Announcements

• HW2 due
• Quiz 2 Tuesday (Adversarial Search, Propositional Logic and Inference)
• Read Chapter 9.1,2,5 (Inference in First order logic)
Today

- First order Logic and inference (chapters 8.1-3, 9.1,2,5)
Key Elements of FOL

• FOL introduces:
  – Objects, named by constant symbols
  – Relations between objects (predicates)
  – Functions that map objects to other objects
  – Variables that act as placeholders for objects
  – Quantifiers that allow us to talk about (infinite) collections of objects
Functions and Predicates

• A **function** takes a set of objects and returns an object, e.g. $Mother(A) = B$

• A **predicate/relation** takes a set of objects and returns true/false, e.g. $isMotherOf(A,B)$?
Syntax: Terms

- A term is a generic representation of an object
- A constant or variable symbol is a term
- If \( t_1, \ldots, t_n \) are terms and \( f \) is a function of arity \( n \), \( f(t_1, \ldots, t_n) \) is a term
- Examples: \( \text{Wumpus}, \text{Pit}, x, y, \text{XWestOf}(x,y) \)
Syntax: Atomic Formulae

• If $P$ is a predicate symbol of arity $n$, and $t_1, \ldots, t_n$ are terms [representing objects], $P(t_1, \ldots, t_n)$ is an atomic formula

• Examples: $\text{Contains}(x, y, \text{Wumpus}), \text{Smelly}(x, y), \text{Breezy}(x, y)$

• Analogous to proposition symbols in propositional logic
Syntax: Complex Formulae

• All atomic formulae are complex formulae
• If $F_1$ and $F_2$ are formulae, so is ($\neg F_1$), ($F_1 \land F_2$), etc (for all connectives)
• If $F_1$ is a formula, so is ($\forall x \ F_1$) and ($\exists x \ F_1$), where $x$ is a variable symbol
• Example:

$$\forall x \ \forall y (\text{Cell}(x,y) \land \text{Contains}(x,y,\text{Pit}) \land \text{Cell}(X\text{NorthOf}(x,y),Y\text{NorthOf}(x,y))) \Rightarrow \text{Breezy}(X\text{NorthOf}(x,y),Y\text{NorthOf}(x,y))$$
Semantics: Quantifiers

• FOL has “quantifiers” ∀ and ∃
• ∀ is the “universal quantifier”, read “For all”
  – ∀xP(x) is a formula that says P is true for every object x
  – Can be thought of as a conjunction over the universe of objects: P(a) ∧ P(b) ∧ P(c)...
• ∃ is the “existential quantifier”, read “There exists”
  – ∃xP(x) is a formula that says there is some object x for which P is true
  – Can be thought of as a disjunction over the universe of objects: P(a) ∨ P(b) ∨ P(c)...
Quantifier nesting

• Quantifiers can be nested:
\[ \exists x \exists y \text{Contains}(x, y, \text{Wumpus}) \]
(or \[ \exists x, y \text{Contains}(x, y, \text{Wumpus}) \] )

• The order of nesting is important
  – \[ \forall x \exists y \text{Friend}(x,y) \] : “Everybody has a friend”
  – \[ \exists y \forall x \text{Friend}(x,y) \] : “There is someone who is everyone’s friend”
Variable scoping

• If two quantifiers use the same variable, the variable “belongs” to the innermost quantifier

\[ \forall x P(x) \land \exists x \forall y Q(x, y) \equiv \]

\[ \forall x P(x) \land \exists a \forall y Q(a, y) \]

• Generally bad idea, don’t do this
  – Variables are placeholders, so can always use new variables as needed
  – Only use the same variable if you need to talk about the same object(s)
Standardizing Apart

• To prevent scoping related confusion, inference engines will generally rename all variables uniquely before inference
  – This is called “standardizing apart”
  – You should do this as well when answering questions, unless it is given that this has been done already
Quantifier relationships

• Observe: if “for all $x$, $P(x)$”, then there is no $x$ for which $\neg P(x)$
• So $\forall x P(x)$ is the same as $\neg \exists y \neg P(y)$
• Similarly, $\exists x P(x)$ is the same as $\neg \forall y \neg P(y)$
• However, using the two symbols improves readability, so we’ll keep them
Definitions

- If a formula has no variables, it is said to be a “ground” formula (vs non-ground) (also “ground fact” or “fact”)
  - Contains(1,2,Wumpus) vs Contains(x, y, Wumpus)

- A variable that appears within the scope of some quantifier is “bound” (vs free)
  - \( \exists x, y \) Contains(x, y, Wumpus) vs. \( \exists x \) Contains(x, y, Wumpus)

- A formula with only bound variables or constants is said to be “closed”
  - We will always work with closed formulae
  - In particular, if you see a formula with a variable but no quantifier, you can assume the variable is universally quantified
( Canonical ) Semantics of FOL

- In propositional logic, a model assigned truth values to propositional symbols
- FOL is much more expressive; a model contains objects that the terms refer to (could be infinite!)
  - The “domain” of a model is the set of objects it has
  - Example: In the wumpus world, the objects are the wumpus, pits, and the numbers representing grid coordinates
Predicates and functions

- A predicate $P(t_1, ..., t_n)$ is a list of tuples of objects for which the relation holds.

- A function $f(t_1, ..., t_n)$ is a mapping from the (cross product of) model’s domain to itself.

- Together with the assignments for constant symbols, these constitute an “interpretation”.
Examples

• Contains \((x, y, z)\)

<table>
<thead>
<tr>
<th>X</th>
<th>Y</th>
<th>Z</th>
</tr>
</thead>
<tbody>
<tr>
<td>*3</td>
<td>*2</td>
<td>*Wumpus</td>
</tr>
<tr>
<td>*1</td>
<td>*3</td>
<td>*Pit</td>
</tr>
<tr>
<td></td>
<td></td>
<td>(etc)</td>
</tr>
</tbody>
</table>

• \text{XWestOf}(x, y)

<table>
<thead>
<tr>
<th>x</th>
<th>y</th>
<th>\text{XWestOf}(X,Y,Wumpus)</th>
</tr>
</thead>
<tbody>
<tr>
<td>*1</td>
<td>*1</td>
<td>*None</td>
</tr>
<tr>
<td>*2</td>
<td>*2</td>
<td>*1</td>
</tr>
<tr>
<td>*3</td>
<td>*1</td>
<td>*2</td>
</tr>
<tr>
<td></td>
<td></td>
<td>(etc)</td>
</tr>
</tbody>
</table>
Compositional Semantics

• FOL semantics are compositional
  – To find the object for $f(t_1, \ldots, t_n)$, find the objects for $t_1, \ldots, t_n$
  – Then look up the table for $f$ to find the object for $f(t_1, \ldots, t_n)$
  – Likewise for predicates
Semantics of a formula

• For atomic formula $P(t_1, ..., t_n)$: Given the interpretation of the terms, does the tuple appear in the interpretation’s list for $P$?

• For complex formulae:
  – Use semantics for logical connectives
Semantics of a formula

• For Quantifiers:
  – Need to consider bindings of the quantified variable
    • Plug in each possible object in the model’s domain in place of the variable and evaluate as below
  – Universal Quantifier: Formula holds if it is true for all possible bindings in the given interpretation
  – Existential Quantifier: Formula holds if a binding exists that makes it true in the given interpretation
Higher order logic

• Propositional logic ("zeroth-order logic") has no concept of objects
• FOL allows objects
• Second order logic allows relations over objects to be objects
  – E.g., can say "there exists a relation that is transitive"
FOL Inference

• As before, we want to find out if certain formulae are entailed by what we know

• In propositional logic, all we were interested in was whether a formula was entailed

• In FOL, we are generally also interested in what variable bindings lead to entailment
FOL Inference

\[ \forall x \ Feathers(x) \land Flies(x) \Rightarrow Bird(x) \]

\[ Feathers(Crow) \]

\[ Flies(Crow) \]

\[ Feathers(Ostrich) \]

\[ \exists y Bird(y) \]
Lifting

• Semantic inference via model checking is no longer possible, because models can be infinite

• To develop syntactic FOL inference, we can take the inference rules we know about from propositional logic and make FOL versions of them
  – This is called “lifting”
(Lifted) Resolution

• Like in propositional logic, resolution is a general purpose proof procedure

• Sound and refutation complete

• In general, not efficient; however, special versions of resolution exist that trade off completeness and efficiency
Resolution inference rule

\[
l_1 \lor l_2 \lor m \ldots \lor l_k, \quad r_1 \lor r_2 \lor \neg m \ldots \lor r_k
\]

\[
l_1 \lor l_2 \ldots \lor l_k \lor r_1 \lor r_2 \ldots \lor r_k
\]

\[\forall x \ Feathers(x) \land Flies(x) \Rightarrow Bird(x)\]

\[Feathers(Crow)\]

\[Flies(Crow)\]

\[Feathers(Ostrich)\]

\[? \exists y Bird(y)\]

\[\forall x \neg Feathers(x) \lor \neg Flies(x) \lor Bird(x), \ Feathers(Crow)\]

???
Lifted resolution inference rule

\[
l_1 \vee l_2 \vee m \ldots \vee l_k, \quad r_1 \vee r_2 \vee \neg m \ldots \vee r_k
\]

\[
l_1 \vee l_2 \ldots \vee l_k \vee r_1 \vee r_2 \ldots \vee r_k
\]

Clause: Universally quantified disjunction

Atomic Formula or negation: \(P(t_1, \ldots, t_n)\) (universally quantified)

Unification: Substitution \(\theta\) makes \(m_i\) syntactically identical to \(n_j\)
Lifted resolution inference rule

\( \forall x \ Feathers(x) \land Flies(x) \Rightarrow Bird(x) \)

\( Feathers(Crow) \)

\( Flies(Crow) \)

\( Feathers(Ostrich) \)

\( ? \exists x Bird(x) \)

\begin{align*}
\forall x \neg Feathers(x) \lor \neg Flies(x) \lor Bird(x), & \neg Flies(Crow) \lor Bird(Crow), \forall y \neg Bird(y) \{y / Crow\} \\
\neg Flies(Crow) \lor Bird(Crow) & \\
Bird(Crow), & \forall y \neg Bird(y) \{y / Crow\} \\
\square & 
\end{align*}
Substitution

• We write $p\{v/c\}$ to mean the formula $p$ with the variable $v$ substituted with the term $c$
  — *Simultaneously* replace every occurrence of $v$ with $c$
  — $c$ is a binding for $v$

• Examples: Suppose $\theta=\{a/f(b,c), b/c\}$, $\sigma=\{b/\text{Const}\}$
  — Then $p(a,b)\theta=p(f(b,c),c)$, $p(a,b)\sigma=p(a,\text{Const})$

• The order of substitution matters
  — $p(a,b)\theta\sigma=p(f(\text{Const},c),c)$, $p(a,b)\sigma\theta=p(f(b,c),\text{Const})$
Unification

• A key step in FOL inference is finding substitutions that make two FOL formulae syntactically identical
  – This is called unification
  – E.g. \( \forall x \text{Feathers}(x) \land \text{Flies}(x) \Rightarrow \text{Bird}(x) \)

  \text{Feathers}(\text{Crow})

  \text{Flies}(\text{Crow})

  – To do FOL inference on this KB, we need to unify \text{Feathers}(x) and \text{Feathers}(\text{Crow})
Examples

\[
\text{Unify}(P(A, x), P(A, C)) = \{x / C\}
\]

\[
\text{Unify}(P(A, x), P(y, B)) = \{x / B, y / A\}
\]

\[
\text{Unify}(P(A, x), Q(m, n)) = \text{fail}
\]

\[
\text{Unify}(P(v, x), P(y, f(u))) = \{x / f(u), v / y\}
\]
Suppose we have two FOL formulae 
\[ P(x, \text{Const1}) \] and \[ P(\text{Const2}, x) \]

- Are these unifiable?
- Yes! Variables are placeholders. Assuming these \( x \)'s come from different universal quantifiers, we can replace the second \( x \) with \( y \).

- This is done during standardizing apart
- Always do this before unification unless it is stated that this has already been done