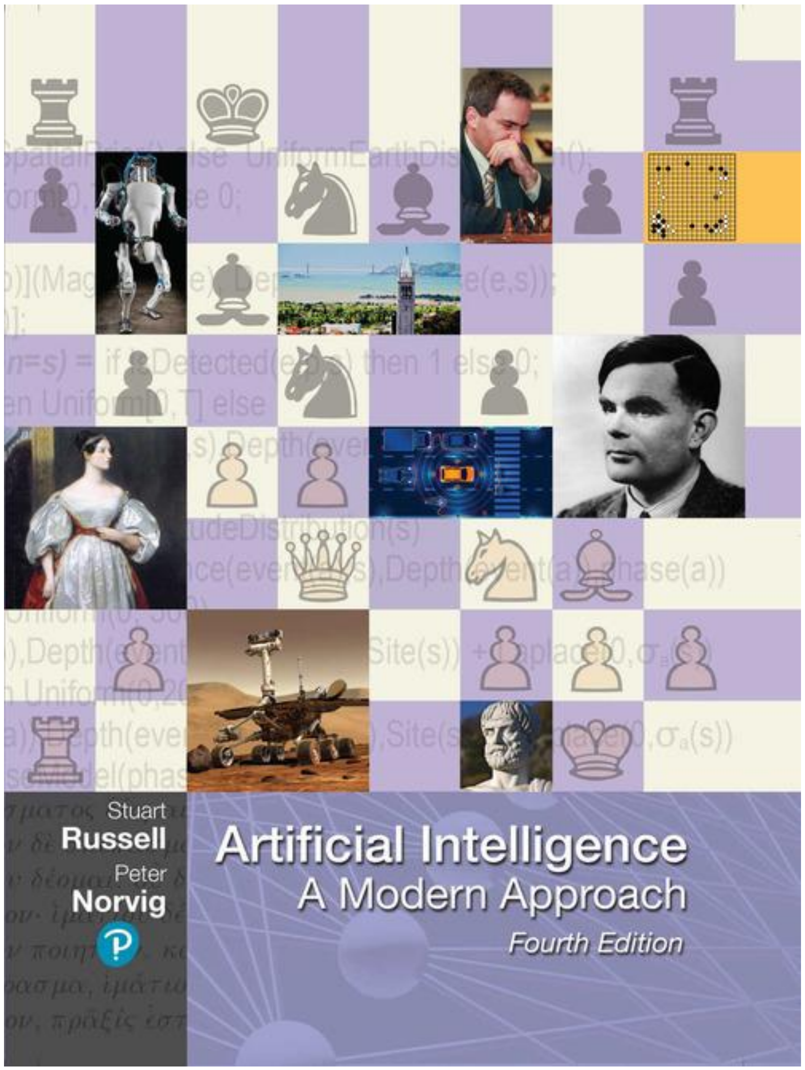
Artificial Intelligence Fundamentals

2024-2025



“The answer is logic. Or, to put it another way, the ability to reason analytically. Applied properly, it can overcome any lack of wisdom, which one only gains through age and experience.”

- Christopher Paolini

AIMA Chapter 7

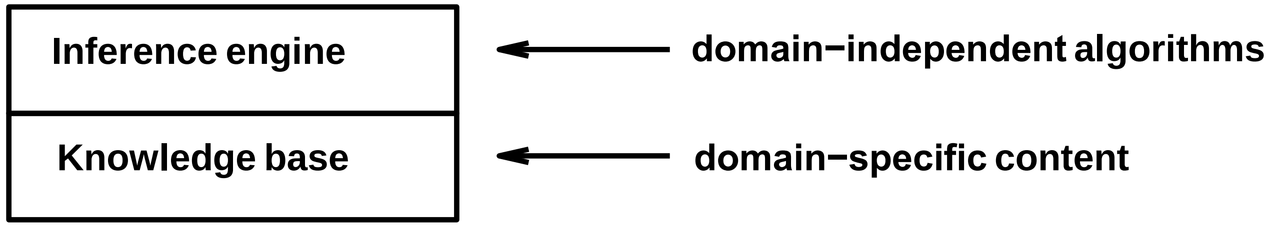
Logical agents

Outline

"In which we design agents that can form representations of a complex world, use a process of inference to derive new representations about the world, and use these new representations to deduce what to do"

- Knowledge-based agents
- Wumpus world
- Logic in general—models and entailment
- Propositional (Boolean) logic
- Equivalence, validity, satisfiability
- Inference rules and theorem proving
    - resolution
    - forward chaining
    - backward chaining

Knowledge bases



Knowledge base = set of sentences in a formal language

Declarative approach to building an agent (or other system):

Tell it what it needs to know

Then it can Ask itself what to do—answers should follow from the

Knowledge Base (KB)

Agents can be viewed at the knowledge level

i.e., what they know, regardless of how implemented

Or at the implementation level

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
```

The agent must be able to:

- Represent states, actions, etc.
- Incorporate new percepts
- Update internal representations of the world
- Deduce hidden properties of the world
- Deduce appropriate actions

Wumpus World PEAS description

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

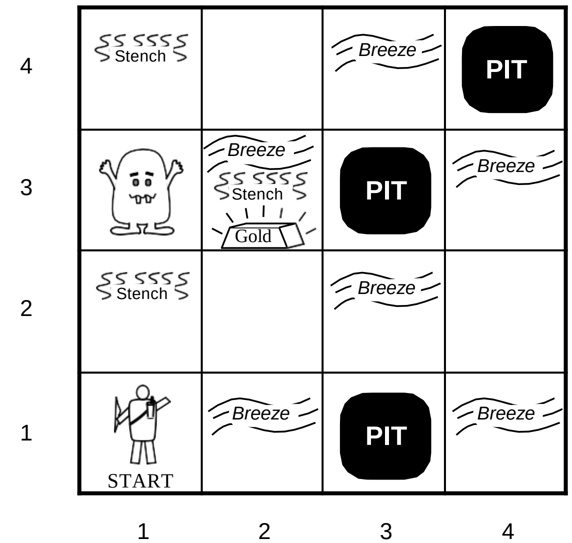
Glitter iff gold is in the same square

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



Actuators: Left turn, Right turn, Forward, Grab, Release, Shoot

Sensors: Breeze (B), Glitter (G), Smell (S)

[Play here → Wumpus World Simulator (thiagodnf.github.io)]

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

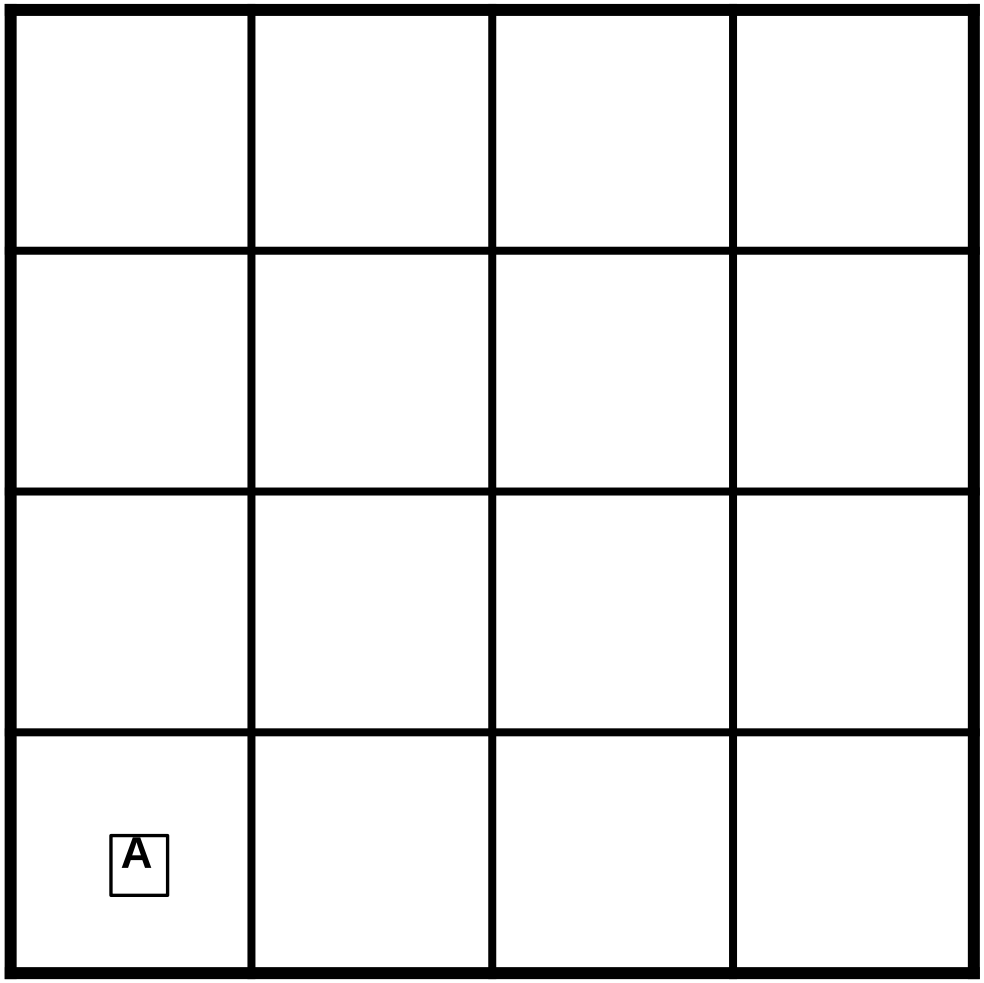
Episodic? No—sequential at the level of actions

Static? Yes—Wumpus and Pits do not move

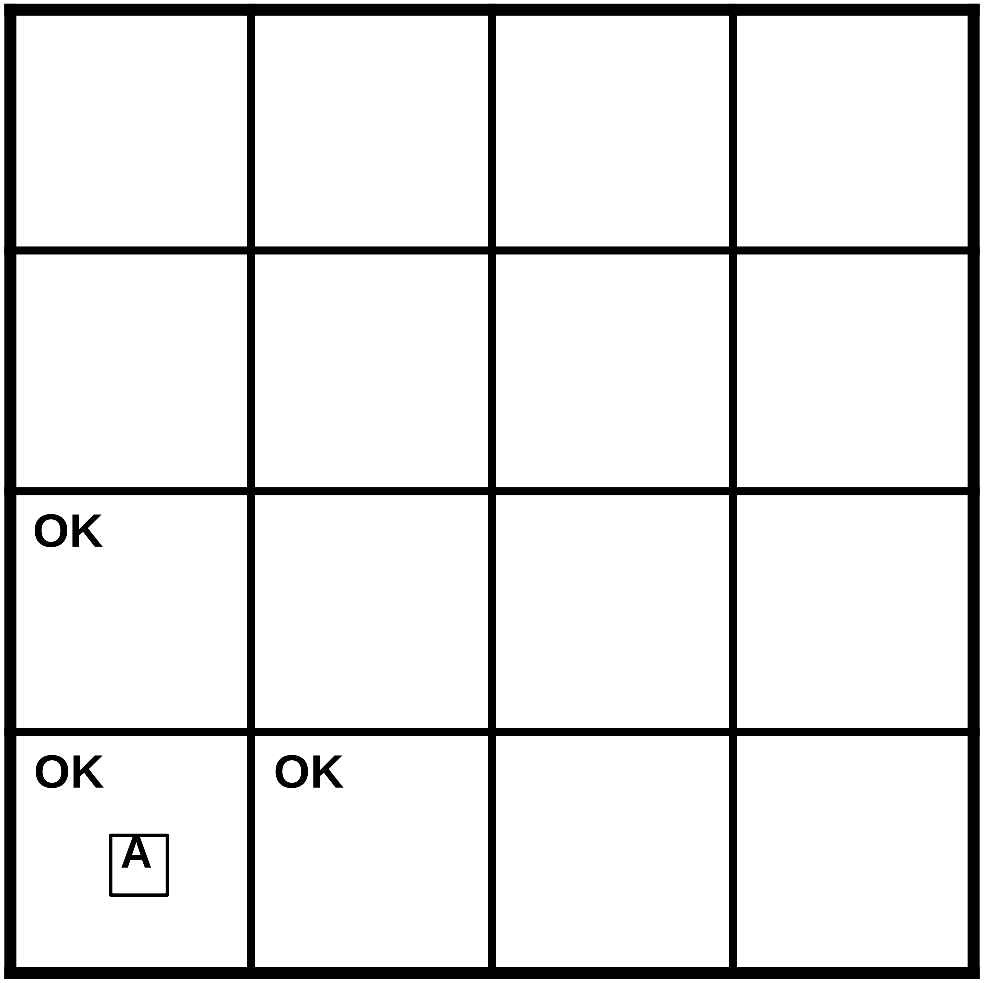
Discrete? Yes

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

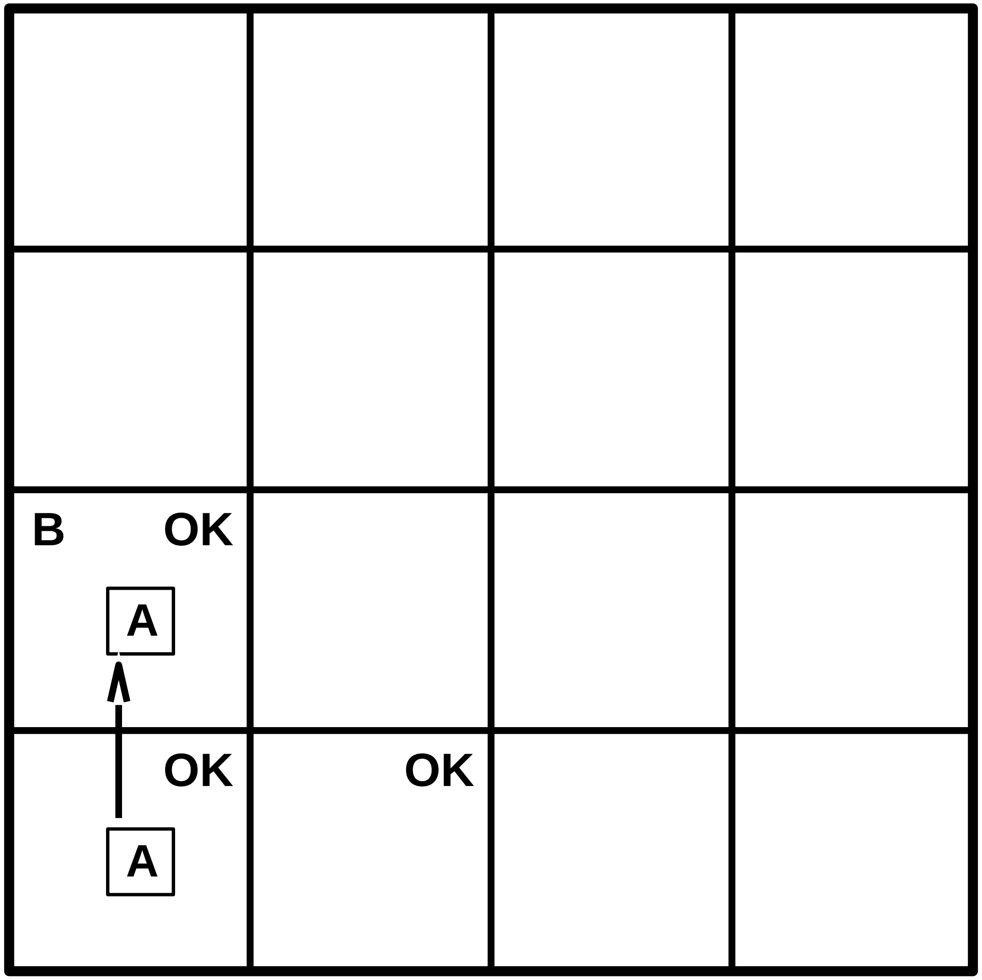
Exploring a wumpus world



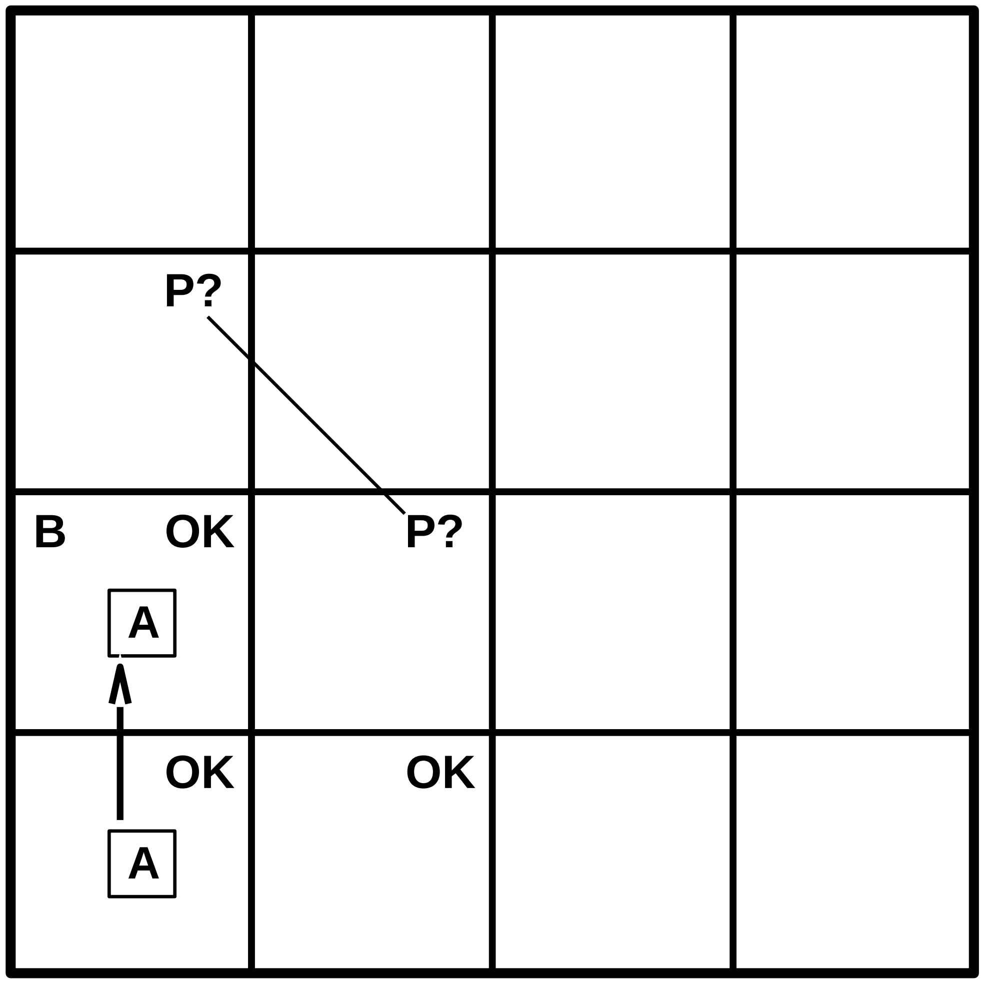
Exploring a wumpus world



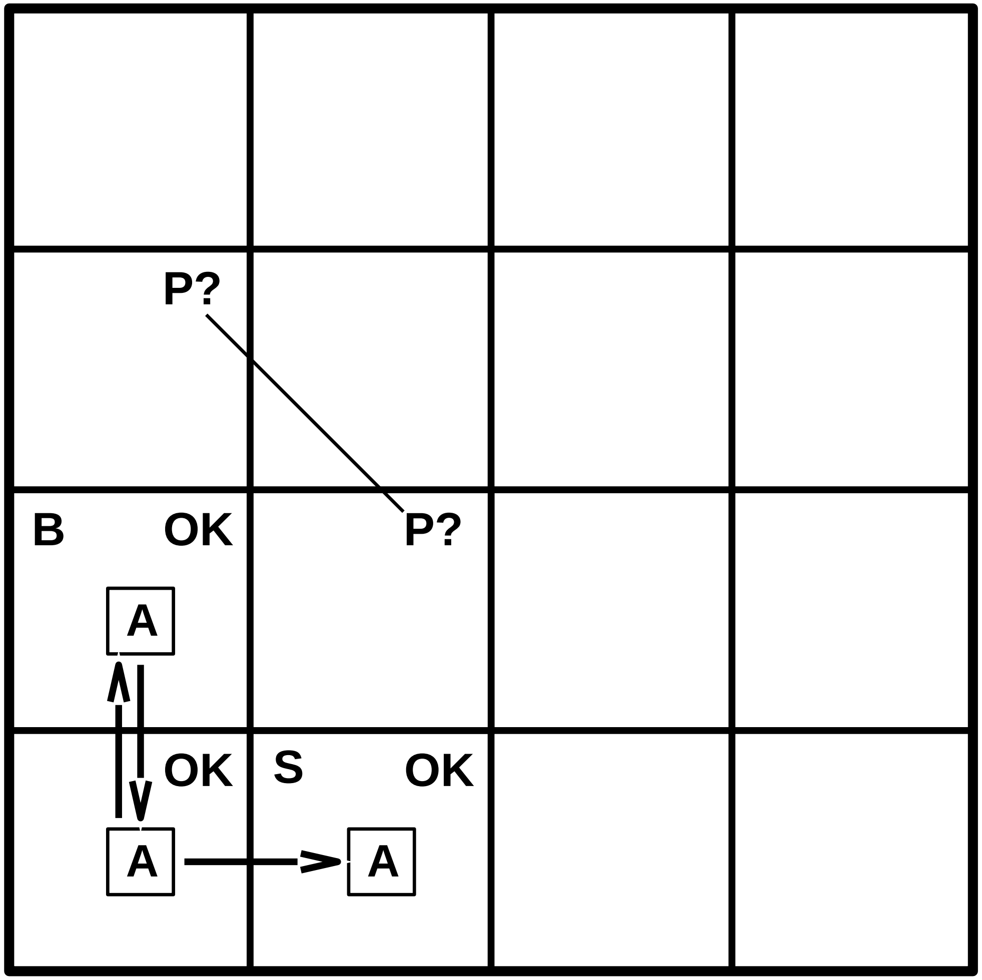
Exploring a wumpus world



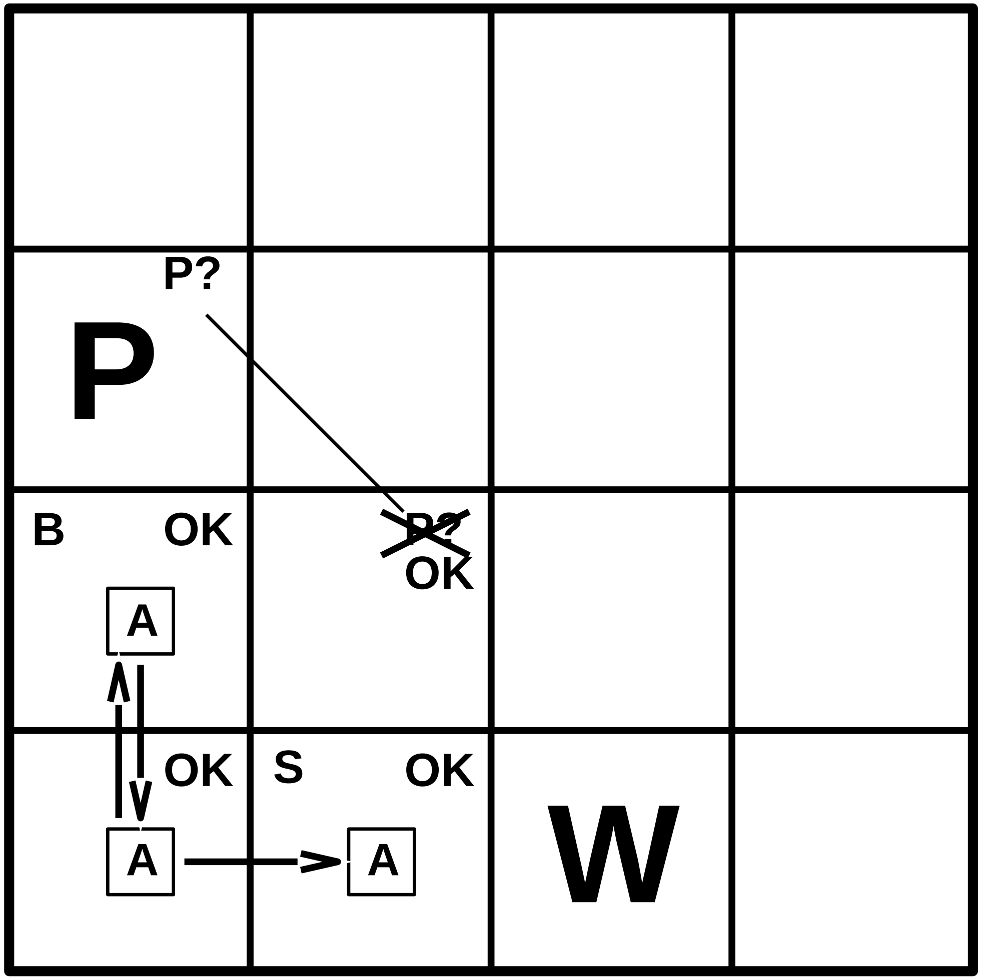
Exploring a wumpus world



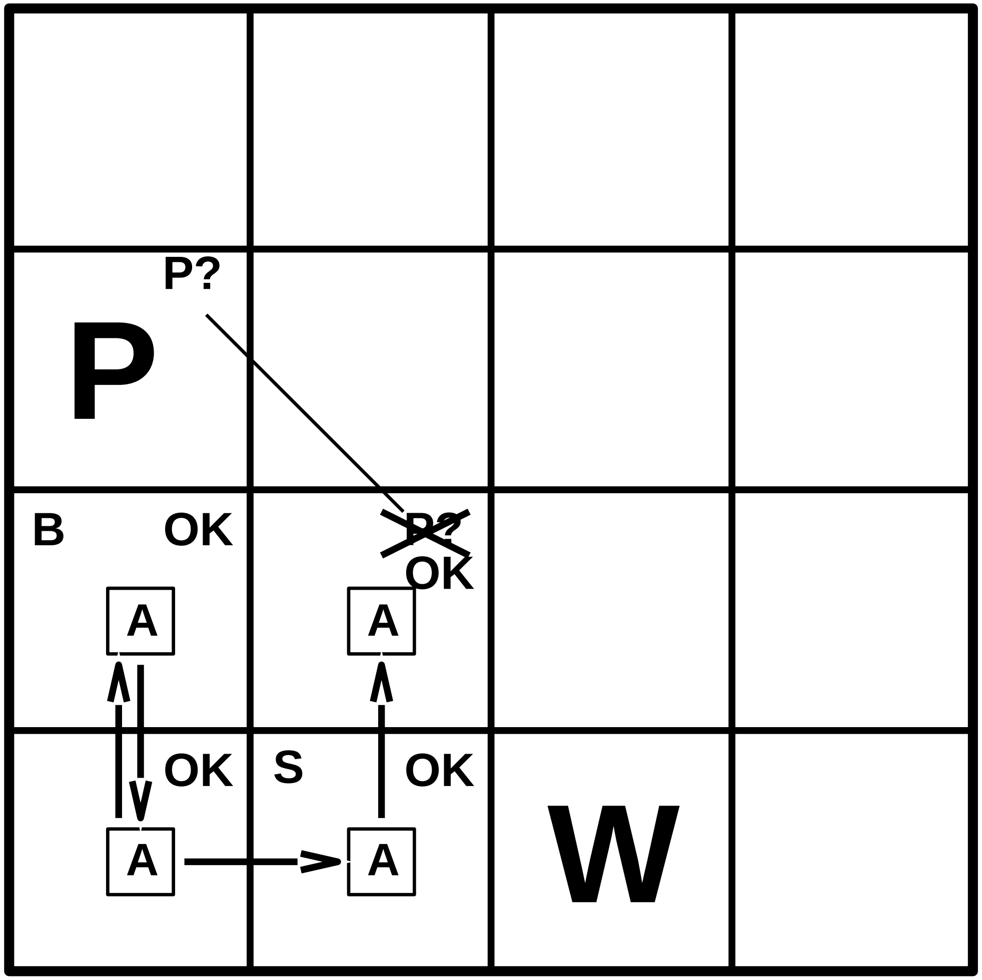
Exploring a wumpus world



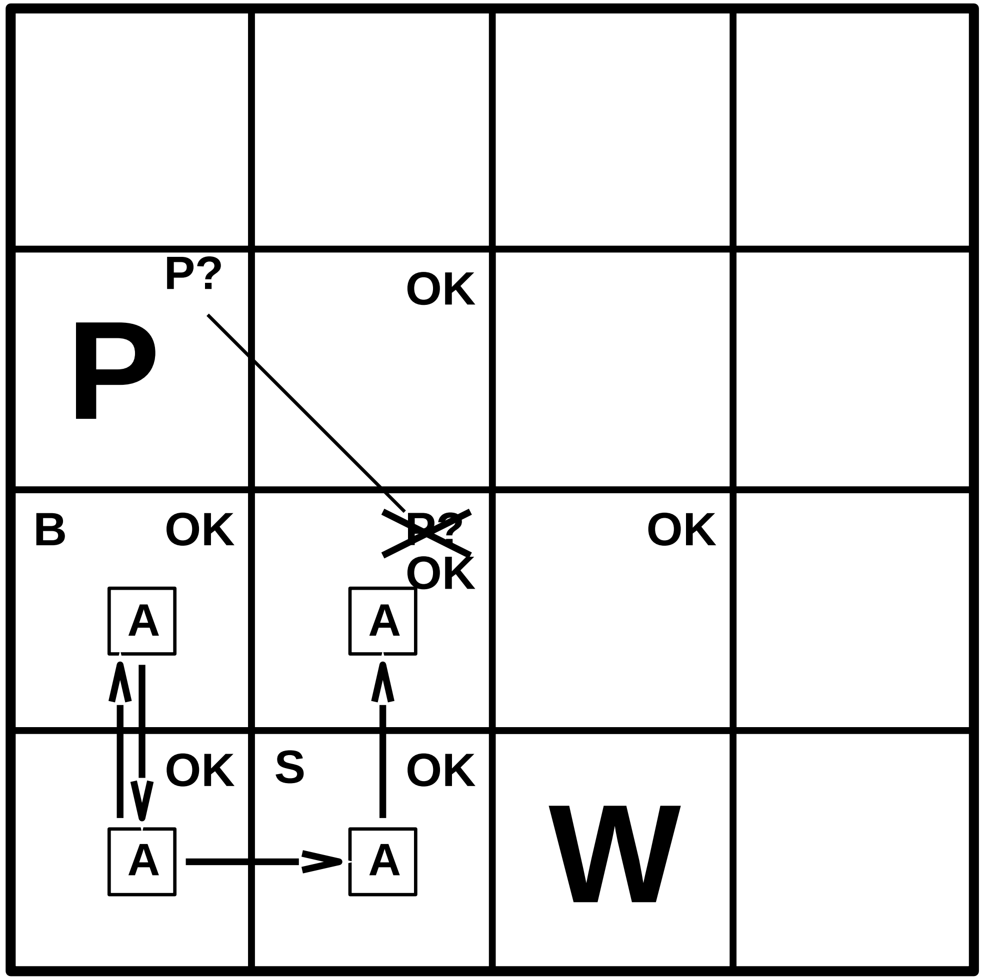
Exploring a wumpus world



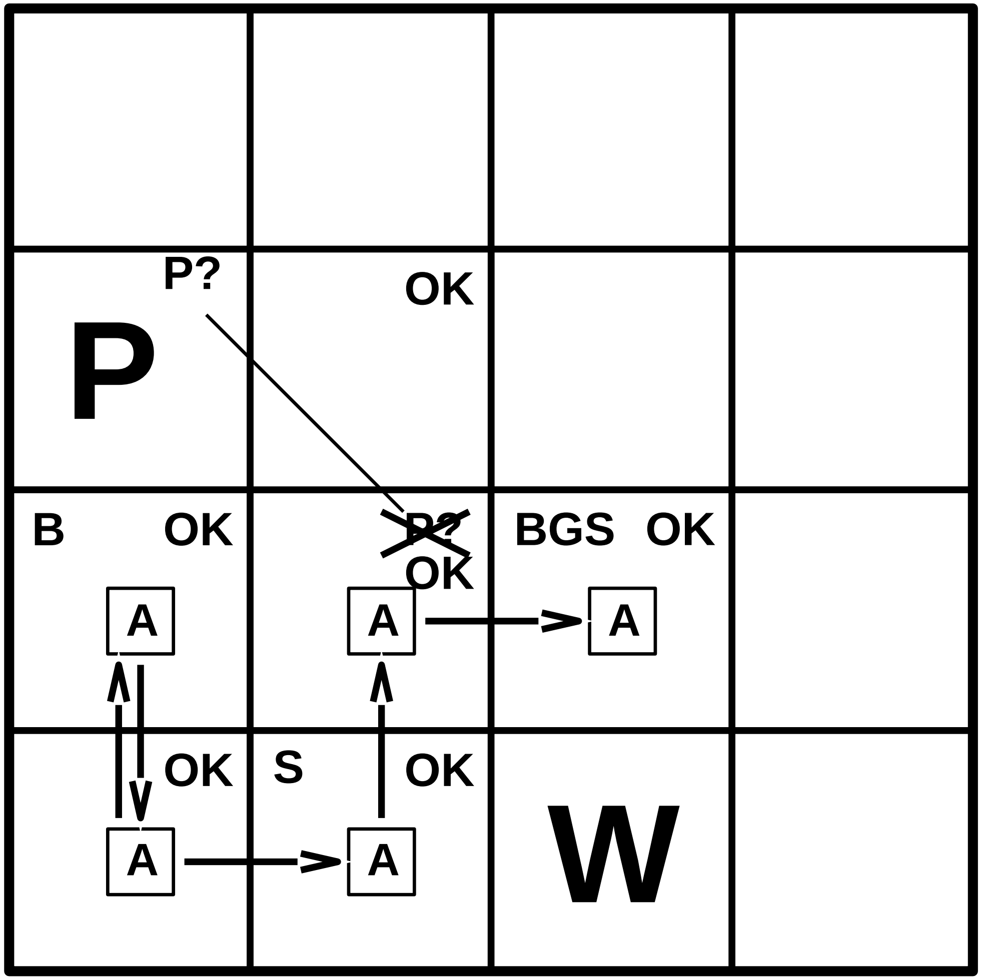
Exploring a wumpus world



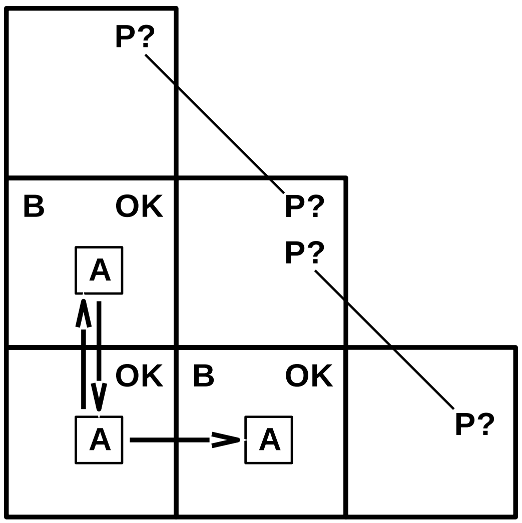
Exploring a wumpus world



Exploring a wumpus world

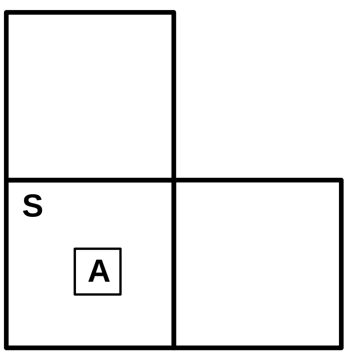


Other tight spots



Breeze in (1,2) and (2,1)
⇒ no safe actions

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



Smell in (1,1)
⇒ cannot move

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

Logic in general

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

Syntax defines the sentences in the language

Semantics define the "*meaning*" of sentences;
i.e., 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

Entailment means that one thing follows from another:

$$KB \models \alpha$$

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

E.g., 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) that is based on semantics

Note: brains process syntax (of some sort)

Models

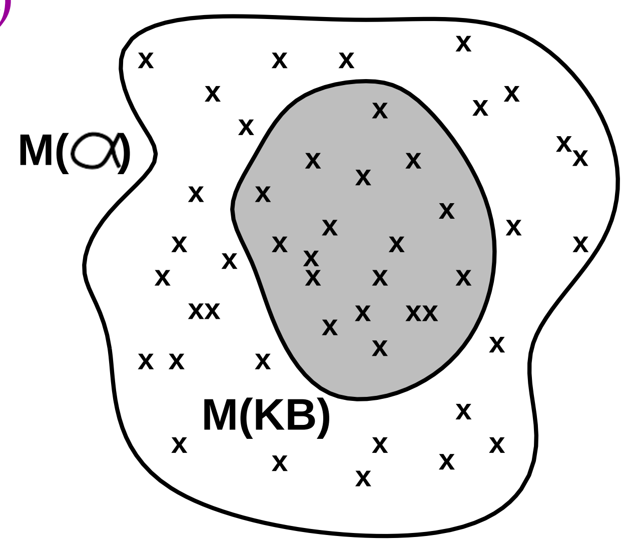
Logicians typically think in terms of **models**, which are *formally structured worlds* with respect to which truth can be evaluated

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

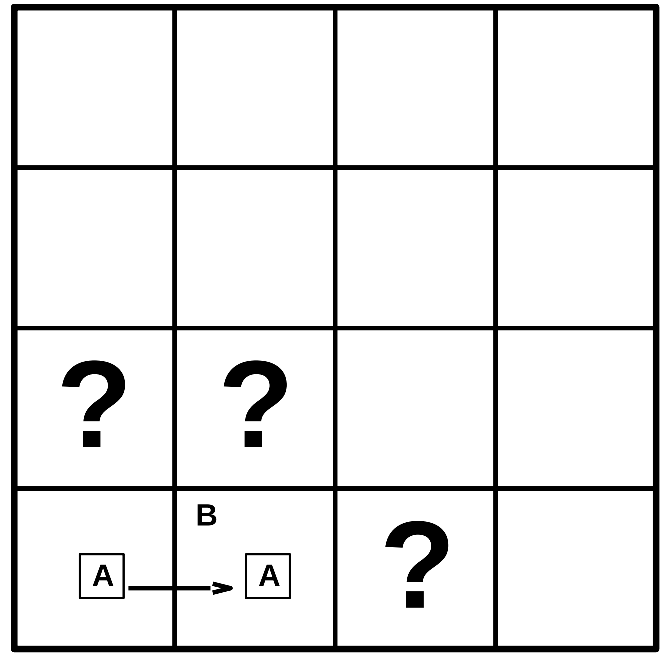
$M(\alpha)$ is the set of all models of α

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

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



Entailment in the wumpus world

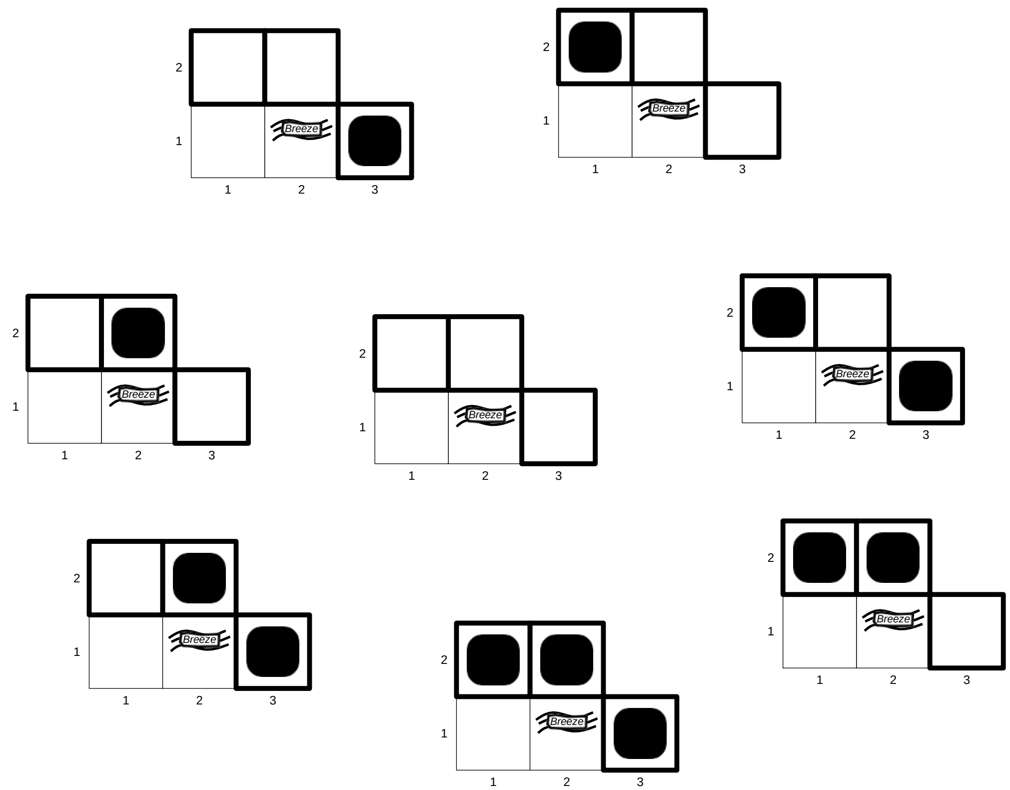


Situation after detecting nothing in [1,1], moving right, breeze in [2,1]

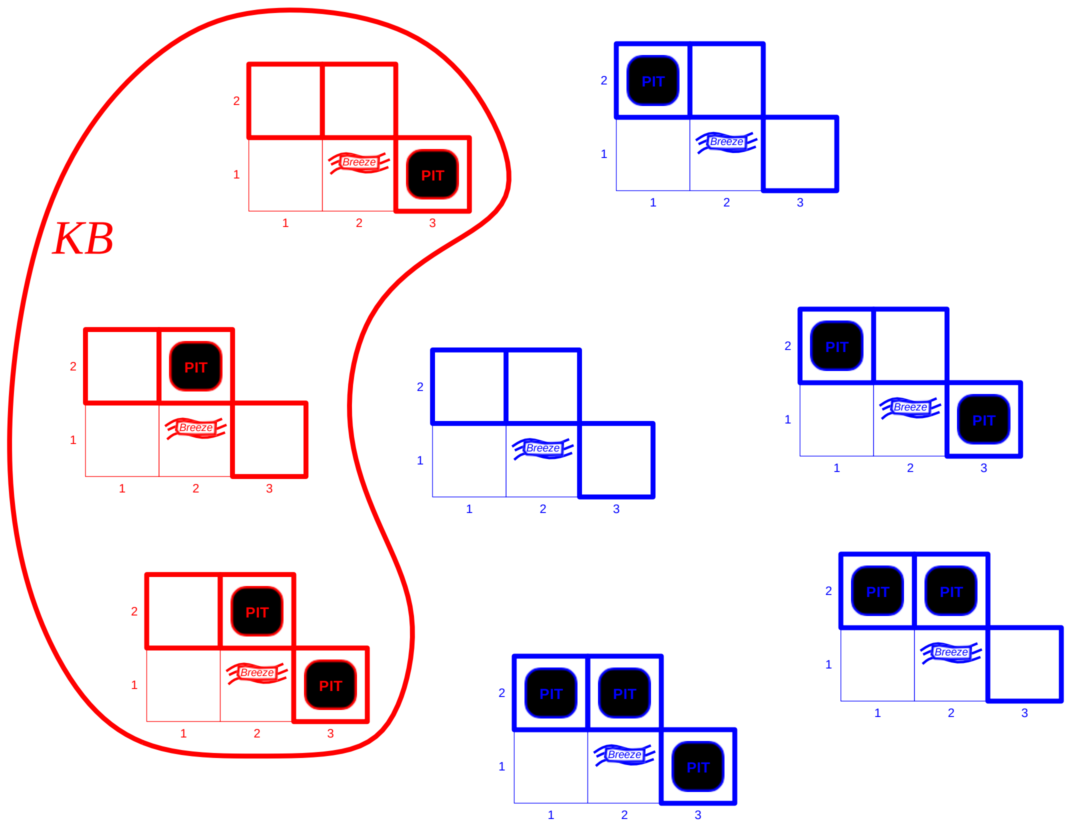
Consider possible models for "?" assuming only pits

3 Boolean choices ⇒ 8 possible models

Wumpus models

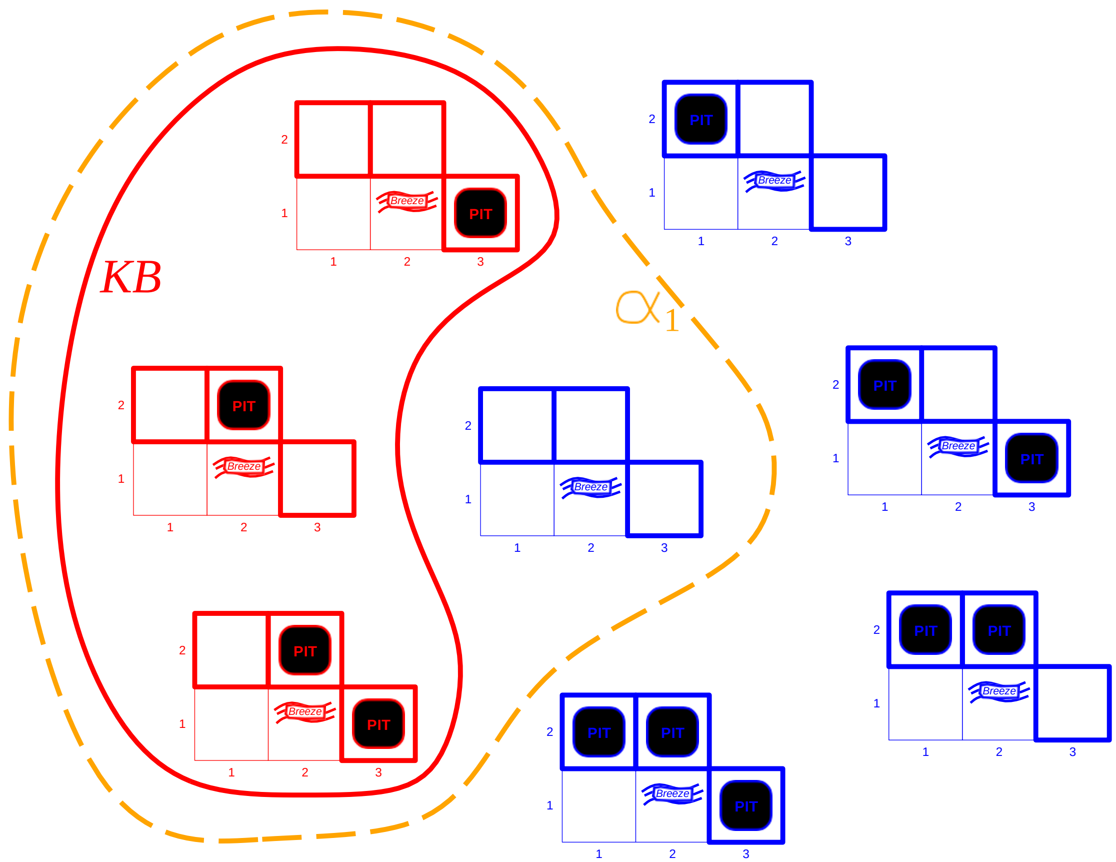


Wumpus models



KB = wumpus-world rules + observations

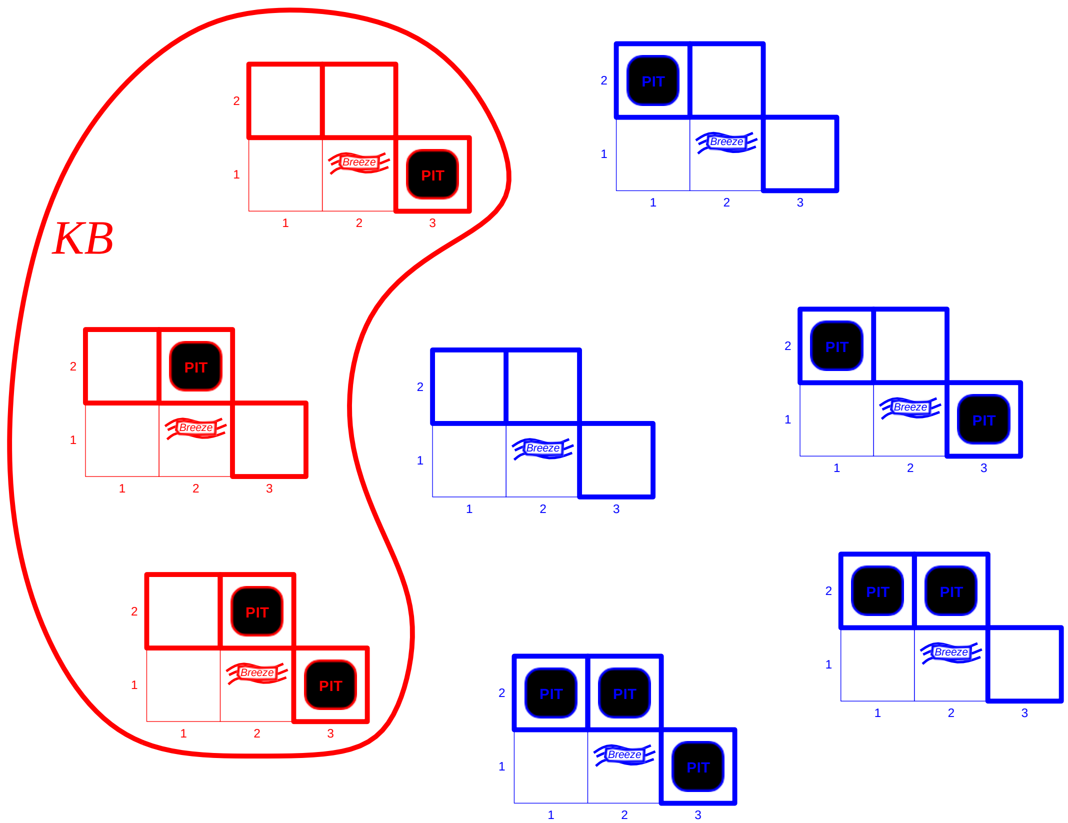
Wumpus models



KB = wumpus-world rules + observations

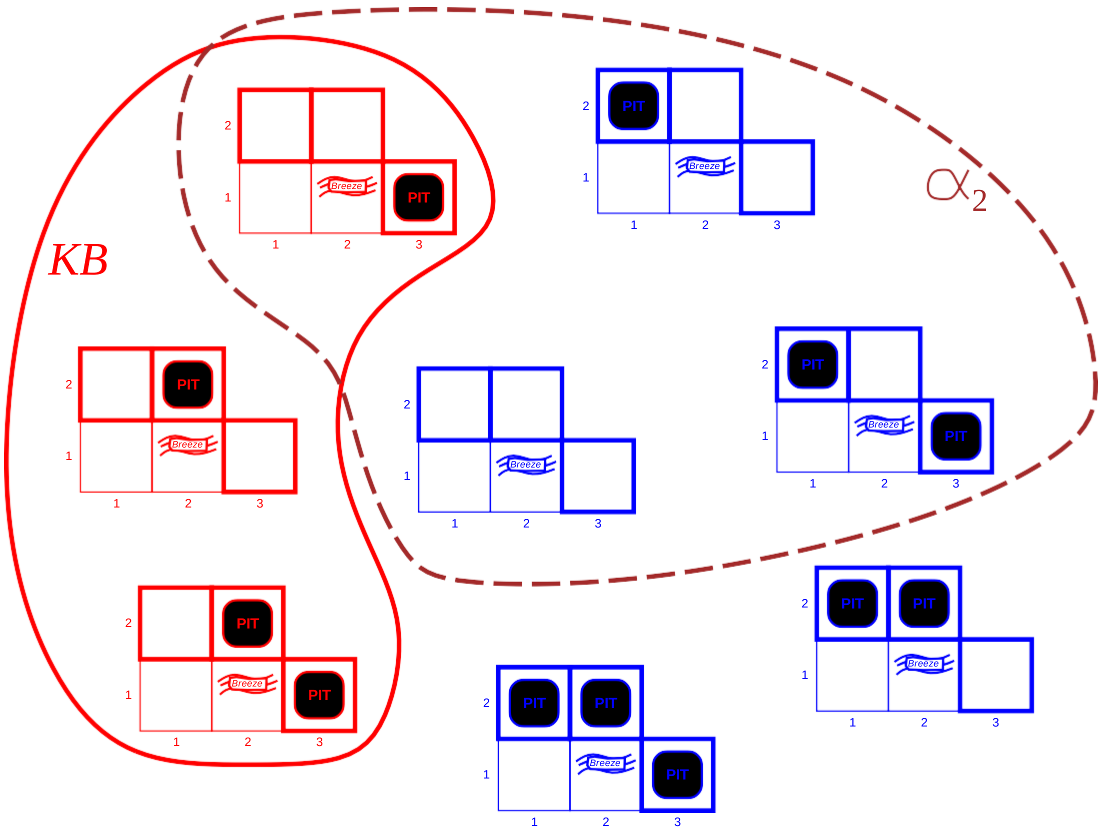
α_1 = "[1,2] is safe", $KB \models \alpha_1$, proved by model checking

Wumpus models



KB = wumpus-world rules + observations

Wumpus models



KB = wumpus-world rules + observations

α_2 = "[2,2] is safe", $KB \models \alpha_2$

in some models in which KB is true, α2 is false.
Hence, KB does not entail α2.

Inference

$KB \vdash_i \alpha$ = sentence a can be derived from KB by procedure i

Consequences of KB are a haystack; a is a needle.
Entailment = needle in haystack; *inference* = finding it

Soundness: i is sound if
whenever $KB \vdash_i \alpha$ it is also true that $KB \models a$

Completeness: i is complete if
whenever $KB \models a$, it is also true that $KB \vdash_i \alpha$

Preview: we will define a logic (first-order logic) 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.

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

(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
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.,

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

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*

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 $[i, j]$.

$\neg P_{1,1}$

$\neg B_{1,1}$ — Observations

$B_{2,1}$

"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"

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

«Rules» valid in every wumpus world

"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*
⋮	⋮	⋮	⋮	⋮	⋮	⋮	⋮	⋮	⋮	⋮	⋮	⋮
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*

Figure 7.9 A truth table constructed for the knowledge base given in the text. KB is true if R_1 through R_5 are true, which occurs in just 3 of the 128 rows (the ones underlined in the right-hand column). In all 3 rows, $P_{1,2}$ 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].

Enumerate rows (different assignments to symbols),

7 symbols = 2^7 = 128 possible models

if KB is true in row, check that α is too

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))
```

when KB is false, always return true

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

Logical equivalence

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

$(\alpha \land \beta)$	$\equiv$	$(\beta \land \alpha)$ commutativity of $\land$
$(\alpha \lor \beta)$	$\equiv$	$(\beta \lor \alpha)$ commutativity of $\lor$
$((\alpha \land \beta) \land \gamma)$	$\equiv$	$(\alpha \land (\beta \land \gamma))$ associativity of $\land$
$((\alpha \lor \beta) \lor \gamma)$	$\equiv$	$(\alpha \lor (\beta \lor \gamma))$ associativity of $\lor$
$\neg(\neg \alpha)$	$\equiv$	α double-negation elimination
$(\alpha \Rightarrow \beta)$	$\equiv$	$(\neg \beta \Rightarrow \neg \alpha)$ contraposition
$(\alpha \Rightarrow \beta)$	$\equiv$	$(\neg \alpha \lor \beta)$ implication elimination
$(\alpha \Leftrightarrow \beta)$	$\equiv$	$((\alpha \Rightarrow \beta) \land (\beta \Rightarrow \alpha))$ biconditional elimination
$\neg(\alpha \land \beta)$	$\equiv$	$(\neg \alpha \lor \neg \beta)$ De Morgan
$\neg(\alpha \lor \beta)$	$\equiv$	$(\neg \alpha \land \neg \beta)$ De Morgan
$(\alpha \land (\beta \lor \gamma))$	$\equiv$	$((\alpha \land \beta) \lor (\alpha \land \gamma))$ distributivity of $\land$ over $\lor$
$(\alpha \lor (\beta \land \gamma))$	$\equiv$	$((\alpha \lor \beta) \land (\alpha \lor \gamma))$ distributivity of $\lor$ over $\land$

Validity and satisfiability

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:

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

A sentence is satisfiable if it is true in some model

e.g., $R1 \wedge R2 \wedge R3 \wedge R4 \wedge R5$ *(in the previous KB)*

A sentence is unsatisfiable if it is true in no model

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 unsatisfiable i.e., prove α by *reductio ad absurdum*

Proof methods

Proof (“*a chain of conclusions that leads to the desired goal*”) 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 alg.
- Typically require translation of sentences into a normal form

Model checking

- 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

Resolution

Conjunctive Normal Form (CNF—universal)

conjunction of $\underbrace{\text{disjunctions of literals}}_{\text{Clauses}}$

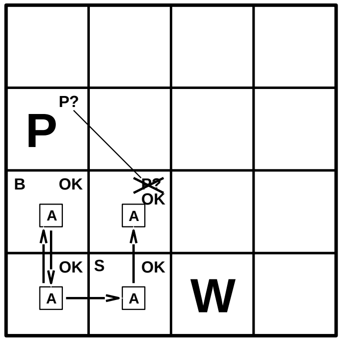
E.g., $(A \lor \neg B) \land (B \lor \neg C \lor \neg D)$

Resolution inference rule (for CNF):

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

where l_i and m_j are complementary literals. E.g.,

$$\frac{P_{1,3} \lor P_{2,2}, \qquad \neg P_{2,2}}{P_{1,3}}$$



Resolution is **sound** and **complete** for propositional logic

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

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
```

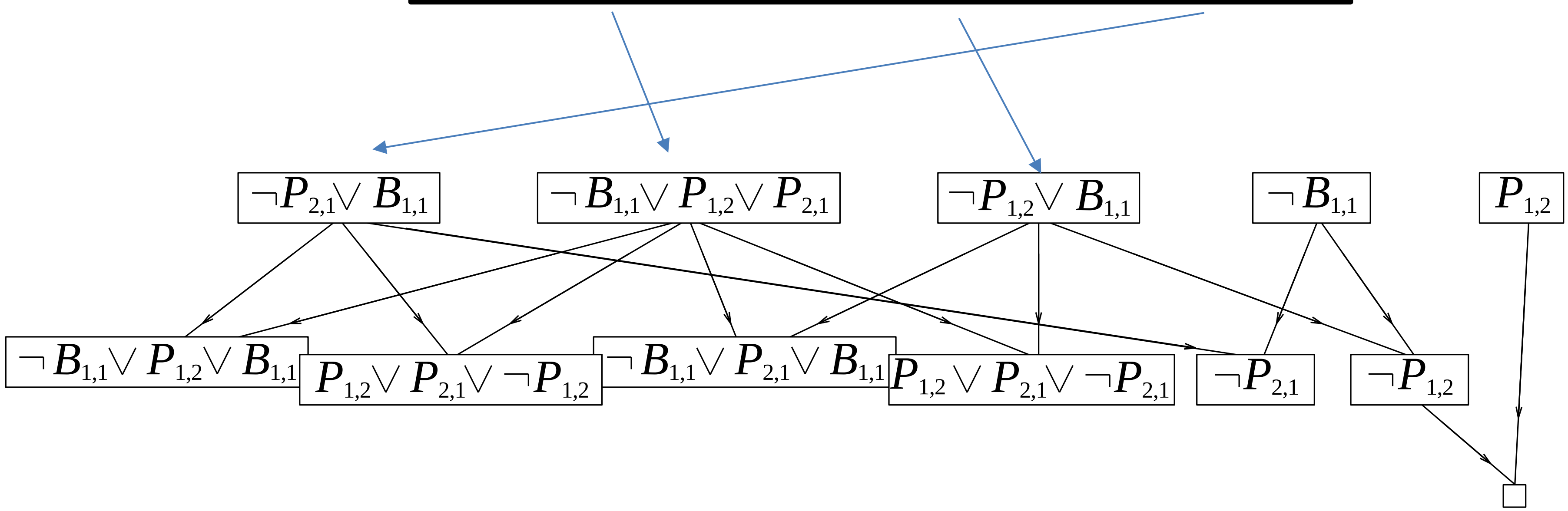
Resolution example

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

$a = \neg P_{1,2}$

CNF

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



Forward and backward chaining

Horn Form (restricted)

KB = conjunction of Horn clauses

Horn clause =

1. proposition symbol; or
2. (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

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

Can be used with forward chaining or backward chaining.

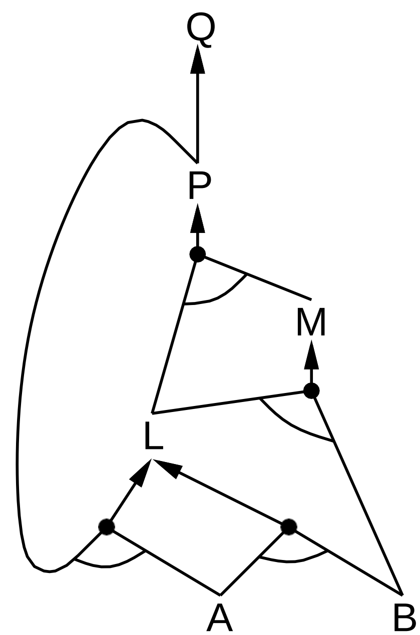
These algorithms are very natural and run in linear time

Pearson

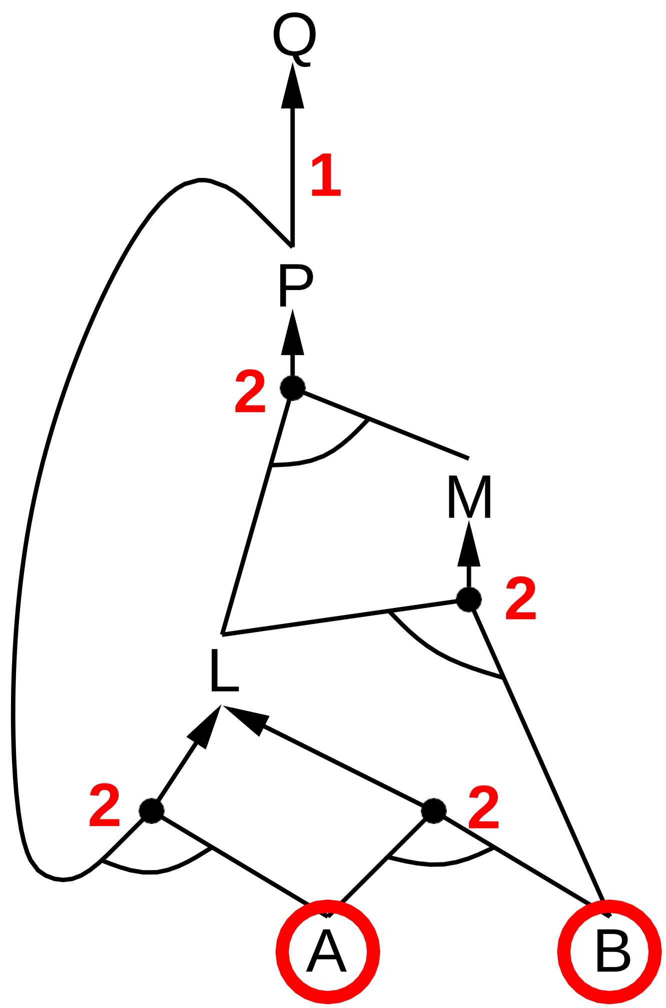
Forward chaining

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

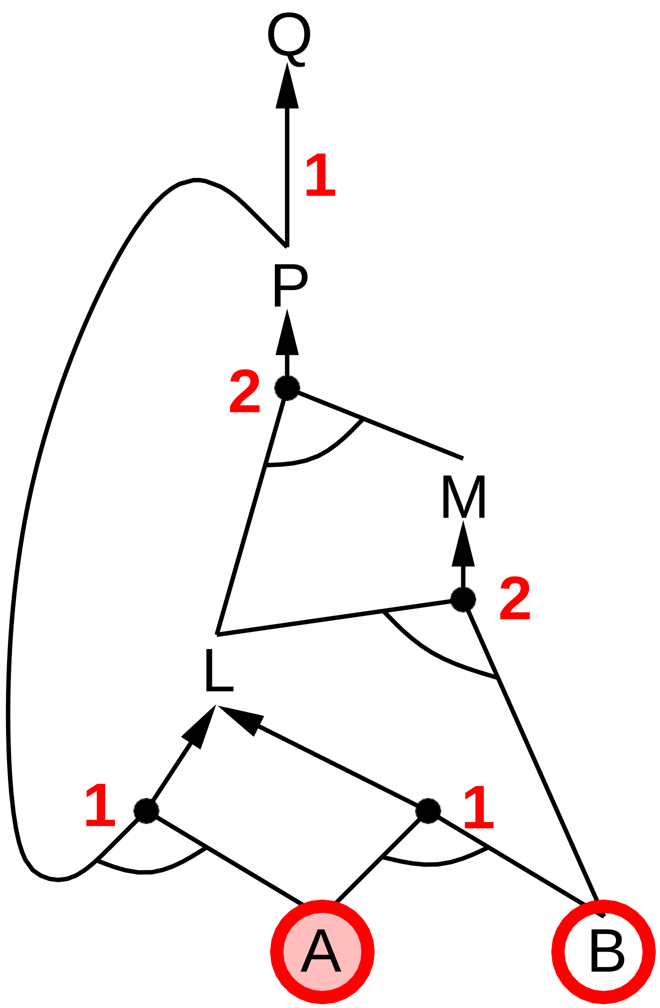
$$P \Rightarrow Q$$
$$L \wedge M \Rightarrow P$$
$$B \wedge L \Rightarrow M$$
$$A \wedge P \Rightarrow L$$
$$A \wedge B \Rightarrow L$$
$$A$$
$$B$$



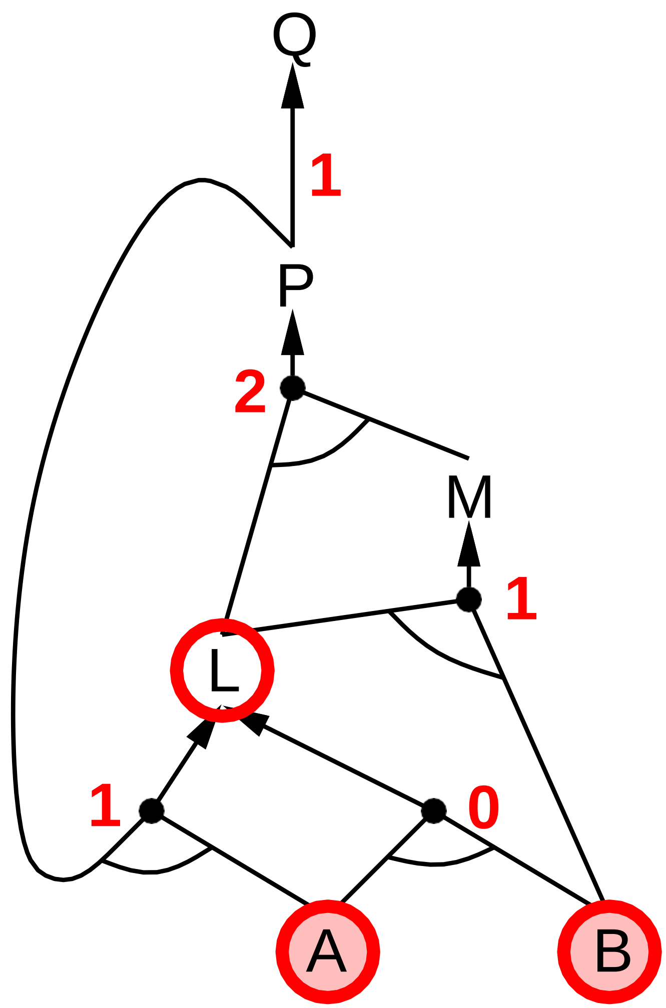
Forward chaining example



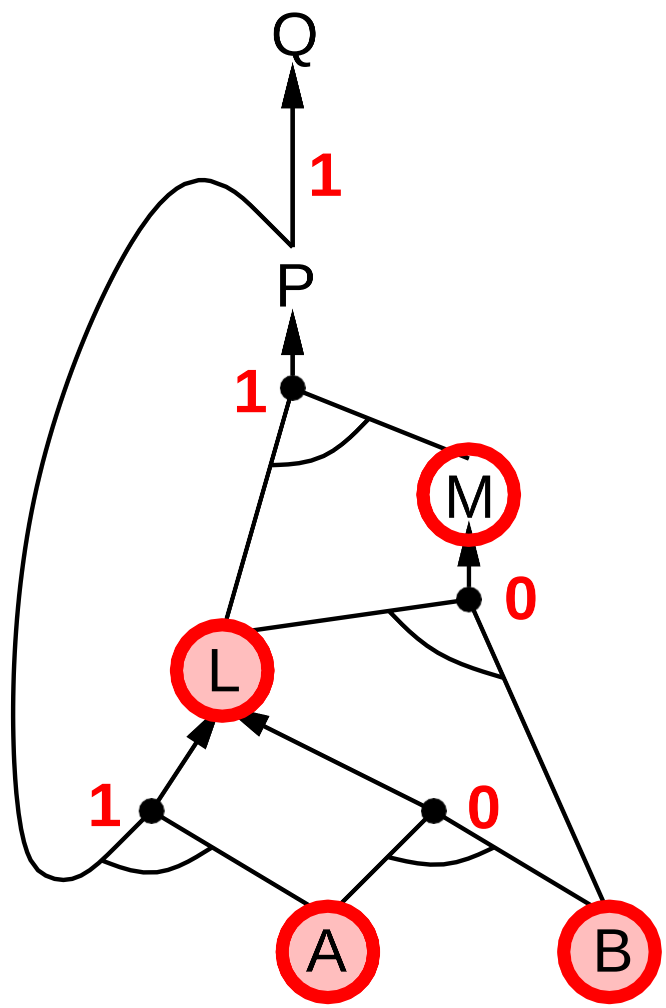
Forward chaining example



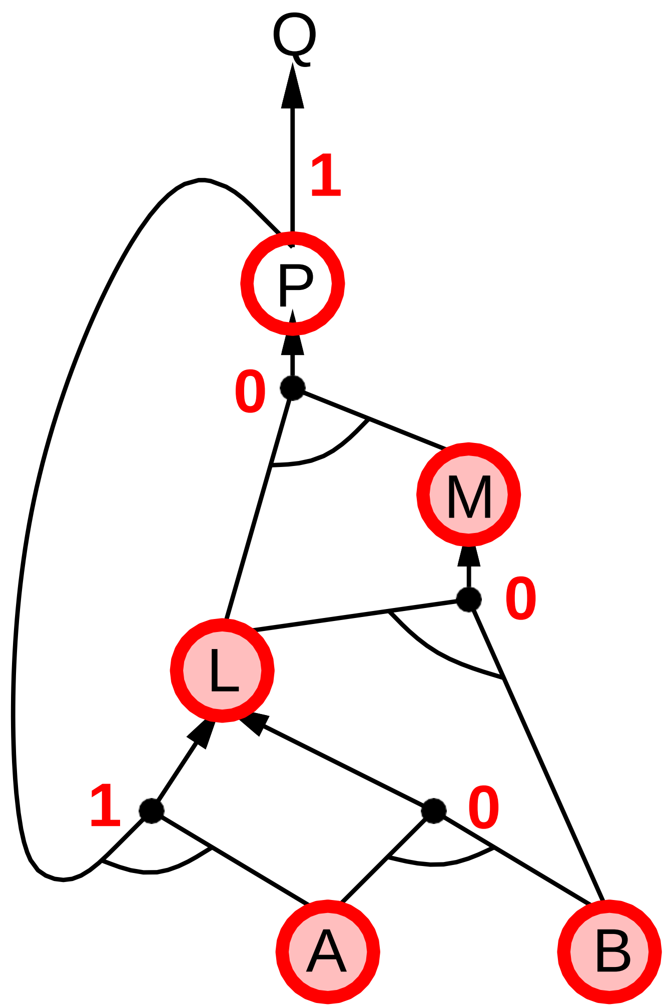
Forward chaining example



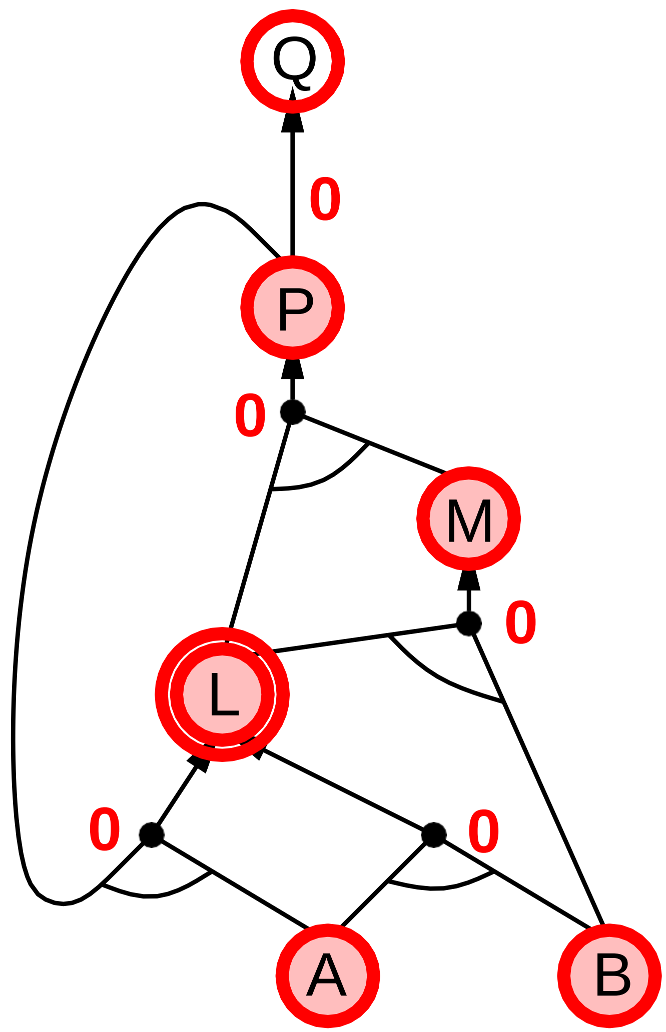
Forward chaining example



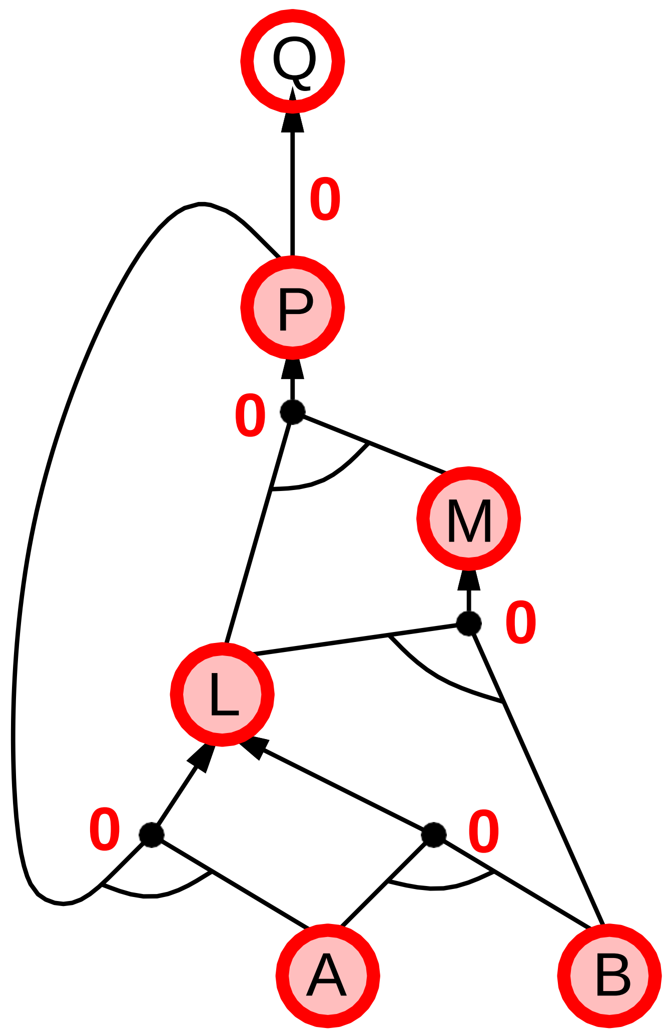
Forward chaining example



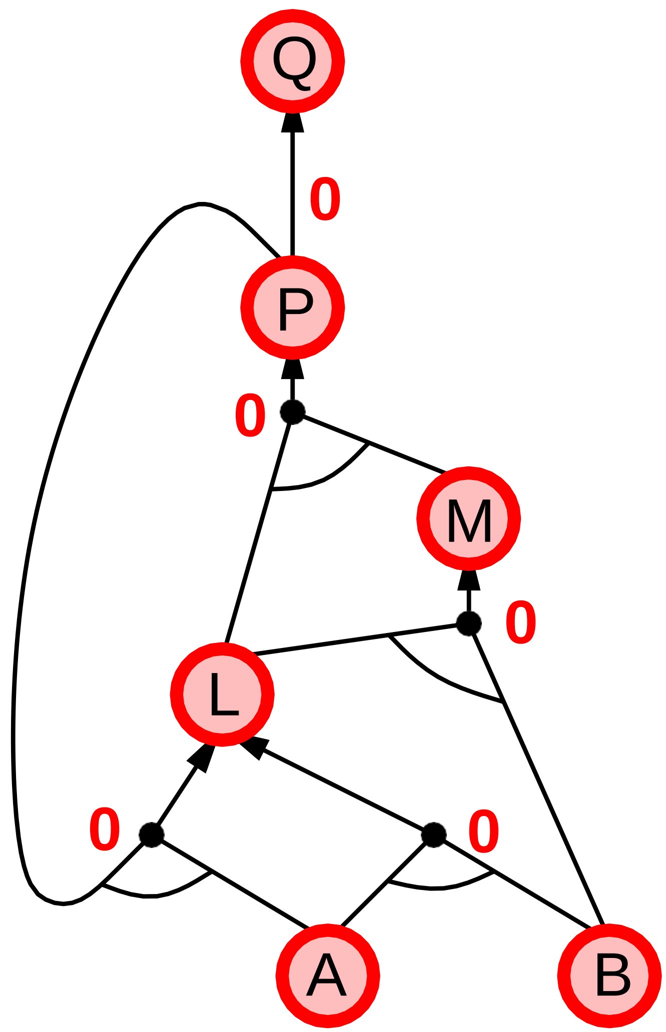
Forward chaining example



Forward chaining example



Forward chaining example



Forward chaining algorithm

```
function PL-FC-ENTAILS?(KB, q) returns true or false
  inputs: KB, the knowledge base, a set of propositional definite clauses
          q, the query, a proposition symbol
  count ← a table, where count[c] is initially the number of symbols in clause c's premise
  inferred ← a table, where inferred[s] is initially false for all symbols
  queue ← a queue of symbols, initially symbols known to be true in KB

  while queue is not empty do
     p ← POP(queue)
     if p = q then return true
     if inferred[p] = false then
        inferred[p] ← true
        for each clause c in KB where p is in c.PREMISE do
           decrement count[c]
           if count[c] = 0 then add c.CONCLUSION to queue
  return false
```

Pearson

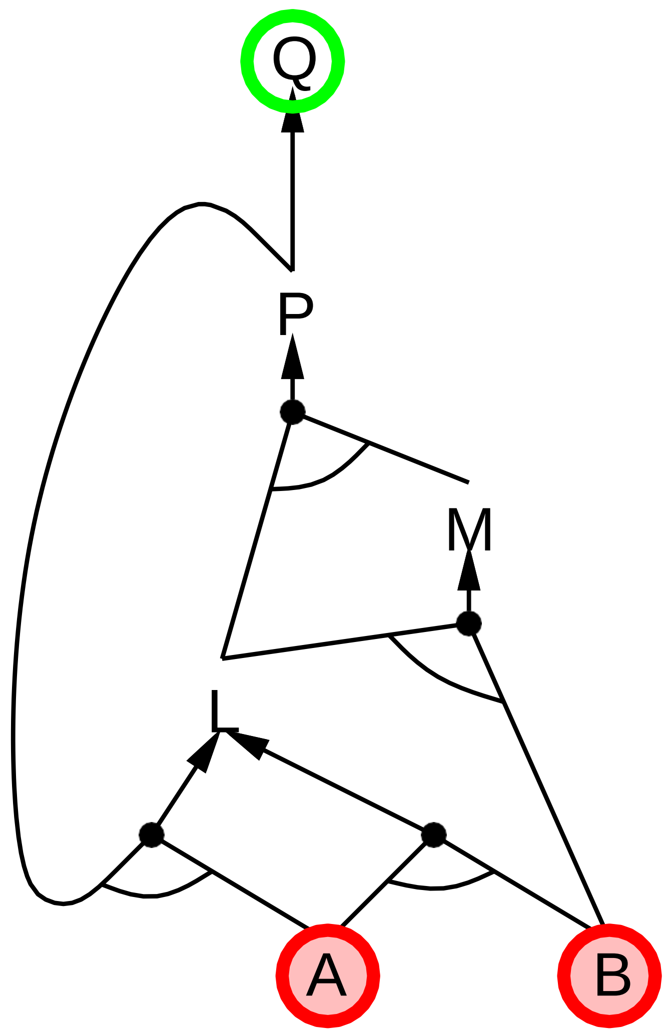
Backward chaining

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

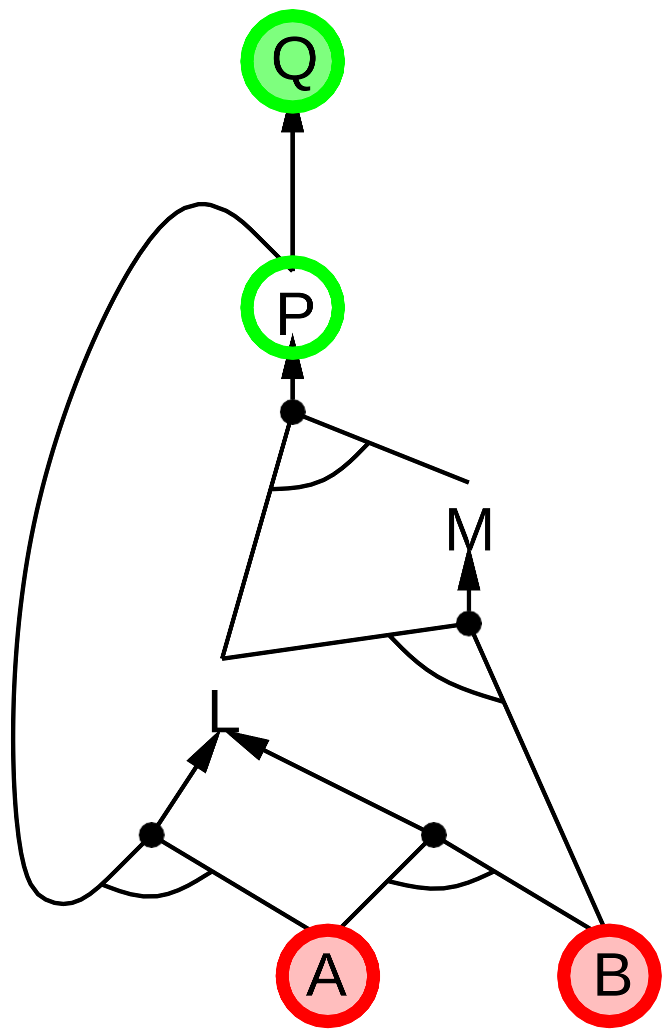
Avoid loops: check if new subgoal is already on the goal stack

Avoid repeated work: check if new subgoal
1) has already been proved true, or
2) has already failed

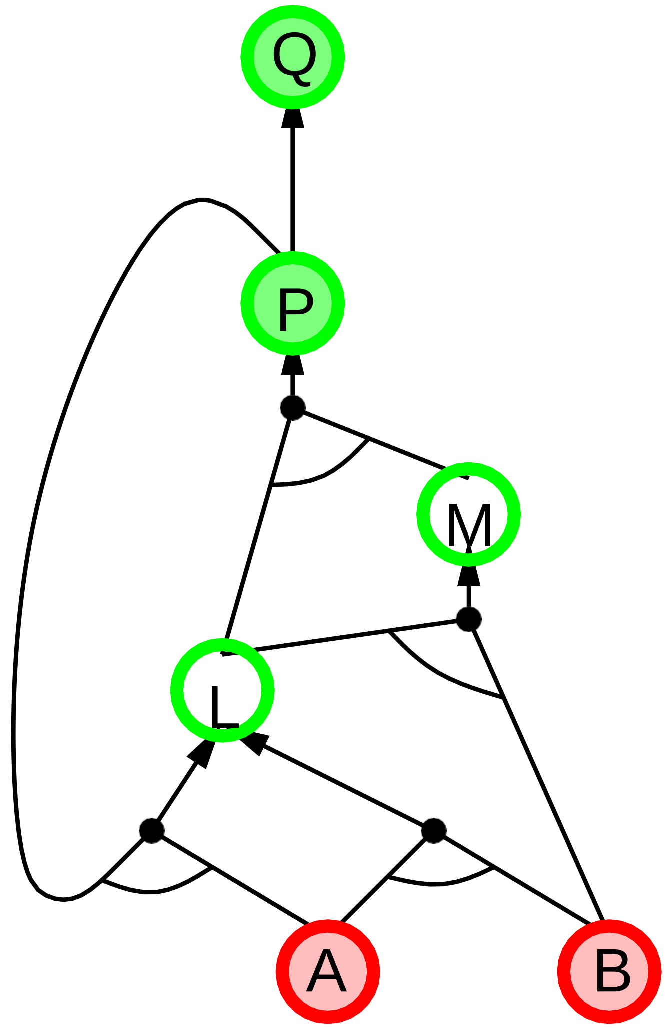
Backward chaining example



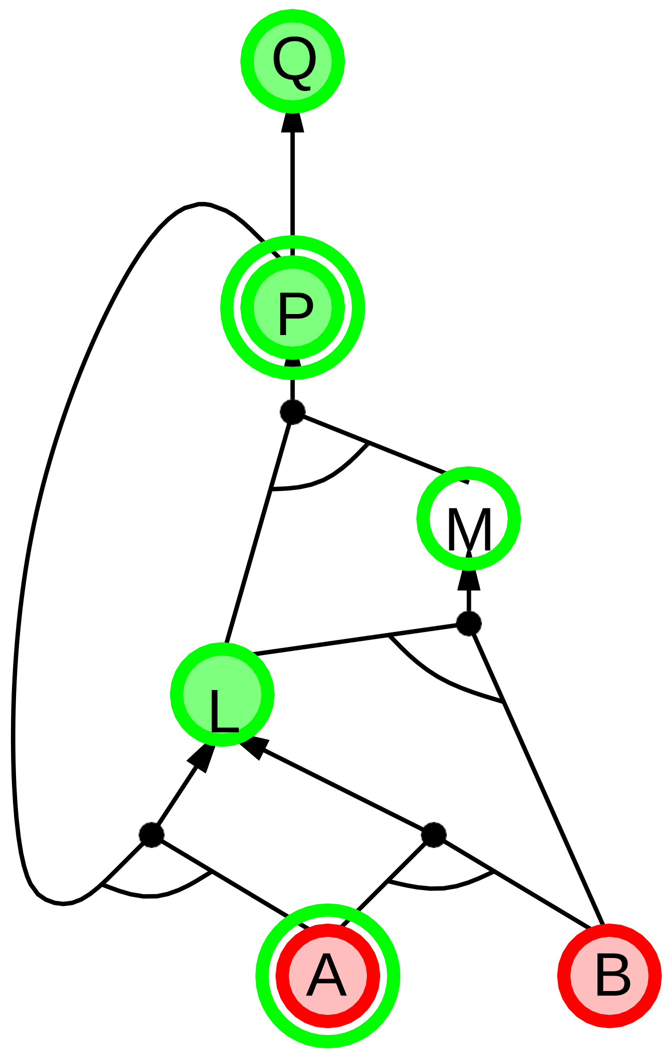
Backward chaining example



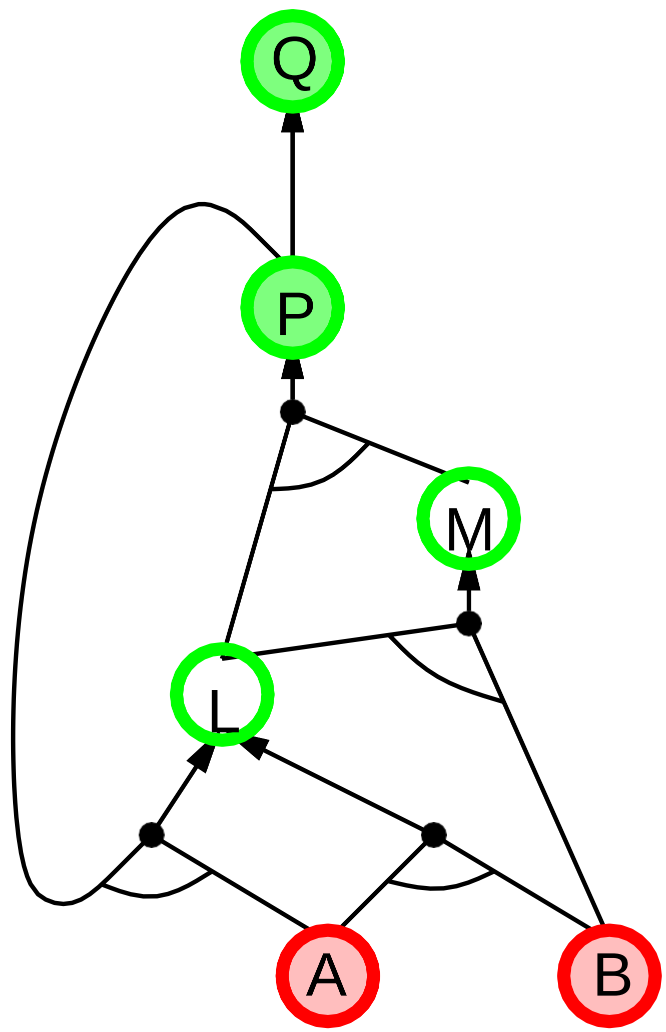
Backward chaining example



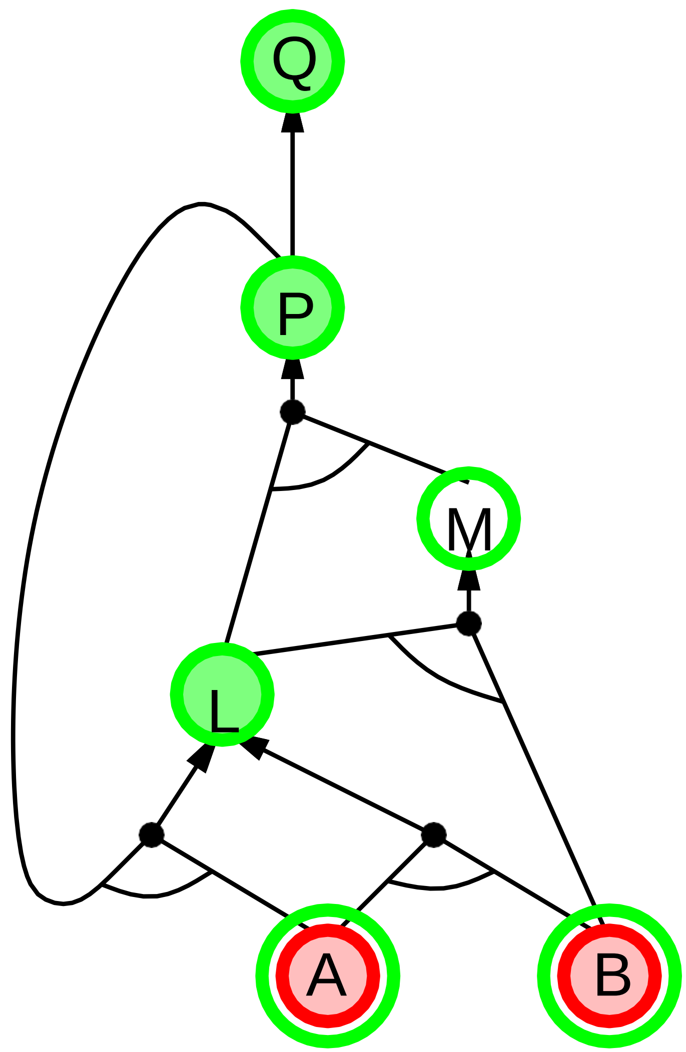
Backward chaining example



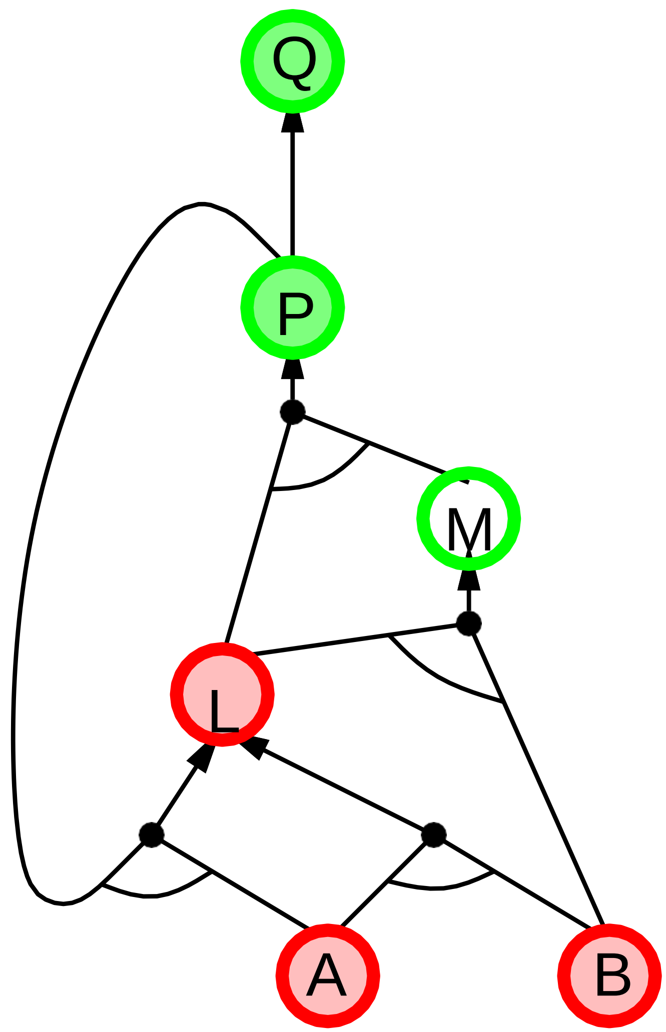
Backward chaining example



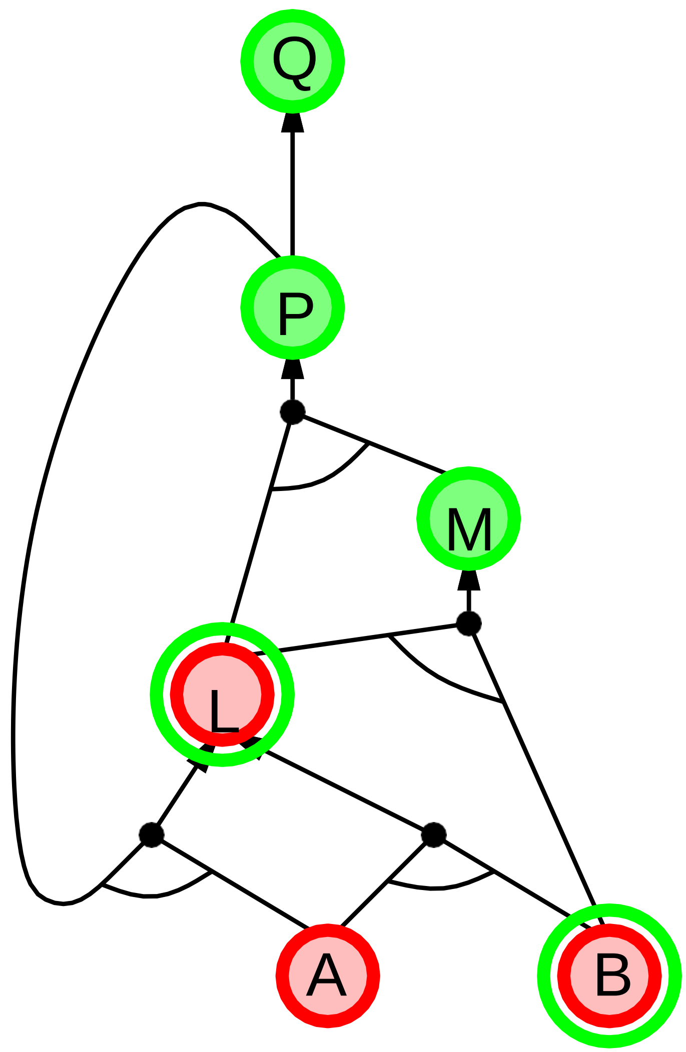
Backward chaining example



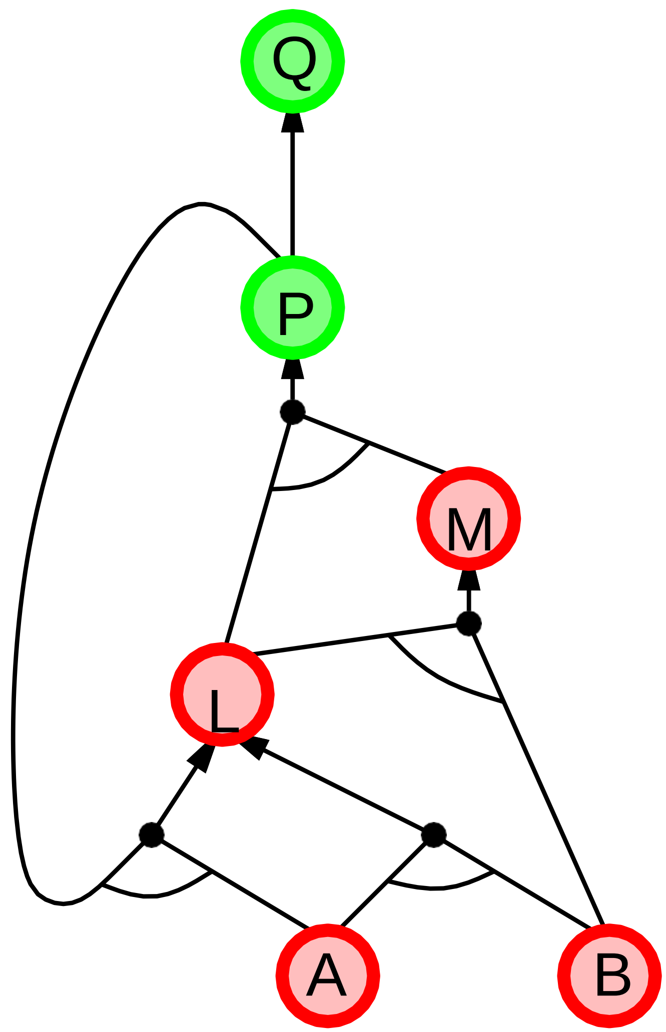
Backward chaining example



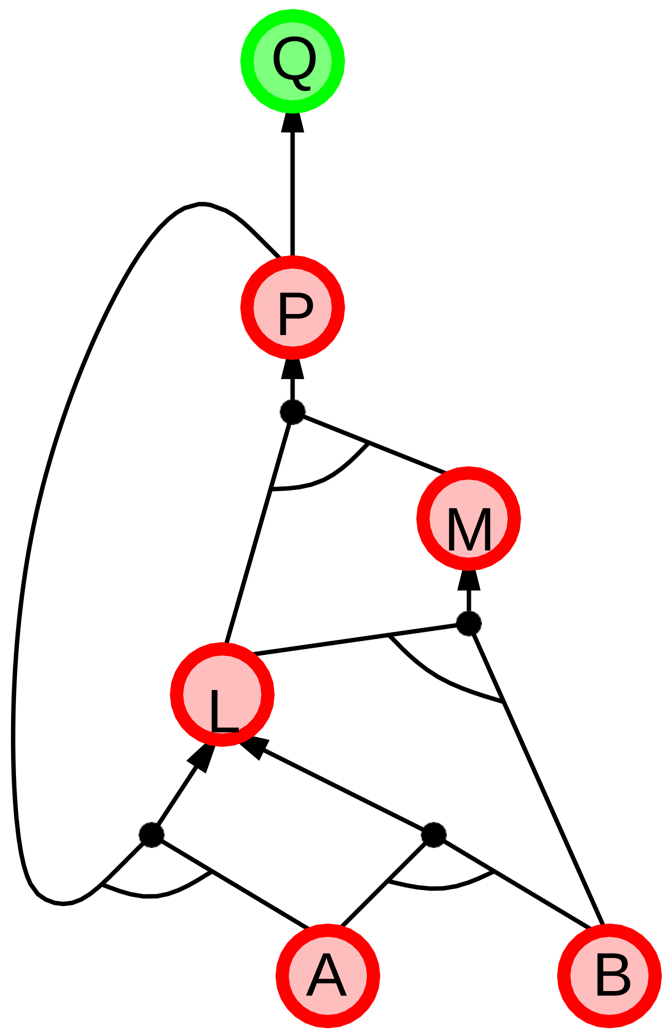
Backward chaining example



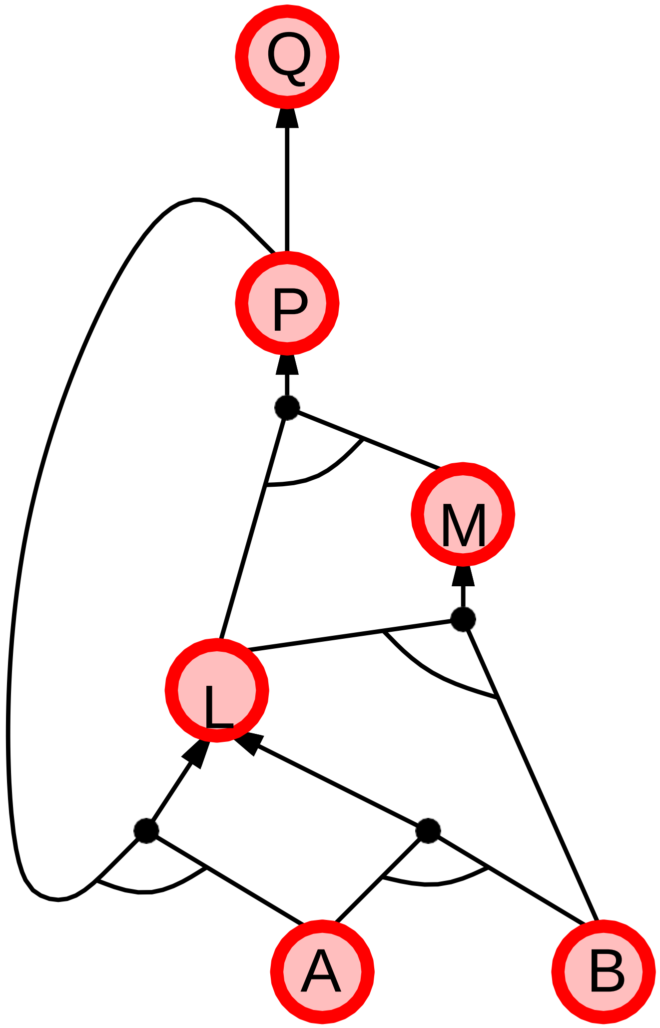
Backward chaining example



Backward chaining example



Backward chaining example



Forward vs. backward chaining

FC is data-driven, 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, 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

AIMA notebooks: «logic.ipynb»

Logic

This Jupyter notebook acts as supporting material for topics covered in **Chapter 6 Logical Agents**, **Chapter 7 First-Order Logic** and **Chapter 8 Inference in First-Order Logic** of the book *Artificial Intelligence: A Modern Approach*. We make use of the implementations in the logic.py module. See the intro notebook for instructions.

Let's first import everything from the `logic` module.

In [1]:
```
from utils import *
from logic import *
from notebook import psource
```

CONTENTS

https://github.com/aimacode/aima-python/blob/master/logic.ipynb

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
- soundess: 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 ...but... Propositional logic lacks expressive power

Local search methods (WALKSAT) find solutions (sound but not complete).

In the next lecture...

- Why First-Order Logic (FOL)?
- Syntax and semantics of FOL
- Fun with sentences
- Wumpus world in FOL
- Knowledge Engineering in FOL