First-Order Logic
(First-Order Predicate Calculus)

Propositional vs. Predicate Logic

- In propositional logic, each possible atomic fact requires a separate unique propositional symbol.
- If there are *n* people and *m* locations, representing the fact that some person moved from one location to another requires nm^2 separate symbols.
- Predicate logic includes a richer **ontology:**
  - objects (terms)
  - properties (unary predicates on terms)
  - relations (*n*-ary predicates on terms)
  - functions (mappings from terms to other terms)
- Allows more flexible and compact representation of knowledge

  Move(x, y, z) for person x moved from location y to z.

Syntax for First-Order Logic

Sentence $\rightarrow$ AtomicSentence
| Sentence Connective Sentence
| Quantifier Variable Sentence
| $\neg$Sentence
| (Sentence)

AtomicSentence $\rightarrow$ Predicate(Term, Term, ...)
| Term=Term

Term $\rightarrow$ Function(Term,Term,...)
| Constant
| Variable

Connective $\rightarrow$ $\vee$ | $\wedge$ | $\Rightarrow$ | $\Leftrightarrow$

Quanitfier $\rightarrow$ $\exists$ | $\forall$

Constant $\rightarrow$ A | John | Car1

Variable $\rightarrow$ x | y | z |...

Predicate $\rightarrow$ Brother | Owns | ...

Function $\rightarrow$ father-of | plus | ...

First-Order Logic: Terms and Predicates

- Objects are represented by **terms**:
  - **Constants**: Block1, John
  - **Function symbols:** father-of, successor, plus
    An *n*-ary function maps a tuple of *n* terms to another term: father-of(John), succesor(0), plus(plus(1,1),2)
- Terms are simply names for objects. Logical functions are not procedural as in programming languages. They do not need to be defined, and do not really return a value. Allows for the representation of an infinite number of terms.
- Propositions are represented by a **predicate** applied to a tuple of terms. A predicate represents a property of or relation between terms that can be true or false: Brother(John, Fred), Left-of(Square1, Square2) GreaterThan(plus(1,1), plus(0,1))
- In a given interpretation, an *n*-ary predicate can defined as a function from tuples of *n* terms to {True, False} or equivalently, a set tuples that satisfy the predicate:

  {<John, Fred>, <John, Tom>, <Bill, Roger>, ...}

Sentences in First-Order Logic

- An atomic sentence is simply a predicate applied to a set of terms.

  Owns(John,Car1)
  Sold(John,Car1,Fred)

  Semantics is True or False depending on the interpretation, i.e. is the predicate true of these arguments.

- The standard propositional connectives ($\vee$ $\neg$ $\wedge$ $\Rightarrow$ $\Leftrightarrow$) can be used to construct complex sentences:

  Owns(John,Car1) $\vee$ Owns(Fred, Car1)
  Sold(John,Car1,Fred) $\Rightarrow$ $\neg$Owns(John, Car1)

  Semantics same as in propositional logic.

Quantifiers

- Allows statements about entire collections of objects rather than having to enumerate the objects by name.

- Universal quantifier: $\forall$x
  Asserts that a sentence is true for all values of variable x

  $\forall$x Loves(x, FOPC)
  $\forall$x Whale(x) $\Rightarrow$ Mammal(x)
  $\forall$x Grackles(x) $\Rightarrow$ Black(x)
  $\forall$x ($\forall$y Dog(y) $\Rightarrow$ Loves(x,y)) $\Rightarrow$ ($\forall$z Cat(z) $\Rightarrow$ Hates(x,z))

- Existential quantifier: $\exists$
  Asserts that a sentence is true for at least one value of a variable x

  $\exists$x Loves(x, FOPC)
  $\exists$x(Cat(x) $\wedge$ Color(x,Black) $\wedge$ Owns(Mary,x))
  $\exists$x($\forall$y Dog(y) $\Rightarrow$ Loves(x,y)) $\wedge$ ($\forall$z Cat(z) $\Rightarrow$ Hates(x,z))

Use of Quantifiers

- Universal quantification naturally uses implication:

  $\forall$x Whale(x) $\wedge$ Mammal(x)

  Says that everything in the universe is both a whale and a mammal.

- Existential quantification naturally uses conjunction:

  $\exists$x Owns(Mary,x) $\Rightarrow$ Cat(x)

  Says either there is something in the universe that Mary does not own or there exists a cat in the universe.

  $\forall$x Owns(Mary,x) $\Rightarrow$ Cat(x)

  Says all Mary owns is cats (i.e. everthing Mary owns is a cat). Also true if Mary owns nothing.

  $\forall$x Cat(x) $\Rightarrow$ Owns(Mary,x)

  Says that Mary owns all the cats in the universe.
  Also true if there are no cats in the universe.

Nesting Quantifiers

- The order of quantifiers of the same type doesn't matter

  $\forall$x$\forall$y(Parent(x,y) $\wedge$ Male(y) $\Rightarrow$ Son(y,x))
  $\exists$x$\exists$y(Loves(x,y) $\wedge$ Loves(y,x))

- The order of mixed quantifiers does matter:

  $\forall$x$\exists$y(Loves(x,y))

  Says everybody loves somebody, i.e. everyone has someone whom they love.

  $\exists$y$\forall$x(Loves(x,y))

  Says there is someone who is loved by everyone in the universe.

  $\forall$y$\exists$x(Loves(x,y))

  Says everyone has someone who loves them.

  $\exists$x$\forall$y(Loves(x,y))

  Says there is someone who loves everyone in the universe.

Variable Scope

- The **scope** of a variable is the sentence to which the quantifier syntactically applies.

- As in a block structured programming language, a variable in a logical expression refers to the closest quantifier within whose scope it appears.

  $\exists x\ (Cat(x) \wedge \forall x(Black\ (x)))$

  The x in Black(x) is universally quantified

  Says cats exist and everything is black

- In a **well-formed formula** (**wff**) all variables should be properly introduced:

  $\exists x P(y)$ not well-formed

- A **ground** expression contains no variables.

Relation Between Quantifiers

- Universal and existential quantification are logically related to each other:

  $\forall x\ \neg Love(x,Saddam) \Leftrightarrow \neg\exists x\ Loves(x,Saddam)$

  $\forall x\ Love(x,Princess\text{-}Di) \Leftrightarrow \neg\exists x\ \neg Loves(x,Princess\text{-}Di)$

- General Identities

  - $\forall x\ \neg P \Leftrightarrow \neg\exists x\ P$
  - $\neg\forall x\ P \Leftrightarrow \exists x\ \neg P$
  - $\forall x\ P \Leftrightarrow \neg\exists x\ \neg P$
  - $\exists x\ P \Leftrightarrow \neg\forall x\ \neg P$

  - $\forall x\ P(x)\wedge Q(x) \Leftrightarrow \forall x P(x) \wedge \forall x Q(x)$
  - $\exists x\ P(x)\vee Q(x) \Leftrightarrow \exists x P(x) \vee \exists x Q(x)$

Equality

- Can include equality as a primitive predicate in the logic, or require it to be introduced and axiomitized as the **identity relation**.

- Useful in representing certain types of knowledge:

  $\exists x\exists y(Owns(Mary, x) \wedge Cat(x) \wedge Owns(Mary,y) \wedge Cat(y) \wedge \neg(x=y))$

  Mary owns two cats. Inequality needed to insure x and y are distinct.

  $\forall x\ \exists y\ married(x, y) \wedge \forall z(married(x,z) \Rightarrow y=z)$

  Everyone is married to exactly one person. Second conjunct is needed to guarantee there is only one unique spouse.

Higher-Order Logic

- FOPC is called **first-order** because it allows quantifiers to range over objects (terms) but not properties, relations, or functions applied to those objects.

- **Second-order** logic allows quantifiers to range over predicates and functions as well:

  $\forall x\ \forall y\ [\ (x=y) \Leftrightarrow (\forall p\ p(x) \Leftrightarrow p(y))\ ]$

  Says that two objects are equal if and only if they have exactly the same properties.

  $\forall f\ \forall g\ [\ (f=g) \Leftrightarrow (\forall x\ f(x) = g(x))\ ]$

  Says that two functions are equal if and only if they have the same value for all possible arguments.

- Third-order would allow quantifying over predicates of predicates, etc.

  For example, a second-order predicate would be Symetric(p) stating that a binary predicate p represents a symmetric relation.

Notational Variants

- In Prolog, variables in sentences are assumed to be universally quantified and implications are represented in a particular syntax.

  son(X, Y) :- parent(Y,X), male(X).

- In Lisp, a slightly different syntax is common.

  (forall ?x (forall ?y (implies (and (parent ?y ?x) (male ?x))
  (son ?x ?y)))

- Generally argument order follows the convention that P(x,y) in English would read “x is (the) P of y”

Logical KB

- KB contains general **axioms** describing the relations between predicates and **definitions** of predicates using ⇔.

  ∀x,y Bachelor(x) ⇔ Male(x) ∧ Adult(x) ∧ ¬∃yMarried(x,y).
  ∀x Adult(x) ⇔ Person(x) ∧ Age(x) >=18.

- May also contain specific ground facts.

  Male(Bob), Age(Bob)=21, Married(Bob, Mary)

- Can provide **queries** or **goals** as questions to the KB:

  Adult(Bob) ?
  Bachelor(Bob) ?

- If query is existentially quantified, would like to return **substitutions** or **binding lists** specifying values for the existential variables that satisfy the query.

  ∃x Adult(x) ?
  {x/Bob}

  ∃x Married(Bob,x) ?
  {x/Mary}

  ∃x,y Married(x,y) ?
  {x/Bob, y/Mary}

Sample Representations

- There is a barber in town who shaves all men in town who do not shave themselves.

  ∃x (Barber(x) ∧ InTown(x) ∧
  ∀y (Man(y) ∧ InTown(y) ∧ ¬Shave(y,y) ⇒ Shave(x,y)))

- There is a barber in town who shaves only and all men in town who do not shave themselves.

  ∃x (Barber(x) ∧ InTown(x) ∧
  ∀y (Man(y) ∧ InTown(y) ∧ ¬Shave(y,y) ⇔ Shave(x,y)))

- Classic example of Bertrand Russell used to illustrate a paradox in set theory: Does the set of all sets contain itself?