Last update: October 11, 2012

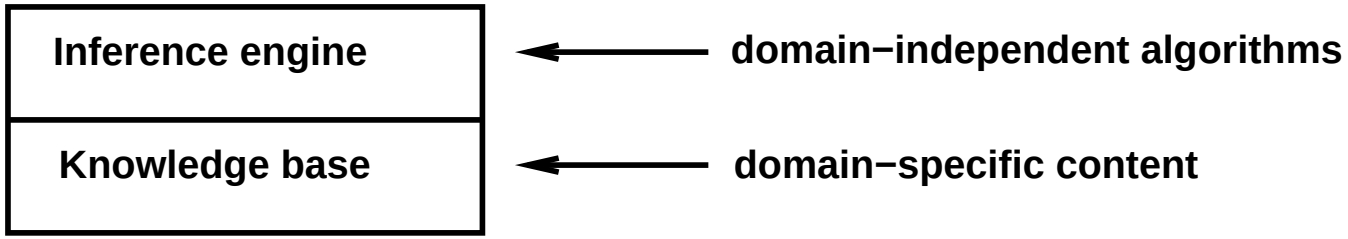
Logical agents

CMSC 421: 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

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
```

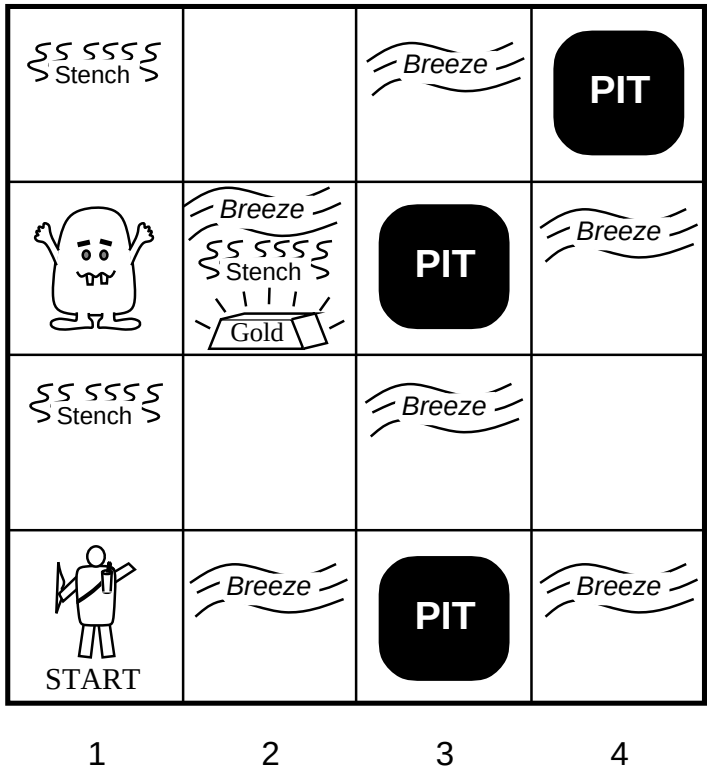
♢ The agent must be able to:

- 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

♢ **E**nvironment:

- One wumpus, one heap of gold
- $P(\text{pit}) = 0.2$ for each square
- Squares next to wumpus are smelly
- Shooting into wumpus's square kills it
- Shooting uses up the only arrow
- Squares next to pit are breezy
- Glitter iff the gold is in your square
  - ◇ Grabbing picks it up
  - ◇ Releasing drops it



♢ **P**erformance measure:

- gold +1000, death −1000, −1 per step, −10 for using the arrow

♢ **A**ctuators: Left turn, Right turn, Forward, Grab, Release, Shoot

♢ **S**ensors: Breeze, Glitter, Smell

Wumpus world characterization

- *Fully observable?*

Wumpus world characterization

- *Fully observable?* No—only local perception
- *Deterministic?*

Wumpus world characterization

- *Fully observable?* No—only local perception
- *Deterministic?* Yes—outcomes exactly specified
- *Episodic?*

Wumpus world characterization

- *Fully observable?* No—only local perception
- *Deterministic?* Yes—outcomes exactly specified
- *Episodic?* No—sequential at the level of actions
- *Static?*

Wumpus world characterization

- *Fully observable?* No—only local perception
- *Deterministic?* Yes—outcomes exactly specified
- *Episodic?* No—sequential at the level of actions
- *Static?* Yes—Wumpus, pits, and gold do not move
- *Discrete?*

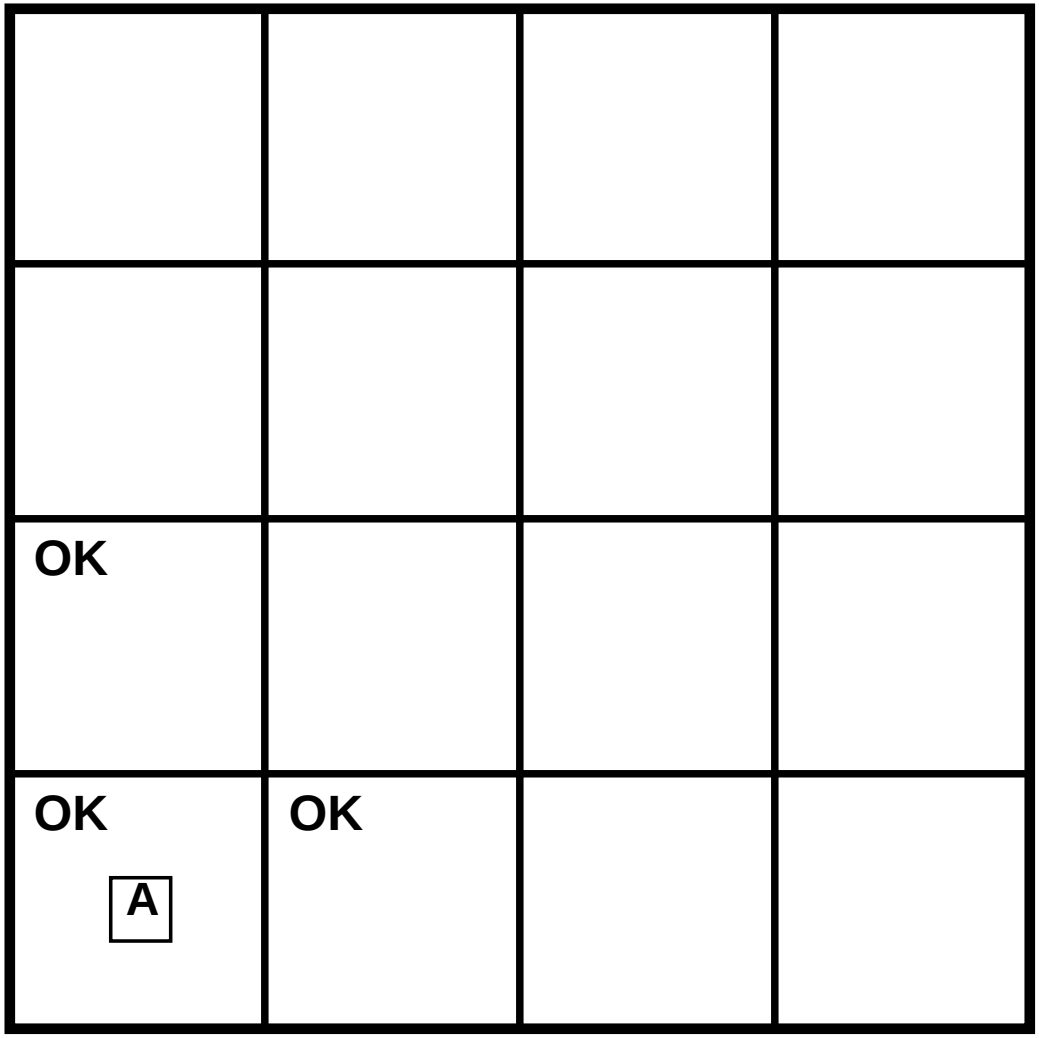
Wumpus world characterization

- *Fully observable?* No—only local perception
- *Deterministic?* Yes—outcomes exactly specified
- *Episodic?* No—sequential at the level of actions
- *Static?* Yes—Wumpus, pits, and gold do not move
- *Discrete?* Yes
- *Single-agent?*

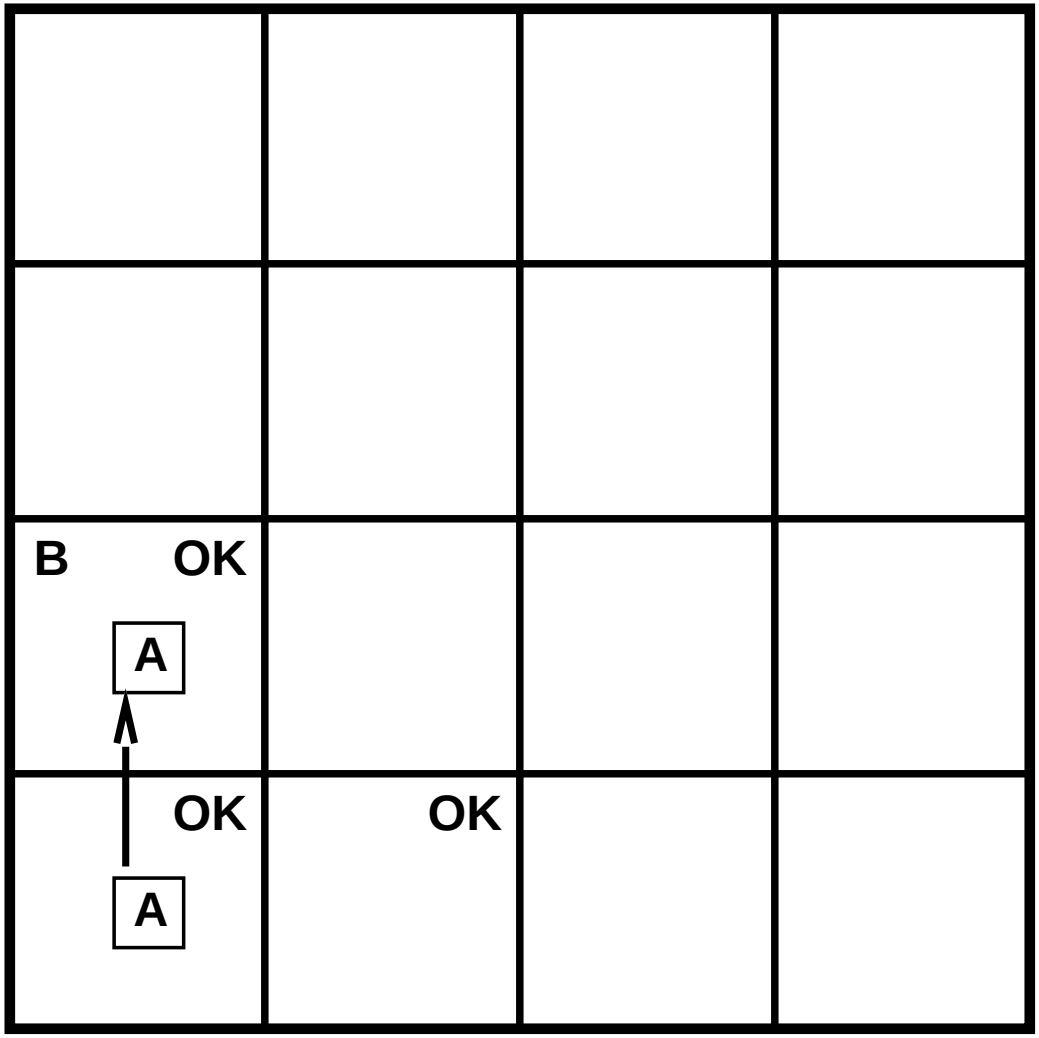
Wumpus world characterization

- *Fully observable?* No—only local perception
- *Deterministic?* Yes—outcomes exactly specified
- *Episodic?* No—sequential at the level of actions
- *Static?* Yes—Wumpus, pits, and gold do not move
- *Discrete?* Yes
- *Single-agent?* Yes—Wumpus is essentially a natural feature

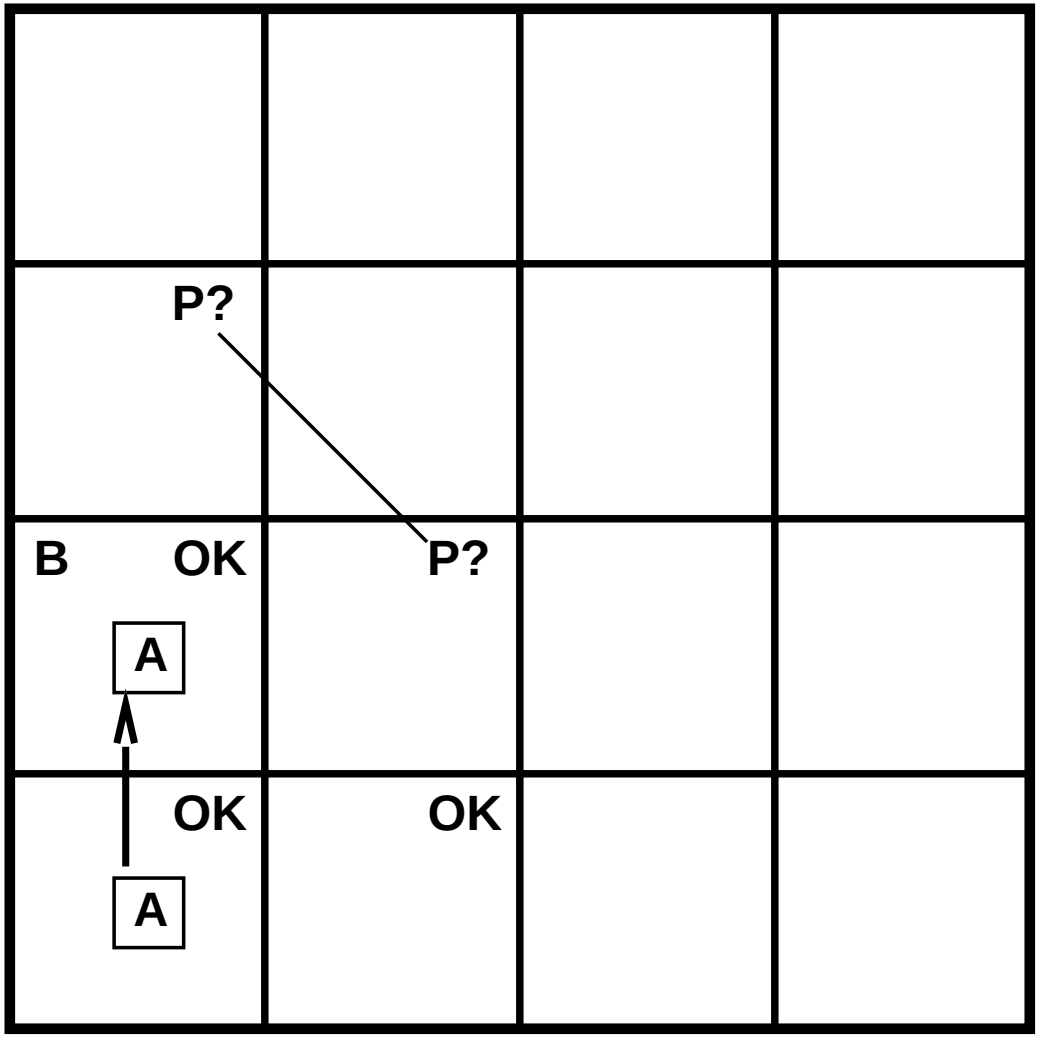
Exploring a wumpus world



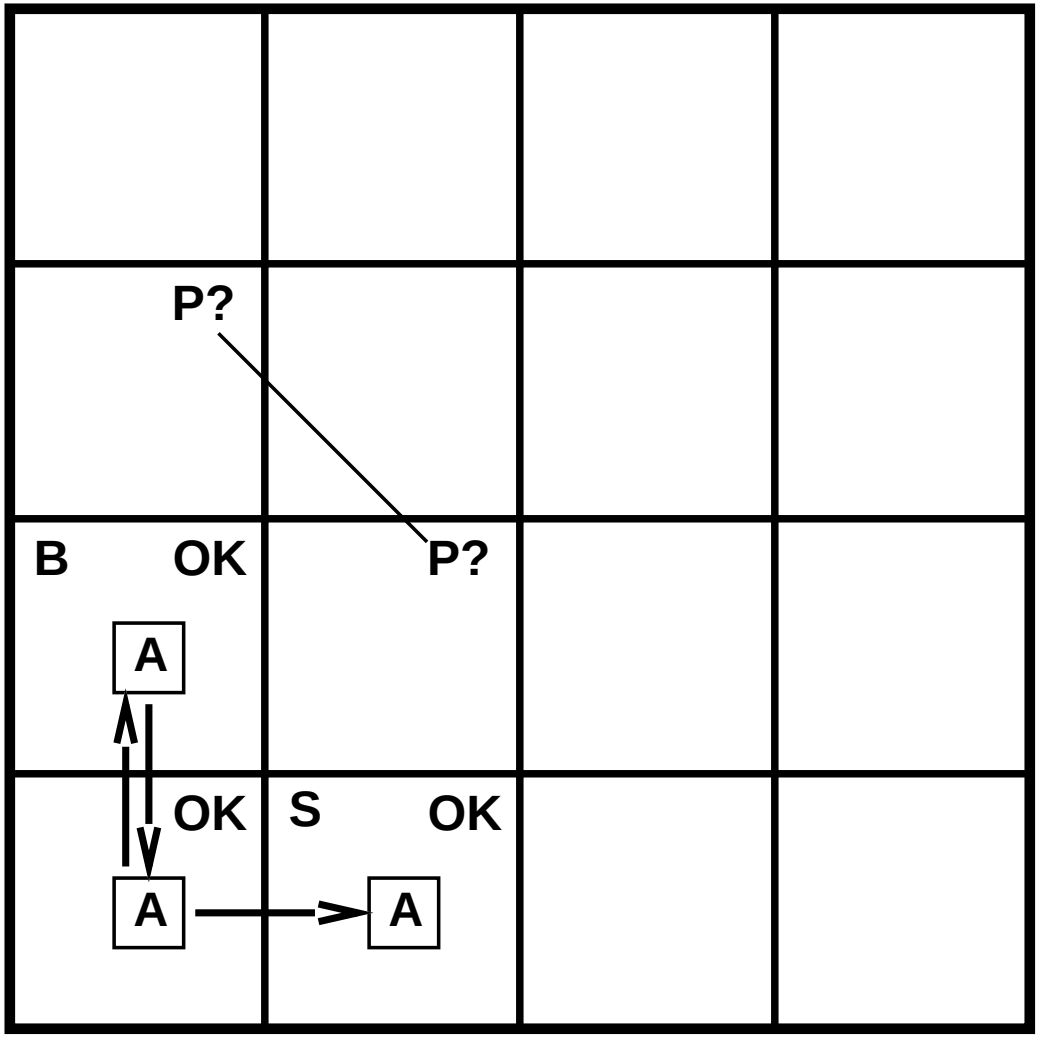
Exploring a wumpus world



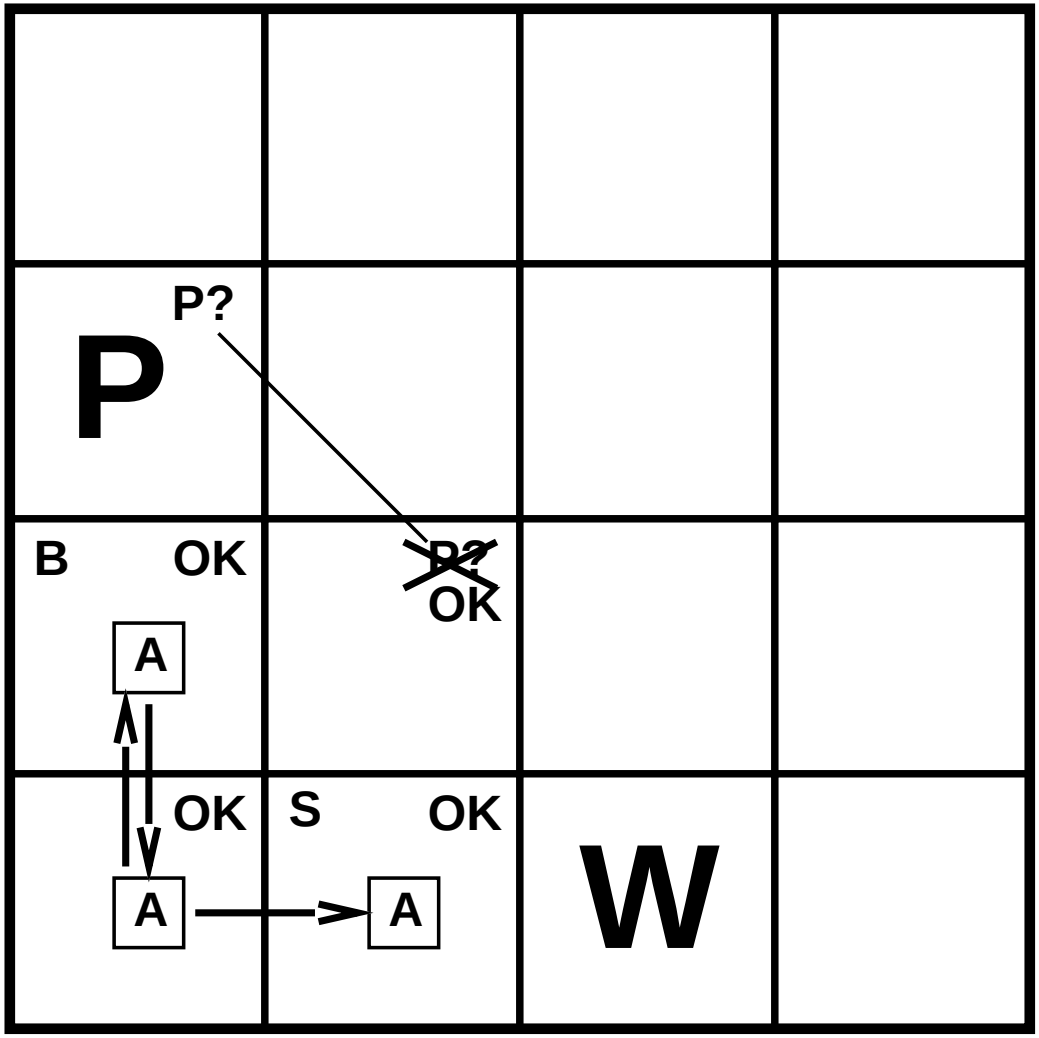
Exploring a wumpus world



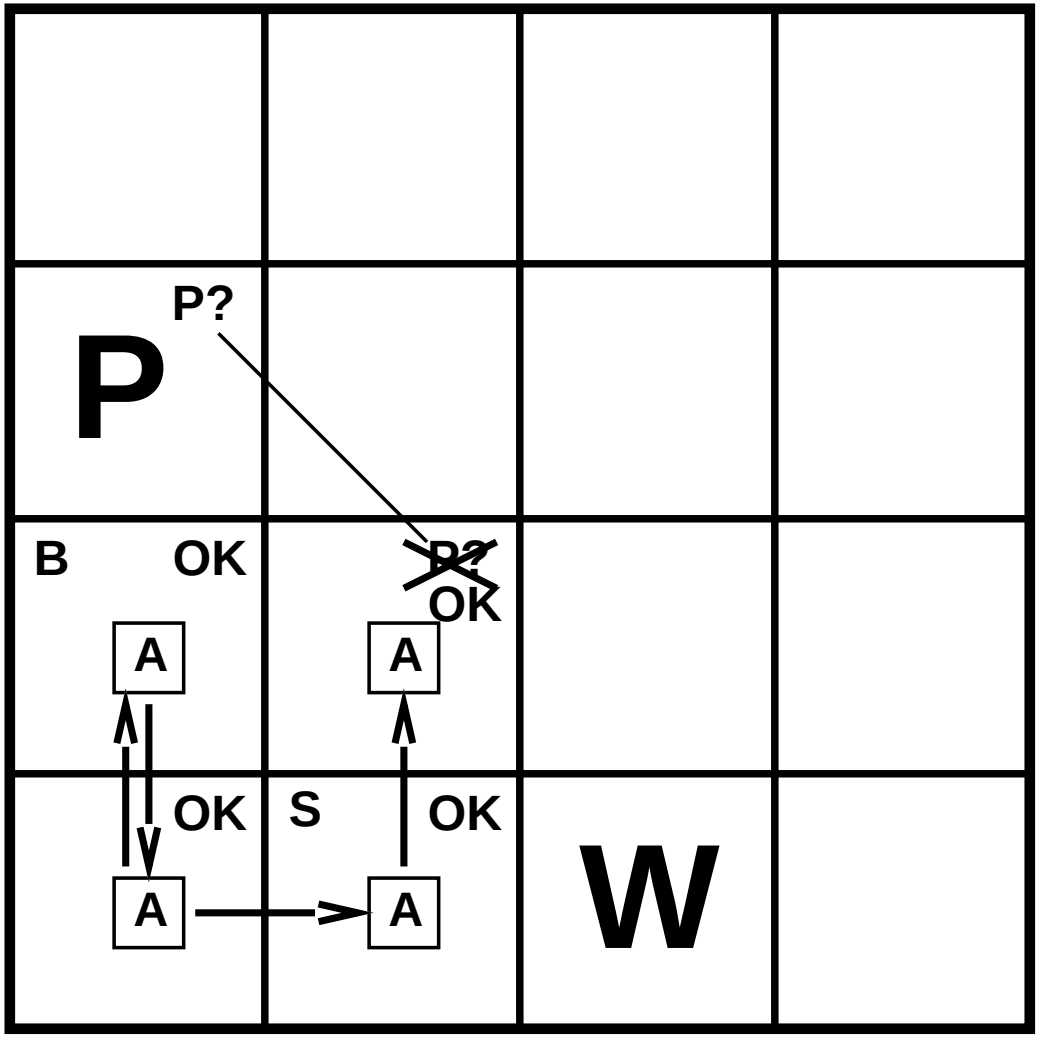
Exploring a wumpus world



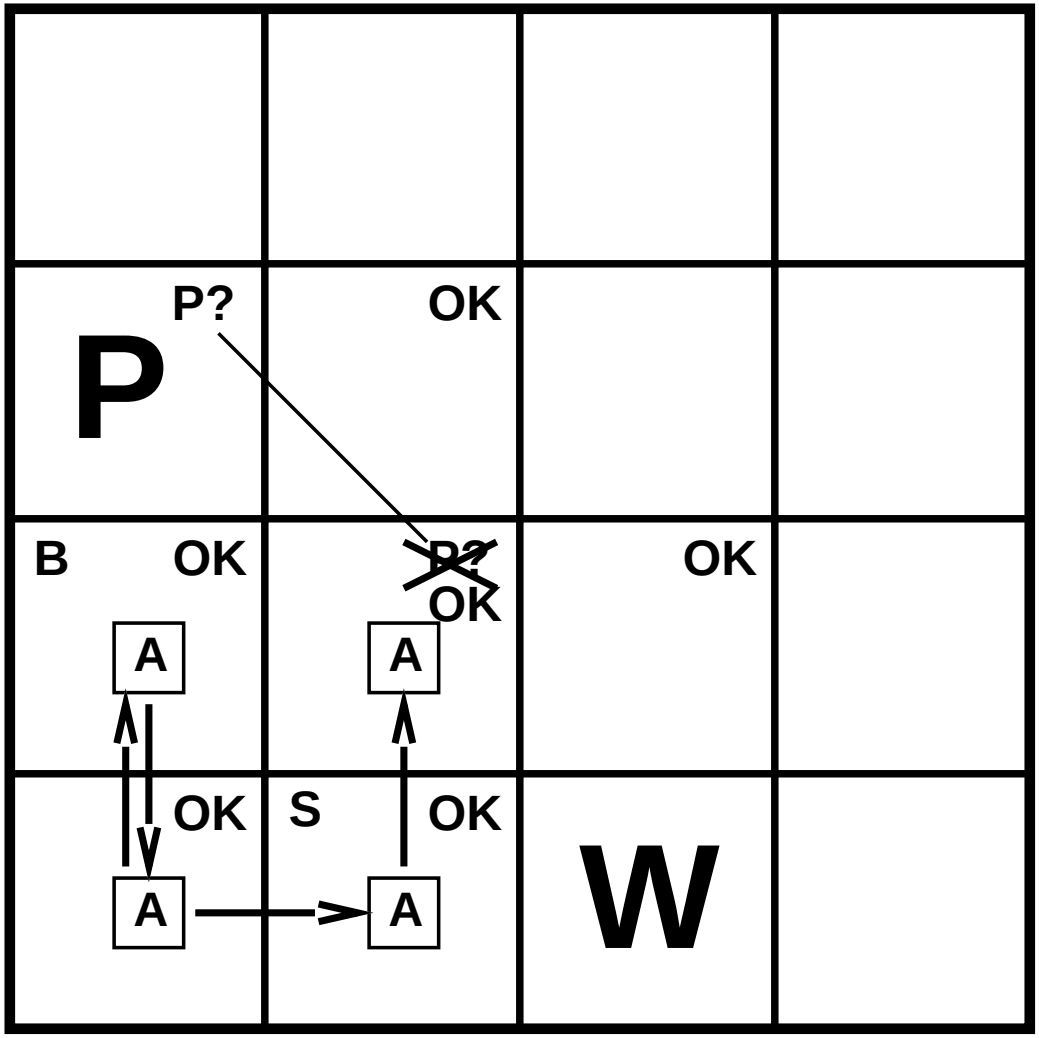
Exploring a wumpus world



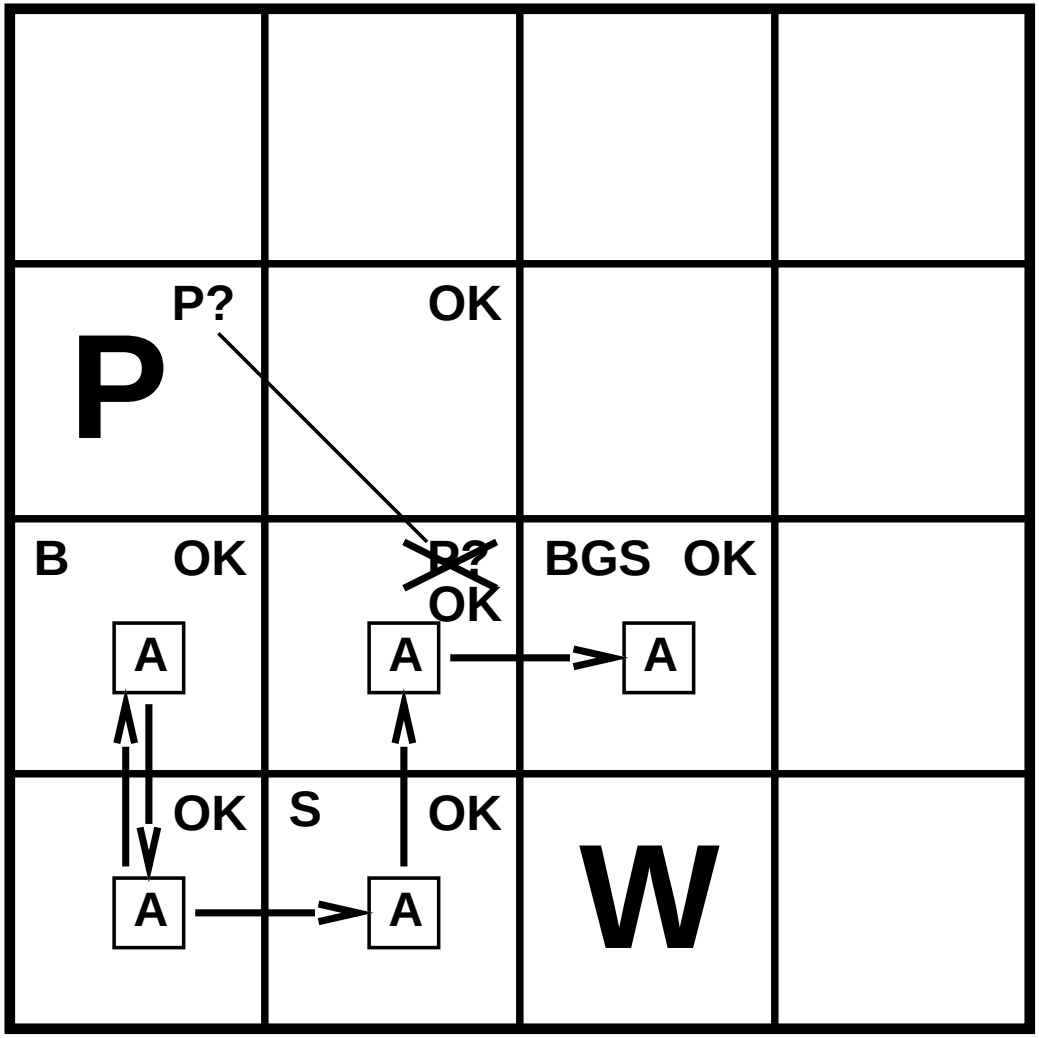
Exploring a wumpus world



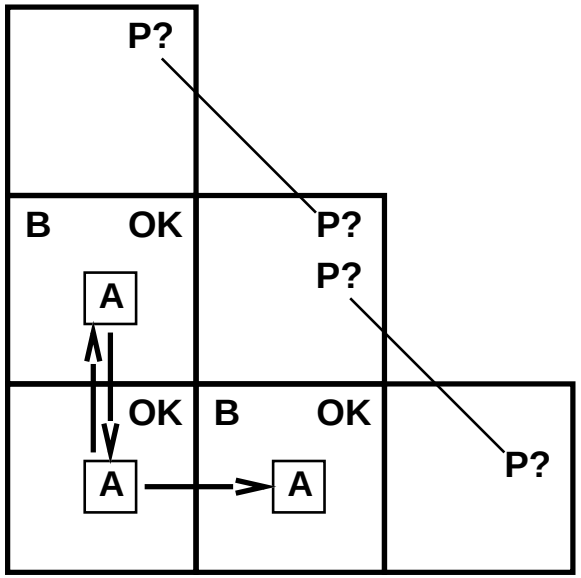
Exploring a wumpus world



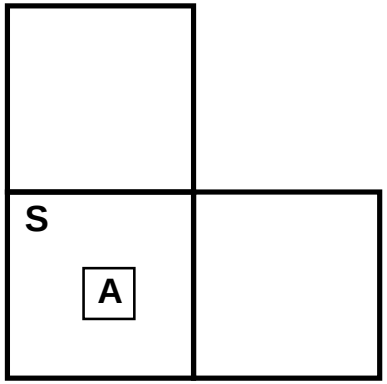
Exploring a wumpus world



Other tight spots



- ♢ Breeze in (1,2) and (2,1)
  ⇒ no safe actions
  - Pr[pit in $(2,2)$] ≈ 0.86
  - Pr[pits in $(1,3)$ and $(3,1)$) ≈ 0.31
- ♢ A later chapter will discuss ways for an agent to infer this



- ♢ Smell in (1,1) ⇒ can't move safely
- ♢ Can use a *coercion* strategy: shoot straight ahead
  - wumpus was there ⇒ dead ⇒ safe
  - wumpus wasn't there ⇒ safe

Logic in general

♢ *Logics* are formal languages for representing information from which conclusions can be inferred

- *Syntax* defines the sentences in the language
- *Semantics* define the "meaning" of sentences
  - ◇ what a sentence means will determine whether it's true or false

♢ 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 at least as big as 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

- A relationship between sentences (i.e., **syntax**) that is based on **semantics**

♢ Knowledge base KB entails sentence α

  ◇ written $KB \models \alpha$

- iff α is true in every world where KB is true

♢ Examples:

- $x + y = 4$ entails $4 = x + y$
- KB: Maryland beat UNC and Duke beat UNC

  ◇ $KB \models$ Maryland beat UNC or Duke beat UNC

inclusive or

Models

♢ Look at *possible worlds*:

- Formally structured worlds in which truth can be evaluated
- A possible world gives you a meaning for each sentence, so you can evaluate whether it is true or false
- A possible world m is a *model* of a sentence α if α is true in m

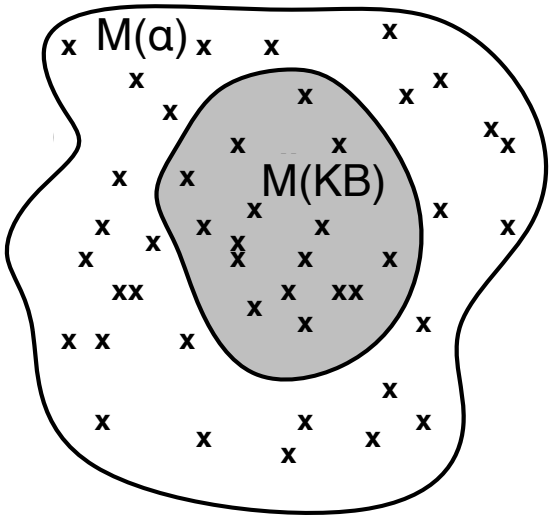
♢ $M(\alpha)$ is the set of all models of α

  ◇ All worlds in which α is true

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

♢ Example

- KB: Maryland beat UNC and Duke beat UNC
- α: Maryland beat UNC

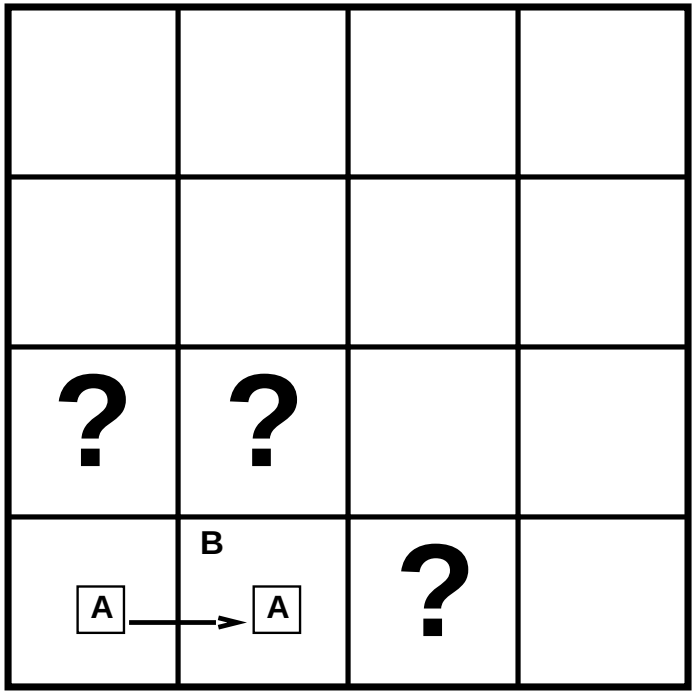


Entailment in the wumpus world

♢ Suppose you
- detect nothing in [1,1]
- move right
- feel a breeze in [2,1]

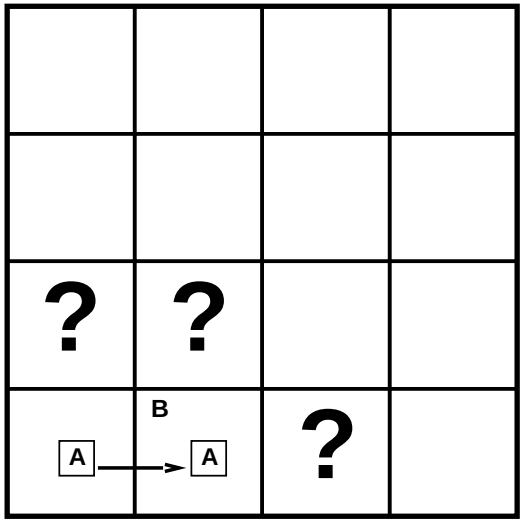
♢ For now,
- Ignore the wumpus and gold
- Ignore all squares other than the **?**s

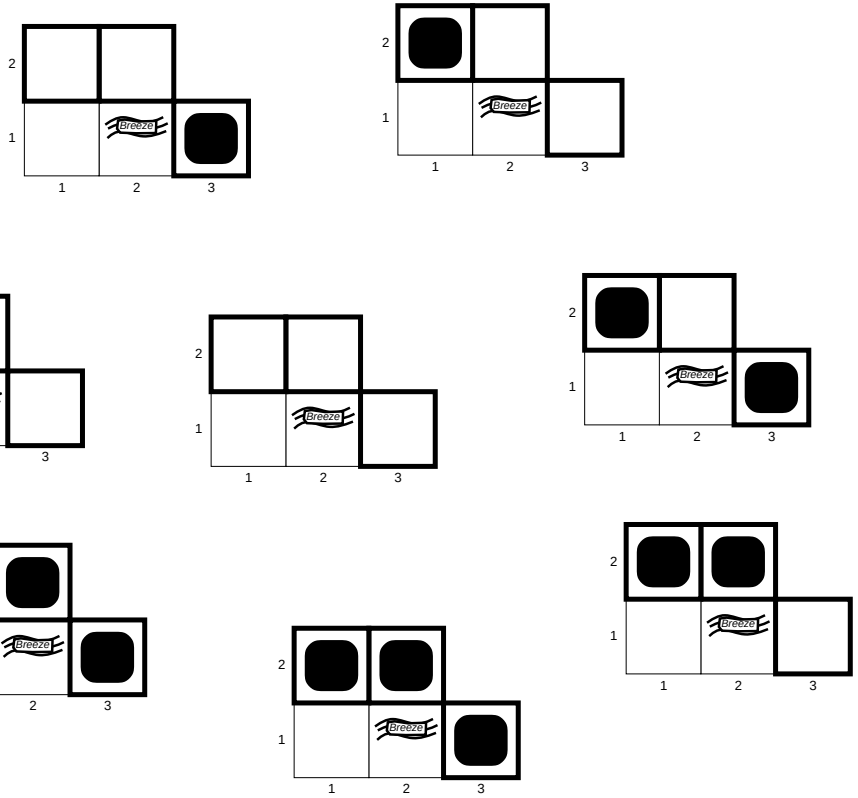
♢ Which **?**s are pits?



♢ **Model checking:**
- 3 Boolean choices $\Rightarrow$ 8 possible worlds
- For each one, check whether it's a model

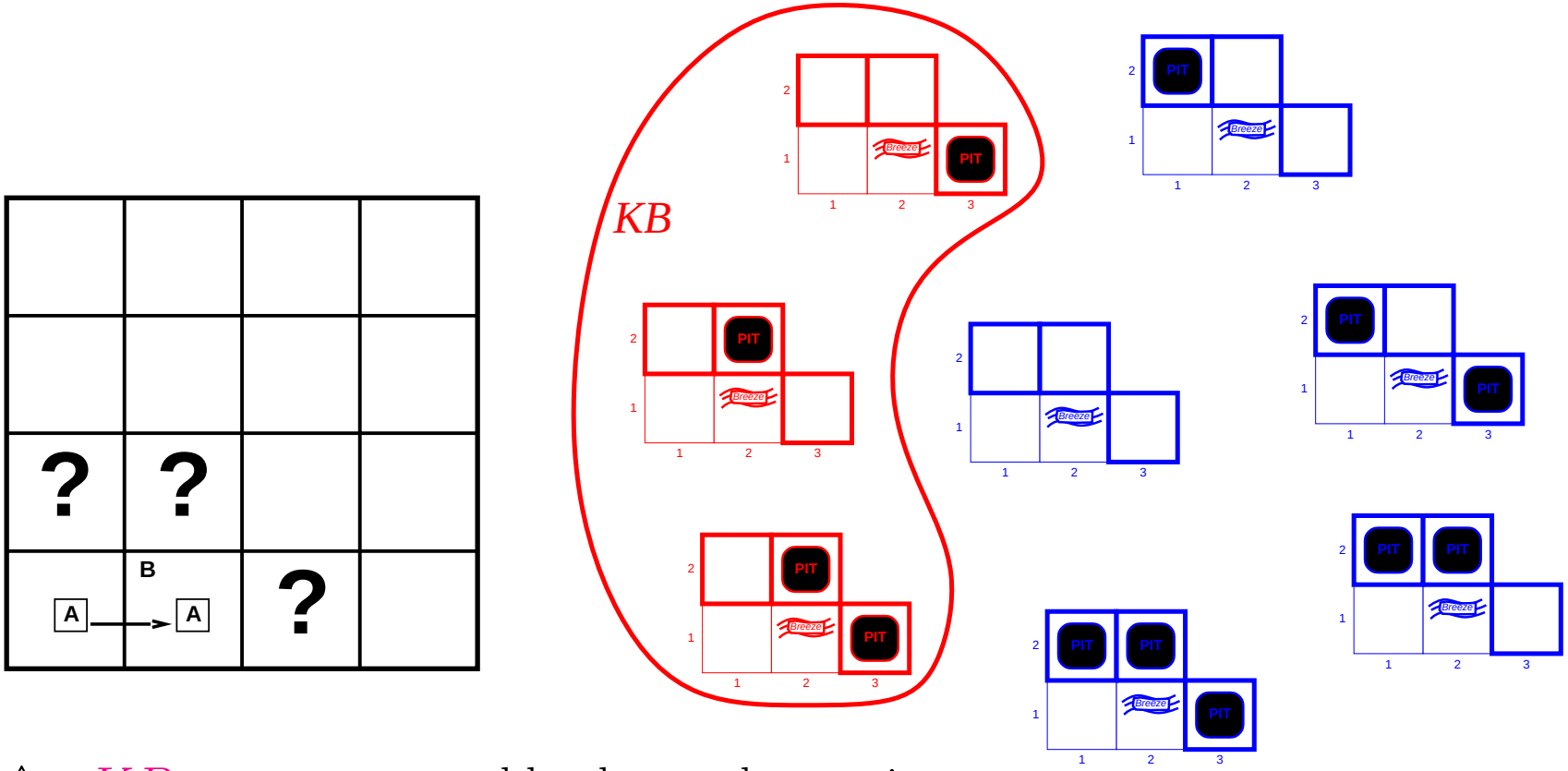
Wumpus models





- KB = wumpus-world rules + observations
  - Eight possible combinations of pit locations
  - Which ones are models of KB?

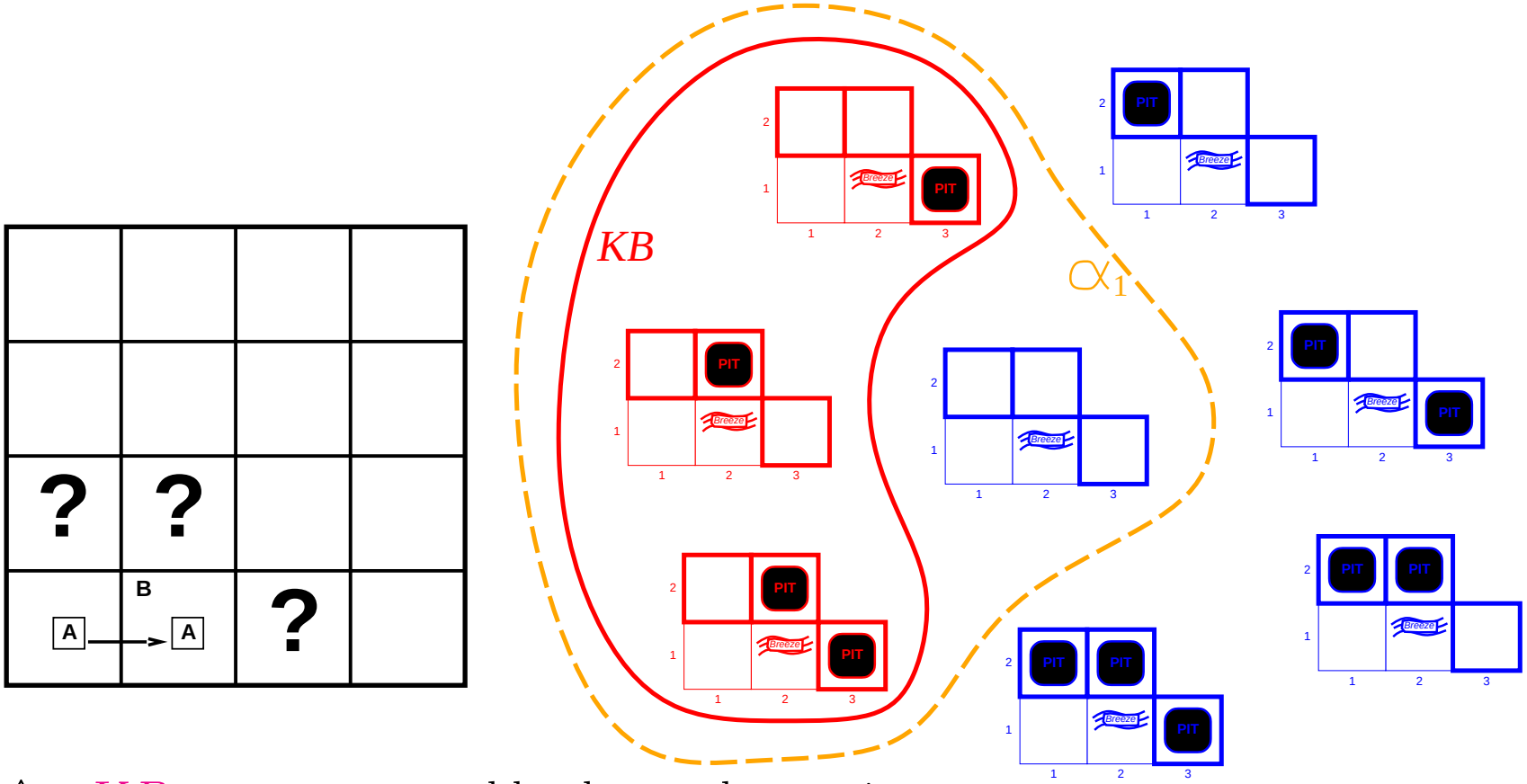
Wumpus models



♢ KB = wumpus-world rules + observations

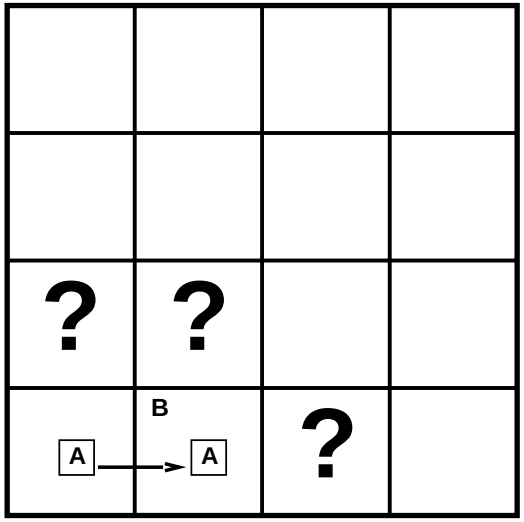
- Three models of KB

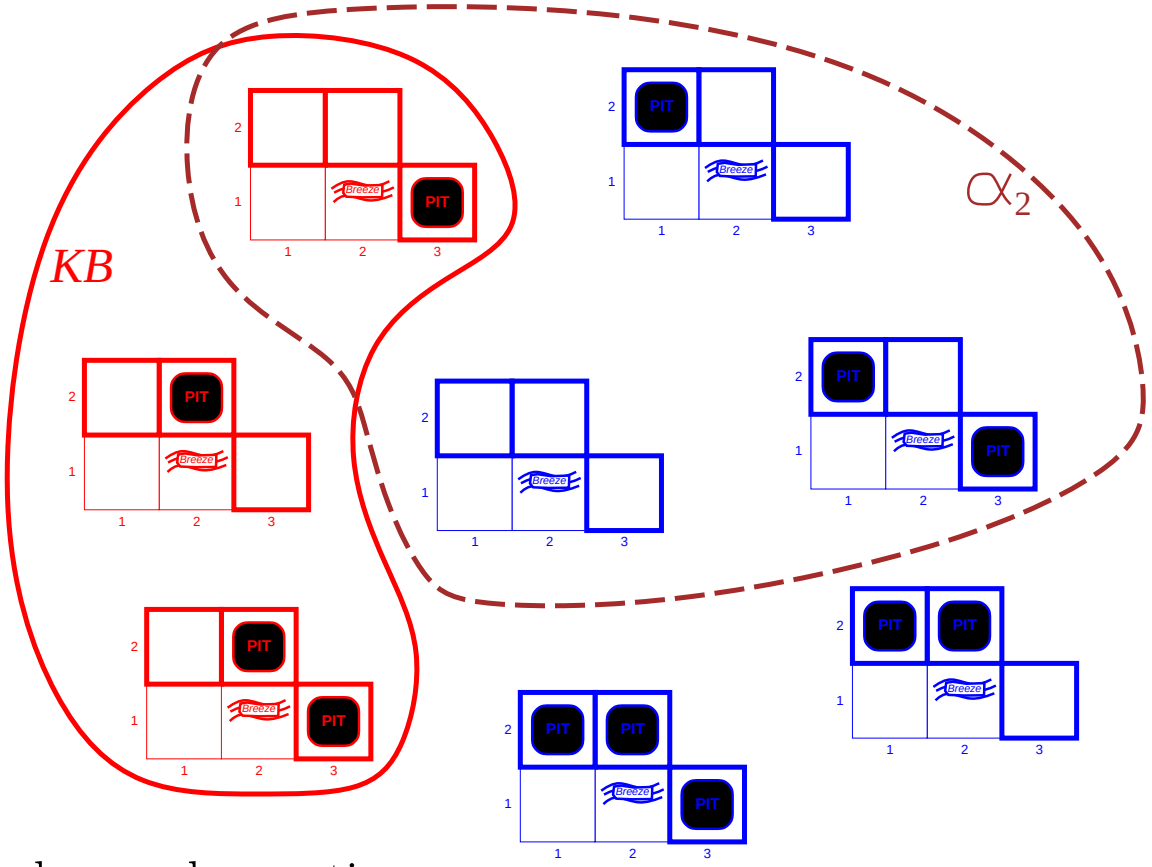
Wumpus models



- KB = wumpus-world rules + observations
  - Three models of KB
- Let α_1 = "[1,2] is safe"
  - Then $KB \models \alpha_1$, proved by model checking

Wumpus models





♢ KB = wumpus-world rules + observations

- Three models of KB

♢ Let α_2 = "[2,2] is safe"

- Then $KB \not\models \alpha_2$

Inference

♢ Entailment (e.g, $KB \models \alpha$) is like looking for a needle in haystack

- Consequences of KB are the haystack α is the needle
- Inference procedure = a procedure for finding α

♢ $KB \vdash_i \alpha$ means inference procedure i can derive sentence α from KB

- *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 $KB \vdash_i \alpha$

♢ Model checking is one kind of inference procedure (not the only one)

- It is sound
- It is complete if the set of possible worlds is finite

Preview of where we're going

- ♢ Later, we'll talk about first-order logic
  - Expressive enough to say almost anything of interest
- ♢ There are sound and complete inference procedures for first-order logic
  - Will answer any question whose answer follows from what's in the KB
- ♢ But first, let's look at propositional logic
  - The simplest logic; illustrates basic ideas

Propositional logic: Syntax

- ♢ 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 (*equivalence*)

Propositional logic: Semantics

♢ Each world specifies a true/false value for every proposition symbol

♢ Rules for evaluating truth with respect to a world w:

$\neg S$ 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	

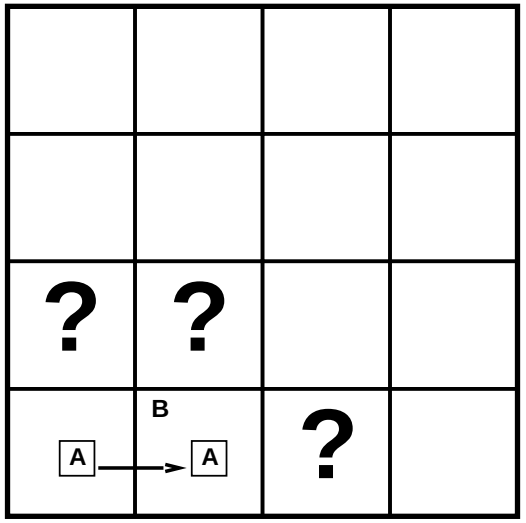
♢ Simple recursive process to evaluate an arbitrary sentence

- Consider a world in which $P_{1,2}$ is $true$
  $P_{2,2}$ is $true$
  $P_{3,1}$ is $false$

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

Wumpus world sentences

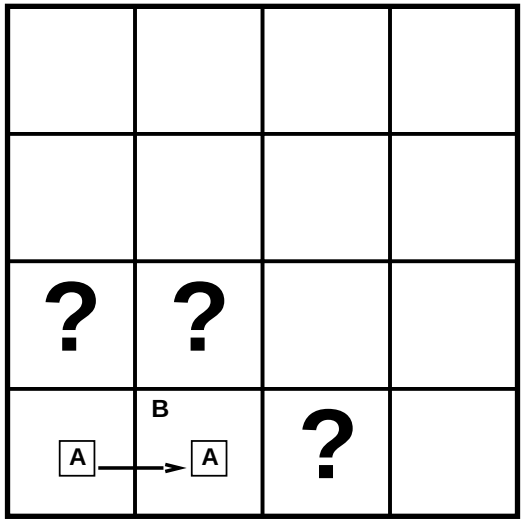
- Let $B_{i,j}$ be true if there is a breeze in $[i, j]$
- Let $P_{i,j}$ be true if there is a pit in $[i, j]$
- KB: $\neg B_{1,1}$, $B_{2,1}$, $\neg P_{1,1}$, $\neg P_{2,1}$
- How to represent "Pits cause breezes in adjacent squares"?



Wumpus world sentences

- Let $B_{i,j}$ be true if there is a breeze in $[i, j]$
- Let $P_{i,j}$ be true if there is a pit in $[i, j]$
- KB: $\neg B_{1,1}, \quad B_{2,1}, \quad \neg P_{1,1}, \quad \neg P_{2,1}$
- How to represent "Pits cause breezes in adjacent squares"?
  - "A square is breezy **if and only if** there is an adjacent pit"
- In propositional logic, we can't write this directly
  - But we can write instantiations of it
- In the example:

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



Truth tables for inference

♢ Model checking in propositional logic = inference using truth tables

♢ Each row is a possible world

- In each row where KB is true, is α true too?

♢ KB: $\neg B_{1,1}$, $B_{2,1}$, $\neg P_{1,1}$, $\neg P_{2,1}$

$B_{1,1}$	$B_{2,1}$	$P_{1,1}$	$P_{1,2}$	$P_{2,1}$	$P_{2,2}$	$P_{3,1}$	KB	α_1
false	*false*	*false*	*false*	*false*	*false*	*false*	*false*	don't care
false	*false*	*false*	*false*	*false*	*false*	*true*	*false*	don't care
⋮	⋮	⋮	⋮	⋮	⋮	⋮	⋮	⋮
false	*true*	*false*	*false*	*false*	*false*	*false*	*false*	don't care
false	*true*	*false*	*false*	*false*	*false*	*true*	*true*	*true*
false	*true*	*false*	*false*	*false*	*true*	*false*	*true*	*true*
false	*true*	*false*	*false*	*false*	*true*	*true*	*true*	*true*
false	*true*	*false*	*false*	*true*	*false*	*false*	*false*	don't care
⋮	⋮	⋮	⋮	⋮	⋮	⋮	⋮	⋮
true	*true*	*true*	*true*	*true*	*true*	*true*	*false*	don't care

Inference by enumeration

♢ Depth-first enumeration of all models is sound and complete

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

function TT-ENTAILS?(KB, α)

  $symbols \leftarrow$ a list of the proposition symbols in KB and α

  return TT-CHECK-ALL($KB, \alpha, symbols, [\,]$)

---

function TT-CHECK-ALL($KB, \alpha, symbols, world$)

  if $symbols$ is empty **then**

    if KB is true in $world$ **then return** α's truth value in $world$

    else return $true$

  else

    $P \leftarrow$ first symbol in $symbols$

    $rest \leftarrow$ the other symbols in $symbols$

    return TT-CHECK-ALL($KB, \alpha, rest,$ EXTEND($P, true, world$)) **and**

      TT-CHECK-ALL($KB, \alpha, rest,$ EXTEND($P, false, world$))

Proof methods

♢ Proof methods divide into (roughly) two kinds:

- 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., hill-climbing with min-conflicts

- Application of inference rules
  - ◇ Legitimate (sound) generation of new sentences from old
  - ◇ *Proof*: a sequence of applications of inference rules
    - Use a search algorithm with inference rules as operators
    - Typically requires translation of sentences into a *normal form*

Logical equivalence

♢ Two sentences are *logically equivalent* iff true in same models:

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

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

Validity and satisfiability

- A sentence is *valid* if it is true in **all** possible worlds,
  - 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 **at least one** possible world
  - e.g., $A \vee B$, C
- A sentence is *unsatisfiable* if it is true in **no** possible worlds
  - 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*

Forward and backward chaining

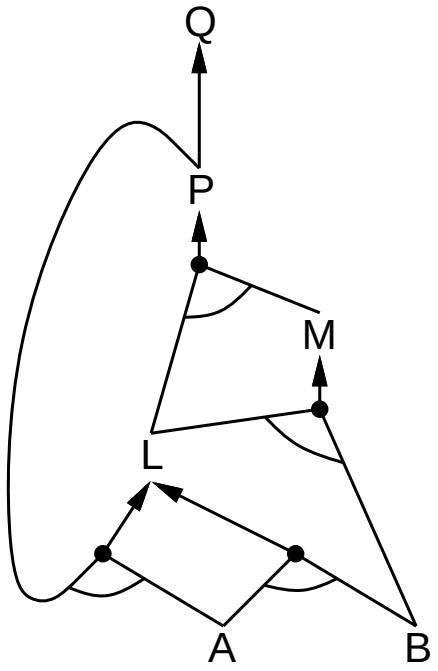
- A *Horn clause* is one of the following:
  - proposition symbol
  - conjunction of proposition symbols $\Rightarrow$ symbol
- *Horn form*: conjunct of Horn clauses, e.g.,
  - $C \wedge (B \Rightarrow A) \wedge (C \wedge D \Rightarrow B)$
  - Often just write the separate clauses:
    - $C, \quad B \Rightarrow A, \quad C \wedge D \Rightarrow B$
- This is a restricted subset of propositional logic
  - e.g., $A \vee B$ isn't in Horn form, and can't be translated into it
- *Modus Ponens* for Horn clauses: $\dfrac{\alpha_1, \ \ldots, \ \alpha_n, \qquad \alpha_1 \wedge \cdots \wedge \alpha_n \ \Rightarrow \ \beta}{\beta}$
  - Complete for KBs in Horn form
  - Can be used with *forward chaining* or *backward chaining*
    - These algorithms are very natural and run in **linear** time

Forward chaining

♢ Given a query q:

- loop until q is found:
  - ◇ apply any rule whose premises are satisfied in the KB
  - ◇ add its conclusion to the KB

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



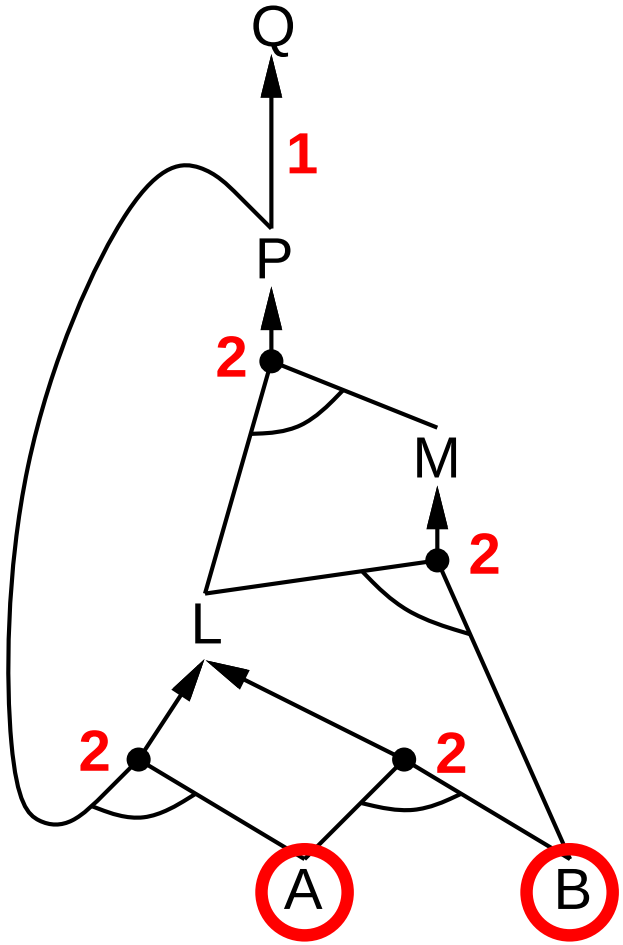
Forward chaining algorithm

```
function PL-FC-Entails?(KB, q)
    for every clause c in KB do
        count[c] ← number of premises of c
    for every proposition symbol p that appears anywhere in KB do
        inferred[p] ← false
    agenda ← all proposition symbols that are clauses in KB
    while agenda is not empty do
        p ← Pop(agenda)
        unless inferred[p] do
            inferred[p] ← true
            for each Horn clause c that has p in its premises 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

c	count$[c]$
$P \Rightarrow Q$	1
$L \wedge M \Rightarrow P$	2
$B \wedge L \Rightarrow M$	2
$A \wedge P \Rightarrow L$	2
$A \wedge B \Rightarrow L$	2
A	0
B	0

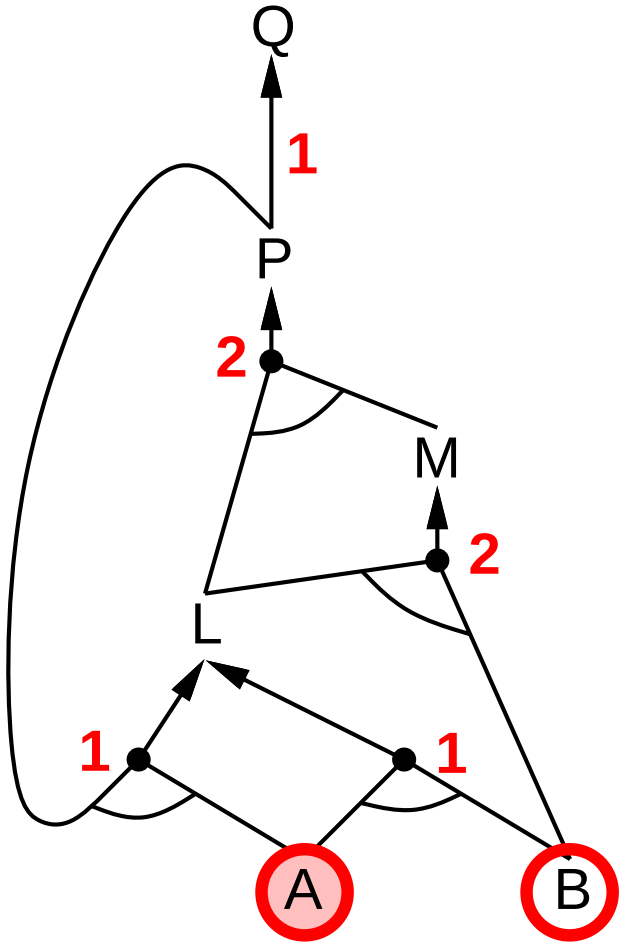
$agenda = \langle A, B \rangle$



Forward chaining example

c	count$[c]$
$P \Rightarrow Q$	1
$L \wedge M \Rightarrow P$	2
$B \wedge L \Rightarrow M$	2
$\mathbf{A} \wedge P \Rightarrow L$	1
$\mathbf{A} \wedge B \Rightarrow L$	1
A	0
B	0

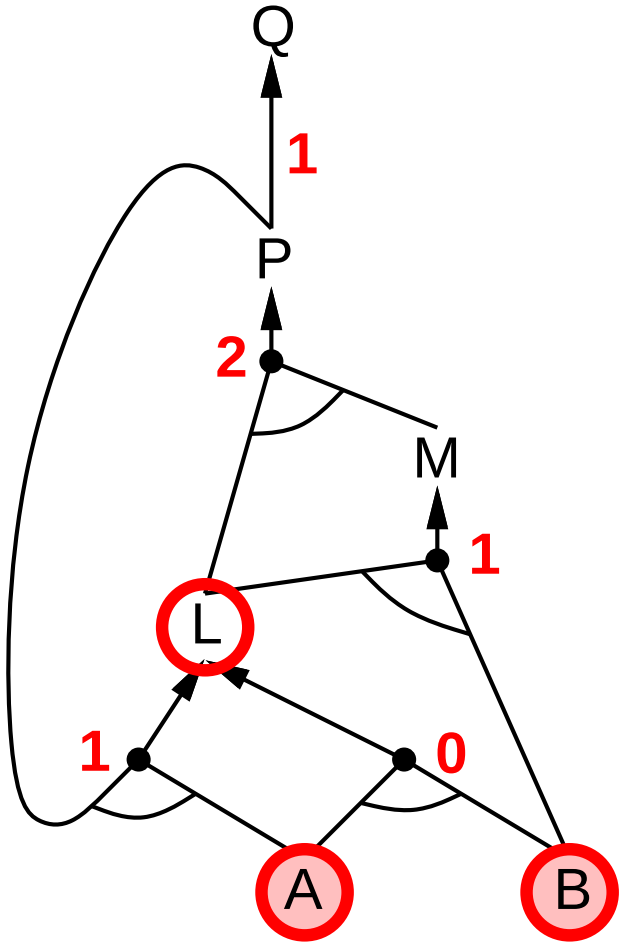
$agenda = \langle B \rangle$



Forward chaining example

c	count$[c]$
$P \Rightarrow Q$	1
$L \wedge M \Rightarrow P$	2
$\mathbf{B} \wedge L \Rightarrow M$	1
$A \wedge P \Rightarrow L$	1
$A \wedge \mathbf{B} \Rightarrow L$	0
A	0
B	0

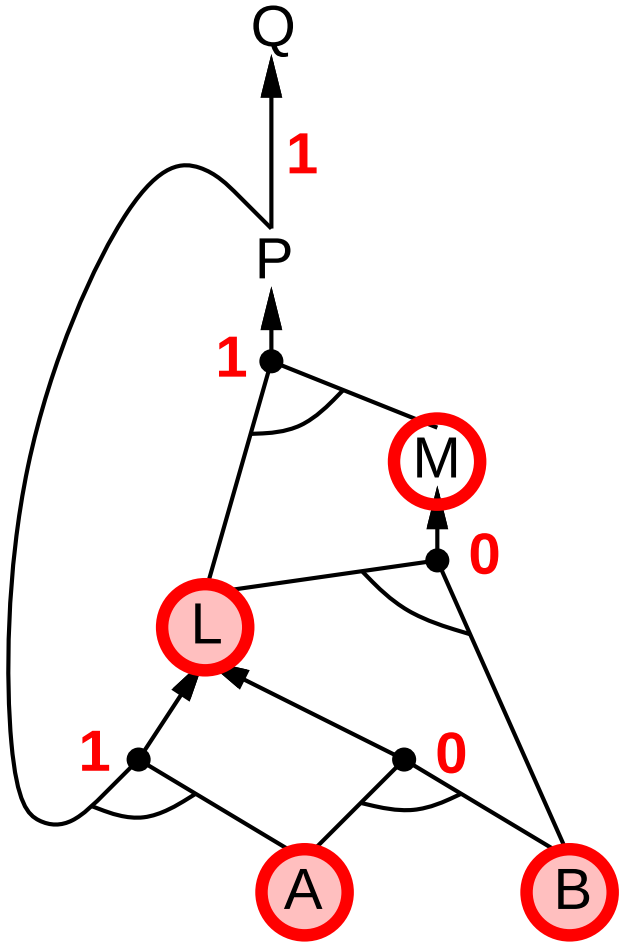
$agenda = \langle L \rangle$



Forward chaining example

c	count$[c]$
$P \Rightarrow Q$	1
$\mathbf{L} \wedge M \Rightarrow P$	1
$B \wedge \mathbf{L} \Rightarrow M$	0
$A \wedge P \Rightarrow L$	1
$A \wedge B \Rightarrow L$	0
A	0
B	0

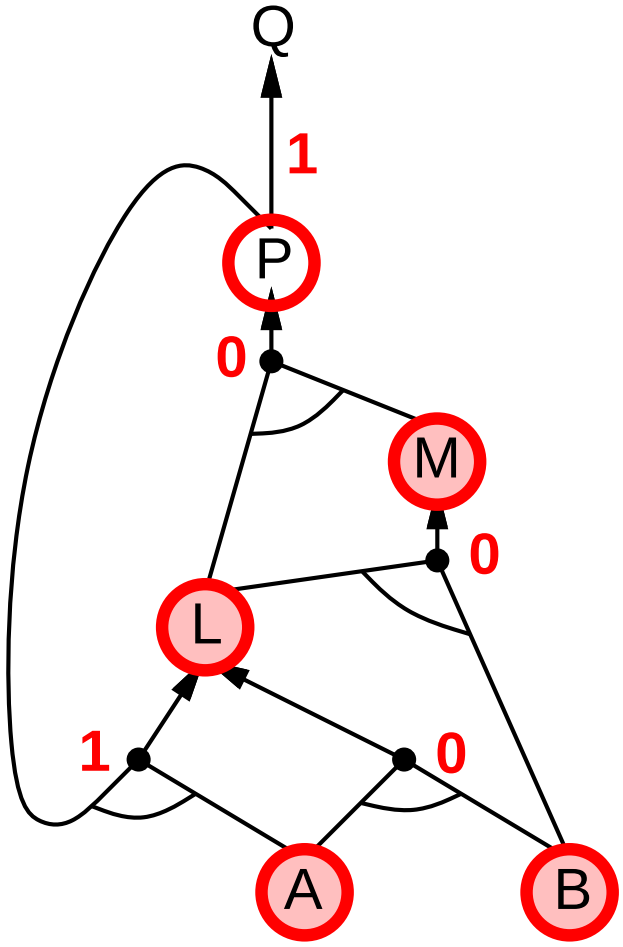
$agenda = \langle M \rangle$



Forward chaining example

c	count$[c]$
$P \Rightarrow Q$	1
$L \wedge \mathbf{M} \Rightarrow P$	0
$B \wedge L \Rightarrow M$	0
$A \wedge P \Rightarrow L$	1
$A \wedge B \Rightarrow L$	0
A	0
B	0

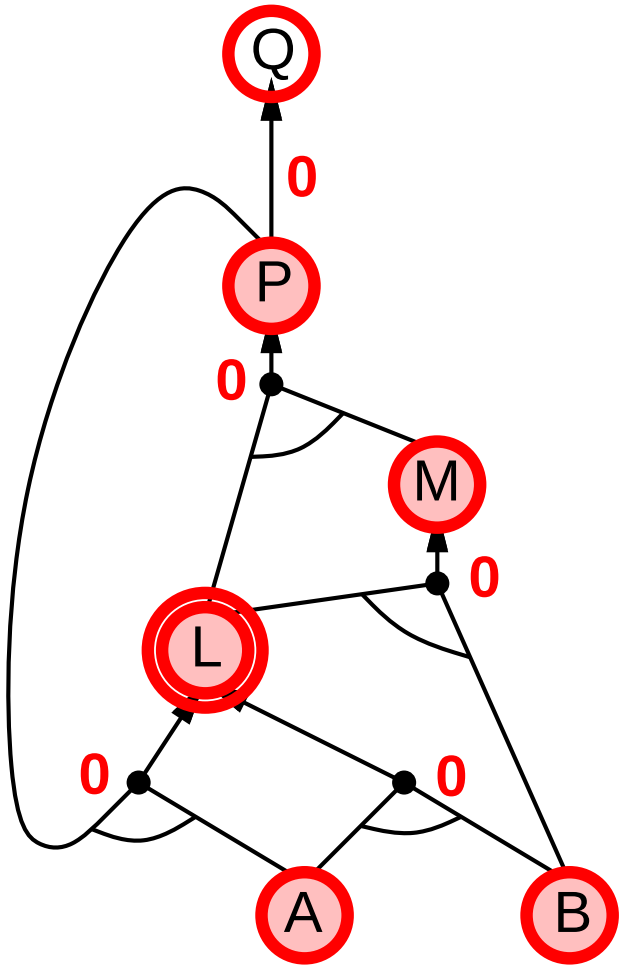
$agenda = \langle P \rangle$



Forward chaining example

c	count$[c]$
$\mathbf{P} \Rightarrow Q$	0
$L \wedge M \Rightarrow P$	0
$B \wedge L \Rightarrow M$	0
$A \wedge \mathbf{P} \Rightarrow L$	0
$A \wedge B \Rightarrow L$	0
A	0
B	0

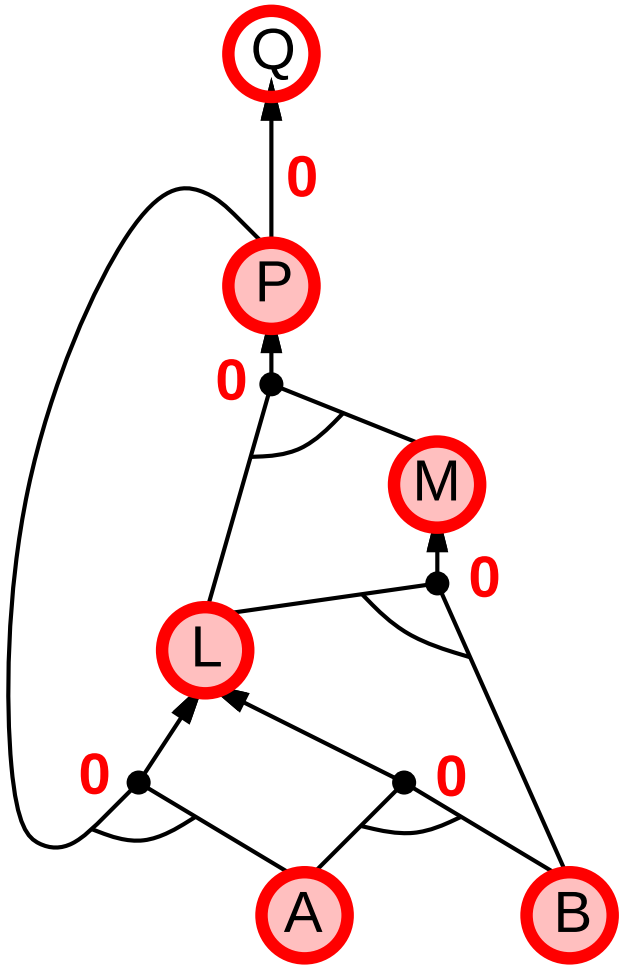
$agenda = \langle L, Q \rangle$



Forward chaining example

c	count$[c]$
$P \Rightarrow Q$	0
$L \wedge M \Rightarrow P$	0
$B \wedge L \Rightarrow M$	0
$A \wedge P \Rightarrow L$	0
$A \wedge B \Rightarrow L$	0
A	0
B	0

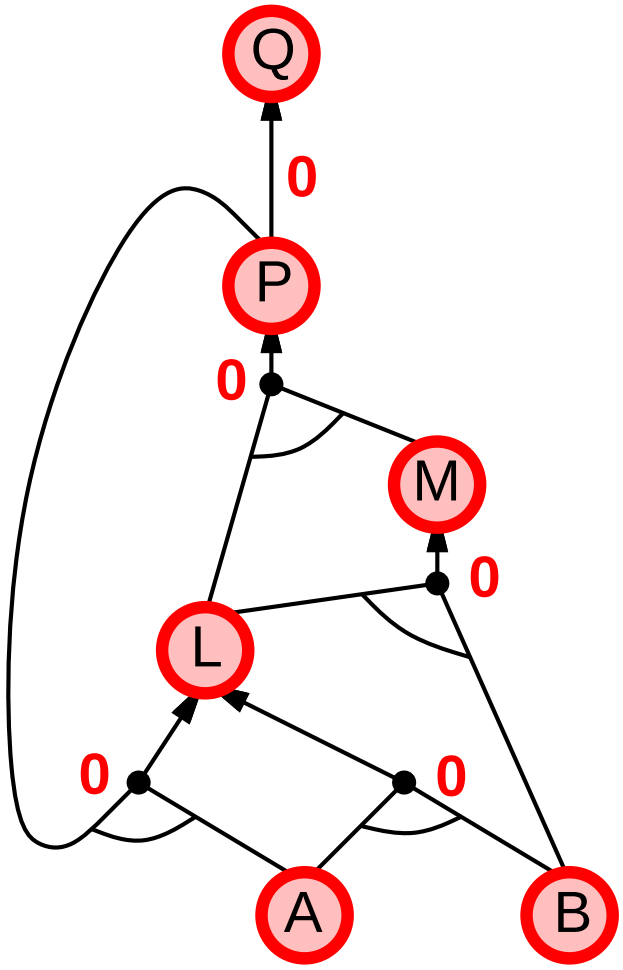
$agenda = \langle Q \rangle$



Forward chaining example

c	count$[c]$
$P \Rightarrow Q$	0
$L \wedge M \Rightarrow P$	0
$B \wedge L \Rightarrow M$	0
$A \wedge P \Rightarrow L$	0
$A \wedge B \Rightarrow L$	0
A	0
B	0

$agenda = \langle\rangle$



Proof of completeness

FC derives every atomic sentence that is entailed by KB

1. At i'th iteration of the "while" loop, can create a world m_i as follows:
   assign true to every atomic symbol that has been derived
   assign false to all other atomic symbols

2. There are only finitely many atomic sentences, so FC must reach a *fixed point* where no new atomic sentences are derived. Let n be the iteration where this happens.

3. Every clause in the original KB is true in m_n. Proof:
   - ◇ Suppose a clause $a_1 \wedge \ldots \wedge a_k \Rightarrow b$ is false in m_i.
   - ◇ Then $a_1 \wedge \ldots \wedge a_k$ is true in m_i and b is false in m_i.
   - ◇ Thus $\text{count}(b) = 0$, so b will be inferred in a future iteration, so iteration i isn't a fixed point.
   - Hence at the fixed point, m_n is a model of KB.

5. If q is atomic and $KB \models q$, then q is true in **every** model of KB, including m. Hence FC must have derived q.

Backward chaining

- Idea: work backwards from the query q
  - To prove q by BC, either
    - Check if q is known already, or
    - Recursively call BC to prove the premises of a rule that infers q
- To avoid loops: check if q is already on the recursion stack
- To avoid repeated work: check if q
  - has already been proved true, or
  - has already failed

Backward chaining example

$P \Rightarrow Q$

$L \wedge M \Rightarrow P$

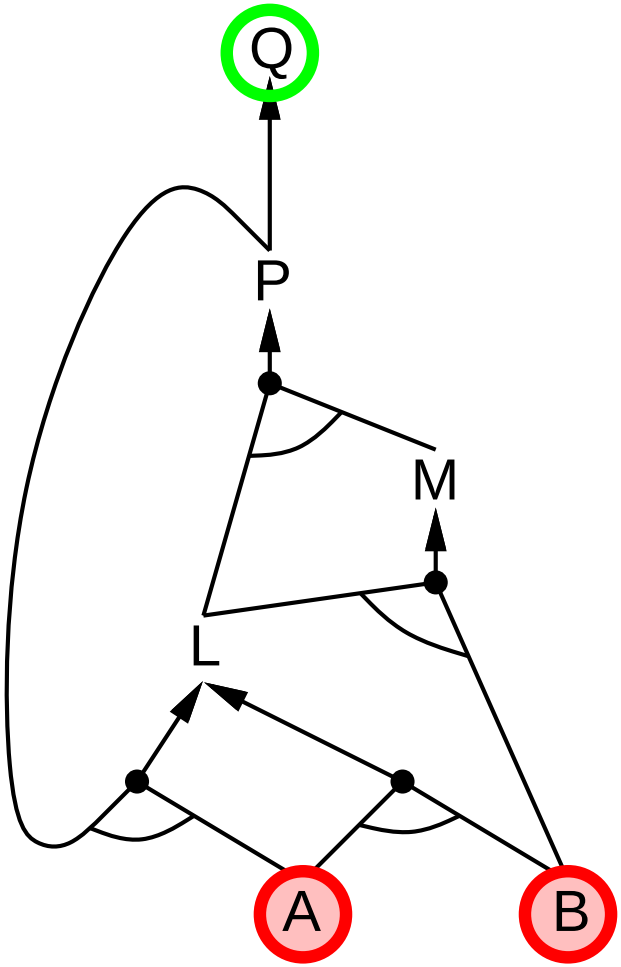
$B \wedge L \Rightarrow M$

$A \wedge P \Rightarrow L$

$A \wedge B \Rightarrow L$

A

B



Backward chaining example

$\mathbf{P} \Rightarrow \mathbf{Q}$

$L \wedge M \Rightarrow P$

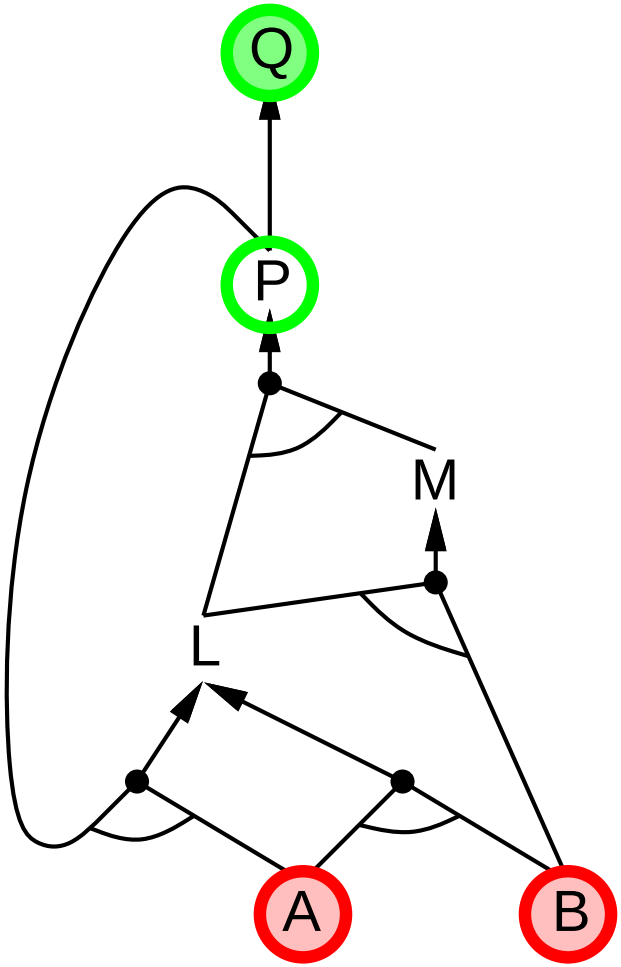
$B \wedge L \Rightarrow M$

$A \wedge P \Rightarrow L$

$A \wedge B \Rightarrow L$

A

B



Backward chaining example

$\mathbf{P} \Rightarrow \mathbf{Q}$

$\mathbf{L} \wedge \mathbf{M} \Rightarrow \mathbf{P}$

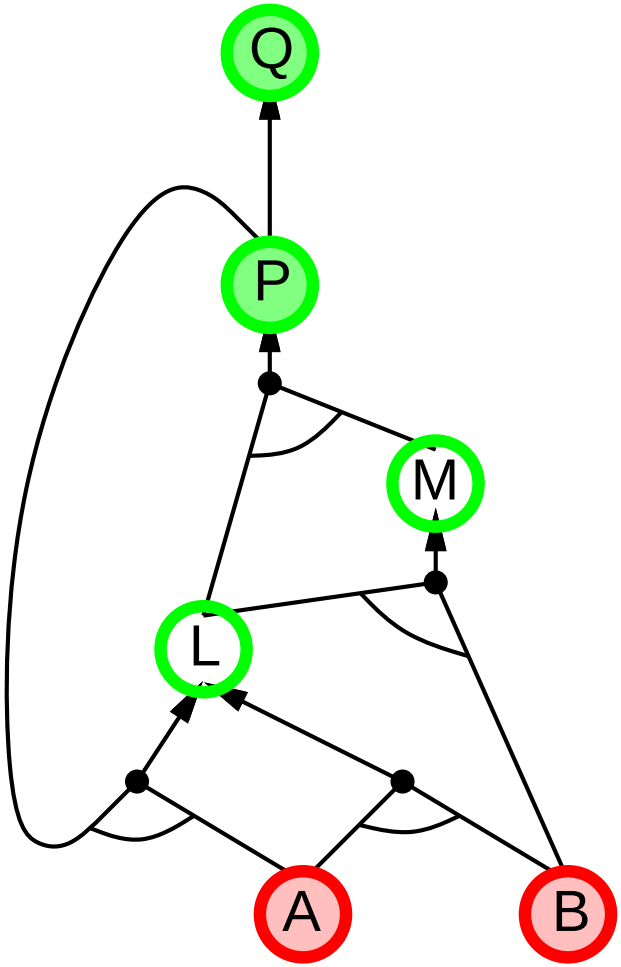
$B \wedge L \Rightarrow M$

$A \wedge P \Rightarrow L$

$A \wedge B \Rightarrow L$

A

B



Backward chaining example

$\mathbf{P} \Rightarrow \mathbf{Q}$

$\mathbf{L} \wedge \mathbf{M} \Rightarrow \mathbf{P}$

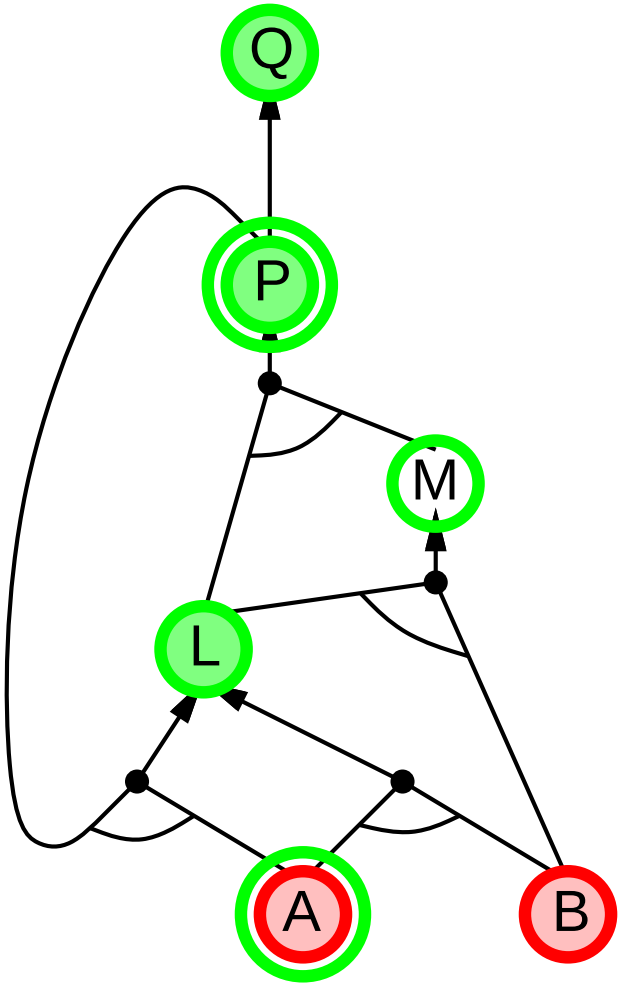
$B \wedge L \Rightarrow M$

$\mathbf{A} \wedge \mathbf{P} \Rightarrow \mathbf{L}$

$A \wedge B \Rightarrow L$

A

B



Backward chaining example

$\mathbf{P} \Rightarrow \mathbf{Q}$

$\mathbf{L} \wedge \mathbf{M} \Rightarrow \mathbf{P}$

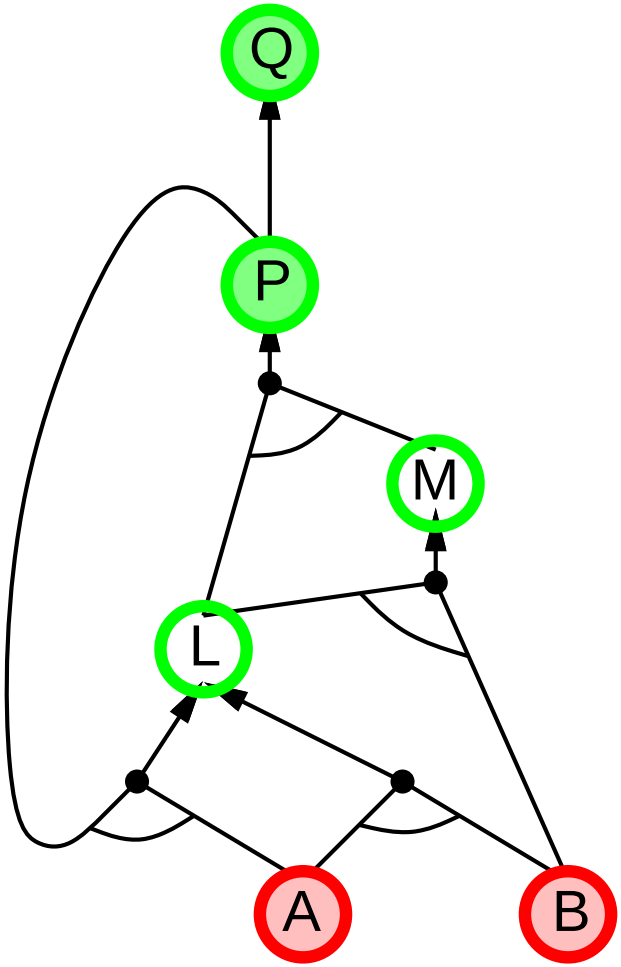
$B \wedge L \Rightarrow M$

$A \wedge P \Rightarrow L$

$A \wedge B \Rightarrow L$

A

B



Backward chaining example

$\mathbf{P} \Rightarrow \mathbf{Q}$

$\mathbf{L} \wedge \mathbf{M} \Rightarrow \mathbf{P}$

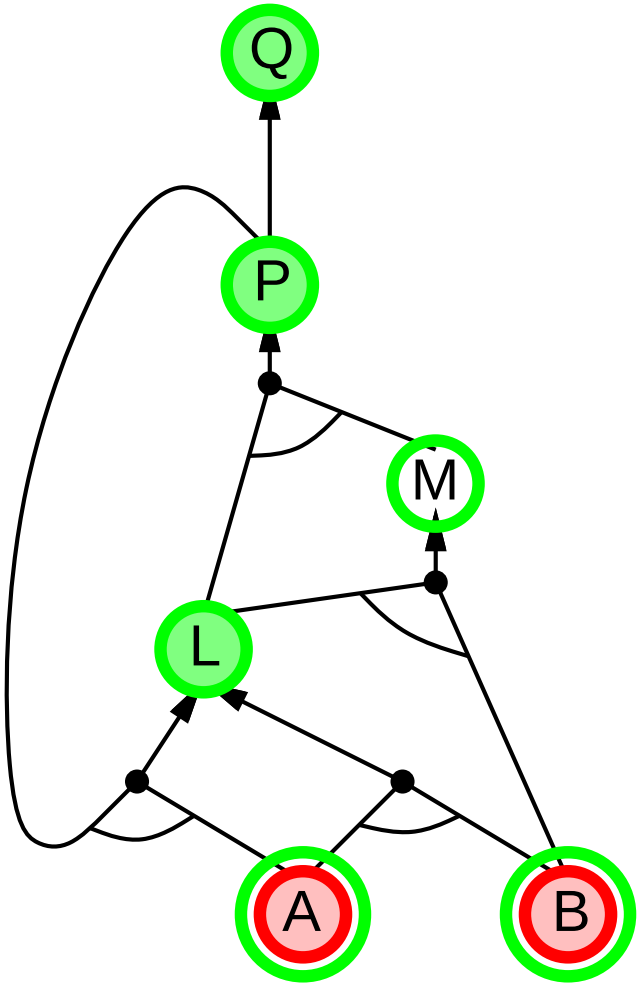
$B \wedge L \Rightarrow M$

$A \wedge P \Rightarrow L$

$\mathbf{A} \wedge \mathbf{B} \Rightarrow \mathbf{L}$

A

B



Backward chaining example

$\mathbf{P} \Rightarrow \mathbf{Q}$

$\mathbf{L} \wedge \mathbf{M} \Rightarrow \mathbf{P}$

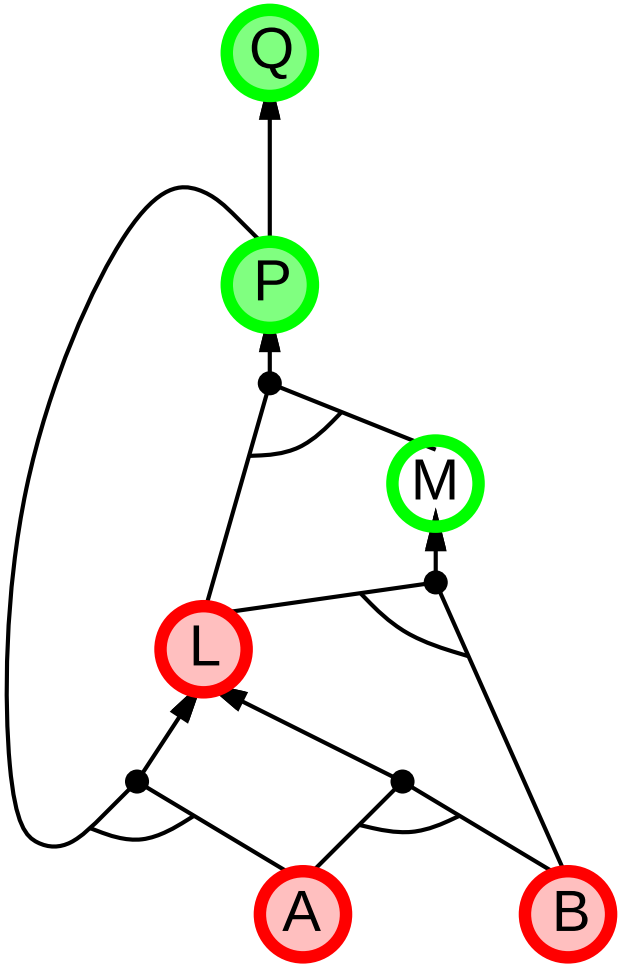
$B \wedge L \Rightarrow M$

$A \wedge P \Rightarrow L$

$\mathbf{A} \wedge \mathbf{B} \Rightarrow \mathbf{L}$ ✓

A

B



Backward chaining example

$\mathbf{P} \Rightarrow \mathbf{Q}$

$\mathbf{L} \wedge \mathbf{M} \Rightarrow \mathbf{P}$

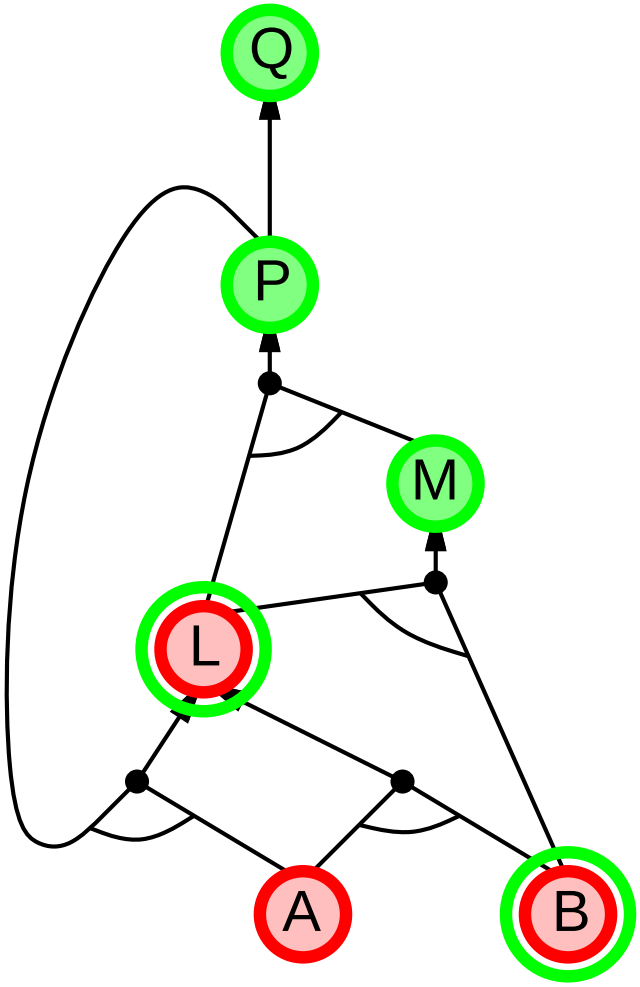
$\mathbf{B} \wedge \mathbf{L} \Rightarrow \mathbf{M}$

$A \wedge P \Rightarrow L$

$\mathbf{A} \wedge \mathbf{B} \Rightarrow \mathbf{L}$ ✓

A

B



Backward chaining example

$\mathbf{P} \Rightarrow \mathbf{Q}$

$\mathbf{L} \wedge \mathbf{M} \Rightarrow \mathbf{P}$

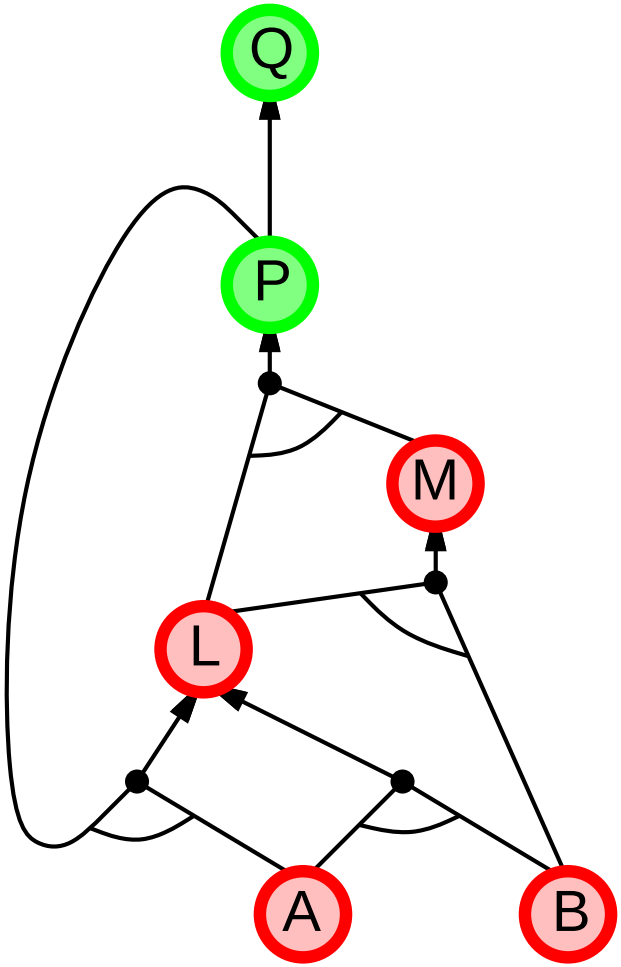
$\mathbf{B} \wedge \mathbf{L} \Rightarrow \mathbf{M}$ ✓

$A \wedge P \Rightarrow L$

$\mathbf{A} \wedge \mathbf{B} \Rightarrow \mathbf{L}$ ✓

A

B



Backward chaining example

$\mathbf{P} \Rightarrow \mathbf{Q}$

$\mathbf{L} \wedge \mathbf{M} \Rightarrow \mathbf{P}$ ✓

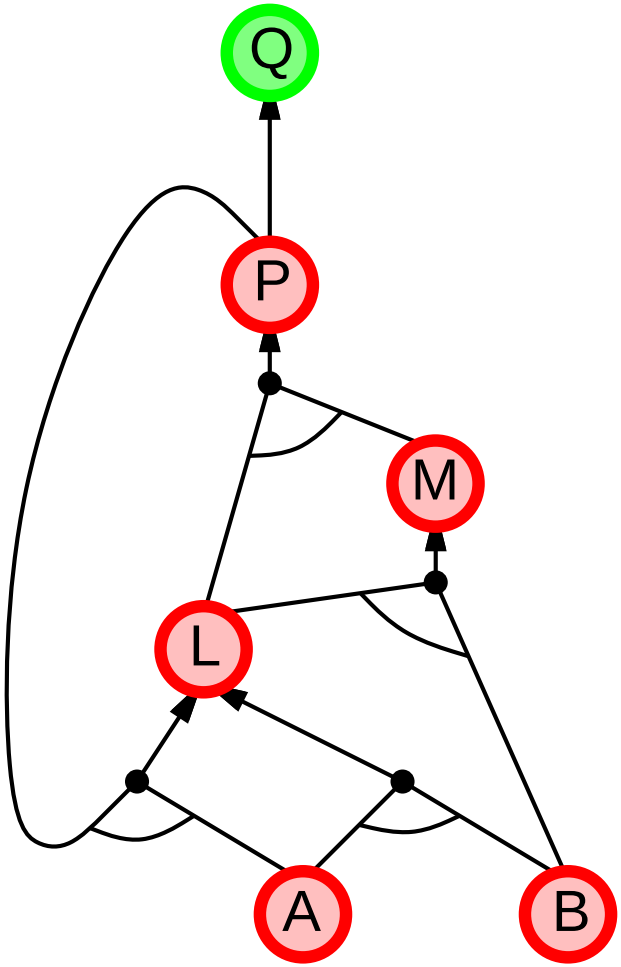
$\mathbf{B} \wedge \mathbf{L} \Rightarrow \mathbf{M}$ ✓

$A \wedge P \Rightarrow L$

$\mathbf{A} \wedge \mathbf{B} \Rightarrow \mathbf{L}$ ✓

A

B



Backward chaining example

$\mathbf{P} \Rightarrow \mathbf{Q}$ ✓

$\mathbf{L} \wedge \mathbf{M} \Rightarrow \mathbf{P}$ ✓

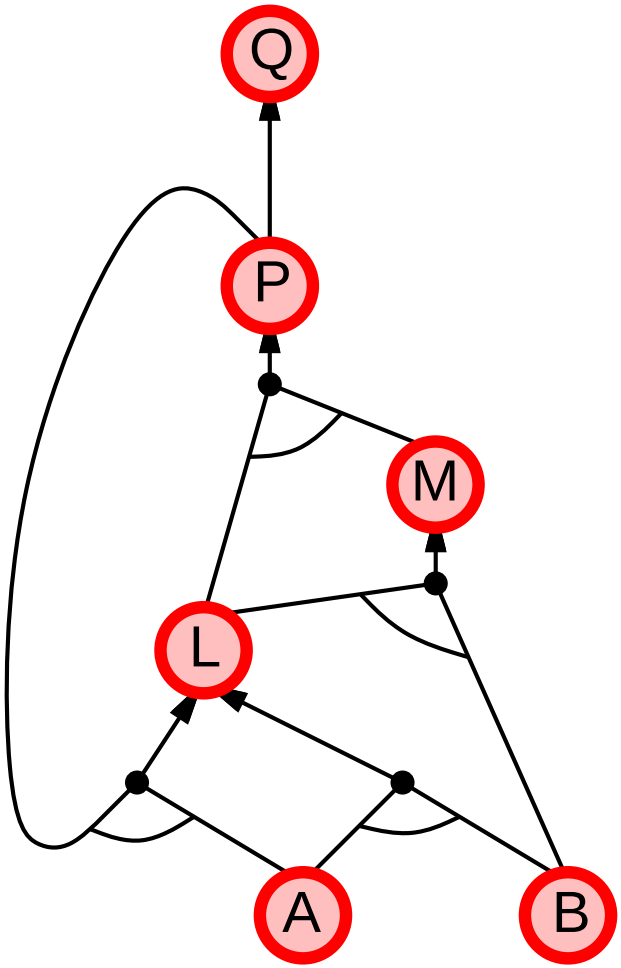
$\mathbf{B} \wedge \mathbf{L} \Rightarrow \mathbf{M}$ ✓

$A \wedge P \Rightarrow L$

$\mathbf{A} \wedge \mathbf{B} \Rightarrow \mathbf{L}$ ✓

A

B



Forward vs. backward chaining

- FC is *data-driven*
  - data-driven algorithms can be used for automatic, unconscious processing
  - e.g., object recognition, routine decisions
- May do lots of work that is irrelevant to the goal

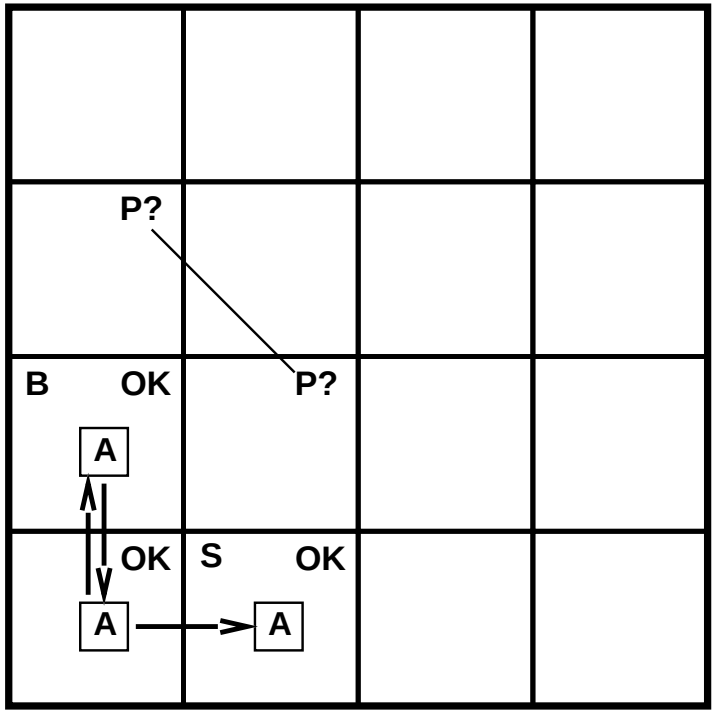
- BC is *goal-driven*, appropriate for problem-solving,
  - e.g., Where are my keys? How do I get into UMD's computer science program?
- Complexity of BC can be **much less** than linear in size of KB

Resolution

Conjunctive Normal Form (CNF): **conjunct** of **disjuncts** of **literals**

clauses

- E.g., $(A \vee \neg B) \wedge (B \vee \neg C \vee \neg D)$
- *Resolution* inference rule: if clause $\mathcal{C}$ contains a literal ℓ and clause $\mathcal{D}$ contains $\neg\ell$, then
  - infer the disjunct of all literals in $\mathcal{C}$ and $\mathcal{D}$ other than ℓ and $\neg\ell$
- E.g., $\dfrac{A \vee \neg B \qquad B \vee \neg C \vee \neg D}{A \vee \neg C \vee \neg D}$
- Wumpus world: $\dfrac{P_{1,3} \vee P_{2,2} \qquad \neg P_{2,2}}{P_{1,3}}$
- Proof by contradication: $\dfrac{P \qquad \neg P}{\square}$



Resolution

♢ Resolution is equivalent to Modus Ponens:

- Example of doing a resolution inference using modus ponens:

start with clauses: $A \vee \neg B$ $\quad$ $B \vee \neg C \vee \neg D$

rewrite as implications: $\neg A \Rightarrow \neg B$ $\quad$ $\neg B \wedge C \Rightarrow \neg D$

apply modus ponens: $\neg A \wedge C \Rightarrow \neg D$

rewrite as clauses: $A \vee \neg C \vee \neg D$

- Example of doing a modus ponens inference using resolution:

start with implications: $\neg A \Rightarrow \neg B$ $\quad$ $\neg B \wedge C \Rightarrow \neg D$

rewrite as clauses: $A \vee \neg B$ $\quad$ $B \vee \neg C \vee \neg D$

apply resolution: $A \vee \neg C \vee \neg D$

convert back to implications: $\neg A \wedge C \Rightarrow \neg D$

Conversion to CNF

♢ Resolution is sound and complete for propositional logic

- But to use it, you first must convert all everything to CNF

♢ Example:

- There's a breeze in (1,1) iff there's a pit in (1,2) or (2,1):
- $B_{1,1} \Leftrightarrow (P_{1,2} \vee P_{2,1})$

1. Eliminate $\Leftrightarrow$, by 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 $\Rightarrow$, by 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

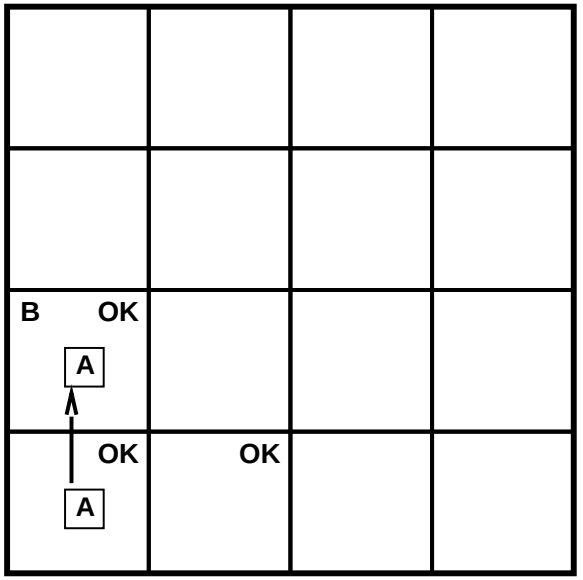
♢ Proof by contradiction

function PL-RESOLUTION(KB, α)
  $clauses \leftarrow$ {all clauses in the CNF representation of $KB \wedge \neg\alpha$}
  loop
    $new \leftarrow \{\}$
    for each C_i, C_j **in** $clauses$ **do**
      $resolvents \leftarrow$ PL-RESOLVE(C_i, C_j)
      if $resolvents$ contains the empty clause **then return** $true$
      $new \leftarrow new \cup resolvents$
    if $new \subseteq clauses$ **then return** $false$
    $clauses \leftarrow clauses \cup new$

---

function PL-RESOLVE(C_1, C_2) **returns** a set of clauses
  $resolvents \leftarrow \{\}$
  for each $l \in C_1$ and $m \in C_2$ such that $l \equiv \neg m$ **do**
    $C \leftarrow$ disjunct of all literals in C_1 and C_2 except for l and m
    $resolvents \leftarrow resolvents \cup \{C\}$
  return $resolvents$

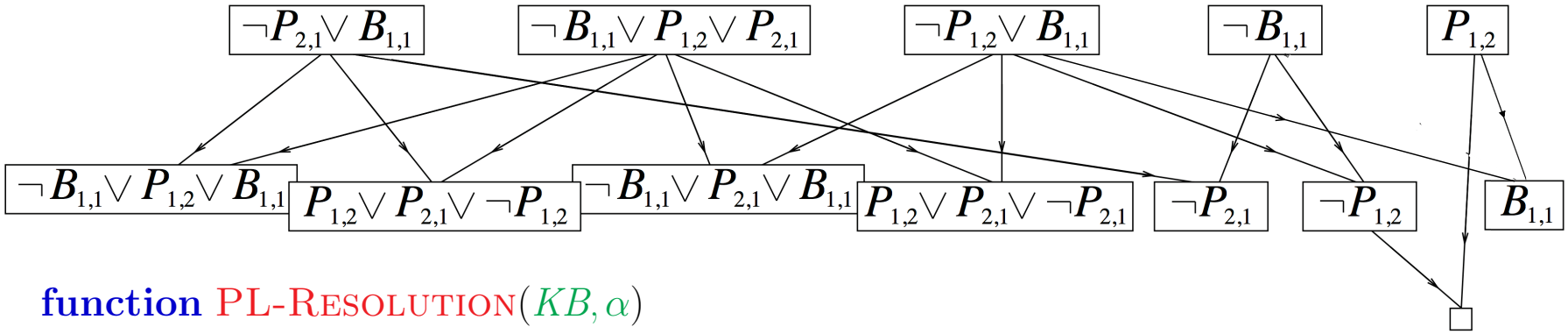
Resolution example



- KB:
  - There's a breeze in (1,1) iff there's a pit in (1,2) or (2,1):
  - There's no breeze in (1,1):
  - $KB = (B_{1,1} \Leftrightarrow (P_{1,2} \vee P_{2,1})) \wedge \neg B_{1,1}$
- KB converted to CNF:

  $(\neg B_{1,1} \vee P_{1,2} \vee P_{2,1}) \wedge (\neg P_{1,2} \vee B_{1,1}) \wedge (\neg P_{2,1} \vee B_{1,1}) \wedge \neg B_{1,1}$
- Want to show there's no pit in (1,2): $\alpha = \neg P_{1,2}$
- $clauses$ = {all clauses in the CNF representation of $KB \wedge \alpha$}

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



function PL-RESOLUTION(KB, α)
  $clauses \leftarrow$ {all clauses in the CNF representation of $KB \wedge \neg\alpha$}
  loop
    $new \leftarrow \{\}$
    for each C_i, C_j **in** $clauses$ **do**
      $resolvents \leftarrow$ PL-RESOLVE(C_i, C_j)
      if $resolvents$ contains the empty clause **then return** $true$
      $new \leftarrow new \cup resolvents$
    if $new \subseteq clauses$ **then return** $false$
    $clauses \leftarrow clauses \cup new$

---

function PL-RESOLVE(C_1, C_2) **returns** a set of clauses
  $resolvents \leftarrow \{\}$
  for each $l \in C_1$ and $m \in C_2$ such that $l \equiv \neg m$ **do**
    $C \leftarrow$ disjunct of all literals in C_1 and C_2 except for l and m
    $resolvents \leftarrow resolvents \cup \{C\}$
  return $resolvents$

Axioms for the Wumpus world

♢ For each square, an axiom telling whether it's breezy

- $B_{1,1} \Leftrightarrow (P_{1,2} \vee P_{2,1})$
- $B_{1,2} \Leftrightarrow (P_{1,1} \wedge P_{2,2} \wedge P_{1,3})$
- ...
- $B_{2,2} \Leftrightarrow (P_{1,2} \vee P_{2,1} \vee P_{2,3} \vee P_{3,2})$
- ...
- $B_{4,4} \Leftrightarrow (P_{1,1} \wedge P_{2,2} \wedge P_{1,3})$

♢ For each square, an axiom telling whether it's smelly

- $S_{1,1} \Leftrightarrow (W_{1,2} \vee W_{2,1})$
- ...

♢ For each square, an axiom telling whether it's glittery

- ...

Axioms for the Wumpus world

♢ Axioms saying there's exactly one square that contains a Wumpus:

- $W_{1,1} \vee W_{1,2} \vee \ldots \vee W_{4,3} \vee W_{4,4}$
- $\neg W_{1,1} \vee \neg W_{1,2}$
- $\neg W_{1,1} \vee \neg W_{1,3}$
- $\ldots$
- $\neg W_{4,3} \vee \neg W_{4,4}$

Fluents

♢ Suppose the agent perceives a breeze at time $t = 3$ and a stench at time $t = 4$

- Can't just add $Breeze$ and $Stench$ to KB
  - ◇ They are perceived at some times but not others
- Instead, use new propositions called *fluents*
  - ◇ $Breeze^3$ and $Stench^4$

♢ Need fluents for everything that can change over time

- ◇ $L^5_{3,4}$ at square [3,4] at time $t = 5$
- ◇ $East^8$ facing east at time $t = 8$
- ◇ $Have\_arrow^{17}$ have the arrow at time $t = 17$
- ◇ $Wumpus\_alive^4$ wumpus is alive at time $t = 4$

♢ Need fluents like these for $t = 0, 1, 2, \ldots, t_{\max}$,

- where $t_{\max}$ is the largest time that you'll ever need to consider

Mapping fluents to domain properties

♢ Axioms for mapping "breeze" percepts into properties of the domain:

- $L^1_{1,1} \Rightarrow (Breeze^1 \Leftrightarrow B_{1,1})$
- $L^1_{1,2} \Rightarrow (Breeze^1 \Leftrightarrow B_{1,2})$
- ...

and similarly for all other time points

♢ Similarly,

- $L^1_{1,1} \Rightarrow (Stench^1 \Leftrightarrow S_{1,1})$
- $L^1_{1,2} \Rightarrow (Stench^1 \Leftrightarrow S_{1,2})$
- ...

and similarly for all other time points

♢ Likewise for all of the other percepts

How to describe actions?

♢ One way of writing axioms to describe movement:

- $L^0_{1,1} \wedge East^0 \wedge Forward \Rightarrow (L^1_{2,1} \wedge \neg L^1_{1,1})$
- $L^0_{1,2} \wedge East^0 \wedge Forward \Rightarrow (L^1_{2,2} \wedge \neg L^1_{1,2})$
- $\ldots$

one for each possible combination of location, time point, and direction

♢ Add similar axioms for each of the other actions

- $Grab, Shoot, Turn\_right, \ldots$

♢ These are called *effect* axioms

♢ Problem: they don't represent all of the consequences of each action

- What do they leave out?

How to describe actions?

♢ The *frame* problem:

- An "effect" axiom describes the **changes** due to an action

$$L^0_{1,1} \wedge East^0 \wedge Forward \Rightarrow (L^1_{2,1} \wedge \neg L^1_{1,1})$$

- But it doesn't describe what **doesn't** change
- If we have the arrow and we move forward, we'll still have the arrow
  - ◇ We can't infer this unless we have an axiom that says it

♢ *Qualification problem*: real actions require endless caveats

- what if gold is slippery or nailed down or . . .

♢ *Ramification problem*: real actions have many secondary consequences

- dust on the gold, wear and tear on gloves, . . .

♢ The Wumpus world is so simple that it's easy to specify each action's qualifications and ramifications

- But we still need to handle the frame problem

♢ Instead of "effect" axioms, we'll use *successor state axioms*

Successor state axioms

- If a fluent f is true at time $t+1$, then either it was true at time t or we performed an action that made it true
- If we have the arrow at time $t+1$, then we had it and didn't shoot it
  - $HaveArrow^2 \Leftrightarrow (HaveArrow^1 \wedge \neg Shoot^1)$
  - $HaveArrow^3 \Leftrightarrow (HaveArrow^2 \wedge \neg Shoot^2)$
  - ...
- If we're at location l at time $t+1$, then we moved there, or we were already there and we didn't move, or we were already there and we bumped into the wall
  - $L_{1,1}^{t+1} \Leftrightarrow (L_{1,1}^t \wedge \neg Forward^t \vee Bump^{t+1})$
    $\vee (L_{1,2}^t \wedge South^t \wedge Forward^t) \vee (L_{2,1}^t \wedge West^t \wedge Forward^t)$
  - need one of those for each location and each value of t
- Need similar axioms for all the other fluents

Inferring where it's OK to move

- $OK^1_{1,1} \Leftrightarrow \neg P_{1,1} \wedge (\neg W_{1,1} \vee \neg Wumpus\_Alive^1)$
- $OK^1_{1,2} \Leftrightarrow \neg P_{1,2} \wedge (\neg W_{1,2} \vee \neg Wumpus\_Alive^1)$
- $\ldots$
- $OK^1_{4,4} \Leftrightarrow \neg P_{4,4} \wedge (\neg W_{4,4} \vee \neg Wumpus\_Alive^1)$

- $OK^2_{1,1} \Leftrightarrow \neg P_{1,1} \wedge (\neg W_{1,1} \vee \neg Wumpus\_Alive^2)$
- $OK^2_{1,2} \Leftrightarrow \neg P_{1,2} \wedge (\neg W_{1,2} \vee \neg Wumpus\_Alive^2)$
- $\ldots$
- $OK^2_{4,4} \Leftrightarrow \neg P_{4,4} \wedge (\neg W_{4,4} \vee \neg Wumpus\_Alive^2)$

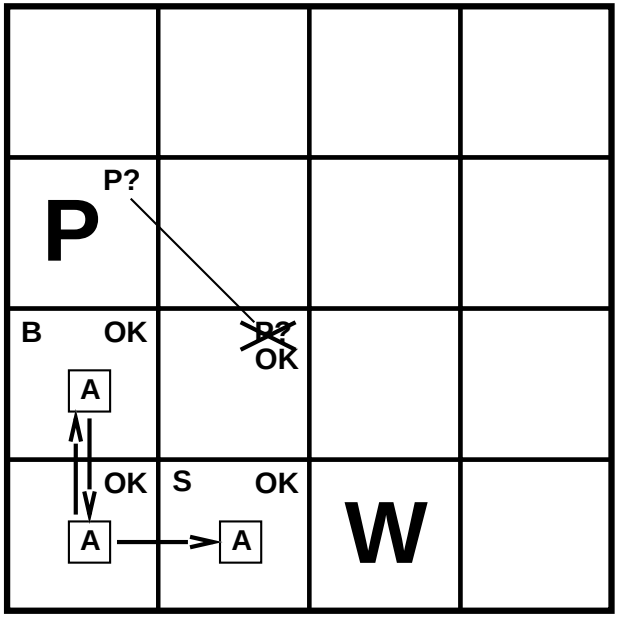
- $\ldots$ for each value of t $\ldots$

Inferring facts about the state of the world

$$\neg Stench^0 \wedge \neg Breeze^0 \wedge \neg Glitter^0 \wedge \neg Bump^0 \wedge \neg Scream^0; \quad Forward^0$$
$$\neg Stench^1 \wedge \quad Breeze^1 \wedge \neg Glitter^1 \wedge \neg Bump^1 \wedge \neg Scream^1; \quad Turn\_right^1$$
$$\neg Stench^2 \wedge \quad Breeze^2 \wedge \neg Glitter^2 \wedge \neg Bump^2 \wedge \neg Scream^2; \quad Turn\_right^2$$
$$\neg Stench^3 \wedge \quad Breeze^3 \wedge \neg Glitter^3 \wedge \neg Bump^3 \wedge \neg Scream^3; \quad Forward^3$$
$$\neg Stench^4 \wedge \neg Breeze^4 \wedge \neg Glitter^4 \wedge \neg Bump^4 \wedge \neg Scream^4; \quad Turn\_Left^4$$
$$\neg Stench^5 \wedge \neg Breeze^5 \wedge \neg Glitter^5 \wedge \neg Bump^5 \wedge \neg Scream^5; \quad Forward^5$$
$$Stench^6 \wedge \neg Breeze^6 \wedge \neg Glitter^6 \wedge \neg Bump^6 \wedge \neg Scream^6$$

♢ With all of the axioms and a resolution theorem prover, we should be able to get

- $ASK(KB, L^6_{2,1}) = true$
- $ASK(KB, P_{1,3}) = true$
- $ASK(KB, W_{3,1}) = true$
- $ASK(KB, OK^6_{2,2}) = true$



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 representing partial and negated information, reasoning by cases, etc.
- ♢ Forward and backward chaining are linear-time, complete for Horn clauses
- ♢ Resolution is complete for propositional logic
- ♢ Propositional logic lacks expressive power