In this section we present the basics of classical first order logic. The treatment is similar to that of standard mathematical logic texts, but limited to properties that are directly relevant to our applications.

1 Formulas of First Order Logic

In any application of first order logic, we start by choosing a vocabulary of symbols. A vocabulary $L$ is a tuple of symbols consisting of:

1. A (possibly empty) set of function symbols, each with an associated arity. We typically use $f, g$, etc., for function symbols, as well as symbols such as $+ \text{ and } -$. 0-ary function symbols are constant symbols, for which we typically use $c, 0, 1$, etc.

2. A (non-empty) set of predicate symbols, each with an associated arity. We typically use $P, Q, R$, etc., for predicate symbols, as well as standard symbols such as $=, \leq$, and words in camel font.

Example 1.

1. The vocabulary of graphs is $L_G = [E, =]$, where $E$ (for the edge relation) and $=$ are both binary predicate symbols.

2. The standard vocabulary for arithmetic, $L_A = [0, s, +, \cdot ; =]$, has the constant symbol $0$, the unary function symbol $s$ (for “successor”), the binary function symbols $+$ and $\cdot$, and the binary relation symbol $=.$

The formulas of first order logic for a given vocabulary $L$ (or $L$-formulas) are over an alphabet consisting of the symbols in $L$, a countably infinite set $x_1, x_2, \ldots$ of variable
symbols, and the symbols $\land, \lor, \neg, \forall, \exists, (, )$. We often use $x, y, z$ etc. for variables.

We first define the terms (or expressions), which are strings denoting objects in the universe of discourse.

**Definition 1.** The terms for a vocabulary $L$ (or $L$-terms) are defined inductively by the following rules.

1. Each variable symbol $x_i$ is an $L$-term;
2. If $f$ is a $k$-ary function symbol of $L$ and $t_1, \ldots, t_k$ are $L$-terms, then $f t_1 \ldots t_k$ is an $L$-term.

**Example 2.**

1. If $L$ contains unary function symbol $f$ and binary function symbol $g$, then the following are $L$-terms: $fx$, $gxy$, $gyx$, $gfx$ and $ggfxgxyz$.

2. Terms of the vocabulary $L_A = [0, s, +, \cdot; =]$ (terms of the language of arithmetic), include $0$, $s0$, $+ss0$, and $\cdot ss0ss0$. In the standard interpretation of the symbols, these terms denote the numbers 0, 1, 2, and 4, respectively.

When it improves readability, will often use infix notation for binary functions, for example writing $t_1 + t_2$ rather than $+t_1t_2$.

No punctuation is required to ensure terms to have a unique parsing, but we may sometimes use the notation $f (x)$ and $g(x, y)$ if it helps human readability. For example $g gfxgxyz$ is arguably easier to understand if we write it $f (g(f(x), f(g(y, z))))$.

**Definition 2.** The first order formulas for vocabulary $L$ (or $L$-formulas) are defined inductively by the following rules.

1. If $P$ is a $k$-ary predicate symbol of $L$, and $t_1, \ldots, t_k$ are $L$-terms, then $Pt_1 \ldots t_k$ is an $L$-formula. We call $Pt_1 \ldots t_k$ atomic, because no sub-string of it is a formula.
2. If $A$ and $B$ are $L$-formulas, then $\neg A$, $(A \land B)$, and $(A \lor B)$ are $L$-formulas.
3. If $A$ is an $L$-formula, and $x$ a variable symbol, then $\forall x A$ and $\exists x A$ are $L$-formulas.

Connectives $\land, \lor$ and $\neg$ are read “and”, “or” and “not”, as usual. $\forall$ and $\exists$ are, respectively, the universal and existential quantifiers. $\forall x A$ is read “for every $x$ $A$”, and $\exists x A$ as “for some $x$ $A$”. 

2
We have the usual abbreviations: \((A \rightarrow B)\) means \((\neg A \lor B)\), and \((A \leftrightarrow B)\) means \(((A \rightarrow B) \land (B \rightarrow A))\). We also will use several more, for example \(t_1 = t_2\) means \(t_1 t_2\) and \(t_1 \neq t_2\) means \(\neg (t_1 = t_2)\).

**Example 3.**

1. Let \(L = [f, g; P, Q]\), where \(f\) is a unary function symbol, \(g\) is a binary function symbol, \(P\) is a unary predicate symbol, and \(Q\) is a binary predicate symbol. Then, the \(L\)-formulas include \(P x, Q x y, P f x, (P f x \land Q f x y x), \text{ and } \forall x (P x \rightarrow \exists y Q x y)\).

2. Formulas of \(L_G\), the vocabulary of graphs, include \(E x y, E x x, \forall u \forall v E u v\), and \(\forall x \exists y E x y\).

3. Formulas of \(L_A\) include \(0 = 0, s 0 \neq 0,\) and \(\forall x \forall y (x + y) = (y + x)\).

A sub-formula of a \(A\) is any string which is a sub-string of \(A\) and is a formula.

An occurrence of variable \(x\) in a formula \(A\) is called bound if it is in a sub-formula of \(A\) (not necessarily proper) of the form \(\forall x B\) or \(\exists x B\), and otherwise is free. A formula with no free variables is called a sentence.

### 2 Semantics of First Order Logic

The semantics of propositional logic are defined by the satisfaction relation, \(|=\), between truth assignments and formulas. First order logic is much more expressive, and defining its semantics requires a more complex kind of object than truth assignments. The family of objects in question are called (mathematical) structures, and are simply defined but very general. A structure \(M\) is simply a set \(M\) (the universe or domain of discourse), together with a collection of functions and relations over \(M\). Simple structures which might be familiar include graphs and algebraic objects such as groups, rings, and fields. The role of a structure in logical semantics is to assign meaning to each term and each predicate symbol occurring in a formula.

**Definition 3.** A structure \(M\) for vocabulary \(L\) (or \(L\)-structure) is a tuple consisting of:

1. A non-empty set, \(M\), called the universe (or domain) of \(M\);
2. For each \(k\)-ary function symbol \(f\) of \(L\), a \(k\)-ary function \(f^M : M^k \rightarrow M\);
3. For each \(k\)-ary predicate symbol \(P\) of \(L\), a \(k\)-ary relation \(P^M \subseteq M^k\).
A structure \( \mathcal{M} \) for \( \mathcal{L} \) is sometimes called an interpretation for \( \mathcal{L} \), and for vocabulary symbol \( S \) of \( \mathcal{L} \), \( S^\mathcal{M} \) may be called the interpretation of \( S \) in \( \mathcal{M} \).

If \( = \) is included in a vocabulary, it always denotes true equality. All other predicate symbols may be given arbitrary interpretations. For example, if \( < \) is a binary predicate symbol, then \( <^\mathcal{M} \) can be any binary relation, and need not be an order. (In practice, of course, we normally try to use mnemonically appropriate symbols.)

**Example 4.**

1. The standard structure for \( \mathcal{L}_A \) is the structure \( \mathcal{N} \) of the natural numbers with:
   - \( N = \mathbb{N} = \{0, 1, 2, \ldots\} \);
   - \( 0^\mathcal{N} = 0 \);
   - \( s^\mathcal{N} \) is the successor function, i.e., the function that adds one;
   - \( +^\mathcal{N} \) and \( \cdot^\mathcal{N} \) are the standard operations of addition and multiplication;
   - \( =^\mathcal{N} \) is normal equality.

2. Any graph \( G \) is a structure for the vocabulary of graphs, with \( G = \langle V, E^G \rangle \).

The size of structure \( \mathcal{M} \) is the size of \( M \), and we call \( \mathcal{M} \) finite if \( M \) is finite.

We wish to define the satisfaction relation between \( \mathcal{L} \)-structures and \( \mathcal{L} \)-formulas. An effect of this definition will be that, for every \( \mathcal{L} \)-structure \( \mathcal{M} \), every sentence for \( \mathcal{L} \) will be either true or false in \( \mathcal{M} \).

If a formula \( A \) has free variables, then we will not be able to say whether it is true or not, without knowing which objects those variables are to denote. For example, we do not know if \( x + y < 5 \) unless we know which numbers \( x \) and \( y \) denote.

An object assignment (sometimes scaled a valuation) for structure \( \mathcal{M} \) is a function (here denoted by \( \sigma \)) mapping each variable \( x_i \) to an element of the universe \( M \).

Now, we are in a position to define the semantics of terms, and then of formulas.

**Definition 4.** The meaning, or denotation, of term \( t \) in structure \( \mathcal{M} \) with valuation \( \sigma \), written \( t^\mathcal{M}[\sigma] \), is defined recursively by:

1. If \( t \) is a variable \( x \), then \( t^\mathcal{M}[\sigma] = \sigma(x) \);
2. If \( t \) is a term of the form \( ft_1 \ldots t_k \), then \( t^\mathcal{M}[\sigma] = f^\mathcal{M}(t_1^\mathcal{M}[\sigma], \ldots t_k^\mathcal{M}[\sigma]) \).
Although notationally a bit heavy, this says the obvious: to figure out what the value \( f(t_1 \ldots t_k) \) denotes (according to \( M \) and \( \sigma \)), first figure out what values the arguments \( t_1, \ldots t_k \) denote (according to \( M \) and \( \sigma \)), figure out what function \( f \) denotes (according to \( M \)), and then apply the function to the values of arguments.

**Example 5.** \( s^0 \mathcal{N} = s^0(s^N(0^\mathcal{N})) = s^N(0) = s^N(1) = 2 \).

**Notation:** If \( \sigma \) is an object assignment, then \( \sigma(x/a) \) denotes the object assignment such that \( \sigma(a/x)(x) = a \) and \( \sigma(a/x)(y) = \sigma(y) \) for every variable \( y \) other than \( x \).

**Definition 5.** \( \mathcal{L} \)-structure \( M \) satisfies \( \mathcal{L} \)-formula \( A \) with object assignment \( \sigma \), written \( M \models A[\sigma] \), according to the following induction in the structure of \( A \).

1. \( M \models Pt_1 \ldots t_k [\sigma] \) iff \( \langle t_1^M[\sigma], \ldots t_k^M[\sigma] \rangle \in P^M \)
2. \( M \models t_1 = t_2 [\sigma] \) iff \( t_1^M[\sigma] = t_2^M[\sigma] \)
3. \( M \models \neg B [\sigma] \) iff \( M \not\models B [\sigma] \)
4. \( M \models (B \lor C) [\sigma] \) iff \( M \models B [\sigma] \) or \( M \models C [\sigma] \);
5. \( M \models (B \land C) [\sigma] \) iff \( M \models B [\sigma] \) and \( M \models C [\sigma] \);
6. \( M \models \forall x B [\sigma] \) iff for every \( a \in M \), \( M \models B [\sigma(a/x)] \)
7. \( M \models \exists x B [\sigma] \) iff for some \( a \in M \), \( M \models B [\sigma(a/x)] \)

Notice that item 2 in Definition 5 follows from item 1 and the requirement that \( = \) always denotes equality.

**Example 6.**

1. \( \mathcal{N} \models (x + y) = sssss0 [\sigma] \) iff \( \sigma \) is such that \( \sigma(x) + \sigma(y) = 5 \).
2. \( \mathcal{N} \not\models \forall x \exists y (y = ssx) \), regardless what \( \sigma \) is, because the formula has no free variables, and for every natural number there is another that is larger by two.
3. \( \mathcal{N} \not\models \forall x \exists y (ssy = x) \), regardless what \( \sigma \) is, because the formula has no free variables, and there is no natural number two smaller than 1 or 0.
4. Let \( \mathcal{M} \) be an \( \mathcal{L}_A \)-structure with \( M = \{0, 1, 2\} \), \( 0^M = 0 \), \( s^M(x) = x + 1 \mod 3 \), and \( +^M \) be addition modulo 3. Then \( \mathcal{M} \models (x + y) = sssss0 [\sigma] \) iff \( \sigma \) is such that \( \sigma(x) + \sigma(y) = 2 \mod 3 \).
5. Let $\mathcal{M}$ be an $L_G$-structure with $M = \{a, b, c\}$ and $E^M = \{(a, b), (b, c)\}$. Then $\mathcal{M} \models Euv$ if $\sigma(u) = a$ and $\sigma(v) = b$, but $\mathcal{M} \not\models Euv$ if $\sigma(u) = a$ and $\sigma(v) = c$.

6. Let $\mathcal{G}$ be a graph. Then $\mathcal{G} \models \forall u \forall v Euv$ if and only if $\mathcal{G}$ is a complete graph.

If $A$ is a sentence, then $\sigma$ plays no role in the truth of $A$: For each structure $\mathcal{M}$ (for the same vocabulary as $A$), either $\mathcal{M} \models A[\sigma]$ for every $\sigma$ or for no $\sigma$. Thus, we may leave out $\sigma$ and simply write $\mathcal{M} \models A$, or $\mathcal{M} \not\models A$. In the case that $\mathcal{M} \models A$, we say that “$A$ is true in $\mathcal{M}$”, or that “$\mathcal{M}$ is a model for $A$” or “$\mathcal{M}$ satisfies $A$”.

**Definition 6.** The problem of model checking for first order logic (FO) on a class $C$ of finite structures is:

- Given: A (string encoding a) structure $\mathcal{M}$ from $C$ and a FO sentence $A$;
- Question: Does $\mathcal{M} \models A$?

**Fact 1.**

1. If $C$ is all finite structures, FO model checking is PSPACE complete.
2. If $C$ is all finite structures, FO model checking is in time $O(n^k)$, where $n$ is the size of the structure, and $k$ the size of the sentence.
3. The problem “given $A$, does $\mathcal{N} \models A$” is undecidable.

To see that item 2 is true, observe that a recursive algorithm directly implementing the recursion of Definitions 5 and 4 runs in the claimed time.

### 3 Defining a Relation in a Structure

Consider our language $L_A$, and the standard $L_A$ structure $\mathcal{N}$. Every $L_A$ sentence is either true for false in $\mathcal{N}$. As we observed earlier, if a formula has free variables, its truth in a structure depends on what objects those free variables denote. For example, $\mathcal{N} \models x = sy [\sigma]$ iff sigma is such that $\sigma(x) = \sigma(y) + 1$. A bit more interesting is the formula $\exists y(x = y + y)$, which is true in $\mathcal{N}$ with $\sigma$ iff $\sigma(x)$ is an even number. We say that $\exists y(x = y + y)$ defines the set of even numbers in $\mathcal{N}$.

**Notation:** We may write $A(x_1, \ldots, x_n)$ to indicate that the free variables of $A$ are among $x_1, \ldots, x_n$. So, if we let Even be the formula $\exists y(x = y + y)$, we write that Even$(x)$ defines the even numbers in $\mathcal{N}$. We sometimes also use tuple notation, for example $A(\bar{x})$, to indicate that $\bar{x}$ is a tuple of variables which includes all the free variables of $A$. 
Definition 7. The relation defined by formula $A(x_1, \ldots, x_n)$ in structure $M$ is

$$\{\langle a_1, \ldots, a_n \rangle \mid M \models A(x_1, \ldots, x_n) [\sigma(a_1/x_1) \ldots (a_n/x_1)]\}.$$  

In Definition 7, $\sigma$ does not matter, because we have explicitly enumerated the assignments to the variables that matter.

Example 7.

1. $x + y = sssss0$ defines, in $\mathcal{N}$, the set of pairs of numbers which sum to 5.
2. If $G = \langle V, E \rangle$ is a graph, then $Euv \wedge Evw \wedge Ewu \wedge u \neq v \wedge v \neq w \wedge w \neq u$ defines the set of triangles in $G$.

Exercise 1.

1. Write a formula $\text{Odd}(x)$ that defines the odd natural numbers in $\mathcal{N}$.
2. Write a formula $\text{Source}(x)$ that defines the set of vertices with in-degree zero, in any directed graph $G = \langle V, E \rangle$.

Exercise 2. Suppose $\text{Prime}(x)$ is a formula that defines the prime numbers in $\mathcal{N}$, $\text{Even}(x)$ is as described previously, and $x < y$ — which is an abbreviation for $(x, y)$ — defines the standard ordering on the naturals. Then Goldbach’s conjecture, that every even integer greater than 2 is the sum of two primes, is expressed by

$$\forall x((\text{Even}(x) \wedge x > 2) \rightarrow \exists y \exists z(\text{Prime}(y) \wedge \text{Prime}(z) \wedge x = y + z)).$$

Write formulas for $\text{Prime}(x)$ and $x < y$.

4 Satisfiability, Validity, Equivalence and Logical Consequence

Definition 8.

1. Formula $A$ is satisfiable iff $M \models A[\sigma]$, for some $M$ and $\sigma$;
2. Set $\Phi$ of formulas is satisfiable iff there is some $M$ and $\sigma$, such that $M \models A[\sigma]$ for every $A \in \Phi$;
3. $A$ is a logical consequence of $\Phi$ (written $\Phi \models A$) iff for every $M$ and $\sigma$, $M \models A[\sigma]$;
4. $A$ is valid (written $\models A$), iff $M \models A[\sigma]$ for every $M$ and $\sigma$;
5. A and B are logically equivalent (written $A \equiv B$) iff, for all $\mathcal{M}$ and $\sigma$, $\mathcal{M} \models A[\sigma]$ iff $\mathcal{M} \models B[\sigma]$.

Example 8.

1. Does $(\forallxA \lor \forallxB) \models A \lor B$ hold for every two formulas A,B? Yes. Suppose $\mathcal{M}$ and $\sigma$ are such that $\mathcal{M} \models (\forallxA \lor \forallxB)[\sigma]$. Then, by Definition 5, either $\mathcal{M} \models \forallxA[\sigma]$ or $\mathcal{M} \models \forallxB[\sigma]$. Suppose the first case. Then, for every $a \in M$, $\mathcal{M} \models A[\sigma(a/x)]$, so, for every $a \in M$, $\mathcal{M} \models (A \lor B)[\sigma(a/x)]$. Therefore $\mathcal{M} \models \forallx(A \lor B)[\sigma]$. The symmetric argument applies if $\mathcal{M} \not\models \forallxA[\sigma]$, but $\mathcal{M} \models \forallxB[\sigma]$.

2. Does $\forallx(A \lor B) \models (\forallxA \lor \forallxB)$ hold for every two formulas A,B? No. Let A be $Px$, and B be $Qx$, and let $\mathcal{M}$ be the structure with $M = \mathbb{N}$, $P^M$ the set of even natural numbers, and $Q^M$ the set of odd natural numbers. Then $\mathcal{M} \models \forallx(A \lor B)$, because every natural number is either even or odd, but $\mathcal{M} \not\models (\forallxA \lor \forallxB)$, because some natural numbers are not even, and some are not odd.

Exercise 3. Verify each of the following.

1. For every formula A, $\neg\forallxA \equiv \existsx\neg A$.
2. For every formula A, $\neg\existsxA \equiv \forallx\neg A$.
3. For every two formulas A, B, $(\forallxA \land \forallxB) \equiv \forallx(A \land B)$.
4. For every two formulas A, B, $(\existsxA \lor \existsxB) \equiv \existsx(A \lor B)$.
5. For every two formulas A, B, $\existsx(A \land B) \models (\existsxA \land \existsxB)$.
6. There are formulas A, B for which $(\existsxA \land \existsxB) \models \existsx(A \land B)$.
7. For every formula A, $\forallx\forallyA \equiv \forally\forallxA$.
8. For every formula A, $\existsx\forallyA \models \forally\existsxA$.
9. There are formulas A for which $\forallx\existsyA \not\models \existsy\forallxA$.
10. For every formula A, $\forallxA \models \existsxA$.

Fact 2.

1. The problem “given a FO formula A, is A satisfiable?”, is undecidable.
2. The problem “given a FO formula A, does A have a finite model?”, is undecidable.
3. The problem “given two FO formulas A, B, does $\mathcal{M} \models B$”, is undecidable.
4. The problem “given a FO formula A, is A valid”, is undecidable.