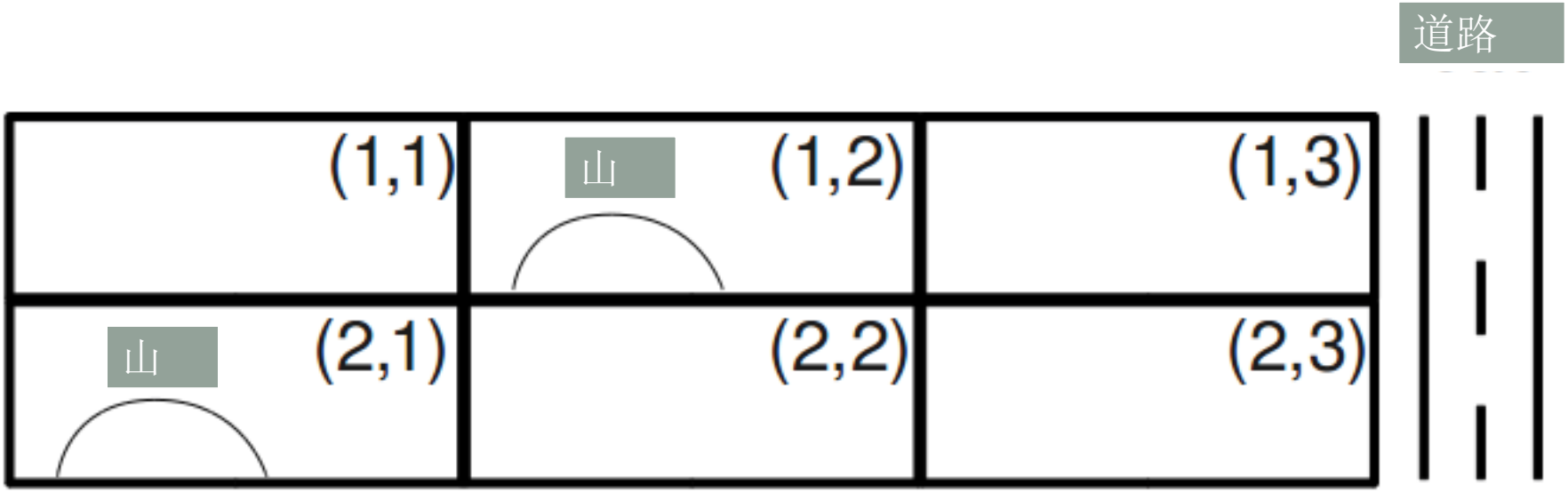
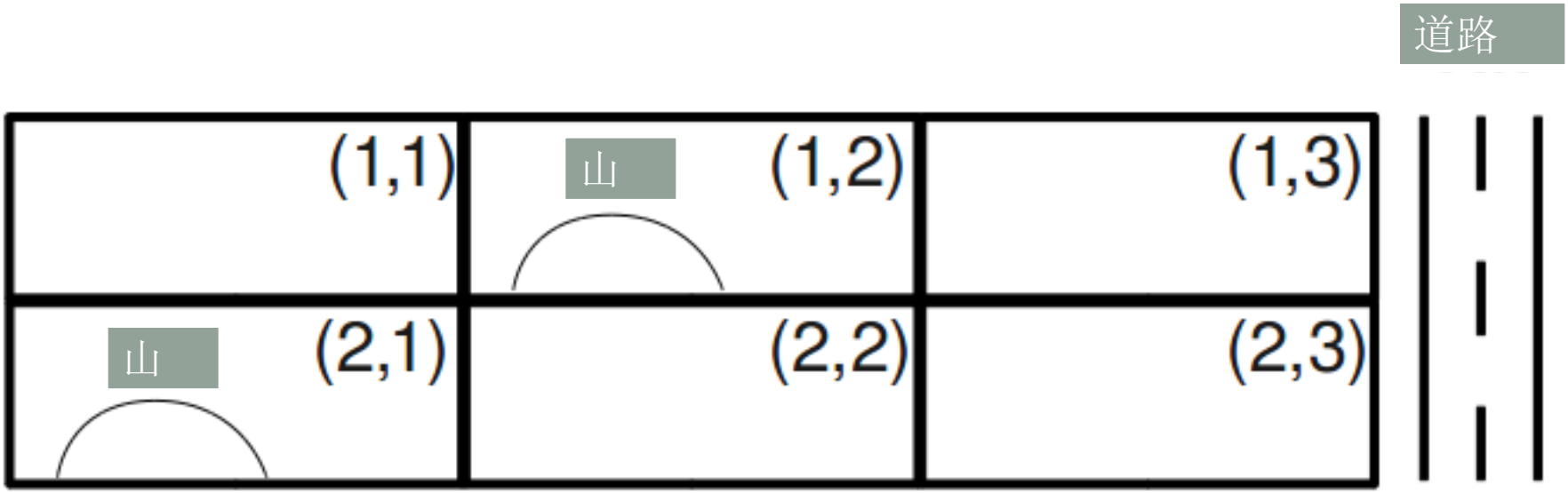
举例：校园规划

- 由你来决定一个校园中新区域的规划设计。这个新区域中有四个建筑：一个行政管理楼（A），一个汽车站（B），一个教室楼（C), 和一个学生宿舍楼(D)。这些建筑必须坐落于以下的网格里。



举例：校园规划（继续）



- 规划必须满足以下约束：
  1. 汽车站(B)必须邻近道路
  2. 行政管理楼(A)和教室楼(C)必须邻近汽车站(B)
  3. 教室楼(C)必须邻近宿舍楼(D)
  4. 行政管理楼(A)一定不能靠近宿舍楼(D)
  5. 行政管理楼(A)一定不能建在一个山上
  6. 宿舍楼(D)一定要么建在一个山上，或者靠近道路
  7. 所有建筑必须在不同的网格里。
- 邻近的意思是建筑所在网格必须共享一个边，而不是一个角。

举例：校园规划（继续）

- 一元约束
  - 哪些约束是一元的？
  - 应用一元约束以后的各变量的值域是什么？
- 弧的一致性检查（AC-3算法）
  - 从以上的值域结果开始，强化弧的一致性。初始时，队列里包括所有的弧（按字母顺序排序）
  - 检查弧A→B的一致性后，A,B的值域是什么？
  - 从以上结果继续弧一致性的检查，证实检查A->C,A->D,B->A,B->C,B->D，C->A 不改变任何值域
  - 接下来检查C->B, 各变量值域如何变化？哪些弧被加入弧队列中？
  - 继续该算法，直至弧队列为空，此时各变量的值域是什么？

举例：校园规划（继续）

- 开始搜索
  - 使用最小剩余值启发信息来选择从哪个变量开始赋值？
  - 使用最少约束值启发信息来选择该变量的值？
  - 赋值后，再递归检查弧的一致性，然后各变量的值域是什么？
  - 此时是否找到了一个解？

今天的内容

- 命题逻辑
  - 语法，语义和推理
- 逻辑型智能体

人工智能导论

命题逻辑(Propositional Logic)：语法，语义和推理

知识

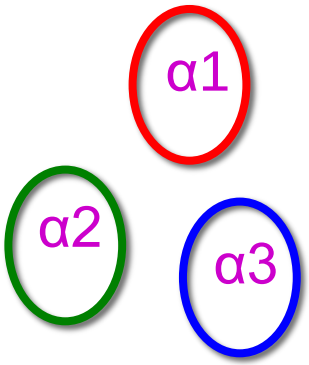
- 知识库= 在正式语言中定义的一个句子的集合
- 声明式法构建智能体(或其他系统):
  - 告诉 智能体它需要知道的(或让它自己*学习*这些知识)
  - 然后它可以询问 自己在一个环境里如何行动—询问的结果应遵循知识库里的知识
- 在*知识层*描述智能体
  - 具体说明智能体 *知道 什么*, 它的目标是什么，和实现细节无关
- 一个推理算法可以回答任何可以回答的问题
  - 比较而言，一个搜索算法只能回答 “如何从A 到 B” 的问题

领域特定事实

领域独立的通用算法和代码

逻辑

- 语法: 定义句子
- 语义:
  - *可能的世界有哪些?*
  - 这些句子在哪些世界里为 *真*？(句子真实性的*定义*)



语法空间

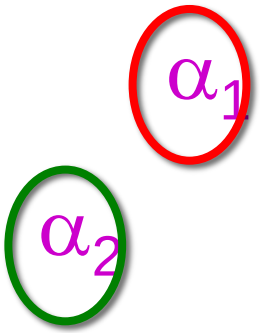
语义空间

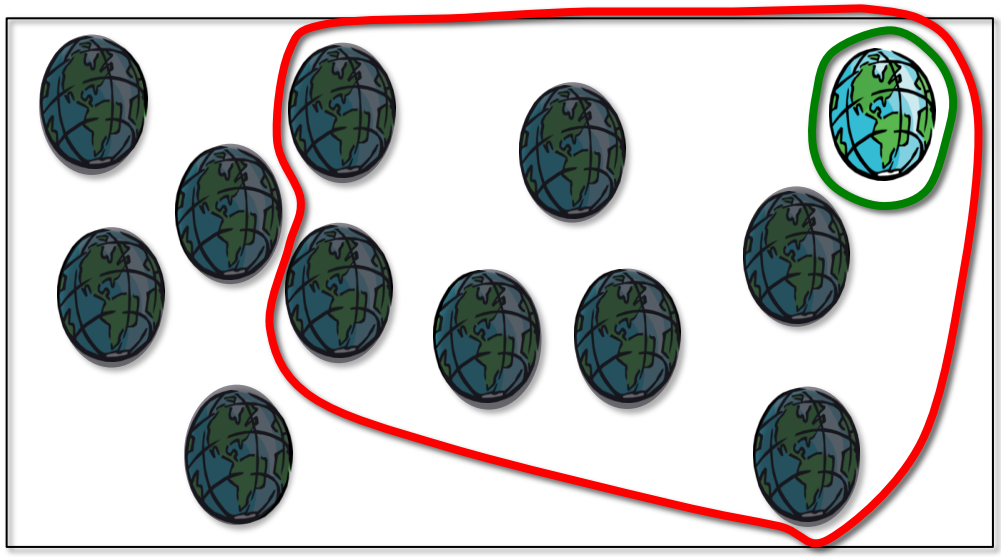
举例

- 命题逻辑（Propositional logic）
  - 语法: $P \vee (\neg Q \wedge R)$; $X_1 \Leftrightarrow (Raining \Rightarrow Sunny)$
  - 可能的一个世界: {P=true,Q=true,R=false}, {X1=true, Raining = false, Sunny=true} , or {110}， {101}
  - 语义: $\alpha \wedge \beta$ 是真 当且仅当 α 真 并且 β 真 (等)

推理的内容: 蕴涵(entailment)

- ***蕴涵***: $\alpha \models \beta$ ("α 牵涉(entails) β" or "β 遵循于(follows from) α")
  当且仅当在 α 为真的每个世界里, β 也是真
  - 换句话说, α-的世界（为真的那些世界） 是 β-的世界的一个子集
    [***models***$(\alpha) \subseteq$ ***models***(β)]
- 例如, $\alpha_2 \models \alpha_1$
- (比如 α_2 是 $\neg Q \wedge R \wedge S \wedge W$
  α_1 是 $\neg Q$)





推理的过程: 证明

- 证明指 α 和 β 之间的蕴涵关系的***演示证明***
- 方法 1: ***模型检查 (model-checking)***
  - 对于每一个可能的世界里, 如果 α 为真 ， 那么确认 β 也真
  - 有限多的世界里可行（比如命题逻辑）
- 方法 2: ***定理证明 (theorem-proving)***
  - 搜寻一系列的证明步骤 (应用 推理规则(***inference rules)***) 从 α 引导到 β
  - 例如, 从 $P \wedge (P \Rightarrow Q)$, 推理出 Q 通过 肯定前件式推理(***Modus Ponens***)
- ***合理的(Sound)*** 算法: 所有被推理证明出来的，实际上也都是被蕴涵的
- ***完全的(Complete)*** 算法: 所有被蕴涵的（句子）， 都可以被推理证明出来

问题

- 完全的 逻辑推理算法和 完全的搜索算法 之间的关联是什么？
- 回答 1: 都含有 “完全的…算法”
- 回答 2: 都可用来求解任何相关的 可解决解答的 问题
- 回答 3: 推理可以建成一个搜索问题
  - 初始状态: KB（知识库） 包含 α
  - 行动: 应用任何可以和 KB 相匹配的推理规则, 并添加结论句子
  - 目标检测: KB 包含 β （KB |= β ）

因此， 任何一个完全的 搜索算法 产生一个完全的推理算法

命题逻辑：语法

- 给定: 一组 命题字符 {X_1,X_2,…, X_n ,P, Q, R, North,…} （可为真或假）
  - (True 和 False 也是，真值固定)
- X_i 是一个句子（原子句）
- 复杂句
  - 如果 α 是句子，那么 $\neg\alpha$ 是一个句子 （否定）
  - 如果α 和 β是句子，那么$\alpha \wedge \beta$是一个句子 （结合）
  - 如果α 和 β是句子，那么$\alpha \vee \beta$是一个句子 （分离）
  - 如果α 和 β是句子，那么$\alpha \Rightarrow \beta$是一个句子 （暗含,或条件的if …then）
  - 如果α 和 β是句子，那么$\alpha \Leftrightarrow \beta$是一个句子 （双向条件的 if and only if ）
  - 逻辑连接符，和（）的组合
- 文字(literal): 原子语句和否定的原子语句
- 没有其他样式的句子!

命题逻辑: 语义

- 给定一个模型（model），决定一个句子的真值
- 真值表

P	Q	$\neg P$	$P \wedge Q$	$P \vee Q$	$P \Rightarrow Q$	$P \Leftrightarrow Q$
false	*false*	*true*	*false*	*false*	*true*	*true*
false	*true*	*true*	*false*	*true*	*true*	*false*
true	*false*	*false*	*false*	*true*	*false*	*false*
true	*true*	*false*	*true*	*true*	*true*	*true*

命题逻辑(Propositional Logic): 语义

function PL-TRUE?(α,model) **returns** true or false

if α 是一个命题符号 **then return** Lookup(α, model)

if Op(α) = ¬ **then return** not(PL-TRUE?(Arg1(α),model))

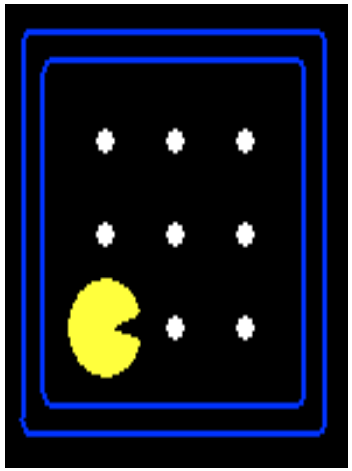
if Op(α) = ∧ **then return** and(PL-TRUE?(Arg1(α),model),
PL-TRUE?(Arg2(α),model))

，等。

(也叫做 “语法上的递归”)

举例：部分可观察的 Pacman

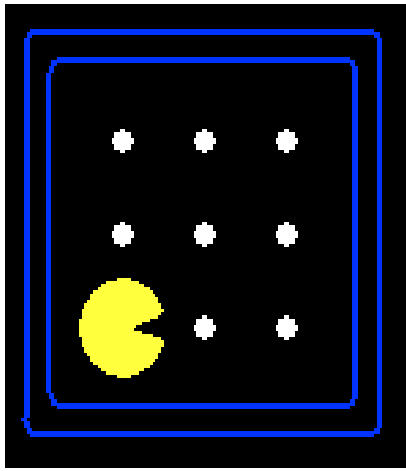
- Pacman 仅感知局部的墙/障碍物
- 问题建立: 我们需要哪些变量?
  - Pacman的位置
    - At_1,1_0 (Pacman 在[1,1] 在时刻 0)　At_3,3_1　等
  - 墙的位置
    - Wall_0,0　Wall_0,1　…
  - 感知到的
    - Blocked_W_0 (向西被墙阻挡，在时刻 0) 等.
  - 行动
    - W_0 (Pacman 向西移动，在时刻 0), E_0 等.



- NxN 个世界，　T 个时刻 => $N^2T + N^2 + 4T + 4T = O(N^2T)$ 变量数
- 2^{N^2T} 可能的世界（模型）！，　N=10, T=100 => 10^{32}

传感器模型

- 描述导致 Pacman的感知变化的事实
- 如果在西面 *有* 一堵墙, 那么Pacman 感知到它
- 如果在时刻 t 他在位置 x,y 并且 有一堵墙在位置 x-1,y, 则Pacman 在时刻 t 感知到一堵墙在他的西面
  - $At\_1,1\_0 \wedge Wall\_0,1 \Rightarrow Blocked\_W\_0$
  - $At\_1,1\_1 \wedge Wall\_0,1 \Rightarrow Blocked\_W\_1$
  - ….
  - $At\_3,3\_9 \wedge Wall\_3,4 \Rightarrow Blocked\_N\_9$



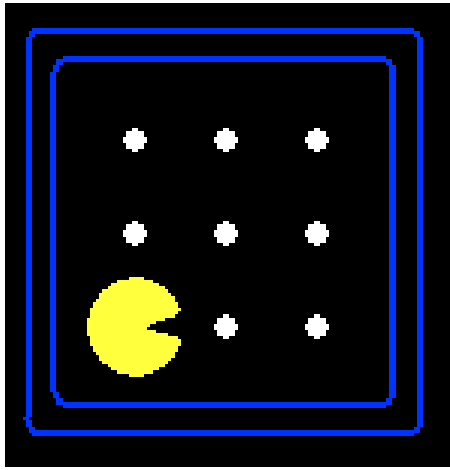
效率低……；如果左边为假，右边是真还是假？

问题在哪???

- 一个暗含（=>）句子，当左边为假时，没有指明右边的状态
  - 苹果落地 ⇒ 牛顿发现万有引力 是真，当 苹果落地 为假
  - 是否“*如果你在截至日期后交作业那么作业将会被拒收*”暗示“*如果你在截至日期之前交作业那么作业将不会被拒收*”?
- 我们需要指出什么时候感知变量为真，何时为假
- 某个感知 ⇔ <世界中的某个条件 >

传感器模型

- 描述导致 Pacman的感知变化的事实
- Pacman 在时刻 t 感知到一堵墙 在西面，*当且仅当(if and only if)* 他在位置 x,y 并且 有一堵墙在位置 x-1,y
  - Blocked_W_0 $\Leftrightarrow$ ((At_1,1_0 $\wedge$ Wall_0,1) v
    (At_1,2_0 $\wedge$ Wall_0,2) v
    (At_1,3_0 $\wedge$ Wall_0,3) v ….)



(根据这些公理(axioms), 再加上一系列的位置和感知信息 , Pacman 可以推理出完整的地图 ….)

推理方法（之前提过）

- 方法 1: 模型检查 ***model-checking***
  - 对于每个可能的世界, 如果 α 为真 则 β 亦真
- 方法 2: 定理证明 ***theorem-proving***
  - 搜索一系列的证明步骤 (应用推理规则 ***inference rules***) 从 α 导致到 β
- ***合理的（Sound）*** 算法:所有被推理证明出来的，实际上也都是被蕴涵的
- ***完全的（Complete）*** 算法: 所有被蕴涵的都是能够被推导证明出来的

逻辑上的一致性

$$
\begin{aligned}
(\alpha \wedge \beta) &\equiv (\beta \wedge \alpha) && \text{commutativity of } \wedge \\
(\alpha \vee \beta) &\equiv (\beta \vee \alpha) && \text{commutativity of } \vee \\
((\alpha \wedge \beta) \wedge \gamma) &\equiv (\alpha \wedge (\beta \wedge \gamma)) && \text{associativity of } \wedge \\
((\alpha \vee \beta) \vee \gamma) &\equiv (\alpha \vee (\beta \vee \gamma)) && \text{associativity of } \vee \\
\neg(\neg\alpha) &\equiv \alpha && \text{double-negation elimination} \\
(\alpha \Rightarrow \beta) &\equiv (\neg\beta \Rightarrow \neg\alpha) && \text{contraposition} \\
(\alpha \Rightarrow \beta) &\equiv (\neg\alpha \vee \beta) && \text{implication elimination} \\
(\alpha \Leftrightarrow \beta) &\equiv ((\alpha \Rightarrow \beta) \wedge (\beta \Rightarrow \alpha)) && \text{biconditional elimination} \\
\neg(\alpha \wedge \beta) &\equiv (\neg\alpha \vee \neg\beta) && \text{De Morgan} \\
\neg(\alpha \vee \beta) &\equiv (\neg\alpha \wedge \neg\beta) && \text{De Morgan} \\
(\alpha \wedge (\beta \vee \gamma)) &\equiv ((\alpha \wedge \beta) \vee (\alpha \wedge \gamma)) && \text{distributivity of } \wedge \text{ over } \vee \\
(\alpha \vee (\beta \wedge \gamma)) &\equiv ((\alpha \vee \beta) \wedge (\alpha \vee \gamma)) && \text{distributivity of } \vee \text{ over } \wedge
\end{aligned}
$$

简单的通过定理进行推断证明的过程: 前向推理(Forward chaining)

- 应用肯定前件式推理(Modus Ponens) 产生新的事实(单一正字符构成的语句):
  - 给定 $X_1 \wedge X_2 \wedge \ldots X_n \Rightarrow Y$ 和 $X_1, X_2, \ldots, X_n$
  - 推理出 Y
- 前向推理持续应用这个规则, 不断添加新的事实, 直到没有可添加的为止
- 要求 KB（知识库） 只包含 **限定子句(*definite clauses*)**:
  - 一组分离(disjunction)的文字(literals),只有一个是正的（其他都含否定符）; 可转成一下形式
  - (字符的结合(conjunction)) $\Rightarrow$ 字符; 或
  - 一个单一的字符 (注意 X 相当于 True $\Rightarrow$ X)

前向推理算法(Forward chaining)

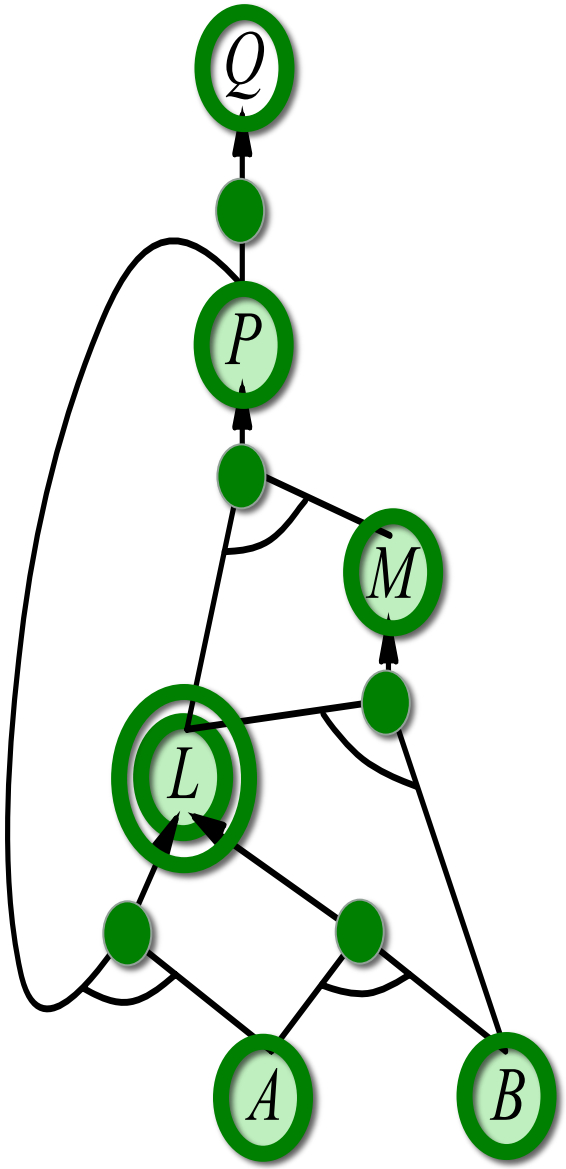
```
function PL-FC-ENTAILS?(KB, q) returns true or false
  count ← 一个表，count[c] 是 子句 c 的前提中的还未知的字符数量
  inferred ← 一个表，inferred[s] 初始化为 false 对于所有字符 s
  agenda ← 一个字符队列, 初始化为 KB 里 (为真的)所有字符
  while agenda is not empty do
      p ← Pop(agenda)
      if p = q then return true
      if inferred[p] = false then
          inferred[p]←true
          for each 子句 c in KB where p 在 c的前提里 do
              减一 count[c]
              if count[c] = 0 then add c的结论 to agenda
  return false
```

前向推理，举例: 证明 Q（被蕴涵）

子句	COUNT	INFERRED
• P ⇒ Q	1 1 0	A false true
• L ∧ M ⇒ P	2 1 1 0	B false true
• B ∧ L ⇒ M	2 1 1 0	L false true
• A ∧ P ⇒ L	2 1 1 0	M false true
• A ∧ B ⇒ L	2 1 1 0	P false true
• A	2 1 1 0	Q false true
• B	0	
	0	

AGENDA

~~A B L M P Q~~



前向推理(Forward Checking)的性质

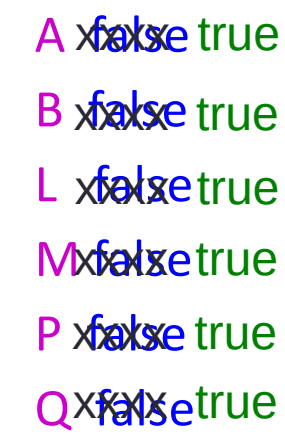
- 定理: FC 是合理的(sound) 和 完全的(complete)，对于限定子句(definite-clause)组成的KBs
- 合理性: 遵循于肯定前件式（ Modus Ponens）的合理性
- 完全性证明:

1. 假设FC 达到了一个固点，即 没有新的原子语句被推导出
2. 最终的 *inferred* 表可以被考虑成一个模型 ***m***, 即字符被赋给了 true/false 值
3. 在原始的 KB 中的每一个子句在 ***m*，** 为真

   证明: 假定一个子句 $a_1 \wedge \ldots \wedge a_k \Rightarrow b$ 在 ***m*** 中为假

   那么 $a_1 \wedge \ldots \wedge a_k$ 必为真在 ***m*** 并且 b 为假 在 ***m***

   如此说明算法还未达到一个固点!（与假设矛盾）
4. 因此 ***m*** 是 KB 的一个模型 （KB 在 m 里为真）
5. 如果 KB ⊨ q, q 则在KB中的每一个模型里为真, 包括 ***m*；** 即q可以被算法推导出来

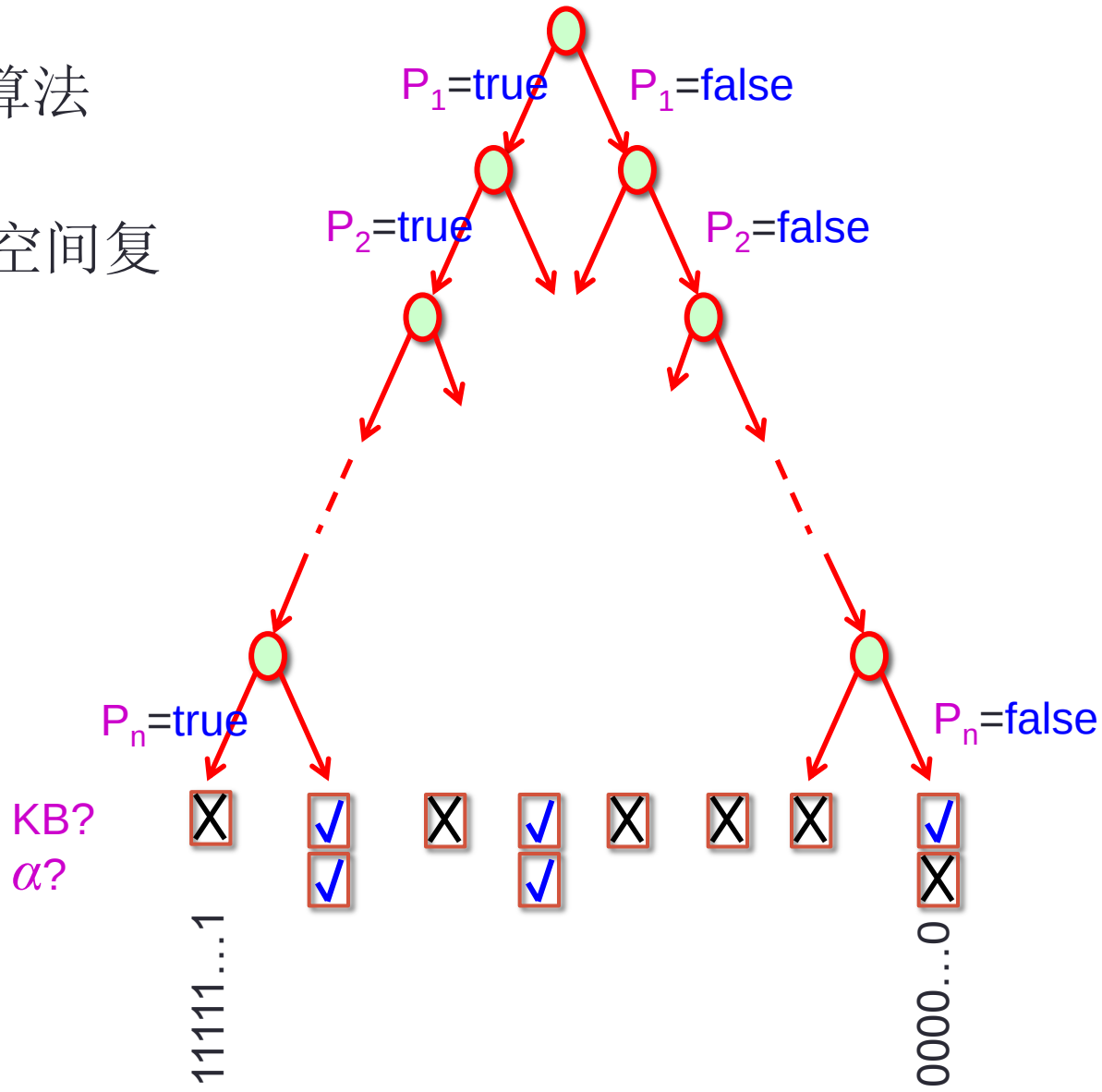


简单的模型检查(model checking)

```
function TT-ENTAILS?(KB, α) returns true or false
    return TT-CHECK-ALL(KB,α,symbols(KB) U symbols(α),{})
function TT-CHECK-ALL(KB,α,symbols,model) returns true or false
    if empty?(symbols) then
        if PL-TRUE?(KB,model) then return PL-TRUE?(α,model)
        else return true
    else
        P ← first(symbols)
        rest ← rest(symbols)
        return and (TT-CHECK-ALL(KB,α,rest,model ∪ {P = true})
                    TT-CHECK-ALL(KB,α,rest,model ∪ {P = false }))
```

简单的模型检查, 继续

- 深度优先，类似于回溯算法(backtracking)的递归
- $O(2^n)$ 时间复杂度, 线性空间复杂度
- 可以有更高效的算法!



可满足性 和 导出(蕴涵)

- 一个语句是 ***可满足的*** ， 如果它至少在一个世界里为真 (参见 CSPs!)
- 假设我们有一个超高效的 SAT solver; 我们如何能用它来测试蕴涵关系?
  - 假定 $\alpha \models \beta$
  - 那么 $\alpha \Rightarrow \beta$ 为真 在所有世界（演绎公理）
  - 因此 $\neg(\alpha \Rightarrow \beta)$ 为假 在所有世界
  - 因此 $\alpha \wedge \neg\beta$为假 在所有世界, i.e., 不可满足的(unsatisfiable)
- 所以, 把否定的结论添加到 所知道的语句里, 测试其不可满足性 (un)satisfiability; 也叫 归谬法(reductio ad absurdum)
- 高效的 SAT solvers 需要 合取范式(***conjunctive normal form)***

合取范式(CNF)

- 每个句子都能表达成一个子句
- 每个子句都是 文字(正的或否[illegible]符)的析取
- 到 CNF的标准变换:
  - At_1,1_0 ⇒ (Wall_0,1 ⇔ Blocked_W_0)
  - At_1,1_0 ⇒ ((Wall_0,1 ⇒ Blocked_W_0) ∧ (Blocked_W_0 ⇒Wall_0,1))
  - ¬At_1,1_0 v ((¬Wall_0,1 v Blocked_W_0) ∧ (¬Blocked_W_0 v Wall_0,1))
  - (¬At_1,1_0 v ¬Wall_0,1 v Blocked_W_0) ∧ (¬At_1,1_0 v ¬Blocked_W_0 v Wall_0,1)

替换双向条件，用两个暗示条件

替换 α ⇒ β 用 ¬α v β

分配 v 到 ∧

高效的 SAT solvers

- DPLL (Davis-Putnam-Logemann-Loveland) 是 现代SAT求解器的核心算法
- 本质上是一个对模型的回溯搜索，和一些额外的技术:
  - ***提早终止***: 如果
    - 所有子句都被满足; e.g., $(A \vee B) \wedge (A \vee \neg C)$ 被满足，通过 {A=true}
    - 任何一个子句为假; e.g., $(A \vee B) \wedge (A \vee \neg C)$ 为假，当 {A=false,B=false}
  - ***纯净的文字（字符）***: 如果一个字符在剩下所有未满足的子句里的符号都是统一的，那么赋给这个字符那个值
    - 例如, A 是纯净的，并且正号的 $(A \vee B) \wedge (A \vee \neg C) \wedge (C \vee \neg B)$ 所以赋给 true
  - ***单元子句***: 如果一个子句只剩下一个单一的文字字符，那么给这个字符赋值使之满足该子句
    - 例如, 如果 A=false, $(A \vee B) \wedge (A \vee \neg C)$ becomes $(\text{false} \vee B) \wedge (\text{false} \vee \neg C)$, i.e. $(B) \wedge (\neg C)$
    - 满足单元子句的过程中经常会导致进一步的传递，产生新的单元子句。

DPLL 算法

```
function DPLL(子句集,字符集,模型) returns true or false
  if every 子句 in 子句集 is true in 模型 then return true
  if 某个 子句 in 子句集 is false in 模型 then return false
  P,value ←FIND-PURE-SYMBOL(字符集,子句集,模型)
  if P is non-null then return DPLL(子句集, 字符集–P, 模型∪{P=value})
  P,value ←FIND-UNIT-CLAUSE(子句集,模型)
  if P is non-null then return DPLL(子句集, 字符集–P, 模型∪{P=value})
  P ← First(字符集); rest ← Rest(字符集)
  return or(DPLL(子句集,rest,模型∪{P=true}),
            DPLL(子句集,rest,模型∪{P=false}))
```

效率

- DPLL的简单实现: 求解 ~100 变量
- 额外技巧:
  - 变量和值的选取排序 (参见 CSPs)
  - 分治法（divide and conquer）
  - 记录下 无法求解的子情况，作为额外的子句，用来避免重蹈覆辙
  - 索引和 增量计算技巧，使得DPLL 算法的每一步都是高效的 (通常 O(1))
    - 索引子句中每个变量（字符）的符号（正或是否定的）
    - 变量赋值过程中持续记录已满足的子句数量
    - 持续记录每个子句中剩余的文字符号的数量
- DPLL的真正实现: 可以求解 ~10,000,000 变量

SAT 求解器在现实中的应用

- 电路验证: 超大规模集成电路 是否计算正确?
- 软件验证: 程序是否计算正确的结果?
- 软件综合: 哪些程序计算正确结果?
- 协议验证: 这个安全协议能否被攻破?
- 协议合成: 哪些协议对于这个任务是安全的?
- 规划: 智能体的行为规划***?***

总结

- 一种可能的智能体框架: 知识 + 推理
- 逻辑 提供了一种对知识进行编码的正规方法
  - 一个逻辑的定义: 语法, 可能世界的集合, 真值条件
- 逻辑推理计算句子间的蕴涵关系

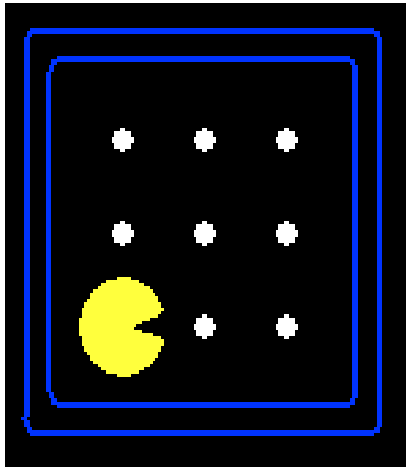
人工智能导论：逻辑型的智能体

一个基于知识的智能体

```
function KB-AGENT(percept) returns 一个行动
  内部记录：KB, 知识库
            t, 整数, 初始为 0
  TELL(KB, MAKE-PERCEPT-SENTENCE(percept, t))
  action ← ASK(KB, MAKE-ACTION-QUERY(t))
  TELL(KB, MAKE-ACTION-SENTENCE(action, t))
  t←t+1
  return action
```

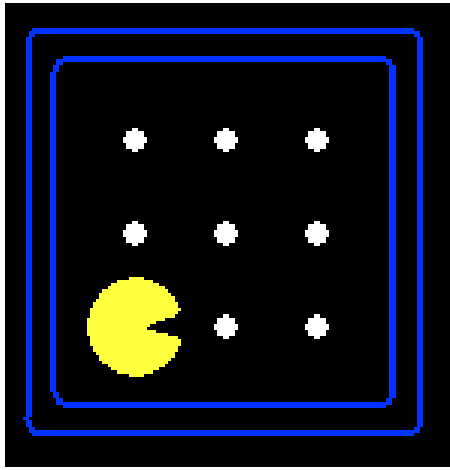
回忆: 部分可观察的Pacman 中的变量

- Pacman 只观察到局部的墙
- 问题建立: 我们需要的变量?
  - Pacman的 位置
    - At_1,1_0 (Pacman 在位置 [1,1] 在时刻 0) At_3,3_1 等
  - 墙的位置
    - Wall_0,0 Wall_0,1 等
  - 感知
    - Blocked_W_0 (在时刻 0，向西有墙阻挡) 等.
  - 行动
    - W_0 (Pacman 向西移动 在时刻 0), E_0 等.



回忆: 传感模型

- 描述引起 Pacman的感知发生变化的事实
- Pacman 感觉到向西有一面墙在时刻 t， *当且仅当* 他在位置 x,y 并且 有一面墙在位置 x-1,y
  - Blocked_W_0 $\Leftrightarrow$ ((At_1,1_0 $\wedge$ Wall_0,1) v
    (At_1,2_0 $\wedge$ Wall_0,2) v
    (At_1,3_0 $\wedge$ Wall_0,3) v ….)



转换模型 (transition model)

- 每个 ***状态变量*** 在每个时刻如何获得它的值?
- 部分可感知的Pacman里的状态变量 是 At_x,y_t , 例如, At_3,3_17
- 一个状态变量获得它的值，根据 ***后继状态公理 (successor-state axiom)***
  - $X_t \Leftrightarrow [X_{t-1} \wedge \neg(\text{某个 action}_{t-1} \text{ 为 false})] \vee [\neg X_{t-1} \wedge (\text{某个 action}_{t-1} \text{ 为 true})]$
- 对于 Pacman 的位置:
  - At_3,3_17 $\Leftrightarrow$ [At_3,3_16 $\wedge$ $\neg$((¬Wall_3,4 $\wedge$ N_16) v (¬Wall_4,3 $\wedge$ E_16) v …)]
    v [¬At_3,3_16 $\wedge$ ((At_3,2_16 $\wedge$ ¬Wall_3,3 $\wedge$ N_16) v (At_2,3_16 $\wedge$ ¬Wall_3,3 $\wedge$ N_16) v …)]

初始状态

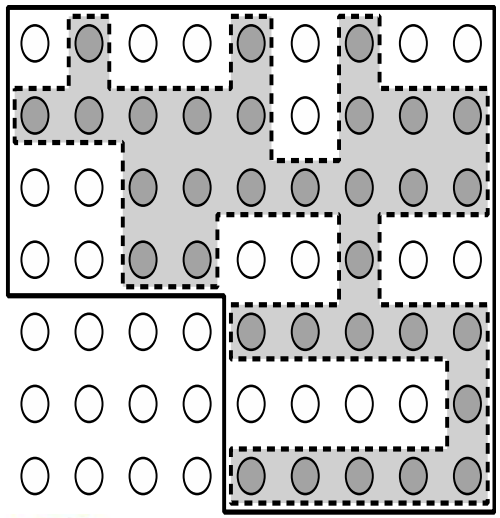
- 智能体可能知道它的初始位置:
  - At_1,1_0
- 或者, 它可能不知道:
  - At_1,1_0 v At_1,2_0 v At_1,3_0 v … v At_3,3_0
- 我们也需要一个 ***值域约束*** – 不能同时在两个不同的位置!
  - ¬(At_1,1_0 ∧ At_1,2_0) ∧ ¬(At_1,1_0 ∧ At_1,3_0) ∧ …
  - ¬(At_1,1_1 ∧ At_1,2_1) ∧ ¬(At_1,1_1 ∧ At_1,3_1) ∧ …
  - …

状态估计

- 回忆智能体在部分可观察情况下的 ***信念状态(belief state)*** 定义:
  - 给定行动和当前的感知，相符合的世界状态的集合
  - ***状态估计*** 是指保持(预测/更新)当前的信念状态
- 对于一个逻辑型的智能体, 计算在当前状态下哪些变量为真，只不过是一个逻辑推理的问题
  - 例如, 询问是否 KB $\wedge$ <actions> $\wedge$ <percepts> $\models$ Wall_2,2
  - 简单但效率低: 每一步的推理涉及到一个智能体整个行动感知的历史

状态估计，继续

- 一个更 “积极主动的” 状态估计形式:
  - 在每个行动和感知以后
    - 对每个状态变量 X_t
      - 如果 X_t 是被蕴涵的, 则加到 KB
      - 如果 $\neg X_t$ 是被蕴涵的, 则加到 KB
- 对于准确的状态评估是否这就足够?
  - 不是! 可能的情况是 X_t 或 $\neg X_t$ 都不被蕴涵, 并且 Y_t 或 $\neg Y_t$ 也都不被蕴涵, 但是某个约束, 例如, $X_t \vee Y_t$, ***是*** 被蕴涵的
    - 例如: 初始不确定性的智能体的位置
- 普遍来讲，完美的状态估计是很难达到的



可满足(satisfiability)来解规划(Planning)问题

- 给定一个超高效的 SAT 求解器，我们能用它来规划智能体的行动吗?
- 是的, 对于完全可观察的, 决定性的环境: 规划问题是可解的当且仅当 存在某个对行动等可满足的赋值
- T = 1 到无穷, 按以下内容构建知识库 (KB)，并运行SAT solver:
  - 初始状态, 值域约束
  - 截至到时刻 T的转换模型语句(包括后继状态转换公理，对于所有可能的行动)
  - 目标(Goal) 在时刻 T 达到
  - ***先决条件公理***：（行动发生的先决条件）$At\_1,1\_0 \wedge N\_0 \Rightarrow \neg Wall\_1,2$ 等
  - ***行动排他公理***: （避免同时采取多个行动）$\neg(N_0 \wedge W_0) \wedge \neg(N_0 \wedge S_0) \wedge ..$等

SAT-Plan: 找到行动规划

- 找到最短行动规划路径

```
function SATPLAN(init, transition, goal, T_max) returns solution or failure
  inputs: init, transition, goal, constitute a description of the problem
          T_max, an upper limit for plan length

  for t = 0 to T_max do
    cnf ← TRANSLATE-TO-SAT(init, transition, goal, t)
    model ← SAT-SOLVER(cnf)
    if model is not null then
      return EXTRACT-SOLUTION(model)
  return failure
```

总结

- 声明法/陈述法(declarative approach) 构建智能体的框架
  - 知识库里是陈述语句（包括公理，感知到的事实）
  - 逻辑推理– 事实被蕴涵的推理证明，规划智能体行动
- 现代超高效的SAT 求解器，使得此法可在实践中应用
- 弱点：语句表达
  - 例如“对于每个时刻 t”，“对于每个方块位置[x,y]”
  - 一阶逻辑(first order logic) 提高了语句的表达性；其逻辑推理方法与命题逻辑的方法一致