产生式系统

张文生　研究员 首席教授

中国科学院自动化研究所
中科院大学人工智能学院

讲课内容

一、产生式系统

二、推理

三、控制策略

四、两类产生式系统

五、基于规则的演绎系统

讲课内容

一、产生式系统

二、推理

三、控制策略

四、两类产生式系统

五、基于规则的演绎系统

产生式

- 一种知识表示方法，常用来表示有因果关系的知识
- 形式:
  - 条件 → 行动
  - 前提 → 结论
  - “if……then……”
- 例如:
  - 烫手→缩手
  - 下雨→地面湿
  - 下雨∧甲未打伞→甲被淋湿
  - 所有人会死∧甲是人→甲会死

- →左边表示条件，右边表示结论
- 一般可以写成 $A_1 \wedge A_2 \wedge \ldots\ldots A_n \rightarrow B$的形式；
  - 下雨∧甲未打伞→甲被淋湿

- **产生式系统**
  - **把一组产生式放在一起，让它们互相配合，协同作用，一个产生式的结论可以供另一个产生式作为前提使用，以这种方式求解问题的系统称为产生式系统。**

- **A→B, B→C, C→D : A→D ???**

产生式的历史

- 1943年，美国数学家Post设计的产生式系统，称为Post系统。
- 目的是构造一种形式化的计算工具。
- 证明它和图灵机具有相同的计算能力。

产生式系统的构成

- 产生式系统的构成
  - 一组产生式规则
  - 综合数据库
  - 控制机制

产生式规则

- 例子
  - 下雨→地面湿
  - 下雨∧甲未打伞→甲被淋湿
  - 所有人会死∧甲是人→甲会死

综合数据库

- 存放已知的事实和推导出的事实;
- 综合数据库 和数据库 不同:
  - 数据库：强调数据的管理（存取、增、删、改等）
  - 综合数据库：产生式系统----抽象的概念
    - 只是说明数据在此存放，和物理实现没关系。
    - 数据是广义的，可以是常量、变量、谓词、图像等。
    - 数据结构：符号串、向量、集合、数组、树、表格、文件等。

控制机制

- 控制机制完成的工作有：
  - 匹配规则条件部分
  - 多于一条规则匹配成功时，选择哪条规则执行
  - 如何将匹配规则的结论部分放入综合数据库
  - 决定系统何时终止

产生式系统的运行过程

- 建立产生式规则
- 将已知的事实放入综合数据库
- 考察每一条产生式规则，如果条件部分和综合数据库中的数据匹配，则规则的结论放入综合数据库

算法

- 令DATA为综合数据库

1．DATA←初始化；

2．如果满足终止条件，终止。

否则：

a) 选择一个可应用于DATA之上的规则R；

b) DATA←R应用于DATA之上产生的结果；

例1

- **八数码游戏(eight puzzle)**

2	3	7
	5	1
4	8	6

1	2	3
8		4
7	6	5

- 游戏说明：
  - 一个棋盘有9个方格，放了8个数（1-8）；
  - 初始时，8个数随机放置；
  - 数字移动规则：空格周围的数字可移动到空格中；
  - 如果通过移动数字，达到一个目标状态，则游戏成功结束；
  - 求一个走步序列；

- 问题：怎样用一个产生式系统描述并解决上述问题？

- 产生式系统的描述:

  - 综合数据库：存放棋盘的状态。
    - 棋盘的状态：8个数字在棋盘上的位置分布;
    - 每走一步，状态就会发生变化;
    - 存放棋盘的当前状态;

  - 规则：规则是数字移动的方法。
    - 空格的移动:
    - 如果空格左边有数字，则将左边的数字移到空格上;
    - 如果空格右边有数字，则将右边的数字移到空格上;
    - 如果空格上边有数字，则将上边的数字移到空格上;
    - 如果空格下边有数字，则将下边的数字移到空格上;

– 控制机制：

- 用当前数据库与规则左半部分进行匹配，确定可执行的规则集；
- 从中选择一条规则执行，用规则的右半部分代替数据库中的状态；
- 如果当前数据库中的状态与目标状态相同，则终止；

例2

- **问题：设字符转换规则**

  A∧B→C

  A∧C→D

  B∧C→G

  B∧E→F

  D→E

  已知：A，B

  求：F

一、综合数据库

{x}，其中x为字符。

二、规则集

1. IF A∧B THEN C
2. IF A∧C THEN D
3. IF B∧C THEN G
4. IF B∧E THEN F
5. IF D THEN E

三、控制策略

顺序排队

四、初始条件

{A，B}

五、结束条件

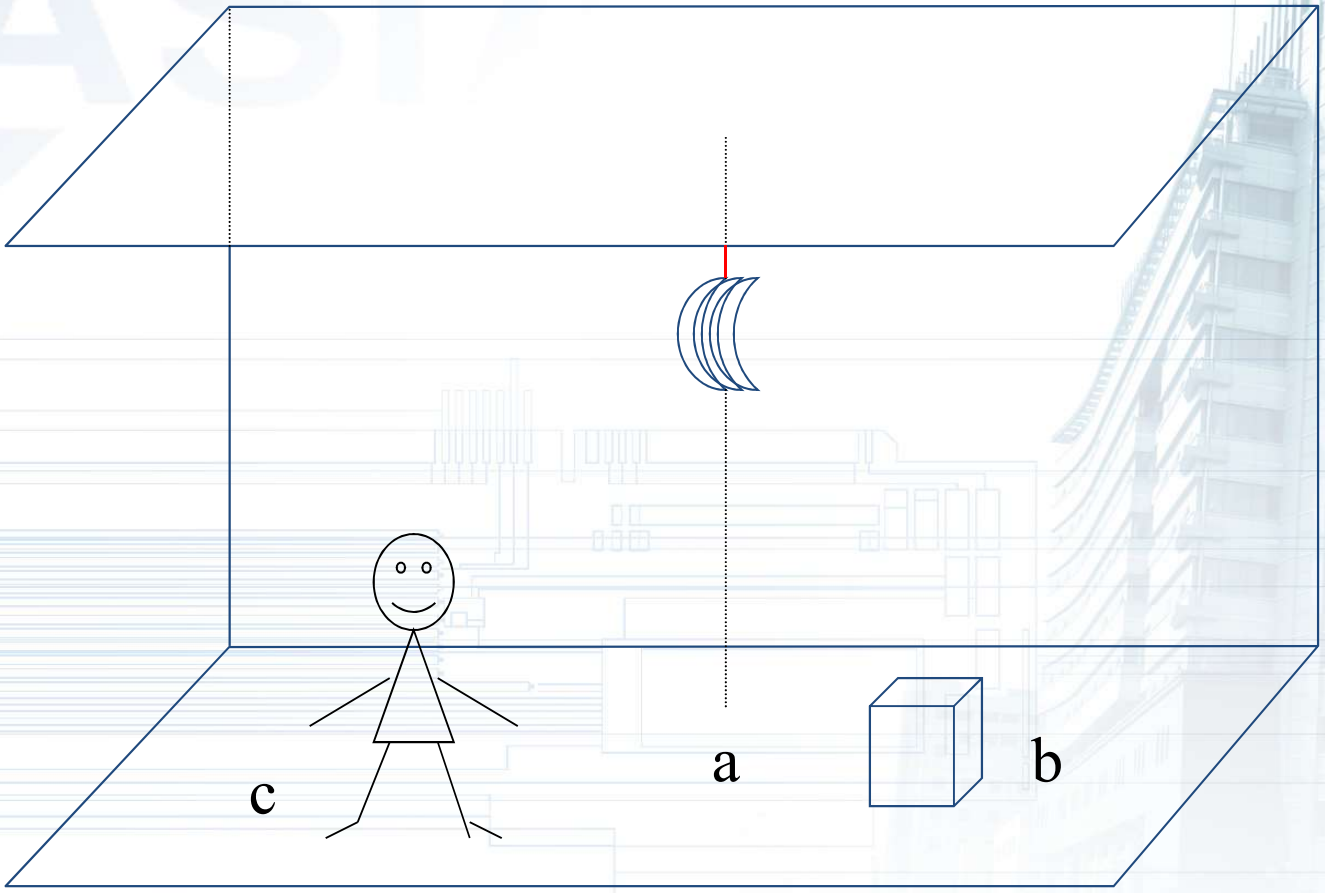
F∈{x}

求解过程

数据库	可触发规则	被触发规则
A，B	（1）	（1）
A，B，C	（2） （3）	（2）
A，B，C，D	（3） （5）	（3）
A，B，C，D，G	（5）	（5）
A，B，C，D，G，E	（4）	（4）
A，B，C，D，G，E，F		

1. IF A∧B THEN C
2. IF A∧C THEN D
3. IF B∧C THEN G
4. IF B∧E THEN F
5. IF D THEN E

例3: 猴子摘香蕉问题



- **综合数据库**

  (M, B, Box, On, H)

M：猴子的位置

B：香蕉的位置

Box：箱子的位置

On=0：猴子在地板上

On=1：猴子在箱子上

H=0：猴子没有抓到香蕉

H=1：猴子抓到了香蕉

- **初始综合数据库**

  (c, a, b, 0, 0)

- **结束综合数据库**

  (x1, x2, x3, x4, 1)

  其中: x1 ~ x4为变量。

(M, B, Box, On, H)

- 规则集

r1: IF (x, y, z, 0, 0) THEN (w, y, z, 0, 0)

r2: IF (x, y, x, 0, 0) THEN (z, y, z, 0, 0)

r3: IF (x, y, x, 0, 0) THEN (x, y, x, 1, 0)

r4: IF (x, y, x, 1, 0) THEN (x, y, x, 0, 0)

r5: IF (x, x, x, 1, 0) THEN (x, x, x, 1, 1)

其中: x, y, z, w为变量

产生式系统的特点

- 规则的表示**形式固定**：
  - 规则分为左半部分和右半部分
  - 左半部分是条件，右半部分是结论

- 知识**模块化**：
  - 知识元是产生式规则的条件中的独立部分，A_i
  - 所有的规则或数据库中的数据都是由知识元构成

- 产生式之间的相互影响是间接的
  - 产生式之间的作用通过综合数据库的变化完成，因此是数据驱动的
  - 易扩展：规则的添加和删除较为自由，因为没有相互作用
  - 添加规则不能造成矛盾；A→B，A→~B

- **可解释性:**
  - 天下雨→张三去医院
    - 天下雨→地滑;
    - 张三不注意安全;
    - 不注意安全的人∧地滑→此人会摔跤;
    - 人摔跤→人骨折;
    - 人骨折→人去医院;

产生式的知识元

- 知识元是常量字符串
  - 完全匹配
    - lecturer∧book→professor
  - 部分匹配
    - 当某条规则的前提中的知识元与综合数据库中的知识元的子串相同，就匹配成功；
    - 匹配成功后，用规则的结论替换综合数据库中的知识元中的那个子串。
    - 置换系统

- 部分匹配的例子:
  - 产生式规则
    - aa→a
    - bb→b
    - ba→ab
    - a→A
    - b→B
  - 将任意由a、b组成的字符串变为AB:
    - abab→aabb→abb→ab→Ab→AB

- 知识元含有谓词
  - ill(x) ∧ not_work(x) → go_to_hospital(x)
  - 谓词描述的知识更为广泛;
  - 将条件和结论中的同一变量同时替换;

- 知识元是**析取式**：
  - 匹配**是非确定性**匹配；
    - (腰背冷痛∨畏寒∨肢冷/1) ∧(腹胀∨浮肿∨嗜睡……/2)…→脾肾阳虚
  - **加权**
    - ((腰背冷痛（0.8）∨畏寒（0.5）∨肢冷（0.2）) ∧加权之和大于等于0.7)…→脾肾阳虚
  - **一种简化的方式是对规则加权**
    - (腰背冷痛∨畏寒∨肢冷) ∧(腹胀∨浮肿∨嗜睡……)…→脾肾阳虚
    - 可信度：0.7

- **常用表示方式:**
  - **（对象，属性，值）**
  - **（细菌，染色斑，革兰式阳性）∧（细菌，形状，球状）∧（细菌，生长结构，链形）→（细菌，类别，链球菌）**
    - **MYCIN系统**

讲课内容

一、产生式系统

二、推理

三、控制策略

四、两类产生式系统

五、基于规则的演绎系统

推理

- 从某些已知事实依照推理规则推出另外一些结论的过程

- 系统状态的转换
- 向前推理(正向推理)
- 向后推理(反向推理)

正向推理

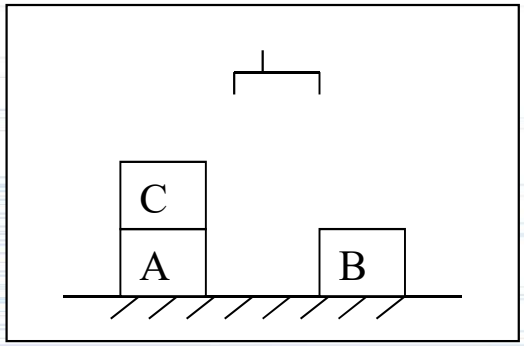
中国科学院
自动化研究所
INSTITUTE OF AUTOMATION
CHINESE ACADEMY OF SCIENCES

- **基本思想：**
  - **从用户提供的初始已知事实出发，在规则集中找出当前可适用的规则；**
  - **然后进行推理，并将推出的新事实加入到综合数据库中作为下一步推理的已知事实；**
  - **在此之后，再在知识库中选取可适用的规则进行推理，如此重复进行这一过程，直到求得了所要求的解或者没有知识可用.**
  - **用综合数据库与规则的条件部分匹配，并用规则的结论部分修改综合数据库；**

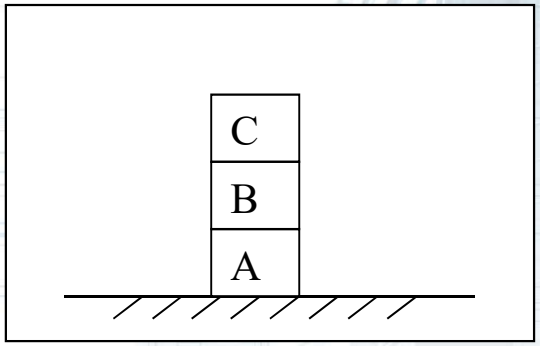
算法

- (1) 将用户提供的初始已知事实送入数据库DB；
- (2) 检查数据库DB中是否已经包含了问题的解，若有，则求解结束，并成功退出；
  否则，转(3)；
- (3) 根据数据库DB中的已知事实，扫描知识库KB，检查KB中是否有可适用(可与DB中已知事实匹配)的知识，若有则转(4)；
  否则，转(6)；
- (4) 把KB中所有的适用知识都选出来，构成可适用的知识集KS.
- (5) 若KS不空，则按某种冲突消解策略从中选出一条知识进行推理，并将推出的新事实加入DB中，然后转(2)；
  若KS空，则转(6)；
- (6) 询问用户是否可进一步补充新的事实，若可补充，则将补充的新事实加入DB中，然后转(3)；
  否则，表示求不出解，失败退出；

积木世界



初始状态

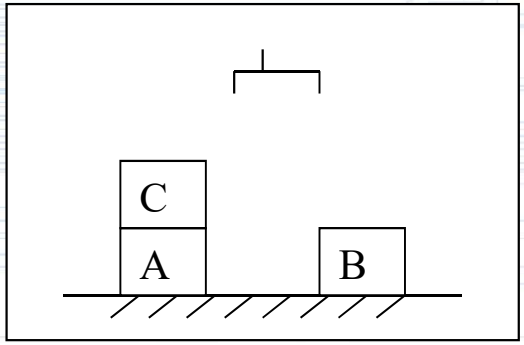


目标状态

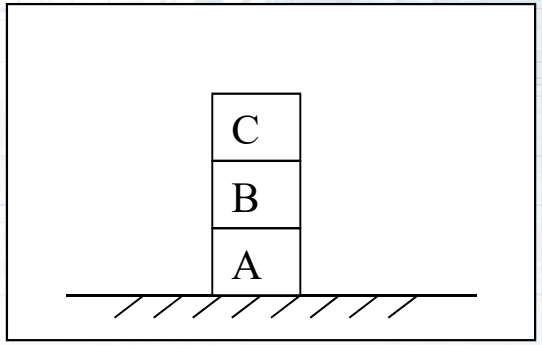
- **知识元：**
  - HANDEMPTY， Holding(x), Ontable(x), Clear(x), On(x, y)
- **规则:**
  - R1: Pickup(x): 机械手从桌子上拿起积木x。
    - HANDEMPTY ∧ Ontable(x) ∧ Clear(x) → Holding(x)
  - R2: Putdown(x): 机械手把拿着的积木放在桌子上。
    - Holding(x) → HANDEMPTY , Ontable(x), Clear(x)
  - R3: Unstack (x, y): 积木x在积木y上, 机械手拿起x。
    - HANDEMPTY ∧ On(x, y) ∧ Clear(x)→ Holding(x), Clear(y)
  - R4: Stack (x, y): 机械手把拿着的积木x放在积木y上。
    - Holding(x) ∧ Clear(y) → HANDEMPTY, On(x, y), Clear(x)

- **综合数据库**

  - **初始状态：**
    - **(HANDEMPTY，Ontable(A)，Ontable(B), On(C, A), Clear(C), Clear(B))**
  - **目标状态：**
    - **(Ontable(A)，On (B, A), On(C, B), Clear(C))**



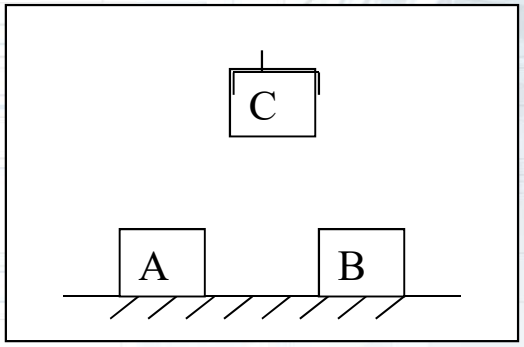
初始状态



目标状态

- (HANDEMPTY，Ontable(A)，Ontable(B), On(C, A), Clear(C), Clear(B))

  - R3: Unstack(C, A)
    - HENDEMPTY ∧ On(x, y) ∧ Clear(x)→ Holding(x), Clear(y)
- (Holding(C)，Ontable(A)，Ontable(B), Clear(B), Clear(A))

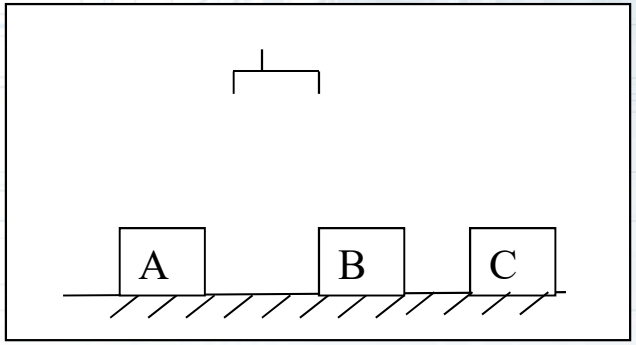


- (Holding(C)，Ontable(A)，Ontable(B), Clear(B), Clear(A))

  – R2: Putdown(c)

    - Holding(x) ⟶ HENDEMPTY , Ontable(x), Clear(x)

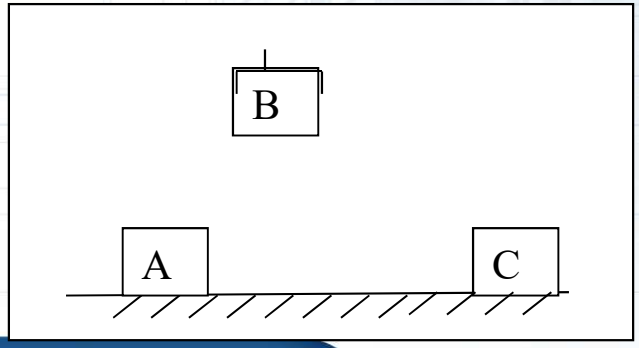
- (HANDEMPTY，Ontable(A)，Ontable(B), Ontable(C), Clear(A), Clear(B), Clear(C)



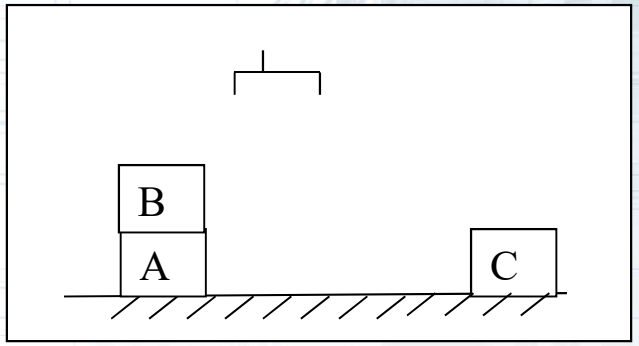
- (HANDEMPTY，Ontable(A)，Ontable(B), Ontable(C), Clear(A), Clear(B), Clear(C)

  – R1: Pickup(B)
    - HENDEMPTY ∧ Ontable(x) ∧ Clear(x) → Holding(x)

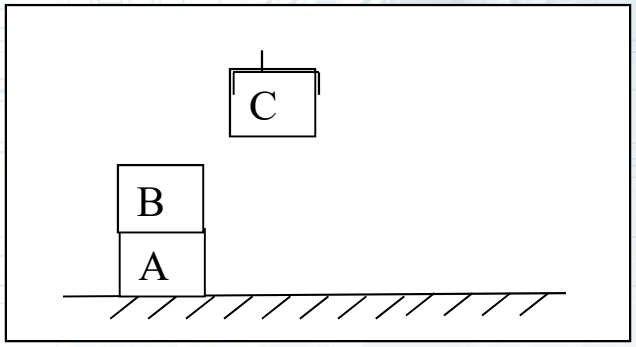
- (Holding(B), Ontable(A)，Ontable(C), Clear(A), Clear(C))



- (Holding(B), Ontable(A)， Ontable(C), Clear(A), Clear(C))
  - R4: Stack(B, A)
    - Holding(x) ∧ Clear(y) → HENDEMPTY, On(x, y), Clear(x)
- (HANDEMPTY, Ontable(A)， Ontable(C), On(B, A), Clear(B), Clear(C))



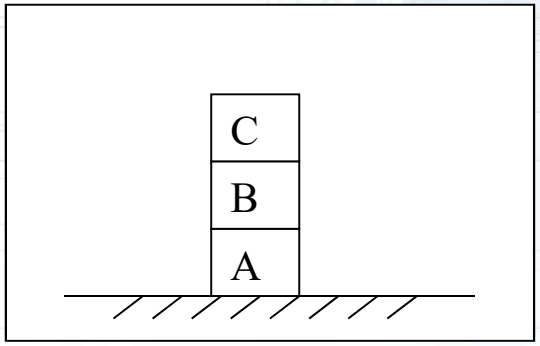
- (HANDEMPTY, Ontable(A)， Ontable(C), On(B, A), Clear(B), Clear(C))
  - R1: Pickup(C)
    - HANDEMPTY ∧ Ontable(x) ∧ Clear(x) → Holding(x)
- (Holding(C), Ontable(A)， On(B, A), Clear(B))



- (Holding(C), Ontable(A)， On(B, A), Clear(B))

  - R4: Stack(C, B)
    - Holding(x) ∧ Clear(y) → HENDEMPTY, On(x, y), Clear(x)

- (HANDEMPTY, Ontable(A)， On(B, A), On(C, B), Clear(C))



反向推理

- 基本思想：
  - 首先选定一个假设目标，然后寻找支持该假设的证据；
  - 若所需的证据都能找到，则说明原假设是成立的；
  - 若无论如何都找不到所需要的证据，说明原假设不成立，此时需要另作新的假设。

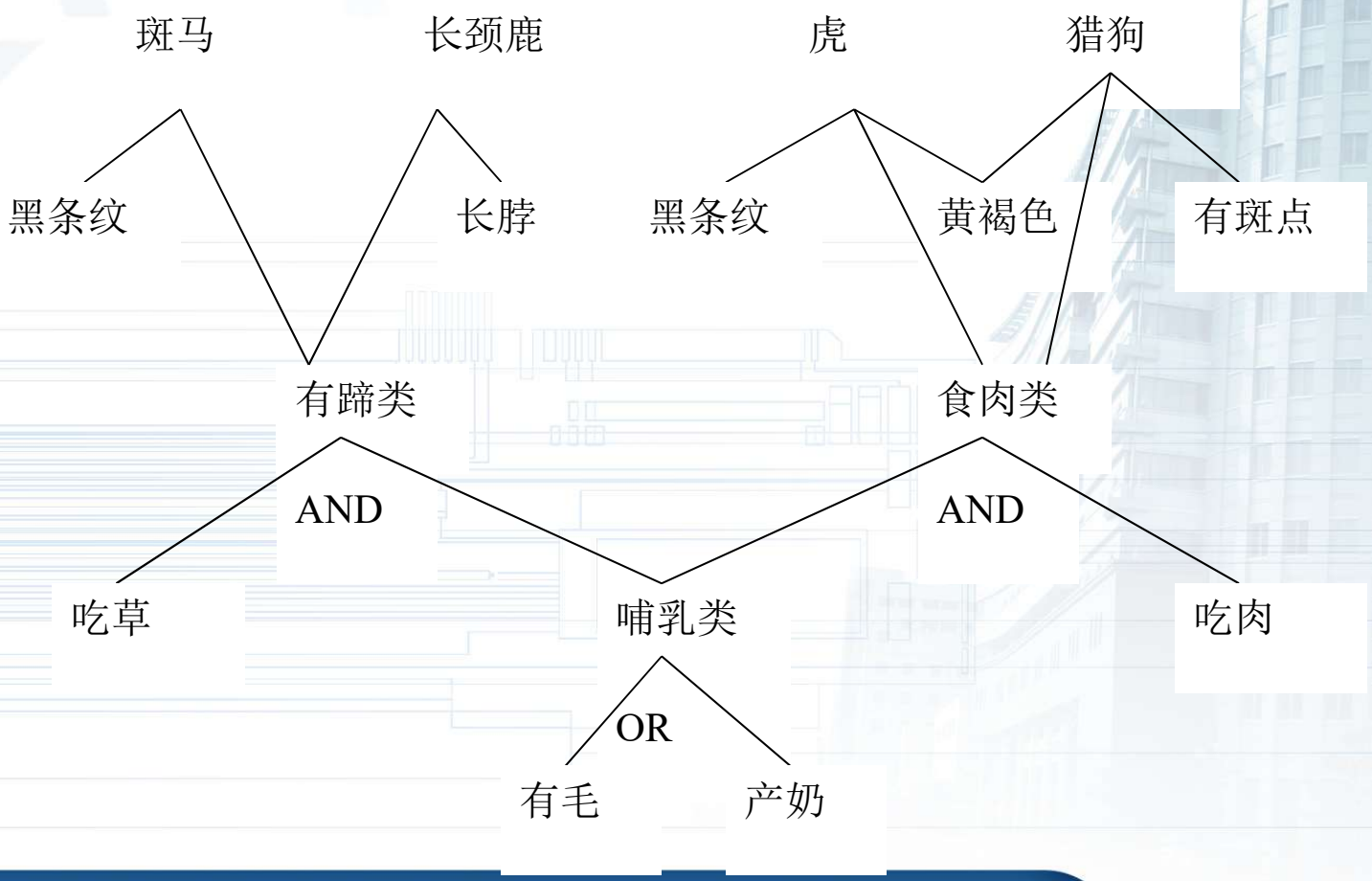
算法

- (1) 提出要求证的目标(假设)；
- (2) 检查该目标是否已在数据库中，若在，则该目标成立，成功地退出推理或者对下一个假设目标进行验证；
  否则，转下一步；
- (3) 判断该目标是否是证据，若是，则询问用户；
  否则，转下一步；
- (4) 在知识库中找出所有能导出该目标的规则，形成适用规则集KS，然后转下一步；
- (5) 从KS中选出一条规则，并将该规则的条件作为新的假设目标，然后转(2)。

动物世界

- 规则:
  - R1：动物有毛 → 哺乳类
  - R2：动物产奶 → 哺乳类
  - R3：哺乳类 ∧ 吃肉 → 食肉类
  - R4：哺乳类 ∧ 吃草 → 有蹄类
  - R5：食肉类 ∧ 黄褐色 ∧ 有斑点→ 猎狗
  - R6：食肉类 ∧ 黄褐色 ∧ 黑条纹→ 虎
  - R7：有蹄类 ∧ 长脖 → 长颈鹿
  - R8：有蹄类 ∧ 黑条纹 → 斑马

规则集的树表示



- 问题：
  - 观察到一种动物是{有毛，吃草，黑条纹}，问是不是斑马？

- 推理过程：
  - 寻找结论是斑马的规则，看它的条件部分是否可以被当前综合数据库满足。如果是，则结束；
  - 否则，看哪些条规则能推出这些条件（规则的结论与这些条件匹配）。
  - 重复这个过程。

具体推理过程

- {有毛，吃草，黑条纹}
  - R1：动物有毛 → 哺乳类
  - R2：动物产奶 → 哺乳类
  - R3：哺乳类 ∧ 吃肉 → 食肉类
  - R4：哺乳类 ∧ 吃草 → 有蹄类
  - R5：食肉类 ∧ 黄褐色 ∧ 有斑点→ 猎狗
  - R6：食肉类 ∧ 黄褐色 ∧ 黑条纹→ 虎
  - R7：有蹄类 ∧ 长脖 → 长颈鹿
  - R8：有蹄类 ∧ 黑条纹 → 斑马

混合推理

- 正向推理
  - 具有盲目、效率低等缺点，可能推出许多与问题求解无关的子目标;
- 逆向推理
  - 若提出的假设目标不合适，也会降低系统的效率。

- 正向推理 +逆向推理?
- 混合推理

讲课内容

一、产生式系统

二、推理

三、控制策略

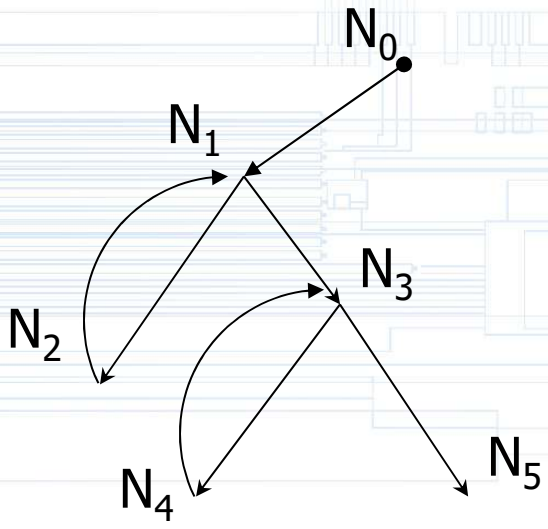
四、两类产生式系统

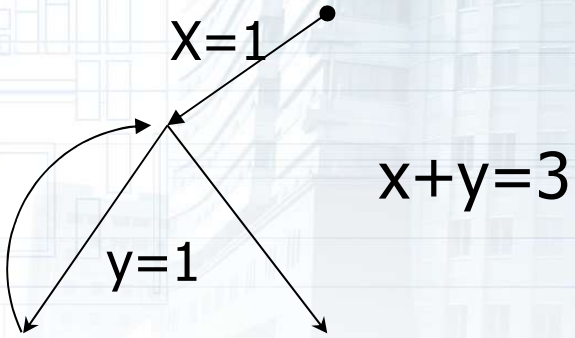
五、基于规则的演绎系统

- 选取规则的原则称为**控制策略**

- 不可撤回的控制策略：
  - 在任意时刻，如果从被激励的规则集中选择点燃的规则都是合适的，即，使用这条规则所求的解，一定在全局解的路径上。
  - 特点：局部解与全局解是一致的

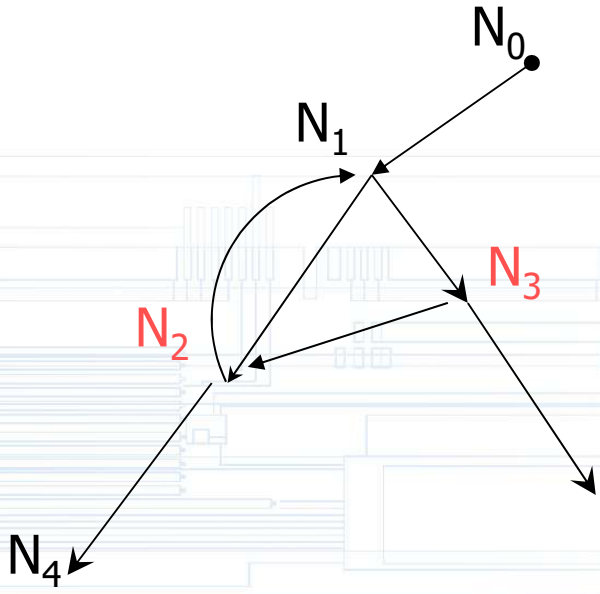
- 试探性控制策略
  - 使用某条规则时, 必须为以后应用另一条规则做好准备;
  - 回溯型:
    - 特点：每个被放弃的状态，可以保证不在解路上





– 图搜索

• 任一个暂时放弃的状态，将来都可能被重新使用;



冲突删除策略

- **规则的状态:**
  - 建议：如果某规则的**部分条件**与综合数据库匹配，此规则处于建议状态。
  - 激励：如果某规则的**全部条件**与综合数据库匹配，此规则处于激励状态。
  - 点燃：根据策略从激励的规则中**选取一个执行**，称此规则被点燃。

- 当激励的规则多于一条时，选择哪条点燃，策略有两种：
  - 领域相关: 启发式知识
  - 领域无关: 冲突删除策略

冲突删除策略

- 重要性排序
  - 产生式规则事先按照重要性排序;
  - 当多条规则被激励时，选择最重要的规则点燃。

- 特殊排序
  - 更特殊的规则先执行。
  - 尺寸排序：条件多的先执行。
    - R1：A∧B∧C → ……
    - R2：D∧E → ……
    - 先执行R1.
  - 包含排序：两个规则的条件部分，一个是另一个的子集，执行全集的那条(尺寸排序的特例)
    - R1：A∧B∧C → ……
    - R2：A∧B → ……
    - 先执行R1.

- 新旧排序
  - 如果规则的获取时间不同，新得到的知识比旧知识更先点燃。
- 匹配程度排序
  - 非确定性匹配中，如果有加权的情况，按照规则的权值大小排序。

- 数据排序
  - 重要性排序：
    - 对知识元事先按重要性排序。
    - 被激励的规则的条件部分第一个文字在综合数据库中的重要性，决定点燃的规则。
    - 例：{A，B，C，D，E}
      - R1：A∧D∧E → ……
      - R2：B∧C → ……
      - 先执行R1.
  - 新旧排序：新进入综合数据库的数据优先级高。

讲课内容

一、产生式系统

二、推理

三、控制策略

四、两类产生式系统

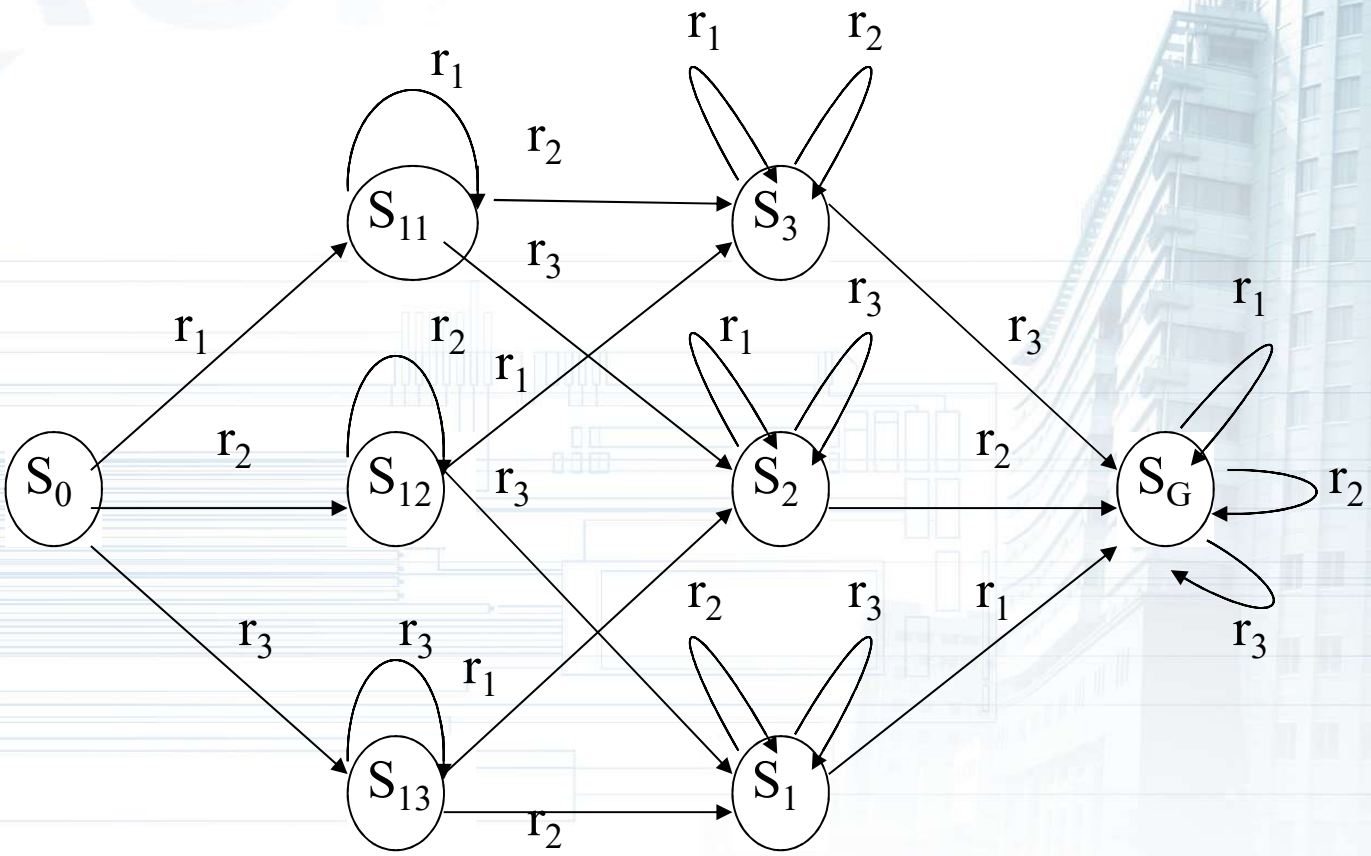
五、基于规则的演绎系统

- 可交换产生式系统
- 可分解产生式系统

可交换产生式系统

- 使用规则的次序不影响产生的结果
- 例子：
  - r1—r2—r3
    - r1—r3—r2
    - r2—r1—r3
    - r2—r3—r1
    - r3—r1—r2
    - r3—r2—r1

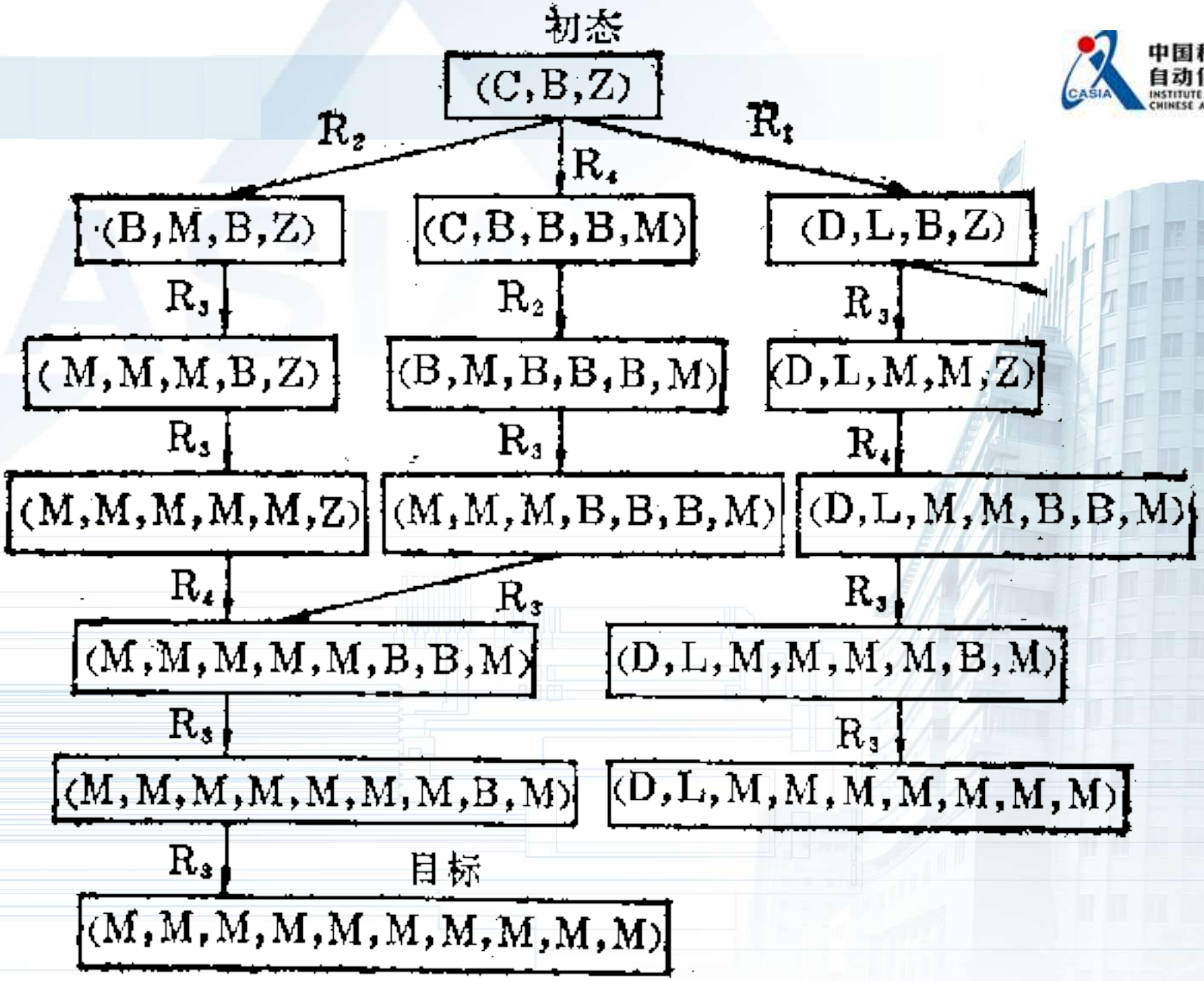
例子



- 如果一个产生式系统对任意数据库都满足如下条件，则称这个产生式系统是可交换的
  - 令R是可作用于数据库D的规则集，$r_i \in R$作用于D产生D’，则R中的任一条规则均可作用于D’
  - 如果D满足目标条件，那么可作用于D上的任一条规则，作用于D后产生D’，也满足目标条件
  - 令R是可作用于数据库D的规则集，对数据库D’来说，使用规则$r_i \in R$的顺序不影响从D 到D’

可分解产生式系统

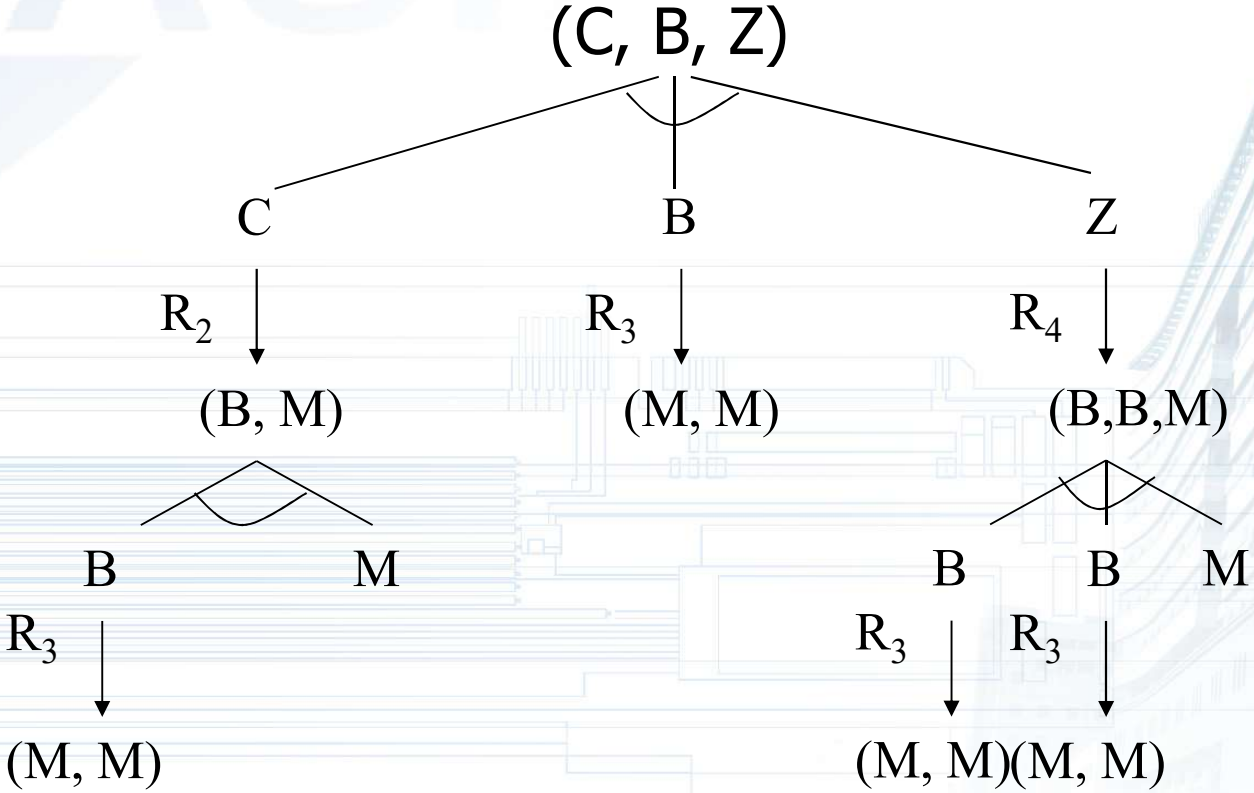
- 例子:
  - 初始数据库: (C, B, Z)
  - 目标数据库: (M, M,……M)
  - 规则:
    - R1: C → (D, L)
    - R2: C → (B, M)
    - R3: B → (M, M)
    - R4: Z → (B, B, M)


初态
(C,B,Z)
R2
R4
R1
(B,M,B,Z)
(C,B,B,B,M)
(D,L,B,Z)
R3
R2
R3
(M,M,M,B,Z)
(B,M,B,B,B,M)
(D,L,M,M,Z)
R3
R3
R4
(M,M,M,M,M,Z)
(M,M,M,B,B,B,M)
(D,L,M,M,B,B,M)
R4
R3
R3
(M,M,M,M,M,B,B,M)
(D,L,M,M,M,M,B,M)
R3
R3
(M,M,M,M,M,M,M,B,M)
(D,L,M,M,M,M,M,M,M)
R3
目标
(M,M,M,M,M,M,M,M,M,M)

中国科学院
自动化研究所
INSTITUTE OF AUTOMATION
CHINESE ACADEMY OF SCIENCES

- **R1: C → (D, L)**
- **R2: C → (B, M)**
- **R3: B → (M, M)**
- **R4: Z → (B, B, M)**

将初始数据库分成几个独立的部分：



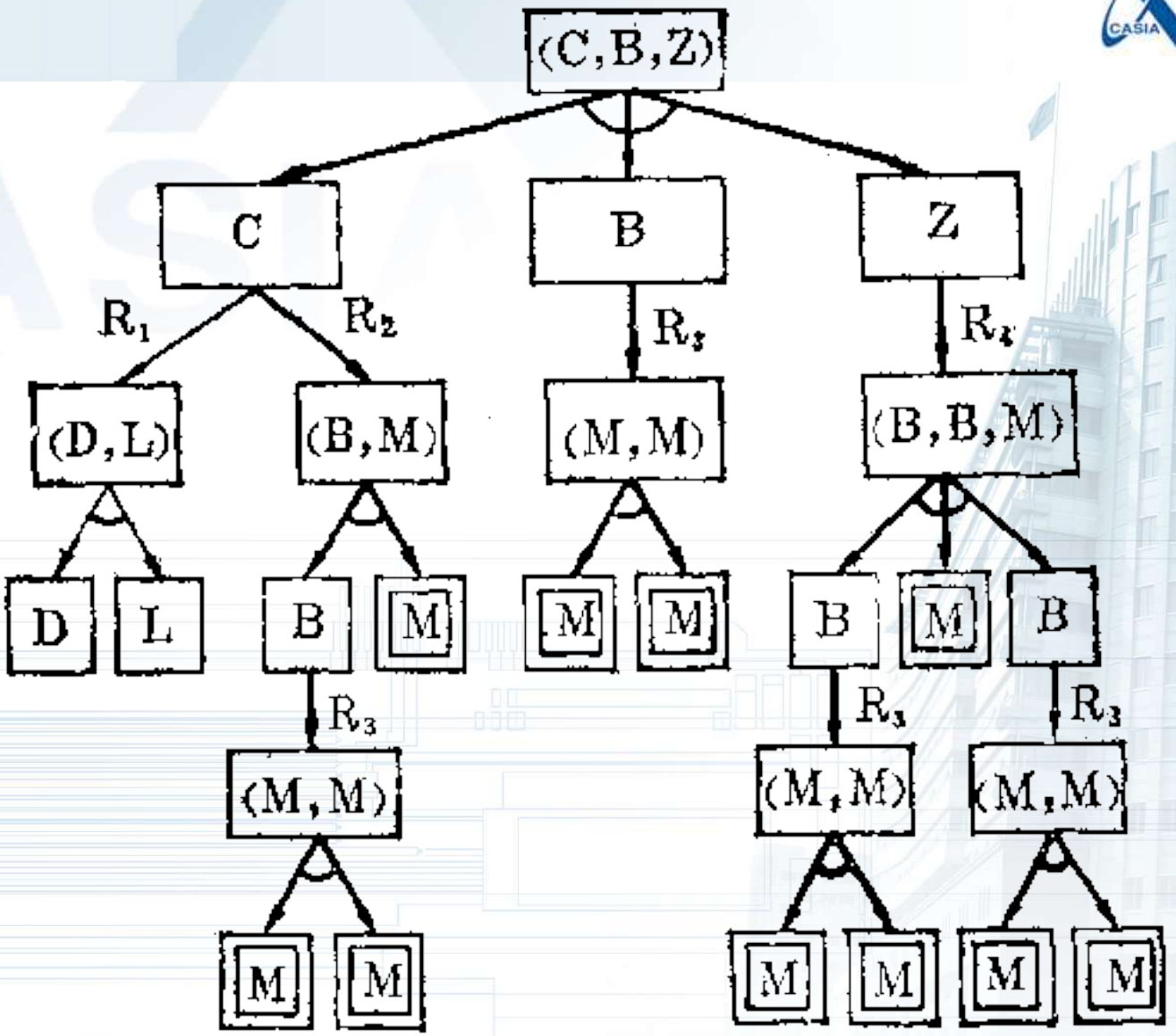
可分解产生式系统

- 定义：
  - 如果一个系统可将其数据库分为几个子数据库，并通过规则作用于子数据库，而达到目标，这个系统称为可分解产生式系统。

SPLIT算法

- 1．DATA←初始数据库；
- 2．$\{D_i\}$←DATA的分解表示。每个D_i为一个子数据库；
- 3．直到所有D_i满足结束条件。

  - 3.1 从$\{D_i\}$中选择不满足结束条件的数据库D*；
  - 3.2 从$\{D_i\}$中删除D*；
  - 3.3 选择一条可作用于D*的规则R；
  - 3.4 D←R作用于D*所产生的结果；
  - 3.5 $\{d_i\}$←D的分解表示；
  - 3.6 将$\{d_i\}$加入$\{D_i\}$中；


(C,B,Z)
C
B
Z
R1
R2
R4
R5
(D,L)
(B,M)
(M,M)
(B,B,M)
D
L
B
M
M
M
B
M
B
R3
R3
(M,M)
(M,M)
M
M
M
M
M
M

讲课内容

一、产生式系统

二、推理

三、控制策略

四、两类产生式系统

五、基于规则的演绎系统

基于规则的演绎系统

- **自动定理证明的方法的问题：**
- **子句表示指使得三大问题（归结方法的缺点）**
  - 与自然语言和思维习惯差别太大，不便于阅读与理解；

例如：

- **“鸟能飞”：$(\forall x)(Bird(x) \rightarrow fly(x))$**
- **子句：$\sim Bird(x) \vee fly(x)$**
- **不够直观、自然**

– 有可能丢失一些重要的控制信息
  - (~A∧~B)→C
  - (~A∧~C)→B
  - (~B∧~C)→A
  - ~A→(B∨C)
  - ~B→(A∨C)
  - ~C→(A∨B)
  - 子句: A ∨ B ∨ C

– 在演算时会产生大量冗余
  - 例如:
  - 子句集 ~P ∨~ Q， ~P ∨~R，~S ∨P， ~U ∨S， U，~W ∨R，W
  - 可以证明，存在一个反演。
  - 如果选C_0= ~P ∨~ Q 作为顶子句，则无论如何也推不出空子句，因为：去掉~P ∨~ Q 剩余的字句不相容。
  - ~P ∨~ Q 作为顶子句将导致冗余。

产生式系统推演

中国科学院
自动化研究所
INSTITUTE OF AUTOMATION
CHINESE ACADEMY OF SCIENCES

- 主要讨论基于规则的推演的形式描述方法；
- 特点：
  - 将一个被证定理的条件与结论分别表示；
  - 将证明这个定理可使用的合理定义和已知的定理单独表示；
- 例如：

(A)吃草 $\wedge$ (B)有黑白斑 $\rightarrow$ (C)斑马

基于规则的推演分为两个对称类

- **正向推演——证明子句集为不相容式（子句为合取式，利用Skolem函数消去∃量词）；**

- **反向推演——证明子句集为用真式（子句为析取式，利用Skolem函数消去∀ 量词）；**

使用如下术语

- 被证明的条件：事实
- 被证明的结论：目标
- 已知定义、公理和定理：规则

- 基于规则的演绎系统(与/或形演绎推理)
  - Nilson
- 事实, 规则, 目标
- 分类
  - 正向演绎
  - 反向演绎
  - 双向演绎

正向演绎

- 事实的表示：
  - 已知事实用不含蕴含符号“→”的与/或树形表示
  - 与或形
    - 无量词约束
    - 否定符只作用于单个文字
    - 只有“与”(∧)、“或”(∨)

- **例子:**

$(\exists u)(\forall v)(Q(v, u)\wedge\sim((R(v)\vee P(v))\wedge S(u, v)))$

$\Rightarrow (\exists u)(\forall v)\,(Q(v, u)\wedge((\sim R(v)\wedge \sim P(v))\vee \sim S(u, v)))$

$\Rightarrow Q(v, a)\wedge((\sim R(v)\wedge \sim P(v))\vee \sim S(a, v))$

Skolem化

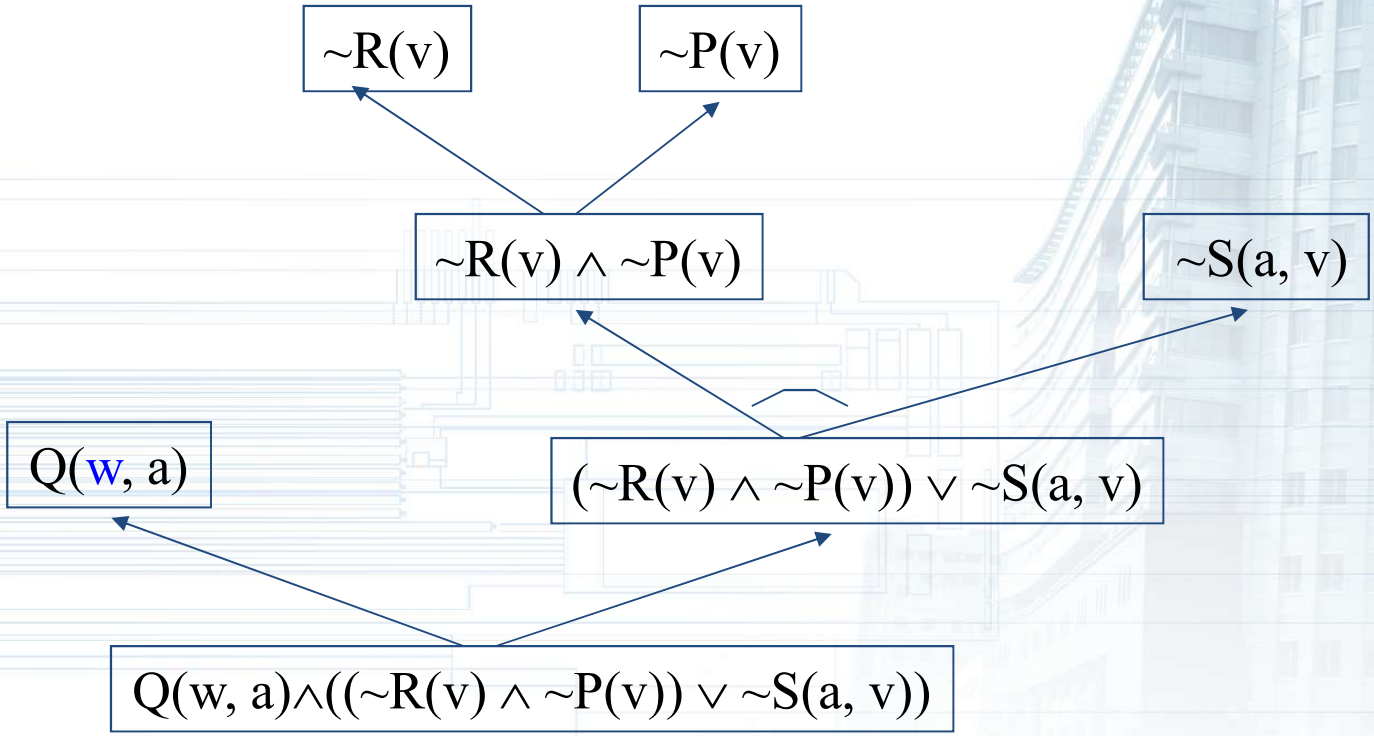
$\Rightarrow Q(w, a)\wedge((\sim R(v)\wedge \sim P(v))\vee \sim S(a, v))$

主合取元变量换名

事实的与或树表示

中国科学院
自动化研究所
INSTITUTE OF AUTOMATION
CHINESE ACADEMY OF SCIENCES

$Q(w, a) \wedge ((\sim R(v) \wedge \sim P(v)) \vee \sim S(a, v))$



解图集： $Q(w, a), \sim R(v) \vee \sim S(a, v), \sim P(v) \vee \sim S(a, v)$

规则表示

中国科学院
自动化研究所
INSTITUTE OF AUTOMATION
CHINESE ACADEMY OF SCIENCES

- 规则的形式(F规则)：$L \rightarrow W$

  其中，L是单文字集合的析取式，

  W是~、 ∧与、 ∨或形，变量受全称量词约束

- 一个规则可以化为多个规则，以使规则的左部只有一个文字

- 例： $(\forall x)((\exists y)(\forall z)P(x, y, z)) \rightarrow (\forall u)Q(x, u))$

  $\Rightarrow (\forall x)(\sim ((\exists y)(\forall z)P(x, y, z)) \vee (\forall u)Q(x, u))$

  $\Rightarrow (\forall x)((\forall y)(\exists z)\sim P(x, y, z) \vee (\forall u)Q(x, u))$

  $\Rightarrow (\forall x)(\forall y)(\exists z)\ (\forall u)(\sim P(x, y, z) \vee Q(x, u))$

  $\Rightarrow \sim P(x, y, f(x, y)) \vee Q(x, u)$

  $\Rightarrow P(x, y, f(x, y)) \rightarrow Q(x, u)$

- 例： $(L1 \vee L2) \rightarrow W \Rightarrow L1 \rightarrow W$ 和 $L2 \rightarrow W$

$(L1 \vee L2) \rightarrow W = \sim (L1 \vee L2) \vee W = (\sim L1 \vee W) \wedge (\sim L2 \vee W)$

对于命题逻辑的情况

- **目标表示：**
  化为析取式

- **例：**

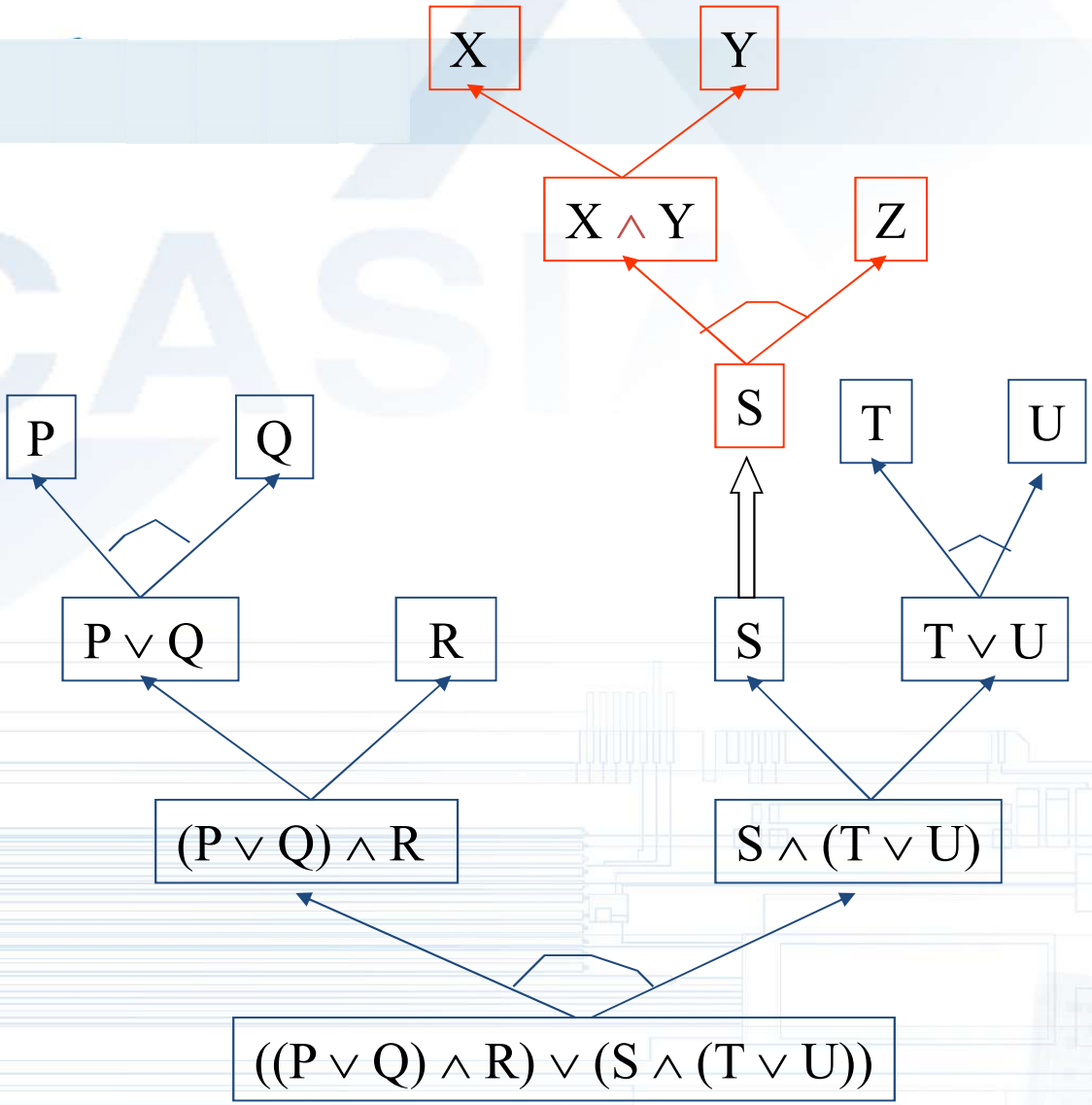
  事实： $((P \vee Q) \wedge R) \vee (S \wedge (T \vee U))$

  规则： $S \rightarrow (X \wedge Y) \vee Z$

  目标： $P \vee Q \vee S$，$P \vee Q \vee T \vee U$，$S \vee R$，$R \vee T \vee U$，
  $P \vee Q \vee X \vee Z$，$P \vee Q \vee Y \vee Z$，
  $P \vee Q \vee T \vee U$，$R \vee X \vee Z$，$R \vee Y \vee Z$，$R \vee T \vee U$

结束条件

- 结束条件
  - 找到一个终止在目标节点上的解图



P∨ Q∨ S

P∨ Q∨ T∨ U

S∨ R

R∨ T∨ U

P∨ Q∨ X∨ Z

P∨ Q∨ Y∨ Z

P∨ Q∨ T∨ U

R∨ X∨ Z

R∨ Y∨ Z

R∨ T∨ U

规则的子句：

$S \rightarrow (X \wedge Y) \vee Z$

=> $\sim S \vee (X \wedge Y) \vee Z$

=> $\sim S \vee X \vee Z$

$\sim S \vee Y \vee Z$

结论：加入规则后得到的解图，是事实与规则对应子句的归结式

例：

事实：$A \vee B$

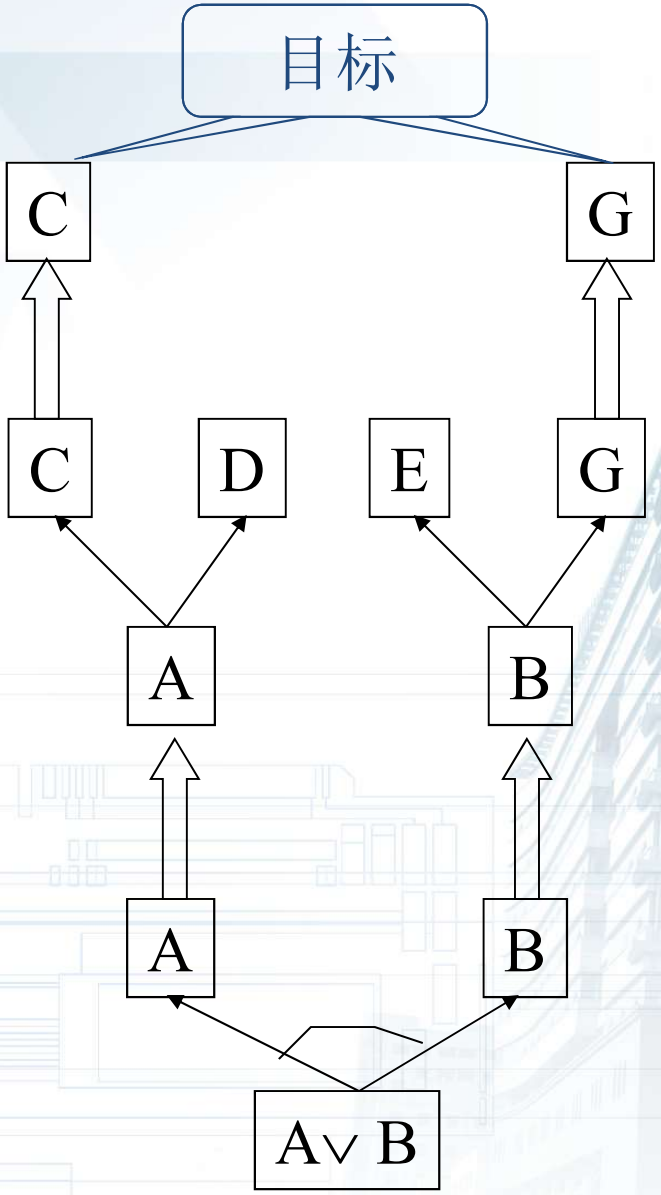
规则集：

$A \rightarrow C \wedge D$

$B \rightarrow E \wedge G$

目标公式：

$C \vee G$



对于谓词逻辑的情况

- 目标用**Skolem** 化的对偶形式，即：
  - 消去全称量词，用**Skolem**函数代替
  - 保留存在量词
  - 对析取元作变量换名

例：$(\exists y)(\forall x)(P(x, y) \vee Q(x, y))$

$\Rightarrow (\exists y)(P(f(y), y) \vee Q(f(y), y))$

$\Rightarrow P(f(y_1), y_1) \vee Q(f(y_2), y_2)$ 换名

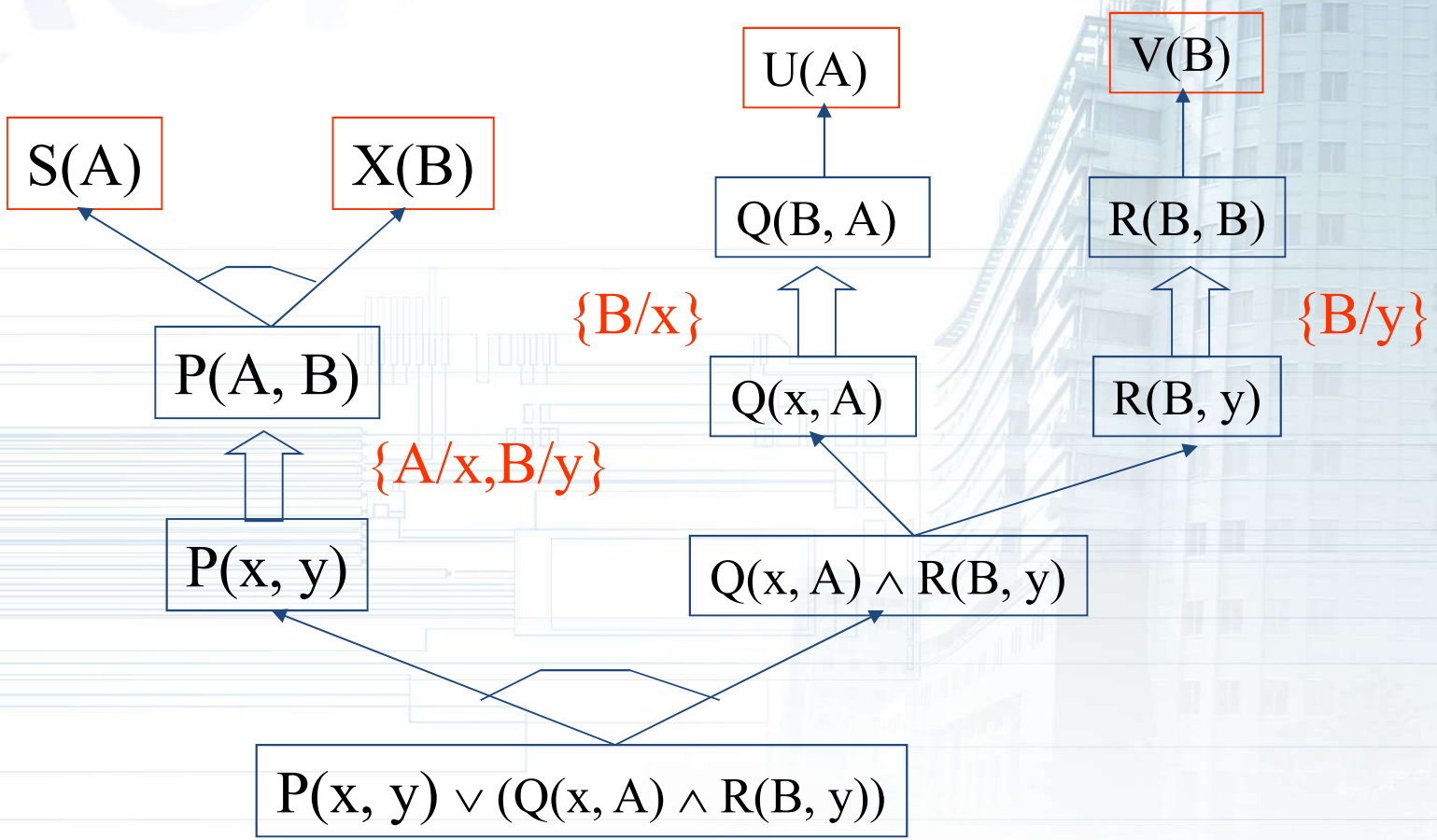
例：事实：$P(x, y) \vee (Q(x, A) \wedge R(B, y))$

规则集：$P(A, B) \rightarrow (S(A) \vee X(B))$

$Q(B, A) \rightarrow U(A)$

$R(B, B) \rightarrow V(B)$

目标：$S(A) \vee X(B) \vee U(A) \vee V(B)$



一个问题

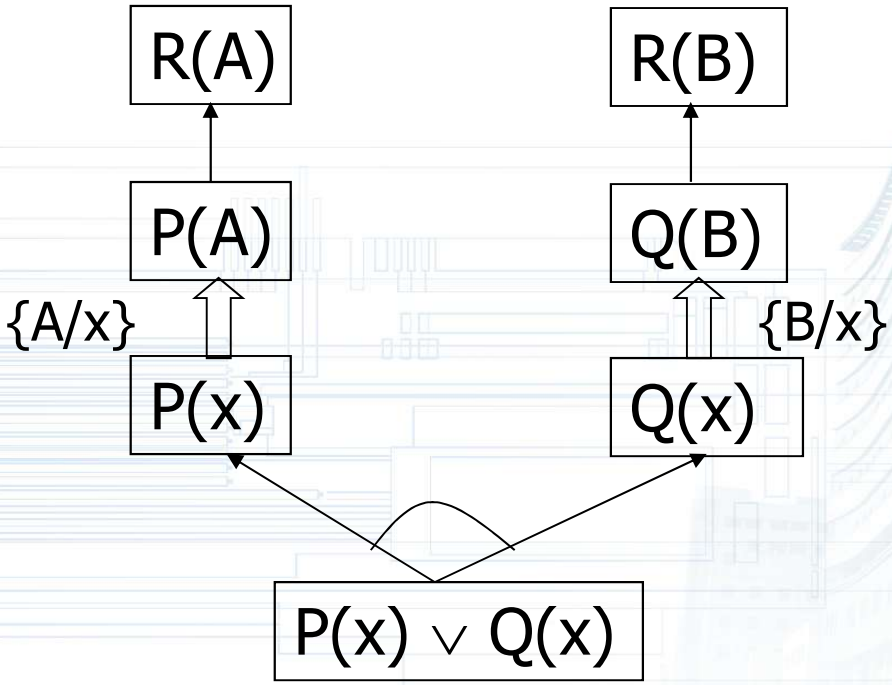
事实: $P(x) \vee Q(x)$

规则: $P(A) \rightarrow R(A)$

$Q(B) \rightarrow R(B)$

目标: $R(A) \vee R(B)$

- 置换是否相容?



一致解图

- 如果一个解图中所涉及的置换是一致的，则该解图称为一致解图。

- 设有置换集$\{u_1, u_2, \ldots, u_n\}$，其中$u_i=\{t_{i1}/v_{i1}, \ldots, t_{im(i)}/v_{im(i)}\}$，定义表达式$U_1=(v_{11}, \ldots, v_{1m1}, \ldots, v_{n1}, \ldots, v_{nm(n)})$，$U_2=(t_{11}, \ldots, t_{1m(1)}, \ldots, t_{n1}, \ldots, t_{nm(n)})$.

  置换集$\{u_1, u_2, \ldots, u_n\}$称为一致的，当且仅当 U_1和U_2是可合一的。

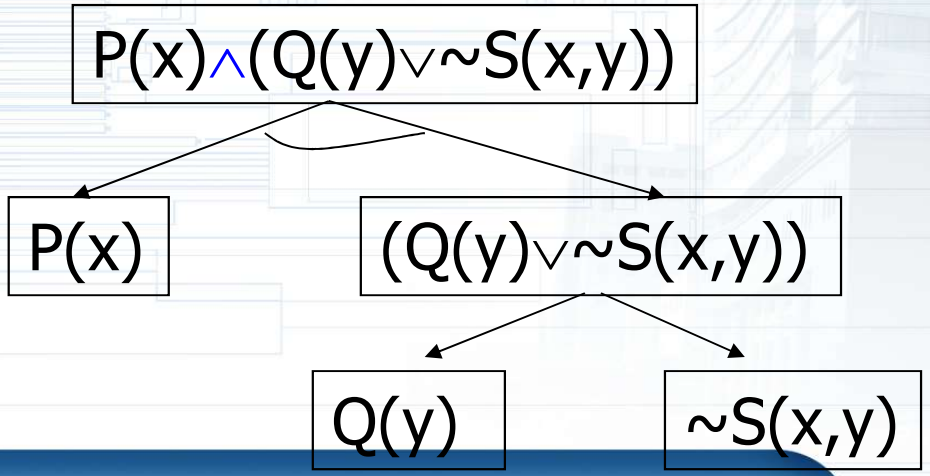
- U_1、U_2的mgu是$\{u_1, u_2, \ldots, u_n\}$的合一复合。

反向演绎系统

- 事实表示
  - 合取式
- 规则的表示（B规则）
  - W → L（读作：欲知L的真假，证明W的真假）

    其中： W为任意一个只用~、 ∧、 ∨连接的逻辑公式；

    L是单文字集合的合取式。
  - 可以将规则化为右部只有一个文字的形式，它们之间的关系是“与”的关系，即：形为W→L1 ∧L2，则变换为L→ W1 和 L→ W2



- 目标表示
  - 用“对偶形”对目标进行Skolem化，即消去全称量词，变量受存在量词约束，对主析取元中的变量换名
  - 用合取连接符连接

- 终止条件:
  - 包含一个终止于事实节点的一致解图.

反向推演的过程

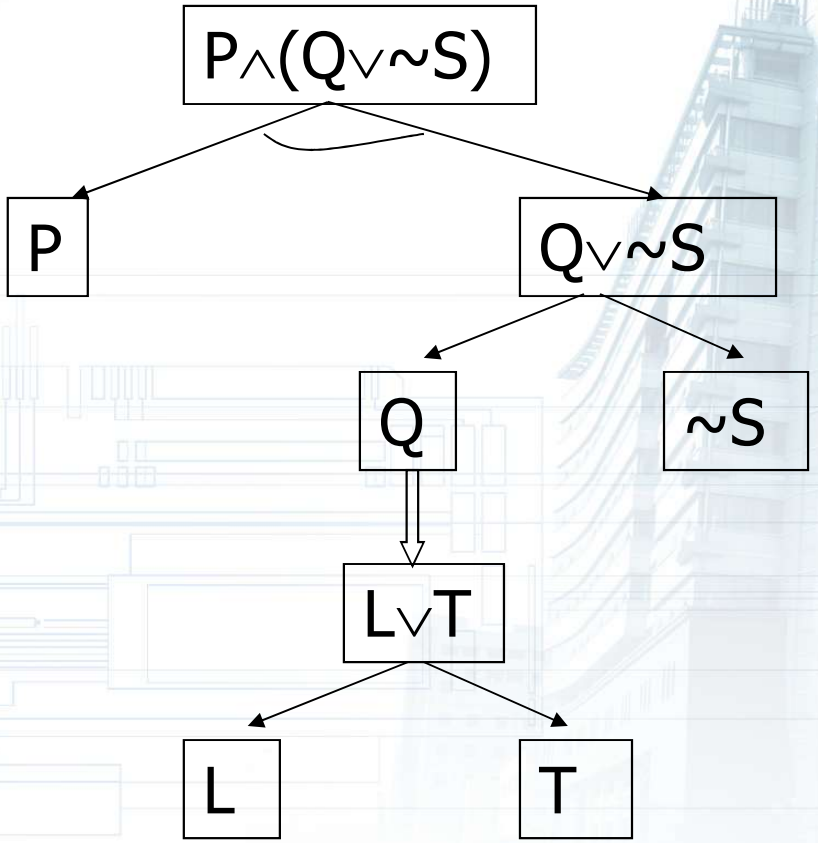
- 规则是以右部与目标的叶节点匹配
- 一般反向推演的事实采用析取式表示

反向演绎的例子

事实: **P ∧ L**

规则: **L∨T→Q**

目标: **P∧(Q∨~S)**



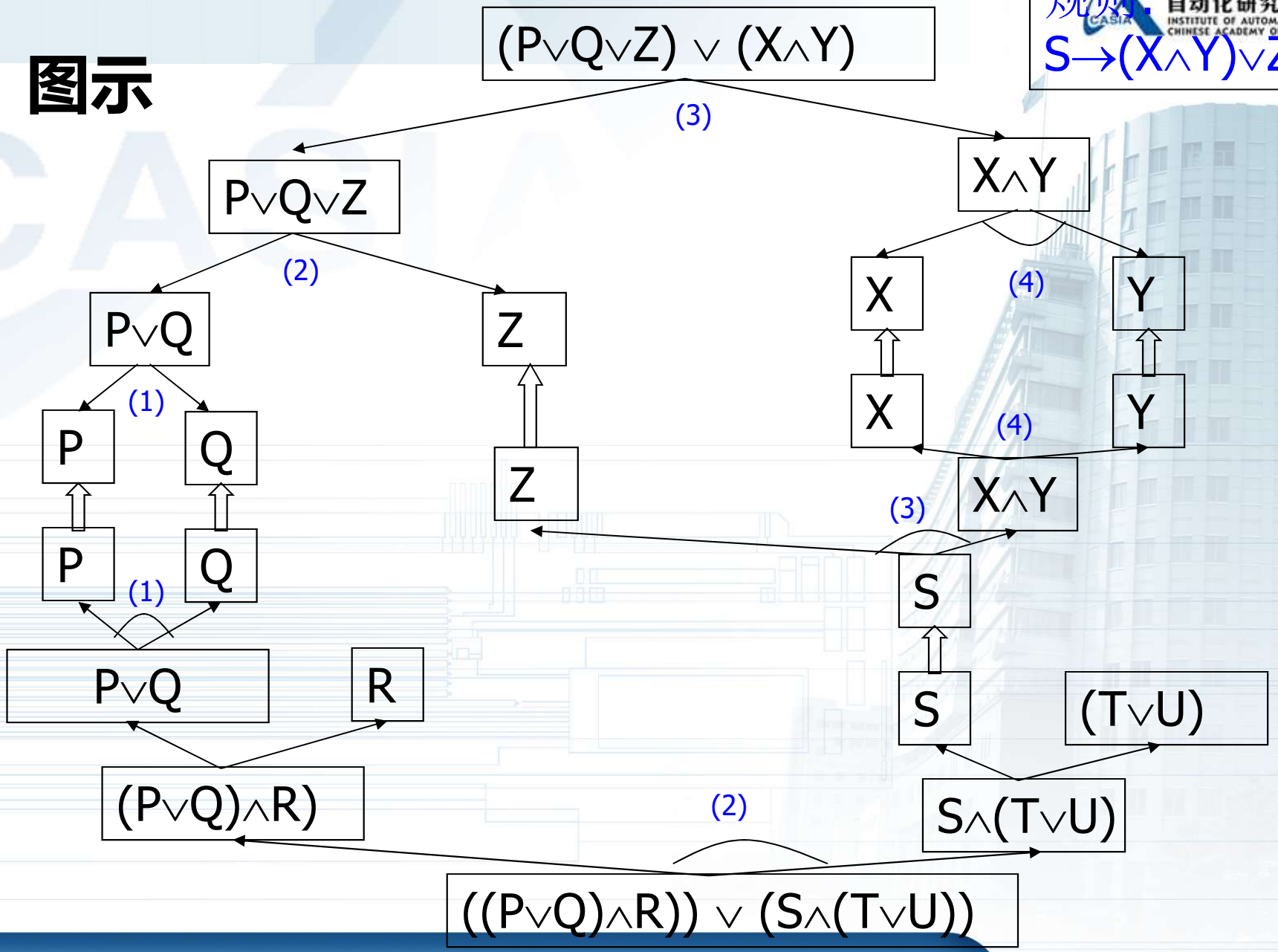
双向演绎

- 起因:
  - 正向演绎要求目标公式是文字的析取式，反向演绎推理要求事实公式为文字的合取式，都有一定的局限性
  - 为克服这些局限性，并充分发挥各自的长处，可进行双向演绎推理

例子

- **事实：** $\{(P \vee Q) \wedge R)) \vee (S \wedge (T \vee U)\}$
- **规则：** $S \rightarrow (X \wedge Y) \vee Z$
- **目标：** $(P \vee Q \vee Z) \vee (X \wedge Y)$

图示


(P∨Q∨Z) ∨ (X∧Y)
(3)
P∨Q∨Z
X∧Y
(2)
(4)
P∨Q
Z
X
Y
(1)
(4)
P
Q
Z
X
Y
(3)
X∧Y
P
Q
S
(1)
P∨Q
R
S
(T∨U)
(P∨Q)∧R)
(2)
S∧(T∨U)
((P∨Q)∧R)) ∨ (S∧(T∨U))

规则：
S→(X∧Y)∨Z

终止条件

- 解图终止
  - 连接弧与非连接弧对应
  - 连接弧必须保证所有出发的枝都相接
  - 非连接弧节点只要保证有一枝相接
  - 一致

中国科学院
自动化研究所
INSTITUTE OF AUTOMATION
CHINESE ACADEMY OF SCIENCES

感谢同学们听课

欢迎讨论与交流