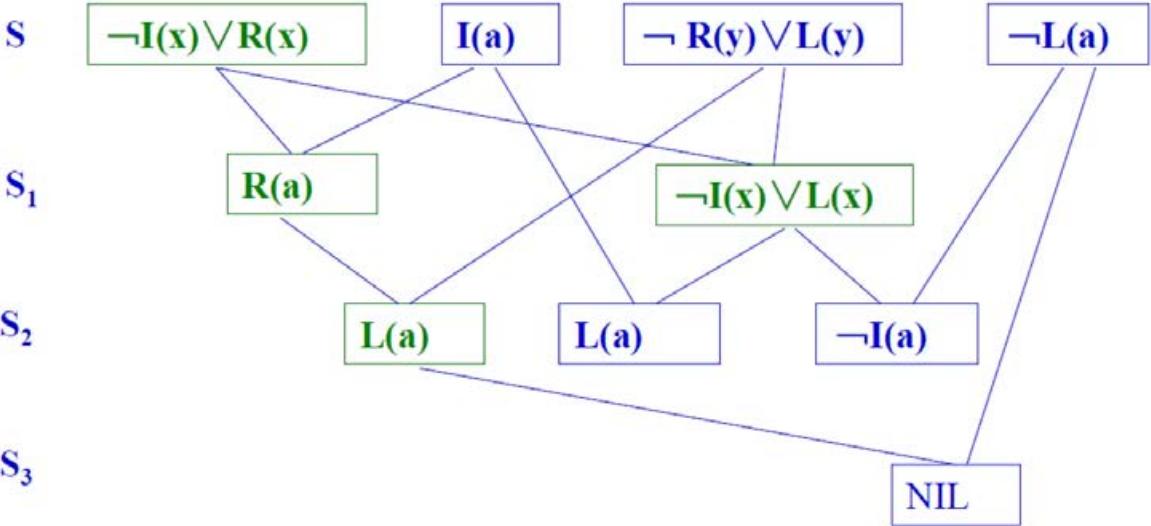
／＼基本思想：

要求每一次参加归结的两个亲本子句中，至少应该有一个是由目标公式的否定所得到的子句或它们的后裔。

- 可以证明支持集策略是完备的，即当子句集为不可满足时，则由支持集策略一定能够归结出一个空子句。
- 也可以把支持集策略看成是在广度优先策略中引入了某种限制条件，这种限制条件代表一种启发信息，因而有较高的效率

例设子句集： $S = \neg I(x) \vee R(x), I(a), \neg R(y) \vee L(y), \neg L(a)$

其中， $\neg I(x) \vee R(x)$ 为目标公式的否定。用支持集策略证明 S 为不可满足：



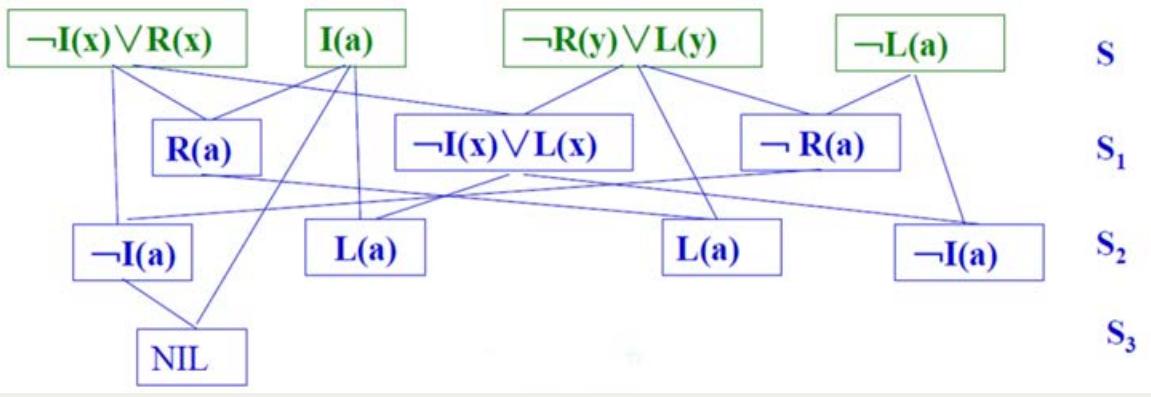
各级归结式数目要比广度优先策略生成的少，但在第二级还没有空子句。即这种策略限制了子句集元素的剧增，但却增加了空子句的深度。

线形输入策略（不完备的）：

- ☛ 基本思想：要求每次参加归结的两个亲本子句中，至少应该有一个是初始子句集中的子句。
- ☛ 所谓初始子句集是指开始归结时所使用的子句集。
- ☛ 该策略可限制生成归结式的数目，简单高效。

用线性输入策略证明子句集为不可满足

$$S = \neg I(x) \vee R(x), I(a), \neg R(y) \vee L(y), \neg L(a)$$



祖先过滤策略（完备的）：

- ☛ 这种策略与线性输入策略有点相似，但放宽了对子句的限制。
- ☛ 基本思想：
 - 每次参加归结的两个亲本子句，只要满足以下两个条件中的任意一个就可进行归结：
 - 两个亲本子句中至少有一个是初始子句集中的子句。
 - 如果两个亲本子句都不是初始子句集中的子句，则一个子句应该是另一个子句的先辈子句。

一个子句（例如 C_1 ）是另一个子句（例如 C_2 ）的先辈子句是指：

C_2 是由 C_1 与别的子句归结后得到的归结式。

单文字子句策略（不完备的）：

如果一个子句只包含一个文字，则称此子句为单文字子句。

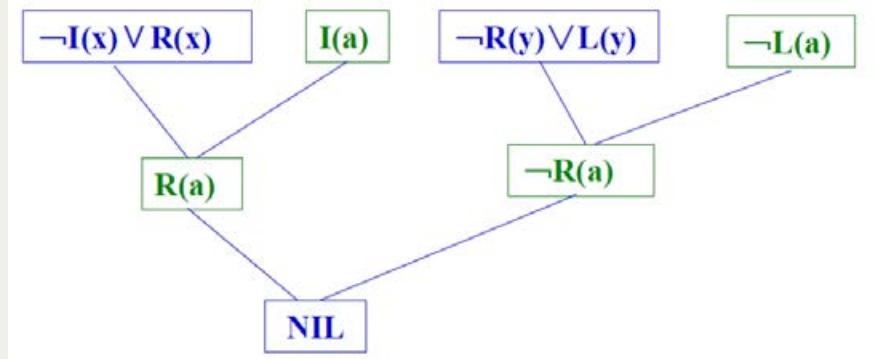
基本思想：

要求每次参加归结的两个亲本子句中至少有一个子句是单文字子句。

是对支持集策略的进一步改进，归结式包含的文字数将少于其亲本子句中的文字数，这将有利于向空子句的方向发展，因此会有较高的归结效率。

设有子句集： $S = \neg I(x) \vee R(x), I(a), \neg R(y) \vee L(y), \neg L(a)$

用单文字子句策略证明不可满足：



3.4.7 归结反演求取问题的答案

归结原理除了可用于定理证明外，还可用来求取问题答案，其思想与定理证明相似。一般步骤为：

1. 把已知条件用谓词公式表示，并化成相应的子句集 S_1
2. 把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 G_1
3. 把 G_1 化为子句集 S_2 ，并把子句集 S_2 与 S_1 合并构成新子句集 S
4. 对子句集 S 应用谓词归结原理进行归结，在归结过程中通过合一置换，改变 ANSWER 中的变元
5. 如果得到归结式 ANSWER，则问题的答案就在 ANSWER 谓词中

已知：

1. 如果 x 和 y 是同班同学，则 x 的老师也是 y 的老师
2. 王先生是小李的老师
3. 小李和小张是同班同学

问：小张的老师是谁？

解：

$T(x, y)$ 表示 x 是 y 的老师, $C(x, y)$ 表示 x 与 y 是同班同学, 则

1. 已知可表示成如下的谓词公式:

$$F1 : (\forall x)(\forall y)(\forall z)(C(x, y) \wedge T(z, x) \rightarrow T(z, y))$$

$$F2 : T(Wang, Li)$$

$$F3 : C(Li, Zhang)$$

将它们化成子句集为:

$$S1 \{ \neg C(x, y) \vee \neg T(z, x) \vee T(z, y), T(Wang, Li), C(Li, Zhang) \}$$

2. 把问题用谓词公式表示, 并将其否定与谓词 ANSWER 作析取:

设小张的老师是 u , 则有 $T(u, Zhang)$

$$G1 : \neg T(u, Zhang) \vee \text{ANSWER}(u)$$

3. 将析取式 $G1$ 化成子句集 $S2$, 并将 $S1$ 与 $S2$ 合并为新子句集 S :

$$S2 = \neg T(u, Zhang) \vee \text{ANSWER}(u)$$

$$S = S1 \cup S2 = \{(a), (b), (c), (d)\}$$

- (a) $\neg C(x, y) \vee \neg T(z, x) \vee T(z, y)$
- (b) $T(Wang, Li)$
- (c) $C(Li, Zhang)$
- (d) $\neg T(u, Zhang) \vee \text{ANSWER}(u)$

4. 应用归结原理进行归结:

- (e) $\neg C(Li, y) \vee T(Wang, y)$ [a 与 b 归结, $(Li/x, Wang/z)$]
- (f) $\neg C(Li, Zhang) \vee \text{ANSWER}(u)$ [d 与 e 归结, $(Wang/u, Zhang/y)$]
- (g) $\text{ANSWER}(Wang)$ [c 与 f 归结]

5. 得到归结式 $\text{ANSWER}(Wang)$, 答案即在其中:

$u = Wang$, 即小张的老师是王先生。

4 搜索策略

4.1 概述

概念:

依靠经验, 利用已有知识, 根据问题的实际情况, 不断寻找可利用知识, 从而构造一条代价最小的推理路线, 使问题得以解决的过程称为搜索。

适用情况:

不良结构或非结构化问题; 难以获得求解所需的全部信息; 更没有现成的算法可供求解使用。结构良好, 理论上有算法可以的, 但问题或算法的复杂性较高。

搜索的分类:

按是否使用启发式信息:

盲目搜索：

按预定的控制策略进行搜索，在搜索过程中获得的中间信息并不改变控制策略

启发式搜索：

在搜索中加入了与问题有关的启发性信息，用于指导搜索朝着最有希望的方向前进，加速问题的求解过程并找到最优解

按问题的表示方式：

状态空间搜索：

用状态空间法来求解问题所进行的搜索

与或树搜索：

用问题归约法来求解问题时所进行的搜索

问题可以形式化地定义为三个组成部分：

初始状态 (Initial state) s_0

后继函数：

操作函数 (Actions) : $\{a_1, a_2, a_3, \dots\}$

路径耗散函数 (PathCost) : PathCost $(s_0 \xrightarrow{a_1} s_1 \dots \xrightarrow{a_{i-1}} s_i)$

目标测试函数 (GoalTest) : GoalTest(s) $\rightarrow T/F$

问题的解：初始状态到目标状态的操作序列， $(s_0 \xrightarrow{a_1} s_1 \dots \xrightarrow{a_{n-1}} s_n)$

4.1.1 状态空间法：

状态 (State)：

是表示问题求解过程中每一步问题状况的数据结构，它可形式地表示为：

$S_k = \{S_{k0}, S_{k1}, \dots\}$ ，当对每一个分量都给以确定的值时，就得到了一个具体的状态。

操作 (Operator)：

也称为算符，它是把问题从一种状态变换为另一种状态的手段。操作可以是一个机械步骤，一个运算，一条规则或一个过程。操作可理解为状态集合上的一个函数，它描述了状态之间的关系。

状态空间 (State space)：

用来描述一个问题的全部状态以及这些状态之间的相互关系。常用一个三元组表示为： (S, F, G) ，其中， S 为问题的所有初始状态集合， F 为操作的集合， G 为目标状态的集合。状态空间也可用一个赋值的有向图来表示，该有向图称为状态空间图。在状态空间图中，节点表示问题的状态，有向边表示操作。

状态空间法求解问题的基本过程：

1. 为问题选择适当的「状态」及「操作」的形式化描述方法
2. 从某个初始状态出发，每次使用一个「操作」，递增地建立起操作序列，直到达到目标状态为止
3. 由初始状态到目标状态所使用的算符序列就是该问题的一个解。

问题:

最优解问题

搜索策略问题

二阶梵塔问题:

设有三根钢针, 它们的编号分别是 1 号、2 号和 3 号。在初始情况下, 1 号钢针上穿有 A、B 两个金片, A 比 B 小, A 位于 B 的上面。要求把这两个金片全部移到另一根钢针上, 而且规定每次只能移动一个金片, 任何时刻都不能使大的位于小的上面。

设用 $S_k = (S_{k0}, S_{k1})$ 表示问题的状态, 其中, S_{k0} 表示金片 A 所在的钢针号, S_{k1} 表示金片 B 所在的钢针号。

全部可能的问题状态共有以下 9 种:

$$\begin{aligned} S_0 &= (1, 1) \\ S_1 &= (1, 2) \\ S_2 &= (1, 3) \\ S_3 &= (2, 1) \\ S_4 &= (2, 2) \\ S_5 &= (2, 3) \\ S_6 &= (3, 1) \\ S_7 &= (3, 2) \\ S_8 &= (3, 3) \end{aligned}$$

初始状态集合 $S = \{S_0\}$

目标状态集合 $G = \{S_4, S_8\}$

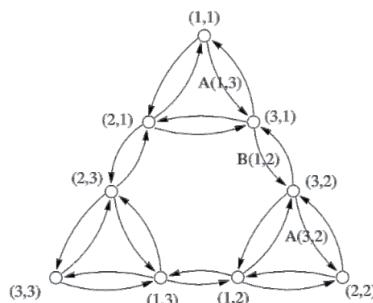
操作:

A_{ij} 表示把金片 A 从第 i 号钢针移到 j 号钢针上

B_{ij} 表示把金片 B 从第 i 号钢针移到到第 j 号钢针上。共有 12 种操作, 它们分别是:

$$A_{12} A_{13} A_{21} A_{23} A_{31} A_{32} B_{12} B_{13} B_{21} B_{23} B_{31} B_{32}$$

根据上述 9 种可能的状态和 12 种操作, 可构成二阶梵塔问题的状态空间图:



从初始节点 (1,1) 到目标节点 (2,2) 及 (3,3) 的任何一条路径都是问题的一个解。其中, 最短的路径长度是 3, 它由 3 个操作组成。例如, 从 (1,1) 开始, 通过使用操作 A_{13} 、 B_{12} 及 A_{32} , 可到达 (2,2)

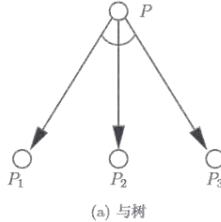
4.1.2 问题规约法

问题的与/或树表示基本思想:

当一问题较复杂时, 可通过分解或变换, 将其转化为一系列较简单的子问题, 然后通过对这些子问题的求解来实现对原问题的求解。

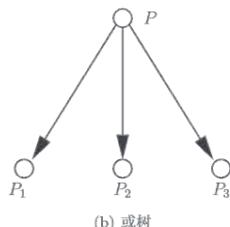
分解:

- 如果一个问题 P 可以归约为一组子问题 P_1, P_2, \dots, P_n , 并且只有当所有子问题 P_i 都有解时原问题 P 才有解
- 任何一个子问题 P_i 无解都会导致原问题 P 无解, 则称此种归约为问题的分解。
- 分解所得到的子问题的「与」与原问题 P 等价。
- 可以用一个「与树」来表示这种分解:



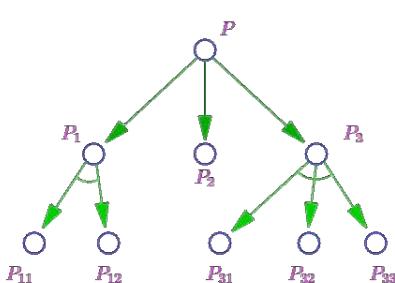
等价变换:

- 如果一个问题 P 可以归约为一组子问题 P_1, P_2, \dots, P_n , 并且子问题 P_i 中只要有一个有解则原问题 P 就有解
- 只有当所有子问题 P_i 都无解时原问题 P 才无解, 称此种归约为问题的等价变换, 简称变换。
- 变换所得到的子问题的「或」与原问题 P 等价。
- 可用一个「或树」来表示这种变换:



与/或树:

如果一个问题既需要通过分解, 又需要通过变换才能得到其本原问题, 则其求解过程可用一个「与/或树」表示:



本原问题: 指那种不能 (或不需要) 再进行分解或变换, 且可以直接解答的子问题。

端节点与终止节点:

端节点:

没有子节点的节点。

终止节点:

本原问题所对应的节点。终止节点一定是端节点，但端节点却不一定时终止节点。

2 可解节点与不可解节点：

在与/或树中，满足以下三个条件之一的节点为可解节点：

- 任何终止节点都是可解节点。
- 对「或」节点，当其子节点中至少有一个为可解节点时，则该或节点就是可解节点。
- 对「与」节点，只有当其子节点全部为可解节点时，该与节点才是可解节点。

可用类似的方法定义不可解节点：

- 不为终止节点的端节点是不可解节点。
- 对「或」节点，若其全部子节点都为不可解节点，则该或节点是不可解节点。
- 对「与」节点，只要其子节点中有一个为不可解节点，则该与节点是不可解节点。

3 解树：

一个由可解节点构成，并且由这些可解节点可以推出初始节点（对应原始问题）为可解节点的子树。

问题归约求解过程实际上就是生成解树，即证明原始节点是可解节点的过程。

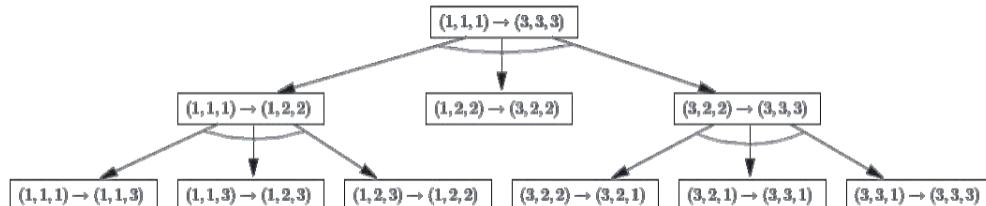
4 三阶梵塔问题：

其中 (i, j, k) 表示 C 在柱 i , B 在柱 j , A 在柱 k

问题可分解为以下三个子问题：

1. 把金片 A 及 B 移到 2 号钢针上的双金片移动问题。即 $(1, 1, 1) \rightarrow (1, 2, 2)$
2. 把金片 C 移到 3 号钢针上的单金片移动问题。即 $(1, 2, 2) \rightarrow (3, 2, 2)$
3. 把金片 A 及 B 移到 3 号钢针上的双金片移动问题。即 $(3, 2, 2) \rightarrow (3, 3, 3)$

子问题 1 和 3 都是二阶梵塔问题，还可以再分解，2 则是本原问题。



在该与/或树中，有 7 个终止节点，它们分别对应着 7 个本原问题。如果把这些本原问题从左至右排列起来，即得到了原始问题的解：

$(1, 1, 1) \rightarrow (1, 1, 3), (1, 1, 3) \rightarrow (1, 2, 3), (1, 2, 3) \rightarrow (1, 2, 2), (1, 2, 2) \rightarrow (3, 2, 2),$
 $(3, 2, 2) \rightarrow (3, 2, 1), (3, 2, 1) \rightarrow (3, 3, 1), (3, 3, 1) \rightarrow (3, 3, 3)$

4.2 状态空间盲目搜索

1 吃豆人为例：

状态空间：

Problem: Path Finding

States: (x, y) location

Actions: NSEW

Successor 后继: update location only

Goal test: is $(x, y) = \text{END}$

💡 Problem: Eat-All-Dots

- 💡 States: $\{(x, y), \text{dot booleans}\}$
- 💡 Actions: NSEW
- 💡 Successor: update location and possibly a dot boolean
- 💡 Goal test: dots all false

💡 状态空间大小:

💡 World state:

- 💡 Agent positions: 120
- 💡 Food count: 30

💡 Ghost positions: 12

💡 Agent facing 面向: NSEW

💡 How many

💡 World states $120 \times (2^{30}) \times (12^2) \times 4$

💡 States for path finding 120

💡 States for eat-all-dots $120 \times (2^{30})$

💡 状态空间搜索的基本思想:

💡 先把问题的初始状态作为当前扩展节点对其进行扩展，生成一组子节点

💡 然后检查问题的目标状态是否出现在这些子节点中：

💡 若出现，则搜索成功，找到了问题的解

💡 若没出现，则再按照某种搜索策略从已生成的子节点中选择一个节点作为当前扩展节点

💡 重复上述过程，直到目标状态出现在子节点中或者没有可供操作的节点为止。

对一个节点进行扩展指对该节点用某个可用操作进行作用，生成该节点的一组子节点。

💡 算法的数据结构和符号约定

💡 Open 表：用于存放刚生成的节点，未扩展的节点，Open 表称为未扩展的节点表。

💡 Closed 表：用于存放已经扩展或将要扩展的节点，Closed 表称为已扩展的节点表。

💡 S_0 ：用表示问题的初始状态

💡 S_g ：用表示问题的目标状态

4.2.1 一般图搜索过程：

1. 把初始节点 S_0 放入 Open 表，并建立目前仅包 S_0 的图 G ，建立一个 Closed 表，置为空；
2. 检查 Open 表是否为空表，若为空，则问题无解，失败退出
3. 把 Open 表的第一个节点取出放入 Closed 表，并记该节点为 n
4. 考察节点 n 是否为目标节点，若是则得到问题的解成功退出。
5. 扩展节点 n ，生成一组节点。把这些节点中不是 n 父节点的那部分子节点计入一个新集合 M ，并把这些子节点作为节点 n 的子节点加 G 中。

6. 针对 M 中子节点的不同情况，分别作如下处理：

- (a) 对那些没有在 G 中出现过的 M 成员设置一个指向其父节点（即节点 n ）的指针，并将它放入 Open 表。
- (b) 对那些原来已经在 G 中出现过，但没有被扩展过的 M 成员，确定是否需要修改它指向父节点的指针

一般是由原始节点到该节点路径上所付出的代价来决定的，哪条路径付出的代价小，相应的节点就作为它的父节点。所谓由原始节点到该节点路径的代价是指这条路径上所有有向边代价之和。

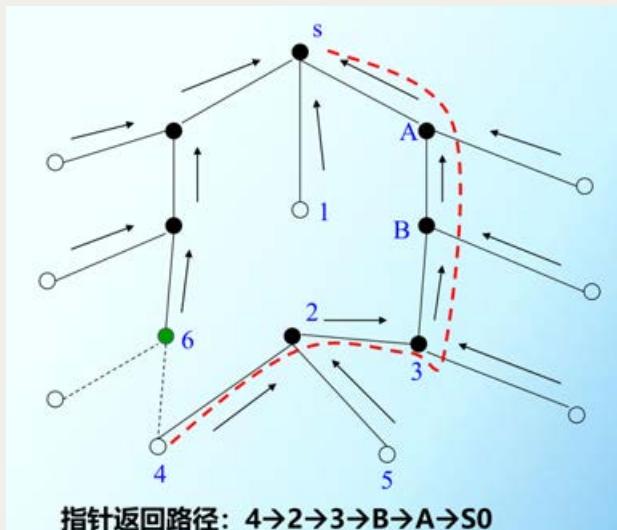
- (c) 对那些原来已经在 G 中出现过，并已经被扩展过的 M 成员，确定是否需要修改其后继节点指向父节点的指针。

7. 按某种策略对 Open 表中的节点进行排序

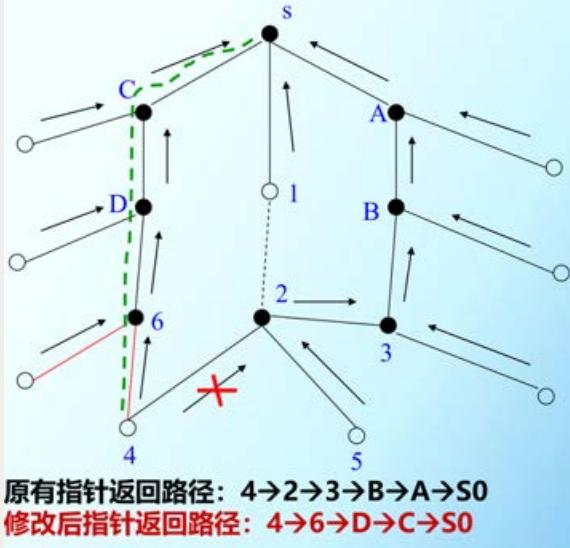
8. 转第 2 步

上述过程是状态空间的一般图搜索算法，它具有通用性，后面所要讨论的各种状态空间搜索策略都是上述过程的特例。各种搜索策略的主要区别在于对 Open 表中节点的排序顺序的不同。

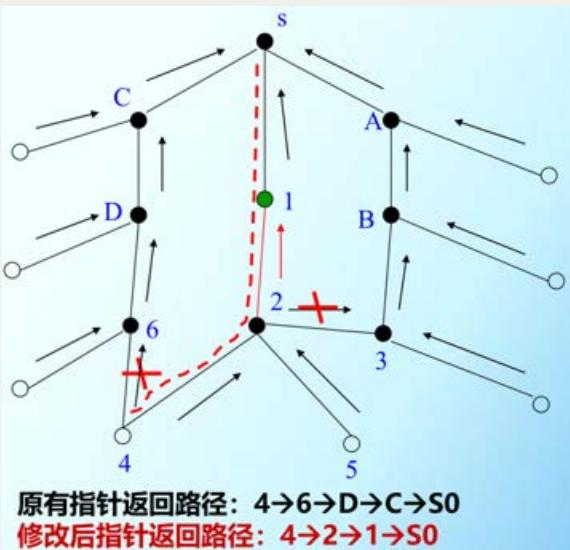
修改返回指针例子：



- (b) 对那些原来已经在 G 中出现过，但没有被扩展过的 M 成员，确定是否需要修改它指向父节点的指针。



(c) 对那些原来已经在 G 中出现过，并已经被扩展过的 M 成员，确定是否需要修改其后继节点指向父节点的指针。



4.2.2 广度优先搜索

❑ Breadth-first search, 一种先生成的节点先扩展的策略。

❑ 宽度优先搜索的基本思想是：

- 从初始节点 S_0 开始，逐层地对节点进行扩展并考察它是否为目标节点
- 在第 n 层的节点没有全部扩展并考察之前，不对第 $n+1$ 层的节点进行扩展。
- Open 表为队列，先进入的节点排在前面，后进入的排在后面。

❑ 过程

1. 把初始节点 S_0 放入 Open 表，建立一个 Closed 表，置为空
2. 检查 Open 表是否为空表，若为空，则问题无解，失败退出
3. 把 Open 表的第一个节点取出放入 Closed 表，并记该节点为 n
4. 考察节点 n 是否为目标节点，若是则得到问题的解成功退出
5. 若节点 n 不可扩展，则转第 2 步

以上和一般的情况相同

6. 扩展节点 n ，将其子节点放入 Open 表的尾部，并为每个子节点设置指向父节点的指针，转向第 2 步。

性质：

- 当问题有解时，一定能找到解
- 搜索效率较低
- 方法与问题无关，具有通用性
- 搜索得到的解是搜索树中路径最短的解（最优解）
- 属于完备搜索策略

4.2.3 深度优先搜索

Depth-first search首先扩展最新产生的（即最深的）节点，是后生成的节点先扩展的策略

基本思想

- 从初始节点 s_0 开始扩展，若没有得到目标节点，则选择最新产生的子节点进行扩展
- 若还是不能到达目标节点，则再对刚才最新产生的子节点进行扩展，一直如此向下搜索。
- 当到达某个子节点，且该子节点既不是目标节点又不能继续扩展时，才选择其兄弟节点进行考察
- Open 表是一种栈结构，最先进入的节点排在最后面，最后进入的节点排最前面。

过程：

- 把初始节点 S_0 放入 Open 表，建立一个 Closed 表，置为空
- 检查 Open 表是否为空表，若为空，则问题无解，失败退出
- 把 Open 表的第一个节点取出放入 Closed 表，并记该节点为 n
- 考察节点 n 是否为目标节点，若是则得到问题的解成功退出
- 若节点 n 不可扩展，则转第 2 步
- 扩展节点 n ，将其子节点放入 Open 表的首部，并为每个子节点设置指向父节点的指针，转向第 2 步

性质：

- 一般不能保证找到最优解
- 最坏情况时，搜索空间等同于穷举
- 通用的与问题无关的方法
- 为了解决深度优先搜索不完备问题，避免搜索过程陷入无穷分支的死循环，提出了有界深度优先搜索方法。

迭代加深的深度优先搜索 (Iterative Deepening)：

思想：

get DFS's space advantage with BFS's time / shallow-solution advantages

Run a DFS with depth limit 1. If no solution

- ∅ Run a DFS with depth limit 2. If no solution
- ∅ Run a DFS with depth limit 3.
- ∅ Generally most work happens in the lowest level searched, so not so bad!

4.2.4 代价一致搜索

前面的各种搜索策略中，实际都假设状态空间中各边的代价都相同，且都一个单位量。从而，可用路径长度来代替路径的代价。但实际问题中，这种假设不现实，它们的状态空间中的各个边的代价不可能完全相同。

为此，我们需要在搜索树中给每条边标上其代价。这种边上有代价的树称为代价树：

- ∅ 在代价树中，可以用 $g(n)$ 表示从初始节点 S_0 到节点 n 的代价，用 $c(n_1, n_2)$ 表示从父节点 n_1 到 n_2 的代价。这样，对节点 n_2 的代价有

$$g(n_2) = g(n_1) + c(n_1, n_2)$$

∅ 在代价树中，最小代价的路径和最短路径是有可能不同的，最短路径不一定是最小代价路径，最小代价路径也不一定是路径最短。代价树搜索的目的是为了找到最佳解，即找到一条代价最小的解路径。

∅ 搜索树中每条连接线上的有关代价，表示时间、距离等花费。

- ∅ Uniform-cost/Cheapest-first search 是宽度优先搜索的一种推广，不是沿着等长度路径断层进行扩展，而是沿着等代价路径断层进行扩展。

∅ 代价一致搜索的基本思想是：

- ／* 在代价一致搜索算法中，把从起始节点 S_0 到任一节点 i 的路径代价记为 $g(i)$ 。
- ／* 从初始节点 S_0 开始扩展，若没有得到目标节点，则优先扩展最少代价 $g(i)$ 的节点，一直如此向下搜索。

∅ 过程：

- ／* 把初始节点 S_0 放入 Open 表，设 $g(s_0) = 0$ ，建立一个 Closed 表，置为空
- ／* 检查 Open 表是否为空表，若为空，则问题无解，失败退出
- ／* 把 Open 表的第一个节点取出放入 Closed 表，并记该节点为 n
- ／* 考察节点 n 是否为目标节点，若是则得到问题的解成功退出
- ／* 若节点 n 不可扩展，则转第 2 步
- ／* 扩展节点 n ，生成子节点 $n_i (i = 1, 2, \dots)$ ，将其子节点放入 Open 表，并为每个子节点设置指向父节点的指针，计算各个节点的代价 $g(n_i)$ ，将 Open 表内的节点按 $g(n_i)$ 从小到大排序，转向第 2 步

4.3 状态空间启发式搜索

∅ 启发性信息：

- 启发性信息是指那种与具体问题求解过程有关的，并可指导搜索过程朝着最有希望方向前进的控制信息。
 - 启发信息的启发能力越强，扩展的无用结点越少，搜索算法越高效。
 - 启发信息包括以下 3 种：
 - 有效地帮助确定扩展节点的信息；
 - 有效地帮助决定哪些后继节点应被生成的信息；
 - 能决定在扩展一个节点时哪些节点应从搜索树上删除的信息。
- 估价函数： $f(n) = g(n) + h(n)$
- $g(n)$ 是从初始节点到节点 n 的实际代价，
 - $h(n)$ 是从节点 n 到目标节点的最优路径的估计代价，需要根据问题自身特性来确定，体现的是问题自身的启发性信息，因此也称为启发函数。
 - 如果使用贪心算法，每次都只选择子节点中 $h(n)$ 最小的进行扩展

4.3.1 A 算法

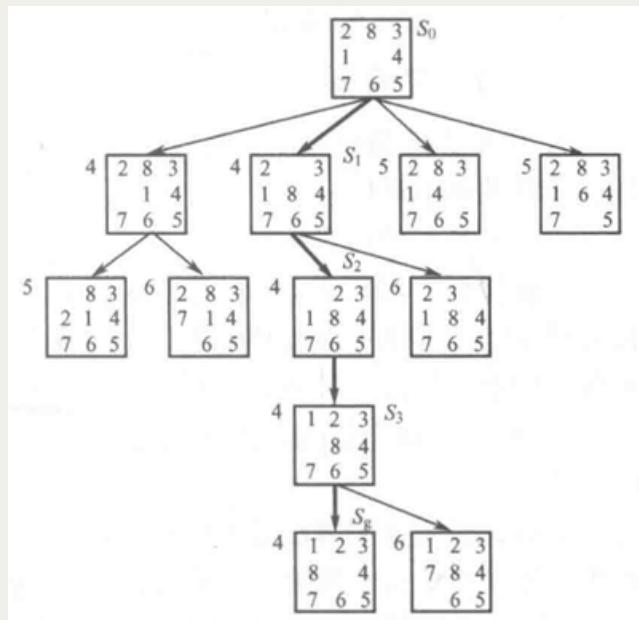
- 在状态空间搜索中，如果每一步都利用估价函数 $f(n) = g(n) + h(n)$ 对 Open 表中的节点进行排序，则称 A 算法，也被称为启发式搜索算法。
- 根据搜索过程中选择扩展节点的范围，启发式搜索算法可分为：
- 全局择优搜索算法（主要是这个）：

每当需要扩展节点时，总是从 Open 表的所有节点中选择一个估价函数值最小的节点进行扩展
 - 局部择优搜索算法：

每当需要扩展节点时，总是从刚生成的子节点中选择一个估价函数值最小的节点进行扩展
- 全局择优搜索算法的搜索过程：
1. 把初始节点 S_0 放入 Open 表中， $f(S_0) = g(S_0) + h(S_0)$
 2. 如果 Open 表为空，则问题无解，失败退出。
 3. 把 Open 表的第一个节点取出放入 Closed 表，并记该节点为 n 。
 4. 考察节点 n 是否为目标节点。若是，则找到了问题的解，成功退出。
 5. 若节点 n 不可扩展，则转第 2 步。
 6. 扩展节点 n ，生成其子节点 $n_i (i = 1, 2, \dots)$ ，计算每个子节点的估价值 $f(n_i) (i = 1, 2, \dots)$ ，并为每个子节点设置指向父节点的指针，然后将它们放入 Open 表中。
 7. 根据各节点的估价函数值，对 Open 表中的全部节点按从小到大的顺序排序。
- 这样在算法第 3 步取出的节点一定是 Open 表的所有节点中估价函数值最小的
8. 转第 2 步

- 如果取估价函数 $f(n) = g(n)$, 则它将退化为代价树的广度优先搜索; 如果取估价函数 $f(n) = d(n)$, 则它将退化为广度优先搜索。它们是全局择优搜索的两个特例。

八数码难题:



设问题的初始状态 S_0 和目标状态 S_g 如图所示, 且估价函数为 $f(n) = d(n) + W(n)$, 其中: $d(n)$ 表示节点 n 在搜索树中的深度, $W(n)$ 表示节点 n 中「不在位」的数码个数。

- 算法可能得到的不是最优解, 因为在估价函数大于实际需要付出的代价时, 可能会选择错误的扩展方向。原因在于对估价函数没有限制。

4.3.2 A* 算法

- 假设 $f^*(n)$ 为从初始节点 S_0 出发, 约束经过节点 n 到达目标节点 S_g 的最小代价值。估价函数 $f(n)$ 则是 $f^*(n)$ 的估计值。显然, $f^*(n)$ 应由以下两部分所组成:

从初始节点 S_0 到节点 n 的最小代价, 记为 $g^*(n)$

从节点 n 到目标节点 S_g 的最小代价, 记为 $h^*(n)$, 当问题有多个目标节点时, 取其中代价最小的一个

即有: $f^*(n) = g^*(n) + h^*(n)$, 且在最佳路径上的所有节点的 f^* 值都相等。

- 对 A 算法中的 $g(n)$ 和 $h(n)$ 分别提出如下限制:

$g(n)$ 是对 $g^*(n)$ 的估计, 且 $g(n) > 0$

$h(n)$ 是 $h^*(n)$ 的下界, 即对任意节点 n 均有 $h(n) \leq h^*(n)$ 。称得到的算法为 A^* 算法。

- A^* 算法的可纳性:

对任一状态空间图, 当从初始节点到目标节点有路径存在时, 如果搜索算法总能在有限步骤内找到一条从初始节点到目标节点的最佳路径, 并在此路径上结束, 则称该搜索算法是可采纳的。

A* 算法可纳性的证明过程可分以下三步：

1. 对有限图，如果从初始节点 S_0 到目标节点 S_g 有路径存在，A* 算法一定能够成功结束。
2. 对无限图，如果从初始节点 S_0 到目标节点 S_g 有路径存在，A* 算法也一定能够成功结束。
 - (a) 对无限图，如果从初始节点 S_0 到目标节点 S_g 有路径存在，则算法 A* 算法不终止的话，则从 Open 表中选出的节点必将具有任意大的 f 值。
 - (b) 在 A* 算法终止前的任何时刻，Open 表中总存在节点 n' ，它是从初始节点 S_0 到目标节点的最佳路径上的一个节点，且满足 $f(n') \leq f^*(S_0)$ 。

(利用 a,b 反证法 证 第二步结论) 假设 A* 算法不结束，由 a 知，Open 表中的节点有任意大的 f 值，这与 b 的结论相矛盾，因此 A* 算法只能成功结束。

3. A* 算法一定能够结束在最佳路径上。

- (a) Open 表中任一具有 $f(n) < f^*(S_0)$ 的节点 n ，最终都被 A* 算法选作为扩展的节点。
- (b) A* 算法只能终止在最佳路径上（反证法）。

假设 A* 算法未能终止在最佳路径上，而是终止在某个目标节点 t 处（不是最优的），则

$$f(t) = g(t) > f^*(S_0)$$

但由 2(b) 可知，在 A* 算法结束前必有最佳路径上的一个节点 n 在 Open 表中，且

$$f(n') \leq f^*(S_0) < f(t)$$

这时算法一定会选择 n' 来扩展，而不可能选择 t ，从而也不会去扩展目标节点 t ，这就与假设 A* 算法终止在目标节点 t 相矛盾。因此，A* 算法只能终止在最佳路径上。

- (c) 在 A* 算法中，对任何被扩展的节点 n ，都有 $f(n) \leq f^*(S_0)$

2 A* 算法的最优化：

★ A* 算法的搜索效率很大程度上取决于估价函数 $h(n)$

★ 在满足 $h(n) \leq h^*(n)$ 的前提下， $h(n)$ 的值越大越好。 $h(n)$ 的值越大，说明它携带的启发性信息越多，A* 算法搜索时扩展的节点就越少，搜索效率就越高。A* 算法的这一特性也称为信息性。

★ 设有两个 A* 算法 A_1^* 和 A_2^* ：

$$\begin{aligned} A_1^* : f_1(n) &= g_1(n) + h_1(n) \\ A_2^* : f_2(n) &= g_2(n) + h_2(n) \end{aligned}$$

如果 A_2^* 比 A_1^* 有更多的启发性信息，即对所有非目标节点均有 $h_2(n) > h_1(n)$ ，则在搜索过程中， A_2^* 扩展的节点集是 A_1^* 扩展的节点集的子集。

2 A* 算法的单调性：

如果启发函数满足以下两个条件：

• $h(S_g) = 0$

• 对任意节点 n_i 及其任一子节点 n_j ，都有

$$0 \leq h(n_i) - h(n_j) \leq c(n_i, n_j)$$

式中， $c(n_i, n_j)$ 是节点 n_i 到其子节点 n_j 的边代价，则称 $h(n)$ 满足单调限制。

如果 h 满足单调条件，则当 A* 算法扩展节点 n 时，该节点已经找到了通往它的最佳路径，即 $g(n) = g^*(n)$ ，就不需要重复修改子节点指针了。

如果 $h(n)$ 满足单调限制，则 A* 算法扩展的节点序列的 f 值是非递减的，即 $f(n_i) \leq f(n_{i+1})$ 。

$h(n)$ 满足单调性限制下的 A* 算法常被称为改进的 A* 算法

4.4 与/或树的启发式搜索

2 解树的代价：

要寻找最优解树，首先需要计算解树的代价。在与/或树的启发式搜索过程中，解树的代价可按如下方法计算：

1. 若 n 为终止节点，则其代价 $h(n) = 0$

2. 若 n 为或节点，且子节点为 n_1, n_2, \dots, n_k ，则 n 的代价为

$$h(n) = \min_{1 \leq i \leq k} \{c(n, n_i) + h(n_i)\}, \quad c(n, n_i) \text{ 是节点 } n \text{ 到其子节点 } n_i \text{ 的边代价。}$$

3. 若 n 为与节点，且子节点为 n_1, n_2, \dots, n_k ，则 n 的代价可用和代价法或最大代价法计算。

(a) 和代价法：

$$h(n) = \sum_{i=1}^k [c(n, n_i) + h(n_i)]$$

(b) 最大代价法：

$$h(n) = \max_{1 \leq i \leq k} \{c(n, n_i) + h(n_i)\}$$

4. 若 n 是端节点，但不是终止节点，则 n 不可扩展，其代价定义为 $h(n) = \infty$ 。

5. 根节点的代价即为解树的代价。

2 希望树：

- ／＼ 为了找到最优解树，搜索过程的任何时刻都应该选择最有希望成为最优解树一部分的节点进行扩展
- ／＼ 由于这些节点及其父节点所构成的与/或树最有可能成为最优解树的一部分，因此称它为希望解树，简称希望树。希望解树是会随搜索过程而不断变化的。
- ／＼ 希望解树 T 满足：
 - ／＼ 初始节点 S_0 在希望解树 T 中；
 - ／＼ 如果 n 是具有子节点 n_1, n_2, \dots, n_k 的或节点，则 n 的某个子节点 n_i 在希望解树 T 中的充要条件是

$$\min_{1 \leq i \leq k} \{c(n, n_i) + h(n_i)\}$$

- ／＼ 如果 n 是与节点，则 n 的全部子节点都在希望解树 T 中。
- ／＼ 与/或树的启发式搜索需要不断地选择、修正希望树，其搜索过程如下（过于复杂，并且老师没认真讲）：

 1. 把初始节点 S_0 放入 Open 表中，计算 $h(S_0)$
 2. 计算希望解树 T
 3. 依次在 Open 表中取出 T 的端节点，放入 Closed 表，并记该节点为 n 。
 4. 如果节点 n 为终止节点，则做下列工作：
 - (a) 标记节点 n 为可解节点
 - (b) 在 T 上应用可解标记过程，对 n 的先辈节点中的所有可解节点进行标记
 - (c) 如果初始节点 S_0 能够被标记为可解节点，则 T 就是最优解树，成功退出
 - (d) 否则，从 Open 表中删去具有可解先辈的所有节点
 - (e) 转第 2 步
 5. 如果节点 n 不是终止节点，但可扩展，则做下列工作：
 - (a) 扩展节点 n ，生成 n 的所有子节点
 - (b) 把这些子节点都放入 Open 表中，并为每个子节点设置指向父节点 n 的指针
 - (c) 计算这些子节点及其先辈节点的 h 值
 - (d) 转第 2 步
 6. 如果节点 n 不是终止节点，且不可扩展，则做下列工作：
 - (a) 标记节点 n 为不可解节点
 - (b) 在 T 上应用不可解标记过程，对 n 的先辈节点中的所有不可解节点进行标记
 - (c) 如果初始节点 S_0 能够被标记为不可解节点，则问题无解，失败退出
 - (d) 否则，从 Open 表中删去具有不可解先辈的所有节点
 - (e) 转第 2 步

4.5 博弈树的启发式搜索

- ／＼ 在双人完备信息博弈过程中，双方都希望自己能够获胜。因此，当任何一方走步时，都是选择对自己最有利而对另一方最不利的行动方案。

假设博弈的一方（先手）为 MAX，另一方为 MIN。在博弈过程的每步，从 MAX 方的观点看，可供自己选择的那些行动方案之间是「或」的关系，原因是主动权掌握在 MAX 手里，选择哪个方案完全是由自己决定的；而那些可供对方选择的行动方案之间是「与」的关系，原因是主动权掌握在 MIN 的手里，任何一个方案都有可能被 MIN 选中，MAX 必须防止那种对自己最为不利的情况的发生。

若把双人完备信息博弈过程用图表示出来，就可得到一棵与/或树，这种与/或树被称为博弈树。在博弈树中，那些下一步该 MAX 走步的节点称为 MAX 节点，而下一步该 MIN 走步的节点称为 MIN 节点。博弈树具有如下特点：

- ★ 博弈的初始状态是初始节点。
- ★ 博弈树中的“或”节点和“与”节点是逐层交替出现的。
- ★ 整个博弈过程始终站在某一方的立场上，所有能使自己一方获胜的终局都是本原问题，相应的节点是可解节点，所有使对方获胜的终局都是不可解节点。

极大/极小过程：

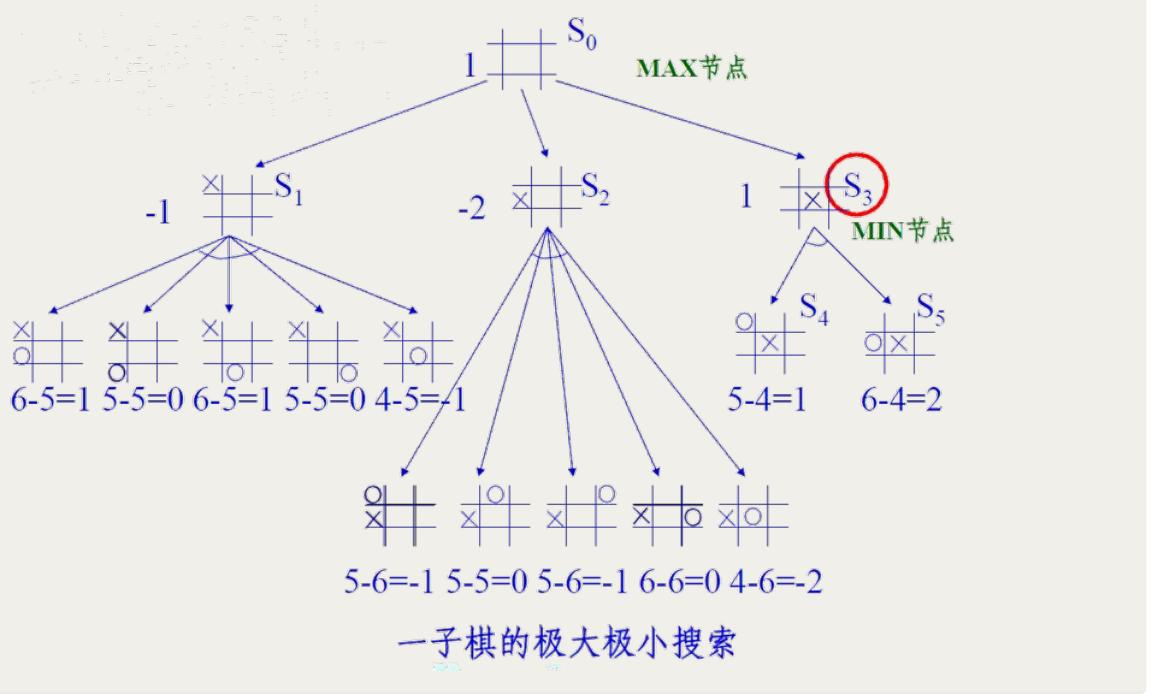
- ★ 用当前考察的节点生成一棵部分博弈树，由于该博弈树的叶节点一般不是哪一方的获胜节点，因此需要利用估价函数 $f(n)$ 对叶节点进行静态估值：
 - 一般来说，那些对 MAX 有利的节点，其估价函数取正值；
 - 那些对 MIN 有利的节点，其估价函数取负值；
 - 那些使双方利益均等的节点，其估价函数取接近于 0 的值。
- ★ 为了计算非叶节点的值，必须从叶节点向上倒退。由于 MAX 方总是选择估值最大的走步，因此，MAX 节点的倒退值应该取其后继节点估值的最大值。相反 MIN 节点的倒退值应取其后继节点估值的最小值。
- ★ 这样一步一步地计算倒退值，直至求出初始节点的倒退值为止。由于此时是站在 MAX 的立场上，因此应选择具有最大倒退值的走步。这一过程称为极大/极小过程。

一字棋游戏例子：

设有一个三行三列的棋盘，两个棋手轮流走步，每个棋手走步时往空格上摆一个自己的棋子，谁先使自己的棋子成三子一线为赢。设 MAX 方的棋子用 \times 标记，MIN 方的棋子用 \circ 标记，并规定 MAX 方先走步。

为了对叶节点进行静态估值，规定估价函数 $e(P)$ 如下：

- ★ 若 P 是 MAX 的必胜局，则 $e(P) = +\infty$ ；
- ★ 若 P 是 MIN 的必胜局，则 $e(P) = -\infty$ ；
- ★ 若 P 对 MAX、MIN 都是胜负未定局，则 $e(P) = e(+P) - e(-P)$ ， $e(+P)$ 表示棋局 P 上有可能使 \times 成三子一线的数目， $e(-P)$ 表示棋局 P 上有可能使 \circ 成三子一线的数目。



② $\alpha - \beta$ 剪枝 100:

上述极大/极小过程是先生成与/或树，再计算各节点的估值，这种生成节点和计算估值相分离的搜索方式，需要生成规定深度内的所有节点，因此搜索效率较低。如果能边生成节点边对节点估值，从而可以剪去一些没用的分支，这种技术称为 $\alpha - \beta$ 剪枝过程。

MAX 节点的 α 值为当前子节点的最大倒退值。

MIN 节点的 β 值为当前子节点的最小倒退值。

$\alpha - \beta$ 剪枝的规则如下：

① β 剪枝：

任何 MAX 节点 n 的 α 值大于或等于它先辈节点的 β 值，则 n 以下的分支可停止搜索，并令节点 n 的倒退值为 α 。

此时 MAX 节点的未扩展节点要剪枝，已扩展节点不需要处理，因为正是从已扩展节点得知 α 值已大于 β 值。

② α 剪枝：任何 MIN 节点 n 的 β 值小于或等于它先辈节点的 α 值，则 n 以下的分支可停止搜索，并令节点 n 的倒退值为 β 。

左图为 α 剪枝，右图为 β 剪枝：

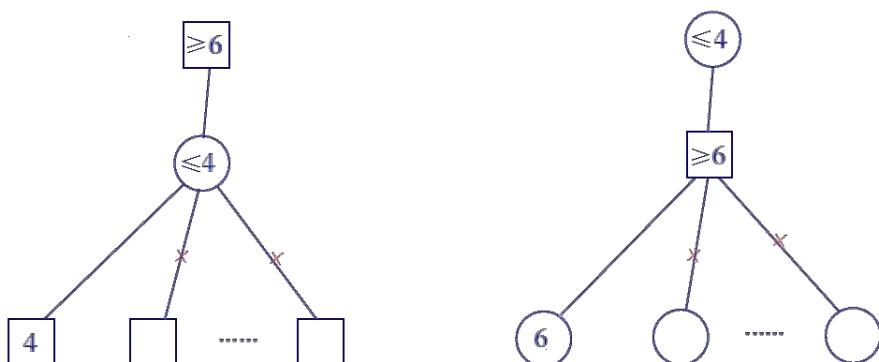


Figure: $\alpha - \beta$ 剪枝

5 不确定性推理

5.1 概述

不确定性推理是指建立在不确定性知识和证据基础上的推理，如不完备、不精确知识的推理、模糊知识的推理等。实际上是一种从不确定的初始证据出发，通过运用不确定性知识，最终推出具有一定程度的不确定性又是合理或基本合理的结论的思维过程。

采用不确定性推理原因

- ❖ 所需知识不完备、不精确
- ❖ 所需知识描述模糊
- ❖ 多种原因导致同一结论（不能真正原因）
- ❖ 解题方案不唯一

知识不确定性表示：

- ❖ 考虑因素：问题的描述能力，推理中不确定性的计算
- ❖ 含义：知识的确定性程度，或动态强度
- ❖ 表示：
 - ❖ 概率： $[0, 1]$ ，0 接近于假，1 接近于真
 - ❖ 可信度： $[-1, 1]$ ，大于 0 接近于真，小于 0 接近于假
 - ❖ 隶属度： $[0, 1]$ ，越接近于 0 隶属度越低，反之越高

证据不确定性的表示：

- ❖ 证据的类型：
 - ❖ 按证据组织：基本证据，组合证据
 - ❖ 按证据来源：初始证据，中间结论
- ❖ 表示方法和知识不确定性一致

不确定性的匹配：

- ❖ 含义：不确定的前提条件与不确定的事实匹配
- ❖ 问题：前提是不确定的，事实也是不确定的
- ❖ 方法：设计一个计算相似程度的算法，给出相似的限度
- ❖ 标志：相似度落在规定限度（阈值）内为匹配，否则为不匹配

组合证据不确定性的计算：

- ❖ 证据的组合方式：析取的关系，合取的关系。
- ❖ 证据的不确定性计算方法：基于基本证据的最大/最小方法、概率方法和有界方法等。

不确定性的更新：

- ❖ 如何用证据的不确定性去更新结论的不确定性：

不同推理方法的解决方法不同

- ❖ 如何在推理中把初始证据的不确定性传递给最终结论：

不同推理方法的解决方法基本相同，即把当前结论及其不确定性作为新的结论放入综合数据库，依次传递，直到得出最终结论

ℳ 不确定性结论的合成:

ℳ 含义: 多个不同知识推出同一结论, 且不确定性程度不同

ℳ 方法: 视不同推理方法而定

5.2 可信度推理

ℳ 可信度推理是一种基于确定性理论 (confirmation theory) 的不确定性推理方法, 可信度推理模型也称为 CF (Certainty Factor) 模型, 结合了概率论和模糊集合论等方法

ℳ 可信度: 根据经验对一个事物或现象为真的相信程度, 不是概率。

ℳ 知识不确定性的表示:

在 CF 模型中, 知识是用产生式规则表示的, 其一般形式为 IF E THEN H ($CF(H, E)$):

ℳ E 是知识的前提证据, 可以是一个简单条件, 也可以是由合取和析取构成的复合条件

ℳ H 是知识的结论, 可以是一个单一的结论, 也可以是多个结论

ℳ $CF(H, E)$ 是知识的可信度, 或称为规则强度, 取值范围是 $[-1, 1]$, 其值表示当证据 E 为真时, 该证据对结论 H 为真的支持程度。值越大, 说明 E 对结论 H 为真的支持程度越大

静态强度 $CF(H, E)$: 知识的强度, 即当 E 所对应的证据为真时对 H 的影响程度。

动态强度 $CF(E)$: 证据 E 当前的不确定性程度。

ℳ 在 CF 模型中, 把 $CF(H, E)$ 定义为

$$CF(H, E) = MB(H, E) - MD(H, E)$$

式中, MB (Measure Belief) 称为信任增长度, 表示因证据 E 的出现, 使结论 H 为真的信任增长度, 定义为

$$MB(H, E) = \begin{cases} 1 & P(H) = 1 \\ \frac{\max\{P(H | E), P(H)\} - P(H)}{1 - P(H)} & \text{其他} \end{cases}$$

MD (Measure Disbelief) 称为不信任增长度, 表示因证据 E 的出现, 对结论 H 为真的不信任增长度, 或称为对结论 H 为假的信任增长度。定义为

$$MD(H, E) = \begin{cases} 1 & P(H) = 0 \\ \frac{\min\{P(H | E), P(H)\} - P(H)}{-P(H)} & \text{其他} \end{cases}$$

在以上两个式子中, $P(H)$ 表示 H 的先验概率; $P(H | E)$ 表示在证据 E 下, 结论 H 的条件概率。

可得到 $CF(H, E)$ 的计算公式:

$$CF(H, E) = \begin{cases} MB(H, E) - 0 = \frac{P(H | E) - P(H)}{1 - P(H)} & \text{IF } P(H | E) > P(H) \\ 0 & \text{IF } P(H | E) = P(H) \\ 0 - MD(H, E) = -\frac{P(H) - P(H | E)}{P(H)} & \text{IF } P(H | E) < P(H) \end{cases}$$

由上得：

互斥性：

- 当 $MB(H, E) > 0$ 时, $MD(H, E) = 0$, 有 $P(H | E) > P(H)$, $CF(H, E) > 0$, 说明由于证据 E 的出现增加了 H 的信任程度
- 当 $MD(H, E) > 0$ 时, $MB(H, E) = 0$, 有 $P(H | E) < P(H)$, $CF(H, E) < 0$, 说明由于证据 E 的出现增加了 H 的不信任程度

MB, MD 都是非负的

- 若 $CF(H, E) = 0$, 则 $P(H | E) = P(H)$, 即 H 的后验概率等于其先验概率, 说明证据 E 与 H 无关
- 对 H 的信任增长度等于对非 H 的不信任增长度

对同一证据 E , 若支持若干个不同的结论 $H_i (i = 1, 2, \dots, n)$, 则

$$\sum_{i=1}^n CF(H_i, E) \leq 1$$

证据不确定性的表示（同样用可信度可信度 $CF(E)$ 衡量）：

1. $CF(E) = 1$, 证据 E 肯定为真。
2. $CF(E) = -1$, 证据 E 肯定为假。
3. $CF(E) = 0$, 对证据 E 一无所知。
4. $0 < CF(E) < 1$, 证据 E 以 $CF(E)$ 程度为真。
5. $-1 < CF(E) < 0$, 证据 E 以 $CF(E)$ 程度为假。
6. 该证据的否定记为 $\neg E$ 。若已知 E 的可信度为 $CF(E)$, 则 $CF(\neg E) = -CF(E)$

组合证据：

当组合证据是多个单一证据的合取时, 若已知 $CF(E_1), CF(E_2), \dots, CF(E_n)$, 则

$$CF(E) = \min \{CF(E_1), CF(E_2), \dots, CF(E_n)\}$$

当组合证据是多个单一证据的析取时, 若已知 $CF(E_1), CF(E_2), \dots, CF(E_n)$, 则

$$CF(E) = \max \{CF(E_1), CF(E_2), \dots, CF(E_n)\}$$

CF 模型中的不确定性推理:

从不确定的初始证据出发, 通过运用相关的不确定性知识, 最终推出结论并求出结论的可信度值。结论 H 的可信度由下式计算:

$$CF(H) = CF(H, E) \times \max\{0, CF(E)\}$$

在该模型中没有考虑证据为假时对结论 H 所产生的影响。

当证据为真, 即 $CF(E) = 1$ 时, 由上式可推出 $CF(H) = CF(H, E)$ 。

不确定性结论的合成:

如果可由多条知识推出一个相同结论, 并且这些知识的前提证据相互独立, 结论的可信度又不相同, 则可用不确定性的合成算法求出该结论的综合可信度。

其合成过程是先把第一条与第二条合成, 再用该合成后的结论与第三条合成, 以此类推, 直到全部合成为止。

两两合成过程:

设有如下知识:

$\textcircled{1}$ IF E_1 THEN $H(CF(H, E_1))$

$\textcircled{2}$ IF E_2 THEN $H(CF(H, E_2))$

1. 分别对每条知识求出其 $CF(H)$, 即

$$CF_1(H) = CF(H, E_1) \cdot \max\{0, CF(E_1)\}$$

$$CF_2(H) = CF(H, E_2) \cdot \max\{0, CF(E_2)\}$$

2. 用如下公式求 E_1 与 E_2 对 H 的综合可信度:

$$CF(H) = \begin{cases} CF_1(H) + CF_2(H) - CF_1(H)CF_2(H) & \text{IF } CF_1(H) \geq 0 \text{ AND } CF_2(H) \geq 0 \\ CF_1(H) + CF_2(H) + CF_1(H)CF_2(H) & \text{IF } CF_1(H) < 0 \text{ AND } CF_2(H) < 0 \\ \frac{CF_1(H) + CF_2(H)}{1 - \min\{|CF_1(H)|, |CF_2(H)|\}} & \text{IF } CF_1(H) \text{ 与 } CF_2(H) \text{ 异号} \end{cases}$$

例子:

设有如下一组知识:

$\textcircled{1}$ r_1 : IF E_1 THEN $H(0.9)$

$\textcircled{2}$ r_2 : IF E_2 THEN $H(0.6)$

$\textcircled{3}$ r_3 : IF E_3 THEN $H(-0.5)$

$\textcircled{4}$ r_4 : IF E_4 AND (E_5 OR E_6) THEN $E_1(0.8)$

已知: $CF(E_2) = 0.8$, $CF(E_3) = 0.6$, $CF(E_4) = 0.5$, $CF(E_5) = 0.6$, $CF(E_6) = 0.8$, 求: $CF(H)$

解:

由 r_4 得

$$\begin{aligned}
\text{CF}(E_1) &= 0.8 \times \max \{0, \text{CF}(E_4 \text{ AND } (E_5 \text{ OR } E_6))\} \\
&= 0.8 \times \max \{0, \min \{\text{CF}(E_4), \text{CF}(E_5 \text{ OR } E_6)\}\} \\
&= 0.8 \times \max \{0, \min \{\text{CF}(E_4), \max \{\text{CF}(E_5), \text{CF}(E_6)\}\}\} \\
&= 0.8 \times \max \{0, \min \{\text{CF}(E_4), \max \{0.6, 0.8\}\}\} \\
&= 0.8 \times \max \{0, \min \{0.5, 0.8\}\} \\
&= 0.8 \times \max \{0, 0.5\} \\
&= 0.4
\end{aligned}$$

由 r_1 得

$$\begin{aligned}
\text{CF}_1(H) &= \text{CF}(H, E_1) \times \max \{0, \text{CF}(E_1)\} \\
&= 0.9 \times \max \{0, 0.4\} = 0.36
\end{aligned}$$

由 r_2 得

$$\begin{aligned}
\text{CF}_2(H) &= \text{CF}(H, E_2) \times \max \{0, \text{CF}(E_2)\} \\
&= 0.6 \times \max \{0, 0.8\} = 0.48
\end{aligned}$$

由 r_3 得

$$\begin{aligned}
\text{CF}_3(H) &= \text{CF}(H, E_3) \times \max \{0, \text{CF}(E_3)\} \\
&= -0.5 \times \max \{0, 0.6\} = -0.3
\end{aligned}$$

根据结论不确定性的合成算法得

$$\begin{aligned}
\text{CF}_{1,2}(H) &= \text{CF}_1(H) + \text{CF}_2(H) - \text{CF}_1(H)\text{CF}_2(H) \\
&= 0.36 + 0.48 - 0.36 \times 0.48 \\
&= 0.84 - 0.17 = 0.67 \\
\text{CF}_{1,2,3}(H) &= \frac{\text{CF}_{1,2}(H) + \text{CF}_3(H)}{1 - \min \{|\text{CF}_{1,2}(H)|, |\text{CF}_3(H)|\}} \\
&= \frac{0.67 - 0.3}{1 - \min \{0.67, 0.3\}} = \frac{0.37}{0.7} = 0.53
\end{aligned}$$

这就是所求的综合可信度, 即 $\text{CF}(H) = 0.53$ 。

5.3 主观 Bayes 推理

⊗ 全概率公式:

$$P(B) = \sum_{i=1}^n P(A_i) \times P(B | A_i) \quad (A_i \text{ 与 } A_j (i \neq j) \text{ 互不相容 })$$

⊗ Bayes 公式:

$$P(A_k | B) = \frac{P(A_k \cap B)}{P(B)} = \frac{P(A_k)P(B | A_k)}{\sum_i P(A_i)P(B | A_i)}$$

5.3.1 知识不确定性的表示形式：

主观 Bayes 方法中的知识是用产生式表示的，其形式为 IF E THEN (LS, LN) H

其中，(LS, LN) 用来表示该知识的知识强度，LS 和 LN 的表示形式分别为

$$\begin{aligned} \text{LS} &= \frac{P(E | H)}{P(E | \neg H)} \\ \text{LN} &= \frac{P(\neg E | H)}{P(\neg E | \neg H)} = \frac{1 - P(E | H)}{1 - P(E | \neg H)} \end{aligned}$$

LS 和 LN 的取值范围均为 $[0, +\infty)$ 。

2 LS 和 LN 的含义：

$$\begin{aligned} P(H | E) &= \frac{P(E | H)P(H)}{P(E)} \\ P(\neg H | E) &= \frac{P(E | \neg H)P(\neg H)}{P(E)} \end{aligned}$$

将两式相除，得

$$\frac{P(H | E)}{P(\neg H | E)} = \frac{P(E | H)}{P(E | \neg H)} \times \frac{P(H)}{P(\neg H)}$$

为讨论方便，下面引入几率函数：

$$O(X) = \frac{P(X)}{1 - P(X)} = \frac{P(X)}{P(\neg X)}$$

随着 $P(X)$ 的增大， $O(X)$ 也在增大，并且 $P(X) = 0$ 时，有 $O(X) = 0$ ； $P(X) = 1$ 时，有 $O(X) = +\infty$ 。这样，就可以把取值为 $[0, 1]$ 的 $P(X)$ 放大到取值为 $[0, +\infty)$ 的 $O(X)$ 。

将几率函数代入公式

$$O(H | E) = \frac{P(E | H)}{P(E | \neg H)} O(H)$$

再把 LS 代入此式，可得

$$O(H | E) = \text{LS} \times O(H)$$

同理，可得到 LN 的公式

$$O(H | \neg E) = \text{LN} \times O(H)$$

上面两式就是修改的 Bayes 公式。从这两个公式可以看出：

当 E 为真时，可以利用 LS 将 H 的先验几率 $O(H)$ 更新为其后验几率 $O(H | E)$

当 E 为假时，可以利用 LN 将 H 的先验几率 $O(H)$ 更新为其后验几率 $O(H | \neg E)$

2 LS 的性质：

反映的是 E 的出现对 H 为真的影响程度，称 LS 为知识的充分性度量。

- ／＊ 当 $LS > 1$ 时， $O(H | E) > O(H)$ ，说明 E 支持 H ，增大了 H 为真的概率：
 - ／＊ LS 越大， $O(H | E)$ 比 $O(H)$ 大得越多， E 对 H 的支持越充分。
 - ／＊ 当 $LS \rightarrow \infty$ 时， $O(H | E) \rightarrow \infty$ ，即 $P(H | E) \rightarrow 1$ ，表示由于 E 的存在，将导致 H 为真。
- ／＊ 当 $LS = 1$ 时， $O(H | E) = O(H)$ ，说明 E 对 H 没有影响。
- ／＊ 当 $LS < 1$ 时， $O(H | E) < O(H)$ ，说明 E 不支持 H ，减小了 H 为真的概率。
- ／＊ 当 $LS = 0$ 时， $O(H | E) = 0$ ，说明 E 的存在使 H 为假。

3 LN 的性质：

反映的是当 E 不存在时对 H 为真的影响，因此称 LN 为知识的必要性度量。

- ／＊ 当 $LN > 1$ 时， $O(H | \neg E) > O(H)$ ，说明 $\neg E$ 支持 H ：
 - ／＊ LN 越大， $P(H | \neg E)$ 就越大，即 $\neg E$ 对 H 为真的支持就越强。
 - ／＊ 当 $LN \rightarrow \infty$ 时， $O(H | \neg E) \rightarrow \infty$ ，即 $P(H | \neg E) \rightarrow 1$ ，表示由于 $\neg E$ 的存在，将导致 H 为真。
- ／＊ 当 $LN = 1$ 时， $O(H | \neg E) = O(H)$ ，说明 $\neg E$ 对 H 没有影响。
- ／＊ 当 $LN < 1$ 时， $O(H | \neg E) < O(H)$ ，说明 $\neg E$ 不支持 H ，即由于 $\neg E$ 的存在，将使 H 为真的可能性下降。
- ／＊ 当 $LN \rightarrow 0$ 时， $O(H | \neg E) \rightarrow 0$ ，即 LN 越小， E 的不出现就越反对 H 为真，这说明 H 越需要 E 的出现。
- ／＊ 当 $LN = 0$ 时， $O(H | \neg E) = 0$ ，说明 $\neg E$ 的存在（即 E 不存在）将导致 H 为假。

在实际系统中，LS 和 LN 的值均是由领域专家根据经验给出的，而不是由 LS 和 LN 计算出来的。当证据 E 越是支持 H 为真时，则 LS 的值应该越大；当证据 E 对 H 越是重要时，则相应的 LN 的值应该越小。

4 LS 与 LN 的关系：

由于 E 和 $\neg E$ 不会同时支持或同时排斥 H ，因此只有下述 3 种情况存在：

- ／＊ $LS > 1$ 且 $LN < 1$
- ／＊ $LS < 1$ 且 $LN > 1$
- ／＊ $LS = LN = 1$ 。

事实上，如果 $LS > 1$ ，即

$$\begin{aligned}
LS > 1 &\Leftrightarrow \frac{P(E | H)}{P(E | \neg H)} > 1 \Leftrightarrow P(E | H) > P(E | \neg H) \\
&\Leftrightarrow 1 - P(E | H) < 1 - P(E | \neg H) \\
&\Leftrightarrow P(\neg E | H) < P(\neg E | \neg H) \Leftrightarrow \frac{P(\neg E | H)}{P(\neg E | \neg H)} < 1 \\
&\Leftrightarrow LN < 1
\end{aligned}$$

5.3.2 证据的不确定性表示

基本证据的表示：

在主观 Bayes 方法中，证据 E 的不确定性是用其概率或几率来表示的。概率与几率之间的关系为

$$O(E) = \frac{P(E)}{1 - P(E)} = \begin{cases} 0 & E \text{ 为假时} \\ \infty & E \text{ 为真时} \\ (0, +\infty) & E \text{ 非真也非假时} \end{cases}$$

在有些情况下，除需要考虑证据 E 的先验概率与先验几率外，往往还需要考虑在当前观察下证据 E 的后验概率或后验几率。以概率情况为例，对初始证据 E ，用户可以根据当前观察 S 将其先验概率 $P(E)$ 更改为后验概率 $P(E | S)$ ，即相当于给出证据 E 的动态强度。

组合证据：

合取：

如果已知在当前观察 S 下，每个单一证据 E_i 有概率 $P(E_1 | S), P(E_2 | S), \dots, P(E_n | S)$ ，则

$$P(E | S) = \min \{P(E_1 | S), P(E_2 | S), \dots, P(E_n | S)\}$$

析取：

如果已知在当前观察 S 下，每个单一证据 E_i 有概率 $P(E_1 | S), P(E_2 | S), \dots, P(E_n | S)$ ，则

$$P(E | S) = \max \{P(E_1 | S), P(E_2 | S), \dots, P(E_n | S)\}$$

5.3.3 不确定性的更新

主观 Bayes 方法推理的任务：

根据证据 E 的概率 $P(E)$ 及 LS 和 LN 的值，把 H 的先验概率 $P(H)$ 或先验几率 $O(H)$ 更新为当前观察 S 下的后验概率 $P(H | S)$ 或后验几率 $O(H | S)$ 。目的就是看 E 对最终结论 H 的影响。根据证据的不同情况去计算其后验概率或后验几率。

证据在当前观察下肯定为真：

$\nearrow P(E) = P(E | S) = 1$

\nearrow 将 H 的先验几率更新为后验几率的公式 $O(H | E) = LS \times O(H)$

\nearrow 如果是把 H 的先验概率更新为其后验概率，公式为：

$$P(H | E) = \frac{LS \times P(H)}{(LS - 1) \times P(H) + 1}$$

2 证据在当前观察下肯定为假：

✓ $P(E) = P(E | S) = 0, P(\neg E) = 1$

✓ 将 H 的先验几率更新为后验率的公式为式，即 $O(H | \neg E) = LN \times O(H)$ 。

✓ 如果把 H 的先验概率更新为其后验概率，公式为：

$$P(H | \neg E) = \frac{LN \times P(H)}{(LN - 1) \times P(H) + 1}$$

3 证据在当前观察下既非为真又非为假：

✓ 当证据既非为真又非为假时，不能再用上面的方法计算 H 的后验概率，而需要使用：

$$P(H | S) = P(H | E)P(E | S) + P(H | \neg E)P(\neg E | S)$$

✓ 分 4 种情况讨论公式：

✓ $P(E | S) = 1, P(\neg E | S) = 0$ ：

$$P(H | S) = P(H | E) = \frac{LS \times P(H)}{(LS - 1) \times P(H) + 1}$$

这实际上是证据肯定为真的情况。

✓ $P(E | S) = 0, P(\neg E | S) = 1$ ：

$$P(H | S) = P(H | \neg E) = \frac{LN \times P(H)}{(LN - 1) \times P(H) + 1}$$

这实际上是证据肯定不存在的情况。

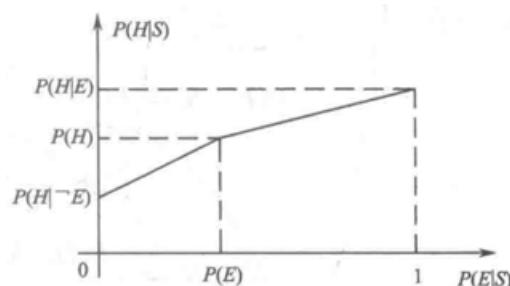
✓ $P(E | S) = P(E)$ ：

表示 E 与 S 无关，由全概率公式可得：

$$P(H | S) = P(H | E)P(E | S) + P(H | \neg E)P(\neg E | S) = P(H | E)P(E) + P(H | \neg E)P(\neg E)$$

✓ 其他情况：

分段插值



$$P(H | S) = \begin{cases} P(H | \neg E) + \frac{P(H) - P(H | \neg E)}{P(E)} \times P(E | S) & 0 \leq P(E | S) < P(E) \\ P(H) + \frac{P(H | E) - P(H)}{1 - P(E)} \times [P(E | S) - P(E)] & P(E) \leq P(E | S) \leq 1 \end{cases}$$

🔗 不确定性结论的合成:

💡 假设有 n 条知识都支持同一结论 H , 并且这些知识的前提条件分别是 n 个相互独立的证据 E_1, E_2, \dots, E_n , 而每个证据所对应的观察又分别是 S_1, S_2, \dots, S_n 。

💡 在这些观察下, 求 H 的后验概率的方法是:

1. 对每条知识分别求出 H 的后验几率 $O(H | S_i)$
2. 利用这些后验几率并按下述公式求出所有观察下 H 的后验几率:

$$O(H | S_1, S_2, \dots, S_n) = \frac{O(H | S_1)}{O(H)} \times \frac{O(H | S_2)}{O(H)} \times \dots \times \frac{O(H | S_n)}{O(H)} \times O(H)$$

主观 Bayes 方法的主要优点是理论模型精确, 灵敏度高, 不仅考虑了证据间的关系, 还考虑了证据存在与否对假设的影响, 因此是一种较好的方法。

主要缺点是需要的主观概率太多, 专家不易给出。

5.3.4 例子

设有规则:

$r_1 : \text{IF } E_1 \text{ THEN } (2, 0.001) H_1$

$r_2 : \text{IF } E_1 \text{ AND } E_2 \text{ THEN } (100, 0.001) H_1$

$r_3 : \text{IF } H_1 \text{ THEN } (200, 0.01) H_2$

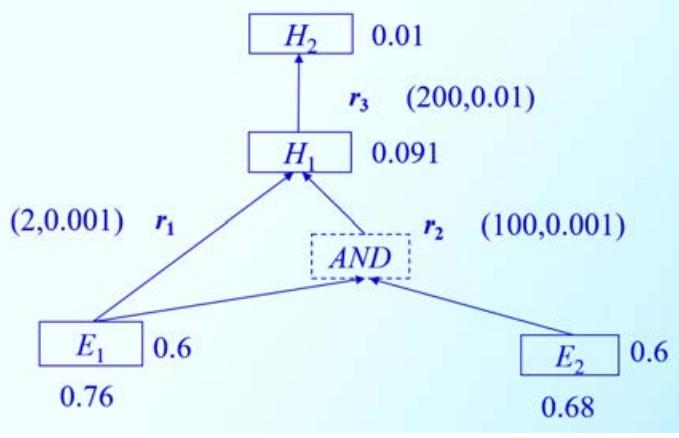
已知: $P(E_1) = P(E_2) = 0.6, P(H_1) = 0.091, P(H_2) = 0.01$ 。

用户回答: $P(E_1 | S_1) = 0.76, P(E_2 | S_2) = 0.68$

求: $P(H_2 | S_1, S_2)$

解:

由已知知识得到的推理网络如图所示:



1. 计算 $O(H_1 | S_1)$:

(a) 先把 H_1 的先验概率 $P(H_1)$ 更新为在 E_1 下的后验概率:

$$P(H_1 | E_1) = \frac{LS_1 \times P(H_1)}{(LS_1 - 1) \times P(H_1) + 1} = \frac{2 \times 0.091}{(2 - 1) \times 0.091 + 1} = 0.167$$

(b) 由于 $P(E_1 | S_1) = 0.76 > P(E_1)$, 使用分段插值的后半部分, 得到在当前观察 S_1 下 H_1 的后验概率

$$\begin{aligned} P(H_1 | S_1) &= P(H_1) + \frac{P(H_1 | E_1) - P(H_1)}{1 - P(E_1)} (P(E_1 | S_1) - P(E_1)) \\ &= 0.091 + \frac{(0.167 - 0.091)}{1 - 0.6} \times (0.76 - 0.6) = 0.121 \\ O(H_1 | S_1) &= \frac{P(H_1 | S_1)}{1 - P(H_1 | S_1)} = \frac{0.121}{1 - 0.121} = 0.138 \end{aligned}$$

2. 计算 $O(H_1 | (S_1 \text{ AND } S_2))$:

由于 r_2 的前件是 E_1, E_2 的合取关系, 且 $P(E_1 | S_1) = 0.76, P(E_2 | S_2) = 0.68$, 即 $P(E_2 | S_2) < P(E_1 | S_1)$ 。按合取取最小的原则, 这里仅考虑 E_2 对 H_1 的影响, 即把计算 $P(H_1 | (S_1 \text{ AND } S_2))$ 的问题转化为计算 $O(H_1 | S_2)$ 的问题。

把 H_1 的先验概率 $P(H_1)$ 更新为在 E_2 下的后验概率:

$$\begin{aligned} P(H_1 | E_2) &= \frac{LS_2 \times P(H_1)}{(LS_2 - 1) \times P(H_1) + 1} \\ &= \frac{100 \times 0.091}{(100 - 1) \times 0.091 + 1} = 0.909 \end{aligned}$$

又由于 $P(E_2 | S_2) > P(E_2)$, 使用后半部分, 得到在当前观察 S_2 下 H_1 的后验概率

$$\begin{aligned} P(H_1 | S_2) &= P(H_1) + \frac{P(H_1 | E_2) - P(H_1)}{1 - P(E_2)} \times (P(E_2 | S_2) - P(E_2)) \\ &= 0.091 + \frac{(0.909 - 0.091)}{1 - 0.6} \times (0.68 - 0.6) = 0.255 \\ O(H_1 | S_2) &= \frac{P(H_1 | S_2)}{1 - P(H_1 | S_2)} = \frac{0.255}{1 - 0.255} = 0.342 \end{aligned}$$

3. 计算 $O(H_1 | S_1, S_2)$:

(a) 将 H_1 的先验概率转换为先验几率

$$O(H_1) = \frac{P(H_1)}{1 - P(H_1)} = \frac{0.091}{1 - 0.091} = 0.100$$

(b) 根据合成公式计算 H_1 的后验几率

$$O(H_1 | S_1, S_2) = \frac{O(H_1 | S_1)}{O(H_1)} \times \frac{O(H_1 | S_2)}{O(H_1)} \times O(H_1) = \frac{0.138}{0.1} \times \frac{0.342}{0.1} \times 0.1 = 0.472$$

(c) 将后验几率转换为后验概率

$$P(H_1 | S_1, S_2) = \frac{O(H_1 | S_1, S_2)}{1 + O(H_1 | S_1, S_2)} = \frac{0.472}{1 + 0.472} = 0.321$$

4. 计算 $P(H_2 | S_1, S_2)$:

(a) 对 r_3, H_1 相当于已知事实, H_2 为结论。将 H_2 的先验概率 $P(H_2)$ 更新为在 H_1 下的后验概率

$$\begin{aligned} P(H_2 | H_1) &= \frac{LS_3 \times P(H_2)}{(LS_3 - 1) \times P(H_2) + 1} \\ &= \frac{200 \times 0.01}{(200 - 1) \times 0.01 + 1} = 0.669 \end{aligned}$$

(b) 由于 $P(H_1 | S_1, S_2) = 0.321 > P(H_1)$, 仍使用后半部分, 得到在当前观察 S_1 和 S_2 下 H_2 的后验概率

$$\begin{aligned} P(H_2 | S_1, S_2) &= P(H_2) + \frac{P(H_2 | H_1) - P(H_2)}{1 - P(H_1)} \times [P(H_1 | S_1, S_2) - P(H_1)] \\ &= 0.01 + \frac{0.669 - 0.01}{1 - 0.091} \times (0.321 - 0.091) = 0.177 \end{aligned}$$

可以看出, H_2 先验概率是 0.01, 通过运用知识 r_1, r_2, r_3 及初始证据的概率进行推理, 最后推出的 H_2 的后验概率为 0.177, 相当于概率增加了 16 倍

5.4 证据理论

5.4.1 证据理论的形式化描述

ℳ 证据理论的基本思想是:

↗ 先定义一个概率分配函数

↗ 再利用该概率分配函数建立相应的信任函数、似然函数及类概率函数, 分别用于描述知识的精确信任度、不可驳斥信任度和估计信任度

↗ 最后利用这些不确定性度量, 按照证据理论的推理模型, 去完成其推理工作。

ℳ 概率分配函数:

- 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，其作用是把命题的不确定性转化为集合的不确定性。
- 概率分配函数的定义需要用到幂集的概念：

- 设 Ω 为变量 x 的所有可能取值的有限集合（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 ，且其中的每个元素都对应一个关于 x 取值情况的命题

例子：

设 $\Omega = \{ \text{红}, \text{黄}, \text{白} \}$ ，求 Ω 的幂集 2^Ω ：

解：

Ω 的幂集包括如下子集：

$A_0 = \emptyset, A_1 = \{ \text{红} \}, A_2 = \{ \text{黄} \}, A_3 = \{ \text{白} \}, A_4 = \{ \text{红}, \text{黄} \}, A_5 = \{ \text{红}, \text{白} \}, A_6 = \{ \text{黄}, \text{白} \}, A_7 = \{ \text{红}, \text{黄}, \text{白} \}$

其中， \emptyset 表示空集，也可表示为 $\{\}$ 。上述子集的个数正好是 $2^3 = 8$ 。

- 一般的概率分配函数：

设函数 $m : 2^\Omega \rightarrow [0, 1]$ ，且满足

$$\begin{aligned} m(\emptyset) &= 0 \\ \sum_{A \subseteq \Omega} m(A) &= 1 \end{aligned}$$

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。

对上例所给出的有限集 Ω ，若给定 2^Ω 上的一个基本函数

$$\begin{aligned} m : m(\{\}) &, m(\{\text{红}\}), m(\{\text{黄}\}), m(\{\text{白}\}), \\ &m(\{\text{红}, \text{黄}\}), m(\{\text{红}, \text{白}\}), m(\{\text{黄}, \text{白}\}), m(\{\text{红}, \text{黄}, \text{白}\}) \\ &= (0, 0.3, 0, 0.1, 0.2, 0.2, 0, 0.2) \end{aligned}$$

请说明该函数满足概率分配函数的定义。

解： $(0, 0.3, 0, 0.1, 0.2, 0.2, 0, 0.2)$ 分别是幂集 2^Ω 中各子集的基本概率数。显然 m 满足

$$\begin{aligned} m(\emptyset) &= 0 \\ \sum_{A \subseteq \Omega} m(A) &= 1 \end{aligned}$$

概率分配函数的定义。

- 概率分配函数的作用是把 Ω 的任意一个子集都映射为 $[0, 1]$ 上的一个数 $m(A)$ 。

当 $A \subset \Omega$ 且 A 由单个元素组成时, $m(A)$ 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, 但不知道这部分信任度该分给 A 中哪些元素; 当 $A = \Omega$ 时, 则 $m(A)$ 也表示不知道该如何分配。

对上例给出的有限集 Ω 及基本函数 m :

当 $A = \{\text{红}\}$ 时, 有 $m(A) = 0.3$, 表示对命题「 x 是红色」的精确信任度为 0.3。

当 $B = \{\text{红, 黄}\}$ 时, 有 $m(B) = 0.2$, 表示对命题「 x 或者是红色, 或者是黄色」的精确信任度为 0.2, 却不知道该把这 0.2 分给 { 红 } 还是分给 { 黄 }。

当 $C = \Omega = \{\text{红, 黄, 白}\}$ 时, 有 $m(\Omega) = 0.2$, 表示不知道该对这 0.2 如何分配, 但它不属于 { 红 }, 就一定属于 { 黄 } 或 { 白 }, 只是在现有认识下, 还不知道该如何分配而已。

概率分配函数不是概率。

在上例中, m 符合概率分配函数的定义, 但是

$$m(\{\text{红}\}) + m(\{\text{黄}\}) + m(\{\text{白}\}) = 0.3 + 0 + 0.1 = 0.4 < 1$$

因此 m 不是概率, 而概率 P 要求 $P(\text{红}) + P(\text{黄}) + P(\text{白}) = 1$

信任函数 (下限函数):

对任何命题 $A \subseteq \Omega$, 其信任函数为

$$\text{Bel}(A) = \sum_{B \subseteq A} m(B)$$

$$\text{Bel}(\Omega) = \sum_{B \subseteq \Omega} m(B) = 1$$

$\text{Bel}(A)$ 表示对 A 的总体信任度。

似然函数 (不可驳斥函数或上限函数):

对任何命题 $A \subseteq \Omega$, 其

$$\begin{aligned}\text{Pl}(A) &= 1 - \text{Bel}(\neg A) = 1 - \sum_{B \cap A = \emptyset} m(B) \\ \text{Pl}(\Omega) &= 1 - \text{Bel}(\neg \Omega) = 1 - \text{Bel}(\emptyset) = 1\end{aligned}$$

$\text{Pl}(A)$ 表示对 A 非假的信任度。

从上面的定义可以看出, 对任何命题 $A \subseteq \Omega$ 和 $B \subseteq \Omega$ 均有

$$\text{Pl}(A) - \text{Bel}(A) = \text{Pl}(B) - \text{Bel}(B) = m(\Omega)$$

它表示对 A (或者 B) 不知道的程度。

类概率函数:

命题 A 的类概率函数 f 定义:

$$f(A) = \text{Bel}(A) + \frac{|A|}{|D|} [\text{Pl}(A) - \text{Bel}(A)] \quad \forall A \in 2^D \subseteq D$$

其中, $|A|$ 与 $|D|$ 分别为 A 与 D 中的元素个数。

类概率函数 $f(A)$ 具有如下性质:

(1) $\sum_{i=1}^n f(\{A_i\}) = 1$, 对于 $A = \{A_1, A_2, \dots, A_n\}$

(2) 对于任何 $A \subseteq D$, 有 $\text{Bel}(A) \leq f(A) \leq \text{Pl}(A)$

(3) 对于任何 $A \subseteq D$, 有 $f(\neg A) = 1 - f(A)$

2 概率分配函数的正交和 (证据的组合):

设 m_1 和 m_2 是两个概率分配函数, 则其正交和 $m = m_1 \oplus m_2$:

$$m(A) = \frac{1}{K} \sum_{B \cap C = A} [m_1(B) \times m_2(C)]$$
$$K = 1 - \sum_{B \cap C = \emptyset} [m_1(B) \times m_2(C)] = \sum_{B \cap C \neq \emptyset} [m_1(B) \times m_2(C)]$$

如果 $K \neq 0$, 则正交和 m 也是一个概率分配函数

如果 $K = 0$, 则不存在正交和 m , 即不可能存在概率函数, 称 m_1 与 m_2 矛盾

5.4.2 证据理论的推理模型

2 知识不确定性的表示:

在 DS 理论中, 不确定性知识的表示形式为:

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

E 为前提条件, 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, \dots, h_n 是该子集中的元素

CF 是可信度因子, 用集合形式表示, 其中的元素 c_1, c_2, \dots, c_n 用来表示 h_1, h_2, \dots, h_n 的可信度, c_i 与 h_i 一一对应, c_i 应满足如下条件:

$$\begin{cases} c_i \geq 0 \\ \sum_{i=1}^n c_i \leq 1 \quad (i = 1, 2, \dots, n) \end{cases}$$

2 证据不确定性的表示:

DS 理论中将所有输入的已知数据、规则前提条件及结论部分的命题都称为证据。证据的不确定性用该证据的确定性表示。

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

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

条件部分命题 A 的确定性为:

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

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

在实际系统中, 如果是初始证据, 其确定性是由用户给出的。如果是推理过程中得出的中间结论, 则其确定性由推理得到。

组合证据:

合取:

$$\text{CER}(E | S) = \min \{\text{CER}(E_1 | S), \text{CER}(E_2 | S), \dots, \text{CER}(E_n | S)\}$$

析取:

$$\text{CER}(E | S) = \max \{\text{CER}(E_1 | S), \text{CER}(E_2 | S), \dots, \text{CER}(E_n | S)\}$$

不确定性的更新:

设有知识 IF E THEN $H = \{h_1, h_2, \dots, h_n\}$ CF = $\{c_1, c_2, \dots, c_n\}$, 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

$$m(\{h_1\}, \{h_2\}, \dots, \{h_n\}) = (\text{CER}(E) \times c_1, \text{CER}(E) \times c_2, \dots, \text{CER}(E) \times c_n)$$
$$m(\Omega) = 1 - \sum_{i=1}^n \text{CER}(E) \times c_i$$

如果有两条知识支持同一结论 H , 即

$$\text{IF } E_1 \text{ THEN } H = \{h_1, h_2, \dots, h_n\} \text{ CF}_1 = \{c_{11}, c_{12}, \dots, c_{1n}\}$$

$$\text{IF } E_2 \text{ THEN } H = \{h_1, h_2, \dots, h_n\} \text{ CF}_2 = \{c_{21}, c_{22}, \dots, c_{2n}\}$$

则按正交和求 $\text{CER}(H)$, 即先求出每一知识的概率分配函数

$$m_1(\{h_1\}, \{h_2\}, \dots, \{h_n\})$$
$$m_2(\{h_1\}, \{h_2\}, \dots, \{h_n\})$$

再用公式 $m = m_1 \oplus m_2$ 对 m_1 和 m_2 求正交和, 从而得到 H 的概率分配函数 m 。

如果有多条规则支持同一结论, 则用公式 $m = m_1 \oplus m_2 \oplus \dots \oplus m_n$ 求出 H 的概率分配函数 m 。

2. 求 $\text{Bel}(H)$ 、 $\text{Pl}(H)$ 及 $f(H)$:

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

$$\text{Pl}(H) = 1 - \text{Bel}(\neg H)$$

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

3. 求 $\text{CER}(H)$: 按公式 $\text{CER}(H) = \text{MD}(H | E') \times f(H)$ 计算结论 H 的确定性。

例子:

例3.8 设有规则

$r_1: IF E_1 AND E_2 THEN A=\{a_1, a_2\} CF=\{0.3, 0.5\}$

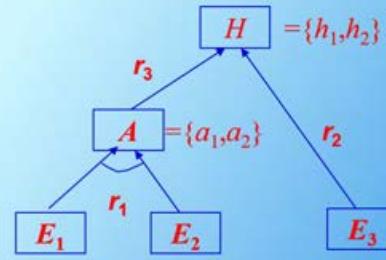
$r_2: IF E_3 THEN H=\{h_1, h_2\} CF=\{0.4, 0.2\}$

$r_3: IF A THEN H=\{h_1, h_2\} CF=\{0.1, 0.5\}$

已知: $\text{CER}(E_1)=0.8$ $\text{CER}(E_2)=0.6$ $\text{CER}(E_3)=0.9$

并假设 D 中的元素个数为 10, 求: $\text{CER}(H)=?$

解: 由给定知识形成的推理网络如右图所示:



1. 求 $\text{CER}(A)$:

由 r_1 可得

$$\text{CER}(E_1 \text{ AND } E_2) = \min \{\text{CER}(E_1), \text{CER}(E_2)\} = \min\{0.8, 0.6\} = 0.6$$

$$m(\{a_1\}, \{a_2\}) = \{0.6 \times 0.3, 0.6 \times 0.5\} = \{0.18, 0.3\}$$

$$\text{Bel}(A) = m(\{a_1\}) + m(\{a_2\}) = 0.18 + 0.3 = 0.48$$

$$\text{Pl}(A) = 1 - \text{Bel}(\neg A) = 1 - 0 = 1$$

$$f(A) = \text{Bel}(A) + \frac{|A|}{|\Omega|} \cdot [\text{Pl}(A) - \text{Bel}(A)] = 0.48 + \frac{2}{10} \times 0.52 = 0.58$$

故 $\text{CER}(A) = \text{MD}(A | E') \times f(A) = 0.58$

2. 求 $\text{CER}(H)$:

由 r_2 可得

$$m_1(\{h_1\}, \{h_2\}) = \{\text{CER}(E_3) \times 0.4, \text{CER}(E_3) \times 0.2\} = \{0.9 \times 0.4, 0.9 \times 0.2\} = \{0.36, 0.18\}$$

$$m_1(\Omega) = 1 - [m_1(\{h_1\}) + m_1(\{h_2\})] = 1 - (0.36 + 0.18) = 0.46$$

再由 r_3 可得

$$m_2(\{h_1\}, \{h_2\}) = \{\text{CER}(A) \times 0.1, \text{CER}(A) \times 0.5\} = \{0.58 \times 0.1, 0.58 \times 0.5\} = \{0.06, 0.29\}$$

$$m_2(\Omega) = 1 - [m_2(\{h_1\}) + m_2(\{h_2\})] = 1 - (0.06 + 0.29) = 0.65$$

求正交和 $m = m_1 \oplus m_2$:

$$K = m_1(\Omega) \times m_2(\Omega) + m_1(\{h_1\}) \times m_2(\{h_1\}) + m_1(\{h_1\}) \times m_2(\Omega) + m_1(\Omega) \times m_2(\{h_1\}) \\ + m_1(\{h_2\}) \times m_2(\{h_2\}) + m_1(\{h_2\}) \times m_2(\Omega) + m_1(\Omega) \times m_2(\{h_2\}) = 0.88$$

$$m(h_1) = \frac{1}{K} \times [m_1(\{h_1\}) \times m_2(\{h_1\}) + m_1(\{h_1\}) \times m_2(\Omega) + m_1(\Omega) \times m_2(\{h_1\})] = 0.32$$

$$m(h_2) = \frac{1}{K} \times [m_1(\{h_2\}) \times m_2(\{h_2\}) + m_1(\{h_2\}) \times m_2(\Omega) + m_1(\Omega) \times m_2(\{h_2\})] = 0.34$$

$$m(\Omega) = 1 - [m(h_1) + m(h_2)] = 1 - 0.66 = 0.34$$

再根据 m 可得

$$\text{Bel}(H) = m(\{h_1\}) + m(\{h_2\}) = 0.32 + 0.34 = 0.66$$

$$\text{Pl}(H) = m(\Omega) + \text{Bel}(H) = 0.34 + 0.66 = 1$$

$$f(H) = \text{Bel}(H) + \frac{|H|}{|\Omega|} \times [\text{Pl}(H) - \text{Bel}(H)] = 0.66 + \frac{2}{10} \times (1 - 0.66) = 0.73$$

$$\text{CER}(H) = \text{MD}(H/E') \times f(H) = 0.73$$

5.5 模糊推理

5.5.1 模糊集及其运算

论域: 所讨论的全体对象, 用 U 等表示。

元素: 论域中的每个对象, 常用 a, b, c, x, y, z 表示。

集合: 论域中具有某种相同属性的确定的、可以彼此区别的元素的全体, 常用 A, B 等表示。

元素 a 和集合 A 的关系: a 属于 A 或 a 不属于 A , 即只有两个真值 True 和 False。

模糊集定义:

设 U 是给定论域, $\mu_F(u)$ 是把任意 $u \in U$ 映射为 $[0, 1]$ 上某个实值的函数:

$$\begin{aligned} \mu_F(u) : U &\rightarrow [0, 1] \\ u &\rightarrow \mu(u) \end{aligned}$$

则称 $\mu_F(u)$ 为定义在 U 上的一个隶属函数, 由 $\mu_F(u)$ 对所有 $u \in U$ 所构成的集合 $F = \{\mu_F(u) \mid u \in U\}$, 则称为 U 上的一个模糊集, $\mu_F(u)$ 称为 u 对 F 的隶属度。

当 $\mu_F(u)$ 仅取 0 和 1 两个值时, 模糊集 F 便退化为一个普通集合。

一个非空论域可以对应多个不同的模糊集, 一个空的论域只能对应一个空的模糊集。

一个模糊集与其隶属函数之间是一一对应关系

常见的隶属函数: 正态分布、三角分布、梯形分布等。

隶属函数确定方法: 模糊统计法、专家经验法、二元对比排序法、基本概念扩充法等

2 模糊集的表示:

2.1 向量表示法:

对离散且有限论域 $U = \{u_1, u_2, \dots, u_n\}$, 其模糊集可表示为
 $F = \{\mu_F(u_1), \mu_F(u_2), \dots, \mu_F(u_n)\}$ 。

为了表示论域中元素与其隶属度之间的对应关系, 扎德 Zadeh 引入了一种模糊集的表示方式, 为论域中的每个元素都标上其隶属度, 再用 + 把它们连接起来, 即

$$F = \mu_F(u_1)/u_1 + \mu_F(u_2)/u_2 + \dots + \mu_F(u_n)/u_n$$

也可写成

$$F = \sum_{i=1}^n \mu_F(u_i)/u_i$$

式中, $\mu_F(u_i)$ 为 u_i 对 F 的隶属度

$\mu_F(u_i)/u_i$ 不是相除关系, 只是一个记号; + 也不是算术意义上的加, 只是一个连接符号。

在这种表示方法中, 当某个 u_i 对 F 的隶属度 $\mu_F(u_i) = 0$ 时, 可省略不写。

2.2 序偶表示法:

$$F = \{(\mu_F(u_1), u_1), (\mu_F(u_2), u_2), \dots, (\mu_F(u_n), u_n)\}$$

3 模糊集运算:

设 F 和 G 分别是 U 上的两个模糊集,

若对任意 $u \in U$, 都有 $\mu_F(u) = \mu_G(u)$ 成立, 则称 F 等于 G , 记为 $F = G$ 。

对任意 $u \in U$, 都有 $\mu_F(u) \leq \mu_G(u)$ 成立, 则称 F 含于 G , 记为 $F \subseteq G$ 。

则 $F \cup G$ 和 $F \cap G$ 分别称为 F 与 G 的并集、交集, 它们的隶属函数分别为

$$\begin{aligned} F \cup G : \mu_{F \cup G}(u) &= \max_{u \in U} \{\mu_F(u), \mu_G(u)\} \\ F \cap G : \mu_{F \cap G}(u) &= \min_{u \in U} \{\mu_F(u), \mu_G(u)\} \end{aligned}$$

为叙述简便, 模糊集合论中通常用 \vee 代表 \max , \wedge 代表 \min , 即

$$\begin{aligned} F \cup G : \mu_{F \cup G}(u) &= \mu_F(u) \vee \mu_G(u) \\ F \cap G : \mu_{F \cap G}(u) &= \mu_F(u) \wedge \mu_G(u) \end{aligned}$$

称 $\neg F$ 为 F 的补集, 其隶属函数为

$$\neg F : \mu_{\neg F}(u) = 1 - \mu_F(u)$$

设 $U = \{1, 2, 3\}$, F 和 G 分别是 U 上的两个模糊集, F 代表概念「小」, G 代表概念「大」, 且

$$\begin{aligned}
F &= 1/1 + 0.6/2 + 0.1/3 \\
G &= 0.1/1 + 0.6/2 + 1/3 \\
F \cup G &= (1 \vee 0.1)/1 + (0.6 \vee 0.6)/2 + (0.1 \vee 1)/3 = 1/1 + 0.6/2 + 1/3 \\
F \cap G &= (1 \wedge 0.1)/1 + (0.6 \wedge 0.6)/2 + (0.1 \wedge 1)/3 = 0.1/1 + 0.6/2 + 0.1/3 \\
\neg F &= (1 - 1)/1 + (1 - 0.6)/2 + (1 - 0.1)/3 = 0.4/2 + 0.9/3
\end{aligned}$$

可以看出，两个模糊集之间的运算实际上是逐点对隶属函数进行相应的运算。

 在 $U_1 \times U_2 \times \cdots \times U_n$ 上的一个 n 元模糊关系 R 是指以 $U_1 \times U_2 \times \cdots \times U_n$ 为论域的一个模糊集，记为

$$R = \int_{U_1 \times U_2 \times \cdots \times U_n} \mu_R(u_1, u_2, \dots, u_n) / (u_1, u_2, \dots, u_n)$$

在上面的两个定义中， $\mu_{F_i}(u_i)(i = 1, 2, \dots, n)$ 是模糊集 F_i 的隶属函数； $\mu_R(u_1, u_2, \dots, u_n)$ 是模糊关系 R 的隶属函数，把 $U_1 \times U_2 \times \cdots \times U_n$ 上的每个元素 (u_1, u_2, \dots, u_n) 都映射为 $[0, 1]$ 上的一个实数，该实数反映出 u_1, u_2, \dots, u_n 具有关系 R 的程度。

二元例子：

设有一组学生 $U = \{u_1, u_2\} = \{\text{秦学, 郝玩}\}$ 和一些在计算机上的活动 $V = \{v_1, v_2, v_3\} = \{\text{编程, 上网, 玩游戏}\}$

设每个学生对各种活动的爱好程度分别为 $\mu_R(u_i, v_j)$ ($i = 1, 2, j = 1, 2, 3$)，即

$$\mu_R(\text{秦学, 编程}) = 0.9, \mu_R(\text{秦学, 上网}) = 0.4, \mu_R(\text{秦学, 玩游戏}) = 0.1$$

$$\mu_R(\text{郝玩, 编程}) = 0.2, \mu_R(\text{郝玩, 上网}) = 0.5, \mu_R(\text{郝玩, 玩游戏}) = 0.8$$

则 $U \times V$ 上的模糊关系：

$$R = \begin{bmatrix} 0.9 & 0.4 & 0.1 \\ 0.2 & 0.5 & 0.8 \end{bmatrix}$$

此外， U 与 V 可以有相同的论域，即 $U = V$ ，从而 R 应该是 $U \times U$ 上的模糊关系。

 模糊关系的合成：

设 R_1 与 R_2 分别是 $U \times V$ 与 $V \times W$ 上的两个模糊关系，则 R_1 与 R_2 的合成是从 U 到 W 的一个模糊关系，记为 $R_1 \circ R_2$ ，其隶属函数为

$$\mu_{R_1 \circ R_2}(u, w) = \vee \{\mu_{R_1}(u, v) \wedge \mu_{R_2}(v, w)\}$$

式中， \wedge 和 \vee 分别表示取最小和取最大。

设有以下两个模糊关系

$$R_1 = \begin{bmatrix} 0.4 & 0.5 & 0.6 \\ 0.8 & 0.3 & 0.7 \end{bmatrix} \quad R_2 = \begin{bmatrix} 0.7 & 0.9 \\ 0.2 & 0.8 \\ 0.5 & 0.3 \end{bmatrix}$$

则 R_1 与 R_2 的合成

$$R = R_1 \circ R_2 = \begin{bmatrix} 0.5 & 0.5 \\ 0.7 & 0.8 \end{bmatrix}$$

把 R_1 的第 i 行元素分别与 R_2 的第 j 列的对应元素相比较, 两两取最小, 再在所得的一组数中取最大的一个, 并以此数作为 $R_1 \circ R_2$ 的元素 $R(i, j)$ 。例如:

$$R(1, 1) = (0.4 \wedge 0.7) \vee (0.5 \wedge 0.2) \vee (0.6 \wedge 0.5) = 0.4 \vee 0.2 \vee 0.5 = 0.5$$

5.5.2 模糊知识表示

2 模糊谓词:

设 x 为在 U 中取值的变量, F 为模糊谓词, 即 U 中的一个模糊关系, 则命题可表示为 x is F

模糊谓词可以是大、小、年轻、年老、冷、暖、长、短等。

2 模糊量词:

模糊逻辑中使用了大量的模糊量词, 如极少、很少、几个、少数、多数、大多数、几乎所有等。

这些模糊量词 F 可以使我们方便地描述类似于下面的命题:

大多数成绩好的学生学习都很刻苦。

很少有成绩好的学生特别贪玩。

2 模糊修饰语 (程度词):

设 m 是模糊修饰语, x 是变量, F 为模糊谓词, 则模糊命题可表示为 x is mF

常用的程度词有「很」, 「非常」, 「有些」, 「绝对」等。

2 模糊知识的表示方式:

在扎德的推理模型中, 产生式规则的表示形式是 IF x is F THEN y is G

x 和 y 是变量, 表示对象; F 和 G 分别是论域 U 及 V 的模糊集, 表示概念。

条件部分可以是多个 x_i is F_i 的逻辑组合, 此时诸隶属函数间的运算按模糊集的运算进行。

模糊推理中所用的证据是用模糊命题表示的, 其一般形式为 x is F' , F' 是论域 U 上的模糊集

5.5.3 模糊概念匹配

2 语义距离:

语义距离用于刻画两个模糊概念之间的差异

设 A 与 B 分别是论域 $U = \{x_1, x_2, \dots, x_n\}$ 上的表示两个模糊概念的模糊集

它们之间的海明距离定义为:

$$d_h(A, B) = \frac{1}{n} \times \sum_{i=1}^n |\mu_A(x_i) - \mu_B(x_i)|$$

★ 欧几里德距离定义为:

$$d_o(A, B) = \frac{1}{\sqrt{n}} \times \sqrt{\sum_{i=1}^n (\mu_A(x_i) - \mu_B(x_i))^2}$$

○ 贴近度:

贴近度是指两个概念的接近程度, 可直接用来作为匹配度。设 F 和 G 分别是论域 $U = \{u_1, u_2, \dots, u_n\}$ 上的两个模糊概念的模糊集, 则它们的贴近度定义为:

$$(F, G) = \frac{1}{2}(F \cdot G + (1 - F \odot G))$$

式中

$$\begin{aligned} F \cdot G &= \bigvee_U (\mu_F(u_i) \wedge \mu_G(u_i)) \\ F \odot G &= \bigwedge_U (\mu_F(u_i) \vee \mu_G(u_i)) \end{aligned}$$

称 $F \cdot G$ 为 F 与 G 的内积, $F \odot G$ 为 F 与 G 的外积。

设论域 U 及其上的模糊集 F 和 G 如例 3.15 所示, 求 F 和 G 的贴近度。

解:

$$\begin{aligned} F \cdot G &= 0.8 \wedge 0.9 \vee 0.5 \wedge 0.6 \vee 0.1 \wedge 0.2 \vee 0 \wedge 0 \vee 0 \wedge 0 = 0.8 \vee 0.5 \vee 0.10 \vee 0 \vee 0 = 0.8 \\ F \odot G &= (0.8 \vee 0.9) \wedge (0.5 \vee 0.6) \wedge (0.1 \vee 0.2) \wedge (0 \vee 0) \wedge (0 \vee 0) \\ &= 0.9 \wedge 0.6 \wedge 0.2 \wedge 0 \wedge 0 = 0 \\ (F, G) &= 0.5 \times (0.8 + (1 - 0)) = 0.5 \times 1.8 = 0.9 \end{aligned}$$

5.5.4 模糊推理方法

○ 模糊关系的构造:

设 F 和 G 分别是论域 U 和 V 上的两个模糊集。

★ 扎德模糊关系 R_m :

$$R_m = \int_{U \times V} (\mu_F(u) \wedge \mu_G(v)) \vee (1 - \mu_F(u)) / (u, v)$$

★ 麦姆德尼 (Mamdani) 模糊关系 R_c :

$$R_c = \int_{U \times V} (\mu_F(u) \wedge \mu_G(v)) / (u, v)$$

★ 米祖莫托 (Mizumoto) 模糊关系 R_g :

$$R_g = \int_{U \times V} (\mu_F(u) \rightarrow \mu_G(v)) / (u, v)$$

$$\mu_F(u) \rightarrow \mu_G(v) = \begin{cases} 1 & \mu_F(u) \leq \mu_G(v) \\ \mu_G(v) & \mu_F(u) > \mu_G(v) \end{cases}$$

ℳ 模糊推理基本模式:

设 F 和 G 分别是 U 和 V 上的两个模糊集, 且有知识 IF x is F THEN y is G 一个模糊集。三种模式表示形式:

模糊假言推理

$$\text{规则: } \text{IF } x \text{ is } F, \text{ THEN } y \text{ is } G$$

$$\text{证据: } \underline{x \text{ is } F},$$

$$\text{结论: } \underline{\underline{y \text{ is } G'}}$$

模糊拒取式推理

$$\text{规则: } \text{IF } x \text{ is } F, \text{ THEN } y \text{ is } G$$

$$\text{事实: } \underline{\underline{y \text{ is } G'}}$$

$$\text{结论: } \underline{x \text{ is } F'}$$

模糊假言三段式

$$\text{规则 1: } \text{IF } x \text{ is } F, \text{ THEN } y \text{ is } G$$

$$\text{规则 2: } \text{IF } y \text{ is } G, \text{ THEN } z \text{ is } H$$

$$\text{结论 (规则 3): } \text{IF } x \text{ is } F, \text{ THEN } z \text{ is } G$$

ℳ 模糊假言推理:

ℳ 若有 U 上的一个模糊集 F' , 且 F 可以和 F' 匹配, 则可以推出 y is G' , 且 G' 是 V 上的一个模糊集。这种推理模式称为模糊假言推理。

ℳ 表示形式为:

ℳ 知识: IF x is F THEN y is G

ℳ 证据: x is F'

ℳ 结论: y is G'

ℳ 在这种推理模式下, 模糊知识 IF x is F THEN y is G 表示在 F 与 G 之间存在着确定的模糊关系, 设此模糊关系为 R 。那么, 当已知的模糊事实 F' 可以与 F 匹配时, 则可通过 F' 与 R 的合成得到 G' , 即

$$G' = F' \circ R$$

式中, 模糊关系 R 可以是 R_m 、 R_c 或 R_g 中的任何一种。

设 $U = \{1, 2, 3\}$, F 和 G 分别是 U 上的两个模糊集, 即

$$F = \text{小} = 1/1 + 0.6/2 + 0.1/3$$

$$G = \text{大} = 0.1/1 + 0.6/2 + 1/3$$

并设「较小」的模糊集为: $F' = \text{较小} = 1/1 + 0.7/2 + 0.2/3$, 求在此证据下的模糊结论 G'

$$R_c = \begin{bmatrix} 1 \\ 0.6 \\ 0.1 \end{bmatrix} \cdot [0.1 \quad 0.6 \quad 1] = \begin{bmatrix} 0.1 & 0.6 & 1 \\ 0.1 & 0.6 & 0.6 \\ 0.1 & 0.1 & 0.1 \end{bmatrix}$$
$$G' = F' \circ R_c = [1 \quad 0.7 \quad 0.2] \circ \begin{bmatrix} 0.1 & 0.6 & 1 \\ 0.1 & 0.6 & 0.6 \\ 0.1 & 0.1 & 0.1 \end{bmatrix} = [0.1 \quad 0.6 \quad 1]$$

💡 模糊拒取式推理:

📌 若有 V 上的一个模糊集 G' , 且 G' 可以与 G 的补集 $\neg G$ 匹配, 则可以推出 x is F' , 且 F' 是 U 上的一个模糊集。这种推理模式称为模糊拒取式推理

📌 表示为:

📎 知识: IF x is F THEN y is G

📎 证据: y is G'

📎 结论: x is F'

📌 模糊知识 IF x is F THEN y is G 也表示在 F 与 G 之间存在着确定的模糊关系, 设此模糊关系为 R 。当已知的模糊事实 G' 可以与 $\neg G$ 匹配时, 则可通过 R 与 G' 的合成得到 F' , 即

$$F' = R \circ G'$$

同上例, 已知事实为 y is 较大, 且模糊概念「较大」的模糊集

$$G' = 0.2/1 + 0.7/2 + 1/3$$

若 G' 与 $\neg G$ 匹配, 以模糊关系 R_c 为例, 推出 F' 。

解: 本例的模糊关系 R_c 已在前面求出, 通过 R_c 与 G' 的合成即可得到所求的 F' 。

$$F' = R_c \circ G' = \begin{bmatrix} 0.1 & 0.6 & 1 \\ 0.1 & 0.6 & 0.6 \\ 0.1 & 0.1 & 0.1 \end{bmatrix} \circ \begin{bmatrix} 0.2 \\ 0.7 \\ 1 \end{bmatrix} = \begin{bmatrix} 1 \\ 0.6 \\ 0.1 \end{bmatrix}$$

即求出的 F' 为

$$F' = 1/1 + 0.6/2 + 0.1/3$$

模糊假言三段式:

- 设 F, G, H 分别是 U, V, W 上的三个模糊集, 且由知识 IF x is F THEN y is G 和 IF y is G THEN z is H , 可推出 IF x is F THEN z is H , 这种推理模式称为模糊假言三段论推理,
- 在这种推理模式下, 模糊知识 r_1 : IF x is F THEN y is G 表示在 F 与 G 之间存在着确定的模糊关系, 设此模糊关系为 R_1 。
- 模糊知识 r_2 : IF y is G THEN z is H 表示在 G 与 H 之间存在着确定的模糊关系设此模糊关系为 R_2 。
- 若模糊假言三段论成立, 则 r_3 的模糊关系 R_3 可由 R_1 与 R_2 的合成得到, 即

$$R_3 = R_1 \circ R_2$$

设:

$$U = W = V = \{1, 2, 3\}$$

$$E = 1/1 + 0.6/2 + 0.2/3$$

$$F = 0.8/1 + 0.5/2 + 0.1/3$$

$$G = 0.2/1 + 0.6/2 + 1/3$$

按 R_g 求 $E \times F \times G$ 上的关系 R 。

解: 先求 $E \times F$ 上的关系 R_{g_1} :

$$R_{g_1} = \begin{bmatrix} 0.8 & 0.5 & 0.1 \\ 1 & 0.5 & 0.1 \\ 1 & 1 & 0.1 \end{bmatrix}$$

再求 $F \times G$ 上的关系 R_{g_2} :

$$R_{g_2} = \begin{bmatrix} 0.2 & 0.6 & 1 \\ 0.2 & 1 & 1 \\ 1 & 1 & 1 \end{bmatrix}$$

最后求 $E \times F \times G$ 上的关系:

$$R = R_{g_1} \circ R_{g_2} = \begin{bmatrix} 0.2 & 0.6 & 0.8 \\ 0.2 & 0.6 & 1 \\ 0.2 & 1 & 1 \end{bmatrix}$$

5.6 Bayesian 网络

5.6.1 概念

三种概率:

联合概率: $P(a, b)$

边缘概率: $P(a) = \sum_b P(a, b)$

条件概率: $P(a | b) = \frac{P(a, b)}{P(b)}$

贝叶斯网络（信念网络或概率网络）是概率论与图论的结合，其拓扑结构是一个有向无环图，图中的节点表示问题求解中的命题或随机变量，节点间的有向边表示条件依赖关系，这些依赖关系可用条件概率来描述。

定义：

设 $X = \{X_1, X_2, \dots, X_n\}$ 是任何随机变量集，其上的贝叶斯网络可定义为 $BN = (B_S, B_p)$

B_S 是贝叶斯网络的结构，即一个定义在 X 上的有向无环图：

每个节点 X_i 都唯一地对应着 X 中的一个随机变量，并需要标注定量的概率信息

每条有向边都表示它所连接的两个节点之间的条件依赖关系

若存在一条从节点 X_j 到节点 X_i 的有向边，则称 X_j 是 X_i 的父节点， X_i 是 X_j 的子节点

B_p 为贝叶斯网络的条件概率集合， $B_p = \{P(X_i | \text{par}(X_i))\}$ ：

$\text{par}(X_i)$ 表示 X_i 的所有父节点的相应取值

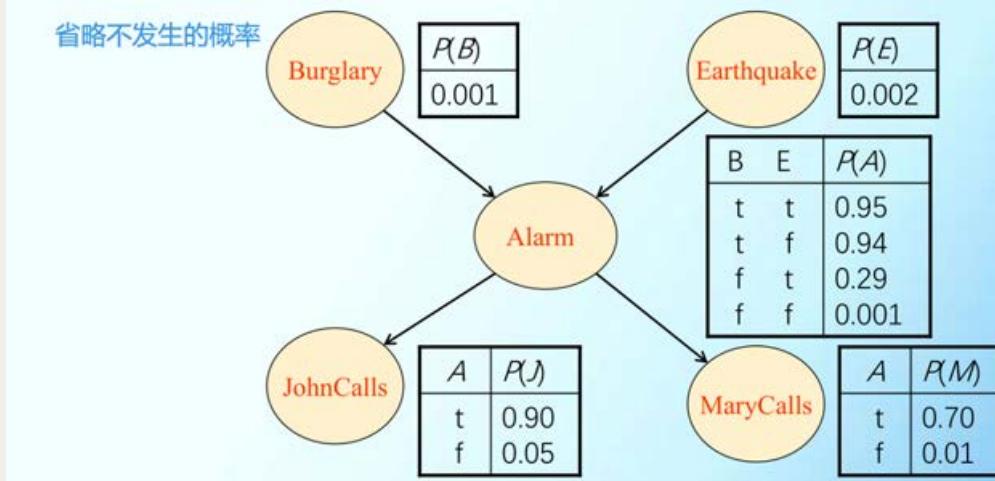
$P(X_i | \text{par}(X_i))$ 是节点 X_i 的一个条件概率分布函数，描述 X_i 的每个父节点对 X_i 的影响，即节点 X_i 的条件概率表

贝叶斯网络中的弧是有方向的，且不能形成回路，因此图有始点和终点。

在始点上有一个初始概率，在每条弧所连接的节点上有一个条件概率。

下图例子省略了每个节点不发生的概率：

例 假设“出现盗贼”和“地震”会“启动报警”，而“启动报警”之后，John 和 Mary 会给房屋主任打电话。可以用下图所示的贝叶斯网络来描述。



全联合概率分布（全联合概率或联合概率分布），是概率的一种合取形式：

设 $X = \{X_1, X_2, \dots, X_n\}$ 为任何随机变量集，其全联合概率分布是指当对每个变量取特定值时 $x_i (i = 1, 2, \dots, n)$ 的合取概率，即

$$P(X_1 = x_1 \wedge X_2 = x_2 \wedge \dots \wedge X_n = x_n)$$

其简化表示形式为 $P(x_1, x_2, \dots, x_n)$ 。

由全联合概率分布，再重复使用乘法法则

$P(x_1, x_2, \dots, x_n) = P(x_n | x_{n-1}, x_{n-2}, \dots, x_1)P(x_{n-1}, x_{n-2}, \dots, x_1)$, 可以把每个合取概率简化为更小的条件概率和更小的合取式，直至得到如下全联合概率分布表示：

$$P(x_1, x_2, \dots, x_n) = \prod_{i=1}^n P(x_i | x_{i-1}, x_{n-2}, \dots, x_1) = \prod_{i=1}^n \{P(x_i | \text{par}(x_i))\}$$

前半段恒等式对任何随机变量都是成立的，亦称为链式法则。

后半段公式就是贝叶斯网络的联合概率分布表示，比全联合概率分布简单得多，其计算复杂度小

上例中：

报警器响了，但既没有盗贼间入，也没有发生地震，同时 John 和 Mary 都给你打电话的概率：

$$\begin{aligned} & P(j, m, a, \neg b, \neg e) \\ &= P(j | a)P(m | a)P(a | \neg b, \neg e)P(\neg b)P(\neg e) \\ &= 0.9 \times 0.7 \times 0.001 \times 0.999 \times 0.998 \\ &= 0.00062 = 0.062\% \end{aligned}$$

贝叶斯网络的条件独立关系（书上说等价）：

给定父节点，一个节点与它的非后代节点是条件独立的

给定一个节点的父节点、子节点以及子节点的父节点一起构成了马尔科夫覆盖（Markov blanket），该节点和网络中的所有其它节点是条件独立的

贝叶斯网络的构造：

贝叶斯网络的联合概率分布表示同时给出了贝叶斯网络的构造方法，其主要依据是随机变量之间的条件依赖关系，即要确保满足联合概率分布。贝叶斯网络的构造过程如下：

- 建立不依赖于其他节点的根节点，并且根节点可以不止一个。
- 加入受根节点影响的节点，并将这些节点作为根节点的子节点。此时，根节点已成为父节点。
- 进一步建立依赖于已建立节点的子节点。重复这一过程，直到叶节点为止。
- 对每个根节点，给出其先验概率；对每个中间节点和叶节点，给出其条件概率表。

贝叶斯网络构造过程应遵循的主要原则如下：

- 忽略过于微弱的依赖关系。对于两个节点之间的依赖关系，是不是一定要在语义网络中用相应的有向边将其表示出来，取决于计算精度要求与计算代价之间的权衡。
- 随机变量之间的因果关系是最常见、最直观的依赖关系，可用来指导贝叶斯网络的构建过程

5.6.2 推理

用 X 表示某查询变量, E 表示证据变量集 $\{E_1, E_2, \dots, E_n\}$, s 表示一个观察到的特定事件, Y 表示非证据变量(隐含变量)集 $\{y_1, y_2, \dots, y_m\}$, 则全部变量的集合 $V = \{X\} \cup E \cup Y$, 其推理就是要查询后验概率 $P(X | s)$

$$P(X | s) = \frac{P(X, s)}{P(s)} = \frac{\sum_y P(X, y, s)}{\sum_x P(x, s)} = \frac{\sum_y P(X, y, s)}{\sum_{x,y} P(x, y, s)}$$

贝叶斯网络推理的一般步骤是:

1. 确定各相邻节点之间的初始条件概率分布
2. 对各证据节点取值
3. 选择适当推理算法对各节点的条件概率分布进行更新
4. 得到推理结果。

贝叶斯网络推理算法可根据对查询变量后验概率计算的精确度, 分为精确推理和近似推理两大类。近似推理算法是在不影响推理正确性的前提下, 通过适当降低推理精确度来提高推理效率的一类方法。常用的近似推理算法主要有马尔科夫链蒙特卡洛 (Markov Chain Monte Carlo, MCMC) 算法等。

精确推理:

主要方法包括基于枚举的算法、基于变量消元的算法和基于团树传播的算法等。

最基本的方法是基于枚举的算法, 使用全联合概率分布去推断查询变量的后验概率:

$$P(X | s) = \alpha P(X, s) = \alpha \sum_Y P(X, s, Y)$$

X 表示查询变量, s 表示一个观察到的特定事件, Y 表示隐含变量集 $\{y_1, y_2, \dots, y_m\}$, α 是归一化常数, 用于保证相对于 X 所有取值的后验概率总和等于 1。

为了对贝叶斯网络进行推理, 可利用贝叶斯网络的概率分布公式

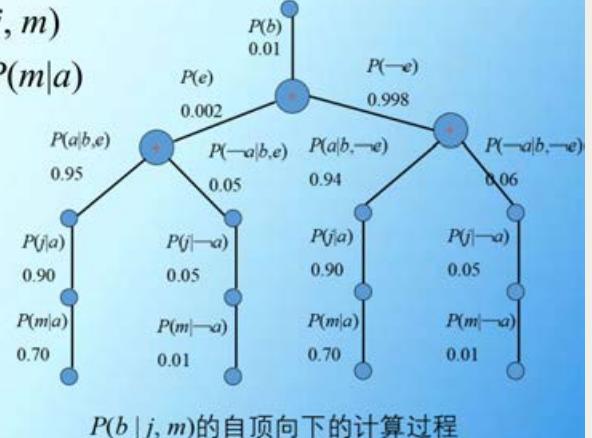
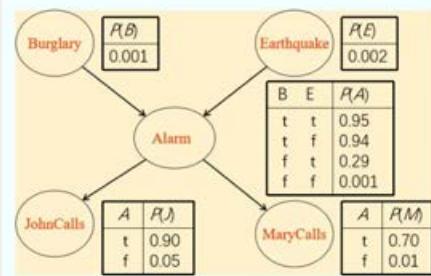
$$P(x_1, x_2, \dots, x_n) = \prod_{i=1}^n P(X_i | \text{par}(X_i))$$

将上式中的 $P(X, s, Y)$ 改写为条件概率乘积的形式。这样就可通过先对 Y 的各枚举值求其条件概率乘积, 再对各条件概率乘积求总和的方式去计算查询变量的条件概率。

例 已知，一个事件 $e = \{JohnCalls = true, and MaryCalls = true\}$ ，试问出现盗贼的概率是多少？

解：

$$\begin{aligned} P(b | j, m) &= \alpha P(b, j, m) = \alpha \sum_e \sum_a P(b, e, a, j, m) \\ &= \alpha \sum_e \sum_a P(b)P(e)P(a|b,e)P(j|a)P(m|a) \\ &= \alpha \times 0.00059224 \end{aligned}$$



$$\begin{aligned} P(\neg b | j, m) &= \alpha P(\neg b, j, m) = \alpha \sum_e \sum_a P(\neg b, e, a, j, m) = \alpha \sum_e \sum_a P(\neg b)P(e)P(a | \neg b, e)P(j | a)P(m | a) \\ &= \alpha \times 0.0014919 \end{aligned}$$

因此, $P(B | j, m) = \alpha < 0.00059224, 0.0014919 > \approx < 0.284, 0.716 >$

即在 John 和 Mary 都打电话的条件下，出现盗贼的概率约为 28%

无需知道 $P(j, m)$ 的准确值也能求出 $P(b | j, m)$

可以使用变量消元优化：

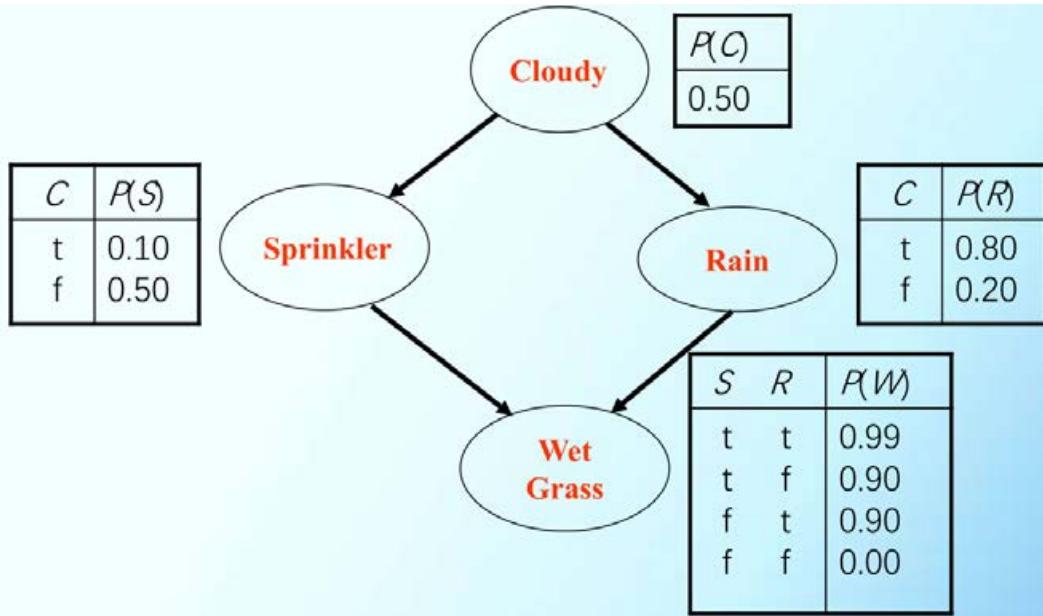
- ✓ 消除重复计算，提高枚举算法的效率
- ✓ 保存中间结果，以备多次使用
- ✓ 从右到左（在树结构中为自底向上）的次序计算 BN 的计算公式

2 近似推理（马尔科夫链蒙特卡洛（MCMC）方法）：

思想：

- ✓ 后验概率计算的主要采样方法对前一个事件进行随机改变而生成事件样本
- ✓ BN 为每个变量指定了一个特定的当前状态
- ✓ 下一个状态是通过对某个非证据变量 X_i 进行采样来产生，取决于 X_i 的马尔可夫覆盖中的变量当前值
- ✓ MCMC 方法可视为：在状态空间中（所有可能的完整赋值空间）的随机走动（每次改变一个变量），但是证据变量的值固定不变

2 执行过程：



【要求】：查询 $P(\text{Rain} | \text{Sprinkler} = \text{true}, \text{WetGrass} = \text{true})$ 的概率

- ↗ 证据变量 Sprinkler, WetGrass 固定为 true
- ↗ 隐变量 Cloudy 和查询变量 Rain 随机初始化, 例如, Cloudy = true, Rain = false
- ↗ 初始状态为: [C = true, S= true, R= false, W= true]
- ↗ 反复执行如下步骤 (反复遍历所有的隐变量和查询变量) :

1. 根据 Cloudy 的马尔科夫覆盖 (MB) 变量 (Sprinkler 和 Rain) 的当前值, 对 Cloudy 采样, 即根据 $P(\text{Cloudy} | \text{Sprinkler} = \text{true}, \text{Rain} = \text{false})$ (即转移概率) 来采样。即:

$$P(C | S, \neg R) = \frac{P(C, S, \neg R)}{P(S, \neg R)} = \frac{P(C)P(S | C)P(\neg R | C)}{P(C)P(S | C)P(\neg R | C) + P(\neg C)P(S | \neg C)P(\neg R | \neg C)}$$

由计算机生成一个随机数 $q \in [0, 1]$ 。判别方法如下:

if $q < P(C | S, \neg R)$ then 转移到下一个新状态;
otherwise 停留在原状态

对于本例子, 假设生成的随机数 $q = 0.0389$, 可知, 转移概率 $P(\text{Cloudy} | \text{Sprinkler} = \text{true}, \text{Rain} = \text{false}) = 0.04762 > q = 0.0389$ 。

所以, Cloudy 由 true 状态转移到新状态 false, 即采样结果为: Cloudy = false。故新的当前状态为: [C=false, S = true, R = false, W = true]

2. 根据 Rain 节点的马尔可夫覆盖变量 (Cloudy、Sprinkler 和 WetGrass) 的当前值, 对 Rain 采样, 即根据 $P(\text{Rain} | \text{Cloudy} = \text{false}, \text{Sprinkler} = \text{true}, \text{WetGrass} = \text{true})$ 来采样。

假设采样结果为: Rain = true。故新的当前状态为: [C=false, S=true, R=true, W=true]

上述过程中所访问的每一个状态都是一个样本, 能对查询变量 Rain 的估计有贡献。

重复上述步骤，直到所要求的访问次数 N 。若 true , false 的次数分别为 n_1, n_2 , 则查询解为:

$$\text{Normalize}(\langle n_1, n_2 \rangle) = \langle n_1/N, n_2/N \rangle$$

若上述过程访问了 20 个 $\text{Rain}=\text{true}$ 的状态和 60 个 $\text{Rain} = \text{false}$ 的状态, 则所求查询的解为 $\langle 0.25, 0.75 \rangle$

每次采样过程分为两步:

1. 依据该隐变量的马尔科夫覆盖所包含的变量的当前值, 计算该状态转移概率
(该隐变量取值改变的概率)
2. 通过生成随机数比较, 确定状态 (该隐变量的取值) 是否需要改变。

6 机器学习

6.1 概述

学习的一般性解释:

一个有特定目的的知识获取和能力增长过程, 其内在行为是获得知识、积累经验、发现规律等, 其外部表现是改进性能、适应环境、实现自我完善等。

机器学习的一般性解释:

让机器 (计算机) 来模拟和实现人类的学习功能。简单来讲, 就是让机器 (计算机) 从数据中学习知识 (即自动获取知识)

分类:

按学习策略分为:

记忆学习、传授学习、演绎学习、归纳学习 (又可分为示例学习、观察发现学习) 等。

按学习方式分为:

有导师学习 (监督学习) 、无导师学习 (非监督学习) 和强化学习 (增强学习)

监督学习的三要素:

标注数据: 标识了类别信息的数据

学习模型: 如何学习得到映射模型

损失函数: 如何对学习结果进行度量

监督学习方法分类:

生成 (式) 方法(generative):

所学到的模型分别称为生成 (式) 模型

生成模型从数据中学习联合概率分布 $P(X, Y)$ (通过似然概率 $P(X | Y)$ 和类概率

$$P(Y) \text{ 的乘积来求取) } P(Y | X) = \frac{P(X, Y)}{P(X)} = \frac{P(X | Y) \times P(Y)}{P(X)}$$

典型方法为贝叶斯方法、隐马尔可夫链

联合分布概率 $P(X, Y)$ 或似然概率 $P(X | Y)$ 求取很困难

判别(式)方法(discriminative):

直接学习判别函数 $f(X)$ 或者条件概率分布 $P(Y|X)$ 作为预测的模型，即判别模型。

判别模型关心在给定输入数据下，预测该数据的输出是什么。

典型判别模型包括回归模型、决策模型、神经网络、支持向量机和集成模型等。

损失函数：

损失函数就是用来计算 x_i 真实值 y_i 与预测值 $f(x_i)$ 之间差值的函数。

在训练过程中希望映射函数在训练数据集上得到「损失」之和最小，即

$$\min \sum_{i=1}^n \text{Loss}(f(x_i), y_i) .$$

损失函数名称	损失函数定义
0-1损失函数	$\text{Loss}(y_i, f(x_i)) = \begin{cases} 1, & f(x_i) \neq y_i \\ 0, & f(x_i) = y_i \end{cases}$
平方损失函数	$\text{Loss}(y_i, f(x_i)) = (y_i - f(x_i))^2$
绝对损失函数	$\text{Loss}(y_i, f(x_i)) = y_i - f(x_i) $
对数损失函数/对数似然损失函数	$\text{Loss}(y_i, P(y_i x_i)) = -\log P((y_i x_i))$

映射函数训练目标：

经验风险最小化 (empirical risk, minimization, ERM)

$$\min_{f \in \Phi} \frac{1}{n} \sum_{i=1}^n \text{Loss}(y_i, f(x_i))$$

期望风险最小化 (expected risk minimization)，测试数据集数据无穷多

$$\min_{f \in \Phi} \int_{x \times y} \text{Loss}(y, f(x)) P(x, y) dx dy$$

经验风险小（训练集上表现好）	期望风险小（测试集上表现好）	泛化能力强
经验风险小（训练集上表现好）	期望风险大（测试集上表现不好）	过拟合 过学习（模型过于复杂）
经验风险大（训练集上表现不好）	期望风险大（测试集上表现不好）	欠学习
经验风险大（训练集上表现不好）	期望风险小（测试集上表现好）	“神仙算法”或“黄粱美梦”

结构风险最小化 (structural risk minimization):

为了防止过拟合，在经验风险上加上表示模型复杂度的正则化项 (regulatizer) 或惩罚项 (penalty term):

$$\min_{f \in \Phi} \frac{1}{n} \sum_{i=1}^n \text{Loss}(y_i, f(x_i)) + \lambda J(f)$$

在最小化经验风险与降低模型复杂度之间寻找平衡

6.2 线性回归

在现实生活中，往往需要分析若干变量之间的关系，如碳排放量与气候变暖之间的关系、某一商品广告投入量与该商品销售量之间的关系等，这种分析不同变量之间存在关系的研究叫回归分析，刻画不同变量之间关系的模型被称为回归模型。如果这个模型是线性的，则称为线性回归 (Linear Regression) 模型。

回归模型: $y = ax + b$

训练集中 n 个样本所产生误差总和(即损失函数)为: $L(a, b) = \sum_{i=1}^n (y_i - a \times x_i - b)^2$

目标: 寻找一组 a 和 b , 使得误差总和 $L(a, b)$ 最小。在线性回归中, 解决此目标的方法叫最小二乘法。

要使函数具有最小值, 可对 $L(a, b)$ 参数 a 和 b 分别求导, 令其导数值为零, 再求取参数 a 和 b 的取值:

$$b = \bar{y} - a\bar{x} \quad a = \frac{\sum_{i=0}^n x_i y_i - n\bar{x}\bar{y}}{\sum_{i=1}^n x_i^2 - n\bar{x}^2}$$

6.3 决策树学习

决策树是一种由节点和边构成的用来描述分类过程的层次数据结构:

根节点: 表示分类的开始

叶节点: 表示一个实例的结束

中间节点: 表示相应实例中的某一属性

边: 某一属性可能的属性值

- 路径：从根节点到叶节点的每一条路径都代表一个具体的实例，并且同一路径上的所有属性之间为合取关系，不同路径（即一个属性的不同属性值）之间为析取关系。
- 决策树的分类过程：从树的根节点开始，按照给定的事例的属性值去测试对应的树枝，并依次下移，直至到达某个叶节点为止。
- 常用算法 ID3 (Iterative Dichotomous version3)、C4.5、C5.0 和随机森林
- ID3 算法的学习过程是一个以整个样本集为根节点，以信息增益最大为原则，选择条件属性进行扩展，逐步构造出决策树的过程。
- 若假设 $S = \{s_1, s_2, \dots, s_n\}$ 为整个样本集， $X = \{x_1, x_2, \dots, x_m\}$ 为全体属性（特征）集， $Y = \{y_1, y_2, \dots, y_k\}$ 为样本类别。则 ID3 算法描述如下：

 - 从根节点 (S, X) 开始，计算所有可能的属性（特征）的信息增益，选择信息增益最大的属性（特征）作为节点的划分特征

信息熵：对信息源整体不确定性的度量

$$\text{Entropy}(S) = H(S) = - \sum_{i=1}^k P_i \log P_i$$

其中 P_i 表示第 i 类样本在样本集 S 中所占比例

信息增益：两个信息量之间的差的度量，描述的是信息的确定性

$$\text{Gain}(S, A) = \text{Entropy}(S) - \sum_{i=1}^n \frac{|S_i|}{|S|} \text{Entropy}(S_i, a_i) = \text{Entropy}(S) - \text{Entropy}(S | A_i) =$$

S_i 为按属性值 a_i 分类后

- 由该属性（特征）的不同取值建立子节点
- 再对子节点递归 1-2 步，构建决策树
- 直到没有特征可以选择或类别完全相同为止，得到最终的决策树。

如果属性集为空，将该节点标记为叶节点

叶节点的类别根据各个类别的样本数量，按照少数服从多数的原则，是样本数最多的那个类别

6.4 支持向量机

- 支持向量机 (SVM) 以 VC 维理论为基础，利用最大间隔算法去近似地实现结构风险最小化原理，是一种基于统计学习理论的新型通用机器学习方法。该方法不仅可以很好地解决线性可分问题，而且可以利用核函数有效地解决线性不可分问题。
- 线性可分与最优分类超平面：
 - 线性可分问题分类是 SVM 学习方法的基础，SVM 是通过最优分类超平面来实现其分类的。

假设有以下 n 个独立、同分布且线性可分的训练样本 $S=\{(x_1, y_1), (x_2, y_2), \dots, (x_n, y_n)\}$, $Y=\{-1, +1\}$ 为样本类别。SVM学习的目标就是要找到一个最优超平面

$$w \cdot x + b = 0$$

将两类不同的样本完全分开。其中， w 是权重向量，“ \cdot ”是向量的点积， x 是输入向量， b 是一个阈值。

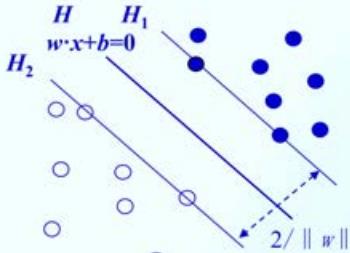


图5.11 线性可分的最优超平面

- (1) H 为分类超平面， H_1 和 H_2 分别为两个不同类的边界分割平面， H_1 和 H_2 均与 H 平行，且 H_1 和 H_2 分别通过相应类中离 H 最近的样本点。
- (2) 对 H ，若能满足 H 与两个类边界分割平面 H_1 和 H_2 等距，且能使 H_1 和 H_2 之间的间隔最大，则称该分类超平面为**最优分类超平面**。
- (3) 最优分类超平面仅与分布在 H_1 和 H_2 上的样本点有关，这些样本点被称为**支持向量**

非线性可分与核函数：

对非线性可分问题，SVM采用特征空间映射方式，即将非线性可分的样本集映射到高维空间，使其在高维空间中被转变为线性可分。即采用核函数的方法。

6.5 集成学习

集成学习是指为解决同一问题，先训练出一系列个体学习器（或称弱学习器），然后再根据某种规则把这些个体学习器的学习结果整合到一起，得到比单个个体学习器更好的学习效果。

包括两大基本问题：

- ／＼ 个体学习器的构造
- ／＼ 个体学习器的合成

集成学习的两种方式

／＼ 同质集成（狭义集成学习）：

构造个体学习器时使用相同类型的学习方法，个体学习器称为基学习器，所用的学习算法称为基学习算法。

／＼ 异质集成（广义集成学习）：

构造个体学习器时可以使用不同类型的学习方法

Boosting 方法：

- ／＼ 为每个训练样本平均分配初始权重，并训练出弱学习器 1
- ／＼ 通过提高错误率高的训练样本的权重，降低错误率低的训练样本的权重，得到新的权重分布，并在其上训练出弱学习器 2
- ／＼ 重复 1 和 2，直到到达最大迭代轮数
- ／＼ 将训练出来的这些弱学习器合成为一起，形成最终的强学习器。

Bagging 方法：

- ／＼ 从初始训练集中使用可重采样的随机抽样方法产生出本轮的训练子集，利用选定的弱学习算法训练出本轮迭代的弱学习器

- 重复 1，直到到达最大迭代轮数
- 按照某种方式将训练出来的这些弱学习器合成到一起，形成最终的强学习器。

弱学习器的合成方法：

代数合成法：

- 通过代数表达式对诸弱学习器进行合成，获得表达式最大支持的结果作为合成结果输出。
- 若假设 D 为训练集， X 为训练集的输入向量， Y 为训练集的输出向量
- T 为训练过程需要迭代的总轮数， $t = 1, 2, \dots, T$ ，为训练过程的当前迭代轮数
- J 为分类问题的最大类别个数，为弱学习器的分类结果
- $h_{t,j}(X)$ 是弱学习器 t 得出分类结果 j 的判断，若分类结果为 $j (j = 1, 2, \dots, J)$ ，则 $h_{t,j}(X) = 1$ ，否则 $h_{t,j}(X) = 0$ 。

平均法：

$$H_{\text{final}}(X) = \arg \max_j \frac{1}{T} \sum_{t=1}^T h_{t,j}(X)$$

加权平均法为

$$H_{\text{final}}(X) = \arg \max_j \frac{1}{T} \sum_{t=1}^T w_t h_{t,j}(X)$$

式中， w_t 为第 t 个弱学习器的权重。

投票法：

- 基本思想是对训练出来的所有弱分类器，按照某种投票原则进行投票表决。
- 常用的投票方法有相对多数投票、加权投票法等。
- 相对多数投票法就是少数服从多数，即在 T 个弱学习器中，选择对样本 X 预测结果中得票最多的弱学习器的学习结果作为强学习器的学习结果
- 加权投票法与加权平均法一样，需要对每个弱学习器的票数再乘以其自身的权重，再对加权票数求和。以简单的多数投票法为例，有

$$H(X) = \arg \max_{y \in Y} \sum_{t=1}^T I(h_t(X) = y)$$

其中，符号 I 为取真值运算。当弱学习器 $h_t(x)$ 的输出与训练样本中 x 对应的输出 y 相同时， $I(h_t(x) = y)$ 取 1，否则取 0

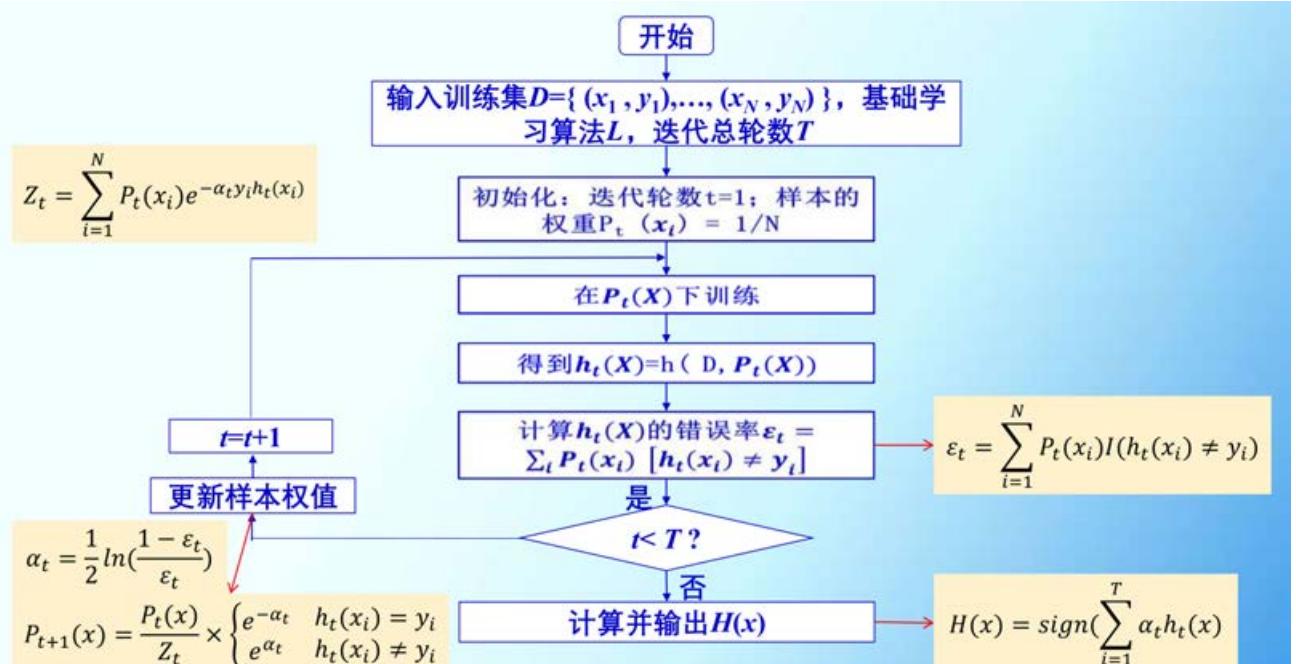
AdaBoost 算法：

符号和定义：

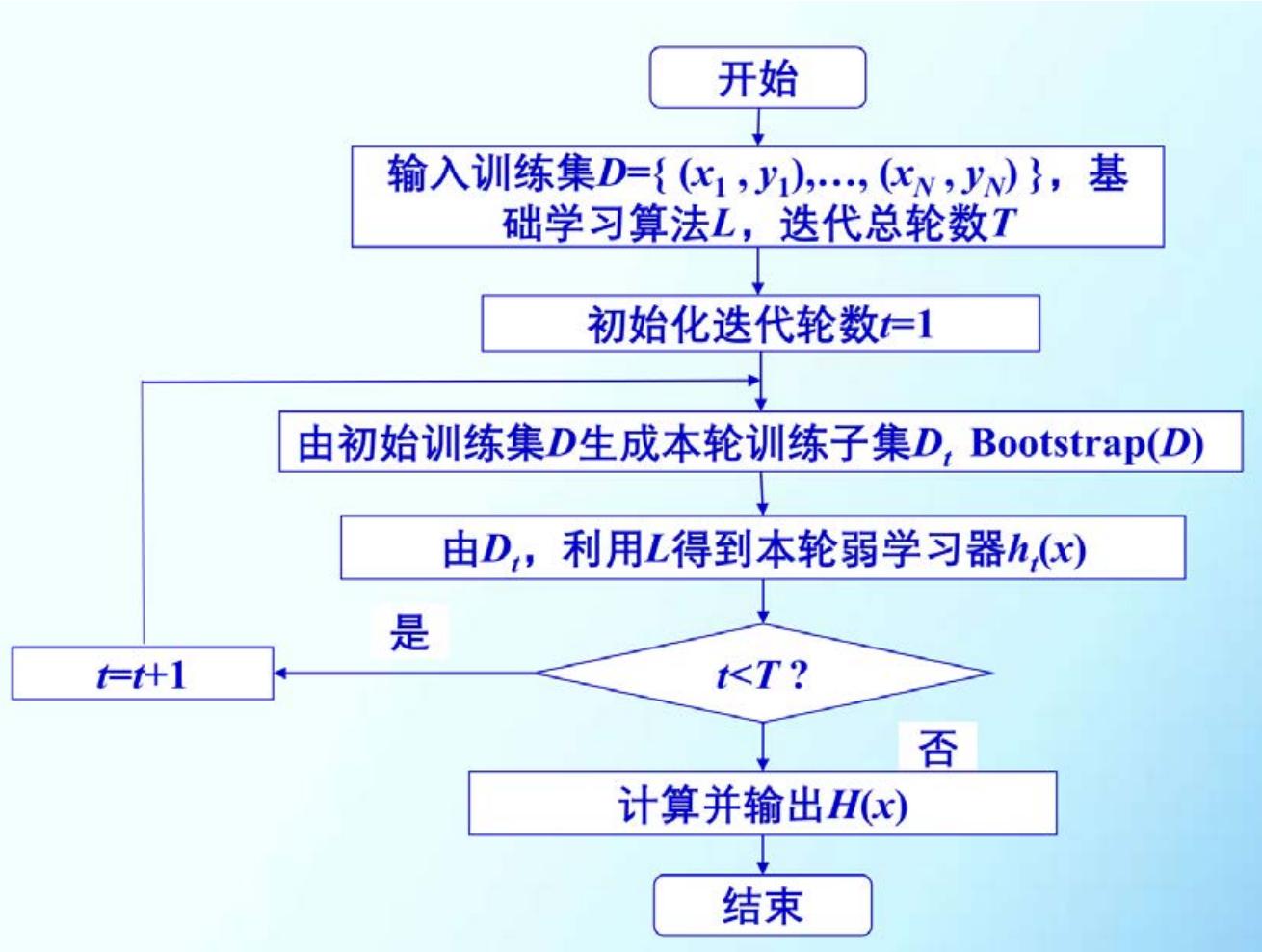
- $D = \{(x_1, y_1), (x_2, y_2), \dots, (x_N, y_N)\}$ 为训练样本集， L 为基础学习算法
- T 为训练过程需要迭代的总轮数
- t 为训练过程的当前迭代轮数， $t = 1, 2, \dots, T$

- $P_t(x)$ 为第 t 轮迭代过程中, 样本空间中第 i 个样本 $x_i (i = 1, 2, \dots, N)$ 被抽取用于训练第 t 个弱学习器的权重, 实际上为概率
- $h_t(x)$ 为第 t 次迭代所得到的弱学习器, $h_t(x) = h(D, P_t(x))$, $h_t(x) : x \rightarrow \{-1, +1\}$
- ε_t 为 $h_t(x)$ 在训练集上的分类误差率, 即 $h_t(x) \neq y_i$ 的概率
- α_t 为弱学习器 h_t 在集成过程中所具有的权重
- $Z_t = \sum_{i=1}^N P_t(x_i) e^{-\alpha_t y_i h_t(x_i)}$ 为归一化因子, 其作用是使 $P_{t+1}(x)$ 成为一个概率分布。

AdaBoost 算法的训练过程如下:



Bagging 算法:



7 神经网络

7.1 概述

- ⌚ 深度学习是机器学习的子领域，其主要出发点是模拟人脑神经系统的深层结构和人脑认知过程的逐层抽象、逐次迭代机制
- ⌚ 深度学习基于深层网络模型、面向低层数据对象、采用逐层抽象机制、逐层学习、最终形成高层概念的机器学习方式

⌚ 神经元：

- ⚡ 深度学习模型中的基本单元
 - ⚡ 对输入信息进行加权累加： $In = \sum_{i=1}^n w_i * a_i$
 - ⚡ 对累加结果进行非线性变换（通过激活函数）： $g(x)$
 - ⚡ 神经元的输出： $Out = g(In + b)$
 - ⌚ 常用激活函数 Softmax：
- 一般用于多分类问题中（输出层），其将输入数据 x_i 映射到第 i 个类别的概率 y_i ，并可将输出概率最大的作为分类目标：

$$y_i = \text{softmax}(x_i) = \frac{e^{x_i}}{\sum_{j=1}^k e^{x_j}}$$

7.2 前馈神经网络

2 Feedforward Neural Network 基本形式：

- ／ 分为输入层，隐藏层和输出层
- ／ 各个神经元接受前一级的输入，并输出到下一级，模型中没有反馈
- ／ 层与层之间通过全连接进行链接，即两个相邻层之间的神经元完全成对连接，但层内的神经元不相互连接

2 参数优化（梯度下降法）：

- ／ 目标：

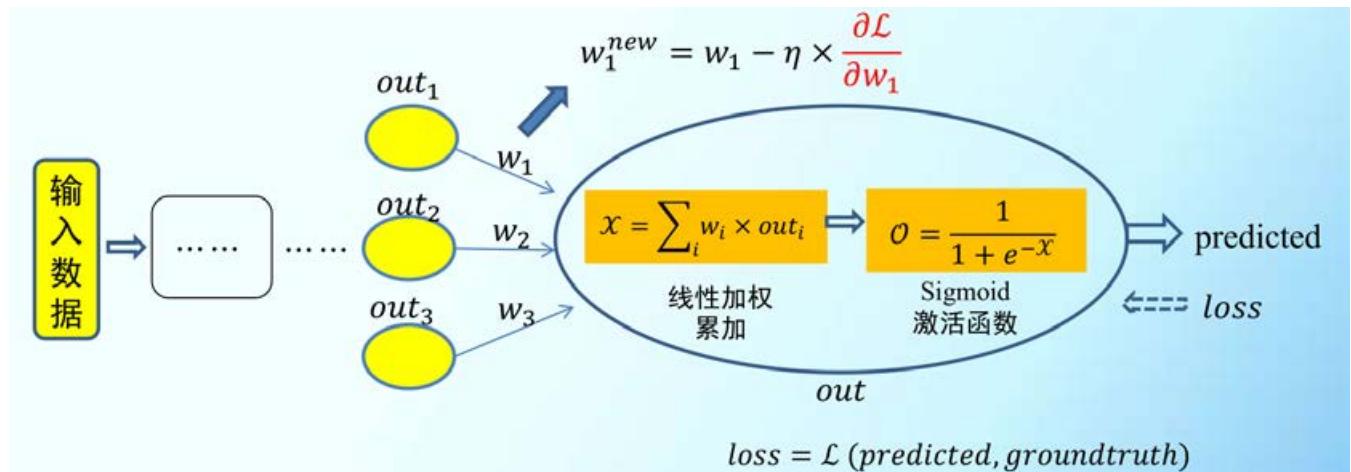
求 w_{ij} ($1 \leq i \leq n, 1 \leq j \leq m$) n 为神经网络层数， m 为每层中神经元个数

- ／ 假设损失函数 $f(x)$ 是连续可微的多元变量函数，其泰勒展开如下 (Δx 是微小的增量)：

$$\begin{aligned} f(x + \Delta x) &= f(x) + f'(x)\Delta x + \frac{1}{2}f''(x)(\Delta x)^2 + \dots + \frac{1}{n!}f^{(n)}(x)(\Delta x)^n \\ f(x + \Delta x) - f(x) &\approx (\nabla f(x))^T \Delta x = \|\nabla f(x)\| \|\Delta x\| \cos \theta \end{aligned}$$

当 $\theta = 180^\circ$ 时， $f(x + \Delta x) - f(x) = -\|\Delta x\| \|\nabla f(x)\|$ ，沿着损失函数梯度的反方向选取 x 的增量 Δx 就会保证损失误差下降最多、下降最快，在实际中引入（可固定）步长 η ，用 $x - \eta \nabla f(x)$ 来更新 x

2 参数优化—误差反向传播（Error Back Propagation, BP）



- ／ BP 算法是一种将输出层误差反向传播给隐藏层进行参数更新的方法
- ／ 将误差从后向前传递，将误差分摊给各层所有单元，从而获得各层单元所产生的误差，进而依据这个误差来让各层单元负起各自责任、修正各单元参数
- ／ 为了使损失函数 \mathcal{L} 取值减少（从而保证模型预测结果与实际结果之间的差距越来越小），需要先求损失函数 \mathcal{L} 对 w_1 的偏导 $\frac{d\mathcal{L}}{dw_1}$ ，然后按照损失函数梯度的反方向选取一个微小的增量，来调整 w_1 的取值，就能够保证损失函数取值减少。即将 w_1 变为 $w_1 - \eta \frac{d\mathcal{L}}{dw_1}$ 后，能使得损失误差减少

求 $\frac{d\mathcal{L}}{dw_1}$:

由于 w_1 与加权累加函数 \mathcal{X} 和 sigmoid 函数均有关, 因此 $\frac{d\mathcal{L}}{dw_1} = \frac{d\mathcal{L}}{dO} \frac{dO}{dX} \frac{dx}{dw_1}$ 。

在这个链式求导中:

与损失函数的定义有关

是对 sigmoid 函数求导, 结果为 $\frac{1}{1 + e^{-x}} \times \left(1 - \frac{1}{1 + e^{-x}}\right)$

$\frac{dx}{dw_1}$ 是加权累加函数 $w_1 \times out_1 + w_2 \times out_2 + w_3 \times out_3$ 对 w_1 求导, 结果为 out_1

参数更新: 在下一轮迭代中参数 w_1 的取值被调整为:

$$\begin{aligned} w_1^{\text{new}} &= w_1 - \eta \times \frac{d\mathcal{L}}{dw_1} = w_1 - \eta \times \frac{d\mathcal{L}}{dO} \frac{dO}{dX} \frac{dx}{dw_1} \\ &= w_1 - \eta \times \frac{d\mathcal{L}}{dO} \times \frac{1}{1 + e^{-X}} \times \left(1 - \frac{1}{1 + e^{-X}}\right) \times out_1 \end{aligned}$$

按照同样的方法, 可调整 w_2 、 w_3 以及其它图中未显示参数的取值。经过这样的调整, 在模型参数新的取值作用下, 损失函数 $\mathcal{L}(\theta)$ 会以最快下降方式逐渐减少, 直至减少到最小值 (即全局最小值)。

链式求导实现了损失函数对某个自变量求偏导, 好比将损失误差从输出端向输入端逐层传播, 通过这个传播过程来更新该自变量取值。

一旦神经网络在当前参数下给出了预测结果 $(\hat{y}_1, \hat{y}_2, \hat{y}_3)$ 后, 通过均方误差损失函数来计算模型预测值与真实值 (y_1, y_2, y_3) 之间误差, 记为 $loss = \frac{1}{3} \sum_{i=1}^3 (\hat{y}_i - y_i)^2$ 。

接下来通过梯度下降和误差反向传播方法, 沿着损失函数梯度的反方向来更改参数变量取值, 使得损失函数快速下降到其最小值。由于损失函数对 $w_7 \sim w_{12}$ 的偏导计算相似,

在此以 w_7 为例来介绍如何更新 w_7 这一参数取值。记损失函数为

$$\mathcal{L}(w), \mathcal{L}(w) = \frac{1}{3} \sum_{i=1}^3 (\hat{y}_i - y_i)^2$$

损失函数 \mathcal{L} 对参数 w_7 的偏导可如下计算: $\delta_7 = \frac{\partial \mathcal{L}}{\partial w_7} = \frac{\partial \mathcal{L}}{\partial \hat{y}_1} * \frac{\partial \hat{y}_1}{\partial Ino_1} * \frac{\partial Ino_1}{\partial w_7}$

上述公式中每一项结果如下: $\frac{\partial \mathcal{L}}{\partial \hat{y}_1} = \hat{y}_1 - y_1$, $\frac{\partial \hat{y}_1}{\partial Ino_1} = \hat{y}_1 * (1 - \hat{y}_1)$, $\frac{\partial Ino_1}{\partial w_7} = Out_{h_1}$

7.3 卷积神经网络

⌚ 卷积：

⚡ 基本过程是：

针对图像的某一类特征，先构造其特征过滤器 (filter, 卷积核, convolution kernel)，然后利用该滤器对图像进行特征提取，得到相应特征的特征图

⚡ 单个卷积核：

$$O_{i,j} = \sum_{s=1}^{h_k} \sum_{t=1}^{w_k} I_{i+s-1,j+t-1} K_{s,t}$$

⚡ 填充 (padding)：

在进行卷积层的处理之前，有时要向输入数据的周围填入固定的数据 (比如 0 等)

⌚ 池化 (降采样、聚合, Pooling) 操作：

⚡ 模仿人的视觉系统对数据进行降维，池化操作通常也叫做子采样 (Subsampling) 或降采样 (Downsampling)，在构建卷积神经网络时，往往会在卷积层之后，通过池化来降低卷积层输出的特征维度，有效减少网络参数的同时还可以防止过拟合现象。

⚡ 池化层没有激活函数

7.4 循环神经网络 (RNN)

⌚ Recurrent Neural Network, RNN：

⚡ 是一类以序列 (sequence) 数据为输入，在序列的演进方向进行递归 (recursion) 且所有节点 (循环单元) 按链式连接的递归神经网络 (recursive neural network)

⚡ 长短期记忆 (Long Short-Term Memory, LSTM)：

是为了解决一般的 RNN (循环神经网络) 存在的长期依赖问题而专门设计出来的，所有的 RNN 都具有一种重复神经网络模块的链式形式。

⚡ 门控循环单元GRU(Gated Recurrent Unit)：

可看成是 LSTM 的一个变种，能在保留长期序列信息下，减少梯度消失问题。

⌚ Transformer 是 Google 的团队在 2017 年提出的一种 NLP 经典模型，现在比较火热的 Bert 也是基于 Transformer。Transformer 模型使用了 Self-Attention 机制，不采用 RNN 的顺序结构，使得模型可以并行化训练，而且能够拥有全局信息。