Note

- Submission of HW1 closed. No late submission allowed.
- Fast Prototyping Exercise #1 on PCA starts next week
  - **HW assignment: Continue studying MATLAB**
  - **HW assignment: Read the reference paper:** https://bidal.sfsu.edu/~kazokada/csc872/PD1.pdf
  - **Download:** https://bidal.sfsu.edu/~kazokada/csc872/DATA/FaceRecognition_Data.zip

1

PF

Knowledge-Based Agents with Propositional Logic

KR/PS

CSC 872

Pattern Analysis and Machine Intelligence

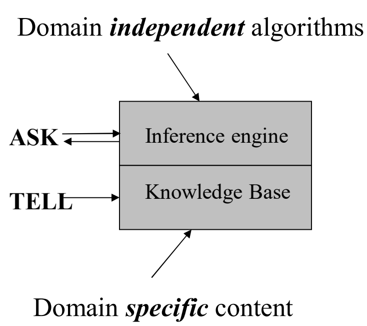
2

Review

- Last Lecture: Search Methods
  - One instance of the AI agent
  - Problem-Solving Agent
  - Goal-based (Uninformed Search)
  - Utility-based (Informed Search) A^*

- Today: knowledge-based agent!
  - Another instance for realizing AI agent
  - (Simple or Model-based) Reflex Agent
  - <u>How do we describe the condition-action rules for more complex problem</u>?

Knowledge-Based Agent



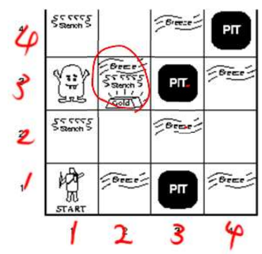
- **TELL** agent what to know
- **ASK** agent to query what to do
- **Knowledge Base (KB):** contains a set of <u>representations</u> of facts about the Agent's environment
- **Sentence** = each representation
- **Knowledge Representation Language** = formal language used to **TELL** facts
- **Inference** = reasoning to answer the query by deducing new facts from **TELL**ed facts
- versus Condition-Action Rules…
  - Use a formal language = Logic
  - Use a general inference algorithm

Toy Problem: Wumpus World

Percepts Breeze, Glitter, Smell

Actions Left turn, Right turn,
Forward, Grab, Release, Shoot

Goals Get gold back to start
without entering pit or wumpus square

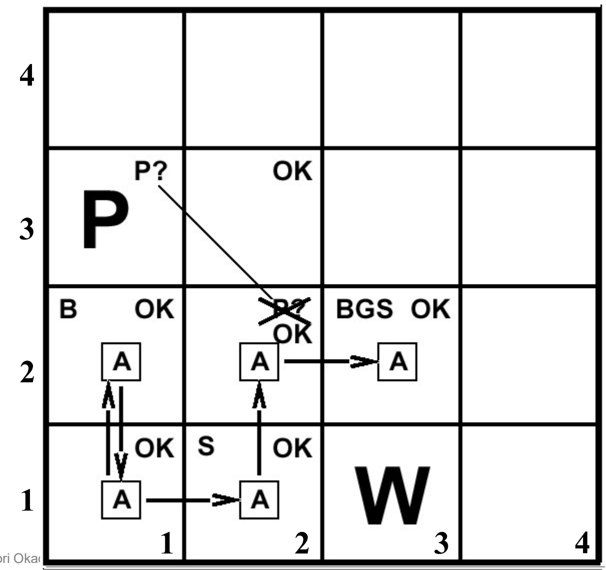


Environment
- Squares adjacent to wumpus are smelly
- Squares adjacent to pit are breezy
- Glitter if and only if gold is in the same square
- Shooting kills the wumpus if you are facing it
- Shooting uses up the only arrow
- Grabbing picks up the gold if in the same square
- Releasing drops the gold in the same square

Wumpus World Characteristics

- Fully Observable **No** – only local perception Partially observable
- Deterministic **Yes** – outcomes exactly specified
- Episodic **No** – sequential at the level of actions
- Static **Yes** – Wumpus and Pits do not move
- Discrete **Yes**
- Single-agent **Yes** – Wumpus is essentially a natural feature

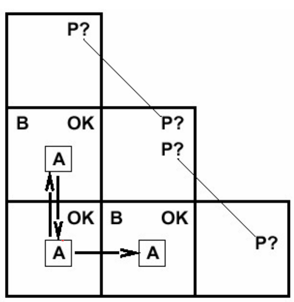
Exploring Wumpus World



A= Agent
B= Breeze
S= Smell
P= Pit
W= Wumpus
OK = Safe
V = Visited
G = Glitter

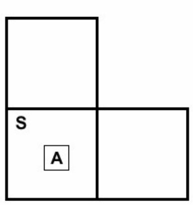
7

Some Tight Spots



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

Assuming pits uniformly distributed, (2,2) is most likely to have a pit



Smell in (1,1)
⇒ cannot move

Can use a strategy of <u>coercion</u>:
shoot straight ahead
wumpus was there ⇒ dead ⇒ safe
wumpus wasn't there ⇒ safe

8

KR: Logic

- Logic is formal language for representing information such that conclusions can be drawn
- Syntax: defines the sentences in the language
- Semantics: define the "meaning" of sentences or "truth of a sentence in a world"
- E.g., the language of arithmetic
  - $x+2 \geq y$ is a sentence; $x2y+\geq \{\}$ 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 $x = 0, y = 6$

Grammar

Meaning

grammatical error

syntax error

PF: Entailment

- **Entailment** means that one sentence follows from another:

$$KB \models \alpha$$

- Knowledge base *KB* entails sentence α if and only if α is true in all worlds where *KB* is true
  - E.g., the **KB** containing "**the GGate won**" and "**the Giants won**" entails "**Either the GGate won or the Giants won**"
  - E.g., $x+y=4$ entails $4=x+y$
  - Entailment is a relationship between sentences (i.e., syntax) that is based on semantics
  - **Entailment is different from Inference**

Knowledge Representation by Logic

KB $\qquad$ α

Representation: Sentences $\xrightarrow{\text{entails}}$ Sentence

Refers to (Semantics)

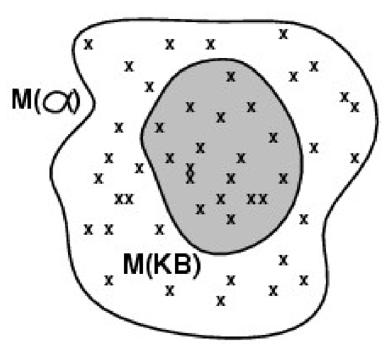
World Facts $\xrightarrow{\text{follows}}$ Fact

Models

- Logicians typically think in terms of **models**, which are formally structured **worlds/interpretations** with respect to which truth can be evaluated $x+2 \geq y$ $\quad$ m_1: $x=1$, $y=0$ $\quad$ m_2: $x=2$ $y=0$ ...

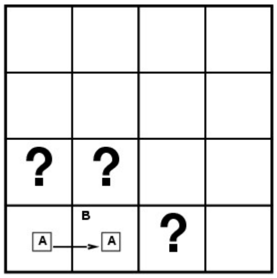
- We say ***m*** is a model of a sentence **α** if **α** is true in ***m***
  - ***α***: $x + 7 \geq y$
  - ***m***: $(x, y) = (3, 4)$ $(1, 4.4), (5, 1), \ldots$
- ***M(α)*** is the set of all models of **α**
  - ***M(α)***: $\{(x, y)\colon x + 7 \geq y\}$

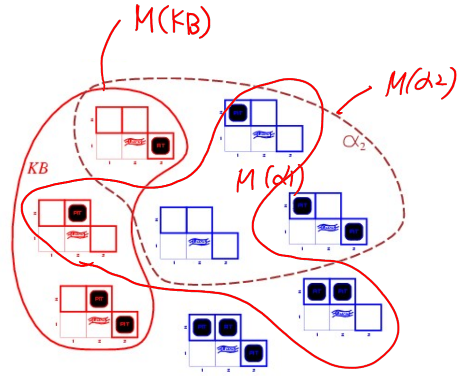
- **KB** $\models$ **α** *iff* ***M(KB)*** $\subseteq$ ***M*(α)**
  - *KB:* GGate won and Giants won
  - **α** : either GGate or Giants won



Entailment in the Wumpus World

α3 = "[3, 1] is safe"





- *KB* = wumpus-world rules + observations
- *α1* = "[1,2] is safe", $KB \models \alpha 1$
- *α2* = "[2,2] is safe", $KB \not\models \alpha 2$

PF: Model Checking

PF: Logical Inference

- $KB \vdash_i \alpha$
  - **sentence α can be derived from *KB* by procedure i**
  - α is inferred from *KB by using procedure i*
  - *Query "**Is α** **true given KB**?"* is proven true by "i"
  - Deductive Reasoning

- Property of the inference procedure "i"
- **Soundness**: All inference is entailment
  - "i" is sound if whenever $KB \vdash_i \alpha$, it is also true that $KB \models \alpha$
- **Completeness**: All entailment is inference
  - "i" is complete if whenever $KB \models \alpha$, it is also true that $KB \vdash_i \alpha$

Sound & Complete i → $KB \vdash_i \alpha \equiv KB \models \alpha$

Propositional Logic: Syntax

- The simplest logical language
- **If P and Q are sentences, following are also sentences with logical connectives:** $\neg, \vee, \wedge, \Rightarrow, \Leftrightarrow$

- P "P is true"
- $\neg P$ ¬ **negation** "P is false"
- $P \vee Q$ ∨ **disjunction** "either P is true or Q is true or both"
- $P \wedge Q$ ∧ **conjunction** "both P and Q are true"
- $P \Rightarrow Q$ → **implication** "if P is true, then Q is true"
- $P \Leftrightarrow Q$ ↔ **equivalence** "P and Q are either both true or both false" biconditional

15

Propositional Logic: Semantics

- **Propositional logic only deal with facts**:
  - Symbols and expressions only evaluate to either "**true**" or "**false**"
- A model "m" specifies true/false for each proposition symbol
-

E.g.	S_1	S_2	S_3
m_1	false	true	false
m_2	true	true	false

Rules for evaluating truth with respect to a model m:

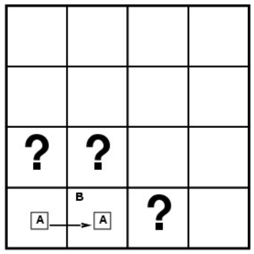
$\neg S$	is true iff	S	is false		
$S_1 \wedge S_2$	is true iff	S_1	is true <u>and</u>	S_2	is true
$S_1 \vee S_2$	is true iff	S_1	is true <u>or</u>	S_2	is true
$S_1 \Rightarrow S_2$	is true iff	S_1	is false <u>or</u>	S_2	is true
i.e.,	is false iff	S_1	is true <u>and</u>	S_2	is false
$S_1 \Leftrightarrow S_2$	is true iff	$S_1 \Rightarrow S_2$	is true <u>and</u>	$S_2 \Rightarrow S_1$	is true

16

Wumpus World by Propositional Logic

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"

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

Truth Table

- Truth value: whether a sentence is true or false.
- Truth table: complete list of truth values for a sentence given all possible values of the individual atomic expressions (defining their semantics).

$2^2=4$

P	Q	$\neg P$	$P \wedge Q$	$P \vee Q$	$P \Rightarrow Q$	$P \Leftrightarrow Q$
$false$	$false$	$true$	$false$	$false$	$true$	$true$
$false$	$true$	$true$	$false$	$true$	$true$	$false$
$true$	$false$	$false$	$false$	$true$	$false$	$false$
$true$	$true$	$false$	$true$	$true$	$true$	$true$

PS: Propositional Inference by Enumeration Method (Model Checking)

$M(KB) = \{m4, m5, m7, m8\}$

$M(\alpha) = \{m3, m4, m5, m6, m7, m8\}$

Let $\alpha = A \vee B$ and $KB = (A \vee C) \wedge (B \vee \neg C)$

Is it the case that $KB \models \alpha$? $M(KB) \subseteq M(\alpha)$

Check all possible models—α must be true wherever KB is true

	A	B	C	$A \vee C$	$B \vee \neg C$	KB	α
m1	*False*	*False*	*False*	*False*	*True*	*False*	*False*
m2	*False*	*False*	*True*	*True*	*False*	*False*	*False*
m3	*False*	*True*	*False*	*False*	*True*	*False*	*True*
m4	*False*	*True*	*True*	*True*	*True*	*True*	*True*
m5	*True*	*False*	*False*	*True*	*True*	*True*	*True*
m6	*True*	*False*	*True*	*True*	*False*	*False*	*True*
m7	*True*	*True*	*False*	*True*	*True*	*True*	*True*
m8	*True*	*True*	*True*	*True*	*True*	*True*	*True*

Proof Methods as KB Query

- The procedure we are interested is essentially the same as performing mathematical proof!!!
- Two types of proof methods:
  - **Application of inference rules (Deductive)**
    - **Inference rule** = *Sound* generation of new sentences from old ones
    - **Proof** = a sequence of inference rule applications
      Use inference rules as operators in a standard search algorithm
    - Typically require transformation of sentences into a **normal form**
  - **Model checking (Enumerative)**
    - truth table enumeration (always exponential in n)
    - improved backtracking,
      - e.g., Davis--Putnam-Logemann-Loveland (DPLL)
    - heuristic search in model space (*sound but incomplete*)
      - e.g., min-conflicts-like hill-climbing algorithms

Validity and Satisfiability

always true → Tautology

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

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

A sentence is **satisfiable** if it is true in some world

e.g., $A, \neg A, True, A \lor B$

always false → contradiction

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

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

Validity is connected to inference via the Deduction Theorem:

$KB \models \alpha \ \leftrightarrow \ (KB \Rightarrow \alpha)$ is valid

Satisfiability is connected to inference via the following:

$KB \models \alpha \ \leftrightarrow \ (KB \land \neg \alpha)$ is unsatisfiable

Logical Equivalence

- Two sentences are logically equivalent

$$\boldsymbol{\alpha \equiv \beta} \ \textit{iff} \ \boldsymbol{\alpha \models \beta \land \beta \models \alpha}$$

$$
\begin{array}{rcll}
(\alpha \land \beta) & \equiv & (\beta \land \alpha) & \text{commutativity of } \land \\
(\alpha \lor \beta) & \equiv & (\beta \lor \alpha) & \text{commutativity of } \lor \\
((\alpha \land \beta) \land \gamma) & \equiv & (\alpha \land (\beta \land \gamma)) & \text{associativity of } \land \\
((\alpha \lor \beta) \lor \gamma) & \equiv & (\alpha \lor (\beta \lor \gamma)) & \text{associativity of } \lor \\
\neg(\neg \alpha) & \equiv & \alpha & \text{double-negation elimination} \\
(\alpha \Rightarrow \beta) & \equiv & (\neg \beta \Rightarrow \neg \alpha) & \text{contraposition} \\
(\alpha \Rightarrow \beta) & \equiv & (\neg \alpha \lor \beta) & \text{implication elimination} \\
(\alpha \Leftrightarrow \beta) & \equiv & ((\alpha \Rightarrow \beta) \land (\beta \Rightarrow \alpha)) & \text{biconditional elimination} \\
\neg(\alpha \land \beta) & \equiv & (\neg \alpha \lor \neg \beta) & \text{de Morgan} \\
\neg(\alpha \lor \beta) & \equiv & (\neg \alpha \land \neg \beta) & \text{de Morgan} \\
(\alpha \land (\beta \lor \gamma)) & \equiv & ((\alpha \land \beta) \lor (\alpha \land \gamma)) & \text{distributivity of } \land \text{ over } \lor \\
(\alpha \lor (\beta \land \gamma)) & \equiv & ((\alpha \lor \beta) \land (\alpha \lor \gamma)) & \text{distributivity of } \lor \text{ over } \land
\end{array}
$$

(Sound) Inference Rules

α: I am hungry
β: I eat

- **Modus Ponens** or **Implication-Elimination:** (From an implication and the premise of the implication, you can infer the conclusion.)

$$\frac{\alpha \Rightarrow \beta, \quad \alpha}{\beta}$$

$\equiv (\alpha \to \beta) \land \alpha \to \beta$ Tautology

- **And-Elimination**: (From a conjunction, you can infer any of the conjuncts.)

$$\frac{\alpha_1 \land \alpha_2 \land \ldots \land \alpha_n}{\alpha_i}$$

- **And-Introduction**: (From a list of sentences, you can infer their conjunction.)

$$\frac{\alpha_1, \alpha_2, \quad \ldots, \quad \alpha_n}{\alpha_1 \land \alpha_2 \land \ldots \land \alpha_n}$$

- **Or-Introduction**: (From a sentence, you can infer its disjunction with anything else at all.)

$$\frac{\alpha_i}{\alpha_1 \lor \alpha_2 \lor \ldots \lor \alpha_n}$$

$$\frac{a}{a \lor \text{anything}}$$

(Sound) Inference Rules cond.

- **Double-Negation Elimination**: (From a doubly negated sentence, you can infer a positive sentence.)

$$\frac{\neg\neg\alpha}{\alpha}$$

- **Unit Resolution:** (From a disjunction, if one of the disjuncts is false, then you can infer the other one is true.)

$$\frac{\alpha \lor \beta, \quad \neg\beta}{\alpha}$$

disjunctive syllogism

- **Resolution**: (This is the most difficult. Because β cannot be both true and false, one of the other disjuncts must be true in one of the premises. Or equivalently, implication is transitive.)

$$\frac{\alpha \lor \beta, \quad \neg\beta \lor \gamma}{\alpha \lor \gamma} \quad \text{or equivalently} \quad \frac{\overset{A \to B}{\neg\alpha \Rightarrow \beta}, \quad \overset{B \to C}{\beta \Rightarrow \gamma}}{\underset{A \to C}{\neg\alpha \Rightarrow \gamma}}$$

PS: Resolution

- **Conjunctive Normal Form** (CNF)
  - *conjunction of disjunctions of literals/clauses*

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

    clauses

$(\circ \lor \square \lor \star) \land (\star \lor \square) \land$

- **Resolution** inference rule (for CNF):

$$\frac{(l_1 \lor \ldots \lor l_k), \qquad (m_1 \lor \ldots \lor m_n)}{l_1 \lor \ldots \lor l_{i-1} \lor l_{i+1} \lor \ldots \lor l_k \lor m_1 \lor \ldots \lor m_{j-1} \lor m_{j+1} \lor \ldots \lor m_n}$$

$l_i = \neg m_j$

where l_i and m_j are complementary literals.

E.g., $\dfrac{(P_{1,3} \lor P_{2,2}), \ \neg P_{2,2}}{P_{1,3}}$

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

PS: Conversion to CNF (example)

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

1. **Eliminate** $\Leftrightarrow$, replacing $\alpha \Leftrightarrow \beta$ with $(\alpha \Rightarrow \beta) \land (\beta \Rightarrow \alpha)$.

   $(B_{1,1} \Rightarrow (P_{1,2} \lor P_{2,1})) \land ((P_{1,2} \lor P_{2,1}) \Rightarrow B_{1,1})$

2. **Eliminate** $\Rightarrow$, replacing $\alpha \Rightarrow \beta$ with $\neg \alpha \lor \beta$.

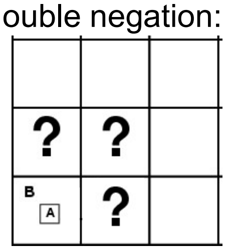
   $(\neg B_{1,1} \lor P_{1,2} \lor P_{2,1}) \land (\neg(P_{1,2} \lor P_{2,1}) \lor B_{1,1})$

3. **Move $\neg$ inwards** using de Morgan's rules and double negation:

   $(\neg B_{1,1} \lor P_{1,2} \lor P_{2,1}) \land ((\neg P_{1,2} \land \neg P_{2,1}) \lor B_{1,1})$

4. **Apply distributivity law** ($\land$ over $\lor$) and flatten:

   $(\neg B_{1,1} \lor P_{1,2} \lor P_{2,1}) \land (\neg P_{1,2} \lor B_{1,1}) \land (\neg P_{2,1} \lor B_{1,1})$

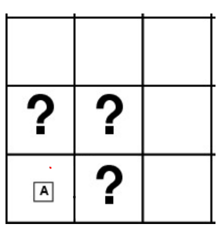


PS: Resolution Algorithm

- **Proof by Contradiction** of $KB \models \alpha$ contradiction
- *i.e.*, ***show (KB ∧ ¬α) is unsatisfiable***

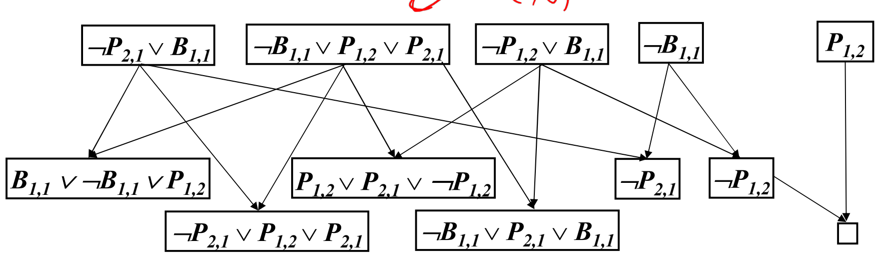
1) First convert ($KB \wedge \neg\alpha$) into CNF.
2) Then apply the resolution rule to resulting clauses.
3) The process continues until:
   a) *there are no new clauses* that can be added
      (*KB* does not entail α)
   b) two clauses *resolve to yield empty clause*
      (*KB* entails α)

Resolution Example



- There is no PIT in (1,2)?
- $KB = (B_{1,1} \Leftrightarrow (P_{1,2} \vee P_{2,1})) \wedge \neg B_{1,1}$
- $\alpha = \neg P_{1,2}$

CNF



PS: Resolution Algorithm

- Proof by Contradiction of $KB \models \alpha$

```
function PL-RESOLUTION(KB, α) returns true or false
   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
```


PS: Forward & Backward Chaining

- **Horn Form:**
  - *conjunction of Horn Clauses*

$$\frac{A \Rightarrow B, \ A}{B}$$

- **Horn Clause:**
  - proposition symbol; or
  - (conjunction of symbols) $\Rightarrow$ symbol
  - E.g., $\boldsymbol{C} \wedge (\boldsymbol{B} \Rightarrow \boldsymbol{A}) \wedge (\boldsymbol{C} \wedge \boldsymbol{D} \Rightarrow \boldsymbol{B})$

- **Modus Ponens:**

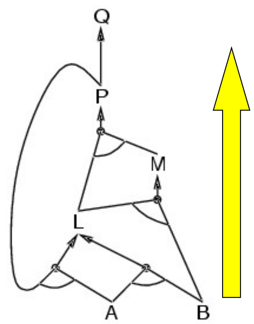
$$\frac{\alpha_1, \ldots, \alpha_n, \qquad \alpha_1 \wedge \ldots \wedge \alpha_n \Rightarrow \beta}{\beta}$$

- **Forward Chaining and Backward Chaining**
  - **uses Modus Ponens on Horn Forms.**
- **They are sound and complete for Horn Form**
- **They run in *LINEAR* time**

PS: Forward Chaining

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

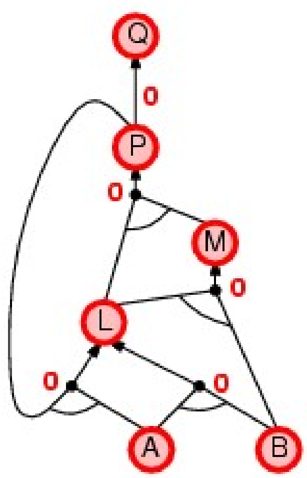
$$\begin{aligned}
&P \Rightarrow Q\\
&L \wedge M \Rightarrow P\\
&B \wedge L \Rightarrow M\\
&A \wedge P \Rightarrow L\\
&A \wedge B \Rightarrow L\\
&A\\
&B
\end{aligned}$$



PS: FC Algorithm Example

$KB \models Q$?

$$\begin{aligned}
&P \Rightarrow Q\\
&L \wedge M \Rightarrow P\\
&B \wedge L \Rightarrow M\\
&A \wedge P \Rightarrow L\\
&A \wedge B \Rightarrow L\\
&A\\
&B
\end{aligned}$$



PS: FC Algorithm

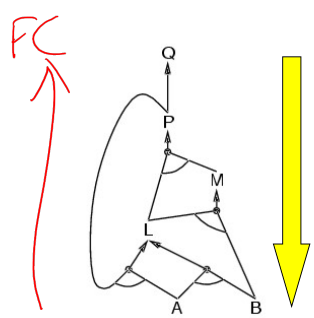
```
function PL-FC-ENTAILS?(KB, q) returns true or false
  local variables: count, a table, indexed by clause, initially the number of premises
                   inferred, a table, indexed by symbol, each entry initially false
                   agenda, a list of symbols, initially the symbols known to be true

  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 is sound and complete for Horn-Form *KB*

PS: Backward Chaining

- Motivation: Need goal-directed reasoning in order to keep from getting overwhelmed with irrelevant consequences
- Main idea:
  - Work backwards from query *Q*
    - Prove by backward chaining all premises of some rule concluding *Q*

$$\begin{aligned} P &\Rightarrow Q \\ L \wedge M &\Rightarrow P \\ B \wedge L &\Rightarrow M \\ A \wedge P &\Rightarrow L \\ A \wedge B &\Rightarrow L \\ A & \\ B & \end{aligned}$$



PS: Forward & Backward Chaining

- Forward Chaining is **data-driven**
  - automatic, unconscious processing,
  - e.g., object recognition, routine decisions
  - May do lots of work that is irrelevant to the goal
- Backward Chaining 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*

Limitation of Propositional Logic

- **Limited expressiveness**
  - Each situation (e.g., location, time) requires separate rule sentence
  - e.g., "don't go forward if the wumpus is in front of you" takes *64* rules when you have an *8*x*8* grid
  - e.g., to track *100* steps over time, we'll then need *6400* rules for the previous example. -> cannot keep track of changes over time
- **Huge Knowledge- and Rule-Base**
  - Hard to write and maintain such huge base
  - Inference becomes intractable

Summary

- Knowledge-Based 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
  - soundness: derivations produce only entailed sentences
  - completeness: derivations can produce all entailed sentences
- Resolution is complete for propositional logic
- Forward, backward chaining are linear-time, complete for Horn clauses
- *Propositional logic lacks expressive power*

- Next
  - First Order Logic
  - Fast Prototyping #1: **READ THE PAPER!!!**