人工智能技术
Artificial Intelligence

——人工智能: 经典智能+计算智能+机器学习
AI: Classical Intelligence + Computing Intelligence + Machine Learning

王鸿鹏、杜月、王润花、许丽

南开大学人工智能学院

第二部分 经典人工智能
Classical Artificial Intelligence

——第三章：知识表示
Chapter 3: Knowledge Representation

知识 Knowledge

- 数据、信息≠知识
- 知识层次

数据：一些无关联的事实(fact)

例：下雨了

信息：建立了事实间的联系后形成信息，提供对what, who, when, where等问题的回答

例：温度下降了15度后就下雨了

知识：当能建立模式之间的联系后便涌现了知识，提供对how的回答

例：如果湿度非常大并且温度下降显著，大气无法保留湿气便下雨了

智慧：能描述模式之间关系的规律，提供对why的回答

例：理解下雨、蒸发、空气状况、温度等级及它们的变化之间的相互作用

知识 Knowledge

- “知识”是我们熟悉的名词。但究竟什么是知识呢？我们认为，知识就是人们对客观事物（包括自然的和人造的）及其规律的认识，知识还包括人们利用客观规律解决实际问题的方法和策略等。
- 就内容而言，知识可分为（客观）原理性知识和（主观）方法性知识两大类。
- 就形式而言，知识可分为显式的和隐式的。
- 知识的属性包括：
  - 真伪性
  - 相对性
  - 不完全性
  - 不确定性
  - 可表示性
  - 可存储性、可传递性和可处理性
  - 相容性

知识表示 Knowledge Representation

- 知识表示是问题求解的基础
  - 问题求解是人工智能的核心问题之一
  - 问题求解的目的
    - 机器自动找出某问题的正确解决策略
    - 更进一步，能够举一反三，具有解决同类问题的能力
- 智能系统中
  - 知识是对世界的描述
    - 决定系统的能力
  - 表示是知识的编码方式
    - 决定系统的性能
- 不同类型的知识需要不同的表示方式
- 不同的表示方法需要不同的求解技术

主要表示方法 Knowledge Representation Methods

- GOFAI (Good Old-Fashioned Artificial Intelligence) philosophy:
  - Problem Solving = Knowledge Representation
  - + Search
- State Space **(**状态空间**)** Representation
- Problem Reduction (问题归约) Representation
- Predicate Logic（谓词逻辑）
- Semantic Network （语义网络） Representation
- Frame, Script, Procedure（框架，剧本，过程）

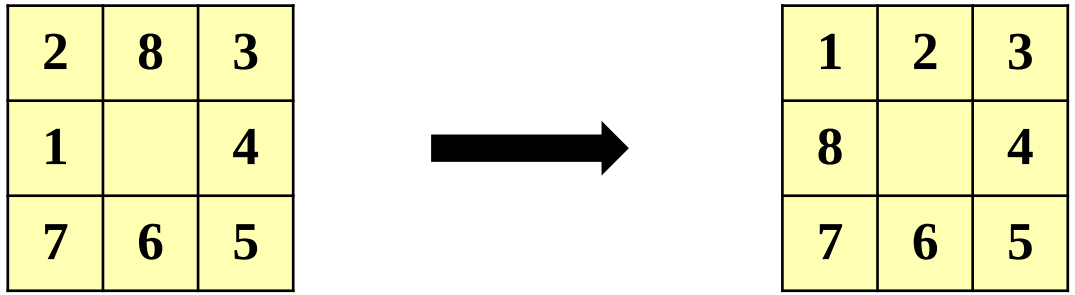
状态空间法

State Space

问题示例

- 状态空间法示例1：八数码难题（后续详解）

在3×3的棋盘，摆有八个棋子，每个棋子上标有1至8的某一数字。棋盘上还有一个空格，与空格相邻的棋子可以移到空格中。

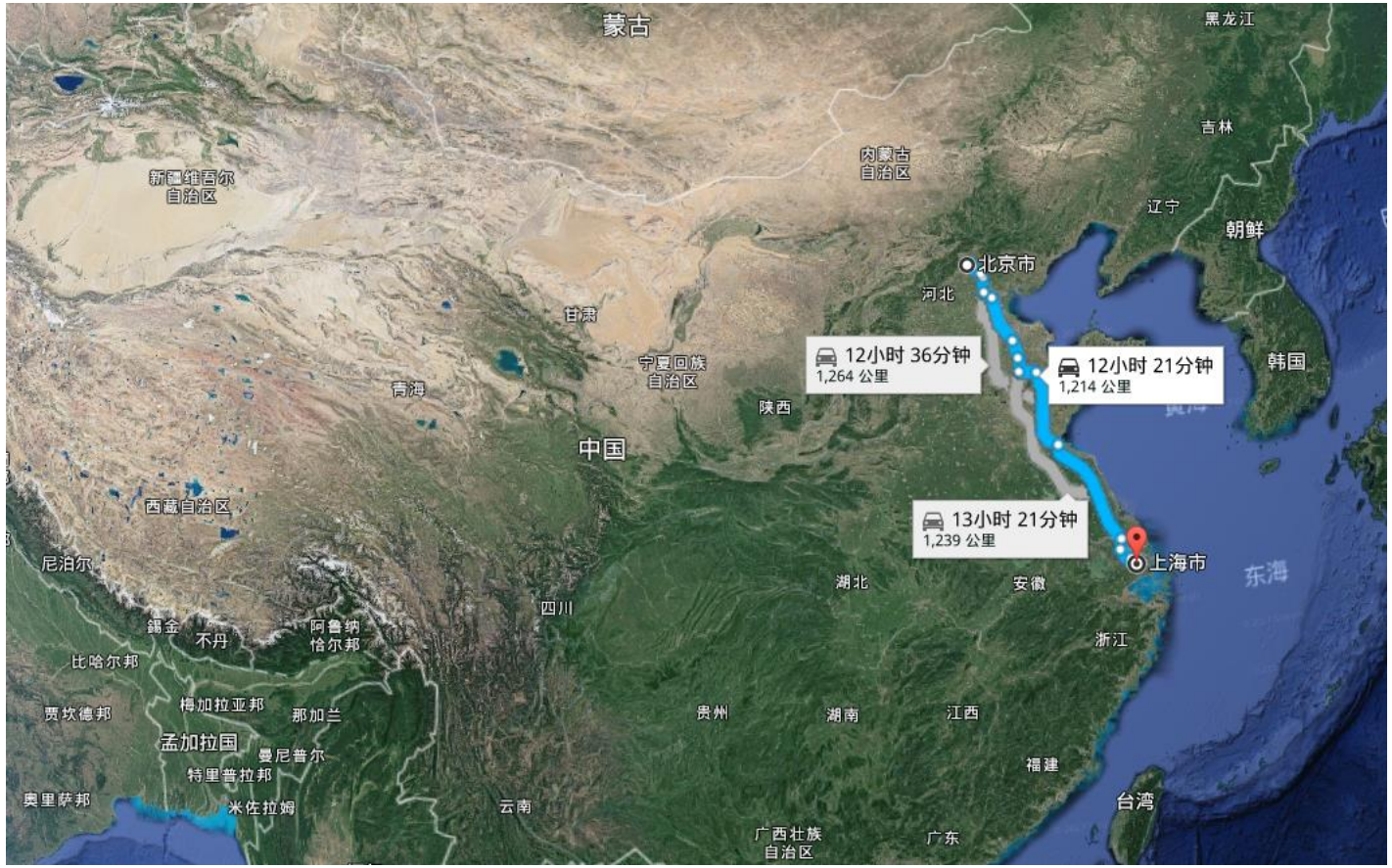


初始状态　　目标状态

- 如何将棋盘从某一初始状态变成最后的目标状态?

问题示例

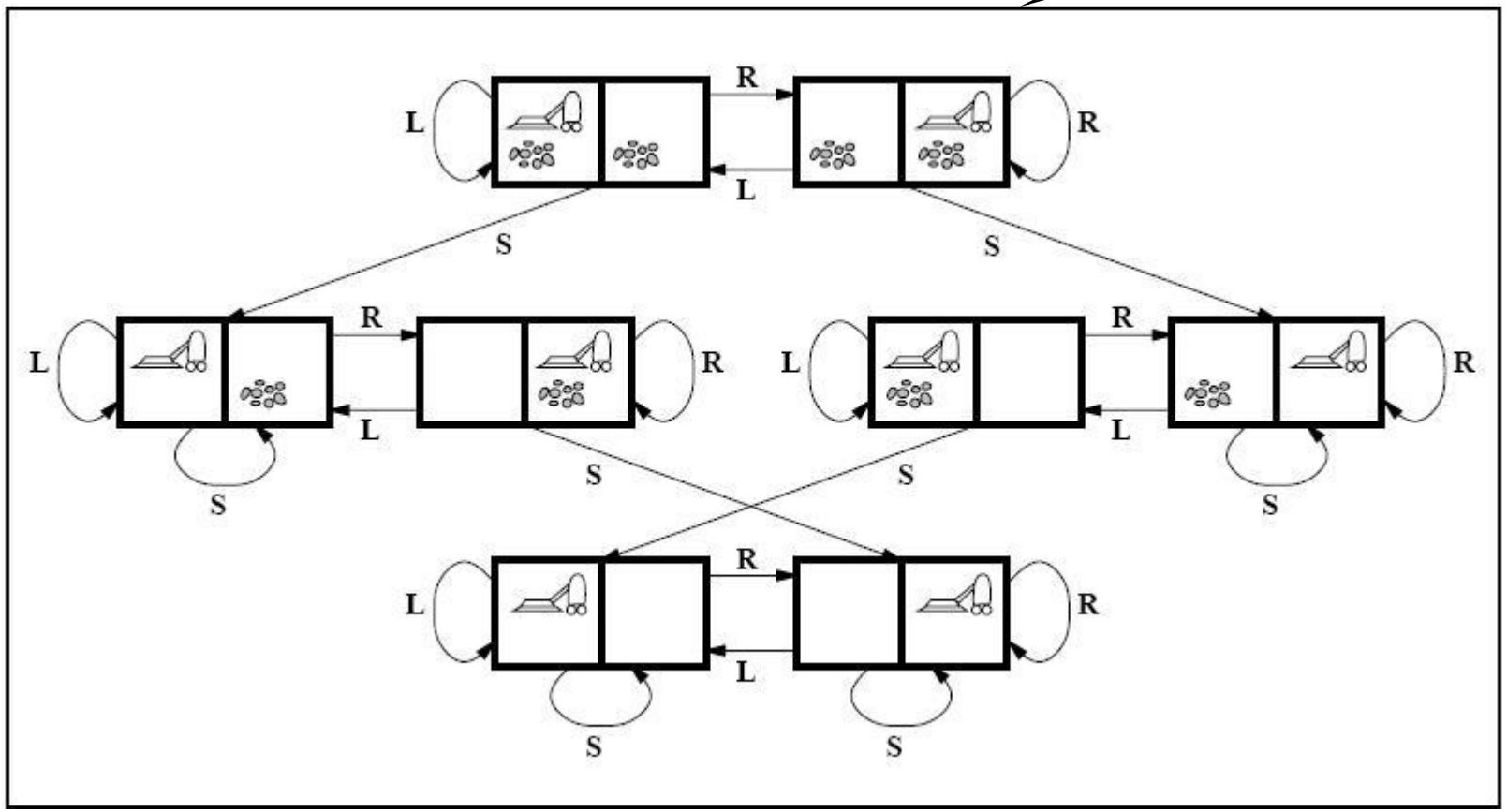
- 怎样找到两点之间的最短路径呢?



状态转换

- 状态之间可以互相转换

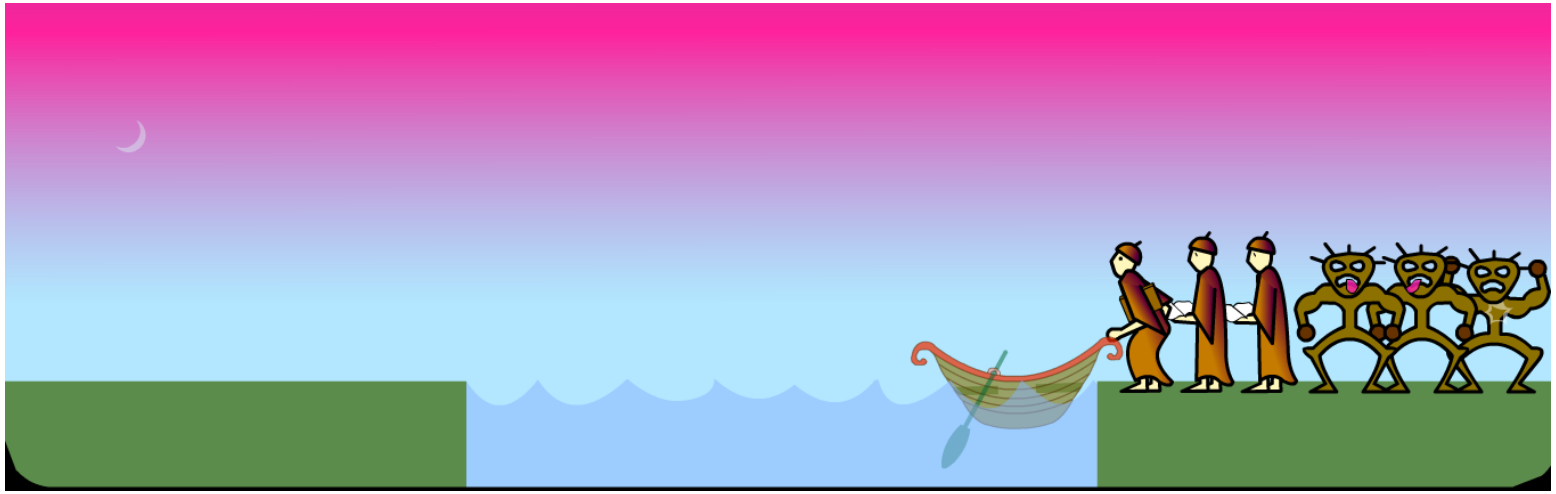
状态空间图



问题示例

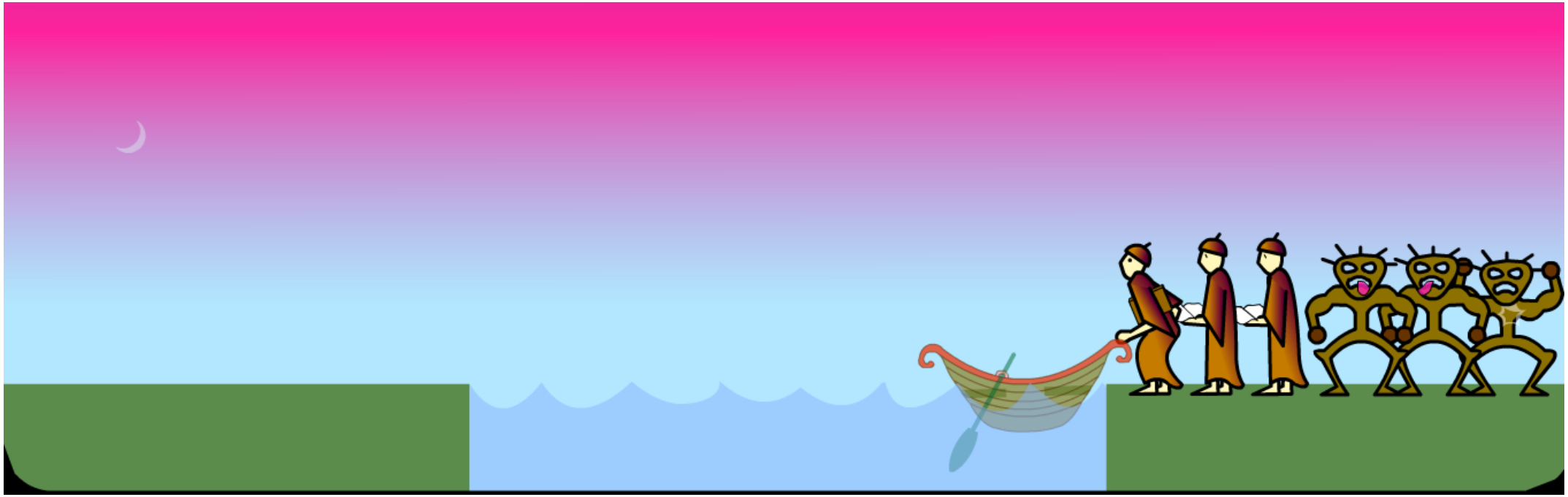
状态空间法示例2：传教士过河问题

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，在河的一方或者船上，如果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法呢?



问题示例

- 传教士过河问题



状态及其表示

- 状态(State)：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

(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

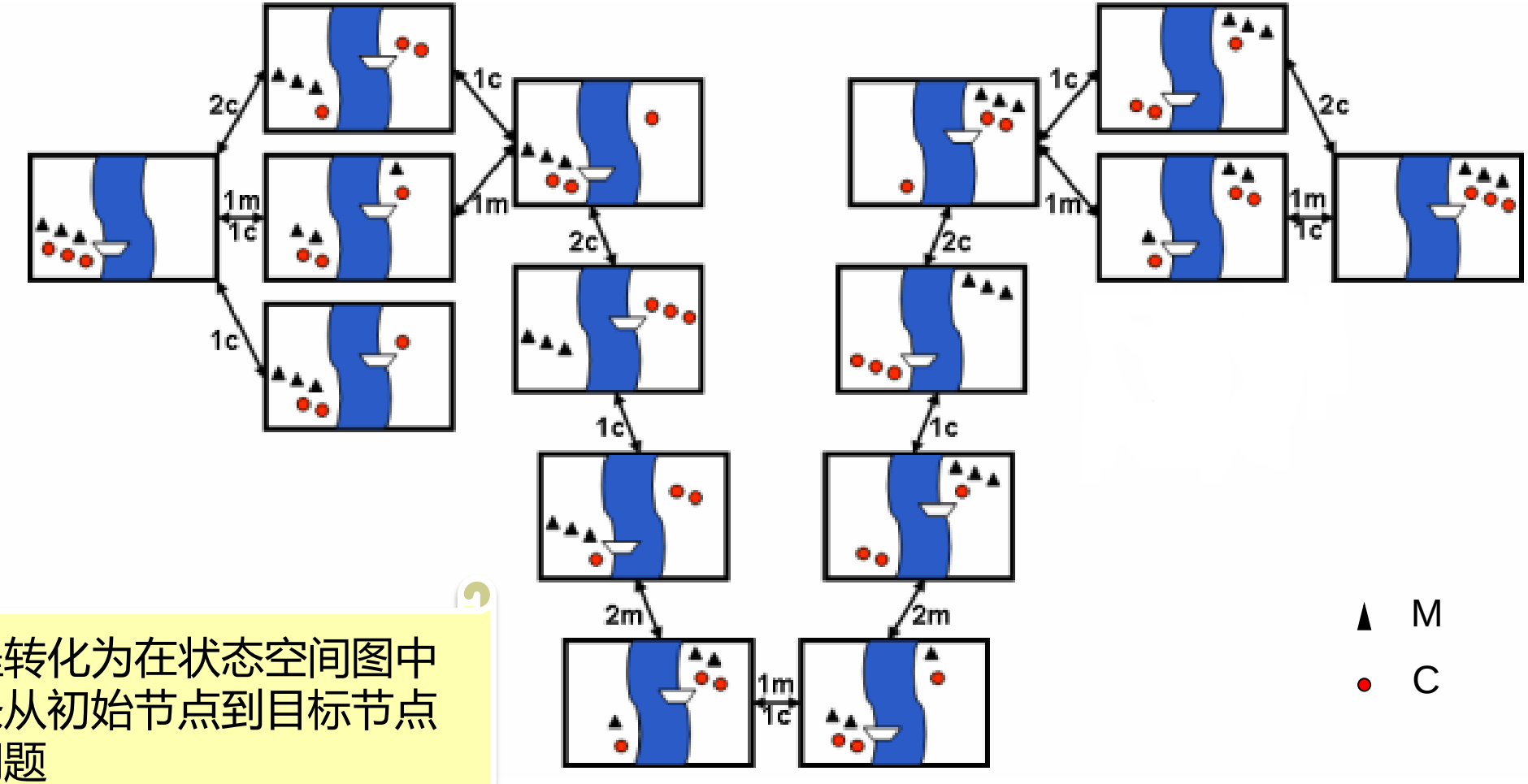
哪些操作能导致状态变化?

状态的转换

- 算子（Operator, 算符，操作符）——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)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



求解过程转化为在状态空间图中搜索一条从初始节点到目标节点的路径问题

状态空间表示

状态空间法：是一种基于解答空间的问题表示和求解方法，它是以状态和算符为基础的。
在利用状态空间图表示时，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实验序列，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”，因而只适用于比较简单的问题。

状态问题描述

- State Space Method : A method for problem representation and problem-solving based on the solution space.
- It is based on the State and Operator.
  - State(状态) : some variables, describing the difference among different things or events of some class

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

  - Operator(算符): A means that is used to transform a problem from one state to another.

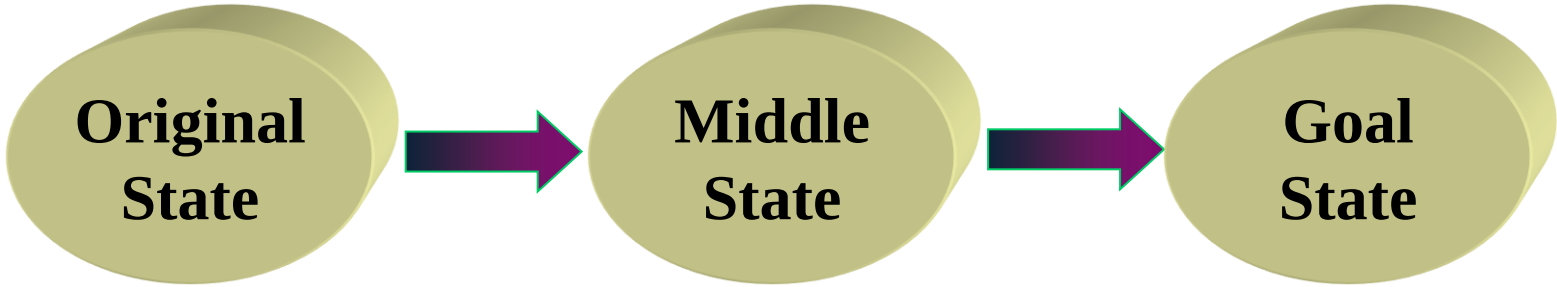
状态空间表示

- State Space(状态空间): A graph that represents all possible states and their relationships with the problem.

  {*S*,*F*,*G*}- *S*: set of all possible initial states of problem

  F: set of operators

  G: set of goal states



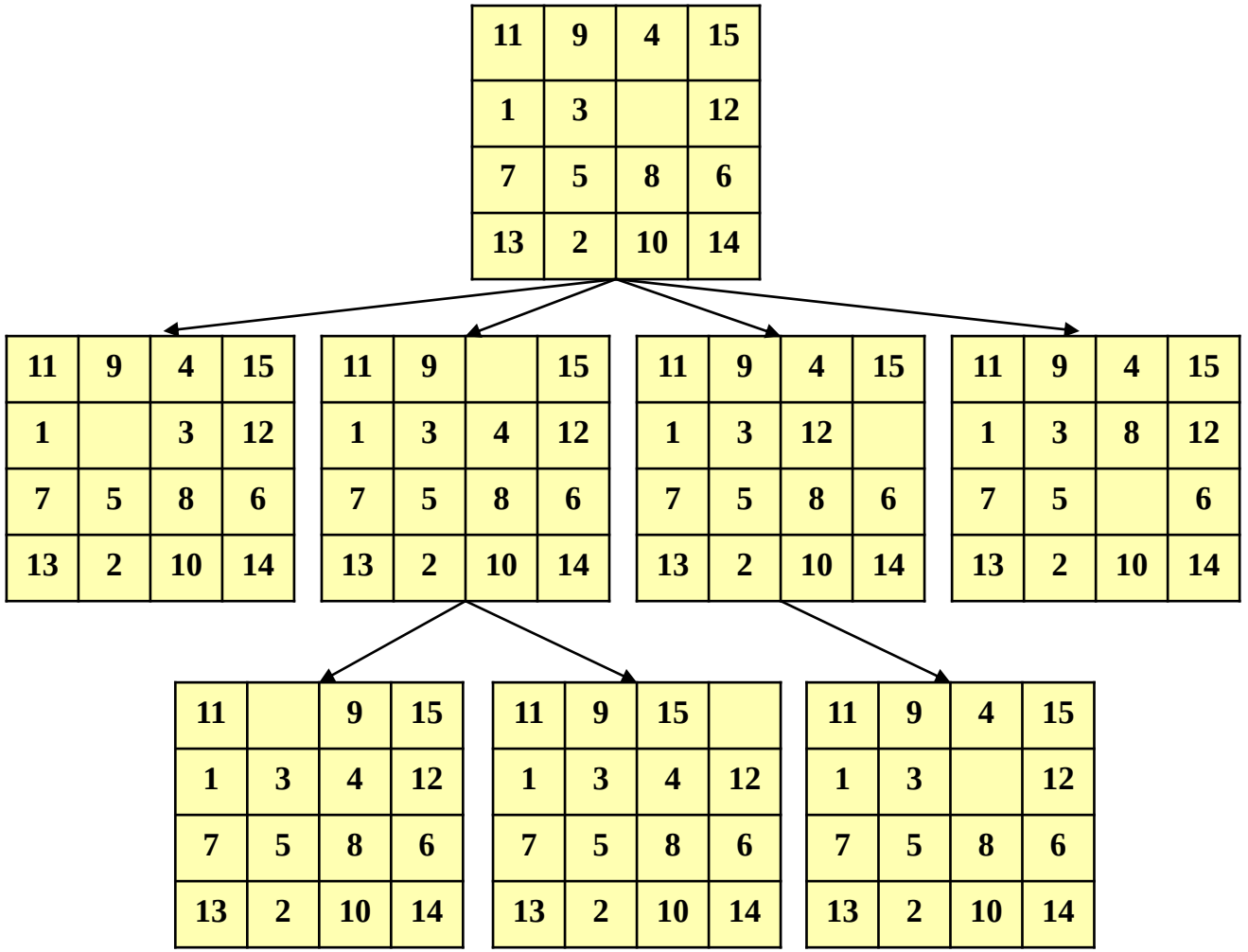
状态空间法示例1：N-数码问题

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

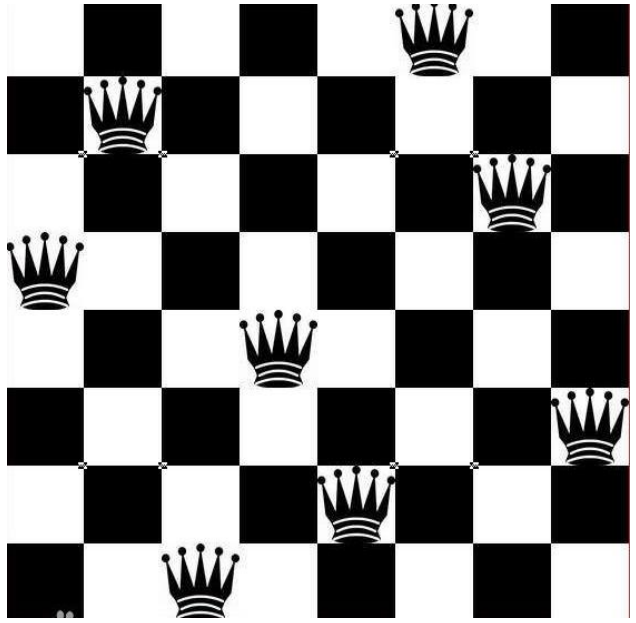
1	2	3	4
5	6	7	8
9	10	11	12
13	14	15	

N-数码游戏（N-数码问题）描述为：在m×m组成的N+1宫格棋盘上，摆有N个牌，每一个牌都刻有1-N个数码中的某一个数码。棋盘中留有一个空格，允许其周围的某一个将牌向空格移动，这样通过移动将牌就可以不断改变将牌的布局。这种游戏求解的问题是：给定一种初始的将牌布局或结构（称初始状态）和一个目标的布局（称目标状态），问如何移动将牌，实现从初始状态到目标状态的转变。

状态空间法示例1：N-数码问题

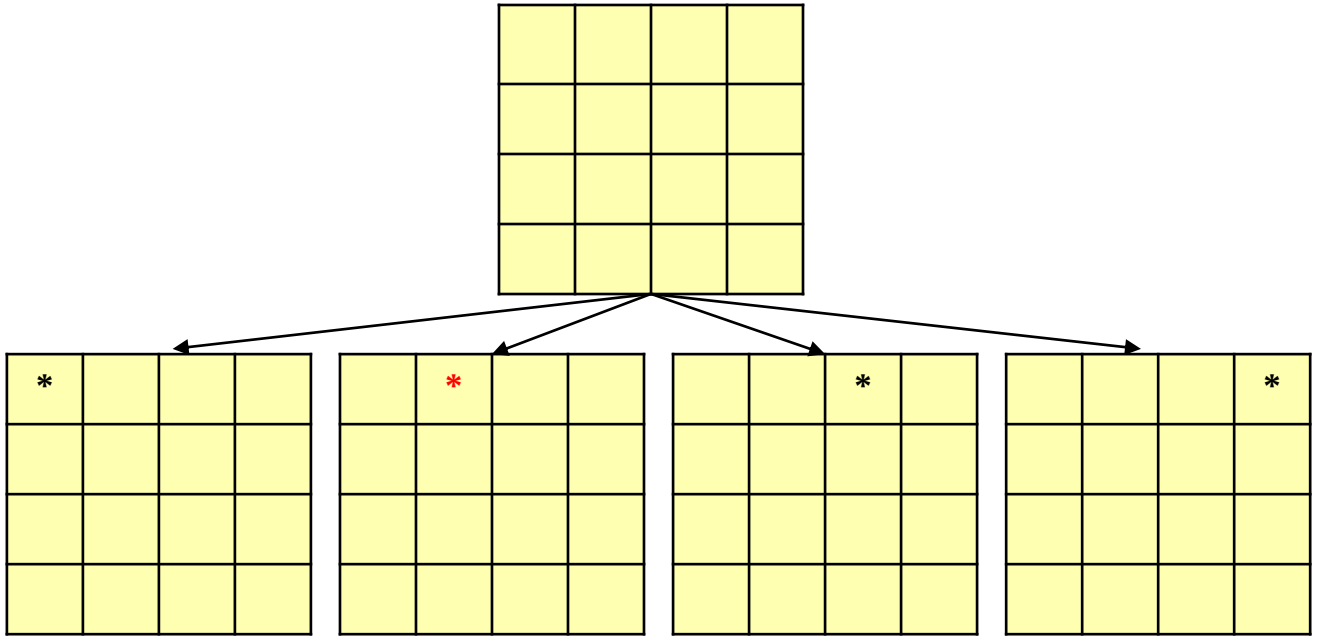


状态空间法示例2：n-皇后问题



八皇后问题是一个以国际象棋为背景的问题：如何能够在 8×8 的国际象棋棋盘上放置八个皇后，使得任何一个皇后都无法直接吃掉其他的皇后？为了达到此目的，任两个皇后都不能处于同一条横行、纵行或斜线上。八皇后问题可以推广为更一般的n皇后摆放问题：这时棋盘的大小变为n1×n1，而皇后个数也变成n2。而且仅当 n2 = 1 或 n1 ≥ 3 时问题有解。

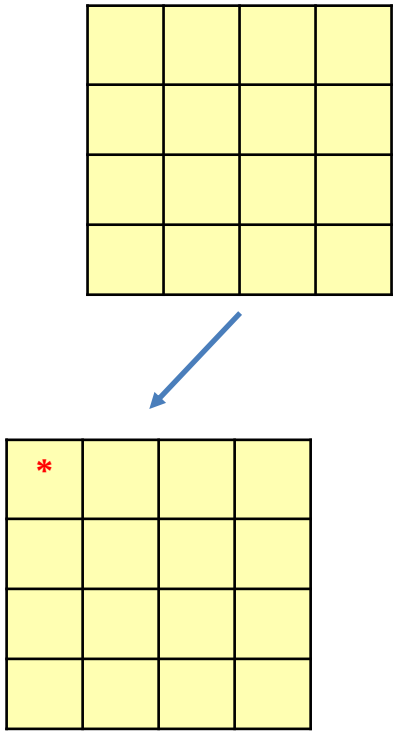
状态空间法示例2：n-皇后问题

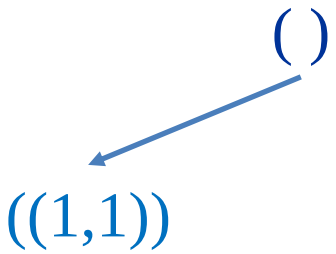


状态空间图：把初始状态可达到的各个状态所组成的空间设想成一幅由各种状态对应的节点组成的图

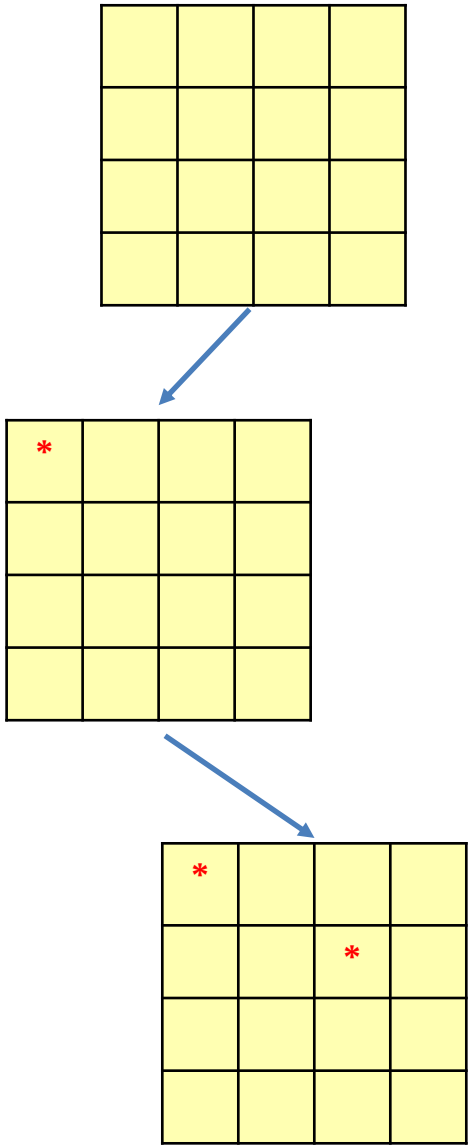
状态空间法示例2：n-皇后问题

Step 1

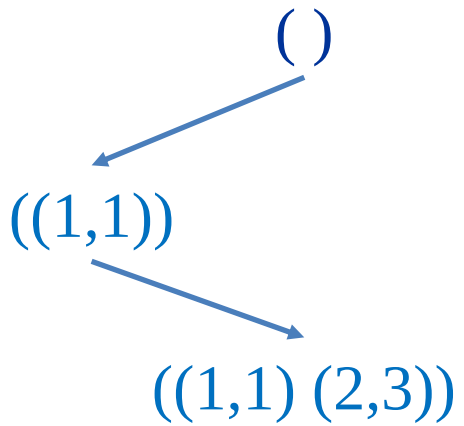




状态空间法示例2：n-皇后问题

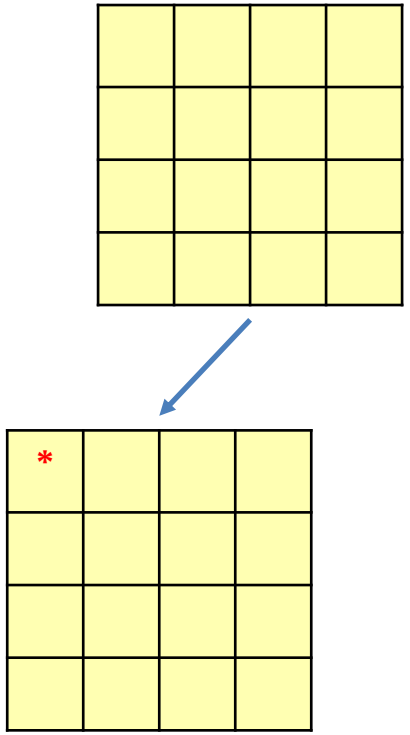


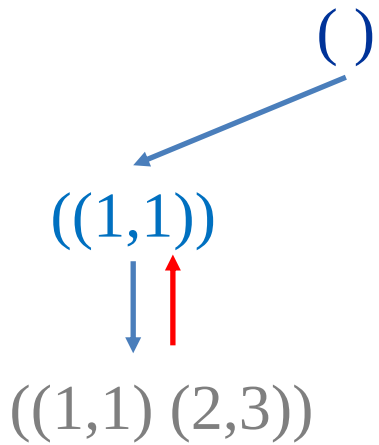
Step 2



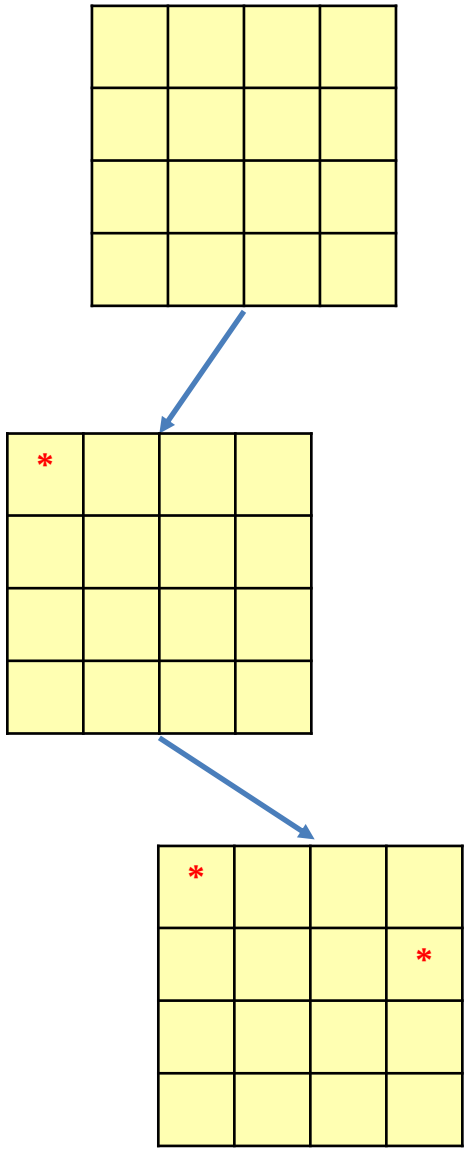
状态空间法示例2：n-皇后问题

Step 3

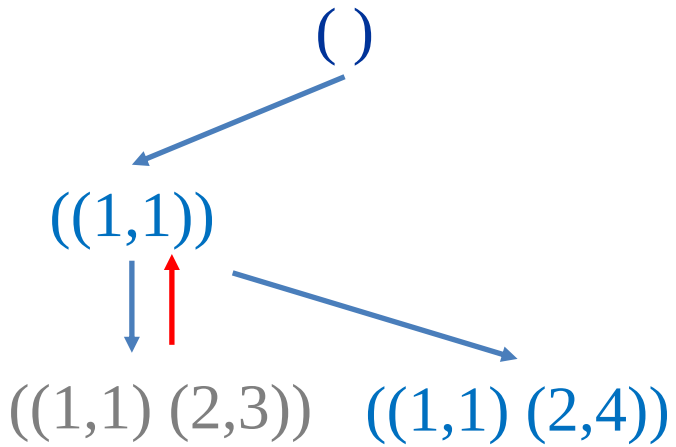




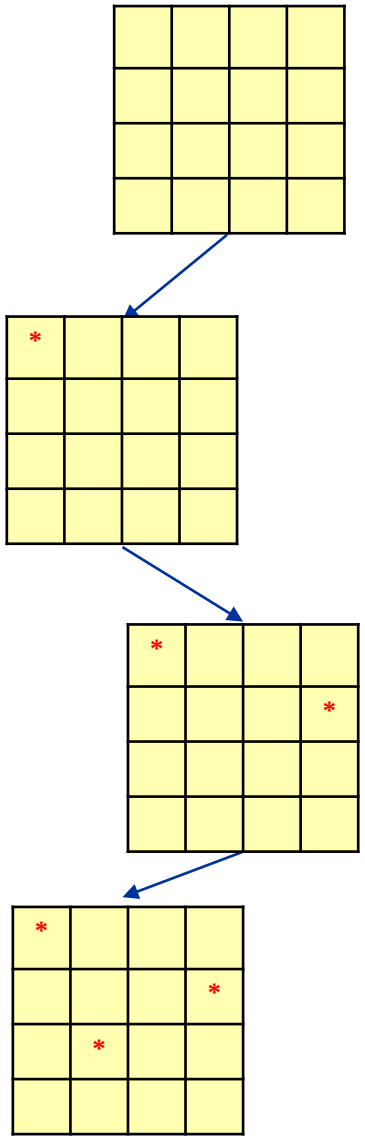
状态空间法示例2：n-皇后问题



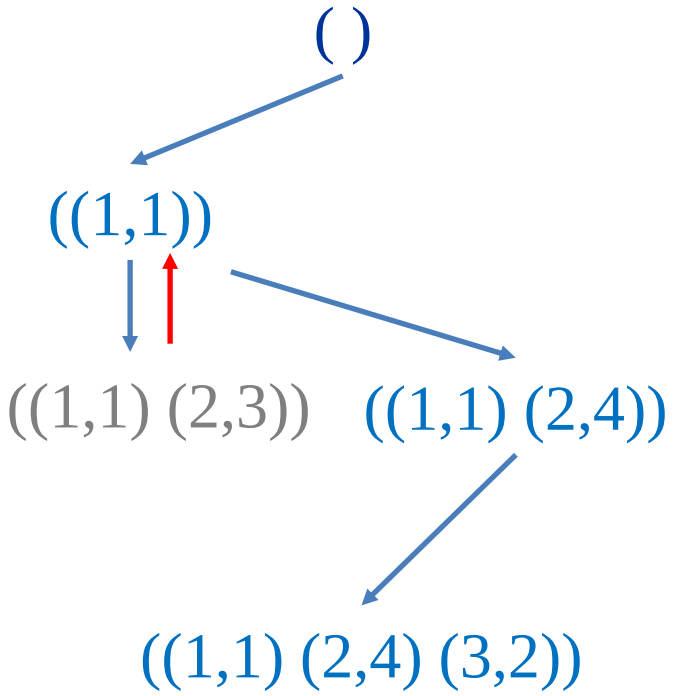
Step 4



状态空间法示例2：n-皇后问题

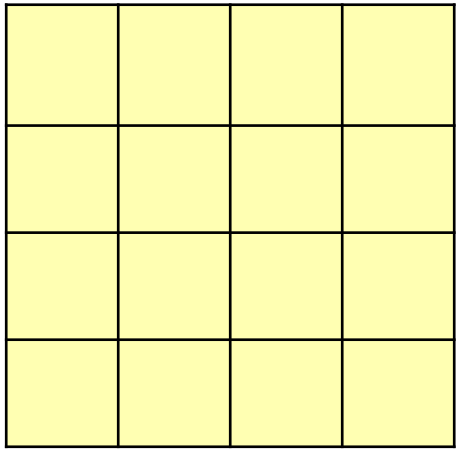


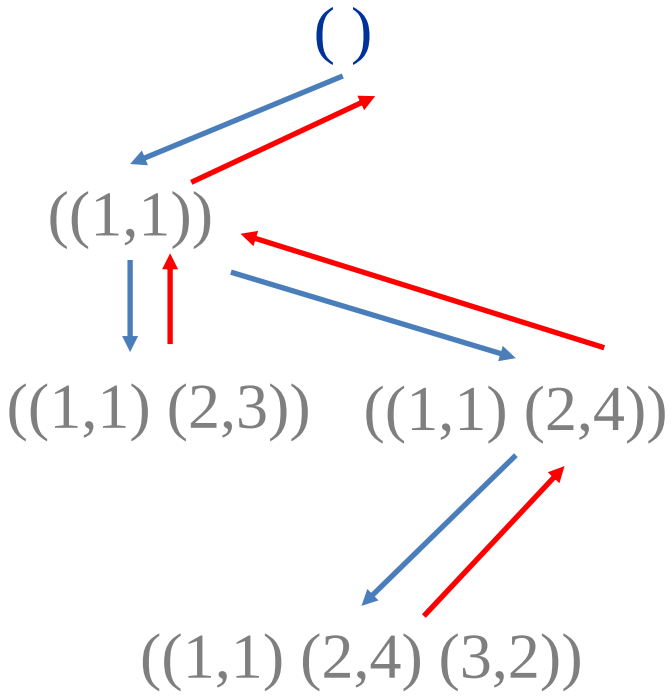
Step 5



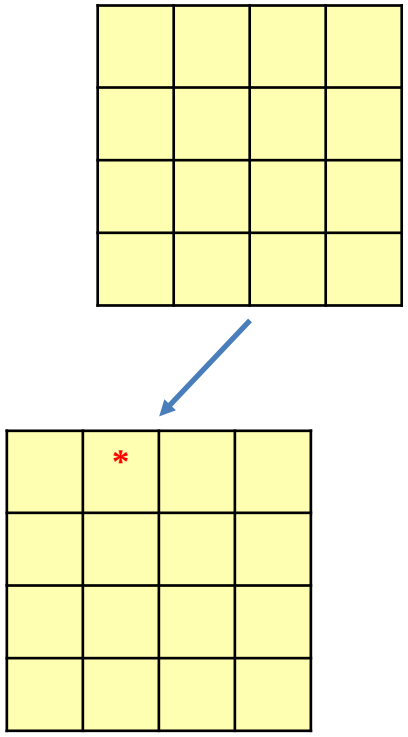
状态空间法示例2：n-皇后问题

Step 6

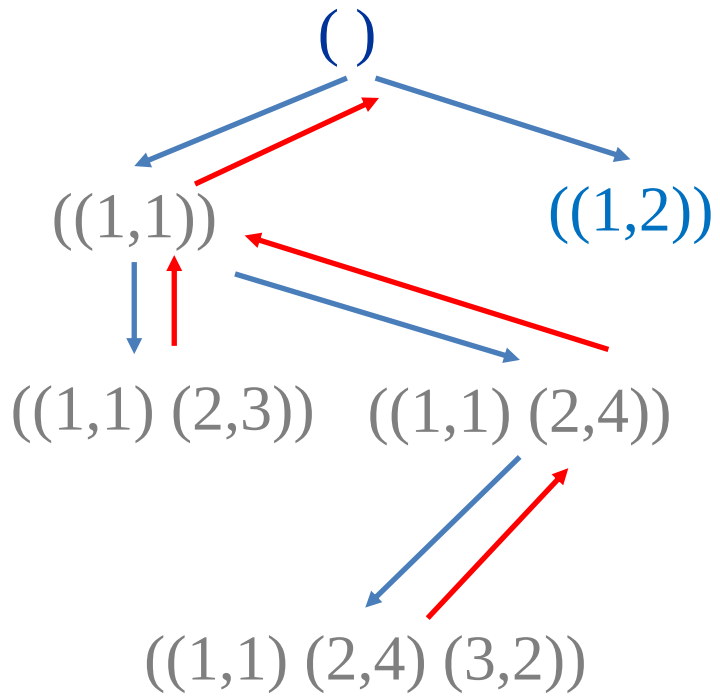




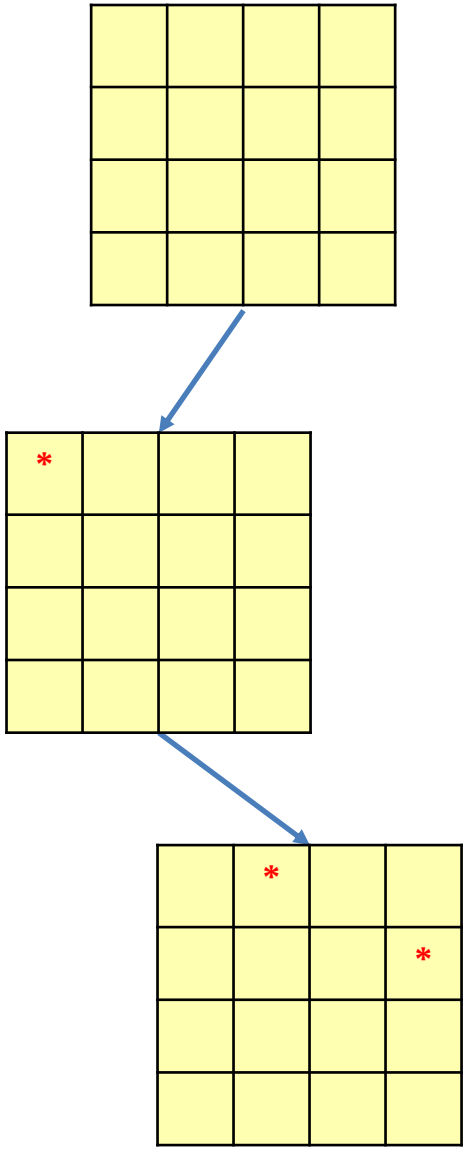
状态空间法示例2：n-皇后问题



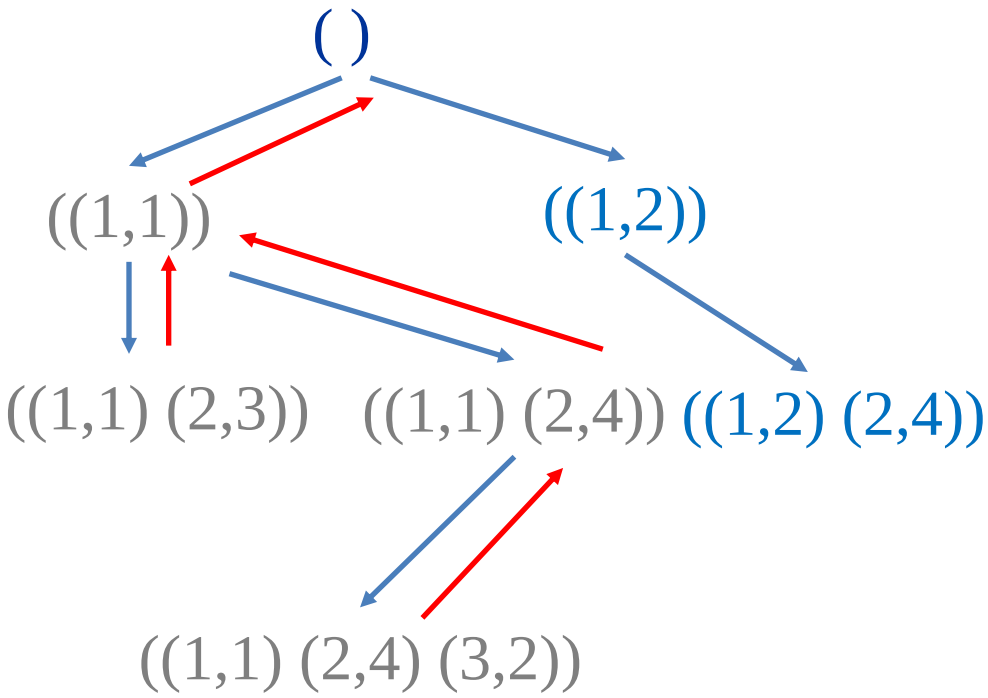
Step 7



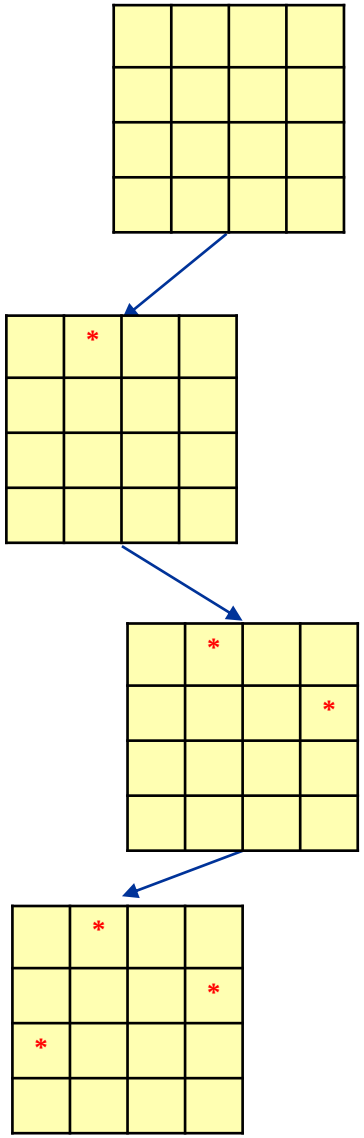
状态空间法示例2：n-皇后问题



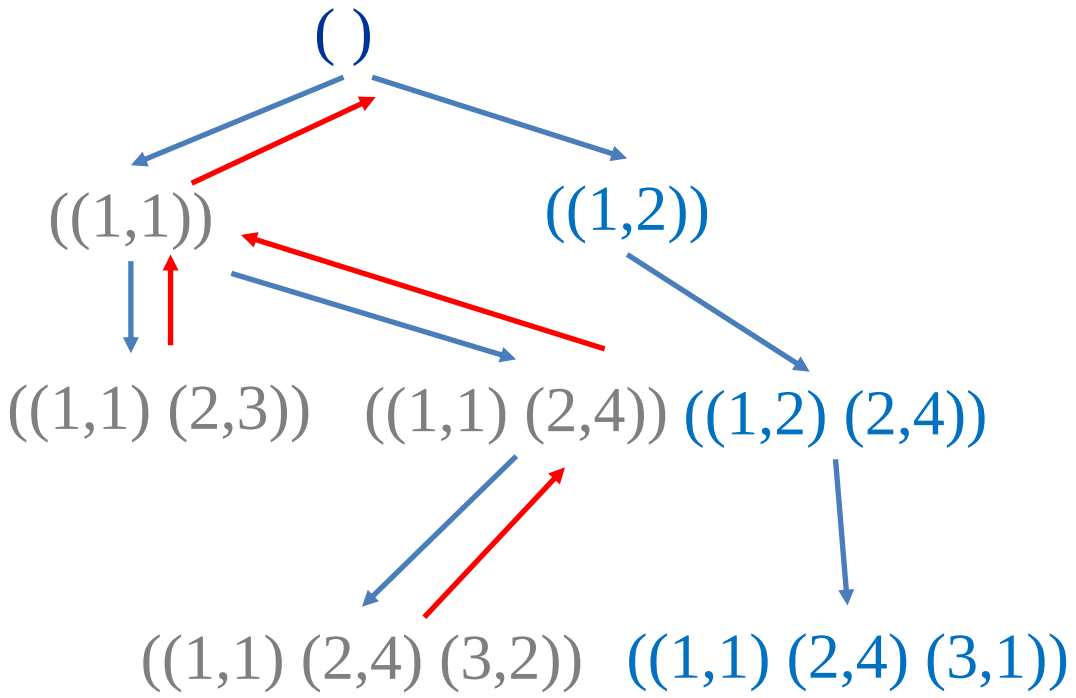
Step 8



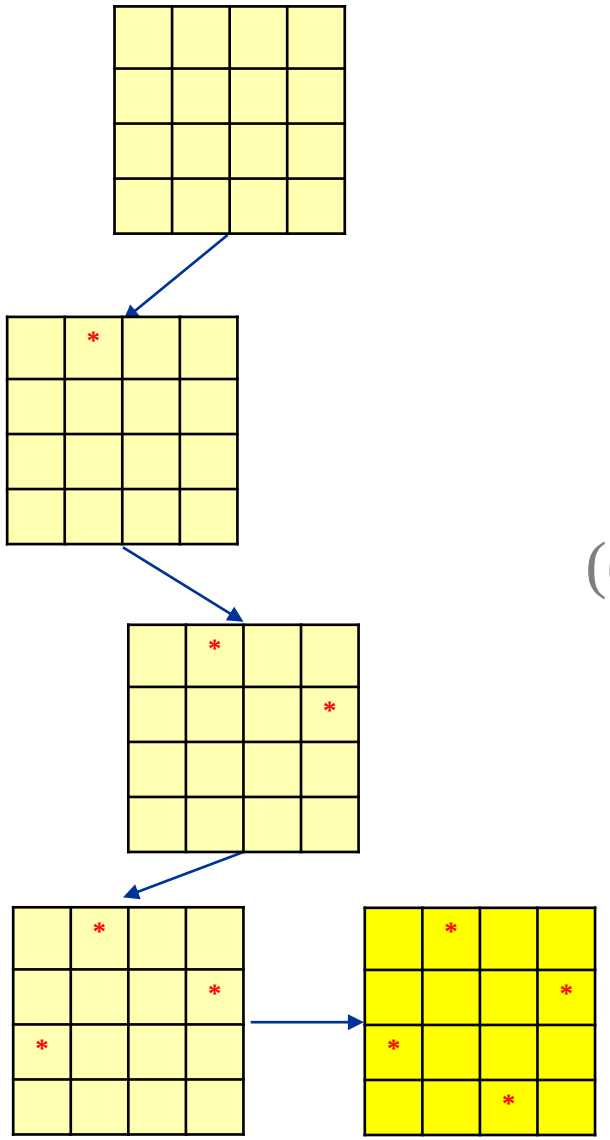
状态空间法示例2：n-皇后问题

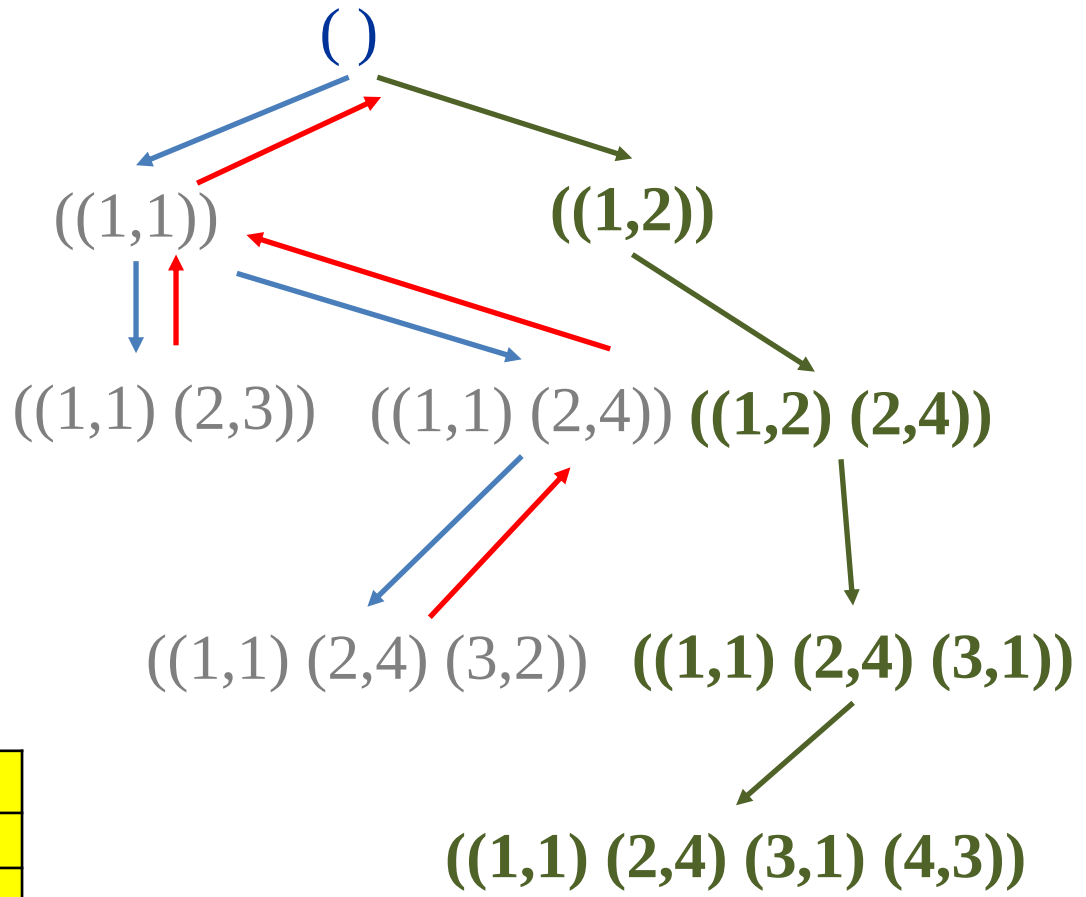


Step 9



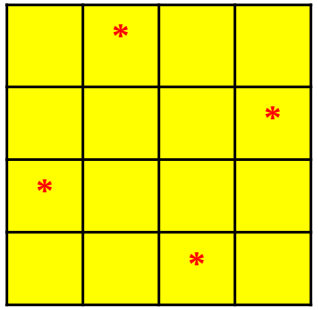
状态空间法示例2：n-皇后问题

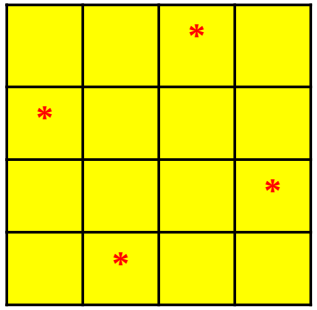




状态空间法示例2：n-皇后问题

- n-皇后问题状态空间法解的数量





4-皇后问题：2个解

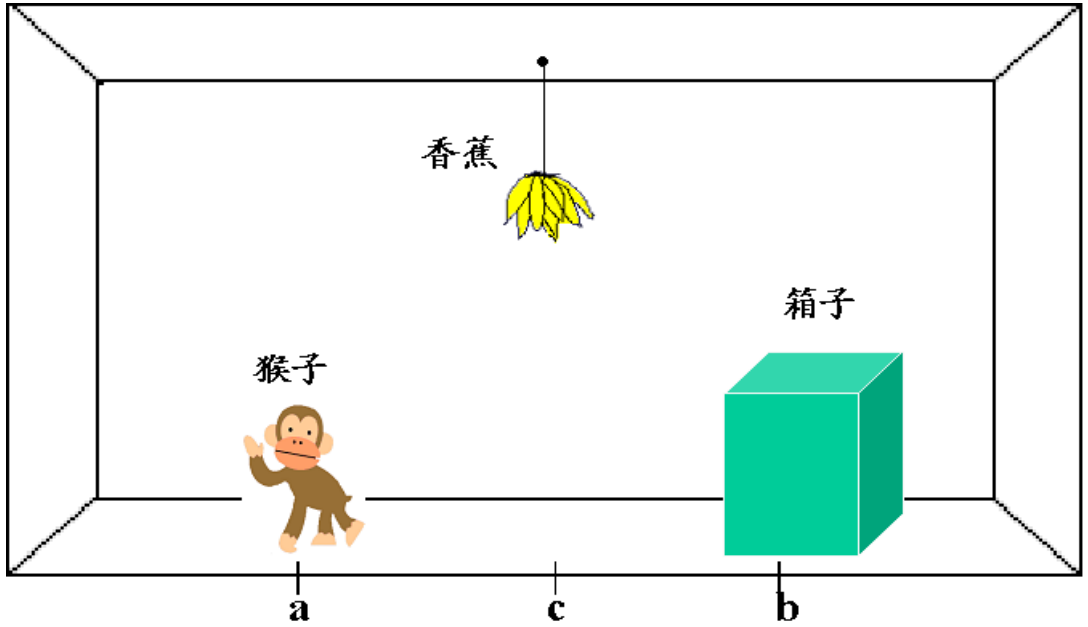
N-皇后	4	5	6	7	8	9
解数量	2	10	4	40	92	352
N-皇后	10	16		25		
解数量	724	14772512		2207893435808352		

*由于状态空间法需要扩展过多的节点，容易出现“组合爆炸”

状态及其表示

状态空间法示例4：猴子与香蕉问题

在一个房间内有一只猴子(可把这只猴子看做一个机器人)、一个箱子和一束香蕉。香蕉挂在天花板下方，但猴子的高度不足以碰到它。那么这只猴子怎样才能摘到香蕉呢?



状态及其表示

状态空间法示例4：猴子与香蕉问题

解：用一个四元组（W, x, Y, z）来表示问题的状态

- W：猴子的水平位置
- x：当猴子爬到箱子顶上取1，否则取0
- Y：箱子的水平位置
- z：当猴子摘到香蕉时取1，否则取0

初始状态是（a, 0, b, 0）

目标状态是（c, 1, c, 1）

状态及其表示

状态空间法示例4：猴子与香蕉问题

操作符：

- 猴子从当前位置W走到水平位置U：goto(U)：

  （W, 0 , Y, z）→(U, 0 , Y, z)

  注：猴子必须不在箱子上

- 猴子将箱子从W位置推到水平位置V：pushbox(V)：

  （W, 0, W, z） → (V, 0, V, z)

  注：猴子与箱子必须在同一位置，猴子不在箱子上

- 猴子爬到箱子上：climbbox：

  （W, 0, W, z） → (W, 1, W, z)

- 猴子摘到香蕉：grasp：

  （c, 1, c, 0） → (c, 1, c, 1)

产生式规则的先决条件

状态及其表示

- 状态空间法示例4：猴子与香蕉问题

初始状态

(a, 0, b, 0)

↓ goto(b)

(b, 0, b, 0)

pushbox　　climbbox

(c, 0, c, 0)　　(b, 1, b, 0)

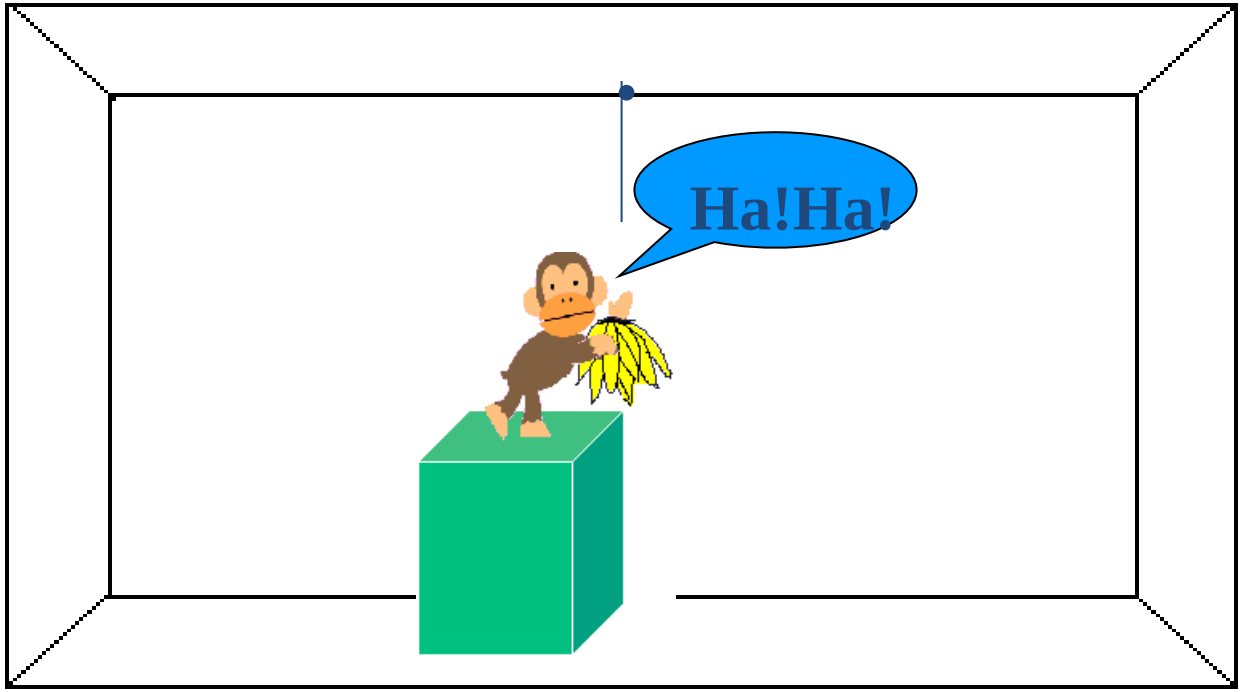
↓ climbbox

(c, 1, c, 0)

↓ grasp

(c, 1, c, 1)

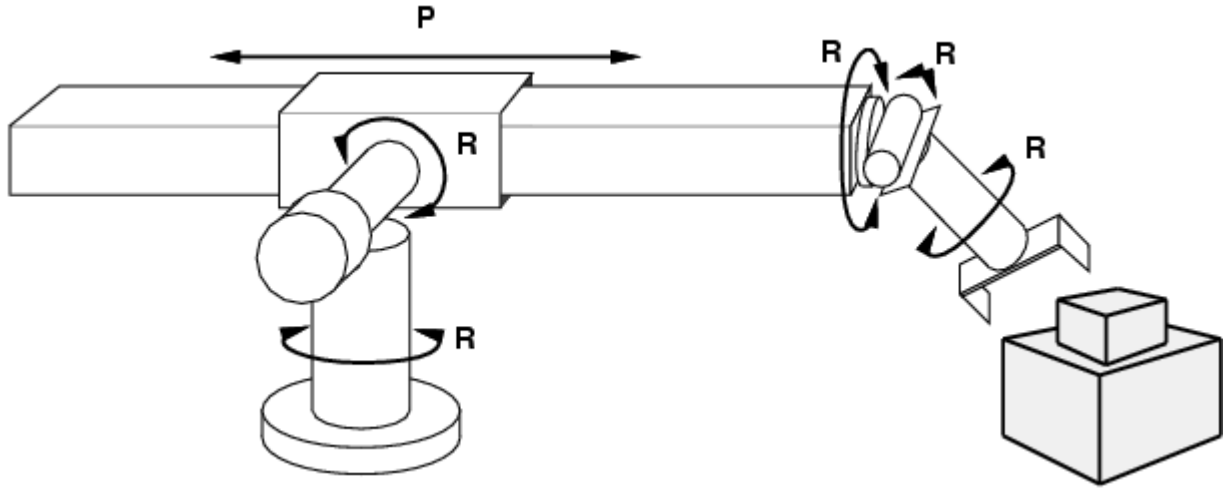
目标状态



解序列为：{Goto(b), Pushbox(c), Climbbox, Grasp}

状态及其表示

- 状态空间法示例5：Robotic Assembly



- states?: real-valued coordinates of robot joint angles parts of the object to be assembled
- actions?: continuous motions of robot joints
- goal test?: complete assembly
- path cost?: time to execute

状态及其表示

- 状态空间法示例6：Route Planning

  - Initial state
    - City the journey starts in Changsha
  - Operators
    - Driving from city to city
  - Goal state
    - Destination city is Beijing

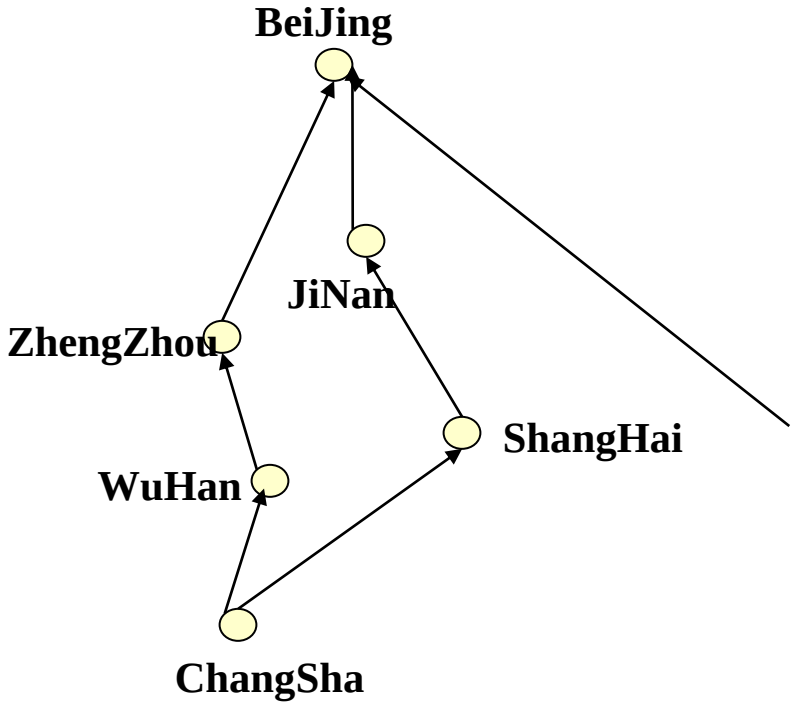


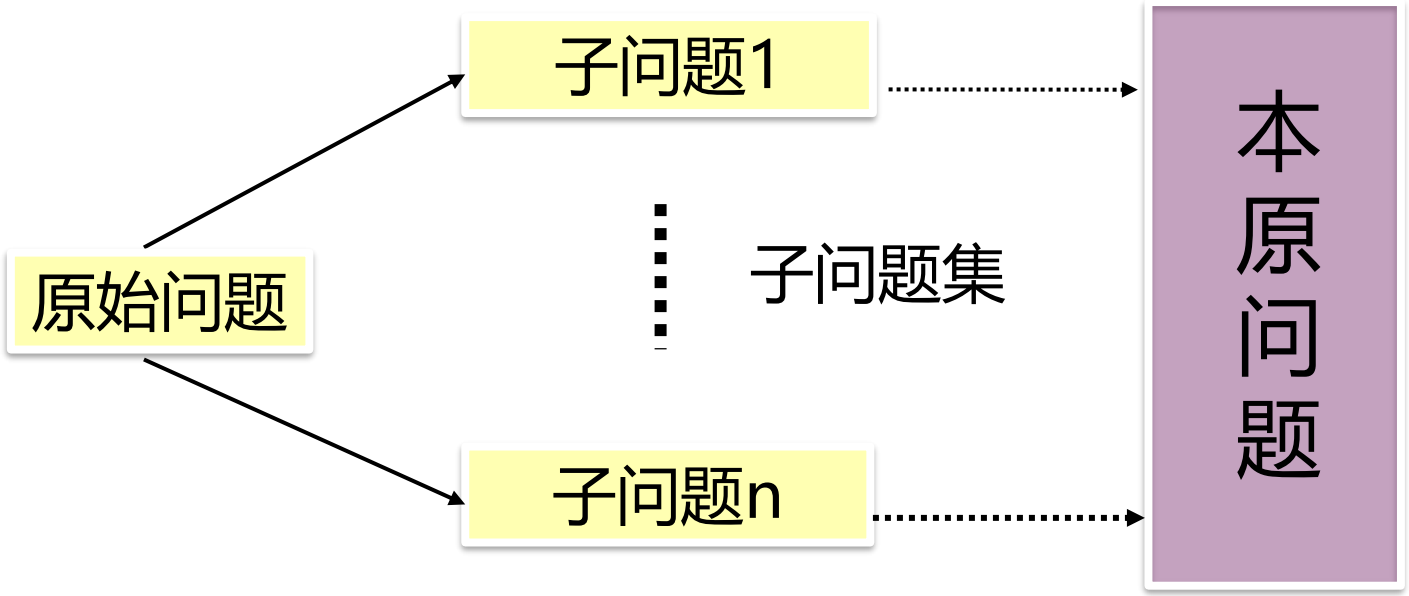
Fig. Route Planning

问题归约法

Problem Reduction

问题归约

- 思路
  - 给定一个问题描述，将原始问题通过一系列转化操作转化为一些子问题的集合
  - 最终把初始问题归约为一个平凡的本原问题（可直接求解的问题）集合
  - 通过求解这些问题来求解原始问题



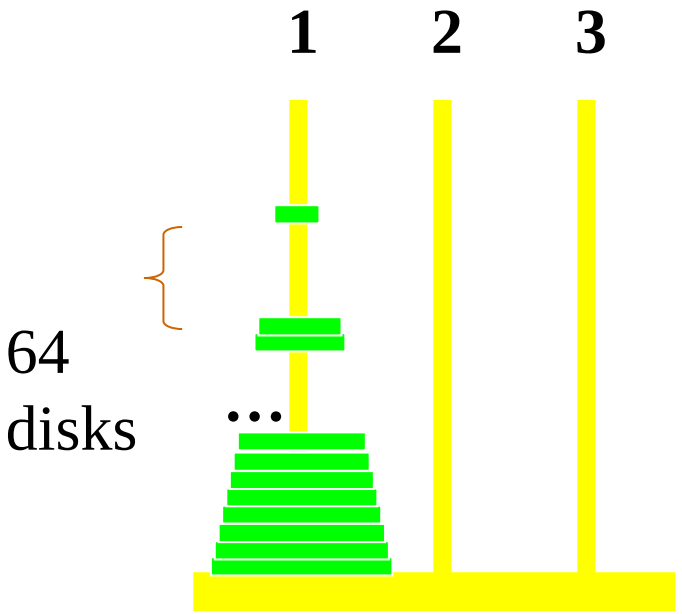
问题归约

- 问题归约表示可用三元组(G,O,P)描述
  - G：一个初始问题描述
  - O：一套把问题通过AND或OR分解变换为子问题的操作符
  - P：一套本原问题描述
- 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(可直接求解的问题)集合
- 问题规约法是更通用、更一般化的状态空间表示方法

问题归约

- **Three pegs and 64 disks**

- Constrain Conditions(约束条件)
  - Only one disk can be moved at a time;
  - Only the top disk on a peg can be moved;
  - Larger disk may never be placed on top of a smaller one



Moving times:

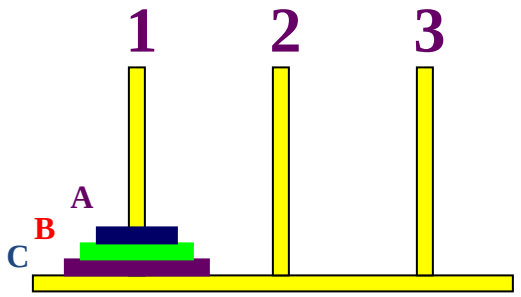
64 disks $2^{64}-1 \approx 2^{64} = 10^{19.27}$

If one person move 1 disk in one second, then to finish this problem needs more than 3000 billion years.(30000多亿年)

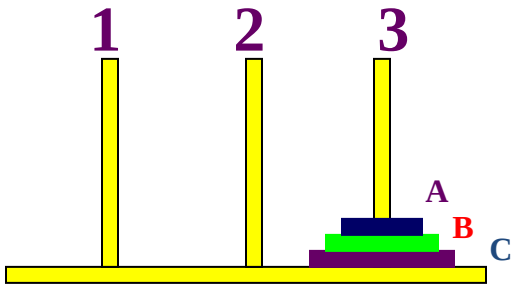
问题归约

问题归约法示例1：汉诺塔问题

Three disk Hanoi-Tower: There are three pegs-1,2 and 3 and three disks of different sizes A, B and C. The disks have holes in their centers so that they can be stacked on the pegs.



(a) initial configuration



(b) Goal configuration

问题归约

- 问题归约法示例1：汉诺塔问题（Hanoi-Tower Problem）
  - 问题表示
    - 与状态空间法不同，不能仅表示“状态”，需要表示目前的“问题”
    - 假设用向量（$D_n, D_{n-1}, \ldots, D_1$）表示从大到小的圆盘所在的柱子号，则
      - 初始状态：(1, 1, …, 1)
      - 目标状态：(3, 3, …, 3)
    - 初始问题则为要将初始状态转变为目标状态
      - $(1, 1, \ldots, 1) \Rightarrow (3, 3, \ldots, 3)$

问题归约

- 问题归约法示例1：汉诺塔问题（Hanoi-Tower Problem）

- Reduce the original problem to a set of simpler problems

  移动步骤和约束条件

  - Move disk C to peg 3 in order to move all of the disks to peg 3, and peg 3 must be empty .
  - Can not move disk C anywhere until disks A and B are moved first, and disk A and B had better not be moved to peg 3 since then we would not be able to move disk C there.
  - Then the key step of moving disk C from peg 1 to peg 3 can be completed, and go on to solve the rest of problem.

问题归约

- 问题归约法示例1：汉诺塔问题（Hanoi-Tower Problem）

  - The two-disk puzzle of moving disks A and B to peg 2 see Fig (a)

（111） (a) （122）

Fig. Reduction for the Hanoi Tower problem

问题归约

- 问题归约法示例1：汉诺塔问题（Hanoi-Tower Problem）

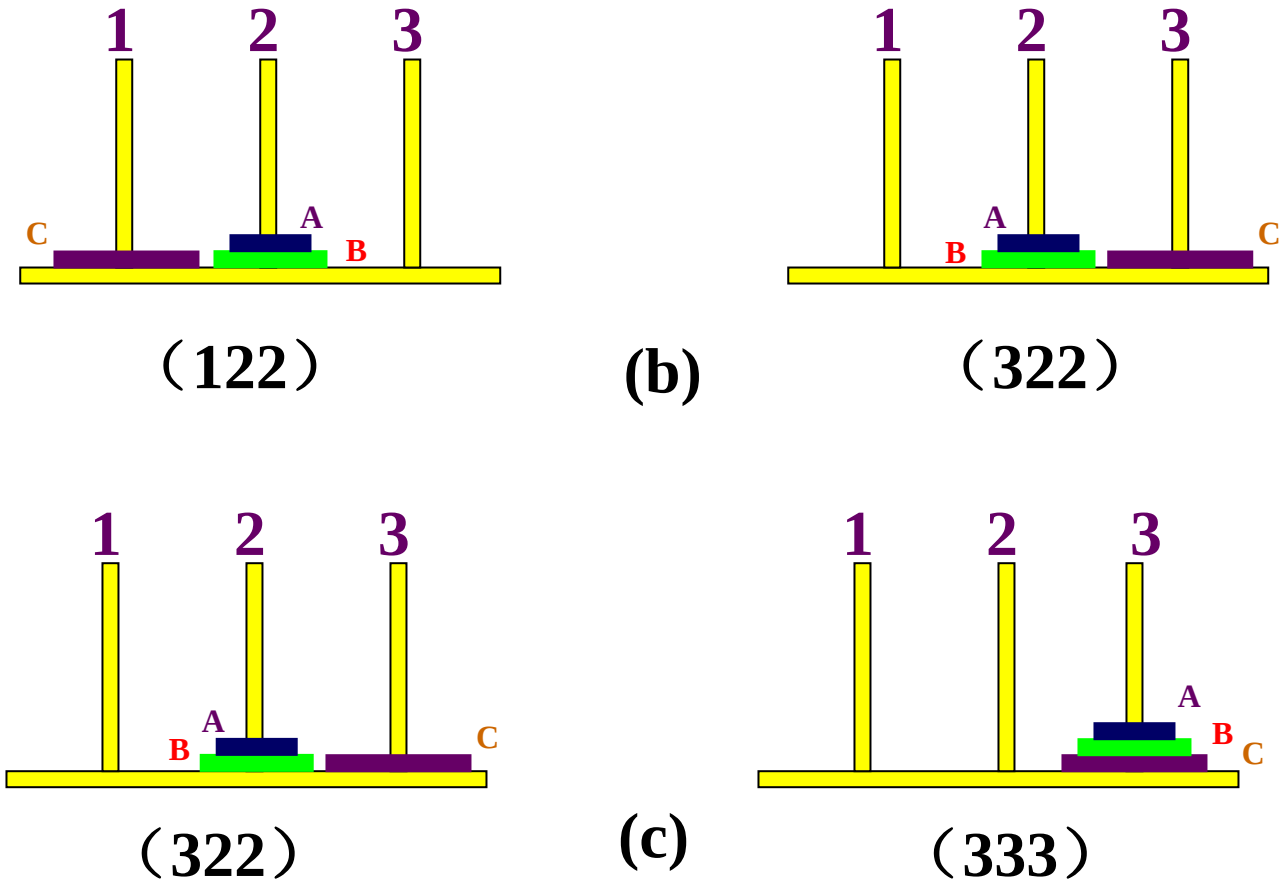


Fig. Reduction for the Hanoi Tower problem

问题归约

- 问题归约法示例1：汉诺塔问题（Hanoi-Tower Problem）

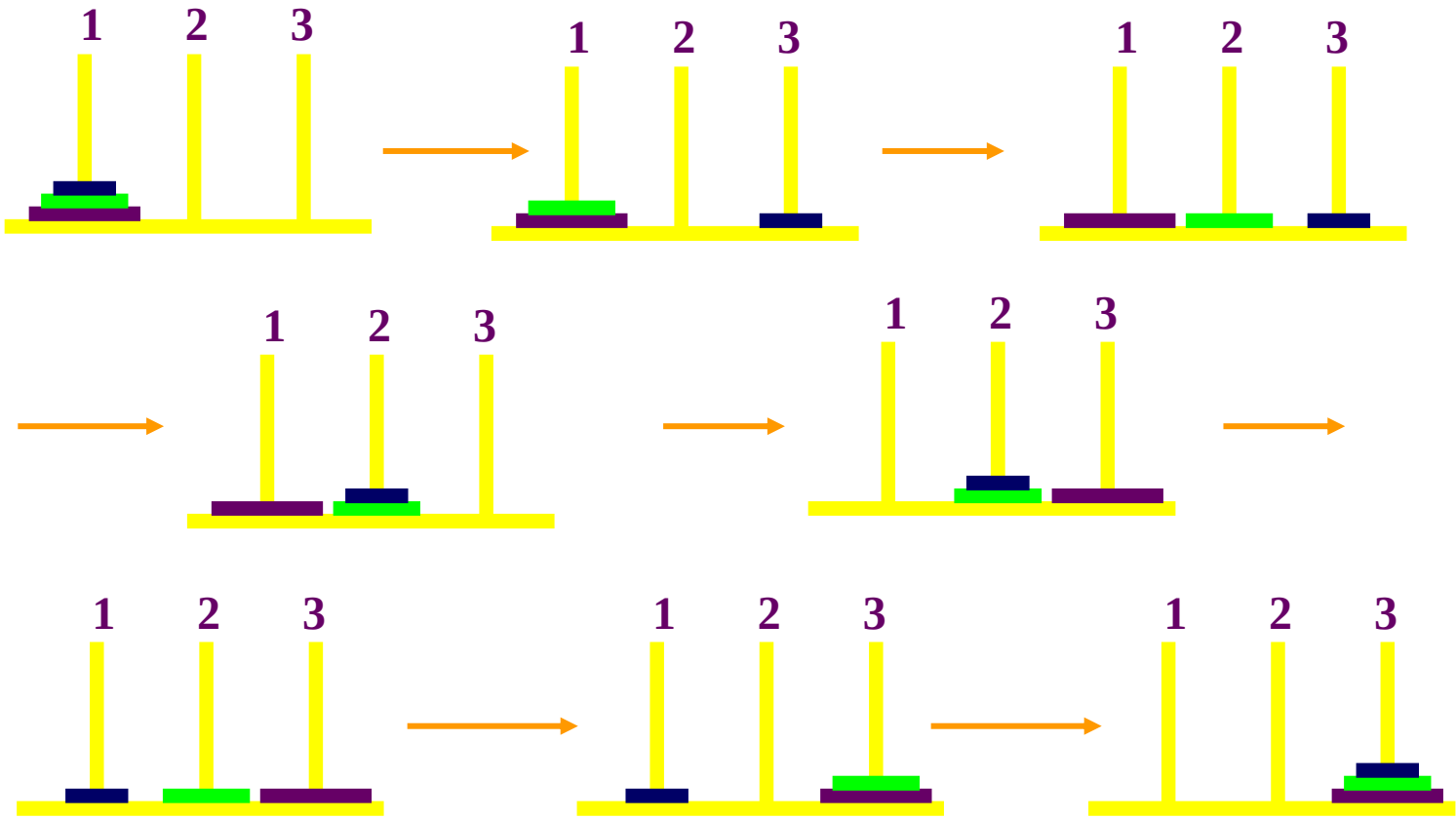


Fig. The solving process of Hanoi Tower problem

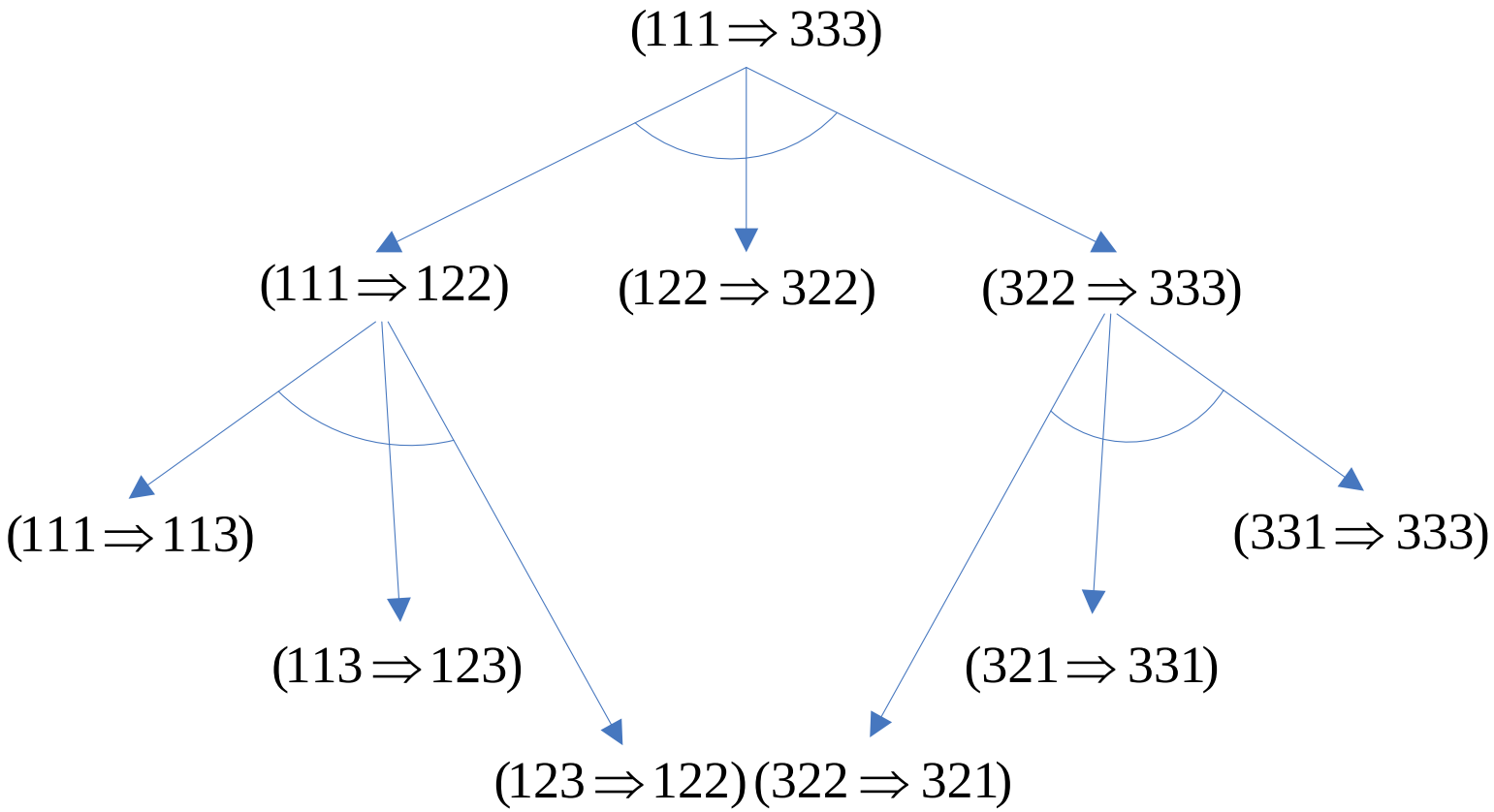
问题归约

- 问题归约法示例1：汉诺塔问题（Hanoi-Tower Problem）

- 归约描述
  - 问题归约方法是应用算符来把问题描述变换为子问题描述;
  - 可以用状态空间表示的三元组合(S、F、G)来规定与描述问题;对于汉诺塔问题,子问题[(111)→(122)]， [(122)→(322)]以及[(322)→(333)规定了最后解答路径将要通过的脚踏石状态(122)和(322)] ;
  - 问题归约方法可以应用状态、算符和目标这些表示法来描述问题,这并不意味着问题归约法和状态空间法是一样的。

问题归约

- 问题归约法示例1：汉诺塔问题（Hanoi-Tower Problem）



谓词逻辑法

Predicate Logic

谓词逻辑

- 逻辑推理是人类另一常用的问题求解技术
- 如何进行逻辑推理?
- 推理的过程怎样?
- 怎么实现自动推理?
- 所有这些问题的基础首先必须是怎样用适合逻辑推理的方式表示问题

谓词逻辑

- 概念：

  - 谓词逻辑法采用谓词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。

  - 谓词逻辑是一种形式语言，能够把数学中的逻辑论证符号化。谓词逻辑常与其他方法混合使用，灵活方便，可以表示比较复杂的问题。

谓词逻辑

- 推理示例：马普尔小姐探案

谁是马克谁是约尔?

谓词逻辑

- 观察结果
  - 马克是右撇子，手表戴在左手
  - 约尔是左撇子，手表戴在右手

  } 事实
- 推理规则
  - 如果手表戴在左手，那么他是马克
  - 如果手表戴在右手，那么他是约尔

  } 规则
- 结论
  - 只是穿着不同衣服的同一个人——约尔

人类的推理可以理解语义
机器如何进行这样类似的推理？
需要将推理的过程与理解分割开，将其形式化

谓词逻辑

- 推理的一般形式

  已知：事实1，事实2，…

  如果 事实1 那么 结论1

  如果 事实2 那么 结论2

  ….

  得到：结论1，结论2，…

- 将事实与规则等抽象出来，不涉及具体内容，借助一些符号来表示，推理过程可以被形式化

  P：某已知事实

  P→Q：如果 P 那么 Q

  结论： Q

  这个过程不需要直觉和解释

谓词逻辑

- 符号与形式语言

- 自然语言不适合计算机处理
  - 例：用红墨水写一个“蓝”字，请问，这个字是红字还是蓝字?
- 需要一种无歧义，方便存储和表达的形式化符号表征体系
  - 数理逻辑
    - 命题逻辑
    - 谓词逻辑

谓词逻辑

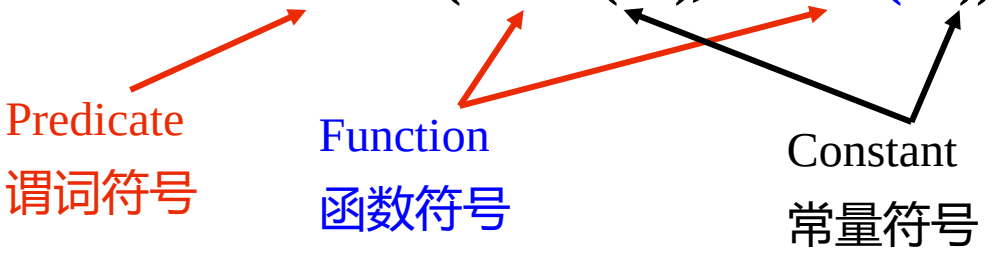
- 符号与形式语言
  - 什么是谓词?
    - 原子命题中刻画个体的性质或个体间关系的成分
  - 谓词逻辑是一种形式语言
  - 是目前为止能够表达人类思维活动规律的一种最精确的语言
  - 接近自然语言，又方便存入计算机处理
  - 最早应用于人工智能中表示知识
  - 适合于表示事物的状态、属性、概念等，也可用来表示事物间确定的因果关系

谓词逻辑

- 概念：
  - 1. Predicate Calculus（谓词演算）
    - Syntax and Semantics （语法和语义）
      - Basic components: predicates, variables, function, constants, punctuation marks
    - Atomic Formulas （原子公式）
      - Atomic Formulas = Predicate + Terms

INROOM(ROBOT, r1)

MARRIED(father(LI), mother(LI))



谓词逻辑

- 连词和量词(Connective &Quantifiers)
  - Connective (∧, ∨ , =>, ~)

    Conjunction (与及合取): Conjunction ∧ is used to represent a compound sentence with AND relationship

Example: LIKE(I，MUSIC)∧LIKE(I，PAINTING)

(我喜爱音乐和绘画。)

LIVES(L1，HOUSE-1)∧COLOR(HOUSE-1，YELLOW)

(李住在一幢黄色的房子。)

谓词逻辑

- 连词和量词(Connective &Quantifiers)
  - Connective (∧, ∨ , =>, ~)

    Disjunction（或及析取）： Disjunction∨ is used to represent a compound sentence with OR relationship

Example：
PLAYS(LIMING，BASKETBALL) ∨ PLAYS(LIMING，FOOTBALL)
(李明打篮球或踢足球。)

谓词逻辑

- 连词和量词(Connective &Quantifiers)
  - Connective (∧, ∨ , =>, ~)

Implication（蕴涵）： "=>"is used to represent a compound sentence with "IF-THEN" relationship

Ex： RUNS (LIUHUA , FASTEST) => WINS (LIUHUA, CHAMPION)

If Liuhua runs fastest, then he wins the champion.

Not（非）： Symbol ~ (in some literature, a symbol ¬) is used to negate the value of a formula

Ex： ~ INROOM(ROBOT, r2)

Robot is not in Room 2

谓词逻辑

- 连词和量词(Connective &Quantifiers)

Ex1：李明是个学生，他住的房间是白色的。

$Isa(Liming, Student) \wedge Lives(Liming,House1) \wedge Color(House1, White)$

Ex2：李明在学校的话，Wang也在学校。

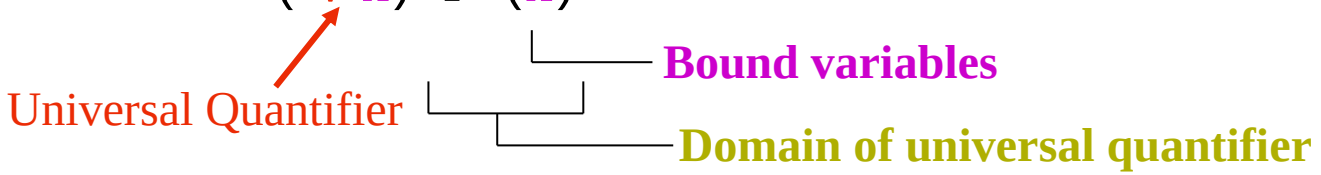
$At(Liming,School) \Rightarrow At(Wang, School)$

谓词逻辑

- 连词和量词(Connective &Quantifiers)

Universal Quantifier（全称量词）： An atomic formula P（x）has the value T for all variables x:

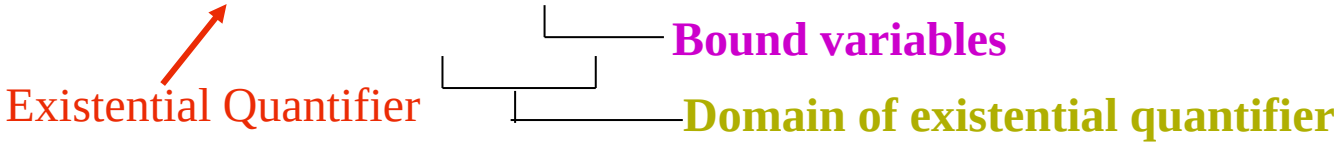
（∀ x）P（x）



Existential Quantifier（存在量词）

If there exists at least one variable of x that makes P (x)has the value T : (∃ x) P(x)

Ex：(∃x)INROOM(x,r1) (1号房间内有个物体)



谓词逻辑

- 量词(Quantifiers)

Ex1: All roads lead to Roma

（∀x）[Road(x)⇒ Lead(x, Roma)]

Ex2: For all variables x，if x is an integer，then x is positive or negative

（∀x）[I(x) ⇒ (P(x) ∨ N(x))]

Ex3: Mary gives each person one book.

（∀x） （∃y）[Person(x)∧Book(y)∧Give(Mary,x,y)]

谓词逻辑

- 量词公式解读

(∀x) : for every object *x* in the universe the following holds

(∃x) : there exists an object *x* in the universe which satisfies the following (for some object x in the universe the following holds)

***F*(*x*, *y*) : "*x* flies faster than *y*" , the universe be the set of airplanes**

(∀x)(∀y) F(x, y)	For every airplane *x* the following holds: *x* is faster than every (any) airplane *y*
(∀x)(∃y) F(x, y)	For every airplane *x* the following holds: for some airplane *y*, *x* is faster than *y*
(∃x)(∀y) F(x, y)	There exist an airplane *x* which satisfies the following: for every airplane *y*, *x* is faster than *y*
(∃x)(∃y) F(x, y)	

谓词逻辑

- 量词公式优先级

(1) F the inside: the one closest to the atomic formula is applied first

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

(2) The positions of the same type of quantifiers can be switched

$(\exists x)(\exists y)(\exists z)\ P(x, y, z)$ → $(\exists y)(\exists x)(\exists z)\ P(x, y, z)$

$(\exists x)(\exists y)(\exists z)\ P(x, y, z)$ → $(\exists z)(\exists y)(\exists x)\ P(x, y, z)$

(3) The positions of different types of quantifiers can not be switched

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

谓词逻辑

- 量词公式
  - variable

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

y is free variable in the domain of (∃ x), but y is bound variable in the domain of (∀ y).

谓词逻辑

- 谓词公式(Predicate Formula)

  - Definition: Atomic Formula
    - An atomic formula or atom is simply a predicate applied to a tuple of terms; that is, an atomic formula is a formula of the form $\mathbf{P(x_1, x_2, \ldots, x_n)}$ for P an predicate, and the $x_1, x_2, \ldots, x_n$ terms.
  - Definition: Predicate calculus formula
    - Predicate calculus formula is obtained by composing atoms with logical connectives.

谓词逻辑

- 谓词公式(Predicate Formula)
  - Well-Formed Formulas （WFF，合适公式）
    - For predicate logic, WFF is obtained by:

(1) Each atomic formula is WFF.

(2) If A is WFF, then～A is WFF too.

(3) If both A and B are WFF, then (A∧B), (A∨B), (A=>B) and (A ⇄ B) are WFF too.

(4) If A is WFF, and x is a free variable in A, then(∀ x)A and (∃ x)A are WFF.

(5) Only such a predicate formula that is derived according to the above four rules is a wff.

谓词逻辑

- 谓词公式(Predicate Formula)
  - Well-Formed Formula's Characters
    - Truth Values

Truth Table

P	Q	P∨Q	P∧Q	P⇒Q	～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

    - Equivalence

      Two WFFs w1 and w2 are equivalent if and only if the truth values of w1 and w2 are the same for all interpretations.

谓词逻辑

- 谓词公式(Predicate Formula)

(1) The negation of negation： $\sim(\sim P) = P$

(2) $P \vee Q = \sim P \Rightarrow Q$

(3) Demorgan's theorems： $\sim(P \vee Q) = \sim P \wedge \sim Q$

$\sim(P \wedge Q) = \sim P \vee \sim Q$

(4) Distributive law： $P \wedge (Q \vee R) = (P \wedge Q) \vee (P \wedge R)$

$P \vee (Q \wedge R) = (P \vee Q) \wedge (P \vee R)$

(5) Commutative law： $P \wedge Q = Q \wedge P$

$P \vee Q = Q \vee P$

(6) Associative law： $(P \wedge Q) \wedge R = P \wedge (Q \wedge R)$

$(P \vee Q) \vee R = P \vee (Q \vee R)$

(7) Contrapositive law： $P \Rightarrow Q = \sim Q \Rightarrow \sim P$

谓词逻辑

- 谓词公式(Predicate Formula)

(8) $\sim(\exists x)P(x) = (\forall x)[\sim P(x)]$

$\sim(\forall x)P(x) = (\exists x)[\sim P(x)]$

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

$(\exists x)[P(x) \vee Q(x)] = (\exists x)P(x) \vee (\forall x)Q(x)$

(10) $(\forall x)P(x) = (\forall y)P(y)$

$(\exists x)P(x) = (\exists y)P(y)$

谓词逻辑

- 置换与合一(Substitution and Unification)
  - Substitution
    - Concept
      - Modus ponens

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

      - Universal specialization

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

bound var　　free var

      - Combination

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

- 置换与合一(Substitution and Unification)

Example:

4 substitutions of P[x, f(y), B]

s1={z/x, w/y} then P[**x**, f(**y**), B]s1= P[**z**, f(**w**), B]

s2={A/y} then P[x, f(**y**), B]s2= P[x, f(**A**), B]

s3=(q(z)/x, A/y) then P[**x**, f(**y**), B]s3= P[**q(z)**, f(**A**), B]

s4=(c/x, A/y) then P[**x**, f(**y**), B]s4= P[**c**, f(**A**), B]

谓词逻辑

- 置换与合一(Substitution and Unification)
  - Definition
    - Apply term (A) to substitute the variable (x) in function, marked as **Es**. An expression is obtained by substituting terms for variables in that expression.
  - Characters
    - Associative

      $$(E\ s1)\ s2 = E\ (s1\ s2)$$
      $$(s1\ s2)\ s3 = s1\ (s2\ s3)$$

      where E is an expression, $s1$,$s2$,$s3$ are instances of expression
    - Noncommutative

      $$s1\ s2 \neq s2\ s1$$

谓词逻辑

- 置换与合一(Substitution and Unification)
  - Unification
    - Definition

      Unification is to find substitution *s* of terms for variables to make expression set {Ei} identical, which is represented by {Ei} *s*.
    - Unifiable

      If

      E1 *s* = E2 *s* = E3 *s* =…

      Expression set {Ei} is unifiable, *s* is unifier.

- 置换与合一(Substitution and Unification)

Example:

For expression set **{P[x,f(y),B], P[x,f(B),B]}**

order

***s*={A/x, B/y}**

then

P[x,f(y),B] *s*= P[x,f(B),B]*s*

P[x,f(B),B]*s*= P[A,f(B),B]

} **the same**

So, *s*={A/x, B/y} is {P[x,f(y),B], P[x,f(B),B]}'s unifier,

and *s*={B/y} is {P[x,f(y),B], P[x,f(B),B]}'s simplest unifier.

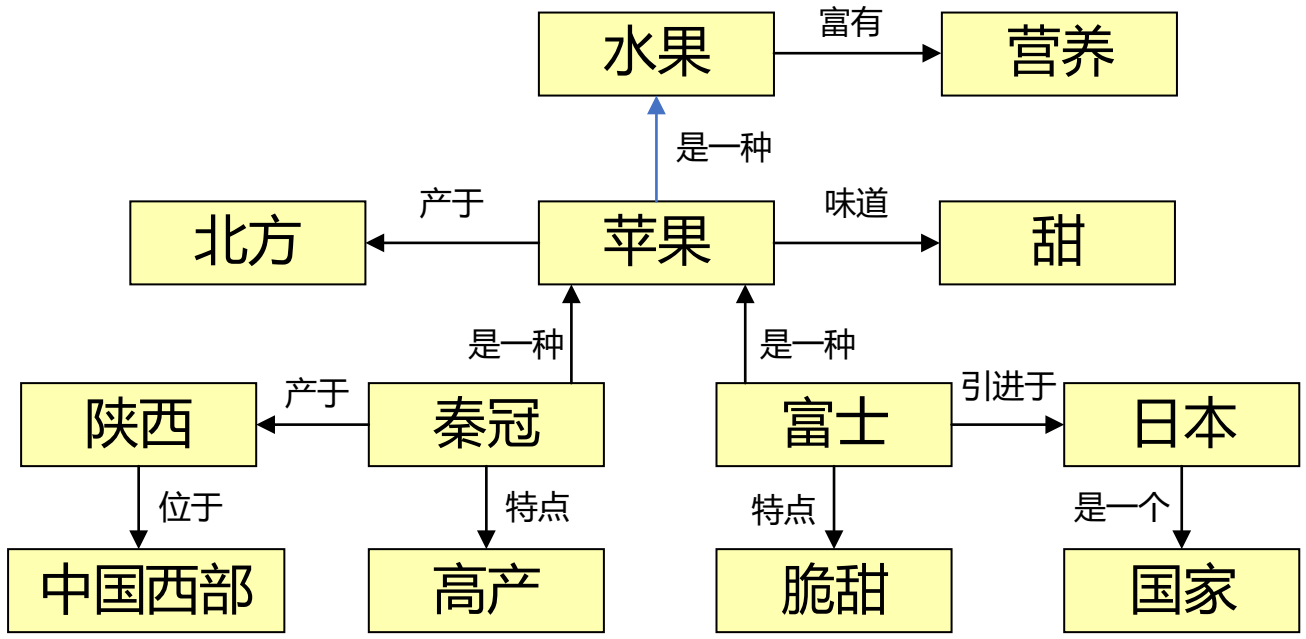
语义网络法

Semantic Network

- J.R.Quillian在其博士论文中作为人类联想记忆的一个显式心理学模型最先提出
- 1972年，Simmons首先将语义网络表示法用于自然语言理解系统

语义网络

苹果是一种富含营养的水果，苹果产于北方，味道甜。
秦冠和富士是两种不同的苹果，秦冠产于陕西，产量高；富士引自日本，苹果脆甜



语义网络

- 语义网络是知识的一种结构化图解表示，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（有向边）用于表示节点间的关系，其中边上的标记就是边的语义。
- 语义网络表示由下列四个相关部分组成：
  - 词法部分：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。
  - 结构部分：叙述符号排列的约束条件，指定各弧线连接的节点对。
  - 过程部分：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。
  - 语义部分：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占有物和对应弧线。

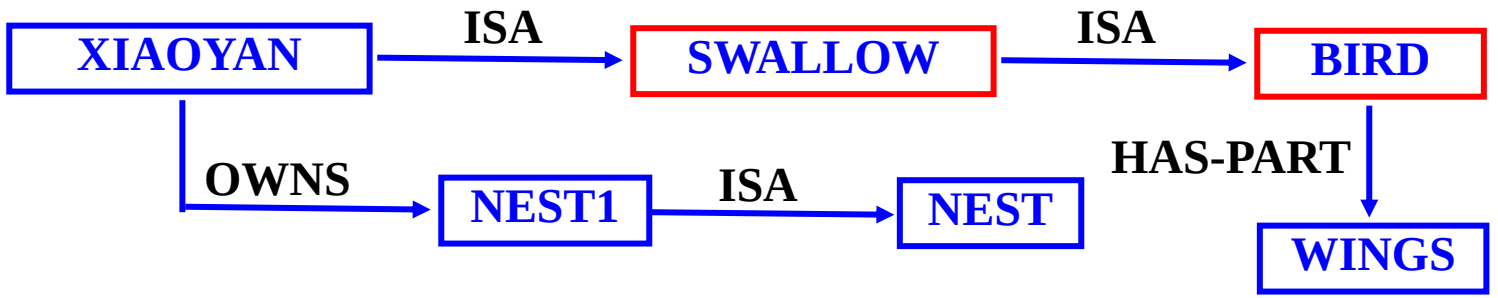
语义网络

- 语义网络
  - 通过概念及其语义关系来表达知识的一种网络图
  - 带标志的有向图
  - 语义网络是知识的一种结构化表示方法
  - 由节点(node)与有向弧(arc)组成
  - 节点表示概念、事物、事件、情况等
  - 弧表示语义关系

语义网络

- Representation of Two-Element Semantic Network

Ex1: All swallows are birds



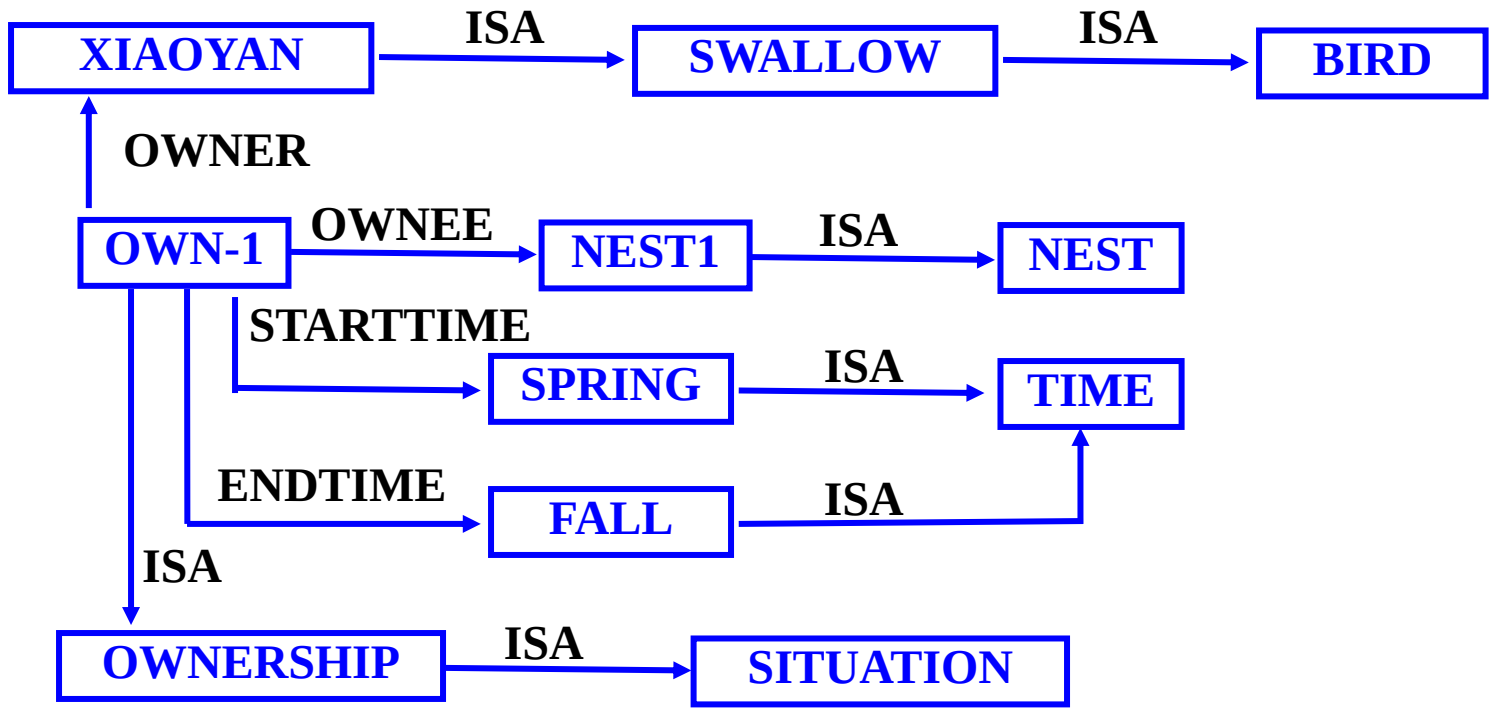
XIAOYAN is a swallow.

Bird has wings.

XIAOYAN owns a nest.

语义网络

Ex2: XIAOYAN owns a nest from spring to autumn.

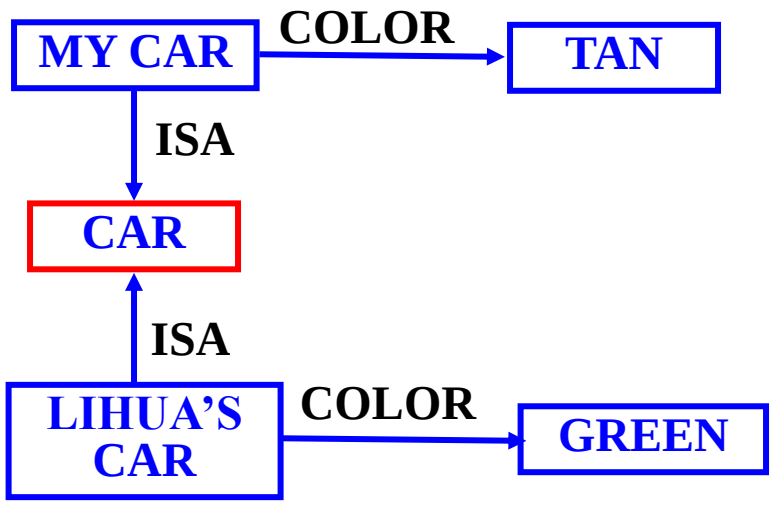


语义网络

- Selecting semantic primitives
  - looking for basic concepts and basic arcs

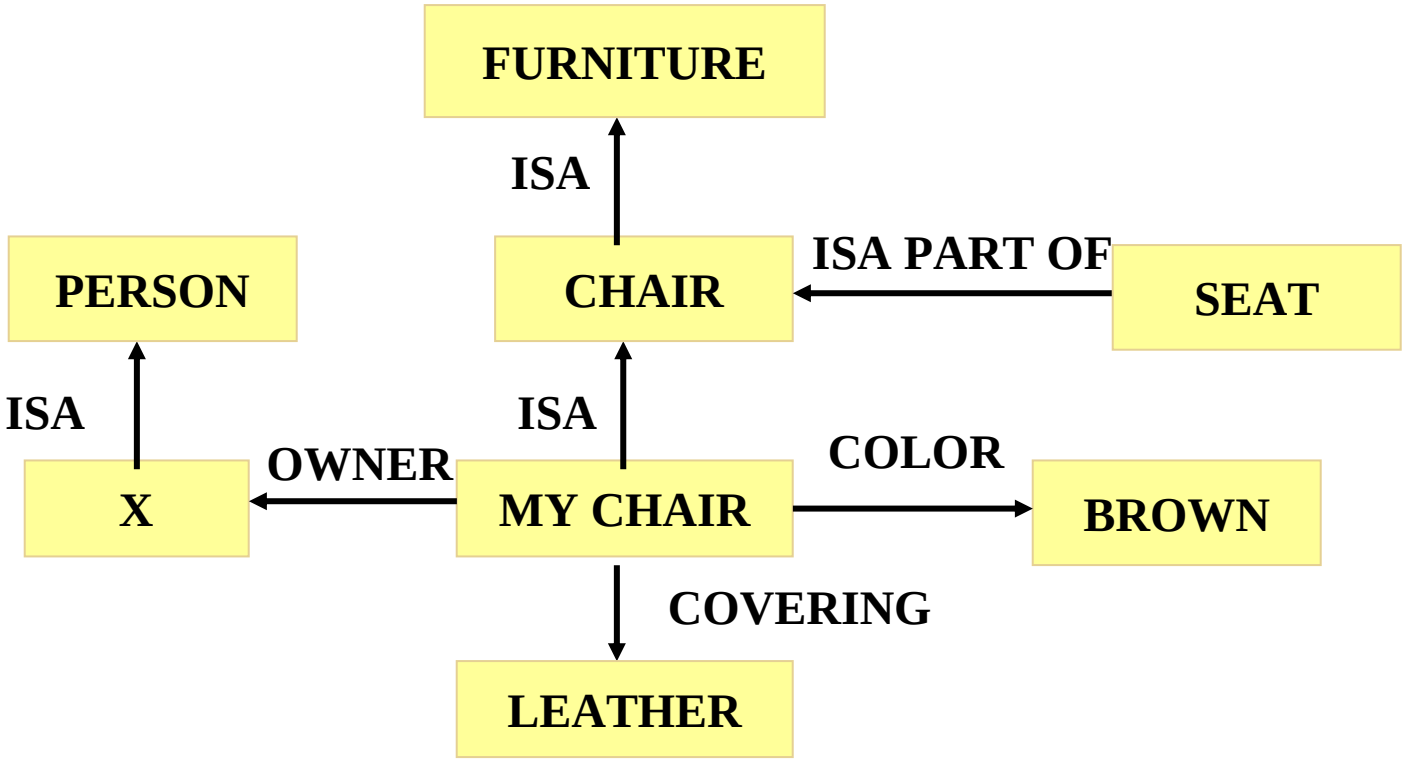
My car is tan.

Lihua's car is green

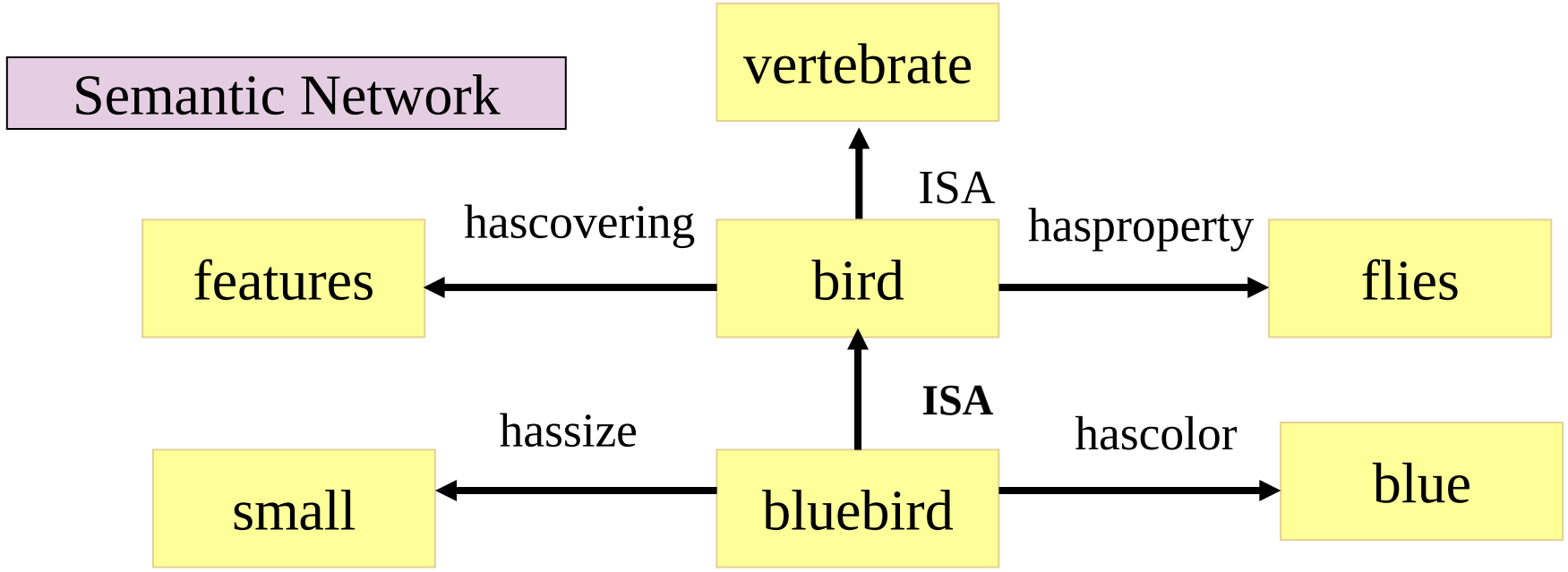


语义网络

Example 3. The color of my chair is brown; the covering of the chair is leather; chair is one kind of furniture; chair is part of seat; chair's owner is X; X is a person。



语义网络



Predicate Calculus

hassize(bluebird, small)
hascovering(bird,features)
hascolor(bluebird,blue)
hadproperty(bird,flies)
isa(bluebird,bird)
isa(bird,vertebrate)

语义网络

- Representation of Multi-Element Semantic Network
  - Predicate Logic = Semantic Network

（Predicate Logic）

ISA（LIMING，MAN）或 MAN（LIMING）

（Semantic Network）

语义网络

- Essence of Multi-Element Semantic Network
  - Turn a Multi-Element to a set of Two-Element by conjunction.

$$R(X_1, X_2, \ldots, X_n)$$

$$R_{12}(X_1, X_2) \wedge R_{13}(X_1, X_3) \wedge \ldots \wedge R_{1n}(X_1, X_n)$$
$$\ldots\ldots$$
$$R_{n-1\ n}(X_{n-1}, X_n)$$

语义网络

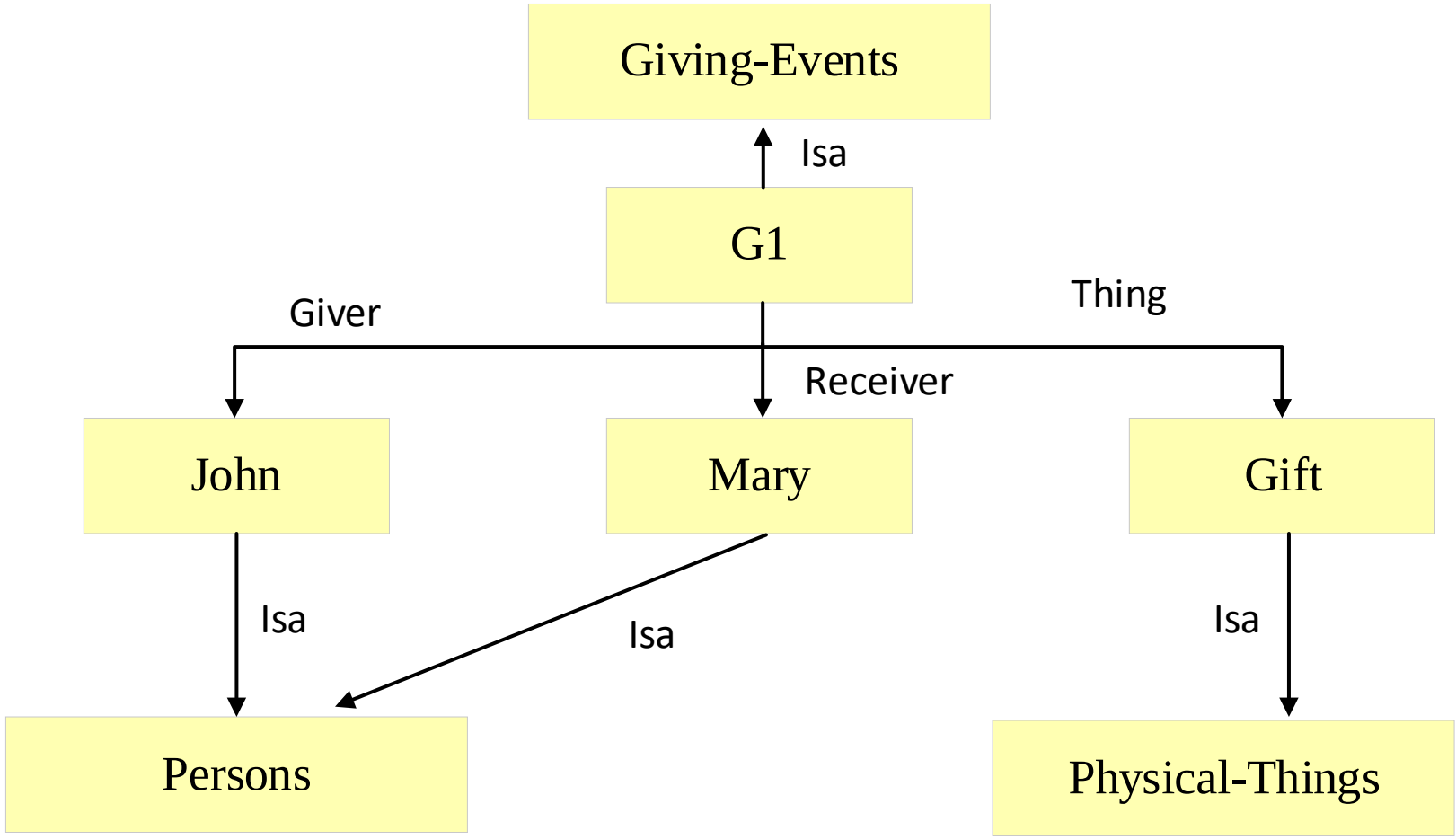
Ex4: John gives Mary a gift.

Gives(John, Mary, Gift)

- Three-element →Two-element
  - The whole sentence describes a instance of Giving-Event case, as G1.
  - John is the Giver in G1.
  - Mary is the Receiver in G1.
  - Gift is a thing in G1.

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

语义网络

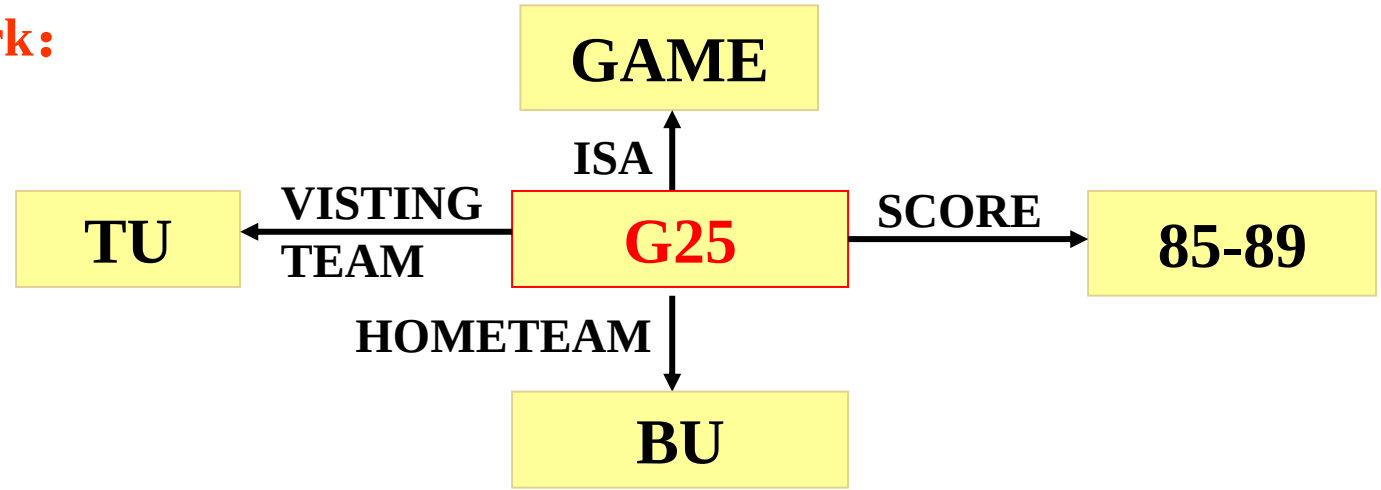

Giving-Events
Isa
G1
Giver
Thing
Receiver
John
Mary
Gift
Isa
Isa
Isa
Persons
Physical-Things

语义网络

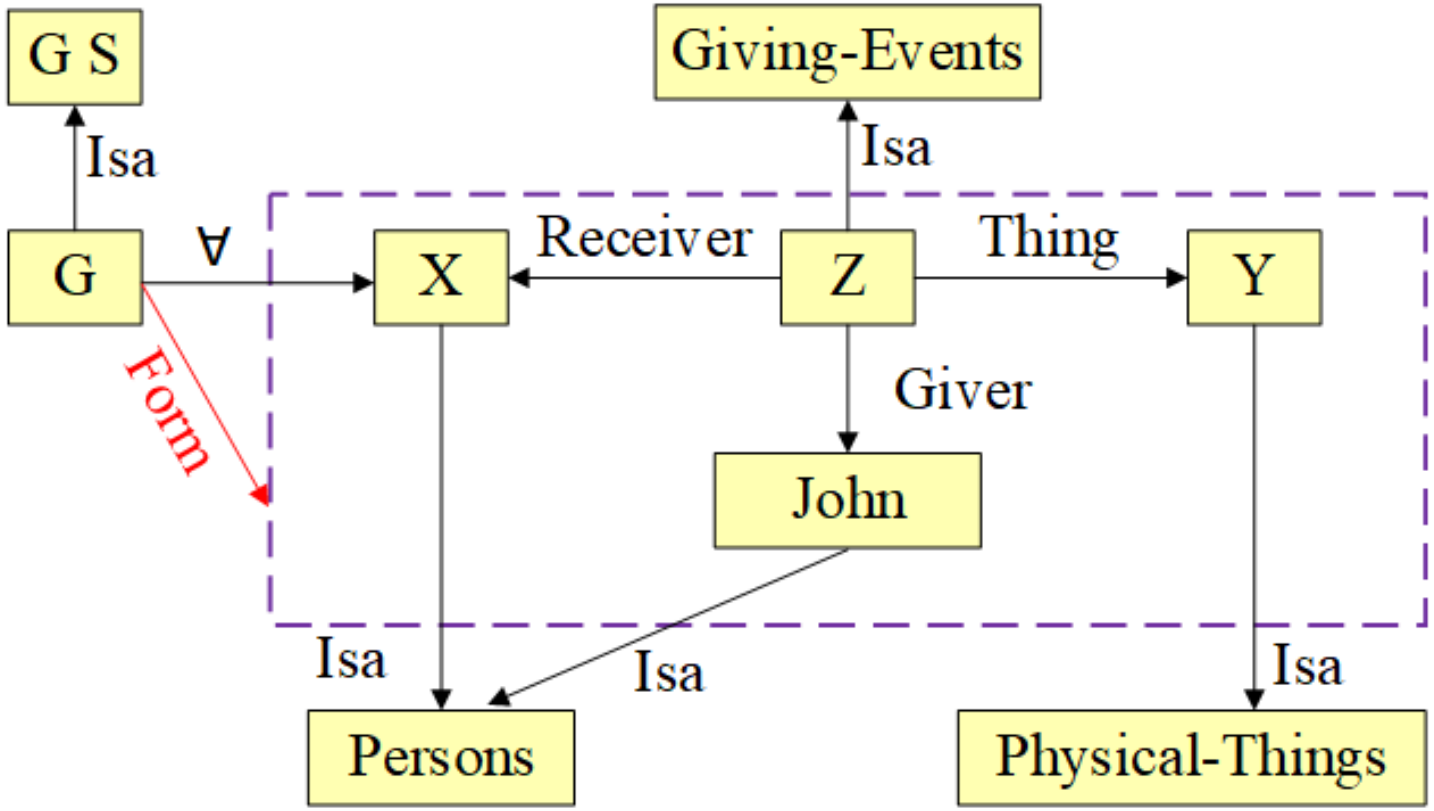
Ex5: There is a basketball game between two teams: Boston University(home team.BU) and TSINGHUA University (TU).The game is hold on Boston,1996 and the score is 85:89.

Predicate Logic: **SCORE(BU, TU, (85:89))**

Semantic Network：

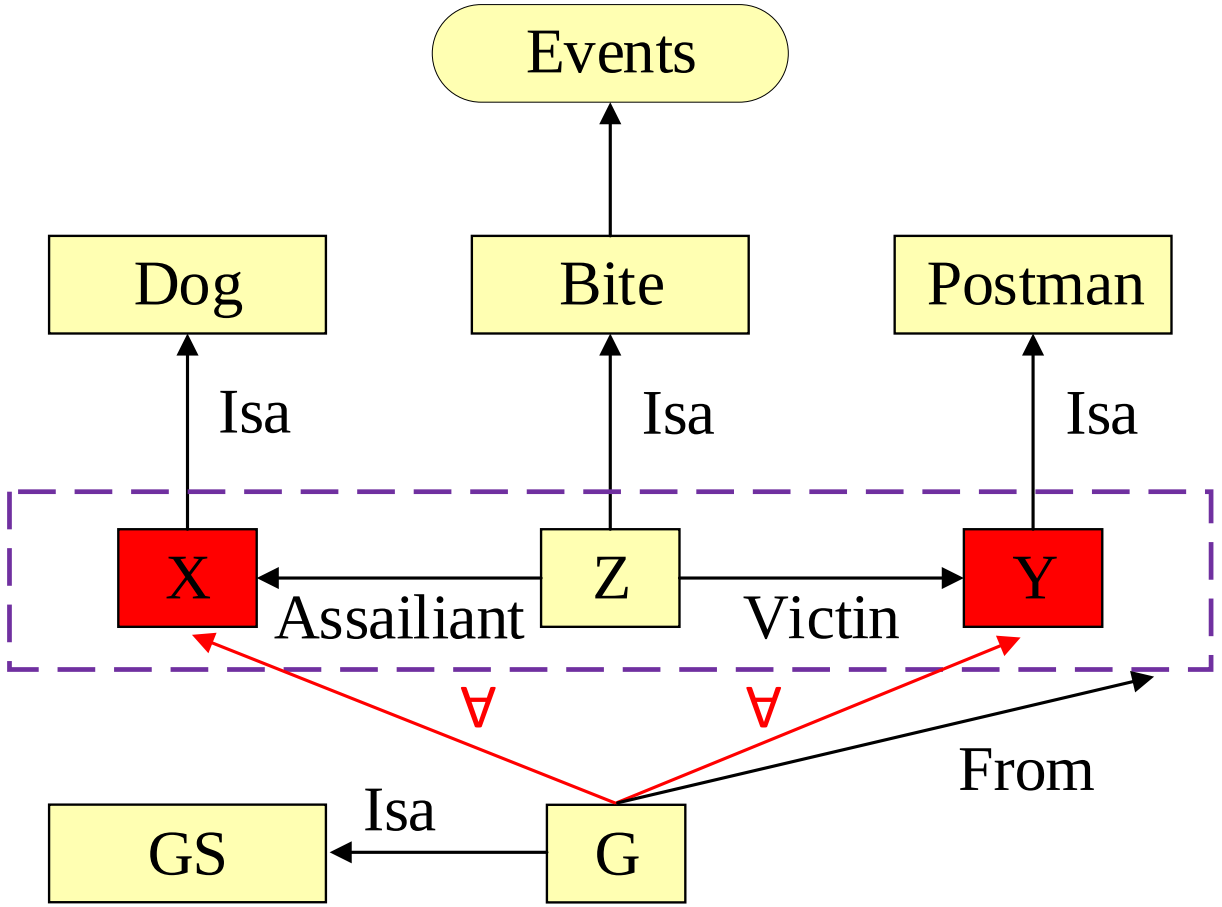


语义网络

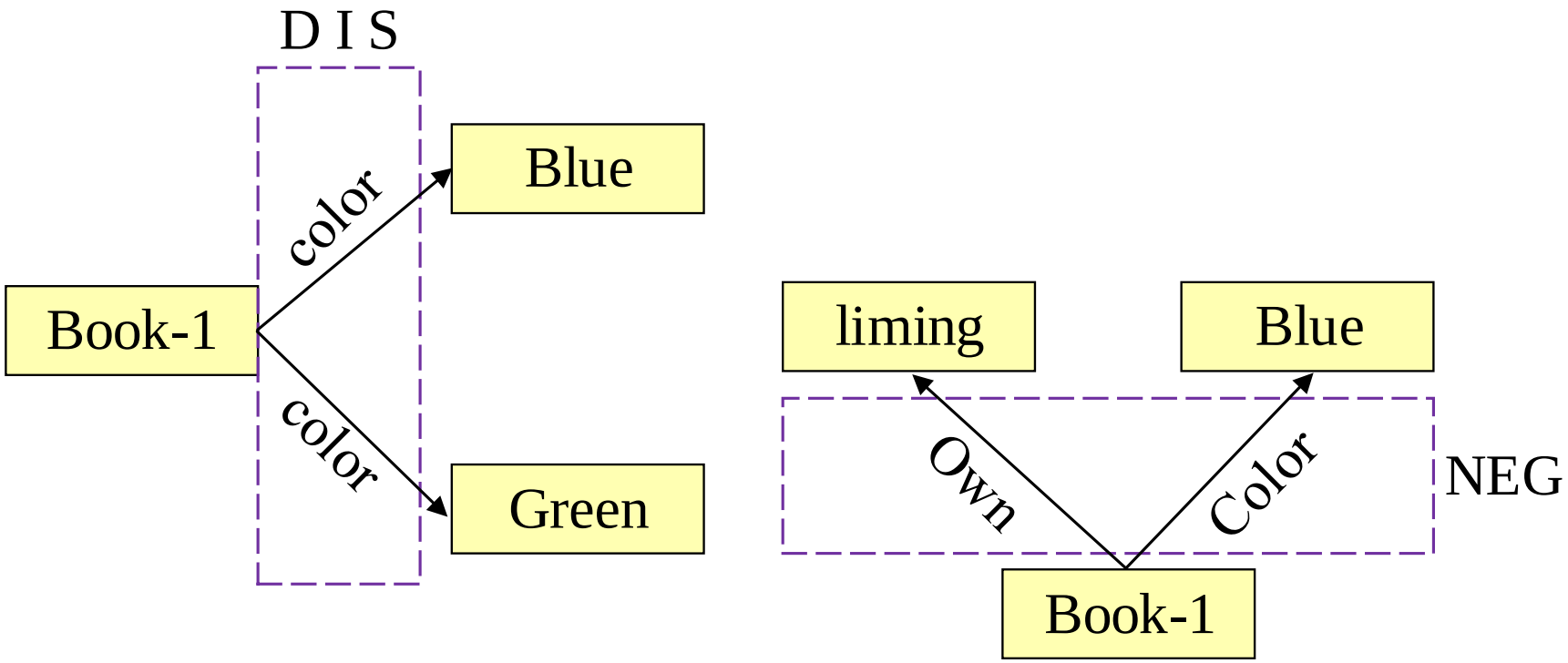

G S
Giving-Events
Isa
Isa
G
∀
X
Receiver
Z
Thing
Y
Form
Giver
John
Isa
Isa
Isa
Persons
Physical-Things

语义网络

- About 2 bound variables by universal quantifiers

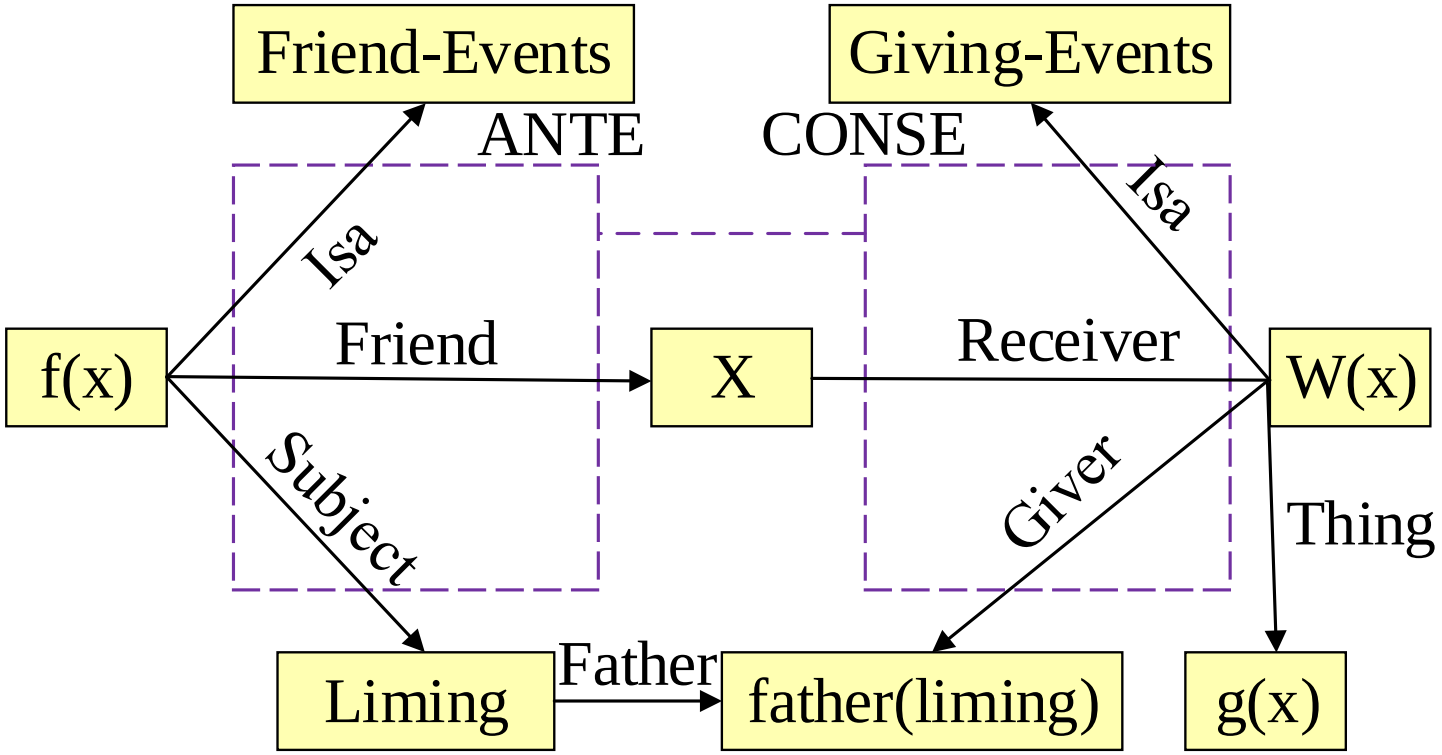


语义网络


D I S
Book-1
color
Blue
color
Green
liming
Blue
Own
Color
NEG
Book-1

语义网络

Ex7: Liming’s father gave Liming’s each friend a gift



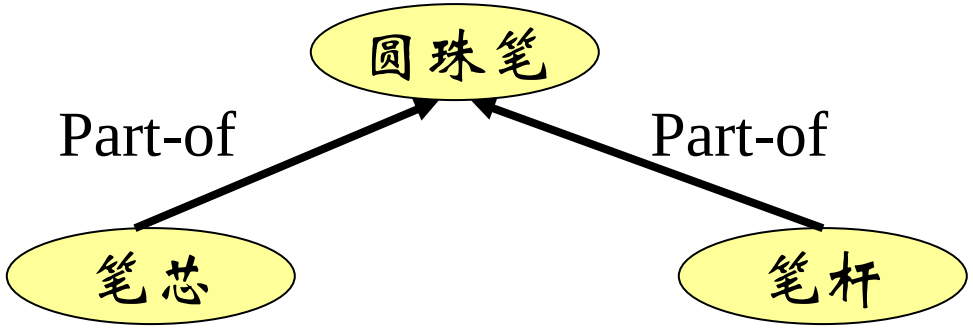
语义网络

- 节点 Node
  - 概念节点
    - 用以表示基本概念
  - 实例节点
    - 用以表示具体事物或属性

语义网络

- 弧
  - 以个体为中心（对应节点一般是名词性个体或概念）
    - ISA （实例联系）
    - AKO（A Kind Of, 泛化联系，子类）

- Part-of（聚集联系）

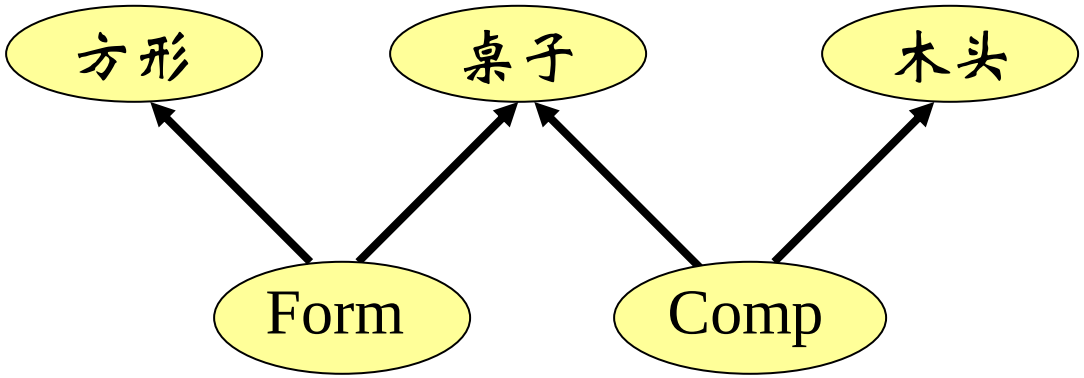


语义网络

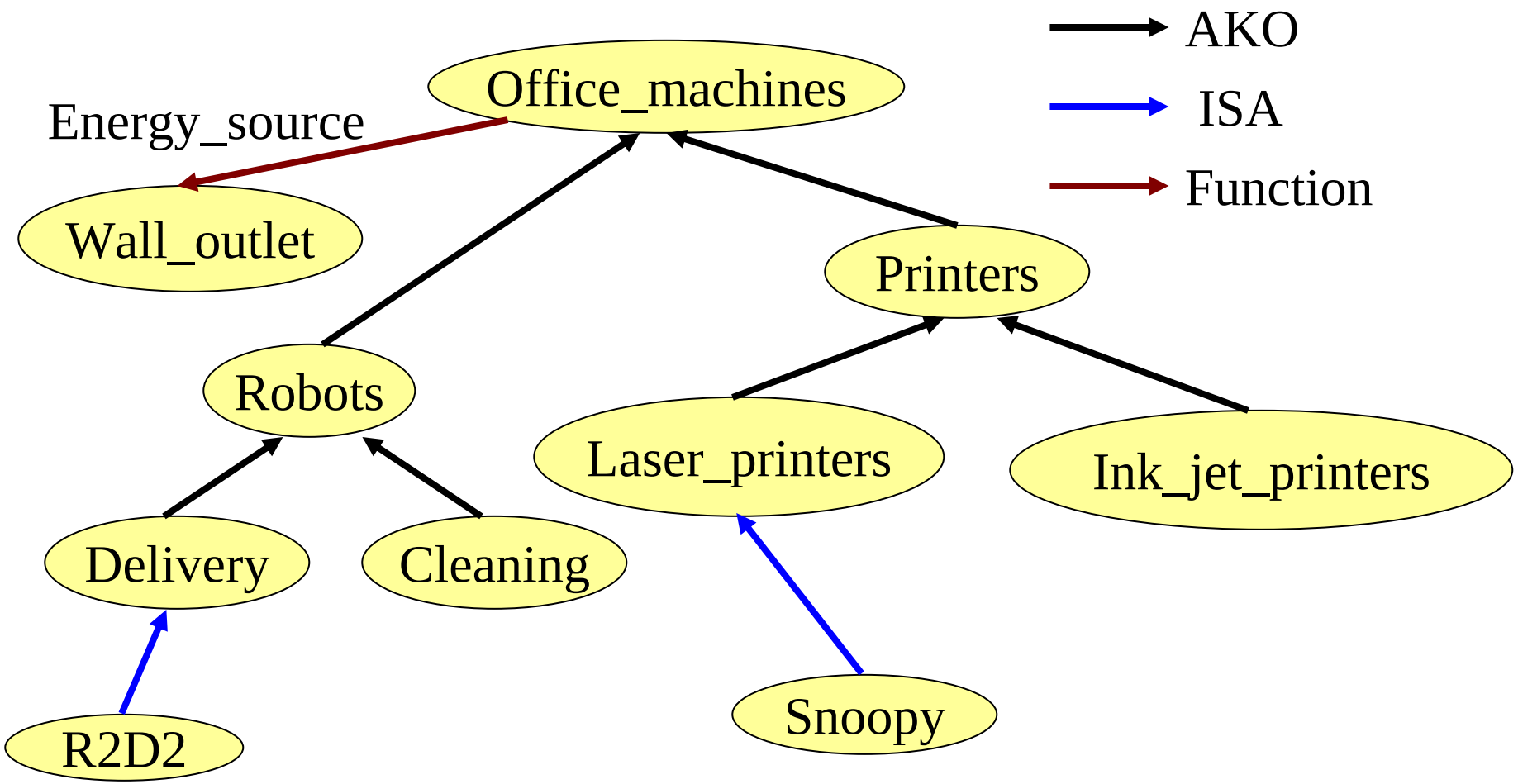
- 以谓词或关系为中心

  例：有一张木头做的方桌

Form (桌子,方形) Comp (桌子,木头)



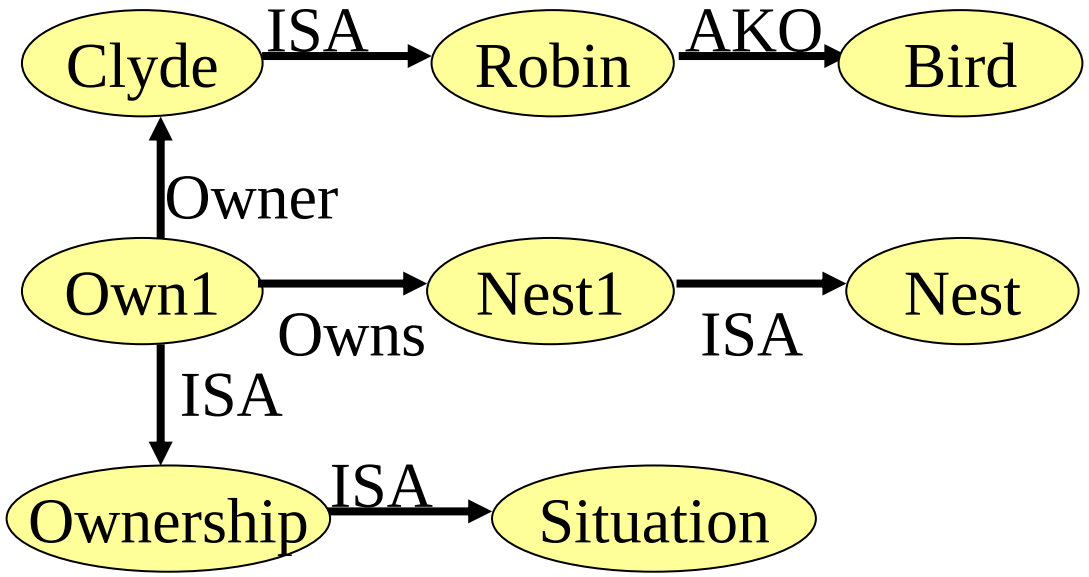
语义网络


AKO
ISA
Function
Office_machines
Energy_source
Wall_outlet
Printers
Robots
Laser_printers
Ink_jet_printers
Delivery
Cleaning
R2D2
Snoopy

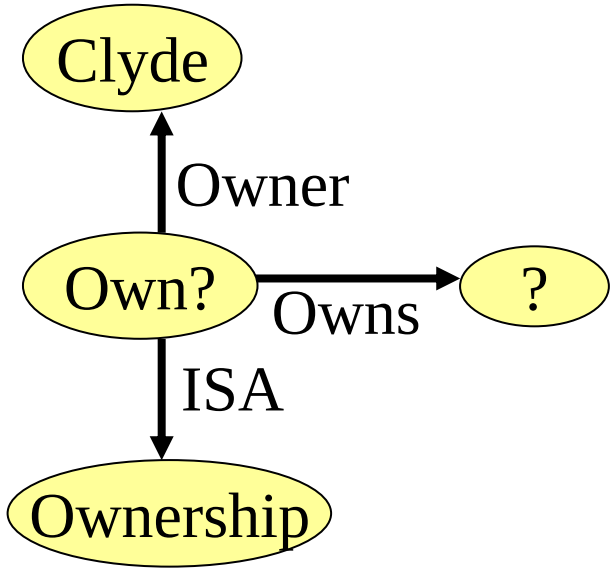
语义网络

两种推理机制：匹配、继承

- 匹配(包括节点和弧的匹配)
  - 构造目标网络块
  - 在事实网络中寻找匹配



What does Clyde own?



语义网络

- 继承

  把对事物的描述从概念节点或类节点传递到实例节点

三类继承性

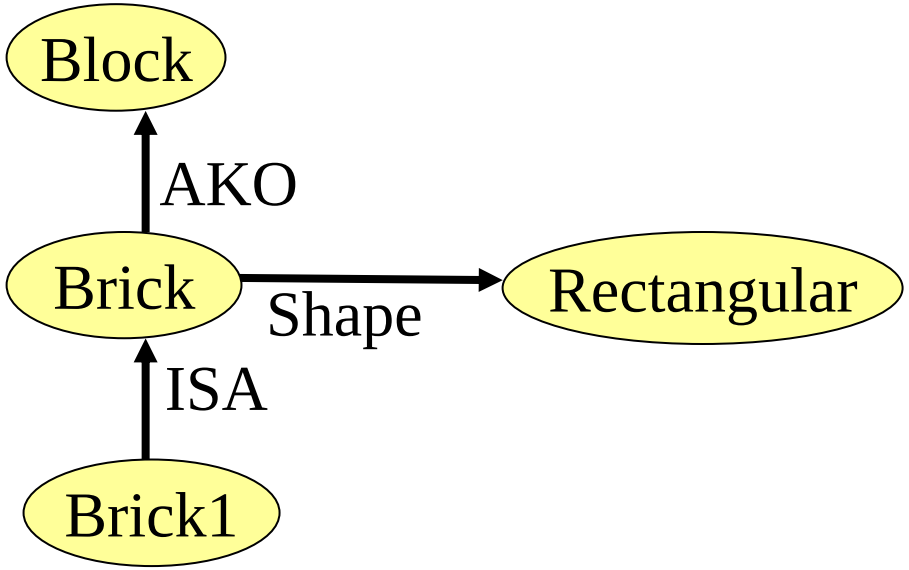
- 直接传递(pass) —— 子节点直接继承父节点的属性
- 附加传递(add) —— 子节点把父节点的属性和自己的属性相综合
- 排斥传递(exclude) —— 子节点与父节点的属性不相容，抑止传递

语义网络

- 三种继承方式
  - 值继承

is-a, AKO(a kind of)

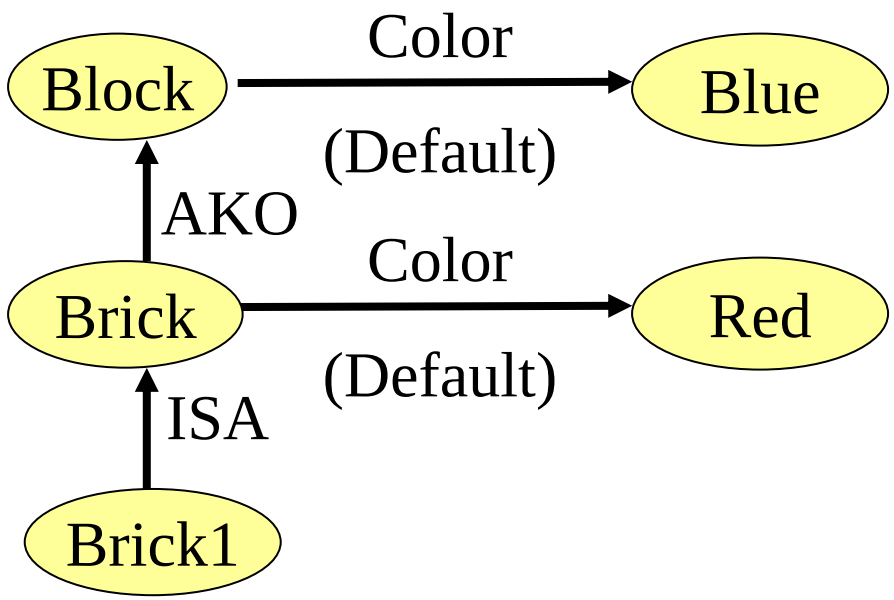
求Brick1的形状



语义网络

- 默认继承(Default)

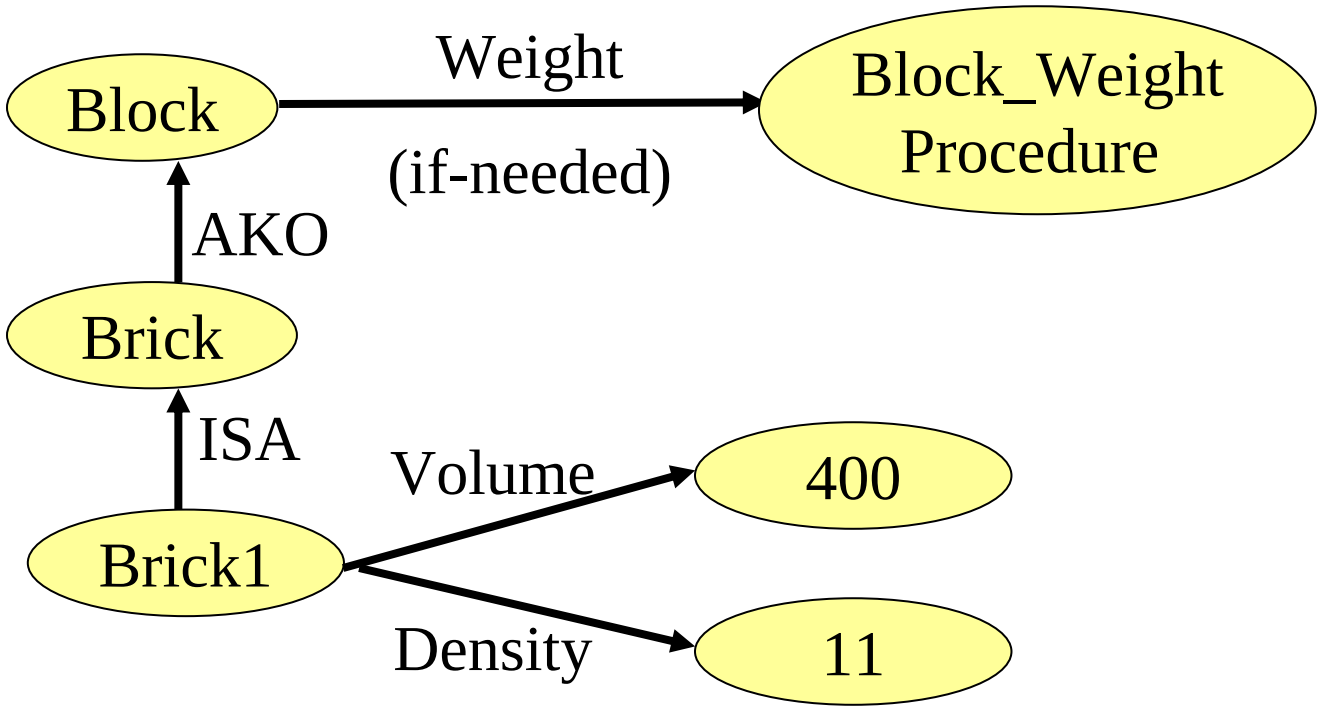
默认值——具有相当程度的真实性但不能十分肯定的值
如：鸟会飞；头疼可能是感冒



语义网络

- if-needed (“如果需要”继承，“附加过程”继承)

某些情况下，对事物的描述不能直接从概念节点或类节点继承得到，但可利用已知信息来计算，这种计算程序称为if-needed程序



语义网络

- 具有结构性

  适于表示分类学型知识和事物特性的知识
- 灵活性

  可以任意定义新的节点与弧
- 继承性，表达能力强
- 语义网络上的继承推理具有非单调性
- 处理上的复杂性
- 非严格性
- 多重继承冲突问题是重要的研究课题

语义网络

- Use semantic networks to represent:
  - 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.

Methodologies of Knowledge Representation 5

框架, 剧本, 过程法

Frame, Script, Procedure

框架

- 75年由美国人工智能学者明斯基提出
- 认为人们对现实世界的认识都是以一种类似于框架的结构存储在记忆中
- Information retrieval when facing a new situation
  - The information is stored in frames with slots(槽)
  - Some of the slots trigger actions, causing new situations
- 框架是一种描述所论对象属性的数据结构
  - Frames are templates which are to be filled-in in a situation
- 在框架理论中，框架是知识表示的一个基本单位
- 也是一种语义网络
- 继承性是其重要特性
  - Also very similar to objects in OOP

框架

- 框架的结构
  - 框架描述一类物体，由框架名与一些描述物体各个方面的槽(slot)组成
  - 每个槽可分为多个侧面(facet)
  - 每个侧面可有一个或多个值

<框架名>
<槽名1>: <侧面11>(值111, 值112, ...)
<侧面12>(值121, 值122, ...)
<槽名2>: <侧面21>(值211, 值212, ...)
<侧面22>(值221, 值222, ...)
...
约束： 约束条件1，约束条件2…

框架

- 框架的结构

<假冒伪劣商品>
商品名称:
生产厂家:
出售商店:
处　　罚: 处理方式:
处理依据:
处罚时间: 单位（年、月、日）
经办部门:

框架

- 槽（侧面类型）
  - 值
  - Default
  - 继承
  - if-needed

<CHAIR>
Specialization-of:FURNITURE
Number-of-Legs:DEFAULT 4
Style-of-Back:Straight,Cushioned
Number-of-Arms:0,1 or 2

<JOHN’S-CHAIR>
Specialization-of:CHAIR
Style-of-Back:Cushioned
Number-of-Arms:0

框架

- 四种侧面填写方式
  - 由已知情况或物体属性提供
  - 通过默认隐含
  - 由继承获得
  - 对附加过程侧面通过执行附加过程实现
- 框架中的槽与侧面可任意定义

  槽与侧面也可以是另一框架，形成框架网络

框架

- 框架的结构
  - 框架间的联系由槽名指明
  - 常用槽
    - ISA槽

<ATHLETE>
Name:
Age:
Sex: range: (male, female)
default: male

<CHESS PLAYER>
ISA: ATHLETE
brain: excellent

框架

- Subclass 槽
- AKO 槽
- Instance 槽

是AKO槽的逆关系

<ATHLETE>
Instance: <Chess player>,
<football player>,
<basketball player>
Name:
Age:
Sex: range: (male, female)
default: male

框架

- Part-of槽
- Infer槽

指出两个框架间的逻辑推理关系
可表示相应的产生式规则

- Possible-Reason槽

与Infer槽相反

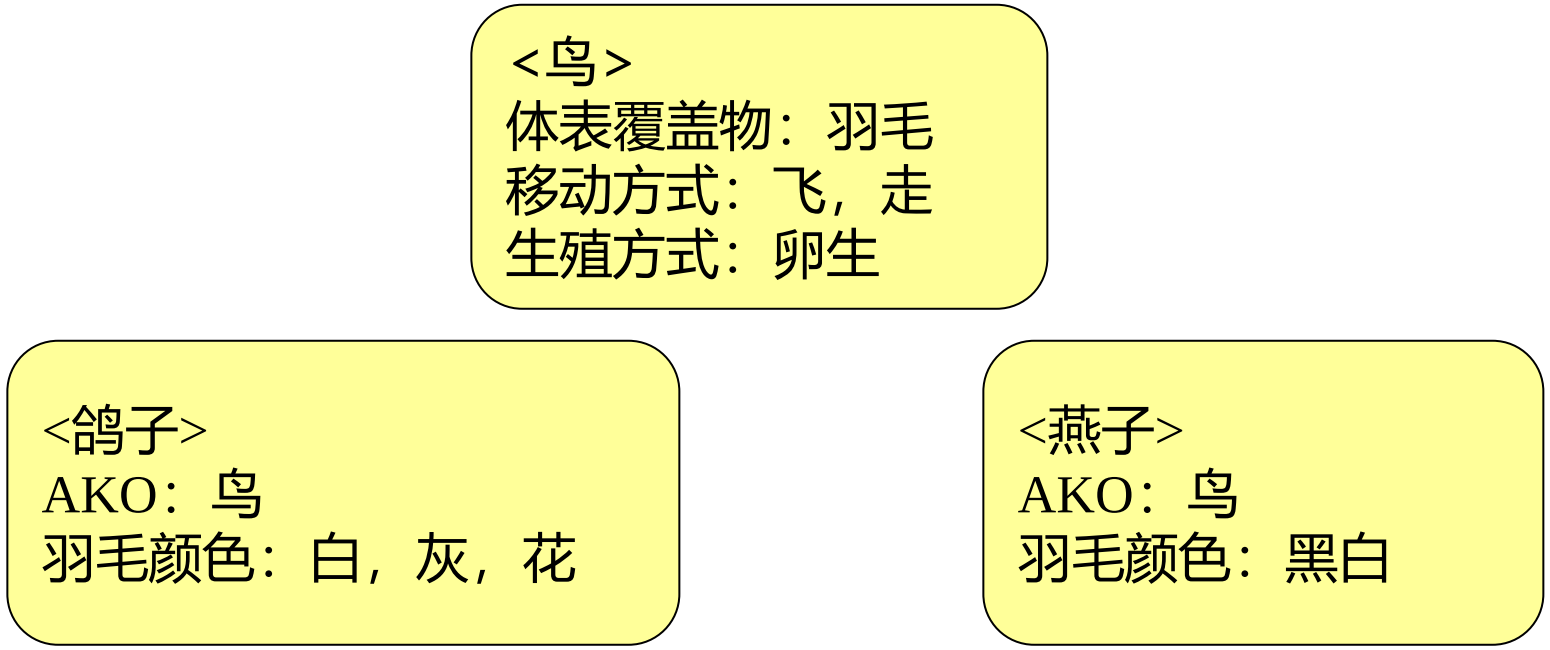
<诊断规则>
症状1：咳嗽
症状2：发烧
症状3：流涕
Infer：<结论>
可信度：0.8

<结论>
病名：感冒
治疗方法：服用感冒胶囊，
一日3次，每次2-3粒
注意事项：多喝开水
Possible-Reason：<诊断规则>

框架

- 对框架及侧面进行合理组织
  - 减少重复性信息
  - 尽量将不同框架描述的相同属性取出构成上层框架

    如：用框架描述鸽子、啄木鸟、布谷鸟、燕子及鹦鹉五种动物



框架

两种推理活动：匹配、填槽

- 匹配
  - 将待解问题用框架表示
  - 匹配通过对相应槽的槽名和值逐个比较实现
  - 比较往往牵涉到其它框架
  - 问题的随机性也使匹配复杂化
- 填槽

  四种填槽方式
  - 查询：中间结果或用户输入
  - 默认
  - 继承
  - 附加过程计算

框架

<师生员工>
姓名：
年龄：
性别：
range:男,女
default:男
健康状况：
range:健康,
一般,差
default:一般
住址：<住址>

<教职工>
AKO：<师生员工>
工作类别：
range:教师,干部,
工人
default:教师
开始工作时间：
截止工作时间：
default:现在
离退休状况：
range:离休,退休
default:退休

<教师>
AKO：<教职工>
部门：单位(系,教研室)
语种：
range:英,法,德,日,俄
default:英语
外语水平：
range:优,良,中,差
default:良
职称：
range:教授,副教授,
讲师,助教
default:讲师
研究方向：

<教师1>
AKO：<教师>
姓名：孙林
年龄：28
健康状况：健康
部门：
计算机系软件教研室
语种：德语
开始工作时间：
1985.9
…

<教师x>
AKO：<教师>
姓名：
年龄：<30
性别：男
健康状况：健康
职称：讲师

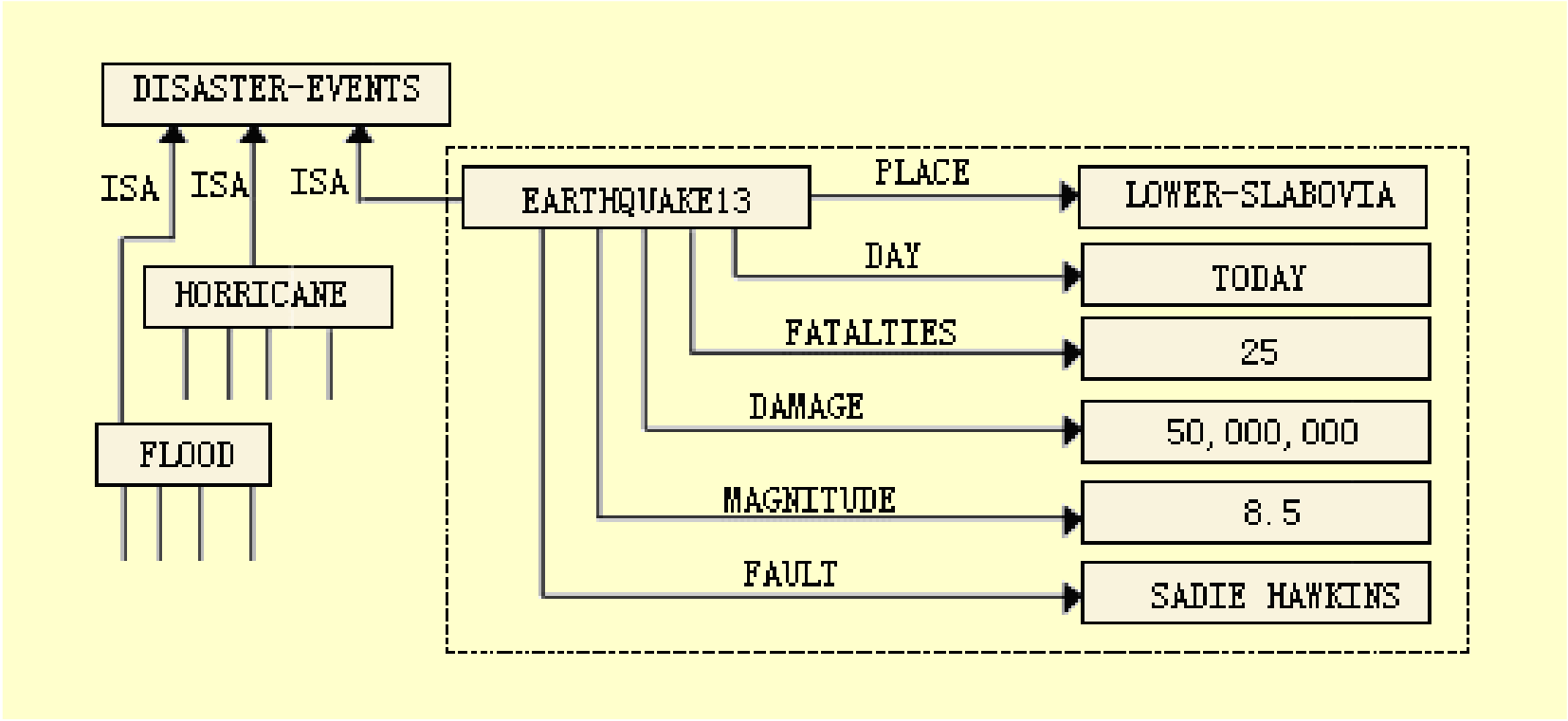
例：在关于师生员工的框架网络种找满足如下条件的教师：
男性，30岁以下，身体健康，讲师

框架

- 框架表示的特点与不足
  - 结构性
  - 继承性
  - 自然性
  - 模块性
  - 没有形成完整的理论体系
  - 缺乏清晰的语义
  - 多重继承的冲突

框架

Example: Today, a magnitude-8.5 earthquake struck the area of Lower-Slovenia, resulting in 25 deaths and 500 million U.S. dollars in property damage. The Lower-Slovenian President said close to the Sa Dihao Kings hit has been a dangerous area over the years.



框架

- XML框架设计与应用
  - XML是Extensible Markup Language的缩写，即可扩展标记语言。它是一种用来创建的标记的标记语言。
  - 1996年，万维网协会（或者叫W3C）开始设计一种可扩展的标记语言，1998年2月，XML1.0成为了W3C的推荐标准。
  - 这种XML语言继承了SGML的规范，Standard Generalized Markup Language (SGML)是一种基于记号文本的语言。XML还保持了对现有的面向SGML系统的向下兼容性。

框架

- XML框架设计与应用：文档组成

XML文档也属于纯文本文件，该文档一般如下四部分组成：

XML文档的声明

XML文档类型定义

XML文档注释

前三部分都是可选的

XML标识及其内容

按照这种文档格式来编写的一个XML文件，如下所示：

```
<?xml version="1.0" encoding="UTF-8"?>
        <!--XML文档注释-->
        <?xml:stylesheet type="text/xsl"
href="stu.xsl"?>
        <!--班级中学生的信息-->
        <class>
                <student>
                        <name>Jone</name>
                        <age>20</age>
                </student>
        </class>
```

框架

- XML框架设计与应用：元素

元素是XML文档的重要组成部分，在XML文档中必须存在元素。
XML文档的元素一般是由标记头、标记末和标记间的字符串数据构成，如下代码所示：

```
<root>
        <a>this is test</a>
</root>
```

元素a的值

元素a的元素名或标签名

XML文档中的第一个元素被称为根元素，在任何一个XML文档中有且只有一个元素被称为根元素。其余所有的元素都是子元素，子元素必须正确的嵌套在根元素中。
标记间的字符串数据就是该元素的值，在XML中，如果元素的值中存在空格，那么这些空格将按原样解析出来。

框架

- XML框架设计与应用：属性

属性是用来修饰某个元素的，如：

```
<root>
        <a attribute="aa">this is test</a>
</root>
```

关于元素的属性需注意如下几个问题：

属性的值必须用引号括起来，如： attribute1=“aa” 或attribute3=‘aa'；

元素的属性以名和值成对出现；

用来修饰同一个元素的属性的属性名不能相同；

属性值不能包含“&”、“’”、“”、“<”等字符。

框架

- XML框架设计与应用：实例
  - 编写某班级的学生信息，要求符合XML语法的规范。
  - 学生信息包括姓名、年龄、电子邮箱、身高、电话、单位等；单位又包含地址、邮编等信息，每个学生都要有一个“编号”属性作为标识。例如，姓名为“张三”的学生有两个电子邮箱，每个学生有电话或手机。

框架

- XML框架设计与应用

```
<?xml version="1.0" encoding="UTF-8"?>
        <!--以下是某班级的学生信息，每个学生有姓名、年龄、电子邮箱、
        身高、电话、单位等信息，单位又有地址、邮编等信息，每个学生
        都要有一个“编号”属性作为标识。名为“张三”的学生有两个电子
        邮箱，每个学生要有电话或手机。-->
        <班级>
                <学生 编号="A0001">
                        <姓名>张三</姓名>
                        <年龄>23</年龄>
                        <电子邮箱>zhangsan@163.com</电子邮箱>
                        <电子邮箱>zhangsan@yahoo.com</电子邮箱>
                        <身高>179.5</身高>
                        <电话>686868</电话>
                        <单位>
                                <地址>上海</地址>
                                <邮编>100002</邮编>
                        </单位>
                </学生>
```

框架

- XML框架设计与应用

```
<学生 编号="A0003">
        <姓名>李四</姓名>
        <年龄>24</年龄>
        <电子邮箱>lisi@263.com</电子邮箱>
        <身高>168.0</身高>
        <手机>135013562554</手机>
        <单位>
            <地址>北京</地址>
        </单位>
</学生>
<学生 编号="A0002">
        <姓名>王五</姓名>
        <年龄>21</年龄>
        <电子邮箱>wangwu@163.com</电子邮箱>
        <身高>179.5</身高>
        <电话>686868</电话>
        <单位>XXXX公司</单位>
</学生>
</班级>
```

脚本(Script)

- 依据概念依赖理论提出的
- 表示特定领域内事件的发生序列
- 脚本的组成
  - 进入条件
  - 角色
  - 道具
  - 场景
  - 结局

脚本(Script)

脚本：餐厅
进入条件：顾客饿了，需要进餐；顾客有钱
角色：顾客、服务员、厨师、老板
道具：食品、桌子、菜单、钱
场景：
　　第一场：进入餐厅
　　　　顾客走入餐厅
　　　　顾客寻找桌子
　　　　在桌旁坐下
　　第二场：点菜
　　　　服务员给顾客菜单
　　　　顾客点菜
　　　　顾客把菜单给服务员
　　　　服务员告诉厨师所要食品
　　　　厨师做菜

脚本(Script)

第三场：上菜进餐

厨师把菜给服务员

服务员把菜送给顾客

顾客吃菜

第四场：顾客离开

顾客告诉服务员结帐

服务员拿来帐单

顾客付钱

顾客离开餐厅

结局：顾客吃了饭；

顾客花了钱；

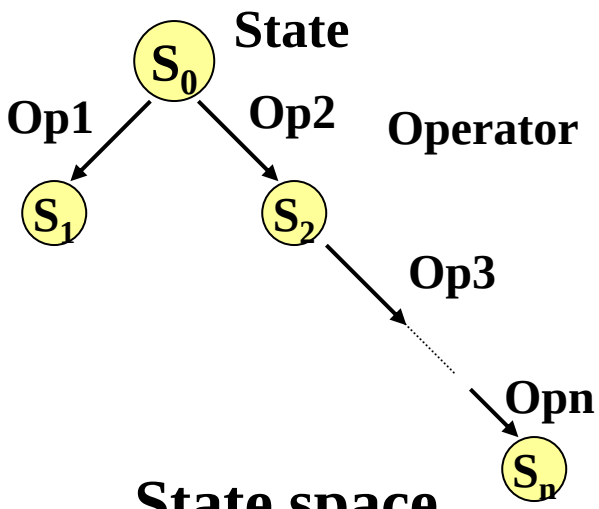
老板挣了钱；

餐厅食品少了

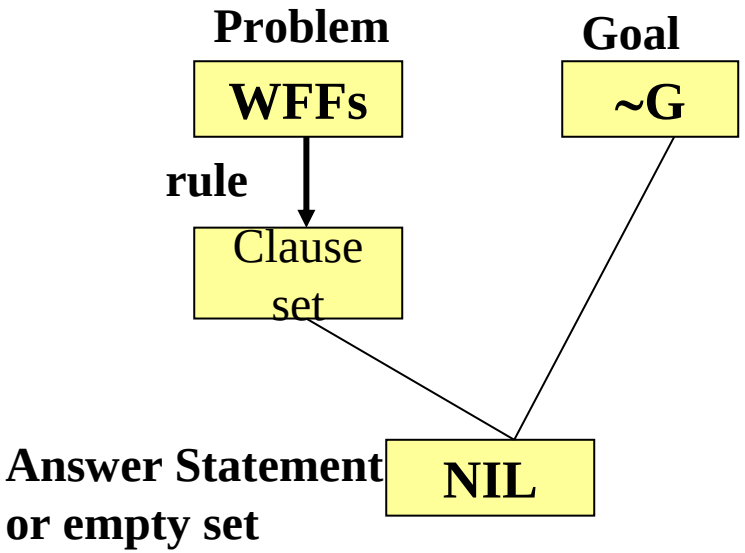
- 在脚本适用于给定事件时，可通过它预测没有明显提及的事实

过程

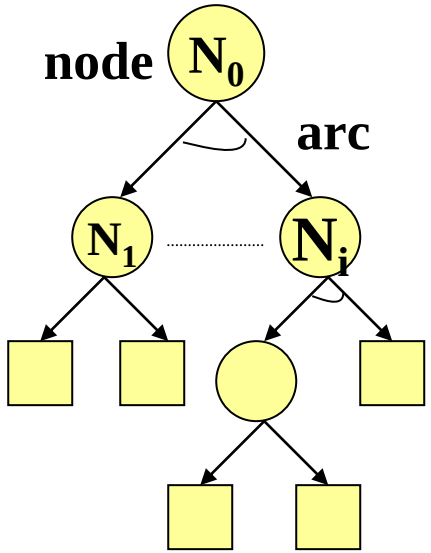
- 语义网络、框架等知识表示方法，均是对知识和事实的一种静止的表示方法，是知识的一种显式表达形式。而对于如何使用这些知识，则通过控制策略来决定。
- 和知识的陈述式表示相对应的是知识的过程式表示。所以过程式表示就是将有关某一问题领域的知识，连同如何使用这些知识的方法，均隐式地表达为一个求解问题的过程。它所给出的是事物的一些客观规律，表达的是如何求解问题。知识的描述形式就是程序，所有信息均隐含在程序之中。从程序求解问题的效率上来说，过程式表达要比陈述式表达高得多。但因其只是均隐含在程序中，因而难于添加新知识和扩充功能，适用范围较窄。



State space



Predicate logic



Primitive Problems

Problem Reduction

Semantic network

Method	Problem	Operation	Goal	Solution
State Space	state	operator	Goal state	Solution path
Problem Reduction	node	arc	terminal nodes	Solution tree
Predicate Logic	wffs	rule	root of tree	answer statement
Semantic Network	node	link	network	network

Relations among four approaches of representation

Q&A

THANKS!