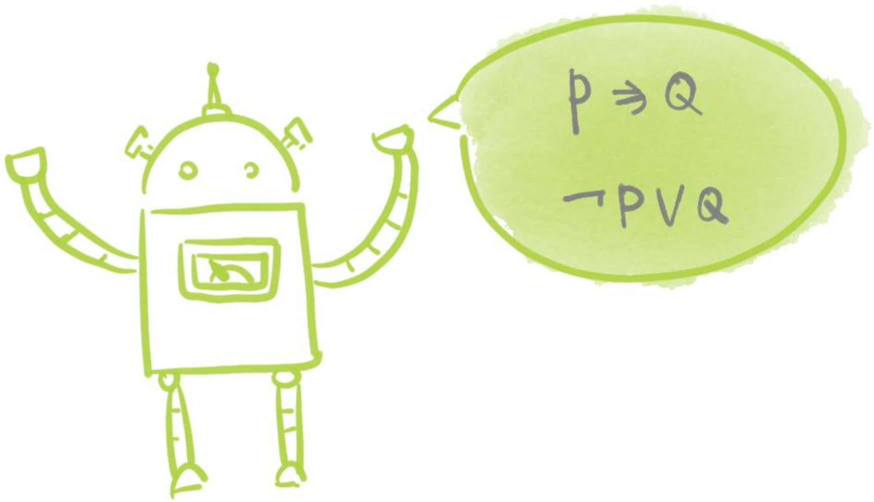
CS5491: Artificial Intelligence

Logical Agents

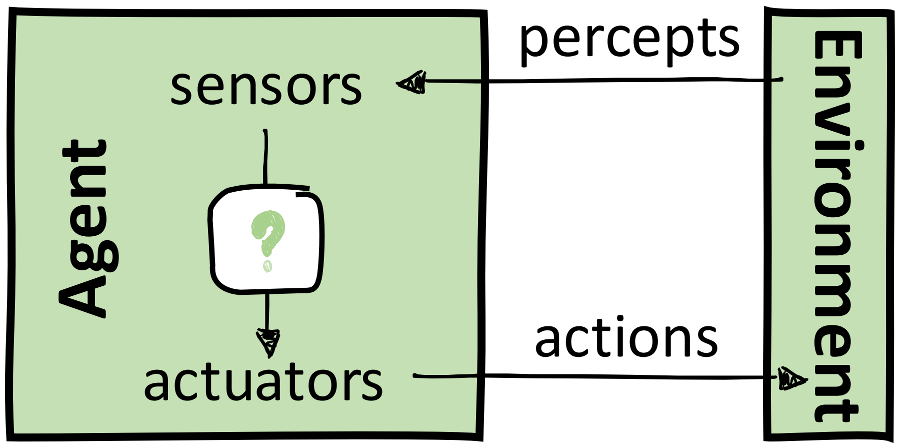


Instructor: Kai Wang

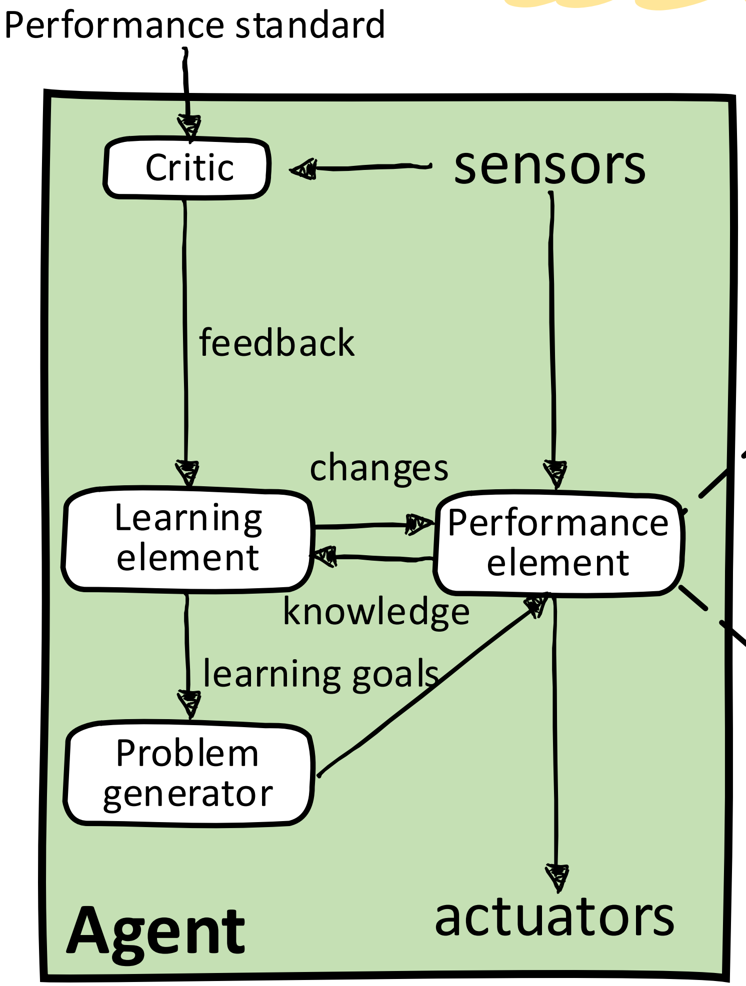
Recap: Agent

An agent is anything that can be viewed as perceiving its environment through sensors and acting upon that environment through actuators.

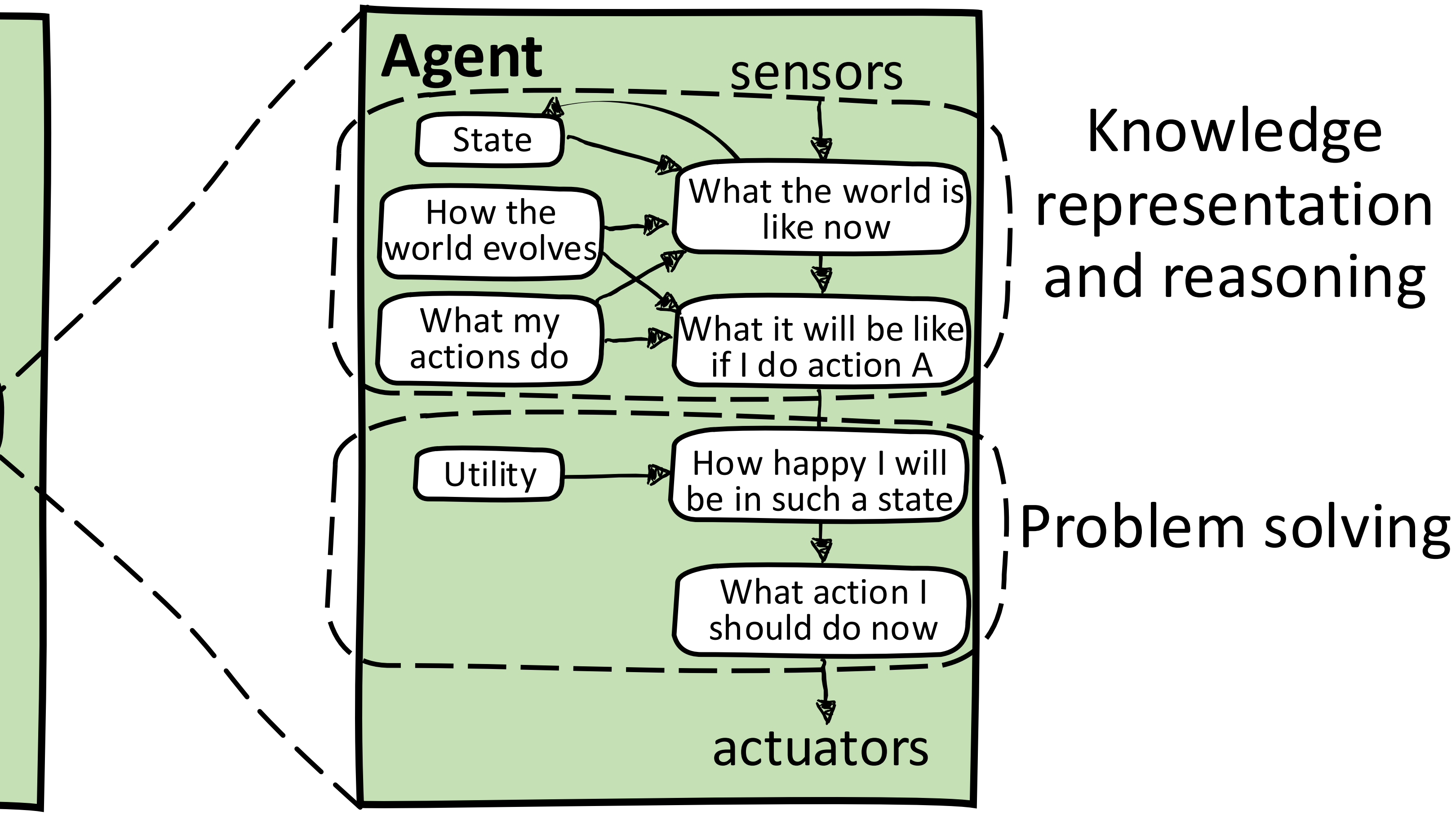
Agent = architecture + program



Recap: The Big Picture



Agents with learning



Agents without learning

The Big Idea

Current Agents

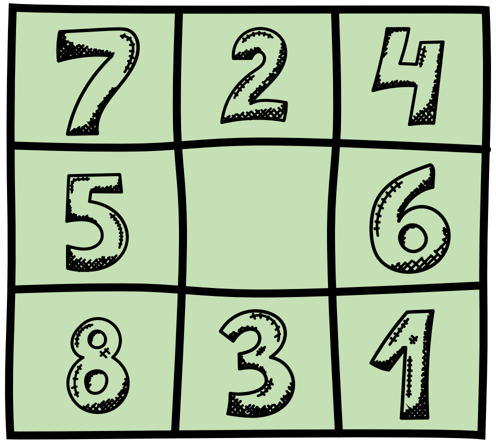
- Very limited / inflexible knowledge
- Knowledge encoded in
  - Successor functions
  - Heuristics
  - Performance measures
  - Goal tests

Reasoning Agents

- Combine information
- Adapt to new tasks
- Learn about environment
- Update in response to environmental changes

The Big Idea

- Can it predict outcomes of future actions?
- Can it conclude that a state is unreachable?
- Can it prove that certain states are always unreachable from others?



Today

Knowledge-based agents

Models and entailment

Propositional logic

Inference and theorem proving

Today

Knowledge-based agents

Models and entailment

Propositional logic

Inference and theorem proving

Knowledge-based Agents

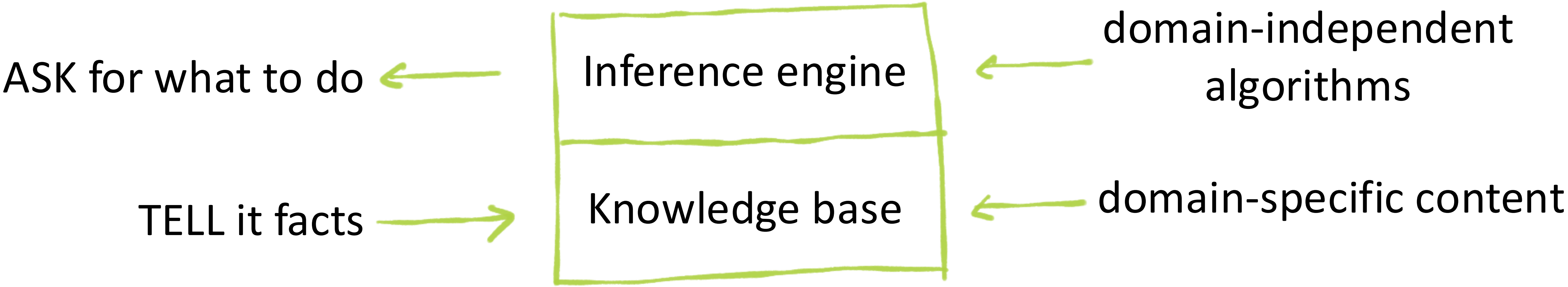
A knowledge base keeps track of things and consists a set of sentences in a formal representation language.

Example:

- TELL: mother of James is Jane
- TELL: Susan is James' sister
- TELL: James's mother is the same as James's sister's mother
- ASK: who is Susan's mother?

Knowledge-based Agents

A knowledge base keeps track of things and consists a set of sentences in a formal representation language.



Knowledge-based Agents

```
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
```

Construct a sentence to incorporate new percepts

↓

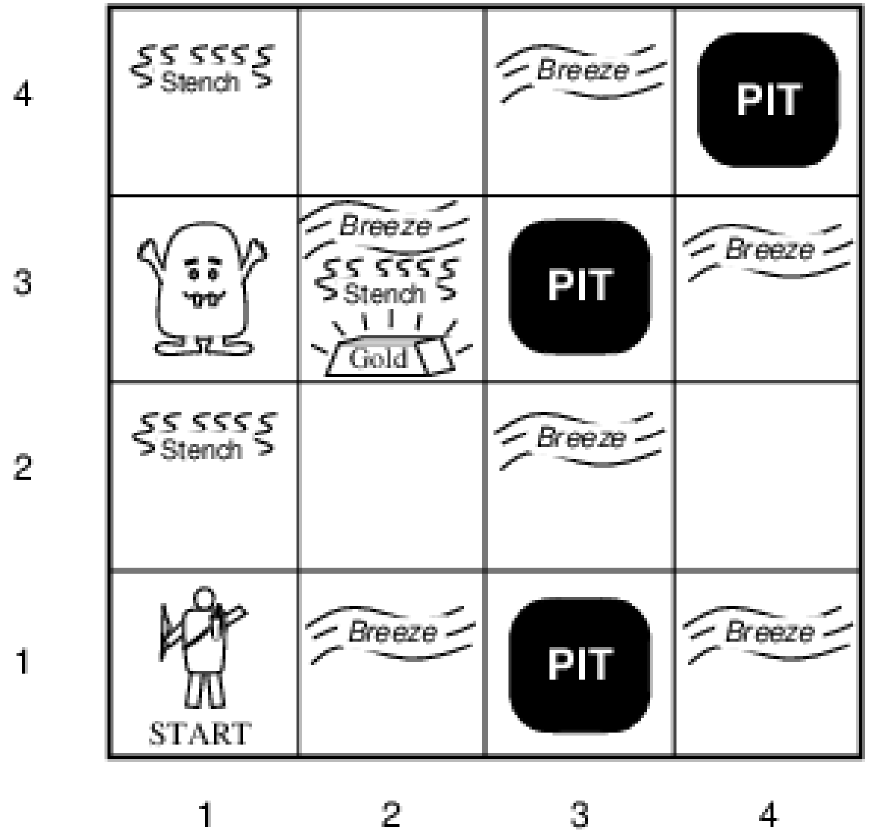
Construct a sentence asking what action is next

↓

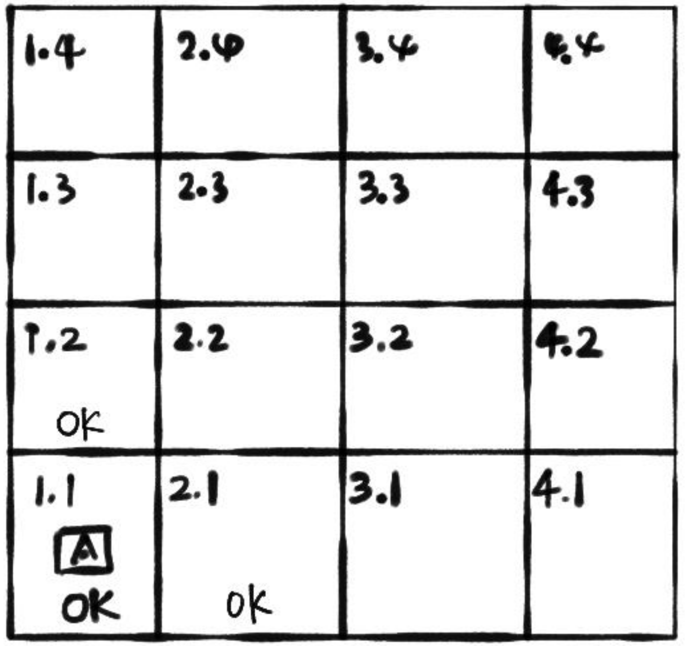
Construct a sentence asserting that action

Wumpus World

- Performance measure
  - + 1000 for gold; -1000 for dying
  - -1 for each 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
- Actuators move forward, left, right, shoot
- Sensors smell, breeze, glitter



Exploring a Wumpus World



A = agent

B = Breeze

G = Glitter, Gold

OK = safe square

P = Pit

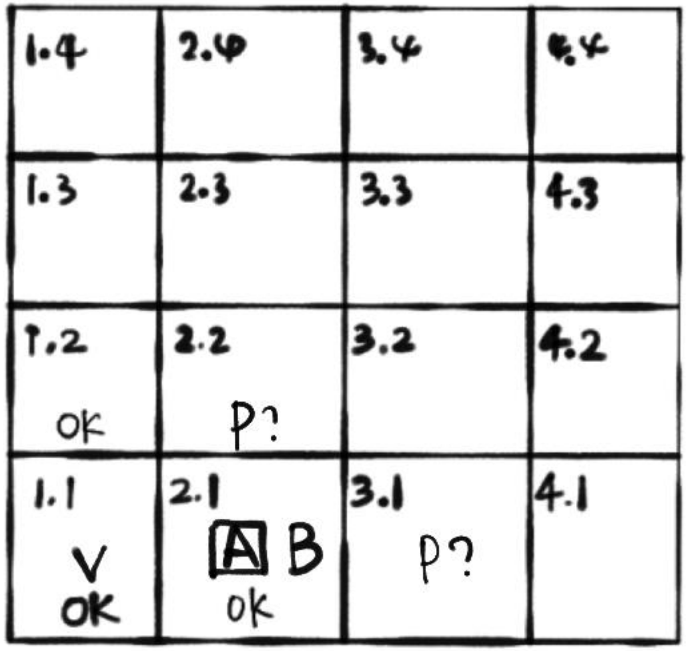
S = Stench

V = Visited

W = Wumpus

Sensor: [none, none, none]

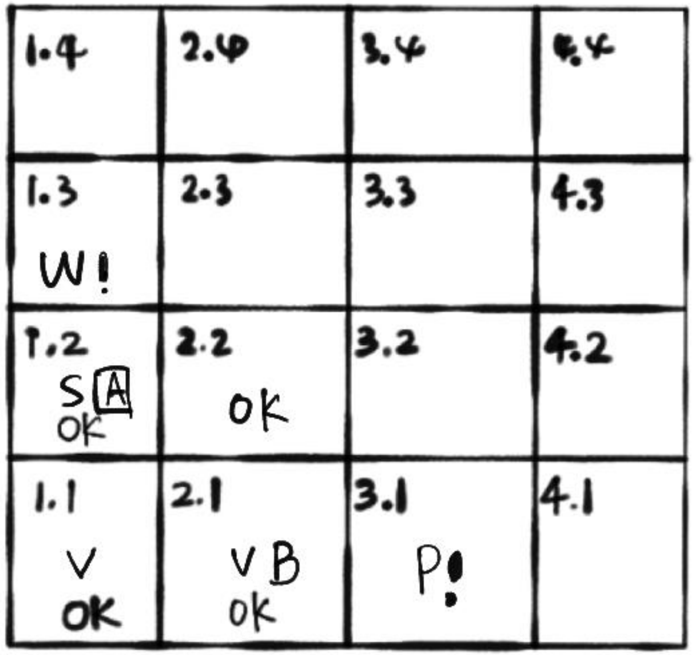
Exploring a Wumpus World



A = agent
B = Breeze
G = Glitter, Gold
OK = safe square
P = Pit
S = Stench
V = Visited
W = Wumpus

Sensor: [none, breeze, none]

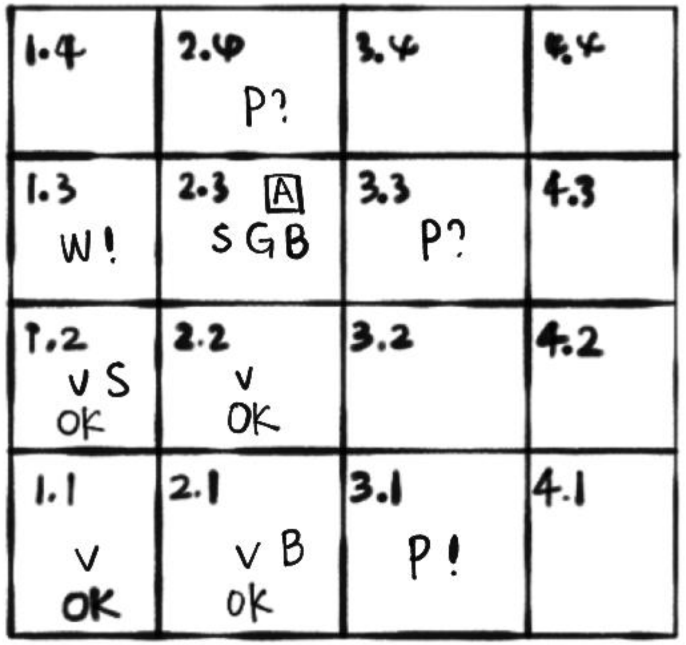
Exploring a Wumpus World



A = agent
B = Breeze
G = Glitter, Gold
OK = safe square
P = Pit
S = Stench
V = Visited
W = Wumpus

Sensor: [stench, none, none]

Exploring a Wumpus World



$\boxed{A}$ = agent
B = Breeze
G = Glitter, Gold
OK = safe square
P = Pit
S = Stench
V = Visited
W = Wumpus

Sensor: [stench, breeze, glitter]

Today

Knowledge-based agents

Models and entailment

Propositional logic

Inference and theorem proving

Basics in Logic

- **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 the truth of a sentence in a world.

Example:

- $x+2 \geq y$ 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 $x=0, y=6$

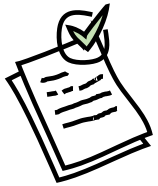
Entailment

Entailment means that a sentence follows from another: $KB \vDash \alpha$. Knowledge base KB entails the sentence α iff α is true in all worlds where KB is true.

Example:

- $(x + y = 4) \vDash (4 = x + y)$
- $(x = 0) \vDash (xy = 0)$
- $(p = True) \vDash (p \vee q)$

Entailment



Clicker question: Which one is right?

False $\models$ True

$(p \wedge q) \models (p \vee q)$

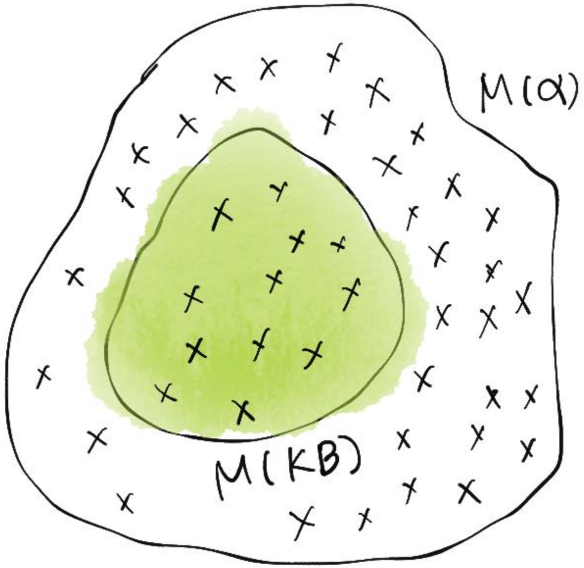
$(x + y > 3) \models (y > 3)$

$(x > y) \models (x > y - 3)$

Models

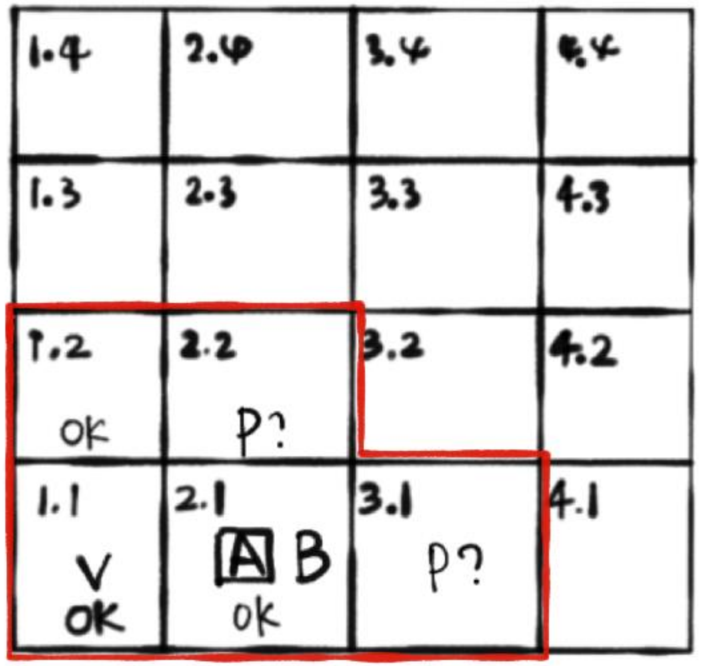
Models are formally structured worlds with respect to which truth can be evaluated. We say m is a model of a sentence α if α is true in m. We denote that $M(\alpha)$ as the set of all models of α.

$KB \vDash \alpha$ iff $M(KB) \subseteq M(\alpha)$.



Entailment in the Wumpus World

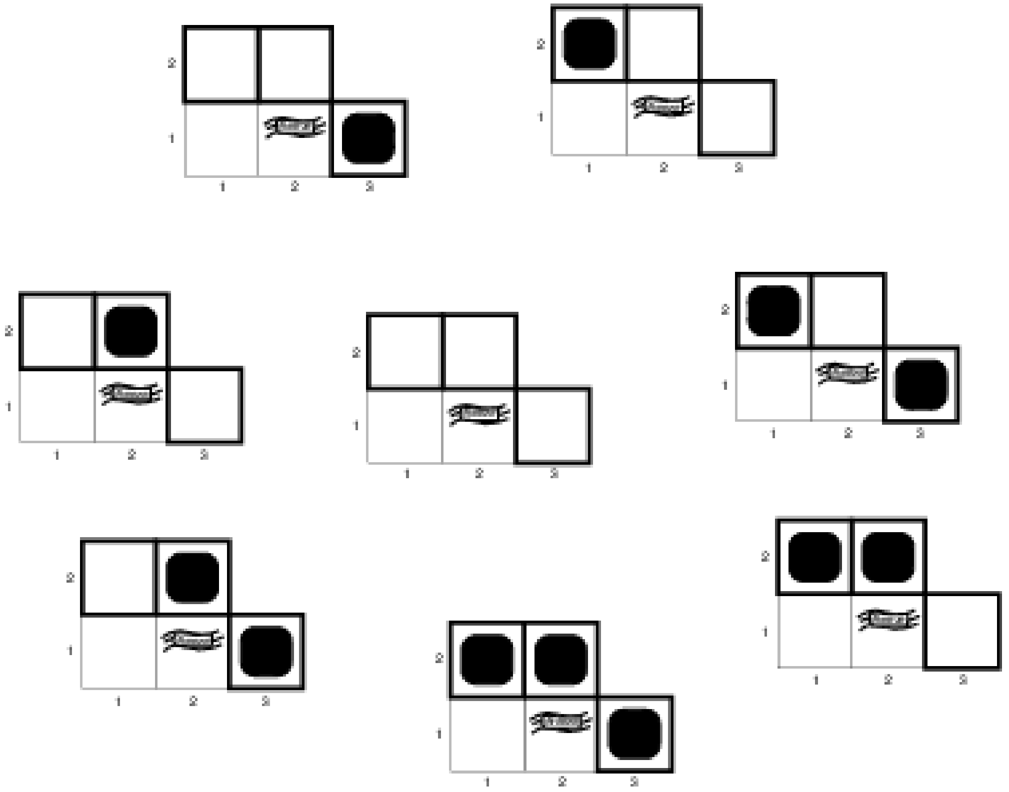
- Model the presence of pits in squares [1,2], [2,2], and [3,1]



A = agent
B = Breeze
G = Glitter, Gold
OK = safe square
P = Pit
S = Stench
V = Visited
W = Wumpus

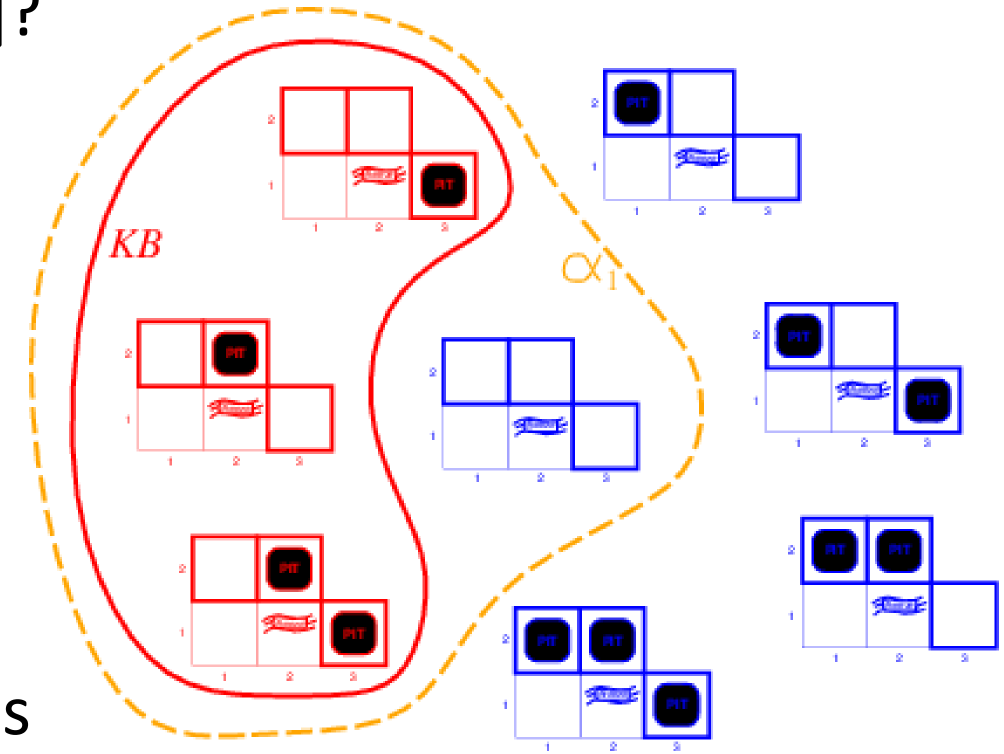
Sensor: [none, breeze, none]

Possible Wumpus Models



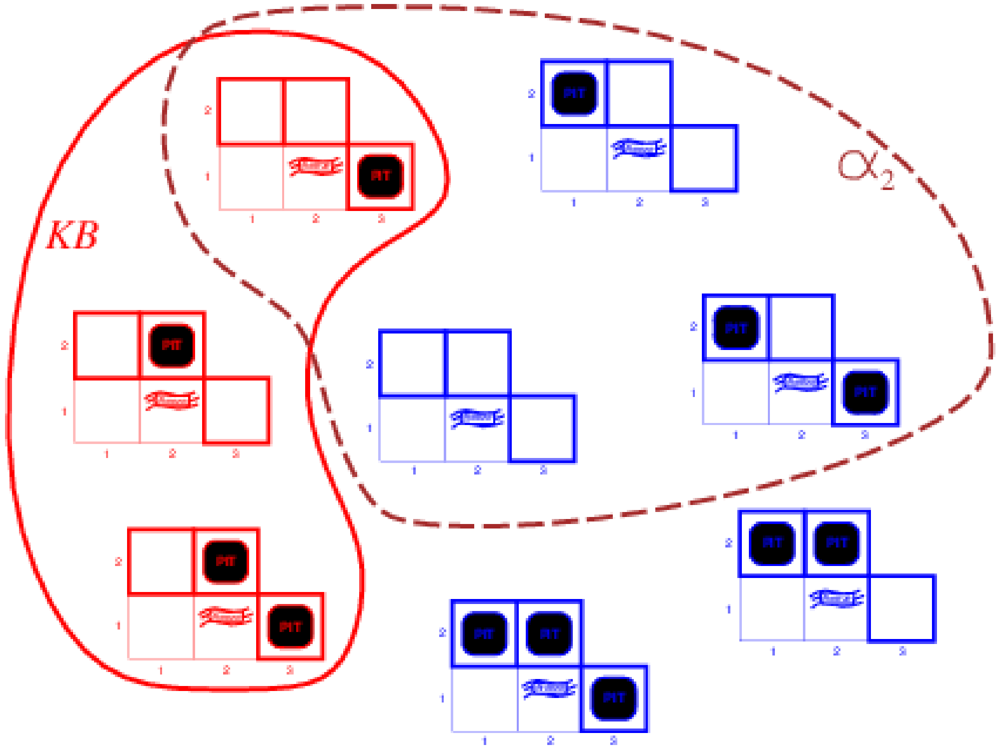
Entailment

- Is it true that there are no pits in [1,2]?
- α_1 = no pits in [1,2]
- This question = $KB \vDash \alpha_1$?
- In every model where KB is true, α_1 is also true.



Not Entailed

- Is it true that there are no pits in [2,2]?
- α_2 = no pits in [2,2]
- This question = $KB \vDash \alpha_2$?
- In every model where KB is true, α_2 is not necessarily true.



Inference

The goal of inference is to decide whether $KB \vDash \alpha$. $KB \vDash_i \alpha$ specifically says α can be derived from KB by procedure i.

Soundness: i is sound if whenever $KB \vDash_i \alpha$, it is also true that $KB \vDash \alpha$.

Completeness: i is complete if whenever $KB \vDash \alpha$, it is also true that $KB \vDash_i \alpha$

Today

Knowledge-based agents

Models and entailment

Propositional logic

Inference and theorem proving

Propositional Logic: Syntax

- The proposition symbols P_1, P_2 etc are sentences.
- Negation: If P is a sentence, $\neg P$ is a sentence.
- Conjunction: If P_1 and P_2 are sentences, $P_1 \wedge P_2$ is a sentence.
- Disjunction: If P_1 and P_2 are sentences, $P_1 \vee P_2$ is a sentence.
- Implication: If P_1 and P_2 are sentences, $P_1 \Rightarrow P_2$ is a sentence.
- Biconditional: If P_1 and P_2 are sentences, $P_1 \Leftrightarrow P_2$ is a sentence.

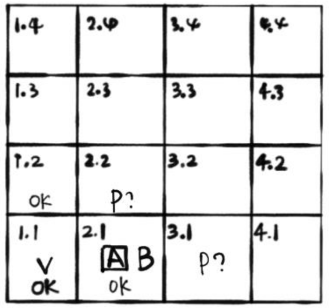
Propositional Logic: Semantics

- Each model specifies true / false for each proposition symbol.
  - E.g., $m = \{P_1 = false, P_2 = true, P_3 = false\}$
- Rules for evaluating truth with respect to a model m:
  - $\neg P$ is true iff P is false
  - $P_1 \wedge P_2$ is true iff P_1 is true and P_2 is true
  - $P_1 \vee P_2$ is true iff P_1 is true or P_2 is true
  - $P_1 \Rightarrow P_2$ is true iff P_1 is false or P_2 is true
    (OR is false iff P_1 is true. and P_2 is false)
  - $P_1 \Leftrightarrow P_2$ is true iff $P_1 \Rightarrow P_2$ is true and $P_2 \Rightarrow P_1$ is true
- Simple recursive process evaluates an arbitrary sentence.
  - E.g., $\neg P_1 \wedge (P_2 \vee P_3) = true \wedge (true \vee false) = true \wedge true = true$

Truth Tables

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*

Revisiting the Wumpus World



Symbols

- P_{ij} is true if there is a pit in $[i, j]$
- W_{ij} is true if there is a Wumpus in $[i, j]$
- B_{ij} is true if the agent perceives a breeze in $[i, j]$
- S_{ij} is true if the agent perceives a stench in $[i, j]$

Knowledge Base

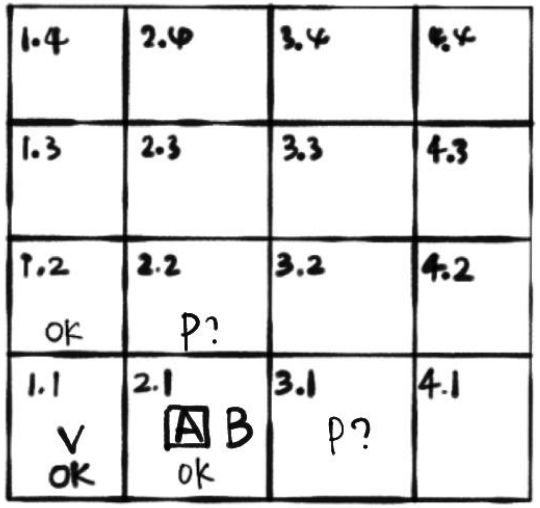
- Wumpus world in general:
  - $R_1: \neg P_{1,1}$
  - $R_2: B_{1,1} \Leftrightarrow \left(P_{1,2} \vee P_{2,1}\right)$
  - $R_3: B_{2,1} \Leftrightarrow \left(P_{1,1} \vee P_{2,2} \vee P_{3,1}\right)$
- After visiting [1,2] and [2,1]:
  - $R_4: \neg B_{1,1}$
  - $R_5: B_{21}$
- $KB = R_1 \wedge R_2 \wedge R_3 \wedge R_4 \wedge R_5$

Inference for the Wumpus World

Goal: Whether my knowledge base says there is no pit in [1,2]?

Denote the sentence $\alpha = \neg P_{1,2}$.

That is, does $KB \vDash \alpha$?



Logical Equivalence

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

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

Validity and Satisfiability

- A sentence is valid if it is true in all models,
  - E.g., $True,\ A \lor \neg A,\ A \Rightarrow A,\ \left(A \land (A \Rightarrow B)\right) \Rightarrow B$
- A sentence is satisfiable if it is true in some models.
  - E.g., $A \lor B,\ C$
- A Sentence is unsatisfiable if it is true in no models.
  - E.g., $A \land \neg A$

$KB \vDash \alpha$ if and only if $KB \Rightarrow \alpha$ is valid.
$KB \vDash \alpha$ if and only if $KB \land \neg\alpha$ is unsatisfiable.

Today

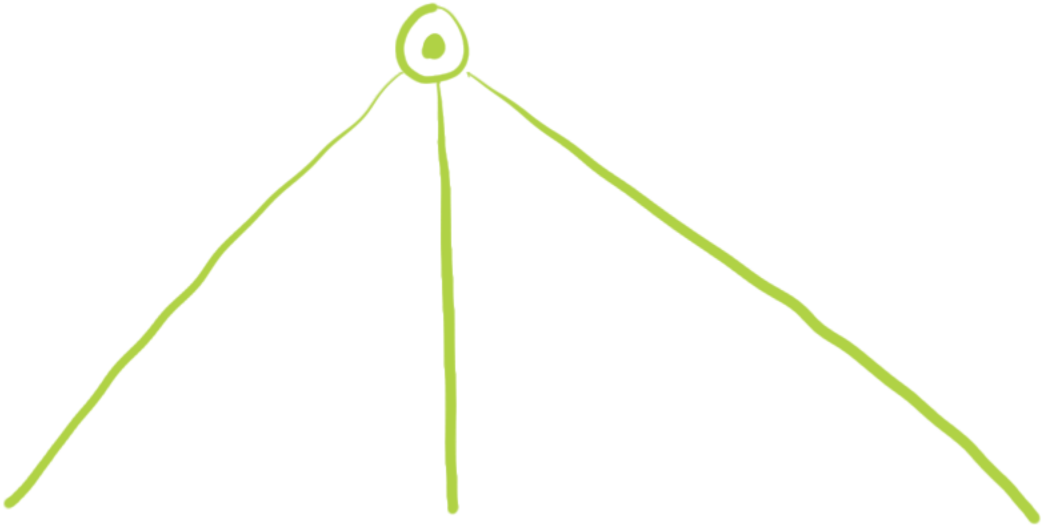
Knowledge-based agents

Models and entailment

Propositional logic

Inference and theorem proving

Inference Methods



Enumeration

Forward and Backward Chaining

Resolution

Enumeration

- Enumerate the models

- For each model, check if what is true in KB has to be true in α.

- In Wumpus, we have 7 relevant symbols and $2^7 = 128$ models.
$B_{1,1}, B_{2,1}, P_{1,1}, P_{1,2}, P_{2,1}, P_{2,2}, P_{3,1}$

Enumeration

- Does $KB \vDash \neg P_{1,2}$?
- Does $KB \vDash \neg P_{1,1}$?

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

Enumeration

- Depth-first enumeration of all models is sound and complete.
- Space complexity: $O(n)$ for n symbols.
- Time complexity: $O(2^n)$

```
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))
```

Forward and Backward Chaining

- Horn Form
  - KB = conjunction of Horn clauses
- Horn clause: at least one literal is positive.
  - E.g., $(\neg P \vee \neg Q \vee V)$ and $(\neg P \vee Q \vee V)$ are Horn clauses.
  - E.g., $(\neg P \vee \neg W)$ is not a Horn clause.
- Horn clauses can be re-written as implications.
  - E.g., $(\neg P \vee \neg Q \vee V)$ becomes $(P \wedge Q \Rightarrow V)$
- Modus Ponens for Horn KB (used for forward / backward chaining):

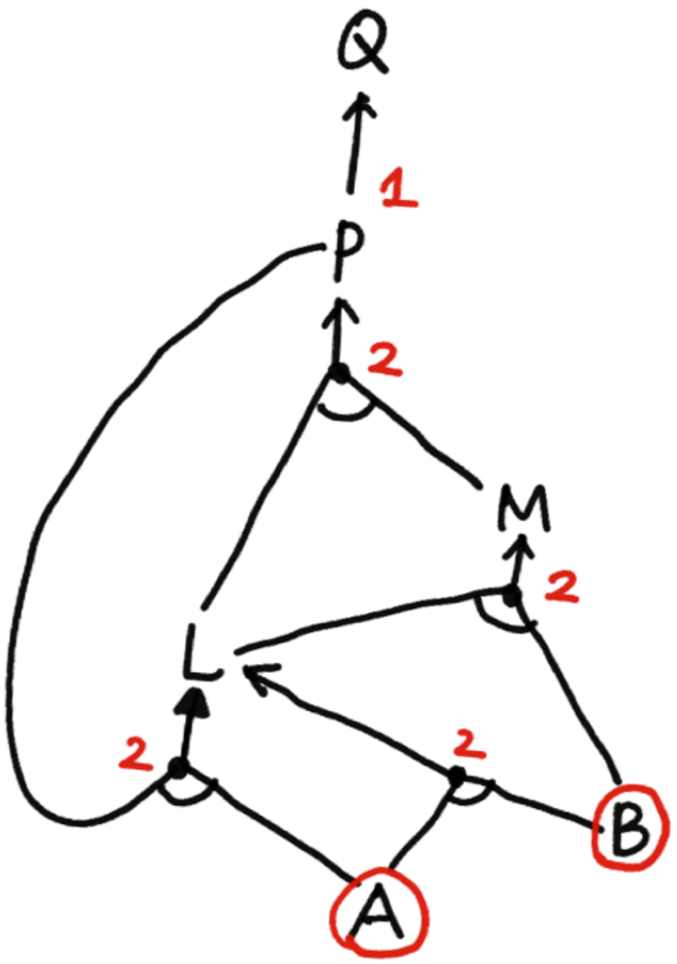
$$\frac{\alpha_1, \cdots, \alpha_n, \quad \alpha_1 \wedge \cdots \wedge \alpha_n \Rightarrow \beta}{\beta}$$

Forward Chaining

```
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, init. 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

- Agenda: A, B
- Annotate Horn clauses with the number of premises

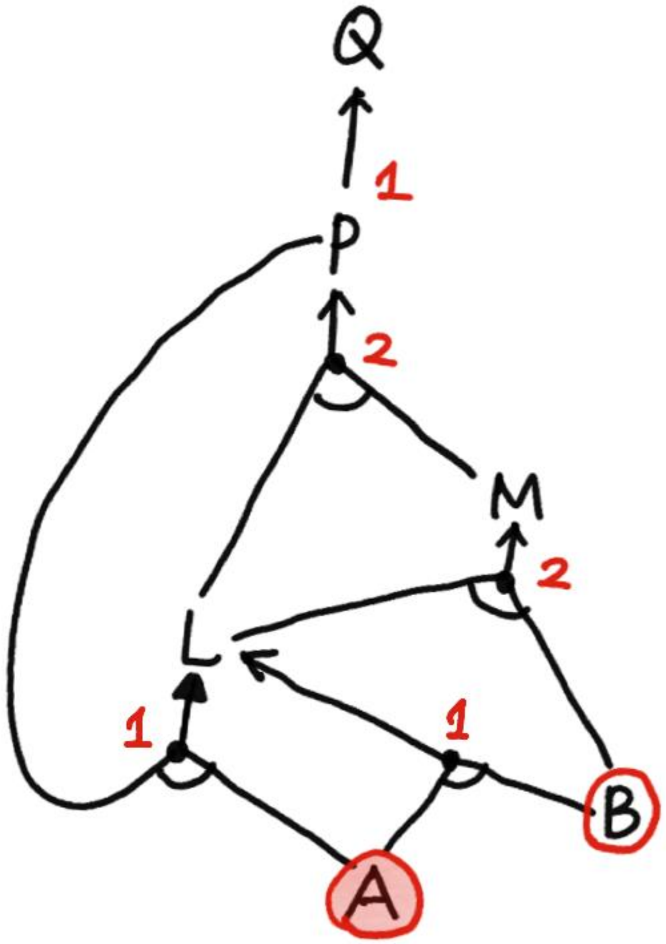


$P \Rightarrow Q$
$L \wedge M \Rightarrow P$
$B \wedge L \Rightarrow M$
$A \wedge P \Rightarrow L$
$A \wedge B \Rightarrow L$
A
B

Forward Chaining

- Process agenda item A

- Decrease count for Horn clauses in which A is premise



$P \Rightarrow Q$

$L \wedge M \Rightarrow P$

$B \wedge L \Rightarrow M$

$A \wedge P \Rightarrow L$

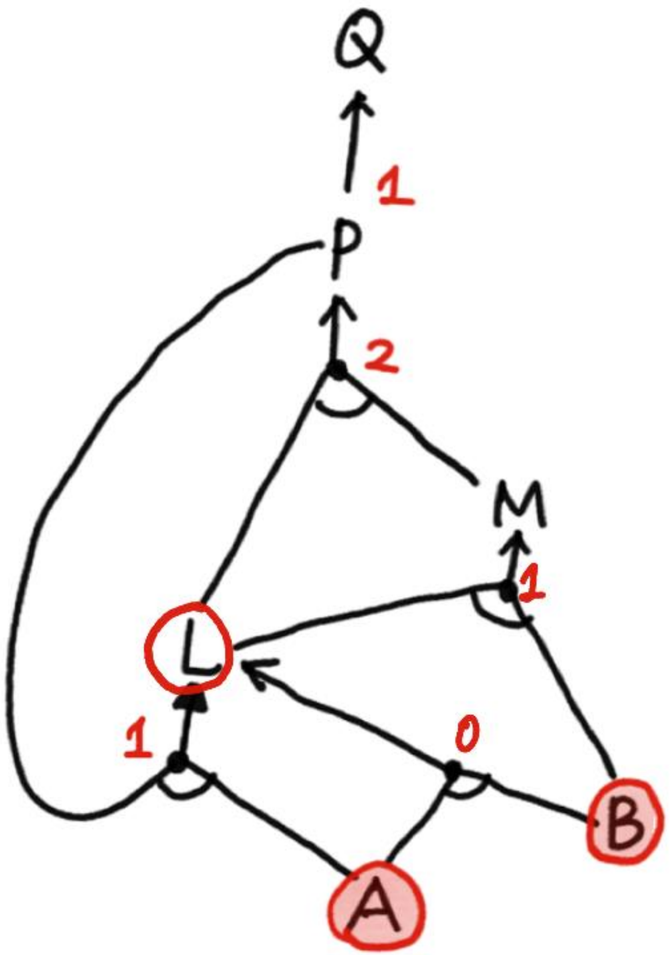
$A \wedge B \Rightarrow L$

A

B

Forward Chaining

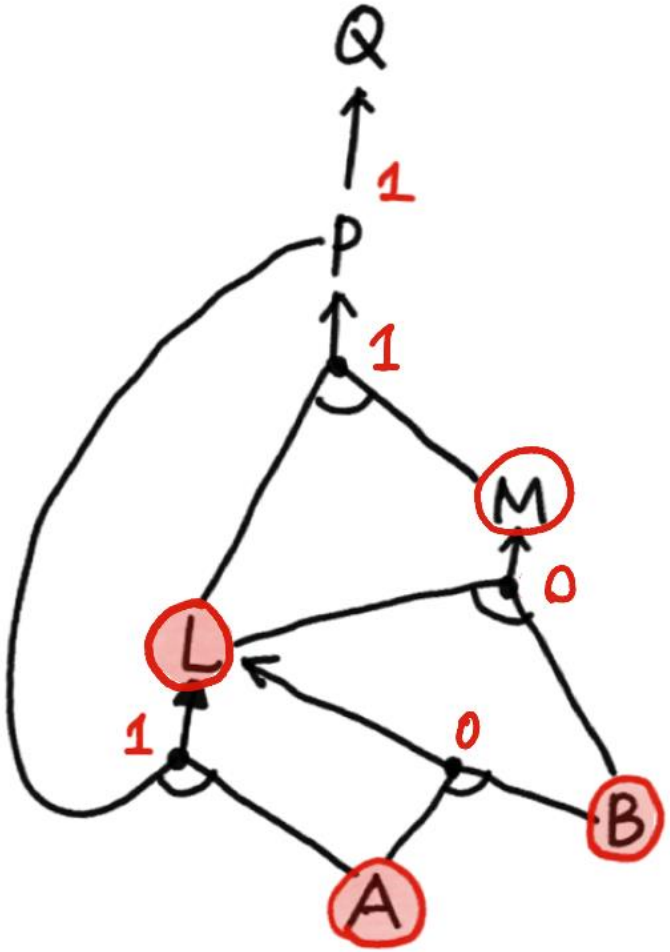
- Process agenda item B
- Decrease count for Horn clauses in which B is premise
- $A \wedge B \Rightarrow L$ has now fulfilled premise
- Now L is the agenda



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

Forward Chaining

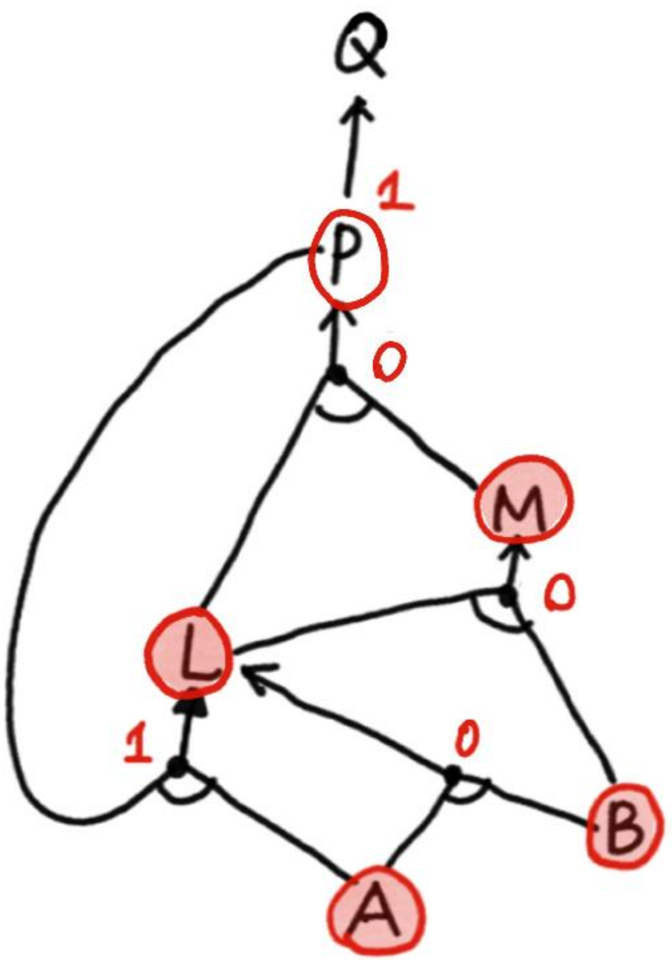
- Process agenda item L
- Decrease count for Horn clauses in which L is premise
- $B \land L \Rightarrow M$ has now fulfilled premise
- Now M is the agenda



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

Forward Chaining

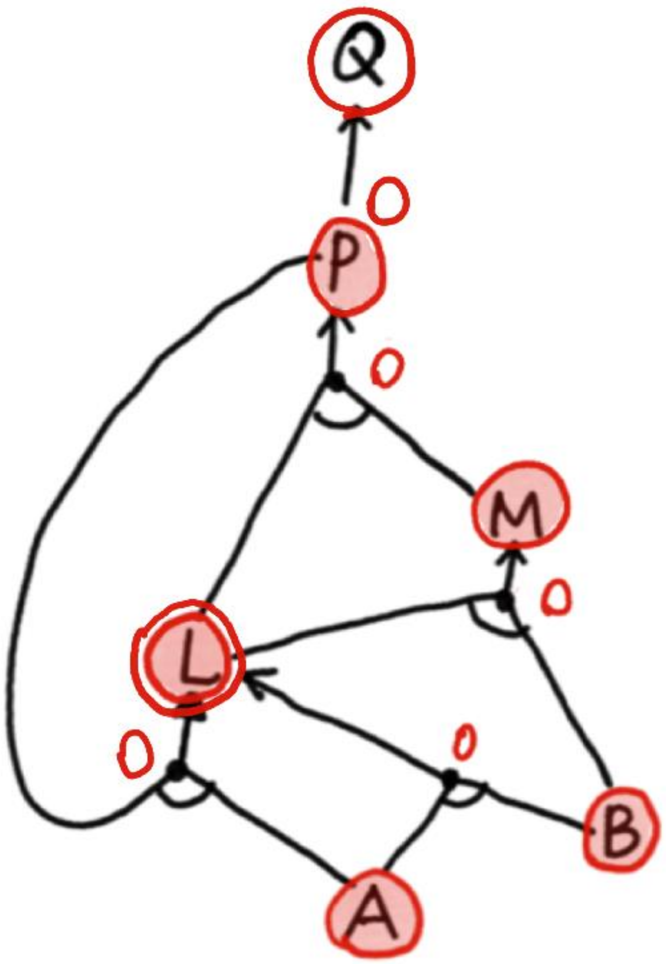
- Process agenda item M
- Decrease count for Horn clauses in which M is premise
- $L \wedge M \Rightarrow P$ has now fulfilled premise
- Now P is the agenda



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

Forward Chaining

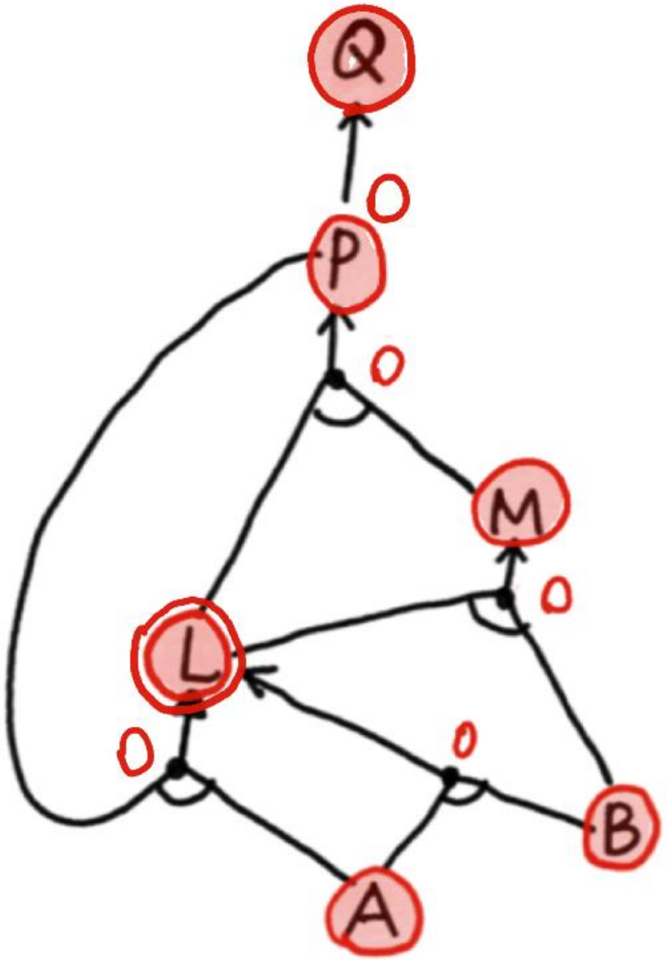
- Process agenda item P
- Decrease count for Horn clauses in which P is premise
- $P \Rightarrow Q$ has now fulfilled premise
- Now Q is the agenda
- $A \wedge P \Rightarrow L$ has now fulfilled premise



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

Forward Chaining

- Process agenda item Q
- Q is finished and done



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

Backward Chaining

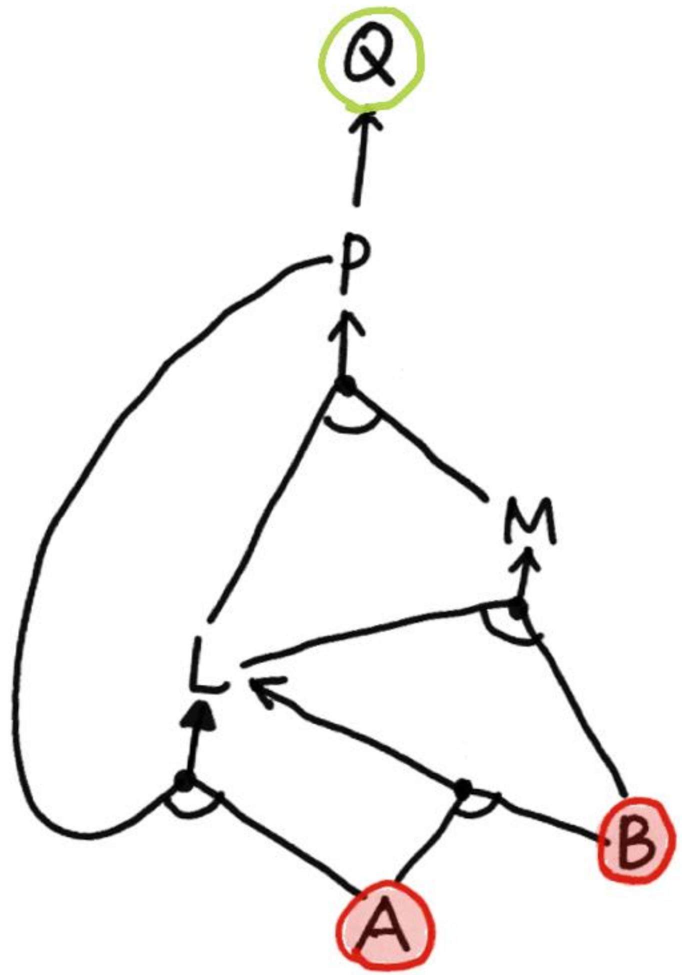
- Idea: work backwards from the query Q
  - To prove Q by backward chaining
  - Check if Q is known or
  - Prove by backward chaining all premises of some rule concluding Q

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

- Avoid repeated work: check of subgoal
  - has already been proved true, or
  - has already failed

Backward Chaining

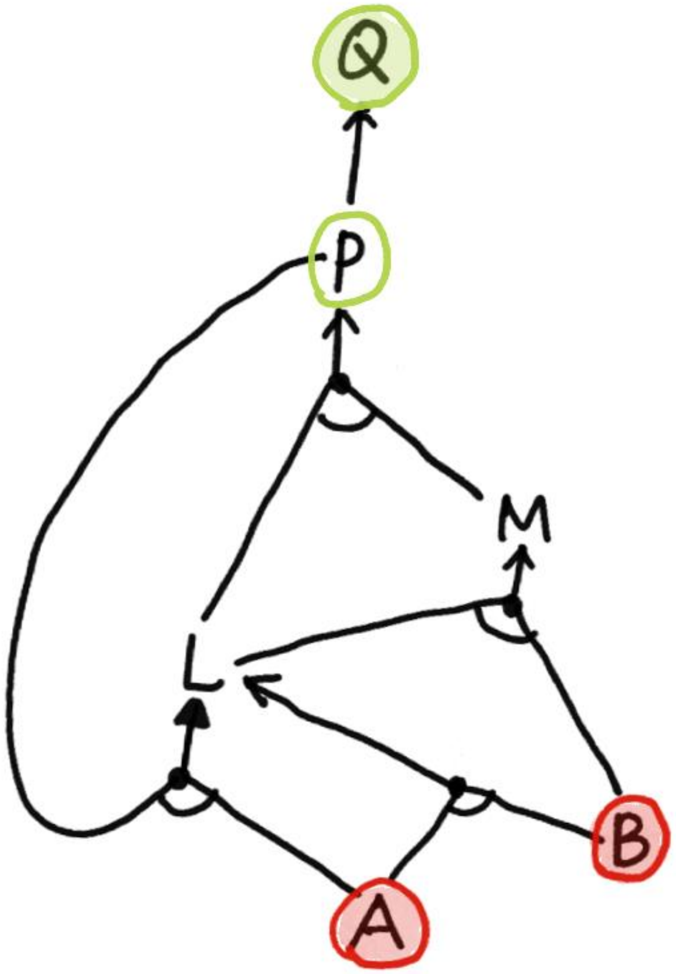
- A and B are known to be true.
- Q needs to be proven.



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

Backward Chaining

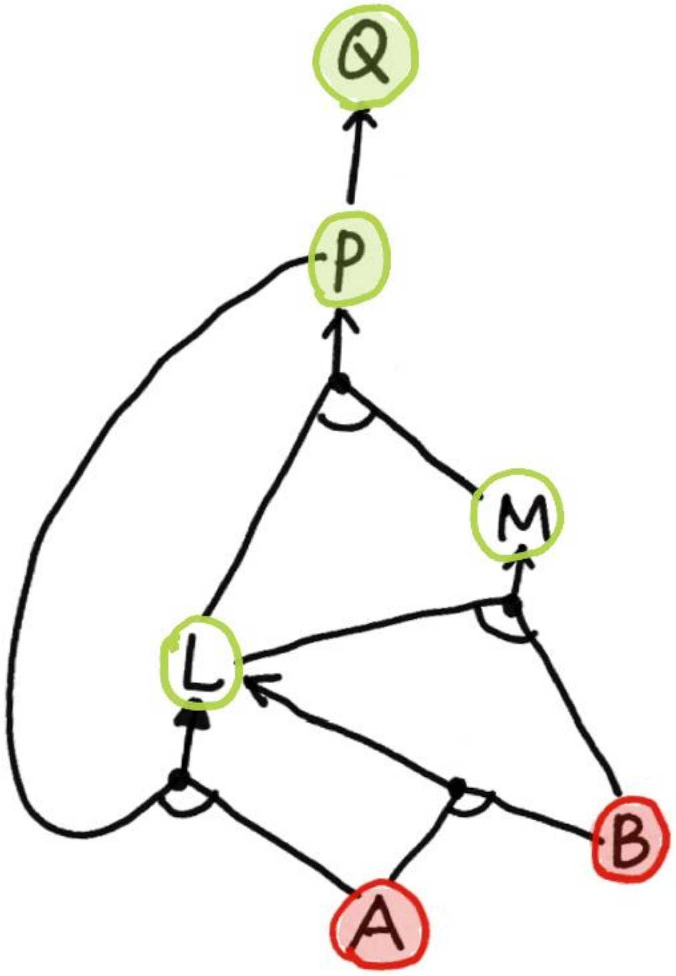
- Current goal: Q
- Q can be inferred by $P \Rightarrow Q$
- P needs to be proven.



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

Backward Chaining

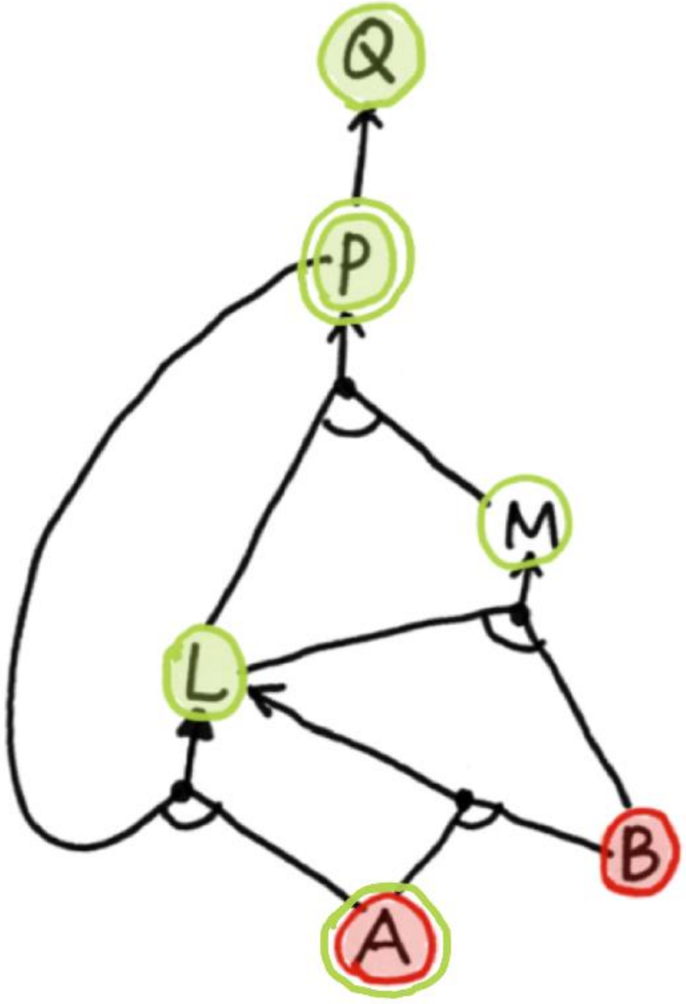
- Current goal: P
- P can be inferred by $L \wedge M \Rightarrow P$
- L and M need to be proven.



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

Backward Chaining

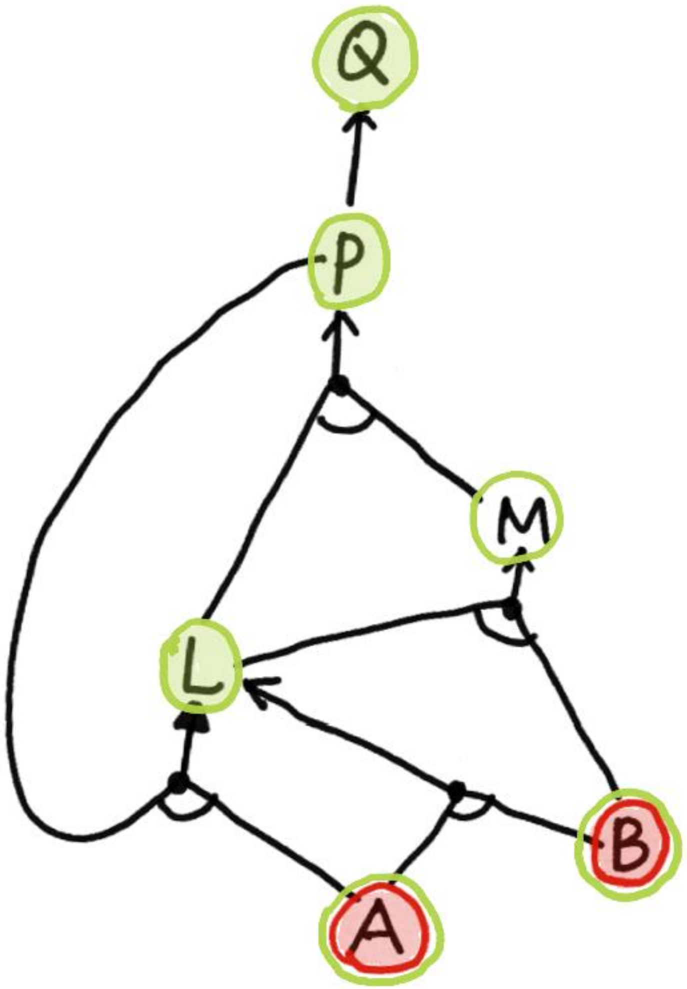
- Current goal: L
- L can be inferred by $A \wedge P \Rightarrow L$
- A is already true
- P is already a goal (repeated subgoal)



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

Backward Chaining

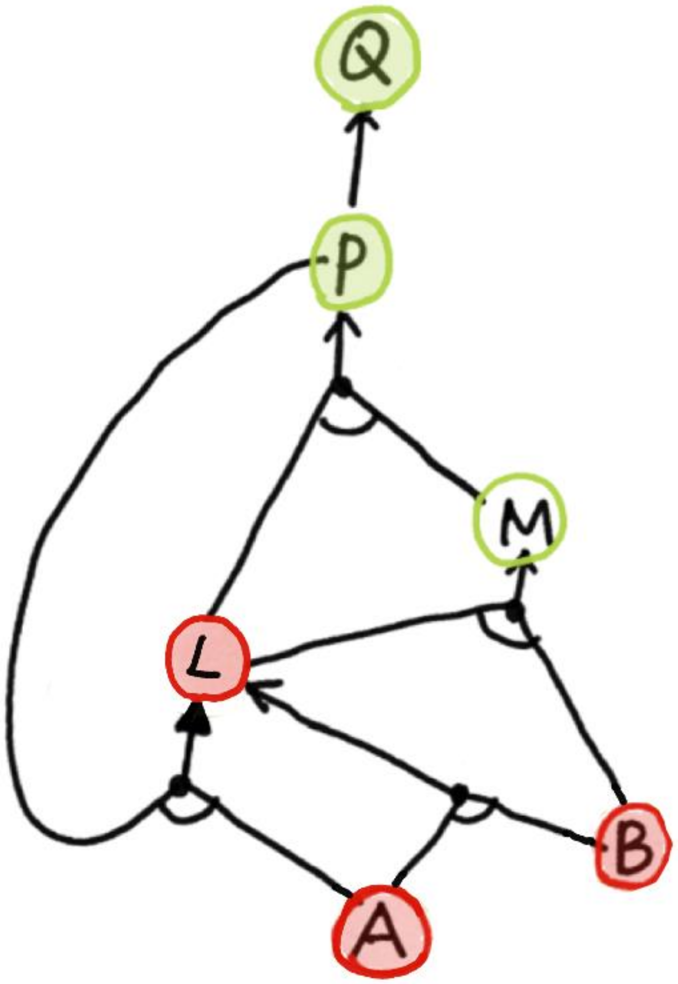
- Current goal: L
- L can be inferred by $A \wedge B \Rightarrow L$
- Both are true



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

Backward Chaining

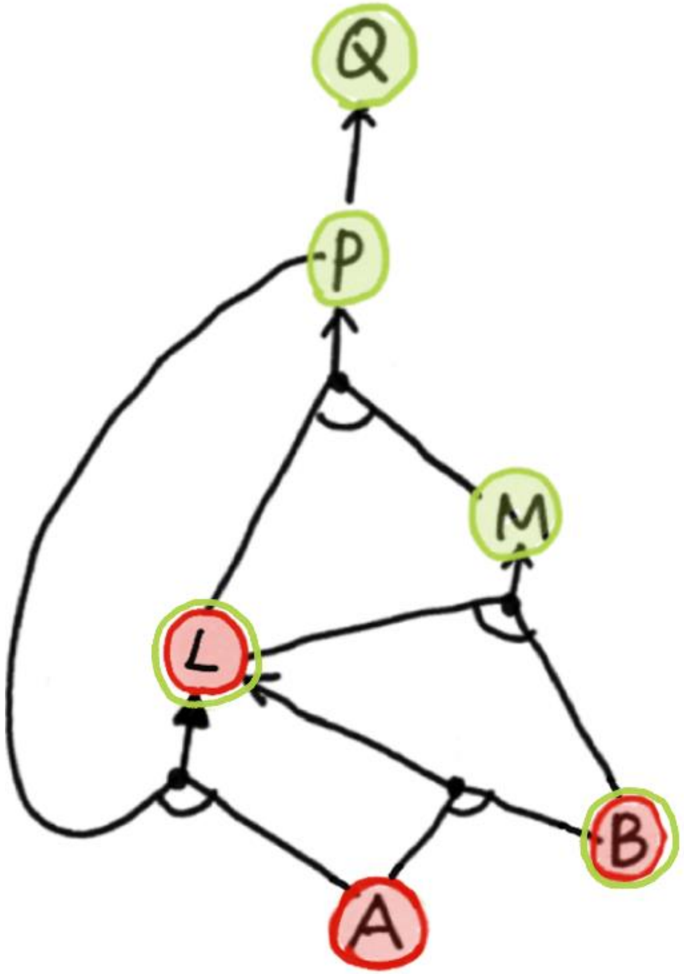
- Current goal: L
- L can be inferred by $A \wedge B \Rightarrow L$
- Both are true
- Thus, L is true.



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

Backward Chaining

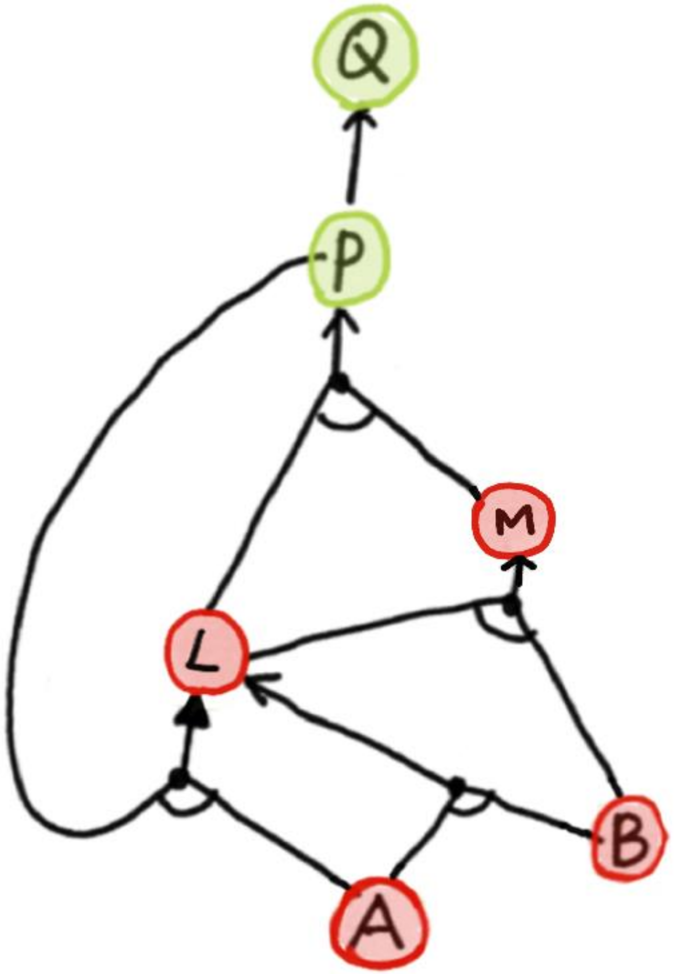
- Current goal: M
- M can be inferred by $B \wedge L \Rightarrow M$
- Both are true



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

Backward Chaining

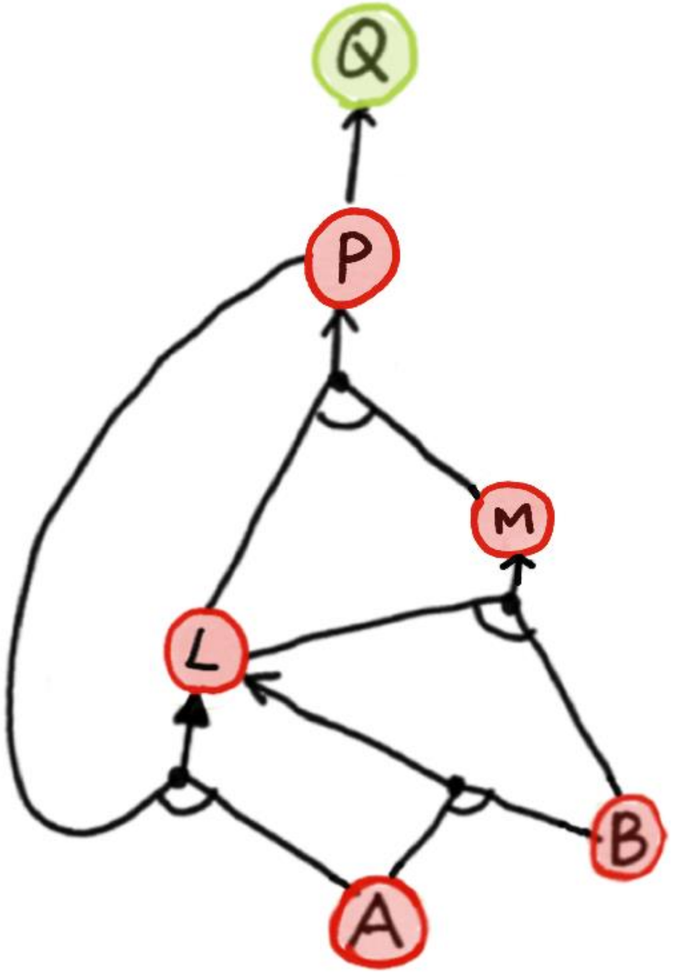
- Current goal: M
- M can be inferred by $B \wedge L \Rightarrow M$
- Both are true
- Thus, M is true.



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

Backward Chaining

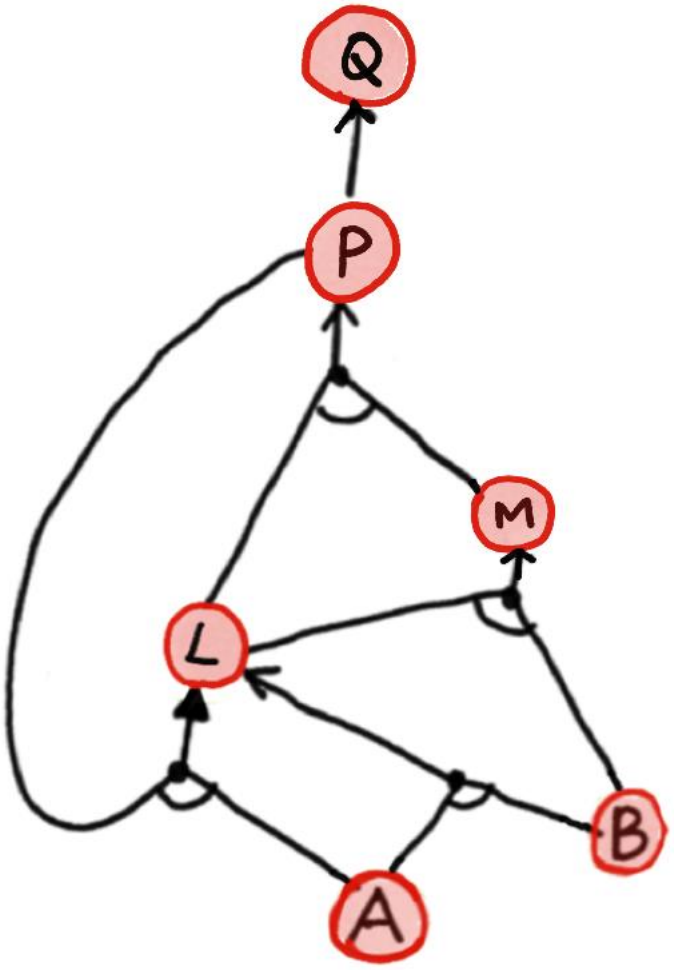
- Current goal: P
- P can be inferred by $L \wedge M \Rightarrow P$
- Both are true
- Thus, P is true.



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

Backward Chaining

- Current goal: Q
- Q can be inferred by $P \Rightarrow Q$
- P is true
- Thus, Q is true.



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

Forward vs. Backward Chaining

Forward

- Data driven, e.g., object recognition, routine decisions
- May do lots of work that is irrelevant to the goal

Backward

- Goal driven, e.g., where are my keys? How do I get into a PhD?
- Complexity of backward chaining can be much less than linear size of KB.

Resolution

- To show $KB \vDash \alpha$, we show that $KB \wedge \neg\alpha$ is not satisfiable.

- Conjunctive Normal Form: conjunction of disjunctions of literals
  - E.g., $(A \vee \neg B) \wedge (B \vee \neg C \vee \neg D)$

- Apply resolution to $KB \wedge \neg\alpha$ in CNF

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

where l_i and m_j are complementary literals.

until
  - there are no new clauses to be added.
  - two clauses resolve to the empty class, which means $KB \vDash \alpha$.

Resolution

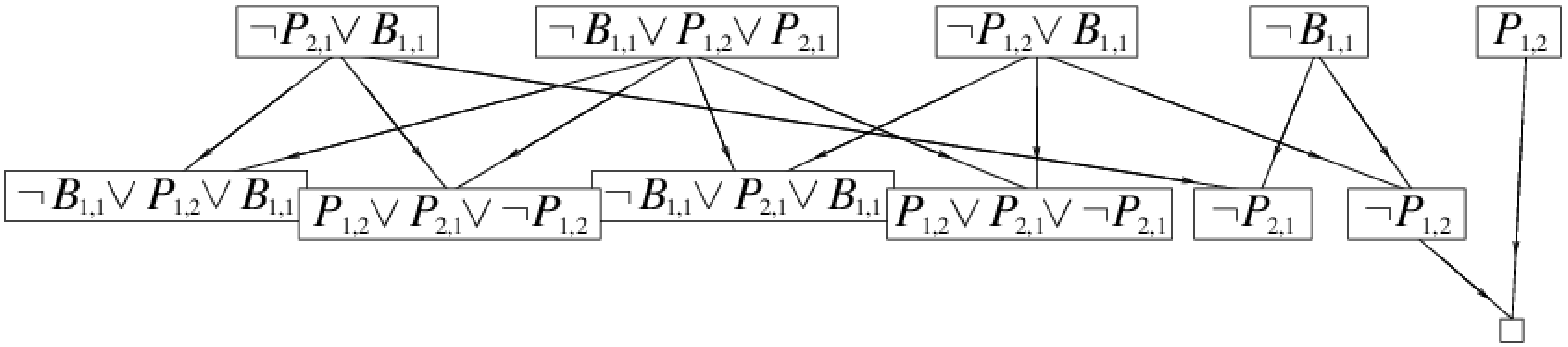
```
function PL-RESOLUTION(KB, α) returns true or false
   inputs: KB, the knowledge base, a sentence in propositional logic
           α, the query, a sentence in propositional logic

   clauses ← the set of clauses in the CNF representation of KB ∧ ¬α
   new ← { }
   loop do
      for each Ci, Cj in clauses do
         resolvents ← PL-RESOLVE(Ci, Cj)
         if resolvents contains the empty clause then return true
         new ← new ∪ resolvents
      if new ⊆ clauses then return false
      clauses ← clauses ∪ new
```

Resolution in Wumpus

- We take a subset of the knowledge base, say

$$KB = R_2 \wedge R_4 = \left(B_{1,1} \Leftrightarrow \left(P_{1,2} \vee P_{2,1}\right)\right) \wedge \neg B_{1,1}$$

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



Goals

- Understand how knowledge-based agents work.
- Understand the basic concepts in logic.
- Understand the propositional logic and its implementation in Wumpus.
- Understand the three inference methods for propositional logic.
- Know how to implement the forward and backward chaining.
- Know how to implement the resolution algorithm.

Important This Week

- Do more exercises in Chapter 7 in the textbook.