

# 人工智能

## 第2章 知识表示方法

## Methodologies of Knowledge Representation

# 主要知识表示方法 Basic Methodologies

- ✿ 状态空间 (State Space)
- ✿ 问题归约 (Problem Reduction)
- ✿ 谓词逻辑 (Predicate Logic)
- ✿ 语义网络 (Semantic Network)
- ✿ 本体 (Ontology)
- ✿ 真体 (Agent)
- ✿ 框架, 剧本, 过程 (Frame, Script, Procedure)

## Problem Solving

= Knowledge Representation + Search

# Main Contents

## 第2章 知识表示方法

- 2.1 状态空间表示
- 2.2 问题归约表示
- 2.3 谓词逻辑表示
- 2.4 语义网络表示
- 2.5 框架表示
- 2.6 本体技术
- 2.7 过程表示
- 2.8 小结

## 2.1 状态空间表示

### State Space Representation

- ❁ 问题求解技术主要是两个方面：
  - ❁ 问题的表示
  - ❁ 求解的方法
- ❁ 状态空间表示
  - ❁ 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

## 15 Puzzle Problem (15数码难题)

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

初始棋局

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

目标棋局

- ❖ 状态：棋局
- ❖ 算符： $15 \times 4 = 60$ 个？  
移动空格4个？
- ❖ 求解方法：从初始棋局开始，试探由每一合法走步得到的各种新棋局，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。  
尝试各种不同的走步，直到偶然得到该目标棋局为止。  
这种尝试本质上涉及某种试探搜索。

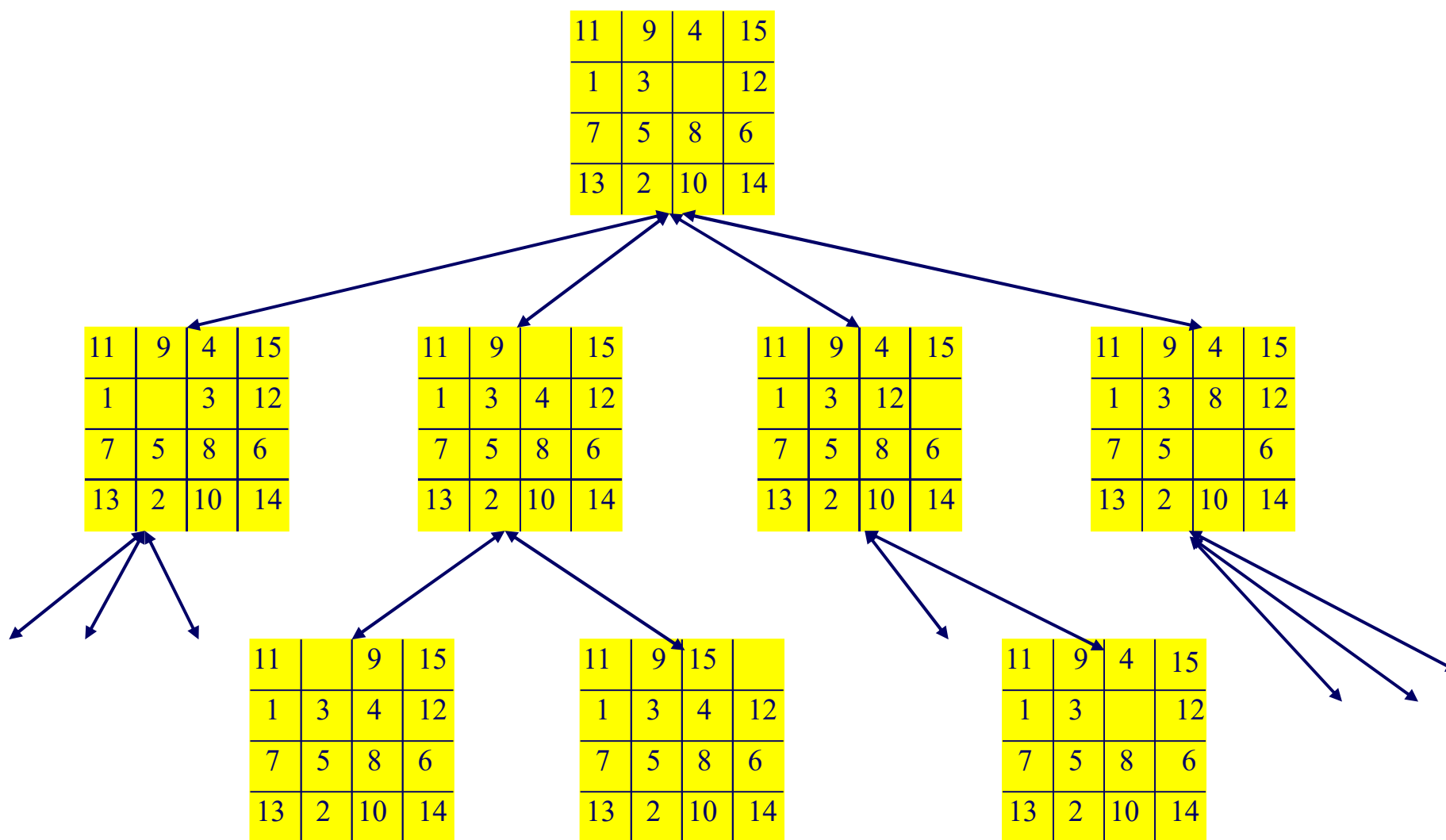


图2.2 十五数码难题部分状态空间图



# 月历魔柱

例子：2013年4月

SUN	MON	TUE	WED	THU	FRI	SAT
0	1	2	3	4	5	6
7	8	9	10	11	12	13
14	15	16	17	18	19	20
21	22	23	24	25	26	27
28	29	30	31			

# 问题一

你是否转过魔方？8数码  
难题应如何移动求解？





## 2.1.2 状态图示法



⊕ 节点

⊕ 弧线

⊕ 有向图

一对节点用弧线连接起来，从一个节点指向另一个节点, 这种图叫做有向图。

## ✚ 路径

某个节点序列  $(n_{i1}, n_{i2}, \dots, n_{ik})$  当  $j = 2, 3, \dots, k$  时, 如果对于每一个  $n_{i, j-1}$  都有一个后继节点  $n_{i, j}$  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(cost)。两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。

## ✚ 图的显式说明

对于显式说明，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。

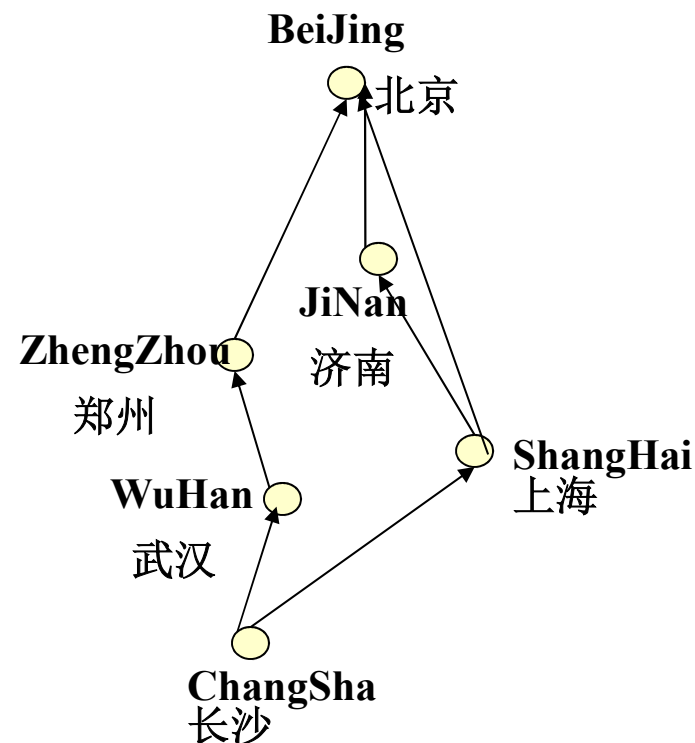
## ✚ 图的隐式说明

节点的无限集合 $\{S\}$ 作为起始节点是已知的。后继节点算符 $\Gamma$ 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。

# 状态空间表示举例

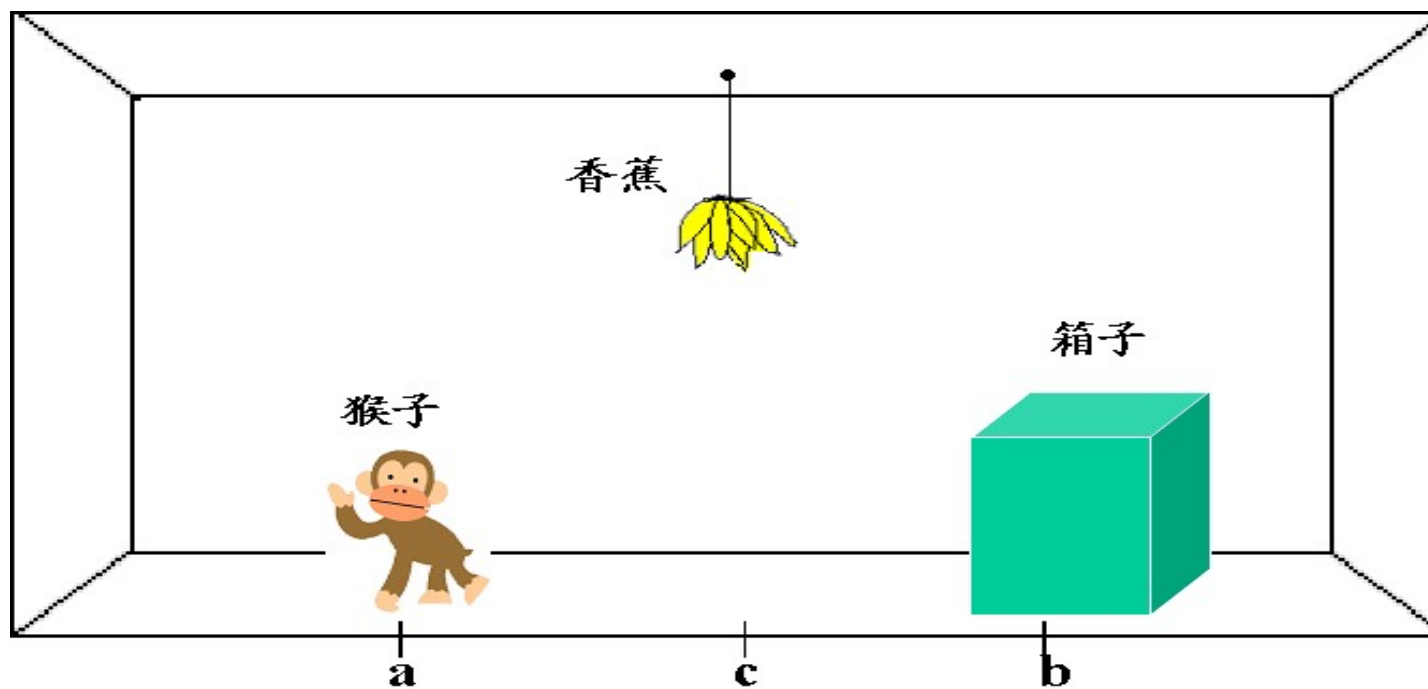
## 例1 路线规划

- ⊕ Initial state
  - ⊞ The journey starting from Changsha
- ⊕ Operators
  - ⊞ Driving from a city to another city
- ⊕ Goal state
  - ⊞ Destination city — Beijing



路线规划

## 例2 猴子和香蕉问题



## 解题过程:

✿ 用一个四元表列  $(W, x, Y, z)$  来表示这个问题状态.

- ✦  $W$  猴子的水平位置
- ✦  $x$  当猴子在箱子顶上时取  $x = 1$ ; 否则取  $x = 0$
- ✦  $Y$  箱子的水平位置
- ✦  $z$  当猴子摘到香蕉时取  $z = 1$ ; 否则取  $z = 0$

✿ 这个问题的操作（算符）如下:

- ✦  $\text{goto } (U)$ : 表示猴子走到水平位置  $U$

$$(W, 0, Y, z) \xrightarrow{\text{goto } (U)} (U, 0, Y, z)$$

▣ pushbox ( $V$ ) : 表示猴子把箱子推到水平位置 $V$

$$(W, 0, W, z) \xrightarrow{\text{pushbox}(V)} (V, 0, V, z)$$

•注意：要应用算符pushbox ( $V$ )，就要求规则的左边，猴子与箱子必须在同一位置上，并且，猴子不是箱子顶上。这种强加于操作的适用性条件，叫做产生式规则的先决条件。

▣ climbbox : 猴子爬上箱顶

$$(W, 0, W, z) \xrightarrow{\text{climbbox}} (W, 1, W, z)$$

应用算符climbbox的先决条件是什么？

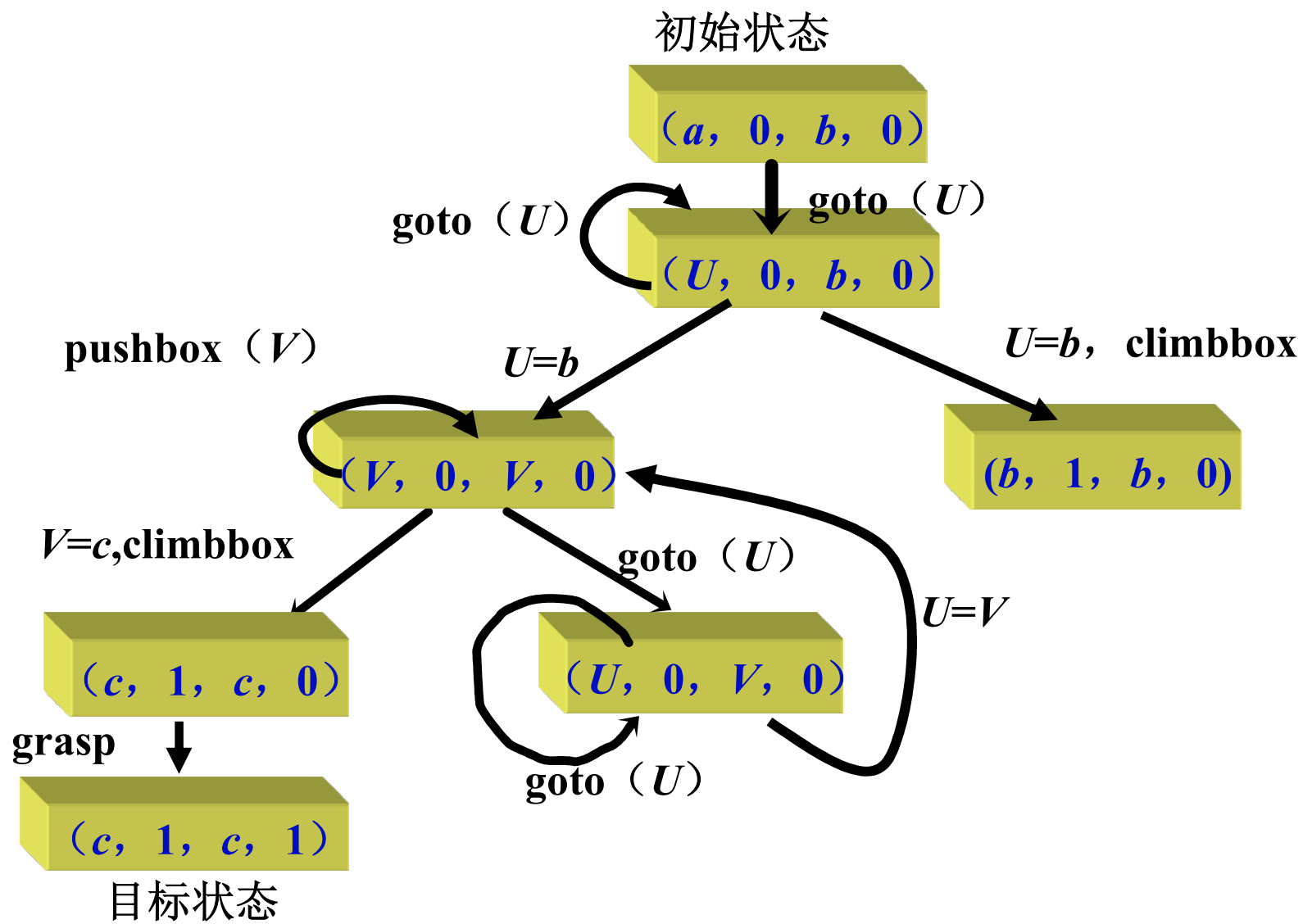
■ grasp : 表示猴子摘到香蕉

$$(c, 1, c, z) \xrightarrow{\text{grasp}} (c, 1, c, 1)$$

•令初始状态为 $(a, 0, b, 0)$ 。这时，goto( $U$ )是唯一适用的操作，并导致下一状态 $(U, 0, b, 0)$ 。现在有3个适用的操作，即goto( $U$ )，pushbox( $V$ )和climbbox(若 $U=b$ )。把所有适用的操作继续应用于每个状态，就能够得到状态空间图，如下图所示。从图不难看出，把该初始状态变换为目标状态的操作序列为：

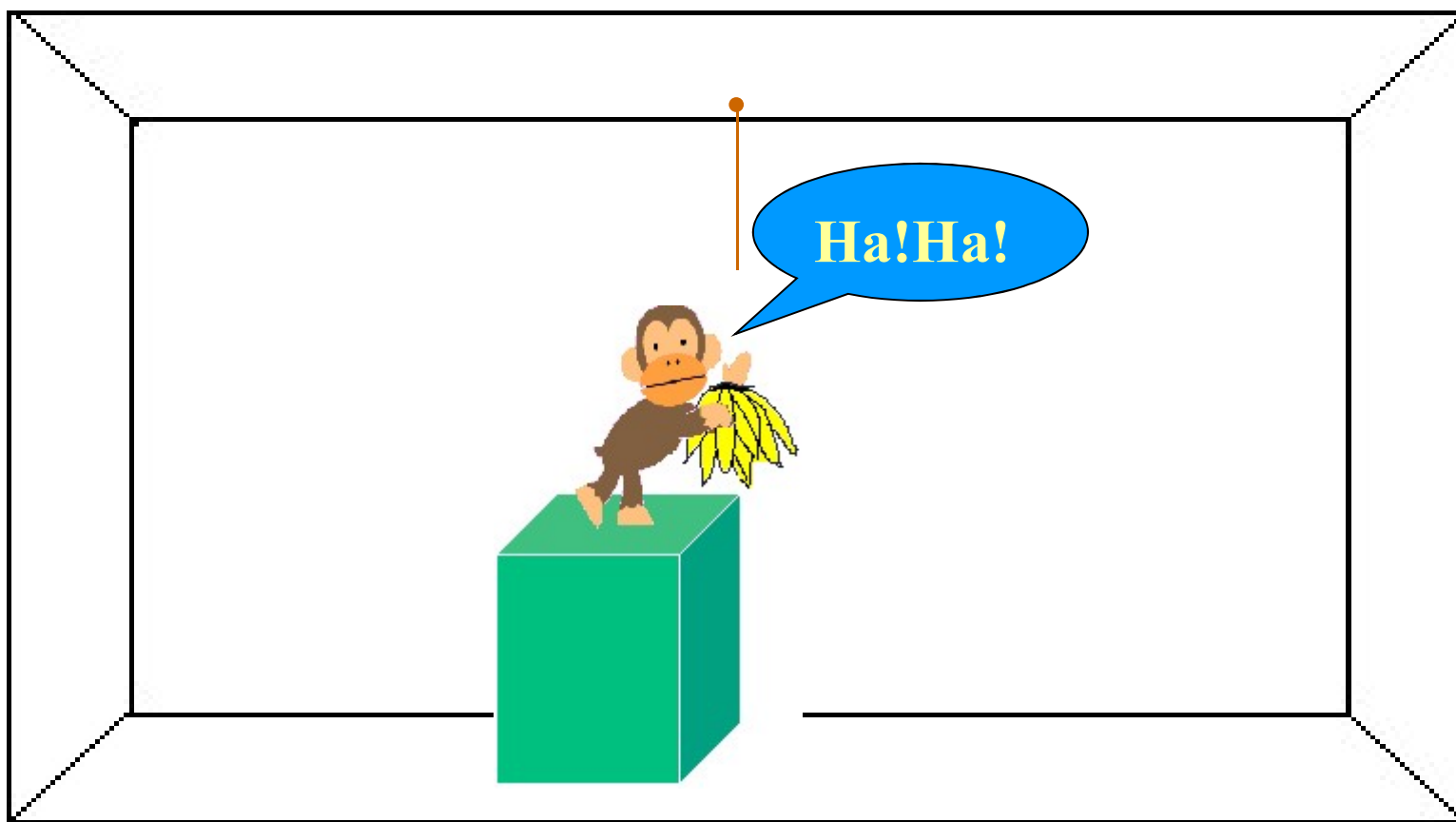
$\{\text{goto}(b), \text{pushbox}(c), \text{climbbox}, \text{grasp}\}$





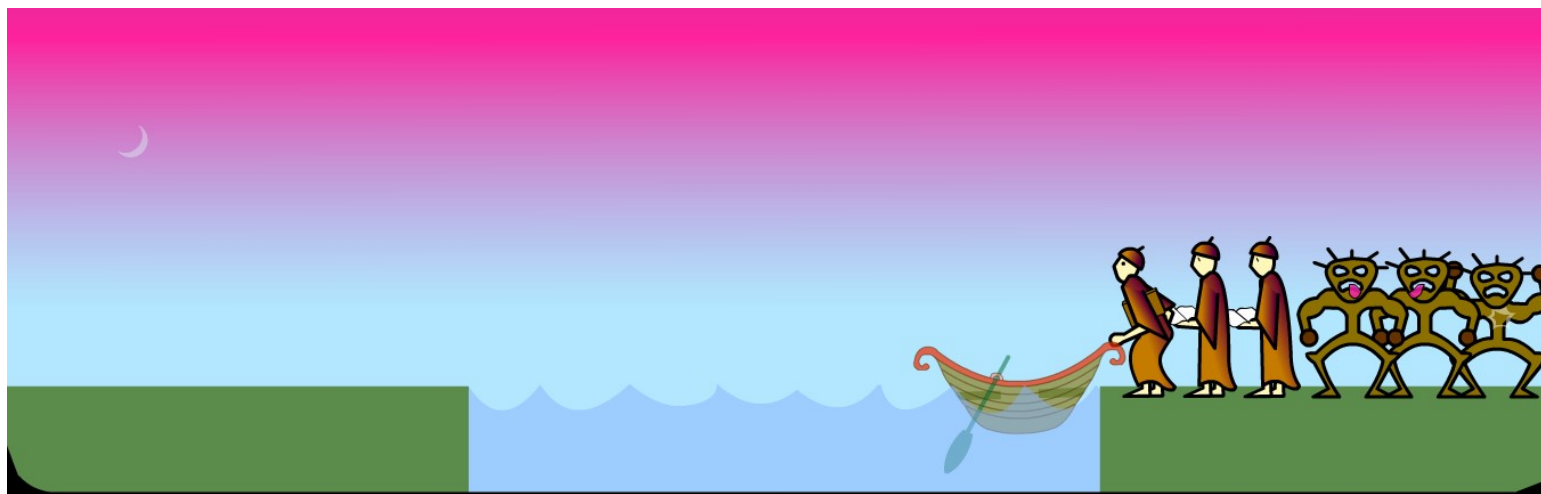
猴子和香蕉问题的状态空间图

## 猴子和香蕉问题自动演示：



### 例3 传教士野人问题（Missionaries& Cannibals Problem, MC问题）

有三个传教士M和三个野人C过河，只有一条能装下两个人的船。如果在河的一方（含船上），野人的人数大于传教士的人数，那么传教士就会有危险，你能不能提出一种安全的渡河方案呢？



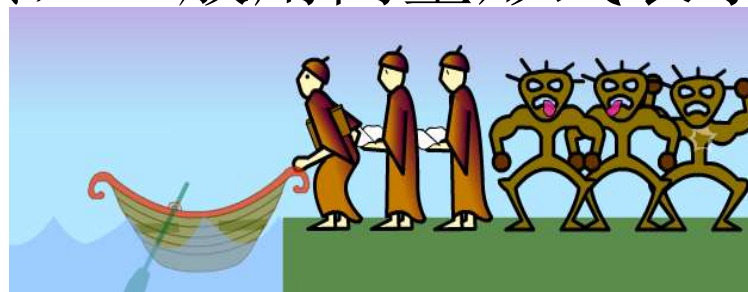
## 问题二

采用哪些操作（操作符）  
能导致传教士与野人  
问题的状态发生变化？



## 状态及其表示

- 状态：问题在某一时刻所处的“位置”，“情况”等
- 根据问题所关心的因素，一般用向量形式表示，每一位表示一个因素。



状态可有多种表示方法：

(左岸传教士数, 右岸传教士数, 左岸野人数, 右岸野人数, 船的位置)

或简化表示为：

(左岸传教士数, 左岸野人数, 船的位置)

初始状态：(0, 0, 0)

目标状态：(3, 3, 1)

┌  
└  
0: 右岸  
1: 左岸

# 状态的转换

✚ 算子（算符，操作符 Operator）——使状态发生改变的操作

✚ MC问题中的算子

✚ 将传教士或野人运到河对岸

✚ **Move-1m1c-lr**: 将一个传教士(m)和一个野人(c)从左岸(L)运到右岸(R)

✚ 所有可能操作

✚ **Move-1m1c-lr**

**Move-1m1c-rl**

**Move-2c-lr**

**Move-2c-rl**

**Move-2m-lr**

**Move-2m-rl**

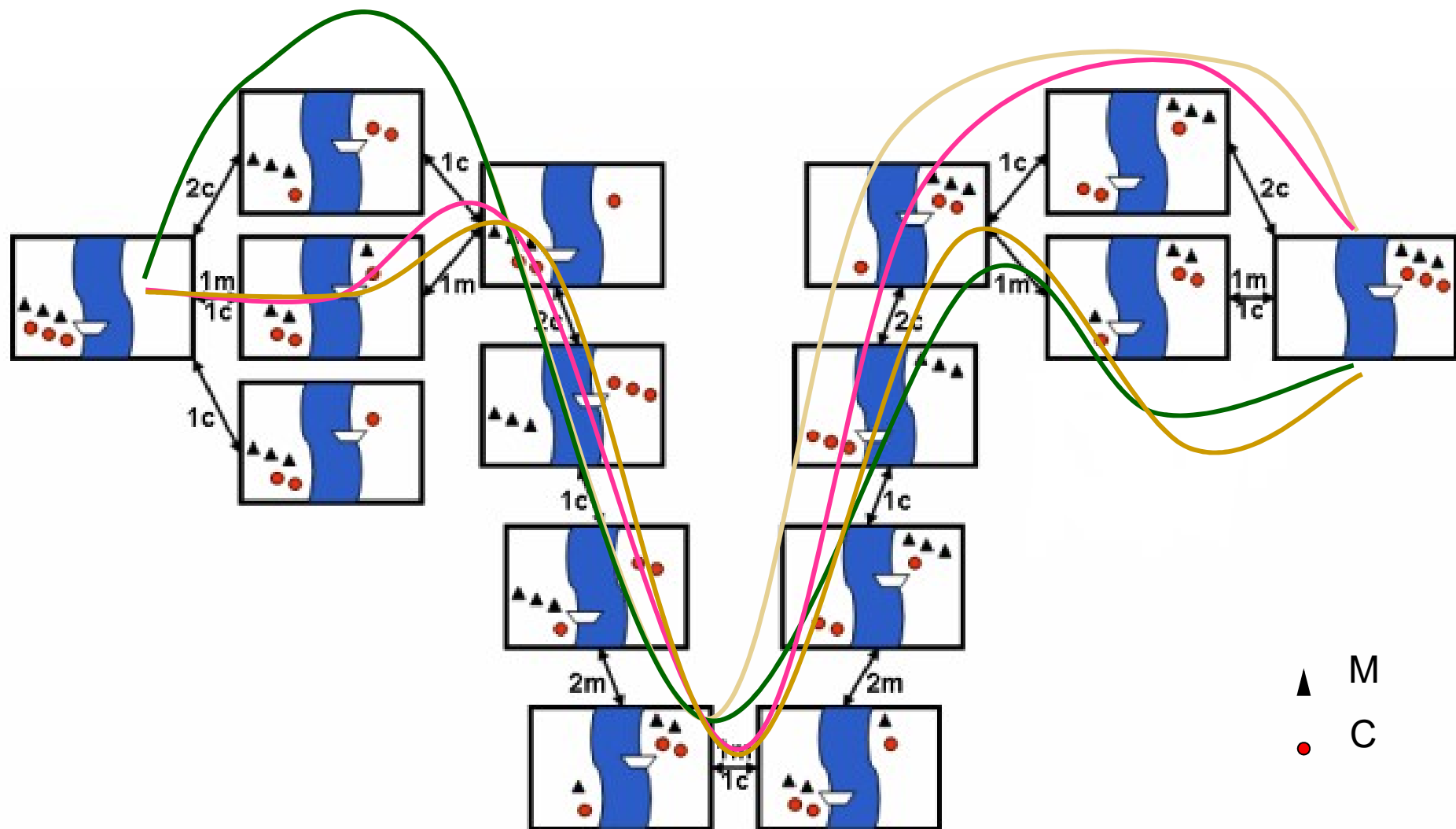
**Move-1c-lr**

**Move-1c-rl**

**Move-1m-lr**

**Move-1m-rl**

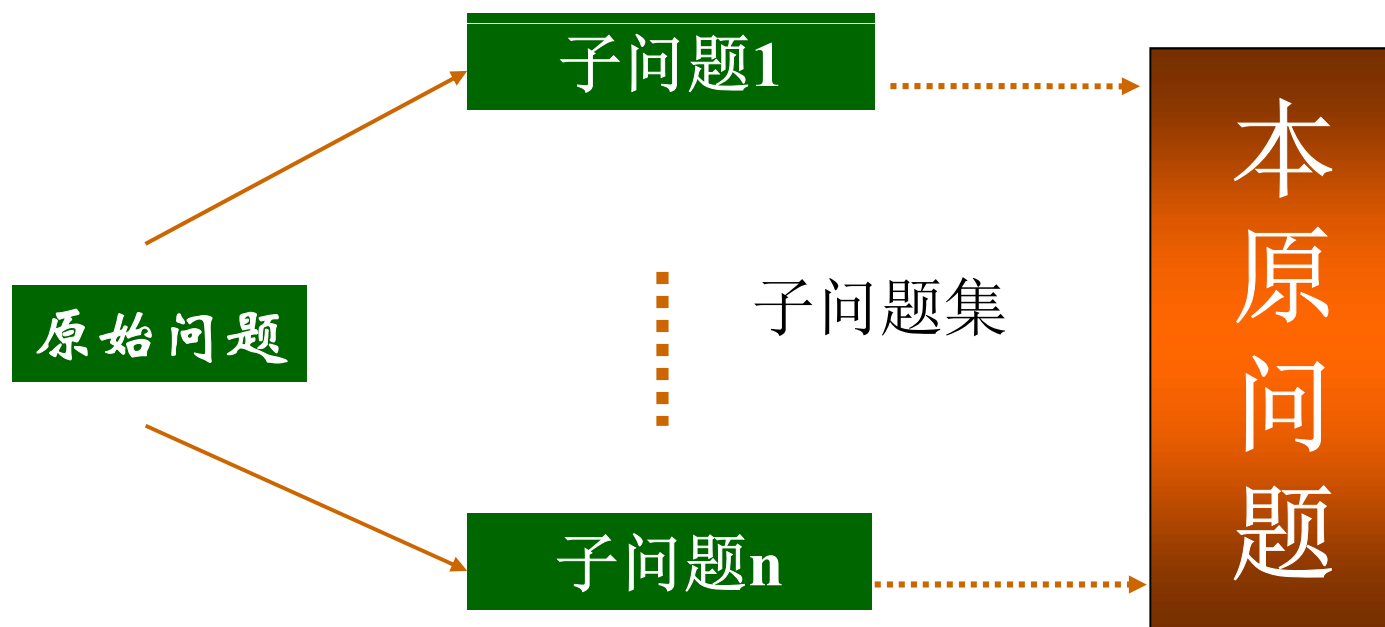
# 传教士野人问题状态空间图



## 2.2 问题归约表示 (Problem Reduction Representation)

### ❁ 问题归约法思想

先把问题分解为子问题及子-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。

## ❁ 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。

## 2.2.1 问题归约描述 (Problem Reduction Description)

### ❁ 梵塔难题(Tower of Hanoi Puzzle)

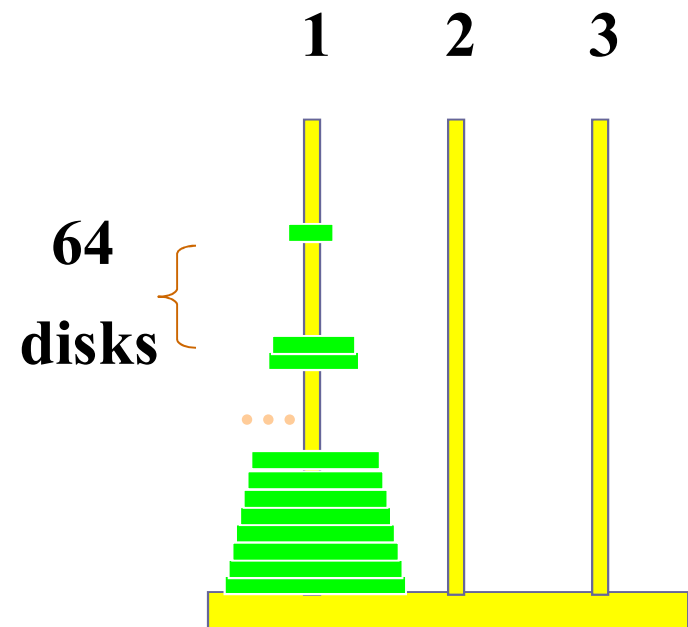
■ 3个柱子和64个圆盘

■ 移动时间: 64 个圆盘

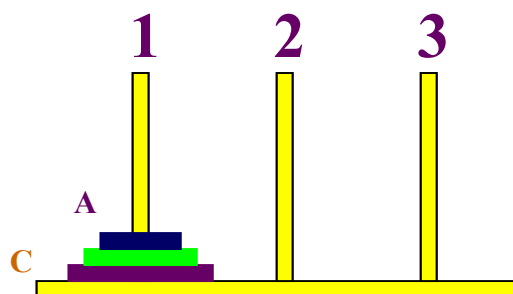
$$2^{64}-1 \approx 2^{64} = 10^{19.27}$$

If one person moves 1 disk in one second, then to finish this problem needs more than 3000 billion years.

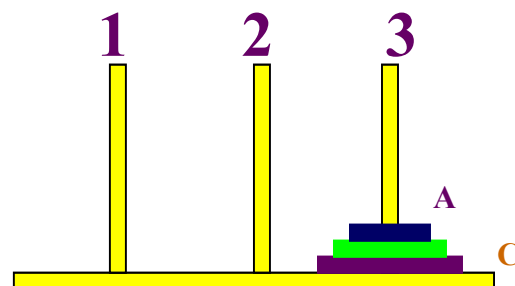
(30000多亿年)



### ❁ 3圆盘梵塔难题



(a) 初始配置



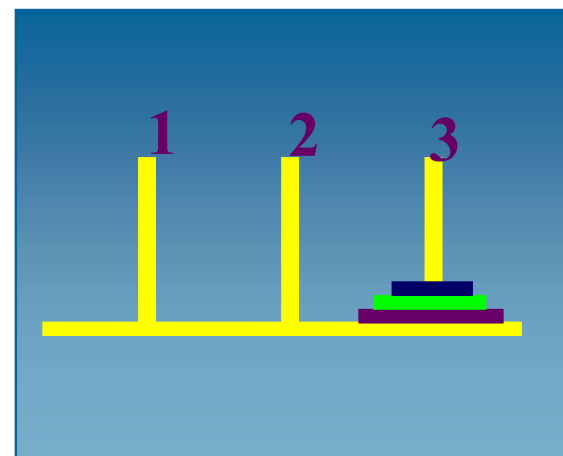
(b) 目标配置

## ➤ 约束条件

(1) 一次只能移动一个圆盘;

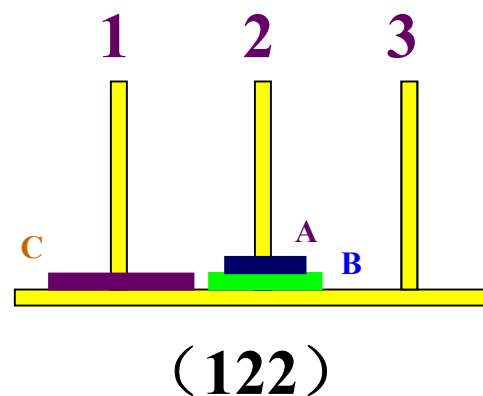
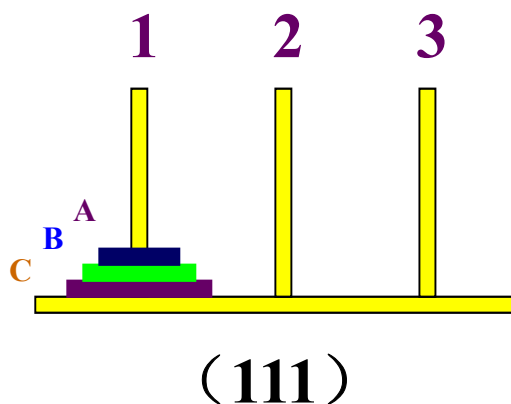
(2) 只有柱子最上面的圆盘可以移动;

(3) 大圆盘不能放在小圆盘上面。

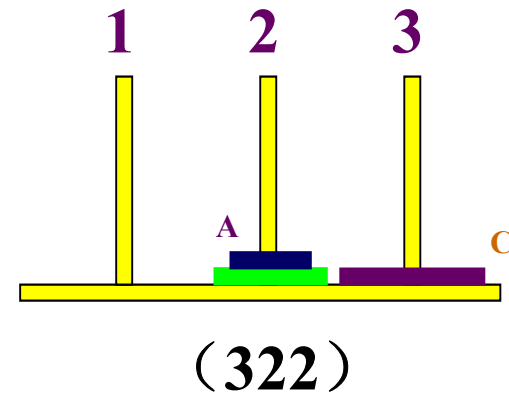
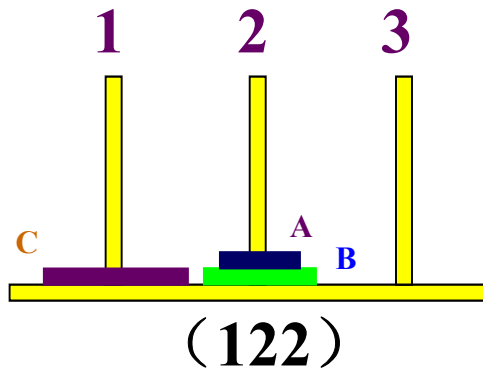


## 解题过程:

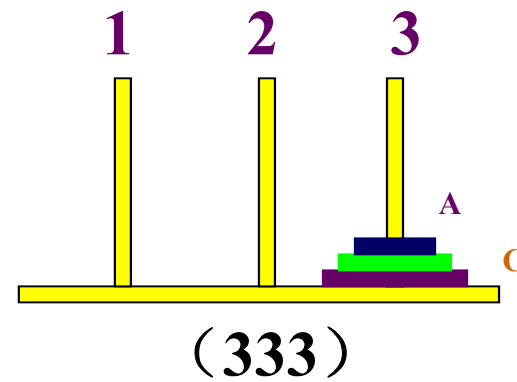
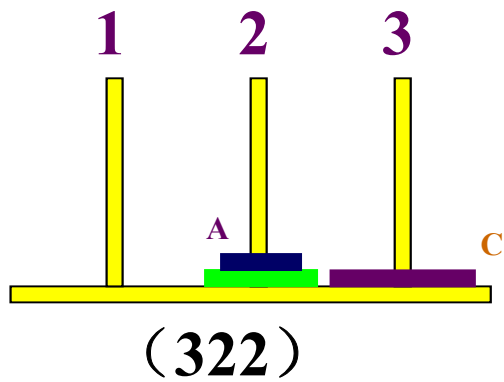
- ✿ 将原始问题归约为一个较简单问题集合  
把原始梵塔难题归约（简化）为下列3个子难题：
  - ✦ 移动圆盘A和B至柱子2的双圆盘难题；



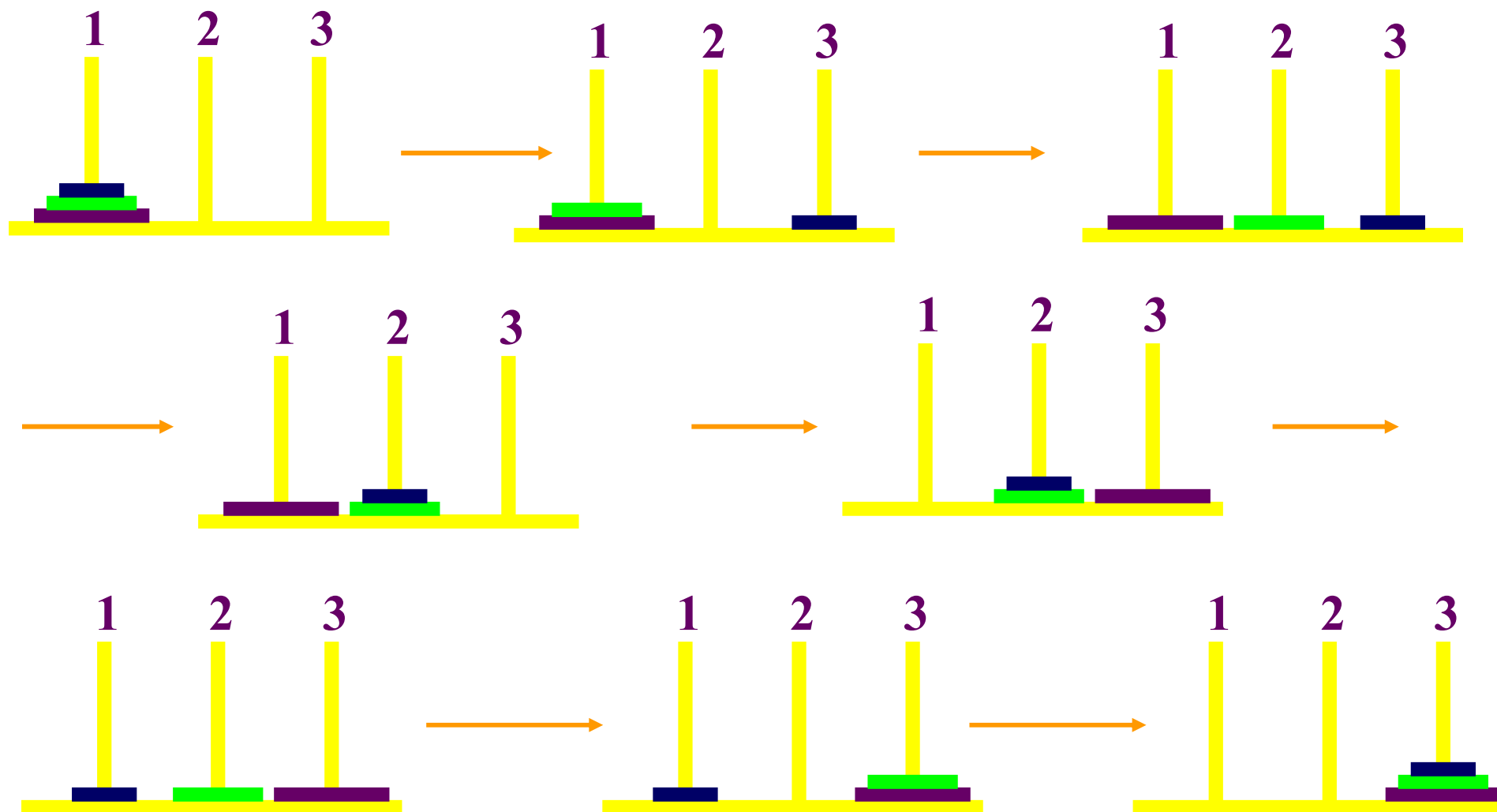
❖ 移动圆盘C至柱子3的单圆盘难题；



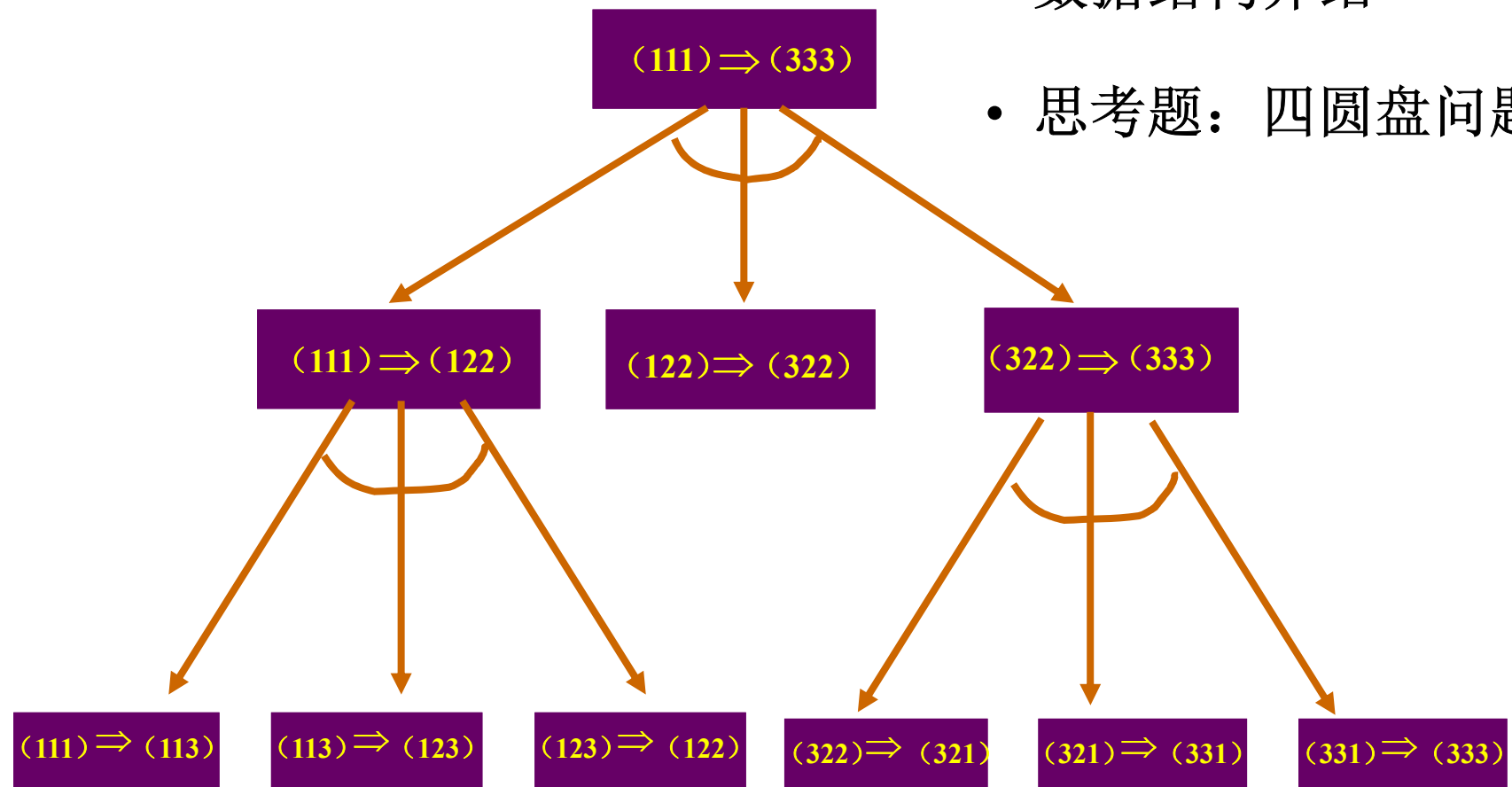
❖ 移动圆盘A和B至柱子3的双圆盘难题。



## 解题过程（3个圆盘梵塔难题）



## 梵塔问题归约图（与或图）



- 数据结构介绍
- 思考题：四圆盘问题



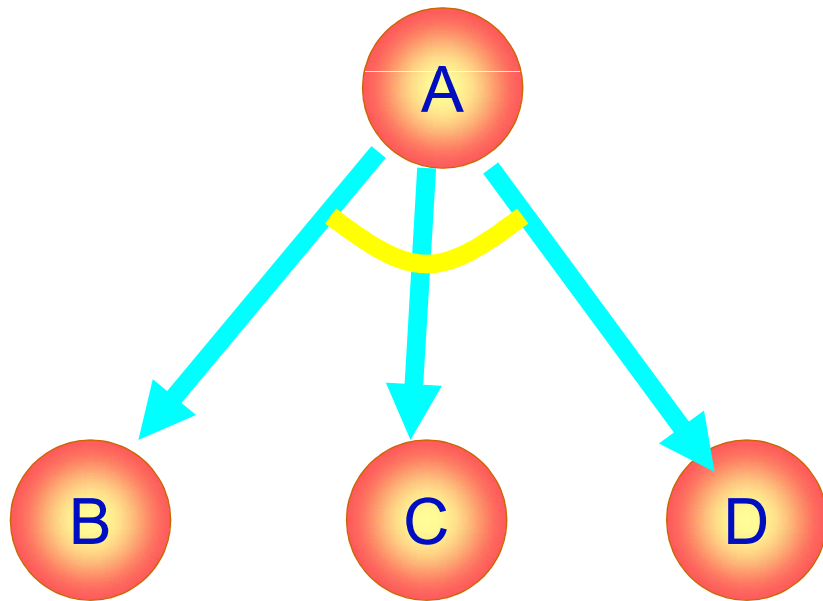
# Four-disk Hanoi Tower



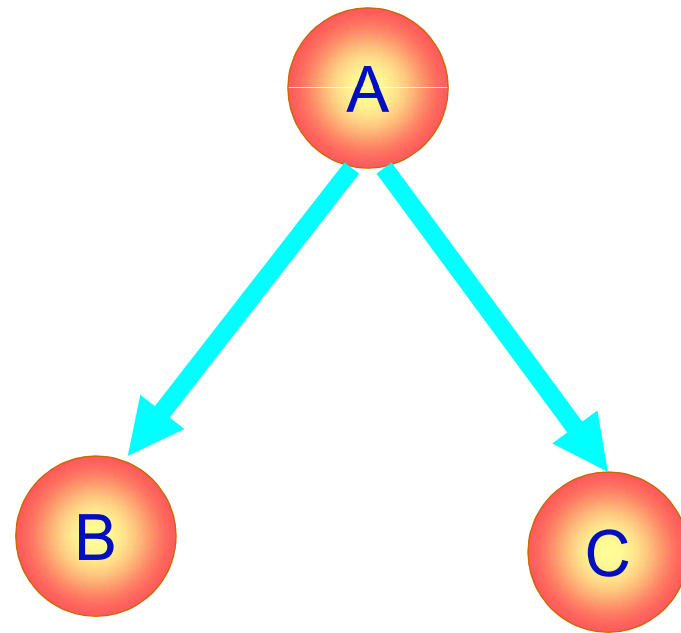
## 2.2.2 与或图表示 (AND/OR Graph Representation)

### ✚ 与图、或图、与或图

一般，用一个似图结构来表示把问题归约为后继问题的替换集合，这一似图结构叫做问题归约图，或叫与或图。如下所示



与图



或图

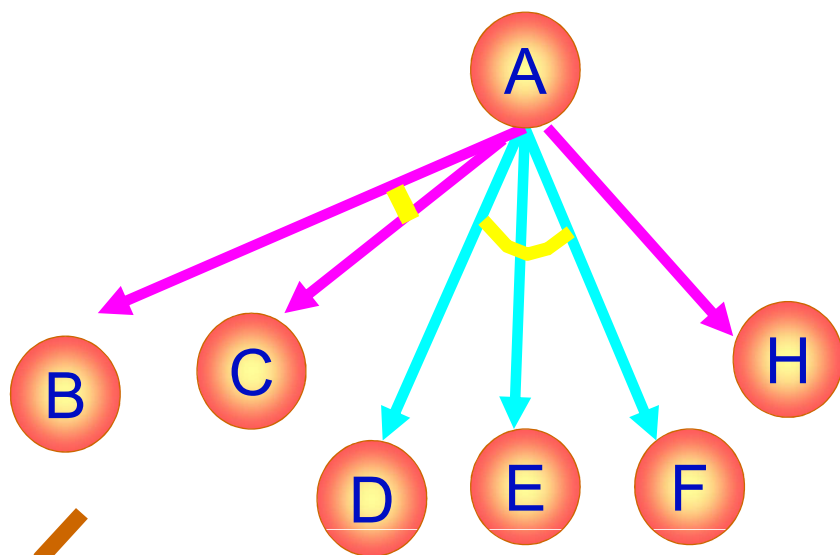


图2.6 子问题替换集合结构图

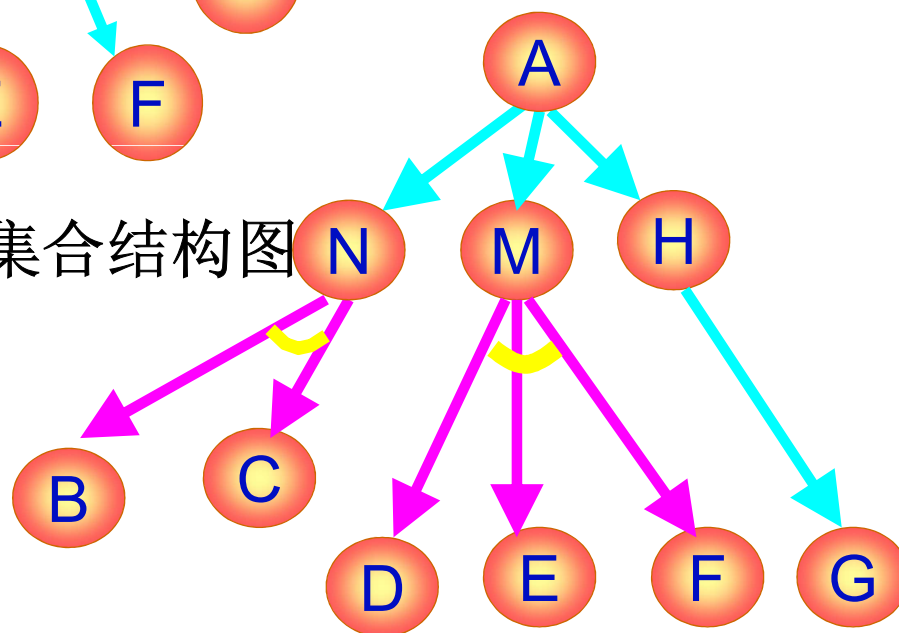
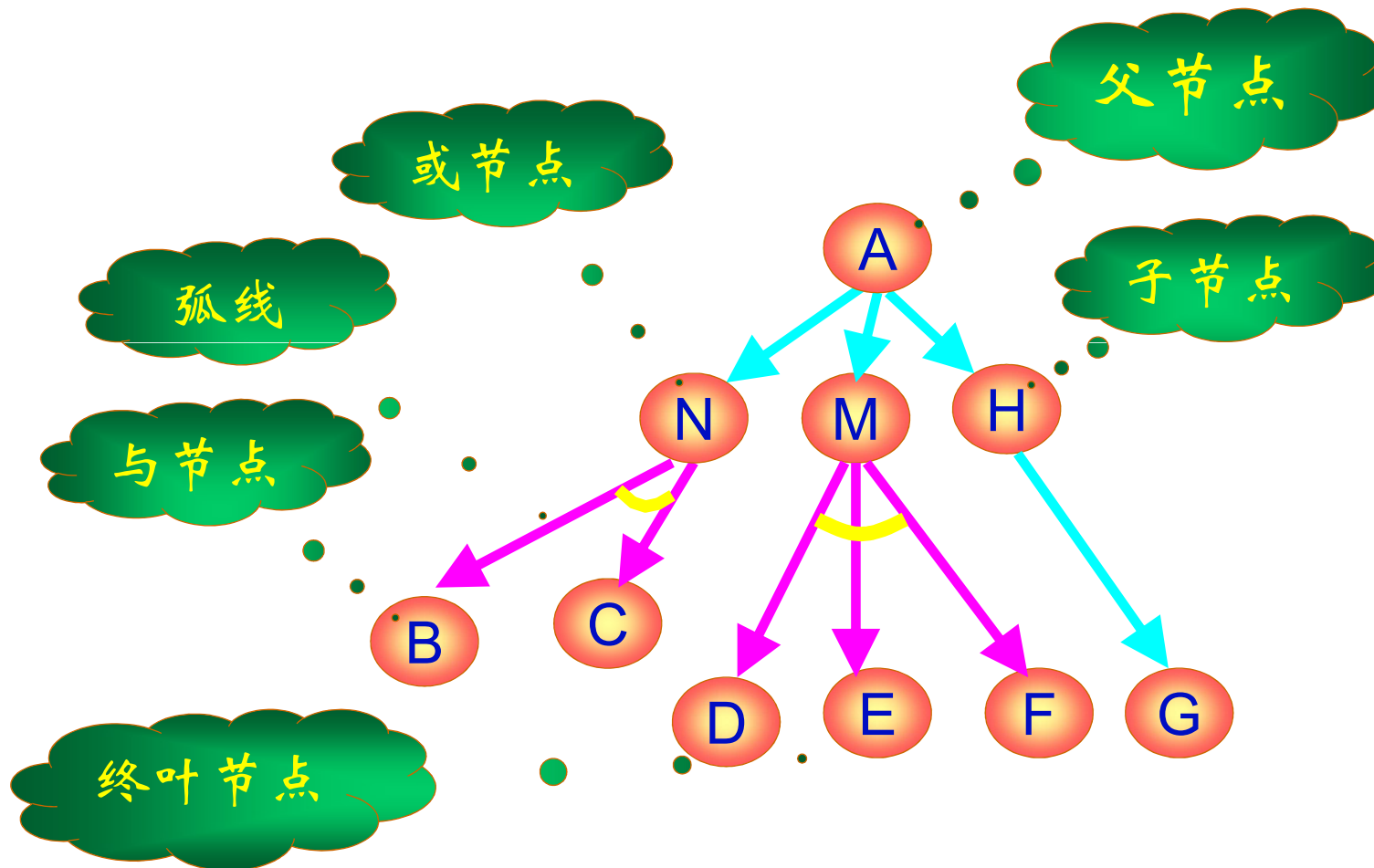


图2.7 一个与或图

## 一些关于与或图的术语



## ✿ 一些关于与或图的术语

- ✿ 父节点、子（后继）节点、弧线
- ✿ 起始节点：对于于原始问题描述的节点
- ✿ 终叶节点：对应于本原问题的节点
- ✿ 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）。各个节点之间用一段小圆弧连接标记。
- ✿ 与或图：由与节点及或节点组成的结构图。

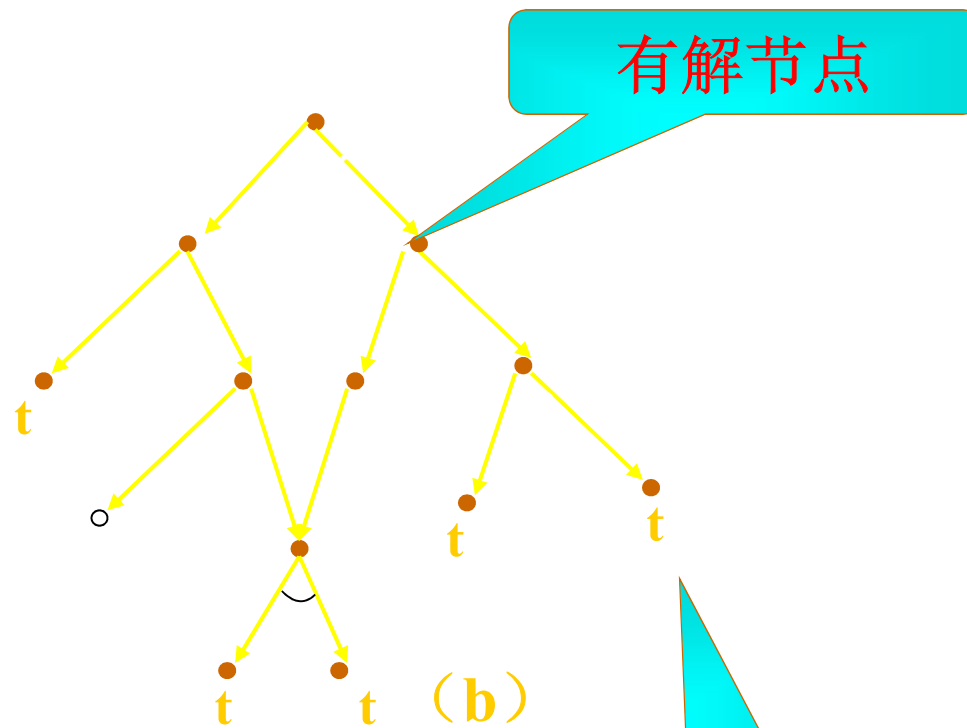
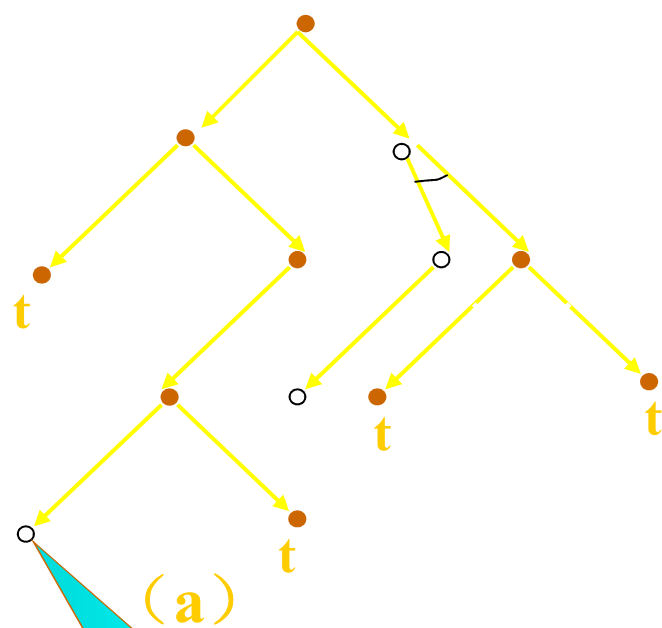
## ❁ 可解节点的一般定义

- ❁ 终叶节点是可解节点(因为它们与本原问题相关联)。
- ❁ 如果某个非终叶节点含有或后继节点，那么只要有一个后继节点是可解的时，此非终叶节点就是可解的。
- ❁ 如果某个非终叶节点含有与后继节点，那么只有其全部后继节点为可解时，此非终叶节点才是可解的。

## ❁ 不可解节点的一般定义

- ❁ 没有后裔的非终叶节点为不可解节点。
- ❁ 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。
- ❁ 如果某个非终叶节点含有与后继节点，那么只要当其后裔有一个为不可解时，此非终叶节点就是不可解的。

如图所示



与或图例子



## ❁ 与或图构成规则

- ❁ 与或图中的每个节点代表一个要解决的单一问题或问题集合。起始节点对应于原始问题。
- ❁ 对应于本原问题的节点，叫做终叶节点。
- ❁ 对于把算符应用于问题A的每种可能情况，都把问题变换为一个子问题集合；有向弧线自A指向后继节点，表示所求得的子问题集合，这些子问题节点叫做或节点。
- ❁ 一般对于代表两个或两个以上子问题集合的每个节点，有向弧线从此节点指向此子问题集合中的各个节点,这些子问题节点叫做与节点。

### 问题三

状态空间与问题归约两种表示方法有何关系？

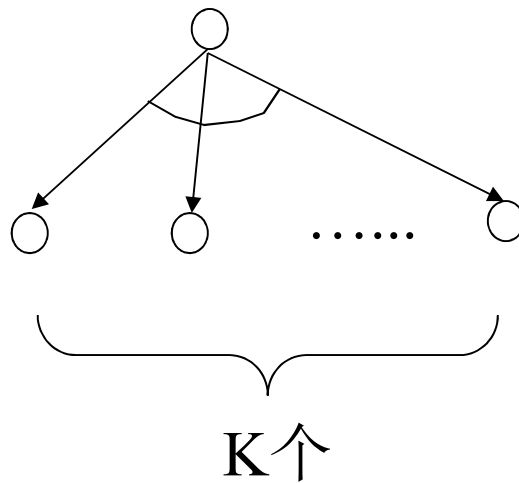


## 问题四

状态空间法和问题归约法  
都是图示法。是否可以用  
逻辑符号表示知识？



- 与或图是一个超图，节点间通过连接符连接。
- K-连接符：**

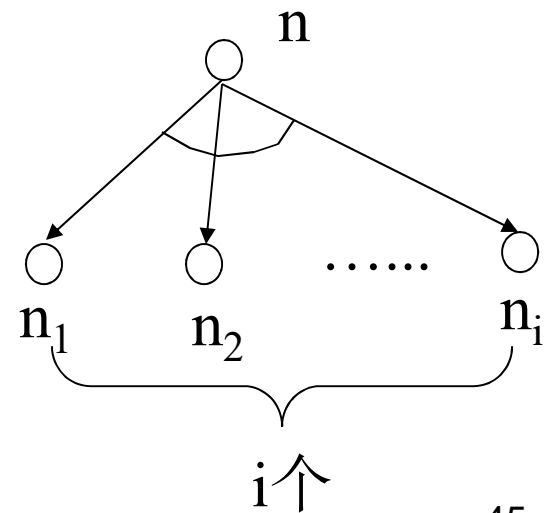


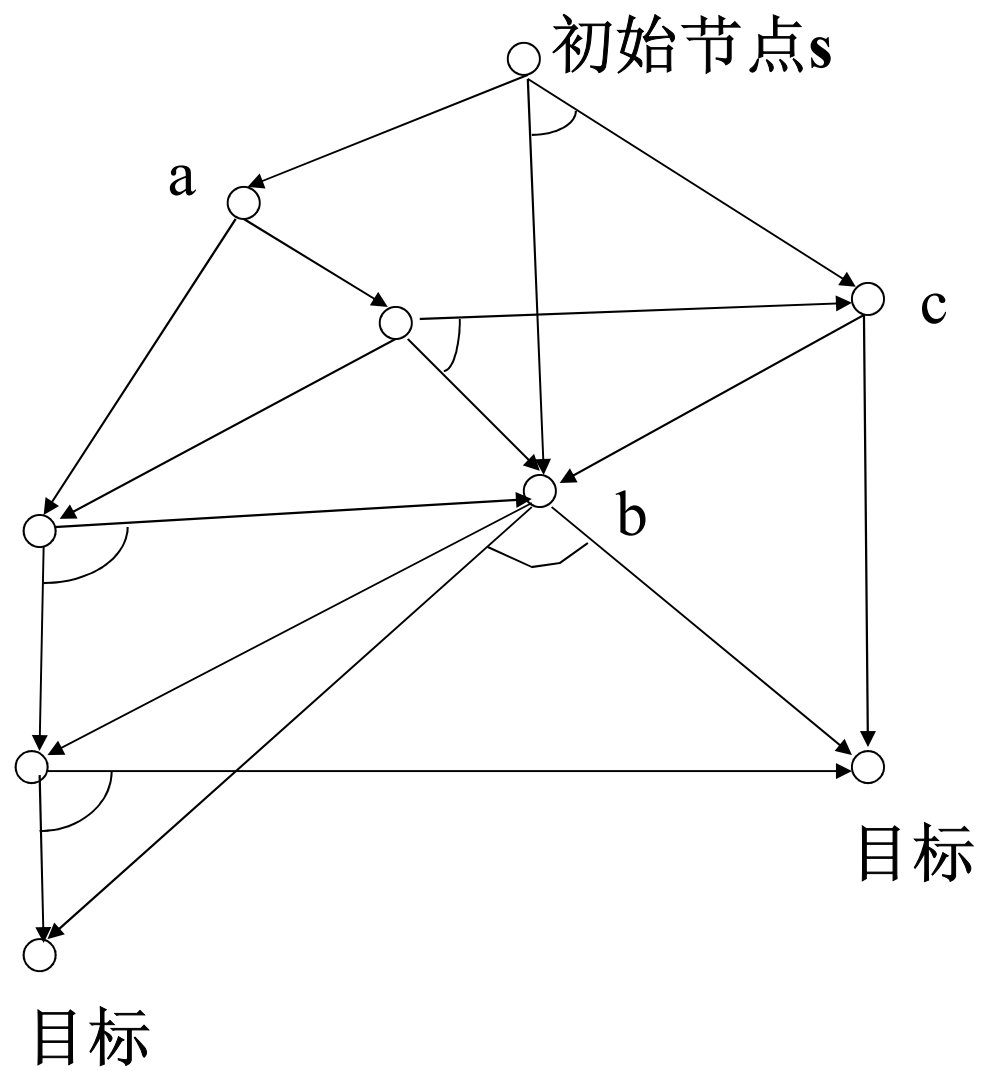
## 耗散值的计算

$$k(n, N) = C_n + k(n_1, N) + \dots + k(n_i, N)$$

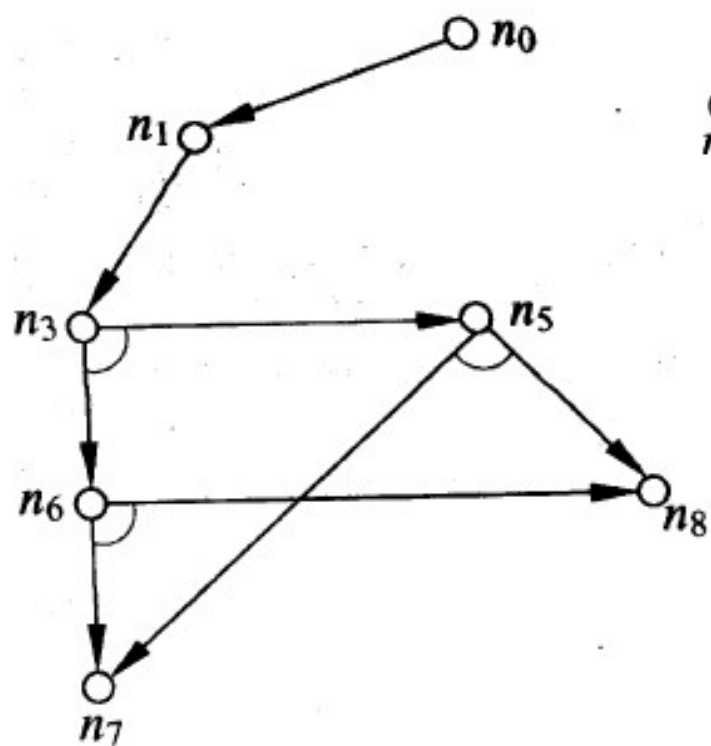
其中：**N**为终节点集

**C<sub>n</sub>**为连接符的耗散值

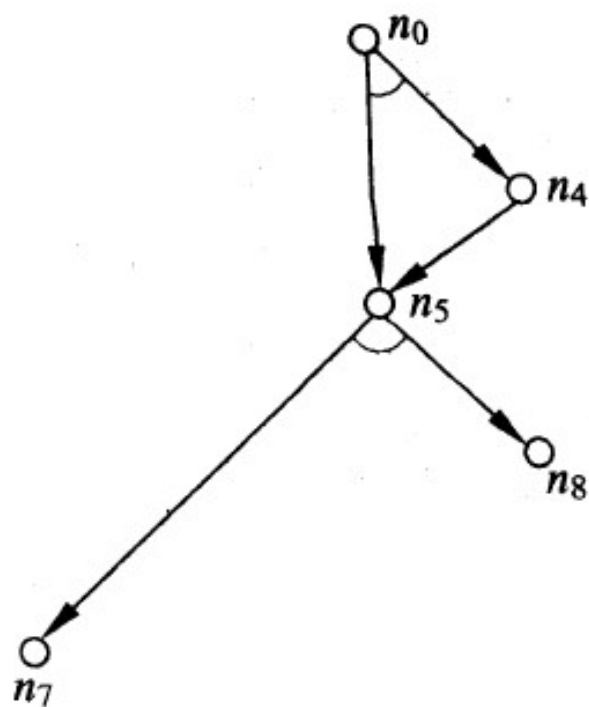




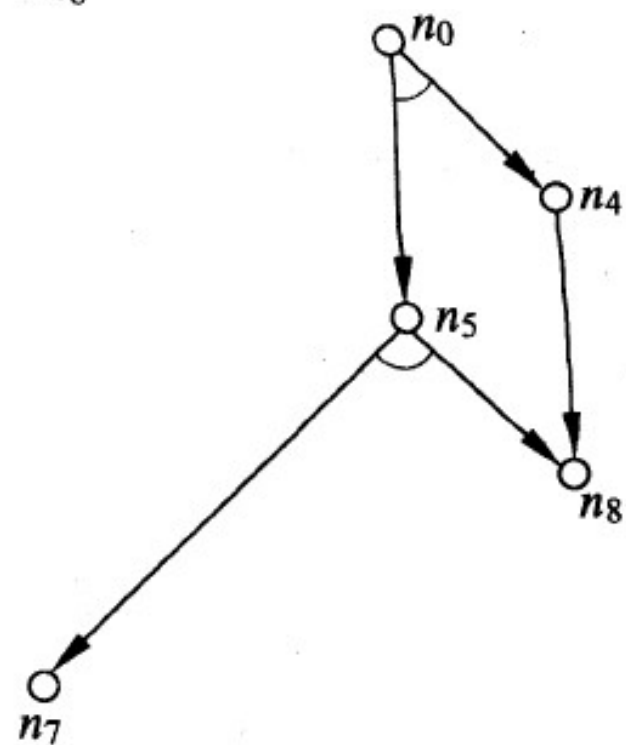
• 解图:



(a)



(b)



(c)

# 能解节点

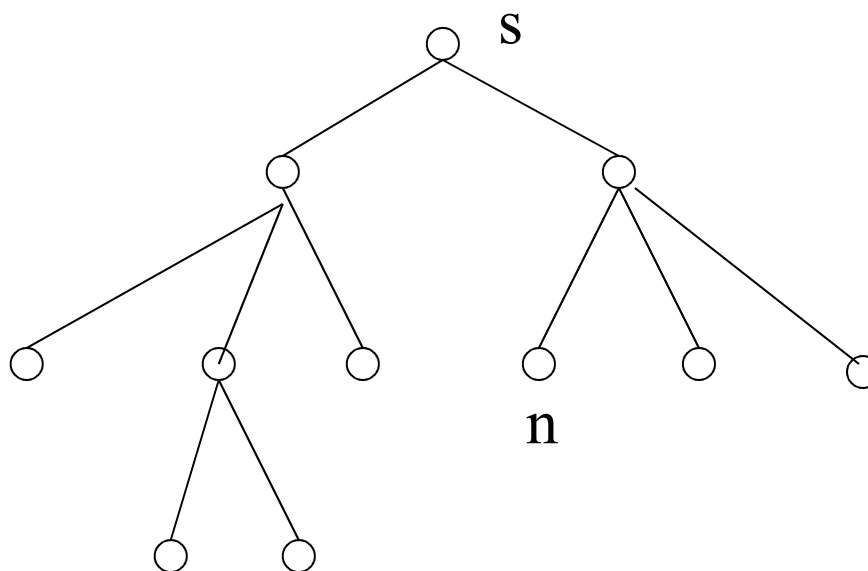
- ✿ 终节点是能解节点
- ✿ 若非终节点有“或”子节点时，当且仅当其子节点至少有一能解时，该非终节点才能解。
- ✿ 若非终节点有“与”子节点时，当且仅当其子节点均能解时，该非终节点才能解。



## 不能解节点

- ❖ 没有后裔的非终节点是不能解节点。
- ❖ 若非终节点有“或”子节点，当且仅当所有子节点均不能解时，该非终节点才不能解。
- ❖ 若非终节点有“与”子节点时，当至少有一个子节点不能解时，该非终节点才不能解。

# 与或图的启发式搜索算法AO\*

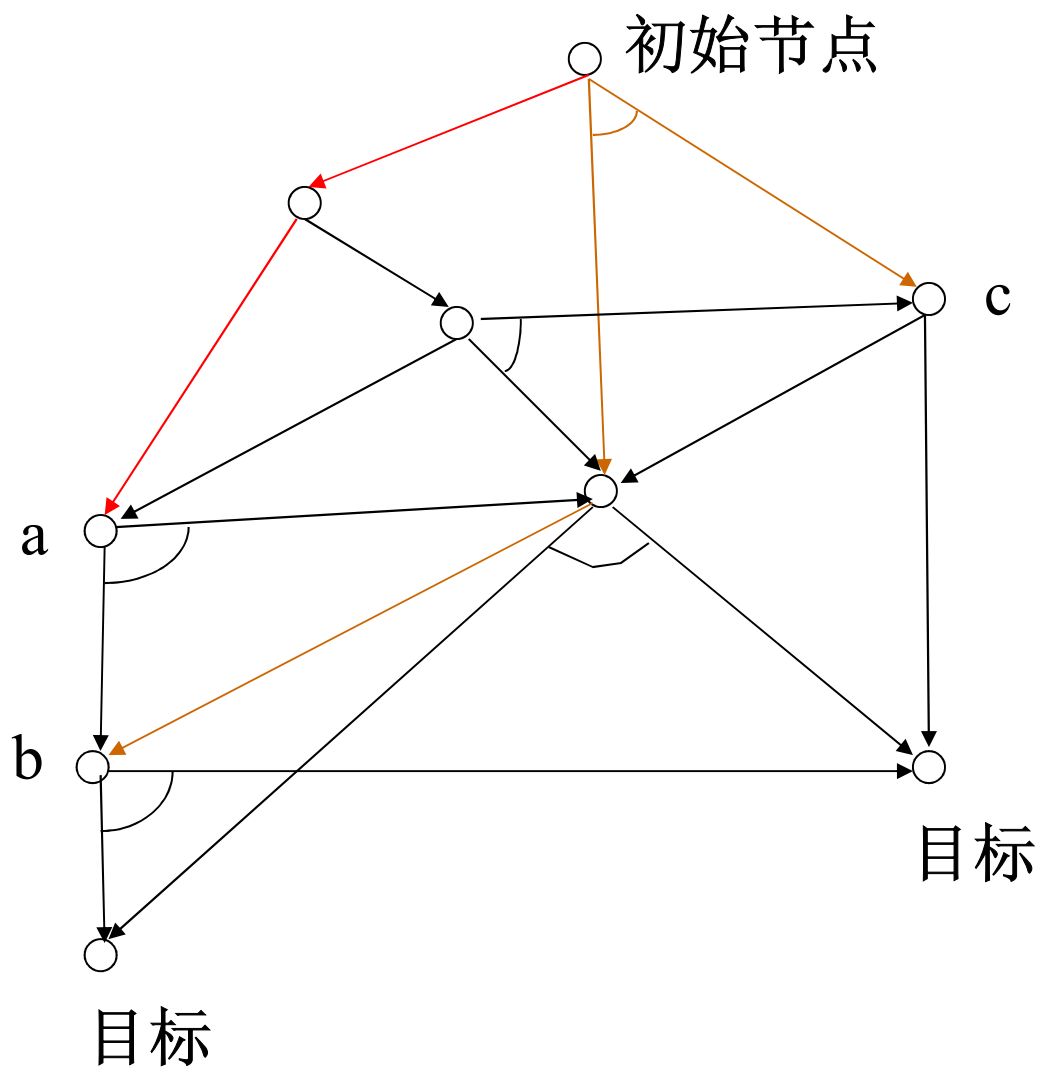


$$f(n) = g(n) + h(n)$$

对 $n$ 的评价实际是对从 $s$ 到 $n$ 这条路径的评价

普通图搜索的情况

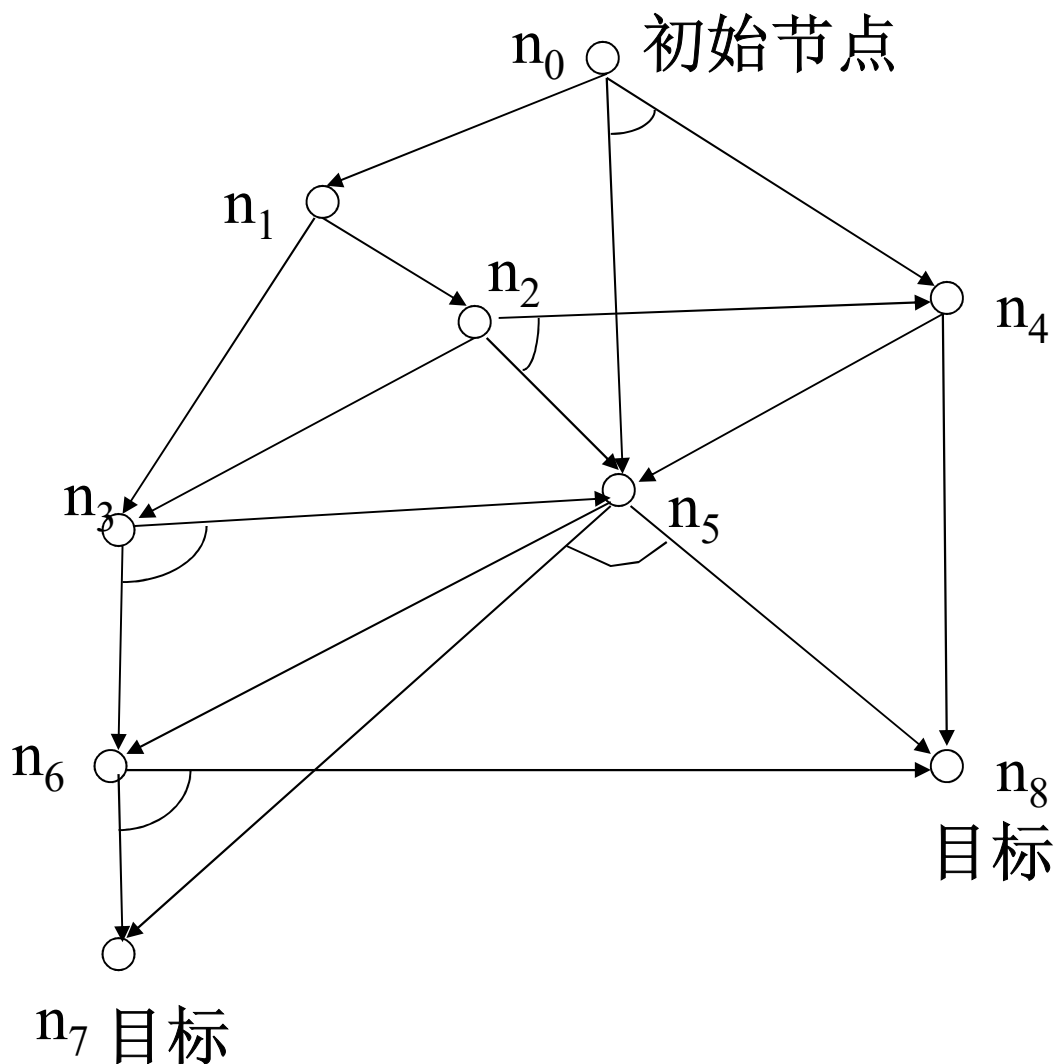
## 与或图：对局部图的评价



## 两个过程

- ✚ 图生成过程，即扩展节点
  - ▣ 从最优的局部途中选择一个节点扩展
- ✚ 计算耗散值的过程
  - ▣ 对当前的局部图重新新计算耗散值

## AO\*算法举例



其中：

$$h(n_0)=3$$

$$h(n_1)=2$$

$$h(n_2)=4$$

$$h(n_3)=4$$

$$h(n_4)=1$$

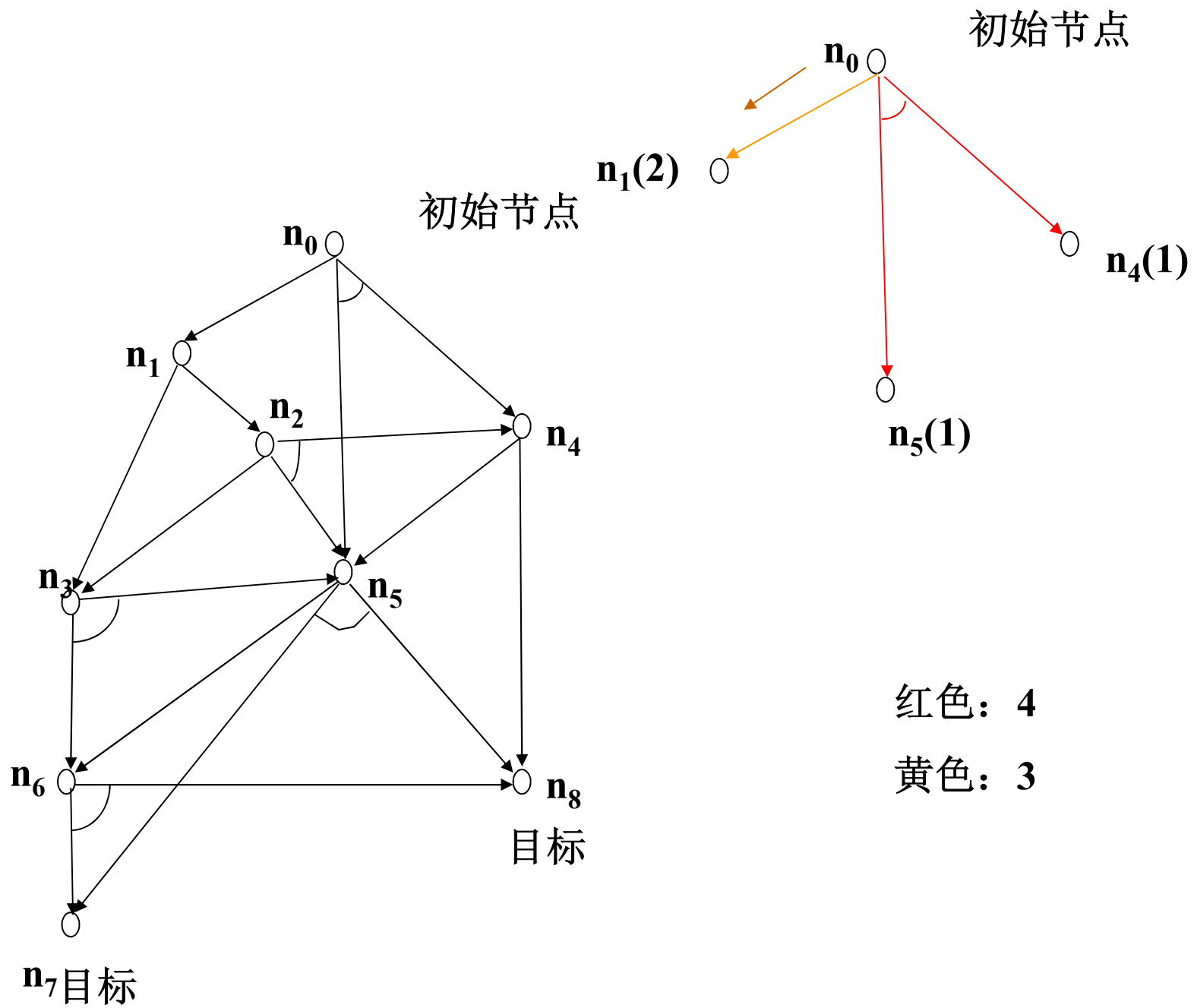
$$h(n_5)=1$$

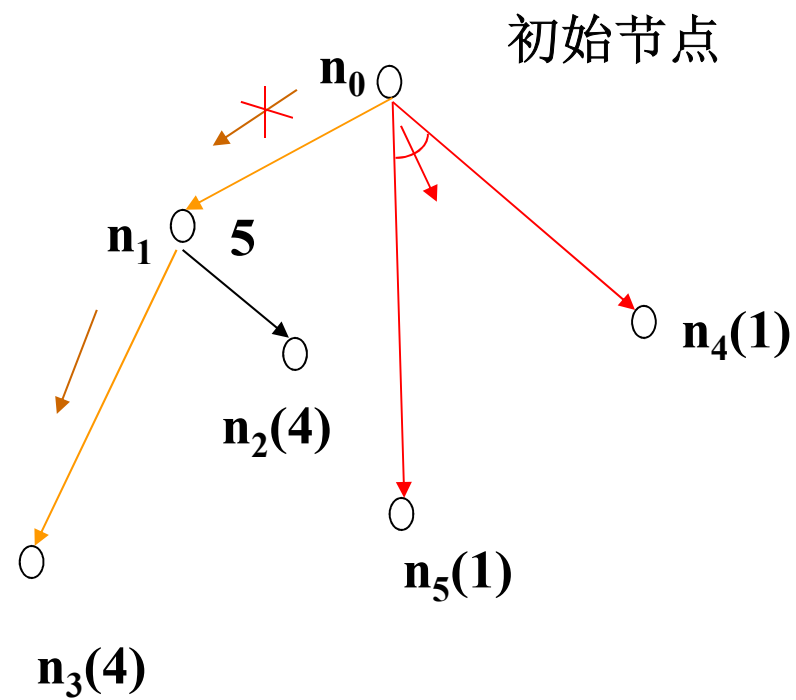
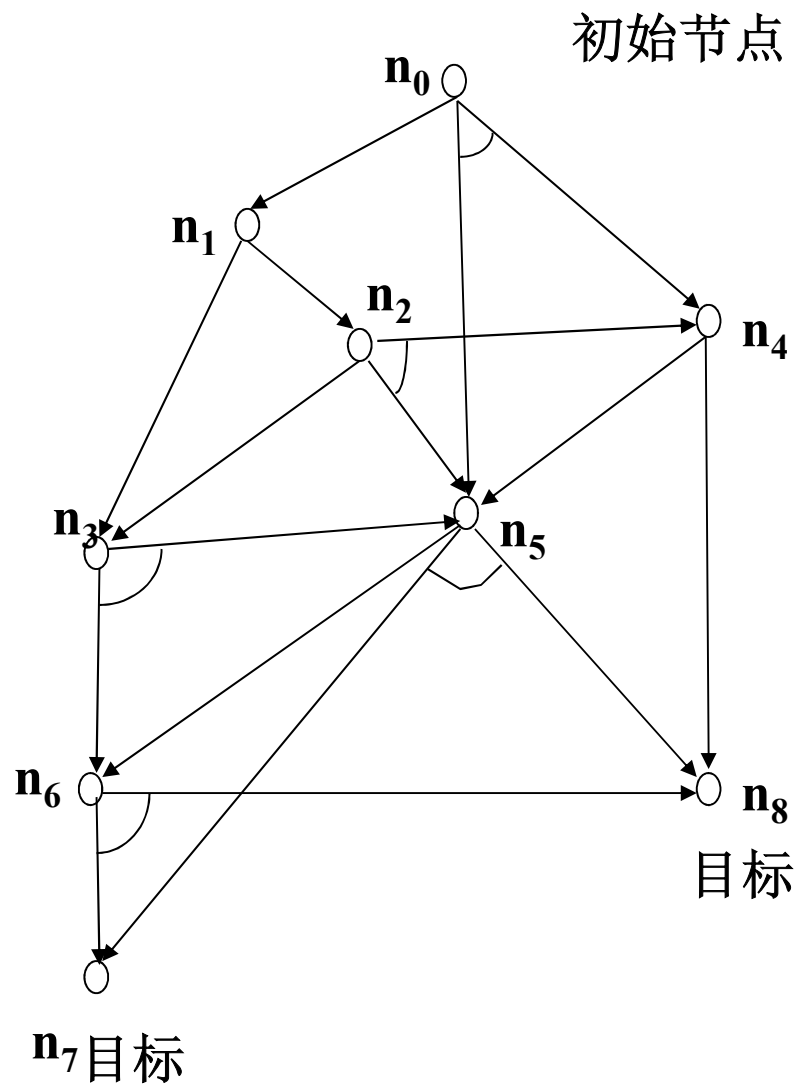
$$h(n_6)=2$$

$$h(n_7)=0$$

$$h(n_8)=0$$

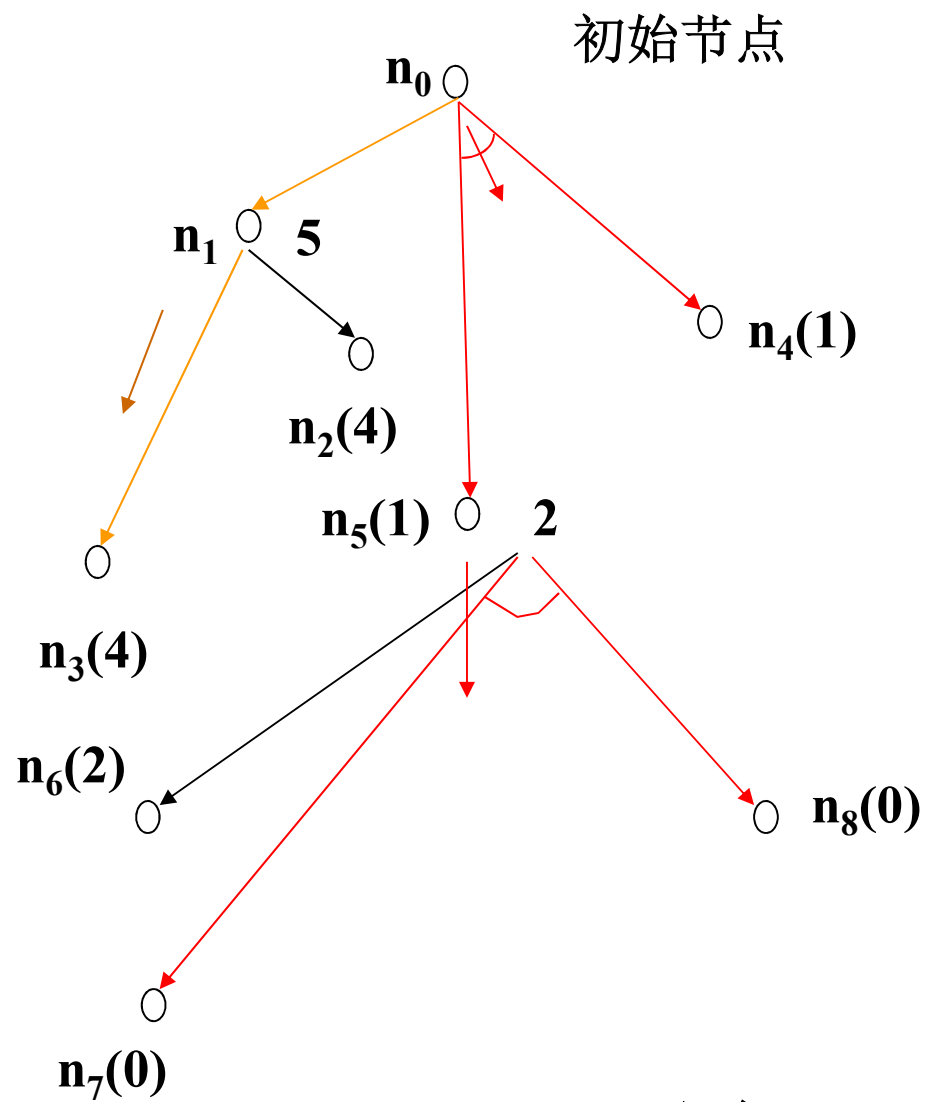
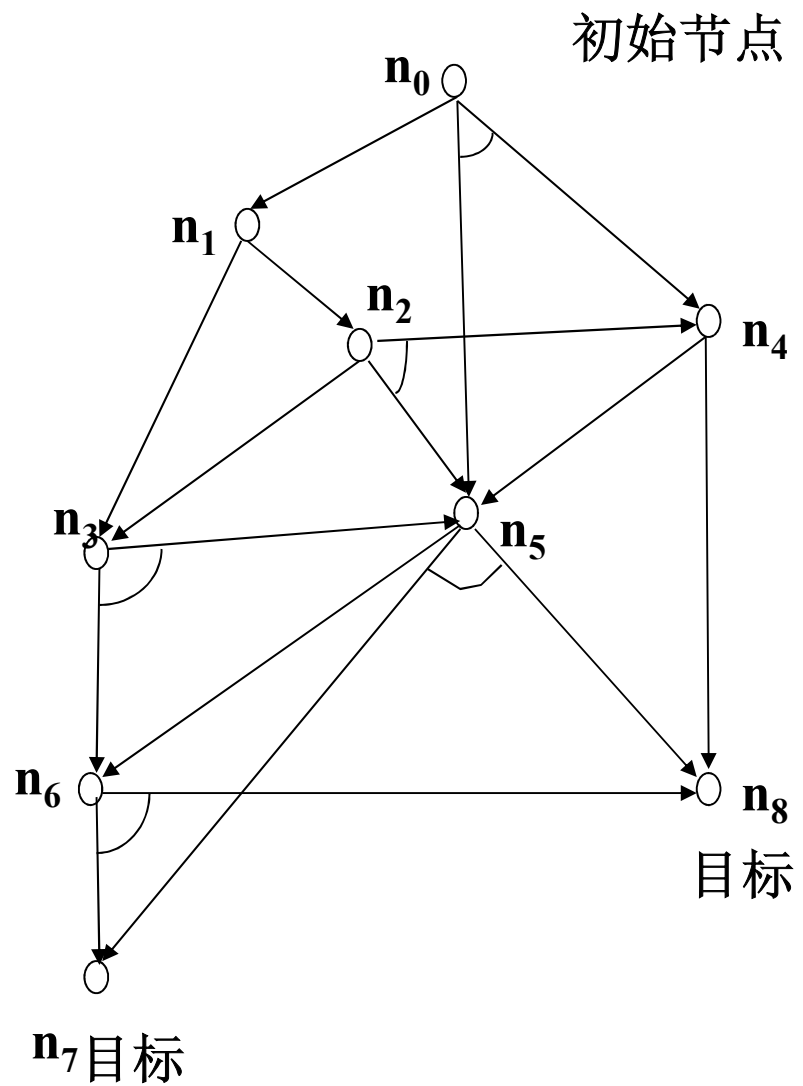
设：K连接符  
的耗散值为K





红色: 4

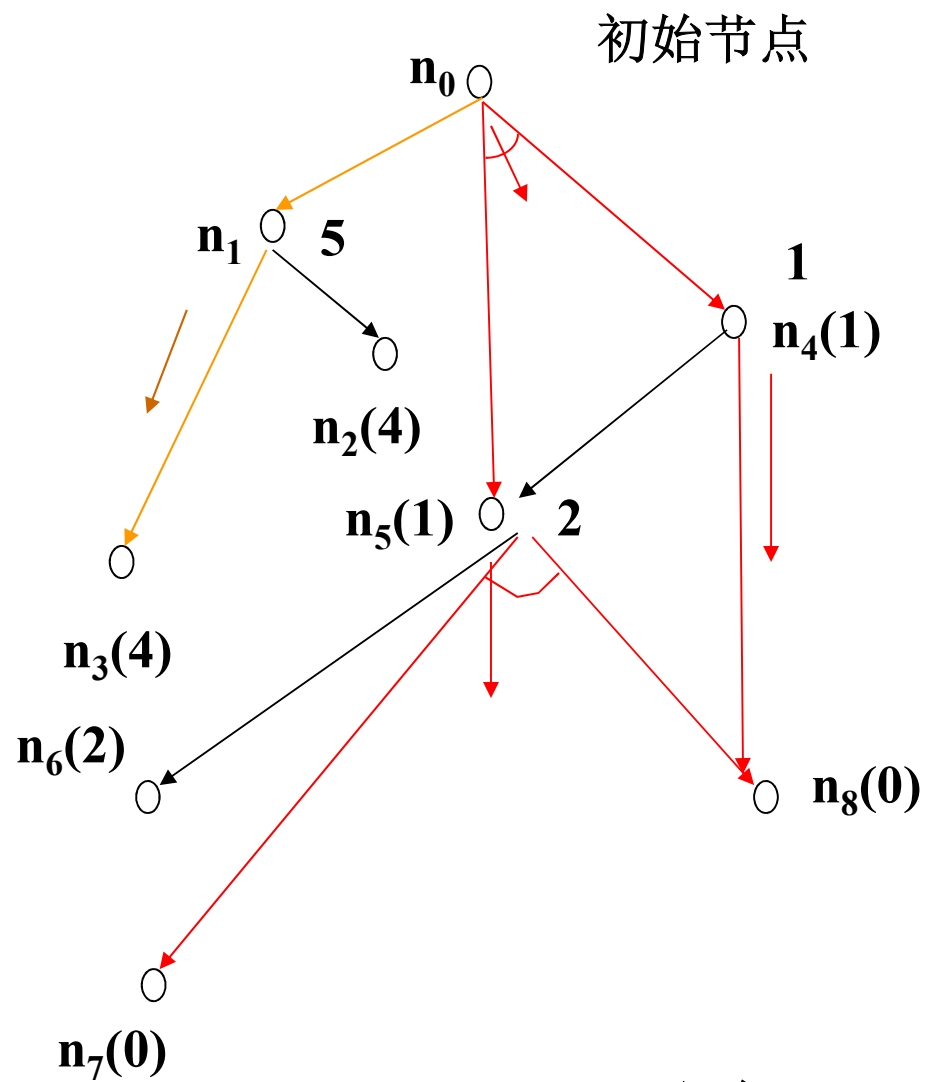
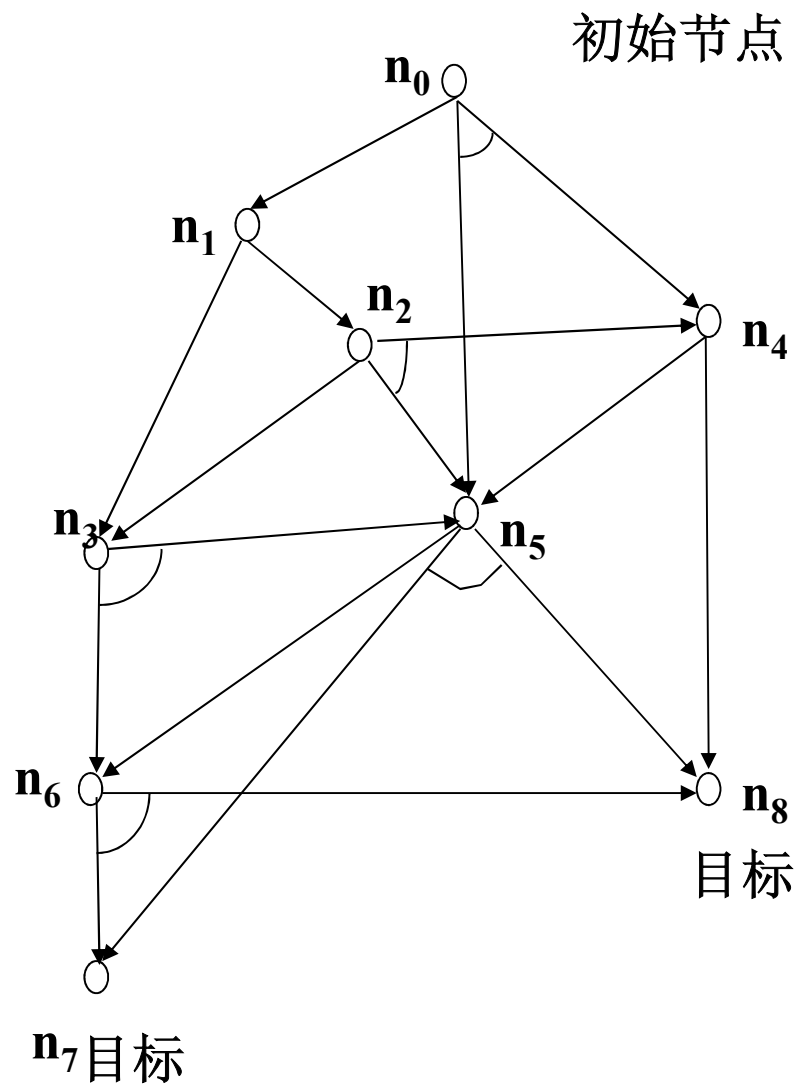
黄色: 6



红色: 5

黄色: 6





红色: 5

黄色: 6

# AO\*算法

AO\*算法可划分成两个操作阶段:

第一阶段是完成自顶向下的图生成操作,先通过有标记的连接符,找到目前为止最好的一个局部解图,然后对其中一个非终结点进行扩展,并对其后继结点赋估计耗散值和加能解标记。

# AO\*算法

第二阶段是完成自下向上的耗散值修正计算、连接符(即指针)的标记以及结点的能解标记。

耗散值的修正从刚被扩展的结点 $n$ 开始，其修正耗散值 $q(n)$ 取估计 $h(n)$ 的所有值中最小的一个，然后根据耗散值递归计算公式逐级向上修正其先辈结点的耗散值，只有下层耗散值修正后，才可能影响上一层结点的耗散值，因此必须自底向上一一直修正到初始结点。这由算法中的内循环过程完成。

## AO\*算法与A算法的区别

- ❖ AO\*算法不能像A算法那样，单纯靠评价某一个结点来评价局部图。
- ❖ AO\*算法由于k-连接符连接的有关子结点，对父结点能解与否以及耗散值都有影响，因而显然不能像A算法那样优先扩展其中具有最小耗散值的结点。

## AO\*与A的区别

- ❖ AO\*算法仅适用于无环图的假设，否则耗散值递归计算不能收敛，因而在算法中还必须检查新生成的结点已在图中时，是否是正被扩展结点的先辈结点。

## AO\*与A的区别

- ❖ A算法有OPEN表和CLOSED表，而AO\*算法只用一个与或图结构，它代表到目前为止已显式生成的部分搜索图，图中每一个结点的 $h(n)$ 值是估计最佳解图，而不是估计解路径。

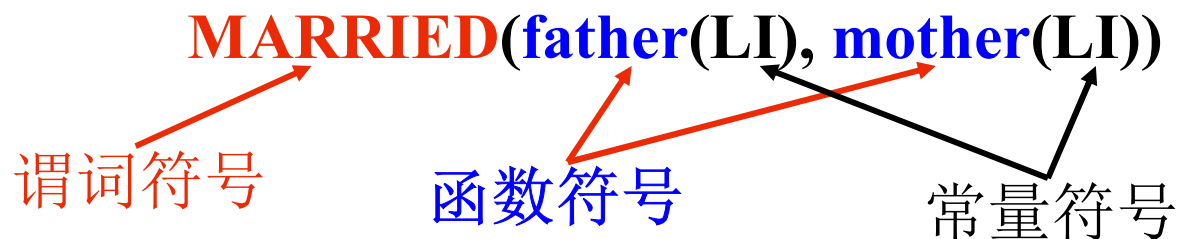
## 2.3 谓词逻辑表示 (Predict Logic)

- 逻辑语句：一种形式语言，它能够把逻辑论证符号化，并用于证明定理，求解问题。
- 形式语言：严格地按照相关领域的特定规则，以数学符号（符号串）形式描述该领域有关客体的表达式。

### 2.3.1 谓词演算

#### 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**：由若干谓词符号和项组成的谓词演算。  
原子公式是谓词演算的基本积木块。

如：

**INROOM(ROBOT, r1)**

（机器人在1号房间内）



## 连词和量词 (Connective & Quantifiers)

■ 连词( $\wedge$ ,  $\vee$ ,  $\Rightarrow$ ,  $\sim$ )

与及合取 (conjunction): 用连词 $\wedge$ 把几个公式连接起来而构成的公式。合取项是合取式的每个组成部分。

例:  $\text{LIKE}(\text{I}, \text{MUSIC}) \wedge \text{LIKE}(\text{I}, \text{PAINTING})$

(我喜爱音乐和绘画。)

$\text{LIVES}(\text{L1}, \text{HOUSE-1}) \wedge \text{COLOR}(\text{HOUSE-1}, \text{YELLOW})$

(李住在一幢黄色的房子。)

**或及析取** (disjunction) : 用连词  $\vee$  把几个公式连接起来而构成的公式。析取项是析取式的每个组成部分

例:

$\text{PLAYS}(\text{LILI}, \text{BASKETBALL}) \vee \text{PLAYS}(\text{LILI}, \text{FOOTBALL})$

(李力打篮球或踢足球。)

**蕴涵 (Implication)** : “ $\Rightarrow$ ”表示“如果—那么”  
(IF—THEN) 关系, 其所构成的公式叫做蕴涵。  
蕴涵的左式叫做前件, 后式叫做后件。

例:  $\text{RUNS (LIUHUA, FASTEST)} \Rightarrow \text{WINS (LIUHUA, CHAMPION)}$

**非 (Not)** : 表示否定,  $\sim$ 、 $—$ 均可表示, 用来否定一个公式的真值。

例:  $\sim \text{INROOM (ROBOT, } r2 \text{ )}$

- 例1：李明是个学生，他住的房间是白色的。

**$\text{Isa}(\text{Liming}, \text{Student}) \wedge \text{Lives}(\text{Liming}, \text{House1}) \wedge \text{Color}(\text{House1}, \text{White})$**

- 例2：李明在学校的话，Wang也在学校。

**$\text{At}(\text{Liming}, \text{School}) \Rightarrow \text{At}(\text{Wang}, \text{School})$**

以上讲的是命题演算(谓词演算的一个子集), 但它缺乏用有效的方法来表达多个命题的能力, 如:

“所有机器人都是灰色的”

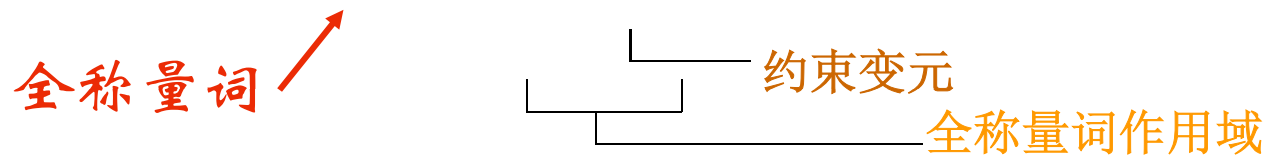
可以表示为:

$(\forall x)[\text{ROBOT}(x) \Rightarrow \text{COLOR}(x, \text{GRAY})]$

但命题演算就无法表示, 所以需要使公式中的命题带有变量。

## ⊕ 量词

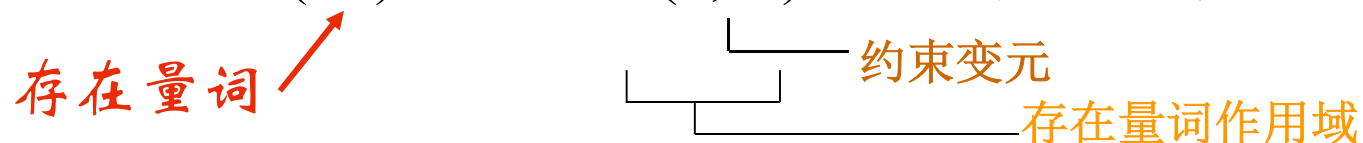
全称量词 (Universal Quantifier) : 若一个原子公式  $P(x)$  , 对于所有可能变量  $x$  都具有  $T$  值, 则用  $(\forall x) P(x)$  表示



## 存在量词 (Existential Quantifier)

若一个原子公式  $P(x)$  , 至少有一个变元  $x$  , 可使  $P(x)$  为  $T$  值, 则用  $(\exists x) P(x)$  表示。

例:  $(\exists x) \text{INROOM}(x, r1)$  (1号房间内有个物体)



例3：条条大路通罗马

$$(\forall x) [ \text{Road}(x) \Rightarrow \text{Lead}(x, \text{Roma}) ]$$

• 例4：Mary给每个人一本书

$$(\forall x) (\exists y) [ \text{Person}(x) \wedge \text{Book}(y) \wedge \text{Give}(\text{Mary}, x, y) ]$$

## 自由变元

$$(\forall y) [ P(y) \Rightarrow (\exists x) Q(x,y) ]$$

$y$ 在存在量词  $(\exists x)$  辖域内是自由变量，但在全称量词  $(\forall y)$  辖域内成为约束变量。



## 2.3.2 谓词公式

### ❁ 1、谓词公式的定义

- ❖ 用  $P(x_1, x_2, \dots, x_n)$  表示一个  $n$  元谓词公式，其中  $P$  为  $n$  元谓词， $x_1, x_2, \dots, x_n$  为客体变量或变元。通常把  $P(x_1, x_2, \dots, x_n)$  叫做谓词演算的原子公式，或原子谓词公式。

### ❁ 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)

❖ 在谓词演算中，合式公式的递归定义如下：

- (1) 原子谓词公式是合式公式。
- (2) 若 $A$ 为合式公式，则 $\sim A$ 也是一个合式公式。
- (3) 若 $A$ 和 $B$ 都是合式公式，则 $(A \wedge B)$ ， $(A \vee B)$ ， $(A \Rightarrow B)$ 和 $(A \Leftrightarrow B)$ 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 $(\forall x)A$ 和 $(\exists x)A$ 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。

## ❁ 2、合式公式的性质

### ❁ 合式公式的真值

表2.1 真值表

$P$	$Q$	$P \vee Q$	$P \wedge Q$	$P \Rightarrow Q$	$\sim P$
T	T	T	T	T	F
F	T	T	F	T	T
T	F	T	F	F	F
F	F	F	F	T	T

### ❁ 等价 (Equivalence)

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。

- (1) 否定之否定:  $\sim(\sim P)$ 等价于 $P$
- (2)  $P \vee Q$ 等价于 $\sim P \Rightarrow Q$
- (3) 狄·摩根定律:  $\sim(P \vee Q)$ 等价于 $\sim P \wedge \sim Q$   
 $\sim(P \wedge Q)$ 等价于 $\sim P \vee \sim Q$
- (4) 分配律:  $P \wedge (Q \vee R)$ 等价于 $(P \wedge Q) \vee (P \wedge R)$   
 $P \vee (Q \wedge R)$ 等价于 $(P \vee Q) \wedge (P \vee R)$
- (5) 交换律:  $P \wedge Q$ 等价于 $Q \wedge P$   
 $P \vee Q$ 等价于 $Q \vee P$
- (6) 结合律:  $(P \wedge Q) \wedge R$ 等价于 $P \wedge (Q \wedge R)$   
 $(P \vee Q) \vee R$ 等价于 $P \vee (Q \vee R)$
- (7) 逆否律:  $P \Rightarrow Q$ 等价于 $\sim Q \Rightarrow \sim P$

(8)  $\sim(\exists x)P(x)$  等价于  $(x) [\sim P(x)]$

$\sim(\forall x)P(x)$  等价于  $(\exists x) [\sim P(x)]$

(9)  $(\forall x) [P(x) \wedge Q(x)]$  等价于  $(\forall x)P(x) \wedge (\forall x)Q(x)$

$(\forall x) [P(x) \vee Q(x)]$  等价于  $(\forall x)P(x) \vee (\forall x)Q(x)$

(10)  $(\forall x)P(x)$  等价于  $(\forall y)P(y)$

$(\exists x)P(x)$  等价于  $(\exists y)P(y)$

## 问题五

在代数课程中，你知道  
“置换”的概念是什么？



## 2.3.3 置换与合一

### ❊ 置换

❊ 概念

❊ 假元推理

$$\left. \begin{array}{l} W_1 \Rightarrow W_2 \\ W_1 \end{array} \right\} \rightarrow \text{产生 } W_2$$

❖ 全称化推理

$$(\forall x) W(x) \rightarrow W(A)$$

└——约束变元    └——任意变量

❖ 综合推理

$$\left. \begin{array}{l} (\forall x) [W_1(x) \rightarrow W_2(x)] \\ W_1(A) \end{array} \right\} \rightarrow \text{产生 } W_2(A)$$

❖ **置换**的定义：就是在表达式中用置换项置换变量。如果用 $E$ 表示表达式， $s$ 为一置换，则置换后的表达式记为 $Es$ 。

❖ **性质**

❖ 可结合律  $(Ls1) s2 = L (s1s2)$   
 $(s1s2) s3 = s1 (s2s3)$

❖ 不可交换律  $s1s2 \neq s2s1$



例如：

表达式  $P[x, f(y), B]$  的4个置换为

$$s1 = \{z/x, w/y\} \quad \text{则} \quad P[x, f(y), B]s1 = P[z, f(w), B]$$



$$s2 = \{A/y\} \quad \text{则} \quad P[x, f(y), B]s2 = P[x, f(A), B]$$



$$s3 = (q(z)/x, A/y) \quad \text{则} \quad P[x, f(y), B]s3 = P[q(z), f(A), B]$$



$$s4 = (c/x, A/y) \quad \text{则} \quad P[x, f(y), B]s4 = P[c, f(A), B]$$



## ❁ 合一 (Unification)

- ❁ 合一：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 $\{E_i\}$ 的每个元素，则我们用 $\{E_i\} s$ 来表示置换例的集。

- ❁ 可合一：称表达式集 $\{E_i\}$ 是可合一的，如果存在一个置换 $s$ 使得：

$$E_1 s = E_2 s = E_3 s = \dots$$

$s$ 称为 $\{E_i\}$ 的合一者。

例如：

表达式集  $\{P[x, f(y), B], P[x, f(B), B]\}$

的合一者为

$$s = \{A/x, B/y\}$$

因为

$$\left. \begin{array}{l} P[x, f(y), B]s = P[x, f(B), B]s \\ P[x, f(B), B]s = P[A, f(B), B] \end{array} \right\} \text{单一形式}$$

所以  $s = \{A/x, B/y\}$  是  $\{P[x, f(y), B], P[x, f(B), B]\}$  的合一者

而  $s = \{B/y\}$  是  $\{P[x, f(y), B], P[x, f(B), B]\}$  最简单的合一者

## 2.4 语义网络表示 (Semantic Network Representation)

- ❖ 由奎廉（Quillian）于1968年提出，作为描述人类联想记忆的一种心理学模型。

- ❖ 语义网络的结构

  - 🟢 定义

语义网络是知识的一种图解表示，它由**节点**和**弧线或链**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。

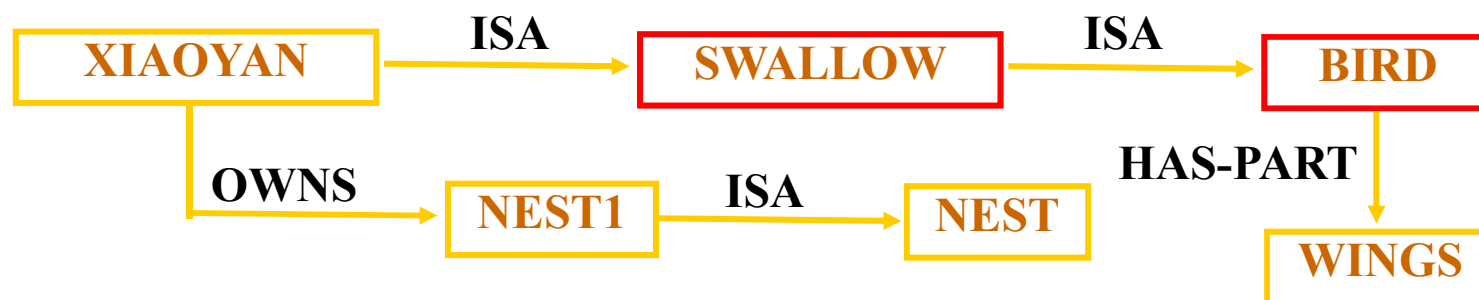
- 组成部分

- 词法 决定表示词汇表中允许有哪些符号，它涉及各个节点和弧线。
- 结构 叙述符号排列的约束条件，指定各弧线连接的节点对。
- 过程 说明访问过程，这些过程能用来建立和修正描述，以及回答相关问题。
- 语义 确定与描述相关的(联想)意义的方法即确定有关节点的排列及其占有物和对应弧线。

## 2.4.1 二元语义网络的表示

### ✿ 表示简单事实和占有关系

例 所有的燕子(SWALLOW)都是鸟(BIRD)

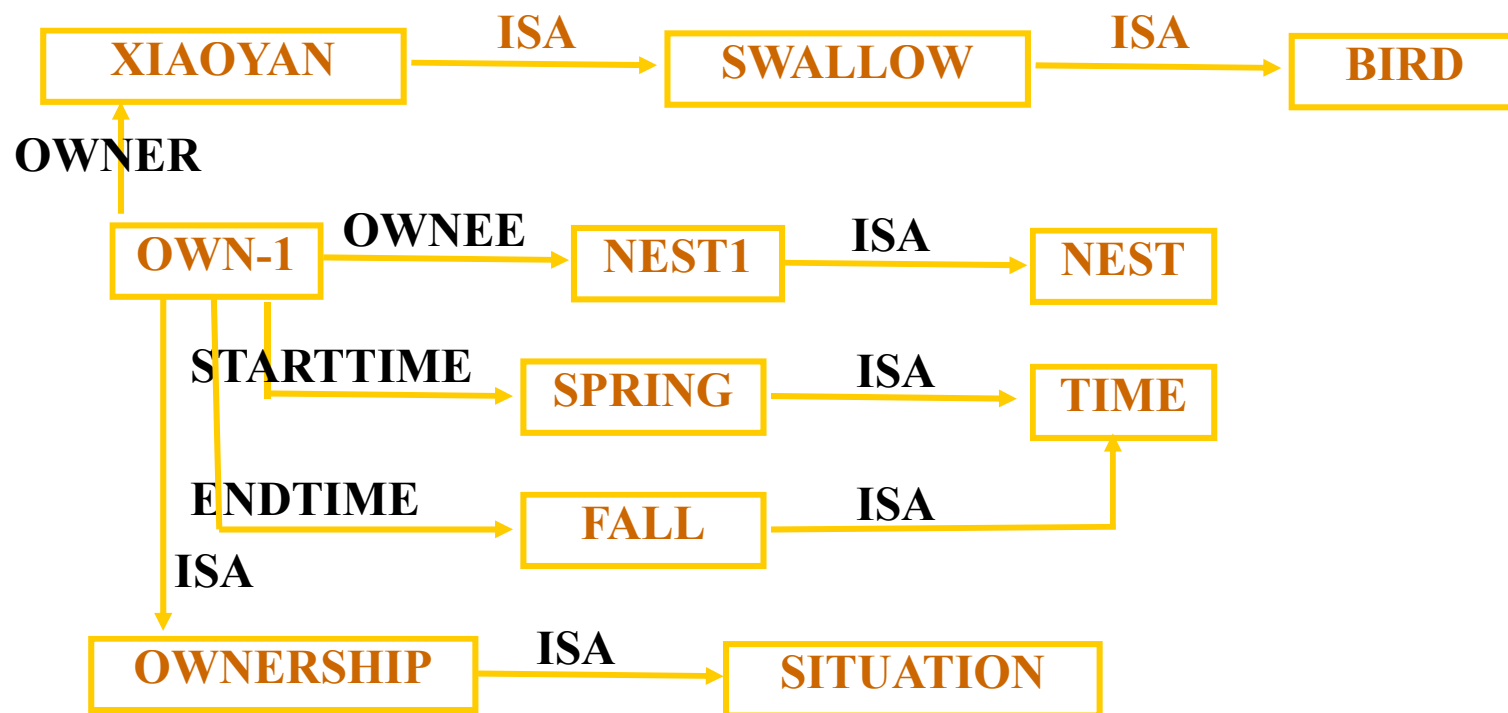


我们希望表示“小燕子(XIAOYAN)是一只燕子”

我们希望表示“鸟有翅膀”

我们希望表示“小燕子有一个巢(nest)”


我们希望把“小燕从春天到秋天占有一个巢”的信息加到网络中去。



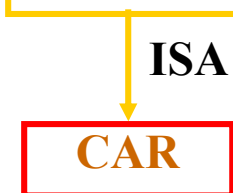
## ❁ 选择语义基元问题

- ❁ 就是试图用一组基元来表示知识，以便简化表示，并可用简单的知识来表示更复杂的知识。

“我的汽车是棕黄色的”表示为：



```
graph LR; MC[MY CAR] -- COLOR --> TAN[TAN]
```



“李华的汽车是棕绿色的”表示为：

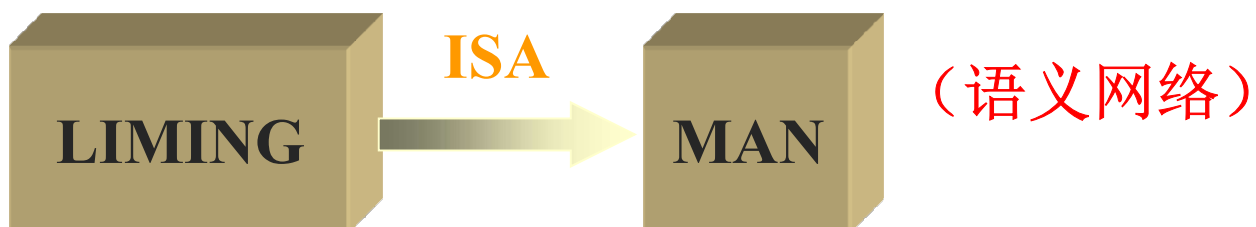


## 2.4.2 多元语义网络的表示

- 表示二元关系

李明是一个人：

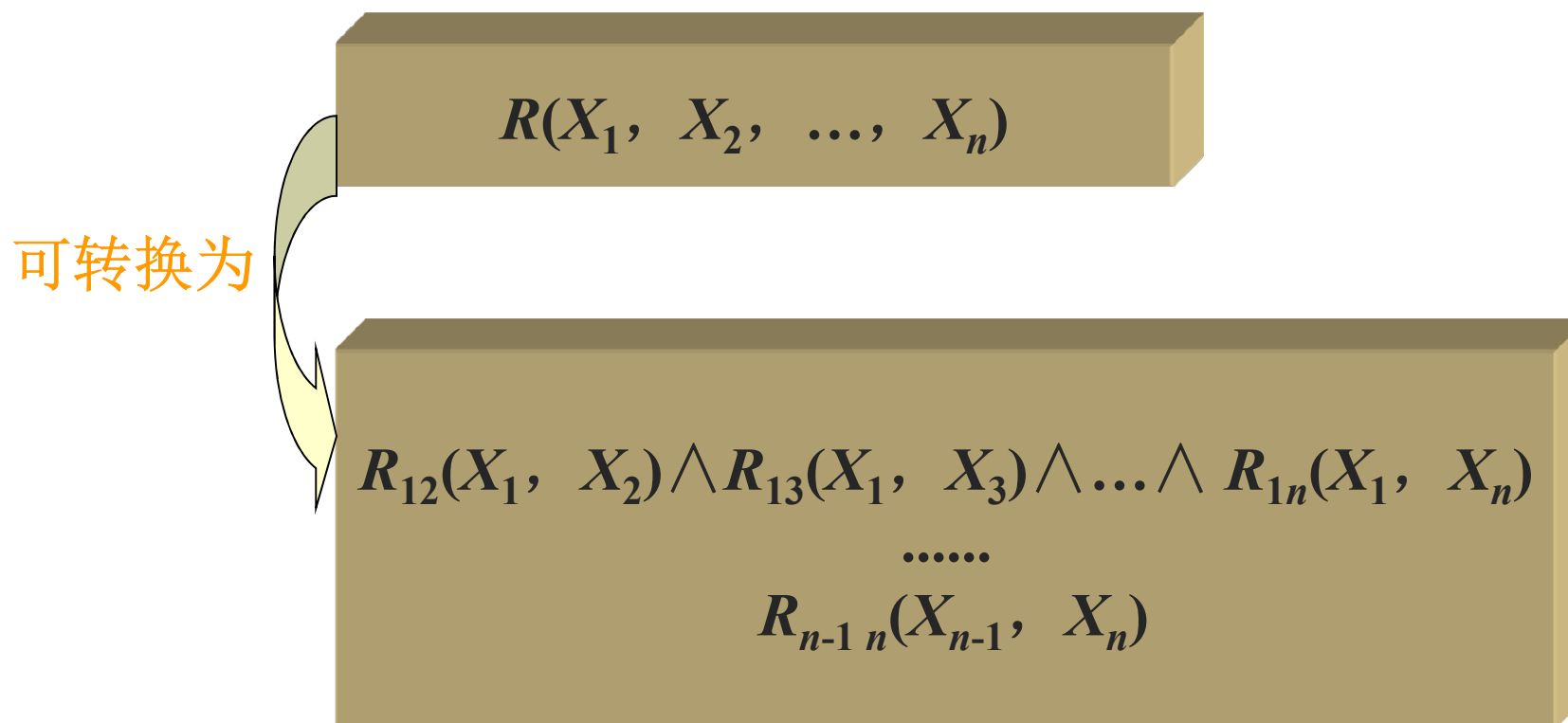
ISA (LIMING, MAN) 或 MAN (LIMING) (谓词逻辑)



说明：语义网络可以毫无困难地表示二元关系

## •表示多元语义

- 把多元关系转化为一组二元关系的组合，或二元关系的合取。

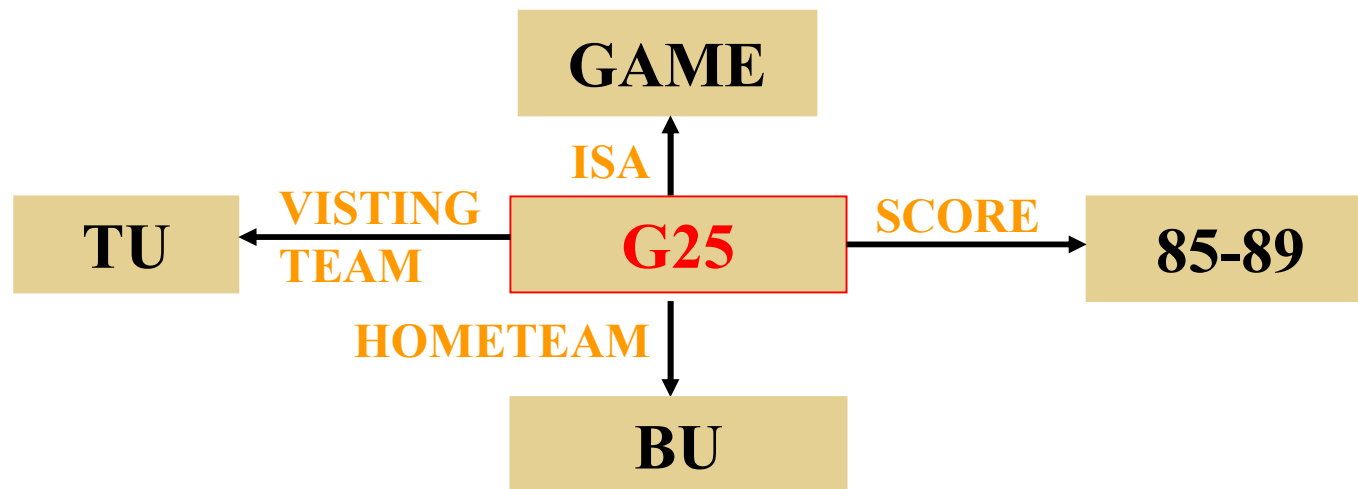


在语义网络中进行上述转换需要引入**附加节点**

例如，要表达北京大学(BEIJING University,简称BU)和清华大学(TSINGHUA University,简称TU)两校篮球队在北大进行的一场比赛的比分是85比89。

谓词逻辑: **SCORE(BU, TU, (85:89))**

语义网络:



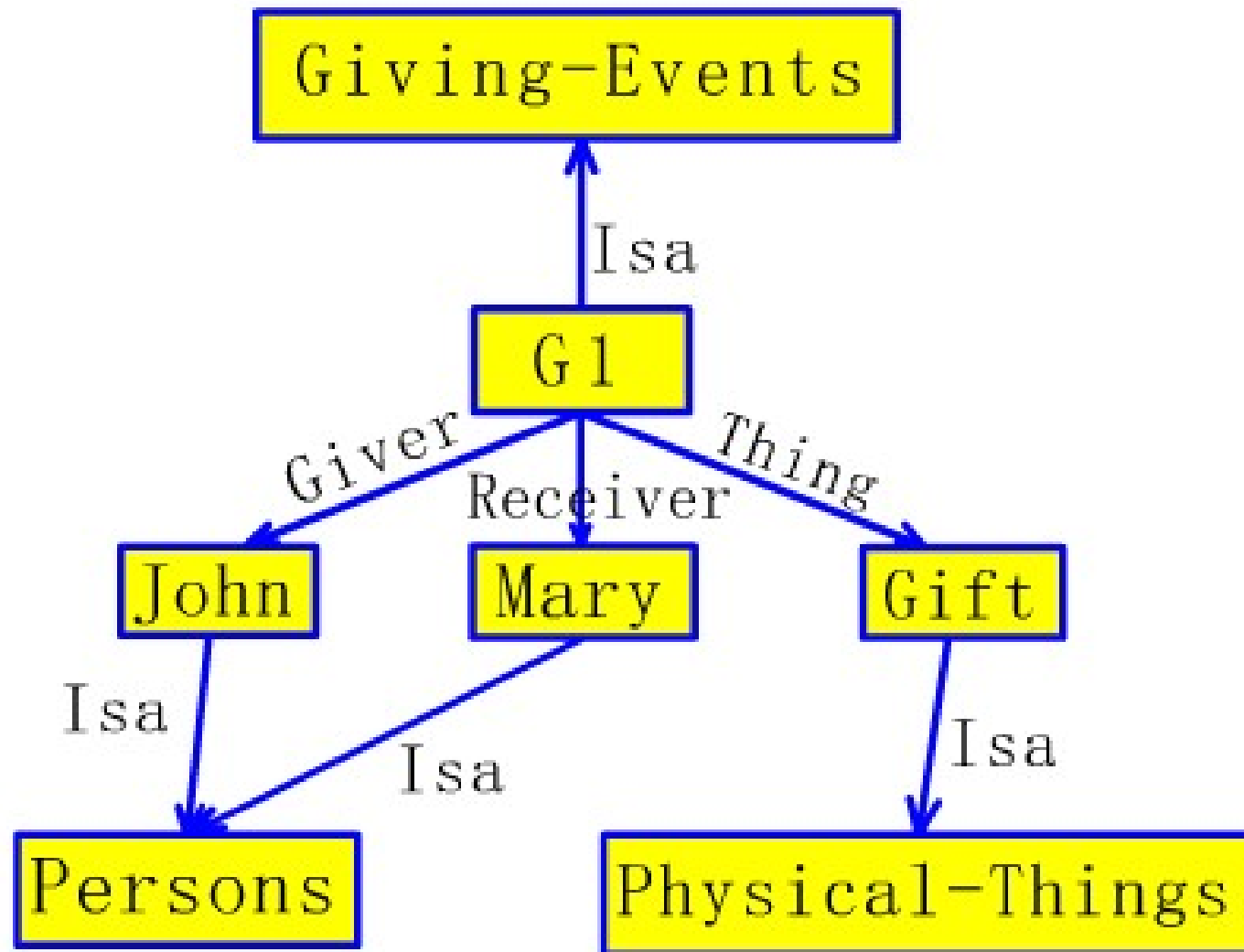
例如, John给Mary一个礼物

**Gives(John, Mary, Gift)。**

❏ 将三元关系转变为多个2元关系的合取

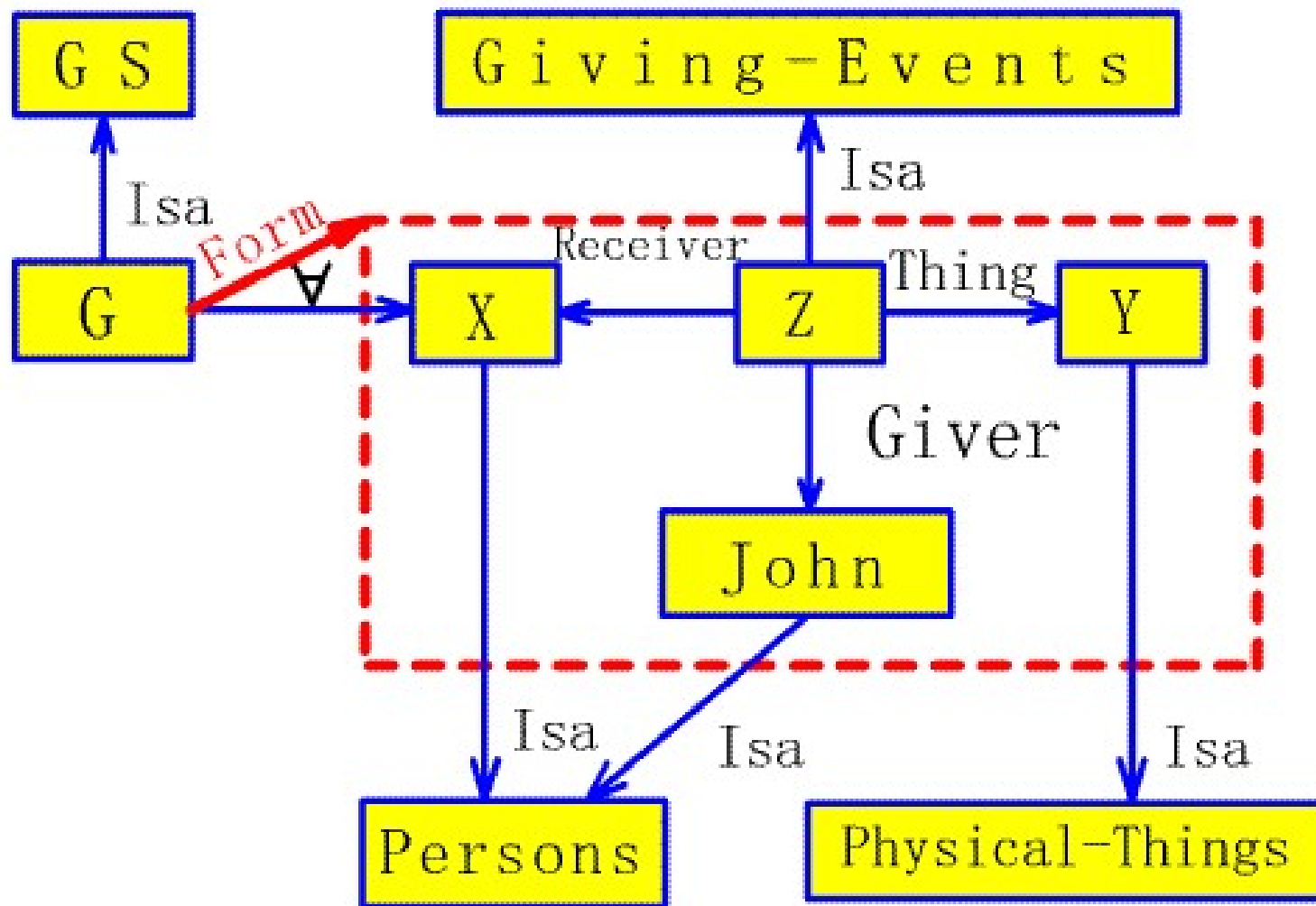
- 整个描述表示为一个给出事件G1, 使其作为事件类 Giving-Event的一个例子
- G1中的John是给出者 (Giver)
- Mary是接受者 (Receiver)
- Gift则是给出的东西 (Thing)

**$\text{Isa}(\text{G1}, \text{Givig-Event}) \wedge \text{Giver}(\text{G1}, \text{John}) \wedge \text{Receiver}(\text{G1}, \text{Mary})$   
 $\wedge \text{Thing}(\text{G1}, \text{Gift})$**

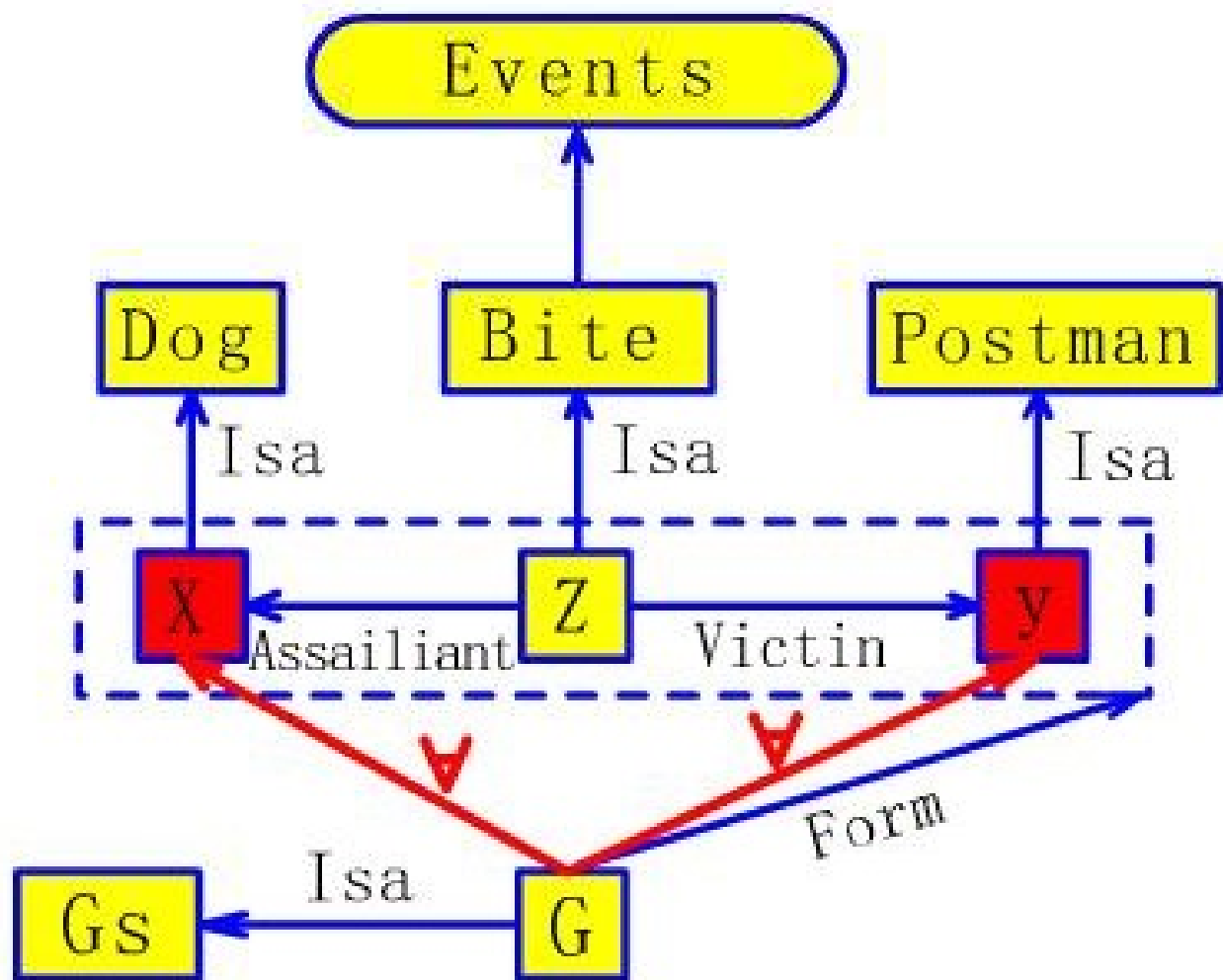


例如，John给每个人一个礼物

$\text{Isa}(G1, \text{Givig-Event}) \wedge \text{Giver}(G1, \text{John}) \wedge \text{Receiver}(G1, \text{Mary})$   
 $\wedge \text{Thing}(G1, \text{Gift})$



- 涉及2个全称量词约束变量的语义网络

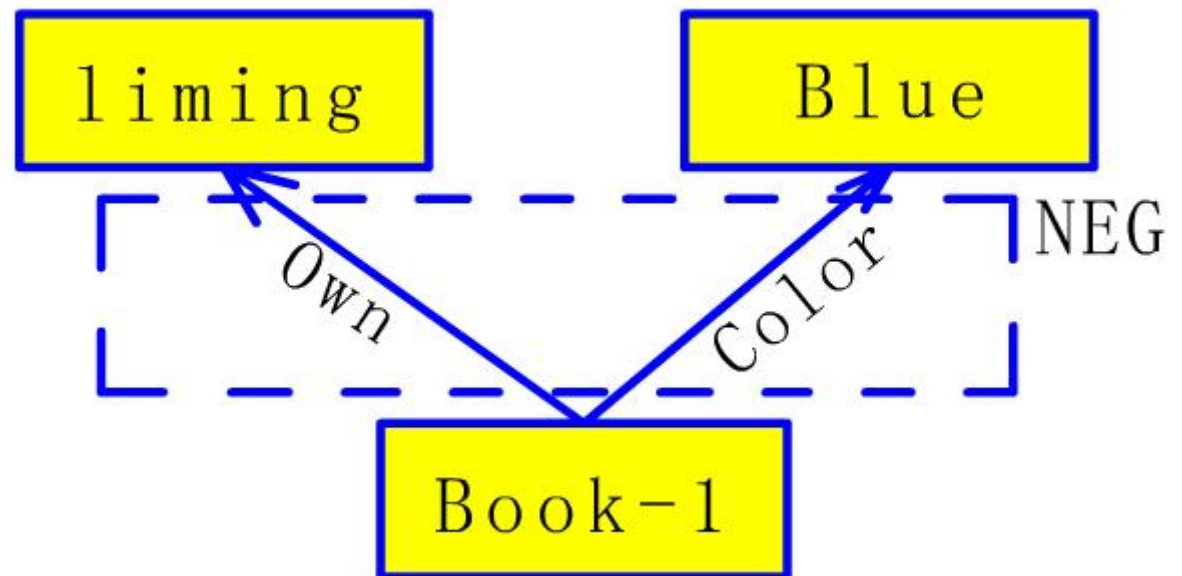
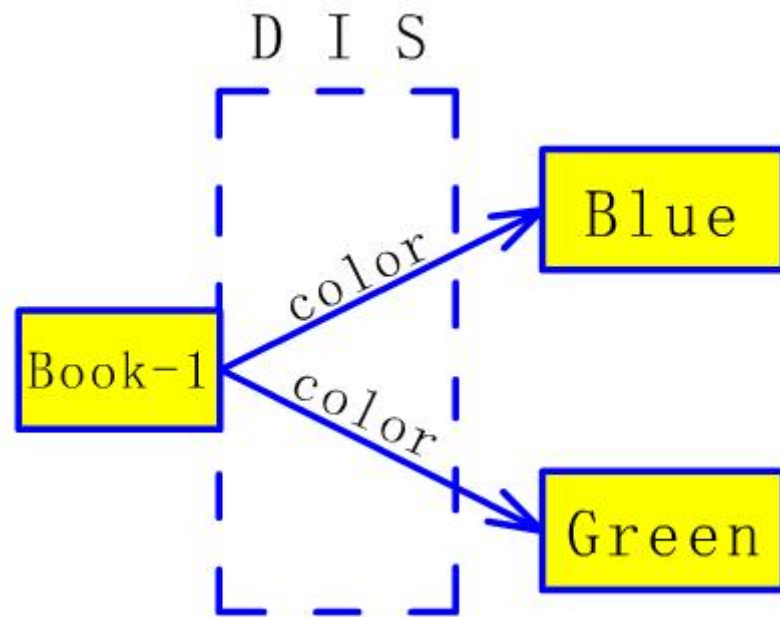


## ❁ 逻辑关系的表示

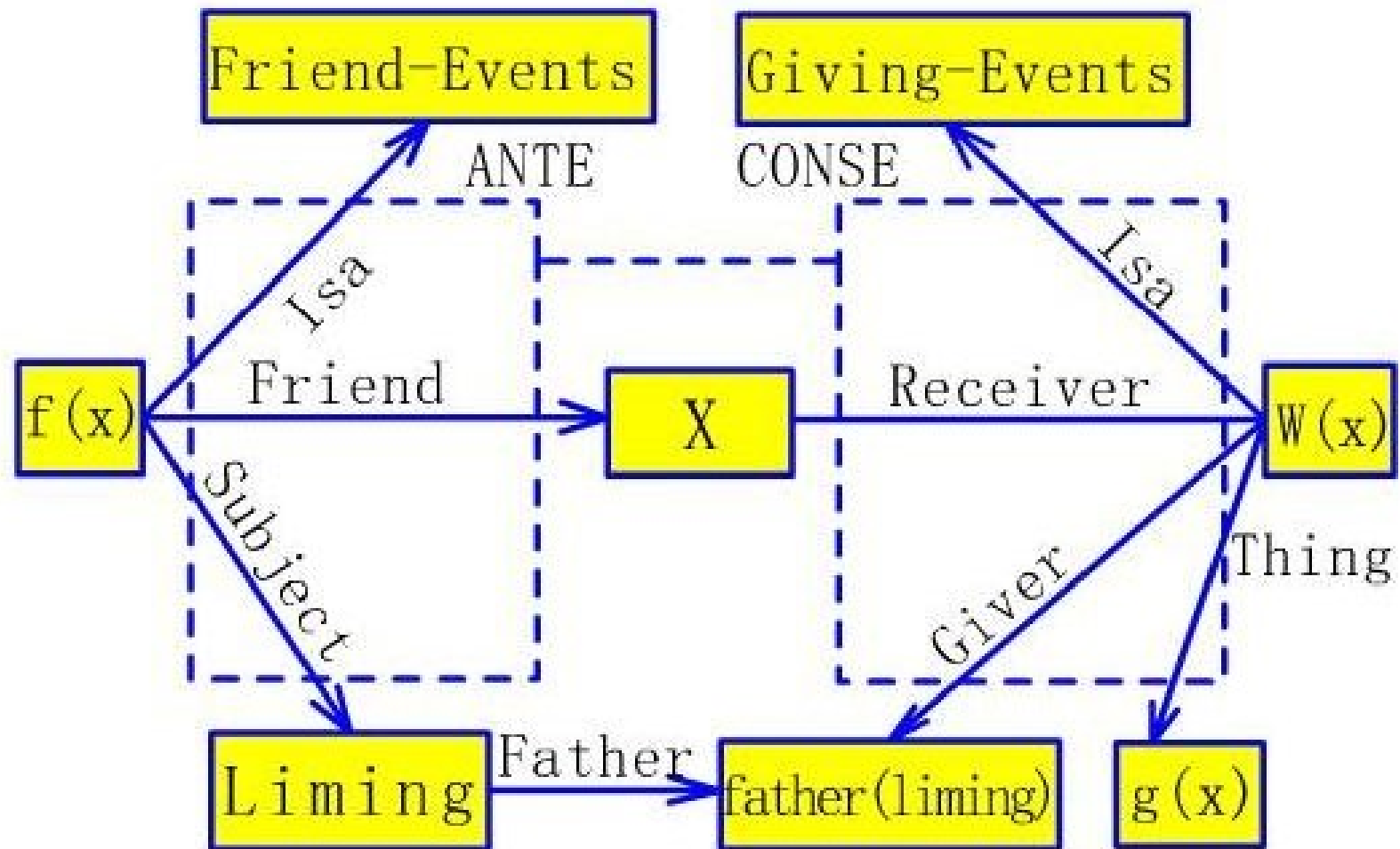
用虚线框将具有某种逻辑关系的关系弧围起来

- ❁ 与：不必作附加处理
- ❁ 或：当2条（或多条）关系弧有“或”关系时，可以用虚线框将在这些弧围起来，并在虚线框上加标记DIS（disjunction）。
- ❁ 非：用加标签NEG(negative)的虚线框围起2条（或多条）关系弧
- ❁ 蕴含：以加标签ANTE(antecedent)的虚线框围住描述蕴涵前项的关系弧；以加标签CONSE(consequent) 的虚线框围住描述蕴涵后项的关系弧；然后再用一条虚线将这两个虚线框连接起来，以表示它们属于同一个蕴涵关系。





李明的父亲给李明的每个朋友一个礼物



## 2.4.3 语义网络的推理过程

语义网络中的推理过程主要有两种，一种是继承，另一种是匹配。

### (1) 继承

继承就是把对事物的描述从概念节点或类节点传递到实例节点。

- ❖ Pass（遗留）— the descendants get the properties from the ancestors
- ❖ Add（添加）—the descendants expand the properties of the ancestors
- ❖ Exclude(排除) — stop inheriting the properties

# Implication for SN

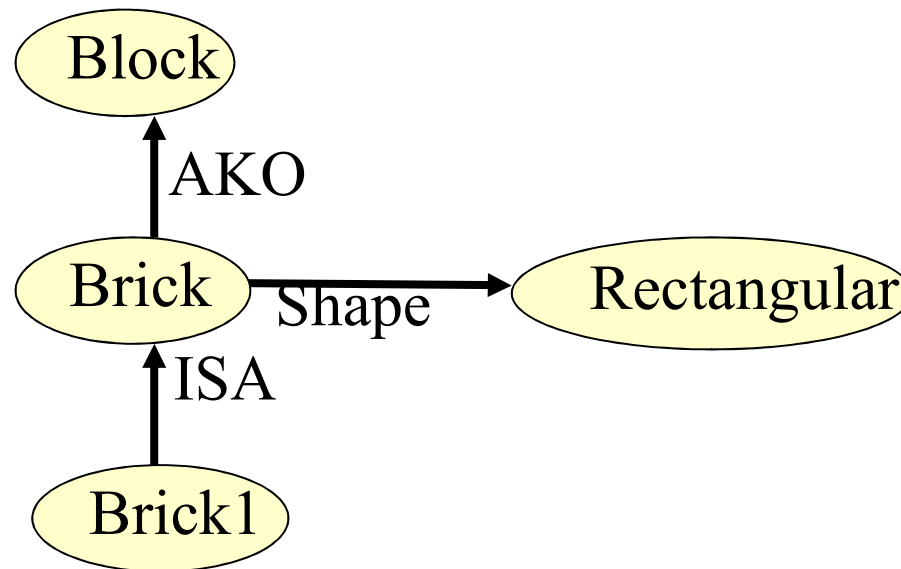
这种推理过程，类似于人的思维过程。一旦知道了某种事物的身份以后，可以联想起很多关于这件事物的一般描述。例如，通常认为鲸鱼很大，鸟比较小，城堡很古老，运动员很健壮等。

一共有3种继承过程：值继承、“如果需要”继承和“缺省”继承。

# Implication for SN

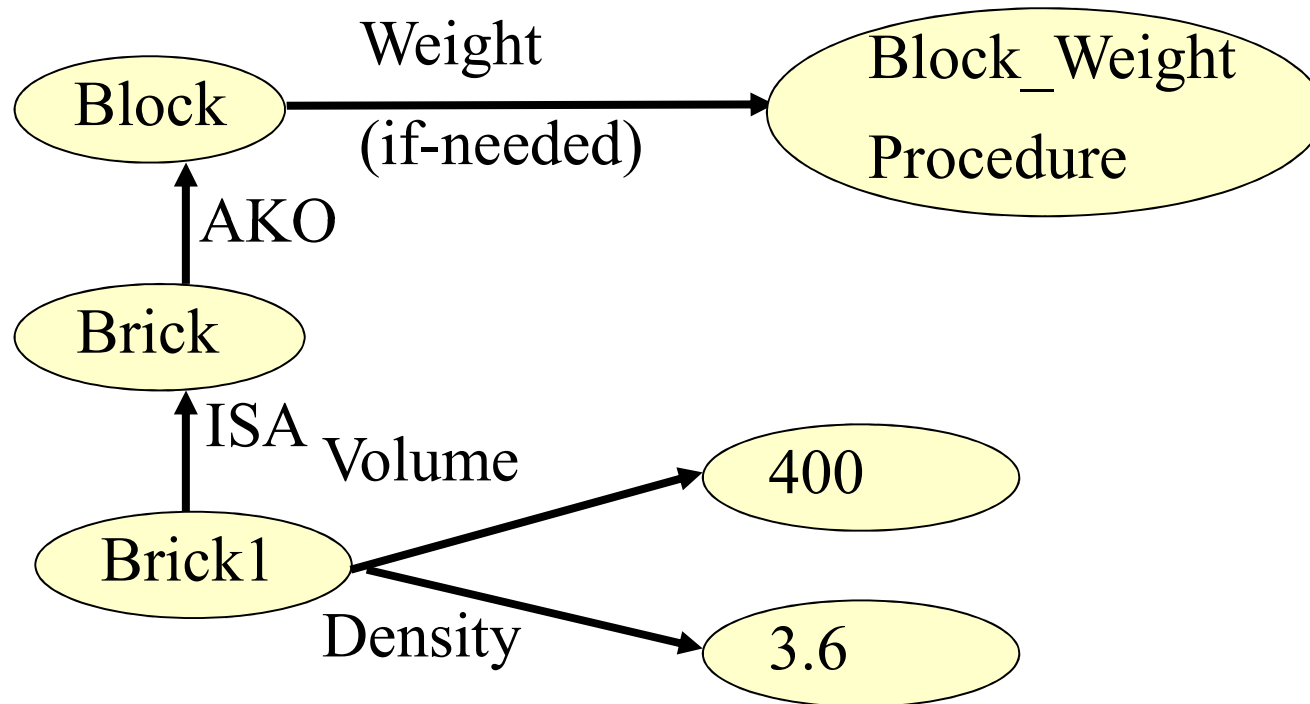
值继承: is-a, AKO(a kind of)

What is the shape of Brick1?



# Implication for SN

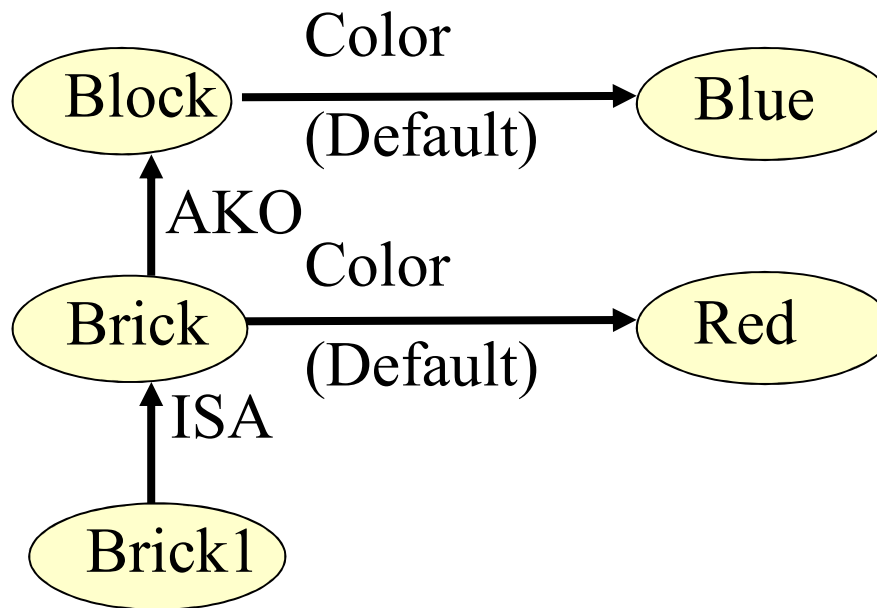
- ❏ “如果一需要” 继承: if-needed (it can not inherit from the ancestors, we get it from other programs )



# Implication for SN

❏ “缺省”继承： Default—it is mostly the truth

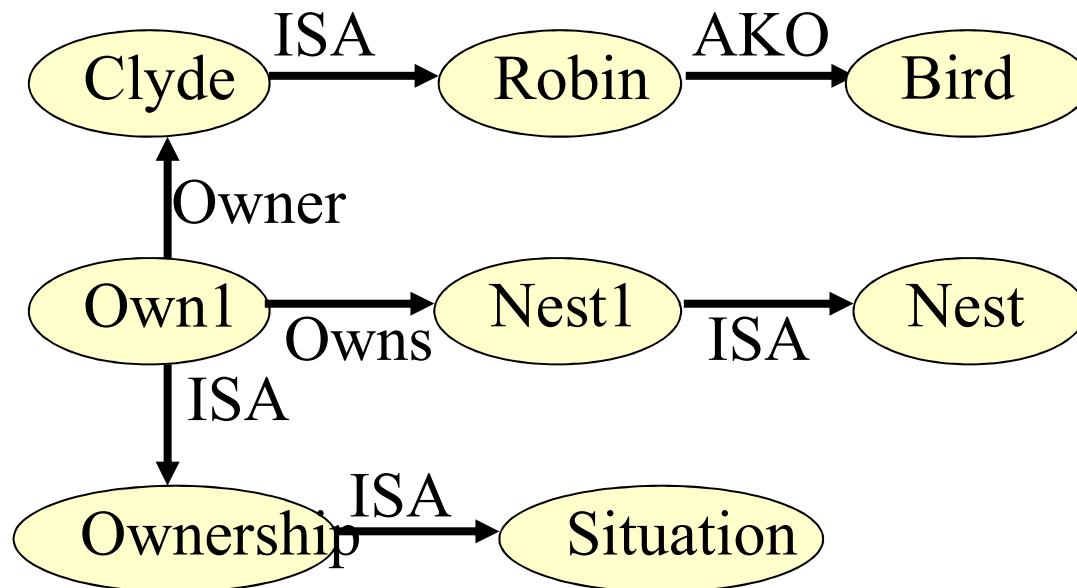
Exp: the birds can fly;



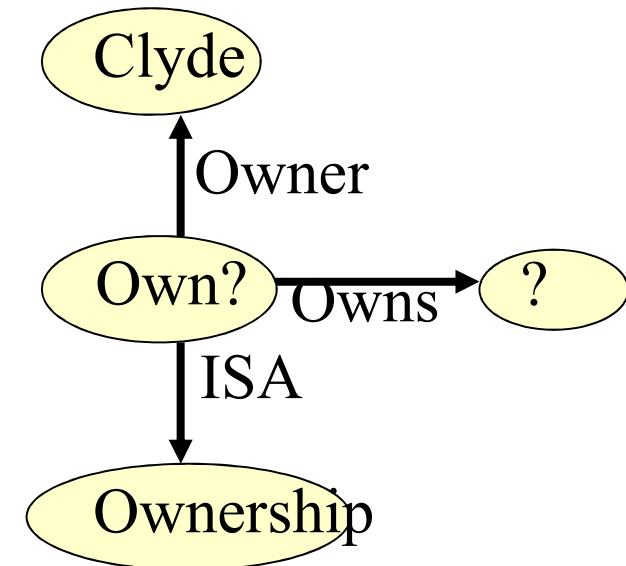
# Implication for SN

## (2) 匹配 (Matching)

当涉及由几个部分组成的事物时，必须考虑值的传递问题。



What does Clyde own?





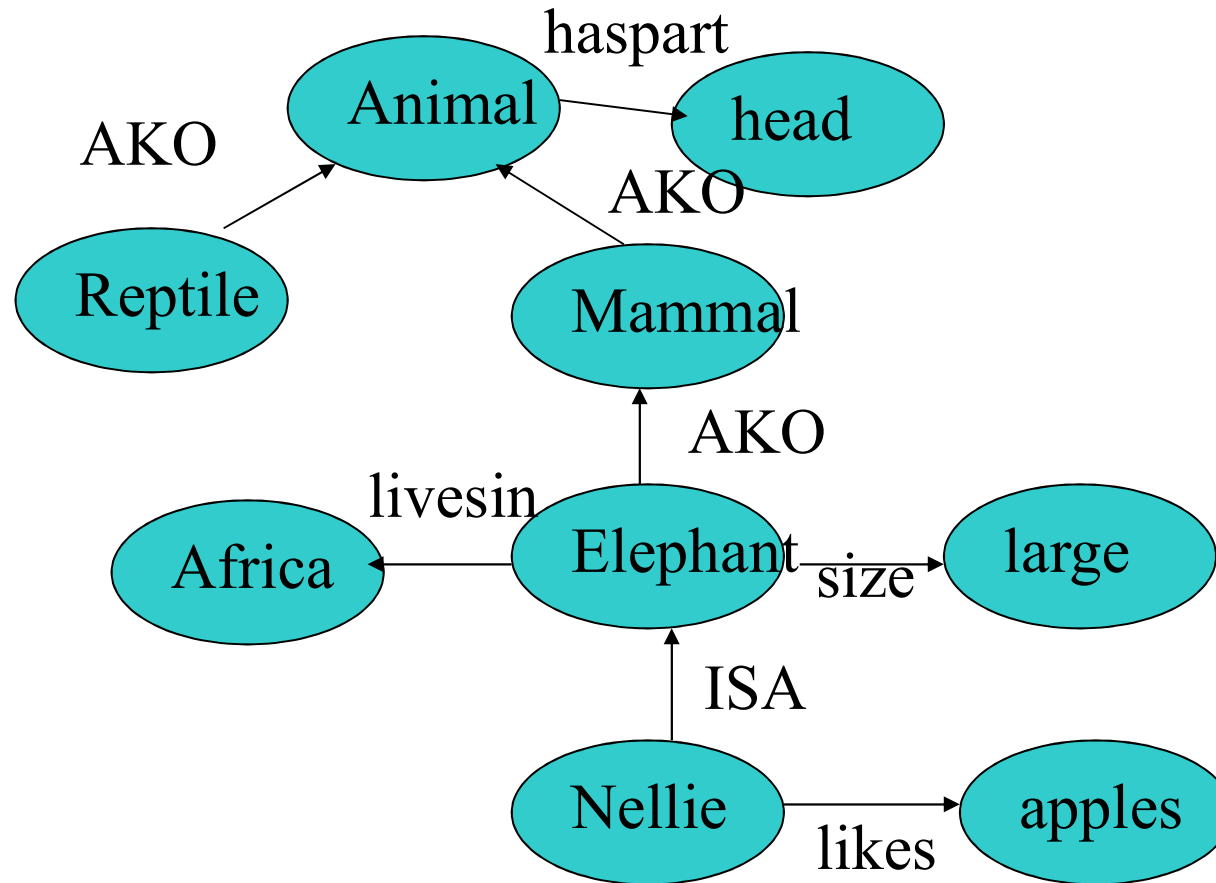
# Semantic Network

## ✚ Example 1 :

Use semantic networks to represent the followings:

- ✚ Nellie is an elephant,
- ✚ he likes apples.
- ✚ Elephants are a kind of mammals,
- ✚ they live in Africa,
- ✚ and they are big animals.
- ✚ Mammals and reptiles are both animals,
- ✚ all animals have head.

# Semantic Network



## 问题六

你是否知道“512”汶川  
地震的基本情况？



## 2.5 框架表示 (Frame Representation)

### ✿ 框架 (Frame) 表示

框架是一种数据结构，在这个结构中，新的资料可以从过去的经验中得到的概念来分析和解释。

框架是一种结构化知识表示法，通常采用语义网络中的节点-槽-值表示结构。这组节点和槽可以描述格式固定的事物、行动和事件。

## 2.5.1 框架的构成

框架通常由描述事物的各个方面的槽组成，每个槽可以拥有若干个侧面，而每个侧面又可以拥有若干个值。这些内容可以根据具体问题的具体需要来取舍，一个框架的一般结构如下：

<框架名>

<槽1><侧面11><值111>...

    <侧面12><值121>...

    ...

<槽2><侧面21><值211>...

    ...

...

<槽 $n$ ><侧面 $n1$ ><值 $n11$ >...

    ...

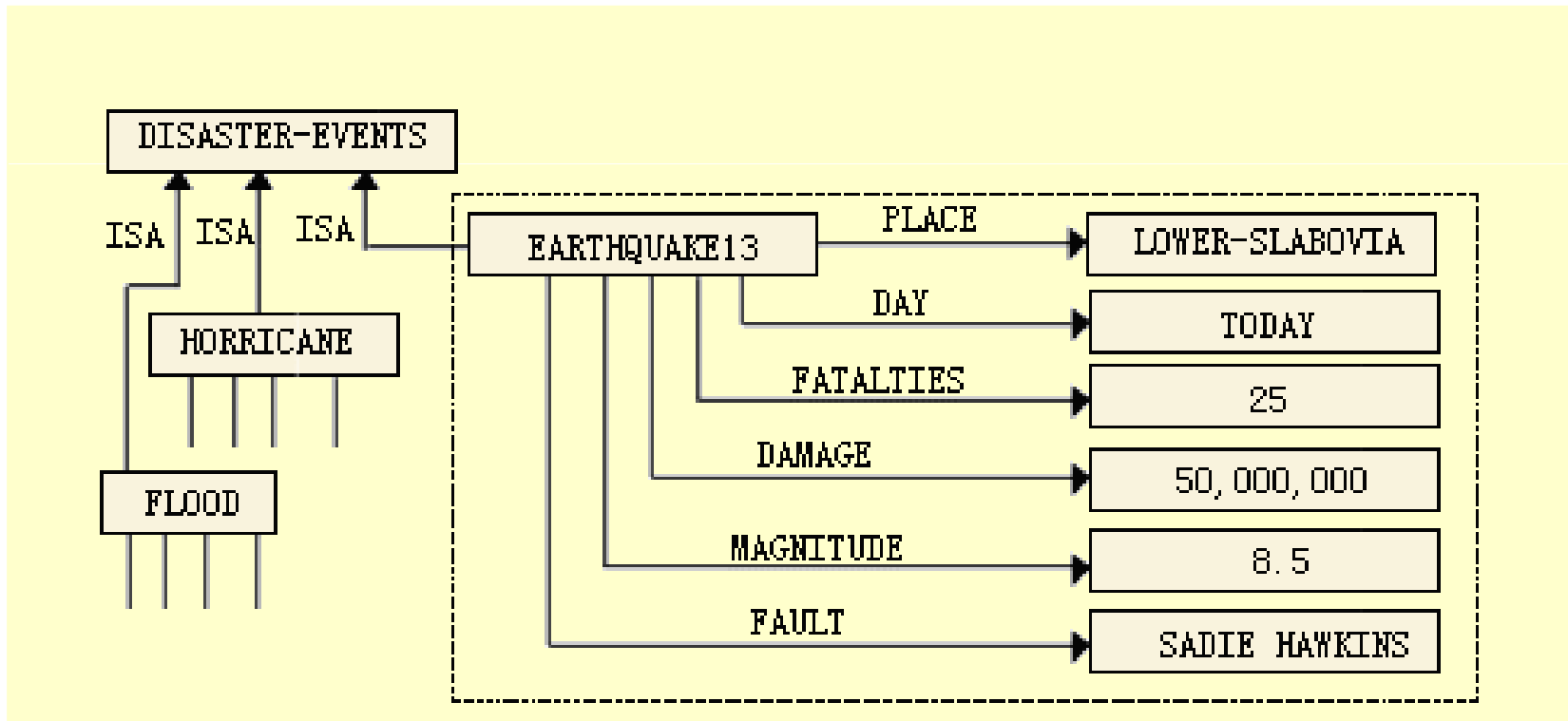
    <侧面 $nm$ ><值 $nm1$ >...

例如，一个人可以用其职业、身高和体重等项描述，因而可以用这些项目组成框架的槽。当描述一个具体的人时，再用这些项目的具体值填入到相应的槽中。下表给出了描述John的框架。

JOHN		
ISA	:	PERSON
Profession	:	PROGRAMMER
Height	:	1.8m
Weight	:	79kg

例如，以下是一段关于某次地震的报道：

"今天一次强度为里氏8.5级的强烈地震袭击了下斯洛文尼亚(Low Slabovia)地区，造成25人死亡和5亿美元的财产损失。下斯洛文尼亚地区主席说，多年来，靠近萨迪豪金斯断层的重灾区一直是一个危险地区。"



## 2.5.2 框架的推理

- 1、框架包含他所描述的情况或物体的多方面的信息
- 2、框架包含物体必须具有的属性
- 3、框架描述他们所代表的概念的典型事例



## 2.6 本体技术 (Ontology Technology)

### 2.6.1 本体的概念

- ✚ Gruber 1993, 本体是概念化的一个显示的规范说明或表示。
- ✚ Guarino & Giaretta 1995, 本体是概念化某些方面的一个显示规范说明或表示。
- ✚ Borst 1997, 本体可定义为被共享的概念化的一个形式规范说明。

## 2.6.2 本体的组成与分类

### ❁ 本体的组成

- ❁ 概念：除一般意义上的概念外，也可以是任务、功能、行为、策略、推理过程等。
- ❁ 关系：表示概念之间的一类关联。
- ❁ 函数
- ❁ 公理：用于描述一些永真式。
- ❁ 实例：属于某个概念的具体实例。

### ❁ 本体的分类

- ❁ 知识表示本体
- ❁ 通用常识本体
- ❁ 领域本体
- ❁ 语言学本体
- ❁ 任务本体

## 2.6.3 本体的建模

### ✚ 本体建模方法

#### ✚ 建模过程可分为

- 非形式化阶段：用自然语言和图表来描述
- 形式化阶段：通过知识表示方法对本体模型进行编码

#### ✚ 本体建模方法

- “骨架”法、“评价法”（TOVE）、KACTUS工程法、Methontology法、SENSUS法、五阶段法

#### ✚ 五阶段法

#### ✚ 语言学本体

#### ✚ 任务本体

## 实例：五阶段法对**NUDT5**本体建模

- ✿ NUDT5本体：用于信息系统与管理各实验室的知识管理
- ✿ 阶段1：数据收集与分析
  - ✦ 从组织的文档、报告中抽取出有关的概念：“Something、Entity、Document、Person、OrganizationGroup、SomeRelation、Author、FamilyName、Title”等
- ✿ 阶段1：建立一个字典
  - ✦ Entity：独立存在的能区别于其它东西的东西；
  - ✦ Document：包含可以表示思想的元素的实体；
  - ✦ Author：表示一个文档被一个人创造的关系；
  - ✦ Title：指定一个文档的文字。

### ❖ 阶段3：对字典进行求精，建立内容更丰富的表

顶层概念表

类	父类	自然语言定义
Something		一种有形的、无形的或者抽象的存在的东西
Entity	Something	独立存在的、能区别于其它东西的东西
...	...	.....

中间层概念表

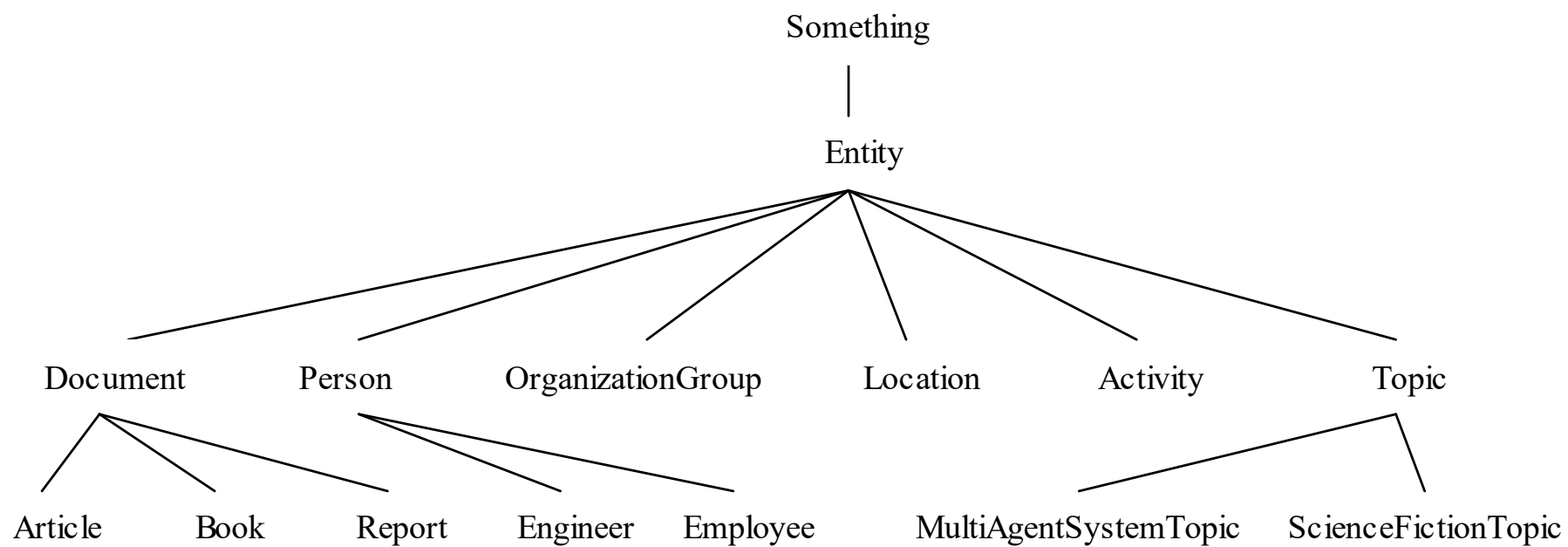
类	父类	自然语言定义
Document	Entity	包含可以表示思想的元素的实体
Person	Entity	人类的一个单独的个体
...	...	.....

### 顶层关系表

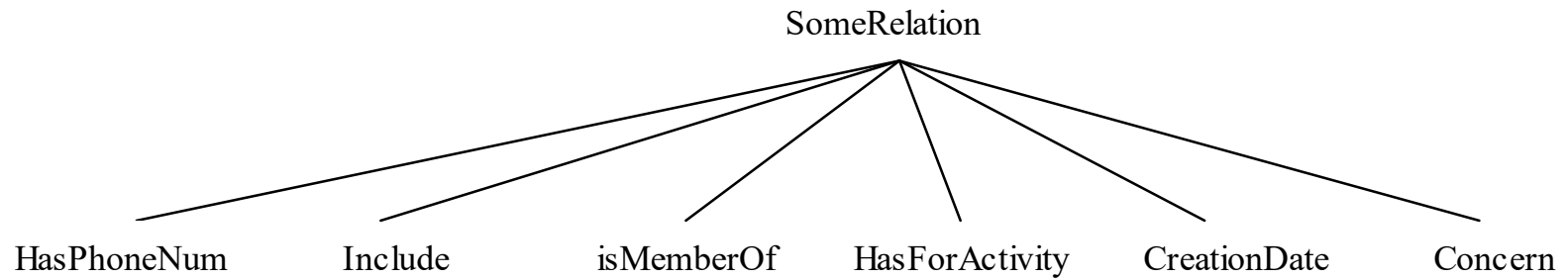
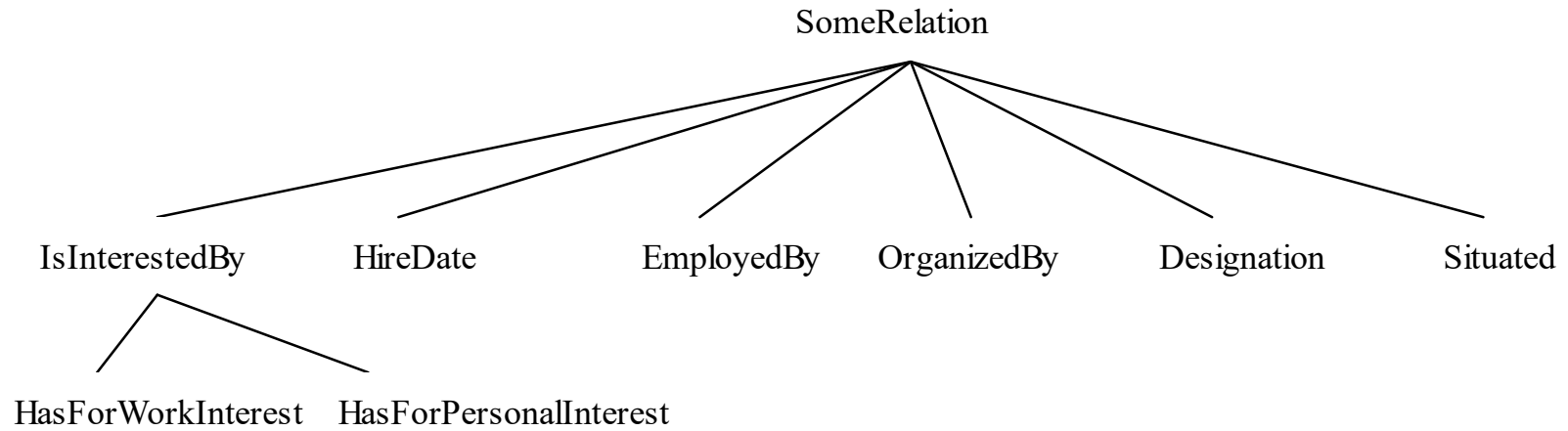
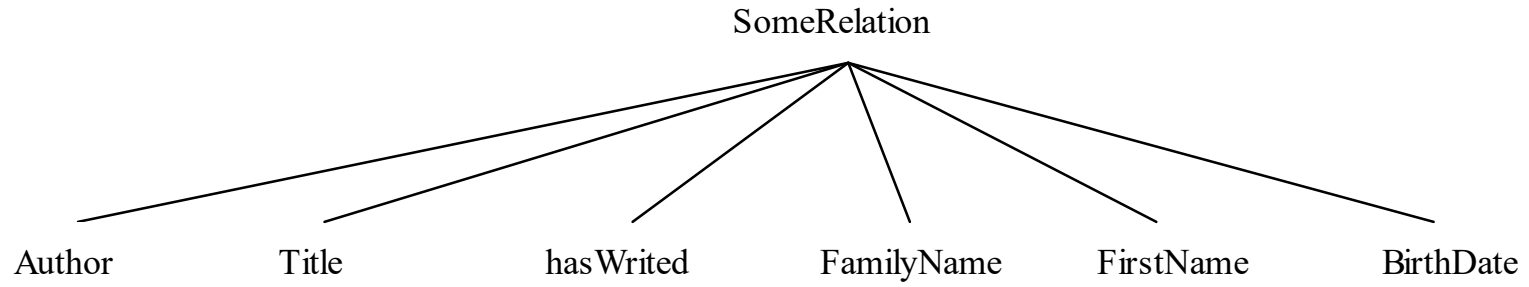
关系	领域	范围	父辈关系	自然语言定义
SomeRelation	Something			两个事物之间属于、连接、刻画等抽象
...	...	...	...	.....

### 中间层关系表

关系	领域	范围	父辈关系	自然语言定义
Author	Document	Person	SomeRelation	表示一个文档被一个人创造的关系
Title	Document	Literal (RDF)	SomeRelation	指定一个文档的文字
...	...	...	...	.....



本体概念的层次结构图



本体关系的层次结构图



## ❁ 阶段4：用**RDFS**语言描述上述各表

- ❁ rdfs和rdf是W3C定义的两个RDFS。
- ❁ `<?xml version="1.0" encoding="ISO-9999-9" ?>`
- ❁ `<rdf:RDF xmlns:rdfs=http://www.w3.org/2000/01/rdf-schema#  
xmlns:rdf="http://www.w3.org/1999/02/22-rdf-syntax-ns#" >`

## ❁ 阶段5：定义关系的代数属性，定义知识的推理规则

- ❁ `<rdf:Property rdf:ID="Author">`
- ❁ `< rdfs:inverse rdf:resource="#hasWrited" />`
- ❁ `</rdf:Property>`
- ❁ `<rdf:Property rdf:ID="hasWrited">`
- ❁ `<rdfs:inverse rdf:resource="#Author" />`
- ❁ `</rdf:Property>`

## ❁ 本体建模语言

- ❁ 语义网的核心概念之一就是本体，本体定义了组成主题领域词汇的基本术语和关系，以及用于组合术语和关系以定义词汇外延的规则。
- ❁ 用来描述本体的语言称为本体描述语言，它应该满足以下要求：
  - 良好定义的语法；
  - 良好定义的语义；
  - 有效的推理支持；
  - 充分的表达能力；
  - 便于表达。
- ❁ 本体描述语言
  - RDF和RDF-S、OIL、DAML、DAML+OIL、OWL、XML、KIF、SHOE、XOL、OCML、Ontolingua、CycL、Loom等

## 2.7 过程表示（Others）

- ❖ 过程式表示就是将有关某一问题领域的知识，连同如何使用这些知识的方法，均隐式地表达为一个求解问题的过程。
- ❖ 过程式表示不象陈述式表示那样具有固定的形式，如何描述知识完全取决于具体的问题。

## 2.8 小结 (Summary)

- ✿ 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。
- ✿ 对于同一问题可以有許多不同的表示方法。不过对于特定问题有的表示方法比较有效，其它表示方法可能不大适用，或者不是好的表示方法。
- ✿ 在表示和求解比较复杂的问题时，采用单一的知识表示方法是远远不够的。往往必须采用多种方法混合表示。
- ✿ 本节着重介绍知识表示方法，而知识推理将在下一章中加以研究。

方法	初始问题	算符	目标	结果
状态空间	状态	算符	目标状态	解答路径 (path)
问题归约	节点	弧线	节点	解答树 (tree)
谓词逻辑	合适公式	子句集 置换合一 消解反演	根节点	NIL
语义网络	节点	链	目标网络	语义网络

4种知识表示方法间的关系