Some modeling paradigms

- State-based models: search problems, games
  - Applications: routing finding, game playing, etc.
  - Think in terms of **states, actions, and costs**
- Variable-based models: CSPs, Bayesian networks
  - Applications: scheduling, medical diagnosis, etc.
  - Think in terms of **variables and potentials**
- Logic-based models: propositional logic, first-order logic
  - Applications: theorem proving, verification, reasoning
  - Think in terms of **logical formulas and inference rules**

Logical Agents
逻辑智能体

Chapter 7

Motivation: smart personal assistant

…

逻辑智能体

- 逻辑智能体：基于知识的智能体
- 知识和推理的重要性
  - 部分可观察的环境
  - 自然语言理解
  - 基于知识的智能体的灵活性

Two goals of logic

- Represent knowledge about the world

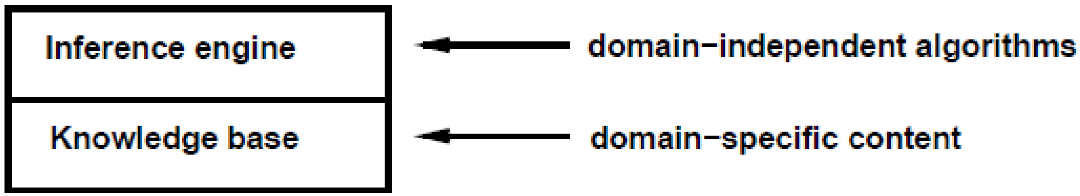
- Reason with that knowledge

Outline

- Knowledge-based agents
- Wumpus world
  - 关于基于知识的智能体运转的例子
- Logic in general — models and entailment（蕴涵）
- Propositional (Boolean) logic 命题逻辑
- Equivalence, validity, satisfiability 等价、合法性和可满足性
- Inference rules and theorem proving
  - forward chaining 前向链接
  - backward chaining 反向链接
  - resolution 归结

Motivation: smart personal assistant

Knowledge bases



Knowledge base（知识库） = set of sentences in a formal language

将新语句添加到知识库——
Declarative approach to building an agent (or other system):
TELL （告诉） it what it needs to know

查询目前所知内容——
Then it can ASK（询问） itself what to do — answers should follow from the KB

Agents can be viewed at the knowledge level（知识层）
i.e., what they know, regardless of how implemented

Or at the implementation level（实现层）
i.e., data structures in KB and algorithms that manipulate them

A simple knowledge-based agent

```
function KB-AGENT( percept) returns an action
    static: KB, a knowledge base
            t, a counter, initially 0, indicating time

    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
```

- TELL→ASK→TELL
- 表示语言的细节隐含于MAKE-PERCEPT-SENTENCE和MAKE-ACTION-QUERY中
- 推理机制的细节隐藏于TELL和ASK中

A simple knowledge-based agent

```
function KB-AGENT(percept) returns an action
    static: KB, a knowledge base
            t, a counter, initially 0, indicating time

    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
```

The agent must be able to:

Represent states, actions, etc.	表示状态和行为
Incorporate new percepts	加入新的感知信息
Update internal representations of the world	更新关于世界的状态表示
Deduce hidden properties of the world	推导关于世界的隐藏信息
Deduce appropriate actions	推导应采取的合适的行为

Outline

- Knowledge-based agents
- Wumpus world
  - 关于基于知识的智能体运转的例子
- Logic in general — models and entailment（蕴涵）
- Propositional (Boolean) logic 命题逻辑
- Equivalence, validity, satisfiability 等价、合法性和可满足性
- Inference rules and theorem proving
  - forward chaining 前向链接
  - backward chaining 反向链接
  - resolution 归结

Wumpus World PEAS description

Performance measure

gold +1000, death -1000

-1 per step, -10 for using the arrow

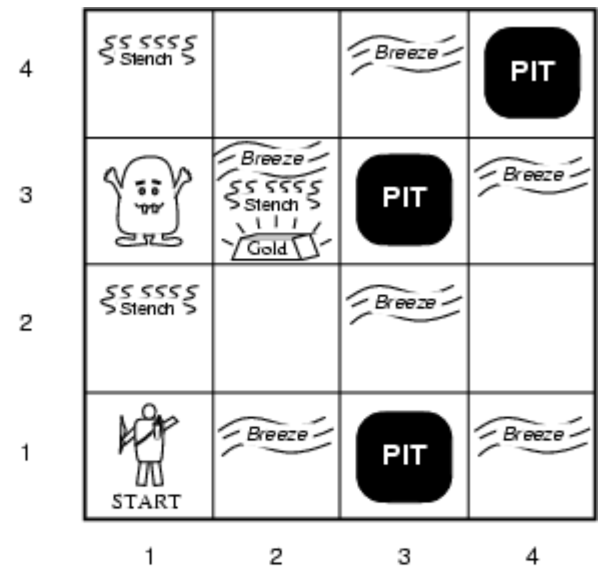
Environment

4×4网格

智能体初始在[1,1]，面向右方

金子和wumpus在[1,1]之外随机均匀分布

[1,1]之外的任意方格是陷阱的概率是0.2



Actuators Left turn, Right turn,

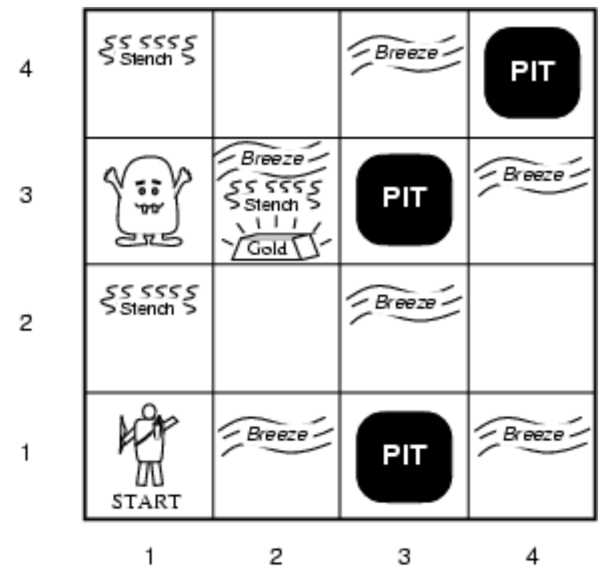
Forward, Grab, Shoot

- 智能体可向前、左转或右转
- 智能体如果进入一个有陷阱或者活着的wumpus的方格，将死去。
- 如果智能体前方有一堵墙，那么向前移动无效
- Grab：捡起智能体所在方格中的一个物体
- Shoot：向智能体所正对方向射箭（只有一枝箭）

Wumpus World PEAS description

Sensors

- Smell: 在wumpus所在之处以及与之直接相邻的方格内，智能体可以感知到臭气。
- Breeze: 在与陷阱直接相邻的方格内，智能体可以感知到微风。
- Glitter(发光): 在金子所处的方格内，智能体可以感知到闪闪金光。
- 当智能体撞到墙时，它感受到撞击。
- 当wumpus被杀死时，它发出洞穴内任何地方都可感知到的悲惨嚎叫。



以5个符号的列表形式将感知信息提供给智能体，例如(stench, breeze, none, none, none)。

Wumpus world characterization

Observable??

Wumpus world characterization

Observable?? No — only local perception

Deterministic??

Wumpus world characterization

Observable?? No — only local perception

Deterministic?? Yes — outcomes exactly specified

Episodic??

Wumpus world characterization

Observable?? No — only local perception

Deterministic?? Yes — outcomes exactly specified

Episodic?? No — sequential at the level of actions

Static??

Wumpus world characterization

Observable?? No — only local perception

Deterministic?? Yes — outcomes exactly specified

Episodic?? No — sequential at the level of actions

Static?? Yes — Wumpus and Pits do not move

Discrete??

Wumpus world characterization

Observable?? No — only local perception

Deterministic?? Yes — outcomes exactly specified

Episodic?? No — sequential at the level of actions

Static?? Yes — Wumpus and Pits do not move

Discrete?? Yes

Single-agent??

Wumpus world characterization

Observable?? No — only local perception

Deterministic?? Yes — outcomes exactly specified

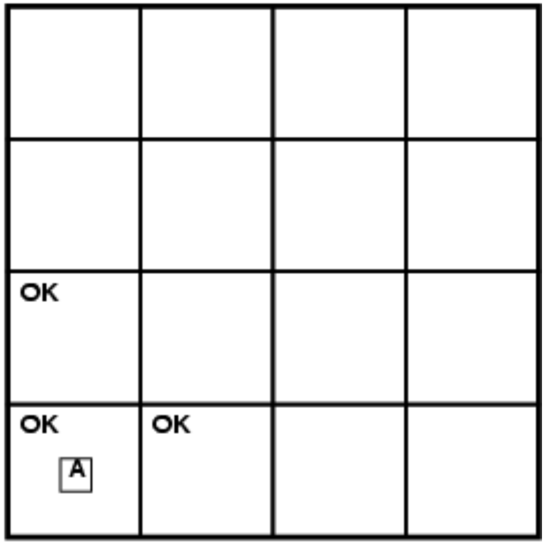
Episodic?? No — sequential at the level of actions

Static?? Yes — Wumpus and Pits do not move

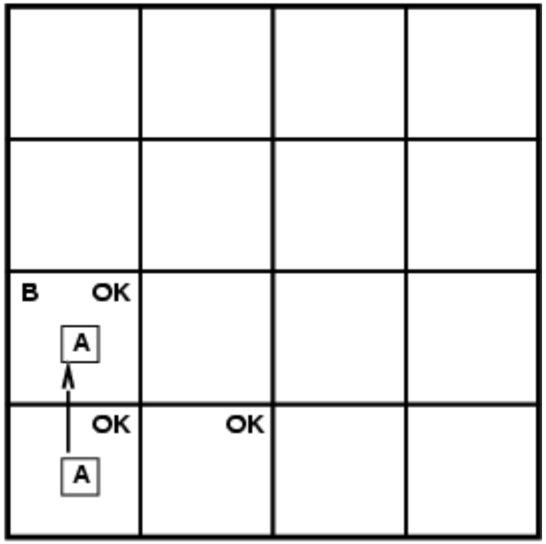
Discrete?? Yes

Single-agent?? Yes — Wumpus is essentially a natural feature

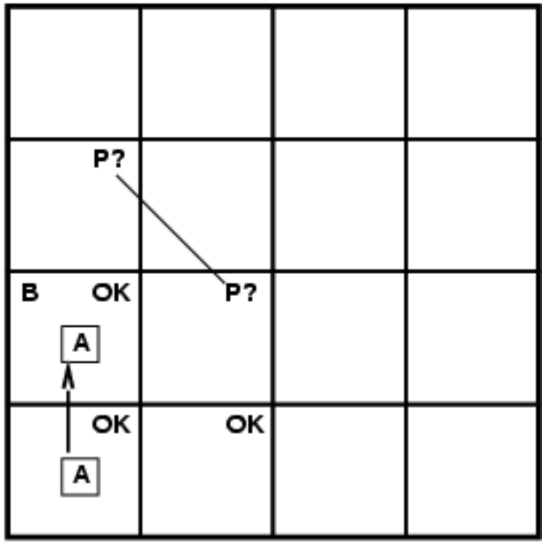
Exploring a wumpus world



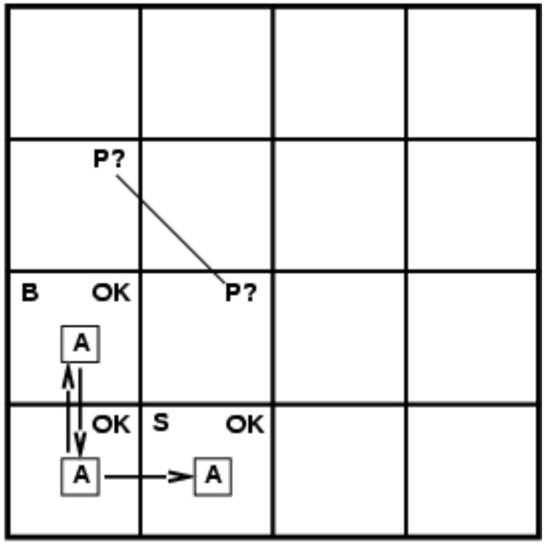
Exploring a wumpus world



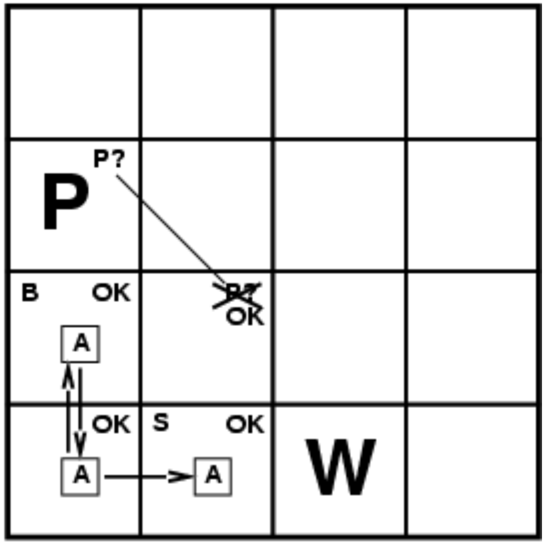
Exploring a wumpus world



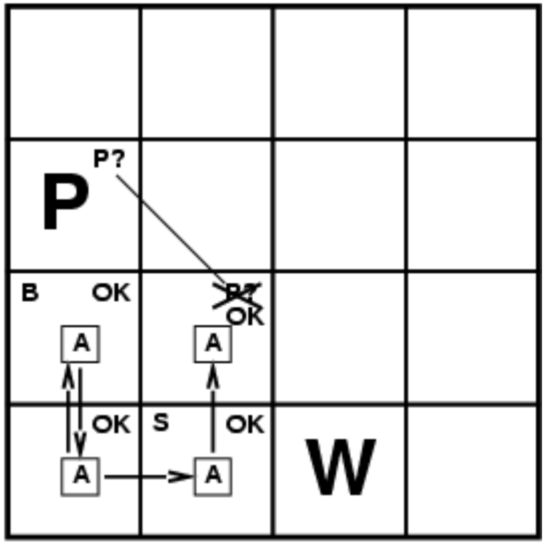
Exploring a wumpus world



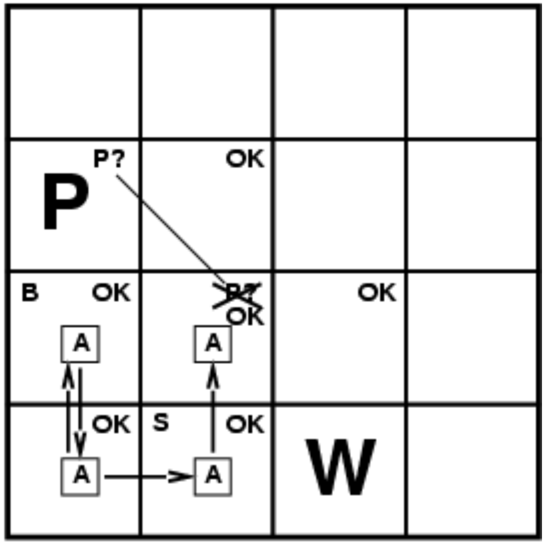
Exploring a wumpus world



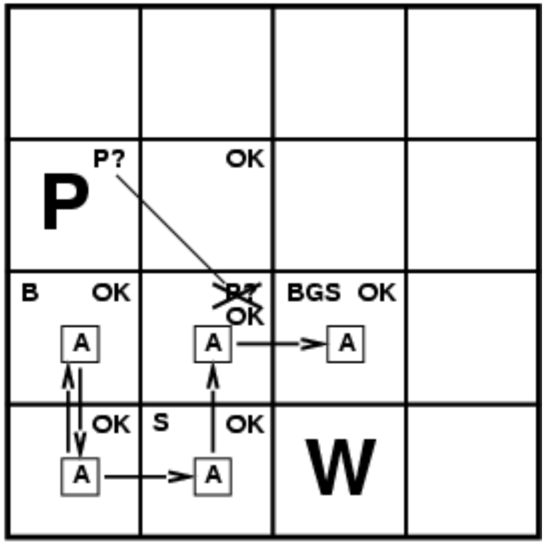
Exploring a wumpus world



Exploring a wumpus world



Exploring a wumpus world



Outline

- Knowledge-based agents
- Wumpus world
  - 关于基于知识的智能体运转的例子
- Logic in general — models and entailment（蕴涵）
- Propositional (Boolean) logic 命题逻辑
- Equivalence, validity, satisfiability 等价、合法性和可满足性
- Inference rules and theorem proving
  - forward chaining 前向链接
  - backward chaining 反向链接
  - resolution 归结

Logic in general

Logics are formal languages for representing information
such that conclusions can be drawn

Syntax（语法） define the sentences in the language

Semantics （语义） define the "meaning" of sentences;
i.e., define truth of a sentence in a world
语义定义了每个语句关于每个可能世界的真值

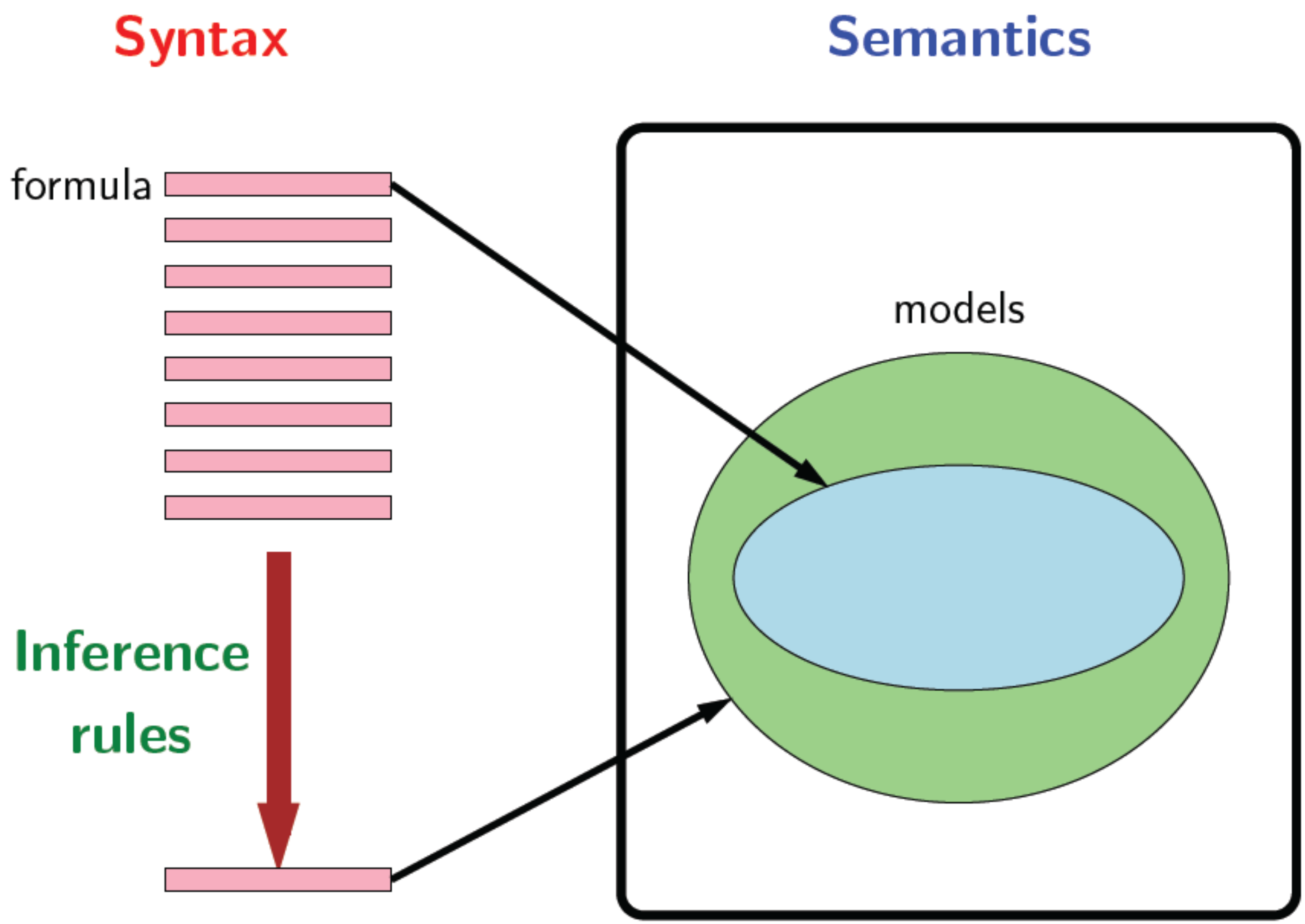
E.g., the language of arithmetic（算术）

$x + 2 \geq y$ is a sentence; $x2 + y >$ is not a sentence

$x + 2 \geq y$ is true if the number $x + 2$ is no less than the number y

$x + 2 \geq y$ is true in a world where $x=7$, $y=1$
$x + 2 \geq y$ is false in a world where $x=0$, $y=6$

Schema for logic



Entailment蕴涵

Entailment(蕴涵) means that one thing follows from another:
一个语句逻辑上跟随另一个语句而出现

$KB \models \alpha$

Knowledge base KB entails sentence α

if and only if

α is true in all worlds where KB is true（在KB为真的每个世界中，α 也为真）

E.g., the KB containing "the Giants won" and "the Reds won" entails "Either the Giants won or the Reds won"

E.g., $x + y = 4$ entails $4 = x + y$

Entailment is a relationship between sentences (i.e., syntax语法) that is based on semantics语义

Models模型

当需要精确描述时，用术语*模型*取代“可能世界”

Logicians typically think in terms of models, which are formally structured worlds with respect to which truth can be evaluated

3 boolean symbols: A, B, C; 8 possible models:

{A : 0; B : 0; C : 0}

{A : 0; B : 0; C : 1}

{A : 0; B : 1; C : 0}

{A : 0; B : 1; C : 1}

{A : 1; B : 0; C : 0}

{A : 1; B : 0; C : 1}

{A : 1; B : 1; C : 0}

{A : 1; B : 1; C : 1}

Models模型

当需要精确描述时，用术语*模型*取代“可能世界”

Logicians typically think in terms of models, which are formally structured worlds with respect to which truth can be evaluated

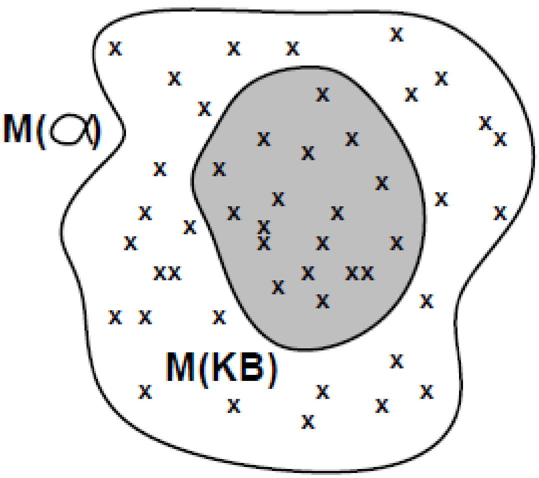
We say *m* is a model of a sentence α if α is true in m

“m是α的一个模型”表示语句α在模型m中为真。

M(α) is the set of all models of α

Then $KB \models \alpha$ if and only if $M(KB) \subseteq M(\alpha)$

在*KB*为真的所有模型中 α 为真



E.g. *KB* = Giants won and Reds won

α = Giants won

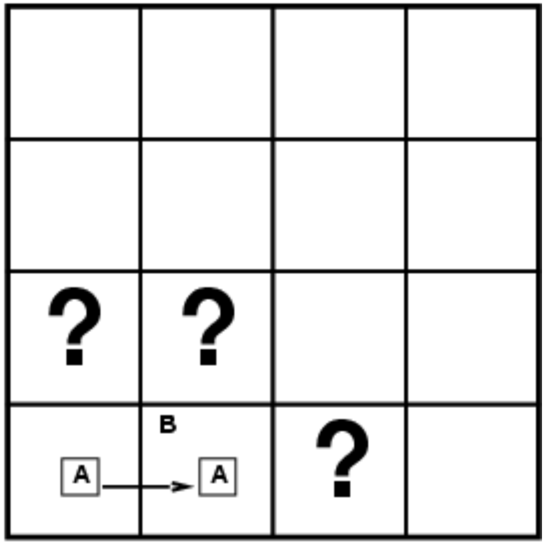
Entailment in the wumpus world

Situation after detecting nothing in [1,1], moving right, breeze in [2,1]—知识库KB

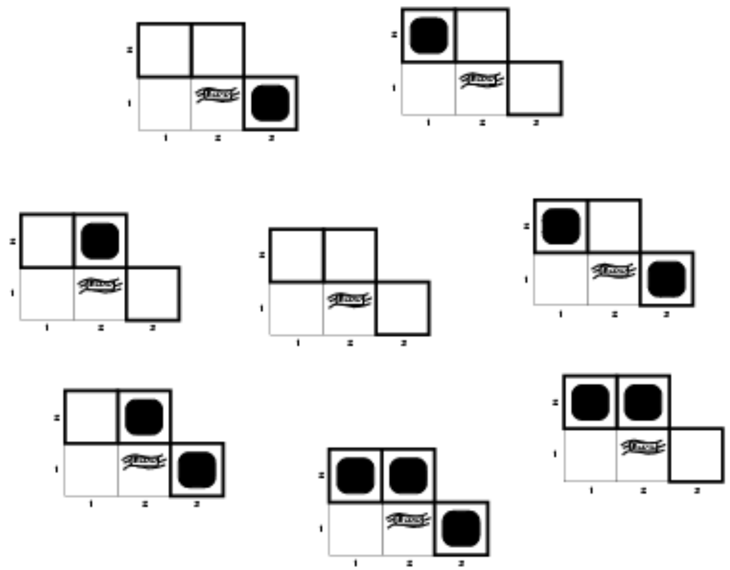
Consider possible models for *KB* assuming only pits

考虑相邻的方格是否包含陷阱

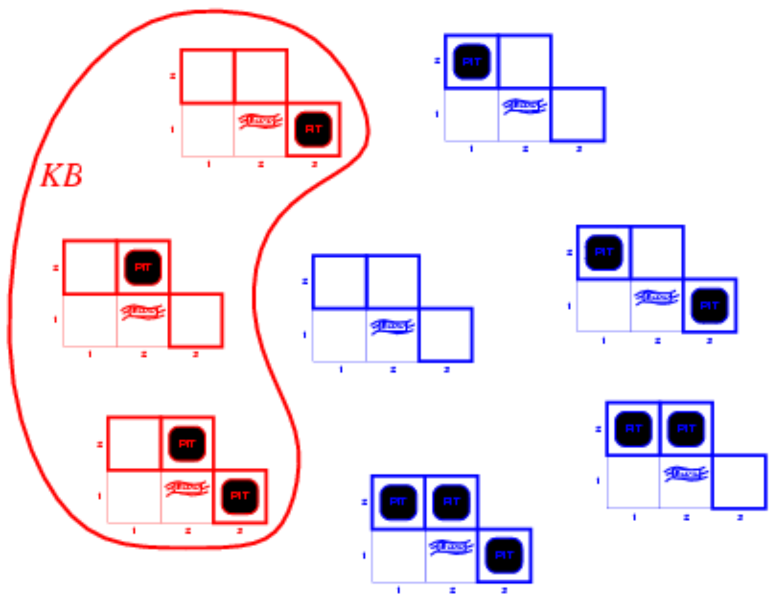
3 Boolean choices $\Rightarrow$ 8 possible models



Wumpus models

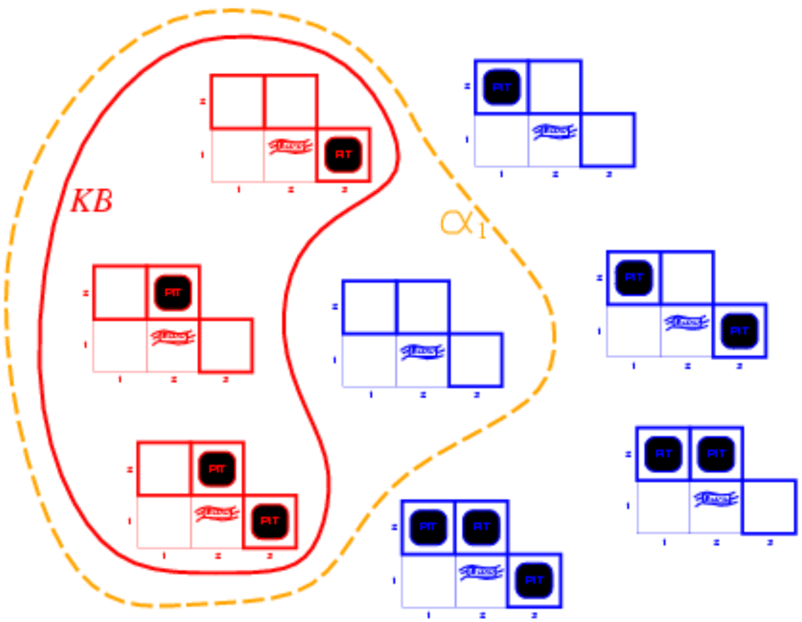


Wumpus models



KB = wumpus-world rules + observations

Wumpus models

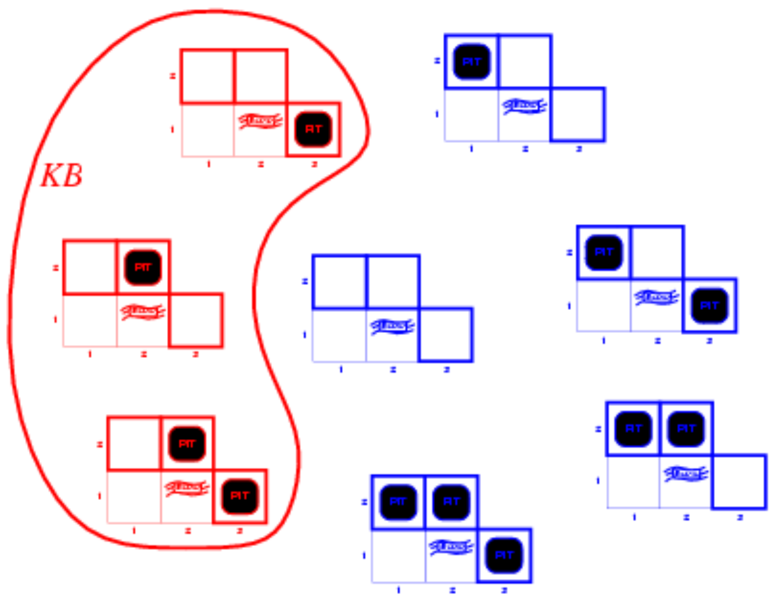


KB = wumpus-world rules + observations

α_1 = "[1,2] is safe", $KB \models \alpha_1$, proved by model checking（模型检验）

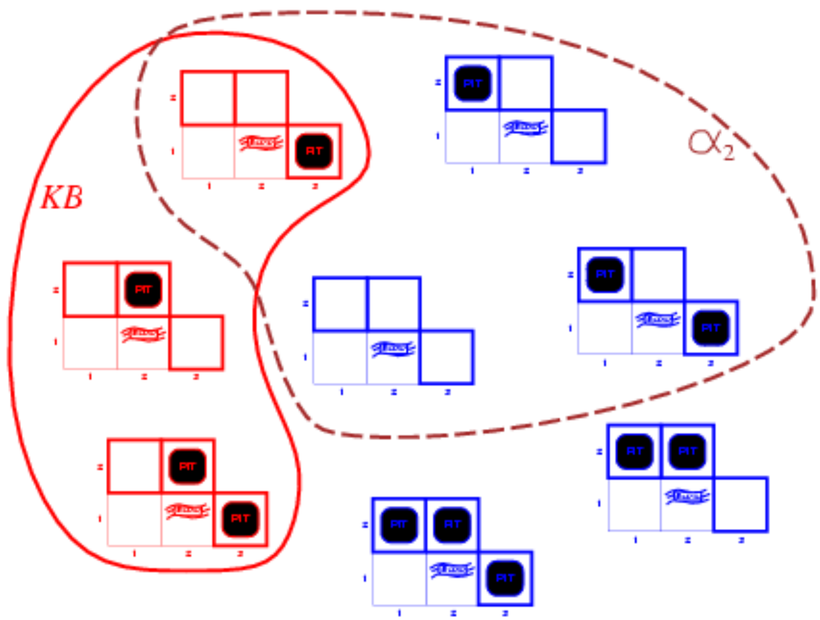
- 在KB为真的每个模型中，α_1 也为真，因此$KB \models \alpha_1$

Wumpus models



KB = wumpus-world rules + observations

Wumpus models



KB = wumpus-world rules + observations

α_2 = "[2,2] is safe", $KB \not\models \alpha_2$

- 在KB为真的某些模型中，α_2 为假，因此$KB \not\models \alpha_2$

Inference推理

$KB \vdash_i \alpha$ =sentence α can be derived from *KB* by procedure *i*
如果推理算法i可以根据KB导出α，我们表示为：$KB \vdash_i \alpha$，读为“i从KB导出α”

Consequences of KB（KB的所有推论集合）are a haystack（干草堆）; α is a needle.
Entailment蕴涵 = needle in haystack; inference推理 = finding it

Soundness（可靠性）—只导出语义蕴涵句: *i* is sound if
whenever $KB \vdash_i \alpha$, it is also true that $KB \models \alpha$
Completeness（完备性）—可以生成任一蕴涵句: *i* is complete if
whenever $KB \models \alpha$, it is also true that $KB \vdash_i \alpha$

Preview: we will define a logic (first-order logic一阶逻辑) which is expressive enough to say almost anything of interest, and for which there exists a sound and complete inference procedure.

That is, the procedure will answer any question whose answer follows from what is known by the *KB*.

Outline

- Knowledge-based agents
- Wumpus world
  - 关于基于知识的智能体运转的例子
- Logic in general — models and entailment（蕴涵）
- Propositional (Boolean) logic 命题逻辑
- Equivalence, validity, satisfiability 等价、合法性和可满足性
- Inference rules and theorem proving
  - forward chaining 前向链接
  - backward chaining 反向链接
  - resolution 归结

Propositional logic: Syntax语法

Propositional logic is the simplest logic – illustrates basic ideas

原子语句:

The proposition symbols（命题符号） P_1, P_2 etc are sentences

代表一个或为真或为假的命题

复合句:

If S is a sentence, $\neg S$ is a sentence (negation非，否定式)

If S_1 and S_2 are sentences, $S_1 \wedge S_2$ is a sentence (conjunction与，合取式)

If S_1 and S_2 are sentences, $S_1 \vee S_2$ is a sentence (disjunction或，析取式)

If S_1 and S_2 are sentences, $S_1 \Rightarrow S_2$ is a sentence (implication蕴涵，蕴涵式)

If S_1 and S_2 are sentences, $S_1 \Leftrightarrow S_2$ is a sentence (biconditional当且仅当，双向蕴涵式)

Propositional logic: Semantics语义

Each model specifies true/false for each proposition symbol
模型简单的固定了每个命题符号的真值

E.g.	$P_{1,2}$	$P_{2,2}$	$P_{3,1}$
	false	true	false

With these symbols, 8 possible models, can be enumerated automatically.

Rules for evaluating truth with respect to a model *m*:

$\neg S$	is true iff	S is false	
$S_1 \wedge S_2$	is true iff	S_1 is true and	S_2 is true
$S_1 \vee S_2$	is true iff	S_1 is true or S_2 is true	
$S_1 \Rightarrow S_2$	is true iff	S_1 is false or	S_2 is true
	is false iff	S_1 is true and	S_2 is false
$S_1 \Leftrightarrow S_2$	is true iff	$S_1 \Rightarrow S_2$ is true and	$S_2 \Rightarrow S_1$ is true

Simple recursive process evaluates an arbitrary sentence, e.g.,

$$\neg P_{1,2} \wedge (P_{2,2} \vee P_{3,1}) = \mathit{true} \wedge (\mathit{true} \vee \mathit{false}) = \mathit{true} \wedge \mathit{true} = \mathit{true}$$

Truth tables for connectives
5种逻辑连接符的真值表

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*

Wumpus world sentences

Let $P_{i,j}$ be true if there is a pit in [i, j] .

Let $B_{i,j}$ be true if there is a breeze in [i, j] .

$\neg P_{1,1}$

$\neg B_{1,1}$

$B_{2,1}$

"Pits cause breezes in adjacent squares"

Wumpus world sentences

Let $P_{i,j}$ be true if there is a pit in [i, j] .

Let $B_{i,j}$ be true if there is a breeze in [i, j] .

$\neg P_{1,1}$

$\neg B_{1,1}$

$B_{2,1}$

"Pits cause breezes in adjacent squares"

$B_{1,1} \Leftrightarrow (P_{1,2} \lor P_{2,1})$

$B_{2,1} \Leftrightarrow (P_{1,1} \lor P_{2,2} \lor P_{3,1})$

"A square is breezy if and only if there is an adjacent pit"

Truth tables for inference

$B_{1,1}$	$B_{2,1}$	$P_{1,1}$	$P_{1,2}$	$P_{2,1}$	$P_{2,2}$	$P_{3,1}$	R_1	R_2	R_3	R_4	R_5	KB
false	*false*	*false*	*false*	*false*	*false*	*false*	*true*	*true*	*true*	*true*	*false*	*false*
false	*false*	*false*	*false*	*false*	*false*	*true*	*true*	*true*	*false*	*true*	*false*	*false*
$\vdots$	$\vdots$	$\vdots$	$\vdots$	$\vdots$	$\vdots$	$\vdots$	$\vdots$	$\vdots$	$\vdots$	$\vdots$	$\vdots$	$\vdots$
false	*true*	*false*	*false*	*false*	*false*	*false*	*true*	*true*	*false*	*true*	*true*	*false*
false	*true*	*false*	*false*	*false*	*false*	*true*	*true*	*true*	*true*	*true*	*true*	*true*
false	*true*	*false*	*false*	*false*	*true*	*false*	*true*	*true*	*true*	*true*	*true*	*true*
false	*true*	*false*	*false*	*false*	*true*	*true*	*true*	*true*	*true*	*true*	*true*	*true*
false	*true*	*false*	*false*	*true*	*false*	*false*	*true*	*false*	*false*	*true*	*true*	*false*
$\vdots$	$\vdots$	$\vdots$	$\vdots$	$\vdots$	$\vdots$	$\vdots$	$\vdots$	$\vdots$	$\vdots$	$\vdots$	$\vdots$	$\vdots$
true	*true*	*true*	*true*	*true*	*true*	*true*	*false*	*true*	*true*	*false*	*true*	*false*

Enumerate rows (different assignments to symbols),
if KB is true in row, check that α is too

Inference by enumeration枚举

Depth-first enumeration of all models is sound and complete

```
function TT-ENTAILS?(KB, α) returns true or false
   inputs: KB, the knowledge base, a sentence in propositional logic
           α, the query, a sentence in propositional logic

   symbols ← a list of the proposition symbols in KB and α
   return TT-CHECK-ALL(KB, α, symbols, [])
```

```
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 do
       P ← FIRST(symbols); rest ← REST(symbols)
       return TT-CHECK-ALL(KB, α, rest, EXTEND(P, true, model)) and
                  TT-CHECK-ALL(KB, α, rest, EXTEND(P, false, model))
```

For *n* symbols, time complexity is $O(2^n)$, space complexity is $O(n)$; problem is co-NP-complete

Logical equivalence

Two sentences are logically equivalent（逻辑等价） iff true in same models:

$\alpha \equiv \beta$ if and only if $\alpha \models \beta$ and $\beta \models \alpha$

$(\alpha \wedge \beta) \equiv (\beta \wedge \alpha)$	commutativity of $\wedge$	$\wedge$的可交换性
$(\alpha \vee \beta) \equiv (\beta \vee \alpha)$	commutativity of $\vee$	$\vee$的可交换性
$((\alpha \wedge \beta) \wedge \gamma) \equiv (\alpha \wedge (\beta \wedge \gamma))$	associativity of $\wedge$	$\wedge$的结合律
$((\alpha \vee \beta) \vee \gamma) \equiv (\alpha \vee (\beta \vee \gamma))$	associativity of $\vee$	$\vee$的结合律
$\neg(\neg\alpha) \equiv \alpha$	double-negation elimination	双重否定消去
$(\alpha \Rightarrow \beta) \equiv (\neg\beta \Rightarrow \neg\alpha)$	contraposition	逆否命题
$(\alpha \Rightarrow \beta) \equiv (\neg\alpha \vee \beta)$	implication elimination	蕴涵消去
$(\alpha \Leftrightarrow \beta) \equiv ((\alpha \Rightarrow \beta) \wedge (\beta \Rightarrow \alpha))$	biconditional elimination	双向蕴涵消去
$\neg(\alpha \wedge \beta) \equiv (\neg\alpha \vee \neg\beta)$	De Morgan	摩根律
$\neg(\alpha \vee \beta) \equiv (\neg\alpha \wedge \neg\beta)$	De Morgan	摩根律
$(\alpha \wedge (\beta \vee \gamma)) \equiv ((\alpha \wedge \beta) \vee (\alpha \wedge \gamma))$	distributivity of $\wedge$ over $\vee$	$\wedge$对$\vee$的分配率
$(\alpha \vee (\beta \wedge \gamma)) \equiv ((\alpha \vee \beta) \wedge (\alpha \vee \gamma))$	distributivity of $\vee$ over $\wedge$	$\vee$对$\wedge$的分配率

Validity and satisfiability
合法性与可满足性

A sentence is valid if it is true in all models,

e.g., True, $A \vee \neg A, A \Rightarrow A, (A \wedge (A \Rightarrow B)) \Rightarrow B$

Validity is connected to inference via the Deduction Theorem（演绎定理）:

$KB \models \alpha$ if and only if ($KB \models \alpha$) is valid

A sentence is satisfiable if it is true in some model

e.g., $A \vee B, \quad C$

A sentence is unsatisfiable if it is true in no models

e.g., $A \wedge \neg A$

Satisfiability is connected to inference via the following:

$KB \models \alpha$ if and only if ($KB \wedge \neg \alpha$) is unsatisable

i.e., prove α by *reduction ad absurdum*（*归谬，反证法*）

假定 α 为假，并证明这将推导出和已知公理KB的一个矛盾

Outline

- Knowledge-based agents
- Wumpus world
  - 关于基于知识的智能体运转的例子
- Logic in general — models and entailment（蕴涵）
- Propositional (Boolean) logic 命题逻辑
- Equivalence, validity, satisfiability 等价、合法性和可满足性
- Inference rules and theorem proving
  - forward chaining 前向链接
  - backward chaining 反向链接
  - resolution 归结

Proof methods

Proof methods divide into (roughly) two kinds:

Application of inference rules 推理规则的应用

- Legitimate (sound) generation of new sentences from old
- Proof = a sequence of inference rule applications推理规则的应用序列
  - Can use inference rules as operators in a standard search alg.
  - 寻找证明的过程与搜索问题中寻找解的过程非常类似：定义后继函数以便生成推理规则所有可能的应用。
- Typically require translation of sentences into a normal form（范式）

Model checking 模型检查

- truth table enumeration (always exponential in *n*)
- improved backtracking, e.g.,Davis-Putnam-Logemann-Loveland
- heuristic search in model space (sound but incomplete)
  - e.g., min-conflicts-like hill-climbing algorithms

Forward and backward chaining 前向链接和反向链接

Horn Form (restricted)

KB = conjunction（与） of Horn clauses 霍恩子句

Horn clause =

- proposition symbol（命题符号）; or
- (conjunction of symbols) $\Rightarrow$ symbol

E.g., $C \wedge (B \Rightarrow A) \wedge (C \wedge D \Rightarrow B)$

Modus Ponens（分离规则，肯定前件的假言推理）(for Horn Form): complete for Horn KBs

$$\frac{\alpha_1, \ldots, \alpha_n, \quad \alpha_1 \wedge \ldots \wedge \alpha_n \Rightarrow \beta}{\beta}$$

Can be used with forward chaining or backward chaining.

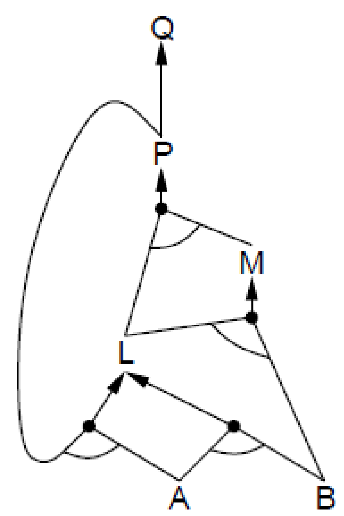
These algorithms are very natural and run in linear time

Forward chaining

Idea: fire any rule whose premises （前提） are satisfied in the *KB*, add its conclusion to the *KB*, until query（询问） is found

从知识库中的已知事实（正文字）开始。如果蕴涵的所有前提已知，那么把它的结论加到已知事实集。持续这一过程，直到询问q被添加或者直到无法进行更进一步的推理

$$P \Rightarrow Q$$
$$L \wedge M \Rightarrow P$$
$$B \wedge L \Rightarrow M$$
$$A \wedge P \Rightarrow L$$
$$A \wedge B \Rightarrow L$$
$$A$$
$$B$$

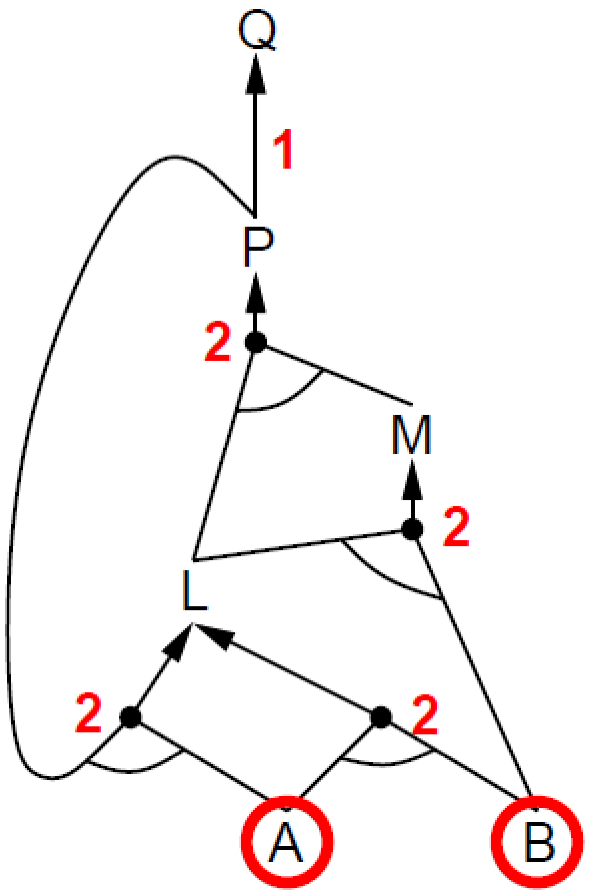


Forward chaining algorithm

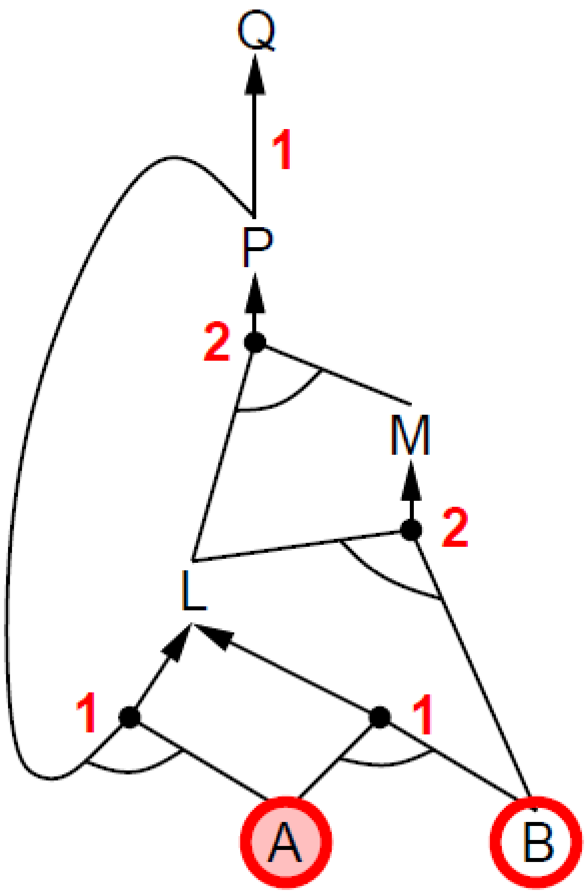
```
function PL-FC-ENTAILS?(KB, q) returns true or false
   inputs: KB, the knowledge base, a set of propositional Horn clauses
           q, the query, a proposition symbol
  local variables: count, a table, indexed by clause, initially the number of premises
                   inferred, a table, indexed by symbol, each entry initially false
                   agenda, a list of symbols, initially the symbols known in KB

   while agenda is not empty do
       p ← POP(agenda)
       unless inferred[p] do
           inferred[p] ← true
           for each Horn clause c in whose premise p appears do
               decrement count[c]
               if count[c] = 0 then do
                   if HEAD[c] = q then return true
                   PUSH(HEAD[c], agenda)
   return false
```

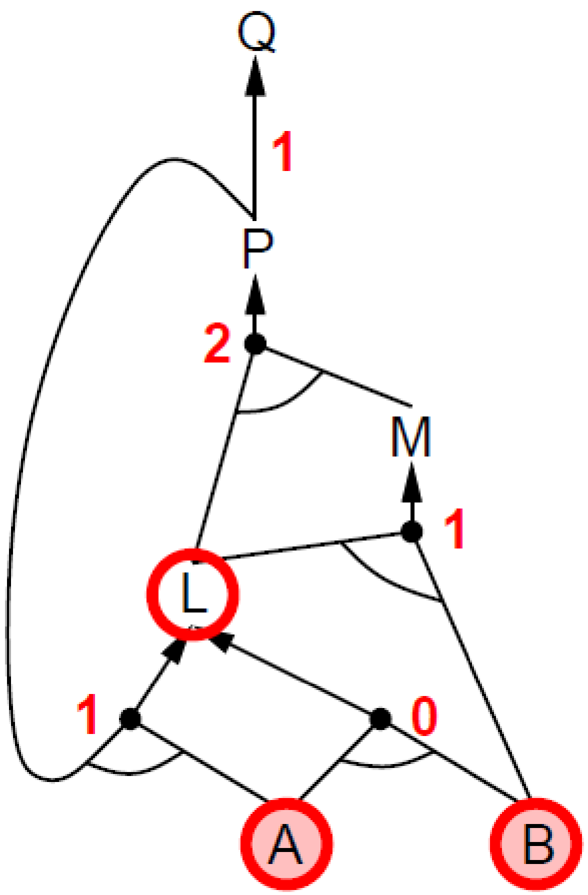
Forward chaining example



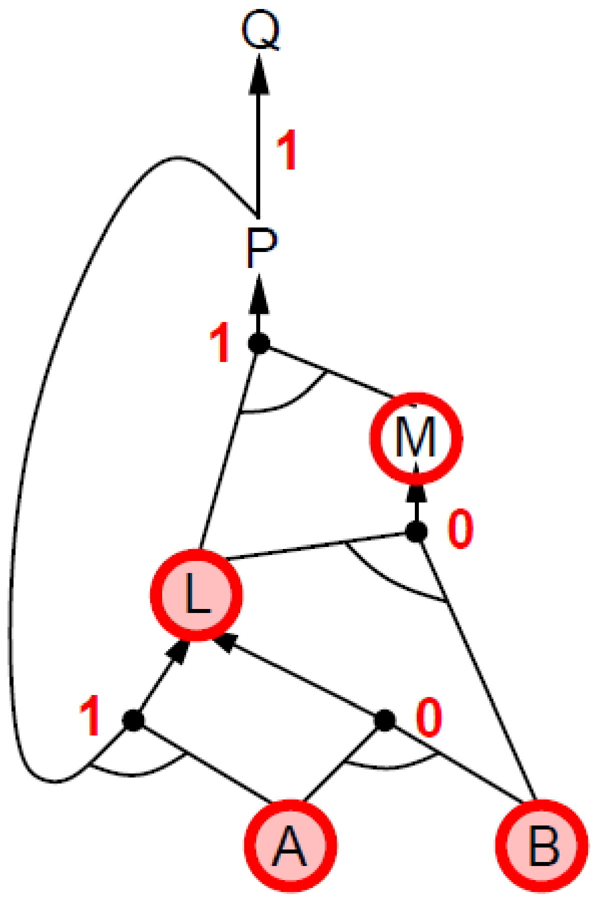
Forward chaining example



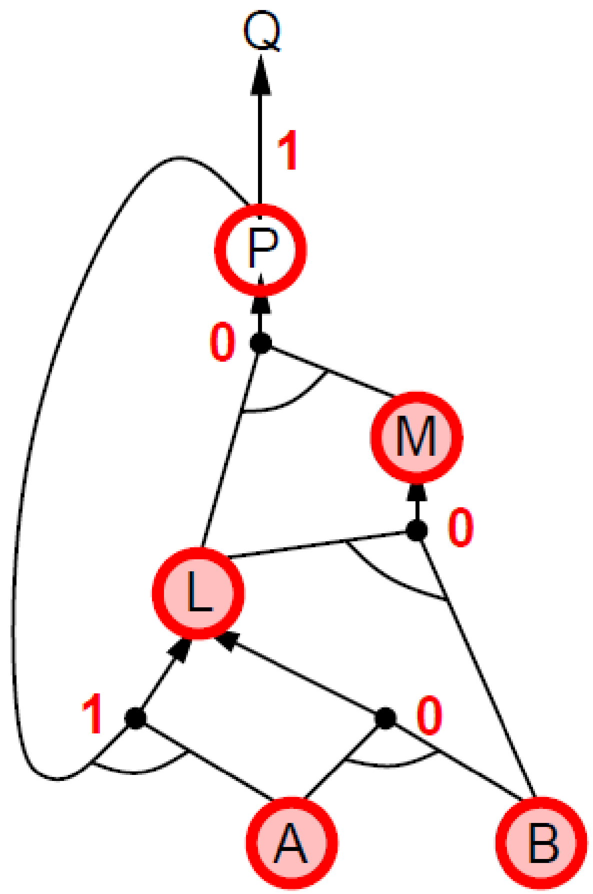
Forward chaining example



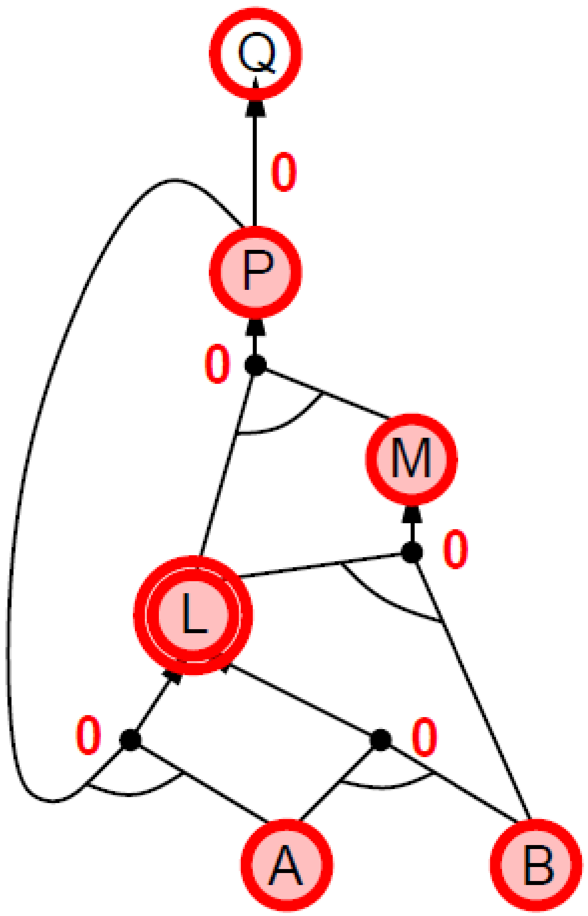
Forward chaining example



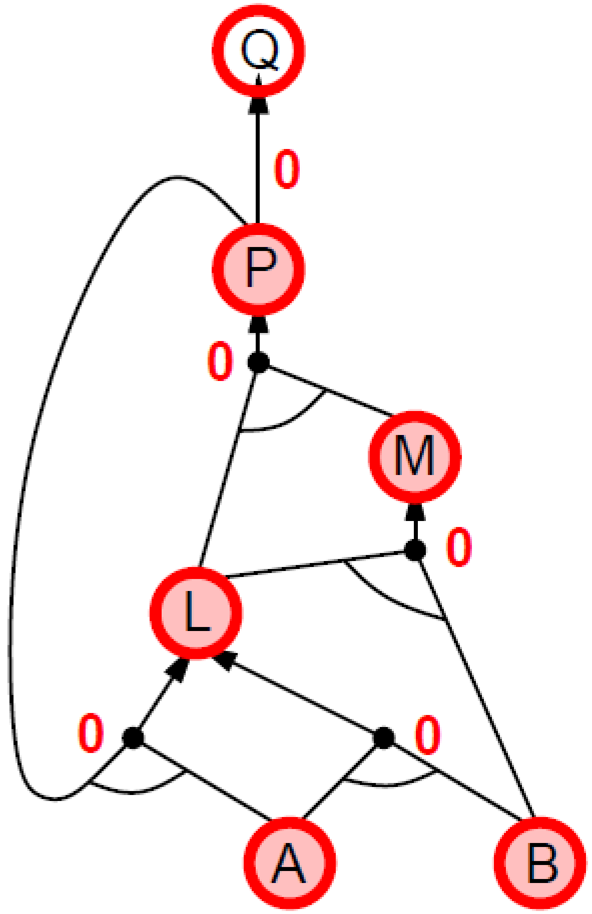
Forward chaining example



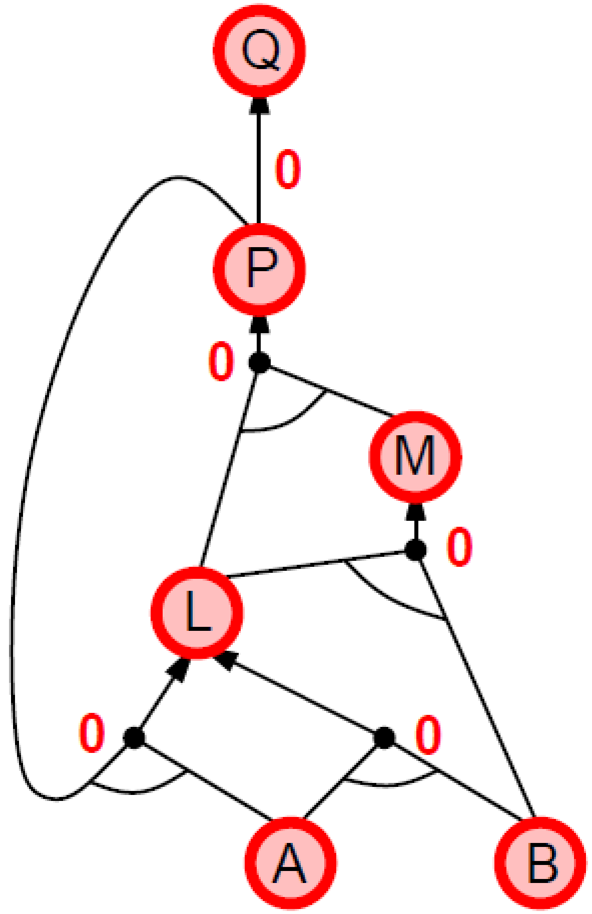
Forward chaining example



Forward chaining example



Forward chaining example



Properties of forward chaining

对于Horn KB，Forward chaining 是

可靠的：每个推理本质上是分离规则的一个应用

完备的：每个被蕴涵的原子语句都将得以生成

Proof of completeness（完备性）

<u>FC可推出每个被*KB*蕴涵的**原子**语句</u>

1. FC到达不动点以后，不可能再出现新的推理。
2. 考察inferred表的最终状态，参与推理过程的每个符号为true，其它为false。把该推理表看做一个逻辑模型*m*
3. *原始KB中的每个确定子句在该模型m中都为真*

   证明：假设某个子句 $a_1 \wedge \ldots \wedge a_k \Rightarrow b$ 在*m*中为false

   那么 $a_1 \wedge \ldots \wedge a_k$ 在*m*中为true，b 在*m*中为false

   与算法已经到达一个不动点相矛盾
4. m是*KB*的一个模型
5. 如果 $KB \models q$，q在*KB*的所有模型中必须为真，包括*m*
6. q在*m*中为真→在inferred表中为真→被FC算法推断出来

Backward chaining

Idea: 从查询q反向进行：

to prove q by BC,

check if q is known already（检查是否q已知为真）, or

prove by BC all premises of some rule concluding q

（寻找知识库中那些以q为结论的蕴涵，证明其中一个蕴涵的所有前提为真）

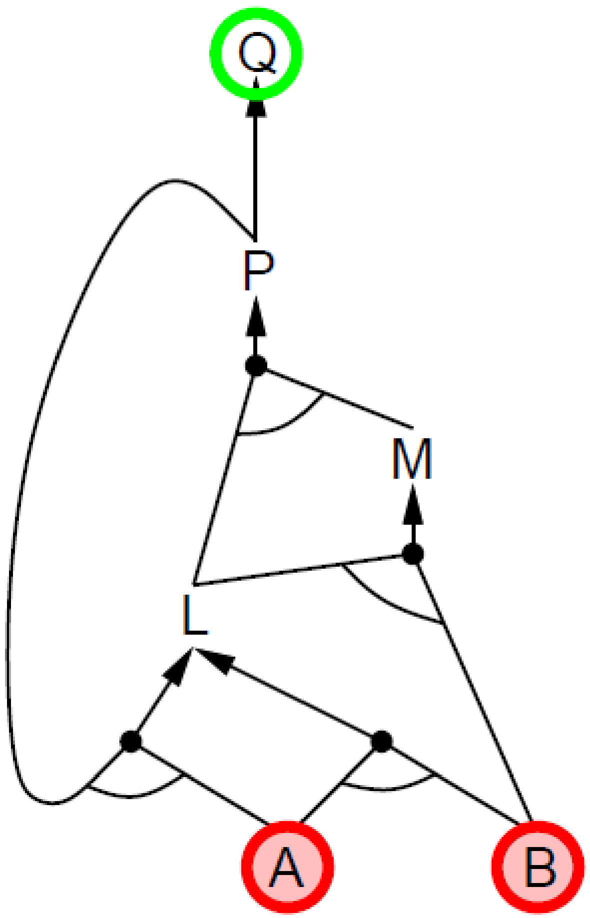
Avoid loops: check if new subgoal is already on the goal stack

Avoid repeated work: check if new subgoal

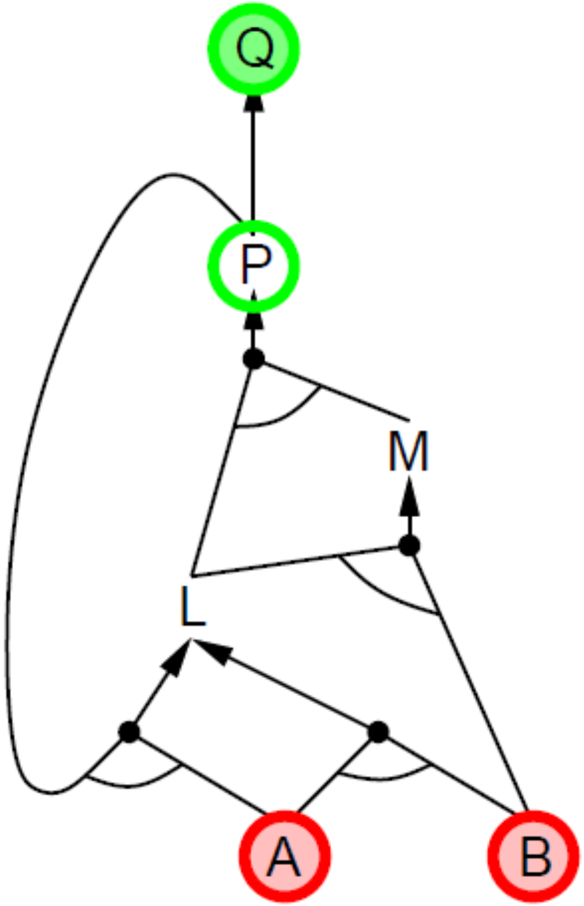
1) has already been proved true, or

2) has already failed

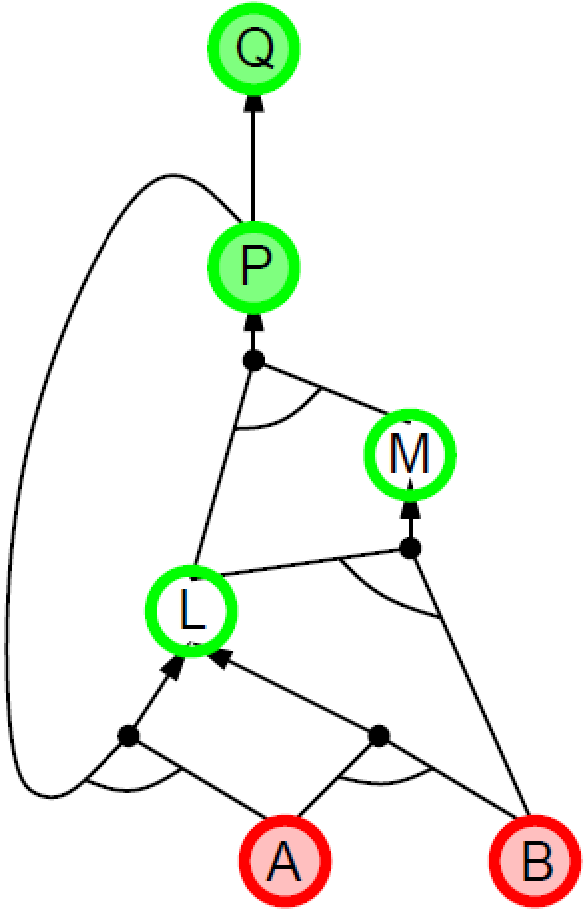
Backward chaining example



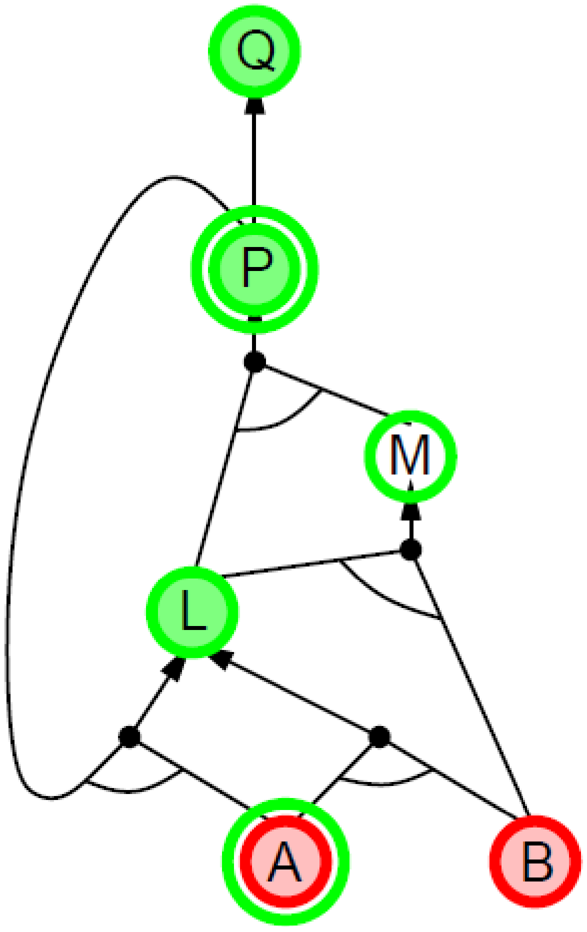
Backward chaining example



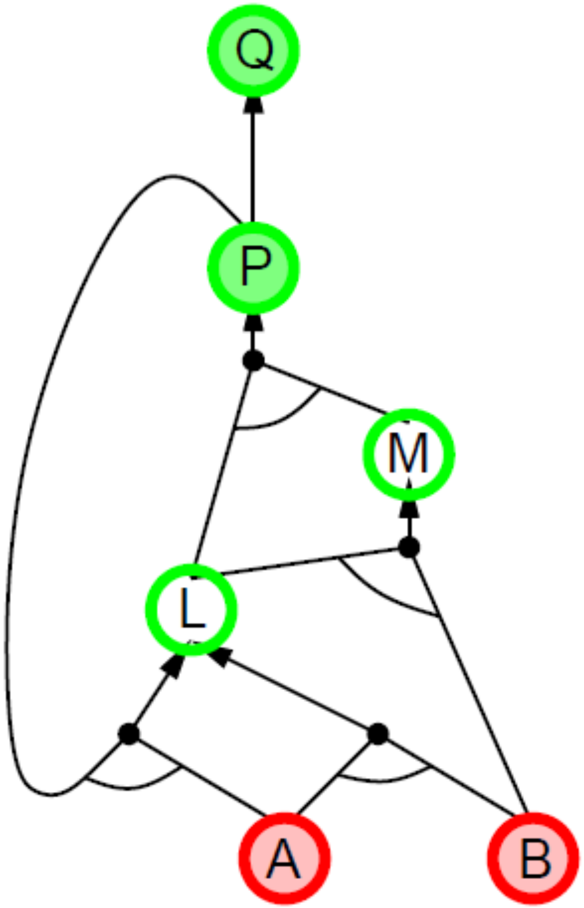
Backward chaining example



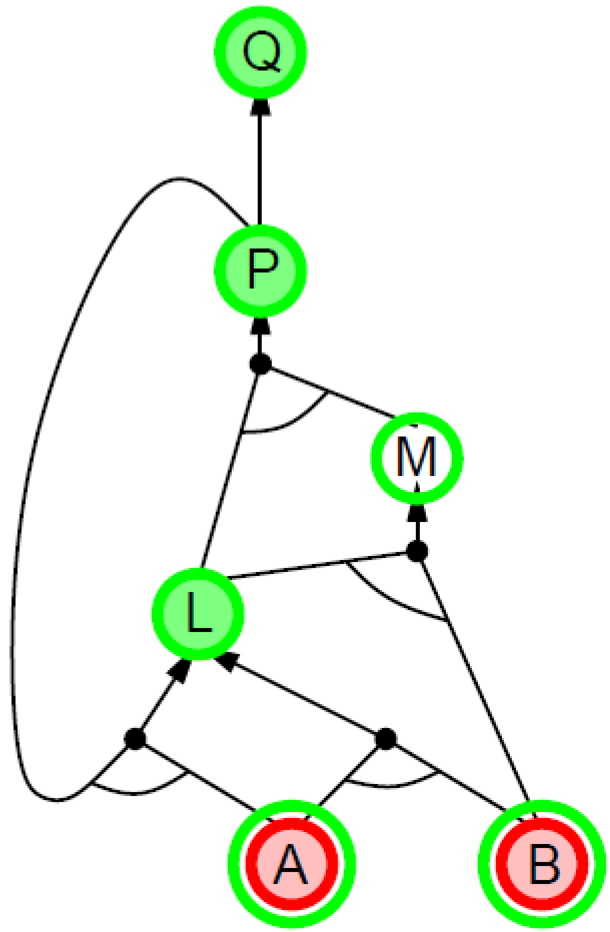
Backward chaining example



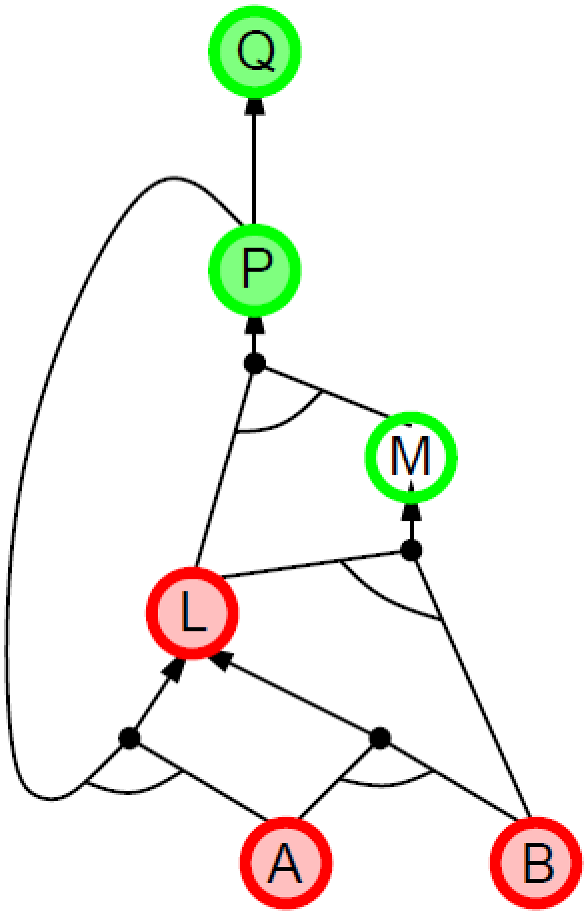
Backward chaining example



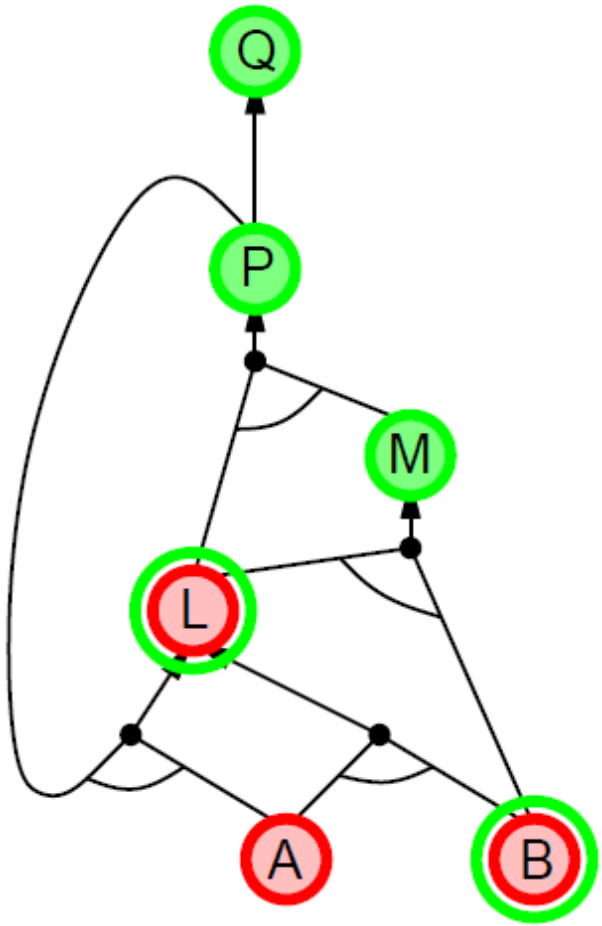
Backward chaining example



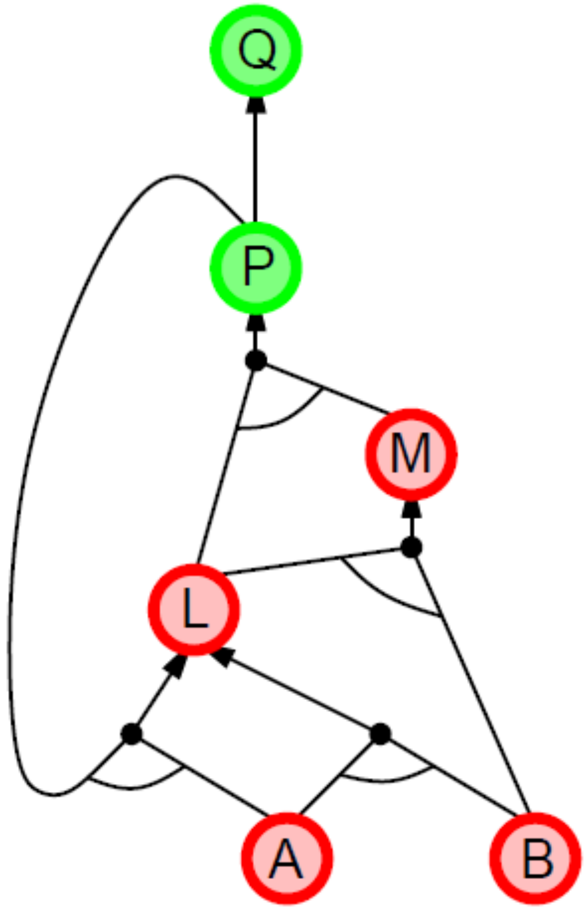
Backward chaining example



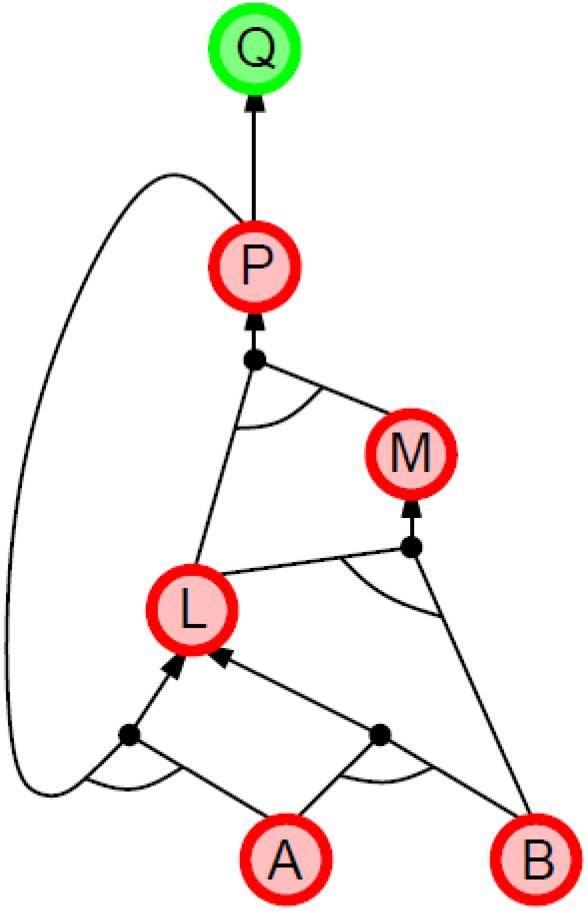
Backward chaining example



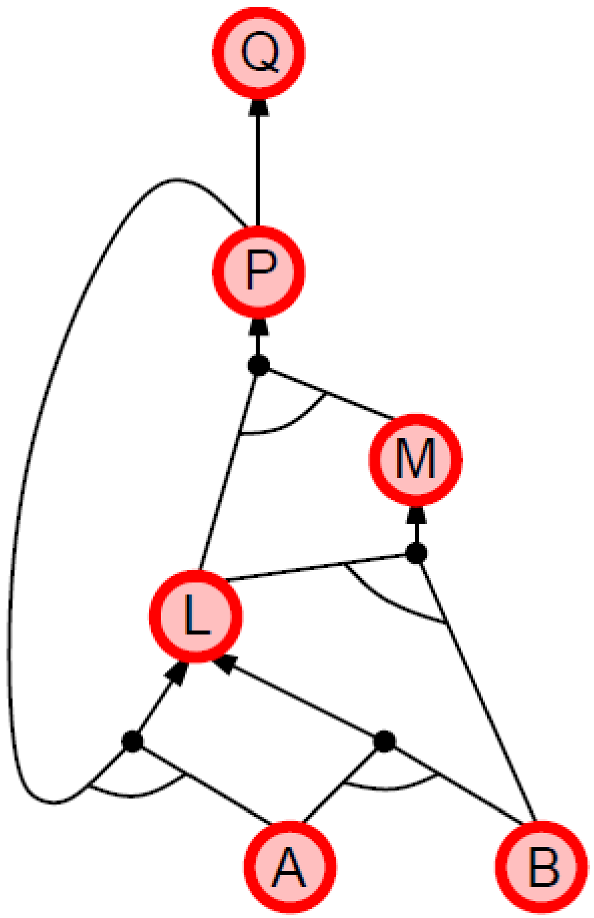
Backward chaining example



Backward chaining example



Backward chaining example



Forward vs. backward chaining

FC is data-driven（数据驱动）, cf. automatic, unconscious processing,

e.g., object recognition, routine decisions

May do lots of work that is irrelevant to the goal

BC is goal-driven（目标指导）, appropriate for problem-solving,

e.g., Where are my keys? How do I get into a PhD program?

Complexity of BC can be much less than linear in size of KB

Resolution归结

Conjunctive Normal Form 合取范式 (CNF)

conjunction of $\underbrace{\text{disjunctions of literals}}_{\text{clauses}}$（文字析取式的合取式）

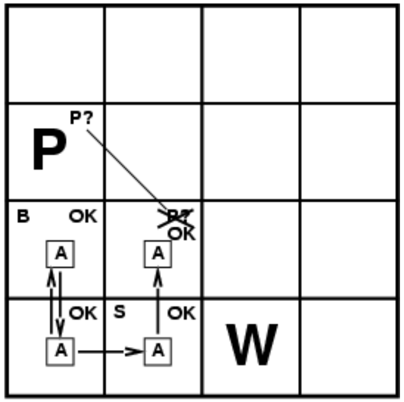
E.g., $(A \vee \neg B) \wedge (B \vee \neg C \vee \neg D)$

Resolution inference rule归结推理规则 (for CNF):

$$\frac{\ell_1 \vee \ldots \vee \ell_k, \qquad m_1 \vee \ldots \vee m_n}{\ell_1 \vee \ldots \vee \ell_{i-1} \vee \ell_{i+1} \vee \ldots \vee \ell_k \vee m_1 \vee \ldots \vee m_{j-1} \vee m_{j+1} \vee \ldots \vee m_n}$$

where ℓ_i and m_j are complementary literals（互补文字）

E.g., $$\frac{P_{1,3} \vee P_{2,2}, \qquad \neg P_{2,2}}{P_{1,3}}$$



Resolution is sound and complete for propositional logic

命题逻辑中归结是可靠和完备的

Conversion to CNF

$B_{1,1} \Leftrightarrow (P_{1,2} \vee P_{2,1})$

1. Eliminate $\Leftrightarrow$, replacing $\alpha \Leftrightarrow \beta$ with $(\alpha \Rightarrow \beta) \wedge (\beta \Rightarrow \alpha)$.

   $(B_{1,1} \Rightarrow (P_{1,2} \vee P_{2,1})) \wedge ((P_{1,2} \vee P_{2,1}) \Rightarrow B_{1,1})$

2. Eliminate $\Rightarrow$, replacing $\alpha \Rightarrow \beta$ with $\neg\alpha \vee \beta$.

   $(\neg B_{1,1} \vee P_{1,2} \vee P_{2,1}) \wedge (\neg(P_{1,2} \vee P_{2,1}) \vee B_{1,1})$

3. Move $\neg$ inwards using de Morgan's rules and double-negation:

   $(\neg B_{1,1} \vee P_{1,2} \vee P_{2,1}) \wedge ((\neg P_{1,2} \wedge \neg P_{2,1}) \vee B_{1,1})$

4. Apply distributivity law ($\vee$ over $\wedge$) and flatten:

   $(\neg B_{1,1} \vee P_{1,2} \vee P_{2,1}) \wedge (\neg P_{1,2} \vee B_{1,1}) \wedge (\neg P_{2,1} \vee B_{1,1})$

Resolution algorithm

- Recall: KB operation boil down to satisfiability

  $KB \models \alpha$ if and only if ($KB \wedge \neg \alpha$) is unsatisfiable

- Algorithm: resolution-based inference
  - Convert all formulas to CNF
  - Repeatedly apply resolution rule
  - Return unsatisfaible iff derive false

Resolution algorithm

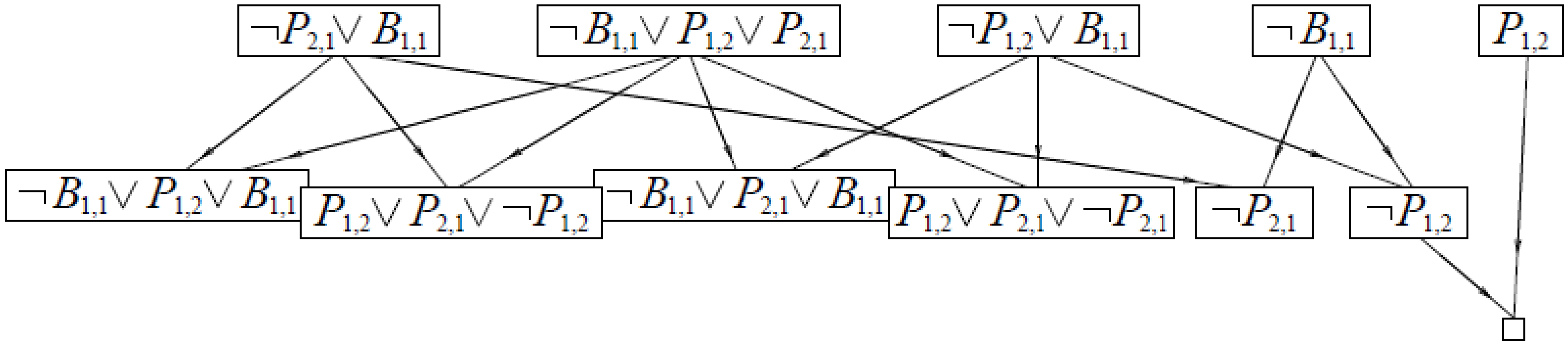
Proof by contradiction, i.e., show $KB \wedge \neg\alpha$ unsatisfiable

```
function PL-RESOLUTION(KB, α) returns true or false
    inputs: KB, the knowledge base, a sentence in propositional logic
            α, the query, a sentence in propositional logic

    clauses ← the set of clauses in the CNF representation of KB ∧ ¬α
    new ← { }
    loop do
        for each C_i, C_j in clauses do
            resolvents ← PL-RESOLVE(C_i, C_j)
            if resolvents contains the empty clause then return true
            new ← new ∪ resolvents
        if new ⊆ clauses then return false
        clauses ← clauses ∪ new
```

Resolution example

$KB = (B_{1,1} \Leftrightarrow (P_{1,2} \vee P_{2,1})) \wedge \neg B_{1,1}$ $\alpha = \neg P_{1,2}$



Time complexity

- Modus ponens inference rule

$$\frac{p_1, \quad \cdots \quad , p_k, \quad (p_1 \wedge \cdots \wedge p_k) \to q}{q}$$

  - Each rule application adds clause with **one** propositional symbol → linear time

- Resolution inference rule

$$\frac{f_1 \vee \cdots \vee f_n \vee h, \quad \neg h \vee g_1 \vee \cdots \vee g_m}{f_1 \vee \cdots \vee f_n \vee g_1 \vee \cdots \vee g_m}$$

  - Each rule application adds clause with **many** propositional symbols → exponential time

Comparison

Horn clauses	any clauses
Modus ponens	resolution
linear time	exponential time
less expressive	more expressive

Summary

Logical agents apply inference to a knowledge base
to derive new information and make decisions

Basic concepts of logic:

— **syntax: formal structure of sentences**
— **semantics: truth of sentences wrt models**
— **entailment: necessary truth of one sentence given another**
— **inference: deriving sentences from other sentences**
— **soundness: derivations produce only entailed sentences**
— **completeness: derivations can produce all entailed sentences**

Wumpus world requires the ability to represent partial and negated information, reason by cases, etc.

Forward, backward chaining are linear-time, complete for Horn clauses
Resolution is complete for propositional logic

Propositional logic lacks expressive power

作业

- 7.12（第二版）=7.13（第三版）
- 证明前向链接算法的完备性