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
  - 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 (¬), 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, 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)

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 containing no "free" variables, i.e., all variables are "bound" by either a universal or existential quantifiers

  (∀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) → 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) ∧ lays_eggs(x)
  - This lets us make a statement about some object without naming 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 "Everyone 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 -> Q = ~P v Q**
  - **Ex student(x) -> smart(x) = Ex ~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 x in people():
        found = False
        for y in people():
            if likes(x,y):
                found = True
                break
         if not Found:
            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 y in people():
        found = True
        for x in people():
            if not likes(x,y):
                found = False
                break
        if found
            return True
    return False
```

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

Connections between $\forall$ and $\exists$

- We can relate sentences involving $\forall$ and $\exists$ 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 $\leftrightarrow$ No dogs like cats
  2. Not all dogs dance $\leftrightarrow$ There is a dog that doesn't dance
  3. All dogs sleep $\leftrightarrow$ There is no dog that doesn't sleep
  4. There is a dog that talks $\leftrightarrow$ Not all dogs can't talk

Quantified inference rules

- Universal instantiation
  - ∀x P(x) ∴ P(A)   *# where A is some constant*
- Universal generalization
  - P(A) ∧ P(B) … ∴ ∀x P(x) *# if AB… enumerate all*
    # individuals
- 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)

* After [Thoralf Skolem]

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.:

  $$\text{Eats(Mickey, Cheese18)} \Rightarrow (\exists x)\ \text{eats(Mickey, x)}$$

- 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

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

You can fool some of the people all of the time

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

You can fool all of the people some of the time

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

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

Note 2 possible readings of NL sentence

All purple mushrooms are poisonous

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

Translating English to FOL

No purple mushroom is poisonous (two ways)

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

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

There are (at least) two purple mushrooms

$\exists x\ \exists y\ mushroom(x) \wedge purple(x) \wedge mushroom(y) \wedge purple(y) \wedge \neg(x=y)$

There are exactly two purple mushrooms

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

Obama is not short

$\neg short(Obama)$

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

$\forall x$ witch(x) $\rightarrow$ burns(x)

$\forall x$ wood(x) $\rightarrow$ burns(x)

$\therefore$ $\forall z$ witch(x) $\rightarrow$ wood(x)

$p \rightarrow q$

$r \rightarrow q$

$p \rightarrow r$

Monty Python Near-Fallacy #2

$\text{wood}(x) \rightarrow \text{can-build-bridge}(x)$

$\therefore\ \text{can-build-bridge}(x) \rightarrow \text{wood}(x)$

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

Monty Python Fallacy #3

$\forall x$ wood(x) $\rightarrow$ floats(x)

$\forall x$ duck-weight (x) $\rightarrow$ floats(x)

$\therefore\ \forall x$ duck-weight(x) $\rightarrow$ wood(x)

$p \rightarrow q$

$r \rightarrow q$

$\therefore\ r \rightarrow p$

Monty Python Fallacy #4

$\forall z$ light(z) $\rightarrow$ wood(z)

light(W)

$\therefore$ wood(W) % ok.............

witch(W) $\rightarrow$ wood(W) % applying universal instan.

% to fallacious conclusion #1

wood(W)

$\therefore$ witch(z)

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

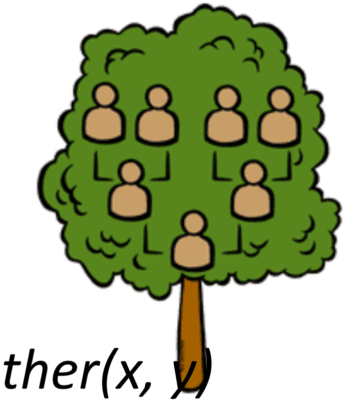
How do we approach this?

- Design an initial ontology of types, e.g.
  - e.g., person, man, woman, gender
- Add general individuals to ontology, e.g.
  - gender(male), gender(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) $\wedge$ has_gender(X, male)

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 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 ∨ Member(x,r)))
5. A set is a subset of another iff all of the 1st set's members are members of the 2nd:
   ∀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) ∨ 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's 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**: facts and rules that capture the (important) facts and concepts about a domain; axioms can be 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 "$p(X) \leftrightarrow \ldots$" and can be decomposed into two parts
  - **Necessary** description: "$p(x) \rightarrow \ldots$"
  - **Sufficient** description "$p(x) \leftarrow \ldots$"
  - 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):

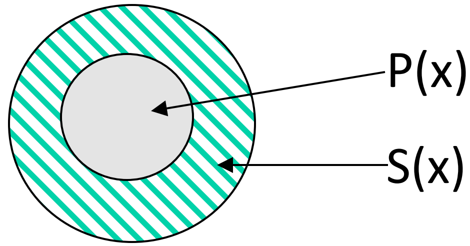
  father(x, y) $\leftarrow$ parent(x, y) ^ male(x) ^ age(x, 35)

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

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

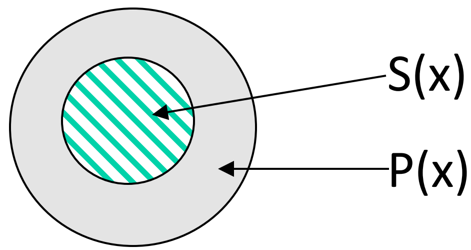
More on definitions

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



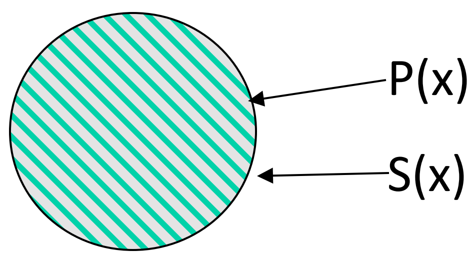
 # all Ps are Ss
$(\forall x)$ P(x) => S(x)

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



 # all Ps are Ss
$(\forall x)$ P(x) <= S(x)

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



 # all Ps are Ss
all Ss are Ps
$(\forall x)$ 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) $\wedge$ $\forall$y (king(y) $\rightarrow$ x=y)
  - $\exists$x king(x) $\wedge$ $\neg\exists$y (king(y) $\wedge$ 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: 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, ...*
  - $\forall$ $\exists$ $\Rightarrow$ $\Leftrightarrow$ $\wedge$ $\vee$ $\neg$ $\bullet$ $\supset$
  - 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