A Very Brief Overview of Quantified First-Order Logic

Step 1: Lexicon
Our logical languages will have the following sorts of symbols:

1. Constants (typically lower-case letters from the beginning of the alphabet)
2. Predicates (typically upper-case letters, and also =) (of any number of argument places)
3. Function Symbols (typically lower-case f or g) (of any number of argument places)
4. Variables (officially, $x_1, x_2, \ldots$; unofficially, lower-case letters from the end of the alphabet)
5. Connectives ($\neg$, $\wedge$, $\vee$, $\rightarrow$, $\leftrightarrow$)
6. Quantifiers ($\forall$, $\exists$)
7. Parentheses

All languages will have the full collection of variables, connectives, quantifiers, and parentheses, and all languages will contain the identity symbol =, but languages can vary in their collection of constants, (non-identity) predicates, and function symbols (the *non-logical* vocabulary). The non-logical vocabulary of a particular language is called its *signature*. A language will sometimes be identified in the form $\mathcal{L}(\Sigma)$, where Σ is the signature. Thus $\mathcal{L}(\cdot)$ or $\mathcal{L}(\cdot, e, ^{-1})$ could be a language for group theory, $\mathcal{L}(\in)$ a language for set theory, and $\mathcal{L}(+, 0, 1)$ a language for arithmetic.

Step 2: Syntax
We first recursively define *terms*:

1. Any variable is a term
2. Any constant is a term
3. If f^n is an n-ary function symbol, and $t_1, \ldots, t_n$ are terms, then $f(t_1, \ldots, t_n)$ is a term

Next, we recursively define *sentences* (sometimes called *formulas*):

1. If P^n is an n-ary predicate and $t_1, \ldots, t_n$ are terms, then $P^n t_1 \ldots t_n$ is a sentence
2. If t_1 and t_2 are terms, then $t_1 = t_2$ is a sentence
3. If A is a sentence, then $\neg A$ is a sentence
4. If A and B are sentences, then $(A \wedge B)$, $(A \vee B)$, $(A \rightarrow B)$, and $(A \leftrightarrow B)$ are all sentences.
5. If A is a sentence and x is a variable, then $\exists x\, A$ is a sentence
6. If A is a sentence and x is a variable, then $\forall x\, A$ is a sentence

An occurrence of a variable x in a sentence A is *bound* if it is in the scope of an $\exists x$ or $\forall x$ quantifier (that is, if it is part of the smallest sentence following the quantifier). It is *free* otherwise. A sentence is *closed* if all of its variable occurrences are bound, and *open* otherwise.

Step 3: Models
A *model* for a language $\mathcal{L}(\Sigma)$ is an ordered pair $(D, [\![\cdot]\!])$, where:

1. D is an arbitrary non-empty set (the *domain of quantification*)
2. $[\![\cdot]\!]$ is the *valuation function*, and has the following features:
   (a) For each constant a in Σ, $[\![a]\!]) \in D$
   (b) For each n-place predicate P^n in Σ, $[\![P^n]\!] \in D^n$
   (c) For each n-place function symbol f^n in Σ, $[\![f^n]\!] \in \{g : \ g : D^n \mapsto D\}$

Step 4: Truth in a Model (Relative to an Assignment)
An *assignment* is a function from the set of variables to the domain of a model. We write $h[a/x]$ to name the assignment which maps the variable x to object a, and maps all other variables in the same way that h maps them. We write $h \sim_x h'$ if $h' = h[a/x]$ for some $a \in D$, and call h' an x-variant of h. First we define an extended valuation function $[\![\cdot]\!]^+_{\mathcal{M},h}$ which assigns, relative to a model $\mathcal{M}$ and an assignment h, to each term an element of the domain:

1. For any constant a, $[\![a]\!]^+_{\mathcal{M},h} = [\![a]\!]$
2. For any variable x, $[\![x]\!]^+_{\mathcal{M},h} = h(x)$
3. For any function symbol F^n and terms $t_1, \ldots, t_n$, $[\![f^n(t_1, \ldots, t_n)]\!]^+_{\mathcal{M},h} = [\![f^n]\!]([\![t_1]\!]^+_{\mathcal{M},h}, \ldots, [\![t_n]\!]^+_{\mathcal{M},h})$

We now define the truth of a sentence relative to a model and an assignment function. We write $\mathcal{M}, h \vDash A$ to mean that A is true relative to model $\mathcal{M}$ and assignment h. We proceed recursively:

1. $\mathcal{M}, h \vDash P^n t_1 \ldots t_n$ iff $< [\![t_1]\!]^+_{\mathcal{M},h}, \ldots, [\![t_n]\!]^+_{\mathcal{M},h} > \in [\![P^n]\!]$
2. $\mathcal{M}, h \vDash t_1 = t_2$ iff $[\![t_1]\!]^+_{\mathcal{M},h} = [\![t_2]\!]^+_{\mathcal{M},h}$
3. $\mathcal{M}, h \vDash \neg A$ iff $\mathcal{M}, h \nvDash A$
4. $\mathcal{M}, h \vDash A \wedge B$ iff $\mathcal{M}, h \vDash A$ and $\mathcal{M}, h \vDash B$
5. $\mathcal{M}, h \vDash A \vee B$ iff $\mathcal{M}, h \vDash A$ or $\mathcal{M}, h \vDash B$
6. $\mathcal{M}, h \vDash A \rightarrow B$ iff $\mathcal{M}, h \nvDash A$ or $\mathcal{M}, h \vDash B$
7. $\mathcal{M}, h \vDash A \leftrightarrow B$ iff either (i) $\mathcal{M}, h \vDash A$ and $\mathcal{M}, h \vDash B$ or (ii) $\mathcal{M}, h \nvDash A$ and $\mathcal{M}, h \nvDash B$
8. $\mathcal{M}, h \vDash \exists x A$ iff there is some $h' \sim_x h$ such that $\mathcal{M}, h' \vDash A$
9. $\mathcal{M}, h \vDash \forall x A$ iff every h' such that $h' \sim_x h$ is such that $\mathcal{M}, h' \vDash A$

It can then be shown that if A is a closed sentence, then given any assignments h and h', $\mathcal{M}, h \vDash A$ iff $\mathcal{M}, h' \vDash A$.

Step 5: The Basic Logical Notions

1. Sentence A *implies* sentence B, which we write '$A \vDash B$', iff given any model $\mathcal{M}$ and any assignment h, if $\mathcal{M}, h \vDash A$, then $\mathcal{M}, h \vDash B$.
2. More generally, if Δ is a set of sentences, $\Delta \vDash B$ iff given any model $\mathcal{M}$ and any assignment h, if for each $A \in \Delta$, $\mathcal{M}, h \vDash A$, then $\mathcal{M}, h \vDash B$.
3. Two sentences A and B are *equivalent* iff given any model $\mathcal{M}$ and any assignment h, $\mathcal{M}, h \vDash A$ iff $\mathcal{M}, h \vDash B$. (Thus A is equivalent to B iff both $A \vDash B$ and $B \vDash A$.)

Step 6: Proofs

Let A, B, and C be arbitrary sentences, τ_1 and τ_2 be arbitrary terms, and χ be a variable that does not occur free in C. Let $A[\alpha/\beta]$ be the sentence that results from simultaneously replacing all occurrences of α in A with β. Then the following sentences are *axioms*:

1. $A \rightarrow (B \rightarrow A)$
2. $(A \rightarrow (B \rightarrow C)) \rightarrow ((A \rightarrow B) \rightarrow (A \rightarrow C))$
3. $(\neg A \rightarrow \neg B) \rightarrow (B \rightarrow A)$

4. $\forall\chi(A \to B) \to (\forall\chi A \to \forall\chi B)$

5. $\forall\chi A \to A[\tau_1/\chi]$

6. $C \to \forall\chi C$

7. $\tau_1 = \tau_1$

8. $\tau_1 = \tau_2 \to (A \to A[\tau_1/\tau_2])$

Also, the following sentences are *definitions*:

1. $(A \wedge B) \to \neg(A \to \neg B)$

2. $\neg(A \to \neg B) \to (A \wedge B)$

3. $(A \vee B) \to (\neg A \to B)$

4. $(\neg A \to B) \to (A \vee B)$

5. $(A \leftrightarrow B) \to ((A \to B) \wedge (B \to A))$

6. $((A \to B) \wedge (B \to A)) \to (A \leftrightarrow B)$

Let *modus ponens* be the rule that takes as input any two sentences A and $A \to B$ and produces as output the sentence B.

A *proof* is a finite sequence of sentences such that each sentence on the list is either (i) an axiom, (ii) a definition, or (iii) the result of applying modus ponens to two sentences earlier on the list.

We say $\vdash A$ if there is a proof whose final sentence is A. We say $A \vdash B$ if there is a proof whose final sentence is $A \to B$. We say $A_1, \ldots, A_n$ if there is a proof whose final sentence is $A_1 \to (A_2 \to (\ldots \to (A_n \to B))\ldots)$. If Δ is a set of sentences, then we say $\Delta \vdash B$ if there are $A_1, \ldots, A_n \in \Delta$ such that $A_1, \ldots, A_n \vdash A$.

(This is only one way of setting up a proof system. We will not be very concerned with the details of proof systems, but just with their existence and broad outlines.)