First-Order Logic: Review

First-order logic

- First-order logic (FOL) models the world in terms of
  - **Objects,** which are things with individual identities
  - **Properties** of objects that distinguish them from others
  - **Relations** that hold among sets of objects
  - **Functions,** which are a subset of relations where there is only one "value" for any given "input"
- Examples:
  - Objects: Students, lectures, companies, cars ...
  - Relations: Brother-of, bigger-than, outside, part-of, has-color, occurs-after, owns, visits, precedes, ...
  - Properties: blue, oval, even, large, ...
  - Functions: father-of, best-friend, second-half, more-than ...

User provides

- **Constant symbols** representing individuals in the world
  - Mary, 3, green
- **Function symbols,** map individuals to individuals
  - father_of(Mary) = John
  - color_of(Sky) = Blue
- **Predicate symbols,** map individuals to truth values
  - greater(5,3)
  - green(Grass)
  - color(Grass, Green)

FOL Provides

- **Variable symbols**
  - E.g., x, y, foo
- **Connectives**
  - Same as in propositional logic: not (¬), and (∧), or (∨), implies (→), iff (↔)
- **Quantifiers**
  - Universal **∀x** or **(Ax)**
  - Existential **∃x** or **(Ex)**

Sentences: built from terms and atoms

- A **term** (denoting a real-world individual) is a constant symbol, variable symbol, or n-place function of n terms.
  - x and $f(x_1, ..., x_n)$ are terms, where each x_i is a term
  - A term with no variables is a **ground term** (i.e., john, father_of(father_of(john))
- An **atomic sentence** (which has value true or false) is an n-place predicate of n terms (e.g., green(Kermit))
- A **complex sentence** is formed from atomic sentences connected by the logical connectives:
  - $\neg P$, $P \vee Q$, $P \wedge Q$, $P \rightarrow Q$, $P \leftrightarrow Q$ where P and Q are sentences

Sentences: built from terms and atoms

- A **quantified sentence** adds quantifiers $\forall$ and $\exists$
- A **well-formed formula (wff)** is a sentence containing no “free” variables. That is, all variables are “bound” by universal or existential quantifiers.
  - $(\forall x)P(x,y)$ has x bound as a universally quantified variable, but y is free

A BNF for FOL

```
S := <Sentence> ;
<Sentence> := <AtomicSentence> |
          <Sentence> <Connective> <Sentence> |
          <Quantifier> <Variable>,... <Sentence> |
          "NOT" <Sentence> |
          "(" <Sentence> ")";
<AtomicSentence> := <Predicate> "(" <Term>, ... ")" |
                    <Term> "=" <Term>;
<Term> := <Function> "(" <Term>, ... ")" |
          <Constant> |
          <Variable>;
<Connective> := "AND" | "OR" | "IMPLIES" | "EQUIVALENT";
<Quantifier> := "EXISTS" | "FORALL" ;
<Constant> := "A" | "X1" | "John" | ... ;
<Variable> := "a" | "x" | "s" | ... ;
<Predicate> := "Before" | "HasColor" | "Raining" | ... ;
<Function> := "Mother" | "LeftLegOf" | ... ;
```

Quantifiers

- **Universal quantification**
  - $(\forall x)P(x)$ means P holds for **all** values of x in domain associated with variable
  - E.g., $(\forall x)$ dolphin(x) $\rightarrow$ mammal(x)
- **Existential quantification**
  - $(\exists x)P(x)$ means P holds for **some** value of x in domain associated with variable
  - E.g., $(\exists x)$ mammal(x) $\wedge$ lays_eggs(x)
  - Permits one to make a statement about some object without naming it

Quantifiers

- Universal quantifiers often used with *implies* to form *rules*:
  - $(\forall x)\ student(x) \rightarrow smart(x)$ means "All students are smart"
- Universal quantification is *rarely* used to make blanket statements about every individual in the world:
  - $(\forall x) student(x) \wedge smart(x)$ means "Everyone in the world is a student and is smart"
- Existential quantifiers are usually used with "and" to specify a list of properties about an individual:
  - $(\exists x)\ student(x) \wedge smart(x)$ means "There is a student who is smart"
- Common mistake: represent this EN sentence in FOL as:
  - $(\exists x)\ student(x) \rightarrow smart(x)$
  - What does this sentence mean?

Quantifier Scope

- FOL sentences have structure, like programs
- In particular, the variables in a sentence have a scope
- For example, suppose we want to say
  - "everyone who is alive loves someone"
  - $(\forall x)\ alive(x) \rightarrow (\exists y)\ loves(x,y)$
- Here's how we scoce the variables

$(\forall x)\ alive(x) \rightarrow (\exists y)\ loves(x,y)$

Scope of x
Scope of y

Quantifier Scope

- **Switching order of universal quantifiers *does not* change the meaning**
  - $(\forall x)(\forall y)P(x,y) \leftrightarrow (\forall y)(\forall x)\ P(x,y)$
  - "Dogs hate cats"
- **You can switch order of existential quantifiers**
  - $(\exists x)(\exists y)P(x,y) \leftrightarrow (\exists y)(\exists x)\ P(x,y)$
  - "A cat killed a dog"
- **Switching order of universals and existentials *does* change meaning:**
  - Everyone likes someone: $(\forall x)(\exists y)\ likes(x,y)$
  - Someone is liked by everyone: $(\exists y)(\forall x)\ likes(x,y)$

Connections between All and Exists

- We can relate sentences involving $\forall$ and $\exists$ using **De Morgan's laws**:
  1. $(\forall x)\ \neg P(x) \leftrightarrow \neg(\exists x)\ P(x)$
  2. $\neg(\forall x)\ P \leftrightarrow (\exists x)\ \neg P(x)$
  3. $(\forall x)\ P(x) \leftrightarrow \neg\ (\exists x)\ \neg P(x)$
  4. $(\exists x)\ P(x) \leftrightarrow \neg(\forall x)\ \neg P(x)$
- Examples
  1. All dogs don't like cats ↔ No dogs like cats
  2. Not all dogs dance ↔ There is a dog that doesn't dance
  3. All dogs sleep ↔ There is no dog that doesn't sleep
  4. There is a dog that talks ↔ Not all dogs can't talk

Quantified inference rules

- Universal instantiation
  - ∀x P(x) ∴ P(A)
- Universal generalization
  - P(A) ∧ P(B) … ∴ ∀x P(x)
- Existential instantiation
  - ∃x P(x) ∴P(F) ←**skolem constant F**
    F must be a "new" constant not appearing in the KB
- Existential generalization
  - P(A) ∴ ∃x P(x)

Universal instantiation (a.k.a. universal elimination)

- If (∀x) P(x) is true, then P(C) is true, where C is *any* constant in the domain of x, e.g.:

  (∀x) eats(John, x) ⇒
  eats(John, Cheese18)
- Note that function applied to ground terms is also a constant

  (∀x) eats(John, x) ⇒
  eats(John, contents(Box42))

Existential instantiation (a.k.a. existential elimination)

- From (∃x) P(x) infer P(c), e.g.:
  - (∃x) eats(Mickey, x) → eats(Mickey, Stuff345)
- The variable is replaced by a **brand-new constant** not occurring in this or any sentence in the KB
- Also known as skolemization; constant is a **skolem constant**
- We don't want to accidentally draw other inferences about it by introducing the constant
- Can use this to reason about unknown objects, rather than constantly manipulating existential quantifiers

Existential generalization (a.k.a. existential introduction)

- If P(c) is true, then (∃x) P(x) is inferred, e.g.:

  Eats(Mickey, Cheese18) ⇒
  (∃x) eats(Mickey, x)
- All instances of the given constant symbol are replaced by the new variable symbol
- Note that the variable symbol cannot already exist anywhere in the expression

Translating English to FOL

Every gardener likes the sun

$\forall x\ \text{gardener}(x) \rightarrow \text{likes}(x,\text{Sun})$

You can fool some of the people all of the time

$\exists x\ \forall t\ \ \text{person}(x) \wedge \text{time}(t) \rightarrow \text{can-fool}(x, t)$

You can fool all of the people some of the time

$\forall x\ \exists t\ (\text{person}(x) \rightarrow \text{time}(t) \wedge \text{can-fool}(x, t))$

$\forall x\ (\text{person}(x) \rightarrow \exists t\ (\text{time}(t) \wedge \text{can-fool}(x, t))$

Note 2 possible readings of NL sentence

All purple mushrooms are poisonous

$\forall x\ (\text{mushroom}(x) \wedge \text{purple}(x)) \rightarrow \text{poisonous}(x)$

Translating English to FOL

No purple mushroom is poisonous (two ways)

$\neg\exists x\ \text{purple}(x) \wedge \text{mushroom}(x) \wedge \text{poisonous}(x)$

$\forall x\ \ (\text{mushroom}(x) \wedge \text{purple}(x)) \rightarrow \neg\text{poisonous}(x)$

There are exactly two purple mushrooms

$\exists x\ \exists y\ \text{mushroom}(x) \wedge \text{purple}(x) \wedge \text{mushroom}(y) \wedge \text{purple}(y) \wedge \neg(x=y) \wedge \forall z\ (\text{mushroom}(z) \wedge \text{purple}(z)) \rightarrow ((x=z) \vee (y=z))$

Obana is not short

$\neg\text{short}(\text{Obama})$

Logic and People

"Logic—the last refuge of a scoundrel."

- People can easily be confused by logic
- And are often suspicious of it, or give it too much weight

Monty Python example (Russell & Norvig)

FIRST VILLAGER: We have found a witch. May we burn her?
ALL: A witch! Burn her!
BEDEVERE: Why do you think she is a witch?
SECOND VILLAGER: She turned *me* into a newt.
B: A newt?
V2 *(after looking at himself for some time)*: I got better.
ALL: Burn her anyway.
B: Quiet! Quiet! There are ways of telling whether she is a witch.

B: Tell me… what do you do with witches?

ALL: Burn them!

B: And what do you burn, apart from witches?

V4: …wood?

B: So **why do witches burn**?

V2 *(pianissimo)*: **because they're made of wood?**

B: Good.

ALL: I see. Yes, of course.

B: So how can we tell if she is made of wood?

V1: Make a bridge out of her.

B: Ah… but can you not also make bridges out of stone?

ALL: Yes, of course… um… er…

B: Does wood sink in water?

ALL: No, no, it floats. Throw her in the pond.

B: Wait. Wait… tell me, what also floats on water?

ALL: Bread? No, no no. Apples… gravy… very small rocks…

B: No, no, no,

KING ARTHUR: A duck!

(They all turn and look at Arthur. Bedevere looks up, very impressed.)

B: Exactly. So… logically…

V1 *(beginning to pick up the thread)*: **If she… weighs the same as a duck… she's made of wood**.

B: And therefore?

ALL: **A witch!**

Fallacy: Affirming the conclusion

$\forall x\ \text{witch}(x) \rightarrow \text{burns}(x)$

$\forall x\ \text{wood}(x) \rightarrow \text{burns}(x)$

$\therefore\ \forall z\ \text{witch}(x) \rightarrow \text{wood}(x)$

$p \rightarrow q$

$r \rightarrow q$

$p \rightarrow r$

Monty Python Near-Fallacy #2

wood(x) → can-build-bridge(x)

∴ can-build-bridge(x) → wood(x)

- B: Ah… but can you not also make bridges out of stone?

Monty Python Fallacy #3

∀x wood(x) → floats(x)

∀x duck-weight (x) → floats(x)

∴ ∀x duck-weight(x) → wood(x)

p → q

r → q

∴ r → p

Monty Python Fallacy #4

∀z light(z) → wood(z)

light(W)

∴ wood(W) % ok…………..

witch(W) → wood(W) % applying universal instan. % to fallacious conclusion #1

wood(W)

∴ witch(z)

Example: A simple genealogy KB by FOL

- **Build a small genealogy knowledge base using FOL that**
  - contains facts of immediate family relations (spouses, parents, etc.)
  - contains definitions of more complex relations (ancestors, relatives)
  - is able to answer queries about relationships between people
- **Predicates:**
  - parent(x, y), child(x, y), father(x, y), daughter(x, y), etc.
  - spouse(x, y), husband(x, y), wife(x,y)
  - ancestor(x, y), descendant(x, y)
  - male(x), female(y)
  - relative(x, y)
- **Facts:**
  - husband(Joe, Mary), son(Fred, Joe)
  - spouse(John, Nancy), male(John), son(Mark, Nancy)
  - father(Jack, Nancy), daughter(Linda, Jack)
  - daughter(Liz, Linda)
  - etc.

- **Rules for genealogical relations**
  - (∀x,y) parent(x, y) ↔ child (y, x)
    (∀x,y) father(x, y) ↔ parent(x, y) ∧ male(x) *;similarly for mother(x, y)*
    (∀x,y) daughter(x, y) ↔ child(x, y) ∧ female(x) *;similarly for son(x, y)*
  - (∀x,y) husband(x, y) ↔ spouse(x, y) ∧ male(x) *;similarly for wife(x, y)*
    (∀x,y) spouse(x, y) ↔ spouse(y, x) ***;spouse relation is symmetric***
  - (∀x,y) parent(x, y) → ancestor(x, y)
    (∀x,y)(∃z) parent(x, z) ∧ ancestor(z, y) → ancestor(x, y)
  - (∀x,y) descendant(x, y) ↔ ancestor(y, x)
  - (∀x,y)(∃z) ancestor(z, x) ∧ ancestor(z, y) → relative(x, y)
    ;related by common ancestry
    (∀x,y) spouse(x, y) → relative(x, y) ;related by marriage
    (∀x,y)(∃z) relative(z, x) ∧ relative(z, y) → relative(x, y) ;**transitive**
    (∀x,y) relative(x, y) ↔ relative(y, x) **;symmetric**
- **Queries**
  - ancestor(Jack, Fred) ; the answer is yes
  - relative(Liz, Joe) ; the answer is yes
  - relative(Nancy, Matthew)
    ;no answer in general, no if under closed world assumption
  - (∃z) ancestor(z, Fred) ∧ ancestor(z, Liz)

Axioms for Set Theory in FOL

1. The only sets are the empty set and those made by adjoining something to a set:
   ∀s set(s) <=> (s=EmptySet) v (∃x,r Set(r) ^ s=Adjoin(s,r))
2. The empty set has no elements adjoined to it:
   ~ ∃x,s Adjoin(x,s)=EmptySet
3. Adjoining an element already in the set has no effect:
   ∀x,s Member(x,s) <=> s=Adjoin(x,s)
4. The only members of a set are the elements that were adjoined into it:
   ∀x,s Member(x,s) <=> ∃y,r (s=Adjoin(y,r) ^ (x=y v Member(x,r)))
5. A set is a subset of another iff all of the 1st set's members are members of the 2^{nd}:
   ∀s,r Subset(s,r) <=> (∀x Member(x,s) => Member(x,r))
6. Two sets are equal iff each is a subset of the other:
   ∀s,r (s=r) <=> (subset(s,r) ^ subset(r,s))
7. Intersection
   ∀x,s1,s2 member(X,intersection(S1,S2)) <=> member(X,s1) ^ member(X,s2)
8. Union
   ∃x,s1,s2 member(X,union(s1,s2)) <=> member(X,s1) v member(X,s2)

Semantics of FOL

- **Domain M:** the set of all objects in the world (of interest)
- **Interpretation I:** includes
  - Assign each constant to an object in M
  - Define each function of n arguments as a mapping M^n => M
  - Define each predicate of n arguments as a mapping M^n => {T, F}
  - Therefore, every ground predicate with any instantiation will have a truth value
  - In general there is an infinite number of interpretations because |M| is infinite
- **Define logical connectives: ~, ^, v, =>, <=>** as in PL
- **Define semantics of (∀x) and (∃x)**
  - (∀x) P(x) is true iff P(x) is true under all interpretations
  - (∃x) P(x) is true iff P(x) is true under some interpretation

- **Model:** an interpretation of a set of sentences such that every sentence is *True*
- **A sentence is**
  - **satisfiable** if it is true under some interpretation
  - **valid** if it is true under all possible interpretations
  - **inconsistent** if there does not exist any interpretation under which the sentence is true
- **Logical consequence:** S |= X if all models of S are also models of X

Axioms, definitions and theorems

- **Axioms** are facts and rules that attempt to capture all of the (important) facts and concepts about a domain; axioms can be used to prove **theorems**
  - Mathematicians don't want any unnecessary (dependent) axioms, i.e. ones that can be derived from other axioms
  - Dependent axioms can make reasoning faster, however
  - Choosing a good set of axioms for a domain is a design problem
- A **definition** of a predicate is of the form "p(X) ↔ …" and can be decomposed into two parts
  - **Necessary** description: "p(x) → …"
  - **Sufficient** description "p(x) ← …"
  - Some concepts don't have complete definitions (e.g., person(x))

More on definitions

Example: define father(x, y) by parent(x, y) and male(x)

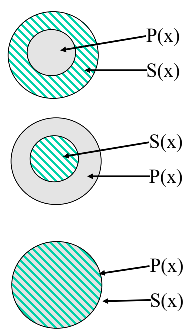
- **parent(x, y)** is a necessary (but not sufficient) description of father(x, y)

  father(x, y) → parent(x, y)
- **parent(x, y) ^ male(x) ^ age(x, 35)** is a sufficient (but not necessary) description of father(x, y):

  father(x, y) ← parent(x, y) ^ male(x) ^ age(x, 35)
- **parent(x, y) ^ male(x)** is a necessary and sufficient description of father(x, y)

  parent(x, y) ^ male(x) ↔ father(x, y)

More on definitions



S(x) is a necessary condition of P(x) — (∀x) P(x) => S(x)

S(x) is a sufficient condition of P(x) — (∀x) P(x) <= S(x)

S(x) is a necessary and sufficient condition of P(x) — (∀x) P(x) <=> S(x)

Higher-order logic

- FOL only allows to quantify over variables, and variables can only range over objects.
- HOL allows us to quantify over relations
- Example: (quantify over functions)

  "two functions are equal iff they produce the same value for all arguments"

  ∀f ∀g (f = g) ↔ (∀x f(x) = g(x))
- Example: (quantify over predicates)

  ∀r transitive(r) → (∀xyz) r(x,y) ∧ r(y,z) → r(x,z))
- More expressive, but undecidable, in general

Expressing uniqueness

- Sometimes we want to say that there is a single, unique object that satisfies a certain condition
- "There exists a unique x such that king(x) is true"
  - ∃x king(x) ∧ ∀y (king(y) → x=y)
  - ∃x king(x) ∧ ¬∃y (king(y) ∧ x≠y)
  - ∃! x king(x)
- "Every country has exactly one ruler"
  - ∀c country(c) → ∃! r ruler(c,r)
- Iota operator: "ι x P(x)" means "the unique x such that p(x) is true"
  - "The unique ruler of Freedonia is dead"
  - dead(ι x ruler(freedonia,x))

Notational differences

- **Different symbols** for *and, or, not, implies, ...*
  - ∀ ∃ ⇒ ⇔ ∧ ∨ ¬ • ⊃
  - p v (q ^ r)
  - p + (q * r)
  - etc
- **Prolog**

  cat(X) :- furry(X), meows (X), has(X, claws)
- **Lispy notations**

```
(forall ?x (implies (and (furry ?x)
                         (meows ?x)
                         (has ?x claws))
                    (cat ?x)))
```


Summary

- First order logic (FOL) introduces predicates, functions and quantifiers
- Much more expressive, but reasoning is more complex
  - Reasoning is semi-decidable
- FOL is a common AI knowledge representation language
- Other KR languages (e.g., OWL) are often defined by mapping them to FOL
- FOL variables range over objects
- HOL variables can ranger over functions, predicates or sentences