15-399 Supplementary Notes: Higher-Order Logic

Robert Harper

March 18, 2004

1 Second- and Higher-Order Logic

First-order logic is the logic of quantification over the elements of a *type*. First-order logic permits us to state properties that hold of every, or some, element of a type. For example, "there are infinitely many prime numbers" or "every natural number may be factored into a product of primes."

Second-order logic[1] is the logic of quantification over *propositions*. Second-order logic permits us to state properties about propositions such as "every proposition implies itself" or "there are decidable propositions".

Higher-order logic is the logic of quantification over *propositional functions*, functions whose range consists of propositions. These include *predicates*, which are propositional functions whose domain is a type such as the natural numbers or tuples of natural numbers, and *connectives*, which are propositional functions whose domain consists of propositions or tuples of propositions.

Second-order Logic

The second-order universal and existential quantifiers are written $\forall_2 p.Q$ and $\exists_2 p.Q$, where p is a *proposition variable* standing for an unspecified proposition. The subscript "2" serves to distinguish the second-order quantifiers from their first-order counterparts, which are, for the time being, written $\forall_1 x{\in}\tau.Q$ and $\exists_1 x{\in}\tau.Q$. Shortly we will unify both forms of quantification into a single, unsubscripted form.

The rules governing second-order universal quantification are as follows:

$$\frac{\Gamma, p\ \mathit{prop} \vdash Q\ \mathit{prop}}{\Gamma \vdash \forall_2 p.\ Q\ \mathit{prop}}\ (\forall_2 F^p)$$

$$\frac{\Gamma, p\ \mathit{prop} \vdash Q\ \mathit{true}}{\Gamma \vdash \forall_2 p.\ Q\ \mathit{true}}\ (\forall_2 I^p) \qquad \frac{\Gamma \vdash \forall_2 p.\ Q\ \mathit{true} \qquad \Gamma \vdash P\ \mathit{prop}}{\Gamma \vdash [P/p]Q\ \mathit{true}}\ (\forall_2 E)$$

[1] Classical second-order logic is sometimes called the logic of *quantified Boolean formulas*, or *QBF*.

These are exactly like the rules for the first-order universal quantifier, except that quantifier ranges over propositions, rather than the elements of a type.

The rules governing second-order existential quantification are as follows:

$$\frac{\Gamma, p\ prop \vdash Q\ prop}{\Gamma \vdash \exists_2 p.\ Q\ prop}\,(\exists_2 F^p) \qquad \frac{\Gamma \vdash P\ prop \qquad \Gamma \vdash [P/p]Q\ true}{\Gamma \vdash \exists_2 p.\ Q\ true}\,(\exists_2 I)$$

$$\frac{\Gamma \vdash \exists_2 p.\ Q\ true \qquad \Gamma, p\ prop, Q\ true \vdash R\ true}{\Gamma \vdash R\ true}\,(\exists_2 E)$$

Here again the rules are essentially the same as those for the first-order existential quantifier, with the difference being that the quantifier ranges over propositions, rather than elements of a type.

First-order quantification is said to be *predicative*, because the domain of quantification is a type whose meaning is given independently of the quantifier itself. This means that the meaning of the type is given without using the first-order quantifiers (using introduction and elimination rules), and that the meaning of quantification is given in terms of the elements of that type.

The situation with second-order quantification is different, because the quantifier ranges over propositions, which includes those that make use of the second-order quantifiers. Thus the meaning of a proposition may involve the meaning of all possible propositions, including the one whose meaning is being given. Because of this element of "circularity" or "self-reference", second-order quantification is said to be *impredicative*.

For example, consider the proposition $P = \forall_2 p.\ Q$. For P to be true means that $[R/p]Q$ is true for *every* proposition R. In particular, $[P/p]Q$ is true, since P is a proposition. If Q is just p, then by this reasoning the truth of P implies the truth of P, which is hardly surprising. But suppose that Q is $p \supset p$. In that case P is true, and as a result $P \supset P$ is also true, again not suprising in itself, since we knew this in advance. What makes many people uneasy, though, is that the truth of $\forall_2 p.\ Q$ implies facts that are much "larger" than Q alone, and which may even involve P itself.

Because of this, impredicativity has been a source of some controversy in the philosophy of constructive mathematics. Rather than attempt to investigate these controversies, we content ourselves here to exploring some of its consequences, which are dramatic.

Encoding Propositional Connectives

One surprising fact about second-order logic is that all other propositional connectives, plus second-order existential quantification, are *definable* in terms of just second-order universal quantification and implication. That is, given the latter two constructs, we may *define* truth, falsehood, conjunction, disjunction, and second-order existential quantification, just as we defined negation in terms of implication and falsehood. Moreover, given in addition first-order universal quantification, we may define first-order existential quantification.

To show that a connective or quantifier is definable in terms of implication and universal quantification, we must do two things:

1. Give a definition of the connective or quantifier in terms of implication and universal quantification.
2. Show that the introduction and elimination rules for the connective or quantifier are derivable from this definition.

We will now show that $\top$, $\bot$, $\wedge$, and $\vee$ are all definable in terms of $\supset$ and $\forall_2$. Please refer to Pfenning's *Constructive Logic* for the rules for these logical connectives.

To see how to arrive at the definition of a connective, let us review our methodology for deriving the rules governing a logical connective. Recall that the overall idea is to specify the conditions for showing that a proposition construct from that connective is true (its introduction rule), and then to derive the conditions for using the truth of a proposition constructed with that connective (its elimination rule) by "inverting" the introduction rule. The elimination rule should extract exactly the information that went into the introduction rule so that applying an elimination rule immediately after its corresponding introduction rule will always cancel via a reduction step. Our strategy for defining a connective will be to capture the "content" of the elimination rule as a proposition of second-order implicational logic, so that the elimination rule is easily seen to be derivable, and then showing that the introduction rules are also derivable. The idea is to examine what we wish to conclude from the truth of a proposition, then ensure that this is consistent with what we put into it.

Let us first consider the two simplest cases, that of $\top$ and $\bot$. The proposition $\top$ has only one introduction rule, with no premises, and no elimination rules. We must define $\top$ to be a formula of second-order implicational logic that is true, given no premises. A handy definition is to take $\top \equiv \forall_2 p.p \supset p$; it is easy to see that $\top$ *true* given this definition. The proposition $\bot$ has only one elimination rule, and no introduction rules. We must define $\bot$ so that if it is true, then any other proposition is true. This suggests defining $\bot \equiv \forall_2 p.p$. If $\bot$ *true*, then taking p to be any proposition P, we may conclude P *true*, as desired.

Let's consider how to define $P \vee Q$. The elimination rule allows us to reason by cases, deriving R *true* from both the assumption P *true* and, separately, from the assumption Q *true*. Since $P \vee Q$ is true only if either P *true* or Q *true*, then one or the other case must apply, allowing us to conclude R *true*. This suggests that we define disjunction as follows:

$$P \vee Q \equiv \forall_2 r.\ (P \supset r) \supset ((Q \supset r) \supset r).$$

This definition captures precisely the meaning of the elimination rule. Indeed, if $P \vee Q$ *true* in the sense of this definition, and, moreover, P *true* $\vdash R$ *true* and Q *true* $\vdash R$ *true*, then $P \supset R$ *true* and $Q \supset R$ *true*, so by two applications of implication elimination we obtain R *true*. It remains to derive the introduction

rules. Suppose that P *true*; we are to show that $P \vee Q$ *true* in the sense of the foregoing definition. To do so, suppose that r *prop*, $P \supset r$ *true*, $Q \supset r$ *true*; we are to show that r *true*. But this follows immediately by substitution, using the assumption that P *true*. The other introduction rule is derived by an analogous argument.

Let's consider how to define $P \wedge Q$. We'd like to encode the elimination rules as a single proposition of second-order implicational logic. But since there are two elimination rules, a direct approach would seem to require some form of conjunction, which is precisely what we seek to avoid. The trick is to formulate an alternative, but equivalent, elimination rule for conjunction:

$$\frac{\Gamma \vdash P \wedge Q \ \mathit{true} \qquad \Gamma, P \ \mathit{true}, Q \ \mathit{true} \vdash R \ \mathit{true}}{\Gamma \vdash R \ \mathit{true}}$$

In words, to derive R *true* from $P \wedge Q$ *true*, it suffices to derive R *true* from P *true* and Q *true*. Given this, we may define conjunction as follows:

$$P \wedge Q \equiv \forall_2 r.\ (P \supset (Q \supset r)) \supset r.$$

It is easy to check that the elimination rule for conjunction is derivable, given this definition. What about the introduction rule? Suppose that P *true* and Q *true*; we are to show that $P \wedge Q$ *true*, with conjunction defined as above. Suppose r *prop*, $P \supset (Q \supset r)$ *true*; we are to show r *true*. But this follows by two uses of implication elimination using our assumptions.

Finally, let's define the second-order existential quantifier:

$$\exists_2 p.\ Q \equiv \forall_2 r.\ (\forall_2 p.\ Q \supset r) \supset r.$$

We leave it as an exercise for the reader to verify that the rules for the second-order existential quantifier are derivable in the sense of this definition.

It is instructive to examine the encodings of proof terms for the logical connectives and quantifiers.[2]

Here are the encodings for truth:

$$\begin{array}{rcl} \top & := & \forall r.\ r \supset r \\ \langle\rangle & := & \lambda r.\ \lambda u{:}r.\ u \end{array}$$

Here are the encodings for falsehood:

$$\begin{array}{rcl} \bot & := & \forall r.\ r \\ \mathbf{abort}_R(M) & : & M(R) \end{array}$$

Here are the encodings for conjunction:

$$\begin{array}{rcl} P \wedge Q & := & \forall r.\ (P \supset Q \supset r) \supset r \\ \langle M, N\rangle_{P,Q} & := & \lambda r.\ \lambda u{:}P \supset Q \supset r.\ u(M)(N) \\ \mathbf{fst}_{P,Q}(M) & := & M(P)(\lambda u : P.\ \lambda v : Q.\ u) \\ \mathbf{snd}_{P,Q}(M) & := & M(Q)(\lambda u : P.\ \lambda v : Q.\ v) \end{array}$$

[2] To avoid notational clutter, some type labels are omitted.

It is easy to check that the expected definitional equalities hold under these encodings:

$$\begin{array}{rcl}
\mathbf{fst}(\langle M, N\rangle) & \equiv & \mathbf{fst}(\lambda r.\ \lambda u{:}P \supset Q \supset r.\ u(M)(N)) \\
& \equiv & (\lambda r.\ \lambda u{:}P \supset Q \supset r.\ u(M)(N))(P)(\lambda u.\ \lambda v.\ u) \\
& \equiv & (\lambda u.\ \lambda v.\ u)(M)(N) \\
& \equiv & M
\end{array}$$

$$\begin{array}{rcl}
\mathbf{snd}(\langle M, N\rangle) & \equiv & \mathbf{snd}(\lambda r.\ \lambda u{:}P \supset Q \supset r.\ u(M)(N)) \\
& \equiv & (\lambda r.\ \lambda u{:}P \supset Q \supset r.\ u(M)(N))(Q)(\lambda u.\ \lambda v.\ v) \\
& \equiv & (\lambda u.\ \lambda v.\ v)(M)(N) \\
& \equiv & N
\end{array}$$

Here are the encodings for disjunction:

$$\begin{array}{rcl}
P \vee Q & := & \forall r.\ (P \supset r) \supset (Q \supset r) \supset r \\
\mathbf{inl}_{P,Q}(M) & := & \lambda r.\ \lambda u : P \supset r.\ \lambda v : Q \supset r.\ u(M) \\
\mathbf{inr}_{P,Q}(M) & := & \lambda r.\ \lambda u : P \supset r.\ \lambda v : Q \supset r.\ v(M) \\
\mathbf{case}_R\, M\ \mathbf{of}\ \mathbf{inl}(u : P) \Rightarrow N \mid \mathbf{inr}(v : Q) \Rightarrow O & & \\
\qquad := & & M\,(R)\,(\lambda u : P.\ N)\,(\lambda v : Q.\ O)
\end{array}$$

We may then check the required definitional equalities:

$$\begin{array}{rcl}
\mathbf{case}_R\, \mathbf{inl}(M)\ \mathbf{of}\ \mathbf{inl}(u : P) \rightarrow N \mid \mathbf{inr}(v : Q) \Rightarrow O & & \\
\quad \equiv & & \mathbf{inl}(M)\,(R)\,(\lambda u.\ N)\,(\lambda v.\ O) \\
\quad \equiv & & (\lambda r.\ \lambda u{:}P \supset r.\ \lambda v : Q \supset r.\ u(M))(R)(\lambda u.\ N)(\lambda v.\ O) \\
\quad \equiv & & (\lambda u{:}P \supset R.\ \lambda v{:}Q \supset R.\ u(M))(\lambda u.\ N)(\lambda v.\ O) \\
\quad \equiv & & (\lambda u.\ N)(M) \\
\quad \equiv & & [M/u]N
\end{array}$$

(The other case is very similar.)

We leave it to the reader to define the proof terms for existential quantification, and demonstrate that the required definitional equality holds under these definitions.

Unifying First- and Higher-Order Quantification

The only distinction between first- and second-order quantification is the domain of quantification. In the former the domain is a type, in the latter it consists of all propositions. We may unify the two forms of quantification into one by simply introducing a type of propositions:

$$\mathbf{prop}\ \textit{type}.$$

The categorical judgement $P\ \textit{prop}$ is replaced by the judgement $P{\in}\mathbf{prop}$.

The introduction rules for the type **prop** are these:

$$\frac{\Gamma \vdash P{\in}\mathbf{prop} \qquad \Gamma \vdash Q{\in}\mathbf{prop}}{\Gamma \vdash P \supset Q{\in}\mathbf{prop}}\supset F \qquad \frac{\Gamma, x{\in}\tau \vdash P{\in}\mathbf{prop}}{\Gamma \vdash \forall x{\in}\tau.\ P{\in}\mathbf{prop}}\forall_\tau F$$

Unlike most other types, there are *no* elimination rules for the type **prop**! There are several reasons for this, the most important of which is that we do not wish to constrain propositions to be *only* those given by these rules. Rather, we wish the system to be *open-ended* in that it may accommodate extensions with additional forms of proposition.

By introducing a type of propositions we may consolidate first- and second-order quantification into a single notion of quantification over a type. Thus $\forall_1 x{\in}\tau.\ P(x)$ stands for $\forall x{\in}\tau.\ P(x)$, and $\forall_2 p.\ Q$ stands for $\forall p{\in}\mathbf{prop}.\ Q$. The definition of the existential quantifiers given above carries over to the more general case, permitting us to reduce our stock of primitive logical notions to just implication and universal quantification (over a type).

Rather than give a list of introduction rules for each logical connective, we may instead treat them as constants, as follows:

$$\begin{array}{rcl} \supset & \in & (\mathbf{prop} \times \mathbf{prop}) \to \mathbf{prop} \\ \forall_\tau & \in & (\tau \to \mathbf{prop}) \to \mathbf{prop} \end{array}$$

Thus the proposition $P \supset Q$ is convenient shorthand for $\supset \langle P, Q\rangle$, the application of the constant $\supset$ to the pair of propositions $\langle P, Q\rangle$. Similarly, $\forall x{\in}\tau.\ P$ is shorthand for $\forall_\tau(\lambda x{\in}\tau.\ P)$. Using this notation, the elimination rule for the universal quantifier may be written as follows:

$$\frac{\Gamma \vdash \forall_\tau(F)\ \mathit{true} \quad \Gamma \vdash t{\in}\tau}{\Gamma \vdash F(t)\ \mathit{true}}\ (\forall E)$$

The "argument", F, to $\forall_\tau$ is a propositional function of type $\tau \to \mathbf{prop}$, which is instantiated by applying it to $t{\in}\tau$ to form $F(\tau)$.

Higher-Order Logic

A *propositional function* is any function whose range is the type **prop** of propositions. Propositional functions include *predicates*, which are functions of type $\tau \to \mathbf{prop}$ for some type τ, *unary connectives*, of type $\mathbf{prop} \to \mathbf{prop}$, *binary connectives*, of type $\mathbf{prop} \to \mathbf{prop} \to \mathbf{prop}$, and so forth.

Predicate quantification may be used to state the principle of mathematical induction as a single proposition:

$$\forall p{\in}\mathbf{nat} \to \mathbf{prop}.\ p(\mathbf{0}) \wedge \forall n{\in}\mathbf{nat}.\ (p(n) \supset p(\mathbf{s}(n))) \supset \forall n{\in}\mathbf{nat}.\ p(n).$$

Similarly, the principle of list induction may be stated as follows:

$$\forall p{\in}\tau\ \mathbf{list} \to \mathbf{prop}.\ p(\mathbf{nil}) \wedge \forall a{\in}\tau.\ \forall l{\in}\tau\ \mathbf{list}(p(l) \supset p(a :: l)) \supset \forall l{\in}\tau\ \mathbf{list}.\ p(l)..$$

Note that both of these propositions are true! That is, we may prove them as theorems using the rules of constructive logic.

The principle of *extensionality* for predicates over the natural numbers states that predicates "respect" equality of natural numbers:

$$\forall p{\in}\mathbf{nat} \to \mathbf{prop}.\ \forall m{\in}\mathbf{nat}.\ \forall n{\in}\mathbf{nat}.\ p(m) \wedge m =_{\mathit{N}} n \supset p(n).$$

The principle of extensionality is sometimes called the principle of *indiscernability of identicals*: equal numbers have the same properties.

The principle of extensionality may be proved by a straightforward inductive argument. More precisely, we may prove by induction on m and n that $\forall p{\in}\mathbf{nat}\to\mathbf{prop}.\ p(m)\wedge m =_N n \supset p(n)$ *true*. This reduces to four proof obligations:

1. $\forall p{\in}\mathbf{nat}\to\mathbf{prop}.\ p(\mathbf{0})\wedge \mathbf{0} =_N \mathbf{0}\supset p(\mathbf{0})$ *true*. This is trivial.
2. $\forall p{\in}\mathbf{nat}\to\mathbf{prop}.\ p(\mathbf{0})\wedge \mathbf{0} =_N \mathbf{s}(n')\supset p(\mathbf{s}(n'))$ *true*. This is vacuously true.
3. $\forall p{\in}\mathbf{nat}\to\mathbf{prop}.\ p(\mathbf{s}(m'))\wedge \mathbf{s}(m') =_N \mathbf{0}\supset p(\mathbf{0})$ *true*. Also vacuously true.
4. Assuming the induction hypothesis

   $$\forall p{\in}\mathbf{nat}\to\mathbf{prop}.\ p(m')\wedge m' =_N n'\supset p(n'),$$

   show that

   $$\forall p{\in}\mathbf{nat}\to\mathbf{prop}.\ p(\mathbf{s}(m'))\wedge \mathbf{s}(m') =_N \mathbf{s}(n')\supset p(\mathbf{s}(n')).$$

   So assume $p{\in}\mathbf{nat}\to\mathbf{prop}$, $p(\mathbf{s}(m'))$ *true*, and $\mathbf{s}(m') =_N \mathbf{s}(n')$ *true*. It follows that $m' =_N n'$, and so by instantiating the induction hypothesis with the predicate $\lambda x{\in}\mathbf{nat}.p(\mathbf{s}(x))$, we obtain $p(\mathbf{s}(n'))$ *true* as required.