CS5010 Artificial Intelligence Principles: Lecture 7

Logical agents

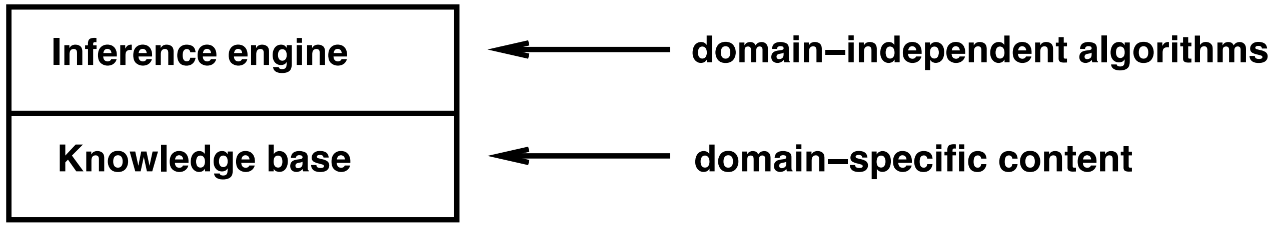
Chapter 7

Outline

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

In machine learning, our agents had evidence with associated probabilities. We now consider agents with knowledge about the environment

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 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

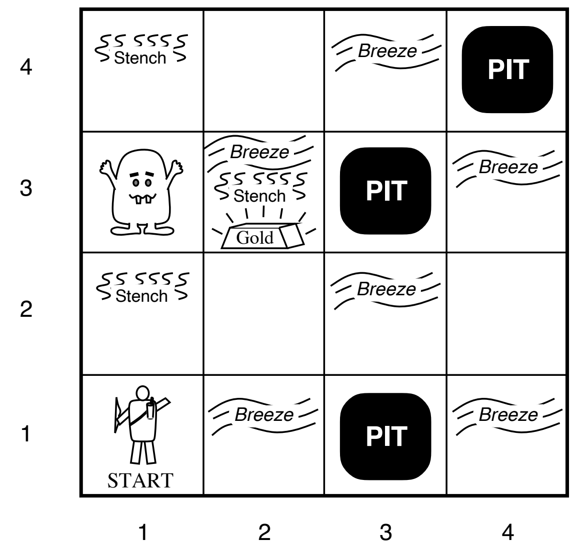
Update internal representations of the world

Deduce hidden properties of the world

Deduce appropriate actions

Wumpus World PEAS description

[4,2] means 4th column, 2nd row



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

also Climb, but only from [1,1]

Sensors Breeze, Glitter, Smell

also Scream and Bump

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

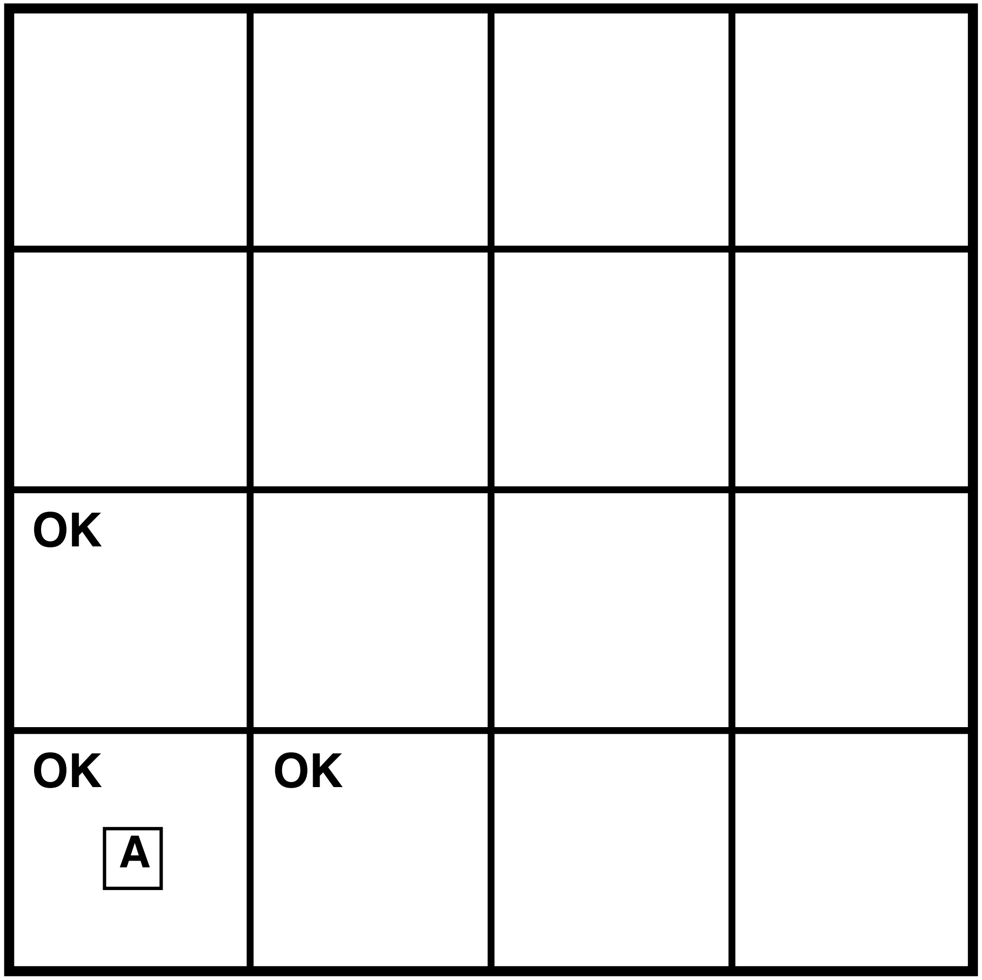
Uncertainty still exists, since the gold and wumpus squares are chosen from a random uniform distribution (excluding the start square). Also, any non-start square can be a pit with probability 0.2

Hence deterministic once these choices have been made

The gold can be in a pit - about 21% of instances - so the game can be unfair

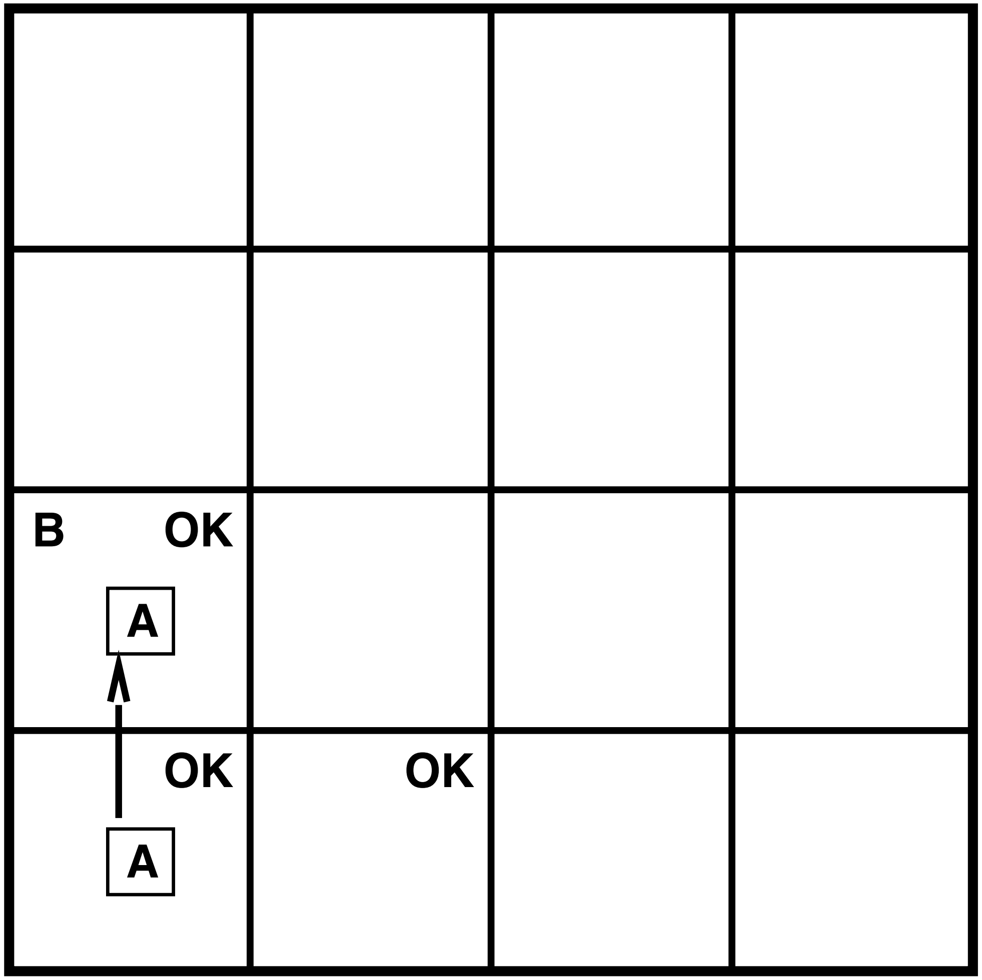
Exploring a wumpus world

Percepts of the form [stench, breeze, glitter, bump, scream]



Initial Percepts: [none, none, none, none, none]
We can deduce that [1,1], [1,2] and [2,1] are OK

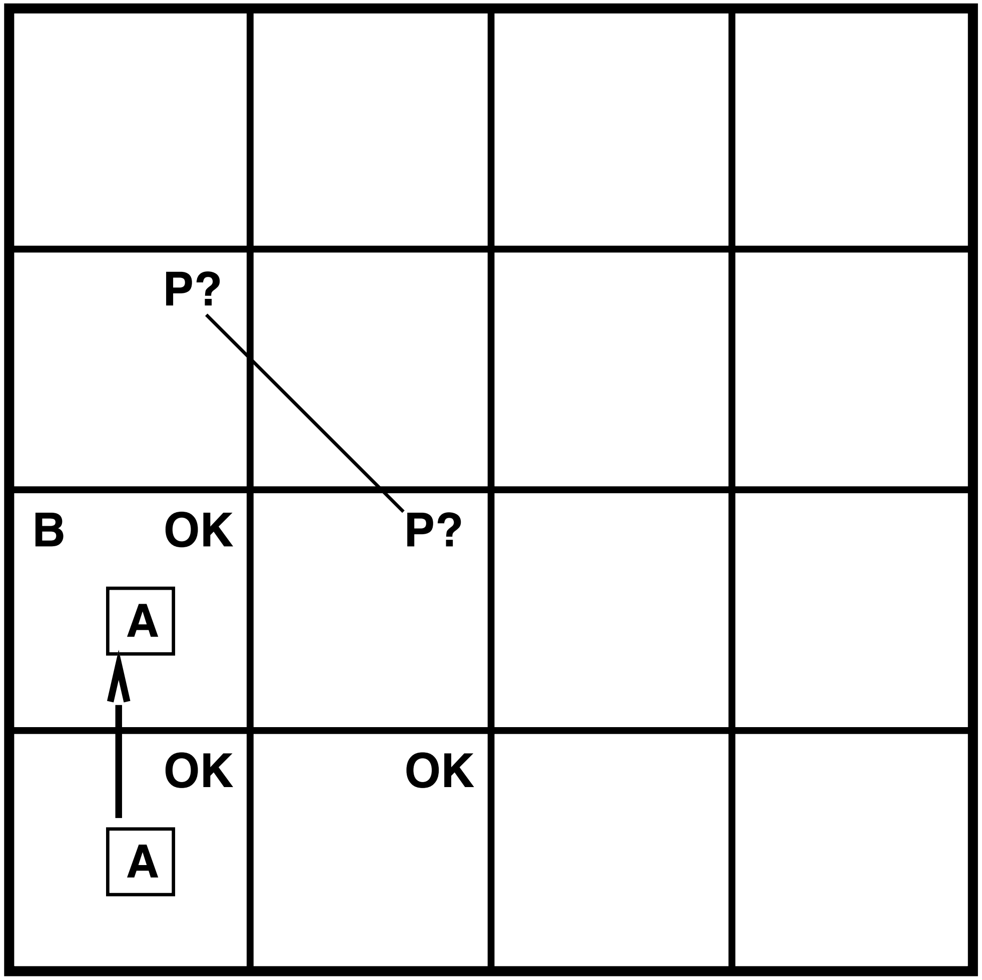
Exploring a wumpus world



Percepts at [1,2]: [none, breeze, none, none, none]

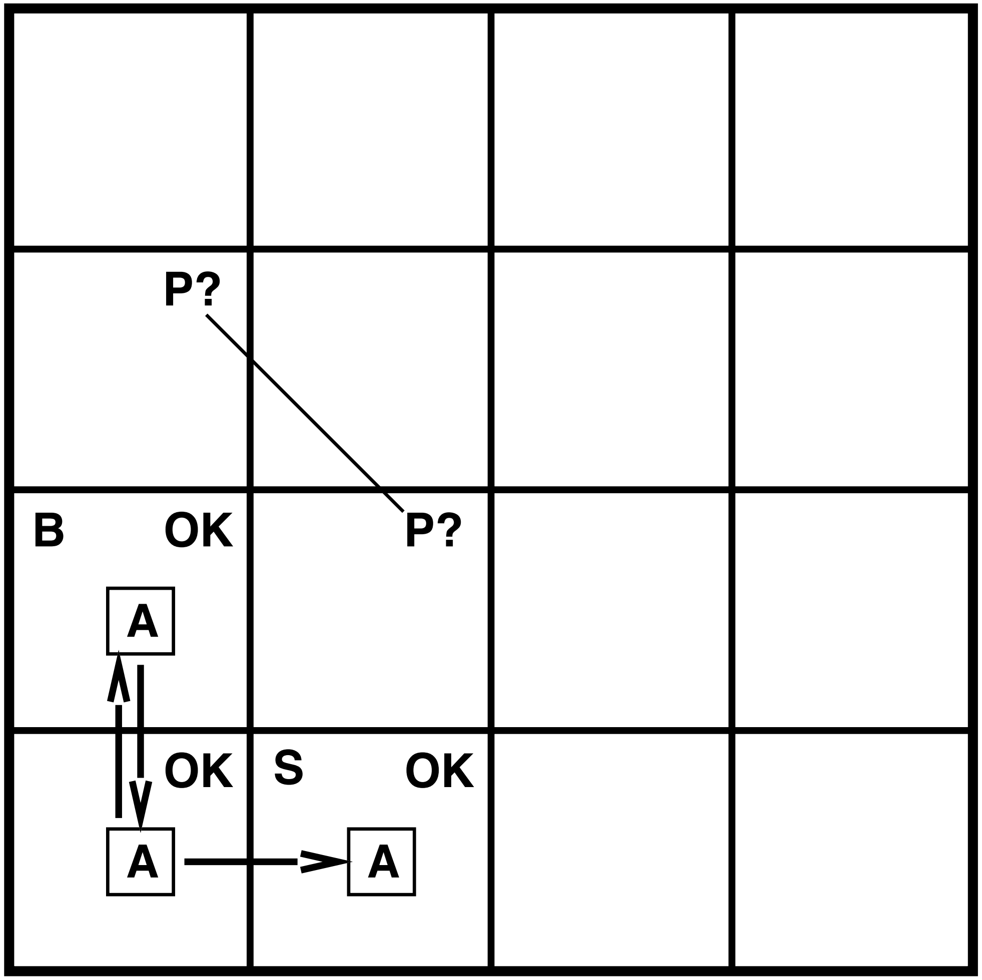
Must be a pit in [1,3] or [2,2] or both, since [1,1] was OK

Exploring a wumpus world



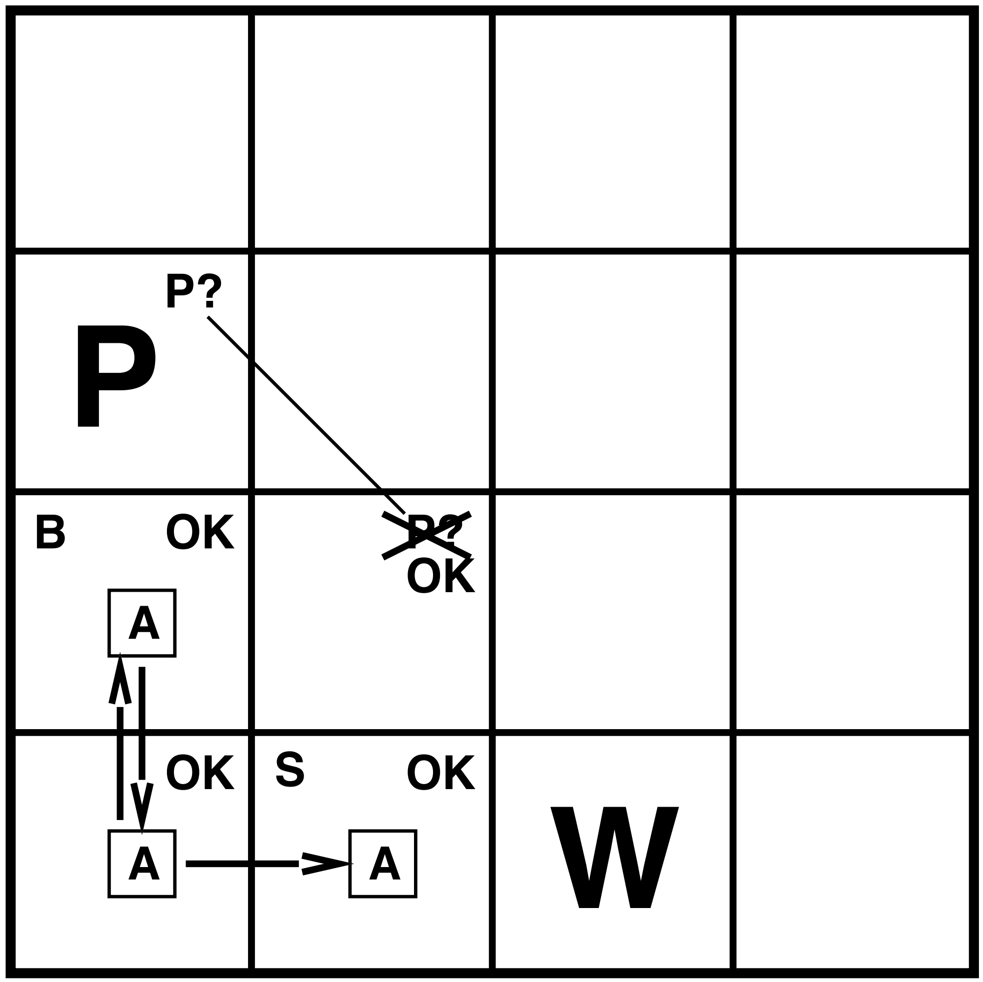
Our agent is prudent, and backtracks to get more information

Exploring a wumpus world



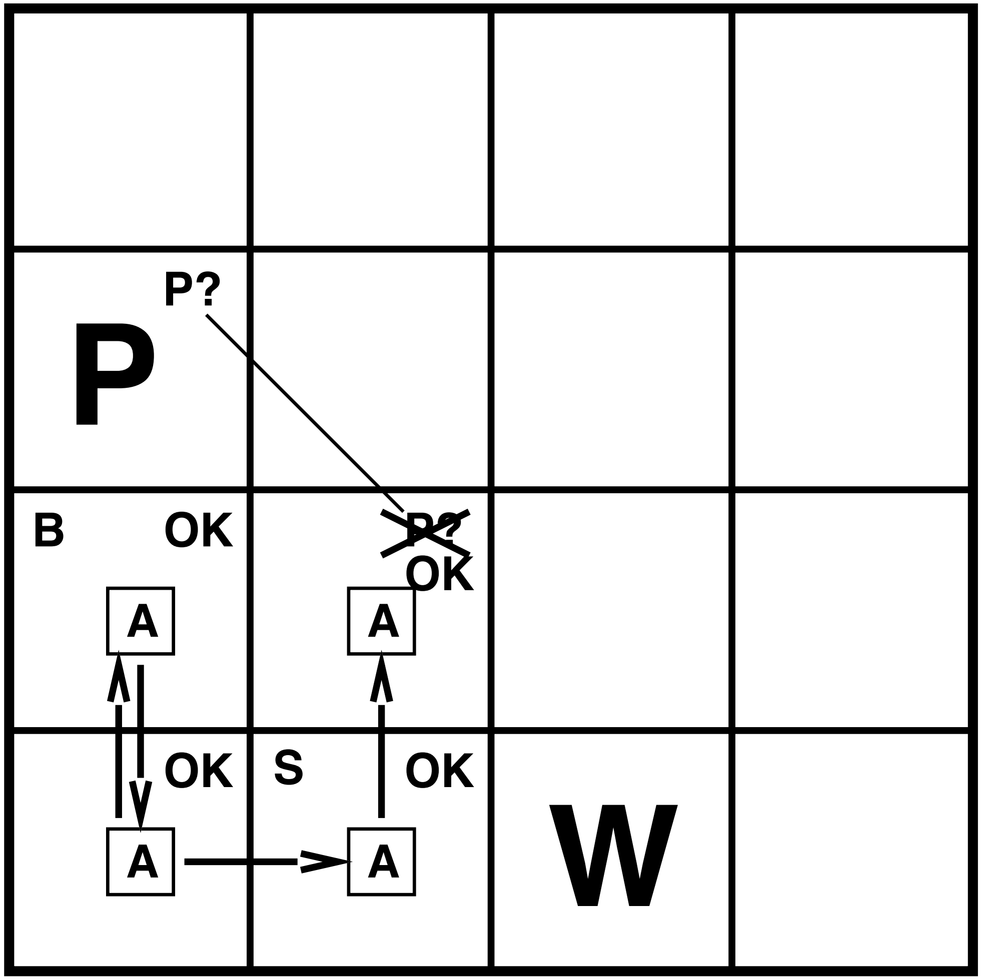
Percept at [2,1] is [stench, none, none, none, none]

Exploring a wumpus world



Positive inference: no Wumpus at [1,1]
Negative inference: no Wumpus at [2,2] since no stench at [1,2]
Combined inference: the Wumpus is in [3,1]

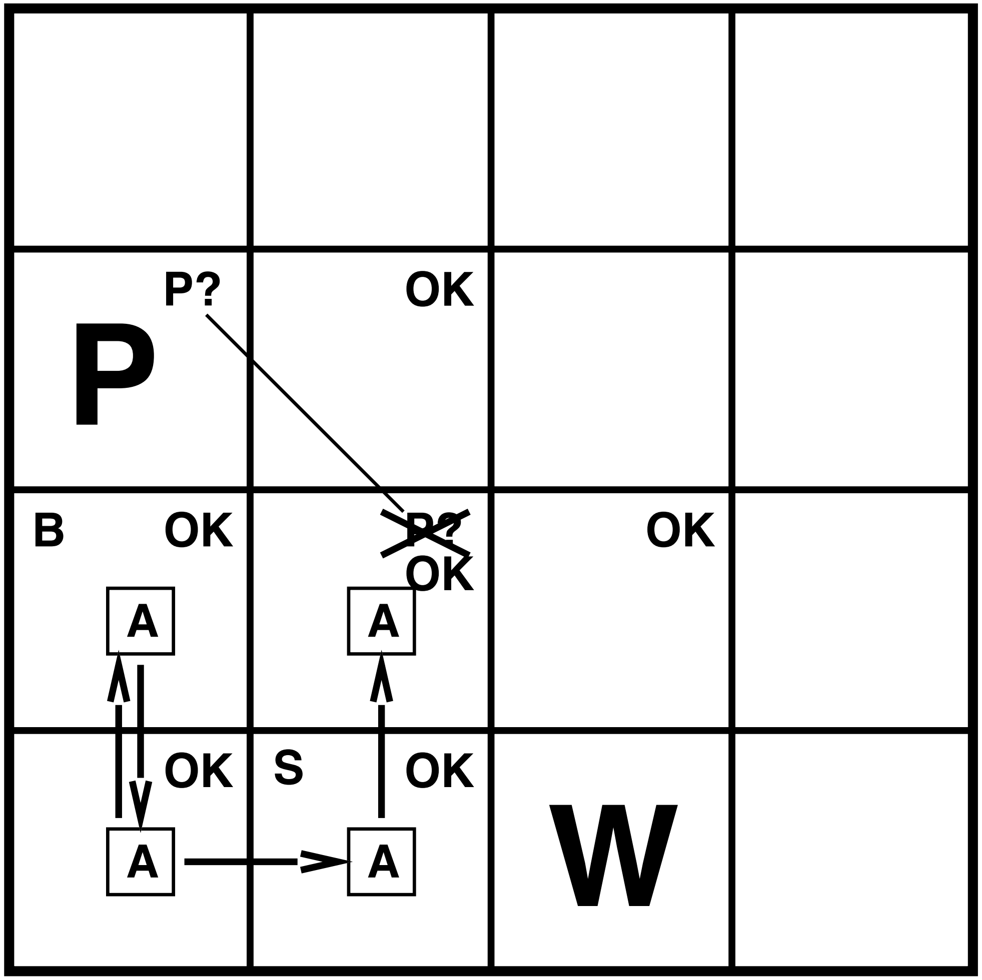
Exploring a wumpus world



Further inference: no breeze at [2,1] means no pit at [2,2]
Which means (a) it's safe to go there and (b) the pit is in [1,3]

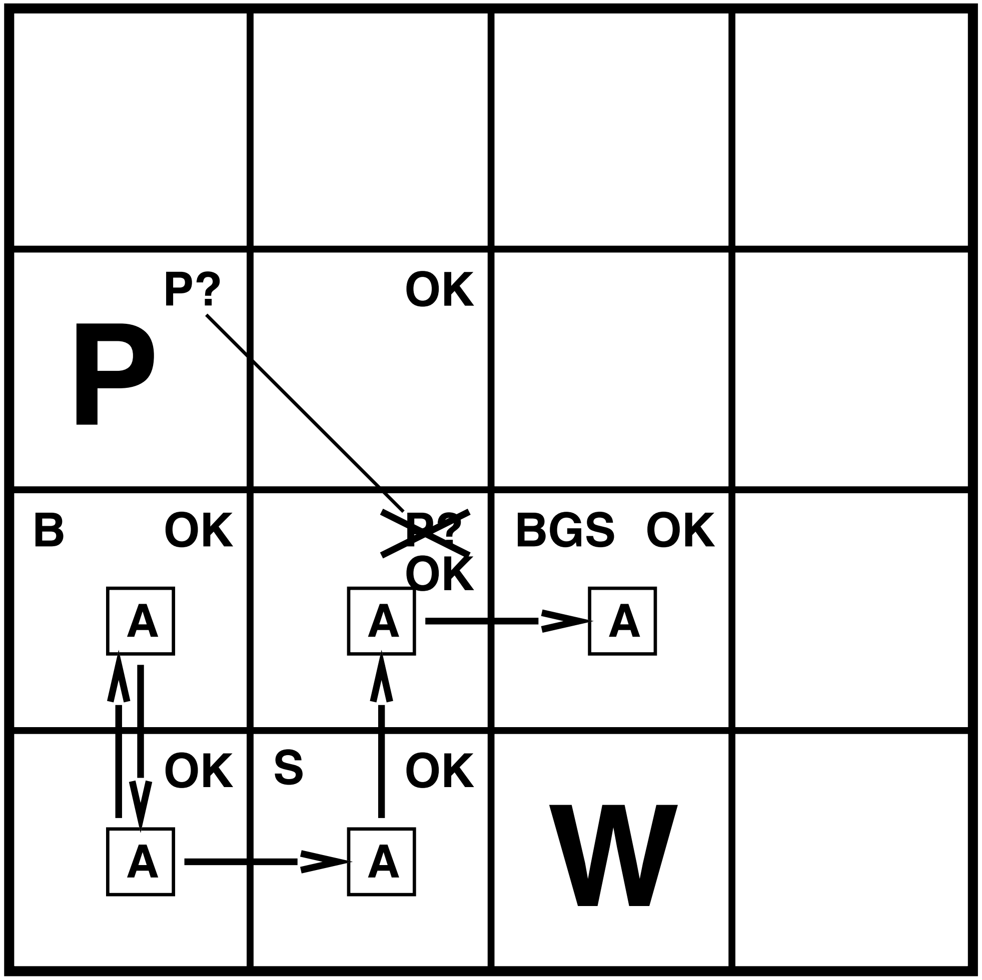
We've combined knowledge from different times and places with knowledge of things NOT happening to make useful inferences

Exploring a wumpus world



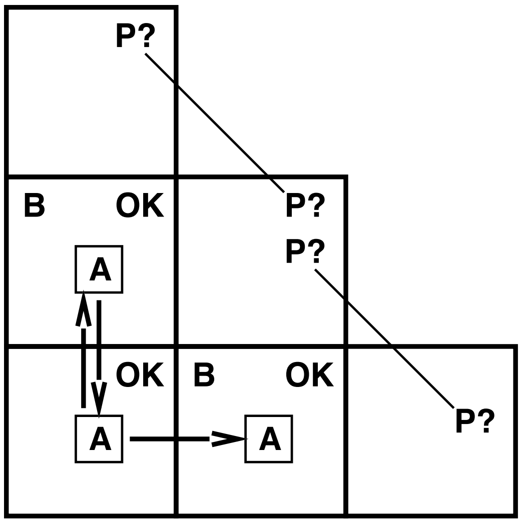
Skip the percepts at [2,2] for this example

Exploring a wumpus world



Assume the agent goes to [3,2], senses glitter, grabs the gold, returns to the start and climbs out of the cave

Other tight spots

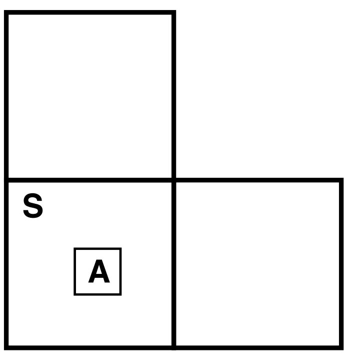


Breeze in (1,2) and (2,1)

$\Rightarrow$ no safe actions

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

This calculation is not easy: we'll return to this later in the course. Look at Ch 13 if interested now



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 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:

$KB \models \alpha$ Sequence of percepts entailed Wumpus in [1,3]

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**

= is syntax. The meaning "equals" is semantics. Could mean string concatenation in javascript, etc.

Note: brains process **syntax** (of some sort)

Models

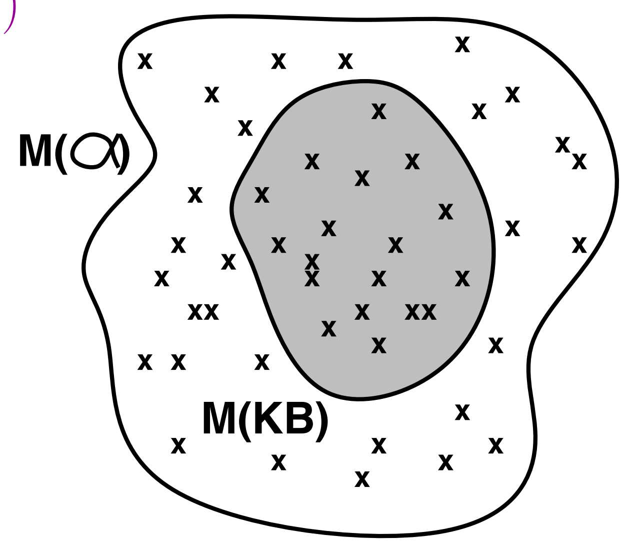
Logicians typically think in terms of models, which are formally structured worlds with respect to which truth can be evaluated

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

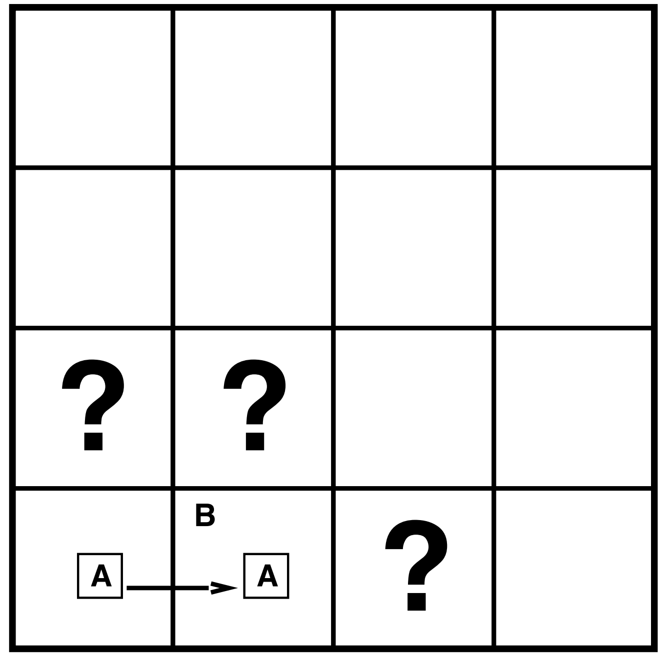
$M(\alpha)$ is the set of all models of α

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

E.g. $KB =$ Giants won and Reds won
$\alpha =$ Giants won



Entailment in the wumpus world



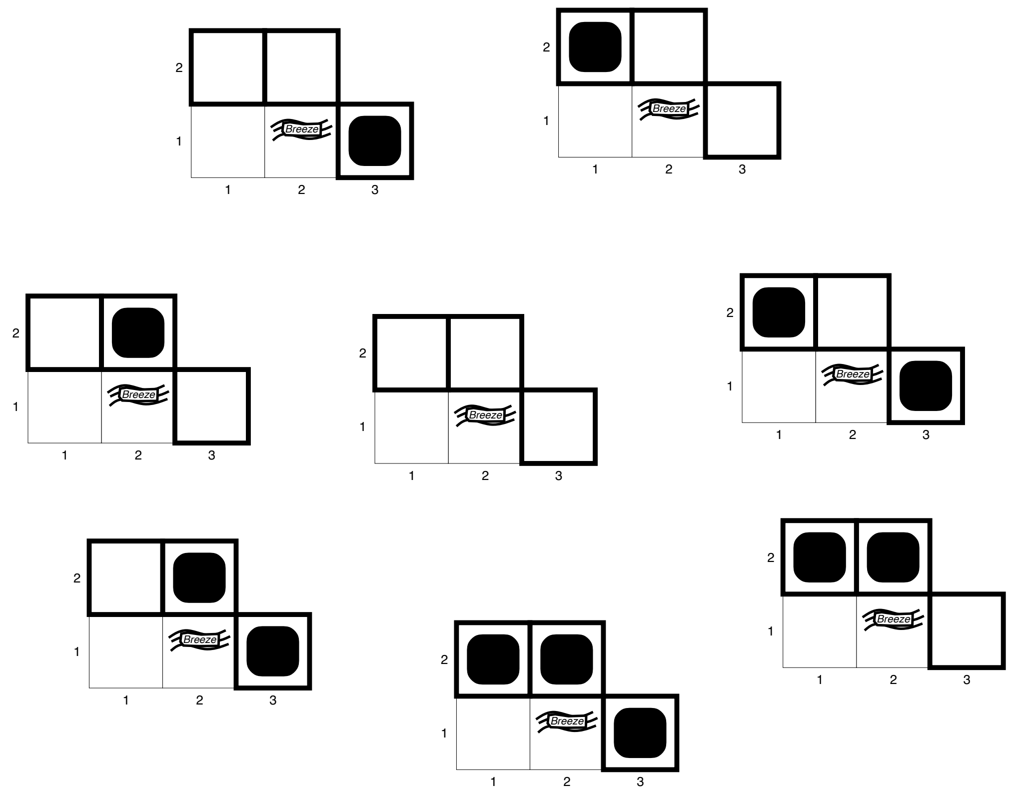
Situation after detecting nothing in [1,1], moving right, breeze in [2,1]

Consider possible models for ?s assuming only pits

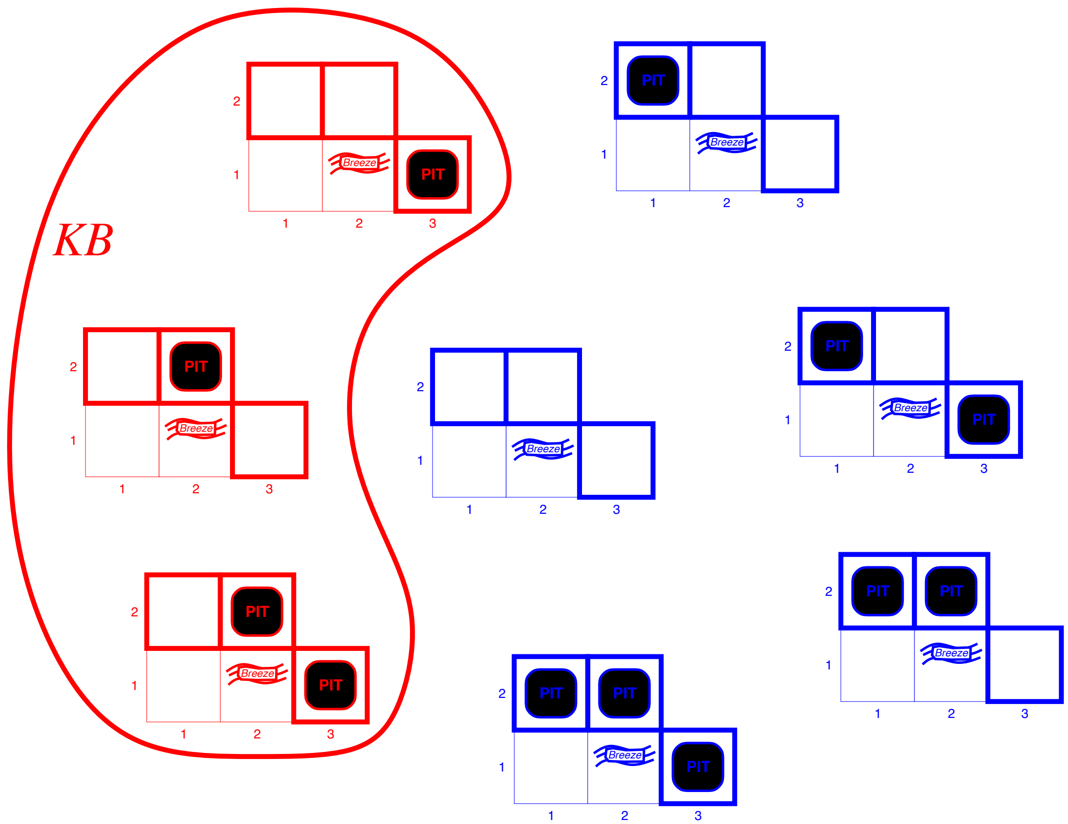
3 Boolean choices $\Rightarrow$ 8 possible models

Pit in zero, one, two or three places

Wumpus models

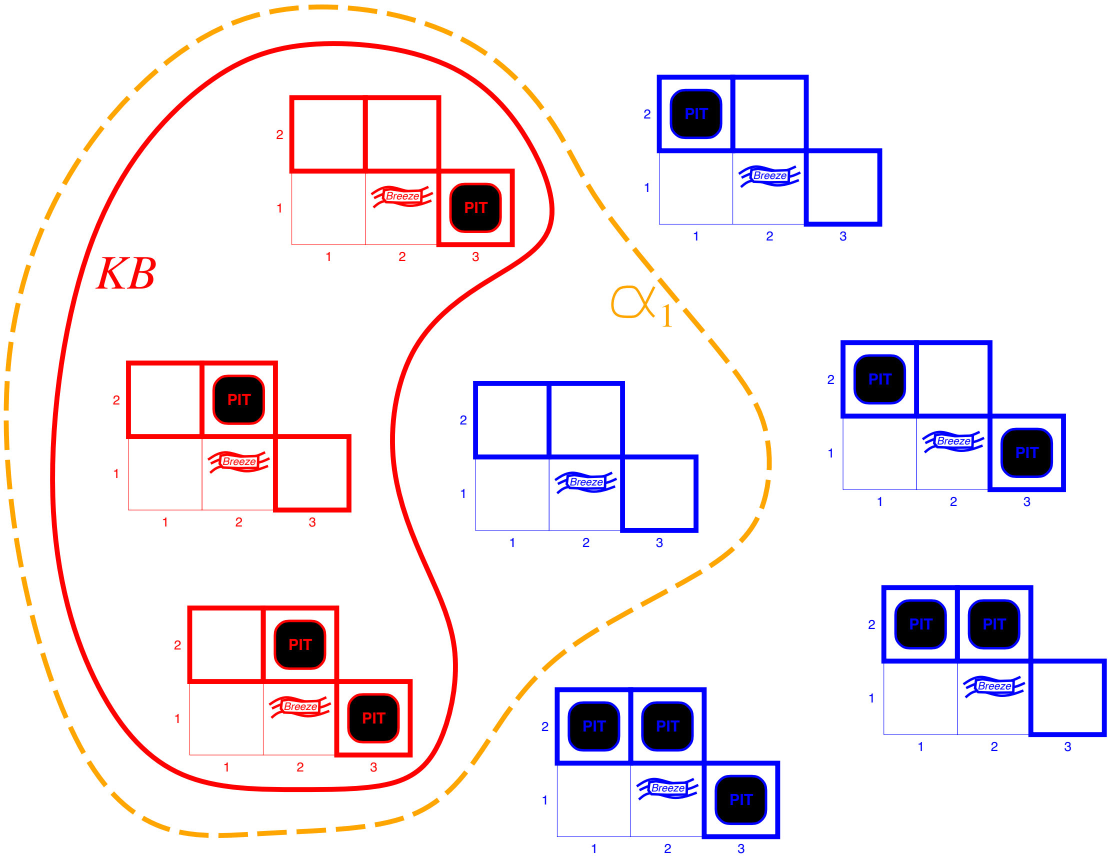


Wumpus models



KB = wumpus-world rules + observations

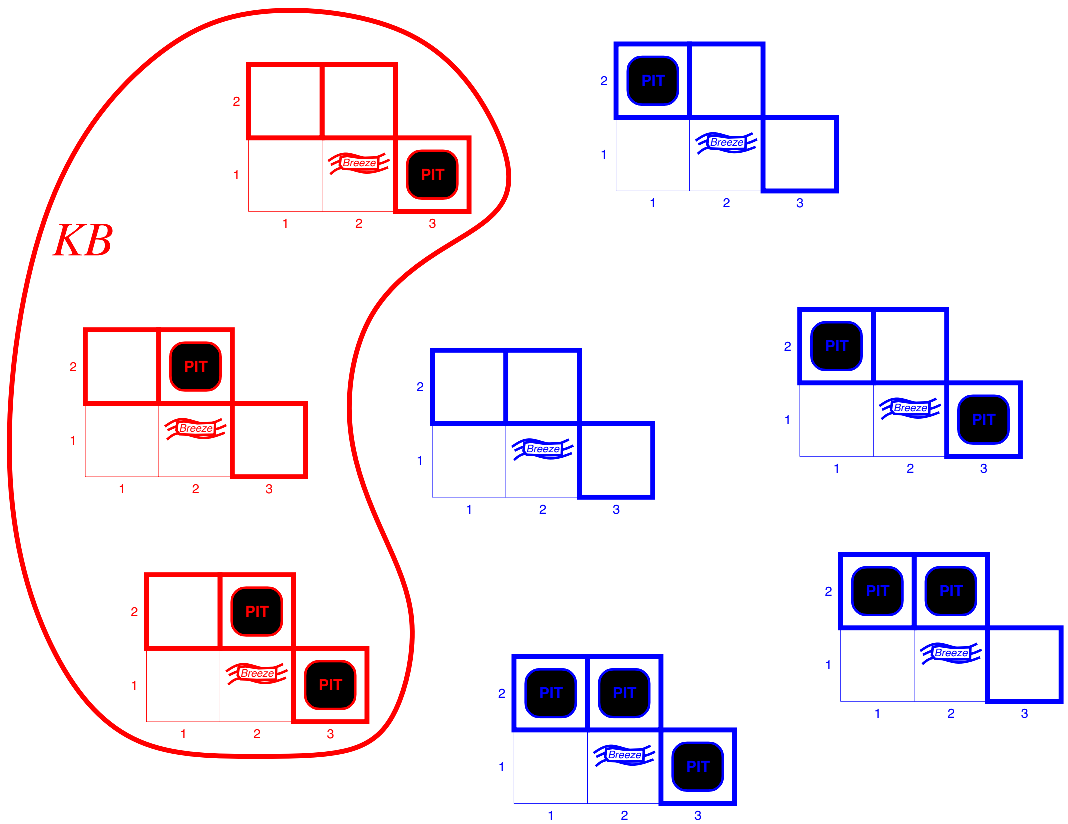
Wumpus models



KB = wumpus-world rules + observations

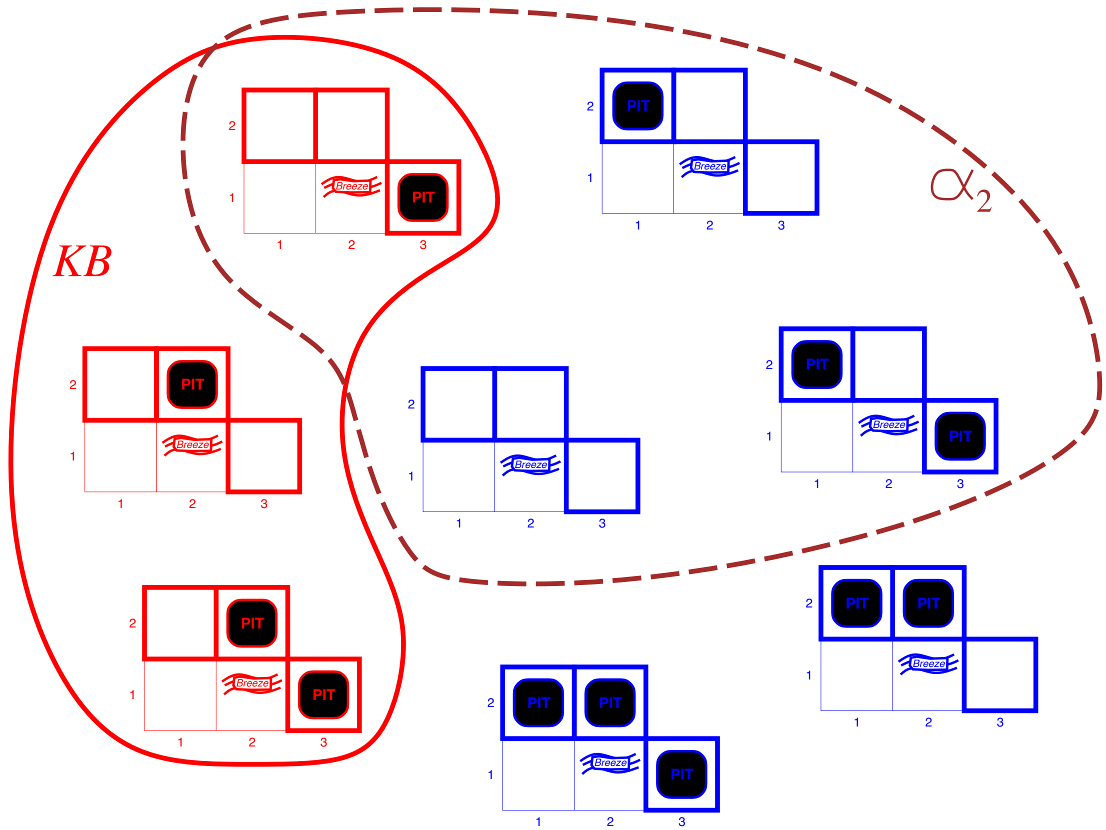
α_1 = "[1,2] is safe", $KB \models \alpha_1$, proved by model checking

Wumpus models



KB = wumpus-world rules + observations

Wumpus models



KB = wumpus-world rules + observations

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

Inference

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

Consequences of KB are a haystack; α 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 $KB \models \alpha$

Completeness: 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) 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 $[i, j]$.

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

"Pits cause breezes in adjacent squares"

Wumpus world sentences

Construct a simple KB:

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 $[i,j]$.

$\neg P_{1,1}$ R1

$\neg B_{1,1}$

$B_{2,1}$

"Pits cause breezes in adjacent squares"

$B_{1,1} \Leftrightarrow (P_{1,2} \vee P_{2,1})$ R2

$B_{2,1} \Leftrightarrow (P_{1,1} \vee P_{2,2} \vee P_{3,1})$ R3

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

R1, R2 and R3 are always true in Wumpus world
Add R4 and R5 as breeze percepts for the specific world our agent is in

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*

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

7 Boolean variables so 2^(7) rows in the truth table
In three of these the KB is true
In those three, there is no pit in [1,2]
So we have derived no pit in [1,2] from the KB by
exhaustive case analysis

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**

Logical equivalence

Two sentences are logically equivalent iff true in same models:

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

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

We use these as inference rules

Validity and satisfiability

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

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

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., $A \lor B$, C

A sentence is unsatisfiable if it is true in **no** models

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

Satisfiability is connected to inference via the following:

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

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

Example claim: No integers a and b exist for which 24y + 12z = 1

Choosing y,z pairs and testing would take forever

Suppose such y and z exist

Then 2y + z = 1/12

Adding integers does not yield a fraction

So the supposition leads to a contradiction

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 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

Searching for proofs is an alternative to enumerating models.
The idea is to ignore irrelevant propositions, and hence be more efficient

Forward and backward chaining

Horn Form (restricted)

KB = **conjunction** of **Horn clauses**

Horn clause =

- ♢ proposition symbol; or
- ♢ (conjunction of symbols) $\Rightarrow$ symbol

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

Whenever sentences of the forms above the line are given, we can infer the sentence below the line

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

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

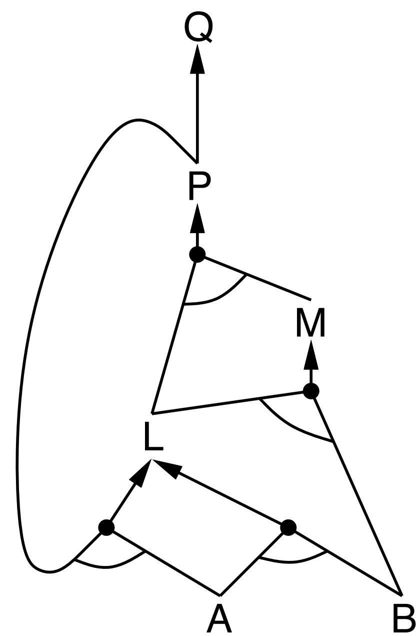
Can be used with forward chaining or backward chaining.
These algorithms are very natural and run in **linear** time

Also use And-Elimination: from a conjunction, any of the conjuncts can be inferred

Forward chaining

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

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



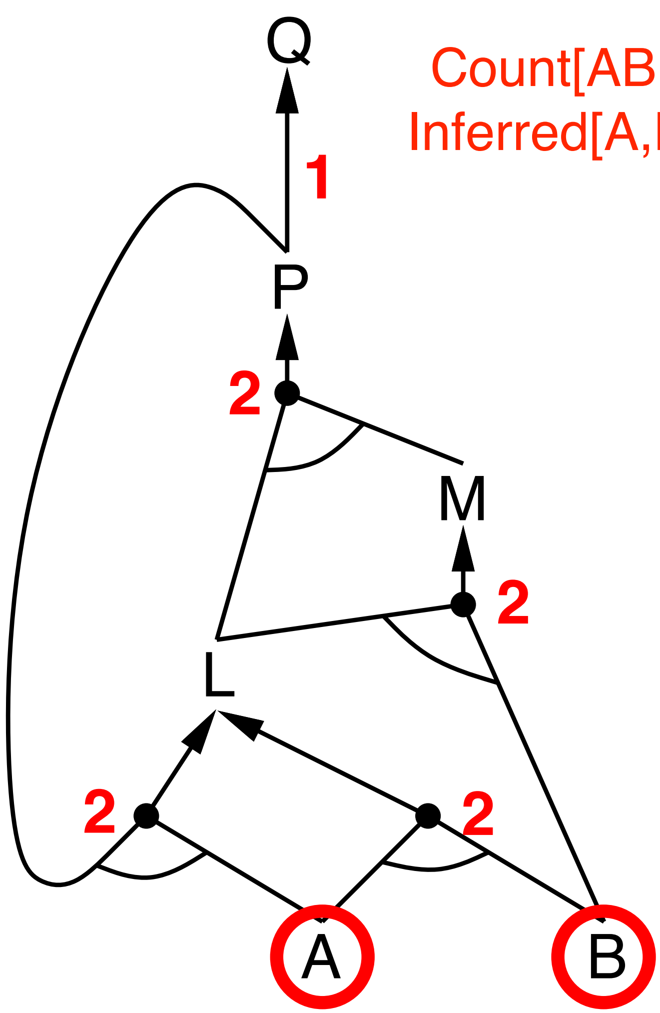
A and B are facts

AND-OR graph
arcs denote a conjunction
no arc denotes disjunction

Forward chaining algorithm

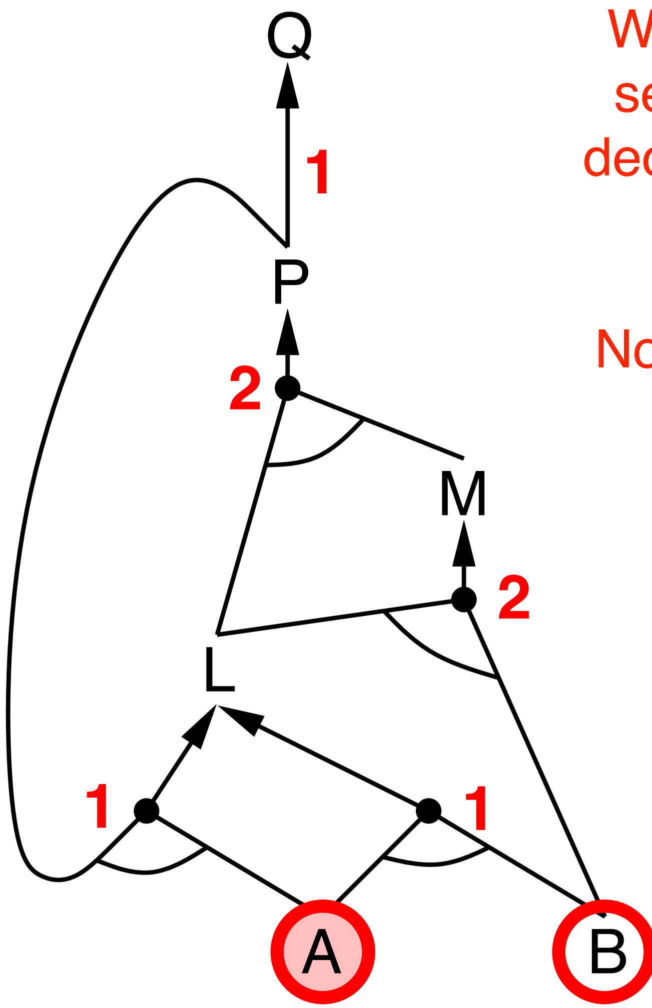
```
function PL-FC-ENTAILS?(KB, q) returns true or false
   inputs: KB, the knowledge base, a set of propositional Horn clauses
           q, the query, a proposition symbol
  local variables: count, a table, indexed by clause, initially the number of premises
                   inferred, a table, indexed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



Count[AB,AP,BL,LM,P] is [2,2,2,2,1]
Inferred[A,B,L,P,M,Q] is [false,...,false]
Agenda is [A,B]

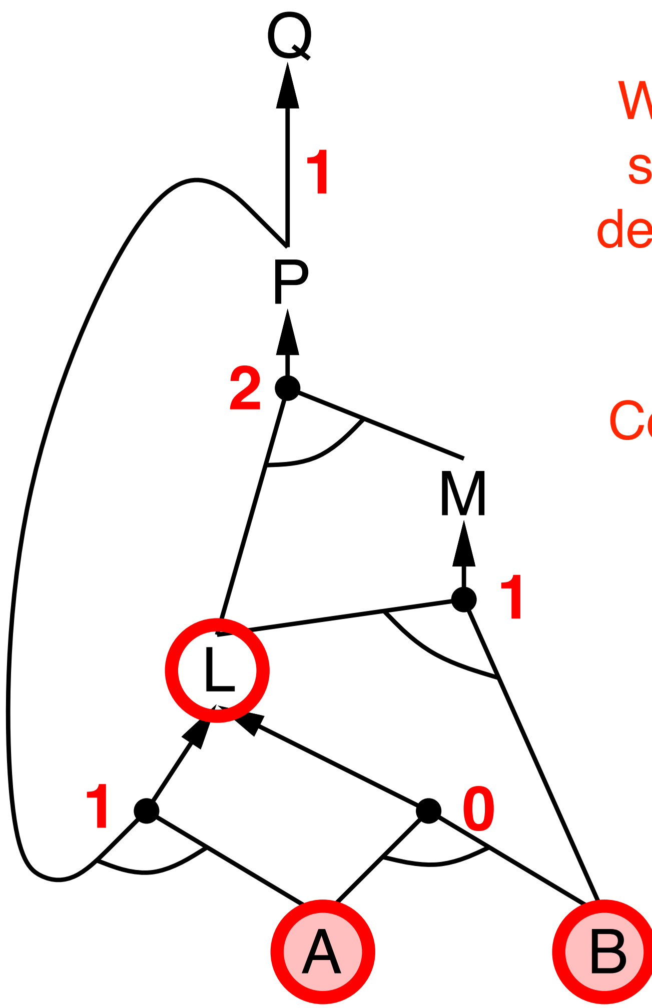
Forward chaining example



We pop A from the agenda, set Inferred[A] as true, and decrement counts associated with A

Note that none of the counts is zero

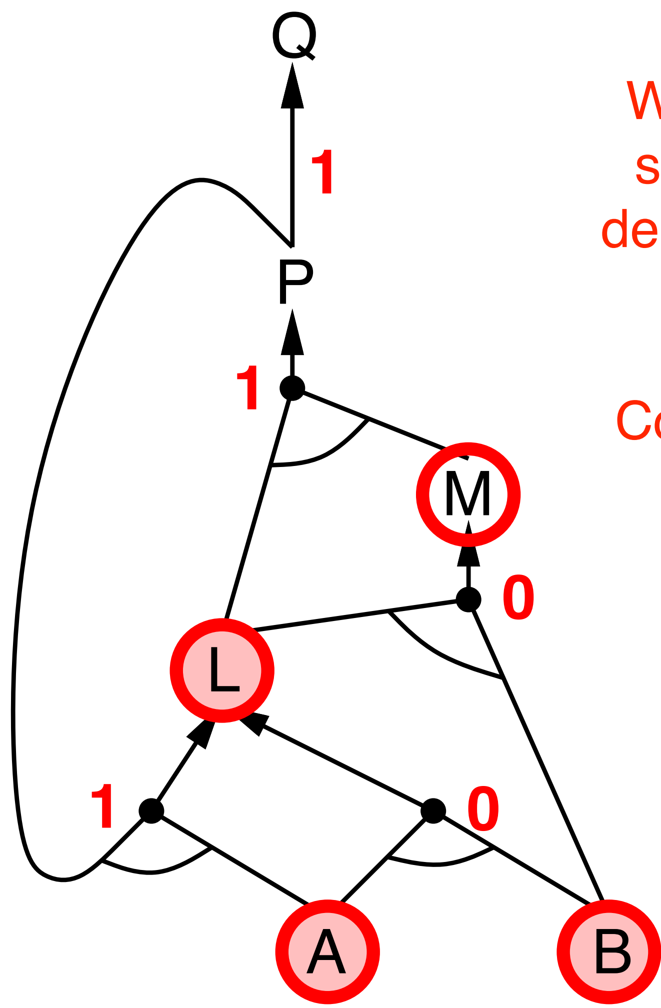
Forward chaining example



We pop B from the agenda, set Inferred[B] as true, and decrement counts associated with B

Count AB is zero, so we add its conclusion, L, to the agenda

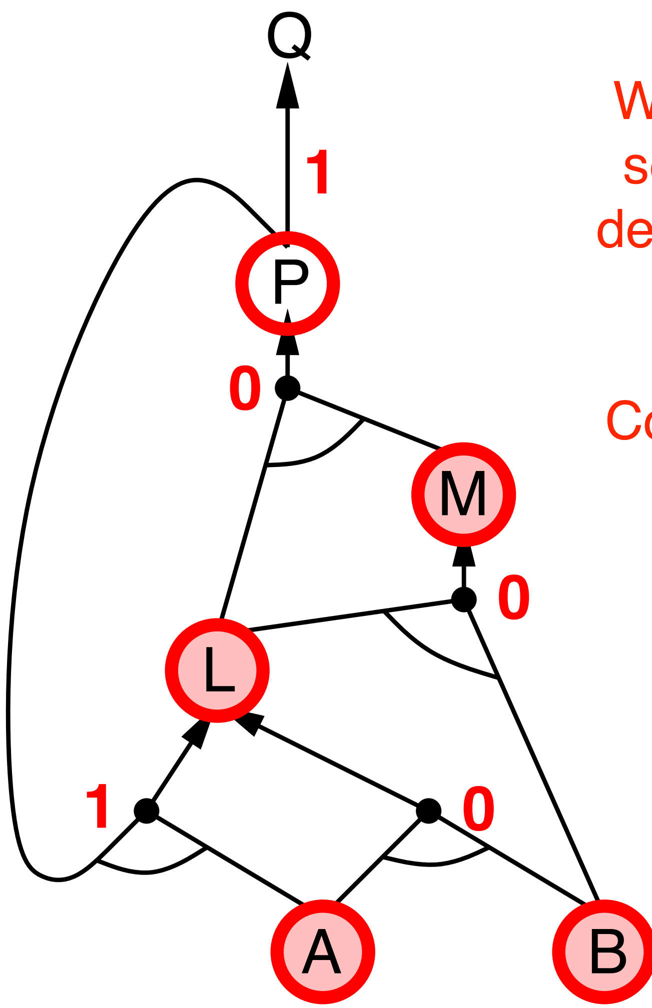
Forward chaining example



We pop L from the agenda, set Inferred[L] as true, and decrement counts associated with L

Count BL is zero, so we add its conclusion, M, to the agenda

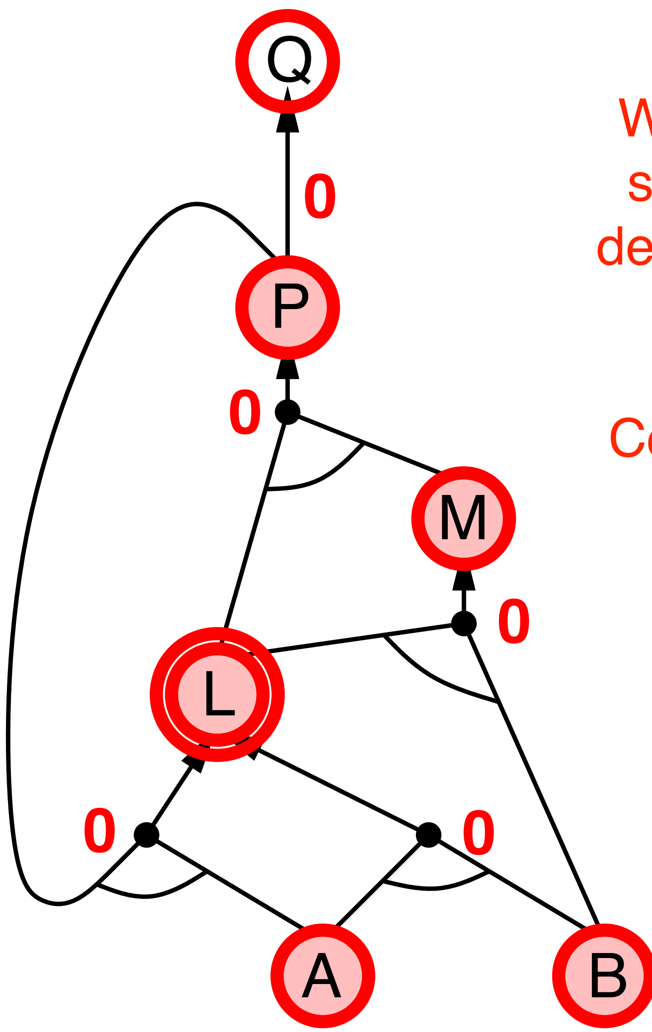
Forward chaining example



We pop M from the agenda, set Inferred[M] as true, and decrement counts associated with M

Count LM is zero, so we add its conclusion, P, to the agenda

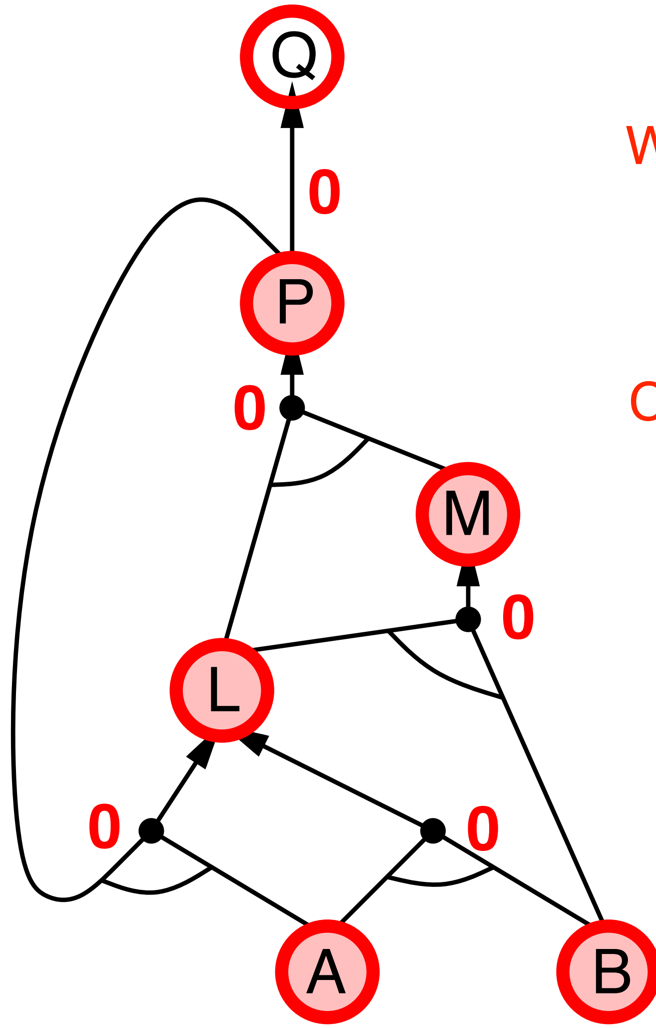
Forward chaining example



We pop P from the agenda, set Inferred[P] as true, and decrement counts associated with P

Count AP is zero, so we add its conclusion, L, to the agenda

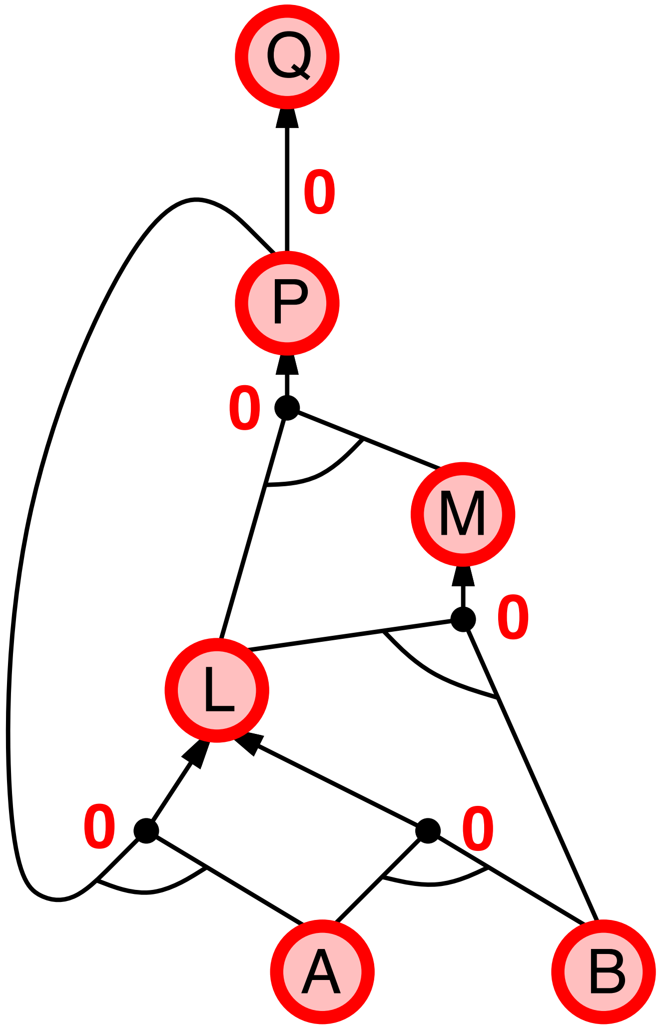
Forward chaining example



We pop L from the agenda, and decrement counts associated with L

Count P is zero, so we add its conclusion, Q, to the agenda

Forward chaining example



The final if statement now holds, so we return true

FC is sound, complete and linear in the size of the KB

Proof of completeness

FC derives every atomic sentence that is entailed by KB

atomic means one variable or its negation

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 α

FC is data-driven reasoning: start with what we know and build forwards

Exercises 7.4, 7.5, 7.6