Note

• Homework #2
  – On Lecture 4-5
  – Due in one week
  – Submit your answers on 10/3 Tuesday 4pm in class

• Fast Prototyping Exercise starts today
  – You must have read the reference paper:

Knowledge-Based Agents with First-Order Predicate Logic

CSC 872
Pattern Analysis and Machine Intelligence
Review

• **Limitation of Propositional Logic**

  - Even for a simple problem like the Wumpus world, KB gets quickly very large.
  - This is because *Propositional Logic* only represents facts in the world.

Propositional Logic Revisited

**Ontological commitments:** what a language assumes about the nature of the *world*.

**Representation:**

- **Sentences** entail **Sentence**
- Refers to (Semantics)

**World**

- **Facts** follows **Fact**
First Order Logic (FOL)

- First-order logic (like natural language) assumes the world contains:
  - **Objects**: car, wheel, door, body, engine, seat, passenger, driver
  - **Relations**: Properties (unary): Red(car), Healthy(body), IsOn(engine)  
    N-ary relations: Inside(car, passenger), Bigger(car, driver)
  - **Functions**: ColorOf(car), MakeOf(car), SizeOf(engine)

- **Function** returns an object.
- **Relation** returns a truth value.
Models for FOL

- Propositional: sets of truth values for all prop. symbols
- FOL: same above + objects & relations!!!

<table>
<thead>
<tr>
<th>P</th>
<th>Q</th>
<th>¬P</th>
<th>P ∧ Q</th>
<th>P ∨ Q</th>
<th>P ⇒ Q</th>
<th>P ⇔ Q</th>
</tr>
</thead>
<tbody>
<tr>
<td>false</td>
<td>false</td>
<td>true</td>
<td>false</td>
<td>false</td>
<td>false</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>true</td>
</tr>
<tr>
<td>true</td>
<td>false</td>
<td>false</td>
<td>false</td>
<td>true</td>
<td>false</td>
<td>false</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>

Syntax: Basic Elements

Example Symbols

- Constants: King, 2, Wumpus,...
- Predicates: Brother, >, HasBreeze,...
- Functions: LeftLegOf, Sqrt,...
- Variables: x, y, a, b,...
- Connectives: ¬, ⇒, ∧, ∨, ⇔
- Equality: =
- Quantifiers: ∀, ∃
Syntax: Terms

- **Term**: logical expression referring to an object

  \[
  \text{Term} \rightarrow \text{Function(Term, \ldots)} \\
  \quad | \text{Constant} \\
  \quad | \text{Variable}
  \]

- **Examples**:
  - TeacherOf(Cody), StudentOf(Kaz)
  - Kaz, Sabiha, Ajinkya, Soumithri, Arjan, Raya, Prateek, Shubhankar, Rohan, Saman, David, Ahmed…
  - \(x\), referring to \{Reed, Ryan,\ldots | all students in csc872\}

Syntax: Atomic Sentences

- **Atomic sentence**: a sentence that states a fact (indicating a proposition)

  \[
  \text{AtomicSentence} \rightarrow \text{Predicate(Term, \ldots)} \\
  \quad | \text{Term = Term}
  \]

- **Examples**:
  - Classmate(Jianhong, Umang),
  - TeacherOf(Khanh) = TeacherOf(Jose)
Syntax: Complex Sentences

Sentence → AtomicSentence
| (Sentence Connective Sentence)
| Quantifier Variable, … Sentence
| ¬ Sentence

• Examples:
  – ((S1 ∧ S2) ∨ S3), ((S1 ∨ S3) ⇒ S2), (S1 ⇔ S3)
  – Student(Kaz, Raghav) ⇒ Teacher(Raghav, Kaz)
  – ∀ x Smart(x, CSC872)
  – ¬ ∀ x Perfect(x, CSC872)

Syntax: Predicate & Function

• Function := Term
  – Returns an object

• Predicate := Sentence
  – Either true or false

• Correspondence between them
  – Function: father_of(Mary) = Bill
  – Predicate: father_of(Mary, Bill)
Semantics: Models & Interpretations

- **Model + Interpretation** → truth of sentences
- **Model** contains
  (objects=domain elements, relations)
- **Interpretation** specifies referents for symbols
  - constant symbols → objects
  - predicate symbols → relations
  - function symbols → functional relations

Semantics: Truth Assignments

- `predicate(term_1,...,term_n)` is true
  iff the objects referred to by `term_1,...,term_n` are in the relation referred to by `predicate`
- `term_1 = term_2` is true under an interpretation
  iff `term_1` and `term_2` refer to the same object
- For complex sentences: Use the same rules for propositional logic
Quantifiers

• Goal: expressing sentences about \textit{collections} of objects without enumeration
• Aim: avoid naming every constants (propositional logic must do this: e.g. Wumpus)

• Variable: x, y, a, b
• Universal quantification (for all): \( \forall \)
  – e.g., All students in CS872 are smart
• Existential quantification (there exists): \( \exists \)
  – e.g., Someone in the class is sleeping

Universal Quantification (for all): \( \forall \)

Syntax: \( \forall <\text{variables}> <\text{sentence}> \)

• English: “All students in CS872 are smart”:
• FOL: \( \forall x \ In872(x) \Rightarrow Smart(x) \)

\( \forall x \) corresponds to the \textit{conjunction} of all instantiations of \( x \)

\[
\begin{align*}
\text{In872( Weisong)} & \Rightarrow \text{Smart(Weisong)} \\
\wedge \text{In872( Ray)} & \Rightarrow \text{Smart(Ray)} \\
\wedge \ldots & \\
\wedge \text{In872( Trent)} & \Rightarrow \text{Smart(Trent)}
\end{align*}
\]
Universal Quantification (for all): $\forall$

- $\Rightarrow$ is a natural connective to use with $\forall$
- Common mistake: to use $\land$ together with $\forall$

- e.g: $\forall \ x \ \text{In(CS872, } x) \land \text{Smart}(x)$
- means “every one is in CS872 and everyone is smart”

Existential quantification (there exists): $\exists$

Syntax $\exists \ <\text{variables}> \ <\text{sentence}>$

- English: “Someone in the class is sleeping”:
- FOL: $\exists \ x \ \text{In(CS872, } x) \land \text{Sleeping}(x)$

- $\exists \ x$ corresponds to the disjunction of all instantiations of $x$

$$
(\text{In(CS872, Sudha)} \land \text{Sleeping(Sudha)}) \lor \\
(\text{In(CS872, Ammar)} \land \text{Sleeping(Ammar)}) \lor \\
... \lor \\
(\text{In(CS872, Gary)} \land \text{Sleeping(Gary)})
$$
Existential quantification (there exists): ∃

• ∧ is a natural connective to use with ∃

• Common mistake: to use ⇒ together with ∃

  – e.g: ∃ x \( \text{In(CS872, x)} \Rightarrow \text{Sleeping(x)} \)
  – True even for someone not in CS872!
  – (False ⇒ True/False) are Valid!

Properties of quantifiers

∀x ∀y is the same as ∀y ∀x (why??)
∃x ∃y is the same as ∃y ∃x (why??)
∃x ∀y is not the same as ∀y ∃x

∃x ∀y Loves(x, y)
"There is a person who loves everyone in the world"
∀y ∃x Loves(x, y)
"Everyone in the world is loved by at least one person"

Quantifier duality: each can be expressed using the other
∀x Likes(x, IceCream) → ∃x ¬Likes(x, IceCream)
∃x Likes(x, Broccoli) → ∀x ¬Likes(x, Broccoli)
Translating FOL to English

The kinship domain:

- \( \forall x,y \ broccoli(x,y) \Rightarrow \ sibling(x,y) \)
  - Brothers are siblings

- \( \forall m,c \ mother(c) = m \Rightarrow (female(m) \land parent(m,c)) \)
  - One's mother is one's female parent

- \( \forall x,y \ sibling(x,y) \Leftrightarrow sibling(y,x) \)
  - "Sibling" is symmetric

Translating English to FOL

- Every gardener likes the sun.
  \( \forall x \ gardener(x) \Rightarrow likes(x, Sun) \)

- You can fool some of the people all of the time.
  \( \exists x \forall t \ (person(x) \land time(t)) \Rightarrow can-fool(x,t) \)

- You can fool everyone some of the time.
  \( \forall x \exists t \ (person(x) \land time(t)) \Rightarrow can-fool(x,t) \)

- All purple mushrooms are poisonous.
  \( \forall x \ (mushroom(x) \land purple(x)) \Rightarrow poisonous(x) \)

- No purple mushroom is poisonous.
  \( \neg \exists x \ broccoli(x) \land purple(x) \land poisonous(x) \)
  \( \equiv \ \forall x \ (mushroom(x) \land purple(x)) \Rightarrow \neg poisonous(x) \)
Higher-Order Logic

- First-order logic allows quantification over objects (= the first-order entities that exist in the world).

- Higher-order logic also allows quantification over relations and functions.

  e.g., “two objects are equal iff all properties applied to them are equivalent”:
  \[ \forall x, y \ (x=y) \iff (\forall p, p(x) \iff p(y)) \]

- Higher-order logics are more expressive than first-order; however, so far we have little understanding on how to effectively reason with sentences in higher-order logic (no sound and complete inference procedure known)

Using FOL in KB Agent

- Ground Terms: a term without variables
- Substitution/Binding List: \{variable/GTerm\}
  - Given a sentence \( S \) e.g., Faster(x,y)
  - Given a substitution \( \sigma \) e.g., \( \{x/Car\}, \{y/Turtle\} \)
  - Plugging \( \sigma \) to \( S \): \( S_\sigma \) e.g., Faster(Car, Turtle)

- Knowledge-Based Agent
  - ASK(\( KB, S \))
  - KB Agent answers if KB entails \( S \) with any \( \sigma \); and also return some/all \( \sigma \) such that \( KB \models S_\sigma \)

- Example
  - TELL(\( KB, Percept([\{Smell,Breeze,\ldots\}], t=5) \))
  - ASK(\( KB, \exists a \ Action(a, t=5) \))
  - Answers: Yes (True) \( \{a/Shoot\} \)
Inference for FOL

- **Propositionalization**
- **Lifting**
  - Generalized Modus Ponens
  - Unification
  - Forward chaining
  - Backward chaining
  - Resolution

Universal Instantiation: UI

- A new inference rule for FOL for eliminating $\forall$ for any sentence $\alpha$, variable $x$ and ground term $\tau$, $\forall x \alpha \vdash \alpha[x/\tau]$

- e.g., $\forall x \ King(x) \land Greedy(x) \Rightarrow Evil(x)$
  - $\forall x \ King(x) \land Greedy(x) \Rightarrow Evil(x)$
  - $\forall x \ King(x) \land Greedy(x) \Rightarrow Evil(x)$
  - $\forall x \ King(x) \land Greedy(x) \Rightarrow Evil(x)$
  - $\forall x \ King(x) \land Greedy(x) \Rightarrow Evil(x)$
Existential Instantiation: EI

- A new inference rule for FOL for eliminating $\exists$ for any sentence $\alpha$, variable $x$ and constant symbol $k$ not in KB,

$$\exists x \alpha \quad \frac{\alpha(x_1) \lor \alpha(x_2) \lor \ldots \lor \alpha(x_n)}{\alpha \{x/k\} \quad k \notin U}$$

- e.g., $\exists x \text{Crown}(x) \land \text{OnHead}(x, John)$

$$\frac{\text{Crown}(C_1) \land \text{OnHead}(C_1, John)}{C_1 \notin U}$$

- $C_1$ Skolem constant

Propositionalization

- Every KB in FOL can be propositionaled so as to preserve entailment
  - Reducing KB in FOL to the new one in PL using UI and EI rules
  - Perform propositional inference on the new KB

- Herbrand (1930) If a sentence $\alpha$ is entailed by an FOL KB, it is entailed by a finite subset of the propositionalized KB

- Idea: For $n = 0$ to $\infty$ do
  - create a propositional KB by instantiating with depth-$n$ terms
  - see if $\alpha$ is entailed by this KB
Problems of Propositionalization

- Infinitely many ground terms
  - e.g., \( \text{Father}(\text{Father}(\text{Father}(\text{John}))) \)

- Generate lots of irrelevant sentences

- The approach based on Herbrand’s theorem works only for entailed sentences

- Turing-Church Theorem (1936):
  - Entailment for FOL is semidecidable
  - Algorithms exist that say yes to every entailed sentence, but no algorithm exists that also says no to every nonentailed sentence.

Lifting

- Raising an inference rule from propositional to first-order logic and apply them directly to FOL KB

- Generalized Modus Ponens (GMP)
  \[
  p_1, p_2, \ldots, p_n, \quad (p_1 \land p_2 \land \ldots \land p_n \Rightarrow q) \quad \text{where } p'_i \theta = p_i \theta \text{ for all } i
  \]
  \[
  \text{King(John)}, \text{Greedy(y)}, \quad (\text{King(x)} \land \text{Greedy(x)} \Rightarrow \text{Evil(x)})
  \]
  
  with \( \theta = \{x/\text{John}\} \)

- GMP used with KB of definite clauses (Horn clause with exactly one positive literal)
- All variables assumed universally quantified
Unification

- \textbf{Unify(\(\alpha, \beta\)) = \theta\) if \(\alpha\theta = \beta\theta\)
- \(\theta\) is called unifier

\[
\begin{align*}
\text{Knows}(\text{John},x) & \quad \text{Knows}(\text{John},\text{Jane}) \quad \{x/\text{Jane}\} \\
\text{Knows}(\text{John},x) & \quad \text{Knows}(y,\text{OJ}) \quad \{x/\text{OJ},y/\text{John}\} \\
\text{Knows}(\text{John},x) & \quad \text{Knows}(y,\text{Mother}(y)) \quad \{x/\text{Mother}(\text{John}),y/\text{John}\} \\
\text{Knows}(\text{John},x) & \quad \text{Knows}(x,\text{OJ}) \quad \{\text{fail}\}
\end{align*}
\]

- \textbf{Standarizing apart}
  - Changing \(x\) in \text{Knows}(x,\text{OJ})\) to \(z\) that does not clash

- \textbf{Most General Unifier}
  - Unique unifier that is most general when more than one unifier possible

Forward & Backward Chaining

- \textbf{Transform KB to Definite Clauses}
- \textbf{Apply Generalized Modus Ponens}
  - From premises: Forward Chaining
  - From conclusion: Backward Chaining
- \textbf{Sound and Complete for FOL in definite clauses}

- \textbf{Forward Chaining}
  - Basis for Deductive Databases (e.g., expert systems)
- \textbf{Backward Chaining}
  - Basis for Logical Programming (e.g., Prolog)
Resolution

- Forward & backward chaining are NOT complete in general FOL
- Complete theorem (Gödel, 1930)
  - any sentence entailed by a set of sentences can be proven from that set.
  - shows that it is possible to find sound and complete inference rules.
- Resolution (Robinson, 1965)
  - Proof by contradiction: (KB ∧ ¬α) is unsatisfiable → KB ⊨ α
  - Conjunctive Normal Form
  - Apply the resolution inference rule sequentially until finding unsatisfiable sentence (empty clause).

Lifted Resolution Rule

\[ \frac{\alpha \lor \beta, \neg \beta \lor \gamma}{\alpha \lor \gamma} \]
\[ \frac{p_1 \lor \ldots p_j \ldots \lor p_m,}{q_1 \lor \ldots q_k \ldots \lor q_n} \]
\[ \frac{(p_1 \lor \ldots p_{j-1} \lor p_{j+1} \ldots p_m \lor q_1 \lor \ldots q_{k-1} \lor q_{k+1} \ldots \lor q_n) \sigma}{U_{h,R}(F, \neg G) \sigma} \]
where \[ p_j \sigma = \neg q_k \sigma \]

For example,

\[ \frac{\neg Rich(x) \lor Unhappy(x)}{Rich(Me)} \]
\[ \frac{Unhappy(Me)}{\neg Rich(x) \lor Unhappy(x)} \]

with \[ \sigma = \{x/Me\} \]
**CNF Conversion for FOL**

\[(\neg v_1 \lor \neg v_2) \land (\neg v_3) \land (v_4 \lor \neg v_5) \land v_6. \quad \]

Any FOL KB can be converted to CNF as follows:

1. Replace \( P \Rightarrow Q \) by \( \neg P \lor Q \)
2. Move \( \neg \) inwards, e.g., \( \neg \forall x \, P \) becomes \( \exists x \, \neg P \)
3. Standardize variables apart, e.g., \( \forall x \, P \lor \exists x \, Q \) becomes \( \forall x \, P \lor \exists y \, Q \)
4. Move quantifiers left in order, e.g., \( \forall x \, P \lor \exists x \, Q \) becomes \( \forall x \exists y \, P \lor Q \)
5. Eliminate \( \exists \) by Skolemization (next slide)
6. Drop universal quantifiers
7. Distribute \( \land \) over \( \lor \), e.g., \( (P \land Q) \lor R \) becomes \( (P \lor Q) \land (P \lor R) \)

**Resolution Example**

- \( \neg \text{American}(y) \lor \text{Weapon}(y) \lor \text{Sells}(x,y) \lor \text{Hostile}(x) \lor \text{Criminal}(x) \)
- \( \neg \text{Criminal}(x) \)
- \( \neg \text{American}(y) \lor \neg \text{Weapon}(y) \lor \text{Sells}(x,y) \lor \text{Hostile}(x) \)
- \( \neg \text{American}(x) \lor \neg \text{Weapon}(x) \lor \text{Sells}(x,x) \lor \text{Hostile}(x) \)
- \( \neg \text{Sells}(x,x) \lor \text{Hostile}(x) \lor \text{Criminal}(x) \)
- \( \neg \text{Hostile}(x) \lor \text{Criminal}(x) \)
- \( \neg \text{American}(x) \lor \neg \text{Weapon}(x) \lor \text{Sells}(x,x) \lor \text{Hostile}(x) \)
- \( \neg \text{American}(y) \lor \neg \text{Weapon}(y) \lor \text{Sells}(x,y) \lor \text{Hostile}(x) \)
- \( \neg \text{Sells}(x,x) \lor \text{Hostile}(x) \lor \text{Criminal}(x) \)
- \( \neg \text{Hostile}(x) \lor \text{Criminal}(x) \)
- \( \neg \text{American}(y) \lor \neg \text{Weapon}(y) \lor \text{Sells}(x,y) \lor \text{Hostile}(x) \)
- \( \neg \text{American}(x) \lor \neg \text{Weapon}(x) \lor \text{Sells}(x,x) \lor \text{Hostile}(x) \)
- \( \neg \text{Sells}(x,x) \lor \text{Hostile}(x) \lor \text{Criminal}(x) \)
- \( \neg \text{Hostile}(x) \lor \text{Criminal}(x) \)
- \( \neg \text{American}(y) \lor \neg \text{Weapon}(y) \lor \text{Sells}(x,y) \lor \text{Hostile}(x) \)
- \( \neg \text{American}(x) \lor \neg \text{Weapon}(x) \lor \text{Sells}(x,x) \lor \text{Hostile}(x) \)
- \( \neg \text{Sells}(x,x) \lor \text{Hostile}(x) \lor \text{Criminal}(x) \)
- \( \neg \text{Hostile}(x) \lor \text{Criminal}(x) \)
## More Complex Applications

- **Ontological Engineering (Ch.10)**
  - Objects, Categories, Inheritance, Taxonomic hierarchy

- **Situation Calculus (Ch.10)**
  - Conventions for describing actions and changes

- **Planning (Ch.11)**
  - Deriving a sequence of actions that will achieve a goal
  - Can formulate planning as inference on a situation calculus KB

## Logic/Knowledge-Based Agents?

- You are driving a car
- You are now at an intersection
- You check the traffic signal
- Signal turns to green
- You move forward
- A car suddenly run over and clash you and you die.
- What went wrong as an agent-based picture???

- Multi-agent (different agents different models)
- Stochastic/Dynamic environment
- Probability Theory (Facts + Uncertainty)
Summary

• FOL is more powerful
  – Objects + Relations as Ontological Commitments
  – Quantifiers introduced

• Sound and Complete Inference available
  – Propositionalization
  – Forward & backward chaining
  – Resolution

• Next
  – Ontological Commitment: Facts + Uncertainty?
  – Probability Theory
  – Bayesian Framework

• Fast Prototyping Exercise #1 after the break.