Note

• Submission of HW1 closed. Reminder sent last night. No late submission allowed.
• Fast Prototyping Exercise #1 on PCA starts next week
  – HW assignment: Continue studying MATLAB
  – Download: https://bidal.sfsu.edu/~kazokada/csc872/FaceRecognition_Data.zip
**Review**

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

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

---

**Knowledge-Based Agent**

- **TELL** agent what to know
- **ASK** agent to query what to do
- **Knowledge Base (KB):** contains a set of representations 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 TELLed 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
- **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

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 coercion:
shoot straight ahead
wumpus was there ⇒ dead ⇒ safe
wumpus wasn’t there ⇒ safe
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; \( xz + 2 \geq \{ \} \) 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 \( x = 0, y = 6 \).

PF: Entailment

- **Entailment** means that one sentence follows from another:
  \[ KB \models \alpha \]

- Knowledge base \( KB \) entails sentence \( \alpha \) if and only if \( \alpha \) 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

**Models**

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

- We say \( m \) is a model of a sentence \( \alpha \) if \( \alpha \) is true in \( m \)
  - \( \alpha: x + 7 \geq y \)
  - \( m: (x, y) = (3, 4) \)

- \( M(\alpha) \) is the set of all models of \( \alpha \)
  - \( M(\alpha): \{(x, y): x + 7 \geq y\} \)

- \( KB \models \alpha \) iff \( M(KB) \subseteq M(\alpha) \)
  - \( KB: \) GGate won and Giants won
  - \( A: \) either GGate or Giants won
Entailment in the Wumpus World

- $\textbf{KB} = \text{wumpus-world rules } + \text{ observations}$
- $\alpha_1 = \text{"[1,2] is safe"}, \text{ } \textbf{KB} \models \alpha_1$
- $\alpha_2 = \text{"[2,2] is safe"}, \text{ } \textbf{KB} \not\models \alpha_2$

PF: Model Checking

PF: Logical Inference

- $\textbf{KB} \models_i \alpha$
  - sentence $\alpha$ can be derived from $\textbf{KB}$ by procedure $i$
  - $\alpha$ is inferred from $\textbf{KB}$ by using procedure $i$
  - Query "Is $\alpha$ true given $\textbf{KB}$?" is proven true by "$i$"
  - Deductive Reasoning

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

\[ \text{Sound } \& \text{ Complete } \iff \text{ } \textbf{KB} \models_i \alpha \iff \text{ } \textbf{KB} \models \alpha \]
Propositional Logic: Syntax

- The simplest logical language
- If P and Q are sentences, following are also sentences with logical connectives:
  \( \neg, \lor, \land, \rightarrow, \leftrightarrow \)

  - \( P \) "P is true"
  - \( \neg P \) negation "P is false"
  - \( P \lor Q \) disjunction "either P is true or Q is true or both"
  - \( P \land Q \) 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"

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 \( m \):

\[
\begin{align*}
\neg S & \text{ is true iff } S \text{ is false} \\
S_1 \land S_2 & \text{ is true iff } S_1 \text{ is true and } S_2 \text{ is true} \\
S_1 \lor S_2 & \text{ is true iff } S_1 \text{ is true or } S_2 \text{ is true} \\
S_1 \rightarrow S_2 & \text{ is true iff } S_1 \text{ is false or } S_2 \text{ is true} \\
\text{i.e.,} & \text{ is false iff } S_1 \text{ is true and } S_2 \text{ is false} \\
S_1 \leftrightarrow S_2 & \text{ is true iff } S_1 \Rightarrow S_2 \text{ is true and } S_2 \Rightarrow S_1 \text{ is true}
\end{align*}
\]
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 $[i,j]$.

- "Pits cause breezes in adjacent squares"
  \[
  B_{1,1} \iff (P_{1,2} \lor P_{2,1}) \\
  B_{2,1} \iff (P_{1,1} \lor P_{2,2} \lor P_{3,1})
  \]

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).

<table>
<thead>
<tr>
<th>$P$</th>
<th>$Q$</th>
<th>$\neg P$</th>
<th>$P \land Q$</th>
<th>$P \lor Q$</th>
<th>$P \Rightarrow Q$</th>
<th>$P \Leftrightarrow Q$</th>
</tr>
</thead>
<tbody>
<tr>
<td>false</td>
<td>false</td>
<td>true</td>
<td>false</td>
<td>true</td>
<td>true</td>
<td>false</td>
</tr>
<tr>
<td>false</td>
<td>true</td>
<td>true</td>
<td>false</td>
<td>true</td>
<td>true</td>
<td>false</td>
</tr>
<tr>
<td>true</td>
<td>false</td>
<td>false</td>
<td>false</td>
<td>false</td>
<td>false</td>
<td>true</td>
</tr>
<tr>
<td>true</td>
<td>true</td>
<td>true</td>
<td>true</td>
<td>true</td>
<td>true</td>
<td>true</td>
</tr>
</tbody>
</table>
**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

A sentence is **valid** if it is true in **all** worlds,
e.g., \( A \lor \neg A, True, \neg False, A \implies A, (A \land (A \implies B)) \implies B \)

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

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

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**:

\[ KB \vDash \alpha \iff (KB \implies \alpha) \text{ is valid} \]

Satisfiability is connected to inference via the following:

\[ KB \vDash \alpha \iff (KB \land \neg \alpha) \text{ is unsatisfiable} \]

Logical Equivalence

- Two sentences are **logically equivalent**

\[ \alpha \equiv \beta \iff \alpha \vDash \beta \land \beta \vDash \alpha \]

\[
\begin{align*}
(\alpha \land \beta) & \equiv (\beta \land \alpha) & \text{commutativity of } \land \\
(\alpha \lor \beta) & \equiv (\beta \lor \alpha) & \text{commutativity of } \lor \\
((\alpha \land \beta) \land \gamma) & \equiv (\alpha \land (\beta \land \gamma)) & \text{associativity of } \land \\
((\alpha \lor \beta) \lor \gamma) & \equiv (\alpha \lor (\beta \lor \gamma)) & \text{associativity of } \lor \\
\neg(\neg\alpha) & \equiv \alpha & \text{double-negation elimination} \\
(\alpha \implies \beta) & \equiv (\neg\beta \implies \neg\alpha) & \text{contraposition} \\
(\alpha \implies \beta) & \equiv (\neg\alpha \lor \beta) & \text{implication elimination} \\
(\alpha \iff \beta) & \equiv ((\alpha \implies \beta) \land (\beta \implies \alpha)) & \text{biconditional elimination} \\
\neg(\alpha \land \beta) & \equiv (\neg\alpha \lor \neg\beta) & \text{de Morgan} \\
\neg(\alpha \lor \beta) & \equiv (\neg\alpha \land \neg\beta) & \text{de Morgan} \\
(\alpha \land (\beta \lor \gamma)) & \equiv ((\alpha \land \beta) \lor (\alpha \land \gamma)) & \text{distributivity of } \land \text{ over } \lor \\
(\alpha \lor (\beta \land \gamma)) & \equiv ((\alpha \lor \beta) \land (\alpha \lor \gamma)) & \text{distributivity of } \lor \text{ over } \land
\end{align*}
\]
(Sound) Inference Rules

\( \alpha: \text{ I'm hungry} \)
\( \beta: \text{ I eat} \)

\[ \alpha \rightarrow \beta, \quad \alpha \]
\[ \beta \]

\( \alpha \land \alpha_2 \land \ldots \land \alpha_n \)
\[ \alpha_i \]

\( \alpha_1, \alpha_2, \ldots, \alpha_n \)
\[ \alpha_1 \land \alpha_2 \land \ldots \land \alpha_n \]

\( \alpha \lor \alpha_2 \lor \ldots \lor \alpha_n \)
\[ \alpha_i \]

\( \alpha \lor \beta, \quad \neg \beta \)
\[ \alpha \quad \text{disjunctive syllogism} \]

\( \alpha \lor \beta, \quad \neg \beta \lor \gamma \)
\[ \alpha \lor \gamma \]

or equivalently
\[ \neg \alpha \rightarrow \beta, \quad \beta \rightarrow \gamma \]
\[ \neg \alpha \rightarrow \gamma \]
**PS: Resolution**

- **Conjunctive Normal Form (CNF)**
  - conjunction of disjunctions of literals/ clauses
  
  \[ (A \lor \neg B) \land (B \lor \neg C \lor \neg D) \]

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

  \[
  \frac{l_i \lor ... \lor l_k, \quad (m_1 \lor ... \lor m_n)}{l_i \lor ... \lor l_{i-1} \lor l_{i+1} \lor ... \lor l_k \lor m_1 \lor ... \lor m_{j-1} \lor m_{j+1} \lor ... \lor m_n}
  \]

  where \( l_i \) and \( m_j \) are complementary literals.

  \[
  \text{E.g.,} \quad (P_{1,3} \lor P_{2,2}), \quad \neg P_{2,2} \\
  \]

- **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 \( KB \models \alpha \)
- *i.e., show \((KB \land \neg \alpha)\) is unsatisfiable*

1) First convert \((KB \land \neg \alpha)\) 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 \( \alpha \))
   b) two clauses *resolve to yield empty clause*
      (KB entails \( \alpha \))

Resolution Example

- There is no PIT in (1,2)?
- \( KB = (B_{1,1} \iff (P_{1,2} \lor P_{2,1})) \land \neg B_{1,1} \)
- \( \alpha = \neg P_{1,2} \)
PS: Resolution Algorithm

- Proof by Contradiction of \( KB \models \alpha \)

```python
def PL-RESOLUTION(KB, \alpha)
    returns true or false
    clauses ← the set of clauses in the CNF representation of \( KB \land \neg \alpha \)
    new ← \{ \}
    loop do
        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 \cup resolvents
            if new \subseteq clauses then return false
        clauses ← clauses \cup new
    end loop
    return false
```

PS: Forward & Backward Chaining

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

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

- **Modus Ponens:**
  \[
  \alpha_1, \ldots, \alpha_n, \alpha_1 \land \ldots \land \alpha_n \Rightarrow \beta
  \]

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

\[
P \Rightarrow Q \\
L \land M \Rightarrow P \\
B \land L \Rightarrow M \\
A \land P \Rightarrow L \\
A \land B \Rightarrow L \\
A \\
B
\]

**PS: FC Algorithm Example**

\[
KB \models Q ?
\]

\[
P \Rightarrow Q \\
L \land M \Rightarrow P \\
B \land L \Rightarrow M \\
A \land P \Rightarrow L \\
A \land B \Rightarrow L \\
A \\
B
\]
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
**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 8x8 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!!!