4.2 Computational Meaning of Quantification

Returning to one of our motivating examples, we saw that a constructive proof of $\forall x{:}\mathsf{nat}.\ \exists y{:}\mathsf{nat}.\ y > x \wedge \mathsf{prime}(x)$ should be a function which, when given a natural number x returns a natural number y which is greater than x and prime. We therefore suspect the computational content of a proof of universal quantifier over $\forall x{:}\tau.\ A(x)$ should be function from objects t of type τ to proofs of $A(t)$. The meaning of an existential quantifier $\exists x{:}\tau.\ A(x)$ should consist of a *withness* term t of type τ and a proof that $A(t)$ holds as specified.

Restating the above, the computational meaning of a proof of $\forall x{:}\tau.\ A(x)$ *true* is a function which, when given an argument t of type τ, returns a proof of $A(t)$. If we don't mind overloading notation, we obtain the following proof term assignment. We use the notation with localized hypotheses, which can now be either assumptions $a{:}\tau$ for object parameters a with their types or $u{:}A$ for propositional hypotheses A.

$$\frac{\Gamma, a{:}\tau \vdash M : A(a)}{\Gamma \vdash \lambda a{:}\tau.\ M : \forall x{:}\tau.\ A(x)}\ \forall I^a$$

$$\frac{\Gamma \vdash M : \forall x{:}\tau.\ A(x) \quad \Gamma \vdash t : \tau}{\Gamma \vdash M\,t : A(t)}\ \forall E$$

The computation rule simply performs the required substitution, and expansion generates an abstraction.

$$\begin{array}{lcll} (\lambda a{:}\tau.\ M)\,t & \Longrightarrow_R & [t/a]M & \\ M : \forall x{:}\tau.\ A & \Longrightarrow_E & \lambda a{:}\tau.\ M\,a & \text{where } a \text{ not free in } M \end{array}$$

At this point we may realize that a parameter a is nothing but a variable bound in proofs. We would not lose anything if we named such variables x to unify the notation.

The existential quantifier $\exists x{:}\tau.\ A(x)$ lies at the heart of constructive mathematics. This is because a proof of this should contain a *witness* t of type τ such that $A(t)$ *true*. The proof term assignment and computational contents of these rules is not particularly difficult. The proof term for an existential introduction is a pair consisting of the witness t and the proof that t satisfies the stated property. The elimination rule destructs the pair, making the components accessible.

$$\frac{\Gamma \vdash t : \tau \quad \Gamma \vdash M : A(t)}{\Gamma \vdash \langle t, M\rangle : \exists x{:}\tau.\ A(x)}\ \exists I$$

$$\frac{\Gamma \vdash M : \exists x{:}\tau.\ A(x) \quad \Gamma, a{:}\tau, u{:}A(a) \vdash N : C}{\Gamma \vdash \mathbf{let}\ \langle a, u\rangle = M\ \mathbf{in}\ N : C}\ \exists E$$

The reduction rule is straightforward, substituting both the witness and the proof term certifying its correctness.

$$\mathbf{let}\langle a, u\rangle = \langle t, M\rangle \mathbf{\ in\ } N \quad \Longrightarrow_R \quad [M/u]\,[t/a]\,N$$

As in the case of the propositional connectives, we now consider various interactions between quantifiers and connectives to obtain an intuition regarding their properties.

By annotating the earlier derivation that universal quantification distributes over conjunction we can extract the following proof term for this judgment (omitting some labels):

$$\begin{array}{rl} & \lambda u.\ \langle \lambda a{:}\tau.\ \mathbf{fst}\,(u\,a), \lambda b{:}\tau.\ \mathbf{snd}\,(u\,b)\rangle \\ : & (\forall x{:}\tau.\ A(x) \wedge B(x)) \supset (\forall x{:}\tau.\ A(x)) \wedge (\forall x{:}\tau.\ B(x)) \end{array}$$

The opposite direction also holds, which means that we can freely move the universal quantifier over conjunctions and vice versa. This judgment (and also the proof above) are parametric in τ. Any instance by a concrete type for τ will be an evident judgment. We show here only the proof term (again omitting some labels):

$$\begin{array}{rl} & \lambda u.\ \lambda a{:}\tau.\ \langle (\mathbf{fst}\ u)\ a, (\mathbf{snd}\ u)\ a\rangle \\ : & (\forall x{:}\tau.\ A(x)) \wedge (\forall x{:}\tau.\ B(x)) \supset (\forall x{:}\tau.\ A(x) \wedge B(x)) \end{array}$$

The proof that an existential can be pushed into the antecedent of an implication has the following proof term.

$$\begin{array}{rl} & \lambda u.\ \lambda a{:}\tau.\ \lambda w.\ u\,\langle a, w\rangle \\ : & ((\exists x{:}\tau.\ A(x)) \supset C) \supset \forall x{:}\tau.\ (A(x) \supset C) \end{array}$$

For the reverse implication we extract from the earlier proof:

$$\begin{array}{rl} & \lambda u.\ \lambda w.\ \mathbf{let}\ \langle a, v\rangle = w\ \mathbf{in}\ (u\,a)\,v \\ : & (\forall x{:}\tau.\ (A(x) \supset C)) \supset ((\exists x{:}\tau.\ A(x)) \supset C) \end{array}$$

4.3 First-Order Logic

First-order logic, also called the predicate calculus, is concerned with the study of propositions whose quantifiers range over a domain about which we make no assumptions. In our case this means we allow only quantifiers of the form $\forall x{:}\tau.\ A(x)$ and $\exists x{:}\tau.\ A(x)$ that are parametric in a type τ. We assume only that τ *type*, but no other property of τ. When we add particular types, such as natural numbers nat, we say that we reason within specific theories. The theory of natural numbers, for example, is called *arithmetic*. When we allow essentially arbitrary propositions and types explained via introduction and elimination constructs (including function types, product types, etc.) we say that we reason in *type theory*. It is important that type theory is open-ended: we can always add new propositions and new types and even new judgment forms, as long as

we can explain their meaning satisfactorily. On the other hand, first-order logic is essentially closed: when we add new constructs, we work in other theories or logics that include first-order logic, but we go beyond it in essential ways.

We have already seen some examples of reasoning in first-order logic in the two previous sections. In this section we investigate the truth of various other propositions in order to become comfortable with first-order reasoning. Just like propositional logic, first-order logic has both classical and constructive variants. We pursue the constructive or intuitionistic point of view. We can recover classical truth either via an interpretation such as Gödel's translation, discussed later in the class, or by adding the law of excluded middle. The practical difference at the first-order level is the interpretation of the existential quantifier. In classical logic, we can prove a proposition $\exists x{:}\tau.\ A(x)$ *true* by proving $\neg\forall x{:}\tau.\ \neg A(x)$ *true* instead. Such a proof may not yield the witness object t such that $A(t)$ is satisfied, which is required under the constructive interpretation of the existential quantifier. But how is it possible to provide witnesses in pure logic, without any assumptions about the domain of quantifiers? The answer is that assumptions about the existence of objects will be introduced locally during the proof. But we have to be careful to verify that the objects we use to witness existential quantifiers or instantiate universal quantifiers are indeed assumed to exist and are available at the right point in the derivation.

As a first concrete example, we investigate the interaction between negation and quantification. We prove

$$(\exists x{:}\tau.\ \neg A(x)) \supset \neg\forall x \in \tau.\ A(x)\ \textit{true}.$$

The subject of the judgment above is a proposition, assuming τ *type* and $x{:}\tau \vdash A(x)$ *prop*. Since all quantifiers range over the same type τ, we will omit the type label from quantification in all propositions below. The reader should keep in mind that this is merely a shorthand. Furthermore, we will not explicitly state the assumption about the propositional or predicate parameters such as $A(x)$ and omit the judgment *true* for the sake of brevity.

$$\dfrac{\dfrac{\dfrac{}{\exists x.\ \neg A(x)}\,u \qquad \dfrac{\dfrac{}{\neg A(c)}\,w \qquad \dfrac{\dfrac{}{\forall x.\ A(x)}\,v \quad \dfrac{}{c:\tau}}{A(c)}\,\forall E}{\bot}\,\supset E}{\bot}\,\exists E^{c,w}}{\neg\forall x.\ A(x)}\,\supset I^{v}}{(\exists x.\ \neg A(x)) \supset \neg\forall x.\ A(x)}\,\supset I^{u}$$

The two-dimensional notation for derivations becomes difficult to manage for large proofs, so we extend the linear notation used in the Tutch proof checker. We use the following concrete syntax.

$\forall x{:}\tau.\ A(x)$	`!x:t. A(x)`
$\exists x{:}\tau.\ A(x)$	`?x:t. A(x)`
$c:\tau$	`c : t`

The quantifiers $\forall$ and $\exists$ act like a prefix operator with minimal binding strength, so that

$$\forall x{:}\tau.\ A(x) \supset B$$

is the same as

$$\forall x{:}\tau.\ (A(x) \supset B).$$

One complication introduced by existential quantification is that the elimination rule introduces two new assumptions, $c : \tau$ and $A(c)$ *true*. In order to distinguish between inferred and assumed judgments, new assumptions are separated by commas and terminated by semicolon. Under these conventions, the four rules for quantification take the following form:

Introduction	Elimination

`c : t;`
`A(c);`
`?x:t. A(x);`
	`?x:t. A(x);`
	`[c : t, A(c);`
	`  ... ;`
	`B];`
	`B;`

`[c : t;`
`  ... ;`
`A(c)];`
`!x:t. A(x)`
	`!x:t. A(x);`
	`c : t;`
	`A(c);`

We use `c` as a new parameter to distinguish parameters more clearly from bound variables. Their confusion is a common source of error in first-order reasoning. And we have the usual assumption that the name chosen for c must be new (that is, may not occur in $A(x)$ or B) in the existential elimination and universal introduction rules.

Below we restate the proof from above in the linear notation.

```
[ ?x:t. ~A(x);
  [ !x:t. A(x);
    [ c : t, ~A(c);
      A(c);
      F ];
    F ];
  ~!x:t. A(x) ];
(?x:t. ~A(x)) => ~!x:t. A(x);
```

The opposite implication does not hold: even if we know that it is impossible that $A(x)$ is true for every x, this does not necessarily provide us with enough information to obtain a witness for $\exists x.\ A(x)$.

Now we return to showing that $(\neg\forall x.\ A(x)) \supset \exists x.\ \neg A(x)$ *true* is not derivable. We search for a normal proof, which means the first step in the bottom-up

construction is forced and we are in the state

$$\dfrac{\dfrac{}{\neg\forall x.\ A(x)\downarrow}\,u}{\begin{array}{c}\vdots\\ \exists x.\ \neg A(x)\uparrow\end{array}}{\scriptstyle\supset I^u}$$
$$(\neg\forall x.\ A(x))\supset\exists x.\ \neg A(x)\uparrow$$

At this point it is impossible to apply the existential introduction rule, because no witness object of type τ is available. So we can only apply the implication elimination rule, which leads us to the following situation.

$$\dfrac{\dfrac{\dfrac{\dfrac{}{\neg\forall x.\ A(x)\downarrow}\,u}{\begin{array}{c}\vdots\\ \forall x.\ A(x)\uparrow\end{array}}\quad\dfrac{}{\neg\forall x.\ A(x)\downarrow}\,u}{\bot\downarrow}{\scriptstyle\supset E}}{\dfrac{}{\exists x.\ \neg A(x)\uparrow}\,{\scriptstyle\bot E}}{\scriptstyle\supset I^u}$$
$$(\neg\forall x.\ A(x))\supset\exists x.\ \neg A(x)\uparrow$$

Now we can either repeat the negation elimination (which leads nowhere), or use universal introduction.

$$\dfrac{\dfrac{\dfrac{\dfrac{\dfrac{}{\neg\forall x.\ A(x)\downarrow}\,u\quad\dfrac{}{c:\tau}}{\begin{array}{c}\vdots\\ A(c)\uparrow\end{array}}}{\forall x.\ A(x)\uparrow}\,{\scriptstyle\forall I^c}\quad\dfrac{}{\neg\forall x.\ A(x)\downarrow}\,u}{\bot\downarrow}{\scriptstyle\supset E}}{\dfrac{}{\exists x.\ \neg A(x)\uparrow}\,{\scriptstyle\bot E}}{\scriptstyle\supset I^u}$$
$$(\neg\forall x.\ A(x))\supset\exists x.\ \neg A(x)\uparrow$$

The only applicable rule for constructing normal deductions now is again the implication elimination rule, applied to the assumption labeled u. This leads to the identical situation, except that we have an additional assumption $d:\tau$ and try to prove $A(d)\uparrow$. Clearly, we have made no progress. Therefore the given proposition has no normal proof and hence, by the completeness of normal proofs, no proof.

As a second example, we see that $(\forall x.\ A(x))\supset\exists x.\ A(x)$ *true* does not have a normal proof. After one forced step, we have to prove

$$\dfrac{\forall x.\ A(x)\downarrow}{\begin{array}{c}\vdots\\ \exists x.\ A(x)\uparrow\end{array}}$$

At this point, no rule is applicable, since we cannot construct *any* term of type τ. Intuitively, this should make sense: if the type τ is empty, then we cannot prove $\exists x{:}\tau.\ A(x)$ since we cannot provide a witness object. Since we make no assumptions about τ, τ may in fact denote an empty type, the above is clearly false.

In classical first-order logic, the assumption is often made that the domain of quantification is non-empty, in which case the implication above is true. In type theory, we can prove this implication for specific types that are known to be non-empty (such as `nat`). We can also model the standard assumption that the domain is non-empty by establishing the corresponding hypothetical judgment:

$$c : \tau \vdash (\forall x{:}\tau.\ A(x)) \supset \exists x{:}\tau.\ A(x)$$

We just give this simple proof in our linear notation.

```
[ c : t;
  [ !x:t. A(x);
    A(c);
    ?x:t. A(x) ];
  (!x:t. A(x)) => ?x:t. A(x)];
```

We can also discharge this assumption to verify that

$$\forall y.\ ((\forall x.\ A(x)) \supset \exists x.\ A(x))\ \mathit{true}$$

without any additional assumption. This shows that, in general, $\forall y.\ B$ is not equivalent to B, even if y does not occur in B! While this may be counterintuitive at first, the example above shows why it must be the case. The point is that while y does not occur in the *proposition*, it does occur in the *proof* and can therefore not be dropped.