Instructor: Sofus A. Macskassy, macskass@usc.edu
TAs: Nadeesha Ranashinghe (nadeeshr@usc.edu)
     William Yeoh (wyeoh@usc.edu)
     Harris Chiu (chiciu@usc.edu)

Lectures: MW 5:00-6:20pm, OHE 122 / DEN
Office hours: By appointment
Class page: http://www.rcf.usc.edu/~macskass/CS561-Spring2010/

This class will use http://www.uscden.net/ and class webpage
    - Up to date information
    - Lecture notes
    - Relevant dates, links, etc.

Course material:
Logistics – HW2

- Homework #2 was due today
- You should have submitted it before class on turnitin
Logistics - MIDTERM

• Midterm 1 is next week

• Date: March 1
• Location: SGM 124
• Time: 5pm – 6:20pm

DEN Students: should have received an email to set up their exam

• Covers: All lectures through this week (Ch. 1-8)
  ◦ It is open book and open notes
  ◦ You can use the book, lecture slides and your notes

• Example midterms available from AIMA site
  ◦ http://aima.cs.berkeley.edu/instructors.html
First-order Logic [AIMA Ch. 8]

- Why FOL?
- Syntax and semantics of FOL
- Fun with sentences
- Wumpus world in FOL
Review: Propositional logic - syntax

- Propositional logic is the simplest logic—illustrates basic ideas
- The proposition symbols $P_1, P_2$ etc are sentences
- If $S$ is a sentence, $\neg S$ is a sentence (negation)
- If $S_1$ and $S_2$ are sentences, $S_1 \wedge S_2$ is a sentence (conjunction)
- If $S_1$ and $S_2$ are sentences, $S_1 \lor S_2$ is a sentence (disjunction)
- If $S_1$ and $S_2$ are sentences, $S_1 \Rightarrow S_2$ is a sentence (implication)
- If $S_1$ and $S_2$ are sentences, $S_1 \Leftrightarrow S_2$ is a sentence (biconditional)
Review: Propositional logic - Semantics

Each model specifies true/false for each proposition symbol

E.g. \( P_{1,2} \) \( P_{2,2} \) \( P_{3,1} \)
\[
\begin{array}{lll}
true & true & false
\end{array}
\]

(With these symbols, 8 possible models, can be enumerated automatically.)

Rules for evaluating truth with respect to a model \( m \):

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

Simple recursive process evaluates an arbitrary sentence, e.g.,

\[
\neg P_{1,2} \land (P_{2,2} \lor P_{3,1}) = true \land (false \lor true) = true \land true = true
\]
Review propositional logic [cont’d]

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

\[
\frac{\alpha \implies \beta, \quad \alpha}{\beta}
\]

◊ **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.)

\[
\frac{\alpha_i}{\alpha_1 \lor \alpha_2 \lor \ldots \lor \alpha_n}
\]

◊ **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

\[
\frac{\neg \alpha \implies \beta, \quad \beta \implies \gamma}{\neg \alpha \implies \gamma}
\]
Why first-order logic?

Pros and cons of propositional logic

😊 Propositional logic is declarative: pieces of syntax correspond to facts

😊 Propositional logic allows partial/disjunctive/negated information (unlike most data structures and databases)

😊 Propositional logic is compositional:
  meaning of $B_{1,1} \land P_{1,2}$ is derived from meaning of $B_{1,1}$ and of $P_{1,2}$

😊 Meaning in propositional logic is context-independent (unlike natural language, where meaning depends on context)

😢 Propositional logic has very limited expressive power (unlike natural language)
  E.g., cannot say “pits cause breezes in adjacent squares” except by writing one sentence for each square
First-order logic (FOL)

- Whereas propositional logic assumes world contains facts, first-order logic (like natural language) assumes the world contains:
  
  ◦ **Objects**: wheel, door, body, engine, seat, car, passenger, driver, people, houses, numbers, theories, Ronald McDonald, colors, baseball games, wars, centuries, ...
  
  ◦ **Relations**: Inside(car, passenger), Beside(driver, passenger), BrotherOf(person, person), BiggerThan(object, object), Inside(), part of(), HasColor(), OccurredAfter(), Owns(), ComesBetween(), ...
  
  ◦ **Functions**: ColorOf(car), FatherOf(person), BestFriend(person), ThirdInningOf(), OneMoreThan(), EndOf(), ...
  
  ◦ **Properties**: Color(car), IsOpen(door), IsOn(engine)
  
- Functions are relations with single value for each object
Logics in general

- Logics are characterized by what they commit to as “primitives”
- Epistemological commitment: what states of knowledge?

<table>
<thead>
<tr>
<th>Language</th>
<th>Ontological Commitment</th>
<th>Epistemological Commitment</th>
</tr>
</thead>
<tbody>
<tr>
<td>Propositional logic</td>
<td>facts</td>
<td>true/false/unknown</td>
</tr>
<tr>
<td>First-order logic</td>
<td>facts, objects, relations</td>
<td>true/false/unknown</td>
</tr>
<tr>
<td>Temporal logic</td>
<td>facts, objects, relations, times</td>
<td>true/false/unknown</td>
</tr>
<tr>
<td>Probability logic</td>
<td>facts</td>
<td>degree of belief 0...1</td>
</tr>
<tr>
<td>Fuzzy logic</td>
<td>facts, degree of truth</td>
<td>known interval value</td>
</tr>
</tbody>
</table>
Semantics/Interpretation

there is a correspondence between

- functions, which return values
- predicates, which are true or false

Function: father_of(Mary) = Bill
Predicate: father_of(Mary, Bill)
Examples:

- “One plus two equals three”
  Objects:
  Relations:
  Properties:
  Functions:

- “Squares neighboring the Wumpus are smelly”
  Objects:
  Relations:
  Properties:
  Functions:
Examples:

• “One plus two equals three”
  Objects: one, two, three, one plus two
  Relations: equals
  Properties: --
  Functions: plus (“one plus two” is the name of the object obtained by applying function plus to one and two; three is another name for this object)

• “Squares neighboring the Wumpus are smelly”
  Objects: Wumpus, square
  Relations: neighboring
  Properties: smelly
  Functions: --
FOL: Syntax of basic elements

- **Constant symbols:** 1, 5, A, B, USC, JPL, Alex, Manos, ...
- **Predicate symbols:** >, Friend, Student, Colleague, ...
- **Function symbols:** +, sqrt, SchoolOf, TeacherOf, ClassOf, ...
- **Variables:** x, y, z, next, first, last, ...
- **Connectives:** ∧, ∨, ⇒, ⇔
- **Quantifiers:** ∀, ∃
- **Equality:** =
FOL: Atomic sentences

AtomicSentence = \textit{predicate}(\textit{term}_1, \ldots, \textit{term}_n)

or \textit{term}_1 = \textit{term}_2

\textit{Term} = \textit{function}(\textit{term}_1, \ldots, \textit{term}_n)

or \textit{constant} or \textit{variable}

- Examples:
  - \textit{SchoolOf(Manos)}
  - \textit{Colleague(TeacherOf(Alex), TeacherOf(Manos))}
  - \textit{>(((+ x y), x)}
FOL: Complex sentences

Complex sentences are made from atomic sentences using connectives

\[ \neg S, \ S_1 \land S_2, \ S_1 \lor S_2, \ S_1 \Rightarrow S_2, \ S_1 \Leftrightarrow S_2 \]

\[ \text{Sentence} \rightarrow \text{AtomicSentence} \]
\[ \quad | \text{Sentence Connective Sentence} \]
\[ \quad | \text{Quantifier Variable, \ldots Sentence} \]
\[ \quad | \neg \text{Sentence} \]
\[ \quad | (\text{Sentence}) \]

- Examples:
  - \( \text{Colleague(Paolo, Maja)} \Rightarrow \text{Colleague(Maja, Paolo)} \)
  - \( \text{Student(Alex, Paolo)} \Rightarrow \text{Teacher(Paolo, Alex)} \)
Truth in first-order logic

- Sentences in FOL are interpreted with respect to a model and an interpretation
- Model contains $\geq 1$ objects (domain elements) and relations among them
- Interpretation specifies refererents for
  - **Constant symbols**: refer to objects
  - **Predicate symbols**: refer to relations
  - **Function symbols**: refer to functional Relations

- An atomic sentence $\text{predicate}(\text{term}_1, \ldots, \text{term}_n)$ is **true** iff the objects referred to by $\text{term}_1, \ldots, \text{term}_n$ are in the relation referred to by $\text{predicate}$
Models for FOL: Example
Truth example

- Consider the interpretation in which
  - \textit{Richard} $\rightarrow$ Richard the Lionheart
  - \textit{John} $\rightarrow$ the evil King John
  - \textit{Brother} $\rightarrow$ the brother relation

- Under this interpretation, \textit{Brother(\textit{Richard, John})} is true just in case Richard the Lionheart and the evil King John are in the brother relation in the model
Models for FOL: Lots!

- Entailment in propositional logic can be computed by enumerating models.

- We can enumerate the FOL models for a given KB vocabulary:
  - For each number of domain elements $n$ from 1 to $\infty$
    - For each $k$-ary predicate $P_k$ in the vocabulary
      - For each possible $k$-ary relation on $n$ objects
        - For each constant symbol $C$ in the vocabulary
          - For each choice of referent for $C$ from $n$ objects ...

Computing entailment by enumerating FOL models is not easy!
Quantifiers

- Expressing sentences about **collections** of objects without enumeration (naming individuals)

- E.g., All Trojans are clever

- Someone in the class is sleeping

- Universal quantification (for all): $\forall$

- Existential quantification (there exists): $\exists$
Universal quantification (for all): $\forall$

$\forall$ <variables> <sentence>

- "Every one in the cs561 class is smart":
  $\forall x \quad In(cs561, x) \Rightarrow Smart(x)$

- $\forall P$ corresponds to the conjunction of instantiations of $P$
  $In(cs561, Manos) \Rightarrow Smart(Manos) \land$
  $In(cs561, Dan) \Rightarrow Smart(Dan) \land$
  ...
  $In(cs561, Bush) \Rightarrow Smart(Bush)$
Universal quantification (for all): \( \forall \)

\( \Rightarrow \) is a natural connective to use with \( \forall \)

**Common mistake:** to use \( \land \) in conjunction with \( \forall \)

\[ \forall x \ (\text{In}(cs561, x) \land \text{Smart}(x)) \]

means “every one is in cs561 and everyone is smart”
Existential quantification (there exists): $\exists$

$\exists \langle \text{variables} \rangle \langle \text{sentence} \rangle$

- “Someone in the cs561 class is smart”:  
  $\exists x \ In(cs561, x) \land Smart(x)$

- $\exists x \ P$ is true in a model $m$ iff $P$ is true with $x$ being some possible object in the model

- $\exists P$ corresponds to the disjunction of instantiations of $P$  
  $(In(cs561, Manos) \land Smart(Manos)) \lor $  
  $(In(cs561, Dan) \land Smart(Dan)) \lor $  
  $\ldots$  
  $(In(cs561, Bush) \land Smart(Bush))$
Existential quantification (there exists): $\exists$

- $\land$ is a natural connective to use with $\exists$

- **Common mistake:** to use $\Rightarrow$ in conjunction with $\exists$

  $$\exists x \quad In(cs561, x) \Rightarrow Smart(x)$$

- is true if there is anyone that is not in cs561! (remember, false $\Rightarrow$ true is valid).
Examples...

∀x In(cs561, x) ⇒ Smart(x)

∀x In(cs561, x) \land Smart(x)

∃x In(cs561, x) \land Smart(x)

∃x In(cs561, x) ⇒ Smart(x)

¬In(cs561, x) \lor Smart(x)
Examples...

\[ \forall x \, \text{In}(\text{cs561}, x) \Rightarrow \text{Smart}(x) \]

\[ \forall x \, \text{In}(\text{cs561}, x) \land \text{Smart}(x) \]

\[ \exists x \, \text{In}(\text{cs561}, x) \land \text{Smart}(x) \]

\[ \exists x \, \text{In}(\text{cs561}, x) \Rightarrow \text{Smart}(x) \]
Examples...

\[ \forall x \ In(cs561, x) \Rightarrow Smart(x) \]
\[ \forall x \ In(cs561, x) \wedge Smart(x) \]
\[ \exists x \ In(cs561, x) \wedge Smart(x) \]
\[ \exists x \ In(cs561, x) \Rightarrow Smart(x) \]
Examples...

\[ \forall x \; \text{In}(cs561, x) \Rightarrow \text{Smart}(x) \]
\[ \forall x \; \text{In}(cs561, x) \land \text{Smart}(x) \]
\[ \exists x \; \text{In}(cs561, x) \land \text{Smart}(x) \]
\[ \exists x \; \text{In}(cs561, x) \Rightarrow \text{Smart}(x) \]

\[ \neg \text{In}(cs561, x) \]
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)
Fun with sentences

- Brothers are siblings
  \( \forall x, y \ \text{Brother}(x, y) \Rightarrow \text{Sibling}(x, y) \)

- “Sibling” is symmetric
  \( \forall x, y \ \text{Sibling}(x, y) \Rightarrow \text{Sibling}(y, x) \)

- One’s mother is one’s female parent
  \( \forall x, y \ \text{Mother}(x, y) \iff (\text{Female}(x) \land \text{Parent}(x, y)) \)

- A first cousin is a child of a parent’s sibling
  \( \forall x, y \ \text{FirstCousin}(x, y) \iff \exists p, ps \ \text{Parent}(p, x) \land \text{Sibling}(ps, p) \land \text{Parent}(ps, y) \)
Translating English to FOL

- Every gardener likes the sun.
  \[ \forall x \ \text{gardener}(x) \implies \text{likes}(x, \text{Sun}) \]

- You can fool some of the people all of the time.
  \[ \exists x \ \forall t \ (\text{person}(x) \land \text{time}(t)) \implies \text{can-fool}(x, t) \]
**Translating English to FOL**

- You can fool all of the people some of the time.
  \[ \forall x \exists t \ (\text{person}(x) \land \text{time}(t)) \implies \text{can-fool}(x,t) \]

- All purple mushrooms are poisonous.
  \[ \forall x \ (\text{mushroom}(x) \land \text{purple}(x)) \implies \text{poisonous}(x) \]
No purple mushroom is poisonous.

¬(∃ x) purple(x) ^ mushroom(x) ^ poisonous(x)

or, equivalently,

(∀ x) (mushroom(x) ^ purple(x)) => ¬poisonous(x)
Equality

- \( term_1 = term_2 \) is true under a given interpretation if and only if \( term_1 \) and \( term_2 \) refer to the same object

- E.g.,
  \[
  \forall x \ (\text{Sqrt}(x), \text{Sqrt}(x)) = x \text{ is satisfiable}
  \]
  \[
  2 = 2 \text{ is valid}
  \]

- E.g., definition of (full) \textit{Sibling} in terms of \textit{Parent}:
  \[
  \forall x, y \ \text{Sibling}(x, y) \iff \neg(x=y) \land \exists m, f \neg(m=f) \land \text{Parent}(m, x) \land \\
  \text{Parent}(f, x) \land \text{Parent}(m, y) \land \text{Parent}(f, y)
  \]
Higher-order logic?

- First-order logic allows us to quantify 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.
Logical agents for the Wumpus world

Remember: generic knowledge-based agent:

```plaintext
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
```

1. TELL KB what was perceived
   Uses a KRL to insert new sentences, representations of facts, into KB

2. ASK KB what to do.
   Uses logical reasoning to examine actions and select best.
Interacting with FOL KBs

• Suppose a wumpus-world agent is using an FOL KB and perceives a smell and a breeze (but no glitter) at \( t = 5 \):

\[
\text{Tell}(KB, \text{Percept}([\text{Smell}, \text{Breeze}, \text{None}], 5)) \\
\text{Ask}(KB, \exists a \text{ Action}(a, 5))
\]

• I.e., does \( KB \) entail any particular actions at \( t = 5 \)?
  Answer: \( \text{Yes, } \{a/\text{Shoot}\} \leftarrow \text{substitution} \) (binding list)

• Given a sentence \( S \) and a substitution \( \alpha \),
  \( S \) denotes the result of plugging \( \alpha \) into \( S \); e.g.,

\[
\begin{align*}
S &= \text{Smarter}(x, y) \\
\alpha &= \{x/\text{Hillary}, y/\text{Bill}\} \\
S\alpha &= \text{Smarter}(\text{Hillary}, \text{Bill})
\end{align*}
\]

• \( \text{Ask}(KB, S) \) returns some/all such that \( KB \models S \)
Knowledge base for the wumpus world

- "Perception"
  \[ \forall b, g, t \ Percept([Smell, b, g], t) \Rightarrow Smelt(t) \]
  \[ \forall s, b, t \ Percept([s, b, Glitter], t) \Rightarrow AtGold(t) \]

- Reflex: \[ \forall t AtGold(t) \Rightarrow Action(Grab, t) \]

- Reflex with internal state: do we have the gold already?
  \[ \forall t AtGold(t) \land \neg Holding(Gold, t) \Rightarrow Action(Grab, t) \]

- \textit{Holding}(Gold, t) cannot be observed
  \[ \Rightarrow \text{keeping track of change is essential} \]
Deducing hidden properties

- Properties of locations:
  \[ \forall x, t \ At(Agent, x, t) \land Smelt(t) \Rightarrow Smelly(x) \]
  \[ \forall x, t \ At(Agent, x, t) \land Breeze(t) \Rightarrow Breezy(x) \]

- Squares are breezy near a pit:
- Diagnostic rule—infer cause from effect
  \[ \forall y \ Breezy(y) \Rightarrow \exists x \ Pit(x) \land \text{Adjacent}(x, y) \]

- Causal rule—infer effect from cause
  \[ \forall x, y \ Pit(x) \land \text{Adjacent}(x, y) \Rightarrow Breezy(y) \]

- Neither of these is complete—e.g., the causal rule doesn't say whether squares far away from pits can be breezy

- Definition for the Breezy predicate:
  \[ \forall y \ Breezy(y) \Leftrightarrow [\exists x \ Pit(x) \land \text{Adjacent}(x, y)] \]
Situation calculus (Keeping track of change)

- Facts hold in situations, rather than eternally
- E.g., \( \text{Holding}(Gold,\text{Now}) \) rather than just \( \text{Holding}(Gold) \)
- Situation calculus is one way to represent change in FOL:
  
  Adds a situation argument to each non-eternal predicate
  
  E.g., Now in \( \text{Holding}(Gold,\text{Now}) \) denotes a situation
- Situations are connected by the \textbf{Result} function
- \( \text{Result}(a, s) \) is the situation that results from doing \( a \) in \( s \)
Describing actions

- **“Effect” axiom**—describe changes due to action
  \[ \forall s \; \text{AtGold}(s) \Rightarrow \text{Holding}(\text{Gold}, \text{Result}(	ext{Grab}, s)) \]

- **“Frame” axiom**—describe non-changes due to action
  \[ \forall s \; \text{HaveArrow}(s) \Rightarrow \text{HaveArrow}(\text{Result}(	ext{Grab}, s)) \]

- **Frame problem**: find an elegant way to handle non-change
  (a) representation—avoid frame axioms
  (b) inference—avoid repeated “copy-overs” to keep track of state

- **Qualification problem**: true descriptions of real actions require endless caveats—what if gold is slippery or nailed down or ...

- **Ramification problem**: real actions have many secondary consequences—what about the dust on the gold, wear and tear on gloves, ...
Describing actions (cont’d)

- Successor-state axioms solve the representational frame problem

- Each axiom is “about” a predicate (not an action per se):

  \[ P \text{ true afterwards} \iff [\text{an action made } P \text{ true} \]
  \[ \lor P \text{ true already and no action made } P \text{ false} \]

- For holding the gold:

  \[ \forall a, s \ \text{Holding}(Gold, \text{Result}(a, s)) \iff \]
  \[ [(a=\text{Grab} \land \text{AtGold}(s)) \]
  \[ \lor (\text{Holding}(Gold, s) \land a \neq \text{Release})] \]
Making plans

- **Initial condition in** $KB$:
  
  $At(Agent, [1, 1], So)$
  
  $At(Gold, [1, 2], So)$

- **Query:** $Ask(KB, \exists s\; Holding(Gold, s))$
  
  i.e., in what situation will I be holding the gold?

- **Answer:** $\{s/Result(Grab, Result(Forward, So))\}$
  
  i.e., go forward and then grab the gold

- This assumes that the agent is interested in plans starting at $So$ and that $So$ is the only situation described in the $KB$
Making plans: A better way

- Represent plans as action sequences \([a_1, a_2, \ldots, a_n]\)

- \(\text{PlanResult}(p, s)\) is the result of executing \(p\) in \(s\)

- Then the query \(\text{Ask}(KB, \exists p \text{ Holding}(\text{Gold}, \text{PlanResult}(p, S_0)))\)
  has the solution \(\{p/[\text{Forward, Grab}]\}\)

- Definition of \(\text{PlanResult}\) in terms of \(\text{Result}\):
  \[
  \forall s \text{ PlanResult}([], s) = s \\
  \forall a, p, s \text{ PlanResult}([a|p], s) = \text{PlanResult}(p, \text{Result}(a, s))
  \]

- Planning systems are special-purpose reasoners designed to do this type of inference more efficiently than a general-purpose reasoner
Summary

- First-order logic:
  - objects and relations are semantic primitives
  - syntax: constants, functions, predicates, equality, quantifiers

- Increased expressive power: sufficient to define wumpus world

- Situation calculus:
  - conventions for describing actions and change in FOL
  - can formulate planning as inference on a situation calculus KB