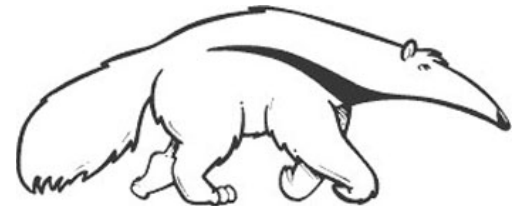


First-Order Logic A: Syntax

CS271P, Fall Quarter, 2018
Introduction to Artificial Intelligence
Prof. Richard Lathrop



Read Beforehand: R&N 8, 9.1-9.2, 9.5.1-9.5.5

Common Sense Reasoning

Example, adapted from Lenat

You are told: John drove to the grocery store and bought a pound of noodles, a pound of ground beef, and two pounds of tomatoes.

- Is John 3 years old?
- Is John a child?
- What will John do with the purchases?
- Did John have any money?
- Does John have less money after going to the store?
- Did John buy at least two tomatoes?
- Were the tomatoes made in the supermarket?
- Did John buy any meat?
- Is John a vegetarian?
- Will the tomatoes fit in John's car?

- Can Propositional Logic support these inferences?

Outline for First-Order Logic (FOL, also called FOPC)

- Propositional Logic is **Useful** --- but **Limited Expressive Power**
- First Order Predicate Calculus (FOPC), or First Order Logic (FOL).
 - FOPC has expanded expressive power, though still limited.
- New Ontology
 - The world consists of OBJECTS.
 - OBJECTS have PROPERTIES, RELATIONS, and FUNCTIONS.
- New Syntax
 - Constants, Predicates, Functions, Properties, Quantifiers.
- New Semantics
 - Meaning of new syntax.
- Unification and Inference in FOL
- Knowledge engineering in FOL

FOL Syntax:

You will be expected to know

- FOPC syntax
 - Syntax: Sentences, predicate symbols, function symbols, constant symbols, variables, quantifiers
- De Morgan's rules for quantifiers
 - connections between \forall and \exists
- Nested quantifiers
 - Difference between " $\forall x \exists y P(x, y)$ " and " $\exists x \forall y P(x, y)$ "
 - $\forall x \exists y \text{ Likes}(x, y)$ --- "Everybody likes somebody."
 - $\exists x \forall y \text{ Likes}(x, y)$ --- "Somebody likes everybody."
- Translate simple English sentences to FOPC and back
 - $\forall x \exists y \text{ Likes}(x, y) \Leftrightarrow$ "Everyone has someone that they like."
 - $\exists x \forall y \text{ Likes}(x, y) \Leftrightarrow$ "There is someone who likes every person."

Pros and cons of propositional logic

- ☺ Propositional logic is **declarative**
 - Knowledge and inference are separate
- ☺ Propositional logic allows **partial/disjunctive/negated information**
 - unlike most programming languages and databases
- ☺ Propositional logic is **compositional**:
 - meaning of $B_{1,1} \wedge P_{1,2}$ 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 **limited expressive power**
 - E.g., cannot say “Pits cause breezes in adjacent squares.”
 - except by writing one sentence for each square
 - Needs to refer to objects in the world,
 - Needs to express general rules

First-Order Logic (FOL), also called First-Order Predicate Calculus (FOPC)

- Propositional logic assumes the world contains **facts**.
- First-order logic (like natural language) assumes the world contains
 - **Objects**: people, houses, numbers, colors, baseball games, wars, ...
 - **Functions**: father of, best friend, one more than, plus, ...
 - Function arguments are objects; function returns an object
 - **Objects generally correspond to English NOUNS**
 - **Predicates/Relations/Properties**: red, round, prime, brother of, bigger than, part of, comes between, ...
 - Predicate arguments are objects; predicate returns a truth value
 - **Predicates generally correspond to English VERBS**
 - **First argument is generally the subject, the second the object**
 - Hit(Bill, Ball) usually means “Bill hit the ball.”
 - Likes(Bill, IceCream) usually means “Bill likes IceCream.”
 - Verb(Noun1, Noun2) usually means “Noun1 verb noun2.”

Aside: First-Order Logic (FOL) vs. Second-Order Logic

- First Order Logic (FOL) allows variables and general rules
 - “First order” because quantified variables represent objects.
 - “Predicate Calculus” because it quantifies over predicates on objects.
 - E.g., “Integral Calculus” quantifies over functions on numbers.
- Aside: Second Order logic
 - “Second order” because quantified variables can also represent predicates and functions.
 - E.g., can define “Transitive Relation,” which is beyond FOL.
- Aside: In FOL we can state that a relationship is transitive
 - E.g., BrotherOf is a transitive relationship
 - $\forall x, y, z \text{ BrotherOf}(x,y) \wedge \text{BrotherOf}(y,z) \Rightarrow \text{BrotherOf}(x,z)$
- Aside: In Second Order logic we can define “Transitive”
 - $\forall P, x, y, z \text{ Transitive}(P) \Leftrightarrow (P(x,y) \wedge P(y,z) \Rightarrow P(x,z))$
 - Then we can state directly, $\text{Transitive}(\text{BrotherOf})$

Syntax of FOL: Basic elements

- Constants KingJohn, 2, UCI,...
- Predicates Brother, >,...
- Functions Sqrt, LeftLegOf,...
- Variables x, y, a, b,...
- Quantifiers \forall, \exists
- Connectives $\neg, \wedge, \vee, \Rightarrow, \Leftrightarrow$ (standard)
- Equality = (but causes difficulties....)

Syntax of FOL: Basic syntax elements are symbols

- **Constant Symbols** (correspond to English nouns)
 - Stand for objects in the world.
 - E.g., KingJohn, 2, UCI, ...
- **Predicate Symbols** (correspond to English verbs)
 - Stand for relations (**maps a tuple of objects to a truth-value**)
 - E.g., Brother(Richard, John), greater_than(3,2), ...
 - $P(x, y)$ is usually read as “ x is P of y .”
 - E.g., Mother(Ann, Sue) is usually “Ann is Mother of Sue.”
- **Function Symbols** (correspond to English nouns)
 - Stand for functions (**maps a tuple of objects to an object**)
 - E.g., Sqrt(3), LeftLegOf(John), ...
- **Model** (world) = set of domain objects, relations, functions
- **Interpretation** maps symbols onto the model (world)
 - Very many interpretations are possible for each KB and world!
 - The KB is to rule out those inconsistent with our knowledge.

Syntax of FOL: Terms

- **Term** = logical expression that **refers to an object**
- **There are two kinds of terms:**
 - **Constant Symbols** stand for (or name) objects:
 - E.g., KingJohn, 2, UCI, Wumpus, ...
 - **Function Symbols** map tuples of objects to an object:
 - E.g., LeftLeg(KingJohn), Mother(Mary), Sqrt(x)
 - This is nothing but a complicated kind of name
 - No “subroutine” call, no “return value”

Syntax of FOL: Atomic Sentences

- **Atomic Sentences** state facts (logical truth values).
 - An **atomic sentence** is a Predicate symbol, optionally followed by a parenthesized list of any argument terms
 - E.g., *Married(Father(Richard), Mother(John))*
 - An **atomic sentence** asserts that some relationship (some predicate) holds among the objects that are its arguments.
- An **Atomic Sentence is true** in a given model if the relation referred to by the predicate symbol holds among the objects (terms) referred to by the arguments.

Syntax of FOL: Atomic Sentences

- Atomic sentences in logic state facts that are true or false.
- Properties and m -ary relations do just that:
 - LargerThan(2, 3) is false.
 - BrotherOf(Mary, Pete) is false.
 - Married(Father(Richard), Mother(John)) could be true or false.Properties and m -ary relations are Predicates that are true or false.
- Note: Functions refer to objects, do not state facts, and form no sentence:
 - Brother(Pete) refers to John (his brother) and is neither true nor false.
 - Plus(2, 3) refers to the number 5 and is neither true nor false.
- BrotherOf(Pete, Brother(Pete)) is True.

↑
Binary relation
is a truth value.

↑
Function refers to John, an object in the
world, i.e., John is Pete's brother.
(Works well iff John is Pete's only brother.)

Syntax of FOL:

Connectives & Complex Sentences

- **Complex Sentences** are formed in the same way, using the same logical connectives, as in propositional logic
- **The Logical Connectives:**
 - \Leftrightarrow biconditional
 - \Rightarrow implication
 - \wedge and
 - \vee or
 - \neg negation
- **Semantics** for these logical connectives are the same as we already know from propositional logic.

Examples

- $\text{Brother}(\text{Richard}, \text{John}) \wedge \text{Brother}(\text{John}, \text{Richard})$
- $\text{King}(\text{Richard}) \vee \text{King}(\text{John})$
- $\text{King}(\text{John}) \Rightarrow \neg \text{King}(\text{Richard})$
- $\text{LessThan}(\text{Plus}(1,2), 4) \wedge \text{GreaterThan}(1,2)$

Syntax of FOL: Variables

- **Variables** range over objects in the world.
- A **variable** is like a **term** because it represents an object.
- A **variable** may be used wherever a **term** may be used.
 - **Variables** may be arguments to functions and predicates.
- (A **term with NO variables** is called a **ground term**.)
- (A **variable not bound by a quantifier** is called **free**.)
 - All variables we will use are bound by a quantifier.

Syntax of FOL: Logical Quantifiers

- There are two **Logical Quantifiers**:
 - **Universal:** $\forall x P(x)$ means “For all x , $P(x)$.”
 - The “upside-down A” reminds you of “ALL.”
 - Some texts put a comma after the variable: $\forall x, P(x)$
 - **Existential:** $\exists x P(x)$ means “There exists x such that, $P(x)$.”
 - The “backward E” reminds you of “EXISTS.”
 - Some texts put a comma after the variable: $\exists x, P(x)$
- You can **ALWAYS** convert one quantifier to the other.
 - $\forall x P(x) \equiv \neg \exists x \neg P(x)$
 - $\exists x P(x) \equiv \neg \forall x \neg P(x)$
 - **RULES:** $\forall \equiv \neg \exists \neg$ and $\exists \equiv \neg \forall \neg$
- **RULES:** To move negation “in” across a quantifier,
Change the quantifier to “the other quantifier”
and negate the predicate on “the other side.”
 - $\neg \forall x P(x) \equiv \neg \neg \exists x \neg P(x) \equiv \exists x \neg P(x)$
 - $\neg \exists x P(x) \equiv \neg \neg \forall x \neg P(x) \equiv \forall x \neg P(x)$

Universal Quantification \forall

- $\forall x$ means “for all x it is true that...”
- Allows us to make statements about all objects that have certain properties
- Can now state general rules:

$\forall x \text{ King}(x) \Rightarrow \text{Person}(x)$ “All kings are persons.”

$\forall x \text{ Person}(x) \Rightarrow \text{HasHead}(x)$ “Every person has a head.”

$\forall i \text{ Integer}(i) \Rightarrow \text{Integer}(\text{plus}(i,1))$ “If i is an integer then i+1 is an integer.”

- **Note: $\forall x \text{ King}(x) \wedge \text{Person}(x)$ is not correct!**

This would imply that all objects x are Kings and are People (!)

$\forall x \text{ King}(x) \Rightarrow \text{Person}(x)$ is the correct way to say this

- **Note that \Rightarrow is the natural connective to use with \forall .**

Universal Quantification \forall

- Universal quantification is conceptually equivalent to:
 - Conjunction of all sentences obtained by substitution of an object for the quantified variable.
 - Not a sentence in the logic --- all logic sentences must be finite.
- Example: All Cats are Mammals.
 - $\forall x \text{ Cat}(x) \Rightarrow \text{Mammal}(x)$
- Conjunction of all sentences obtained by substitution of an object for the quantified variable:
 - $\text{Cat}(\text{Spot}) \Rightarrow \text{Mammal}(\text{Spot}) \wedge$
 - $\text{Cat}(\text{Rebecca}) \Rightarrow \text{Mammal}(\text{Rebecca}) \wedge$
 - $\text{Cat}(\text{LAX}) \Rightarrow \text{Mammal}(\text{LAX}) \wedge$
 - $\text{Cat}(\text{Shayama}) \Rightarrow \text{Mammal}(\text{Shayama}) \wedge$
 - $\text{Cat}(\text{France}) \Rightarrow \text{Mammal}(\text{France}) \wedge$
 - $\text{Cat}(\text{Felix}) \Rightarrow \text{Mammal}(\text{Felix}) \wedge$
 - ...

Existential Quantification \exists

- $\exists x$ means “there exists an x such that....”
 - There is in the world at least one such object x
- Allows us to make statements about some object without naming it, or even knowing what that object is:
 - $\exists x \text{ King}(x)$ “Some object is a king.”
 - $\exists x \text{ Lives_in}(\text{John}, \text{Castle}(x))$ “John lives in somebody’s castle.”
 - $\exists i \text{ Integer}(i) \wedge \text{Greater}(i,0)$ “Some integer is greater than zero.”
- **Note: $\exists i \text{ Integer}(i) \Rightarrow \text{Greater}(i,0)$ is not correct!**

It is vacuously true if anything in the world were not an integer (!)

$\exists i \text{ Integer}(i) \wedge \text{Greater}(i,0)$ is the correct way to say this
- **Note that \wedge is the natural connective to use with \exists .**

Existential Quantification \exists

- Existential quantification is conceptually equivalent to:
 - Disjunction of all sentences obtained by substitution of an object for the quantified variable.
 - Not a sentence in the logic --- all logic sentences must be finite.
- Spot has a sister who is a cat.
 - $\exists x \text{ Sister}(x, \text{Spot}) \wedge \text{Cat}(x)$
- Disjunction of all sentences obtained by substitution of an object for the quantified variable:
 - Sister(Spot, Spot) \wedge Cat(Spot) \vee*
 - Sister(Rebecca, Spot) \wedge Cat(Rebecca) \vee*
 - Sister(LAX, Spot) \wedge Cat(LAX) \vee*
 - Sister(Shayama, Spot) \wedge Cat(Shayama) \vee*
 - Sister(France, Spot) \wedge Cat(France) \vee*
 - Sister(Felix, Spot) \wedge Cat(Felix) \vee*
 - ...*

Combining Quantifiers --- Order (Scope)

The order of “unlike” quantifiers is important.

Like nested variable scopes in a programming language.

Like nested ANDs and ORs in a logical sentence.

$\forall x \exists y \text{ Loves}(x,y)$

- For everyone (“all x”) there is someone (“exists y”) whom they love.
- There might be a different y for each x (y is inside the scope of x)

$\exists y \forall x \text{ Loves}(x,y)$

- There is someone (“exists y”) whom everyone loves (“all x”).
- Every x loves the same y (x is inside the scope of y)

Clearer with parentheses: $\exists y (\forall x \text{ Loves}(x,y))$

The order of “like” quantifiers does not matter.

Like nested ANDs and ANDs in a logical sentence

$$\forall x \forall y P(x, y) \equiv \forall y \forall x P(x, y)$$

$$\exists x \exists y P(x, y) \equiv \exists y \exists x P(x, y)$$

Connections between Quantifiers

- Asserting that all x have property P is the same as asserting that does not exist any x that does not have the property P

$$\forall x \text{ Likes}(x, \text{CS-171 class}) \Leftrightarrow \neg \exists x \neg \text{Likes}(x, \text{CS-171 class})$$

- Asserting that there exists an x with property P is the same as asserting that not all x do not have the property P

$$\exists x \text{ Likes}(x, \text{IceCream}) \Leftrightarrow \neg \forall x \neg \text{Likes}(x, \text{IceCream})$$

In effect:

- \forall is a conjunction over the universe of objects
- \exists is a disjunction over the universe of objects

Thus, DeMorgan's rules can be applied

De Morgan's Law for Quantifiers

De Morgan's Rule

$$P \wedge Q \equiv \neg(\neg P \vee \neg Q)$$

$$P \vee Q \equiv \neg(\neg P \wedge \neg Q)$$

$$\neg(P \wedge Q) \equiv \neg P \vee \neg Q$$

$$\neg(P \vee Q) \equiv \neg P \wedge \neg Q$$

Generalized De Morgan's Rule

$$\forall x P \equiv \neg \exists x (\neg P)$$

$$\exists x P \equiv \neg \forall x (\neg P)$$

$$\neg \forall x P \equiv \exists x (\neg P)$$

$$\neg \exists x P \equiv \forall x (\neg P)$$

AND/OR Rule is simple: if you bring a negation inside a disjunction or a conjunction, always switch between them (\neg OR \rightarrow AND \neg ; \neg AND \rightarrow OR \neg).

QUANTIFIER Rule is similar: if you bring a negation inside a universal or existential, always switch between them ($\neg \exists \rightarrow \forall \neg$; $\neg \forall \rightarrow \exists \neg$).

De Morgan's Law for Quantifiers

De Morgan's Rule

$$P \wedge Q \equiv \neg (\neg P \vee \neg Q)$$

$$P \vee Q \equiv \neg (\neg P \wedge \neg Q)$$

$$\neg (P \wedge Q) \equiv (\neg P \vee \neg Q)$$

$$\neg (P \vee Q) \equiv (\neg P \wedge \neg Q)$$

Generalized De Morgan's Rule

$$\forall x P(x) \equiv \neg \exists x \neg P(x)$$

$$\exists x P(x) \equiv \neg \forall x \neg P(x)$$

$$\neg \forall x P(x) \equiv \exists x \neg P(x)$$

$$\neg \exists x P(x) \equiv \forall x \neg P(x)$$

AND/OR Rule is simple: if you bring a negation inside a disjunction or a conjunction, always switch between them (\neg OR \rightarrow AND \neg ; \neg AND \rightarrow OR \neg).

QUANTIFIER Rule is similar: if you bring a negation inside a universal or existential, always switch between them ($\neg \exists \rightarrow \forall \neg$; $\neg \forall \rightarrow \exists \neg$).

Aside: More syntactic sugar --- uniqueness

- $\exists! x$ is “syntactic sugar” for “There exists a unique x ”
 - “There exists one and only one x ”
 - “There exists exactly one x ”
 - Sometimes $\exists!$ is written as \exists^1
- For example, $\exists! x \text{ PresidentOfTheUSA}(x)$
 - “There is exactly one PresidentOfTheUSA.”
- This is just syntactic sugar:
 - $\exists! x P(x)$ is the same as $\exists x P(x) \wedge (\forall y P(y) \Rightarrow (x = y))$
 - “Syntactic sugar” = a convenient syntax abbreviation/extension

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., definition of *Sibling* in terms of *Parent*, using = is:

$$\begin{aligned} \forall x,y \text{ Sibling}(x,y) \Leftrightarrow \\ [\neg(x = y) \wedge \\ \exists m,f \neg(m = f) \wedge \text{Parent}(m,x) \wedge \text{Parent}(f,x) \\ \wedge \text{Parent}(m,y) \wedge \text{Parent}(f,y)] \end{aligned}$$

- **Equality can make reasoning much more difficult!**
 - (See R&N, section 9.5.5, page 353)
 - You may not know when two objects are equal.
 - E.g., Ancients did not know (MorningStar = EveningStar = Venus)
 - You may have to prove $x = y$ before proceeding
 - E.g., a resolution prover may not know $2+1$ is the same as $1+2$ or $4-1$

Syntactic Ambiguity

- FOPC provides many ways to represent the same thing.
- E.g., “Ball-5 is red.”
 - HasColor(Ball-5, Red)
 - Ball-5 and Red are objects related by HasColor.
 - Red(Ball-5)
 - Red is a unary predicate applied to the Ball-5 object.
 - HasProperty(Ball-5, Color, Red)
 - Ball-5, Color, and Red are objects related by HasProperty.
 - ColorOf(Ball-5) = Red
 - Ball-5 and Red are objects, and ColorOf() is a function.
 - HasColor(Ball-5(), Red())
 - Ball-5() and Red() are functions of zero arguments that both return an object, which objects are related by HasColor.
 - ...
- This can GREATLY confuse a pattern-matching reasoner.
 - Especially if multiple people collaborate to build the KB, and they all have different representational conventions.

Syntactic Ambiguity --- Partial Solution

- FOL can be TOO expressive, can offer TOO MANY choices
- Likely confusion, especially for **teams** of Knowledge Engineers
- Different team members can make different representation choices
 - E.g., represent “Ball43 is Red.” as:
 - a predicate (= verb)? E.g., “Red(Ball43)” ?
 - an object (= noun)? E.g., “Red = Color(Ball43)” ?
 - a property (= adjective)? E.g., “HasProperty(Ball43, Red)” ?
- PARTIAL SOLUTION:
 - An upon-agreed **ontology** that settles these questions
 - Ontology = what exists in the world & how it is represented
 - The Knowledge Engineering teams agrees upon an ontology BEFORE they begin encoding knowledge

Fun with sentences

Brothers are siblings

Fun with sentences

Brothers are siblings

$\forall x, y \text{ Brother}(x, y) \Rightarrow \text{Sibling}(x, y).$

“Sibling” is symmetric

Fun with sentences

Brothers are siblings

$$\forall x, y \text{ Brother}(x, y) \Rightarrow \text{Sibling}(x, y).$$

“Sibling” is symmetric

$$\forall x, y \text{ Sibling}(x, y) \Leftrightarrow \text{Sibling}(y, x).$$

One's mother is one's female parent

Fun with sentences

Brothers are siblings

$$\forall x, y \text{ Brother}(x, y) \Rightarrow \text{Sibling}(x, y).$$

“Sibling” is symmetric

$$\forall x, y \text{ Sibling}(x, y) \Leftrightarrow \text{Sibling}(y, x).$$

One's mother is one's female parent

$$\forall x, y \text{ Mother}(x, y) \Leftrightarrow (\text{Female}(x) \wedge \text{Parent}(x, y)).$$

A first cousin is a child of a parent's sibling

Fun with sentences

Brothers are siblings

$$\forall x, y \text{ Brother}(x, y) \Rightarrow \text{Sibling}(x, y).$$

“Sibling” is symmetric

$$\forall x, y \text{ Sibling}(x, y) \Leftrightarrow \text{Sibling}(y, x).$$

One's mother is one's female parent

$$\forall x, y \text{ Mother}(x, y) \Leftrightarrow (\text{Female}(x) \wedge \text{Parent}(x, y)).$$

A first cousin is a child of a parent's sibling

$$\forall x, y \text{ FirstCousin}(x, y) \Leftrightarrow \exists p, ps \text{ Parent}(p, x) \wedge \text{Sibling}(ps, p) \wedge \text{Parent}(ps, y)$$

More fun with sentences

- **“All persons are mortal.”**
- [Use: Person(x), Mortal (x)]

More fun with sentences

- **“All persons are mortal.”**
[Use: Person(x), Mortal (x)]
- $\forall x \text{ Person}(x) \Rightarrow \text{Mortal}(x)$
- **Equivalent Forms:**
- $\forall x \neg \text{Person}(x) \vee \text{Mortal}(x)$
- **Common Mistakes:**
- $\forall x \text{ Person}(x) \wedge \text{Mortal}(x)$

More fun with sentences

- **“Fifi has a sister who is a cat.”**
- [Use: Sister(Fifi, x), Cat(x)]
-

More fun with sentences

- **“Fifi has a sister who is a cat.”**
- [Use: $Sister(Fifi, x)$, $Cat(x)$]
- $\exists x Sister(Fifi, x) \wedge Cat(x)$
- **Common Mistakes:**
- $\exists x Sister(Fifi, x) \Rightarrow Cat(x)$

More fun with sentences

- **“For every food, there is a person who eats that food.”**

[Use: Food(x), Person(y), Eats(y, x)]

More fun with sentences

- “For every food, there is a person who eats that food.”
[Use: Food(x), Person(y), Eats(y, x)]
- $\forall x \exists y \text{ Food}(x) \Rightarrow [\text{Person}(y) \wedge \text{Eats}(y, x)]$
- **Equivalent Forms:**
- $\forall x \text{ Food}(x) \Rightarrow \exists y [\text{Person}(y) \wedge \text{Eats}(y, x)]$
- $\forall x \exists y \neg \text{Food}(x) \vee [\text{Person}(y) \wedge \text{Eats}(y, x)]$
- $\forall x \exists y [\neg \text{Food}(x) \vee \text{Person}(y)] \wedge [\neg \text{Food}(x) \vee \text{Eats}(y, x)]$
- $\forall x \exists y [\text{Food}(x) \Rightarrow \text{Person}(y)] \wedge [\text{Food}(x) \Rightarrow \text{Eats}(y, x)]$
- **Common Mistakes:**
- $\forall x \exists y [\text{Food}(x) \wedge \text{Person}(y)] \Rightarrow \text{Eats}(y, x)$
- $\forall x \exists y \text{ Food}(x) \wedge \text{Person}(y) \wedge \text{Eats}(y, x)$

More fun with sentences

- **“Every person eats every food.”**

[Use: Person (x), Food (y), Eats(x, y)]

More fun with sentences

- **“Every person eats every food.”**
[Use: Person (x), Food (y), Eats(x, y)]
- $\forall x \forall y [\text{Person}(x) \wedge \text{Food}(y)] \Rightarrow \text{Eats}(x, y)$
- **Equivalent Forms:**
- $\forall x \forall y \neg \text{Person}(x) \vee \neg \text{Food}(y) \vee \text{Eats}(x, y)$
- $\forall x \forall y \text{Person}(x) \Rightarrow [\text{Food}(y) \Rightarrow \text{Eats}(x, y)]$
- $\forall x \forall y \text{Person}(x) \Rightarrow [\neg \text{Food}(y) \vee \text{Eats}(x, y)]$
- $\forall x \forall y \neg \text{Person}(x) \vee [\text{Food}(y) \Rightarrow \text{Eats}(x, y)]$
- **Common Mistakes:**
- $\forall x \forall y \text{Person}(x) \Rightarrow [\text{Food}(y) \wedge \text{Eats}(x, y)]$
- $\forall x \forall y \text{Person}(x) \wedge \text{Food}(y) \wedge \text{Eats}(x, y)$

More fun with sentences

- **“All greedy kings are evil.”**

[Use: King(x), Greedy(x), Evil(x)]

More fun with sentences

- **“All greedy kings are evil.”**
[Use: King(x), Greedy(x), Evil(x)]
- $\forall x [\text{Greedy}(x) \wedge \text{King}(x)] \Rightarrow \text{Evil}(x)$
- **Equivalent Forms:**
- $\forall x \neg \text{Greedy}(x) \vee \neg \text{King}(x) \vee \text{Evil}(x)$
- $\forall x \text{Greedy}(x) \Rightarrow [\text{King}(x) \Rightarrow \text{Evil}(x)]$
- **Common Mistakes:**
- $\forall x \text{Greedy}(x) \wedge \text{King}(x) \wedge \text{Evil}(x)$

More fun with sentences

- **“Everyone has a favorite food.”**
[Use: Person(x), Food(y), Favorite(y, x)]

More fun with sentences

- “Everyone has a favorite food.”

[Use: Person(x), Food(y), Favorite(y, x)]

- **Equivalent Forms:**

- $\forall x \exists y \text{ Person}(x) \Rightarrow [\text{Food}(y) \wedge \text{Favorite}(y, x)]$

- $\forall x \text{ Person}(x) \Rightarrow \exists y [\text{Food}(y) \wedge \text{Favorite}(y, x)]$

- $\forall x \exists y \neg \text{Person}(x) \vee [\text{Food}(y) \wedge \text{Favorite}(y, x)]$

- $\forall x \exists y [\neg \text{Person}(x) \vee \text{Food}(y)] \wedge [\neg \text{Person}(x) \vee \text{Favorite}(y, x)]$

- $\forall x \exists y [\text{Person}(x) \Rightarrow \text{Food}(y)] \wedge [\text{Person}(x) \Rightarrow \text{Favorite}(y, x)]$

- **Common Mistakes:**

- $\forall x \exists y [\text{Person}(x) \wedge \text{Food}(y)] \Rightarrow \text{Favorite}(y, x)$

- $\forall x \exists y \text{ Person}(x) \wedge \text{Food}(y) \wedge \text{Favorite}(y, x)$

More fun with sentences

- **“There is someone at UCI who is smart.”**
[Use: Person(x), At(x, UCI), Smart(x)]

More fun with sentences

- **“There is someone at UCI who is smart.”**

[Use: $\text{Person}(x)$, $\text{At}(x, \text{UCI})$, $\text{Smart}(x)$]

- $\exists x \text{ Person}(x) \wedge \text{At}(x, \text{UCI}) \wedge \text{Smart}(x)$

- **Common Mistakes:**

- $\exists x [\text{Person}(x) \wedge \text{At}(x, \text{UCI})] \Rightarrow \text{Smart}(x)$

More fun with sentences

- **“Everyone at UCI is smart.”**

[Use: Person(x), At(x, UCI), Smart(x)]

More fun with sentences

- **“Everyone at UCI is smart.”**
[Use: Person(x), At(x, UCI), Smart(x)]
- $\forall x [\text{Person}(x) \wedge \text{At}(x, \text{UCI})] \Rightarrow \text{Smart}(x)$
- **Equivalent Forms:**
- $\forall x \neg [\text{Person}(x) \wedge \text{At}(x, \text{UCI})] \vee \text{Smart}(x)$
- $\forall x \neg \text{Person}(x) \vee \neg \text{At}(x, \text{UCI}) \vee \text{Smart}(x)$
- **Common Mistakes:**
- $\forall x \text{Person}(x) \wedge \text{At}(x, \text{UCI}) \wedge \text{Smart}(x)$
- $\forall x \text{Person}(x) \Rightarrow [\text{At}(x, \text{UCI}) \wedge \text{Smart}(x)]$
-

More fun with sentences

- **“Every person eats some food.”**

[Use: Person (x), Food (y), Eats(x, y)]

More fun with sentences

- “Every person eats some food.”

[Use: Person (x), Food (y), Eats(x, y)]

- $\forall x \exists y \text{ Person}(x) \Rightarrow [\text{Food}(y) \wedge \text{Eats}(x, y)]$

-

- **Equivalent Forms:**

- $\forall x \text{ Person}(x) \Rightarrow \exists y [\text{Food}(y) \wedge \text{Eats}(x, y)]$

- $\forall x \exists y \neg \text{Person}(x) \vee [\text{Food}(y) \wedge \text{Eats}(x, y)]$

- $\forall x \exists y [\neg \text{Person}(x) \vee \text{Food}(y)] \wedge [\neg \text{Person}(x) \vee \text{Eats}(x, y)]$

- **Common Mistakes:**

- $\forall x \exists y [\text{Person}(x) \wedge \text{Food}(y)] \Rightarrow \text{Eats}(x, y)$

- $\forall x \exists y \text{ Person}(x) \wedge \text{Food}(y) \wedge \text{Eats}(x, y)$

-

More fun with sentences

- **“Some person eats some food.”**
[Use: Person (x), Food (y), Eats(x, y)]

More fun with sentences

- **“Some person eats some food.”**
[Use: Person (x), Food (y), Eats(x, y)]
- $\exists x \exists y \text{ Person}(x) \wedge \text{Food}(y) \wedge \text{Eats}(x, y)$
- **Common Mistakes:**
- $\exists x \exists y [\text{Person}(x) \wedge \text{Food}(y)] \Rightarrow \text{Eats}(x, y)$

Summary

- First-order logic:
 - Much more expressive than propositional logic
 - Allows objects and relations as semantic primitives
 - Universal and existential quantifiers
- Syntax: constants, functions, predicates, equality, quantifiers
- Nested quantifiers
 - Order of unlike quantifiers matters (the outer scopes the inner)
 - Like nested ANDs and ORs
 - Order of like quantifiers does not matter
 - like nested ANDs and ANDs
- Translate simple English sentences to FOPC and back