Last update: March 4, 2010

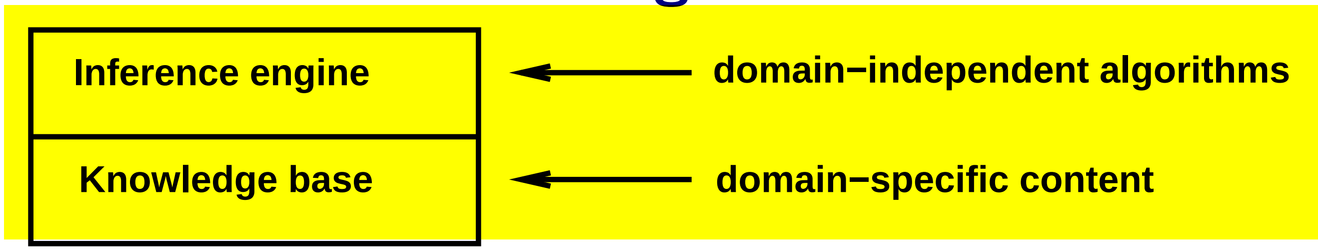
Logical agents

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

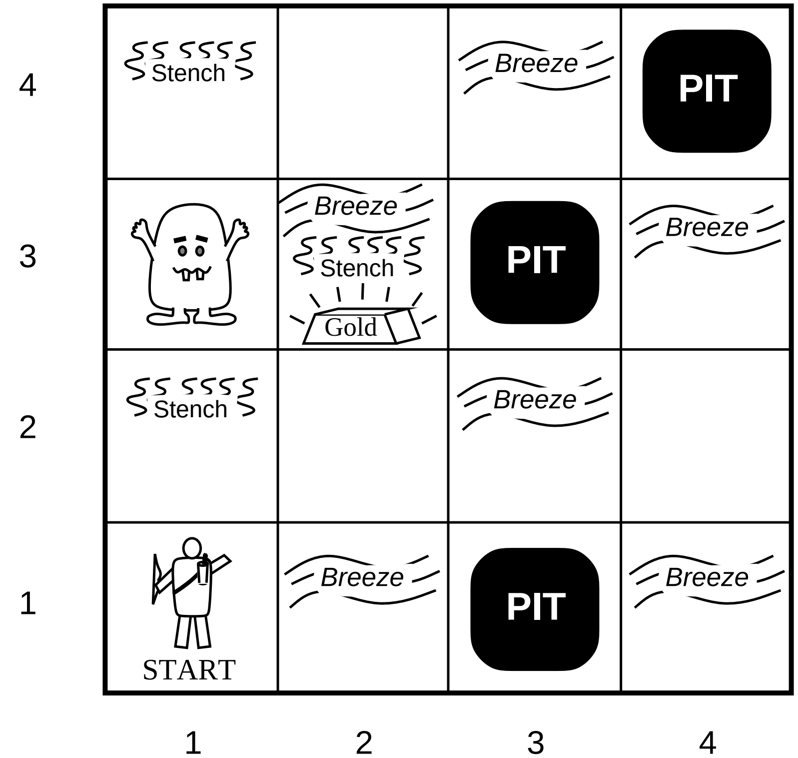
function KB-AGENT(*percept*) **returns** an *action*
  static: *KB*, a knowledge base
    t, a counter, initially 0, indicating time

  TELL(*KB*, MAKE-PERCEPT-SENTENCE(*percept*, *t*))
  action ← ASK(*KB*, MAKE-ACTION-QUERY(*t*))
  TELL(*KB*, MAKE-ACTION-SENTENCE(*action*, *t*))
  $t \leftarrow t + 1$
  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

Environment:
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



Performance measure:
gold $+1000$, death -1000, -1 per step, -10 for using the arrow

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

Sensors: 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

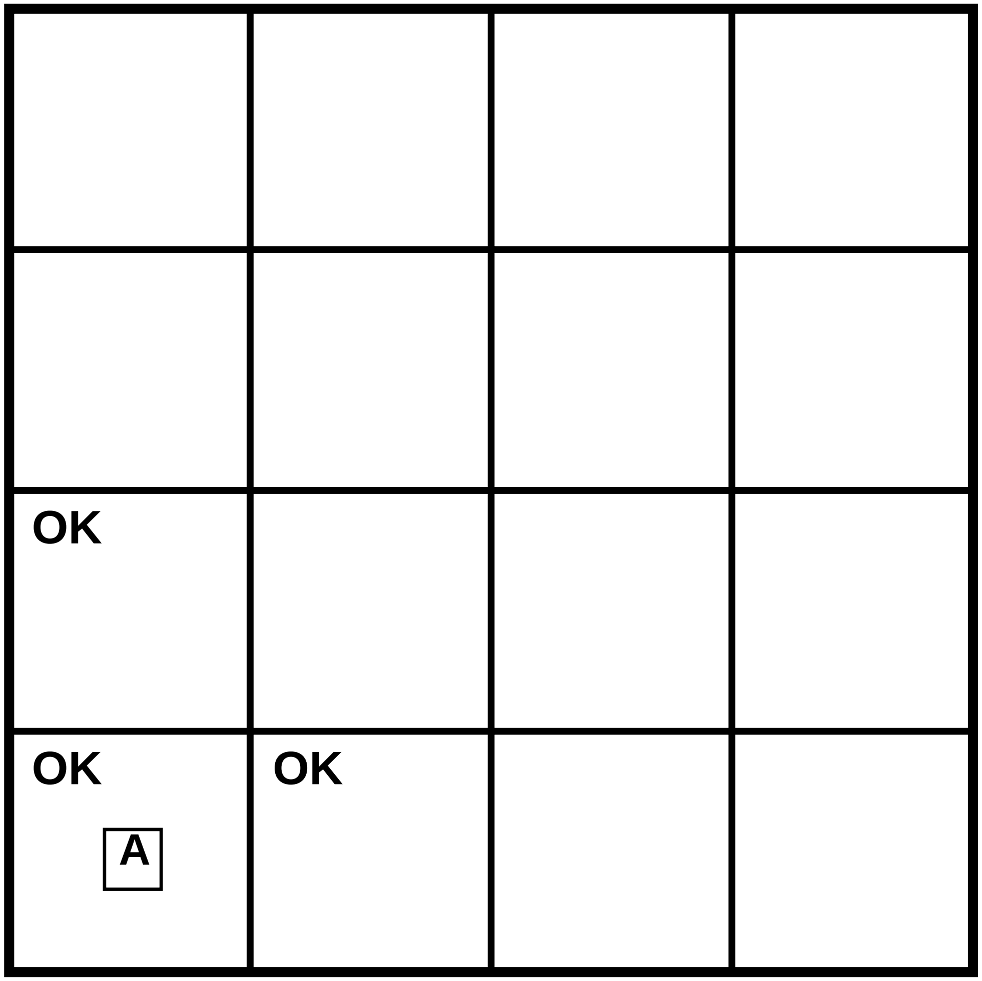
Episodic? No—sequential at the level of actions

Static? Yes—Wumpus, pits, and gold do not move

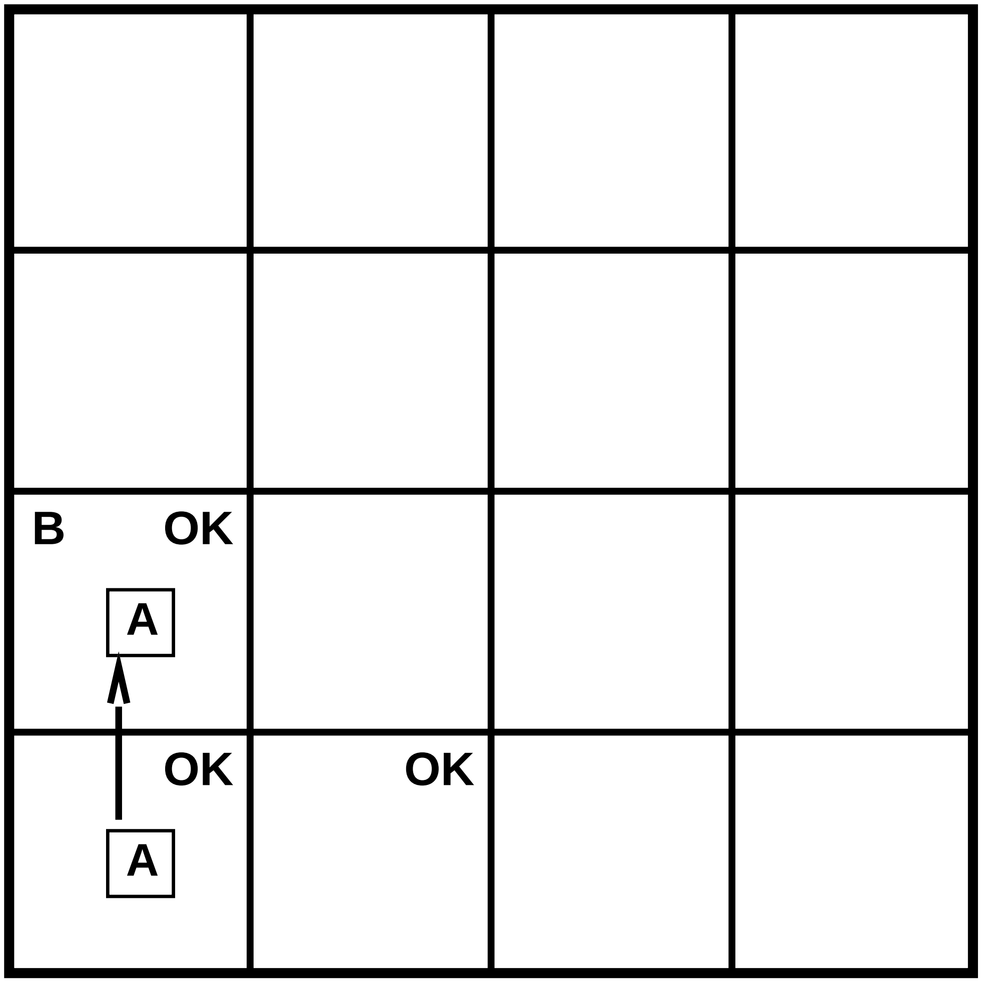
Discrete? Yes

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

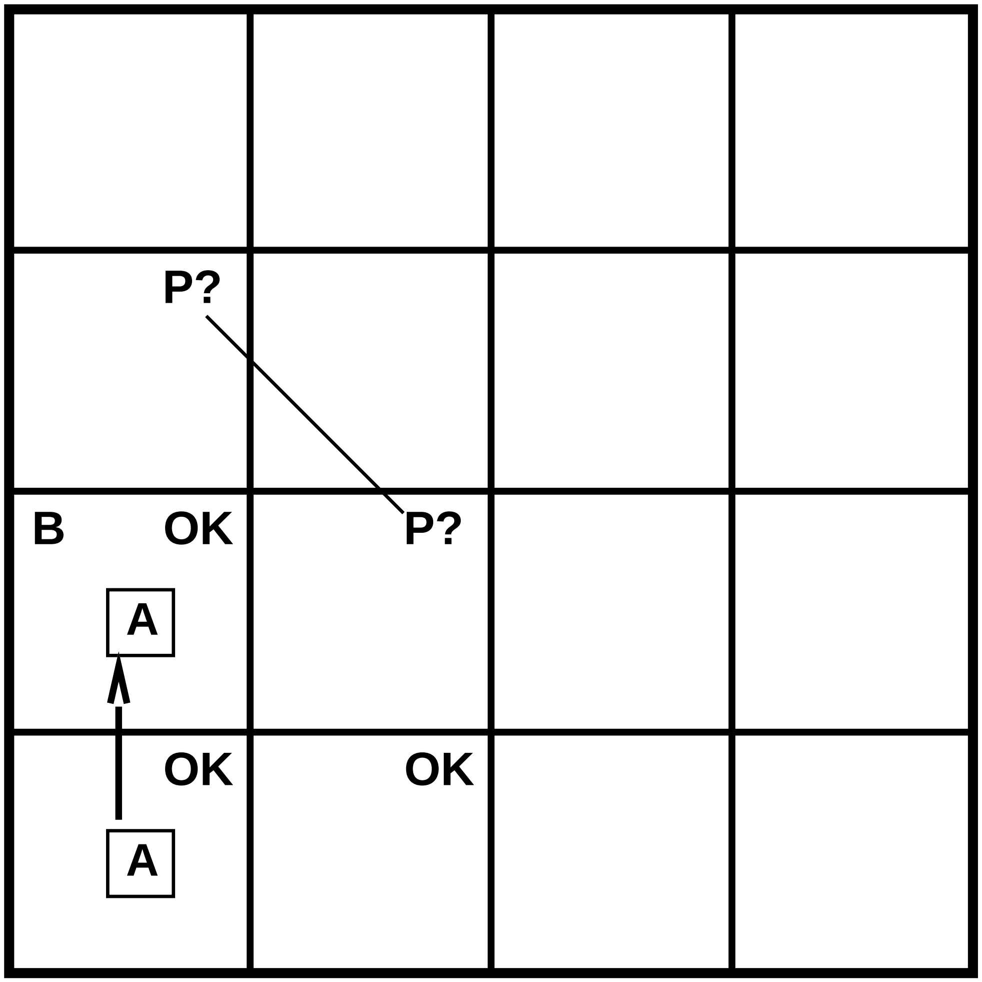
Exploring a wumpus world



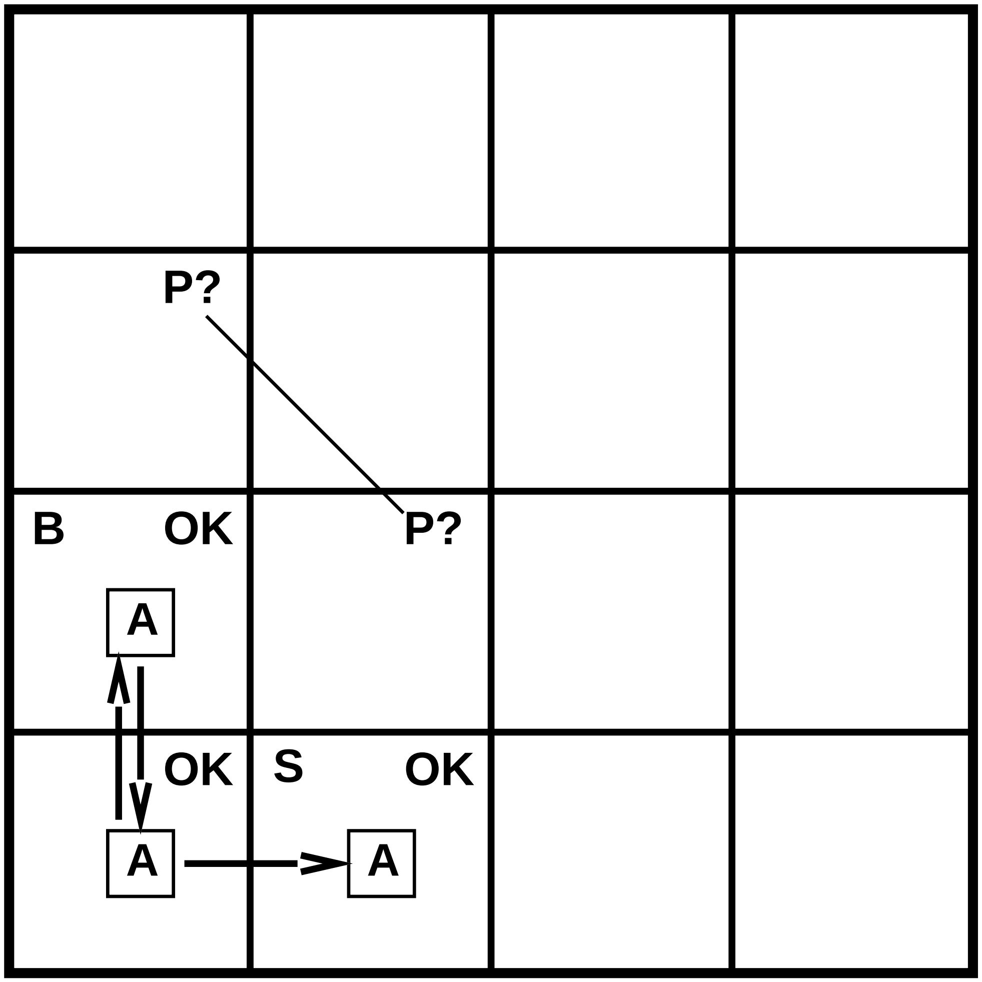
Exploring a wumpus world



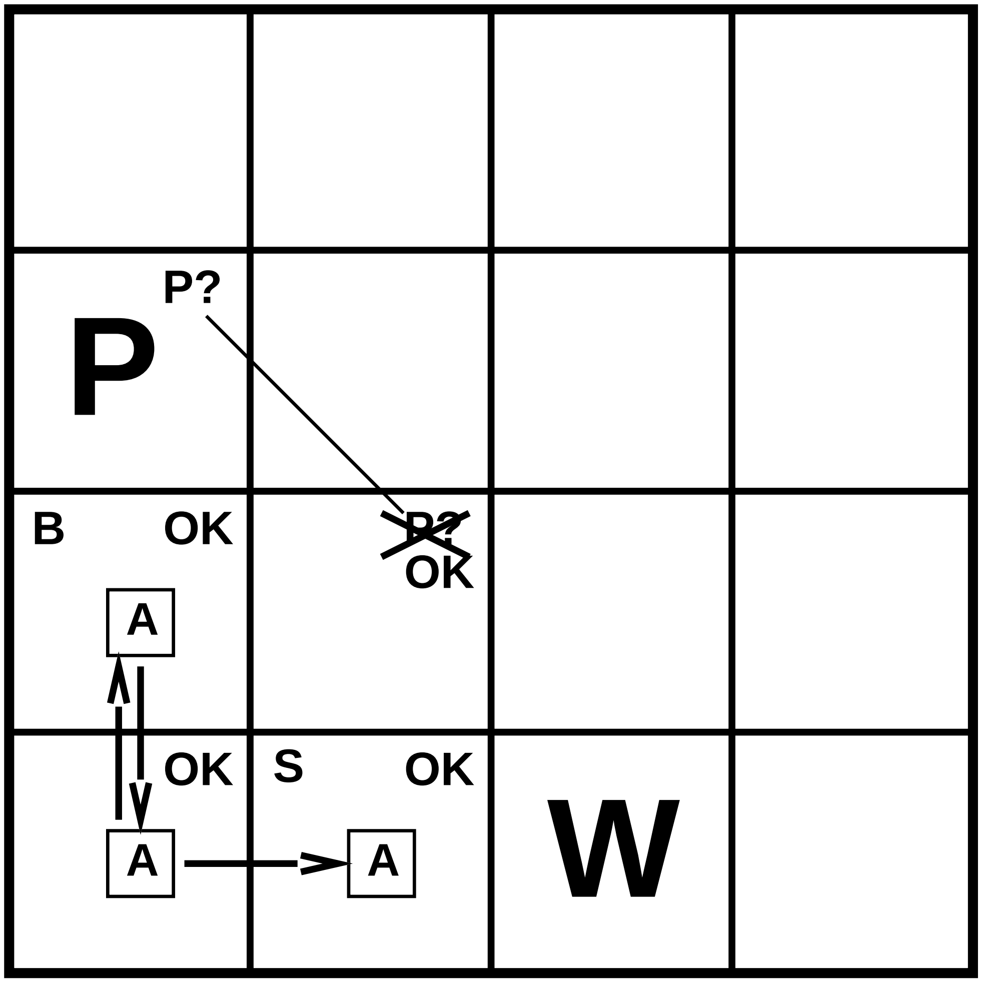
Exploring a wumpus world



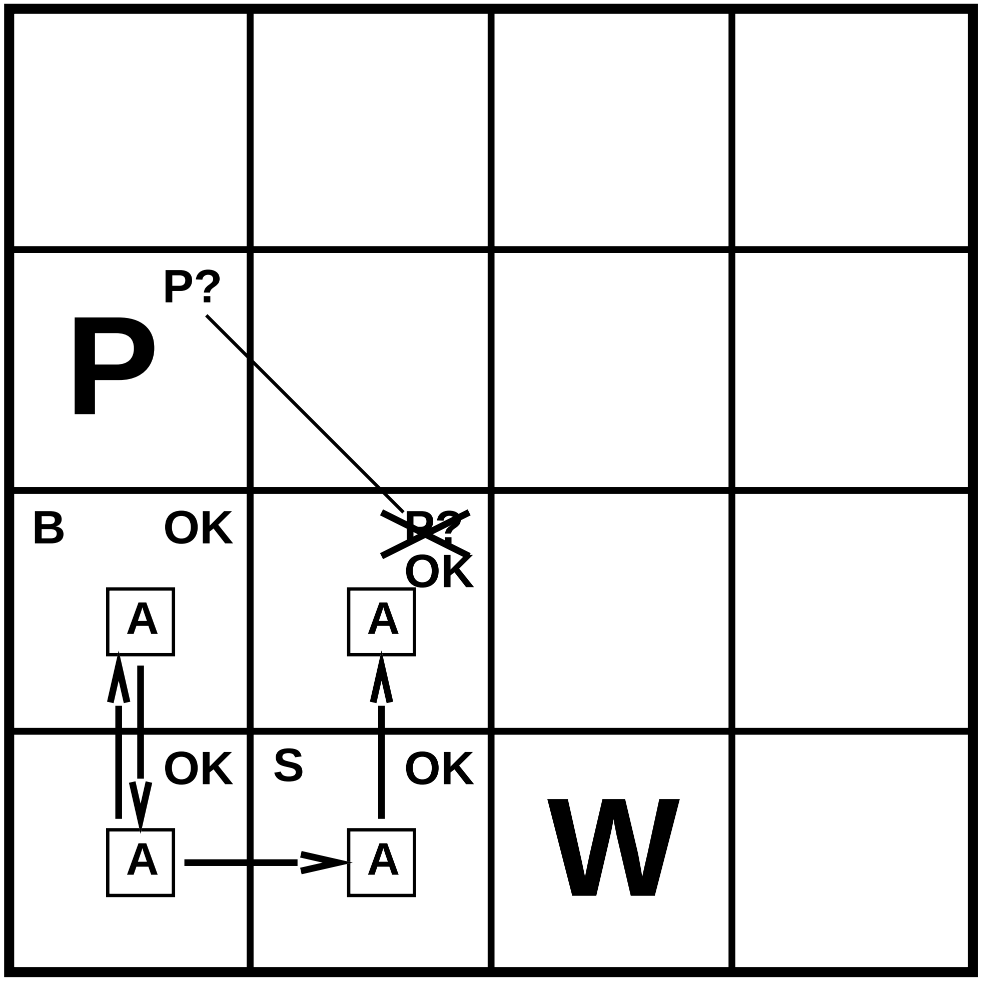
Exploring a wumpus world



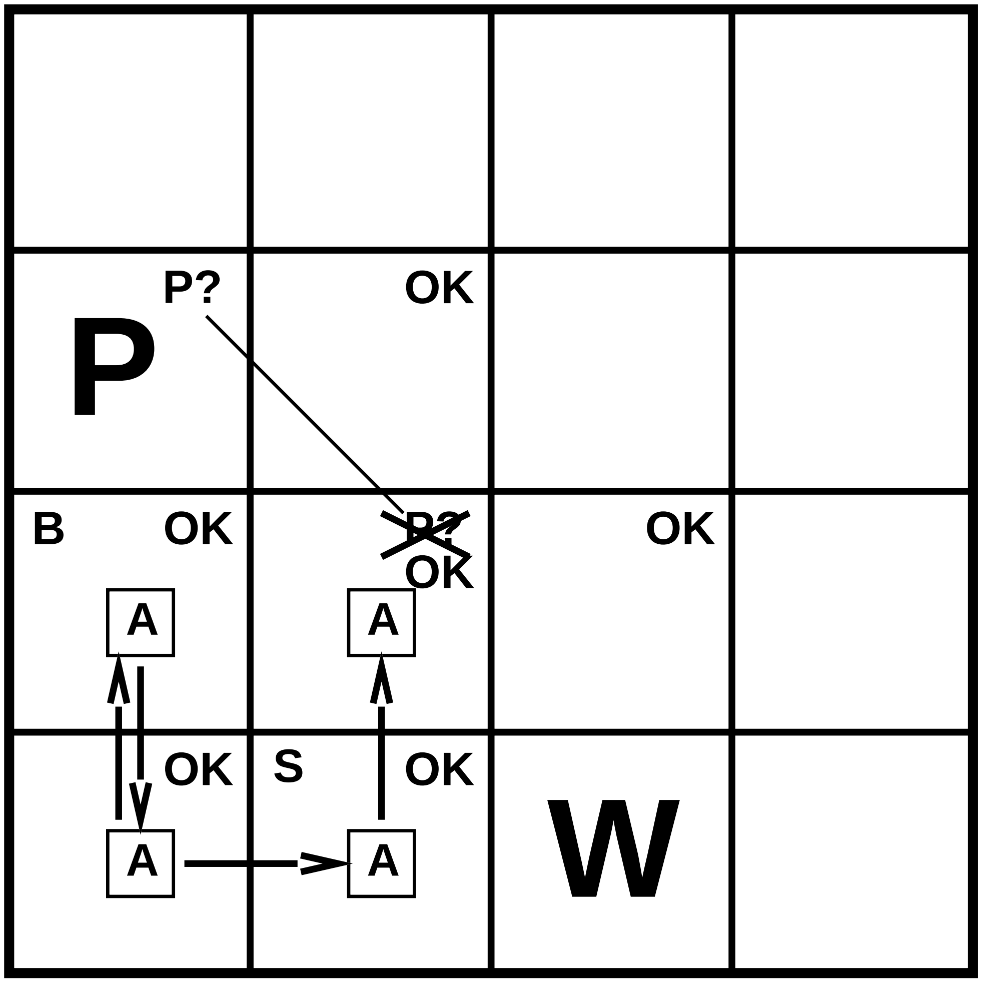
Exploring a wumpus world



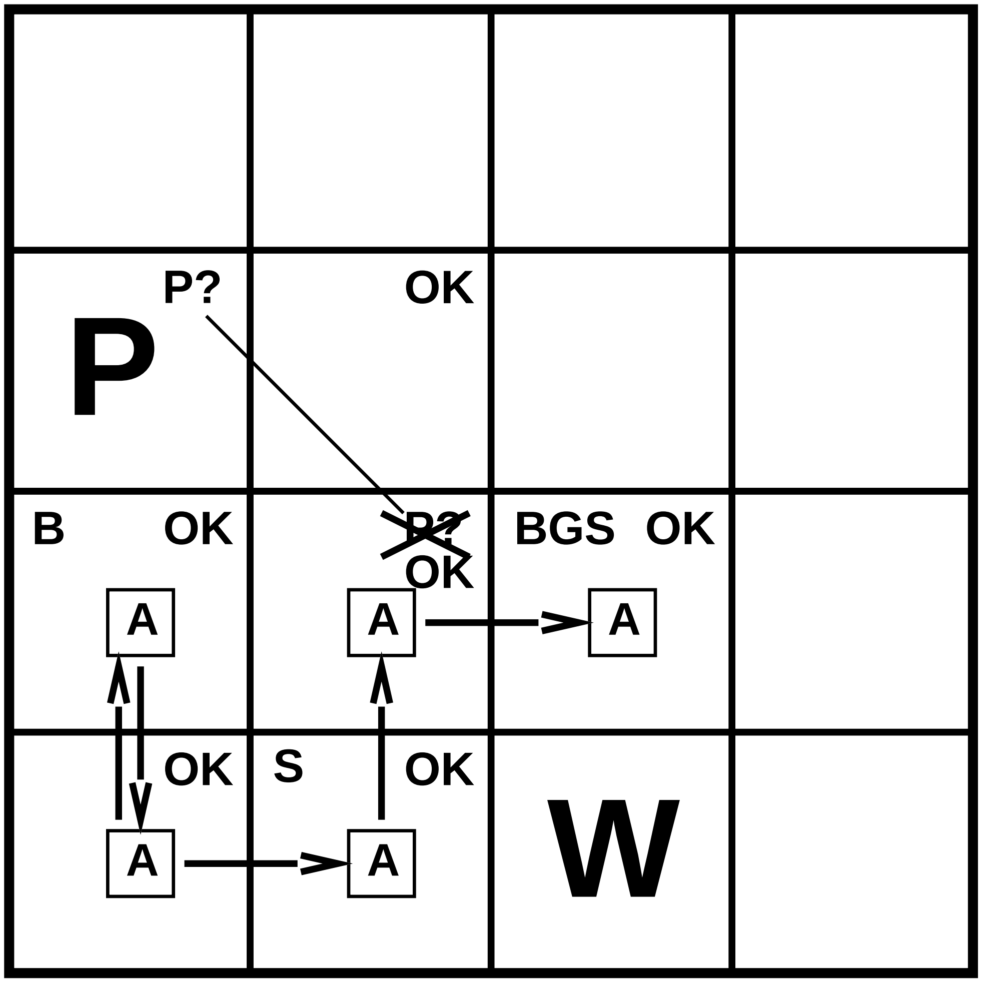
Exploring a wumpus world



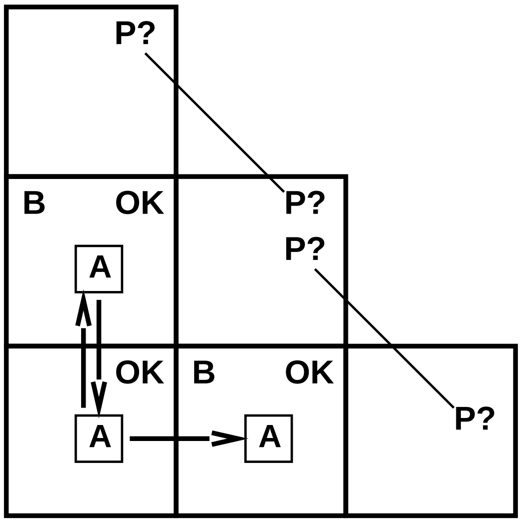
Exploring a wumpus world



Exploring a wumpus world



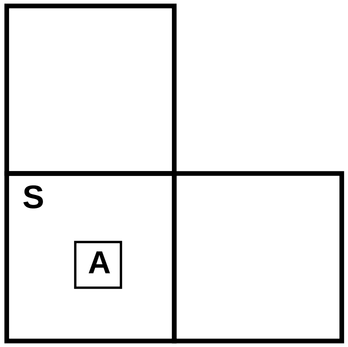
Other tight spots



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

P(pit in (2,2)) $\approx$ 0.86
P(pits in (1,3) and (3,1)) $\approx$ 0.31

In a later chapter we'll see
how to compute this



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

$KB \models \alpha$

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

E.g., if a KB contains "Maryland won" and "Duke won",
the KB entails "Maryland won or* Duke 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)

---

*The "or" is inclusive, not exclusive.

Models

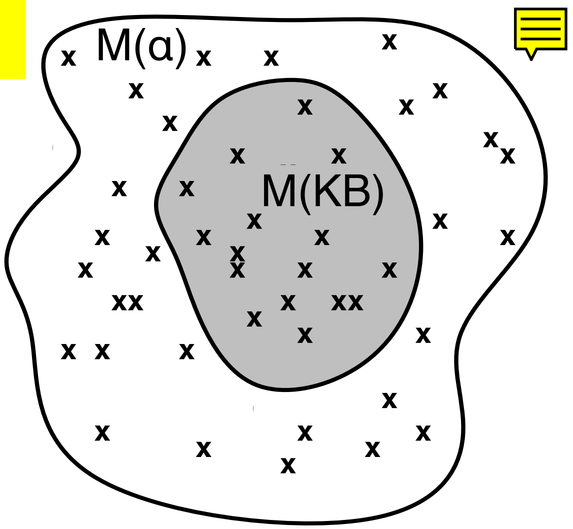
A *model* is a formally structured world in which truth can be evaluated

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. $KB =$ Maryland won and Duke won
$\alpha =$ Maryland won



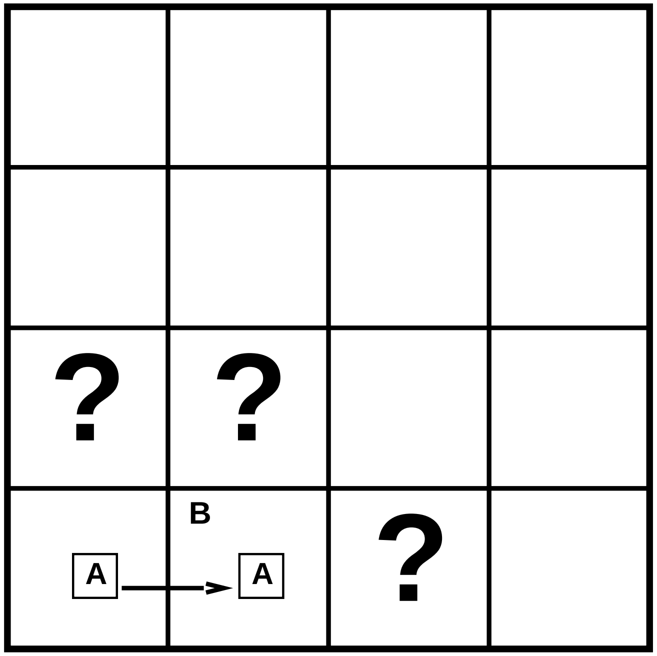
Entailment in the wumpus world

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

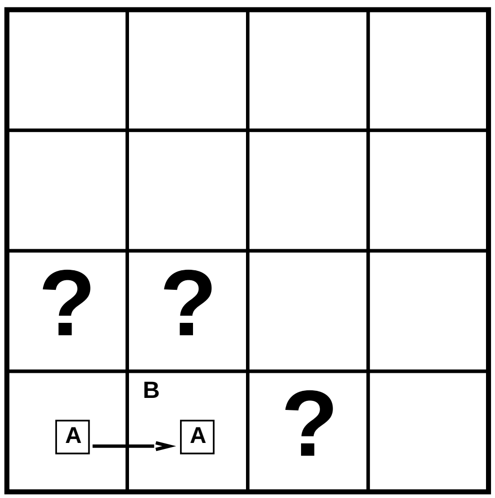
For now, ignore the wumpus and gold. Which of the **?**s are pits?

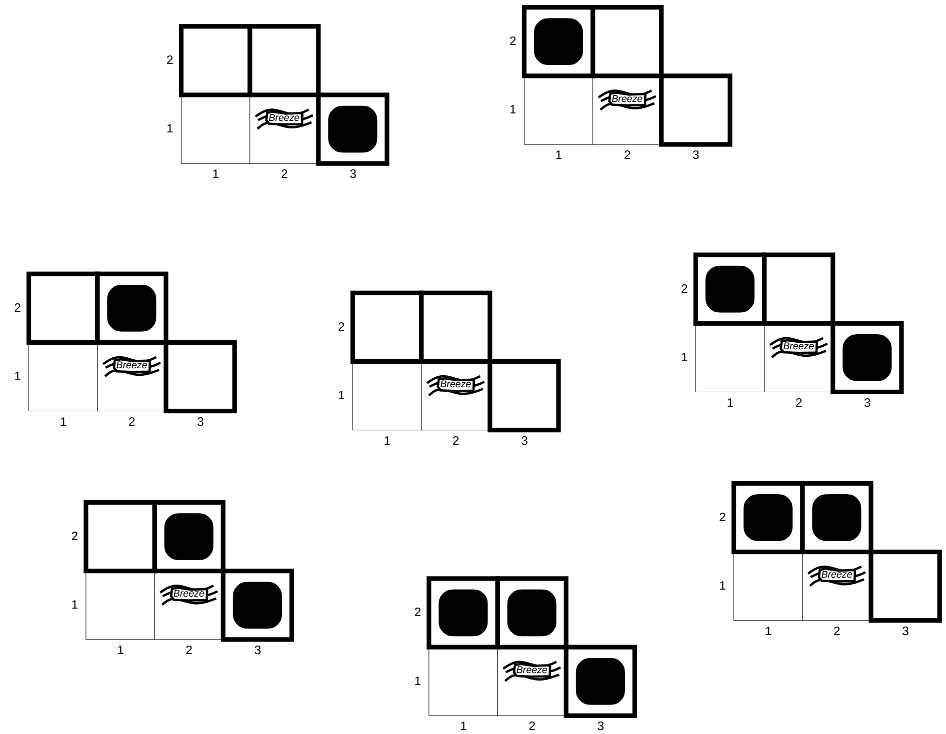
For each possible combination of pit locations, check whether it's a model.

3 Boolean choices $\Rightarrow$ 8 possible models



Wumpus models

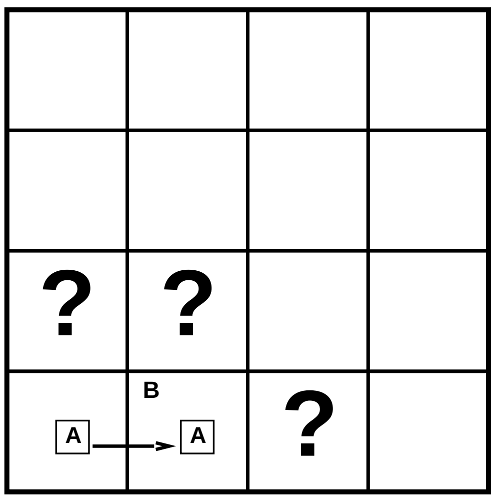


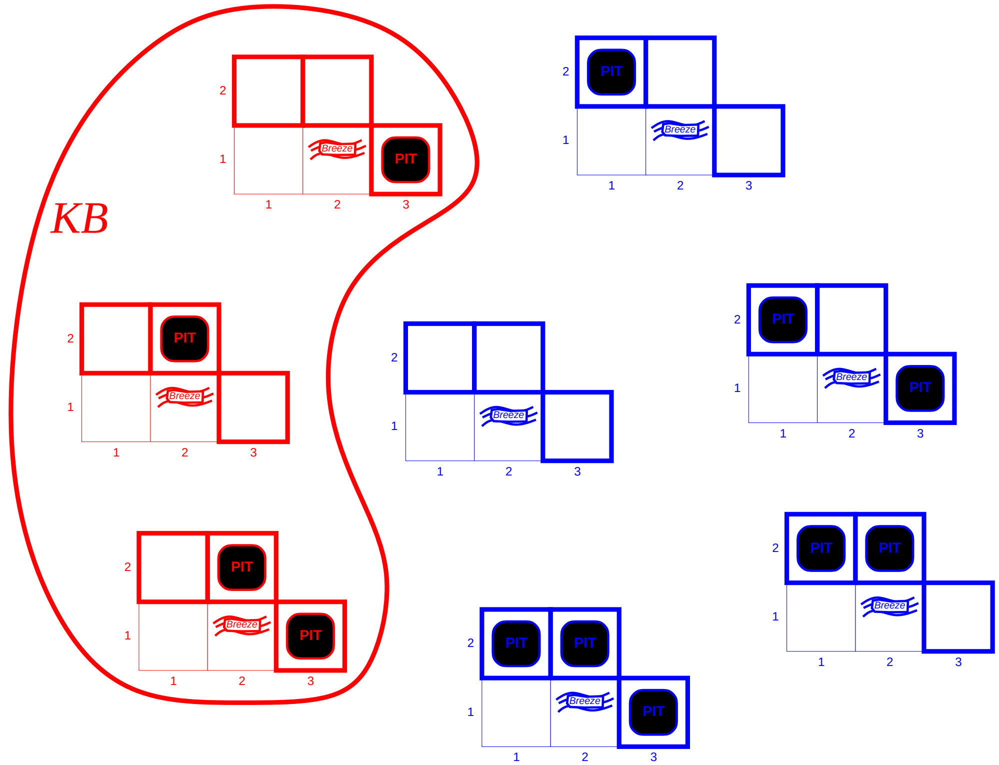


KB = wumpus-world rules + observations

Eight possible combinations of pit locations: which ones are models of KB?

Wumpus models

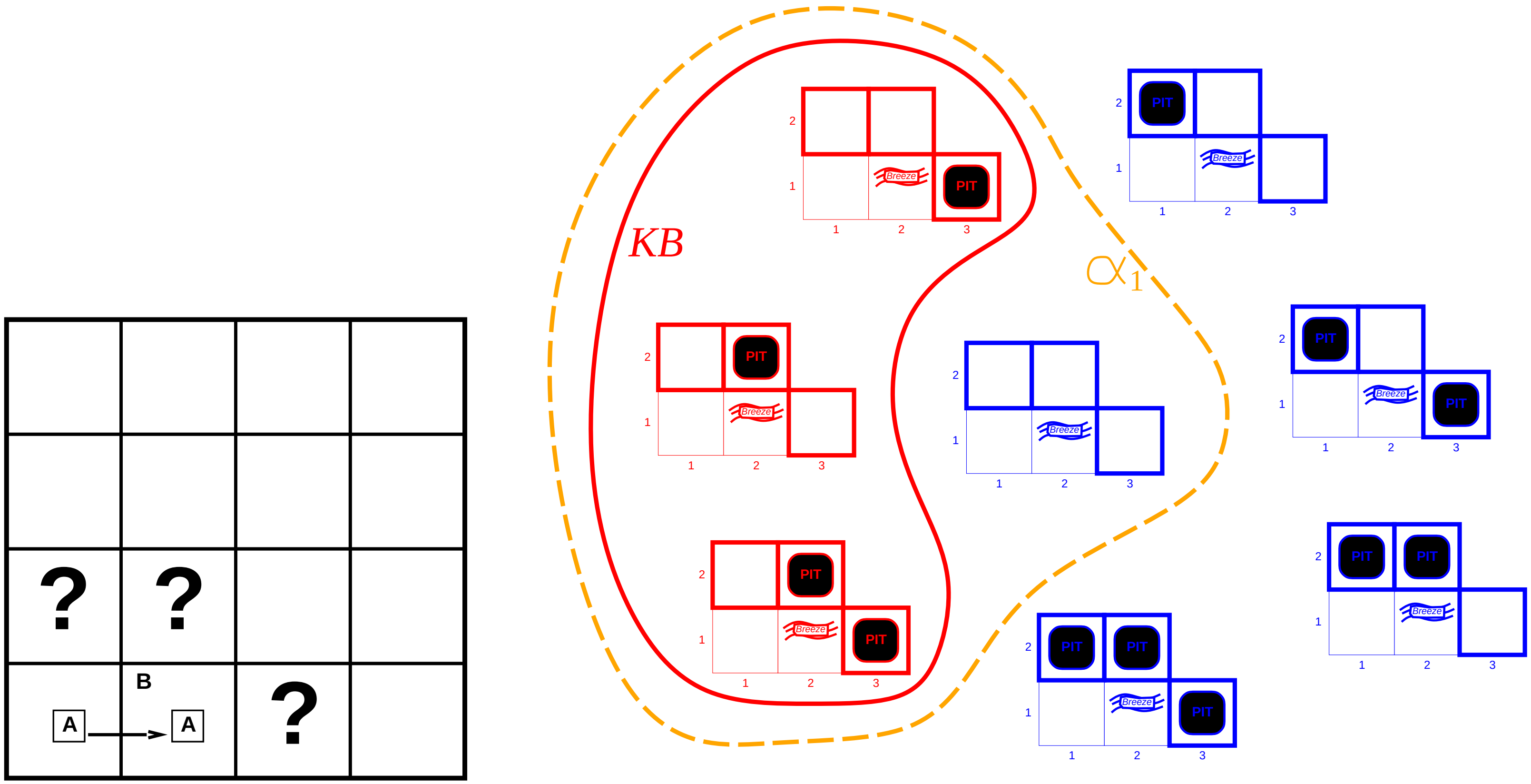




KB = wumpus-world rules + observations

Three models

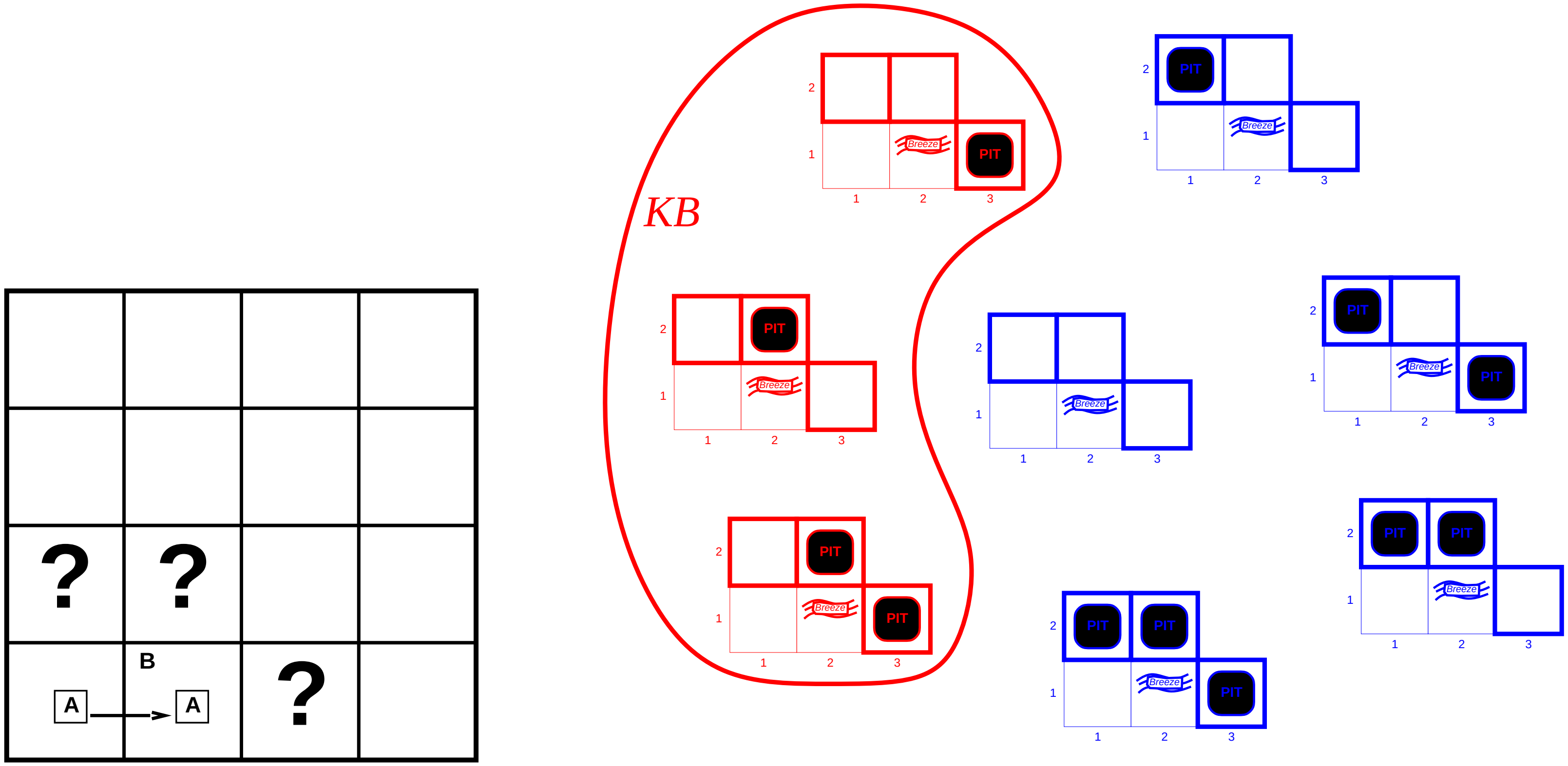
Wumpus models



KB = wumpus-world rules + observations

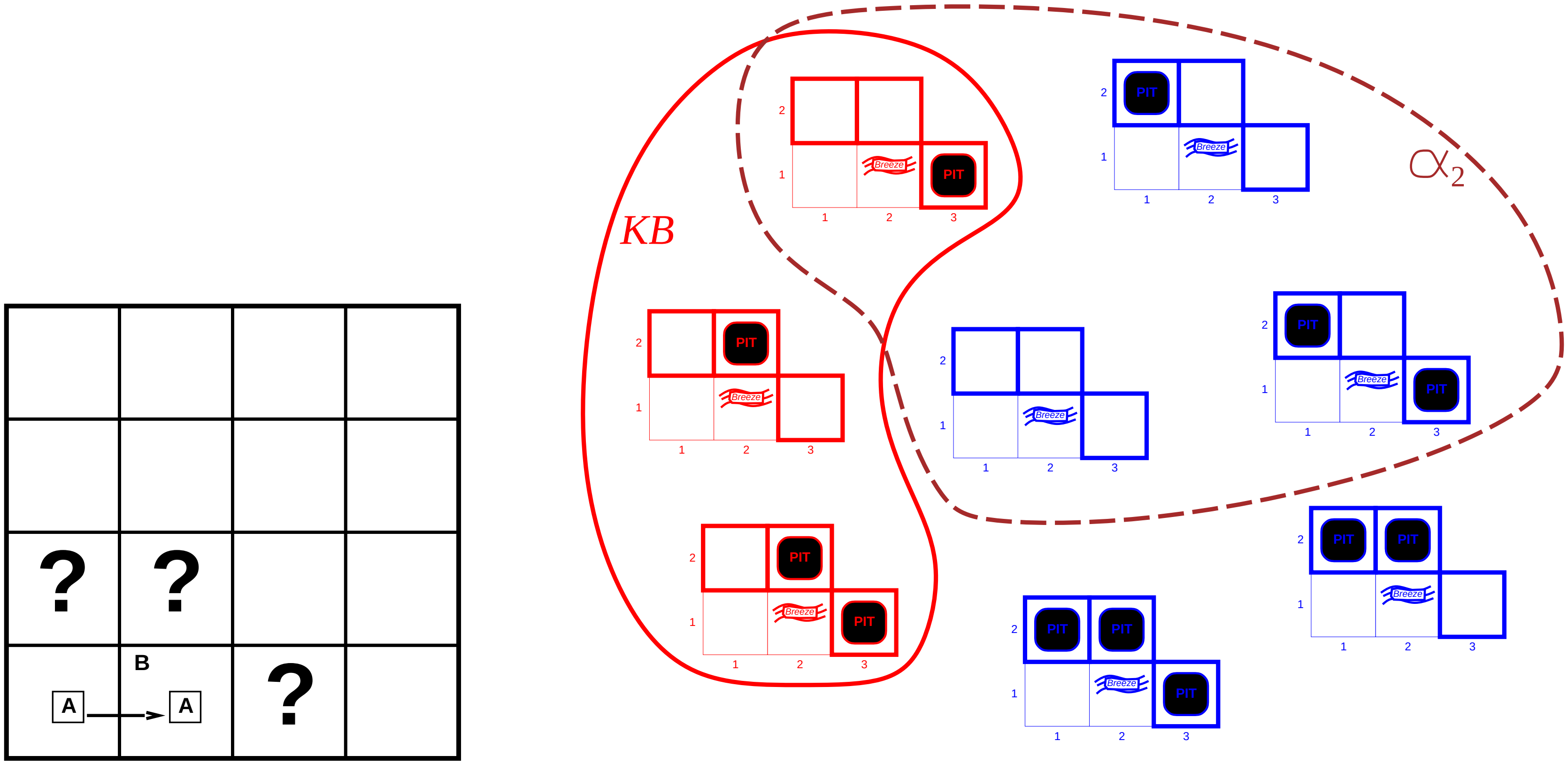
α_1 = "[1,2] is safe", $KB \models \alpha_1$, proved by *model checking*

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 $KB \vdash_i \alpha$

Model checking (what we just did) is one kind of inference procedure,
but not the only one

Model checking is sound
Model checking is complete if the set of all possible models is finite

Preview of where we're going

Later, we'll 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.

But first, let's look at propositional logic.

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 a true/false value for every proposition symbol

E.g. $P_{1,2}$ $P_{2,2}$ $P_{3,1}$
$true$ $true$ $false$
(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

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"

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

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

Truth tables for inference

$B_{1,1}$	$B_{2,1}$	$P_{1,1}$	$P_{1,2}$	$P_{2,1}$	$P_{2,2}$	$P_{3,1}$	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

Model checking in propositional logic = inference using truth tables

Each row is a potential model: an assignment of truth values to symbols

if KB is true in row, is α true 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

  $symbols \leftarrow$ a list of the proposition symbols in KB and α
  return TT-CHECK-ALL($KB, \alpha, symbols, [\,]$)

---

function TT-CHECK-ALL($KB, \alpha, symbols, model$) **returns** *true* or *false*
  if EMPTY?($symbols$) **then**
    if PL-TRUE?($KB, model$) **then return** PL-TRUE?($\alpha, model$)
    else return *true*
  else do
    $P \leftarrow$ FIRST($symbols$); $rest \leftarrow$ REST($symbols$)
    return TT-CHECK-ALL($KB, \alpha, rest$, EXTEND($P, true, model$)) **and**
      TT-CHECK-ALL($KB, \alpha, rest$, 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 $\beta \models \alpha$

$(\alpha \land \beta)$	$\equiv$	$(\beta \land \alpha)$	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., $True$, $A \lor \neg A$, $A \Rightarrow A$, $(A \land (A \Rightarrow B)) \Rightarrow B$

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 **at least one** 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*

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

Forward and backward chaining

Horn Form

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

Horn clause =

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

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

This is a restricted subset of propositional logic
e.g., the following is not a Horn clause, and can't be translated into one:
$A \vee B$

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

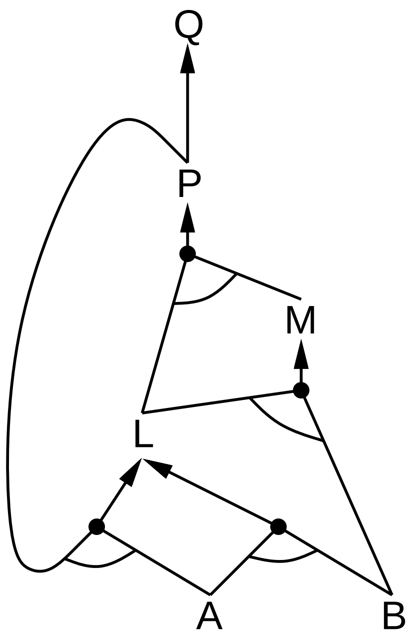
$$\frac{\alpha_1, \ldots, \alpha_n, \qquad \alpha_1 \wedge \cdots \wedge \alpha_n \Rightarrow \beta}{\beta}$$

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

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 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(c), number of c's premises not yet inferred
                    inferred(c), whether or not c has been inferred
                    agenda, {all clauses that are ready to be inferred}

   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

$P \Rightarrow Q$

$L \wedge M \Rightarrow P$

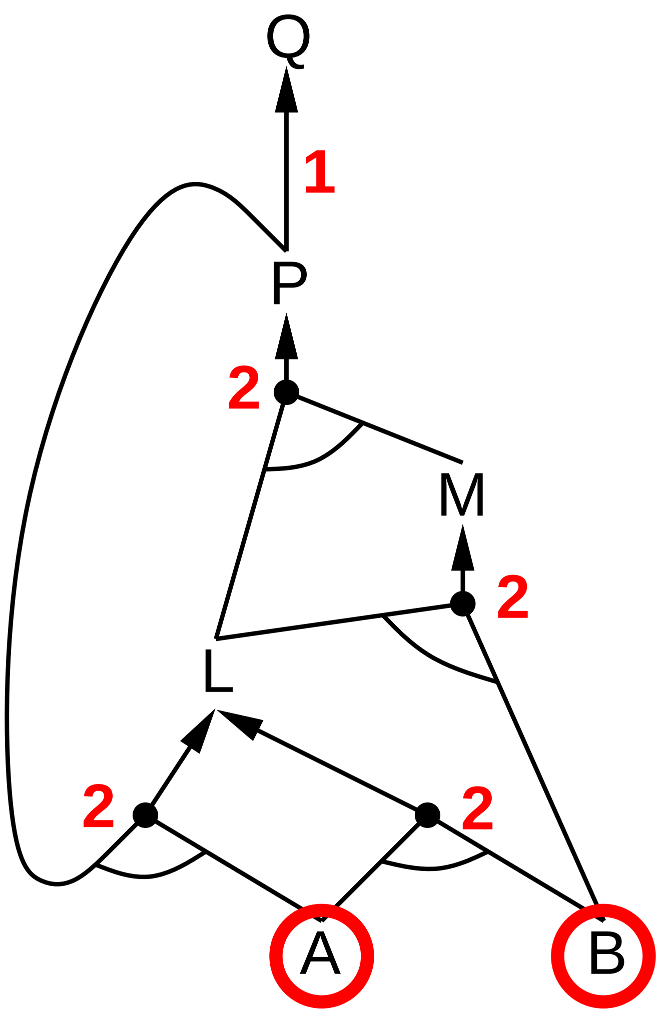
$B \wedge L \Rightarrow M$

$A \wedge P \Rightarrow L$

$A \wedge B \Rightarrow L$

A

B



Forward chaining example

$P \Rightarrow Q$

$L \wedge M \Rightarrow P$

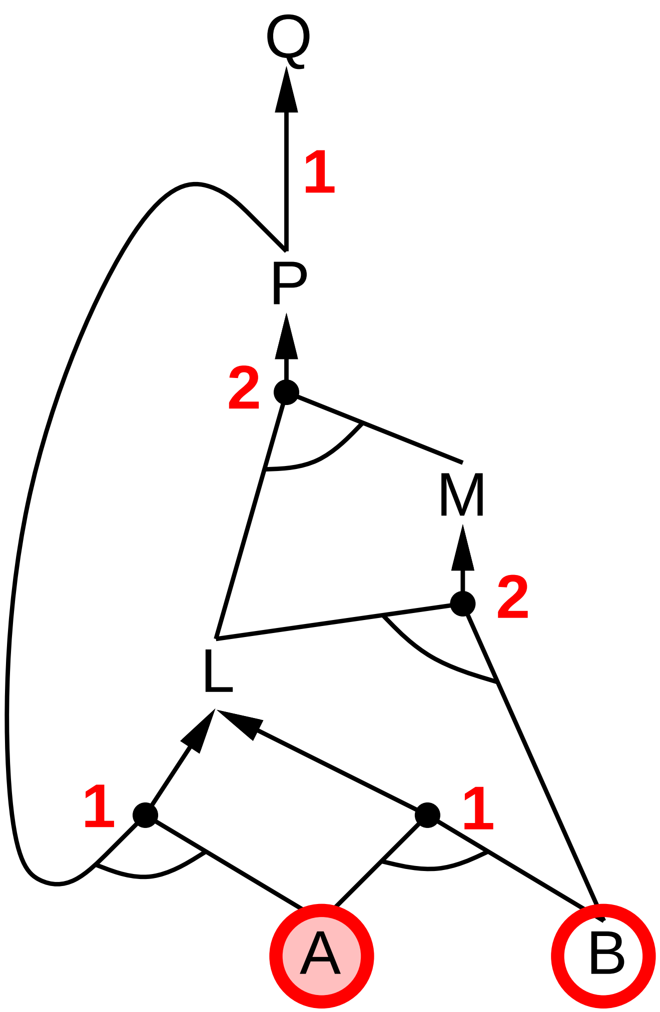
$B \wedge L \Rightarrow M$

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

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

$\mathbf{A}$

$\mathbf{B}$



Forward chaining example

$P \Rightarrow Q$

$L \wedge M \Rightarrow P$

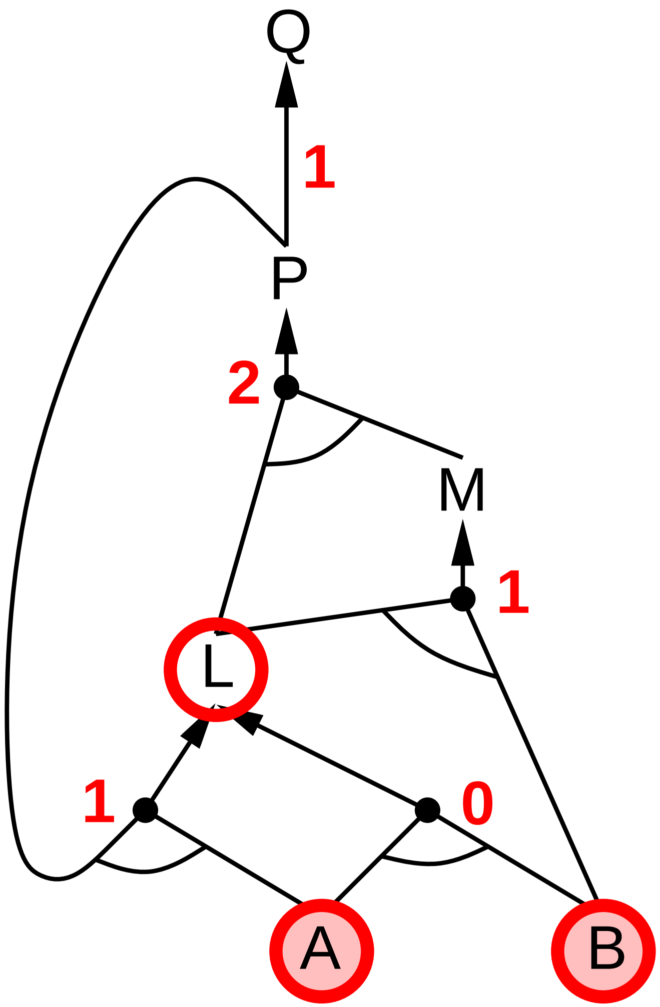
$\mathrm{B} \wedge L \Rightarrow M$

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

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

$\mathbf{A}$

B



Forward chaining example

$P \Rightarrow Q$

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

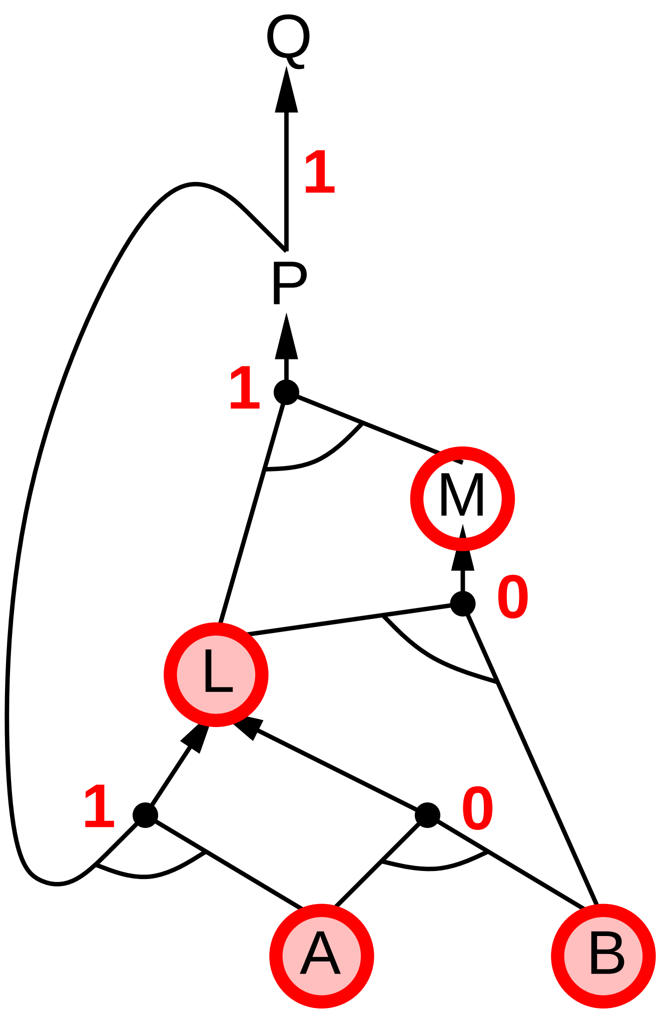
$\mathbf{B} \wedge \mathbf{L} \Rightarrow \mathbf{M}$

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

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

$\mathbf{A}$

$\mathbf{B}$



Forward chaining example

$P \Rightarrow Q$

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

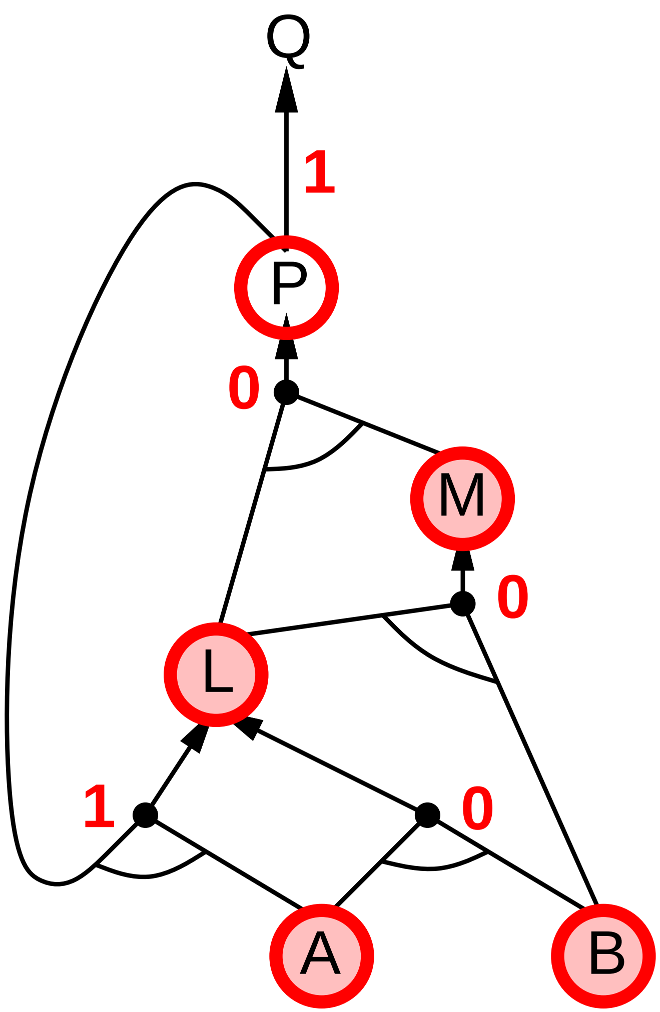
$\mathbf{B} \wedge \mathbf{L} \Rightarrow \mathbf{M}$

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

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

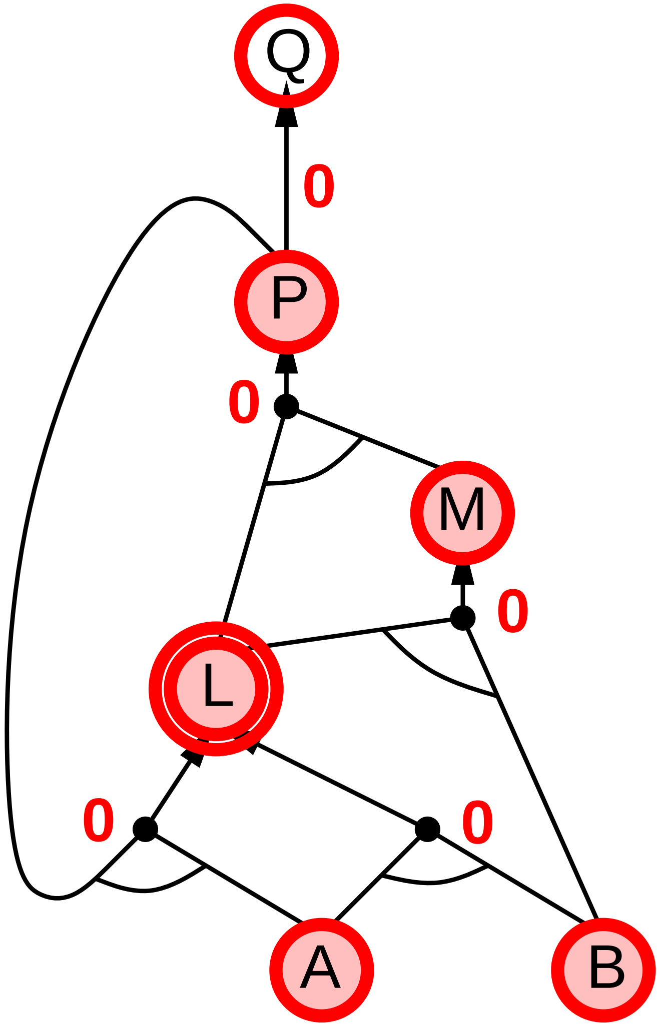
$\mathbf{A}$

$\mathbf{B}$



Forward 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



Forward chaining example

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

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

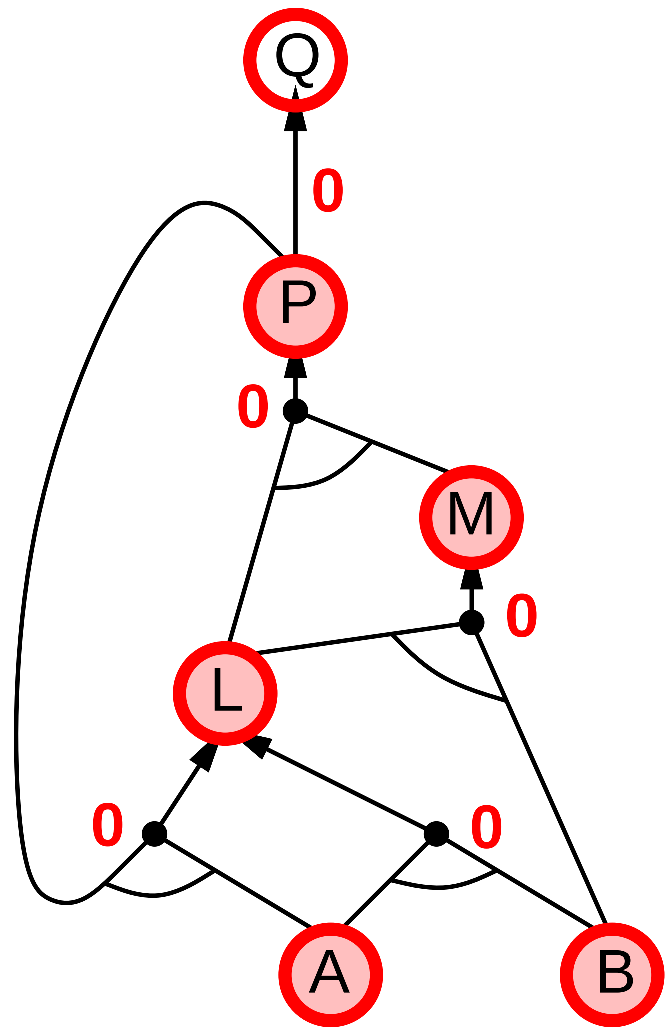
$\mathbf{B} \wedge \mathbf{L} \Rightarrow \mathbf{M}$

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

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

$\mathbf{A}$

$\mathbf{B}$



Forward chaining example

$P \Rightarrow Q$

$L \wedge M \Rightarrow P$

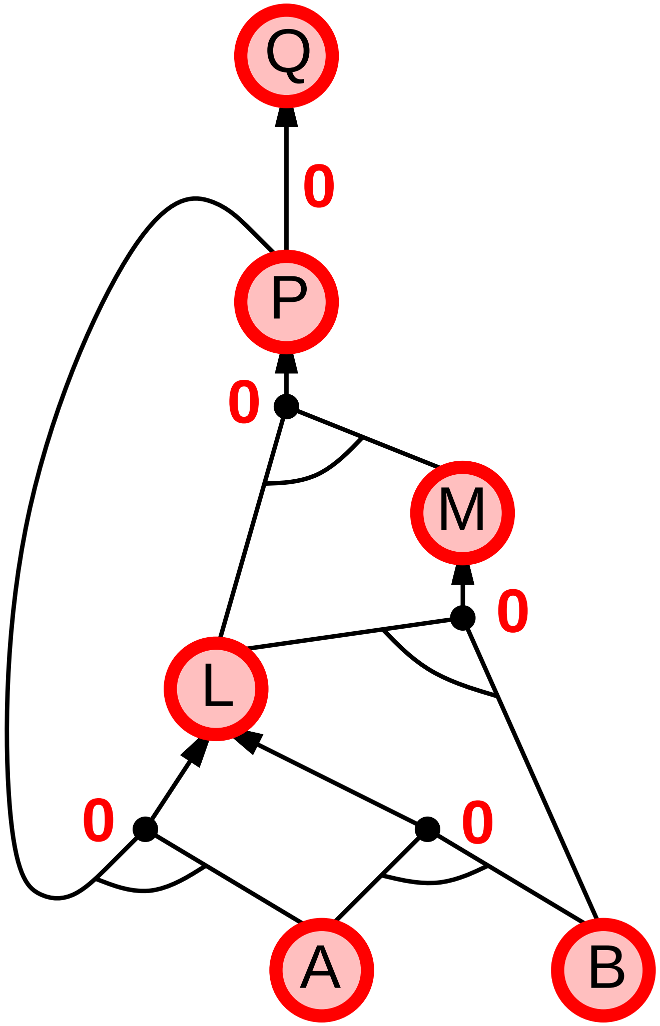
$B \wedge L \Rightarrow M$

$A \wedge P \Rightarrow L$

$A \wedge B \Rightarrow L$

A

B



Backward chaining

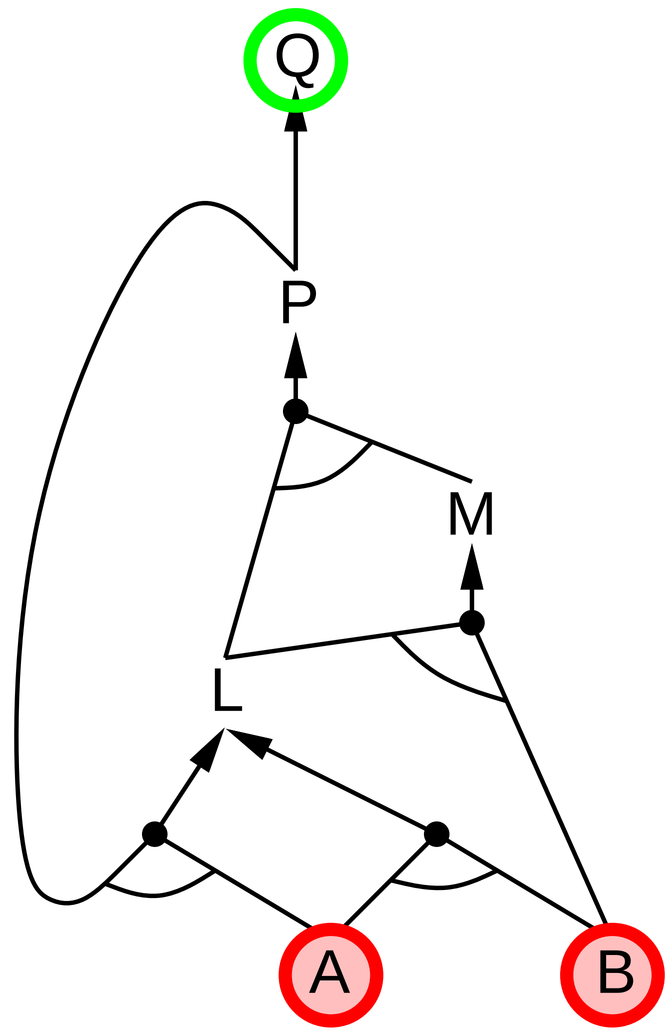
Idea: work backwards from the query q:
to prove q by BC,
check if q is known already, or
recursively call BC to prove all premises of some rule concluding q

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

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

$P \Rightarrow Q$

$L \wedge M \Rightarrow P$

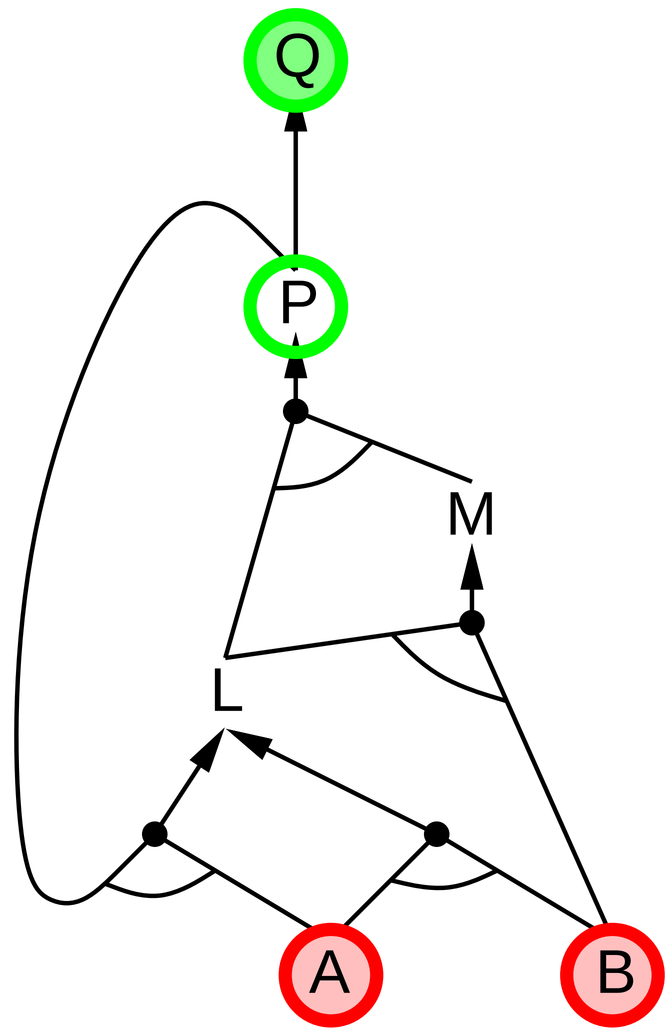
$B \wedge L \Rightarrow M$

$A \wedge P \Rightarrow L$

$A \wedge B \Rightarrow L$

A

B



Backward chaining example

$P \Rightarrow Q$

$L \wedge M \Rightarrow P$

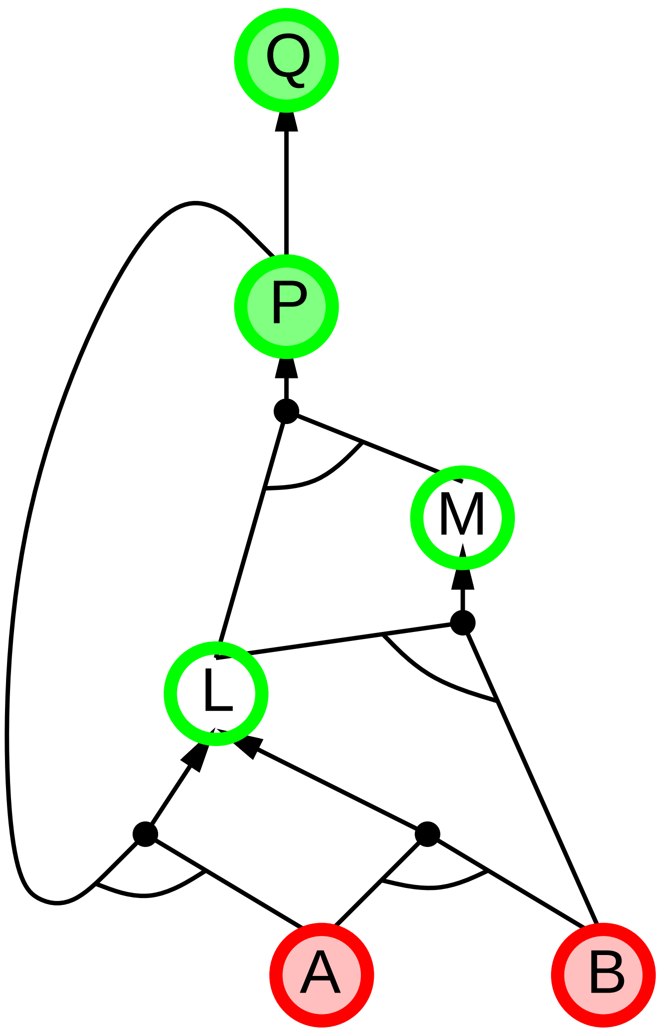
$B \wedge L \Rightarrow M$

$A \wedge P \Rightarrow L$

$A \wedge B \Rightarrow L$

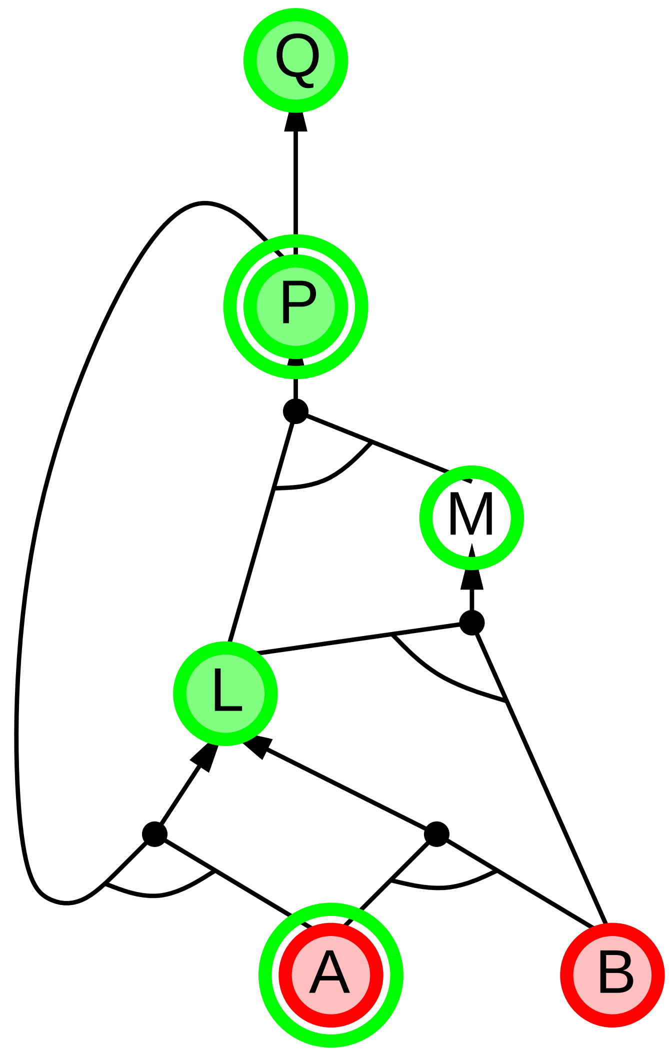
A

B



Backward chaining example

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



Backward chaining example

$P \Rightarrow Q$

$L \wedge M \Rightarrow P$

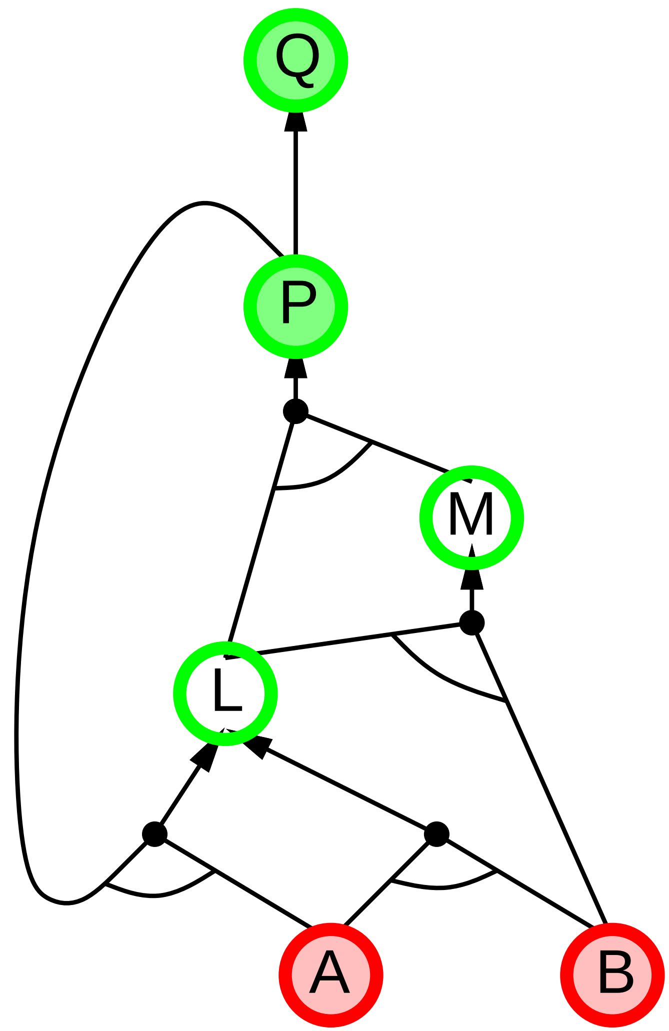
$B \wedge L \Rightarrow M$

$A \wedge P \Rightarrow L$

$A \wedge B \Rightarrow L$

A

B



Backward chaining example

$P \Rightarrow Q$

$L \wedge M \Rightarrow P$

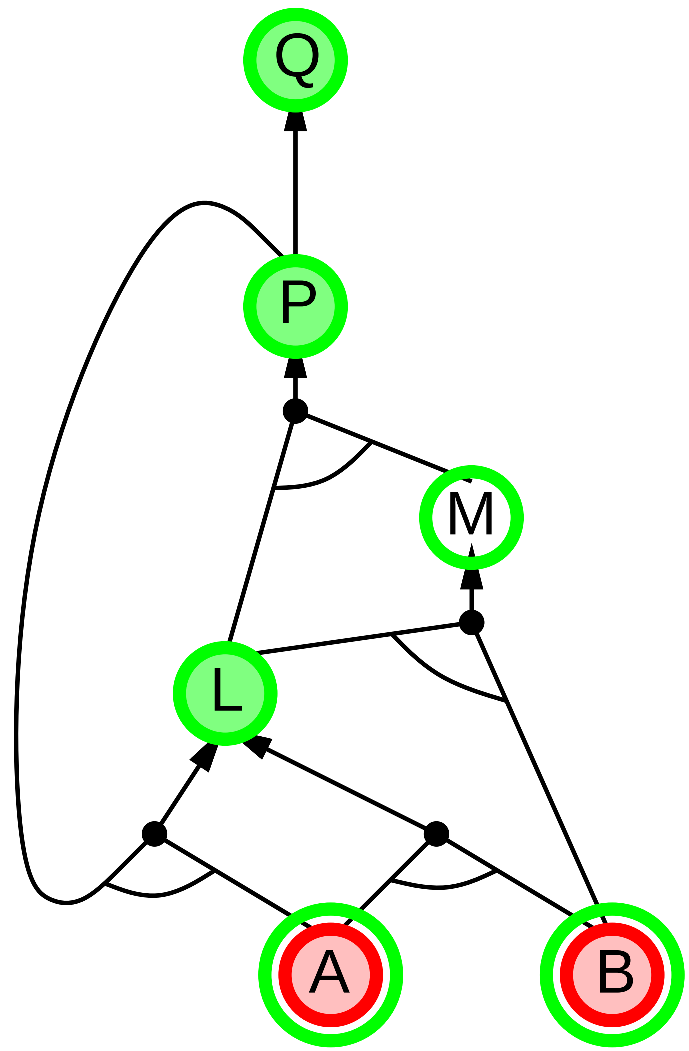
$B \wedge L \Rightarrow M$

$A \wedge P \Rightarrow L$

$A \wedge B \Rightarrow L$

A

B



Backward chaining example

$P \;\Rightarrow\; Q$

$L \wedge M \;\Rightarrow\; P$

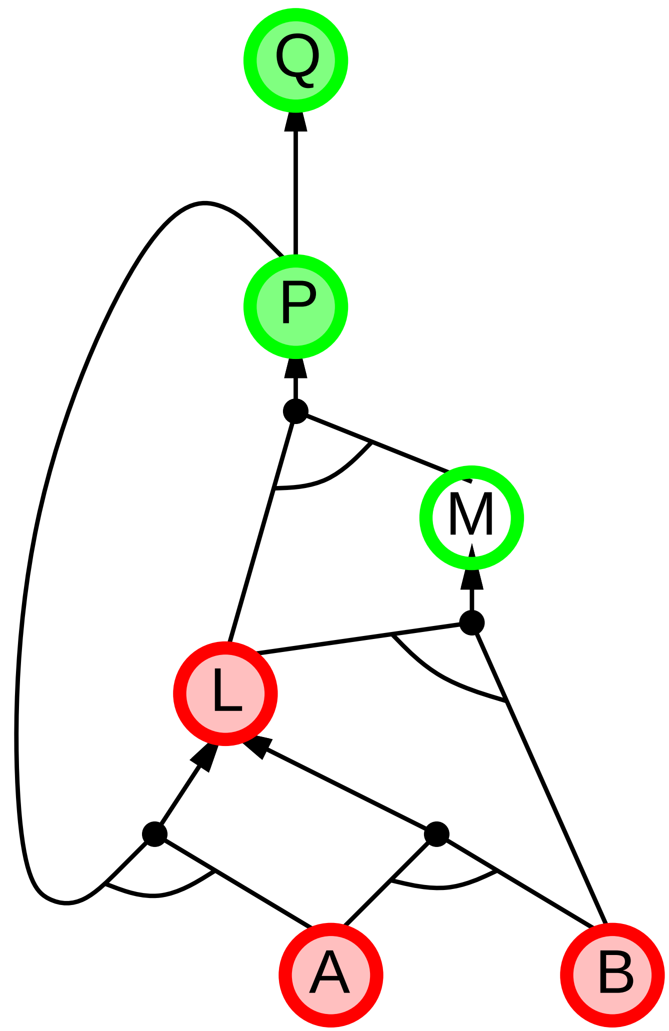
$B \wedge L \;\Rightarrow\; M$

$A \wedge P \;\Rightarrow\; L$

$A \wedge B \;\Rightarrow\; L$

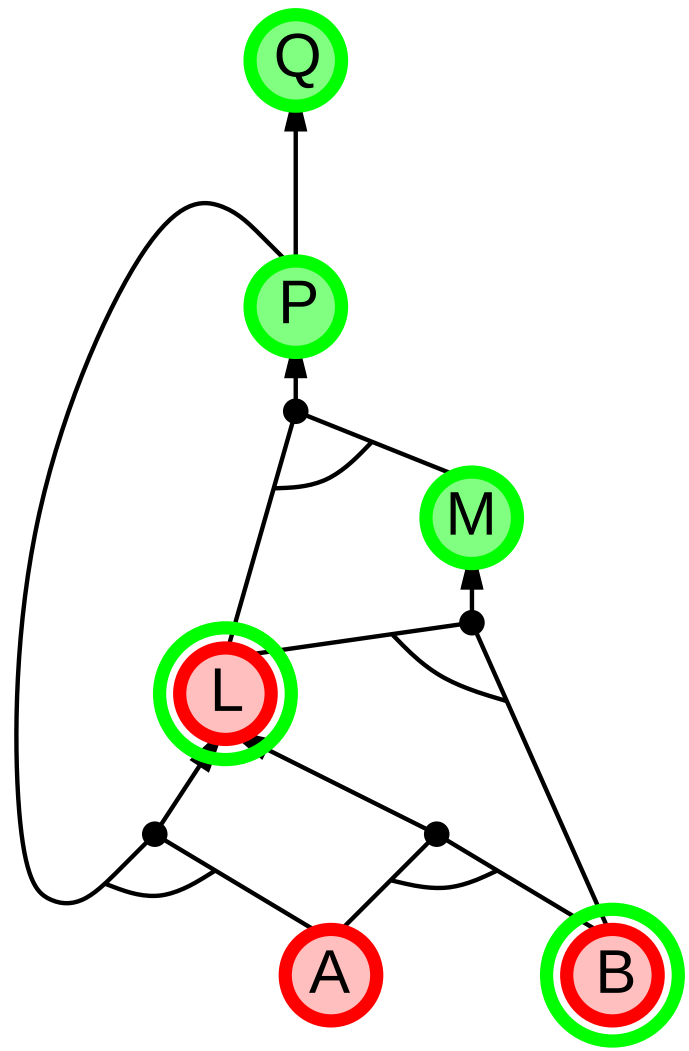
A

B



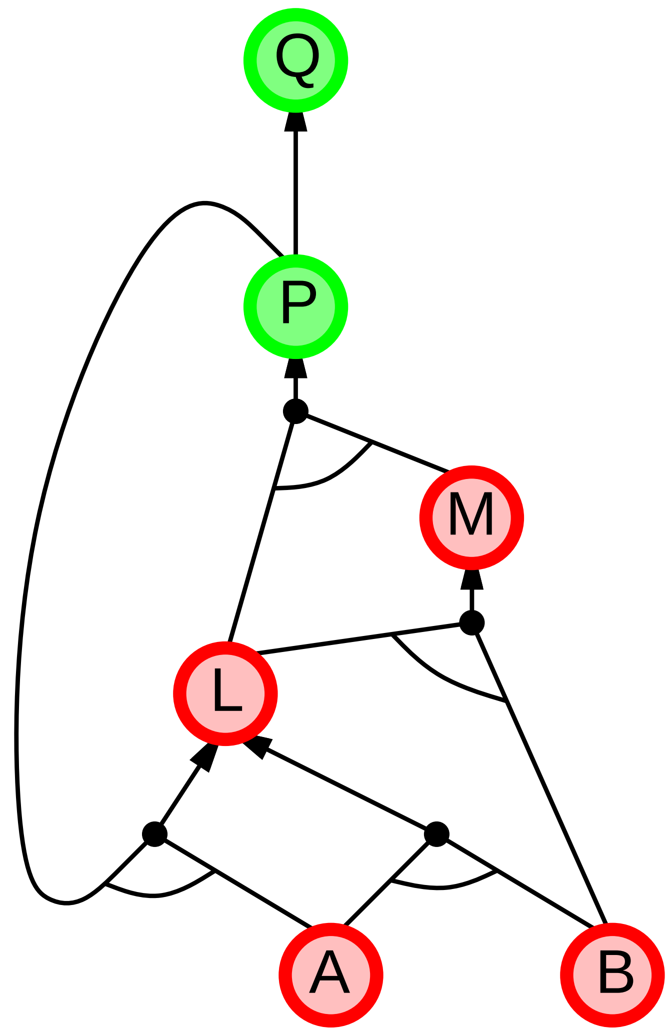
Backward chaining example

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



Backward chaining example

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



Backward chaining example

$P \Rightarrow Q$

$L \wedge M \Rightarrow P$

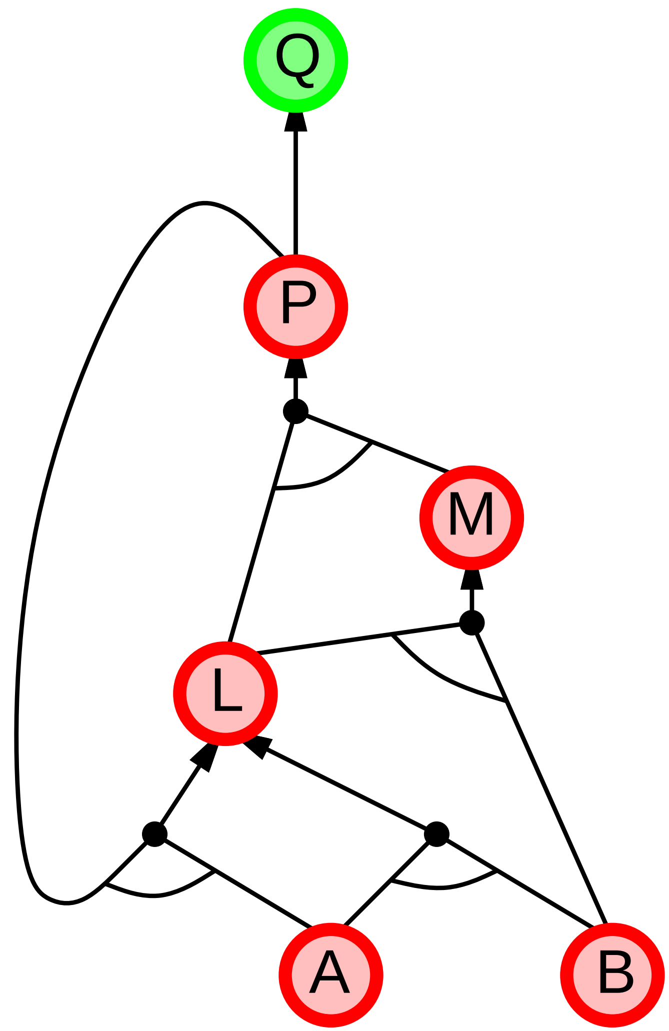
$B \wedge L \Rightarrow M$

$A \wedge P \Rightarrow L$

$A \wedge B \Rightarrow L$

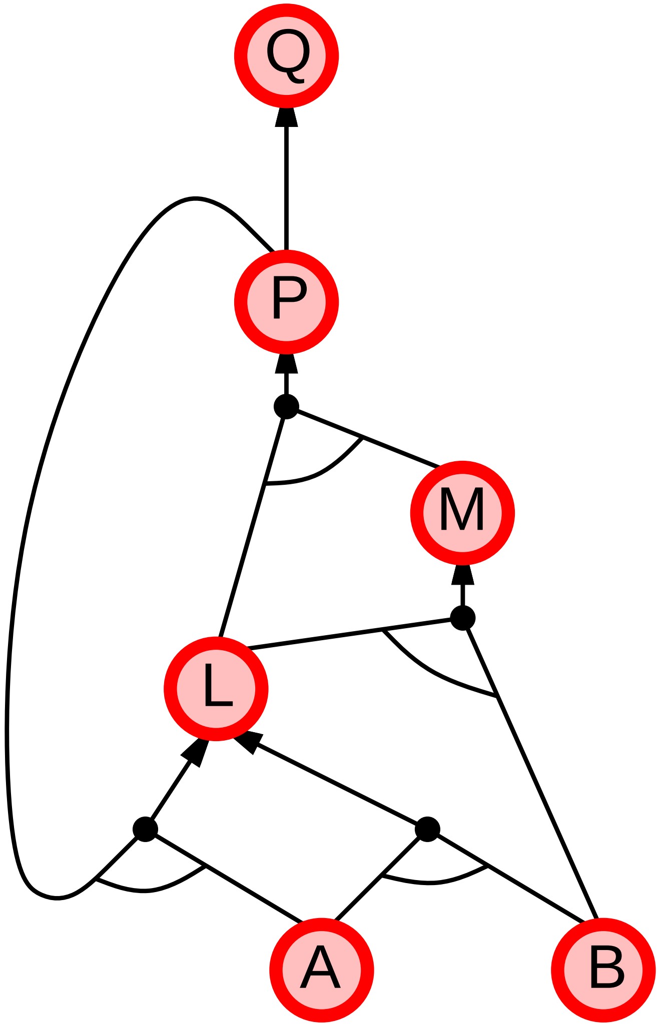
A

B



Backward chaining example

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



Forward 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 a PhD program?

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

Resolution

Conjunctive Normal Form (CNF): **conjunction** of $\underbrace{\textbf{disjunctions}\text{ of }\textbf{literals}}_{\textit{clauses}}$

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

Resolution inference rule:

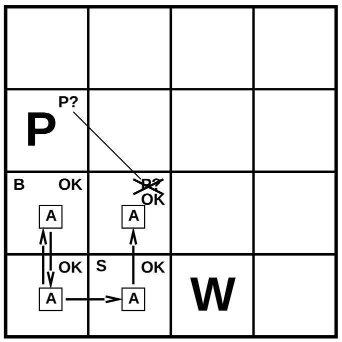
If ℓ_i and m_j are negations of each other,

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

i.e., the disjunct of everything other than ℓ_i and m_j

Example:

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



Resolution

Resolution is equivalent to Modus Ponens:

clauses from CNF:	$A \vee \neg B$	$B \vee \neg C \vee \neg D$
rewrite as implications:	$\neg A \;\Rightarrow\; \neg B$	$\neg B \wedge C \;\Rightarrow\; \neg D$
apply modus ponens:	$\neg A \wedge C \;\Rightarrow\; \neg D$	
rewrite as clauses:	$A \vee \neg C \vee \neg D$	

Resolution is sound and complete for propositional logic

But to use it, you need to convert all your propositional statements to CNF

Conversion to CNF

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: to prove $KB \Rightarrow \alpha$, show $KB \wedge \neg\alpha$ is unsatisfiable

function PL-RESOLUTION(KB, α) **returns** *true* or *false*
  inputs: KB, the knowledge base, a sentence in propositional logic
    α, the query, a sentence in propositional logic

  clauses ← the set of clauses in the CNF representation of $KB \wedge \neg\alpha$
  new ← { }
  loop do
    ;; compute all possible resolvents from *clauses*, and add them to *clauses*
    for each C_i, C_j **in** *clauses* **do**
      resolvents ← PL-RESOLVE(C_i, C_j)
      if *resolvents* contains the empty clause **then return** *true*
      new ← *new* ∪ *resolvents*
    if *new* ⊆ *clauses* **then return** *false*
    clauses ← *clauses* ∪ *new*

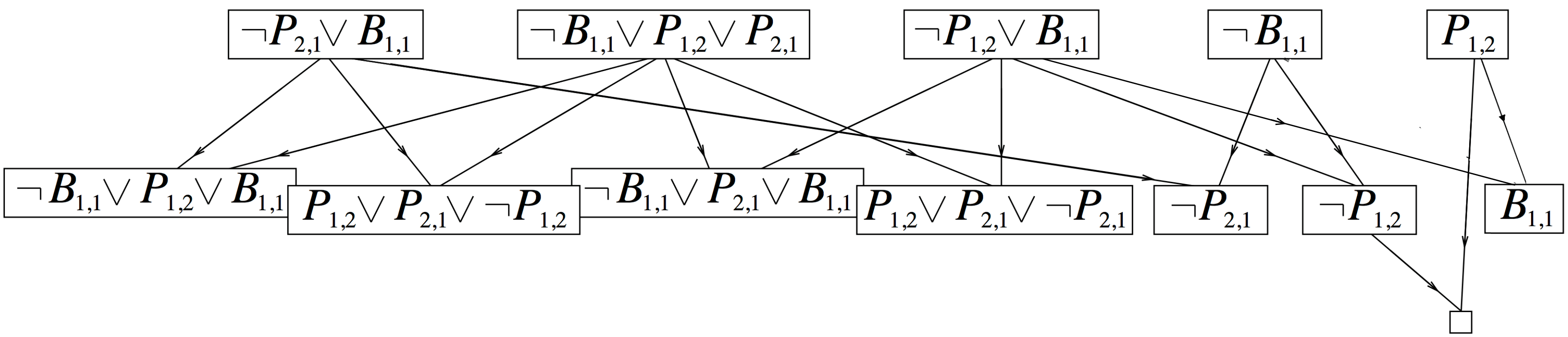
Resolution example

KB: there's a breeze in (1,1) iff there's a pit in (1,2) or (2,1);
and there's no breeze in (1,1)

$$\begin{aligned} KB &= (B_{1,1} \;\Leftrightarrow\; (P_{1,2} \lor P_{2,1})) \land \neg B_{1,1} \\ &= (\neg B_{1,1} \lor P_{1,2} \lor P_{2,1}) \land (\neg P_{1,2} \lor B_{1,1}) \land (\neg P_{2,1} \lor B_{1,1}) \land \neg B_{1,1} \end{aligned}$$

$\alpha = \neg P_{1,2}$ we want to show there's no pit in (1,2)

$\neg\alpha = P_{1,2}$ suppose there is one (for proof by contradiction)



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

Propositional logic lacks expressive power