人工智能概论-知识表示

赵亚伟

zhaoyw@ucas.ac.cn

中国科学院大学 大数据分析技术实验室

2018.5.12

目录

知识表示的重要性

- **知识是智能的基础**
  - 获得知识
  - 运用知识

- **符合计算机要求的知识模式**
  - 计算机能存储、处理的知识表示模式
  - 数据结构（**List, Table, Tree, Graph, etc.**）

知识表示

- 知识表示就是研究用机器表示上述这些知识的可行性、有效性的一般方法，可以看作是将知识符号化并输入到计算机的过程和方法。
- 知识表示 = 数据结构 + 处理机制

基本概念

- **数据 (Data)**
  - 信息的载体和表示
  - 用一组符号及其组合表示信息
- **信息 (Information)**
  - 数据的语义
  - 数据在特定场合下的具体含义
- **知识 (Knowledge)**
  - 信息关联后所形成的信息结构：事实 & 规则
  - 经加工、整理、解释、挑选、改造后的信息

关于知识

- 知识是由经验总结升华出来的，因此知识是经验的结晶。
- 知识在信息的基础上增加了上下文信息，提供了更多的意义，因此也就更加有用和有价值。
- 知识是随着时间的变化而动态变化的，新的知识可以根据规则和已有的知识推导出来。
- 因此知识反映了信息之间的内在关系或必然联系。

人工智能中的知识

- **人工智能所关心的知识**
  - 事实：是关于对象和物体的知识。
  - 规则：是有关问题中与事物的行动、动作相联系的因果关系的知识。
  - 元知识：是有关知识的知识，是知识库中的高层知识。
  - 常识性知识：泛指普遍存在而且被普遍认识了的客观事实一类知识。

知识分类

- **常识性知识、领域性知识**（作用范围）
- **事实性知识、过程性知识、控制知识**（作用及表示）
- **确定性知识、不确定性知识**（确定性）
- **逻辑性知识、形象性知识**（结构及表现形式）
- **零级知识、一级知识、二级知识**（抽象程度）

知识的特性

- **相对正确性**
  - 一定条件下
  - 某种环境中
  - ……
- **不确定性**
  - 存在“中间状态”
  - “真”（“假”）程度
  - 随机性
  - 模糊性
  - 经验性
  - 不完全性
  - ……
- **可表示性 & 可利用性**
  - 语言
  - 文字
  - 图形
  - 图像
  - 视频
  - 音频
  - 神经网络
  - 概率图模型
  - ……

人工智能对知识表示的要求

- **表示能力**
  - 要求能够正确、有效地将问题求解所需要的各类知识都表示出来
- **可理解性**
  - 所表示的知识应易懂、易读。
- **便于知识的获取**
  - 使得智能系统能够渐进地增加知识，逐步进化。
- **便于搜索**
  - 表示知识的符号结构和推理机制应支持对知识库的高效搜索，使得智能系统能够迅速地感知事物之间的关系和变化；同时很快地从知识库中找到有关的知识。
- **便于推理**
  - 要能够从已有的知识中推出需要的答案和结论。

常用的知识表示方法

- 状态空间法
- 问题规约表示
- 谓词逻辑表示
- 语义网表示
- 框架表示法
- 本体表示
- 过程表示法

如何选择适合的表示法?

- 充分表示**领域知识**
- 有利于对**知识的利用**
- 便于**理解和实现**
- 便于**对知识的组织、管理与维护**

目录

状态空间法

- 问题求解(problem solving)是个大课题，它涉及归约、推断、决策、规划、常识推理、定理证明和相关过程的核心概念。
- 在人工智能中，许多问题的求解采用的是试探搜索方法，也就是说，这些方法是通过在某个可能的解空间内寻找一个解来求解问题的。
- 这种基于解答空间的问题表示和求解方法就是状态空间法，它是以状态和算符(operator)为基础来表示和求解问题的。

问题求解

- **问题求解技术两个主要的方面**
  - 问题的表示：如果描述方法不对，对问题求解会带来很大的困难
  - 求解的方法：采用试探搜索方法。

状态空间法的三要点

- **状态空间法三要点**
  - 状态（**state**）：表示问题解法中每一步问题状况的数据结构；
  - 算符（**operator**）：把问题从一种状态变换为另一种状态的手段；
  - 状态空间方法：基于解答空间的问题表示和求解方法，它是以状态和算符为基础来表示和求解问题的。

问题状态描述

- **要完成某个问题的状态描述，必须确定三件事：**
  - **1.**该状态描述方式，特别是初始状态描述；
  - **2.**操作符集合及其对状态描述的作用；
  - **3.**目标状态描述的特性。

定义

- 状态(state)：为描述某类不同事物间的差别而引入的一组最少变量q_0，q_1，…，q_n的有序集合，其矢量形式如下：

$$Q = [q_0, q_1, \ldots, q_n]^T$$

- 式中每个元素$q_i(i=0,1，\ldots，n)$为集合的分量，称为状态变量。
- 给定每个变量的一组值就得到一个具体的状态，如

$$Q_k = [\, q_{0k}, q_{1k}, \cdots, q_{nk} \,]^T$$

它只是问题所有可能状态的罗列，还必须描述这些状态之间的可能变化。

定义

- 所谓操作，或称为算子是引起状态中的某分量发生改变，从而使问题由一个具体状态A变化为另一具体状态B的作用。
- 算符：使问题从一种状态变化为另一种状态的手段称为操作符或算符。操作符可为走步、过程、规则、数学算子、运算符号或逻辑符号等。

定义Cont.

- **问题的状态空间(state space)：**
  - 是一个表示该问题全部可能状态及其关系的图，它包含三种说明的集合，即所有可能的问题初始状态集合S(初始状态$S_0 \in S$)、操作符集合F以及目标状态集合G($G \subset S$)。可把状态空间记为三元状态(S，F，G)。
- **状态空间可用有向图来表示**
- **状态空间的一个解**
  - 使一个有限的操作算子序列，它使初始状态转化为目标状态：$S_0\text{-}f_1\text{->}S_1\text{-}f_2\text{->}...f_k\text{->}G$

例子：十五数码问题

- 由15个编有1至15并放在4×4方格棋盘上的可走动的棋子组成。棋盘上总有一格是空的，以便可能让空格周围的棋子走进空格，这也可以理解为移动空格。
- 绘出两种棋局，即初始棋局和目标棋局，它们对应于该下棋问题的初始状态和目标状态。
- 如何把初始棋局变换为目标棋局呢？问题的解答就是某个合适的棋子走步序列，如"左移棋子12，下移棋子15，右移棋子4，…"等等。

例子：十五数码问题

- 总状态为16! = 20922789888000
- 由于棋盘的对称性，实际状态数减半
- 上、下、左、右移动四种操作

11	9	4	15
1	3		12
7	5	8	6
13	2	10	14

（a）初始棋局

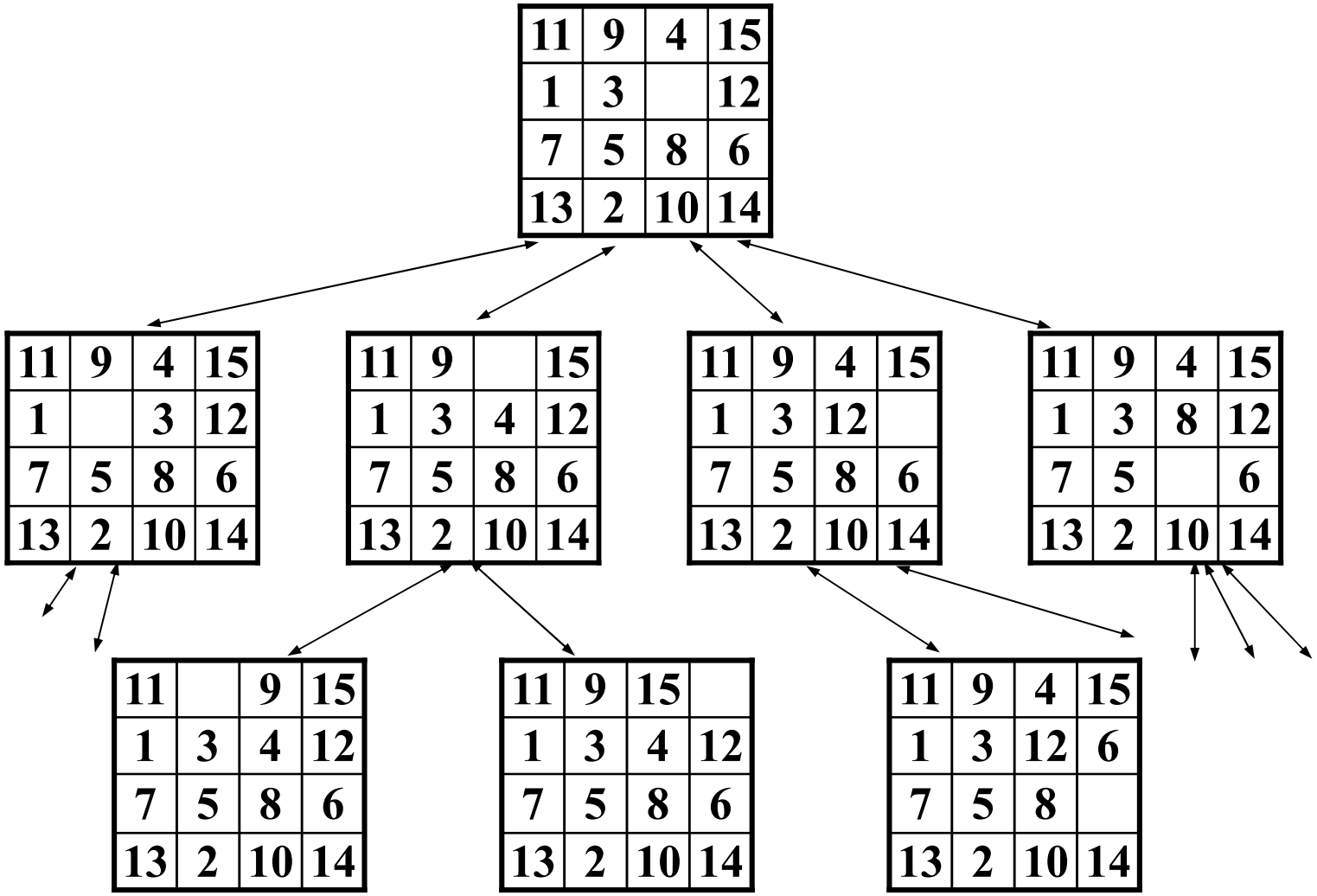
1	2	3	4
5	6	7	8
9	10	11	12
13	14	15	

（b）目标棋局

十五数码难题

例子：十五数码问题

- 十五数码难题最直接的求解方法是尝试各种不同的走步，直到偶然得到该目标棋局为止。
- 本质上是试探搜索。从初始棋局开始，由每一合法走步试探得到的各种新棋局，然后计算再走一步而得到的下一组棋局。
- 这样继续下去，直至达到目标棋局为止。把初始状态可达到的各状态所组成的空间设想为一幅由各种状态对应的节点组成的图。这种图称为状态图。图中每个节点标有它所代表的棋局。
- 首先把适用的算符用于初始状态，以产生新的状态；然后，再把另一些适用算符用于这些新的状态；这样继续下去，直至产生目标状态为止。


11 9 4 15
1 3 12
7 5 8 6
13 2 10 14
11 9 4 15
1 3 4 12
7 5 8 6
13 2 10 14
11 9 15
1 3 4 12
7 5 8 6
13 2 10 14
11 9 4 15
1 3 12
7 5 8 6
13 2 10 14
11 9 4 15
1 3 8 12
7 5 6
13 2 10 14
11 9 4 15
1 3 4 12
7 5 8 6
13 2 10 14
11 9 15
1 3 4 12
7 5 8 6
13 2 10 14
11 9 4 15
1 3 12 6
7 5 8
13 2 10 14

- **我们一般用状态空间法这一术语来表示下述方法：**
  - 从某个初始状态开始，每次加一个操作符，递增地建立起操作符的试验序列，直到目标状态为止。
- **寻找状态空间的全部过程包括从旧的状态描述产生新的状态描述，以及此后检验这些新的状态描述，看其是否描述了该目标。**
- **这种检验往往只是查看某个状态是否与给定的目标状态描述相匹配。**

- **要完成某个问题的状态描述，必须确定三件事：**
  - 1.该状态描述方式，特别是初始状态描述；
  - 2.操作符集合及其对状态描述的作用；
  - 3.目标状态描述的特性。

状态图示法

- **图论中的几个术语**
  - 节点(node)：图形上的汇合点，用来表示状态、事件和时　　间关系的汇合，也可用来指示通路的汇合；
  - 弧线(arc)：节点间的连接线；
  - 有向图(directed graph)：一对节点用弧线连接起来，从一个节点指向另一个节点。

状态图示法 Cont.

- **图论中的几个术语**
  - 后继节点(descendant node)与父辈节点(parent node)：如果某条弧线从节点n_i指向节点n_j，那么节点n_j就叫做节点n_i的后继节点或后裔，而节点n_i叫做节点n_j的父辈节点或祖先。
  - 路径：某个节点序列(n_{i1}, n_{i2}, …, n_{ik})当j=2，3，…，k时，如果对于每一个$n_{i,\ j-1}$都有一个后继节点n_{ij}存在，那么就把这个节点序列叫做从节点n_{i1}至节点n_{ik}的长度为k的路径。
  - 代价：用c(n_i, n_j)来表示从节点n_i指向节点n_j的那段弧线的代价。

状态图示法 Cont.

- **图论中的几个术语**
  - 如果从节点n_i至节点n_j存在有一条路径，那么就称节点n_j是从节点n_i可达到的节点。
  - 两节点间路径的代价等于连接该路径上各节点的所有弧线代价之和。最小者称为最小代价的路径。

状态图示法 Cont.

- **图论中的几个术语**
  - 显式表示：各节点及其具有代价的弧线由一张表明确给出。此表可能列出该图中的每一节点、它的后继节点以及连接弧线的代价。
  - 隐式表示：节点的无限集合$\{s_i\}$作为起始节点是已知的。后继节点算符Γ也是已知的，它能作用于任一节点以产生该节点的全部后继节点和各连接弧线的代价。

图的显式和隐式表示

- **一个图可由显式说明也可由隐式说明**
  - 显式说明对于大型的图是不切实际的，而对于具有无限节点集合的图则是不可能的。
- **引入后继节点算符的概念是方便的**
  - 后继节点算符Γ也是已知的，它能作用于任一节点以产生该节点的全部后继节点和各连接弧线的代价
  - 把后继算符应用于$\{s_i\}$的成员和它们的后继节点以及这些后继节点的后继节点，如此无限制地进行下去，最后使得由Γ和$\{s_i\}$所规定的隐式图变为显示图
  - 把后继算符应用于节点的过程，就是扩展一个节点的过程。

图的表示与搜索

- 因此，搜索某个状态空间以求得算符序列的一个解答的过程，对应于使隐式图足够大一部分变为显式以便包含目标的过程。这样的搜索图是状态空间问题求解的主要基础。
- 问题的表示对求解工作量有很大的影响。人们显然希望有较小的状态空间表示。许多似乎很难的问题，当表示适当时就可能具有小而简单的状态空间。

目录

问题归约法

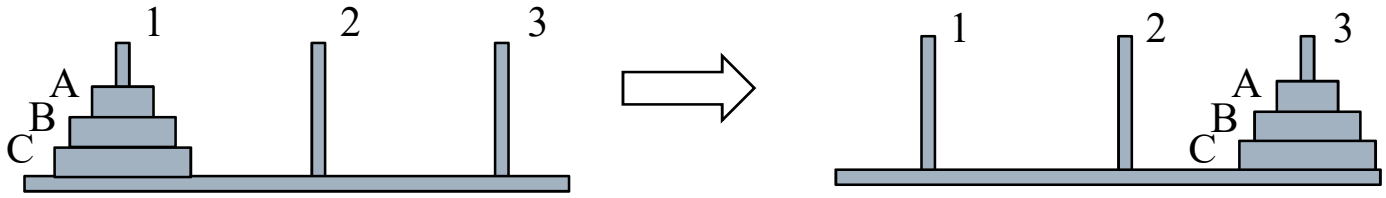
- **问题归约(problem reduction)是另一种问题描述与求解方法。**
  - 先把问题分解为子问题和子-子问题，然后解决较小的问题。
  - 对该问题的某个具体子集的解答就意味着对原始问题的一个解答。

问题归约描述

- **问题归约表示的组成部分：**
  - 一个初始问题描述；
  - 一套把问题变换为子问题的操作符；
  - 一套本原问题描述。
    - 其中的每一个问题是不证明的，自然成立的，如公理、已知的实事等（本原问题集）
- **问题归约的实质：**
  - 从目标(要解决的问题)出发逆向推理，建立子问题以及子问题的子问题，直至最后把初始问题归约为一个平凡的本原问题集合。

梵塔难题

- 有3个柱子(1，2和3)和3个不同尺寸的圆盘（A，B和C）。在每个圆盘的中心有一个孔，所以圆盘可以堆叠在柱子上。
  - 最初，3个圆盘都堆在柱子1上：最大的圆盘C在底部，最小的圆盘A在顶部。
  - 要求把所有圆盘都移到柱子3上，每次只许移动一个，而且只能先搬动柱子顶部的圆盘，还不许把尺寸较大的圆盘堆放在尺寸较小的圆盘上。
  - 这个问题的初始配置和目标配置如图所示。



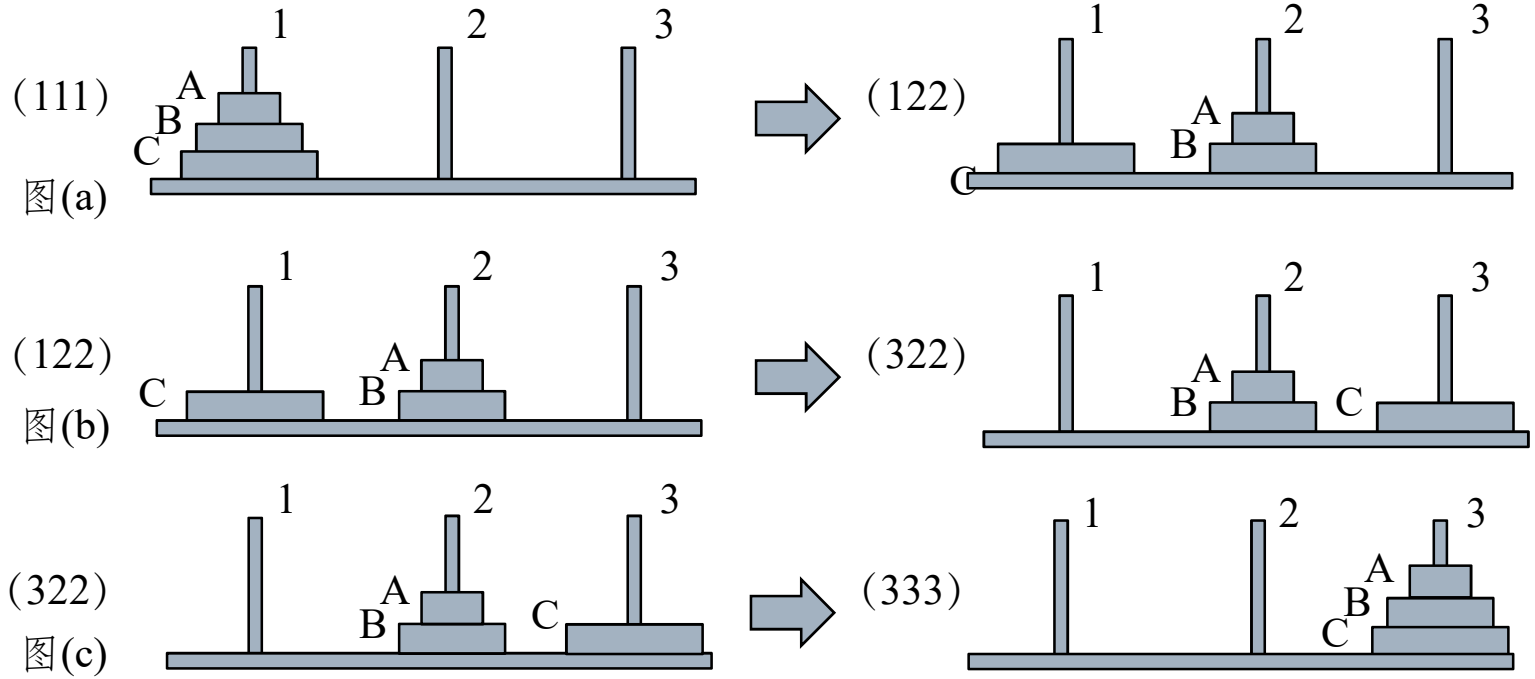
初始状态　　目标状态

梵塔难题 Cont.

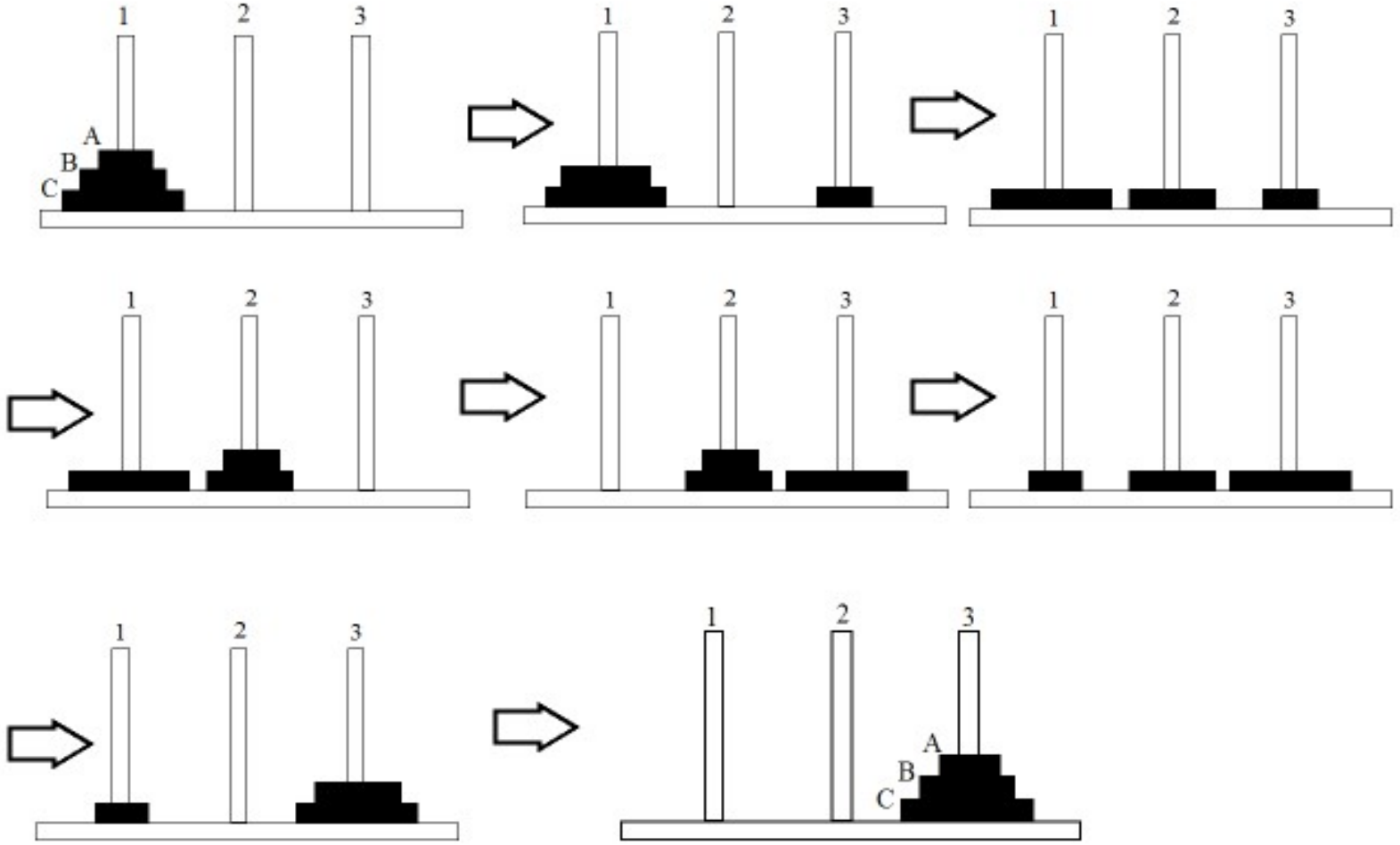
- **解题过程：**
  - 将原始问题归约为一个较简单问题集合，要把所有圆盘都移至柱子3，我们必须首先把圆盘C移至柱子3；而且在移动圆盘C至柱子3之前，要求柱子3必须是空的。只有在移开圆盘A和B之后，才能移动圆盘C；而且圆盘A和B最好不要移至柱子3就不能把圆盘C移至柱子3。因此，首先应该把圆盘A和B移到柱子2上。然后才能够进行关键的一步，把圆盘C从柱子1移至柱子3，并继续解决难题的其余部分。
  - 将原始难题归约（简化）为下列子难题：移动圆盘A和B至柱子2的双圆盘难题，如图(a)所示。

梵塔难题 Cont.

- **把原始难题归约（简化）为以下三个子难题：**
  - 移动圆盘A和B至柱子2的双圆盘难题；如图(a)所示
  - 移动圆盘C至柱子3的单圆盘难题；如图(b)所示
  - 移动圆盘A和B至柱子3双圆盘难题；如图(c)所示



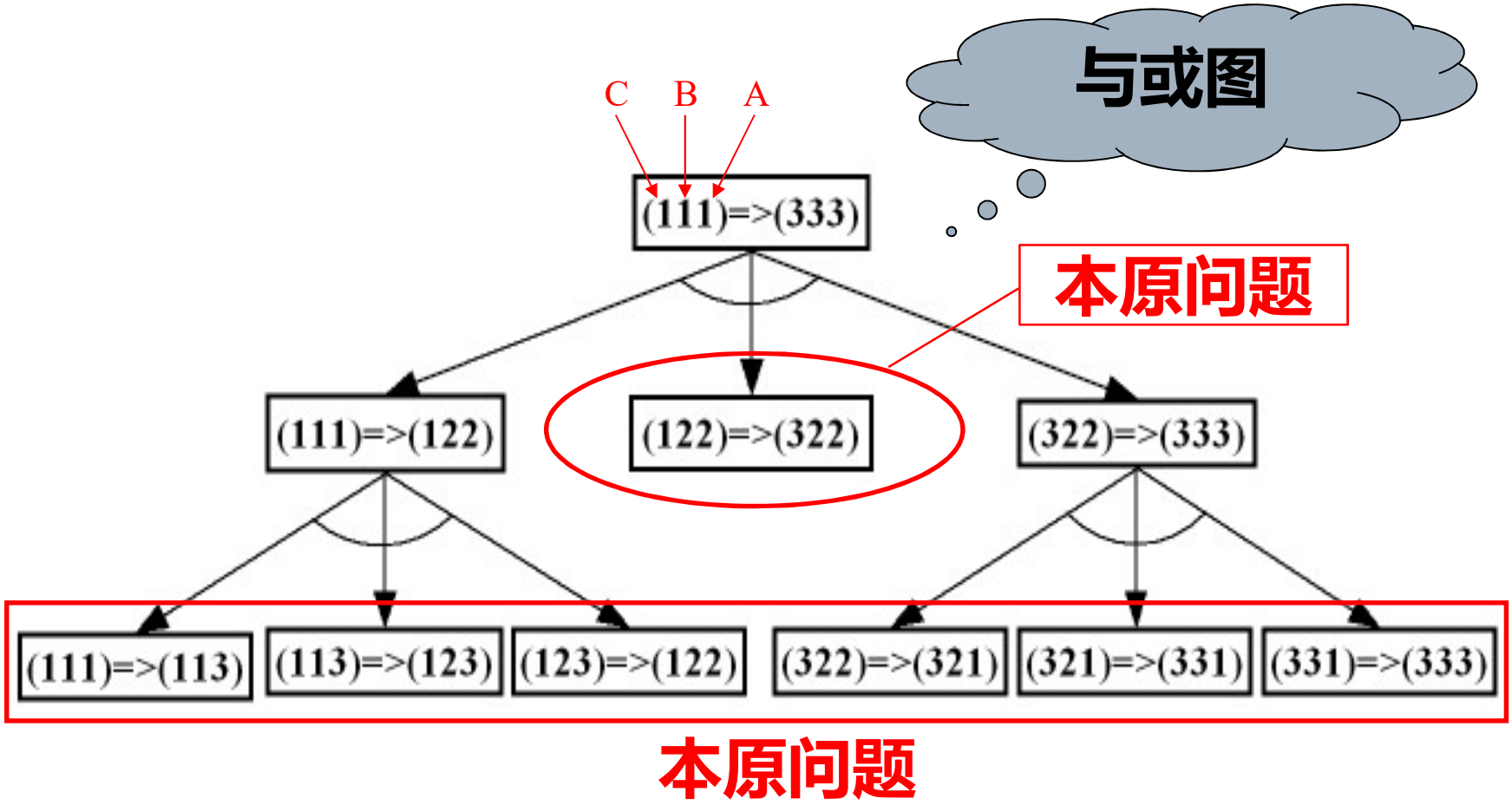
梵塔难题 Cont.



与或图

- **梵塔问题归约图：**
  - 子问题2可作为本原问题考虑，因为它的解只包含一步移动。应用一系列相似的推理，子问题1和子问题3也可被归约为本原问题，如图（下页）所示。这种图式结构，叫做与或图(AND/OR graph)。
- **它能有效地说明如何由问题归约法求得问题的解答。**

与或图Cont.

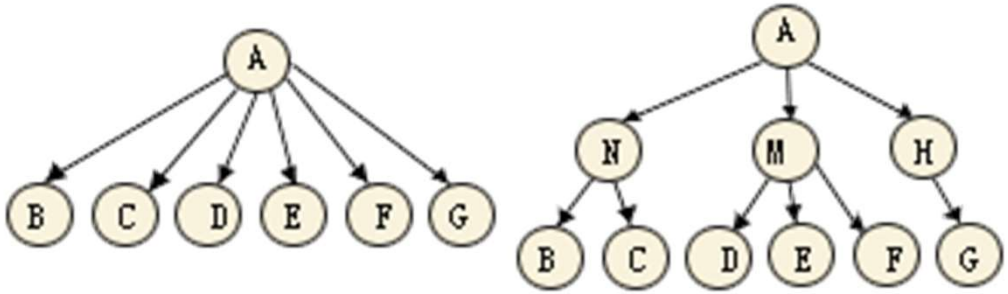

与或图
C
B
A
(111)=>(333)
本原问题
(111)=>(122)
(122)=>(322)
(322)=>(333)
(111)=>(113)
(113)=>(123)
(123)=>(122)
(322)=>(321)
(321)=>(331)
(331)=>(333)
本原问题

归约-本原问题

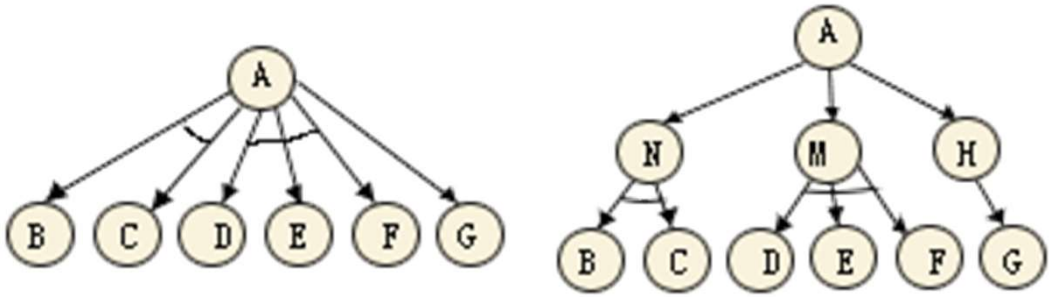
- 把一个问题描述变换为一个归约或后继问题描述的集合，这是由问题归约算符进行的。变换所得所有后继问题的解就意味着父辈问题的一个解。
- 所有问题归约的目的是最终产生具有明显解答的本原问题。这些问题可能是能够由状态空间搜索中走动一步来解决的问题，或者可能是别的具有已知解答的更复杂的问题。

与或图表示

- 一般地，我们用一个类似图的结构来表示把问题归约为后继问题的替换集合，这种结构图叫做问题归约图，或叫与或图。如下图所示：



子问题集合



与或图

与或图术语

- **一些关于与或图的术语：**
  - 父节点、子（后继）节点、弧线、起始节点。
  - 终叶节点：对应于原问题的本原节点。
  - 或节点：只要解决某个问题就可解决其父辈问题的节点集合，如（M，N，H）。
  - 与节点：只有解决所有子问题，才能解决其父辈问题的节点集合，如（B,C)和（D,E,F）各个结点之间用一端小圆弧连接标记。
- **与或图：由与节点及或节点组成的结构图。**

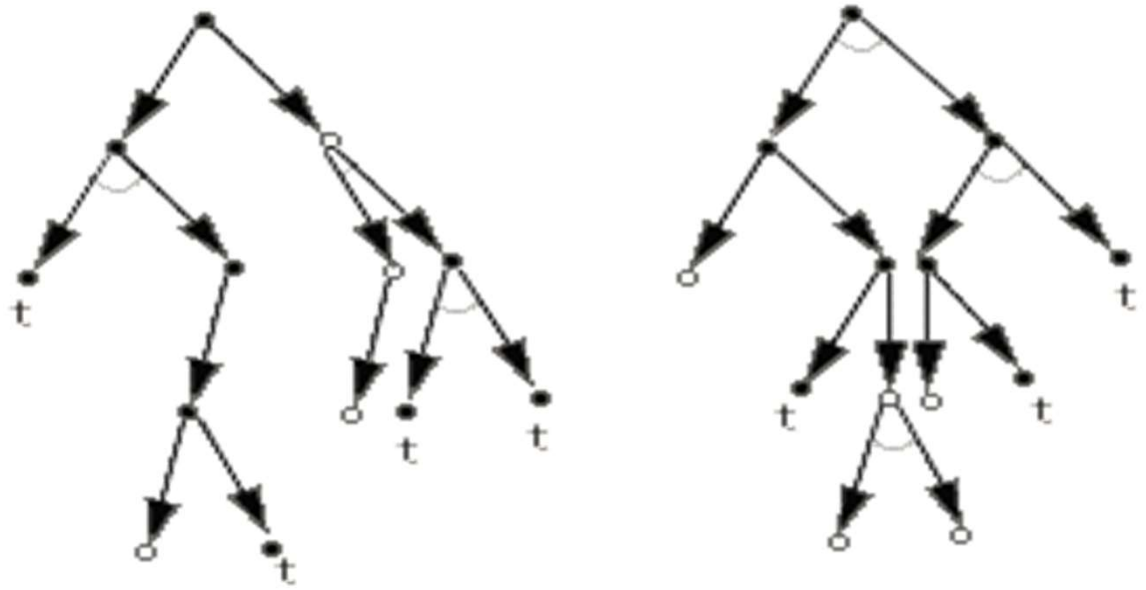
与或图术语Cont.

- **可解节点的一般定义**
  - (1) 终叶节点是可解节点(因为它们与本原问题相关连)。
  - (2) 如果某个非终叶节点含有或后继节点，那么只要当其后继节点至少有一个是可解的时，此非终叶节点才是可解的。
  - (3) 如果某个非终叶节点含有与后继节点，那么只要当其后继节点全部为可解时，此非终叶节点才是可解的。

与或图术语Cont.

- **不可解节点的一般定义:**
  - (1) 没有后裔的非终叶节点为不可解节点。
  - (2) 如果某个非终叶节点含有或后继节点，那么只有当其全部后裔为不可解时，此非终叶节点才是不可解的。
  - (3) 如果某个非终叶节点含有与后继节点，那么只要当其后裔至少有一个为不可解时，此非终叶节点才是不可解的。

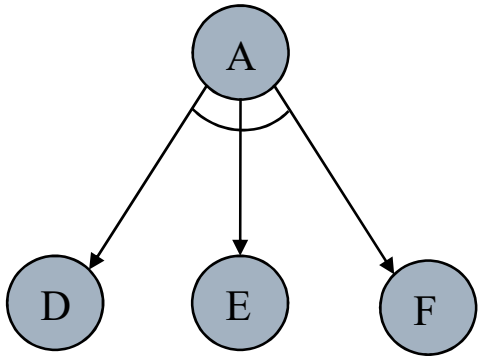
- 图中，终叶节点用字母 *t* 表示，有解节点用小原点表示，而解图用粗线分支表示。



与或图例子

与或图构成规则

- **规则1**：每个节点代表一个要解决的单一问题或问题集合，起始节点对应原始问题。
- **规则2**：终叶节点：没有后裔，本原问题。
- **规则3**：有向弧线自A指向后继节点表示所求得的子问题集合。
- **规则4**：有向弧线A节点指向此子问题集合中的各个节点。由于只有当集合中所有的项都有解时，这个子问题的集合才能获得解答，所以这些子问题节点叫做与节点。
- **规则5**：在特殊情况下，当只有一个算符可应用于问题A，而且这个算符产生具有一个以上子问题的某个集合时，由上述规则3和规则4所产生的图可以得到简化。
- 因此，代表子问题集合的中间或节点可以被略去，如右图所示。



与或图

目录

谓词逻辑法

命题逻辑与谓词逻辑

- ✓ **命题逻辑**与**谓词逻辑**是最先用于人工智能的两种逻辑，对于知识的形式化表示，特别是定理的证明发挥了重要作用
- ✓ 虽然命题逻辑能够把客观世界的各种事实表示为逻辑命题，但是它具有较大的局限性。命题逻辑只能进行命题间关系的推理，无法解决与命题结构和成分有关的推理问题，不适合表示比较复杂的问题。
- ✓ 谓词逻辑是在命题逻辑的基础上发展而来的，命题逻辑可以看作是谓词逻辑的一种特殊形式。

谓词逻辑法

命题

- 命题是具有真假意义的语句
- 命题代表人们进行思维时的一种判断，若命题的意义为真，称它的真值为“真”，记作“T”；若命题的意义为假，称它的真值为“假”，记作“F”。例如：
  - “西安是陕西省省会”“10大于6”是真值为“T”的命题
  - “月亮是方的”“煤炭是白的”是真值为“F”的命题
- 一个命题不能同时即为真又为假，但可以在一定条件下为真，在另一种条件下为假。例如：
  - “1+1=10”在二进制情况下为真，十进制情况下为假

谓词逻辑法

命题

- ✓ 没有真假意义的语句，如感叹句、疑问句等，**不是命题**。
- ✓ 通常用大写英文字母表示一个命题，例如：
  - P：西安是座古老的城市

命题逻辑的局限性

- ✓ 客观事物的结构及逻辑特征？
- ✓ 不同事物间的共同特征？

谓词逻辑法

命题逻辑的局限性?

- ✓ 命题这种表示方法无法把它所描述的客观事物的结构及逻辑特征反映出来，也不能把不同事物间的**共同特征**表述出来
- ✓ 例如，用字母P表示“小张是老张的儿子”这一命题，则无法表述出老张与小张是父子关系
- ✓ 又如，“张三是学生”，“李四是学生”这两个命题，用命题逻辑表示时，无法把两者的共同特征“都是学生”形式的表示出来
- ✓ 可否用 **Student（“张三”）， Student（“李四”）**表示上述命题？——**谓词逻辑**

谓词逻辑法

谓词

- 在谓词逻辑中，命题是用形如$P(x_1,x_2,\ldots,x_n)$的谓词来表述的。一个谓词可分为**谓词名**与**个体**两个部分
- **个体**：是命题的主语，表示独立存在的事物或某个抽象的概念
  - “$x_1,x_2,\ldots,x_n$”是个体，一般用小写字母表示
  - 个体可以是个体常量、变元或函数
- **谓词名**：表示个体的性质、状态或个体之间的关系
  - “P”是谓词名，一般用大写字母表示
  - 称P 是一个n元谓词。

谓词逻辑法

谓词

- 对于命题“张三是学生”，用谓词可以表示为：Student（“张三”）。其中，Student是谓词名，“张三”是个体，Student刻画了“张三”是个学生这一特征。
- 在谓词中，个体可以是常量，也可以是变元，还可以是一个函数。
  - 例如，对于命题“x>10”可以表示为more（x,10），其中x是变元。又如，命题“小张的父亲是老师”，可以表示为Teacher（father（Zhang）），其中，father（Zhang）是一个函数。
- 当谓词中的变元都用特定的个体取代时，谓词就具有一个确定的真值“T”或“F”。

谓词逻辑法

谓词

- 在n元谓词 $P(x_1,x_2,\ldots,x_n)$中，若每个个体均为常量、变元或函数，则称它为**一阶谓词**。
- 如果某个个体本身又是一个一阶谓词，则称它为**二阶谓词**，如此类推。
- 个体变元的取值范围称为**个体域**。个体域可以是有限的，也可以是无限的。例如用I（x）表示“x是整数”，则个体域为所有整数，是无限的。
- **谓词与函数不同**，谓词的真值是“T” 或“F”，而函数的值是个体域中的一个个体，无真值可言。

谓词逻辑法

谓词演算

✓ 谓词逻辑语言的语法和语义

- **基本符号：** 谓词符号、变量符号、函数符号、常量符号、括号和逗号
- **原子公式：** 原子公式由若干谓词符号和项组成
  - 谓词符号规定定义域内的一个相应关系
  - 常量符号是最简单的项，表示论域内的物体或实体
  - 变量符号也是项，不明确涉及是哪一个实体
  - 函数符号表示论域内的函数，是从论域内的一个实体到另外一个实体的映射
  - 例如：原子公式 **Married [father(LI) , mother(LI)]**表示“李（LI）的父亲和他的母亲结婚”

谓词逻辑法

✓ 连词和量词

□ 连词

- 合取：符号“∧ ”， 表示所连结的两个命题之间具有“与”的关系。
- 析取： 符号“∨ ”，表示所连结的两个命题之间具有“或”的关系
- 蕴涵：符号“→ ” ，表示“若…则…”的语义。**P→Q**读作“如果**P**，则**Q**”其中，**P**称为条件的前件，**Q**称为条件的后件。
- 非：符号“¬”，表示对其后面的命题的否定
- 双条件：符号“↔ ”，表示“当且仅当”的语义。**P↔Q**读作“**P**当且仅当**Q**”。

谓词逻辑法

量词

- 全称量词：符号“∀”，意思是“所有的”、“任一个”
  - ∀x读作“对一切x”,或“对每一x”，或“对任一x”。
  - 命题(∀x)P(x)为真，当且仅当对论域中的所有x，都有P(x)为真
  - 命题(∀x)P(x)为假，当且仅当至少存在论域中的一个x，使得P(x)为假
- 存在量词：符号“∃”，意思是“至少有”、“存在”
  - ∃ x读作“存在一个x”,或“对某些x”，或“至少有一x”。
  - 命题(∃x)P(x)为真，当且仅当至少存在论域中的一个x，使得P(x)为真
  - 命题(∃ x)P(x)为假，当且仅当对论域中的所有x，都有P(x)为假

谓词逻辑法

谓词公式

- ✓ 原子谓词公式：是由谓词符号和若干项组成的谓词演算。
- ✓ 分子谓词公式：可以用连词把原子谓词公式组成复合谓词公式，并把它叫做分子谓词公式。
- ✓ 合式公式（**WFF，Well-formed Formulas**）：通常把合式公式叫做谓词公式，递归定义如下：
  - **(1)** 原子谓词公式是合式公式
  - **(2)** 若**A**为合式公式，则¬ **A**也是一个合式公式
  - **(3)** 若**A**，**B**是合式公式，则**A**∨**B**，**A**∧**B**，**A**→**B**，**A**↔**B**也都是合式公式
  - **(4)** 若**A**是合式公式，**x**为**A**中的自由变元，则 (∀**x**)**A**和(∃ **x**)**A**都是合式公式
  - **(5)** 只有按上述规则**(1)**至**(4)**求得的那些公式，才是合式公式。

谓词逻辑法

谓词公式

- ✓ 用谓词公式表示知识时，需要首先定义谓词，然后再用连接词把有关的谓词连接起来，形成一个谓词公式表达一个完整的意义。
- ✓ 例1：设有下列知识
  - ①刘欢比他父亲出名。
  - ②高扬是计算机系的一名学生，但他不喜欢编程 。
  - ③任何整数或者为正或者为负。

为了用谓词公式表示上述知识，首先需要定义谓词：

- **FAMOUS (x, y)： x比y出名**
- **COMPUTER (x) : x 是计算机系的**
- **LIKE (x, y) : x 喜欢 y**

谓词逻辑法

- I(x)表示“x是整数”
- P(x)表示“x是正数”
- N(x)表示“x是负数”

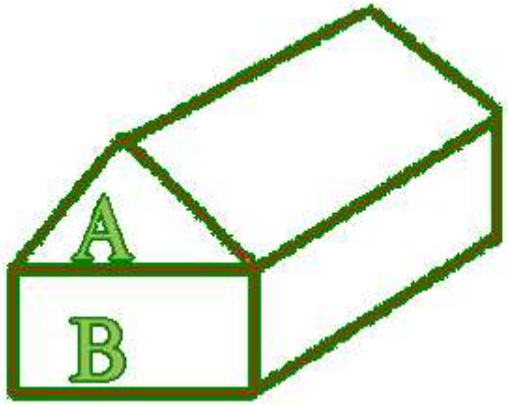
此时可用谓词公式把上述知识表示为:

- ①刘欢比他父亲出名:
  - FAMOUS (liuhuan, father (liuhuan))
- ②高扬是计算机系的一名学生，但他不喜欢编程 :
  - COMPUTER(gaoyang)∧¬LIKE(gaoyang, programing)
- ③任何整数或者为正或者为负:
  - $(\forall x)(I(x) \rightarrow (P(x) \vee N(x)))$

谓词逻辑法

谓词公式

- 例2：用谓词逻辑描述右图中的房子的概念



  - 个体：A , B
  - 谓词：
    - SUPPORT(x,y)：表示 x 被 y支撑着
    - WEDRE (x)：表示 x 是楔形块
    - BRICK(y)：表示 y 是长方块
  - 其中 x , y是个体变元，它们的个体域｛A,B｝
  - 房子的概念可以表示成一组合式谓词公式的合取式：

    SUPPORT(A,B) ∧WEDGE(A) ∧BRICK(B)

谓词逻辑法

合式公式的性质

- ✓ 若P、Q是两个合式公式，则由这两个合式公式所组成的复合表达式可由下列真值表给出。

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

谓词逻辑法

合式公式的性质

- ✓ 如果两个合式公式，无论如何解释，其真值表都是相同的，那么我们就称此两合式公式是等价的，用“=”表示。
- ✓ 应用上述真值表可以确立下列等价关系：
  - （1）否定之否定：$\neg(\neg P) = P$
  - （2）$(P \rightarrow Q) = (\neg P \vee Q)$，或者$(\neg P \rightarrow Q) = (P \vee Q)$
  - （3）狄·摩根定律：
    - $\neg(P \vee Q) = \neg P \wedge \neg Q$
    - $\neg(P \wedge Q) = \neg P \vee \neg Q$
  - （4）分配律：
    - $P \vee (Q \wedge R) = (P \vee Q) \wedge (P \vee R)$
    - $P \wedge (Q \vee R) = (P \wedge Q) \vee (P \wedge R)$

谓词逻辑法

- （5）交换律：
  - $P \vee Q = Q \vee P$
  - $P \wedge Q = Q \wedge P$
- （6）结合律：
  - $P \vee (Q \vee R) = (P \vee Q) \vee R$
  - $P \wedge (Q \wedge R) = (P \wedge Q) \wedge R$
- （7）逆否率：
  - $(P \rightarrow Q) = (\neg Q \rightarrow \neg P)$
- （8）泛界律：
  - $P \vee F = P, P \wedge T = P$
  - $P \wedge F = F, P \vee T = T$
- （9）互余律：
  - $P \vee \neg P = T,\ P \wedge \neg P = F$

谓词逻辑法

- ✓ 此外还可以确立下列等价关系：
  - $\neg (\exists x)\, P(x) = (\forall x)\, [\neg P(x)]$
  - $\neg (\forall x)\, P(x) = (\exists x)\, [\neg P(x)]$
  - $(\forall x)\, [P(x) \wedge Q(x)] = (\forall x)\, P(x) \wedge (\forall x)\, Q(x)$
  - $(\forall x)\, [P(x) \vee Q(x)] = (\forall x)\, P(x) \vee (\forall x)\, Q(x)$
  - $(\forall x)\, P(x) = (\forall y)\, P(y)$
  - $(\exists x)\, P(x) = (\exists y)\, P(y)$

谓词逻辑法

置换与合一

✓ 置换

- 推理规则：用合式公式的集合产生新的合式公式
  - 假言推理

$$\frac{W_1,\ W_1 \rightarrow W_2}{W_2}$$

  - 全称化推理

$$\frac{(\forall x)\,W(x)\ \text{，任意常量}A}{W(A)}$$

  - 综合推理

$$\frac{W_1(A),\ (\forall x)\,W_1(x) \rightarrow W_2(x)}{W_2(A)}$$

寻找A对x的置换，使$W_1(A)$与$W_1(x)$一致

谓词逻辑法

置换与合一

✓ 置换（Substitution）

- 置换的定义：置换是用变元、常量、函数来替换变元，使该变元不在公式中出现。
- 置换是形如{ $t_1/x_1, t_2/x_2, \ldots, t_n/x_n$}的有限集合。
  - t_1，t_2，……，t_n是项
  - x_1，x_2，……，x_n是互不相同的变元
  - t_i/x_i 表示用 t_i 项替换变元 x_i，不允许 t_i 和 x_i 相同，也不允许变元 x_i 循环地出现在另一个 t_j 中

谓词逻辑法

- 置换与合一
  - ✓ 置换（Substitution）
    - 例如
      - {a/x , f(b)/y ， w/z} 是一个置换
      - {g(y)/x , f(x)/y} 不是一个置换
      - {g(a)/x , f(x)/y} 不是一个置换

谓词逻辑法

置换与合一

✓ 置换（Substitution）

例2.2（P40），表达式 P[x, f(y), B]的置换为

- s1={ z/x, w/y}； s2={ A/y}； s3={ q(z)/x , A/y}；
  s4={ c/x , A/y}

用Es表示一个表达式E用置换s所得到的表达式的置换。于是，P[x, f(y), B]的4个置换如下：

- P[x, f(y), B] s1 = P[z, f(w), B]
- P[x, f(y), B] s2 = P[x, f(A), B]
- P[x, f(y), B] s3 = P[q(z), f(A), B]
- P[x, f(y), B] s4 = P[c, f(A), B]

谓词逻辑法

置换与合一

✓ 置换（Substitution）

- 置换是可结合的
  - 用s1s2表示两个置换s1和s2的合成，L表示一个表达式，则有
    - （Ls1）s2 ＝L（s1s2 ）
    - 即用s1和s2相继作用于表达式L是与用s1s2作用于L一样的
  - 进一步推广：（s1s2）s3 ＝s1（s2s3 ）
- 一般说来，置换是不可交换的，即

  $s1s2 \neq s2s1$

谓词逻辑法

置换与合一

✓ 合一（Unification）

- 合一的定义：寻找项对变量的置换，以使两表达式一致。
- 如果一个置换s作用于表达式集合$\{E_i\}$的每个元素，用$\{E_i\}s$表示置换的集。称表达式$\{E_i\}$是可合一的，如果存在一个置换s使得：

$$E_1s = E_2s = E_3s = \cdots\cdots$$

那么，称此s为$\{E_i\}$的合一者（unifier），因为s的作用是使集合$\{E_i\}$成为单一形式。

- 例如，设有公式集 E={ P(x, y, f(y)), P(a, g(x), z) }

则下式是它的一个合一：

s={a/x, g(a)/y, f(g(a))/z}

谓词逻辑法

主要优点

- ✓ 自然：一阶谓词逻辑是一种接近于自然语言的形式语言系统，谓词逻辑表示法接近于人们对问题的直观理解
- ✓ 明确：有一种标准的知识解释方法，因此用这种方法表示的知识明确、易于理解
- ✓ 精确：谓词逻辑的真值只有“真”与“假”，其表示、推理都是精确的
- ✓ 灵活：知识和处理知识的程序是分开的，无须考虑处理知识的细节
- ✓ 模块化：知识之间相对独立，这种模块性使得添加、删除、修改知识比较容易进行

谓词逻辑法

主要缺点

- ✓ 知识表示能力差：只能表示确定性知识，而不能表示非确定性知识、过程性知识和启发式知识
- ✓ 知识库管理困难：缺乏知识的组织原则，知识库管理比较困难
- ✓ 存在组合爆炸：由于难以表示启发式知识，因此只能盲目地使用推理规则，这样当系统知识量较大时，容易发生组合爆炸
- ✓ 系统效率低：它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，往往使推理过程冗长，降低了系统效率

目录

- 关于知识表示
- 状态空间法
- 问题归约法
- 谓词逻辑法
- 语义网络法
- 其他方法
- 小结

语义网络表示法

语义网络的提出及基本思想

- 1968年J.R.Quillian在其博士论文中最先提出语义网络，把它作为人类联想记忆的一个显式心理学模型，并在他设计的可教式语言理解器TLC (Teachable Language Comprehenden)中用作知识表示方法。
- 语义网络的基本思想：在这种网络中，用“节点”代替概念，用节点间的“连接弧”(称为联想弧)代替概念之间的关系，因此，语义网络又称联想网络。它在形式上是一个带标识的有向图。由于所有的概念节点均通过联想弧彼此相连，Quillian希望他的语义网络能用于知识推导。

语义网络的概念

- 语义网络中的节点：表示各种事物、概念、情况、属性、动作、状态等，每个节点可以带有若干属性，一般用框架或元组表示。此外，节点还可以是一个语义子网络，形成一个多层次的嵌套结构。
- 语义网络中的弧：表示各种语义联系，指明它所连接的节点间某种语义关系。
- 节点和弧都必须带有标识，以便区分各种不同对象以及对象间各种不同的语义联系。最简单的语义网络是一个三元组：

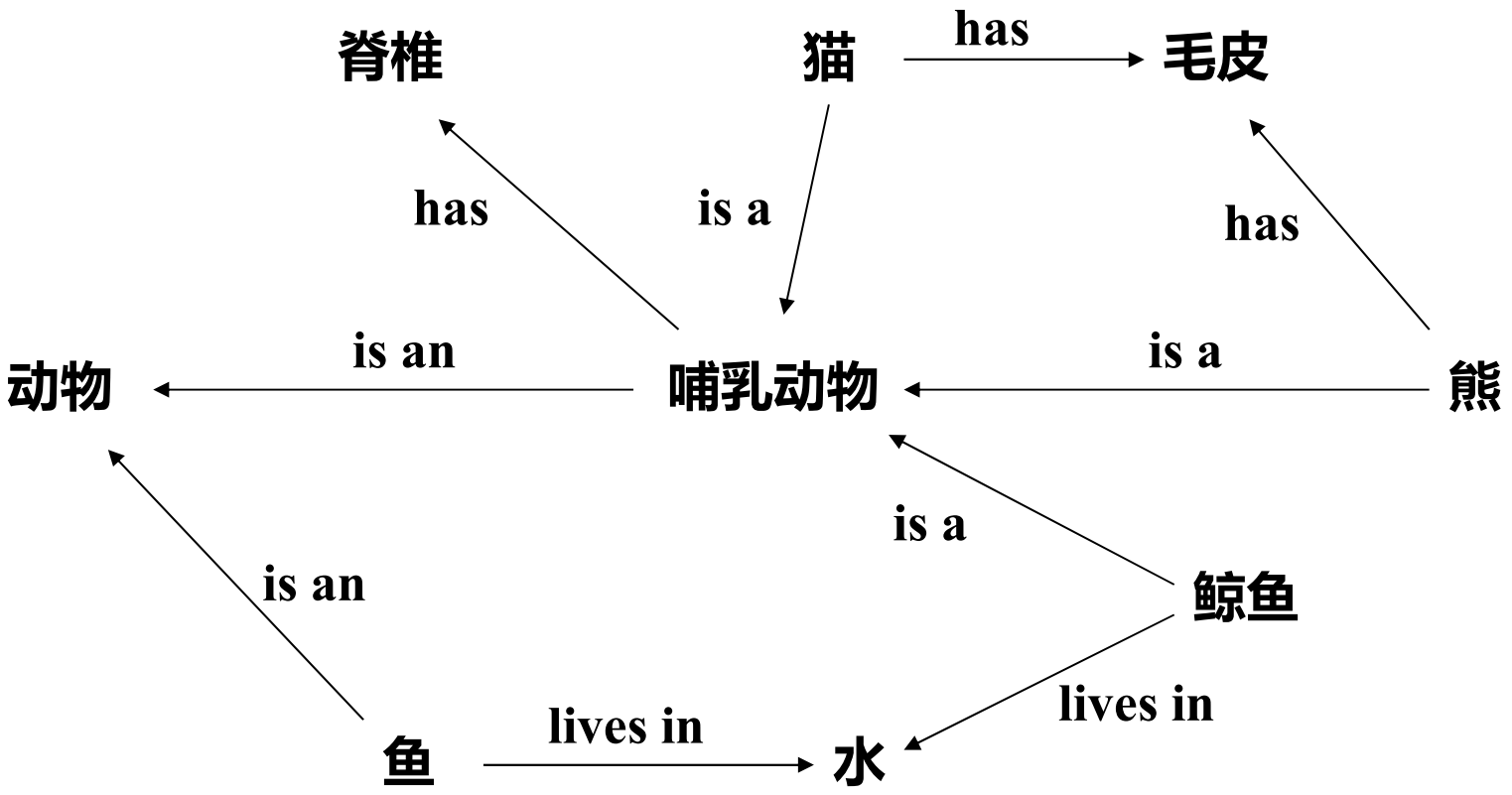
（节点1，弧，节点2）

语义网络表示

语义网络

- 它是表示概念（**concepts**）间语义关系的网络；
- 它是知识表示（**Knowledge Representation**，缩写**KR**）的一种形式；
- 它是一个有向（或无向）图，顶点（**vertices**）表示概念，弧（或边）表示语义关系。

语义网络的示例（http://en.wikipedia.org/wiki/Image:Semantic_Net.svg）

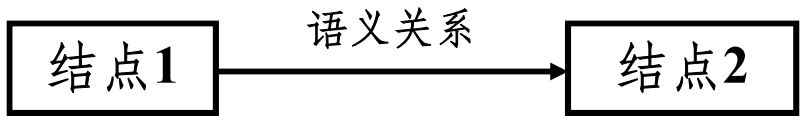


语义网络的历史

- 1956年，剑桥语言研究所Richard H. Richens在研究自然语言机器翻译时发明了“语义网”（Semantic Nets）；
- 60年代初，Robert F. Simmons对其进行了发展；
- 60年代末，Allan M. Collins和M.R. Quillian提出了后来最为流行的语义网络模型；
- 60至80年代，在超文本（hypertext）系统（如Web）中使用了语义链接的思想，该思想影响深远。

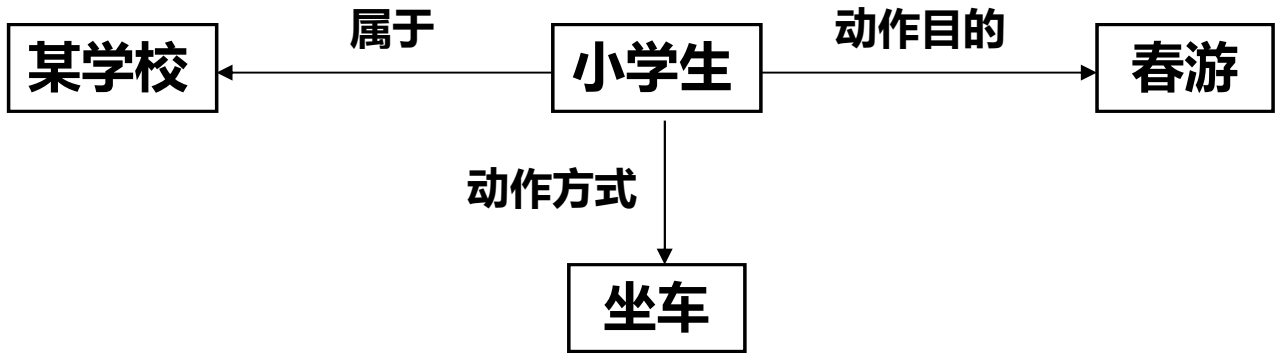
语义网络的结构

- **语义基元**
  - 最基本的语义单元
  - 三元组形式，（结点1，弧，结点2）
  - 有向图形式



  - 结点：实体（事物、概念、情况、属性、状态、事件、动作等）
  - 弧：方向代表实体间的主从关系，即结点1为主，结点2为辅；标注说明语义关系。

- 多个语义基元通过语义联系关联在一起，形成语义网络
- “小学生坐车去春游”



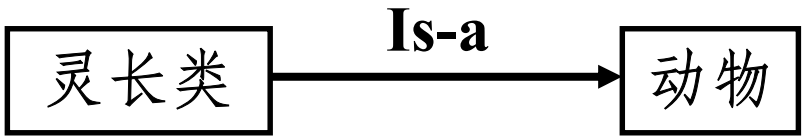
- 语义网络不仅可表示事物的属性、状态等，更适合表示事物间的关系和联系。

语义网络属于非逻辑表示，但可以进行逻辑分析

 **(Li, Friend, Wang)**	语义基元的有向图形式 语义基元的三元组形式
Friend(Li, Wang)	二元谓词

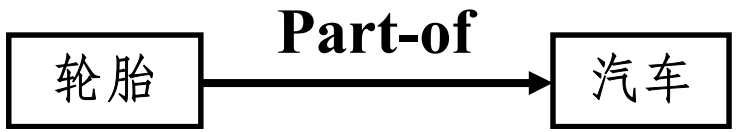
基本的语义关系

- **Is-a**
  - 表示一个事物是另一个事物的实例；
  - 具体和抽象；
  - 有继承性。



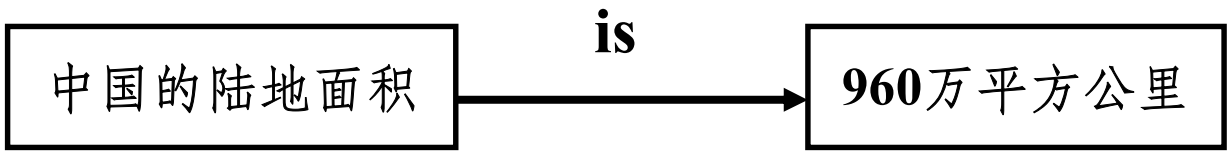
基本的语义关系（续）

- **Part-of**
  - 表示一个事物是另一个事物的一部分；
  - 部分与整体；
  - 无继承性。



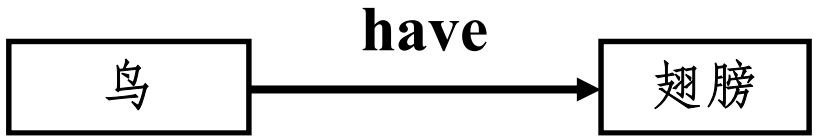
基本的语义关系（续）

- **is**
  - 一个事物是另一个事物的属性



基本的语义关系（续）

- **Have**
  - 事物与属性的拥有关系。



基本的语义关系（续）

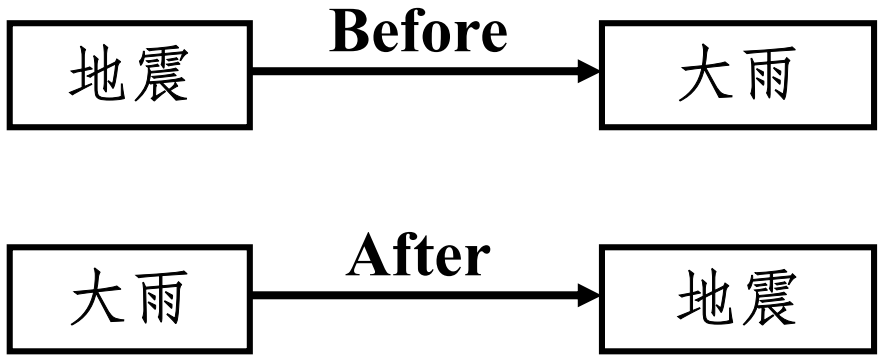
- **A-Kind-of**
  - 表示一个事物是另一个事物的一种类型；
  - 隶属关系；
  - 有继承性；
  - 下层结点可细化、补充、变异上层结点的属性。

基本的语义关系（续）

- **Can**
  - 表示一个结点能对另一个结点做某种事情。

基本的语义关系（续）

- 时间关系
  - Before
  - After

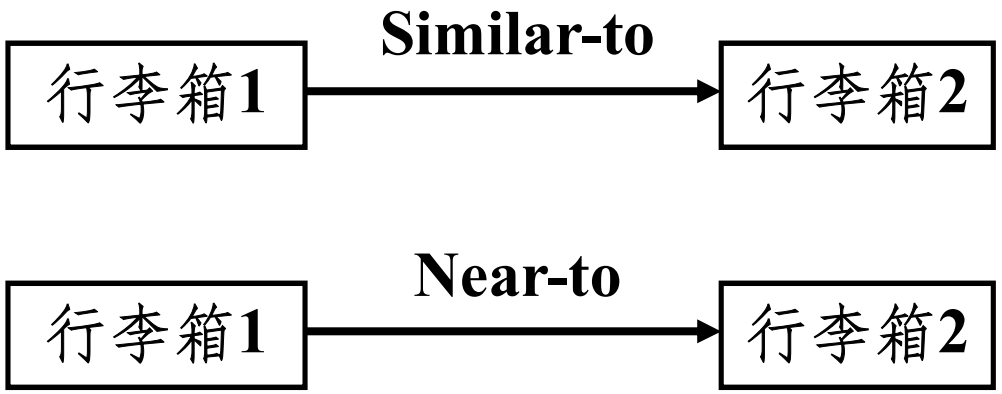


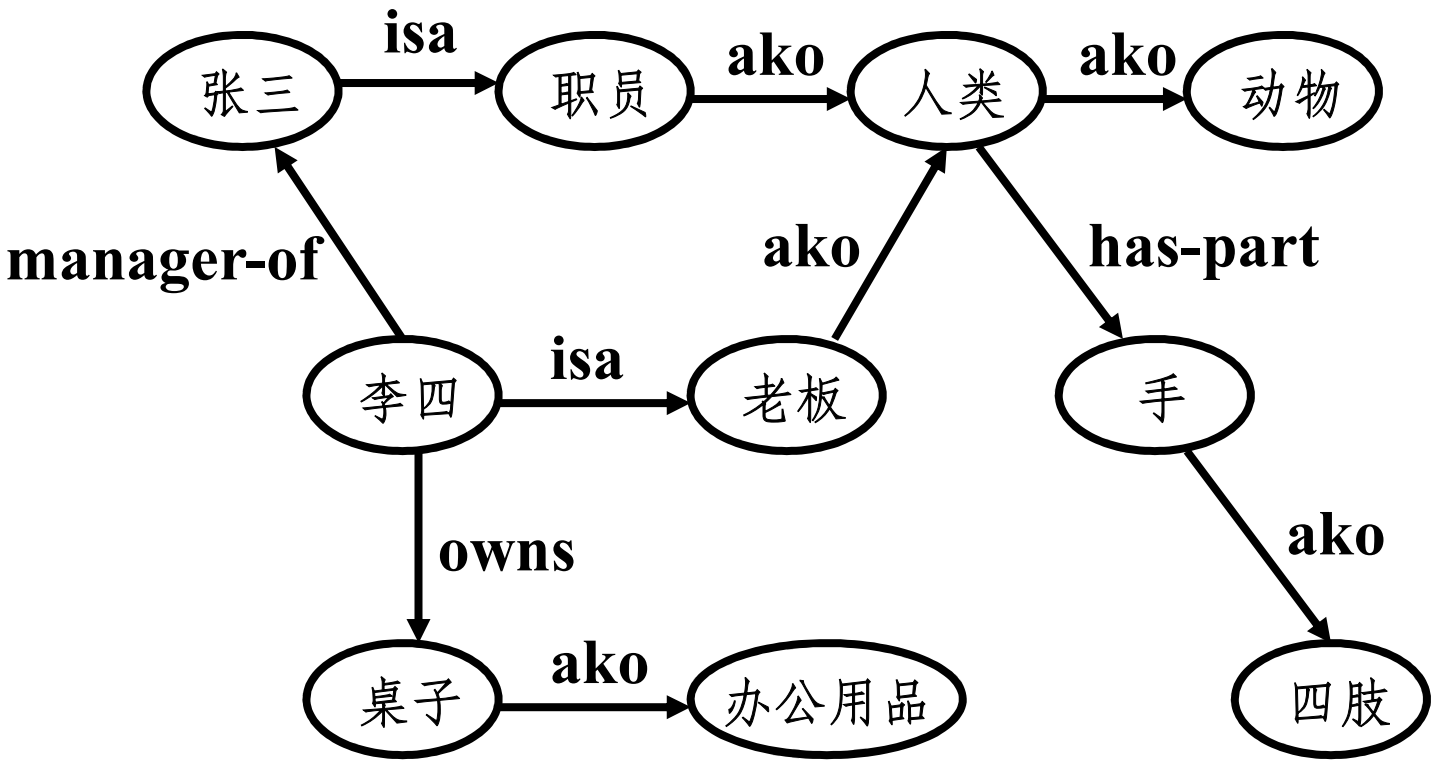
基本的语义关系（续）

- 位置关系
  - Located-on
  - Located-at
  - Located-under
  - Located-inside
  - Located-outside

基本的语义关系（续）

- 相近关系
  - Similar-to
  - Near-to

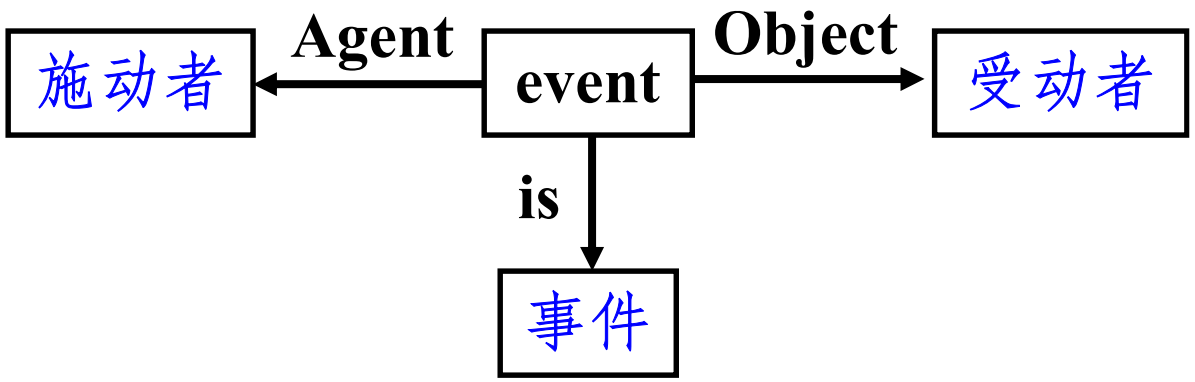



张三
isa
职员
ako
人类
ako
动物
manager-of
ako
has-part
李四
isa
老板
手
owns
ako
桌子
ako
办公用品
四肢

一个语义网络实例

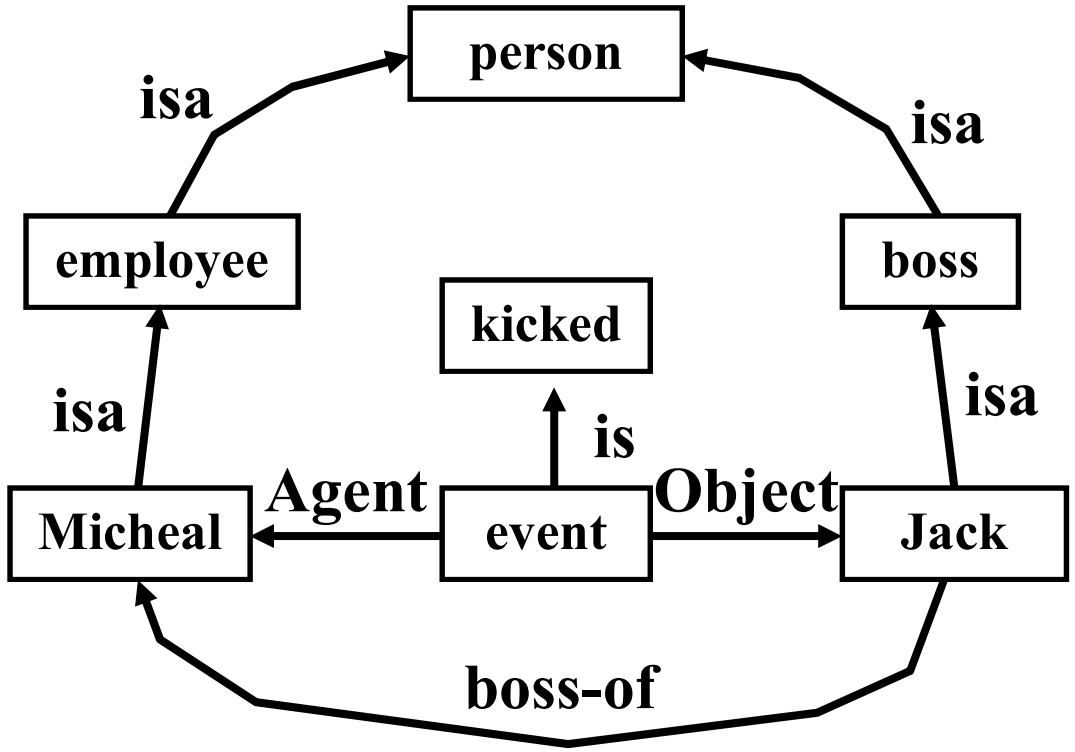
4.3.2 基本的语义关系（续）

- **event**
  - 对现实世界中“事件”的表示。



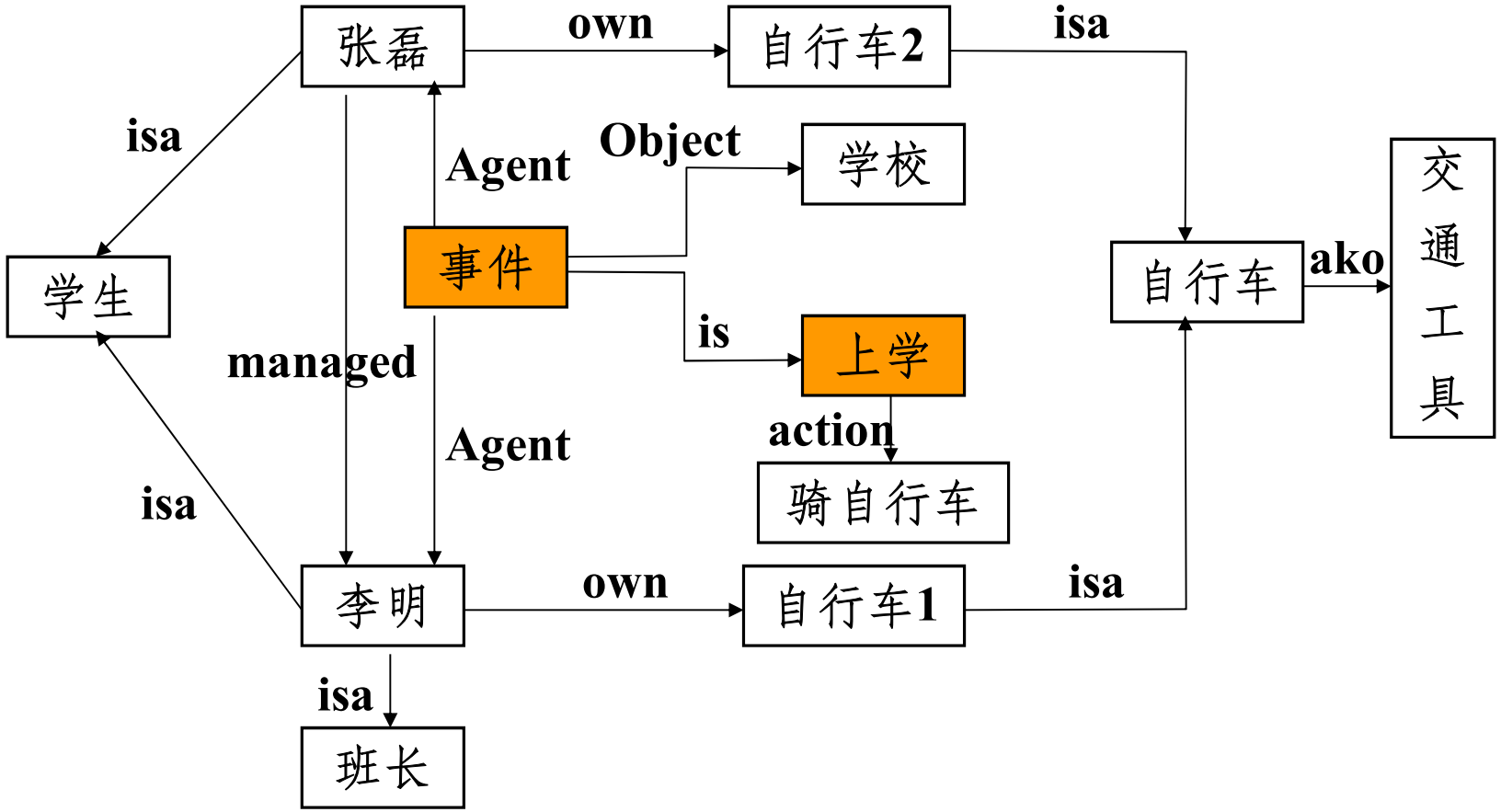
事件的语义网络实例

- 例：Micheal is an employee and Jack is his boss. Someday Micheal kicked his boss.



两个事件关联的语义网络实例

张磊每天都和班长李明一起骑车上学



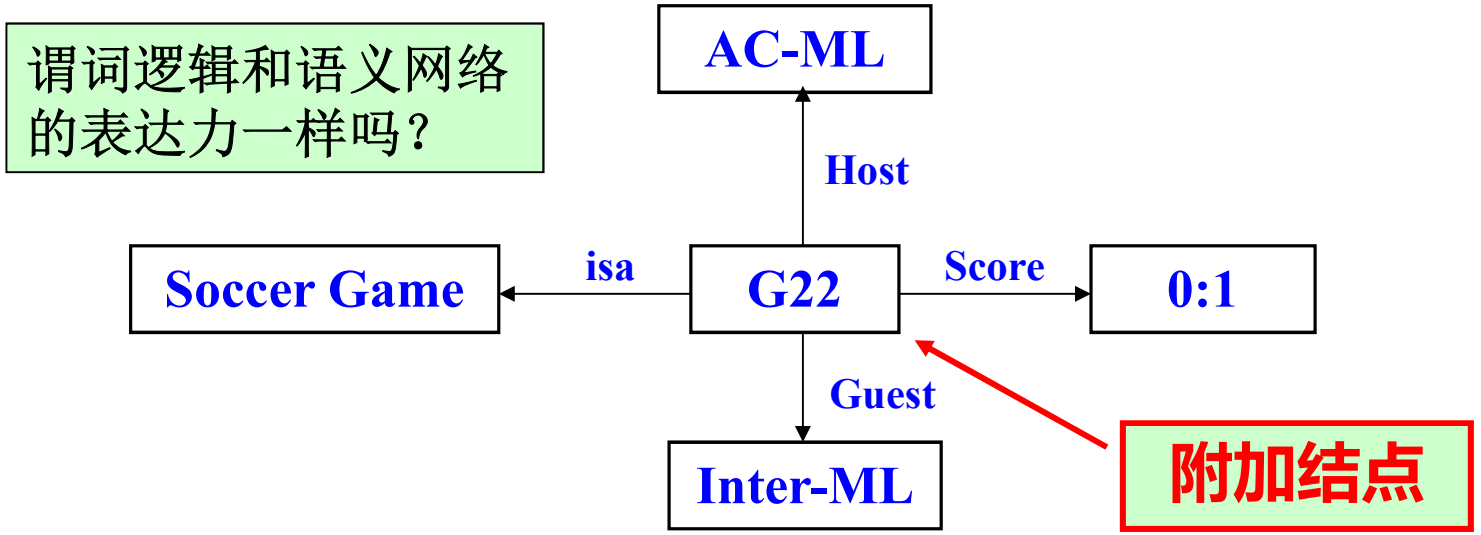
谓词逻辑转化为语义网络

- **一元谓词的转化**

苹果(水果)

- **多元谓词的转化**

SCORE(AC-MILAN, INTER-MILAN, 0:1)



语义网络的推理

- **逻辑表示时，用归结原理；**
- **产生式表示时，用推理机；**
- **语义网络表示时，怎么推理呢?**
  - 引入逻辑含义，利用归结法推理；
  - **1977**年，**Hendrix**提出网络分块，降低求解复杂度；
  - 将网络看作自动机，通过寻找会合点求解；
  - 总体上，没有完善的推理方法。

语义网络的推理Cont.

- **在语义网络上，推理主要通过继承和匹配。**
- **继承**
  - 事物的描述（即属性）从抽象（概念、类）结点传递到具体（实例）结点；
  - 具有继承关系的语义联系有：
    - **Is-a**
    - **A-kind-of**

继承的一般过程

1. 建立结点表，开始只包含待求结点；
2. 检查表中第一个结点；
   ① 若有继承弧，将弧所指的结点放入结点表（末尾），并记录这些结点的属性值；
   ② 删除第一个结点。
3. 如果表空，记录的属性值就是X结点的所有属性，结束推理；否则，转2。

基本的继承关系

- **If X(AKO)Y and Y(AKO)Z Then X(AKO)Z**

  如果麻雀是一种鸟，且鸟是一种动物，则麻雀是一种动物

- **If X(ISA)Y and Y(AKO)Z Then X(ISA)Z**

  如果麻雀1是只麻雀，且麻雀是一种鸟，则麻雀1是一只鸟

- **If X(AKO)Y and Y(属性)Z Then X(属性)Z**

  如果麻雀是一种鸟，且鸟有翅膀，则麻雀有翅膀

基本的继承关系（续）

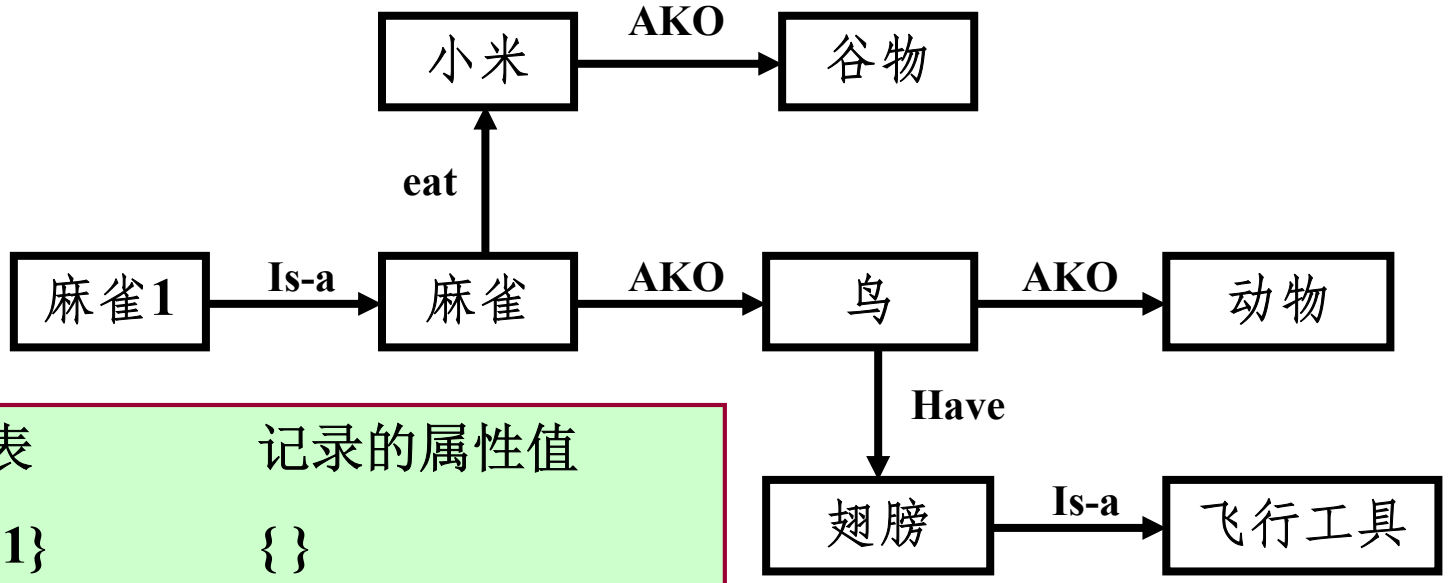
- **If X(ISA)Y and Y(属性)Z Then X(属性)Z**

  如果麻雀1是只麻雀，且麻雀吃小米，则麻雀1吃小米

- **If X(属性)Y and Y(ISA)Z Then X(属性)Z**

  如果鸟有翅膀，且翅膀是个飞行工具，则鸟有飞行工具

“继承”推理的实例



结点表	记录的属性值
{麻雀1}	{ }
{麻雀}	{吃小米}
{鸟}	{吃小米，有翅膀}
{动物}	{吃小米，有翅膀}
{ }	{吃小米，有翅膀}

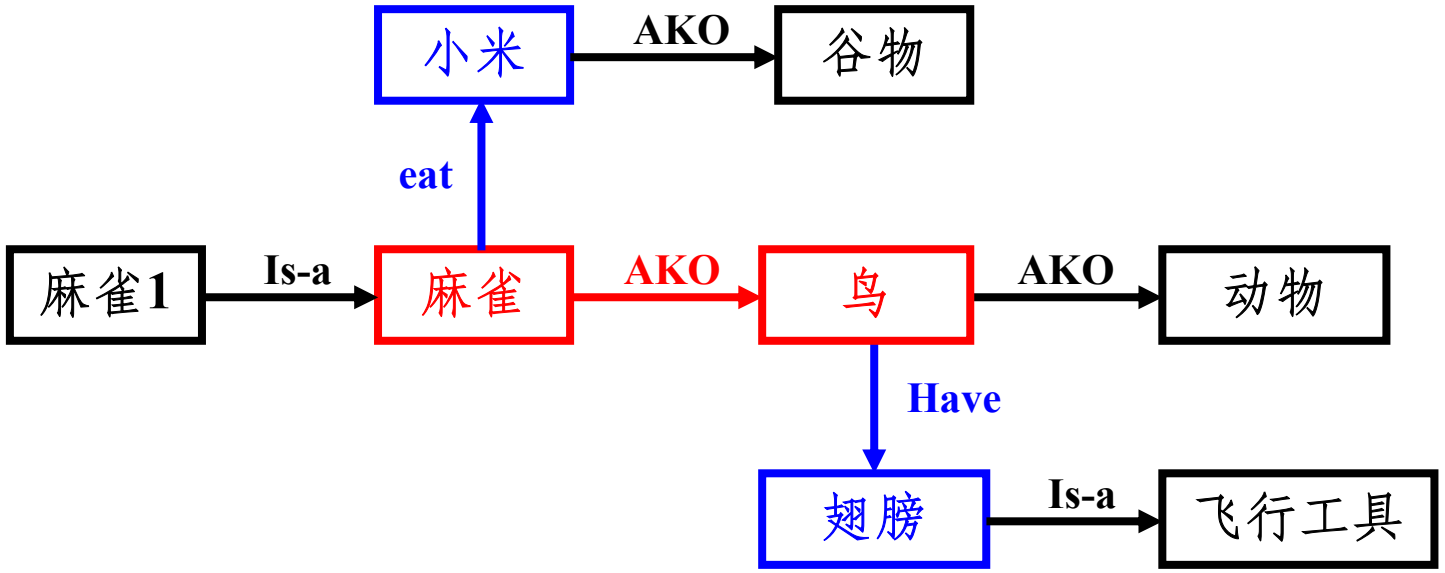
基于匹配的推理

匹配的过程

- 根据问题构造网络片断，片段中空的结点或弧代表待求解的问题。
- 在知识库（语义网络）中，寻找匹配网络片断的部分，从而确定空的结点或弧。

匹配的例子

- “已知麻雀是一种鸟，求麻雀的特点”



语义网络表示法的特点

- **优点**
  - 结构化知识表示；
  - 基于弧的信息查找，而不是搜索整个知识库；
  - 直观，自然语言到语义网络的转换比较容易。
- **缺点**
  - 没有形式化的含义（**formalized meaning**），推理规则不明了；
  - 不适合表示复杂知识，大型语义网络的推理很难进行。

目录

- 关于知识表示
- 状态空间法
- 问题归约法
- 谓词逻辑法
- 语义网络法
- 其他方法
- 小结

其他知识表示方法（自学）

- 框架表示法
- 剧本表示法
- 过程表示法
- ……

框架表示法

- **框架表示法：**框架表示法是在框架理论的基础上发展起来的一种结构化知识表示方法。
- **框架理论：**
  - ✓ 框架理论是明斯基于1975年作为理解视觉、自然语言对话及其它复杂行为的一种基础提出来的。
  - ✓ 框架理论认为，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。

框架表示法

框架理论：

- 框架：是人们认识事物的一种通用的数据结构形式。即当新情况发生时，人们只要把新的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)，这样的通用数据结构就称为框架。
- 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。
- 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便可形成一个框架系统。
- 框架系统推理：由框架之间的协调来完成。

框架表示法

□ 框架结构：

<框架名>

槽名1：　侧面名1_1　　值1_{11}，值1_{12}，……

　　侧面名1_2　　值1_{21}，值1_{22}，……

　　……

槽名2：　侧面名2_1　　值2_{11}，值2_{12}，……

　　侧面名2_2　　值2_{21}，值2_{22}，……

　　……

……

槽名n：　侧面名n_1　　值n_{11}，值n_{12}，……

　　侧面名n_2　　值n_{21}，值n_{22}，……

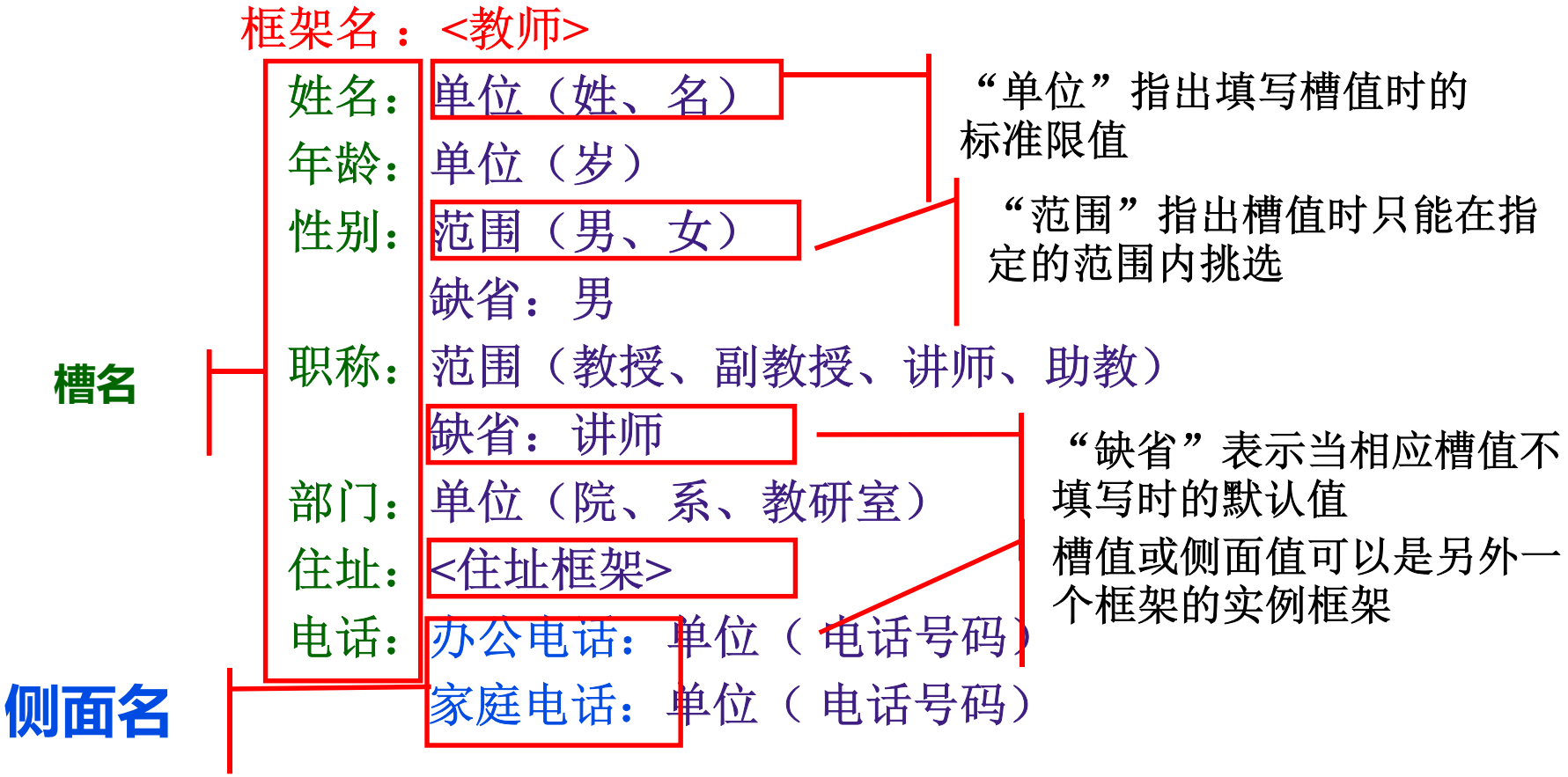
　　……

框架表示法

- **框架结构：**
  - ✓ 每个框架都有框架名，代表某一类对象
  - ✓ 一个框架由若干个槽（项目）组成，用于表示对象的某个方面的属性
  - ✓ 有时一个槽（属性）还要从不同的侧面来描述，每个侧面可具有一个或多个值。

框架表示法

□ 框架表示法的例子：一个描述教师的框架

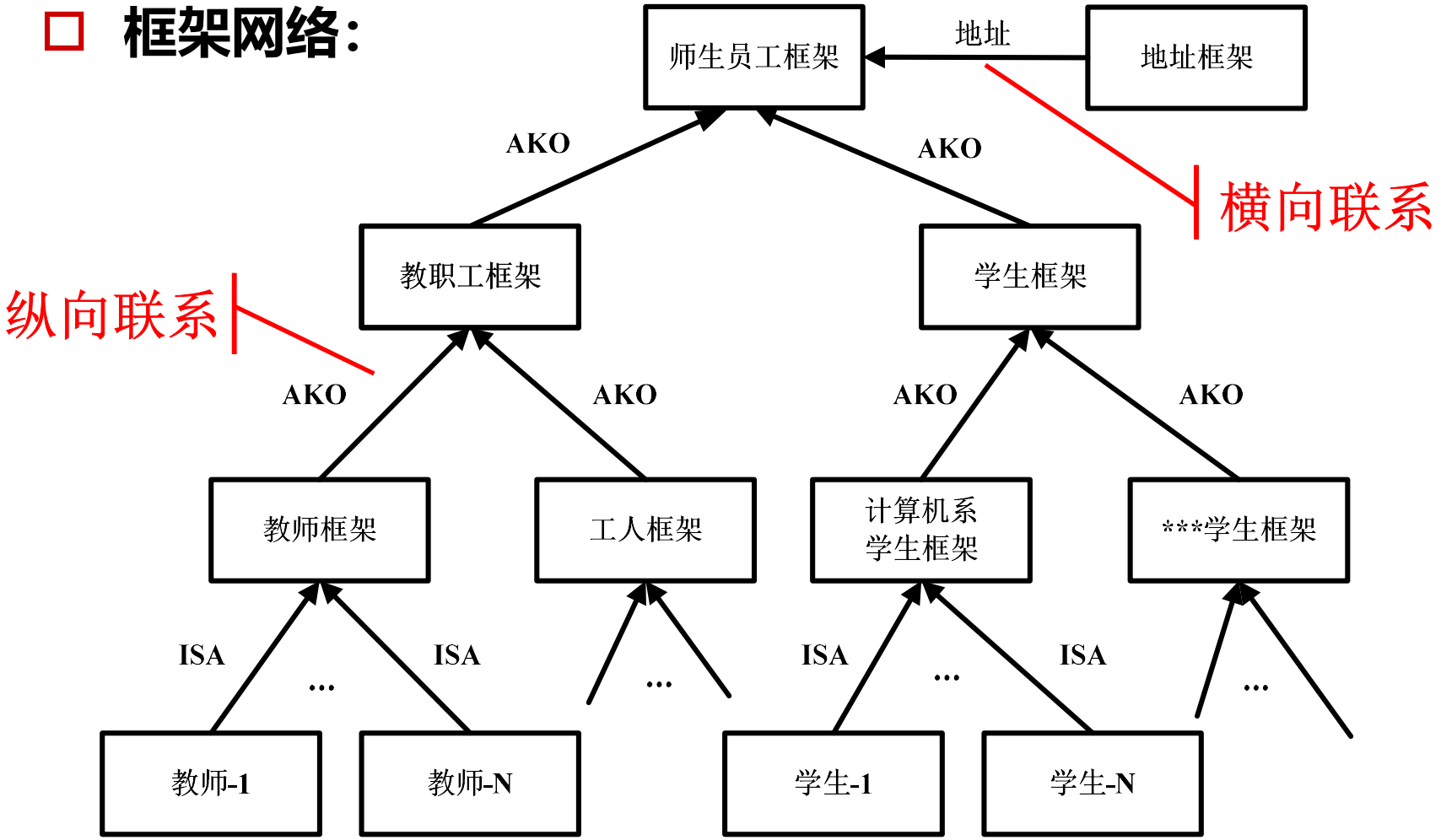


框架表示法

- **框架网络：**
  - ✓ 当知识比较复杂时，往往需要通过多个框架之间的横向或纵向联系形成一种框架网络。
- **框架之间的纵向联系：**
  - ✓ 是指那种具有继承关系的上下层框架之间的联系。框架之间的纵向联系通过预定义槽名**AKO**或**ISA**等来实现。
- **框架之间的横向联系**
  - ✓ 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。

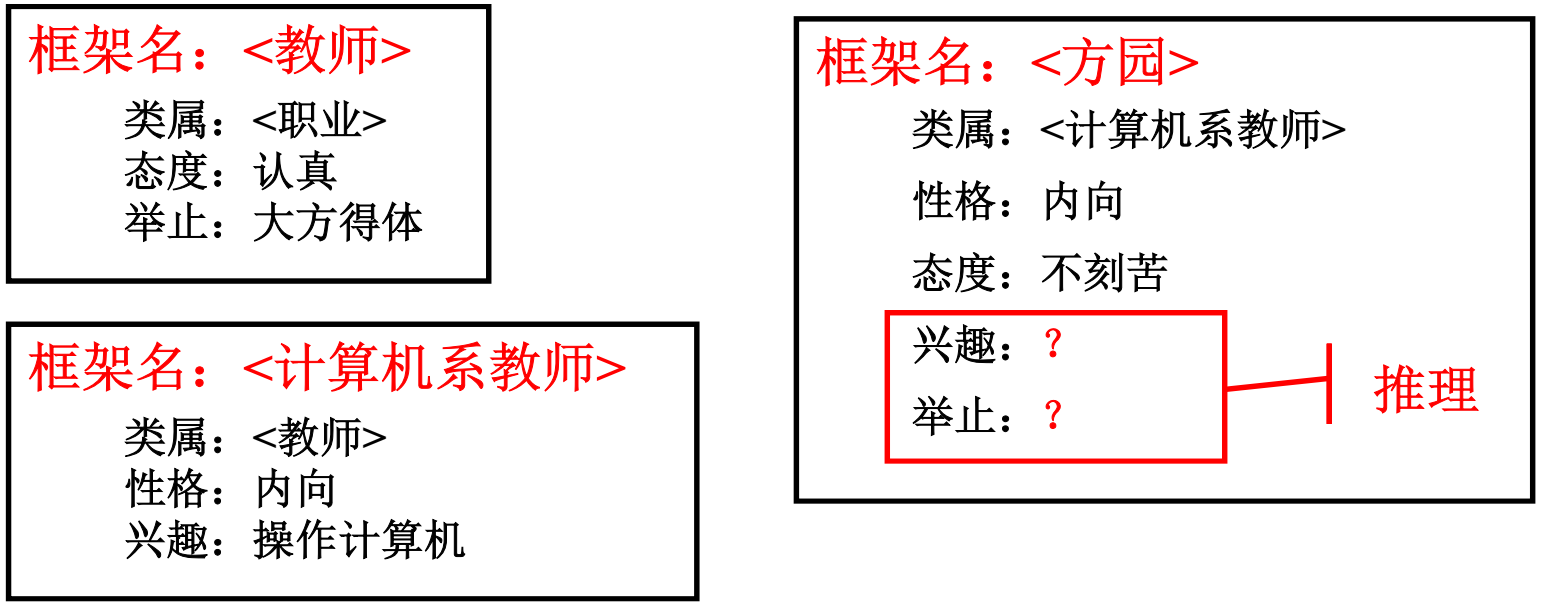
框架表示法

- 框架网络：


师生员工框架
地址
地址框架
AKO
AKO
横向联系
教职工框架
学生框架
纵向联系
AKO
AKO
AKO
AKO
教师框架
工人框架
计算机系
学生框架
***学生框架
ISA
ISA
ISA
ISA
教师-1
教师-N
学生-1
学生-N

框架表示法

- **框架网络的例子：** 一般来讲，教师的工作态度是认真，行为举止大方得体，计算机系教师一般来讲性格内向，喜欢操作计算机。方园是计算机系教师，他性格内向，但工作不刻苦。问他的兴趣和举止如何？
- **框架表示为：**



框架表示法

- **框架的推理：**在框架网络中，问题求解主要是通过对框架的继承与匹配来实现的。
- **继承**
  - ✓ 下层框架从上层框架继承相关属性、属性值、条件
- **匹配**
  - ✓ 框架通常只能与现实做到部分匹配，完全匹配是一个特殊情况。因为框架是对一类事物的完整或典型的描述，待匹配的具体个体不可能做到完全一致。
  - ✓ 不匹配的情况：某个属性不存在，或与规定的属性值不符，或属性类型不符

框架表示法

框架表示法的优点：

- ✓ 结构性：最突出特点是善于表示结构性知识，它能够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：在框架网络中，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。

框架表示法

框架表示法的不足：

- ✓ 缺乏框架的形式理论：至今，还没有建立框架的形式理论，其推理和一致性检查机制并非基于良好定义的语义。
- ✓ 缺乏过程性知识表示：框架网络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，而这些规则在框架网络中又很难表达。
- ✓ 清晰性难以保证：由于各框架本身的数据结构不一定相同，从而框架网络的清晰性很难保证。

剧本表示法

剧本表示法：

- 剧本表示法（又称为脚本表示法）是夏克（**R. C. Schank**）依据他的概念依赖理论提出的一种知识表示方法，时间约在**1975**年。
- 概念依赖理论的基本思想：把人类生活中各类故事情节的基本概念抽取出来，构成一组原子概念，确定这些原子概念的相互依赖关系，然后把所有故事情节都用这组原子概念及其依赖关系表示出来。
- 剧本是框架的一种特殊形式，它用一组槽来描述某些事件的发生序列，就像剧本中的事件序列一样，故称为“剧本”表示法。

剧本表示法

剧本的构成

- ✓ 开场条件： 给出在剧本中描述的事件发生的前提条件。
- ✓ 角色 ：用来表示在剧本所描述的事件中可能出现的有关人物的一些槽。
- ✓ 道具： 这是用来表示在剧本所描述的事件中可能出现的有关物体的一些槽。
- ✓ 场景：描述事件发生的真实顺序，可以由多个场景组成，每个场景又可以是其它的剧本。
- ✓ 结果：给出在剧本所描述的事件发生以后通常所产生的结果。

剧本表示法

- **剧本表示法的例子：**餐厅剧本
  - ✓ 开场条件
    - (a)顾客饿了，需要进餐。
    - (b)顾客有足够的钱。
  - ✓ 角色
    - 顾客，服务员，厨师，老板。
  - ✓ 道具
    - 食品，桌子，菜单，钱。
  - ✓ 场景：5个场景

剧本表示法

- **剧本表示法的例子：**餐厅剧本
  - ✓ 场景：5个场景
    - 场景1：进入餐厅
      - (a) 顾客走入餐厅。
      - (b) 寻找桌子。
      - (c) 在桌子旁坐下
    - 场景2：点菜
      - (a) 服务员给顾客菜单。
      - (b) 顾客点菜。
      - (c) 顾客把菜单还给服务员。
      - (d) 顾客等待服务员送菜。

剧本表示法

- **剧本表示法的例子：**餐厅剧本
  - ✓ 场景：5个场景
    - 场景3：等待
      - (a) 服务员把顾客所点的菜告诉厨师。
      - (b) 厨师做菜。
    - 场景4：吃菜
      - (a) 厨师把做好的菜给服务员。
      - (b) 服务员给顾客送菜。
      - (c) 顾客吃菜。

剧本表示法

- **剧本表示法的例子：**餐厅剧本
  - ✓ 场景：5个场景
    - 场景5：离开
      - (a) 服务员拿来帐单。
      - (b) 顾客付钱给服务员。
      - (c) 顾客离开餐厅。
  - ✓ 结果
    - (a) 顾客吃了饭，不饿了。
    - (b) 顾客花了钱。
    - (c) 老板挣了钱。
    - (d) 餐厅食品少了。

剧本表示法

剧本的推理：

- ✓ 一旦剧本被启用，则可以应用它来进行推理。其中最重要的是运用剧本可以预测没有明显提及的事件的发生。
- ✓ 例如：对于以下情节：“昨晚，约翰到了餐厅。他点了牛排。当他要付款时发现钱已用光。因为开始下雨了，所以他赶紧回家了“。
- ✓ 推理：“昨晚，约翰吃饭了吗？”
- ✓ 虽然上面的情节中没有提到约翰吃没吃饭的问题，但借助于餐厅剧本，可以回答：他吃了。因为启用了餐厅剧本，情节中的所有事件与剧本中所预测的事件序列相对应，可以推断出整个事件正常进行时所得出的结果。

剧本表示法

剧本的推理：

- 但是，一旦一个典型的事件被中断，也就是给定情节中的某个事件与剧本中的事件不能对应时，则剧本便不能预测被中断以后的事件了。
- 例如：如下情节：“约翰走进餐厅。他被带到餐桌旁。点了一大块牛排之后，他坐在那儿等了许久。于是，他生气走了。”
- 该情节中，因为要久等，所以约翰走了，这一事件改变了餐厅脚本中所预测的事件序列，因而被中断了，这时就不能推断约翰是否付了帐等情节，但仍然可以推断出他看了菜单，这是因为看菜单事件发生在中断之前。

剧本表示法

剧本表示法的特点

- ✓ 剧本表示法与框架表示法相比，比较呆板，知识表达的范围也很窄。
- ✓ 人类日常的行为有各种各样，很难用一个剧本就理解各种各样的情节。
- ✓ 剧本表示法对于表达预先构思好的特定知识，如理解故事情节等，是非常有效的。
- ✓ 目前剧本表示法主要在自然语言理解方面获得了一些应用。

过程表示法

- ✓ **过程表示法的例子：**八数码问题
  - 我们用一个3×3的方格阵来表示该问题的一个状态，用a～i来标记这9个方格，如下图所示。

a	b	c
d	e	f
g	h	i

(a)状态描述

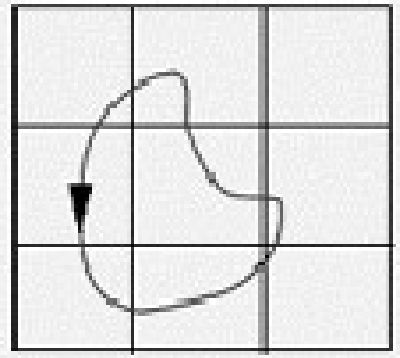
1	2	3
8		4
7	6	5

(b)目标状态

过程表示法

- ✓ **过程表示法的例子：** 八数码问题
- ✓ 当任意给定一初始状态后，求解该问题的过程如下：
  - (1)首先移动棋牌，使得棋子1和空格均不在位置c 上。
  - (2) 依次移动棋牌，使得空格位置沿下图(a)所示的箭头方向移动，直到棋子1位于a为止 。

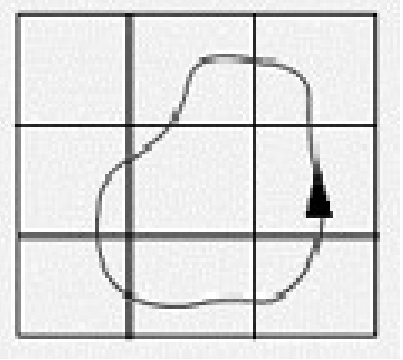
a	b	c
d	e	f
g	h	i



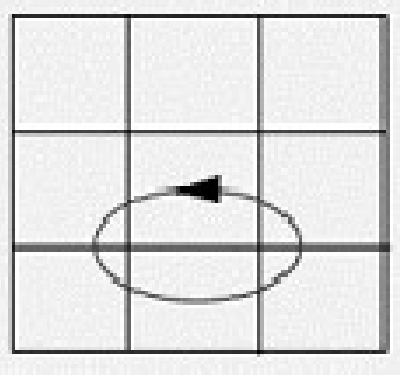
(a)

过程表示法

- ✓ **过程表示法的例子：八数码问题**
  - (3)依次移动棋牌，使得空格位置沿图(b)所示的箭头方向移动，直到数码2位于b为止。若这时刚好数码3在位置c，则转(6)。
  - (4) 依次移动棋牌，使得空格位置沿图(c)所示的箭头方向移动，直到数码3位于e为止。这时空格刚好在位置d。



(b)



(c)

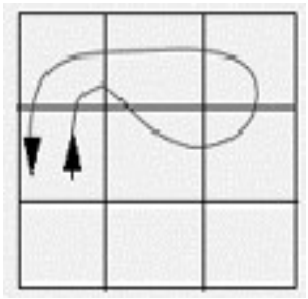
过程表示法

✓ 过程表示法的例子：八数码问题

- 经过以上4步，得到的状态如右图Step(4)所示。其中"×"表示除空格以外的任何棋牌。
- (5)依次移动棋牌，使得空格位置沿图(d)所示的箭头方向移动，直到空格又回到了d为止。此时状态如右图Step(5)所示。

1	2	×
	3	×
×	×	×

Step(4)



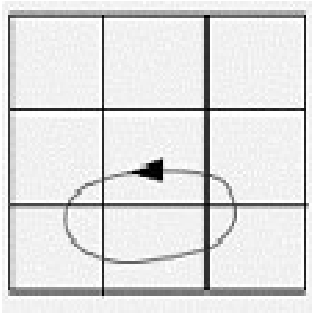
(d)

1	2	3
	×	×
×	×	×

Step(5)

过程表示法

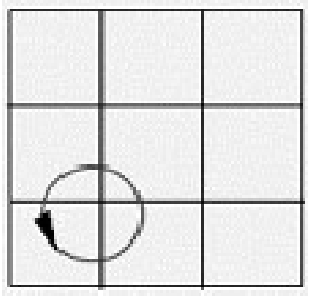
- ✓ **过程表示法的例子：八数码问题**
  - (6)依次移动棋牌，使得空格位置沿图 (e)所示的箭头方向移动，直到数码4在位置f为止，如图Step(6) 中所示。若这时刚好数码5在位置i则转(9)。
  - (7)依次移动棋牌，使得空格位置沿图(f)所示的箭头方向移动，直到数码5位于e为止。这时空格刚好在位置d。



(e)

1	2	3
	×	4
×	×	×

Step(6)



(f)

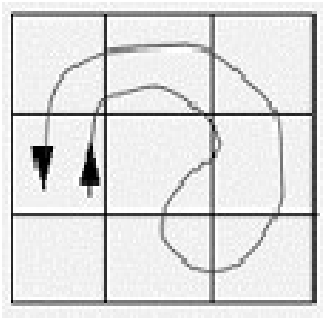
1	2	3
	5	4
×	×	×

Step(7)

过程表示法

✓ 过程表示法的例子：八数码问题

- (8)依次移动棋牌，使得空格位置沿图(g)所示的箭头方向移动，直到空格回到位置d为止，如图Step(8)。

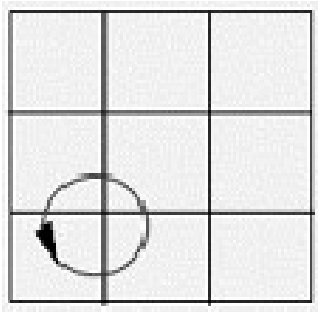


(g)

1	2	3
	×	4
×	×	5

Step(8)

- (9)依次移动棋牌，使得空格位置沿图(h)所示的箭头方向移动，直到数码6在位置h为止，若这时数码7、8分别在位置g和d，则问题得解，否则，说明由所给初始状态达不到所要求的目标状态。



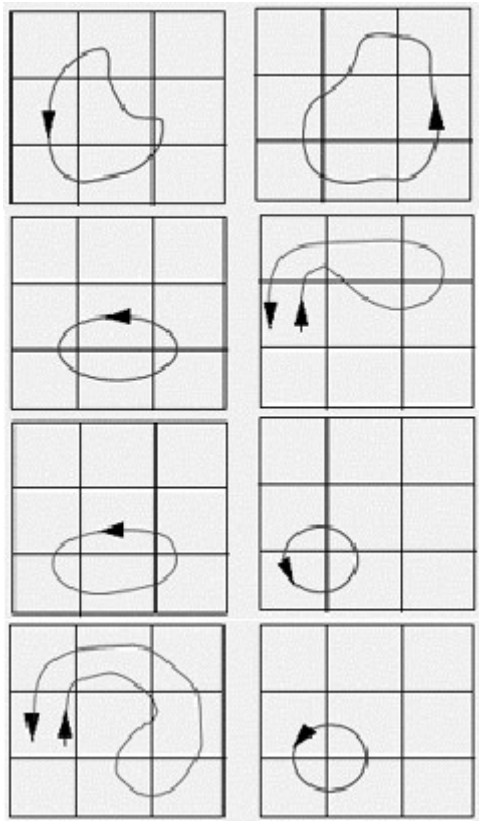
(h)

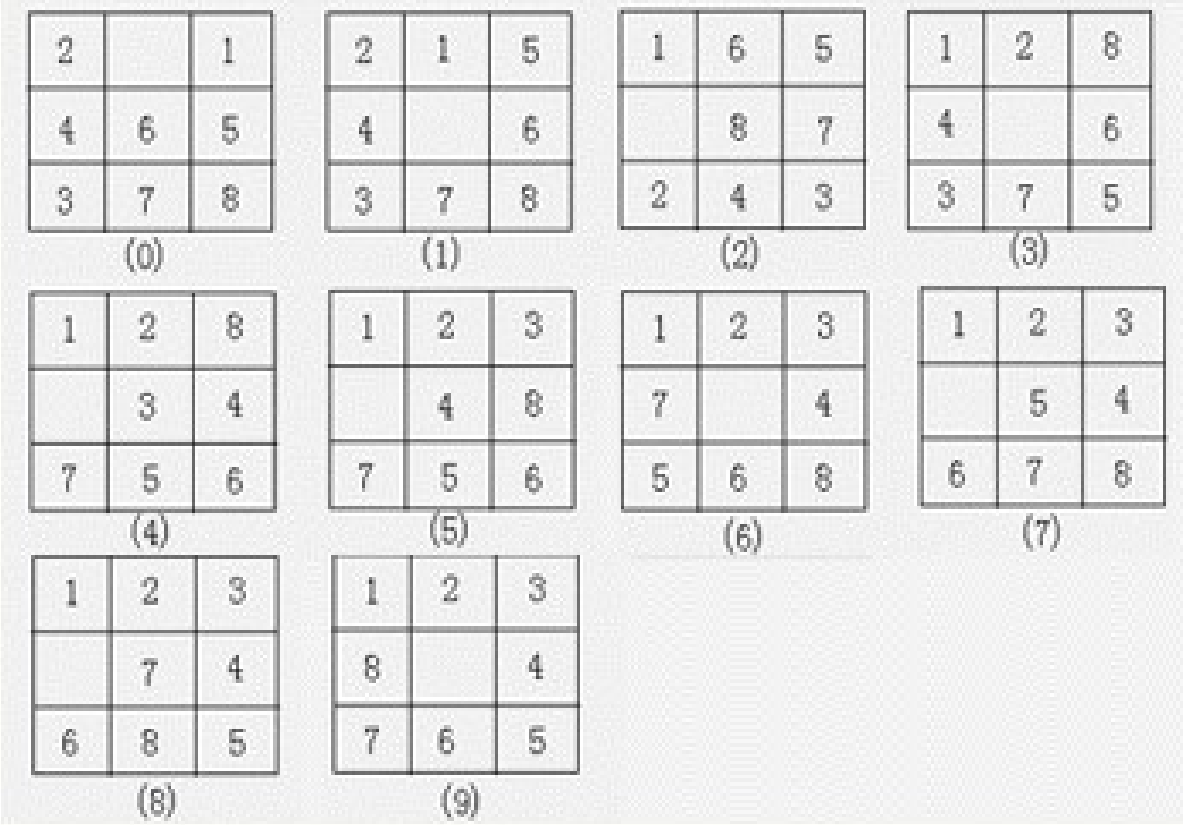
1	2	3
×		4
×	6	5

Step(9)

过程表示法

- ✓ **过程表示法的例子：**八数码问题





过程表示法

过程表示法的优点：

- ✓ 表示效率高：过程表示法是用程序来表示知识的，而程序能准确的表明先做什么，后作什么以及怎样做，并直接嵌入一些启发式的控制信息，因此，可以避免选择及匹配那些无关的知识，也不需要跟踪那些不必要的路径，从而提高了系统的运行效率。
- ✓ 控制系统容易实现：由于控制性质是已嵌入到程序中，因而控制系统就比较容易设计。

过程表示法的不足：

- ✓ 不易修改及添加新知识，而且当对某一过程进行修改时，又可能影响到其它过程，对系统的维护带来不便。

小结

- 本章所讨论的知识表示问题是人工智能研究的核心问题之一。知识表示方法很多，本章介绍了其中的7种，有图示法和公式法，陈述式表示和过程式表示等。
- **状态空间法**：是一种基于解答空间的问题表示和求解方法，它是以状态和操作符为基础的。在利用状态空间图表示时，我们从某个初始状态开始，每次加一个操作符，递增地建立起操作符的试验序列，直到达到目标状态为止。由于状态空间法需要扩展过多的节点，容易出现"组合爆炸"，因而只适用于表示比较简单的问题。

小结

- **问题归约法**：从目标(要解决的问题)出发，逆向推理，通过一系列变换把初始问题变换为子问题集合和子-子问题集合，直至最后归约为一个平凡的本原问题集合。这些本原问题的解可以直接得到从而解决了初始问题，用与或图来有效地说明 问题归约法的求解途径。
- **谓词逻辑法**：采用谓词合式公式和一阶谓词演算把要解决的问题变为一个有待证明的问题，然后采用消解定理和消解反演来证明一个新语句是从已知的正确语句导出的，从而证明这个新语句也是正确的。

小结

- **语义网络：**是知识的一种图解表示，它由节点和弧线或链线组成。节点用于表示实体、概念和情况等，弧线用于表示节点间的关系。
- **框架：**是一种结构化表示方法。框架通常由指定事物各个方面的槽组成，每个槽拥有若干个侧面，而每个侧面又可拥有若干个值。大多数实用系统必须同时使用许多框架，并可把它们联成一个框架系统。
- **剧本表示法：**是框架的一种特殊形式，它使用一组槽来描述事件的发生序列。剧本表示特别适用于描述顺序性动作或事件，但使用不如框架灵活，因此应用范围也不如框架那么广泛。

小结

- **过程表示法**：是一种知识的过程式表示，它将某一有关问题领域知识同这些使用方法一起，隐式地表示为一个问题求解过程。过程表示用程序来描述问题，具有很高的问题求解效率。由于知识隐含在程序中难以操作，所以适用范围较窄。