Propositional Logic Syntax

- symbols
  - logical constants True, False
  - propositional symbols P, Q, …
  - logical connectives
    - conjunction ∧, disjunction ∨,
    - negation ¬,
    - implication ⇒, equivalence ⇔
    - there are other connectives
      - unary, binary, n-ary
  - parentheses ( )
- sentences
  - constructed from simple sentences
  - conjunction, disjunction, implication, equivalence, negation

Propositional Logic Semantics

- Semantics is the *interpretation* of the propositional symbols and constants
  - symbols can stand for any arbitrary fact
    - sentences consisting of only a propositional symbols are limited
      - the value of the symbol can be True or False
      - must be explicitly stated in the model
  - the constants True and False have a fixed interpretation
    - True indicates that the world is as stated
    - False indicates that the world is not as stated
- specification of the logical connectives
  - frequently explicitly via truth tables

BNF Grammar – constructing legal sentences Propositional Logic

- Sentence → AtomicSentence | ComplexSentence
- AtomicSentence → True | False | P | Q | R | ...
- ComplexSentence → (Sentence)
- Sentence → | Sentence Connective Sentence | ¬Sentence
- Connective → ∧ | ∨ | ⇒ | ⇔

- ambiguities are resolved
  - implicitly through precedence: ¬ ∧ ∨ ⇒ ⇔
  - explicitly via parentheses (...)
  - e.g.
    - » ¬P ∨ Q ∧ R ⇒ S is equivalent to ((¬P) ∨ (Q ∧ R)) ⇒ S

Logics in general

Language	Ontological Commitment	Epistemological Commitment
Propositional logic	facts	true/false/unknown
First-order logic	facts, objects, relations	true/false/unknown
Temporal logic	facts, objects, relations, times	true/false/unknown
Probability theory	facts	degree of belief
Fuzzy logic	facts + degree of truth	known interval value

First-order Logic -- FOL (Predicate Logic)

Want a logic that is more expressive than Propositional Logic
Whereas propositional logic assumes world contains facts,
first-order logic (like natural language) assumes the world contains

- Objects:
  - people, houses, numbers, theories,, colors, baseball games, wars, centuries . . .
- Predicates can express:
  - Relations: red, round, bogus, prime, multistoried . . ., brother of, bigger than, inside, part of, has color, occurred after, owns, comes between, . . .
  - Functions: father of, best friend, third inning of, one more than, end of …

Introducing First Order (Predicate) Logic

- Predicate logic uses the following new features:
  - Variables: x, y, z
  - Predicates: $P(x)$, $M(x)$
  - Quantifiers (*to be covered in a few slides*):
- *Propositional functions* are a generalization of propositions.
  - They contain variables and a predicate, e.g., $P(x)$
  - Variables can be replaced by elements from their *domain*.
- Relations

Propositional Functions

- Propositional functions become propositions (and have truth values) when their variables are each replaced by a value from the *domain* (or *bound* by a quantifier, as we will see later).
- The statement *P(x)* is said to be the value of the propositional function *P* at *x*.
- For example, let *P(x)* denote "$x > 0$" and the domain be the integers. Then:
  P(-3) is false.
  P(0) is false.
  P(3) is true.
- Often the domain is denoted by *a Symbol, e.g U*. So, in this example *U* is the integers.

Representing Knowledge in FOL: Objects

- primarily distinguishable things in the real world
  - e.g. people, cars, computers, programs, ...
  - **the set of all objects determines the domain of a model**
- Information about objects can be expressed using predicates
  - frequently includes abstract concepts
    - » colors, stories, light, money, love, ...
    - » in contrast to physical objects
    - » Subdomains of the domain can be defined using propositional functions, e.g. Person(x)
- properties
  - describe specific aspects of objects
    - » green, round, heavy, visible, …
  - can be used to distinguish between objects: e.g. green(tree) $\rightarrow$ True

Representing Knowledge in FOL: Relations

- used to establish connections between objects
  - unary relations refer to a single object
    - » e.g. mother-of(John), brother-of(Jill), spouse-of(Joe)
  - binary relations relate two objects to each other
    - » e.g. twins(John,Jill), married(Joe, Jane)
  - n-ary relations relate n objects to each other
    - » e.g. triplets(Jim, Tim, Wim), seven-dwarfs(D1, ..., D7)

- functions are a special type of relation
  - non-ambiguous: only one output for a given input
  - often distinguished from similar binary relations by appending -of
    - » e.g. father(John, Jim) (T/F) vs. father-of(John) → Jim

Quantifiers

- can be used to express properties of **collections of objects**
  - eliminates the need to explicitly enumerate all objects
- predicate logic uses two quantifiers
  - universal quantifier ∀ : true for all objects in a collection
  - existential quantifier ∃ : true for at least one thing in a collection

Quantifiers

Universal Quantification: states that

- a predicate *P* holds for all objects *x* in the universe under discourse
- written as $\forall x\ P(x)$
- the sentence is true if and only if all the individual sentences where the variable *x* is replaced by the individual objects it can stand for are `true`

Existential Quantification: states that

- a predicate *P* holds for some objects in the universe,
- written as $\exists x\ \ P(x)$
- the sentence is `true` if and only if there is at least one `true` individual sentence where the variable *x* is replaced by the individual objects it can stand for

Usage of Universal Qualification

- universal quantification is frequently used to make statements like "All humans are mortal", "All cats are mammals", "All birds can fly", …
  - this can be expressed through sentences like

$$\forall x\ \ Human(x) \Rightarrow Mortal(x)$$

  - these sentences are equivalent to the explicit sentence about individuals

$$Human(John) \Rightarrow Mortal(John) \wedge$$
$$Human(Jane) \Rightarrow Mortal(Jane) \wedge$$
$$Human(Jill) \Rightarrow Mortal(Jill) \wedge \quad . \quad . \quad .$$

Usage of Existential Qualification

- existential quantification is used to make statements like
  - “Some humans are computer scientists”,
    “John has a sister who is a computer scientist”
    “Some birds can’t fly”, …
- this can be expressed through sentences like
  - $\exists$ x Human(x) $\land$ Computer-Scientist(x)
  - $\exists$ x Sister(x, John) $\land$ Computer-Scientist(x)
  - $\exists$ x TypeBird(x) $\land$ $\neg$Can-Fly(x)
- these sentences are equivalent to the explicit sentence about individuals
  - Human(John) $\land$ Computer-Scientist(John)
  - Sister(Jane, John) $\land$ Computer-Scientist(Jane)
  - TypeBird(Penguin) $\land$ $\neg$Can-Fly(Penguin)
    . . .

Connections between $\forall$ and $\exists$

- all statements made with one quantifier can be converted into equivalent statements with the other quantifier by using negation
  - $\forall$ is a conjunction over all objects under discourse
  - $\exists$ is a disjunction over all objects under discourse
  - De Morgan’s rules apply to quantified sentences

$$\forall x\ \neg P(x) \equiv \neg\exists\ x\ \ P(x) \qquad\qquad \neg\forall x\ P(x) \equiv \exists\ x\ \ \neg P(x)$$
$$\forall x\ P(x) \equiv \neg\exists\ x\ \neg P(x) \qquad\qquad \neg\forall x\ \neg P(x) \equiv \exists\ x\ \ P(x)$$

- strictly speaking, only one quantifier is necessary
  - using both is more convenient

Universal Quantification

$\forall x$, BelongsTo(x, Gryffindor) $\rightarrow$ $\neg$BelongsTo(x, Hufflepuff)

For all objects x, if x belongs to Gryffindor, then x does not belong to Hufflepuff.

Anyone in Gryffindor is not in Hufflepuff.

Existential Quantification

$\exists x$. House(x) $\wedge$ BelongsTo(Minerva, x)

There exists an object x such that x is a house and Minerva belongs to x.

Minerva belongs to a house.

$\forall x$. Person(x) $\rightarrow$ ($\exists y$. House(y) $\wedge$ BelongsTo(x, y))

For all objects x, if x is a person, then there exists an object y such that y is a house and x belongs to y.

Every person belongs to a house.

Multiple Quantifiers

more complex sentences can be formulated by using multiple variables and by nesting quantifiers

- the order of quantification is important
- variables must be introduced by quantifiers, and belong to the innermost quantifier that mention them

Examples

$\forall x, y \; Parent(x,y) \Rightarrow Child(y,x)$
$\forall x \; Human(x) \; \exists \, y \; Mother(y,x)$
$\forall x \; Human(x) \; \exists \, y \; Loves(x, y)$

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

FOL Syntax

- based on sentences
  - more complex than propositional logic
    - constants (propositional symbols), predicates, terms, quantifiers
    - includes propositional logic as a subset
- constant symbols – e.g. A, B, C, knife, study, …
  - stand for unique **objects** (in a specific context)
- relation symbols (a propositional function)
  Adjacent-To(,), Younger-Than (,) , ...
  - describes relations between objects, i.e. the relation is True or False
- function symbols – a short-hand for a type of relation using a function from objects to other objects (syntactic sugar)
  Father-Of, Square-Position, …
  - the given object is related to **exactly one other object**

BNF Grammar: Legal Sentences Predicate Logic

- Sentence → AtomicSentence | (Sentence Connective Sentence) | Quantifier Variable, ... Sentence | ¬Sentence
- AtomicSentence → Predicate(Term, …) | Term = Term
- Term → Function(Term, …) | Constant | Variable
- Connective → ∧ | ∨ | ⇒ | ⇔
- Quantifier → ∀ | ∃
- Constant → A, B, C, X1 , X2, Jim, Jack
- Variable → a, b, c, x1 , x2, counter, position
- Predicate → Adjacent-To, Younger-Than,
- Function → Father-Of, Square-Position, Sqrt, Cosine
- ambiguities are resolved through precedence or parentheses

Semantics

- relates sentences to models
  - in order to determine their truth values
- provided by interpretations for the basic constructs
  - usually suggested by meaningful names
  - Constants identify the object in the real world
  - predicate symbols particular relation in a model
    - may be explicitly defined through the set of tuples of objects that satisfy the relation, or implicitly via already established relations
  - function symbols: identify **the object** referred to by a single or tuple of objects
  - quantifiers: sentences about sets of objects
- interpretations for complex constructs
  - constructed from basic building blocks (“compositional semantics”)

Propositional vs FOL expressiveness

Propositional Symbols
MinervaGryffindor
MinervaHufflepuff
MinervaRavenclaw
MinervaSlytherin
…

Constant Symbol	Predicate Symbol
Minerva	
Pomona	Person
Horace	House
Gilderoy	BelongsTo
Gryffindor	
Hufflepuff	
Ravenclaw	
Slytherin	

Person(Minerva) → Minerva is a person.
House(Gryffindor) → Gryffindor is a house.
¬House(Minerva) → Minerva is not a house.
BelongsTo(Minerva, Gryffindor) → Minerva belongs to Gryffindor.

Validity and Satisfiability

- validity
  - a sentence is valid if it is true under all possible interpretations in all possible world states
    - independent of its intended or assigned meaning
    - independent of the state of affairs in the world under consideration
    - valid sentences are also called tautologies
- satisfiability
  - a sentence is satisfiable if there is some interpretation in some world state (a model) such that the sentence is true
- relationship between satisfiability and validity
  - a sentence is satisfiable iff (“if and only if”) its negation is not valid
  - a sentence is valid iff its negation is not satisfiable

Truth in first-order logic

- Sentences are true with respect to a model and an interpretation
- Model contains ≥ 1 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($term_1, \ldots, term_n$) is true iff the objects referred to by $term_1, \ldots, term_n$ are in the relation referred to by predicate

Example: Family Relationships

- **objects**: people
- **properties**: gender, height, weight, {cat, dog, …} person, …
  - expressed as n-ary predicates or functions
    - *Male(x), Female(y)*
    - *Height (x),*
- **relations**: parenthood, brotherhood, marriage
  - expressed through binary predicates *Parent(x,y), Brother(x,y), …*
- **functions**: motherhood, fatherhood
  - *Mother-of(x), Father-of(y)*
  - because every person has exactly one mother and one father
  - there may also be a relation *Mother(x,y), Father(x,y)*

Family Relationships

$\forall m,c\ Mother\text{-}of(c) = m$	$\Leftrightarrow$	$Female(m) \wedge Parent(m,c)$
$\forall w,h\ Husband(h,w)$	$\Leftrightarrow$	$Male(h) \wedge Spouse(h,w)$
$\forall x\ Male(x)$	$\Leftrightarrow$	$\neg Female(x)$
$\forall g,c\ Grandparent(g,c)$	$\Leftrightarrow$	$\exists\, p\ Parent(g,p) \wedge Parent(p,c)$
$\forall x,y\ Sibling(x,y)$	$\Leftrightarrow$	$\neg(x=y) \wedge \exists\, p\ Parent(p,x) \wedge Parent(p,y)$

. . .

Inference in FOL

- Reduction to Propositional Logic
  - Generates lots of unnecessary sentences and objects

- Unification: Generalized Modus Ponens
  - Makes only those substitutions necessary for a particular inference
  - This enables the use of forward and backward chaining
    - Algorithms used when implications are restricted to Horn clauses
    - Used in Knowledge Based System (e.g. Expert Systems)

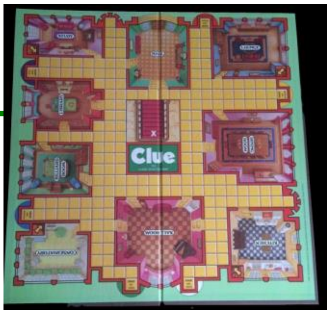
FOL summary

- Predicate logic extends Propositional Logic with new features:
  - Variables: *x*, *y*, *z*
  - Predicates: *P*(*x*), *M*(*x*)
- Relations: unary, binary, n-ary
  - Express relationships between different "objects" in the domain
- *Propositional functions* are a generalization of propositions.
  - They contain variables and a predicate, e.g., *P*(*x*)
  - Variables can be replaced by elements from their *domain*.
- Quantifiers
  - universal quantifier ∀ *For all, For any*
  - existential quantifier ∃ *There exists*
- Inference reduces to Propositional Logic Approaches

Knowledge Engineering

- Knowledge engineering is the process of developing a representation of a problem using a modeling language.
- Here our modeling language is Propositional Logic
- Many other languages can be used for knowledge representation and inference.
- For example, *First order predicate logic* which we will cover later.

Example: Game of Clue



Solve the murder: Who, Where, Weapon?

Col. Mustard	Ballroom	Knife
Prof. Plum	Kitchen	Revolver
Ms. Scarlet	Library	Wrench

Example: Game of Clue (continued)

- Cards for each person, room, weapon. Envelop hides the answer
- Information accumulates throughout the game.
  - Single cards tell you that person, room or weapon not in the answer
  - Incorrect guess implies at least one of the proposed components is not in the answer

People	Rooms	Weapons

CLUE: Propositional Symbols

mustard	ballroom	knife
plum	kitchen	revolver
scarlet	library	wrench

One person, location, weapon

(Mustard ∨ Plum ∨ Scarlet)

(knife ∨ revolver ∨ wrench)

(ballroom ∨ kitchen ∨ library)

player gets three cards, one of each type

¬(Mustard)

¬(kitchen)

¬(revolver)

CLUE: Propositional Symbols

mustard	ballroom	knife
plum	kitchen	revolver
scarlet	library	wrench

One person, location, weapon

(Mustard ∨ Plum ∨ Scarlet)

(knife ∨ revolver ∨ wrench)

(ballroom ∨ kitchen ∨ library)

player gets three cards, one of each type

¬(Mustard)

¬(kitchen)

¬(revolver)

CLUE game progression

- During the game, one can make a guess, suggesting one combination of person, weapon, and location. Suppose that the guess is that Scarlet used a wrench to commit the crime in the library. If this guess is wrong, then the following can be deduced and added to the KB:

(¬Scarlet ∨ ¬library ∨ ¬wrench)

- Now, suppose someone shows us the Plum card. Thus, we can add

¬(Plum) to our KB.

CLUE game progression

- The player now knows the murderer is Scarlet, since it is not Mustard or Plum.
- Adding just one more piece of knowledge, e.g.it is not the ballroom, gives us more information.
  Update our KB with ¬(ballroom)

- Claim: It can be proven that:
- Scarlet committed the murder with a knife in the library.

Expressing the knowledge base in Python

```
knowledge = And(          # Start with the game conditions
   Or(mustard, plum, scarlet),
   Or(ballroom, kitchen, library),
   Or(knife, revolver, wrench),
   Not(mustard),          # The three initial cards we saw
   Not(kitchen),
   Not(revolver),
                          # wrong guess  Scarlet, wrench, library
   Or(Not(scarlet), Not(library), Not(wrench)),
   Not(plum),      ⇒      #must be Scarlet   WHY?
   Not(ballroom)   ⇒      #library   Obvious WHY?
```

But can conclude more that **Scarlet, knife, library**

Knowledge Engineering in FOL

Knowledge engineering: the general process of knowledge-base construction.

The steps used in the knowledge engineering process:

1. Identify the questions.
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 problem instance
6. Pose queries to the inference procedure and get answers
7. Debug and evaluate the knowledge base