Brief Intro to First Order Logic

Laboratory for Applied Ontology, ISTC-CNR, www.loa-cnr.it

S. Borgo – 3 Oct. 2007

Overview

Motivations

- What you will be able to understand
- What you care to understand

Content

- Syntax
- Semantics
- Logical equivalence
- Proof theory (just a bit)
- Consistency, Soundness, Completeness

Motivations

What you will be able to understand

First Incompleteness Theorem (Gödel, 1931)

- Any consistent formal system F within which a certain amount of elementary arithmetic can be carried out is incomplete,

  that is

- there are statements of the language F which can neither be proved nor disproved in F.

  This mathematical result (together with the second Gödel's incompleteness theorem) is the most cited and celebrated theorem outside mathematics. You can find references to it in the writings of mathematicians, logicians, and philosophers as well as of physicists, theologians, literary critics, architects, cognitivists, and so no.

What is important to understand it?

- Any consistent formal system F within which a certain amount of elementary arithmetic can be carried out is incomplete
- Any consistent formal system F within which a certain amount of elementary arithmetic can be carried out is incomplete
- Any consistent formal system F within which a certain amount of elementary arithmetic can be carried out is incomplete

Formal system, Consistent, (In)Complete

Roughly, just to have a hint

- A formal system is a system of axioms (in some fixed language) equipped with rules of reasoning which allow one to generate (*derive / prove*) new statements (*theorems*).
- A formal system is consistent if there is no statement for which the statement itself and its negation are both generated (derivable) in the system.

  `Only consistent systems are interesting.`

- A formal system is complete if for every statement of the language either the statement or its negation can be derived in the system.

Motivations

What you care to understand

What you care to understand

We study formal systems as a tool to explicitly represent what we know (or care to know) of a problem, a situation, a state of the world.

Formal systems for knowledge representation have several advantages:

- the information can be processed automatically
- the consistency of the information is easier to test
- the knowledge is more reliably shared with other people and systems.

To prove vs To be true

There are two ways to use formal systems

- To reason on statements syntactically:
  essentially based on *deduction* (proof theory)
  (this is what we mentioned earlier)
- To reason on statements semantically:
  essentially based on *semantics* (model theory)
  (this is much more relevant to this class, we will see it later)

These alternatives are equivalent only in special cases.

Why formal systems?

Statements in natural language (e.g., English, Italian, Korean) are often ambiguous. They can be interpreted in different and incompatible ways: “Yesterday, I saw a doctor.”

To make sure that different people (or robots, informatic systems etc.) do not reach incompatible conclusions from the same statements, we need a precise and unambiguous language.

Logic provides a series of formal languages that have the property of being precise and unambiguous (provided we use them correctly and with some attention).

Content: First Order Logic (FOL)

Syntax

Syntax - 1

Symbols of first order logic (FOL) are divided in logical and non-logical

1) logical symbols

- ▷ symbols for propositional connectives:
  $\neg$ (not), $\wedge$ (and), $\vee$ (or), $\rightarrow$ (implies), $\leftrightarrow$ (if and only if)
- ▷ symbols for propositional constants: $\top$ and $\bot$
- ▷ symbol for identity: $=$ [it may be omitted]
- ▷ separation symbols (parentheses): (,)
- ▷ symbols for individual variables: $x_1, x_2, \ldots, y_1, y_2 \ldots$
- ▷ symbols for quantifiers: $\forall$ (for all), $\exists$ (exists)

Syntax - 2

2) non-logical symbols (parameters)

- ▷ predicate symbols (each associated to a positive integer (arity)): P, Q, R,...
- ▷ function symbols (each associated to a positive integer (arity)): f, g, h,...
- ▷ constant symbols (names): $a, b, c, \ldots$

A language of FOL is

an enumerable set of non-logical symbols.

Languages: examples 1 and 2

Predicate logic

- ▷ Symbol of identity: NO
- ▷ Predicate symbols (for each $n > 0$): $P^n_1, P^n_2, \ldots$
- ▷ Function symbols (for each $n > 0$): NO
- ▷ Constant symbols: $c_1, c_2, \ldots$

Set theory

- ▷ Symbol of identity: YES
- ▷ Predicate symbols: $\in$ (binary)
- ▷ Function symbols (for each $n > 0$): NO
- ▷ Constant symbols: NO

Languages: example 3

Elementary number theory

- ▷ Symbol of identity: YES
- ▷ Predicate symbols: $<$ (binary)
- ▷ Function symbols: s (unary, the successor function), $+$ (binary, the addition function) and $\times$ (binary, the product function)
- ▷ Constant symbols: 0

Examples for your intuition

A formal language is used to represent information (among other things).

(1) "Some cats are white"
(2) "All the chairs are broken"

(1) $\exists x\,(Cat(x) \wedge White(x))$

(2) $\forall x\,(Chair(x) \rightarrow Broken(x))$

Can you guess the meaning of the following?

(3) $\forall x\,(Cat(x) \rightarrow White(x))$

(4) $(\exists x\,Chair(x)) \rightarrow \exists y\,Broken(y)$

Terms

Given a language L, the set of terms of L is given by the following rules:

- ▷ Every symbol of constant and of variable is a term
- ▷ If $t_1, \ldots, t_n$ are terms and f is a function symbol of arity n, then $f(t_1, \ldots, t_n)$ is a term

Examples of terms: $x, c, x + c, f(x, y + c)$

Formulas - 1

Given a language L, a formula (*of* or *in* L) is any finite sequence of logical and/or non-logical symbols provided the non-logical symbols belong to L.

The formulas which "receive meaning" are called well formed formulas (wff)

We call atomic formulas those obtained by the following rules:

1. $\top$ and $\bot$ are atomic formulas
2. If t_1 and t_2 are terms then $t_1 = t_2$ is an atomic formula
3. If $t_1, \ldots, t_n$ are terms and P is a predicate symbol of arity n, then $P(t_1, \ldots, t_n)$ is an atomic formula.

Examples of atomic fomulas:
$c = d$, $P(x)$, $Q(x, c)$, $R(x, f(x, y + c))$

Formulas - 2

The set of formulas of L is the set obtained by the following rules:

(1) Every atomic formula is a formula

(2) If A is a formula, then $\neg A$ is a formula

(3) If $\circ$ is a binary connective, A and B formulas, then $A \circ B$ is a formula

(4) If A is a formula, x a variable, then $\forall x\, A$ and $\exists x\, A$ are formulas

Examples of formulas:
$Alive(x), \exists x\, Loves(x, Bush), \forall z\, R(x, f(x, y + c))$

More examples

- Is_Big(Home(John)) $\wedge$ Is_Bigger(Home(John),Home(Rob))
- $\forall x$ ($x =$ Author_Of(Divinacommedia) $\rightarrow$ Born_In(Firenze,x))
- $\forall x\ (2 + (x \times 0) = 2)$
- $\forall x\ (5 + (0 \times x) = x)$
- $\forall x \exists y$ *Loves(x, y)* — everybody loves somebody
- $\forall x \exists y$ *Loves(y, x)* — everybody is loved by somebody
- $\exists x \forall y$ *Loves(x, y)* — somebody loves everybody
- $\exists x \forall y$ *Loves(y, x)* — somebody is loved by everybody

Order of the connectives

To simplify the writing, we establish an order on the connectives (from left to right)

$$\forall, \exists, \neg, \wedge, \vee, \rightarrow, \leftrightarrow$$

Thus, formula $\forall x P(x) \rightarrow \exists y \exists z Q(y,z) \wedge \neg \forall x R(x)$
is equivalent to

$$(\forall x(P(x)) \rightarrow ((\exists y(\exists z Q(y,z))) \wedge (\neg(\forall x(R(x))))))$$

We posit that the binary connectives are associative (give precedence) to the right

$A \rightarrow B \rightarrow C$ is equivalent to $A \rightarrow (B \rightarrow C)$

Open and closed formulas

Quantifier: an expression of form $\forall x$ or $\exists x$.

Scope of a quantifier: the subformula to which it applies.
E.g. $P(x)$ is the scope of $\forall x$ in formula $P(y) \wedge Q(x) \wedge \forall x\, P(x)$.

All occurrences of a variable y outside the scope of quantifiers $\forall y$ and $\exists y$ are said to be *free*. The other are said *bound*.

A variable can have bound and free occurrences in the same formula.
E.g., the first occurrence of x and the occurrence of y are free in formula

$$Q(x) \wedge \forall x\, (P(x) \rightarrow Q(y))$$

Sentences

Examples of open and closed formulas. Can you tell which of these are open and which are closed formulas?

(1) $\exists x\, Loves(x, Bob)$

(2) $\forall x, y \exists z\, (z = exp(x + y) \wedge \Re(z))$

(3) $\forall y\, R(c_2, f(c_1, y + c_2)) \wedge \exists z\, P(z)$

(4) $\forall x \exists y\, (Q(x, y) \wedge x = f(y + c))$

(5) $P(x)$

(6) $\forall z\, (R(x, f(x, y + c)) \wedge P(z))$

(7) $\forall z\, R(x, f(x, y + c))$

(8) $(\forall x\, P(x)) \wedge Q(x)$

A *sentence* (or *closed formula*) is a formula without free (occurrences of) variables.

Very good!

We have finished with the syntax of first order logic.

Ready for the next step?

Content

Semantics

Interpretation - 1

So far we:

- constrained languages and grammatical rules.
- isolated some special sets like terms and formulas.
- described some simplifications obtained via ordering the connectives.
- distinguished open and closed formulas depending on the occurring variables.

Still we haven't said anything about when sentences are true or false. We have seen some examples and their "informal reading" in natural language.

Interpretations (or *models* or *structures*) provide the information to correctly and precisely understand formulas in these languages.

Interpretation - 2

An interpretation of a language specifies the following:

(1) A domain D (namely, a non-empty set).

(2) A denotation for each constant in the language, that is, an element of the domain that carries that constant as a name.

(3) A function from D^n to D for each function of arity n in the language.

(4) A set of n-tuples for each predicate of arity n in the language.

Obviously, the interpretation depends on the language since it has to associate each element in the language with its specific "meaning."

Note: sometimes "interpretation" is used to mean (2)-(4) only.

Interpretation: example 1

Assume we have the language to express formula

$$L(b, F(F(b)))$$

(Note: L must be a binary predicate and F a unary function for the formula to be well founded.)

Let us fix the interpretation:

(1) Domain D= {Ben, John, Tom}

(2) The denotation of b is Ben.

(3) The function from D^1 to D for F is F(Ben)=John, F(John)=Tom, and F(Tom)=Tom.

(4) A set of 2-tuples for predicate L is {(Ben,Ben),(Ben,Tom)}.

Can you tell if the formula is true or false in this interpretation?
(true)

Interpretation: example 2

Assume we have the language to express formula

$$\forall x[(\exists y\,(L(x,y) \wedge L(y,b))) \rightarrow \neg L(x,b)]$$

Let us fix the interpretation:

(1) Domain D= {Betty, John, Lucy}

(2) The denotation of b is Betty.

(3) -

(4) A set of 2-tuples for predicate L is {(Betty,Betty), (John,Betty), (Lucy,John)}.

Can you tell if the formula is true or false in this interpretation? (false)

Interpretation: multiple example - 3.1

Assume we have the language to express formula

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

Interpretation 1:

- D = set of human beings
- P is interpreted as the set of pairs (a, b) such that b is the natural father of a

True or False?

Meaning: *each human being has a father*

Interpretation: multiple example - 3.2

As before, take the language to express formula

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

Interpretation 2:

- D = set of human beings
- P is interpreted as the set of all pairs (a, b) where $a \neq b$

True or False?

Meaning: take $P(a, b)$ to mean b knows a
given any human being there is someone that knows him/her

Interpretation: multiple example - 3.3

As before, take the language to express formula

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

Interpretation 3:

- D = set of Italian regions as of year 2007
- P is interpreted as set of pairs (a, b) with regions a and b sharing a boundary

True or False?

Meaning:
every Italian region shares a boundary with another Italian region

The truth-values

We are able to tell if a sentence of a given language is true or false in a given interpretation. But it looks hard if the sentence is complex.

We can make the assignment of truth-values very easy by taking it as the result of a step-by-step process.

Given a sentence φ and an interpretation $\mathcal{I}$ given (domain D and interpretation of the language elements Int), i.e. $\mathcal{I} = \langle D, Int \rangle$, we want to define $\mathcal{I}(\varphi)$ such that

- $\mathcal{I}(\varphi)$= T if φ is true in interpretation $\mathcal{I}$
- $\mathcal{I}(\varphi)$= F if φ is false in interpretation $\mathcal{I}$

Notations

A *structure* is a pair of type $\langle D, Int \rangle$ (i.e., another way to write an interpretation). In terms of structures, the expression $\mathcal{I}(\varphi) = \mathsf{T}$ is written:

$$\langle D, Int \rangle \models \varphi$$

and is read "structure $\langle D, Int \rangle$ satisfies φ" or "structure $\langle D, Int \rangle$ is a model for φ"
Analogously, the notation $\mathcal{I}(\varphi) = \mathsf{F}$ is written:

$$\langle D, Int \rangle \not\models \varphi$$

and is read "structure $\langle D, Int \rangle$ does not satisfy φ" or "structure $\langle D, Int \rangle$ is not a model for φ"

Note: in this notation the denotation of a constant c, predicate P and function f are written c^{Int}, P^{Int}, and f^{Int} (resp.ly).

Sentence true in structure $\langle D, Int \rangle$

Let $\langle D, Int \rangle$ be a structure for the language L.

(1) $\langle D, Int \rangle \models \top$ and $\langle D, Int \rangle \not\models \bot$

(2) If A is an atomic formula of type $P(t_1, \ldots, t_n)$, then

$\langle D, Int \rangle \models P(t_1, \ldots, t_n)$ iff $t_1^{Int}, \ldots, t_n^{Int} \in P^{Int}$

(3) If A is an atomic formula of type $t_1 = t_2$ then

$\langle D, Int \rangle \models t_1 = t_2$ iff $t_1^{Int} = t_2^{Int}$

(4) $\langle D, Int \rangle \models \neg A$ iff $\langle D, Int \rangle \not\models A$

(5) $\langle D, Int \rangle \models A \wedge B$ iff $\langle D, Int \rangle \models A$ and $\langle D, Int \rangle \models B$

(6) $\langle D, Int \rangle \models A \vee B$ iff $\langle D, Int \rangle \models A$ or $\langle D, Int \rangle \models B$

Sentence true in interpretation $\mathcal{I}$

(7) $\langle D, Int \rangle \models (A \rightarrow B)$ iff $\langle D, Int \rangle \not\models A$ or $\langle D, Int \rangle \models B$

(8) $\langle D, Int \rangle \models (A \leftrightarrow B)$ iff both $\langle D, Int \rangle \models A$ and $\langle D, Int \rangle \models B$ or both $\langle D, Int \rangle \not\models A$ and $\langle D, Int \rangle \not\models B$

In the remaining cases, we write $A_{\{d=x\}}$ to indicate that we extend the interpretation to variable x (in a sense, treating it as a constant) by stating that the denotation of x is element d of the domain.

(9) $\langle D, Int \rangle \models \forall x A$ iff **for all** $d \in D$ we have $\langle D, Int \rangle \models A_{\{x=d\}}$

(10) $\langle D, Int \rangle \models \exists x A$ iff **there exists** a $d \in D$ for which $\langle D, Int \rangle \models A_{\{x=d\}}$

(Note: there are some further constraints in these last two cases that we ignore here.)

Example 1

$$\langle D, Int \rangle \models \exists x(P(x) \wedge Q(x))$$

- First, apply case (10)
  we need to verify that there exists $d \in D$ such that
  $\langle D, Int \rangle \models (P(x) \wedge Q(x))_{\{x=d\}}$
  i.e. $\langle D, Int \rangle \models P(x)_{\{x=d\}} \wedge Q(x)_{\{x=d\}}$
- Second, apply case (5)
  we need to verify $\langle D, Int \rangle \models P(x)_{\{x=d\}}$ and $\langle D, Int \rangle \models Q(x)_{\{x=d\}}$
- Finally, because of case (2), it suffices to check if $d \in P^{Int}$ and $d \in Q^{Int}$
- Go back with this result to assign value to the formula

Example 2

$$\langle D, Int \rangle \models \forall x(P(x) \rightarrow Q(x))$$

- First, apply case (9)
  verify that for all $d \in D$ we have $\langle D, Int \rangle \models (P(x) \wedge Q(x))_{\{x=d\}}$
  i.e. $\langle D, Int \rangle \models P(x)_{\{x=d\}} \wedge Q(x)_{\{x=d\}}$
- Second, apply case (7)
  we need to verify $\langle D, Int \rangle \not\models P(x)_{\{x=d\}}$ or $\langle D, Int \rangle \models Q(x)_{\{x=d\}}$
- Finally, because of case (2), it suffices to check if we have $d \notin P^{Int}$ or $d \in Q^{Int}$
- Store this result and go to the first step to assign a different element to x. When you are have checked all the elements, you get the value of the formula from case (9).

(Note that this formula is true for each element whenever P^{Int} is the empty set.)

Validity

A formula A of the language L is *valid* if and only if it is true in all the interpretations (structures) of L. In this cse, we write

$$\models A$$

to indicate that A is true no matter what structure we put on the left of the $\models$ sign.

A set of formulas in a language L is said to be *satisfiable* if and only if there exists an interpretation (structure) $\langle D, Int \rangle$ of L such that $\langle D, Int \rangle \models A$ for each formula A in Γ.

Content

Logical equivalence and related notions

Logical equivalence and logical consequence

Fix a language L and let Γ be a set of closed formulas and A a closed formula in L.

A is a *logical consequence* of Γ, written $\Gamma \models A$, if and only if for each structure $\langle D, Int \rangle$ of L, if $\langle D, Int \rangle \models B$ for all $B \in \Gamma$, then $\langle D, Int \rangle \models A$.

Two formulas A and B of the language L are *logically (or semantically) equivalent*, written $A \equiv B$, if and only if for each structure $\langle D, Int \rangle$ of L we have

$$\langle D, Int \rangle \models A \text{ iff } \langle D, Int \rangle \models B$$

Examples of equivalent formulas

Formulas that differ only because of the following issues are equivalent

- bound variables
  $\forall x P(x) \equiv \forall y P(y)$
- the order of similar quantifiers
  $\forall x \forall y P(x, y) \equiv \forall y \forall x P(x, y)$
  (indeed, we can write $\forall x, y P(x, y)$)
- the elimination of quantifiers that do not bound any variable
  $\forall x P(y) \equiv P(y)$

Examples with quantifiers and negation - 1

Important equivalences

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

Examples with quantifiers and negation - 2

Other important equivalences

- $\forall x(P(x) \wedge Q(x)) \equiv \forall x P(x) \wedge \forall x Q(x)$
- $\exists x(P(x) \vee Q(x)) \equiv \exists x P(x) \vee \exists x Q(x)$
- $\forall x(P(x) \vee Q) \equiv (\forall x P(x)) \vee Q$ if x not free in Q
- $\exists x(P(x) \wedge Q) \equiv (\exists x P(x)) \wedge Q$ if x not free in Q

Ok, we are almost done!

Content

Proof Theory

The role of Proof Theory

The goal is to derive new sentences from others via the systematic application of a well established set of rules. You can think of a rule as a way to manipulate a formula (or a set of formulas) to obtain another formula so that, if the first is (are) accepted, the latter must be accepted as well.

The rules are motivated by our intuition on what the connectives represent in the logic. They are not motivated or justified by semantic considerations.

There are several types of rules for first order logic (Hilbert style, Natural Deduction, Sequent calculus) and each type can provide different sets of rules.

Examples of rules

Beside special rules like *modus ponens*, in general there are two rules for each connective and quantifier. Given a connective (or quantifier), one rule is for obtaining a formula with the connective (quantifier) from other simpler formulas (connective-introduction), one to obtaining a formula without the connective (quantifier) from a formula with the connective (connective-elimination).
For example,

$$\frac{A \rightarrow B \quad A}{B} \text{(modus ponens)}$$

$$\frac{A \quad B}{A \wedge B} \ (\wedge\text{-introduction}) \qquad \frac{A \wedge B}{A} \ (\wedge\text{-elimination})$$

Example of deduction

We prove the formula

$$A \wedge \exists x P(x) \rightarrow \exists x(A \wedge P(x))$$

The deduction does not uses any information external to the logic itself. The goal is to show that formula $\exists x(A \wedge P(x))$ is obtained from $A \wedge \exists x P(x)$ by a correct application of the rules of deduction.

$$\dfrac{\dfrac{\dfrac{A \wedge \exists x P(x)}{A} \qquad \dfrac{\dfrac{A \wedge \exists x P(x)}{\exists x P(x)}}{P(a)}}{A \wedge P(a)}}{\exists x(A \wedge P(x))}$$

Consistency

The relationship between semantics and proof theory is crucial in many aspects.

A set of formulas (theory or formal system) is *syntactically consistent* if it is not possible to derive both a formula and its negation, i.e., both A and $\neg A$.
(This is the notion we introduced at the very beginning)

A set of formulas (theory or formal system) is *semantically consistent* if it has a model.

Soundness and Completeness

Given a semantically consistent set of formulas Γ. A set of deduction rules is sound if each formula that can be derived from Γ using the rules is satisfies by any model of Γ.

A semantically consistent set of formulas Γ together with a set of rules is said to be complete if both

(1) each formula true in all the models of Γ can be proved from Γ using the rules in the set and

(2) each formula derived via the rules from formulas in Γ is true in all the models of Γ.