# Artificial Intelligence

First Order Logic

# Outline

- Why FOL (instead of propositional logic)?
    - Propositional logic is declarative, compositional, and context-independent, but limited in expressiveness.
- Syntax and semantics of FOL
- Using FOL
- Knowledge engineering in FOL

# First-order Logic

- Propositional logic assumes the world contains facts, while first-order logic assumes the world contains:
  - Objects: people, houses, numbers, colors, baseball games, wars, …
  - Relations: red, round, prime_number, siblings, bigger_than, part_of, comes_between, … (each returns true/false)
  - Functions: father of, best friend, one more than, plus, … (each returns an object)

# Syntax of FOL
# Basic Elements

- Constants: Karen, 42, Table_in_room, VIU, …
- Predicates: isEmpty, Sibling, >, …
- Functions: Sqrt, Third_Grade_Teacher_Of, …
- Variables: x, y, a, b, …
- Connectives: $\neg$, $\wedge$, $\vee$, $\Rightarrow$, $\Leftrightarrow$
- Equality: =
- Quantifiers: $\forall$, $\exists$

# Atomic sentences

- atomic sentence ::= predicate ($term_1$, …, $term_n$)
  or $term_1$ = $term_2$

- term ::= function ($term_1$, …, $term_n$)
  or constant
  or variable

- Examples of atomic sentences:
  isEmpty(Children_of(Karen))
  Sibling(Karen, Third_Grade_Teacher_Of(John))
  NumOfLegs(Table_in_room) = 3
  Sqrt(10) = 3

# Complex sentences

- Complex sentences are made from atomic sentences using connectives
  - $\neg S$
  - $S1 \wedge S2$
  - $S1 \vee S2$
  - $S1 \Rightarrow S2$
  - $S1 \Leftrightarrow S2$,
- Examples:
  - $> (1, 2) \vee < (1, 2)$
  - $> (1, 2) \wedge \neg > (1, 2)$

# Truth in first-order logic

- Sentences are true with respect to a model and an interpretation
- Model contains objects (domain elements) and relations among them
- Interpretation specifies referents for
  constant symbols → objects
  predicate symbols → relations
  function symbols → functional relations
- An atomic sentence 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

# Universal Quantification

- ∀<variables> <sentence>

- Example:
  Every course offered at VIU has assignments
  ∀x IsCourse(x, VIU) ⇒ HasAssignments(x)

- ∀x P is true in a model m iff P is true with x being each possible object in the model

- Roughly speaking, equivalent to the conjunction of instantiations of P
  IsCourse(CSCI479, VIU) ⇒ HasAssignments(CSCI479)
  ∧ IsCourse(PSYC200, VIU) ⇒ HasAssignments(PSYC200)
  ∧ IsCourse(ASTR112, VIU) ⇒ HasAssignments(ASTR112)
  ∧ …

- Typically, ⇒ is the main connective with ∀

- Common mistake: using ∧ as the main connective with ∀

# Existential Quantification

- ∃ <variables> <sentence>

- Example:
  George has a son:
  ∃x Male(x) ∧ isParent(George, x)

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

- Roughly speaking, equivalent to the disjunction of instantiations of P
  Male(Mary) ∧ isParent(George, Mary)
  ∨ Male(James) ∧ isParent(George, James)
  ∨ Male(CSCI479) ∧ isParent(George, CSCI479)
  ∨ …

- Typically, ∧ is the main connective with ∃

- Common mistake: using ⇒ as the main connective with ∃

# Properties of Quantifiers

- $\forall x\ \forall y$ is the same as $\forall y\ \forall x$
- $\exists x\ \exists y$ is the same as $\exists y\ \exists x$
- $\exists x\ \forall y$ is not the same as $\forall y\ \exists x$
- $\exists x\ \forall y$ Loves(x, y)
  — "There exists a person (x) who loves everyone/thing in the world"
- $\forall y\ \exists x$ Loves(x, y)
  — "Everyone/thing in the world is loved by at least one person (x)"
- Quantifier duality: each can be expressed using the other
  $\forall x$ Likes(x,IceCream) is equivalent to $\neg(\exists x\ \neg$Likes(x,IceCream))
  $\exists x$ Likes(x,Broccoli) is equivalent to $\neg(\forall x\ \neg$Likes(x,Broccoli))

# Equality

- term1 = term2 is true under a given interpretation if and only if term1 and term2 refer to the same object

- Example — definition of Sibling in terms of Parent:

$$
\begin{aligned}
&\forall x\ \forall y\ Sibling(x, y) \\
&\Leftrightarrow (\neg(x = y) \wedge \\
&\quad (\exists m\ \exists f\ (\neg\ (m = f) \wedge Parent(m, x) \wedge Parent(f, x) \\
&\qquad \wedge Parent(m, y) \wedge\ Parent(f, y) \\
&\qquad ) \\
&\quad ) \\
&)
\end{aligned}
$$

# Knowledge Engineering in FOL

1. Identify the task
2. Assemble the relevant knowledge
3. Decide on a vocabulary of predicates, functions, and constants
4. Encode general knowledge about the domain
5. Encode a description of the specific problem instance
6. Pose queries to the inference procedure and get answers
7. Debug the knowledge base

# Summary

- First-order logic:
  - objects and relations are semantic primitives
  - syntax: constants, functions, predicates, equality, quantifiers
- Increased expressive power — sufficient for most scenarios