Crash Course in the Theory of Logic

Richmond H. Thomason
Email: rich@thomason.org
Web: http://www.eecs.umich.edu/~rthomaso/

Version 2: 1-05-1998

1. Language

First-Order Logic (FOL) is a framework within which first order theories can be developed. The framework defines various syntactic types (i.e., types of expressions,) and provides rules that specify how simpler expressions can be combined to make more complex ones. The following syntactic types of expressions are available.

1. Terms
2. Function expressions
3. Predicate expressions
4. Connectives
5. Formulas
6. Quantifiers

Basic terms include constants (specifically, individual constants) and variables (individual variables).[1] If functions are available, there may be complex terms as well as basic terms; if a is an individual constant and f and g are one-place function symbols, for instance,

$$f(g(a))$$

will be a term.

Function expressions are typed according to the number of arguments that they take. (I.e., FOL, like many programming languages, is not polymorphic.) An n-place function expression combines with n terms to make a term: so, if f is a 2-place function expression, x is a term, and $f(a)$ is a term, then

$$f(x, f(a))$$

is also a term. The usual versions of FOL do not provide for complex function expressions. Logicians usually count beginning with 0; 0-place function expressions are allowed, and in fact these are the same as individual constants.

Predicate expressions combine with terms to make formulas. Like function expressions, predicate expressions are typed according to the number of arguments that they take. An

[1] The difference between FOL and higher-order logics is that in FOL, only individual variables are available; variables that are stipulated to stand for functions from individuals to individuals, for instance, are not allowed in FOL.

n-place predicate expression combines with n terms to make a formula: so, if P is a 2-place predicate expression, x is a variable, and $f(a)$ is a term,

$$P(x, f(a))$$

will be a formula. Note that a 0-place basic predicate expression would be a basic formula. The usual versions of FOL do not provide for complex predicate expressions.

Connectives combine formulas into more complex formulas. In general, the following connectives are available in FOL (either as primitives or by definition):

1. $\neg$ (negation, a 1-place connective)
2. $\wedge$ (conjunction, a 2-place connective)
3. $\vee$ (disjunction, a 2-place connective)
4. $\rightarrow$ (the conditional, a 2-place connective)
5. $\leftrightarrow$ (the biconditional, a 2-place connective)

Thus, for instance, if A, B, and C are formulas then so is $\neg[\neg A \wedge [B \vee \neg C]]$. *It is a crucial part of the design of the language of FOL that formulas are not to be ambiguous.* Therefore, brackets are used for disambiguation in the application of connectives. An expression like

$$A \vee B \wedge C$$

could not be a formula of FOL. But (depending on the bracketing conventions, which differ, but which always render formulas unambiguous), disambiguated combinations like

$$A \vee [B \wedge C]$$

or

$$[A \vee B] \wedge C$$

would be admissible.

The usual versions of FOL do not provide for connectives that are not basic. This does not mean, however, that connectives can't be defined; a defined connective is usually treated as a metalinguistic abbreviation of (i.e., as a macro for) a complex of primitives. That is, logicians like to treat definitions not as primitives of the object language which can somehow be eliminated, but as abbreviations in the metalanguage. On this policy, if $A \rightarrow B$ is defined as $\neg A \vee B$, a formula like '$P(a) \rightarrow Q(a)$' does not appear in the logical language at all, but is merely another way that we use to describe the formula $\neg P(a) \vee Q(a)$. This policy toward definitions should be familiar to computer scientists, since it amounts to treating the extended language with definitions as a "higher level" language that is compiled into the object language by macro expansion. Often, $\neg$ and $\vee$ are taken as primitive connectives; it is a theorem that any boolean function can be defined in terms of these two, so this set is expressively complete.

FOL usually has a very limited set of basic quantifiers; usually either the *universal quantifier* $\forall$ or the *existential quantifier* $\exists$ is taken to be primitive, and the other is then defined. Quantifiers combine with variables and formulas to make formulas: you can universally quantify a formula like $P(x, y)$, for instance, in two interesting ways, obtaining either $\forall x P(x, y)$ or $\forall y P(x, y)$.

The usual versions of FOL do not provide for quantifiers that are not basic. This does not mean, however, that quantifiers can't be defined. Though identity can be regarded as a two-place predicate expression of FOL, it is often singled out as a special expression with distinctive logical properties. When identity is present, a large number of "numerical" quantifiers can be defined, like "for at least three," "for exactly two," and "for either at most three or at least ten." For instance, $(\exists_{\geq 2} x)A$ is defined as $\exists y \exists z[A(y/x) \wedge A(z/x) \wedge \neg y = z]$, where y and z are different variables not occurring in A. And $(\exists_1 xA)$ is defined as $(\exists y)(\forall x)[A \leftrightarrow x = y]$.

2. A bit about variables and substitution

In a formula $\forall xA$ or $\exists xA$, A and everything in it is the *scope* of the quantifier $\forall x$ or $\exists x$. Thus, in $P(x) \wedge \exists xQ(x)$, the first occurrence of x is in the not scope of a quantifier, but the second occurrence of x is. A variable x that is in the scope of a quantifier $\forall x$ or $\exists x$ is *bound* by that quantifier; variables that are not bound are free. Thus, the first occurrence of x in $P(x) \wedge \exists xQ(x)$ is free and the second occurrence of x is bound. A formula with no free occurrences of variables is called a *sentence.*

Where A is a formula (which may or may not have free occurrences of u), $A(t/u)$ is the result of substituting occurrences of t for all free occurrences of u in A. Since substitutions that result in bound occurrences of variables that were not bound in the original formula are anomalous, this notation presupposes that no such substitutions take place. One way to make sure that the substitution $A(t/u)$ is legitimate is to assume that t contains no occurrences of variables that are arguments of quantifiers in A.

3. Choosing a first-order theory

In an application of FOL, a specific language is made up out of the above materials. Suppose, for instance, that you have an application in which there are five things that you want to talk about. Then you might choose a language with five individual constants and no more. And if there are three interesting types of objects and one two-place relation that is interesting in the domain, you will choose three 1-place predicate constants (basic predicate expressions) and one 2-place predicate constants. This, then, will yield your first-order theory. The connectives and quantifiers are always available (and identity is usually available). And it is always assumed that there are infinitely many variables. (For logical purposes, it is necessary to assume that for any formula A you can find a variable that doesn't occur in A.)

4. Macrosemantics

Here is a large-scale picture of semantics, or model theory, that leaves out the details.

Corresponding to each syntactic type, there is a semantic type. Terms correspond to individuals. (That is, each term refers to a member of the semantic type of individuals.) Formulas correspond to truth values. (These values can be any two fixed objects; usually, 0 and 1 are used for this purpose.) An n-place function expression will correspond to an n-place function from individuals to individuals. An n-place predicate expression will correspond to an n-place function from individuals to truth values. An n-place connective will correspond to an n-place function from truth values to truth values (i.e., to an n-place boolean function.) Finally, a quantifier will correspond to a function from sets of individuals to truth values. (The existential quantifier, for instance, takes a set of individuals to the value true (i.e., the value 1) if and only if this set is nonempty.)

5. More semantic detail: models

You can think of a model as a microworld in which a first-order language has been interpreted according to the conventions above. For logical purposes, the most important part of a model is its *domain of individuals*. This is a limited set of objects serving as the range of the quantifiers, and which supplies references for terms.

In a model of a first-order language, each individual constant is assigned some member of the model's domain. As part of the policy that FOL must be unambiguous, we require that a unique domain member is assigned to each individual constant; but we do allow domain members to be assigned to more than one individual constant, or to be assigned to no individual constant at all. Each n-place function constant is assigned some n-place function from the domain to the domain. Each n-place predicate constant is assigned some n-place function from the domain to truth values.

Rules of truth (or of *satisfaction*) will determine a unique truth value for each formula of a first-order language in any model of that language, relative to an assignment $\mathbf{s}$ of values in the domain to each individual variable. I'll state these rules in a separate section. Meanwhile, an example should help us to collect our thoughts about models.

6. A little model

The domain of the model consists of three objects, a, b, and c. The first of these is yellow; the rest aren't. The first is bigger than the second, the second is bigger than the third. So we choose a FOL with individual constants a, b, c,[2] and with the 1-place predicate constant P and the 2-place predicate constant R. To complete the model, we need to choose an assignment of values to variables; we arbitrarity assign the value a to each variable.

The defining features of this model are summarized in the following table.

- **Domain:** $\mathrm{D} = \{\mathrm{a}, \mathrm{b}, \mathrm{c}\}$
- **Assignment of Model Values to Constants:**

FOL Constant	Value in the Model
a	a
b	b
c	c
P	$\mathbf{f}_1^{\mathrm{D}}(\mathrm{a})$
R	$\mathbf{f}_2^{\mathrm{D}}(\langle \mathrm{a}, \mathrm{b}\rangle, \langle \mathrm{b}, \mathrm{c}\rangle)$

[2]*Don't confuse objects with their names!* Typically, names are different from the objects that they name. So we'd expect the individual constant b to differ from the object b of the model. (However, the case in which some of the objects in the domain of the model are names, and where b names itself, is not ruled out.)

- **Assignment of values to variables:** $\mathbf{s}(x) = \mathrm{a}$, for all variables x.

In this table, $\mathbf{f}_n^{\mathrm{D}}(\mathrm{X})$ stands for the characteristic function of the set X. That is, $\mathbf{f}_n^{\mathrm{D}}(\mathrm{X})$ is the function from D^n (where D^n is the set of all n-tuples of members of D) to $\{0,1\}$ such that $\mathbf{f}_n^{\mathrm{D}}(\mathrm{X})(\mathrm{x}) = 1$ if $\mathrm{x} \in \mathrm{X}$, and otherwise $\mathbf{f}_n^{\mathrm{D}}(\mathrm{X})(\mathrm{x}) = 0$. (*Note:* we identify a set from D^n to the set $\{0,1\}$ of truth values with an n-place function from D to $\{0,1\}$; i.e., that is what an n-place function is. Also note that $\mathrm{D}^1 = \mathrm{D}$.)

7. Still more semantic detail: semantic rules

A semantic version of substitution can be defined on assignments: where $\mathbf{s}$ is an assignment of values from the domain D to a set of variables, and one of these variables is x, let $\mathbf{s}(d/x)$ be the assignment that is like $\mathbf{s}$, except that the value of x has been changed to d. That is, $\mathbf{s}(\mathrm{d}/x)(y) = \mathbf{s}(y)$ if $y \neq x$, and $\mathbf{s}(\mathrm{d}/x)(x) = \mathrm{d}$.

Where M is a model involving the assignment $\mathbf{s}$, d is an element of the domain of M, and x is a variable of the language that M interprets, let $M(\mathrm{d}/x)$ be the model that is like M except that its variable assignment is $\mathbf{s}(\mathrm{d}/x)$.

Let M be a model with domain D, of a language that contains all the constants and variables of a term t and formula A. Then M will assign t a member $[\![t]\!]_{\mathrm{M}}$ of the domain and A a truth value, $[\![A]\!]_{\mathrm{M}}$. An induction on the complexity of formulas specifies how this assignment works.

1. *Basic terms.*

   $[\![a]\!]_{\mathrm{M}}$ is specified as part of the definition of M.
   $[\![x]\!]_{\mathrm{M}} = \mathbf{s}(x)$.

2. *Complex terms.*

   $[\![f(t)]\!]_{\mathrm{M}} = [\![f]\!]_{\mathrm{M}}([\![t]\!]_{\mathrm{M}})$

3. *Basic formulas.*

   $[\![P(t_1, \ldots, t_n)]\!]_{\mathrm{M}} = [\![P]\!]_{\mathrm{M}}([\![t_1]\!]_{\mathrm{M}}, \ldots, [\![t_n]\!]_{\mathrm{M}})$

4. *Complex formulas.*

   (a) *Negations.*

   $[\![\neg A]\!]_{\mathrm{M}} = 0$ if $[\![A]\!]_{\mathrm{M}} = 1$, and
   $[\![\neg A]\!]_{\mathrm{M}} = 1$ if $[\![A]\!]_{\mathrm{M}} = 0$

   (b) *Conditionals.*

   $[\![A \to B]\!]_{\mathrm{M}} = 1$ if $[\![A]\!]_{\mathrm{M}} = 0$ or $[\![B]\!]_{\mathrm{M}} = 1$, and
   $[\![A \to B]\!]_{\mathrm{M}} = 0$ if $[\![A]\!]_{\mathrm{M}} = 1$ and $[\![B]\!]_{\mathrm{M}} = 0$.

   (c) *Universal quantifications.*

   $[\![\forall x A]\!]_{\mathrm{M}} = 1$ if $[\![\forall x A]\!]_{\mathrm{M}(\mathrm{d}/x)} = 1$ for all $\mathrm{d} \in \mathrm{D}$, and
   $[\![\forall x A]\!]_{\mathrm{M}} = 0$ if $[\![\forall x A]\!]_{\mathrm{M}(\mathrm{d}/x)} = 0$ for some $\mathrm{d} \in \mathrm{D}$.

Often (in analogy to roots of equations, as when 3 is said to satisfy $x^2-x-6=0$) we say that a model M *satisfies* a formula when $[\![A]\!]_{\mathrm{M}} = 1$. Generalizing to sets of formulas, we say that a model M *simultaneously satisfies* a set T if formulas when $[\![A]\!]_{\mathrm{M}} = 1$ for all $A \in T$.

8. Validity and implication

A formula A is said to be *valid*, $\Vdash A$, if A is assigned the value 1 (or *true*) in every model. And A is said to be *satisfiable* if A is assigned the value 1 in some model. A model M is said to *simultaneously satisfy* a set T of formulas if $[\![A]\!]_{\mathrm{M}} = 1$ for all $A \in T$. And T is said to be *simultaneously satisfiable* if there is a model that simultaneously satisfies T.

Finally, a set T of formulas is said to *imply* a formula A if A is assigned the value 1 (or **true**) by every model that simultaneously satisfies T.[3]

Note that implication is a generalization of validity, in the sense that $\Vdash A$ if and only if $\emptyset \Vdash A$. (You might want to prove this, as a way of checking that you understand the definitions.)

9. Proofs

In their simplest form, proofs are records of how to derive a conclusion from axioms by means of rules of inference. This assumes, of course, that a set of axioms has been presented, as well as a set of rules of inference.[4] So a proof can be defined as a list of formulas such that every entry either (1) is an axiom, or (2) follows from previous entries by a rule of inference. Such a list is a *proof of* A, where A is a formula, in case A is the list's last entry.

A major purpose of proofs is to obtain consensus about the consequences of assumptions. This purpose would be undermined if it were not possible to check mechanically whether a list of formulas is a proof. Therefore, it is usual to require proofs to be algorithmically recognizable. This recognizability condition will hold if for any formula A it is decidable whether A is an axiom, and if for any pair consisting of a finite list X of formulas and a formula A it is decidable whether A follows from X by a rule of inference. Note that this does not require the set of axioms to be finite; if we characterize an infinite set of axioms in terms of some decidable property of their shapes, proofs from these axioms will be recognizable, as long as rules of inference remain recognizable.

A formula A is said to be *provable* in case there is a proof of A. *Notation:* '$\vdash A$' is short for 'A is provable.' Sometimes we want to refer explicitly to the axiomatic basis, and say that $\vdash_{\mathbf{S}} A$, where $\mathbf{S}$ is a system of axioms and rules.

Hypothetical proof is a very natural and common reasoning technique, which allows the reasoner to invoke and discharge assumptions in the course of an argument. (We argue this way, for instance, in reasoning by cases: in this form of reasoning we begin by dividing the possibilities into a list of cases, and alternatively assume each of these cases, trying to reason to the desired conclusion. The cases can be represented by formulas of FOL; if, then, the

[3]Sometimes, A is said to be a *logical consequence* of T when T implies A. But here, I'll use 'logical consequence' more generally, for an abstract kind of consequence that could be characterized in any of a variety of ways.

[4]An inference is an operator taking a finite set of formulas (the premisses) and returning a formula (the conclusion).

cases are A and B and the desired conclusion is C, we know that $A \vee B$ is true, and wish to conclude C. So first we assume A and try to derive C; if this succeeds, we then discharge our first assumption, assume B, and try to derive C. If both hypothetical deductions succeed, we have proved C.)

The above definition of proof can easily be generalized to obtain a simple account of hypothetical proof. Modeling the hypotheses by a set T of formulas,[5] we say that a *deduction from* T is a list of formulas such that every entry either (1) is an axiom, or (2) is a member of T, or (3) follows from previous entries by a rule of inference. Such a list is a *deduction of* A *from* T in case A is the list's last entry. And a formula A is said to be *deducible from* a set T of formulas if there is a deduction of A from T.

Note that deducibility is a generalization of provability, in the sense that $\vdash A$ if and only if $\emptyset \vdash A$. (You might want to prove this, as a way of checking that you understand the definitions.)

10. Abstract properties of the consequence relation

Both the semantic definition of implication and the proof-theoretic definition of deducibility are ways of characterizing an intuitive notion of consequence in FOL; the idea is that A is a consequence of T when A follows from T due to logical considerations alone.

It's useful to look abstractly at some of the general properties that both of these consequence relation share. To do this, let's use $|\!\rhd$ as an abstract consequence relation. Then $|\!\rhd$ will relate a set T of formulas (which may, however, be empty) to a formula A.

Here, then, are a number of general properties. In all cases but one, these are easily provable from the definitions above, whether $|\!\rhd$ is taken semantically to be $\Vdash$, or proof-theoretically as $\vdash$.

1. *Identity.*

   If $A \in T$ then $T \mathrel{|\!\rhd} A$.

2. *Transitivity.*

   If $T \mathrel{|\!\rhd} A$ and $T \cup \{A\} \mathrel{|\!\rhd} B$ then $T \mathrel{|\!\rhd} B$

3. *Monotonicity.*

   If $T \mathrel{|\!\rhd} B$ then $T \cup \{A\} \mathrel{|\!\rhd} B$.

4. *Finiteness.*

   $T \mathrel{|\!\rhd} A$ if and only if $T' \mathrel{|\!\rhd} A$ for some finite subset T' of T.

[5]The set T needn't be finite; for the same conditions that apply to logical axioms, we might want to consider an infinite set of hypotheses satisfying some general conditions. For instance, if we knew that infinitely many things satisfy a one-place predicate P, we might be interested in the set

$$\{\exists_{\geq 1} x P(x), \exists_{\geq 2} x P(x), \ \ldots\ , \exists_{\geq n} x P(x), \ \ldots\}.$$

The only one of these properties that can't be proved trivially for $\vdash$ or $\Vdash$ is the finiteness of $\Vdash$; the proof of this result, which uses an algebraic idea called the ultraproduct construction, is usually one of the first results in a serious textbook in model theory. The rest of the properties all follow immediately from definitions.

Take monotonicity, for instance. Proof-theoretically, it says that if $T \vdash B$ then $T \cup \{A\} \vdash B$. To prove it in this form, suppose that $T \vdash B$. Then there is a deduction $\mathcal{P}$ of B from T. But, by inspecting the definition, we see that $\mathcal{P}$ is also a deduction of B from $T \cup \{A\}$. (Note that it is not required that every member of T be used in a deduction from T.) Semantically, monotonicity says that if $T \Vdash B$ then $T \cup \{A\} \Vdash B$. To prove it in this form, suppose that $T \Vdash B$. Then every model that simultaneously satisfies T also satisfies B. But clearly, every model that simultaneously satisfies $T \cup \{A\}$ also satisfies B; so every model that simultaneously satisfies $T \cup \{A\}$ must satisfies B, so that $T \cup \{A\}$ implies B.

There are also a number of less general properties that hold of consequence in FOL, but that would have to be regarded as properties of FOL, rather than of logical consequence in general: the fact that if $T \mid\!\triangleright \forall x A$ then $T \mid\!\triangleright A(t/x)$ is an example. It turns out that even at this level of detail, semantic and proof theoretic consequence are equivalent in FOL: in fact, it can be proved that $T \vdash A$ if and only if $T \Vdash A$, for all T and A.