# EECS 391: Introduction to AI

Soumya Ray

Website: http://vorlon.case.edu/~sray/eecs391_sp12/index.html

Email: sray@case.edu

Office: Olin 516

Office hours: Tue 2:30-4:00 or by appointment

# Announcements

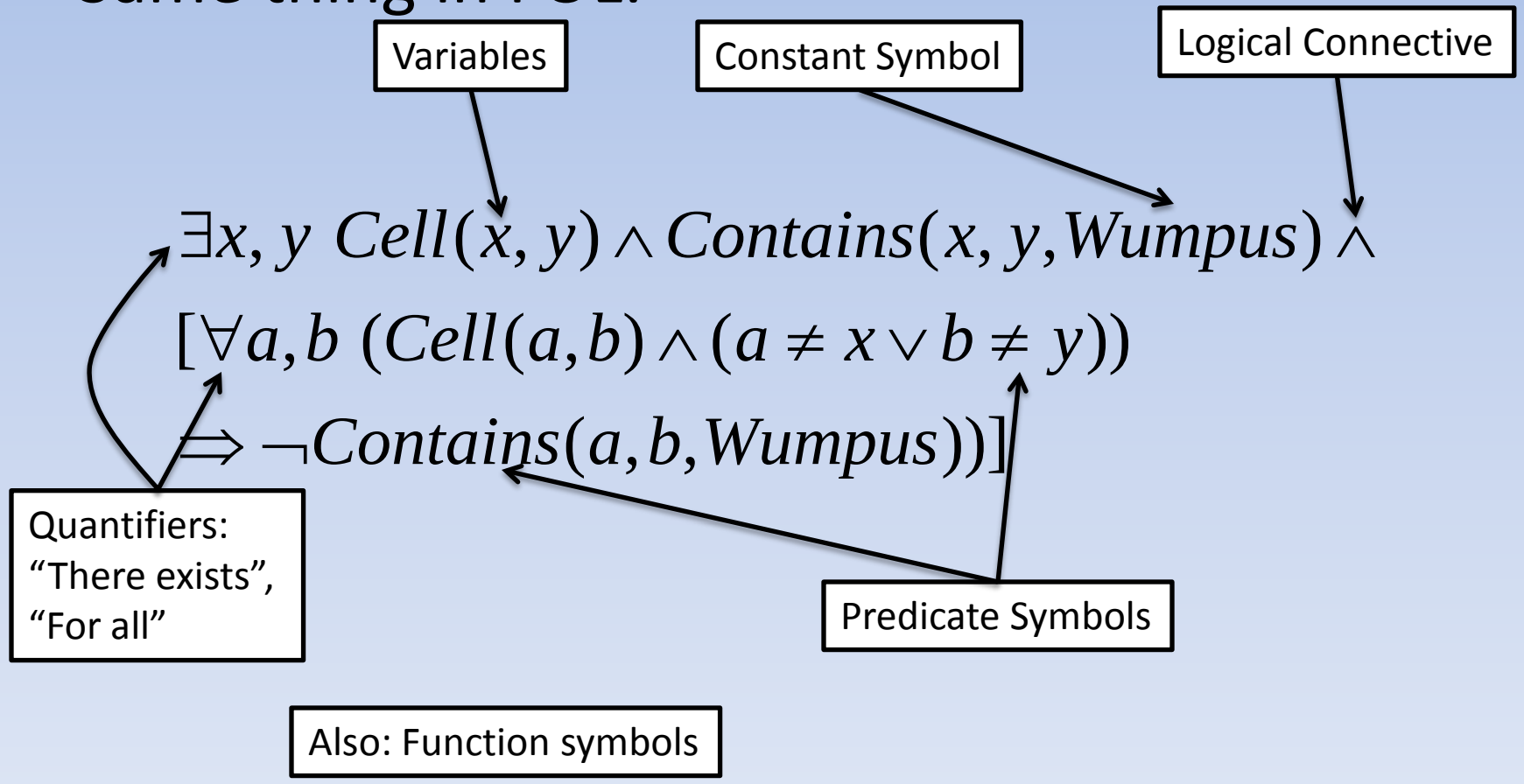
- HW3 due (solutions tomorrow noon)
- Quiz 3 Thursday

# Today

- First order logic (Chapter 8)
- FOL Inference (Chapter 9)

# First Order Logic

- Same thing in FOL:



# Key Elements of FOL

- FOL introduces:
  - Objects
  - *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 takes a set of objects and returns true/false, e.g. $isMotherOf(A,B)$?

# Syntax

- Basic elements:
  - Constant symbols (representing objects in the world)
  - Predicate symbols (representing relations between objects)
  - Function symbols (can be used to map objects to others)
- Also variables (placeholders for objects)

# Terms

- 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: *Wumpus*, *Pit*, *x*, *y*, *XWestOf(x,y)*

- A term is a generic representation of an object

# Atomic Formulae

- If $P$ is a predicate symbol of arity $n$, and $t_1,\ldots,t_n$ are terms, $P(t_1,\ldots,t_n)$ is an atomic formula
- Examples: $Contains(x, y, Wumpus)$, $Smelly(x, y)$, $Breezy(x, y)$

- Analogous to proposition symbols in propositional logic

# Complex Formulae

- All atomic formulae are complex formulae
- If $F_1$ and $F_2$ are formulae, so is $(\neg F_1)$, $(F_1 \wedge F_2)$, 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:

$$\forall x\ \forall y(Cell(x,y) \wedge Contains(x,y,Pit) \wedge Cell(XNorthOf(x, y),YNorthOf(x, y))) \Rightarrow Breezy(XNorthOf(x, y),YNorthOf(x, y))$$

# Quantifiers

- FOL has "quantifiers" $\forall$ and $\exists$
- $\forall$ is the "universal quantifier", read "For all"
  - $\forall x P(x)$ is a formula that says $P$ is true for every object $x$
  - Can be thought of as a conjunction over the universe of objects: $P(a) \wedge P(b) \wedge P(c) \ldots$
- $\exists$ is the "existential quantifier", read "There exists"
  - $\exists x P(x)$ 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: $P(a) \vee P(b) \vee P(c) \ldots$

# Quantifier nesting

- Quantifiers can be nested:

$\exists x \exists y Contains(x, y, Wumpus)$

(or $\exists x,y Contains(x, y, Wumpus)$)

- The order of nesting is important
  - $\forall x \exists y Friend(x,y)$ : "Everybody has a friend"
  - $\exists y \forall x Friend(x,y)$ : "There is someone who is everyone's friend"

# 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

# Variable scoping

- If two quantifiers use the same variable, the variable "belongs" to the innermost quantifier

$$\forall x P(x) \wedge \exists x \forall y Q(x, y) \equiv$$

$$\forall x P(x) \wedge \exists a \forall y Q(a, y)$$

- 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

# Terminology

- 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. Contains(x, y, Wumpus)
- A formula with only bound variables or constants is said to be "closed"
  - We will always work with closed formulae

# 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 constant symbols 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 $P(t_1,\ldots,t_n)$ is a list of tuples of objects for which the relation holds
- A function $f(t_1,\ldots,t_n)$ 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)

| X | Y | Z |
|---|---|---|
| 3 | 2 | Wumpus |
| 1 | 3 | Pit |
| (etc) | | |

- XWestOf(x, y)

| x | y | XWestOf(X,Y,Wumpus) |
|---|---|---|
| 1 | 1 | -1 |
| 2 | 2 | 1 |
| 3 | 1 | 2 |
| (etc) | | |

# 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 $P(t_1,\ldots,t_n)$ : 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
  - 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

$\forall x\ Feathers(x) \wedge Flies(x) \Rightarrow Bird(x)$

$Feathers(Crow)$

$Flies(Crow)$

$Feathers(Ostrich)$

$?\exists x Bird(x)$

# FOL Inference Roadmap

- If we can somehow turn our KB into a propositional KB, we can use techniques we already know—*propositionalization*
  - This has some drawbacks
- Then we'll see how to extend resolution to FOL

# Converting FOL to propositions

- Dealing with $\forall$:
  - If our KB has $\forall x P(x)$, we can *substitute* $x$ with all objects in our model's domain
  - replace it with a sequence of facts: $P(a)$, $P(b)$, $P(c)$, …
  - Notice this could be infinite if there are function symbols: $P(a)$, $P(f(a))$, $P(f(f(a)))$,…

# 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 $\theta=\{a/f(b,c), b/c\}$, $\sigma=\{b/Const\}$
  - Then $p(a,b)\theta=p(f(b,c),c)$, $p(a,b)\sigma=p(a,Const\}$

# Converting FOL to propositions

- Dealing with $\exists$:
  - If our KB has $\exists x P(x)$, we can *substitute* $x$ with *some* object from our model's domain
  - We don't specifically know the name of this object, so let's invent one
    - This is called a "Skolem constant"
  - So $\exists x P(x)$ becomes $P(NewSymbol)$, where $NewSymbol$ is a constant symbol not used by anything else in our KB

# Converting FOL to propositions

- At this point, we've eliminated all quantifiers and are left with a collection of *ground* formulae
- We can treat each atomic formula as a separate propositional symbol
  - This reduces the KB to a propositional one
  - Then to answer questions we can use propositional inference

# Example

$\forall x\ Feathers(x) \wedge Flies(x) \Rightarrow Bird(x)$

$Feathers(Crow)$

$Flies(Crow)$

$Feathers(Ostrich)$

---

$Feathers(Crow)$

$Flies(Crow)$

$Feathers(Ostrich)$

$Feathers(Crow) \wedge Flies(Crow) \Rightarrow Bird(Crow)$

$Feathers(Ostrich) \wedge Flies(Ostrich) \Rightarrow Bird(Ostrich)$

# Example

$$Feathers\_Crow$$

$$Flies\_Crow$$

$$Feathers\_Ostrich$$

$$Feathers\_Crow \wedge Flies\_Crow \Rightarrow Bird\_Crow$$

$$Feathers\_Ostrich \wedge Flies\_Ostrich \Rightarrow Bird\_Ostrich$$

# Issues

- What if there are function symbols?
  - There are theorems (Skolem, Herbrand, Godel 1929-1930) that says that if a FOL formula is entailed by a KB, it must be provable using just a finite subset of the propositionalized KB
  - We could do a sort of “iterative deepening” of the KB
- So if a formula is entailed, it can be proved eventually---completeness!
- But what if it is *not* entailed?

# Entailment is Semidecidable

- If a formula is *not* entailed by a KB, in general there is no way to tell by an algorithm
  - i.e., no algorithm exists that will always finitely determine non-entailment
  - Our “iterative deepening” procedure might in general run forever
- This creates problems for real logic systems
  - Prolog has “negation as failure”

# Issues Continued

- Converting to a propositionalized KB is very expensive
- Results in huge propositional KBs even for small first order KBs
  - Contains large numbers of formulae that might be completely irrelevant to query
  - Still sometimes usable due to fast propositional SAT solvers
- These issues can be solved by FOL (syntactic) inference algorithms

# Lifting

- 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 resolutions exist that trade off completeness and efficiency

# Resolution inference rule

$$\frac{l_1 \vee l_2 \vee m \ldots \vee l_k, \quad r_1 \vee r_2 \vee \neg m \ldots \vee r_k}{l_1 \vee l_2 \ldots \vee l_k \vee r_1 \vee r_2 \ldots \vee r_k}$$

$$\frac{l_1 \vee l_2 \vee m_i \ldots \vee l_k, \quad r_1 \vee r_2 \vee \neg n_j \ldots \vee r_k, m_i\theta = \neg n_j\theta}{l_1 \vee l_2 \ldots \vee l_k \vee r_1 \vee r_2 \ldots \vee r_k \{\theta\}}$$

Atomic Formulae

Unification: This substitution makes $m_i$ identical to $\neg n_j$