Chapter 7 – Logical Agents

AI 501 - 2025

Outline

Knowledge-based agents

Wumpus world

Logic: models and entailment

Propositional (Boolean) logic

Equivalence, validity, satisfiability

Inference rules and theorem proving
- resolution
- forward chaining
- backward chaining

DPLL Algorithm

Effective Propositional Model Checking

Knowledge based agents…

… reason over an internal **representation** of knowledge to decide what actions to take.

*- can accept **new tasks** in the form of explicitly described goals*

*- can **achieve competence** quickly by being told or learning new knowledge about the environment*

*- can **adapt to changes** in the environment by updating the relevant knowledge.*

Knowledge base - KB

Set of sentences in a **formal language**

Comes with a way to add new sentences - **TELL**

and a way to query what is known - **ASK**

INFERENCE

KB = **axioms** + sentences that “follow” from other sentences

---

Knowledge level

Describe the agent by what it knows

KB

domain−specific content

Implementation level

Describing the agent by what it does

Inference engine

domain−independent algorithms

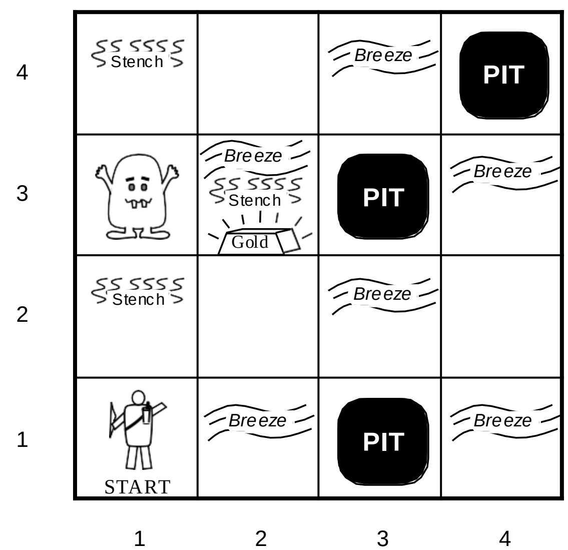
A simple Knowledge-based agent

```
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
```

A generic knowledge-based agent. Given a percept, the agent adds the percept to its knowledge base, asks the knowledge base for the best action, and tells the knowledge base that it has in fact taken that action.

Wumpus world

*The wumpus world is a cave consisting of rooms connected by passageways. Lurking somewhere in the cave is the **terrible wumpus**, a beast that eats anyone who enters its room.*
*Some rooms contain bottomless **pits** that will trap anyone who wanders into these rooms (except for the wumpus,which is too big to fall in).*
*The only redeeming feature of this bleak environment is the possibility of finding a heap of **gold**.*



Wumpus world

Environment

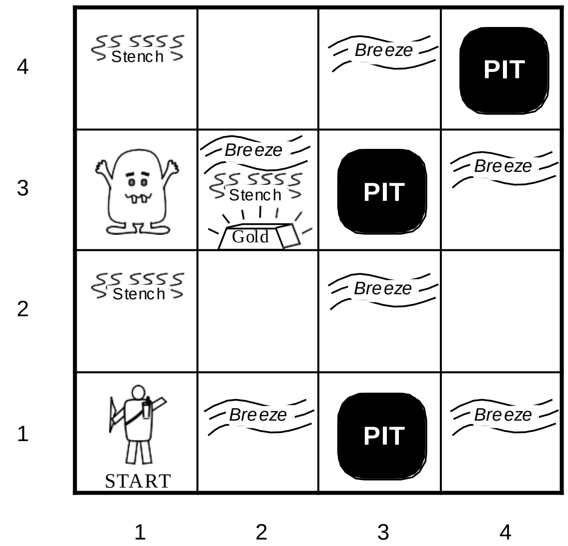
Squares adjacent to wumpus are smelly

Squares adjacent to pit are breezy

Actuators

Left turn, Right turn, Forward

Sensors Breeze, Smell



Exploring a Wumpus world

OK			
OK **A**	**OK**		

A= Agent

B= Breeze

G= Glitter

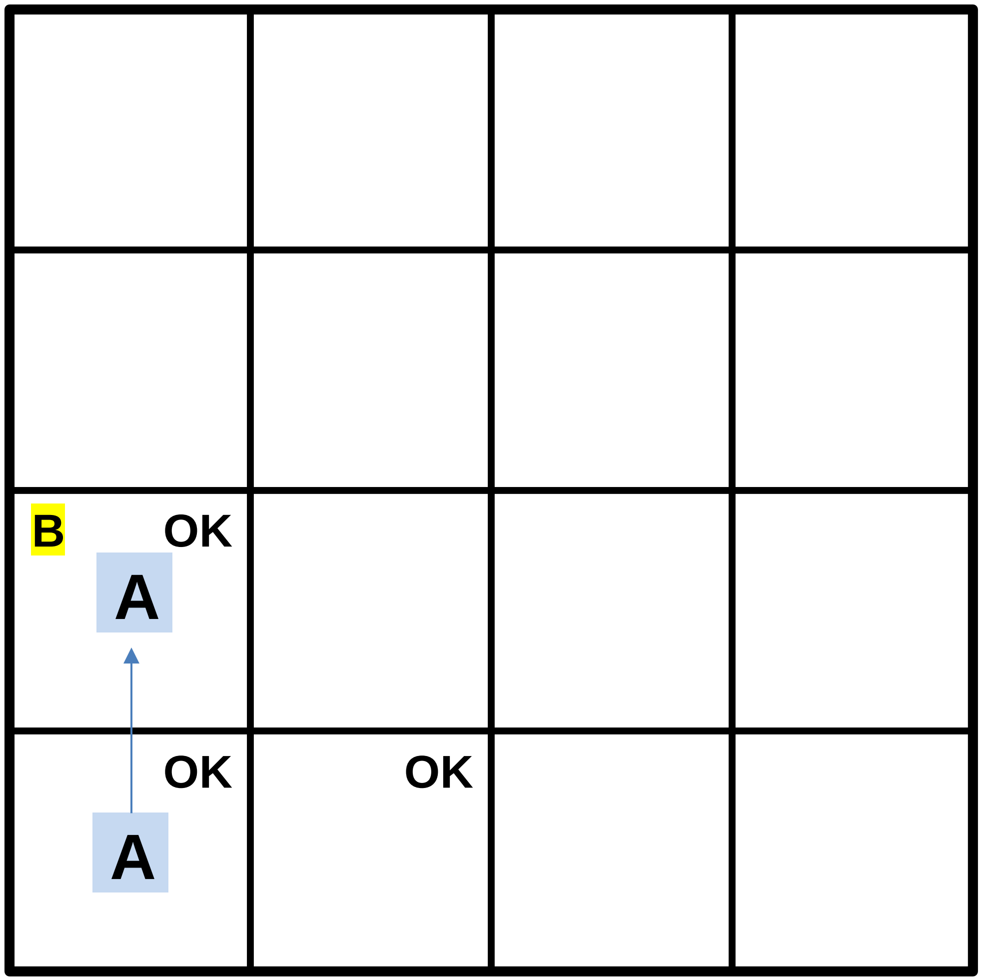
P= Pit

S= Stench

W=Wumpus

OK = Safe

Exploring a Wumpus world



A= Agent

B= Breeze

G= Glitter

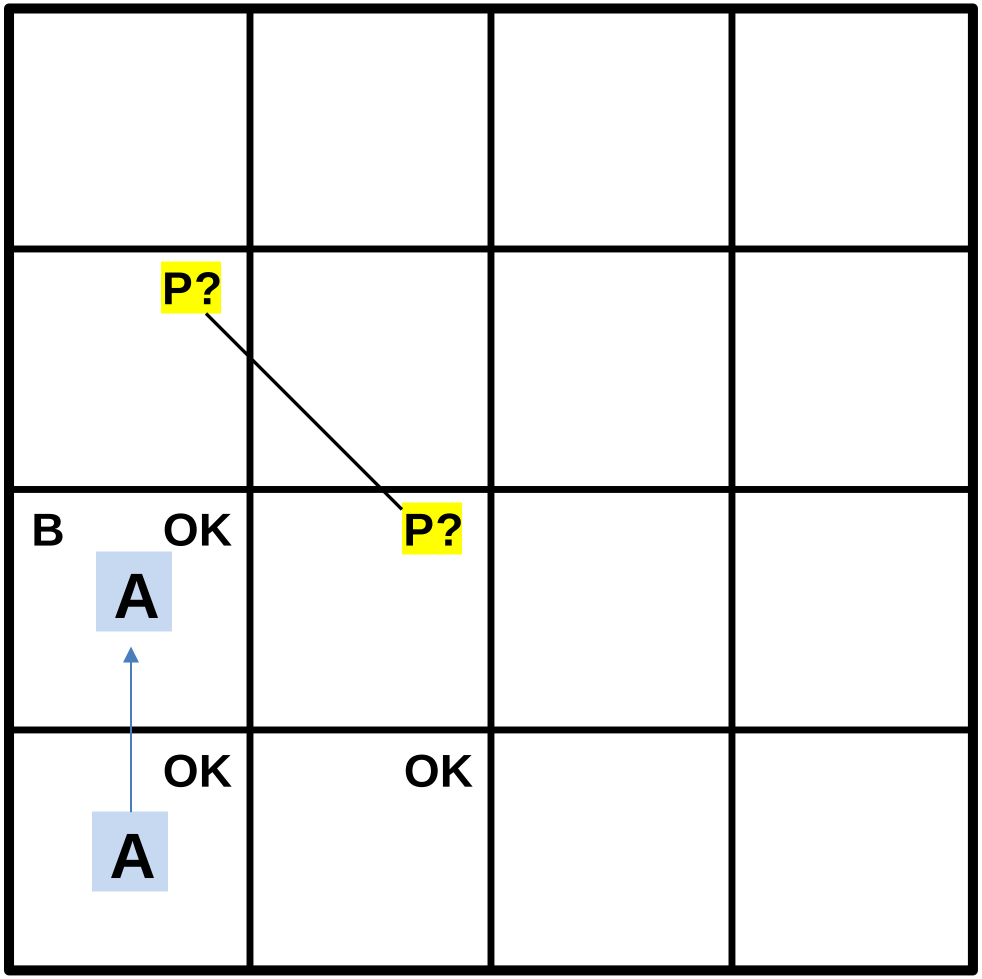
P= Pit

S= Stench

W=Wumpus

OK = Safe

Exploring a Wumpus world



A= Agent

B= Breeze

G= Glitter

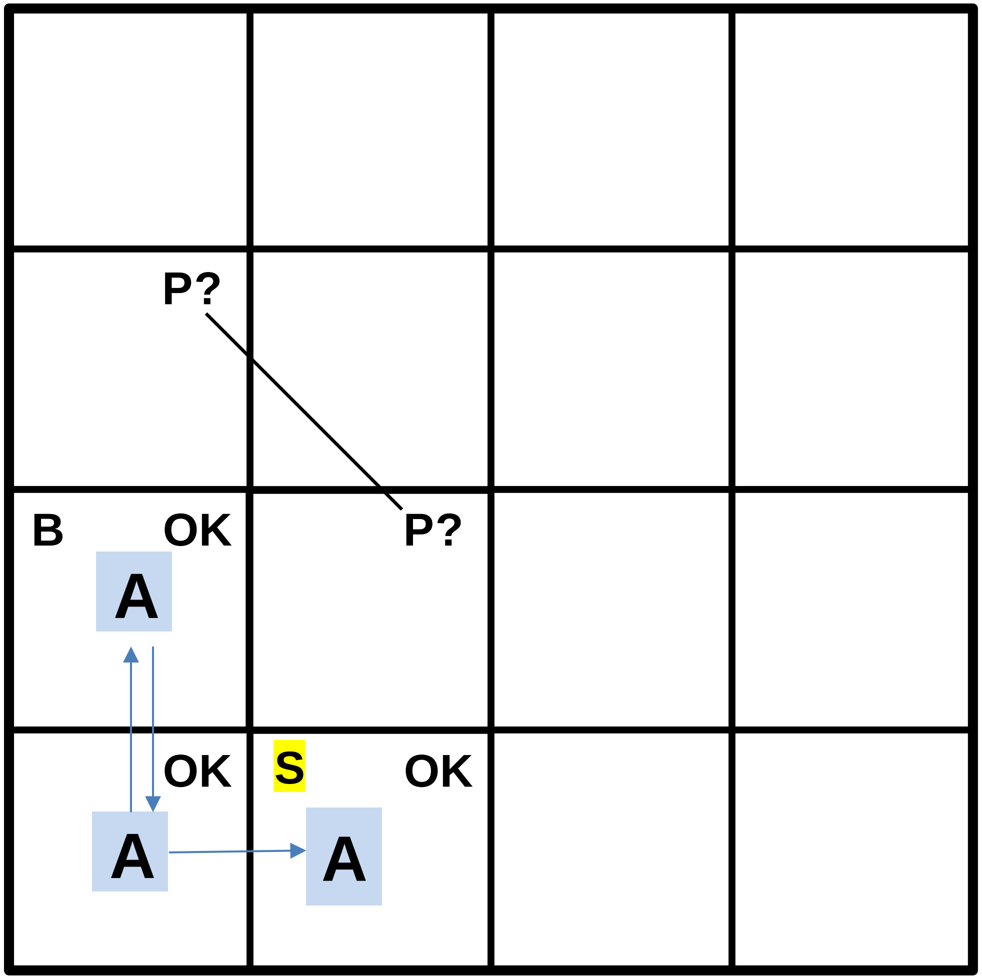
P= Pit

S= Stench

W=Wumpus

OK = Safe

Exploring a Wumpus world



A= Agent

B= Breeze

G= Glitter

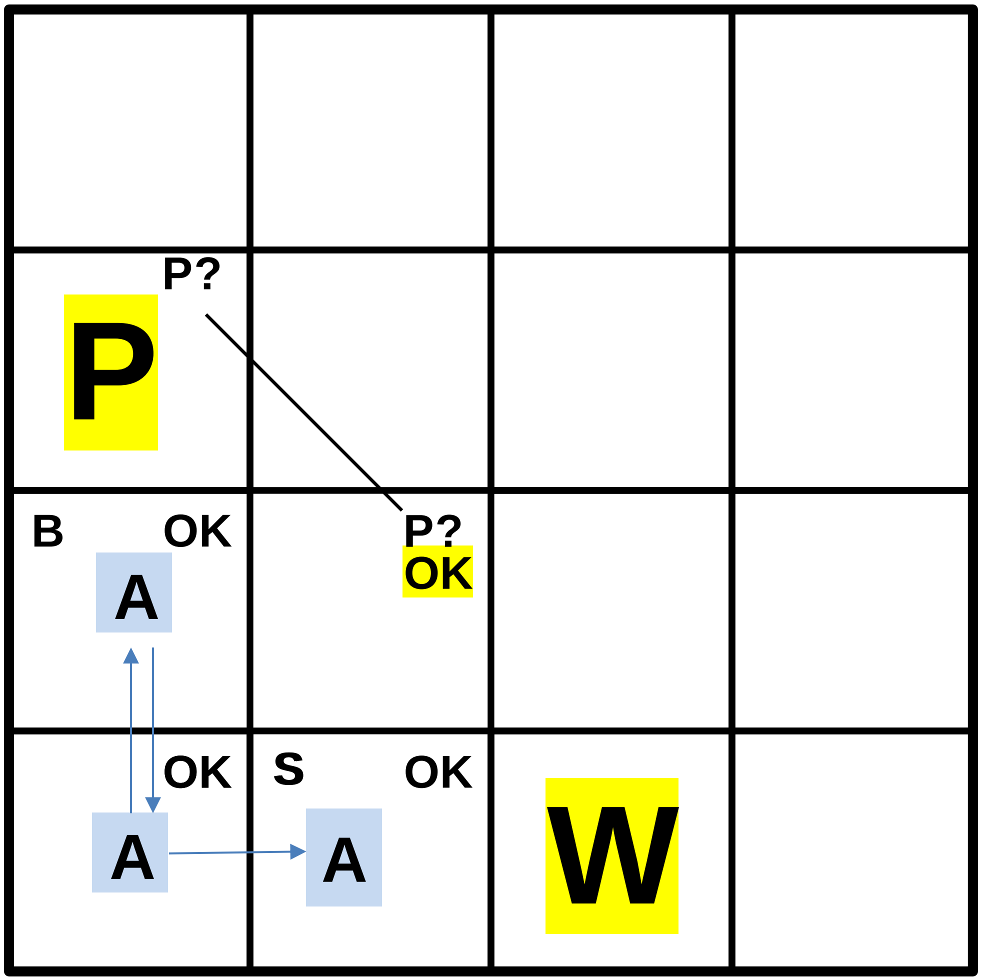
P= Pit

S= Stench

W=Wumpus

OK = Safe

Exploring a Wumpus world



A= Agent

B= Breeze

G= Glitter

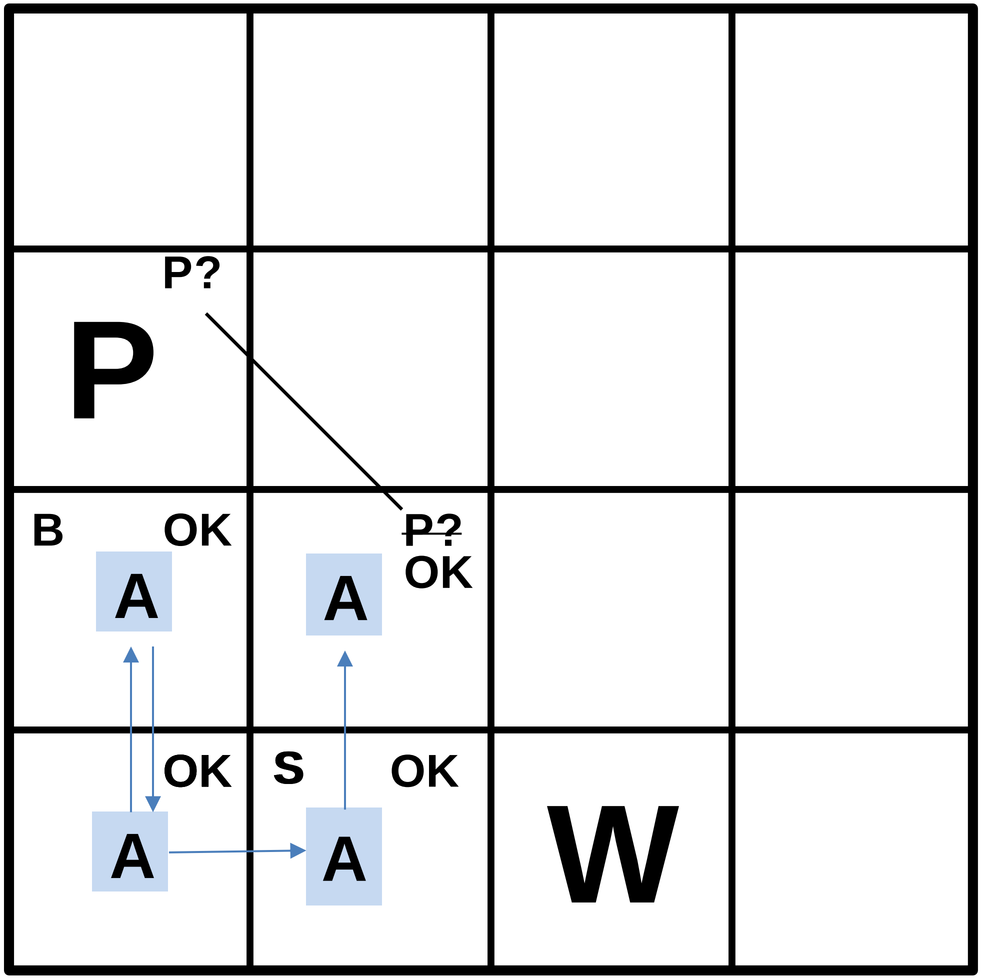
P= Pit

S= Stench

W=Wumpus

OK = Safe

Exploring a Wumpus world



A= Agent

B= Breeze

G= Glitter

P= Pit

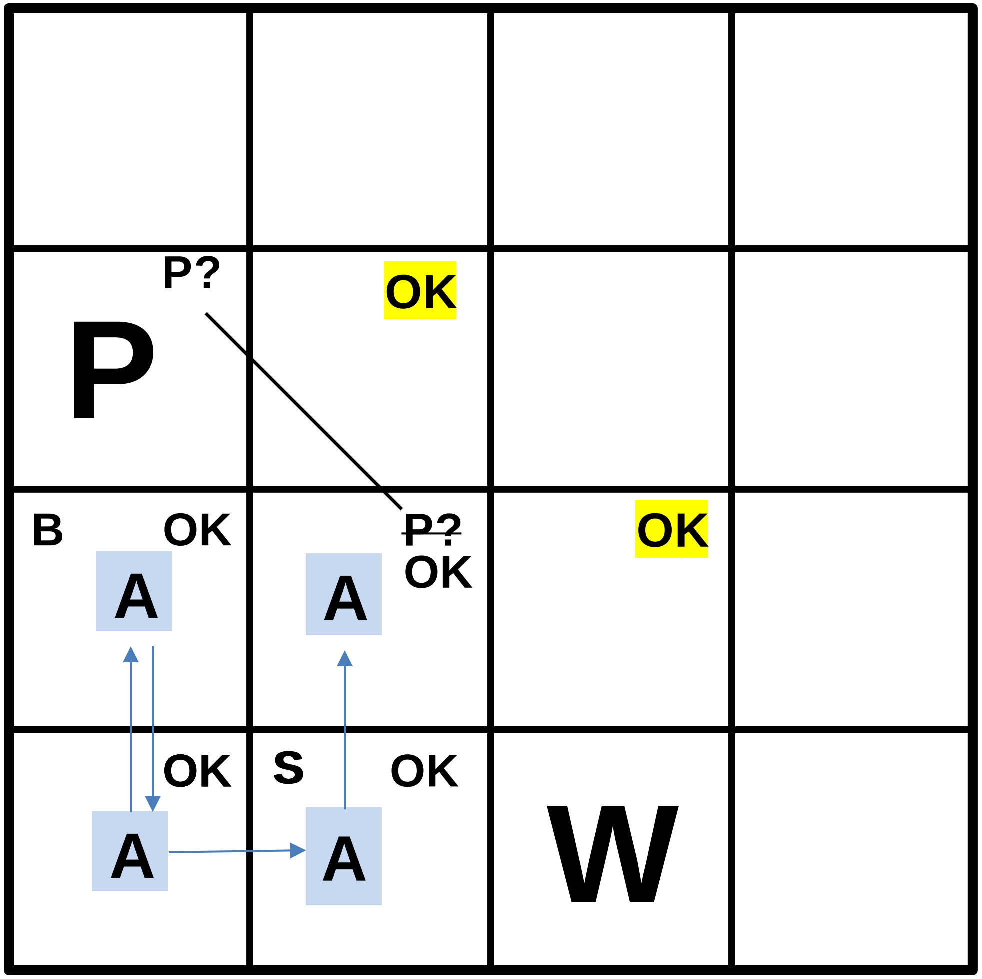
S= Stench

W=Wumpus

OK = Safe

Pearson

Exploring a Wumpus world



A= Agent

B= Breeze

G= Glitter

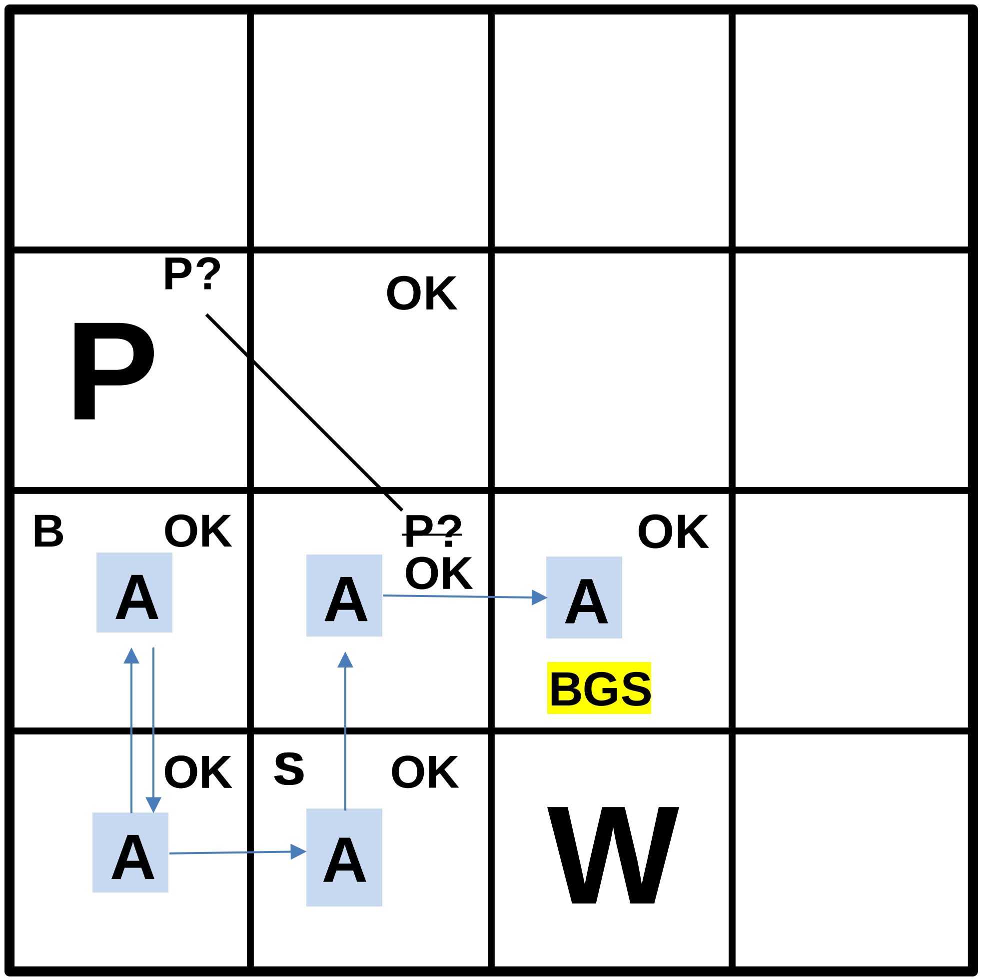
P= Pit

S= Stench

W=Wumpus

OK = Safe

Exploring a Wumpus world



A= Agent

B= Breeze

G= Glitter

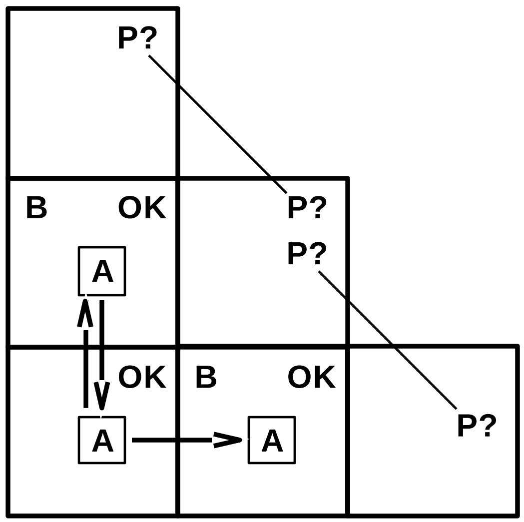
P= Pit

S= Stench

W=Wumpus

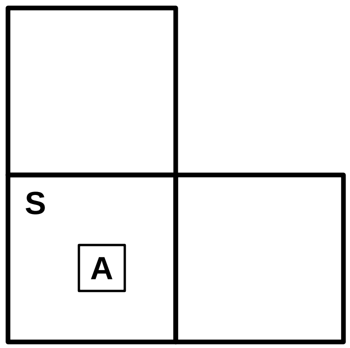
OK = Safe

Other tight spots



Breeze in (1,2) and (2,1)
$\Rightarrow$ no safe actions

Assuming pits uniformly distributed, (2,2) has pit w/ prob 0.86, vs. 0.31



Smell in (1,1)
$\Rightarrow$ cannot move

Can use a strategy of coercion:
- shoot straight ahead
- wumpus was there $\Rightarrow$ dead $\Rightarrow$ safe
- wumpus wasn't there $\Rightarrow$ safe

Logic

Formal representation of reasoning

Syntax: *specifies all the sentences that are well formed.*

Semantics: define the "meaning" of sentences

define truth of a sentence in a world

E.g., the language of arithmetic

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

$x + 2 \geq y$ is true iff 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 $x = 0, \ y = 6$

Entailment

$$KB \models \alpha$$

Knowledge base KB entails sentence α
if and only if
α is true in all worlds where KB is true

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

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

Note: brains process syntax (of some sort)

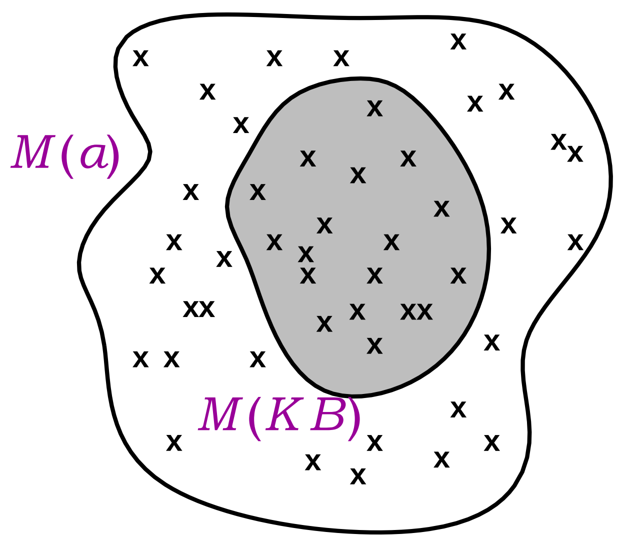
Models

formally structured worlds with respect to which truth can be evaluated

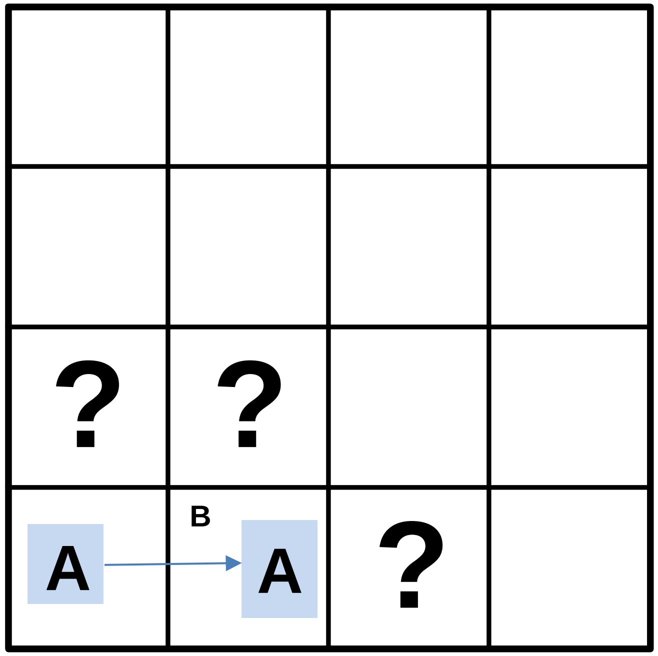
We say m is a model of a sentence α if α is true in m

$M(\alpha)$ is the set of all models of α

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

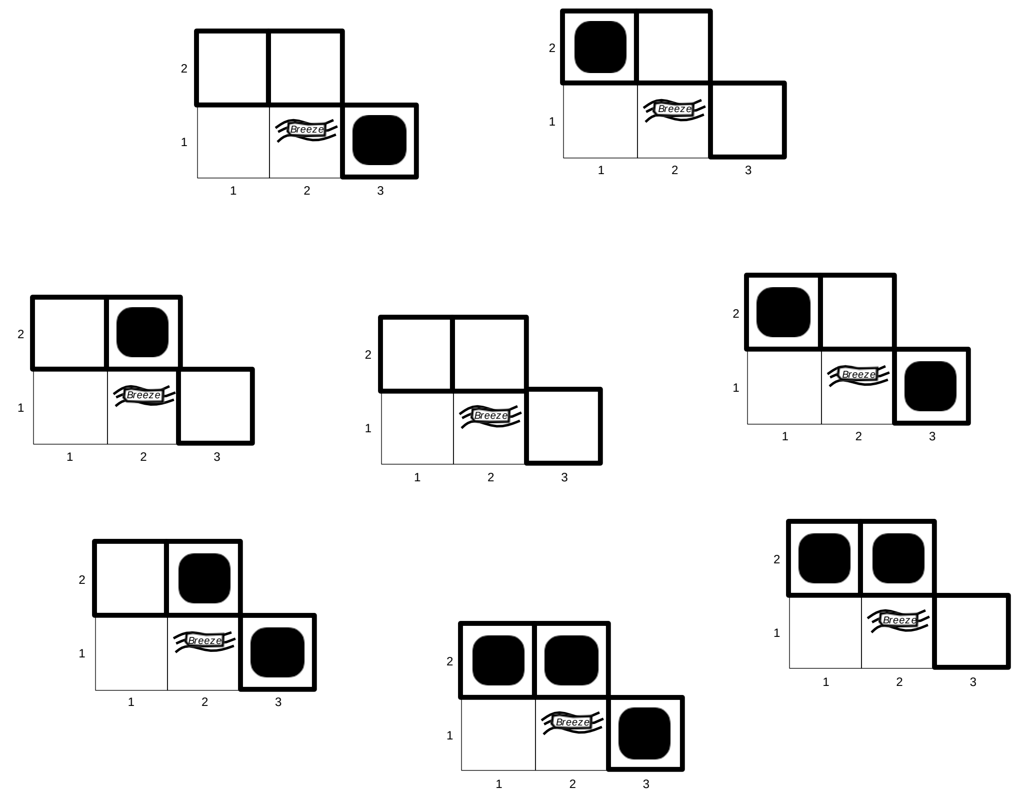


Entailment in the Wumpus world

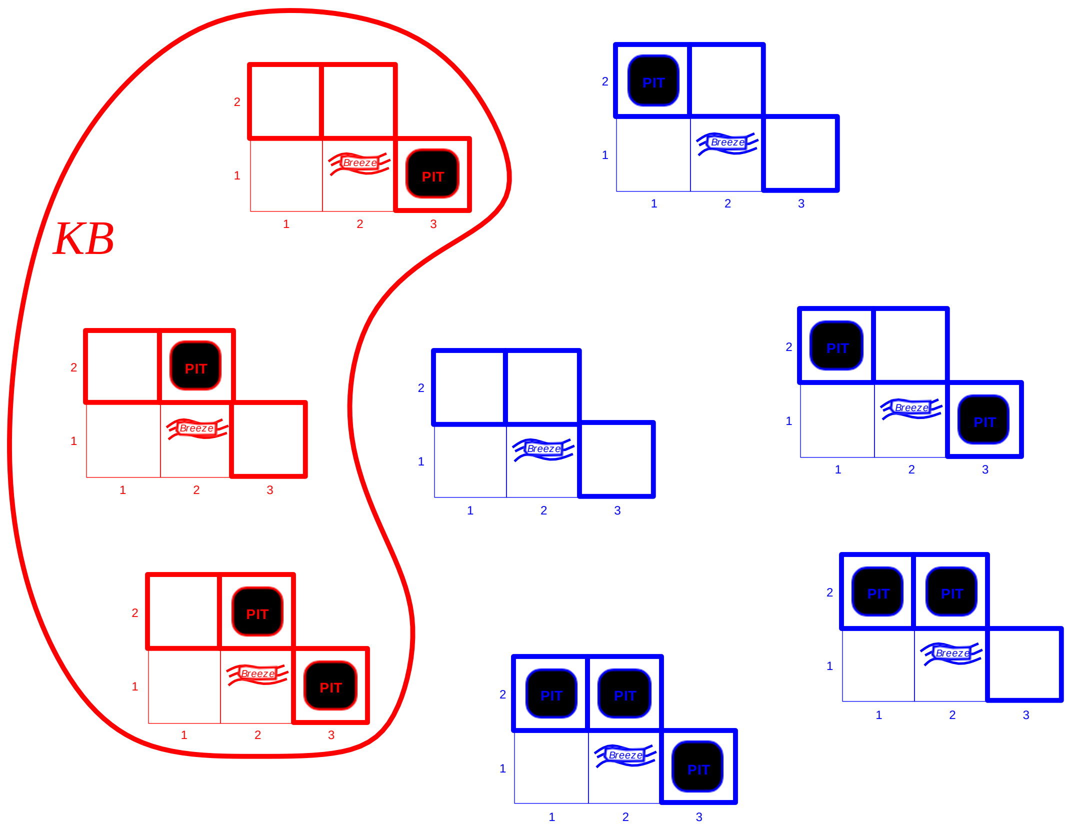


There are 8 possible models

Wumpus models

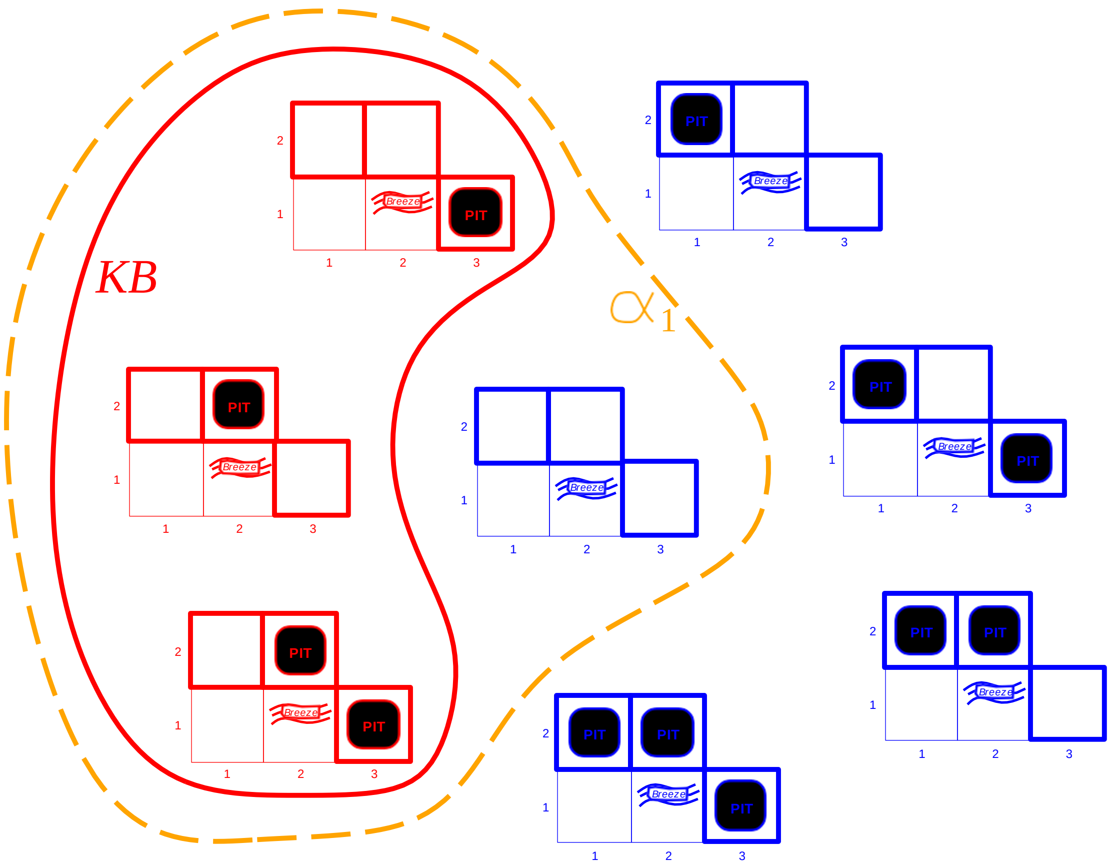


Wumpus models



KB = wumpus-world rules + observations

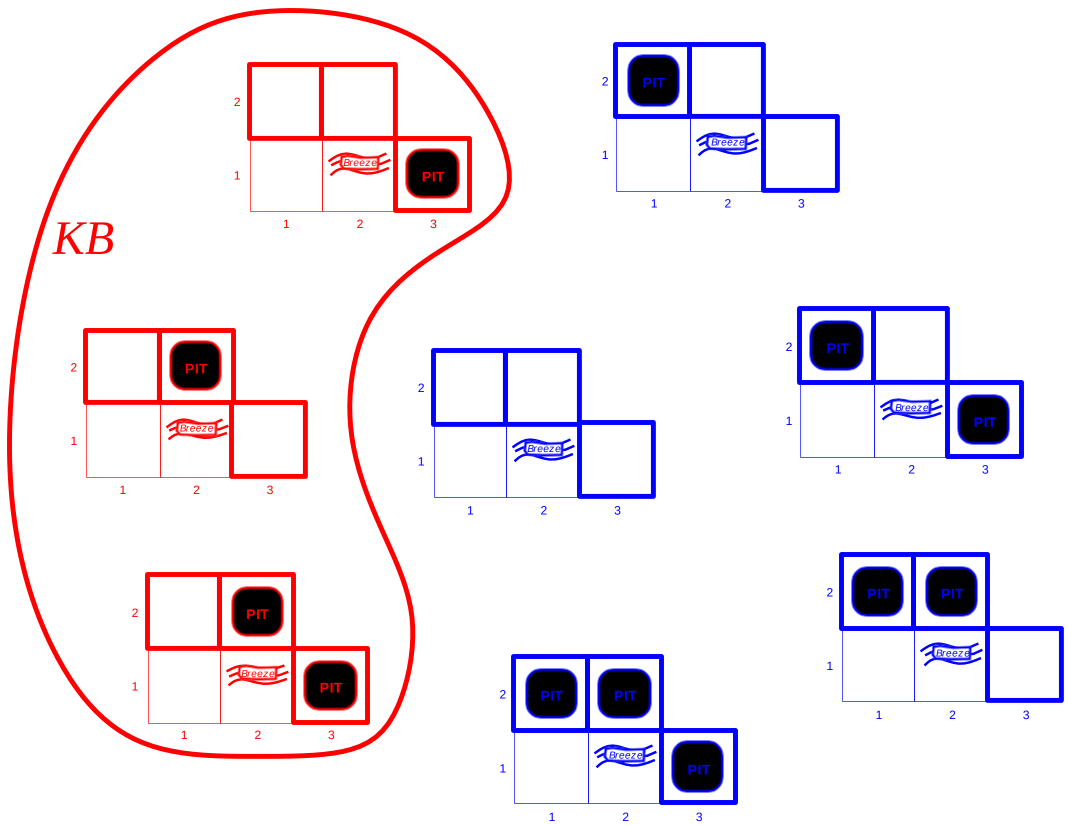
Wumpus models



KB = wumpus-world rules + observations

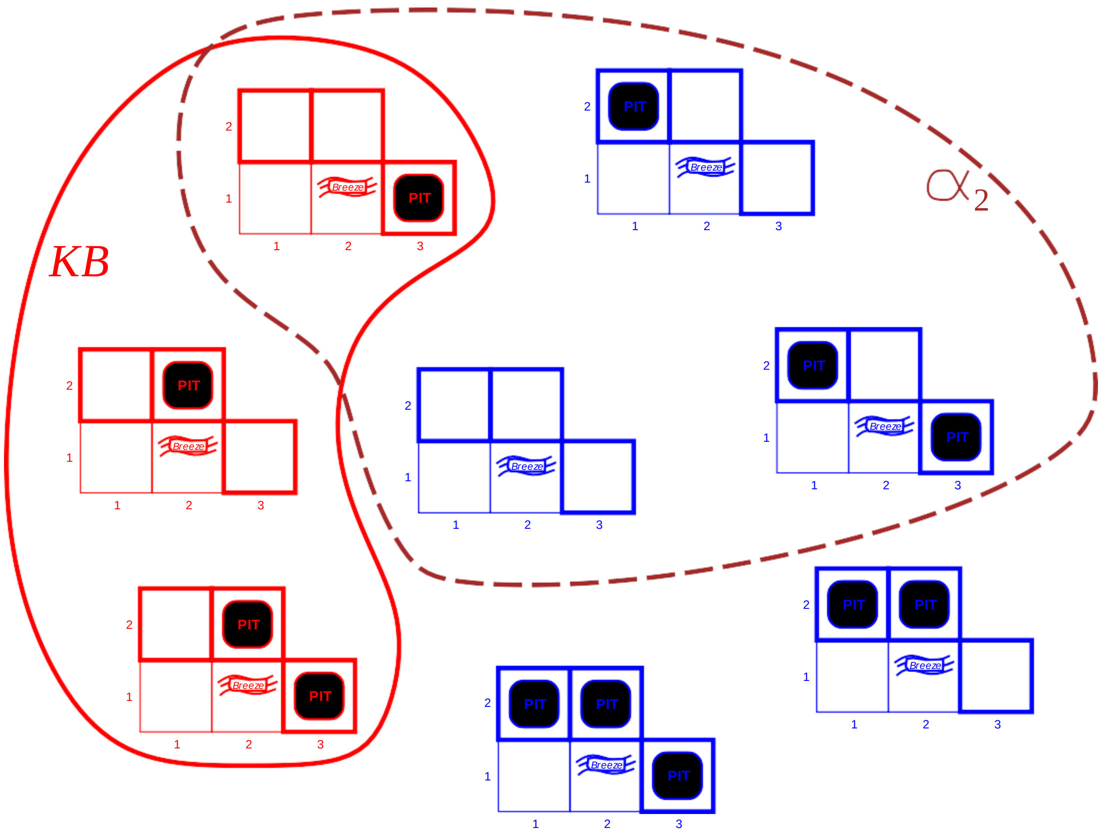
α_1 = "[1,2] is safe", $KB \models \alpha_1$, proved by model checking

Wumpus models



KB = wumpus-world rules + observations

Wumpus models



KB = wumpus-world rules + observations

α_2 = "[2,2] is safe", KB *does not entail* α_2

Model Checking

enumerating models
and showing that the sentence must hold in all models.

Inference

the process of deriving conclusions from premises using valid reasoning.

KB $\vdash_i$ α

sentence a can be derived from KB by procedure i

Soundness: i is sound if

whenever KB $\vdash_i$ α , it is also true that $KB \models a$

Completeness: i is complete if

whenever $KB \models a$, it is also true that KB $\vdash_i$ α

Propositional logic: Syntax

Sentences are :

- Atomic = **proposition symbols** P, Q …

- Complex = constructed from simpler senteces using **logical connectives** …

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. $P_{1,2}$ $P_{2,2}$ $P_{3,1}$
$\quad true$ $true$ $false$

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
i.e.,	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.,

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

Pearson

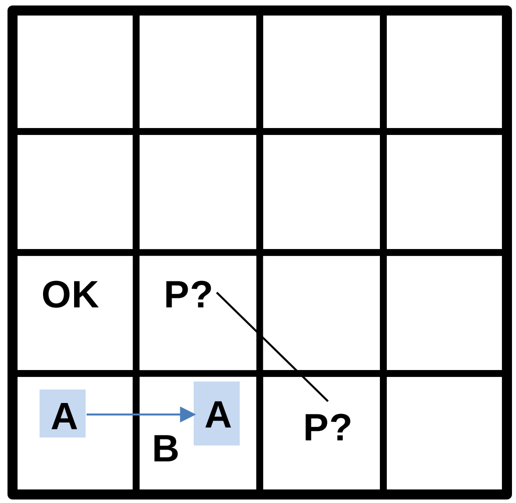
Truth tables for connectives

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*

Pearson

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 $[i,j]$.



R_1	$\neg P_{1,1}$
R_2	$B_{1,1} \Leftrightarrow (P_{1,2} \vee P_{2,1})$
R_3	$B_{2,1} \Leftrightarrow (P_{1,1} \vee P_{2,2} \vee P_{3,1})$
R_4	$\neg B_{1,1}$
R_5	$B_{2,1}$

"Pits cause breezes in adjacent squares"

Truth tables for inference

$B_{1,1}$	$B_{2,1}$	$P_{1,1}$	$P_{1,2}$	$P_{2,1}$	$P_{2,2}$	$P_{3,1}$	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*
.	.	.	.	.	.	.	.	.	.	.	.	.
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*
.	.	.	.	.	.	.	.	.	.	.	.	.
true	*true*	*true*	*true*	*true*	*true*	*true*	*false*	*true*	*true*	*false*	*true*	*false*

A truth table constructed for the knowledge base given in the text. KB is true if R1 through R5 are true, which occurs in just 3 of the 128 rows (the ones underlined in the right-hand column). In all 3 rows, P1,2 is false, so there is no pit in [1,2]. On the other hand, there might (or might not) be a pit in [2,2].

Inference by enumeration

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))
```

$O(2^n)$ for n symbols; problem is co-NP-complete

Propositional Theorem Proving

applying rules of inference directly to the sentences
in our knowledge base to construct a proof of the desired sentence without consulting models.

Some additional concepts…

Logical equivalence

Two sentences are logically equivalent iff true in same models:
$\alpha \equiv \beta$ if and only if $\alpha \models \beta$ and $\beta \models \alpha$

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

Pearson

Some additional concepts…

Validity and satisfiability

A sentence is valid if it is true in all models

e.g., $True$, $A \lor \neg A$, $A \Rightarrow A$, $(A \land (A \Rightarrow B)) \Rightarrow B$

Deduction Theorem:
$KB \models \alpha$ if and only if $(KB \Rightarrow \alpha)$ is valid

A sentence is satisfiable if it is true in some model → **SAT** problem

e.g., $A \lor B$, C

A sentence is unsatisfiable if it is true in no models → Proof by contradiction

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

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

Proof methods

Application of inference rules

- Sound generation of new sentences
- Proof = a sequence of inference rule applications
- Typically require translation of sentences into a normal form

Model checking

truth table enumeration (always exponential in n)

improved backtracking,

e.g., DPLL heuristic search in model space (sound but incomplete)

Modus Ponens

$$\frac{\alpha \Rightarrow \beta \quad \alpha}{\beta}$$

Premise(s)

conclusion

And-Elimination

$$\frac{\alpha \wedge \beta}{\alpha}$$

Biconditional-elim

$$\frac{\alpha \Leftrightarrow \beta}{(\alpha \Rightarrow \beta) \wedge (\beta \Rightarrow \alpha)}$$

Contraposition

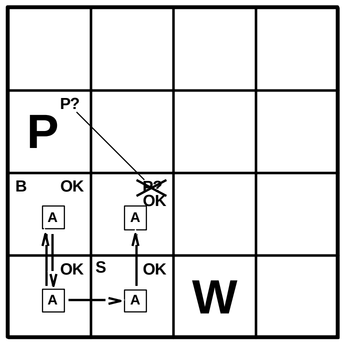
$$\frac{(\alpha \Rightarrow \beta)}{(\neg \beta \Rightarrow \neg \alpha)}$$

Unit Resolution

$$\frac{\alpha \vee \beta \quad \neg \alpha}{\beta}$$

De Morgan

$$\frac{\neg(\beta \vee \alpha)}{\neg \beta \wedge \neg \alpha}$$



R_1	$\neg P_{1,1}$
R_2	$B_{1,1} \Leftrightarrow (P_{1,2} \lor P_{2,1})$
R_3	$B_{2,1} \Leftrightarrow (P_{1,1} \lor P_{2,2} \lor P_{3,1})$
R_4	$\neg B_{1,1}$
R_5	$B_{2,1}$
R_{11}	$\neg B_{1,2}$
R_{12}	$B_{1,2} \Leftrightarrow (P_{1,1} \lor P_{2,2} \lor P_{1,3})$

$$\frac{B_{1,1} \Leftrightarrow (P_{1,2} \lor P_{2,1})}{(B_{1,1} \Rightarrow (P_{1,2} \lor P_{2,1})) \land ((P_{1,2} \lor P_{2,1}) \Rightarrow B_{1,1})} \quad \text{Biconditional-elim}$$

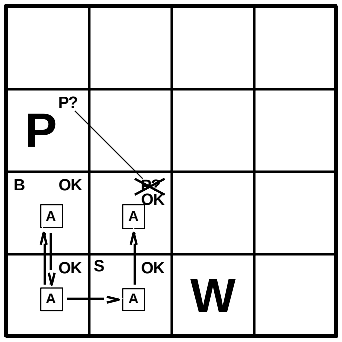
$$\frac{(B_{1,1} \Rightarrow (P_{1,2} \lor P_{2,1})) \land ((P_{1,2} \lor P_{2,1}) \Rightarrow B_{1,1})}{(P_{1,2} \lor P_{2,1}) \Rightarrow B_{1,1}} \quad \text{And-Elimination}$$

$$\frac{(P_{1,2} \lor P_{2,1}) \Rightarrow B_{1,1}}{\neg B_{1,1} \Rightarrow \neg(P_{1,2} \lor P_{2,1})} \quad \text{Contraposition}$$

$$\frac{\neg B_{1,1} \Rightarrow \neg(P_{1,2} \lor P_{2,1}) \qquad \neg B_{1,1}}{\neg(P_{1,2} \lor P_{2,1})} \quad \text{Modus Ponens}$$

$$\frac{\neg(P_{1,2} \lor P_{2,1})}{\neg P_{1,2} \land \neg P_{2,1}} \quad \text{De Morgan}$$

Pearson



R_1	$\neg P_{1,1}$
R_2	$B_{1,1} \Leftrightarrow (P_{1,2} \vee P_{2,1})$
R_3	$B_{2,1} \Leftrightarrow (P_{1,1} \vee P_{2,2} \vee P_{3,1})$
R_4	$\neg B_{1,1}$
R_5	$B_{2,1}$
R_{11}	$\neg B_{1,2}$
R_{12}	$B_{1,2} \Leftrightarrow (P_{1,1} \vee P_{2,2} \vee P_{1,3})$
R_{13}	$\neg P_{2,2}$
R_{14}	$\neg P_{1,3}$

$$\frac{B_{2,1} \Leftrightarrow (P_{1,1} \vee P_{2,2} \vee P_{1,3})}{(B_{2,1} \Rightarrow (P_{1,1} \vee P_{2,2} \vee P_{1,3})) \wedge ((P_{1,1} \vee P_{2,2} \vee P_{1,3}) \Rightarrow B_{2,1}} \quad \text{Biconditional-elim}$$

$$\frac{(B_{2,1} \Rightarrow (P_{1,1} \vee P_{2,2} \vee P_{1,3})) \wedge ((P_{1,1} \vee P_{2,2} \vee P_{1,3}) \Rightarrow B_{2,1}}{(B_{2,1} \Rightarrow (P_{1,1} \vee P_{2,2} \vee P_{1,3}))} \quad \text{And-Elimination}$$

$$\frac{(B_{2,1} \Rightarrow (P_{1,1} \vee P_{2,2} \vee P_{1,3})) \qquad B_{2,1}}{(P_{1,1} \vee P_{2,2} \vee P_{1,3})} \quad \text{Modus Ponens}$$

$$\frac{(P_{1,1} \vee P_{2,2} \vee P_{1,3}) \qquad \neg P_{2,2}}{(P_{1,1} \vee P_{1,3})} \quad \text{Unit Resolution}$$

$$\frac{(P_{1,1} \vee P_{1,3}) \qquad \neg P_{1,1}}{P_{1,3}} \quad \text{Unit Resolution}$$

Pearson

Resolution

Clause : disjunction of literals

Resolution inference rule - from two **clauses** we obtain one :

$$\frac{J_1 \vee \cdots \vee J_k \qquad m_1 \vee \cdots \vee m_n}{J_1 \vee \cdots \vee J_{i-1} \vee J_{i+1} \vee \cdots \vee J_k \vee m_1 \vee \cdots \vee m_{j-1} \vee m_{j+1} \vee \cdots \vee m_n}$$

where J_i and m_j are complementary literals. 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

A resolution-based theorem prover can, for any sentences α and β in propositional logic, decide whether α |= β.

Conjunctive Normal Form (CNF)

conjunction of disjunctions of literals clauses

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

Every sentence of propositional logic is logically equivalent to a conjunction of clauses.

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:

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

Pearson

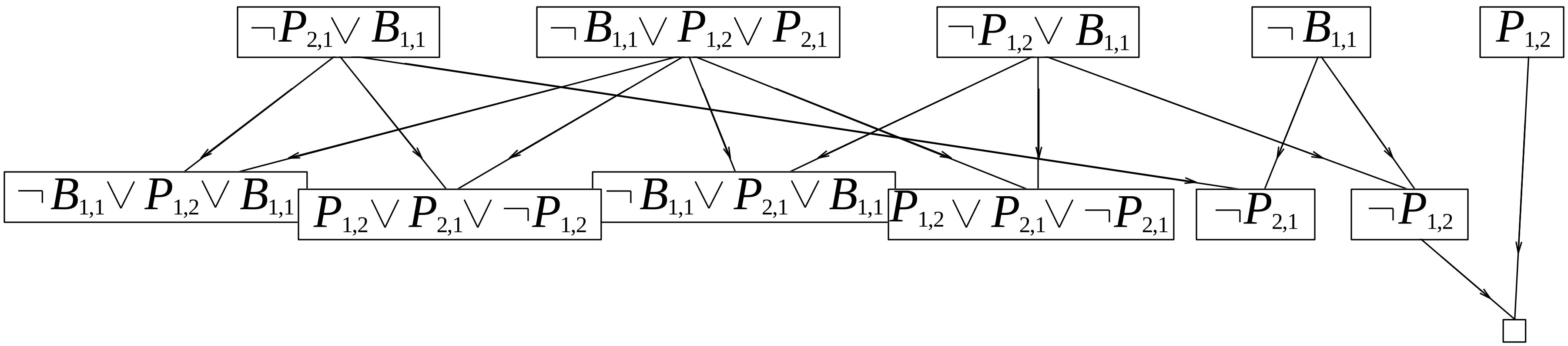
Resolution algorithm

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 Ci, Cj in clauses do
         resolvents ← PL-Resolve(Ci, Cj)
         if resolvents contains the empty clause then return true
         new ← new ∪ resolvents
      if new ⊆ clauses then return false
      clauses ← clauses ∪ new
```

Resolution example

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



Ground resolution theorem

If a group of clauses is unsatisfiable, the empty clause is included in the resolution closure of those clauses. (Completeness)

Forward and backward chaining

Horn clause = *disjunction of literals of which at most one is positive.*

Horn Form = conjunction of Horn clauses

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

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

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

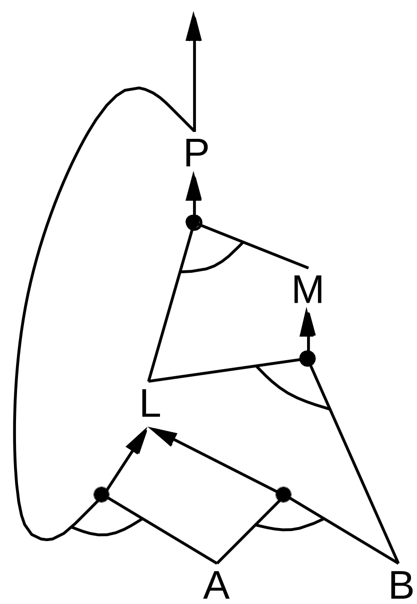
Can be used with forward chaining or backward chaining

in linear time

Forward chaining

Idea: *If all the premises of an implication are known, then its conclusion is added to the set of known facts. This process continues until the query q is added or until no further inferences can be made.*

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

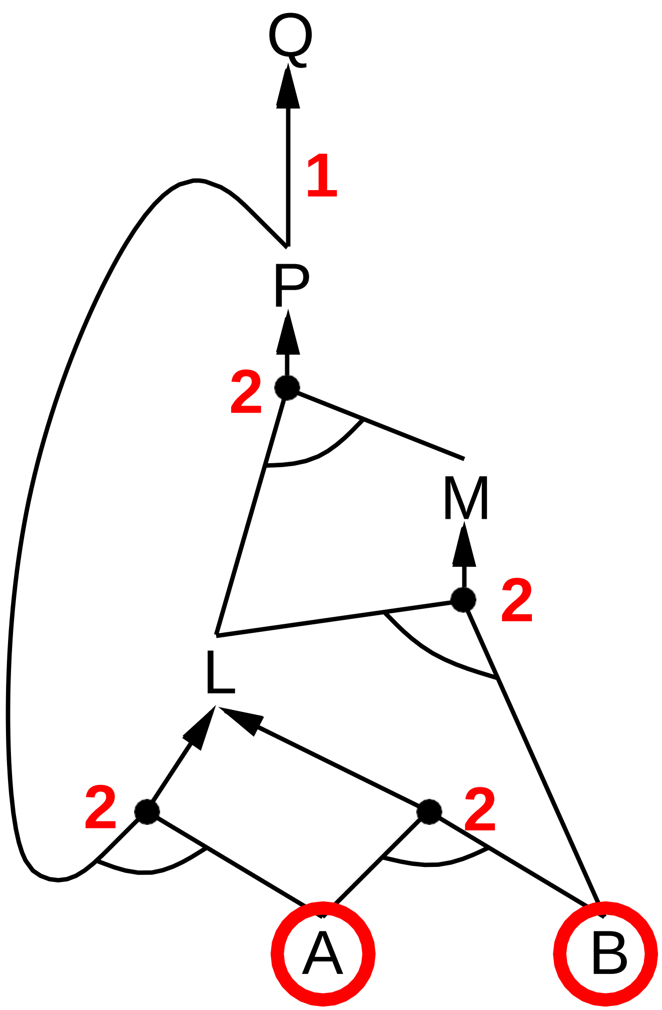


Forward chaining algorithm

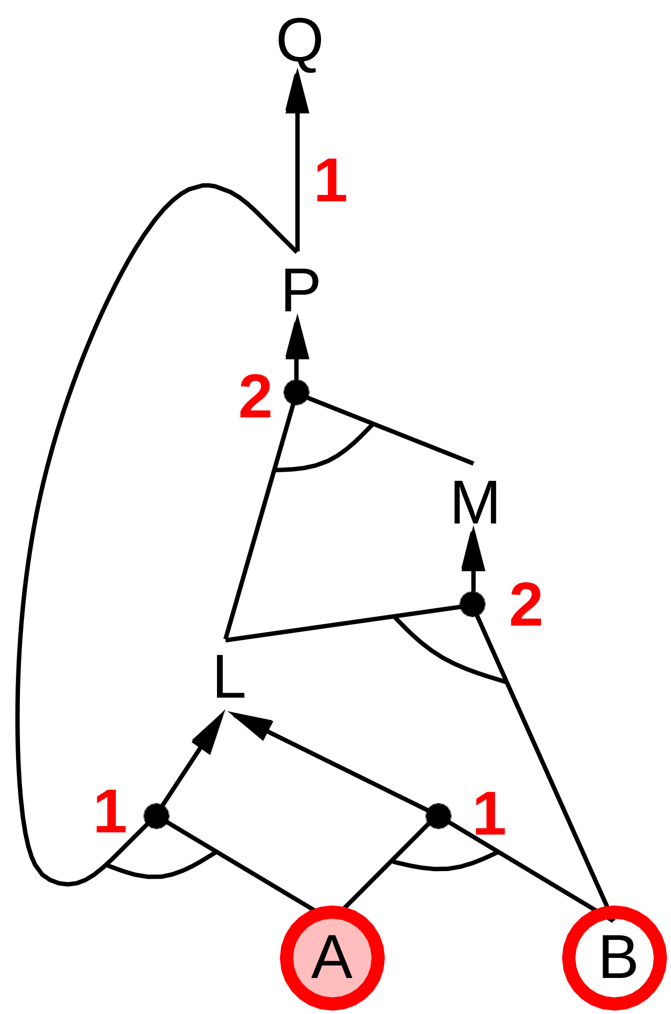
```
function PL-FC-Entails?(KB, q) returns true or false
   inputs: KB, the knowledge base, a set of propositional Horn clauses
           q, the query, a proposition symbol
   local variables: count, a table, indexed by clause, initially the number of premises
                    inferred, a table,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

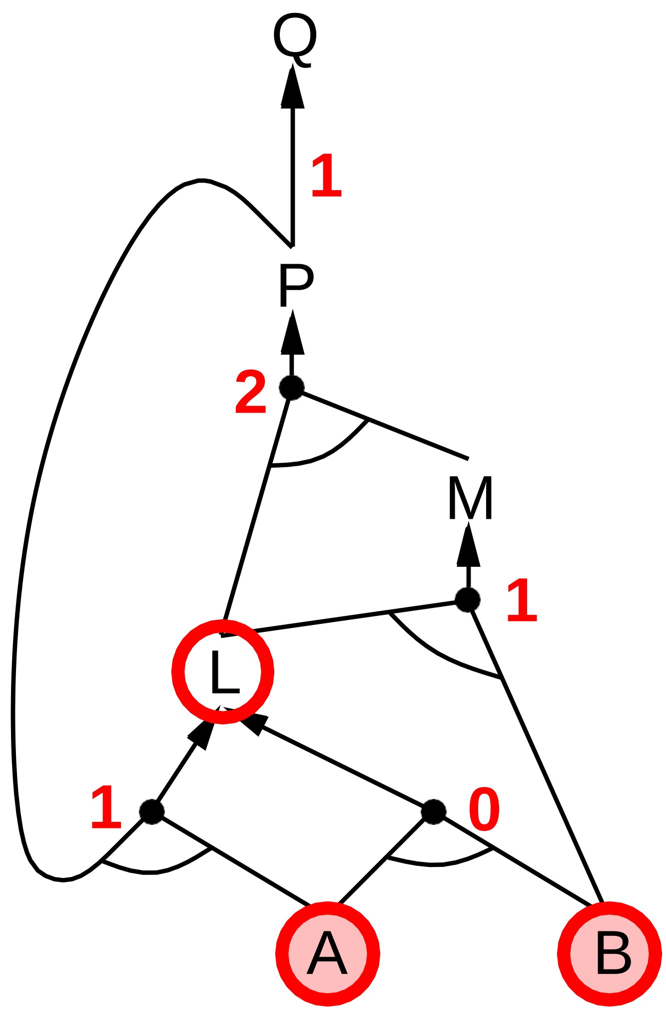
Forward chaining example



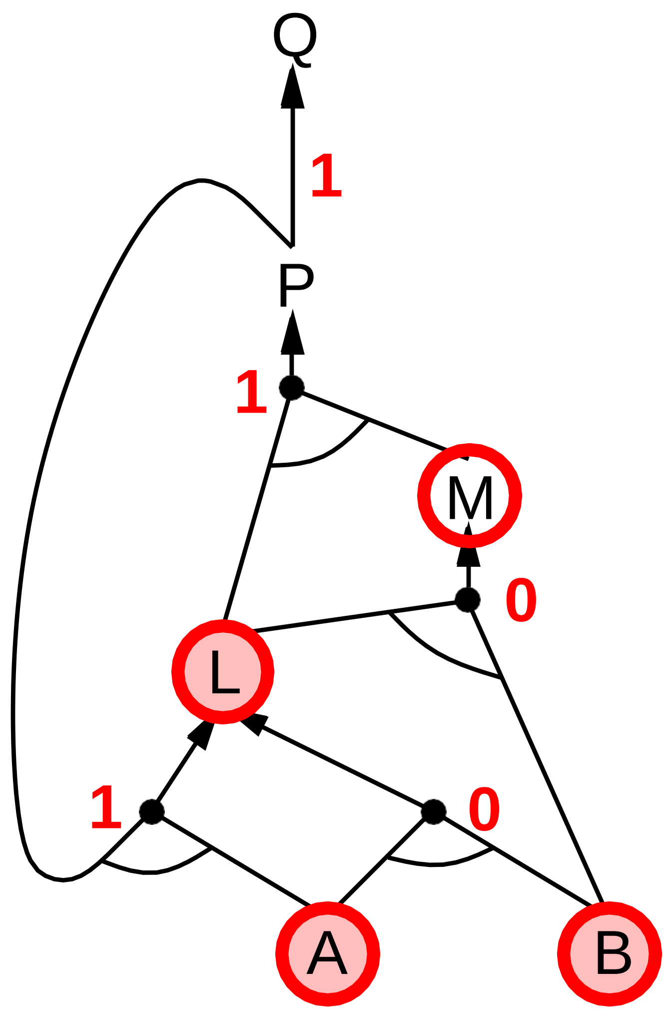
Forward chaining example



Forward chaining example

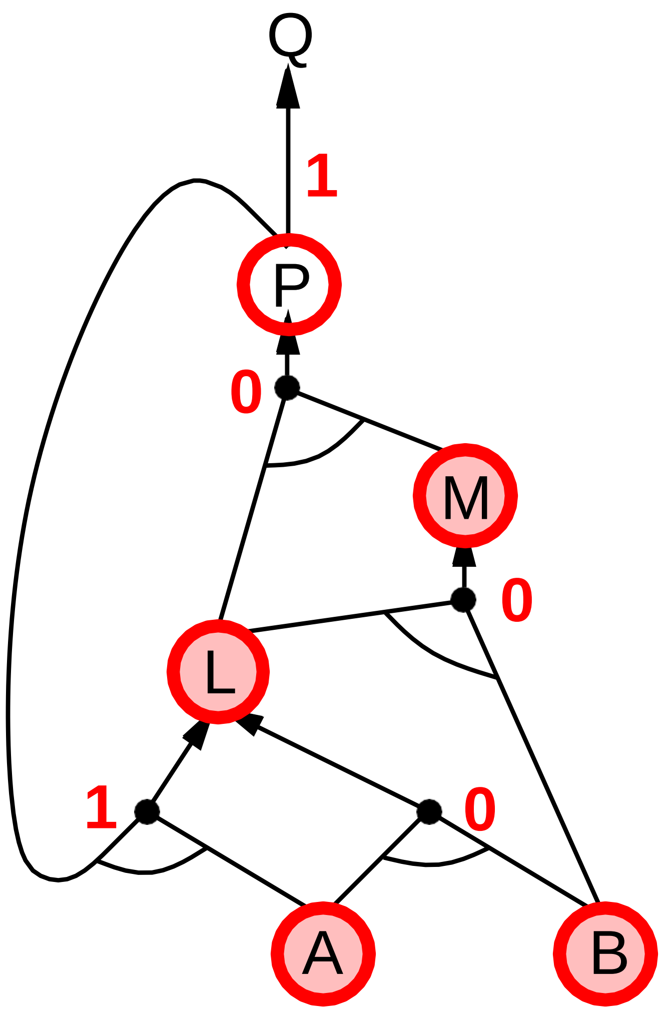


Forward chaining example

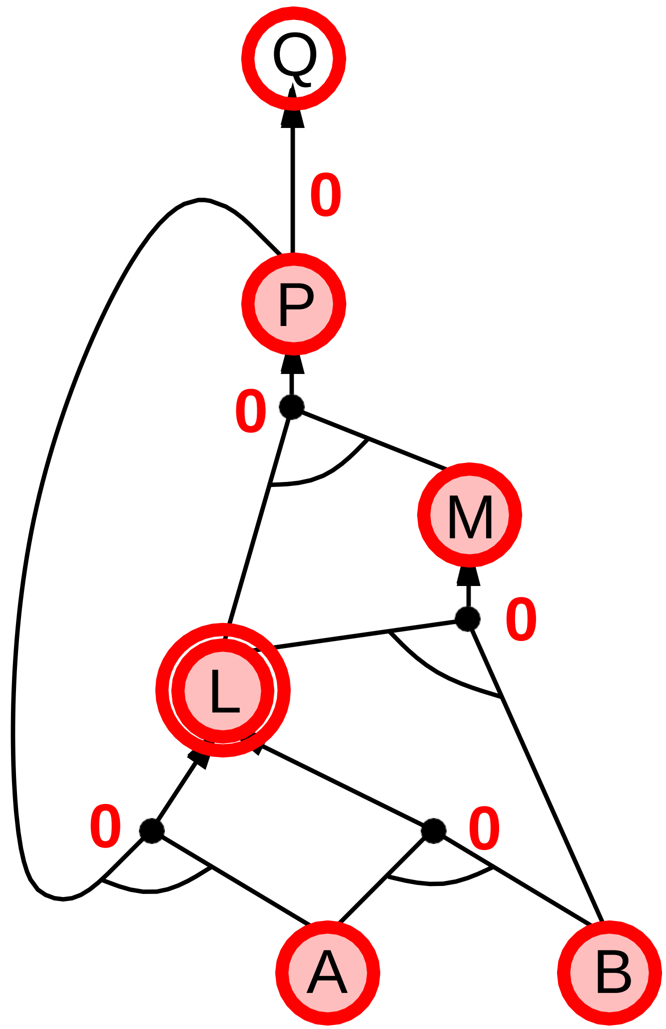


Pearson

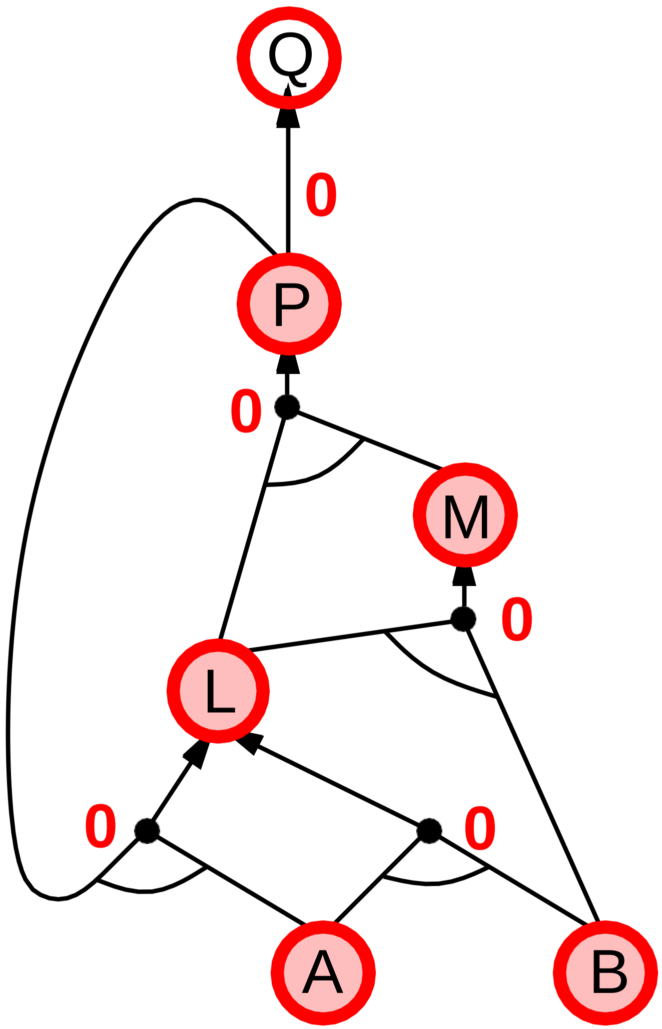
Forward chaining example



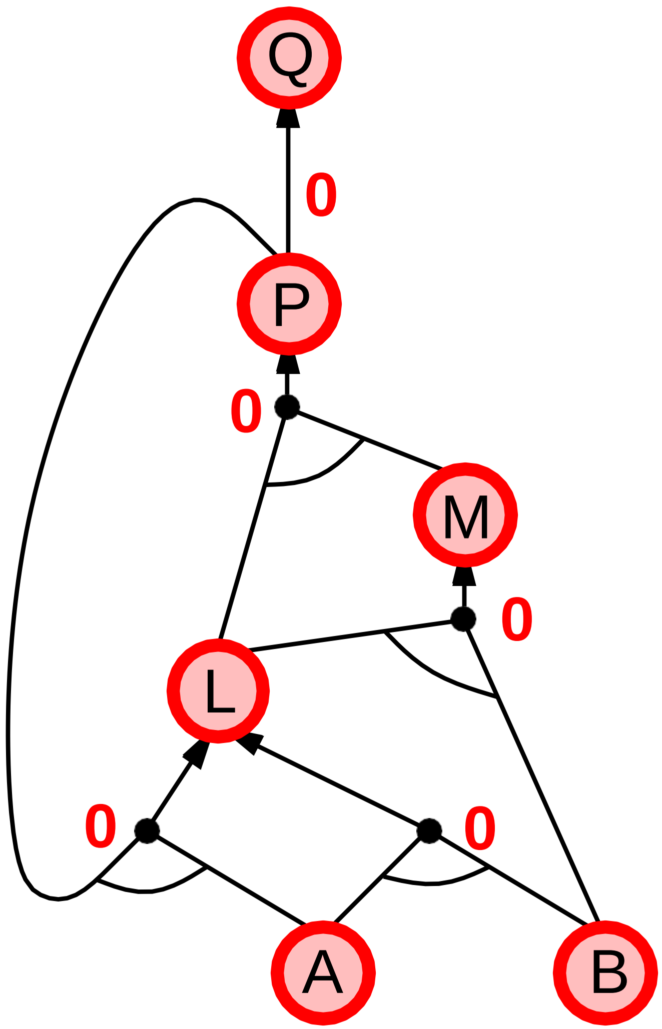
Forward chaining example



Forward chaining example



Forward chaining example



Proof of completeness

FC derives every atomic sentence that is entailed by KB

1. FC reaches a fixed point where no new atomic sentences are derived

2. Consider the final state as a model m, assigning true/false to symbols

3. Every clause in the original KB is true in m
   Proof: Suppose a clause $a_1 \wedge \ldots \wedge a_k \Rightarrow b$ is false in m
   Then $a_1 \wedge \ldots \wedge a_k$ is true in m and b is false in m
   Therefore the algorithm has not reached a fixed point!

4. Hence m is a model of KB

5. If $KB \models q$, q is true in every model of KB, including m

General idea: construct any model of KB by sound inference, check a

Backward chaining

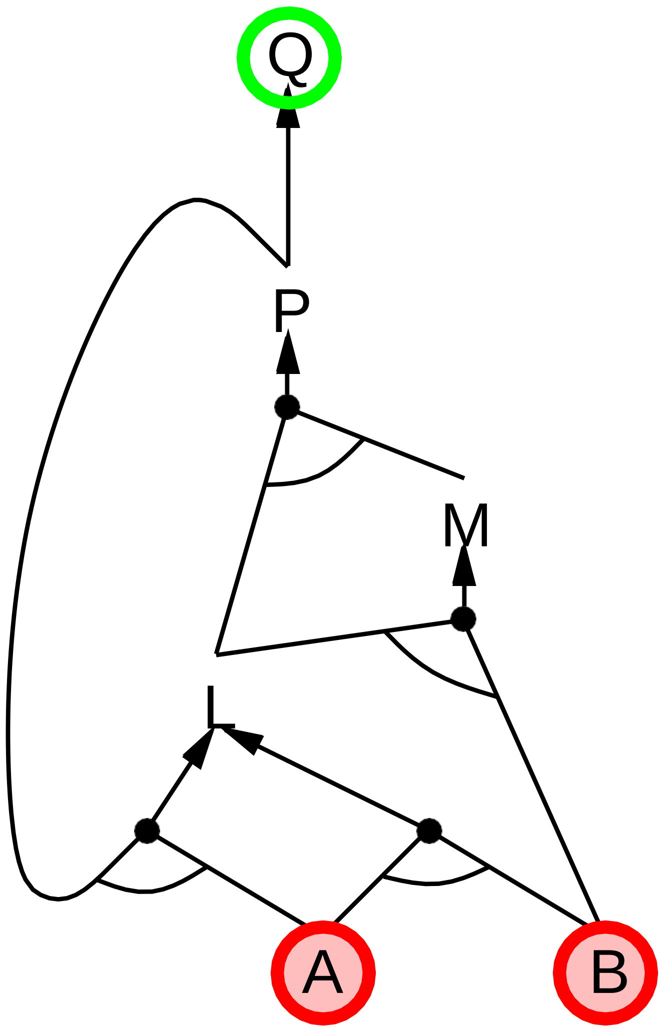
Idea :*If the query q is known to be true, then no work is needed. Otherwise, the algorithm finds those implications in the knowledge base whose conclusion is q. If all the premises of one of those implications can be proved true (by backward chaining), then q is true.*

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

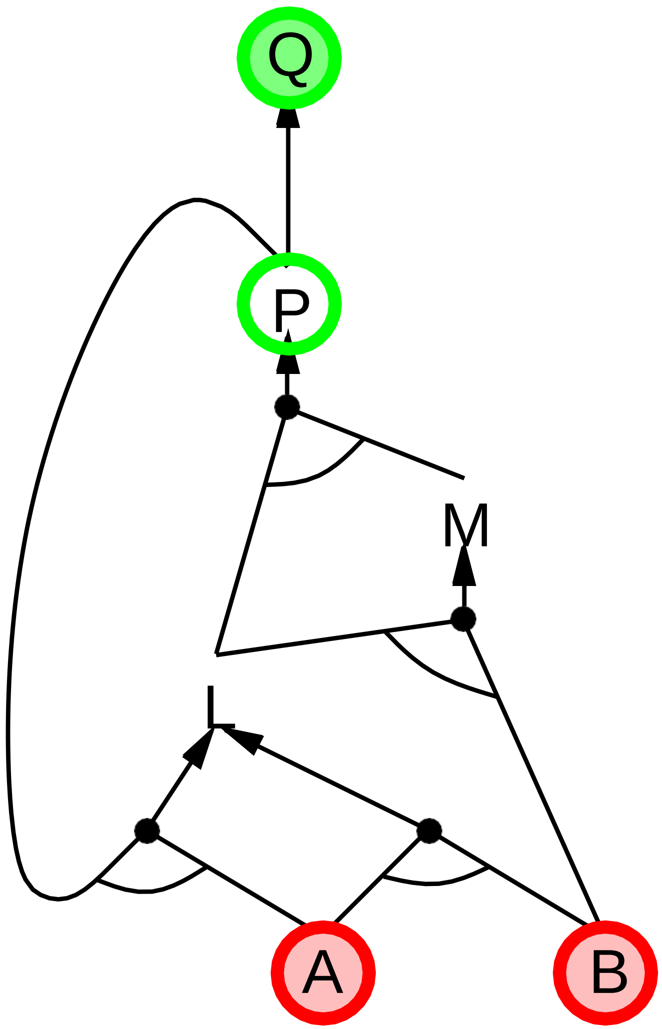
Avoid repeated work: check if new subgoal

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

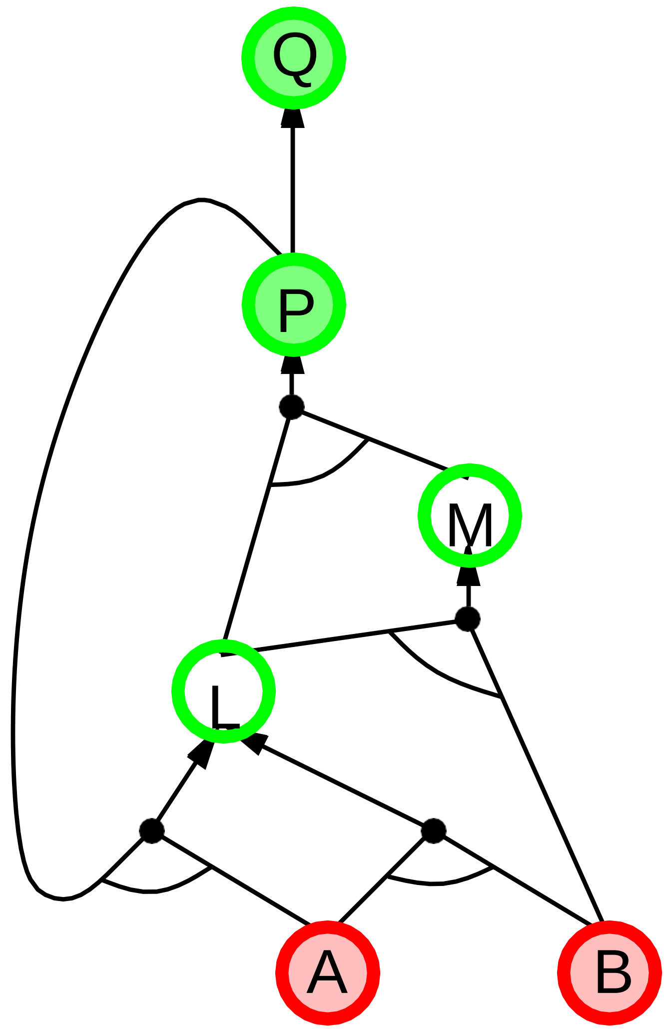
Backward chaining example



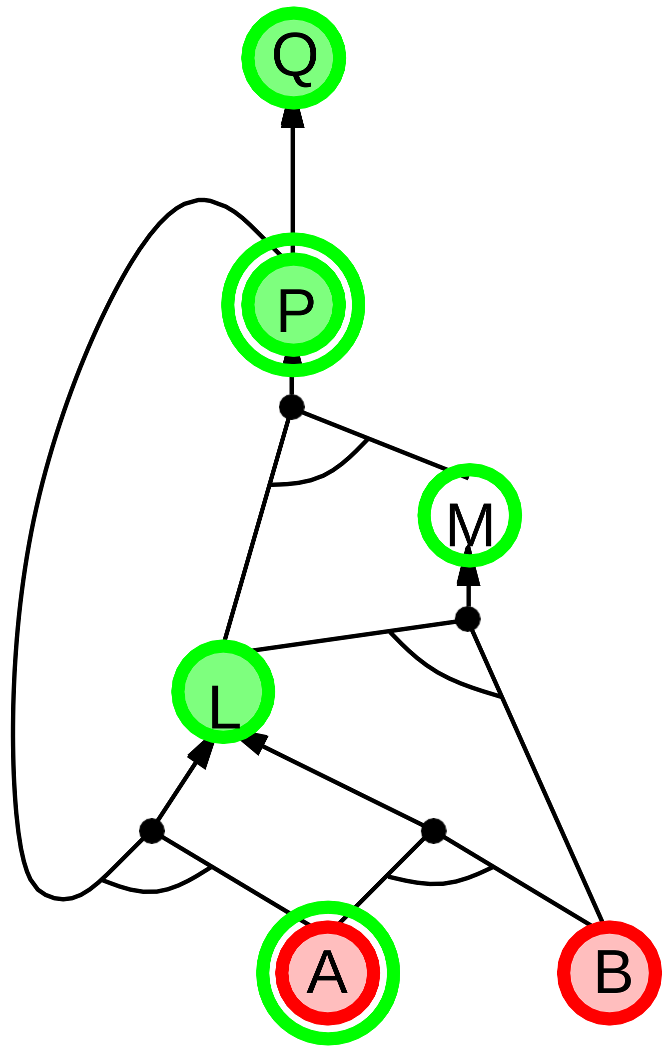
Backward chaining example



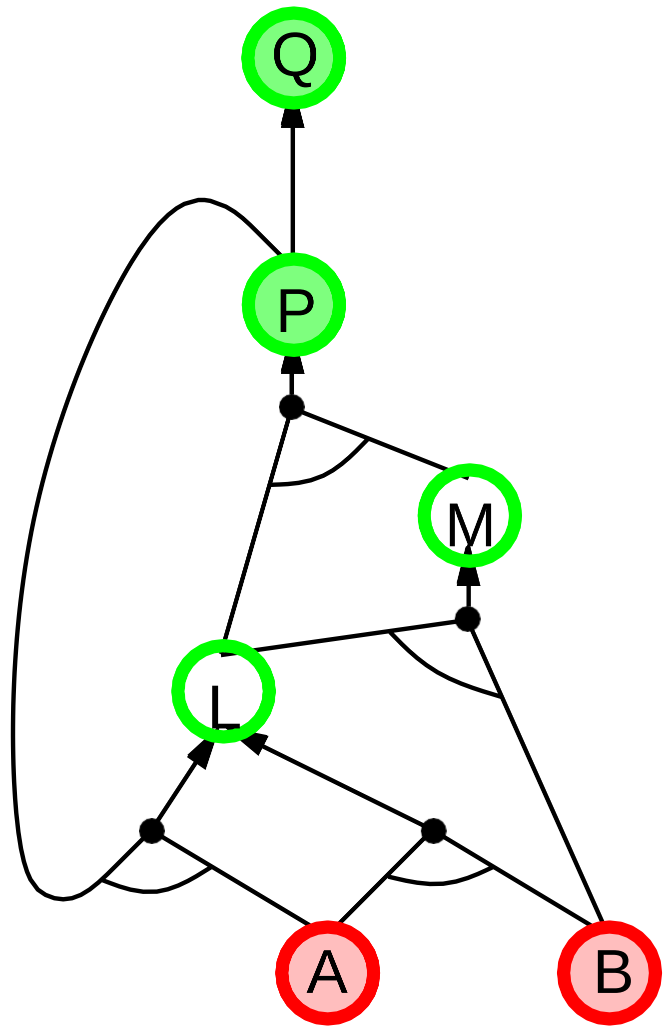
Backward chaining example



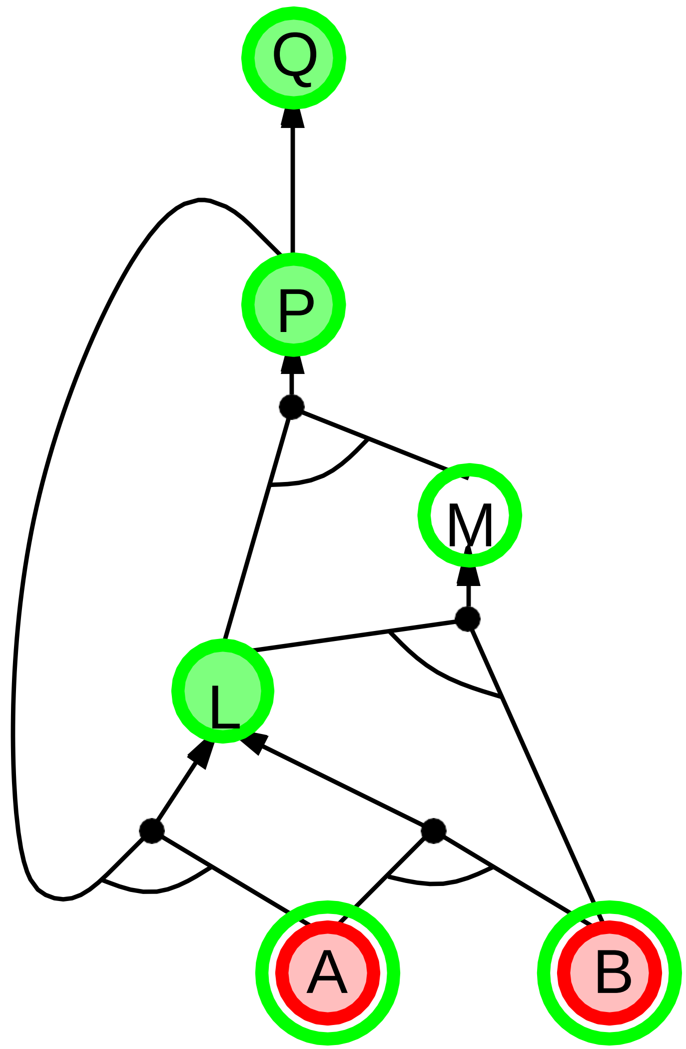
Backward chaining example



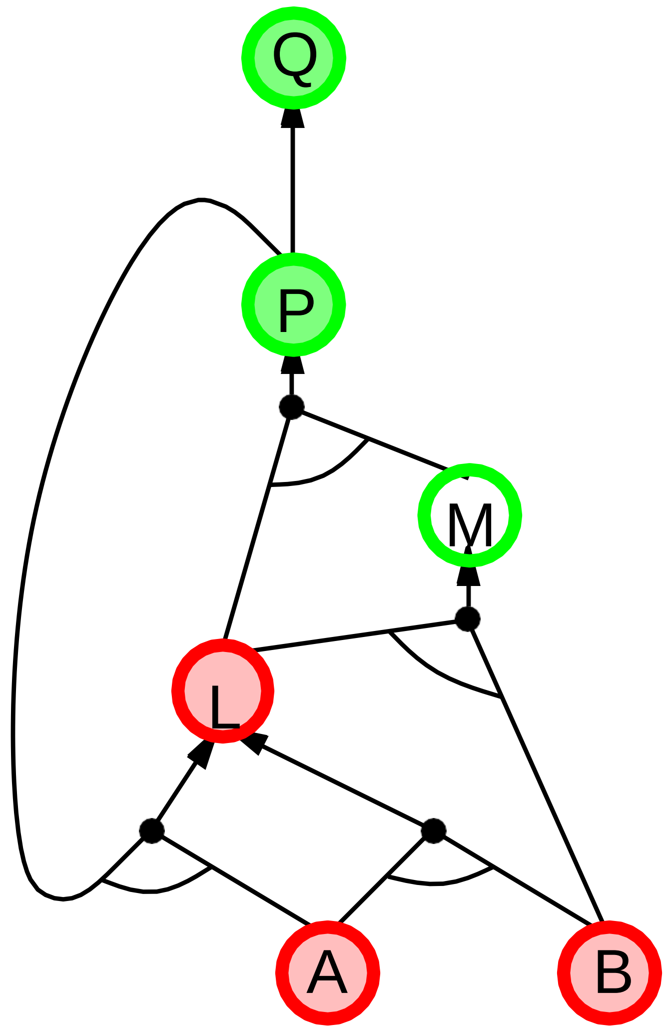
Backward chaining example



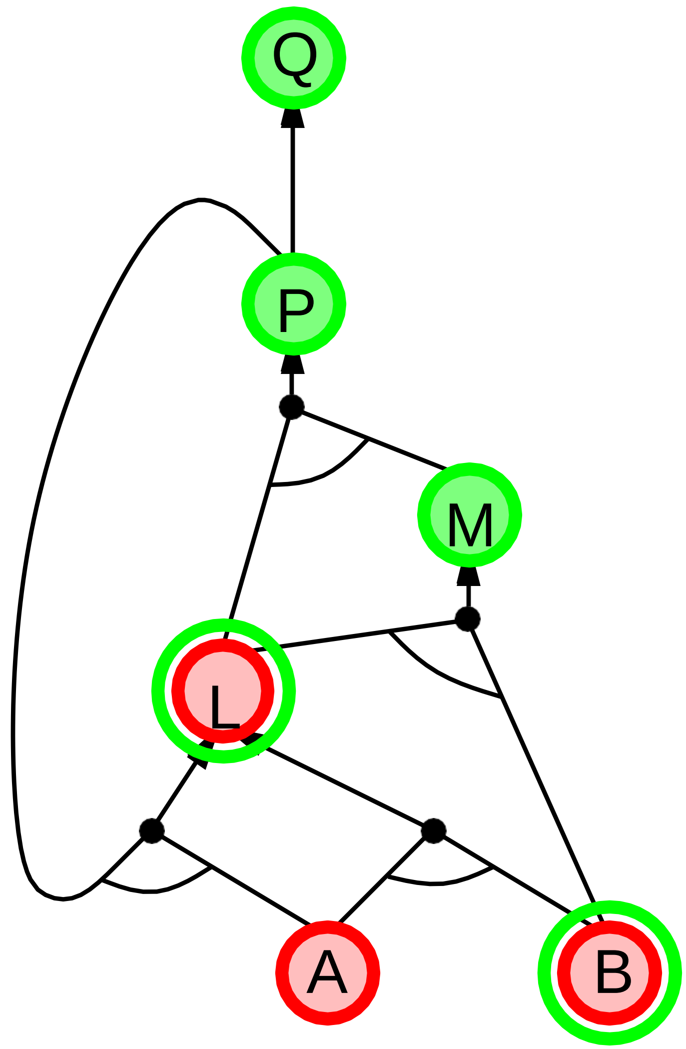
Backward chaining example



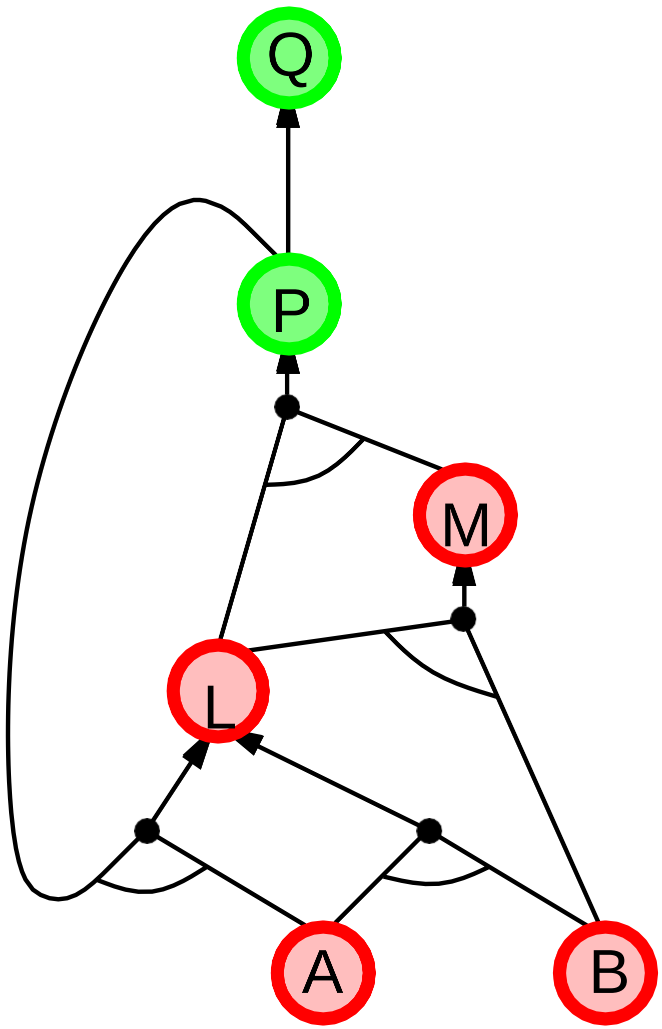
Backward chaining example



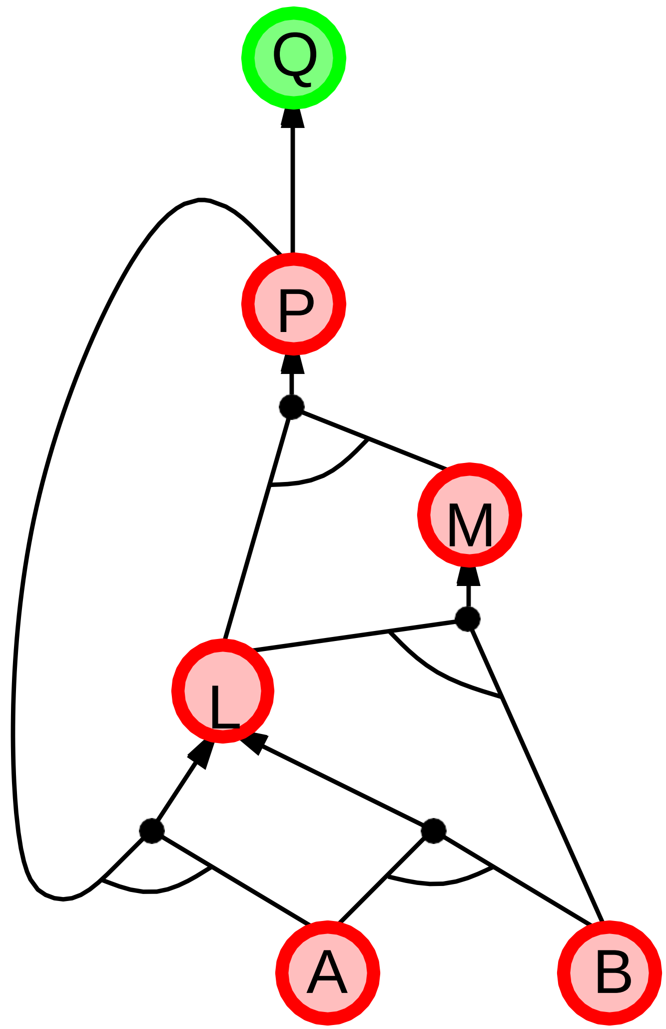
Backward chaining example



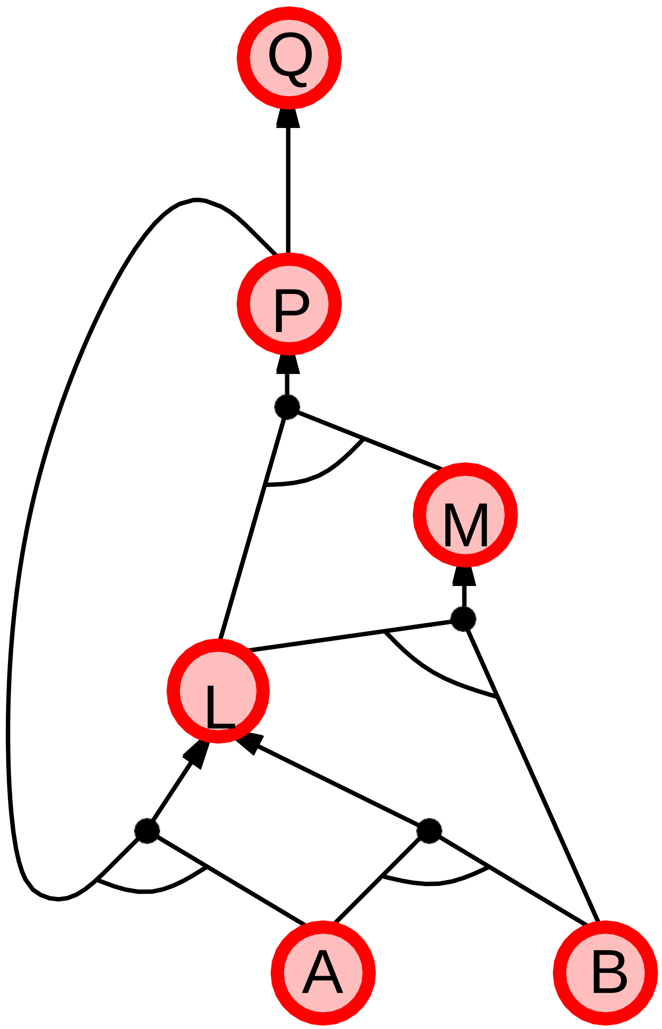
Backward chaining example



Backward chaining example



Backward chaining example



Forward vs. backward chaining

Data-driven
automatic, unconscious processing

e.g. object recognition, routine decisions

lots of work that is irrelevant to the goal

Goal-driven
appropriate for problem-solving

e.g. where are my keys?

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

SAT Problem

A sentence is satisfiable if it is true in some model

Is the sentence q satisfiable?

many combinatorial problems
in computer science can be reduced to
checking the satisfiability of a propositional
sentence…

Pearson

Davis–Putnam algorithm (DPLL)

a *recursive, depth-first enumeration of possible models.*
(similar to BACKTRACKING-SEARCH)

Input a sentence in CNF

Improvements over TT-ENTAILS:

- Early termination:
  The algorithm detects whether the sentence must be true or false, even with a partially completed model.

- Pure symbol heuristic:
  symbol that always appears with the same Pure symbol "sign" in all clauses.

- Unit Clause heuristic:
  clause with one literal OR clauses in which all literals but one are already assigned false by the model.

Pearson

A simple DPLL algorithm

```
function DPLL-SATISFIABLE?(s) returns true or false
    inputs: s, a sentence in propositional logic

    clauses←the set of clauses in the CNF representation of s
    symbols←a list of the proposition symbols in s
    return DPLL(clauses, symbols, {})

function DPLL(clauses, symbols,model) returns true or false
    if every clause in clauses is true in model then return true
    if some clause in clauses is false in model then return false
    P, value←FIND-PURE-SYMBOL(symbols, clauses,model)
    if P is non-null then return DPLL(clauses, symbols – P,model ∪{P=value})
    P, value←FIND-UNIT-CLAUSE(clauses,model)
    if P is non-null then return DPLL(clauses, symbols – P,model ∪ {P=value})
    P←FIRST(symbols); rest←REST(symbols)
    return     DPLL(clauses, rest,model ∪ {P=true}) or
               DPLL(clauses, rest,model ∪ {P=false}))
```

Some DPLL tricks

- Component analysis:
  the set of clauses may become separated into disjoint subsets
- Variable and value ordering:
  choosing the variable that appears most frequently over all remaining clauses.
- Intelligent backtracking:
  that backs up all the way to the relevant point of conflict.
- Random restarts:
  Sometimes a run appears not to be making progress. In this case, we can start over
- Clever indexing:
  fast indexing of such things as “the set of clauses in which variable Xi appears as a positive literal.”

Effective Propositional Model Checking

Local search algorithms such as hill-climbing & simulated annealing.

WALKSAT: On every iteration, the algorithm picks an unsatisfied clause and picks a symbol in the clause to flip.

Summary and reading exercise

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
- soundess: derivations produce only entailed sentences
- completeness: derivations can produce all entailed sentences

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

DPLL algorithm for SAT solving

Read from 7.1 to 7.6 of Artifitial Intelligence - A modern approach

Pearson