Note

- Submit HW1 now.
- Fast Prototyping Exercise #1 on PCA starts next week
  - HW assignment: Read the reference paper: https://bidal.sfsu.edu/~kazokada/csc872/PD1.pdf
  - HW assignment: Continue studying MATLAB

Knowledge-Based Agents with Propositional Logic

CSC 872
Pattern Analysis and Machine Intelligence
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; \( 2x + y > 0 \) 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:
  \[ \text{KB} \models \alpha \]

- Knowledge base \( \text{KB} \) entails sentence \( \alpha \) if and only if \( \alpha \) is true in all worlds where \( \text{KB} \) is true

- E.g., the \( \text{KB} \) 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

**Facts**

- Representation: Facts follows Sentence
- Refers to (Semantics)
- World

```
Knowledge Base (KB) entails Sentence (α)
```

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 \( α \) if \( α \) is true in \( m \).

- \( M(α) \) is the set of all models of \( α \).

- \( KB \models α \) iff \( M(KB) \subseteq M(α) \)
  - \( KB = \text{Giants won and 49ers won} \)
  - \( α = \text{either Giants or 49ers won} \)
Entailment in the Wumpus World

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

PF: Logical Inference

- $KB \models \alpha$
  - sentence $\alpha$ can be derived from $KB$ by procedure $i$
  - $\alpha$ is inferred from $KB$ by using procedure $i$
  - Query “Is $\alpha$ true given $KB$?” is proven true by “$i$”
  - Deductive Reasoning

- Property of the inference procedure “$i$”
- **Soundness:** all inference is consistent
  - “$i$” is sound if whenever $KB \models \alpha$, it is also true that $KB \not\models \alpha$
- **Completeness:** all consistent is inference
  - “$i$” is complete if whenever $KB \not\models \alpha$, it is also true that $KB \models \alpha$

$\text{Sound & Complete } i : \; KB \models \alpha \leftrightarrow KB \not\models \alpha$
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. \( S_1 \), \( S_2 \), \( S_3 \)
  - \( m_1 \):
    - \( S_1 \): false
    - \( S_2 \): true
    - \( S_3 \): false
  - \( m_2 \):
    - \( S_1 \): true
    - \( S_2 \): true
    - \( S_3 \): false

Rules for evaluating truth with respect to a model m:

- \( \neg S \) is true iff \( S \) is false
- \( S_1 \land S_2 \) is true iff \( S_1 \) is true and \( S_2 \) is true
- \( S_1 \lor S_2 \) 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., 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
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 $[i, j]$.

<table>
<thead>
<tr>
<th></th>
<th>$P_{1,1}$</th>
<th>$P_{1,2}$</th>
<th>$P_{2,1}$</th>
<th>$P_{2,2}$</th>
<th>$P_{3,1}$</th>
</tr>
</thead>
<tbody>
<tr>
<td>$P_{2,1}$</td>
<td>$B_{1,1}$</td>
<td>$B_{1,2}$</td>
<td>$B_{2,1}$</td>
<td>$B_{2,2}$</td>
<td>$B_{3,1}$</td>
</tr>
</tbody>
</table>

- "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>false</td>
<td>true</td>
<td>true</td>
</tr>
<tr>
<td>false</td>
<td>true</td>
<td>true</td>
<td>false</td>
<td>false</td>
<td>true</td>
<td>false</td>
</tr>
<tr>
<td>true</td>
<td>false</td>
<td>false</td>
<td>true</td>
<td>true</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!!!
- Proof methods divide into (roughly) two kinds:
  - Application of inference rules
    - 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
    - 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, \( \neg \text{False} \), \( A \Rightarrow A \), \( (A \land (A \Rightarrow B)) \Rightarrow B \)

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

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

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

e.g., \( A \land \neg A \), False, \( \neg \text{True} \)

Validity is connected to inference via the **Deduction Theorem**:
\[ KB \vdash \alpha \iff (KB \Rightarrow \alpha) \text{ is valid} \]

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**
  \[ \alpha \equiv \beta \iff \alpha \models \beta \text{ and } \beta \models \alpha \]

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

- **Modus Ponens or Implication-Elimination**: (From an implication and the premise of the implication, you can infer the conclusion.)
  \[ \frac{\alpha \Rightarrow \beta, \quad \alpha}{\beta} \]

- **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.)
  \[ \frac{\alpha_1, \alpha_2, \ldots, \alpha_n}{\alpha_1 \land \alpha_2 \land \ldots \land \alpha_n} \]

- **Or-Introduction**: (From a sentence, you can infer its disjunction with anything else at all.)
  \[ \frac{\alpha_i}{\alpha_i \lor \alpha_2 \lor \ldots \lor \alpha_n} \]

---

(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.)
  \[ \frac{\alpha \lor \beta, \quad \neg \beta}{\alpha} \]

- **Resolution**: (This is the most difficult. Because \( \beta \) cannot be both true and false, one of the other disjuncts must be true in one of the premises. Or equivalently, implication is transitive.)
  \[ \frac{\alpha \lor \beta, \quad \neg \beta \lor \gamma}{\alpha \lor \gamma} \]
  or equivalently
  \[ \frac{\neg \alpha \Rightarrow \beta, \quad \beta \Rightarrow \gamma}{\neg \alpha \Rightarrow \gamma} \]
PS: Resolution

- **Conjunctive Normal Form (CNF)**
  - conjunction of disjunctions of literals/clauses
  
  \[ (\bigvee \bigwedge (\forall i \forall \overrightarrow{\ell}), \bigwedge (\forall j \forall \overrightarrow{m}) \] clauses

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

\[
\begin{align*}
\left( l_i \lor \cdots \lor l_k \right), &

\left( m_1 \lor \cdots \lor m_n \right) \\
\hline
\left( l_1 \lor \cdots \lor l_{i-1} \lor l_{i+1} \lor \cdots \lor l_k \right), &

\left( m_1 \lor \cdots \lor m_{j-1} \lor m_{j+1} \lor \cdots \lor m_n \right)
\end{align*}
\]

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

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

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

PS: Conversion to CNF (example)

\[ B_{1,1} \iff (P_{1,2} \lor P_{2,1}) \]

1. **Eliminate** \( \iff \), replacing \( \alpha \iff \beta \) with \( (\alpha \implies \beta) \land (\beta \implies \alpha) \).
   
   \[ (B_{1,1} \implies (P_{1,2} \lor P_{2,1})) \land ((P_{1,2} \lor P_{2,1}) \implies B_{1,1}) \]

2. **Eliminate** \( \implies \), replacing \( \alpha \implies \beta \) 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** \( (\land \text{ over } \lor) \) and flatten:
   
   \[ (\neg B_{1,1} \lor P_{1,2} \lor P_{2,1}) \land (\neg P_{2,1} \lor B_{1,1}) \land (\neg P_{2,1} \lor B_{1,1}) \]
**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 $KB \models \alpha$

```java
function 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:**
  \[ \frac{\alpha_1, \ldots, \alpha_n}{\alpha_1 \land \ldots \land \alpha_n \Rightarrow \beta} \]

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

```plaintext
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!!!