4.4 Proof Irrelevance

So far, we have carefully designed the proof term language so that we can reconstruct the original deduction, maintaining a bijection. In many contexts this will give us too much information. Returning to one of our motivating examples,

$$\forall x{:}\mathsf{nat}.\ \exists y{:}\mathsf{nat}.\ y > x \wedge \mathsf{prime}(y)$$

we can see from what we have developed so far, that the computational content of a constructive proof of this proposition will be a function from a natural number x that returns a pair consisting of a witness p and a proof that $p > x \wedge \mathsf{prime}(p)$ which is again a pair of proofs. While it may be important for us to know that these last two proofs exist, in practice we may only be interested in the prime p, and not the proof that it is indeed greater than x or a prime number.

Our objective in this section is to find means to selectively hide portions of a proof. In one direction, this will allows us to extract only part of the computational content of a proof, making the resulting code much more efficient to execute. In the other direction, it will allow us write proof-agnostic functions and reason about them logically.

We will use all concepts and techniques we have developed so to achieve this goal. The basic idea is to introduce a new proposition $[A]$, pronounced "*bracket A*", for any proposition A. $[A]$ should be true if A is true, except that we intend to erase the proof before using it computationally. A first cut at the introduction rule would be

$$\frac{A\ \mathit{true}}{[A]\ \mathit{true}}\ []I?$$

The corresponding elimination rule

$$\frac{A\ \mathit{true}}{[A]\ \mathit{true}}\ []E?$$

is sound with respect to the introduction rule, but unfortunately fails to capture the intent of the type $[A]$. It says that if we have a proof of $[A]$ (which will be erased at runtime) then we can get a proof of A (which will be needed at runtime). It is evident that after erasing the proof of $[A]$ we would have to make up a proof of A out of thin air, which is not possible in general.

In order to capture this, we need a new judgment $A\ \mathit{irr}$ (pronounced "A *irrelevant*") which says that A is true but its proof is not computationally available. The revised elimination rule exploits this judgment to express that if we know $[A]\ \mathit{true}$ we know that A is true, but that we cannot use its proof computationally.

$$\frac{[A]\ \mathit{true} \quad \begin{array}{c}\dfrac{}{A\ \mathit{irr}}\ u\\ \vdots \\ C\ \mathit{true}\end{array}}{C\ \mathit{true}}\ []E^u$$

How do we use assumptions $A\ irr$? From the discussion above it should be clear that it is must remain too weak to prove $A\ true$: the latter requires a computationally relevant proof term but the former cannot offer one. However, when we are trying to prove $[C]\ true$, then we should be allowed to use assumptions $A\ irr$ because a proof of $[C]$ will be erased. The corresponding rule is a bit difficult to express in the two-dimensional format. It says that at the point we introduce $[C]$ we transform every assumption $A\ irr$ to a corresponding assumptione $A\ true$.

$$\dfrac{\dfrac{\overline{A_1\ irr}^{\,u_1}}{A_1\ true} \quad \cdots \quad \dfrac{\overline{A_n\ irr}^{\,u_n}}{A_n\ true}}{\begin{array}{c}\vdots\\ C\ true\end{array}}{}_{[]I}}{[C]\ true}$$

Before carrying out some examples, let us make sure that the rules are locally sound and complete. First, the local reduction which witnesses local soundness.

$$\dfrac{\dfrac{\dfrac{\overline{A_1\ irr}^{\,u_1}}{A_1\ true} \quad \cdots \quad \dfrac{\overline{A_n\ irr}^{\,u_n}}{A_n\ true}}{\begin{array}{c}\mathcal{D}\\ A\ true\end{array}}{}_{[]I}}{[A]\ true} \quad \dfrac{\overline{A\ irr}^{\,u}}{\begin{array}{c}\mathcal{E}\\ C\ true\end{array}}}{C\ true}{}_{[]E^u} \quad \Longrightarrow_R \quad \dfrac{\dfrac{\overline{A_1\ irr}^{\,u_1}}{A_1\ true} \quad \cdots \quad \dfrac{\overline{A_n\ irr}^{\,u_n}}{A_n\ true}}{\begin{array}{c}\mathcal{D}\\ A\ true\end{array}}{}_{u}}{\begin{array}{c}\mathcal{E}\\ C\ true\end{array}}$$

The operation on the right is again not well represented, visually. It requires a new substitution principle to substitute a deduction of $A\ true$ for assumptione $A\ irr$. This works because the assumption $A\ irr$ can only be used when it is promoted to an assumption $A\ true$, at which place we can use the ordinary substitution principle. We will see more formally how this works when we have proof terms below, and hypotheses are written in a localized form.

The local expansion is much simpler.

$$\begin{array}{c}\mathcal{D}\\ [A]\ true\end{array} \quad \Longrightarrow_E \quad \dfrac{\begin{array}{c}\mathcal{D}\\ [A]\ true\end{array} \quad \dfrac{\dfrac{\overline{A\ irr}^{\,u}}{A\ true}}{[A]\ true}{}_{[]I}}{[A]\ true}{}_{[]E^u}$$

Here, $A\ true$ is available to prove the premise of the $[]I$ rule because applying this rule promotes the assumption $A\ irr$.

We consider three examples. The first proves that $A \supset [A]$, that is, we can

forget computational content if we so choose.

$$\dfrac{\dfrac{\dfrac{}{A\ true}\,u}{[A]\ true}\,[\,]I}{A \supset [A]\ true}\,\supset I^u$$

The second one shows that we cannot prove $[A] \supset A$, that is, we cannot spanteously create computational content,

$$\dfrac{\dfrac{\dfrac{}{A\ true}\,u \qquad \begin{array}{c}\dfrac{}{A\ irr}\,w\\ \vdots \\ A\ true\end{array}}{A\ true}\,[\,]E^w}{[A] \supset A\ true}\,\supset I^u$$

Of course, this is not a convincing argument that we cannot prove $[A] \supset A$, but if we also knew that we can always find a proof by using introduction rules from below and elimination rules from above, then indeed there cannot be any proof.

Finally, we can distribute brackets over implication.

$$\dfrac{\dfrac{\dfrac{}{[A \supset B]\ true}\,u \qquad \dfrac{\dfrac{}{[A]\ true}\,w \qquad \dfrac{\dfrac{\dfrac{}{A \supset B\ irr}\,u' \qquad \dfrac{}{A\ irr}\,w'}{A \supset B\ true} \supset E}{B\ true}}{[B]\ true}\,[\,]I}{[B]\ true}\,[\,]E^{w'}}{[A] \supset [B]\ true}\,[\,]E^{u'}}{[A \supset B] \supset [A] \supset [B]\ true}\,\supset I^w}\,\supset I^u$$

Now we move on to proof terms. We write hypotheses in localized form, $\Gamma \vdash M : A$. There are two forms of hypothesis: $u{:}A$ which labels an assumption $A\ true$, and $[u]{:}A$, which labels an assumption $A\ irr$. The brackets around u indicate that it cannot be used directly, but only after a $[\,]$ introduction, which "unlocks" the variable $u{:}A$. In order to describe this process we define the process of *promotion*, written $\Gamma^{\oplus}$.

$$\begin{array}{lcl}(\cdot)^{\oplus} & = & \cdot \\ (\Gamma, u{:}A)^{\oplus} & = & \Gamma^{\oplus}, u{:}A \\ (\Gamma, [u]{:}A)^{\oplus} & = & \Gamma^{\oplus}, u{:}A\end{array}$$

We then have the following two rules.

$$\dfrac{\Gamma^{\oplus} \vdash M : A}{\Gamma \vdash [M] : [A]}\,[\,]I \qquad \dfrac{\Gamma \vdash M : [A] \quad \Gamma, [u]{:}A \vdash N : C}{\Gamma \vdash \mathbf{let}\ [u] = M\ \mathbf{in}\ N : C}\,[\,]E^u$$

What the hypothesis promotion achieves is, intuitively, that a variable $[u]{:}A$ can only be used inside brackets in a term M and not outside. For example:

$$\begin{array}{rcl} \lambda u.\ [u] & : & A \supset [A] \\ \lambda f.\ \lambda x.\ \mathbf{let}\ [f'] = f\ \mathbf{in}\ \mathbf{let}\ [x'] = x\ \mathbf{in}\ [f'\ x'] & : & [A \supset B] \supset [A] \supset [B] \end{array}$$

are well-typed. The first, because we are free to use u (which stands for A *true*) inside brackets, where it will not be used computationally, and the second because both f' and x' are used only inside brackets, where they will not be used computationally. On the other hand

$$\lambda u.\ \mathbf{let}\ [u'] = u\ \mathbf{in}\ u' \quad \not: \quad [A] \supset A$$

because u' is incorrectly used *outside* a bracket context.

Local reductions and expansions, as well as the substitution principle, are now much clearer on proof terms.

$$\textit{If}\ \Gamma^{\oplus} \vdash M : A\ \textit{and}\ \Gamma, [u] : A, \Gamma' \vdash N : C\ \textit{then}\ \Gamma, \Gamma' \vdash [M/u]N : C.$$

This is correct because the only place where $[u]{:}A$ can be used in the second deduction is underneath a $[\,]$ constructor where the context is promoted to $\Gamma^{\oplus}, \Gamma'^{\oplus}$ so that the ordinary substitution principle applies.

$$\begin{array}{lcl} \mathbf{let}\ [u] = [M]\ \mathbf{in}\ N & \Longrightarrow_R & [M/u]N \\ M : [A] & \Longrightarrow_E & \mathbf{let}\ [u] = M\ \mathbf{in}\ [u] \end{array}$$

We now go through a few more examples. In order to write these examples more compactly, we assume that every assumption $v{:}[A]$ is immediately decomposed into $[u]{:}A$, and, moreover, u is longer used. In that case, we can write $\lambda[u].\ M$ instead of $\lambda v.\ \mathbf{let}\ [u] = v\ \mathbf{in}\ M$.

First, we consider variants of the following proposition:

$$(\exists x.\ A(x) \wedge B(x)) \supset (\exists x.\ B(x) \wedge A(x))$$

The starting proof of this is straightforward:

$$\lambda u.\ \mathbf{let}\ \langle y, w\rangle = u\ \mathbf{in}\ \langle y, \langle \mathbf{snd}\, w, \mathbf{fst}\, w\rangle\rangle$$

Hiding in proof information in the output is consistent.

$$\lambda u.\ \mathbf{let}\ \langle y, w\rangle = u\ \mathbf{in}\ \langle y, [\langle \mathbf{snd}\, w, \mathbf{fst}\, w\rangle]\rangle : (\exists x.\ A(x) \wedge B(x)) \supset (\exists x.\ [B(x) \wedge A(x)])$$

We can still hide consistently even if the input proofs (of $A(x)$ and $B(x)$) are not available at runtime.

$$\lambda u.\ \mathbf{let}\ \langle y, [w]\rangle = u\ \mathbf{in}\ \langle y, [\langle \mathbf{snd}\, w, \mathbf{fst}\, w\rangle]\rangle : (\exists x.\ [A(x) \wedge B(x)]) \supset (\exists x.\ [B(x) \wedge A(x)])$$

This is correct because the only occurrences of w in its scope are inside brackets. Erasing w will therefore not lead to any dangling variables, that is, variables

that would be incorrectly assumed to be available at runtime. However, if we also hide the witness y then the result is no longer well-formed.

$$\lambda u.\ \mathbf{let}\ [\langle y, w\rangle] = u\ \mathbf{in}\ \langle y, [\langle \mathbf{snd}\, w, \mathbf{fst}\, w\rangle]\rangle \not: ([\exists x.\ A(x) \wedge B(x)]) \supset (\exists x.\ [B(x) \wedge A(x)])$$

The problem here is that y is bound inside the brackets but used outside. If we erased all the content inside the brackets we would have

$$\lambda u.\ \mathbf{let}\ [\,] = u\ \mathbf{in}\ \langle y, [\,]\rangle \not: [\,] \supset (\exists x.\ [\,])$$

which is not meaningful as a program even if we interpret $[\,]$ at $\top$ in the proposition and the unit element $\langle\,\rangle$ in the term.

It is even possible to be more fine-grained. For example, we can hide the proof of $A(x)$ in the antecedent of the implication of we also hide in the succedent.

Another set of examples comes from a specfication of graph reachability. We specify that for any two nodes x and y in a graph, either there is a path p connecting the two nodes or not.

$$\forall x.\ \forall y.\ (\exists p.\ \mathsf{path}(p, x, y)) \vee \neg(\exists p.\ \mathsf{path}(p, x, y))$$

A proof of this, when viewed computationally, will be a function taking nodes x and y as input and returning either $\mathbf{inl}\, M$ or $\mathbf{inr}\, N$. In the first case, a path exists, and M is a pair consisting of the path p and a proof that it connects x and y in the graph. In the second case, no path exists, and M is a function which derives a contradiction from any candidate path given to it as an argument.

One natural refinement of this specification is to hide the proof that p is indeed a valid path connecting x and y (which can easily verify this ourselves) and the proof that there is no path connecting x and y. For the latter, we would trust the proof (because, say, have verified it before erasing it). To capture this we would write

$$\forall x.\ \forall y.\ (\exists p.\ [\mathsf{path}(p, x, y)]) \vee [\neg(\exists p.\ \mathsf{path}(p, x, y))]$$

Now the function extracted from this proof would at runtime return either $\mathbf{inl}\, \langle p, [\,]\rangle$ (that is, the path connecting x and y) or $\mathbf{inr}\, [\,]$ to indicate there is no such path.

We can take this one step further, also hiding the path itself. In this case, the function returns either $\mathbf{inl}\, [\,]$ if there is a path connecting x and y and $\mathbf{inr}\, [\,]$ if there is none.

$$\forall x.\ \forall y.\ [\exists p.\ \mathsf{path}(p, x, y)] \vee [\neg(\exists p.\ \mathsf{path}(p, x, y))]$$

Summary. We present a summary of the rules for proof irrelevance, using the local form for hypothesis. We have a new form of judgment, $A\ \mathit{irr}$ which means that A is true, but the evidence for that is computationally irrelevant, that is,

may be erased before executing the program. We arrange that $A\ \mathit{irr}$ is only used as a hypothesis. We can *promote* such assumptions using the $\Delta^{\oplus}$ operation:

$$\begin{array}{lcl}(\cdot)^{\oplus} & = & \cdot \\ (\Delta, A\ \mathit{true})^{\oplus} & = & \Delta^{\oplus}, A\ \mathit{true} \\ (\Delta, A\ \mathit{irr})^{\oplus} & = & \Delta^{\oplus}, A\ \mathit{true}\end{array}$$

Then the introduction and elimination rules are

$$\frac{\Delta^{\oplus} \vdash A\ \mathit{true}}{\Delta \vdash [A]\ \mathit{true}}\ []I \qquad \frac{\Delta \vdash [A]\ \mathit{true} \quad \Delta, A\ \mathit{irr} \vdash C\ \mathit{true}}{\Delta \vdash C\ \mathit{true}}\ []E^u$$

In the presence of proof terms, we write $[u]{:}A$ as a labeling of the assumption $A\ \mathit{true}$. Then the promotion operation on labeled contexts Γ becomes

$$\begin{array}{lcl}(\cdot)^{\oplus} & = & \cdot \\ (\Gamma, u{:}A)^{\oplus} & = & \Gamma^{\oplus}, u{:}A \\ (\Gamma, [u]{:}A)^{\oplus} & = & \Gamma^{\oplus}, u{:}A\end{array}$$

With proof terms, the introduction and elimination rules then read as follows:

$$\frac{\Gamma^{\oplus} \vdash M : A}{\Gamma \vdash [M] : [A]}\ []I \qquad \frac{\Gamma \vdash M : [A] \quad \Gamma, [u]{:}A \vdash N : C}{\Gamma \vdash \mathbf{let}\ [u] = M\ \mathbf{in}\ N : C}\ []E^u$$