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,** 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
  - BarackObama, 3, Green
- **Function symbols,** map individuals to individuals
  - father_of(SashaObama) = BarackObama
  - 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 ($\neg$), and ($\wedge$), or ($\vee$), implies ($\rightarrow$), iff ($\leftrightarrow$)
- **Quantifiers**
  - Universal $\forall$**x** or **(Ax)**
  - Existential $\exists$**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, e.g.:
  - Constants: john, umbc
  - Variables: x, y, z
  - Functions: mother_of(john), phone(mother(x))
- **Ground terms** have no variables in them
  - **Ground:** john, father_of(father_of(john))
  - **Not Ground:** father_of(X)

Sentences: built from terms and atoms

- An **atomic sentence** (which has value true or false) is an n-place predicate of n terms, e.g.:
  - green(Kermit))
  - between(Philadelphia, Baltimore, DC)
  - loves(X, mother(X))
- A **complex sentence** is formed from atomic sentences connected by logical connectives:

  $\neg P, P \vee Q, P \wedge Q, P \rightarrow Q, P \leftrightarrow Q$

  where P and Q are sentences

What do atomic sentences mean?

- Unary predicates typically encode a **type** or **is_a** relationship
  - Dolphin(flipper): flipper is a kind of dolphin
  - Green(kermit): kermit is a kind of green thing
  - Integer(x): x is a kind of integer
- Non-unary predicates typically encode relations
  - Loves(john, mary)
  - Greater_than(2, 1)
  - Between(newYork, philadelphia, baltimore)

Ontologies

- Designing a logic representation is similar to modeling in an object-oriented language
- An **ontology** is a “formal naming and definition of the types, properties, and interrelationships of the entities that really exist in a particular domain of discourse”
- See schema.org as for an ontology that’s used by search engines to add semantic data to web sites

Sentences: built from terms and atoms

- **quantified sentences** adds quantifiers ∀ and ∃
  - ∀x loves(x, mother(x))
  - ∃x number(x) ∧ greater(x, 100), prime(x)
- A **well-formed formula (wff)** is a sentence with no *free* variables; all variables are *bound* by either a universal or existential *quantifier*

  In **(∀x)P(x, y)** x is bound and y is free

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)
  - This lets us make a statement about some object without identifying it

Quantifiers (1)

- Universal quantifiers often used with *implies* to form *rules*:

  $(\forall x)\ student(x) \rightarrow smart(x)$ means "All students are smart"

- Universal quantification *rarely* used to make blanket statements about every individual in the world:

  $(\forall x)\ student(x) \wedge smart(x)$ means "Everything in the world is a student and is smart"

Quantifiers (2)

- Existential quantifiers 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 in FOL as:

  $(\exists x)\ student(x) \rightarrow smart(x)$

- What does this sentence mean?
  - ??

Quantifiers (2)

- Existential quantifiers 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 in FOL as:

  $(\exists x)\ student(x) \rightarrow smart(x)$

- What does this sentence mean?
  - **$P \rightarrow Q$ = ~P v Q**
  - **$\exists x$ student(x) $\rightarrow$ smart(x) = $\exists x$ ~student(x) v smart(x)**
  - **There's something that is not a student or is smart**

Quantifier Scope

- FOL sentences have structure, like programs
- In particular, 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 scope 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 (i.e., all dogs hate all 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 universal and existential quantifiers *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)

Procedural example 1

```
def verify1():
    # Everyone likes someone: (∀x)(∃y) likes(x,y)
    for p1 in people():
        foundLike = False
        for p2 in people():
            if likes(p1, p2):
                foundLike = True
                break
         if not FoundLike:
          print(p1, 'does not like anyone ☹')
          return False
    return True
```

Every person has at least one individual that they like.

Procedural example 2

```
def verify2():
    # Someone is liked by everyone: (∃y)( ∀x) likes(x,y)
    for p2 in people():
        foundHater = False
        for p1 in people():
            if not likes(p1, p2):
                foundHater = True
                break
        if not foundHater
            print(p2, ‘is liked by everyone ☺’)
            return True
    return False
```

There is a person who is liked by every person in the universe.

Connections between ∀ and ∃

- We can relate sentences involving ∀ and ∃ using extensions to **De Morgan's laws**:
  1. $(\forall x)\ \neg P(x) \leftrightarrow \neg(\exists x)\ P(x)$
  2. $\neg(\forall x)\ P(x) \leftrightarrow (\exists x)\ \neg P(x)$
  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 dog likes 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

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

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

  $(\forall x)$ eats(John, x) $\Rightarrow$
  eats(John, Cheese18)

- Note that function applied to ground terms is also a constant

  $(\forall x)$ eats(John, x) $\Rightarrow$
  eats(John, contents(Box42))

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

- From $(\exists x)$ P(x) infer P(c), e.g.:
  - $(\exists x)$ eats(Mikey, x) $\rightarrow$ eats(Mikey, 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 $(\exists x)$ P(x) is inferred, e.g.:

  Eats(Mickey, Cheese18) $\Rightarrow$

  $(\exists x)$ 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

$\forall x$ gardener(x) $\rightarrow$ likes(x,Sun)

All purple mushrooms are poisonous

$\forall x$ (mushroom(x) $\wedge$ purple(x)) $\rightarrow$ poisonous(x)

No purple mushroom is poisonous (two ways)

$\neg\exists x$ purple(x) $\wedge$ mushroom(x) $\wedge$ poisonous(x)

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

Translating English to FOL

There are (at least) two purple mushrooms

$\exists x \; \exists y \; mushroom(x) \land purple(x) \land mushroom(y) \land purple(y) \land \neg(x=y)$

There are exactly two purple mushrooms

$\exists x \; \exists y \; mushroom(x) \land purple(x) \land mushroom(y) \land purple(y) \land \neg(x=y) \land$
$\forall z \; (mushroom(z) \land purple(z)) \rightarrow ((x=z) \lor (y=z))$

Obama is not short

$\neg short(Obama)$

Translating English to FOL

What do these mean?

- **You can fool some of the people all of the time**
- **You can fool all of the people some of the time**

Translating English to FOL

What do these mean?

Both English statements are ambiguous

- **You can fool some of the people all of the time**

  There is a nonempty group of people so easily fooled that you can fool that group every time*

  For any given time, there is a non-empty group at that time that you can fool

- **You can fool all of the people some of the time**

  There are one or more times when it's possible to fool everyone*

  Everybody can be fooled at some point in time

* Most common interpretation, I think

Some terms we will need

- **person(x):** True iff x is a person
- **time(t):** True iff t is a point in time
- **canFool(x, t):** True iff x can be fooled at time t

Translating English to FOL

You can fool some of the people all of the time

There is a nonempty group of people so easily fooled that you can fool that group every time*

≡ There's a person that you can fool every time

$\exists x \; \forall t \;\; person(x) \land time(t) \rightarrow canFool(x, t)$

For any given time, there is a non-empty group at that time that you can fool

≡ For every time, there is a person at that time that you can fool

$\forall t \; \exists x \;\; person(x) \land time(t) \rightarrow canFool(x, t)$

* Most common interpretation, I think

Translating English to FOL

You can fool all of the people some of the time

There are one or more times when it's possible to fool everyone*

$\exists t \, \forall x \; time(t) \land person(x) \rightarrow canFool(x, t)$

Everybody can be fooled at some point in time

$\forall x \, \exists t \; person(x) \land time(t) \rightarrow canFool(x, t)$

* Most common interpretation, I think

Simple genealogy KB in FOL

Design a knowledge base using FOL that

- Has facts of immediate family relations, e.g., spouses, parents, etc.
- Defines of more complex relations (ancestors, relatives)
- Detect conflicts, e.g., you are your own parent
- Infers relations, e.g., grandparent from parent
- Answers queries about relationships between people

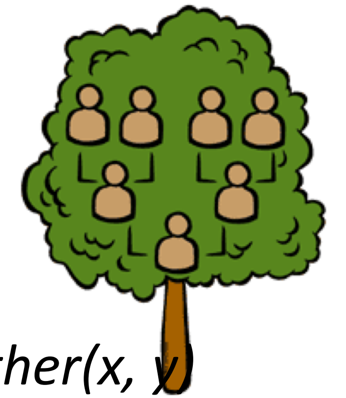
How do we approach this?

- Design an initial ontology of types, e.g.
  - e.g., person, man, woman, male, female
- Extend ontology by defining relations, e.g.
  - spouse, has_child, has_parent
- Add general constraints to relations, e.g.
  - spouse(X,Y) => ~ X = Y
  - spouse(X,Y) => person(X), person(Y)
- Add FOL sentences for inference, e.g.
  - spouse(X,Y) ⇔ spouse(Y,X)
  - man(X) ⇔ person(X) ∧ male(X)

Example: A simple genealogy KB by FOL

- **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.

Example Axioms



$(\forall x,y)$ parent(x, y) $\leftrightarrow$ child (y, x)

$(\forall x,y)$ father(x, y) $\leftrightarrow$ parent(x, y) $\wedge$ male(x) *;similar for mother(x, y)*

$(\forall x,y)$ daughter(x, y) $\leftrightarrow$ child(x, y) $\wedge$ female(x) *;similar for son(x, y)*

$(\forall x,y)$ husband(x, y) $\leftrightarrow$ spouse(x, y) $\wedge$ male(x) *;similar for wife(x, y)*

$(\forall x,y)$ spouse(x, y) $\leftrightarrow$ spouse(y, x) *;spouse relation is symmetric*

$(\forall x,y)$ parent(x, y) $\rightarrow$ ancestor(x, y)

$(\forall x,y)(\exists z)$ parent(x, z) $\wedge$ ancestor(z, y) $\rightarrow$ ancestor(x, y)

$(\forall x,y)$ descendant(x, y) $\leftrightarrow$ ancestor(y, x)

$(\forall x,y)(\exists z)$ ancestor(z, x) $\wedge$ ancestor(z, y) $\rightarrow$ relative(x, y)

$(\forall x,y)$ spouse(x, y) $\rightarrow$ relative(x, y) ;related by marriage

$(\forall x,y)(\exists z)$ relative(z, x) $\wedge$ relative(z, y) $\rightarrow$ relative(x, y) *;transitive*

$(\forall x,y)$ relative(x, y) $\leftrightarrow$ relative(y, x) *;symmetric*

Axioms, definitions and theorems

- **Axioms**: facts and rules that capture (important) facts & concepts in a domain; axioms are used to prove **theorems**
  - Mathematicians dislike unnecessary (dependent) axioms, i.e. ones that can be derived from others
  - Dependent axioms can make reasoning faster, however
  - Choosing a good set of axioms is a design problem
- A **definition** of a predicate is of the form "p(X) $\leftrightarrow$ …" and can be decomposed into two parts
  - **Necessary** description: "p(x) $\rightarrow$ …"
  - **Sufficient** description "p(x) $\leftarrow$ …"
  - Some concepts have definitions (e.g., triangle) and some don't (e.g., person)

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) $\rightarrow$ parent(x, y)

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

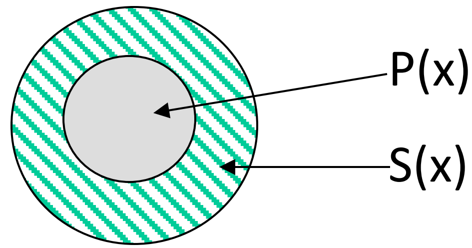
  father(x, y) $\leftarrow$ parent(x, y) ^ male(x) ^ age(x, 35)

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

  parent(x, y) ^ male(x) $\leftrightarrow$ father(x, y)

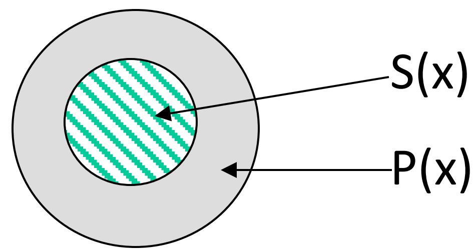
More on definitions

S(x) is a
necessary
condition of P(x)



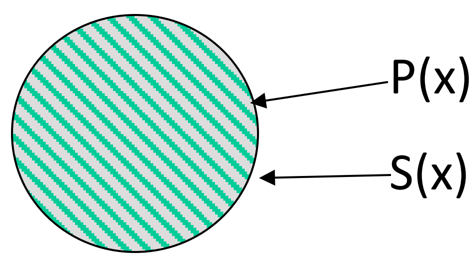
 # all Ps are Ss
(∀x) P(x) => S(x)

S(x) is a
sufficient
condition of P(x)



 # all Ps are Ss
(∀x) P(x) <= S(x)

S(x) is a
necessary and
sufficient
condition of P(x)



all Ps are Ss
all Ss are Ps
(∀x) P(x) <=> S(x)

Higher-order logic

- FOL only lets us quantify over variables, and **variables can only range over objects**
- HOL allows us to quantify over relations, e.g.

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

  $\forall f\ \forall g\ (f = g) \leftrightarrow (\forall x\ f(x) = g(x))$
- E.g.: (quantify over predicates)

  $\forall r\ \text{transitive}(\ r\ ) \rightarrow (\forall xyz)\ r(x,y) \wedge r(y,z) \rightarrow r(x,z))$
- More expressive, but undecidable, in general

Expressing uniqueness

syntactic sugar

- Often want to say that there is a single, unique object that satisfies a condition
- There exists a unique x such that king(x) is true
  - $\exists x\ king(x) \land \forall y\ (king(y) \rightarrow x=y)$
  - $\exists x\ king(x) \land \neg\exists y\ (king(y) \land x \neq y)$
  - $\exists!\ x\ king(x)$
- Every country has exactly one ruler
  - $\forall c\ country(c) \rightarrow \exists!\ r\ ruler(c,r)$
- Iota operator: $\iota\ x\ P(x)$ means "the unique x such that p(x) is true"
  - The unique ruler of Freedonia is dead
  - $dead(\iota\ x\ ruler(freedonia,x))$

Notational differences

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

A example of FOL in use

- Semantics of W3C's Semantic Web stack (RDF, RDFS, OWL) is defined in FOL
- OWL Full is equivalent to FOL
- Other OWL profiles support a subset of FOL and are more efficient
- However, the semantics of schema.org is only defined in natural language text
- …and Google's knowledge Graph probably (!) uses probabilities

FOL Summary

- First order logic (FOL) introduces predicates, functions and quantifiers
- More expressive, but reasoning more complex
  - Reasoning in propositional logic is NP hard, FOL is semi-decidable
- 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 range over functions, predicates or sentences