Lecture Notes on
The Curry-Howard Isomorphism

15-312: Foundations of Programming Languages
Frank Pfenning

Lecture 23
November 18, 2004

In this lecture we explore an interesting connection between logic and programming languages. In brief, logical proofs embody certain constructions which may be interpreted as programs. Under this interpretation, propositions become types. It was first observed by the logicians Haskell Curry [˜1960] and William Howard [˜1969] in different contexts that this is in fact an isomorphism: in a certain fragment of logic, every proof describes a program and every program describes a proof.

We will make the same observation here, using the methodology of judgments that we have used to far in the course, applied to the development of principles of logical reasoning. This formulation is due to Per Martin-Löf [˜1983].

What is the meaning of a proposition? Martin-Löf argues that to understand the meaning of a proposition means to understand when it is true. Consequently, in order to explain the meaning of a logical connective, we have to explain how to derive that a proposition with this connective is true.

Thus the most basic of all judgments of logic is that of truth, written as $A\ \ \mathsf{true}$.

As the simplest example of a connective, consider conjunction. We say that $A \wedge B\ \mathsf{true}$ if $A\ \mathsf{true}$ and $B\ \mathsf{true}$. Written as an inference rule:

$$\frac{A\ \mathsf{true} \quad B\ \mathsf{true}}{A \wedge B\ \mathsf{true}} \wedge I$$

We refer to this as an *introduction rule* because it introduces a connective in the conclusion (here '$\wedge$'). The introduction rule tells us how to conclude that $A \wedge B$ is true, thereby defining its meaning.

The next question is how can we use the information that $A \wedge B$ true. According to the above explanation, if we know $A \wedge B$ true, we should know the two premises, that is A true and B true.

$$\frac{A \wedge B \;\mathsf{true}}{A \;\mathsf{true}} \wedge E_1 \qquad \frac{A \wedge B \;\mathsf{true}}{B \;\mathsf{true}} \wedge E_2$$

We refer to these as *elimination rules*, because they eliminate a connective in the premise (here '$\wedge$').

If we think of the introduction rule as defining the meaning of the connective, how do we know that the elimination rules that we developed from it are actually correct? We will consider this question later in this lecture by introducing the notions of local soundness and completeness of the rules. For now, we continue by filling out the store of available logical connectives.

The next one we want to consider is implication. When is $A \supset B$ true? From our experience with proofs we know that in order to conclude $A \supset B$ true we assume that A is true and try to prove that B is true. If we want to take this as a definition, we need a hypothetical judgment. We write A_1 $\mathsf{true}, \ldots, A_n$ $\mathsf{true} \vdash A$ true for such a hypothetical judgment with assumptions A_1 true through A_n true. We abbreviate a collection of hypotheses with H. From the nature of reasoning from hypotheses we obtain for free a hypothesis rule and a substitution principle.

$$\frac{}{H_1, A \;\mathsf{true}, H_2 \vdash A \;\mathsf{true}} \; \mathit{Hyp}$$

Unfortunately, there is a small ambiguity here: there may be several hypotheses A true and from the form of the rule above we cannot tell which one was meant. In order to make this unambiguous we record the number of the assumption that was used.

Hypothesis rule.

$$\frac{}{A_1 \;\mathsf{true}, \ldots, A_i \;\mathsf{true}, \ldots, A_n \;\mathsf{true} \vdash A_i \;\mathsf{true}} \; \mathit{Hyp}^i$$

Substitution principle.

If $H_1 \vdash A$ true and H_1, A $\mathsf{true}, H_2 \vdash C$ true then $H_1, H_2 \vdash C$ true.

The latter is called a substitution principle because in order to obtain evidence for C true we substitute derivations of A true for uses of the assumption A true.

Now we have the concepts in place to be able to define implication by its introduction rule.

$$\frac{H, A\ \mathsf{true} \vdash B\ \mathsf{true}}{H \vdash A \supset B\ \mathsf{true}} \supset\! I$$

The elimination rule is based on the substitution principle. Assume we know that $A \supset B$ true. By the above rule this means B true under the assumption that A true. Now, if we had a proof of A true we could substitute it for uses of the assumption in the proof of B true. As a rule:

$$\frac{H \vdash A \supset B\ \mathsf{true} \quad H \vdash A\ \mathsf{true}}{H \vdash B\ \mathsf{true}} \supset\! E$$

Now we can prove, for example, that $(A \wedge B) \supset (B \wedge A)$ true.

$$\dfrac{\dfrac{\dfrac{\dfrac{}{A \wedge B\ \mathsf{true} \vdash A \wedge B\ \mathsf{true}}\ Hyp^1}{A \wedge B\ \mathsf{true} \vdash B\ \mathsf{true}}\ \wedge E_2 \quad \dfrac{\dfrac{}{A \wedge B\ \mathsf{true} \vdash A \wedge B\ \mathsf{true}}\ Hyp^1}{A \wedge B\ \mathsf{true} \vdash A\ \mathsf{true}}\ \wedge E_1}{A \wedge B\ \mathsf{true} \vdash B \wedge A\ \mathsf{true}}\ \wedge I}{\cdot \vdash (A \wedge B) \supset (B \wedge A)\ \mathsf{true}} \supset\! I$$

Note that this is holds for any propositions A and B, that is, it is a schematic derivation just like inference rules are schematic.

We continue our analysis of logical connectives with disjunction $A \vee B$. The disjunction is true if either of the disjuncts is true. This means we have two introduction rules.

$$\frac{H \vdash A\ \mathsf{true}}{H \vdash A \vee B\ \mathsf{true}} \vee I_1 \qquad \frac{H \vdash B\ \mathsf{true}}{H \vdash A \vee B\ \mathsf{true}} \vee I_2$$

In order to determine the elimination rule, we must consider how to use the knowledge that $A \vee B$ true. Clearly, we do not know which of A true or B true holds. This means if we are trying to prove C true and we know $A \vee B$ true, we must be able to show C true no matter whether A true or B true. In other words, we must proceed with a proof by cases. In the form of an elimination rule:

$$\frac{H \vdash A \vee B\ \mathsf{true} \quad H, A\ \mathsf{true} \vdash C\ \mathsf{true} \quad H, B\ \mathsf{true} \vdash C\ \mathsf{true}}{H \vdash C\ \mathsf{true}} \vee E$$

As a sample proof, consider the statement

If A or B implies C, then A implies C.

Formally:

$$((A \vee B) \supset C) \supset (A \supset C)\ \mathsf{true}$$

The proof:

$$\dfrac{\dfrac{\dfrac{}{(A \vee B) \supset C\ \mathsf{true}, A\ \mathsf{true} \vdash A\ \mathsf{true}}\ \mathit{Hyp}^2}{(A \vee B) \supset C\ \mathsf{true}, A\ \mathsf{true} \vdash A \vee B\ \mathsf{true}}\ \vee I_1 \qquad \dfrac{}{(A \vee B) \supset C\ \mathsf{true}, A\ \mathsf{true} \vdash (A \vee B) \supset C\ \mathsf{true}}\ \mathit{Hyp}^1}{\dfrac{(A \vee B) \supset C\ \mathsf{true}, A\ \mathsf{true} \vdash C\ \mathsf{true}}{\dfrac{(A \vee B) \supset C\ \mathsf{true} \vdash A \supset C\ \mathsf{true}}{\cdot \vdash ((A \vee B) \supset C) \supset (A \supset C)\ \mathsf{true}}\ \supset I}\ \supset I}\ \supset E$$

Next we look at some degenerate cases. Consider truth ($\top$) as a logical constant. It should be provable no matter what assumptions we have.

$$\dfrac{}{H \vdash \top\ \ \mathsf{true}}\ \top I$$

Because we put no information into the proof of $\top$, we can obtain no information out. Therefore, there is no elimination rule for $\top$. We can observe that $\top$ is like a 0-ary version of conjunction: $\wedge I$ has two premises and consequently we have two elimination rules ($\wedge E_1$ and $\wedge E_2$), while $\top I$ has no premises and consequently no elimination rules.

Now consider falsehood ($\bot$). It represents a contradiction and should therefore not be provable. In other words, there is no introduction rule. Conversely, if we know $\bot$ true we should be able to conclude anything.

$$\dfrac{H \vdash \bot\ \ \mathsf{true}}{H \vdash C\ \ \mathsf{true}}\ \bot E$$

We can recognize falsehood as a disjunction of zero alternatives. Whereas there are two introduction rules for $\vee$ and therefore two cases to consider in the elimination rule, there are no introduction rules for $\bot$ and therefore no branches in the elimination rule.

Figure 1 summarizes the rules, adding hypotheses to the first rules about conjunction in the straightforward way. We list the introduction rules in the left column and elimination rules in the right column. We have stacked the premises of the $\vee E$ rules purely for typographical reasons.

A natural question is if these are all the logical connectives we may be interested in, and if the given rules define logical reasoning completely if restricted to the considered connectives. If we ignore universal and existential quantification, then the main missing connectives are logical equiv-

$$\frac{}{A_1\ \mathsf{true}, \ldots, A_i\ \mathsf{true}, \ldots, A_n\ \mathsf{true} \vdash A_i\ \mathsf{true}}\ \mathit{Hyp}^i$$

$$\frac{H \vdash A\ \mathsf{true} \quad H \vdash B\ \mathsf{true}}{H \vdash A \wedge B\ \mathsf{true}}\ \wedge I \qquad \frac{H \vdash A \wedge B\ \mathsf{true}}{H \vdash A\ \mathsf{true}}\ \wedge E_1 \qquad \frac{H \vdash A \wedge B\ \mathsf{true}}{H \vdash B\ \mathsf{true}}\ \wedge E_2$$

$$\frac{}{H \vdash \top\ \ \mathsf{true}}\ \top I \qquad \text{no } \top E \text{ rule}$$

$$\frac{H, A\ \mathsf{true} \vdash B\ \mathsf{true}}{H \vdash A \supset B\ \mathsf{true}}\ \supset I \qquad \frac{H \vdash A \supset B\ \mathsf{true} \quad H \vdash A\ \mathsf{true}}{H \vdash B\ \mathsf{true}}\ \supset E$$

$$\frac{H \vdash A\ \mathsf{true}}{H \vdash A \vee B\ \mathsf{true}}\ \vee I_1 \qquad \frac{H \vdash B\ \mathsf{true}}{H \vdash A \vee B\ \mathsf{true}}\ \vee I_2 \qquad \frac{\begin{array}{l} H \vdash A \vee B\ \mathsf{true} \\ H, A\ \mathsf{true} \vdash C\ \mathsf{true} \\ H, B\ \mathsf{true} \vdash C\ \mathsf{true} \end{array}}{H \vdash C\ \mathsf{true}}\ \vee E$$

$$\text{no } \bot I \text{ rule} \qquad \frac{H \vdash \bot\ \mathsf{true}}{H \vdash C\ \mathsf{true}}\ \bot E$$

Figure 1: Rules of Intuitionistic Propositional Logic

alence $A \equiv B$ and negation $\neg A$. These can be easily considered abbreviations, using

$$\begin{array}{rcl} A \equiv B & = & (A \supset B) \wedge (B \supset A) \\ \neg A & = & A \supset \bot \end{array}$$

On the question of the completeness of these rules, a debate is possible. With the right proof-theoretic analysis we can show that $A \vee \neg A\ \mathsf{true}$ is *not* provable for an arbitrary A in this logic, essentially because we can prove neither $A\ \mathsf{true}$ nor $\neg A\ \mathsf{true}$, which are the two possibilities if we consider the introduction rules for disjunction ($\vee$).

The logic we have developed so far is, for historical reasons, called *intuitionistic logic*. If we also allow arbitrary instances of the axiom schema of *excluded middle* (*XM*),

$$\frac{}{H \vdash A \vee \neg A\ \mathsf{true}}\ \mathit{XM}$$

we obtain what is called *classical logic*. Note how classical logic, in a rule rather difficult to motivate, destroys the design principles and attempts at explaining the meaning of the connectives. For example, to understand the meaning of disjunction it is no longer sufficient to understand its introduction rules, but we must also understand the law of excluded middle, which contains and appeal to negation and falsehood. All is not lost, but we can say that the Curry-Howard isomorphism (the subject of this lecture whose explanation is yet to come) will fail in the presence of the law of excluded middle.

For the rest of this lecture we will only be interested in the intuitionistic logic as defined with the rules in Figure 1. The first observation is that a derivable judgment $H \vdash A\ \mathsf{true}$ does not contain any information about its derivation. When we assert $H \vdash A\ \mathsf{true}$ is derivable we mean that it has a derivation, but we do not exhibit such a derivation. This makes it difficult to convince someone else of the truth of A under assumptions H. So what we would like to do is to enrich the judgment with a *proof term* M which contains enough information to reconstruct the derivation.

So we would like to uniformly replace the judgment $A\ \mathsf{true}$ with $M : A$ (read: M *is a proof of* A). For assumptions $A\ \mathsf{true}$ we do not actually have a proof of A, we *assume* that there is a proof. We model this by using a *variable*, where each assumption is labeled by a distinct variable.

So we want to translate a judgment

$$A_1\ \mathsf{true}, \ldots, A_n\ \mathsf{true} \vdash A\ \mathsf{true}$$

to the form

$$x_1{:}A_1, \ldots, x_n{:}A_n \vdash M : A$$

in such a why that the derivation of the first judgment can be reconstructed directly from M. First, the hypothesis rule is straightforward:

$$\frac{}{A_1\ \mathsf{true}, \ldots, A_i\ \mathsf{true}, \ldots, A_n\ \mathsf{true} \vdash A_i\ \mathsf{true}}\ \mathit{Hyp}^i$$

becomes

$$\frac{}{x_1{:}A_1, \ldots, x_i{:}A_i, \ldots, x_n{:}A_n \vdash x_i : A_i}\ \mathit{Hyp}$$

Since the assumptions are labeled by distinct variables, we no longer need to annotate the justification with an integer and we just write *Hyp*.

The remaining rules mention a collection of hypotheses $H = (A_1\ \mathsf{true}, \ldots, A_n\ \mathsf{true})$ which we annotate uniformly with distinct variables, leading to a context $\Gamma = (x_1{:}A_1, \ldots, x_n{:}A_n)$.

We begin the logical connectives with conjunction. A proof of a conjunction $A \wedge B$ by the introduction rule $\wedge I$ consists of a pair of proofs, one of A and one for B.

$$\frac{\Gamma \vdash M : A \quad \Gamma \vdash N : B}{\Gamma \vdash \mathsf{pair}(M, N) : A \wedge B} \wedge I$$

We can recover the old rule by ignoring the proof terms, which immediately shows that the rule is sound with respect to the truth judgment. The two elimination rules can be considered as extracting components of this pair of proofs, which is why we use the suggesting names fst and snd.

$$\frac{\Gamma \vdash M : A \wedge B}{\Gamma \vdash \mathsf{fst}(M) : A} \wedge E_1 \qquad \frac{\Gamma \vdash M : A \wedge B}{\Gamma \vdash \mathsf{snd}(M) : B} \wedge E_2$$

The main observations of the Curry-Howard isomorphism should now already be visible:

1. (**Propositions-as-types**) Propositions of logic correspond to types of a programming language
2. (**Proofs-as-programs**) Proofs in logic correspond to expressions in a programming language
3. (**Proof-checking-as-type-checking**) Verifying the correctness of a proof corresponds to type-checking its corresponding expression.

We will consider later how computations are interpreted.

Now we go back to the logical connectives, considering implication. We need to account for the fact that the introduction rule ($\supset I$) introduces a new hypotheses. In the proof term this is handled as a binding construct. To make the reconstruction problem unambiguous we record A in the expression. A more standard notation for $\mathsf{fn}(A, x.M)$ would be $\lambda x{:}A.M$.

$$\frac{\Gamma, x{:}A \vdash M : B}{\Gamma \vdash \mathsf{fn}(A, x.M) : A \supset B} \supset I$$

In words: a proof of $A \supset B$ is a function which maps a proof of A to a proof of B. This is the functional interpretation of implication in intuitionistic logic. The elimination rule just applies such a function to an argument proof term.

$$\frac{\Gamma \vdash M : A \supset B \quad \Gamma \vdash N : A}{\Gamma \vdash \mathsf{apply}(M, N) : B} \supset E$$

In summary, logical implications corresponds to function types, analogous to the way that logical conjunctions correspond to product types.

It is not hard to guess that logical disjunction will correspond to disjoint sum types.

$$\frac{\Gamma \vdash M : A}{\Gamma \vdash \mathsf{inl}(B, M) : A \vee B}\ \vee I_1 \qquad \frac{\Gamma \vdash M : B}{\Gamma \vdash \mathsf{inr}(A, M) : A \vee B}\ \vee I_2$$

Proof by cases corresponds to the case construct over disjoint sums.

$$\frac{\Gamma \vdash M : A \vee B \quad \Gamma, x{:}A \vdash N : C \quad \Gamma, y{:}B \vdash P : C}{\Gamma \vdash \mathsf{case}(M, x.N, y.P) : C}\ \vee E$$

The logical constant $\top$ just becomes the unit type, and the logical constant $\bot$ the void (empty) type.

$$\frac{}{\Gamma \vdash \mathsf{unit} : \top}\ \top I$$

$$\frac{\Gamma \vdash M : \bot}{\Gamma \vdash \mathsf{abort}(C, M) : C}\ \bot E$$

The following table summarizes the correspondence between propositions and types under the Curry-Howard isomorphism:

Conjunction	$A \wedge B$	$\tau \times \sigma$	Product Type
Truth	$\top$	1	Unit Type
Implication	$A \supset B$	$\tau \rightarrow \sigma$	Function Type
Disjunction	$A \vee B$	$\tau + \sigma$	Sum Type
Falsehood	$\bot$	0	Void Type

As example, consider the following sample proofs, now written out in proof terms rather than full derivations. We have elided some (in this case, redundant) types[1] in the terms. As remarked in the lecture about bidirectional type-checking, if the term is *normal* and the type is known on the outside, then no internal type annotations are necessary.

$$\begin{array}{rcl} \mathsf{fn}(x.\mathsf{pair}(\mathsf{snd}(x), \mathsf{fst}(x))) & : & (A \wedge B) \supset (B \wedge A) \\ \mathsf{fn}(x.\mathsf{fn}(y.\mathsf{apply}(x, \mathsf{inl}(y)))) & : & ((A \vee B) \supset C) \supset (A \supset C) \end{array}$$

At this point it should be easy to see that we could actually let ML do some of the proof-checking for us. For example, with definitions

[1]that is, propositions

```
fun pair(x,y) = (x,y);
fun fst(x,y) = x;
fun snd(x,y) = y;
```

the first proof term above can be written as

```
- (fn x => pair (snd x, fst x));
val it = fn : 'a * 'b -> 'b * 'a
```

which constitutes a proof of $(A \wedge B) \supset (B \wedge A)$. The principal difficulty is that the presence of effects and recursion destroys the isomorphism. For example,

```
fun loop(x) = loop(x);
val loop = fn : 'a -> 'b
```

but the corresponding proposition, $A \supset B$ cannot be true in general. This means type-checking along in ML does not implement proof-checking; we also have to verify (by hand) the absence of effects and recursion.

We will not formalize this here, but it follows by straightforward inductions that for a derivation $\mathcal{D}$ of A_1 true$, \ldots, A_n$ true $\vdash A$ true we can systematically construct a derivation of $x_1{:}A_1, \ldots, x_n{:}A_n \vdash M : A$. Moreover, if $x_1{:}A_1, \ldots, x_n{:}A_n \vdash M : A$ then by erasure of terms (and appropriate labeling of the hypothesis rule) we can construct a derivation of A_1 true$, \ldots, A_n$ true $\vdash A$ true. These two translations are inverses of each other. In other words, the correspondence is really an ismorphism between proofs and programs.

It remains to consider what the role of *computation* is on the logical side. The whole construction seems too beautiful and elegant for the operational semantics of programs to be a simple accident without logical counterpart. In order to investigate this we return to the question of how to ascertain the "correctness" of the introduction and elimination rules for each connective. For example, it would clearly be unsound to have an elimination rule that allows us to infer A true from $A \vee B$ true. But how can we formally reject such an incorrect elimination rule?

In our context here we break down the correctness of the elimination rules with respect to the introduction rules into two questions: *local soundness* and *local completeness*.

Local soundness means that the elimination rules are not too strong. We have to verify that we cannot obtain more knowledge from a judgment by an elimination rule than we put into it by an introduction rule. More formally, we must show that if we introduce a connective and then eliminate it, we could derive the conclusion without this detour.

In the example of conjunction, this property is quite easy to see. We consider the possible combinations of introductions followed by eliminations, of which there are two.

$$\dfrac{\dfrac{\begin{array}{c}\mathcal{D}\\ A\ \mathsf{true}\end{array}\quad\begin{array}{c}\mathcal{E}\\ B\ \mathsf{true}\end{array}}{A\wedge B\ \mathsf{true}}\wedge I}{A\ \mathsf{true}}\wedge E_1 \qquad\qquad \dfrac{\dfrac{\begin{array}{c}\mathcal{D}\\ A\ \mathsf{true}\end{array}\quad\begin{array}{c}\mathcal{E}\\ B\ \mathsf{true}\end{array}}{A\wedge B\ \mathsf{true}}\wedge I}{B\ \mathsf{true}}\wedge E_2$$

In the first case, we can eliminate the detour because $\mathcal{D}$ is already a derivation of the conclusion, in the second case it is $\mathcal{E}$. We write this as *local reductions* on proofs that witness the local soundness of the rules.

$$\dfrac{\dfrac{\begin{array}{c}\mathcal{D}\\ A\ \mathsf{true}\end{array}\quad\begin{array}{c}\mathcal{E}\\ B\ \mathsf{true}\end{array}}{A\wedge B\ \mathsf{true}}\wedge I}{A\ \mathsf{true}}\wedge E_1 \qquad \mapsto \qquad \begin{array}{c}\mathcal{D}\\ A\ \mathsf{true}\end{array}$$

$$\dfrac{\dfrac{\begin{array}{c}\mathcal{D}\\ A\ \mathsf{true}\end{array}\quad\begin{array}{c}\mathcal{E}\\ B\ \mathsf{true}\end{array}}{A\wedge B\ \mathsf{true}}\wedge I}{B\ \mathsf{true}}\wedge E_2 \qquad \mapsto \qquad \begin{array}{c}\mathcal{E}\\ B\ \mathsf{true}\end{array}$$

If we annotate the derivations with proof terms we see that each local reduction is a *rule of computation* on proof terms.

$$\dfrac{\dfrac{\begin{array}{c}\mathcal{D}\\ M : A\end{array}\quad\begin{array}{c}\mathcal{E}\\ N : B\end{array}}{\mathsf{pair}(M,N) : A\wedge B}\wedge I}{\mathsf{fst}(\mathsf{pair}(M,N)) : A}\wedge E_1 \qquad \mapsto \qquad \begin{array}{c}\mathcal{D}\\ M : A\end{array}$$

$$\dfrac{\dfrac{\begin{array}{c}\mathcal{D}\\ M : A\end{array}\quad\begin{array}{c}\mathcal{E}\\ N : B\end{array}}{\mathsf{pair}(M,N) : A\wedge B}\wedge I}{\mathsf{snd}(\mathsf{pair}(M,N)) : B}\wedge E_2 \qquad \mapsto \qquad \begin{array}{c}\mathcal{E}\\ N : B\end{array}$$

Written out using only proof terms:

$$\begin{array}{rcl}\mathsf{fst}(\mathsf{pair}(M,N)) & \mapsto & M\\ \mathsf{snd}(\mathsf{pair}(M,N)) & \mapsto & N\end{array}$$

So proof reduction arises from showing that a "detour", that is, the introduction of a connective immediately followed by its elimination, can be avoided, leading to a "more direct" proof of the conclusion. The logically important property of this proof reduction is that it witnesses local soundness: we cannot get more information out of the truth of a proposition than we put into it.

Under the Curry-Howard isomorphism, computation then arises from a notion of proof reduction by imposing a particular strategy of reduction. For ML, this strategy is characterized by the search rules that specify where a reduction may take place. It seems that nothing about the logical meaning of a program forces the particular strategy adopted by ML, which means that the logical reading underdetermines how to evaluate programs but instead provides only the basic building blocks, namely the reductions.

To extend our analysis of proof reductions to implications, we need to consider substitution. Recall the substitution principle:

If $H_1 \vdash A\ \mathsf{true}$ and $H_1, A\ \mathsf{true}, H_2 \vdash C\ \mathsf{true}$ then $H_1, H_2 \vdash C\ \mathsf{true}$.

If we annotate this with proof terms we obtain:

If $\Gamma_1 \vdash M : A$ and $\Gamma_1, x{:}A, \Gamma_2 \vdash N : C$
then $\Gamma_1, \Gamma_2 \vdash \{M/x\}N : C$.

Now the pure proof reduction for an implication introduction followed by its elimination has the form

$$\dfrac{\dfrac{\overset{\mathcal{D}}{H, A\ \mathsf{true} \vdash B\ \mathsf{true}}}{H \vdash A \supset B\ \mathsf{true}}\supset\! I \qquad \overset{\mathcal{E}}{H \vdash A\ \mathsf{true}}}{H \vdash B\ \mathsf{true}}\supset\! E \quad \mapsto \quad \overset{\mathcal{D}'}{H \vdash B\ \mathsf{true}}$$

where the existence of $\mathcal{D}'$ is justified by the substitution property applied to $\mathcal{E}$ and $\mathcal{D}$. With proof terms:

$$\dfrac{\dfrac{\overset{\mathcal{D}}{\Gamma, x{:}A \vdash M : B}}{\Gamma \vdash \mathsf{fn}(A, x.M) : A \supset B}\supset\! I \qquad \overset{\mathcal{E}}{\Gamma \vdash N : A}}{\Gamma \vdash \mathsf{apply}(\mathsf{fn}(A, x.M), N) : B}\supset\! E \quad \mapsto \quad \overset{\mathcal{D}'}{\Gamma \vdash \{N/x\}M : B}$$

Written out using only proof terms:

$$\mathsf{apply}(\mathsf{fn}(A, x.M), N) \quad \mapsto \quad \{N/x\}M$$

For sums, we have to check two combinations of introduction followed by an elimination, because there are to rules for disjunction introduction. We leave it to the reader to write out the proof reduction that witnesses local soundness. The corresponding proof term reductions are

$$\begin{array}{rcl} \mathsf{case}(\mathsf{inl}(B, M), x.N, y.P) & \mapsto & \{M/x\}N \\ \mathsf{case}(\mathsf{inr}(A, M), x.N, y.P) & \mapsto & \{M/y\}P \end{array}$$

For truth ($\top$) and falsehood ($\bot$) no local reductions arise, because truth has only and introduction and falsehood only an elimination. Consequently, there are no reduction rules for the unit and void types, which is consistent with our definition of MinML.

One further remark regarding the connection between proof reductions and rules of computation. The fact that proof reductions transform one valid proof of $\Gamma \vdash M : A$ to another valid proof $\Gamma \vdash M' : A$ ensures *type preservation* for the corresponding computation rules in the programming languages.

There is a second check that is usually applied to the introduction and elimination rules for a connective to verify that the elimination rules are strong enough to recover all the information that has been put into a proposition. We have to verify that if we assume we have a proof of a proposition, we can apply elimination rules in such a way that we can reconstruct a proof of the original proposition by an introduction rule. We call this property *local completeness*, which is witnessed by a local expansion. However, local expansions do not have an immediate computational meaning, but are connected to the canonical forms property (also called value inversion). We do not explore this connection further and just show an example.

$$\begin{array}{c} \mathcal{D} \\ H \vdash A \wedge B\ \mathsf{true} \end{array} \quad \Longrightarrow \quad \dfrac{\dfrac{\begin{array}{c}\mathcal{D}\\ H \vdash A \wedge B\ \mathsf{true}\end{array}}{H \vdash A\ \mathsf{true}}\wedge E_1 \quad \dfrac{\begin{array}{c}\mathcal{D}\\ H \vdash A \wedge B\ \mathsf{true}\end{array}}{H \vdash B\ \mathsf{true}}\wedge E_2}{H \vdash A \wedge B\ \mathsf{true}}\wedge I$$

With proof terms:

$$\begin{array}{c} \mathcal{D} \\ \Gamma \vdash M : A \wedge B \end{array} \quad \Longrightarrow \quad \dfrac{\dfrac{\begin{array}{c}\mathcal{D}\\ \Gamma \vdash M : A \wedge B\end{array}}{\Gamma \vdash \mathsf{fst}(M) : A}\wedge E_1 \quad \dfrac{\begin{array}{c}\mathcal{D}\\ \Gamma \vdash M : A \wedge B\end{array}}{\Gamma \vdash \mathsf{snd}(M) : B}\wedge E_2}{\Gamma \vdash \mathsf{pair}(\mathsf{fst}(M), \mathsf{snd}(N)) : A \wedge B}\wedge I$$

Or purely on terms (indicating the type of the left-hand side)

$$M : A \wedge B \quad \Longrightarrow \quad \mathsf{pair}(\mathsf{fst}(M), \mathsf{snd}(M))$$