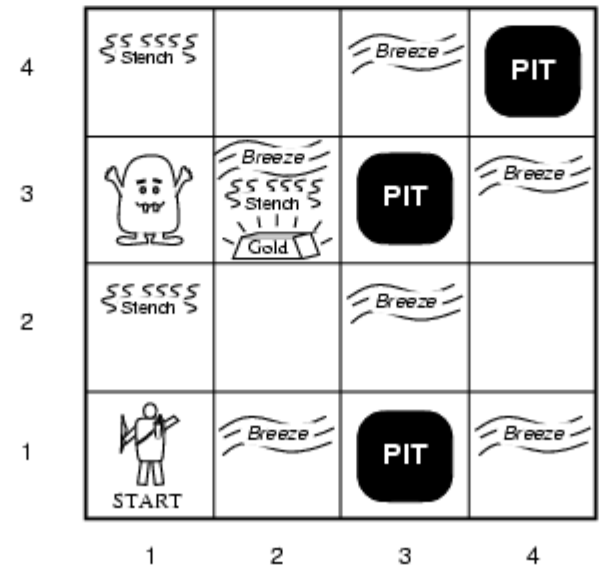
Propositional Logic and AI

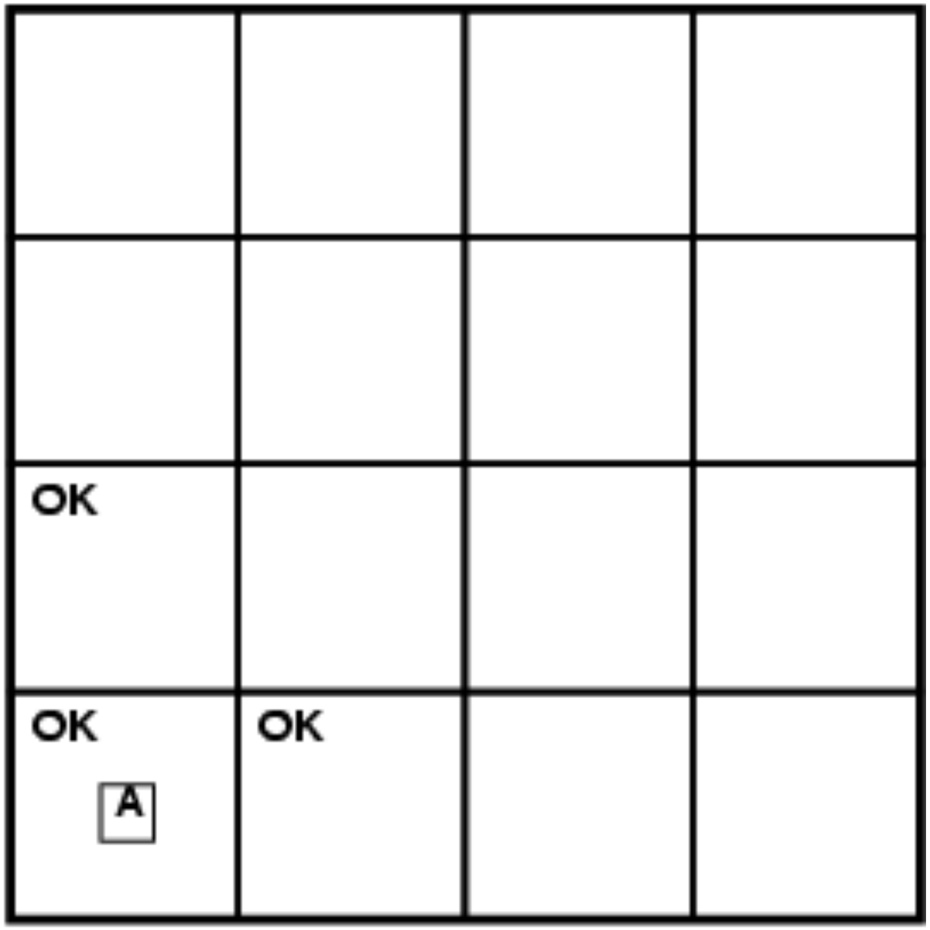
The Wumpus World

- Performance measure
  - gold +1000, death -1000
  - -1 per step, -10 for using the arrow
- Environment
  - Squares adjacent to wumpus are smelly
  - Squares adjacent to pit are breezy
  - Glitter iff gold is in the same square
  - Shooting kills wumpus if you are facing it
  - Shooting uses up the only arrow
  - Grabbing picks up gold if in same square
  - Releasing drops the gold in same square
  - Walking into a wall makes the agent perceive a bump
  - When the wumpus is killed, it emits a scream that is heard throughout the cave



- Sensors: Stench, Breeze, Glitter, Bump, Scream
- Actuators: turn left, turn right, move forward, Grab, Release, Shoot

Exploring a wumpus world



Logic for reasoning

- Goal: Deduce new facts (α) using:
  - The rules about the world
  - Information we gather through perception

  $\left.\begin{array}{l}\text{The rules about the world}\\ \text{Information we gather through perception}\end{array}\right\}$ KB

- Entailment means that one thing follows from another:

$$KB \models \alpha$$

Last night I ate either spaghetti or pizza
I did not eat pizza last night
⊨ Last night I ate spagett

Model

- Assignment of a truth value – true or false – to every atomic sentence

S v P
¬P

KB

M_1: S = False, P = False

M_2: S = False, P = True

M_3: S = True, P = False

M_4: S = True, P = True

A model m is a model of KB iff it is a model of all sentences in KB, that is, all sentences in KB are true in m

Satisfiability of a KB

A KB is satisfiable iff it admits at least one model; otherwise it is unsatisfiable

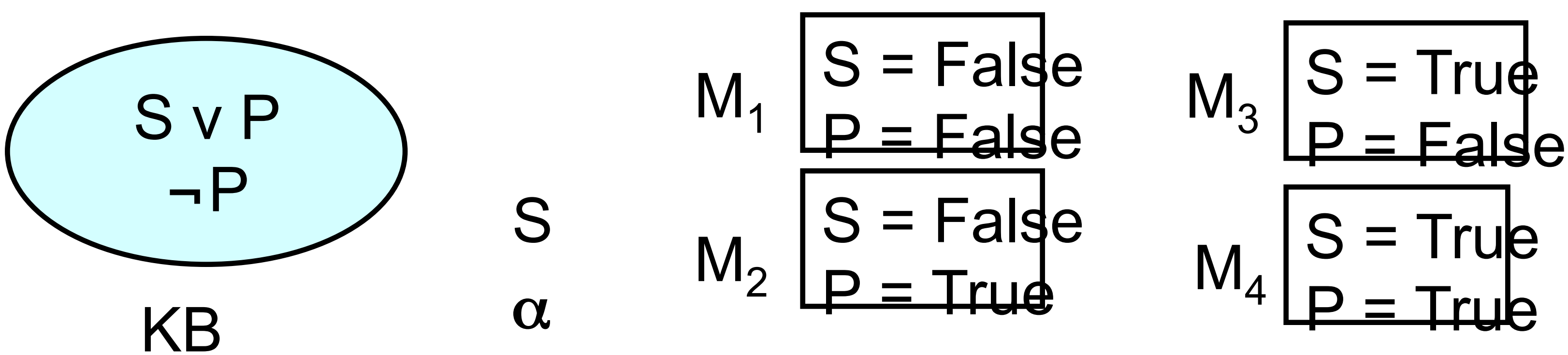
KB1 = $\{P, \neg Q \wedge R\}$ is ____________________

KB2 = $\{\neg P \vee P\}$ is ____________________

KB3 = $\{P, \neg P\}$ is ____________________

Logical Entailment

- KB : set of sentences
- α : arbitrary sentence
- KB entails α – written KB $\models \alpha$ – iff every model of KB is also a model of α
- Alternatively, KB $\models \alpha$ iff
  - $\{KB, \neg\alpha\}$ is unsatisfiable
  - KB $\Rightarrow \alpha$ is valid

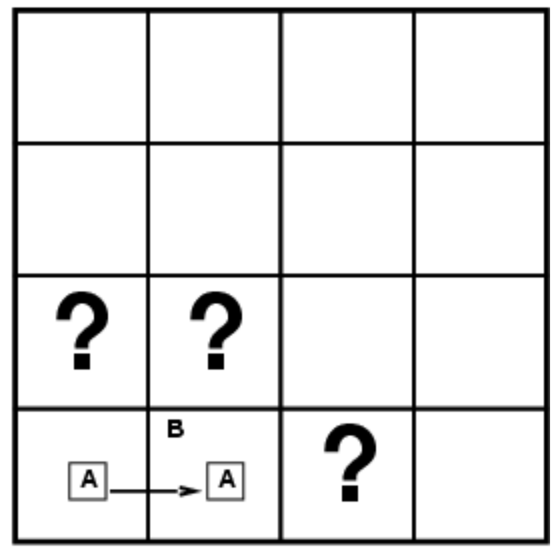


Entailment in the wumpus world

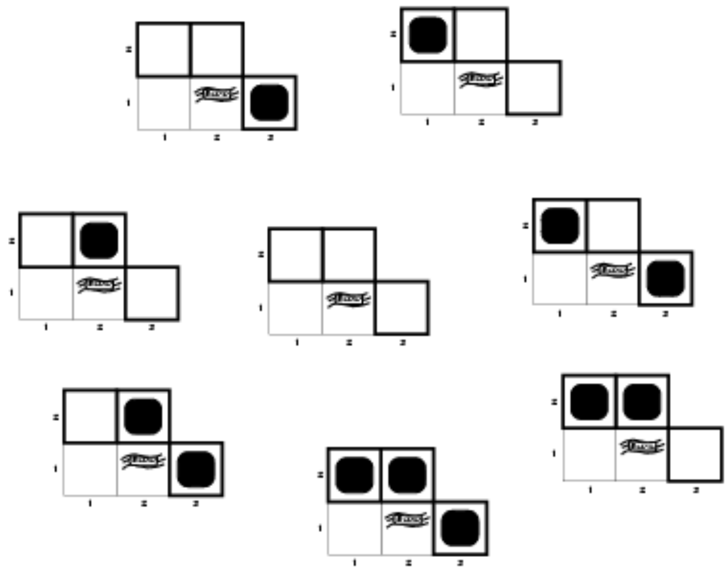
Situation after detecting nothing in [1,1], moving right, breeze in [2,1]

Consider possible models for *KB* assuming only pits

3 Boolean choices ⇒ 8 possible models

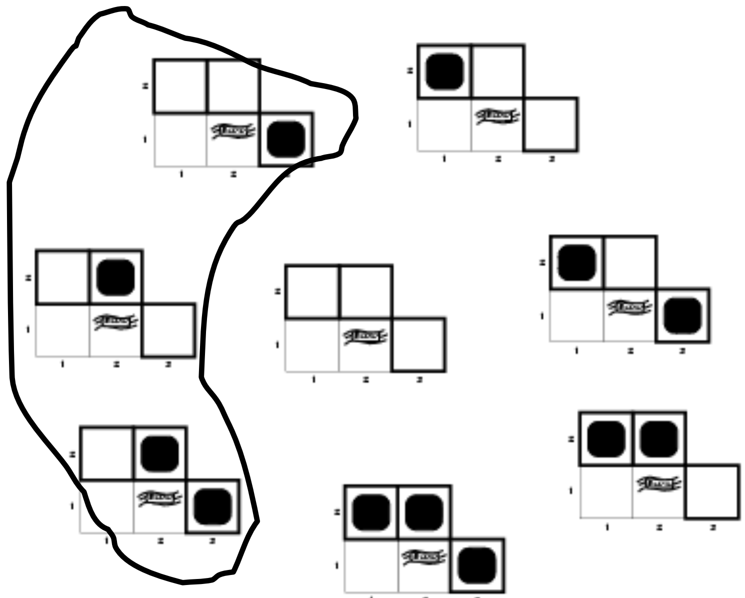


Wumpus models



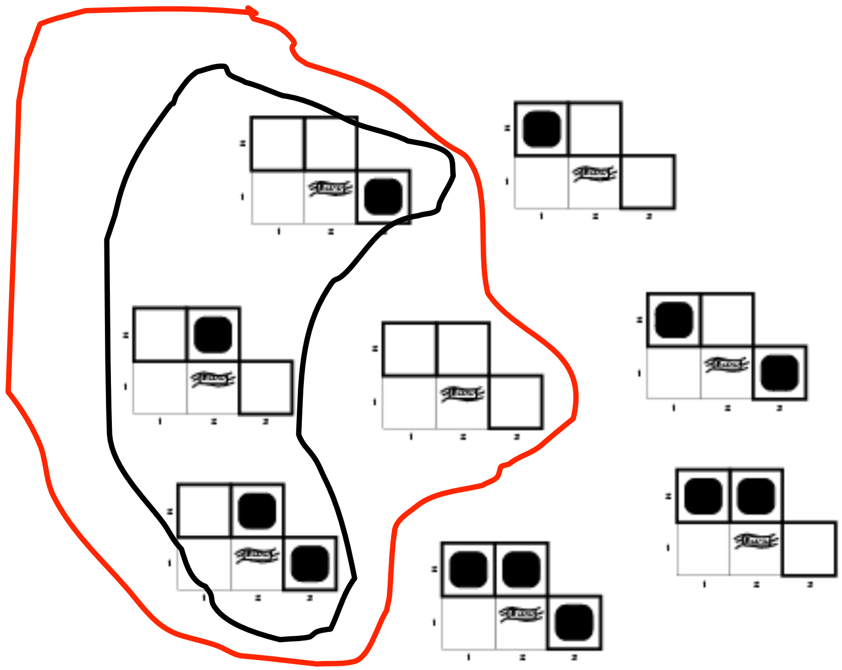
- *KB* = wumpus-world rules +2observations
- α_1 = "[1,2] is safe"

Wumpus models



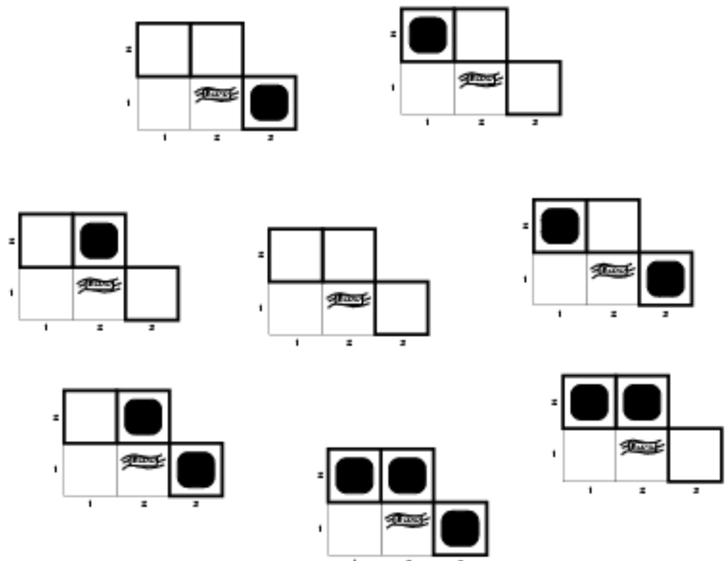
- *KB* = wumpus-world rules + 2observations
- α_1 = "[1,2] is safe"

Wumpus models



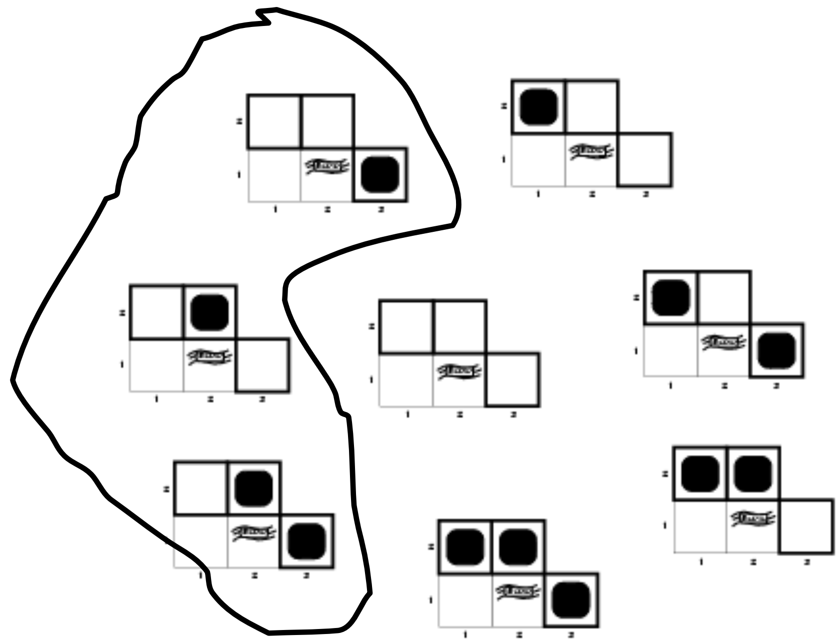
- *KB* = wumpus-world rules + observations
- α_1 = "[1,2] is safe"

Wumpus models



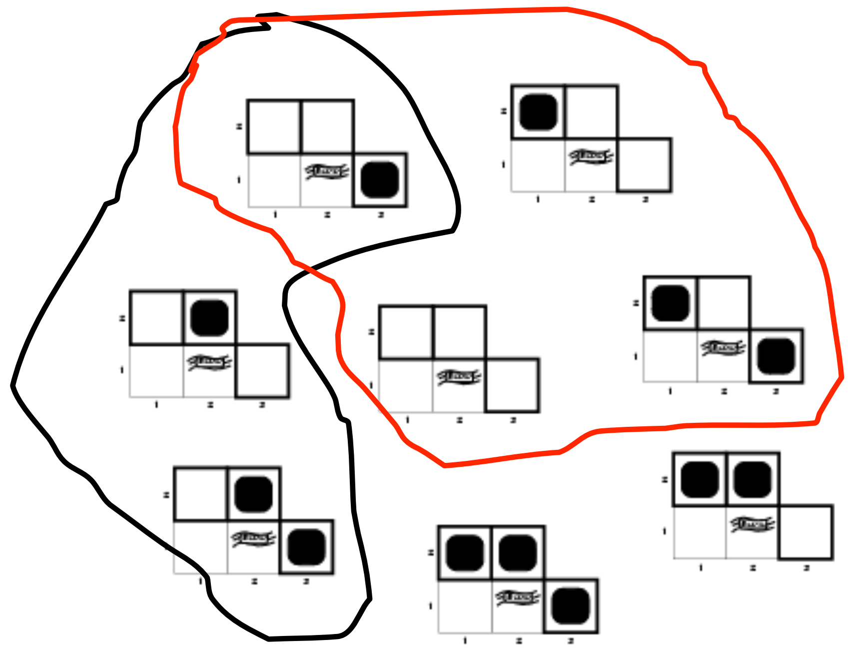
- *KB* = wumpus-world rules + 2observations
- α_2 = "[2,2] is safe"

Wumpus models



- *KB* = wumpus-world rules + 2observations
- α_2 = "[2,2] is safe"

Wumpus models



- *KB* = wumpus-world rules + observations
- α_2 = "[2,2] is safe"

Propositional logic: Syntax

- Propositional logic is the simplest logic – illustrates basic ideas
- The proposition symbols P_1, P_2 etc are sentences

  – If S is a sentence, $\neg S$ is a sentence (negation)
  – If S_1 and S_2 are sentences, $S_1 \wedge S_2$ is a sentence (conjunction)
  – If S_1 and S_2 are sentences, $S_1 \vee S_2$ is a sentence (disjunction)
  – If S_1 and S_2 are sentences, $S_1 \Rightarrow S_2$ is a sentence (implication)
  – If S_1 and S_2 are sentences, $S_1 \Leftrightarrow S_2$ is a sentence (biconditional)

Propositional logic: Semantics

Each model specifies true/false for each proposition symbol

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
$S_1 \Leftrightarrow S_2$	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.,

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

Inference

- Just because a KB entails a sentence doesn't mean we can find (infer) it
- Inference is the process of generating sentences entailed by the KB

Inference Rule

- An inference rule $\{\xi, \psi\} \vdash \varphi$ consists of 2 sentence patterns ξ and ψ called the **conditions** and one sentence pattern φ called the **conclusion**
- If ξ and ψ match two sentences of KB then the corresponding φ can be inferred according to the rule

Example: Modus Ponens

$$\{\ \alpha \Rightarrow \beta\ ,\ \alpha\ \}\ \vdash\ \beta$$

$$\{\xi,\ \psi\} \vdash \varphi$$

Battery-OK ∧ Bulbs-OK ⇒ Headlights-Work ($\alpha \Rightarrow \beta$)
Battery-OK ∧ Starter-OK ∧ ¬Empty-Gas-Tank ⇒ Engine-Starts
Engine-Starts ∧ ¬Flat-Tire ⇒ Car-OK
Battery-OK ∧ Bulbs-OK (α)

Can be used as inference rules…

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

igure 7.11 Standard logical equivalences. The symbols α, β, and γ stand for arbitrary entences of propositional logic.

$\Rightarrow$ Connective symbol (implication)

$\models$ Logical entailment

KB $\models \alpha$ iff KB $\Rightarrow \alpha$ is valid

$\vdash$ Inference

Soundness

- An inference rule is sound if it generates only entailed sentences
- All inference rules previously given are sound, e.g.:
  modus ponens: $\{\alpha \Rightarrow \beta , \alpha\} \vdash \beta$
- Is the following rule sound?
  $$\{\alpha \Rightarrow \beta , \beta\} \vdash \alpha$$

Completeness

- A set of inference rules is complete if every entailed sentences can be obtained by applying some finite succession of these rules
- Modus ponens *alone* is not complete, e.g.:
  from $A \Rightarrow B$ and $\neg B$, we cannot get $\neg A$

Proof

The proof of a sentence α from a set of sentences KB is the derivation of α by applying a series of sound (legal) inference rules

Proof

The proof of a sentence α from a set of sentences KB is the derivation of α by applying a series of sound inference rules

1. Battery-OK $\wedge$ Bulbs-OK $\Rightarrow$ Headlights-Work
2. Battery-OK $\wedge$ Starter-OK $\wedge$ $\neg$Empty-Gas-Tank $\Rightarrow$ Engine-Starts
3. Engine-Starts $\wedge$ $\neg$Flat-Tire $\Rightarrow$ Car-OK
4. Headlights-Work
5. Battery-OK
6. Starter-OK
7. $\neg$Empty-Gas-Tank
8. $\neg$Car-OK

Proof

The proof of a sentence α from a set of sentences KB is the derivation of α by applying a series of sound inference rules

1	1.	Battery-OK $\wedge$ Bulbs-OK $\Rightarrow$ Headlights-Work
2	2.	Battery-OK $\wedge$ Starter-OK $\wedge$ $\neg$Empty-Gas-Tank $\Rightarrow$ Engine-Starts
3	3.	Engine-Starts $\wedge$ $\neg$Flat-Tire $\Rightarrow$ Car-OK
4	4.	Headlights-Work
5	5.	Battery-OK
6	6.	Starter-OK
7	7.	$\neg$Empty-Gas-Tank
8	8.	$\neg$Car-OK
9	9.	Battery-OK $\wedge$ Starter-OK $\leftarrow$ (5+6)
10	10.	Battery-OK $\wedge$ Starter-OK $\wedge$ $\neg$Empty-Gas-Tank $\leftarrow$ (9+7)
11	11.	Engine-Starts $\leftarrow$ (2+10)
12	12.	Engine-Starts $\Rightarrow$ Flat-Tire $\leftarrow$ (3+8)
13	13.	Flat-Tire $\leftarrow$ (11+12)

Inference Problem

- Given:
  - KB: a set of sentence
  - α: a sentence
- Answer:
  - KB $\models \alpha$?

We require an automatic, sound and complete inference method

Inference by enumeration

- We can enumerate all possible models and test whether every model of KB is also a model of α

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]

KB:

$\neg P_{1,1}$
$\neg B_{1,1}$
$B_{2,1}$

Rules of the environment:

"Pits cause breezes in adjacent squares"

Truth tables for inference

$B_{1,1}$	$B_{2,1}$	$P_{1,1}$	$P_{1,2}$	$P_{2,1}$	$P_{2,2}$	$P_{3,1}$	KB	α_1
$false$	$false$	$false$	$false$	$false$	$false$	$false$	$false$	$true$
$false$	$false$	$false$	$false$	$false$	$false$	$true$	$false$	$true$
$\vdots$	$\vdots$	$\vdots$	$\vdots$	$\vdots$	$\vdots$	$\vdots$	$\vdots$	$\vdots$
$false$	$true$	$false$	$false$	$false$	$false$	$false$	$false$	$true$
$false$	$true$	$false$	$false$	$false$	$false$	$true$	$\underline{true}$	$\underline{true}$
$false$	$true$	$false$	$false$	$false$	$true$	$false$	$\underline{true}$	$\underline{true}$
$false$	$true$	$false$	$false$	$false$	$true$	$true$	$\underline{true}$	$\underline{true}$
$false$	$true$	$false$	$false$	$true$	$false$	$false$	$false$	$true$
$\vdots$	$\vdots$	$\vdots$	$\vdots$	$\vdots$	$\vdots$	$\vdots$	$\vdots$	$\vdots$
$true$	$true$	$true$	$true$	$true$	$true$	$true$	$false$	$false$

¬ P1,1

¬B1,1

B2,1

B1,1 ⇔ (P1,2 ∨ P2,1)

B2,1 ⇔ (P1,1 ∨ P2,2 ∨ P3,1)

α_1 ¬$P_{1,2}$

Inference by enumeration

- How efficient is this? How much time/space does it take?

Proof methods

- Proof methods divide into (roughly) two kinds:
  - Application of inference rules
    - Legitimate (sound) generation of new sentences from old
    - Proof = a sequence of inference rule applications
      Can use inference rules as operators in a standard search algorithm
    - Typically require transformation of sentences into a normal form
  - Model checking
    - truth table enumeration (always exponential in *n*)
    - improved backtracking, e.g., Davis--Putnam-Logemann-Loveland (DPLL)
    - heuristic search in model space (sound but incomplete)
      e.g., min-conflicts-like hill-climbing algorithms

Resolution Inference Rule

Conjunctive Normal Form (CNF)

conjunction of disjunctions of literals clauses

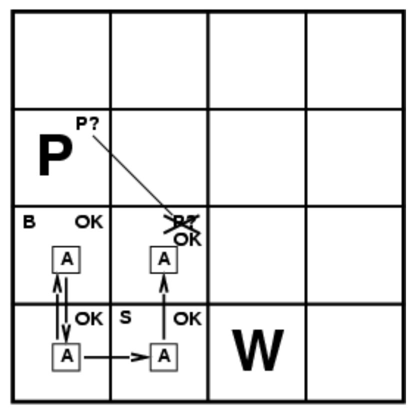
E.g., $(A \lor \neg B) \land (B \lor \neg C \lor \neg D)$

- Resolution inference rule (for CNF):

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

where ℓ_i and m_j are complementary literals.

E.g., $\dfrac{P_{1,3} \lor P_{2,2}, \quad \neg P_{2,2}}{P_{1,3}}$



- Resolution is sound and complete for propositional logic

Conjunctive Normal Form

- Resolution rule only applies to disjunctions of literals
- How could it possibly be compete?
- EVERY sentence in prop logic is equivalent to a conjunction of clauses (disjunction of literals)
- This equivalent form is called CNF
- We have a standard procedure for converting to CNF

Prove that KB ⊨ α by proving that KB $\wedge \neg\alpha$ **is 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 $KB \wedge \neg\alpha$
  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*

Figure 7.12 A simple resolution algorithm for propositional logic. The function PL-RESOLVE returns the set of all possible clauses obtained by resolving its two inputs.

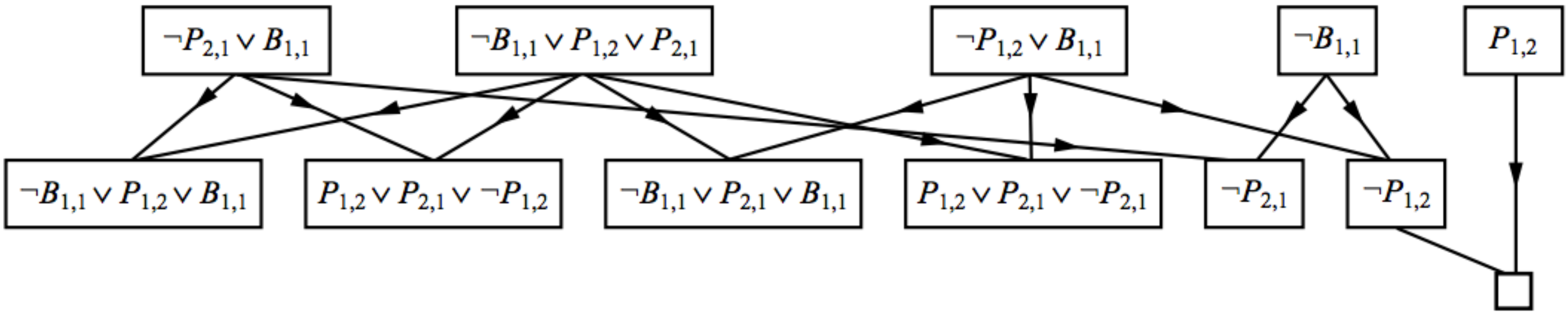


Figure 7.13 Partial application of PL-RESOLUTION to a simple inference in the wumpus world. $\neg P_{1,2}$ is shown to follow from the first four clauses in the top row.

Forward and backward chaining

- Horn Form (restricted)
  KB = conjunction of Horn clauses

  – Horn clause =
    - proposition symbol; or
    - (conjunction of symbols) $\Rightarrow$ symbol
  – E.g., $C \wedge (B \Rightarrow A) \wedge (C \wedge D \Rightarrow B)$

- Modus Ponens (for Horn Form): complete for Horn KBs

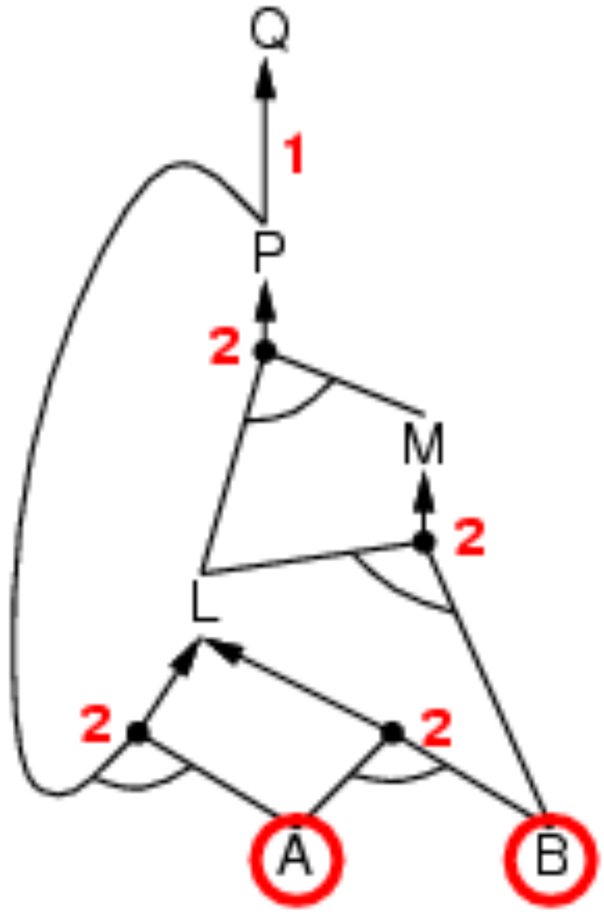
$$\frac{\alpha_1, \ldots, \alpha_n, \qquad \alpha_1 \wedge \ldots \wedge \alpha_n \Rightarrow \beta}{\beta}$$

- Can be used with forward chaining or backward chaining.
- These algorithms are very natural and run in linear time

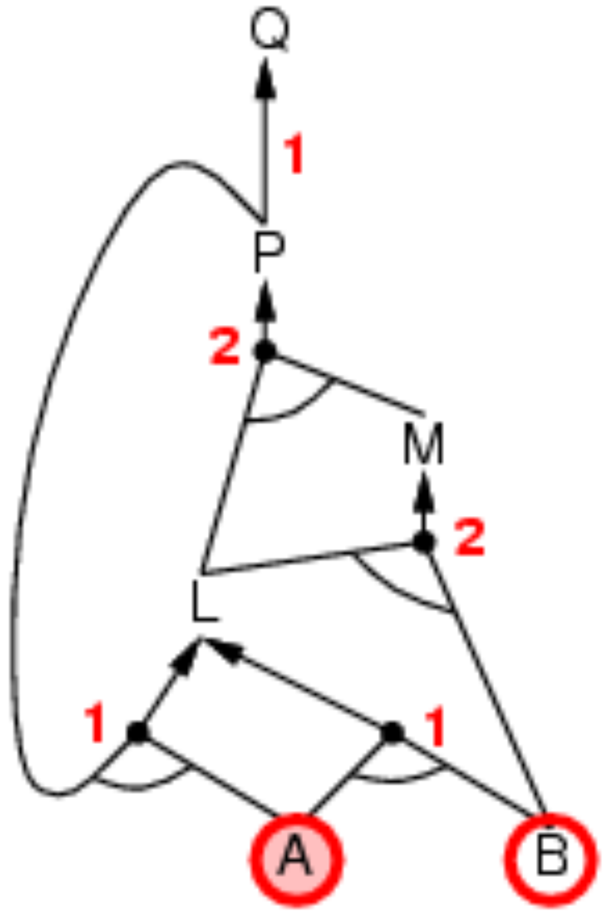
Forward chaining

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

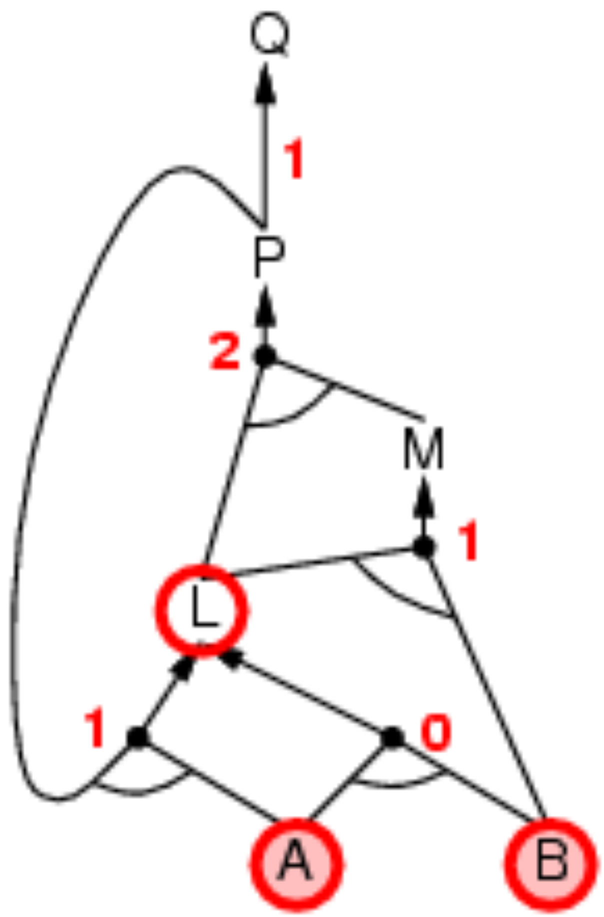
Forward chaining example



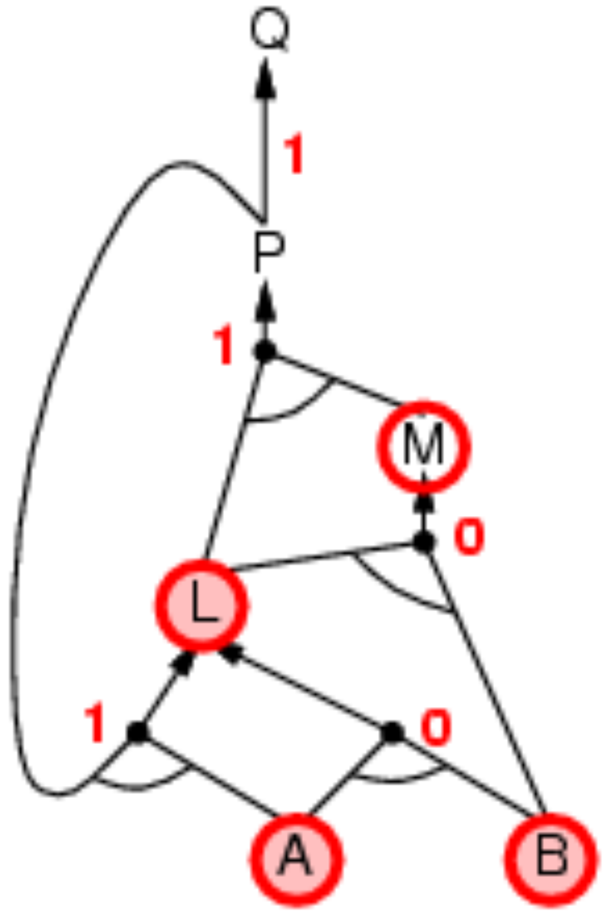
Forward chaining example



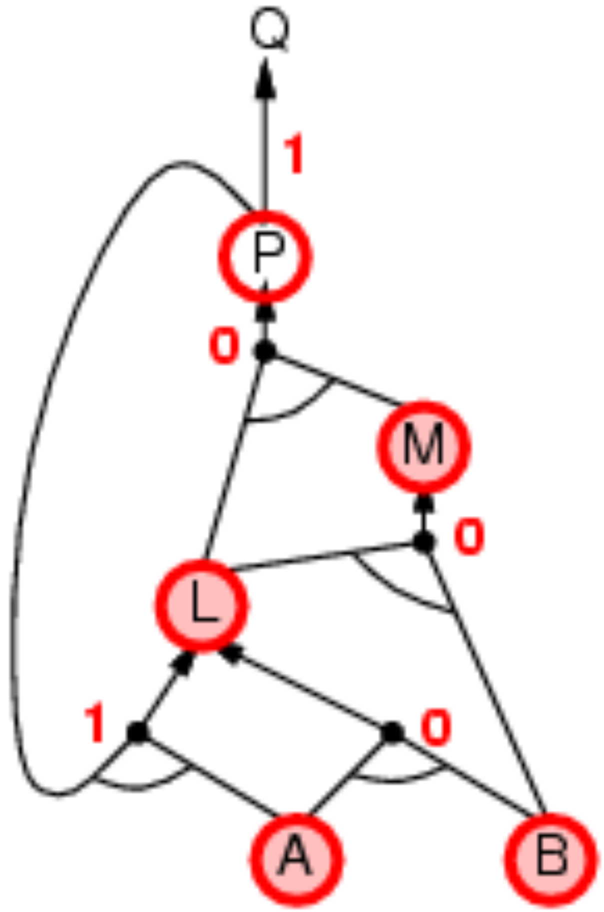
Forward chaining example



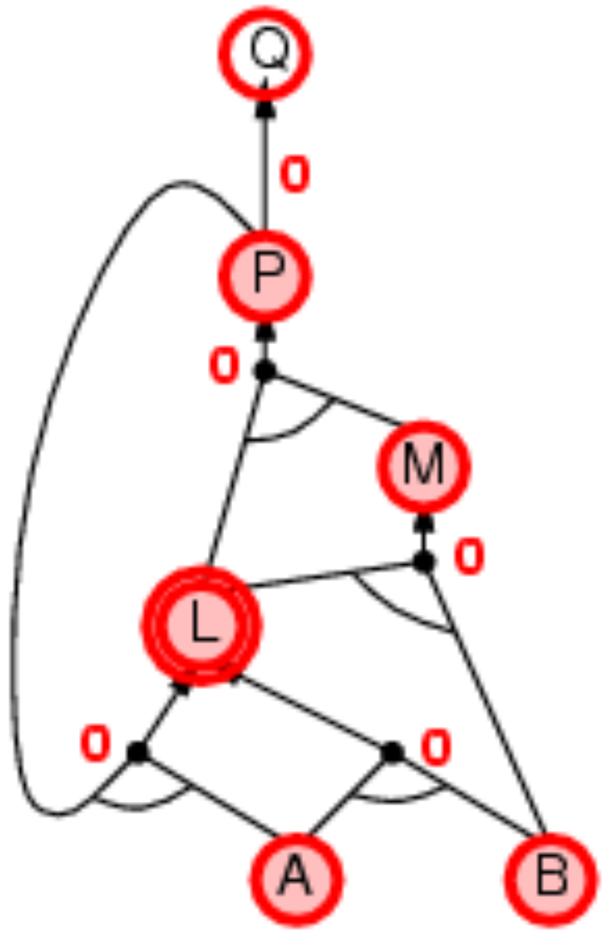
Forward chaining example



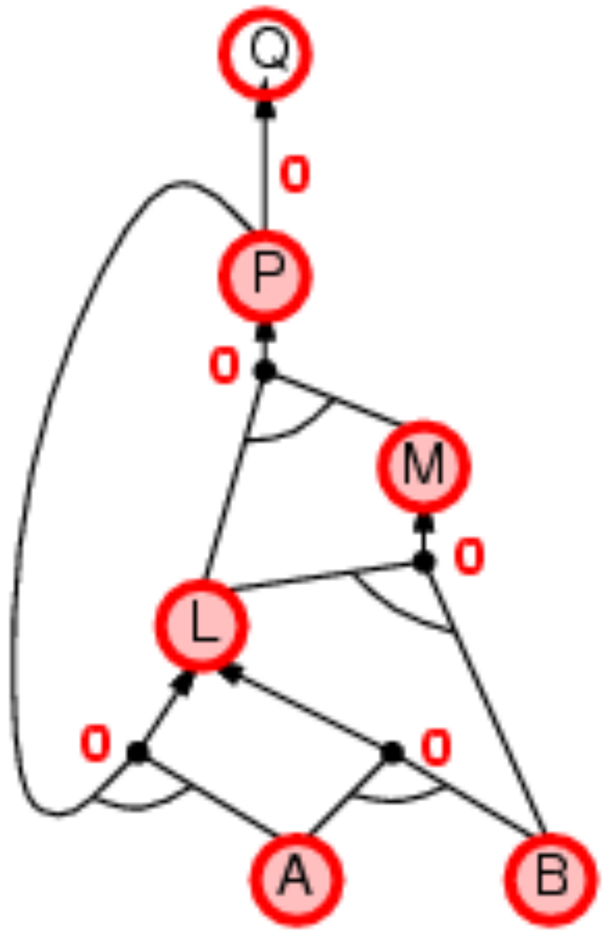
Forward chaining example



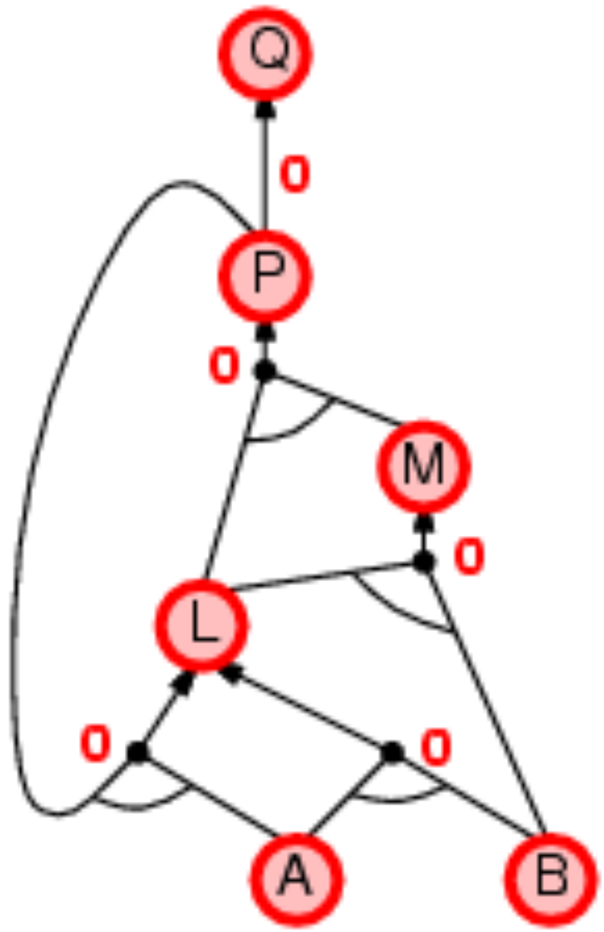
Forward chaining example



Forward chaining example



Forward chaining example



Backward chaining

Idea: work backwards from the query *q*:

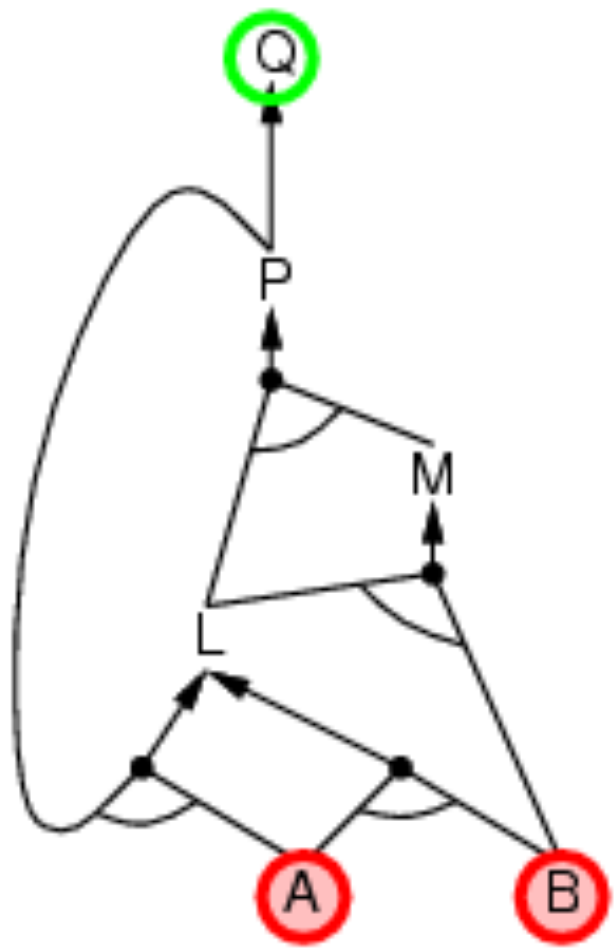
- to prove *q* by BC,
  - check if *q* is known already, or
  - prove by BC all premises of some rule concluding *q*

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

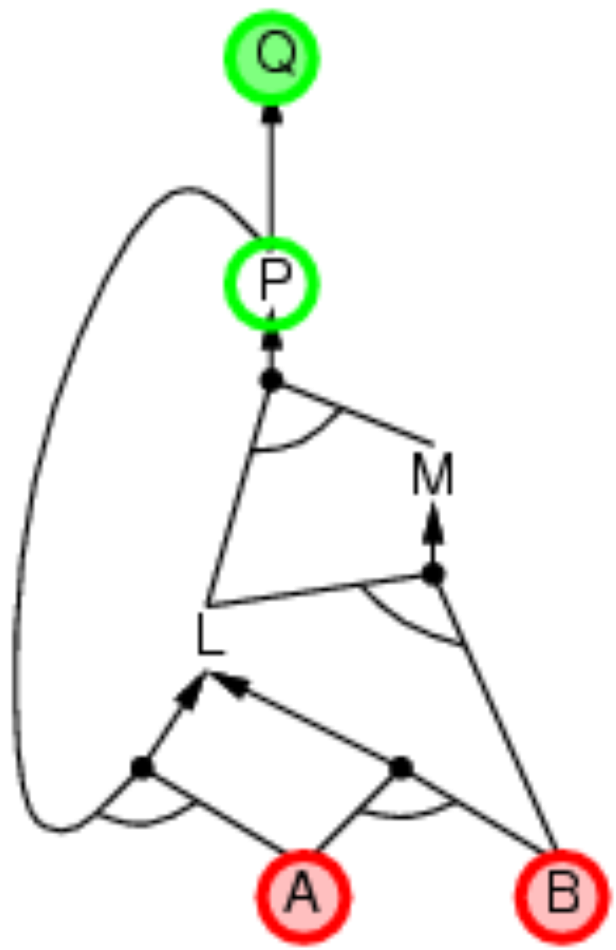
Avoid repeated work: check if new subgoal

1. has already been proved true, or
2. has already failed

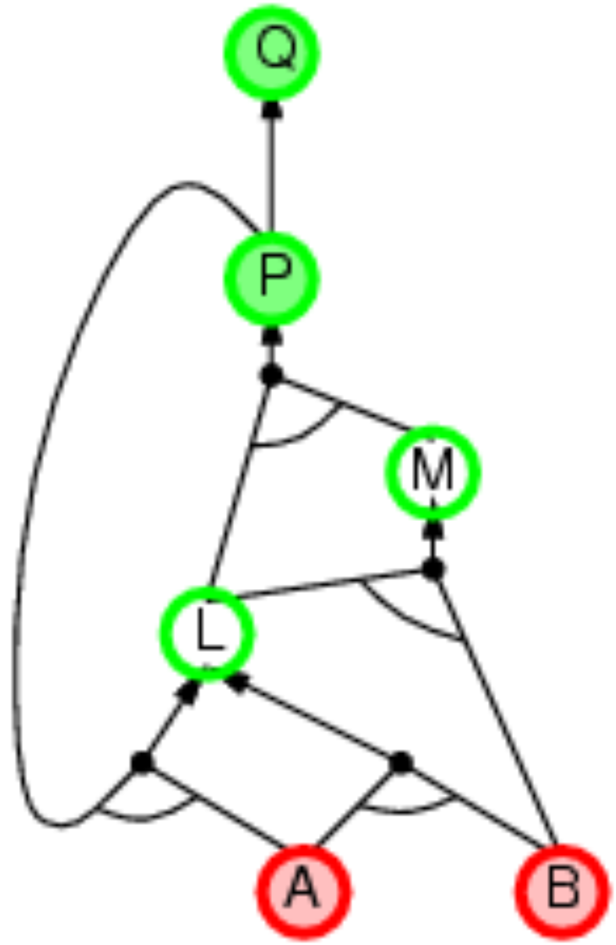
Backward chaining example



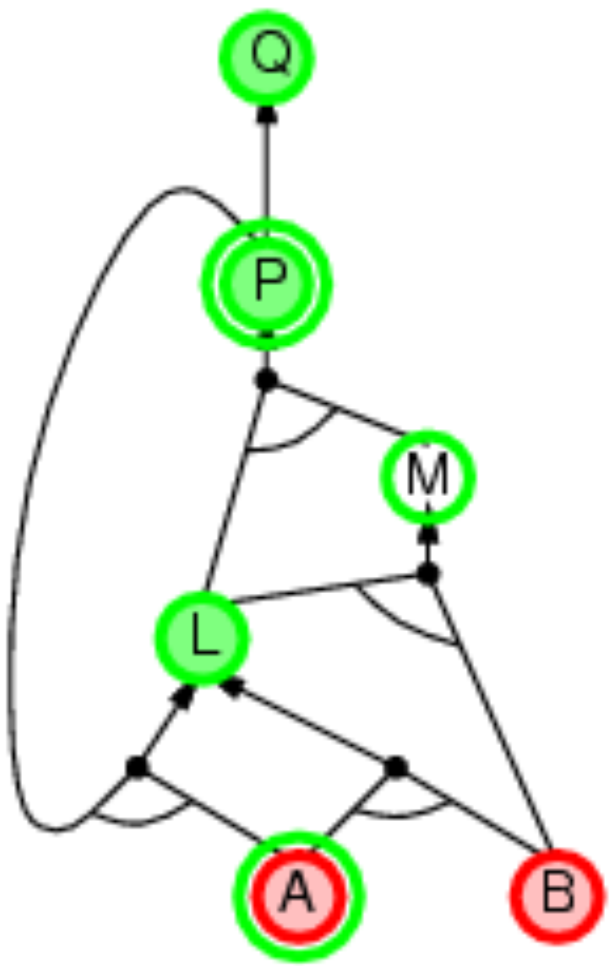
Backward chaining example



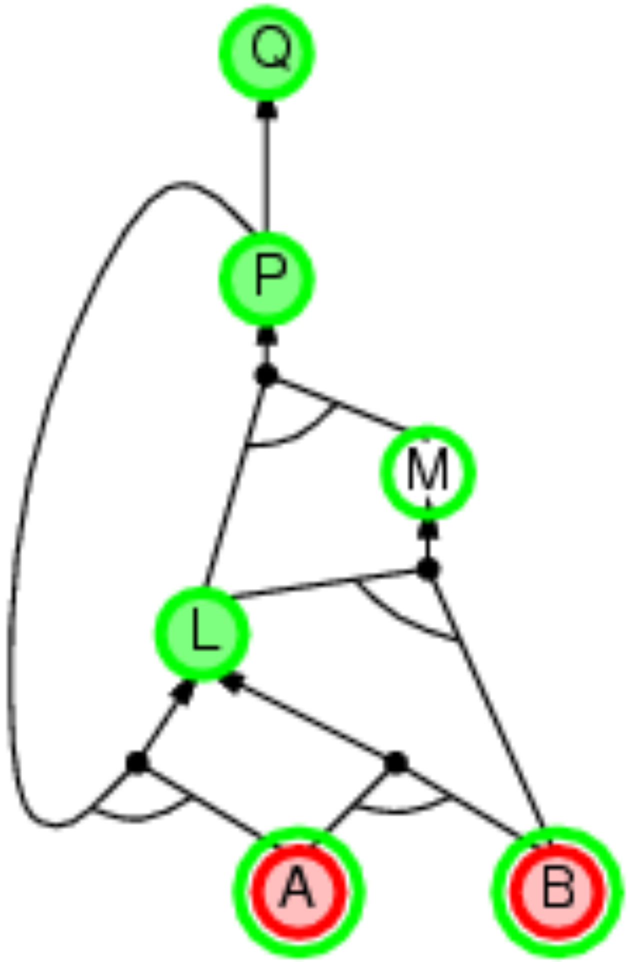
Backward chaining example



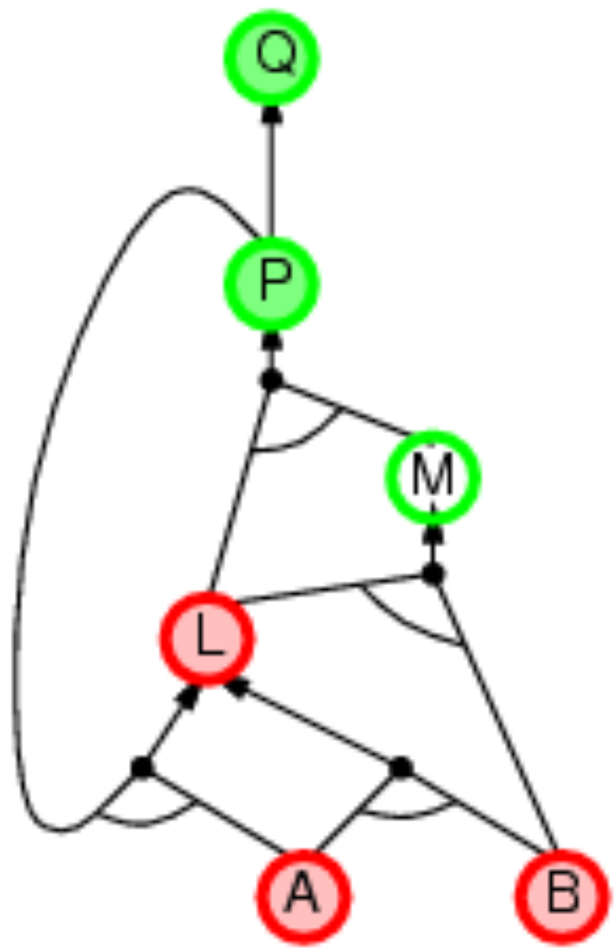
Backward chaining example



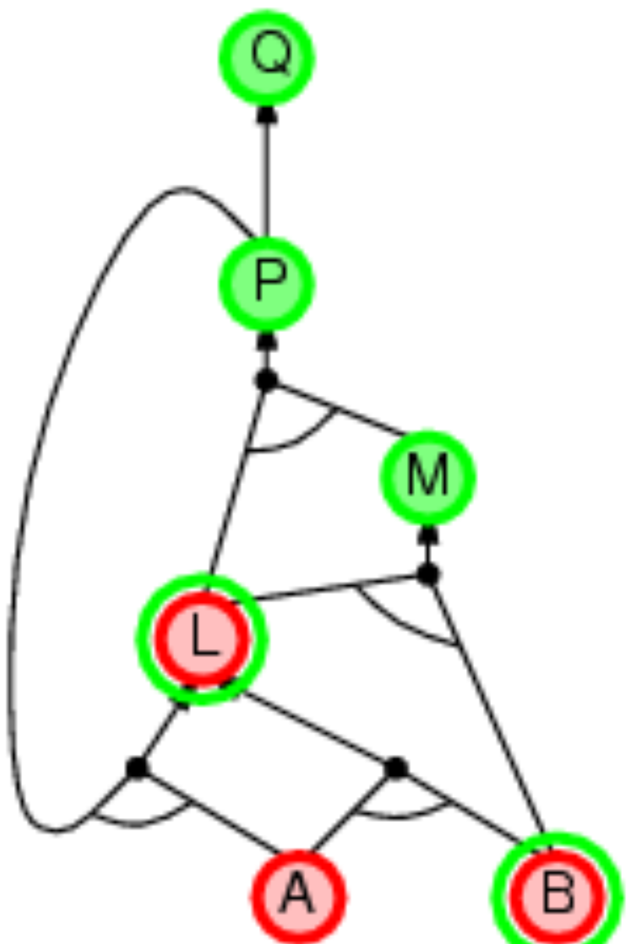
Backward chaining example



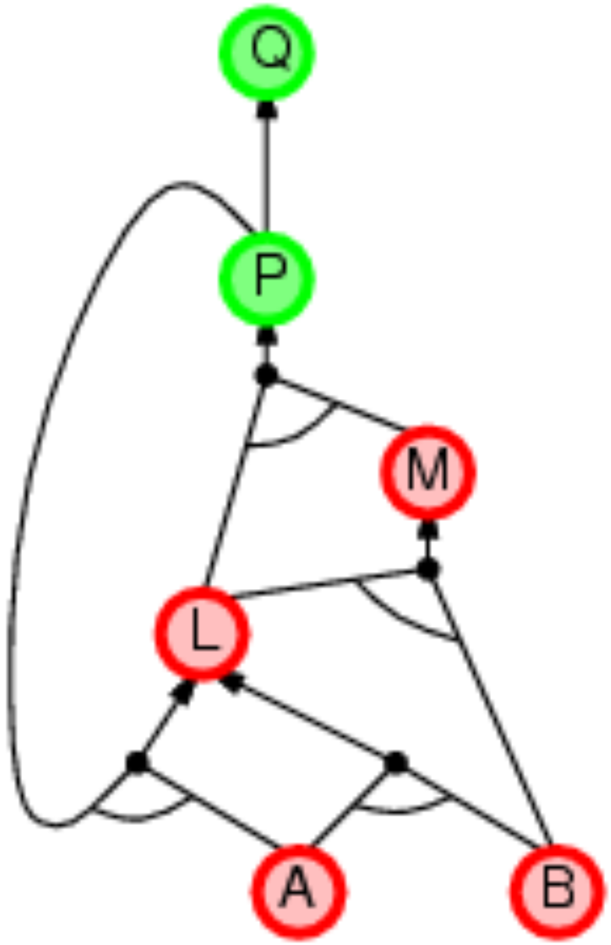
Backward chaining example



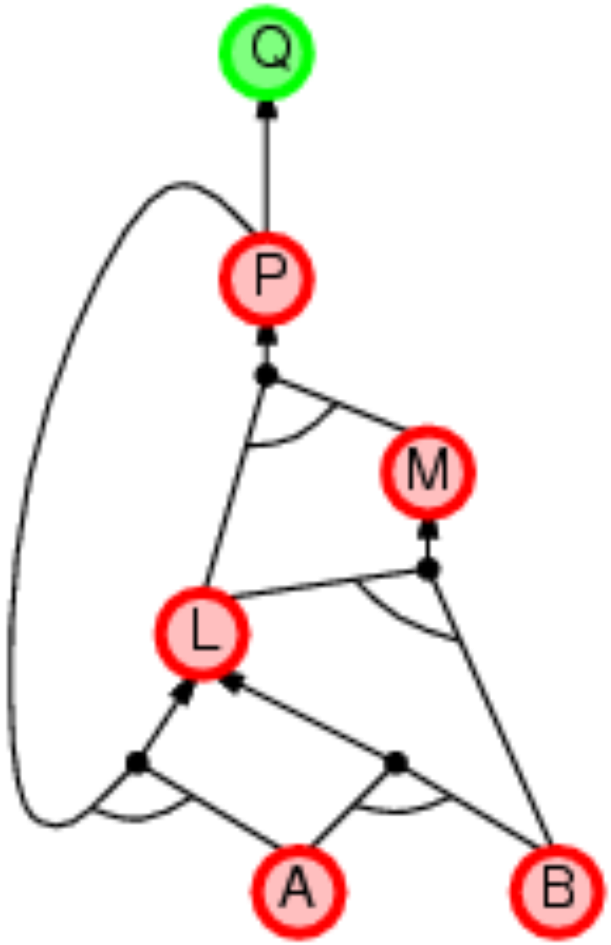
Backward chaining example



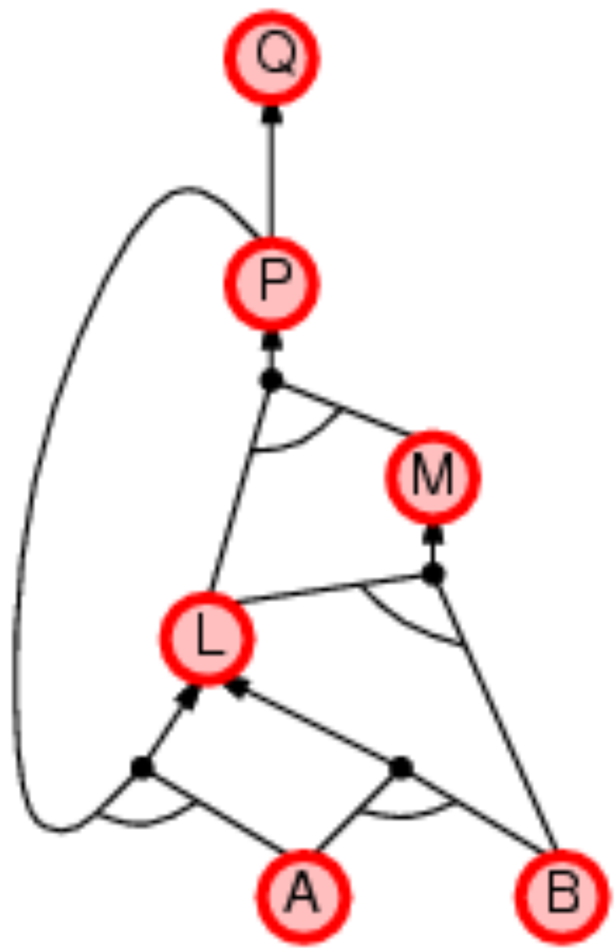
Backward chaining example



Backward chaining example



Backward chaining example



Forward vs. backward chaining

- FC is data-driven, automatic, unconscious processing,
  - e.g., object recognition, routine decisions
- May do lots of work that is irrelevant to the goal
- BC is goal-driven, 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