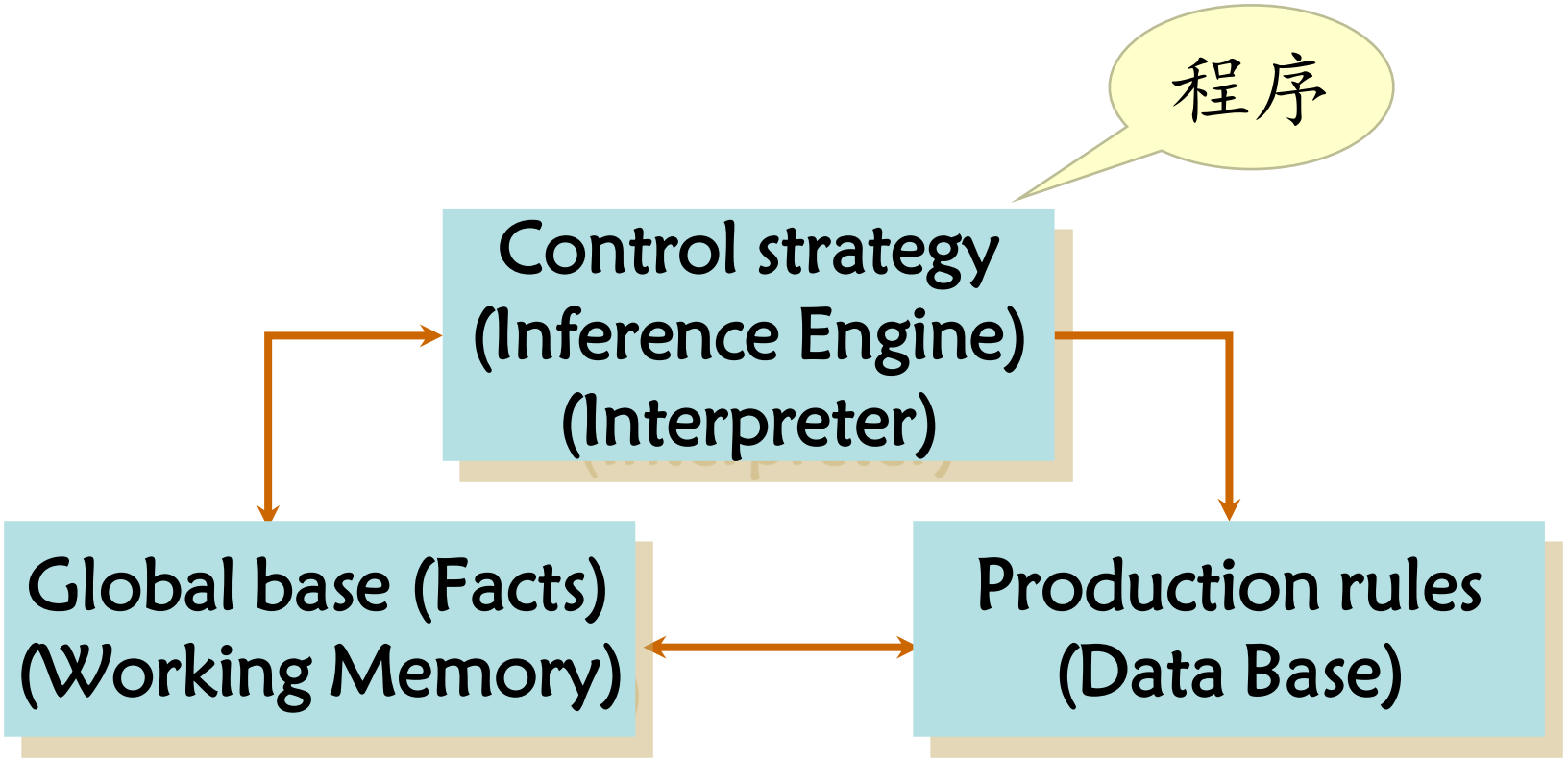
Outline

- What is reasoning?
- Resolution（消解）
- *Production-rule System（产生式系统）*

产生式系统 Production-rule System

- **Definition：**
  - Also called Rule-based System.
  - Proposed by E.Post in 1943. DENDRAL
  - Describes several different things that based on a same basic concept.
- **Essential：** Knowledge separated 2 parts
  - facts represented static knowledge, exp. object, event and relation between them;
  - production rules represented inference process and action.

产生式系统的组成
Architecture of Production-rule System



- 总数据库
  - 又称综合数据库、上下文、黑板等
  - 存放求解过程中各种当前信息的数据，如：问题的初始状态、事实或证据、中间推理结论和结果等。
- 产生式规则（规则库）
  - 存放于求解问题相关的某个领域知识的规则集合
  - 完整性、一致性、准确性、灵活性和合理性
- 控制策略（推理机）
  - 由一组程序组成，用来控制产生式系统的运行

从选择规则到执行

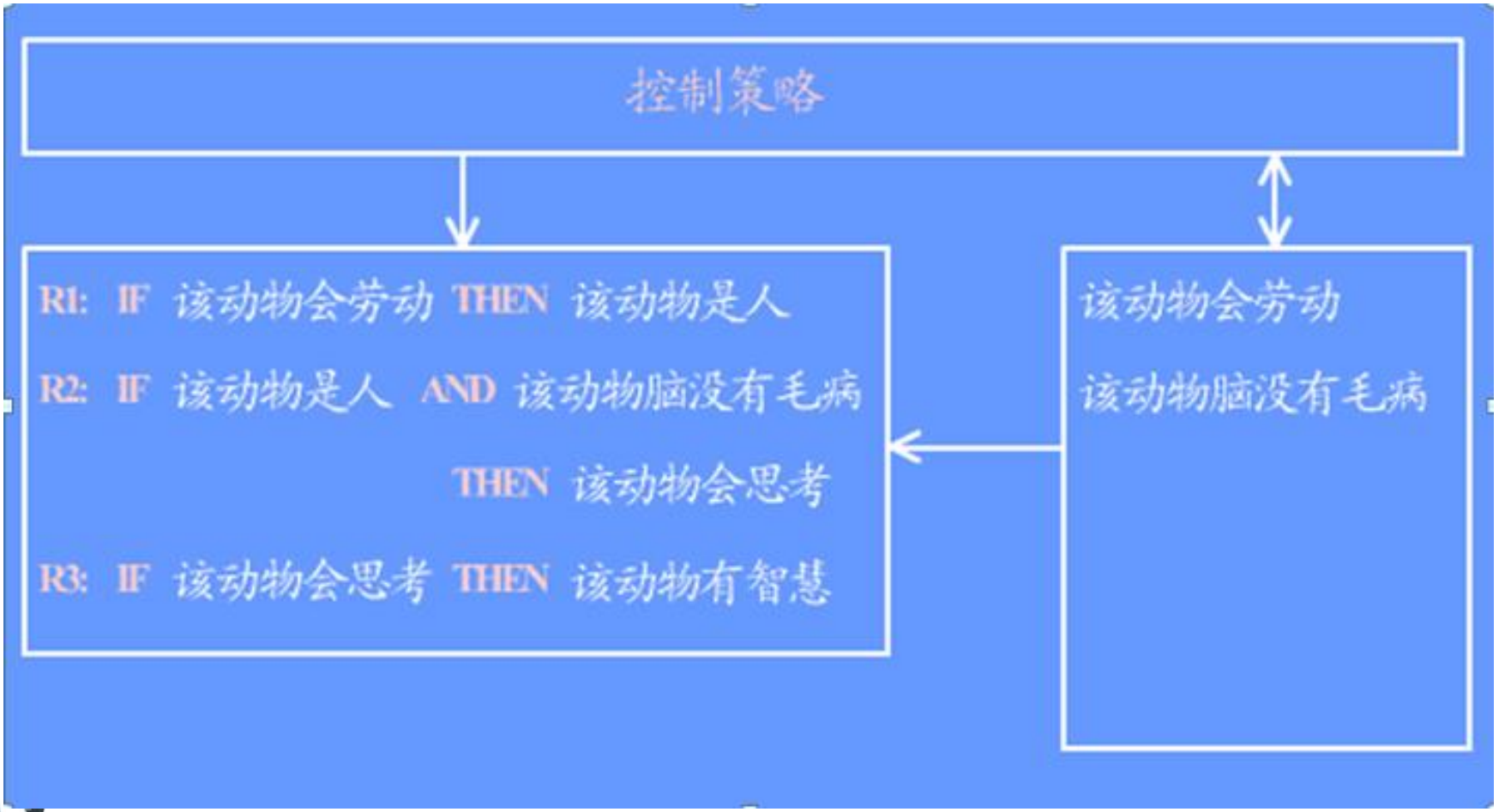
- Matching
  - Current database is matched with rule condition.
- Conflict resolution
  - When more than one rule matched with current database, it should decide which rule is used firstly, which is called conflict resolution.
- Operation
  - Operation means execution of the rule's operation parts

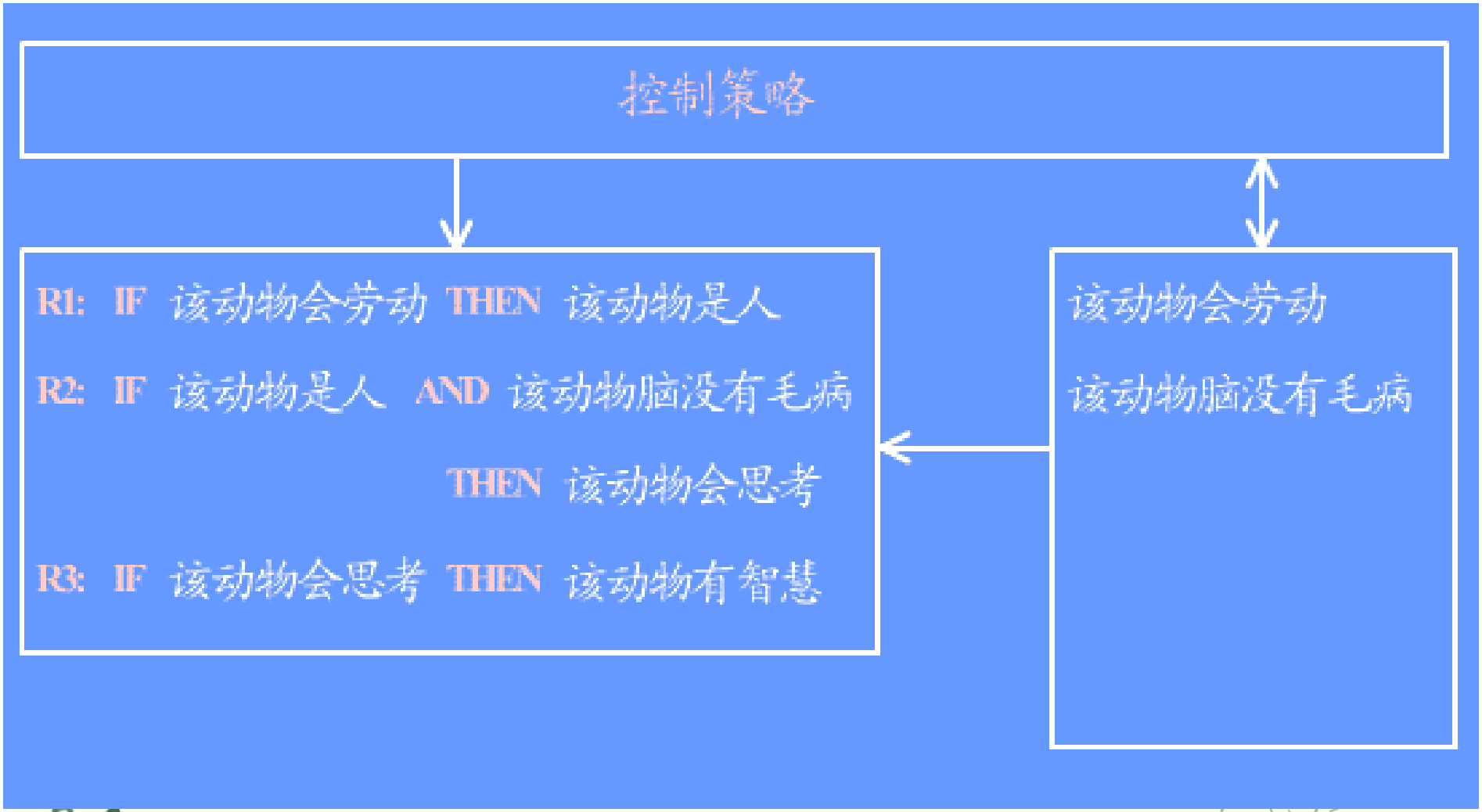
产生式系统的推理

Inference of Production-rule System

- Forward Inference
- Backward Inference
- Bidirectional Inference

例：


控制策略
R1: IF 该动物会劳动 THEN 该动物是人
R2: IF 该动物是人 AND 该动物脑没有毛病
THEN 该动物会思考
R3: IF 该动物会思考 THEN 该动物有智慧
该动物会劳动
该动物脑没有毛病


控制策略
R1: IF 该动物会劳动 THEN 该动物是人
R2: IF 该动物是人 AND 该动物脑没有毛病
THEN 该动物会思考
R3: IF 该动物会思考 THEN 该动物有智慧
该动物会劳动
该动物脑没有毛病

An example of production system

- You want a program that can answer questions and make inferences about food items
- Like:
  - What is purple and perishable?
  - What is packed in small containers?
  - What is green and weighs 5 kg?

A simple production rule system making inferences about food

WORKING MEMORY (WM)
Initially WM = (green, weighs-5-kg)

RULE BASE

R1: **IF** green **THEN** produce
R2: **IF** packed-in-small-container **THEN** delicacy
R3: **IF** [refrigerated **OR** produce] **THEN** perishable
R4: **IF** [weighs-5-kg **AND** inexpensive **AND NOT** perishable] **THEN** staple
R5: **IF** [weighs-5-kg **AND** produce] **THEN** watermelon

INTERPRETER

1. Find all productions whose condition parts are true
2. Deactivate productions that would add a duplicate symbol
3. Execute the lowest numbered production (or quit)
4. Repeat until there is no rule to execute

First cycle of execution

WORKING MEMORY

WM = (green, weighs-5-kg)

CYCLE 1

1. Productions whose condition parts are true: ***R1***
2. No production would add duplicate symbol
3. Execute ***R1***.

   This gives: WM = (***produce,*** green, weighs-5-kg)

RULE BASE

R1: **IF** green **THEN** produce
R2: **IF** packed-in-small-container **THEN** delicacy
R3: **IF** [refrigerated **OR** produce] **THEN** perishable
R4: **IF** [weighs-5-kg **AND** inexpensive **AND NOT** perishable] **THEN** staple
R5: **IF** [weighs-5-kg **AND** produce] **THEN** watermelon

INTERPRETER

1. Find all productions whose condition parts are true
2. Deactivate productions that would add a duplicate symbol
3. Execute the lowest numbered production (or quit)
4. Repeat

Second cycle of execution

WORKING MEMORY

WM = (produce, green, weighs-5-kg)

CYCLE 2

1. Productions whose condition parts are true: ***R1, R3, R5***
2. Production R1 would add duplicate symbol, so ***deactivate R1***
3. Execute ***R3*** because it is the lowest numbered production.

   This gives: WM = (***perishable,*** produce, green, weighs-5-kg)

RULE BASE

R1: **IF** green **THEN** produce
R2: **IF** packed-in-small-container **THEN** delicacy
R3: **IF** [refrigerated **OR** produce] **THEN** perishable
R4: **IF** [weighs-5-kg **AND** inexpensive **AND NOT** perishable] **THEN** staple
R5: **IF** [weighs-5-kg **AND** produce] **THEN** watermelon

INTERPRETER

1. Find all productions whose condition parts are true
2. Deactivate productions that would add a duplicate symbol
3. Execute the lowest numbered production (or quit)
4. Repeat

Third cycle of execution

WORKING MEMORY

WM = (perishable, produce, green, weighs-5-kg)

CYCLE 3

1. Productions whose condition parts are true: ***R1, R3, R5***
2. Productions ***R1, R3*** would add duplicate symbol, so deactivate them
3. Execute ***R5***.

This gives: WM = (***watermelon***, perishable, produce, green, weighs-5-kg)

RULE BASE

R1: **IF** green **THEN** produce
R2: **IF** packed-in-small-container **THEN** delicacy
R3: **IF** [refrigerated **OR** produce] **THEN** perishable
R4: **IF** [weighs-5-kg **AND** inexpensive **AND NOT** perishable] **THEN** staple
R5: **IF** [weighs-5-kg **AND** produce] **THEN** watermelon

INTERPRETER

1. Find all productions whose condition parts are true
2. Deactivate productions that would add a duplicate symbol
3. Execute the lowest numbered production (or quit)
4. Repeat

Fourth cycle of execution

WORKING MEMORY

WM = (watermelon, perishable, produce, green, weighs-5-kg)

CYCLE 4

1. Productions whose condition parts are true: ***R1, R3, R5***
2. Productions ***R1, R3, R5*** would add duplicate symbol, so ***deactivate them***
3. ***Quit.***

What are the conclusions?

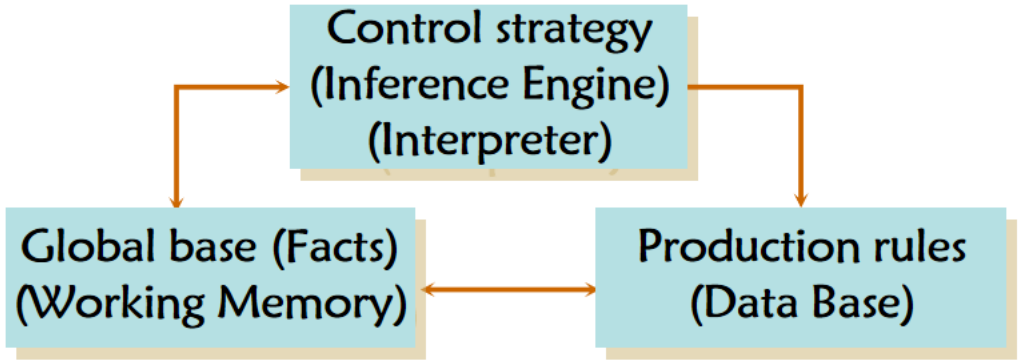
RULE BASE

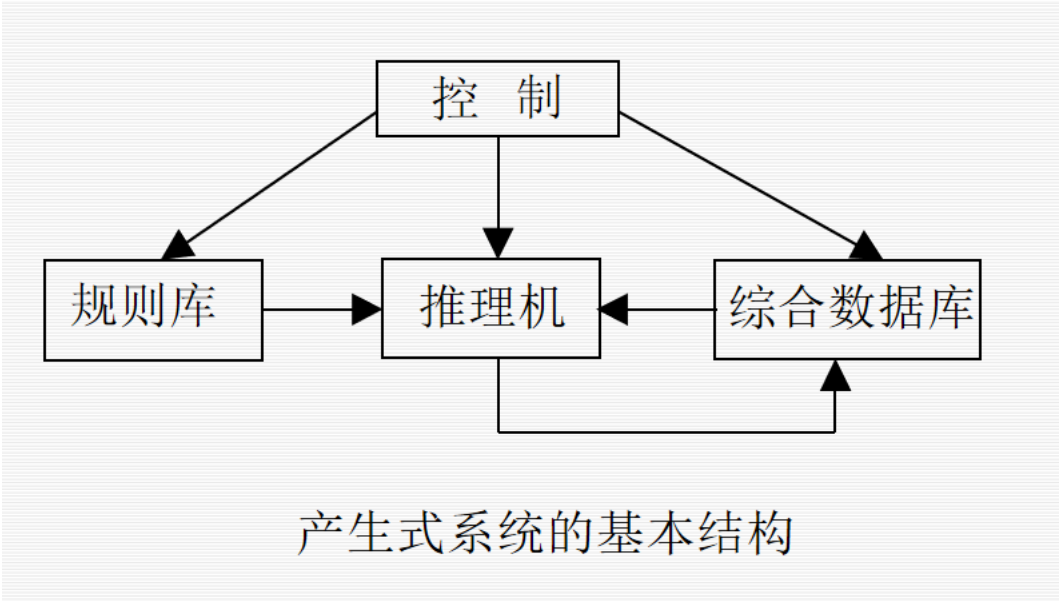
R1: **IF** green **THEN** produce
R2: **IF** packed-in-small-container **THEN** delicacy
R3: **IF** [refrigerated **OR** produce] **THEN** perishable
R4: **IF** [weighs-5-kg **AND** inexpensive **AND NOT** perishable] **THEN** staple
R5: **IF** [weighs-5-kg **AND** produce] **THEN** watermelon

INTERPRETER

1. Find all productions whose condition parts are true
2. Deactivate productions that would add a duplicate symbol
3. Execute the lowest numbered production (or quit)
4. Repeat

小结-产生式系统





产生式系统的基本结构

产生式系统

1. 规则库

- **规则库**：用于描述相应领域内知识的产生式集合。

2. 综合数据库 （动态改变的）

- **综合数据库**(事实库、上下文、黑板等)：一个用于存放问题求解过程中各种当前信息的数据结构。

3. 控制系统

- **控制系统**（推理机构）：由一组程序组成，负责整个产生式系统的运行，实现对问题的求解。

产生式系统

3．控制系统（续）

控制系统要做以下几项工作：

（1）从规则库中选择与综合数据库中的已知事实进行**匹配**。

（2）匹配成功的规则可能不止一条，进行**冲突消解**。

（3）执行某一规则时，如果其右部是一个或多个结论，则把这些结论加入到综合数据库中；如果其右部是一个或多个操作，则执行这些操作。

（4）对于不确定性知识，在执行每一条规则时还要按一定的算法计算结论的不确定性。

（5）检查综合数据库中是否包含了最终结论，决定是否停止系统的运行。

产生式系统

1. 产生式表示法的优点

（1）自然性

（2）模块性

（3）有效性

（4）清晰性

2. 产生式表示法的缺点

（1）效率不高（若规则多更不高）

（2）不能表达结构性知识

（3）不适合求解复杂系统

3. 适合产生式表示的知识

（1）领域知识间关系不密切，不存在结构关系。

（2）经验性及不确定性的知识，且相关领域中对这些知识没有严格、统一的理论。

（3）领域问题的求解过程可被表示为一系列相对独立的操作，且每个操作可被表示为一条或多条产生式规则。

本章小结-确定性推理

- 搜索问题及求解
  - 盲目搜索、启发式搜索
  - 博弈搜索求解
- 推理
  - 用谓词公式、推理规则、转换合一、消解原理求解（或证明）问题
  - 产生式系统的组成、推理过程

3-1- 推理

- 我们从几个维度来考察了人类智能推理的特征。
  - 推理的归纳和演绎；
  - 推理的正向和逆向；
  - 推理的确定和不确定。

- 可见，计算机如果要实现自动推理，最可行的方法就是采用反向、演绎的方法。

- 人工智能要模拟人的推理，也需要从上述几个角度来讨论可行性。

- 比如，第一个维度中，归纳和演绎，两种方法中，归纳方法做推理很难，但在规律发现、学习方面有优势。因此归纳方法适合于学习规律，而不适用于推理，实际上，机器学习可以看作一种归纳。
- 相比而言，演绎更适合描述人类推理。

- 其次，从推理的方向来看，
  - 正向推理，如果知识库中的规则和知识很多，则每推理一步都会产生大量的无用推理，因此不适合计算机。
  - 反向推理则可以在每一步都有具体的目标，目标单一，适合计算机处理。

- 最后，从确定性考虑，
  - 人工智能早期只涉及了确定性推理，如定理证明。
  - 随着产生式系统的发展，逐渐具备了不确定性推理的能力。

3-1- 归结原理

- 归结过程前，9个步骤求取子句集
- 归结原理用于自动推理
- 反证法
- 其前提假设：

– （1）证明一个命题 P 成立，等价于证明其逆命题 ¬P 不成立。
– （2）一个已知为真的事实集，其内部不包含矛盾。
– （3）如果将一个命题加入事实集，并且导致事实集出现矛盾，则原命题不成立。
– （4）最常见的矛盾形式，是 ：$P \wedge \neg P \Rightarrow \square$ 一旦事实集中出现空符号□，则认为事实集中存在矛盾。

3-2- 搜索问题和求解

1、盲目搜索方法
open表、closed表
宽度优先搜索：其算法、优缺点、举例
深度优先搜索：其算法、优缺点、举例
宽度优先搜索 vs. 深度优先搜索
等代价搜索

2、启发式搜索
什么情形下用启发式策略
启发式信息的表示：估计函数、规则
如何构造启发式函数？举例【8数码问题】

- 启发式函数 f1= 数字错放位置的个数，f1=0，则达到目标。
- 当f1值相同时f2决定走步：f2 = 所有数字当前位置以最短路径走到正确位置的步数之和。

如何构造启发式函数？举例【八皇后问题】
爬山法
最佳优先搜索
A*算法

3-3-博弈搜索

1）博弈特征

“双人有限零和顺序游戏”, 状态空间搜索，动态，预估对方

2）极大极小算法：宽度优先搜索

- MAX方、MIN方；MINMAX的基本思想
- 过程：宽度优先法生成规定深度的全部博弈树，然后对其所有端节点计算其静态估计函数值;从底向上逐级求非终结点的倒推估计值，至初始节点的倒推值f(s)为止，标记走步。

3）*α-β*剪枝算法：有限深度优先搜索

- 极大层的这个下界值为α、极小层的这个上界值为β
- *α*剪枝：*α*（先辈层）≥*β*（后继层），则可中止该极小值层中这个节点以下的搜索
- *β*剪枝： *α*（后继层）≥*β*（先辈层），则可中止该极大值层中这个节点以下的搜索

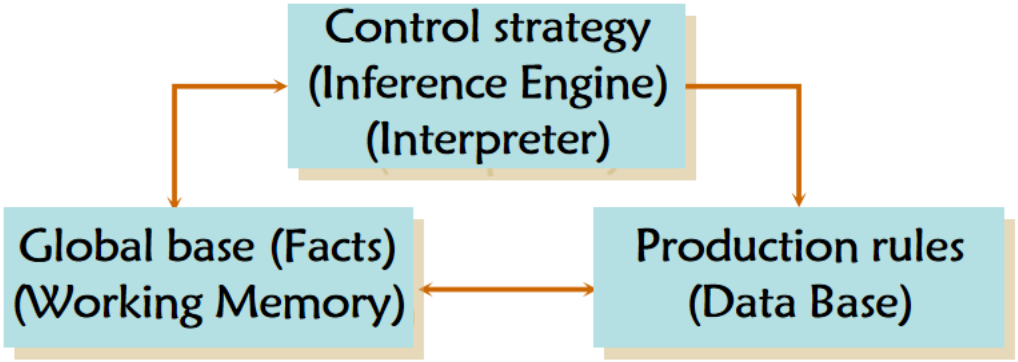
4）蒙特卡洛树算法

ALPHAGO、五子棋类似，只是规模不同；强化学习做决策、蒙特卡洛做搜索

5) 算力

3-4 产生式系统

- 产生式系统的3个组成部分：产生式规则、总数据库、控制策略



- 产生式系统的各部分功能和其工作方式
- 控制策略的作用是说明如何选用规则，如何应用规则；从选择规则到执行操作分3步：匹配、冲突解决、操作

Task1：

课后习题3

3-1

3-2

3-15

Task2：

1） 完成如下练习5.1和5.2

5.1 对于如图 5-7 所示的博弈树，假若 A 在极大值层，它该选什么样的走步？

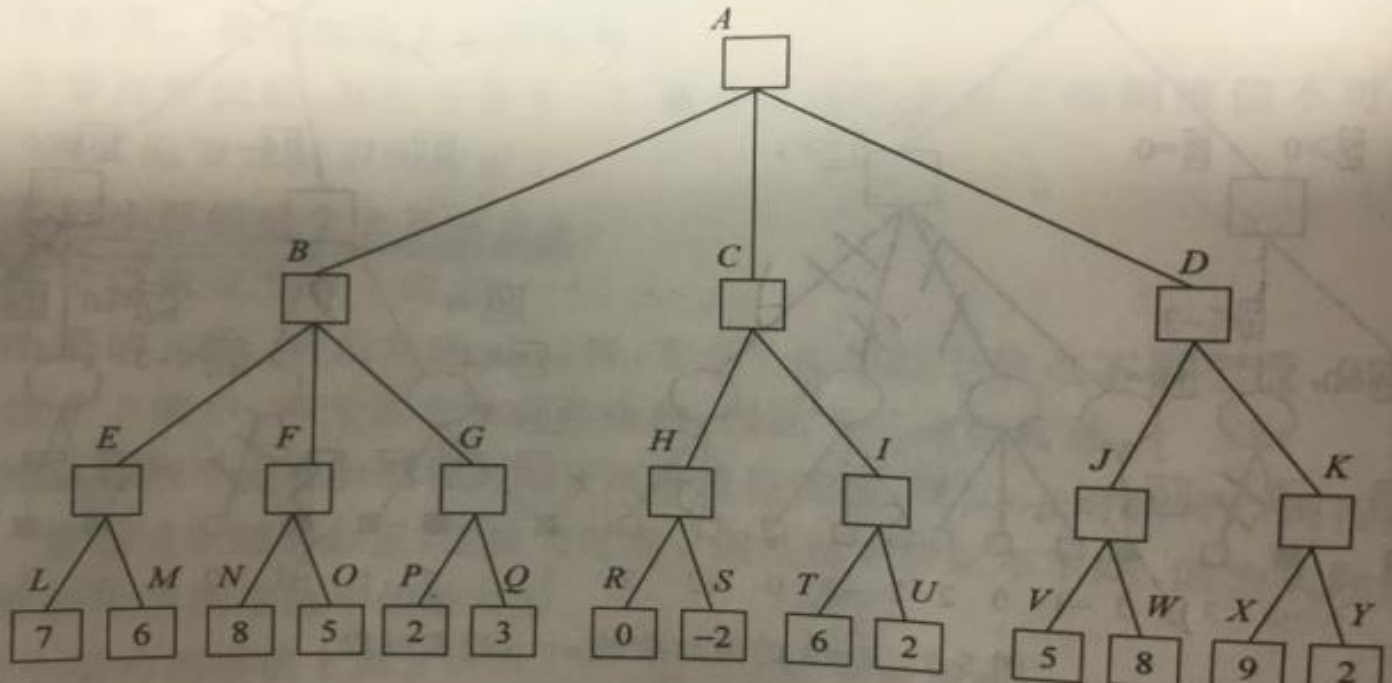


图 5-7 习题 5.1 用图

5.2 在上题的博弈树中，用剪枝过程需要检查哪些节点？

2）“20Q游戏” www.20q.net

3）复习前三章，参照教材每章“小结”和PPT