6.4 An alternative presentation of natural deduction

The above notation for natural deduction derivations suffers from a problem of presentation: since assumptions are first written down, later canceled dynamically, it is not easy to see when each assumption in a finished derivation was canceled.

The following alternate presentation of natural deduction works by deriving entire *judgments*, rather than *formulas*. Rather than keeping track of assumptions as the leaves of a proof tree, we annotate each formula in a derivation with the entire set of assumptions that were used in deriving it. In practice, this makes derivations more verbose, by repeating most assumptions on each line. In theory, however, such derivations are easier to reason about.

A *judgment* is a statement of the form $x_1{:}A_1, \ldots, x_n{:}A_n \vdash B$. It states that the formula B is a consequence of the (labeled) assumptions $A_1, \ldots, A_n$. The rules of natural deduction can now be reformulated as rules for deriving judgments:

1. (Axiom)
$$(ax_x)\ \frac{}{\Gamma, x{:}A \vdash A}$$

2. ($\wedge$-introduction)
$$(\wedge\text{-}I)\ \frac{\Gamma \vdash A \qquad \Gamma \vdash B}{\Gamma \vdash A \wedge B}$$

3. ($\wedge$-elimination)
$$(\wedge\text{-}E_1)\ \frac{\Gamma \vdash A \wedge B}{\Gamma \vdash A} \qquad (\wedge\text{-}E_2)\ \frac{\Gamma \vdash A \wedge B}{\Gamma \vdash B}$$

4. ($\top$-introduction)
$$(\top\text{-}I)\ \frac{}{\Gamma \vdash \top}$$

5. ($\to$-introduction)
$$(\to\text{-}I_x)\ \frac{\Gamma, x{:}A \vdash B}{\Gamma \vdash A \to B}$$

6. ($\to$-elimination)
$$(\to\text{-}E)\ \frac{\Gamma \vdash A \to B \qquad \Gamma \vdash A}{\Gamma \vdash B}$$

6.5 The Curry-Howard Isomorphism

There is an obvious one-to-one correspondence between types of the simply-typed lambda calculus and the formulas of propositional intuitionistic logic introduced in Section 6.3 (provided that the set of basic types can be identified with the set of atomic formulas). We will identify formulas and types from now on, where it is convenient to do so.

Perhaps less obvious is the fact that derivations are in one-to-one correspondence with simply-typed lambda terms. To be precise, we will give a translation from derivations to lambda terms, and a translation from lambda terms to derivations, which are mutually inverse up to α-equivalence.

To any derivation of $x_1{:}A_1, \ldots, x_n{:}A_n \vdash B$, we will associate a lambda term M such that $x_1{:}A_1, \ldots, x_n{:}A_n \vdash M : B$ is a valid typing judgment. We define M by recursion on the definition of derivations. We prove simultaneously, by induction, that $x_1{:}A_1, \ldots, x_n{:}A_n \vdash M : A$ is indeed a valid typing judgment.

1. (Axiom) If the derivation is
$$(ax_x)\ \frac{}{\Gamma, x{:}A \vdash A},$$
then the lambda term is $M = x$. Clearly, $\Gamma, x{:}A \vdash x : A$ is a valid typing judgment by (*var*).

2. ($\wedge$-introduction) If the derivation is
$$(\wedge\text{-}I)\ \frac{\begin{array}{c}\vdots\\ \Gamma \vdash A\end{array} \qquad \begin{array}{c}\vdots\\ \Gamma \vdash B\end{array}}{\Gamma \vdash A \wedge B},$$
then the lambda term is $M = \langle P, Q\rangle$, where P and Q are the terms associated to the two respective subderivations. By induction hypothesis, $\Gamma \vdash P : A$ and $\Gamma \vdash Q : B$, thus $\Gamma \vdash \langle P, Q\rangle : A \times B$ by (*pair*).

3. ($\wedge$-elimination) If the derivation is
$$(\wedge\text{-}E_1)\ \frac{\begin{array}{c}\vdots\\ \Gamma \vdash A \wedge B\end{array}}{\Gamma \vdash A},$$

then we let $M = \pi_1 P$, where P is the term associated to the subderivation. By induction hypothesis, $\Gamma \vdash P : A \times B$, thus $\Gamma \vdash \pi_1 P : A$ by (π_1). The case of ($\wedge$-*E*$_2$) is entirely symmetric.

4. ($\top$-introduction) If the derivation is

$$(\top\text{-}I)\ \frac{}{\Gamma \vdash \top},$$

then let $M = *$. We have $\vdash * : 1$ by $(*)$.

5. ($\to$-introduction) If the derivation is

$$(\to\text{-}I_x)\ \frac{\begin{array}{c}\vdots\\ \Gamma, x{:}A \vdash B\end{array}}{\Gamma \vdash A \to B},$$

then we let $M = \lambda x^A.P$, where P is the term associated to the subderivation. By induction hypothesis, $\Gamma, x{:}A \vdash P : B$, hence $\Gamma \vdash \lambda x^A.P : A \to B$ by (*abs*).

6. ($\to$-elimination) Finally, if the derivation is

$$(\to\text{-}E)\ \frac{\begin{array}{c}\vdots\\ \Gamma \vdash A \to B\end{array} \qquad \begin{array}{c}\vdots\\ \Gamma \vdash A\end{array}}{\Gamma \vdash B},$$

then we let $M = PQ$, where P and Q are the terms associated to the two respective subderivations. By induction hypothesis, $\Gamma \vdash P : A \to B$ and $\Gamma \vdash Q : A$, thus $\Gamma \vdash PQ : B$ by (*app*).

Conversely, given a well-typed lambda term M, with associated typing judgment $\Gamma \vdash M : A$, then we can construct a derivation of A from assumptions Γ. We define this derivation by recursion on the type derivation of $\Gamma \vdash M : A$. The details are too tedious to spell them out here; we simply go though each of the rules (*var*), (*app*), (*abs*), (*pair*), (π_1), (π_2), $(*)$ and apply the corresponding rule (*ax*), ($\to$-*I*), ($\to$-*E*), ($\wedge$-*I*), ($\wedge$-*E*$_1$), ($\wedge$-*E*$_2$), ($\top$-*I*), respectively.

6.6 Reductions in the simply-typed lambda calculus

β- and η-reductions in the simply-typed lambda calculus are defined much in the same way as for the untyped lambda calculus, except that we have introduced some additional terms (such as pairs and projections), which calls for some additional reduction rules. We define the following reductions:

$$\begin{array}{llcll}
(\beta_{\to}) & (\lambda x^A.M)N & \to & M[N/x], & \\
(\eta_{\to}) & \lambda x^A.Mx & \to & M, & \text{where } x \notin FV(M), \\
(\beta_{\times,1}) & \pi_1\langle M, N\rangle & \to & M, & \\
(\beta_{\times,2}) & \pi_2\langle M, N\rangle & \to & N, & \\
(\eta_{\times}) & \langle \pi_1 M, \pi_2 M\rangle & \to & M, & \\
(\eta_1) & M & \to & *, & \text{if } M : 1.
\end{array}$$

Then single- and multi-step β- and η-reduction are defined as the usual contextual closure of the above rules, and the definitions of β- and η-equivalence also follow the usual pattern. In addition to the usual (*cong*) and (ξ) rules, we now also have congruence rules that apply to pairs and projections.

We remark that, to be perfectly precise, we should have defined reductions between typing judgments, and not between terms. This is necessary because some of the reduction rules, notably (η_1), depend on the type of the terms involved. However, this would be notationally very cumbersome, and we will blur the distinction, pretending at times that terms appear in some implicit typing context that we do not write.

An important property of the reduction is the "subject reduction" property, which states that well-typed terms reduce only to well-typed terms of the same type. This has an immediate application to programming: subject reduction guarantees that if we write a program of type "integer", then the final result of evaluating the program, if any, will indeed be an integer, and not, say, a boolean.

Theorem 6.1 (Subject Reduction). *If $\Gamma \vdash M : A$ and $M \to_{\beta\eta} M'$, then $\Gamma \vdash M' : A$.*

Proof. By induction on the derivation of $M \to_{\beta\eta} M'$, and by case distinction on the last rule used in the derivation of $\Gamma \vdash M : A$. For instance, if $M \to_{\beta\eta} M'$ by $(\beta_{\to})$, then $M = (\lambda x^B.P)Q$ and $M' = P[Q/x]$. If $\Gamma \vdash M : A$, then we must have $\Gamma, x{:}B \vdash P : A$ and $\Gamma \vdash Q : B$. It follows that $\Gamma \vdash P[Q/x] : A$; the latter statement can be proved separately (as a "substitution lemma") by induction on P and makes crucial use of the fact that x and Q have the same type.

The other cases are similar, and we leave them as an exercise. Note that, in particular, one needs to consider the (*cong*), (ξ), and other congruence rules as well. □

6.7 A word on Church-Rosser

One important theorem that does *not* hold for $\beta\eta$-reduction in the simply-typed $\lambda^{\to,\times,1}$-calculus is the Church-Rosser theorem. The culprit is the rule (η_1). For instance, if x is a variable of type $A \times 1$, then the term $M = \langle \pi_1 x, \pi_2 x\rangle$ reduces to x by $(\eta_\times)$, but also to $\langle \pi_1 x, *\rangle$ by (η_1). Both these terms are normal forms. Thus, the Church-Rosser property fails.

$$\begin{tikzcd}
 & \langle \pi_1 x, \pi_2 x\rangle \arrow[dl, "\eta_\times"'] \arrow[dr, "\eta_1"] & \\
x & & \langle \pi_1 x, *\rangle
\end{tikzcd}$$

There are several ways around this problem. For instance, if we omit all the η-reductions and consider only β-reductions, then the Church-Rosser property does hold. Eliminating η-reductions does not have much of an effect on the lambda calculus from a computational point of view; already in the untyped lambda calculus, we noticed that all interesting calculations could in fact be carried out with β-reductions alone. We can say that β-reductions are the engine for computation, whereas η-reductions only serve to clean up the result. In particular, it can never happen that some η-reduction inhibits another β-reduction: if $M \to_\eta M'$, and if M' has a β-redex, then it must be the case that M already has a corresponding β-redex. Also, η-reductions always reduce the size of a term. It follows that if M is a β-normal form, then M can always be reduced to a $\beta\eta$-normal form (not necessarily unique) in a finite sequence of η-reductions.

Exercise 29. Prove the Church-Rosser theorem for β-reductions in the $\lambda^{\to,\times,1}$-calculus. Hint: use the same method which we used in the untyped case.

Another solution is to omit the type 1 and the term $*$ from the language. In this case, the Church-Rosser property holds even for $\beta\eta$-reduction.

Exercise 30. Prove the Church-Rosser theorem for $\beta\eta$-reduction in the $\lambda^{\to,\times}$-calculus, i.e., the simply-typed lambda calculus without 1 and $*$.

6.8 Reduction as proof simplification

Having made a one-to-one correspondence between simply-typed lambda terms and derivations in intuitionistic natural deduction, we may now ask what β- and η-reductions correspond to under this correspondence. It turns out that these reductions can be thought of as “proof simplification steps”.

Consider for example the β-reduction $\pi_1\langle M, N\rangle \to M$. If we translate the left-hand side and the right-hand side via the Curry-Howard isomorphism (here we use the first notation for natural deduction), we get

$$(\wedge\text{-}E_1)\,\dfrac{(\wedge\text{-}I)\,\dfrac{\begin{matrix}\Gamma\\ \vdots\\ A\end{matrix}\qquad \begin{matrix}\Gamma\\ \vdots\\ B\end{matrix}}{A\wedge B}}{A} \quad\to\quad \begin{matrix}\Gamma\\ \vdots\\ A\end{matrix}\ .$$

We can see that the left derivation contains an introduction rule immediately followed by an elimination rule. This leads to an obvious simplification if we replace the left derivation by the right one.

In general, β-redexes correspond to situations where an introduction rule is immediately followed by an elimination rule, and η-redexes correspond to situations where an elimination rule is immediately followed by an introduction rule. For example, consider the η-reduction $\langle \pi_1 M, \pi_2 M\rangle \to M$. This translates to:

$$(\wedge\text{-}I)\,\dfrac{(\wedge\text{-}E_1)\,\dfrac{\begin{matrix}\Gamma\\ \vdots\\ A\wedge B\end{matrix}}{A}\qquad (\wedge\text{-}E_2)\,\dfrac{\begin{matrix}\Gamma\\ \vdots\\ A\wedge B\end{matrix}}{B}}{A\wedge B} \quad\to\quad \begin{matrix}\Gamma\\ \vdots\\ A\wedge B\end{matrix}$$

Again, this is an obvious simplification step, but it has a side condition: the left and right subderivation must be the same! This side condition corresponds to the fact that in the redex $\langle \pi_1 M, \pi_2 M\rangle$, the two subterms called M must be equal. It is another characteristic of η-reductions that they often carry such side conditions.

The reduction $M \to *$ translates as follows:

$$\begin{matrix}\Gamma\\ \vdots\\ \top\end{matrix} \quad\to\quad (\top\text{-}I)\,\dfrac{}{\top}$$

In other words, any derivation of $\top$ can be replaced by the canonical such derivation.

More interesting is the case of the $(\beta_{\to})$ rule. Here, we have $(\lambda x^A.M)N \to M[N/x]$, which can be translated via the Curry-Howard Isomorphism as follows:

$$
(\to\text{-}E)\ \dfrac{(\to\text{-}I)\ \dfrac{\begin{array}{c}\Gamma,[x{:}A]\\ \vdots\\ B\end{array}}{A\to B}\,x \qquad \begin{array}{c}\Gamma\\ \vdots\\ A\end{array}}{B} \quad \to \quad \begin{array}{c}\Gamma\\ \vdots\\ \Gamma,\ A\\ \vdots\\ B\end{array}\ .
$$

What is going on here is that we have a derivation M of B from assumptions Γ and A, and we have another derivation N of A from Γ. We can directly obtain a derivation of B from Γ by stacking the second derivation on top of the first!

Notice that this last proof "simplification" step may not actually be a simplification. Namely, if the hypothesis labeled x is used many times in the derivation M, then N will have to be copied many times in the right-hand side term. This corresponds to the fact that if x occurs several times in M, then $M[N/x]$ might be a longer and more complicated term than $(\lambda x.M)N$.

Finally, consider the $(\eta_{\to})$ rule $\lambda x^A.Mx \to M$, where $x \notin FV(M)$. This translates to derivations as follows:

$$
(\to\text{-}I)\ \dfrac{(\to\text{-}E)\ \dfrac{\begin{array}{c}\Gamma\\ \vdots\\ A\to B\end{array} \qquad (ax)\ \dfrac{[x{:}A]}{A}\,x}{B}}{A\to B}\,x \quad \to \quad \begin{array}{c}\Gamma\\ \vdots\\ A\to B\end{array}
$$

6.9 Getting mileage out of the Curry-Howard isomorphism

The Curry-Howard isomorphism makes a connection between logic and the lambda calculus. We can think of it as a connection between "proofs" and "programs". What is such a connection good for? Like any isomorphism, it allows us to switch back and forth and think in whichever system suits our intuition in a given situation. Moreover, we can save a lot of work by transferring theorems that were proved about the lambda calculus to logic, and vice versa. As an example, we will see in the next section how to add disjunctions to propositional intuitionistic logic, and then we will explore what we can learn about the lambda calculus from that.

6.10 Disjunction and sum types

To the BNF for formulas of propositional intuitionistic logic from Section 6.3, we add the following clauses:

$$\text{Formulas:}\quad A,B ::= \ldots \ \big|\ A\vee B\ \big|\ \bot.$$

Here, $A \vee B$ stands for disjunction, or "or", and $\bot$ stands for falsity, which we can also think of as zero-ary disjunction. The symbol $\bot$ is also known by the names of "bottom", "absurdity", or "contradiction". The rules for constructing derivations are extended by the following cases:

7. ($\vee$-introduction)

$$(\vee\text{-}I_1)\ \frac{\Gamma\vdash A}{\Gamma\vdash A\vee B} \qquad (\vee\text{-}I_2)\ \frac{\Gamma\vdash B}{\Gamma\vdash A\vee B}$$

   In other words, if we have proven A or we have proven B, then we may conclude $A \vee B$.

8. ($\vee$-elimination)

$$(\vee\text{-}E_{x,y})\ \frac{\Gamma\vdash A\vee B \qquad \Gamma,x{:}A\vdash C \qquad \Gamma,y{:}B\vdash C}{\Gamma\vdash C}$$

   This is known as the "principle of case distinction". If we know $A \vee B$, and we wish to prove some formula C, then we may proceed by cases. In the first case, we assume A holds and prove C. In the second case, we assume B holds and prove C. In either case, we prove C, which therefore holds independently.

   Note that the $\vee$-elimination rule differs from all other rules we have considered so far, because it involves some arbitrary formula C that is not directly related to the principal formula $A \vee B$ being eliminated.

9. ($\bot$-elimination)

$$(\bot\text{-}E)\ \frac{\Gamma\vdash\bot}{\Gamma\vdash C},$$

   for an arbitrary formula C. This rule formalizes the familiar principle "ex falsum quodlibet", which means that falsity implies anything.

There is no $\bot$-introduction rule. This is symmetric to the fact that there is no $\top$-elimination rule.

Having extended our logic with disjunctions, we can now ask what these disjunctions correspond to under the Curry-Howard isomorphism. Naturally, we need to extend the lambda calculus by as many new terms as we have new rules in the logic. It turns out that disjunctions correspond to a concept which is quite natural in programming: "sum" or "union" types.

To the lambda calculus, add type constructors $A+B$ and 0.

$$\text{Simple types:} \quad A, B ::= \ldots \;\big|\; A+B \;\big|\; 0.$$

Intuitively, $A+B$ is the disjoint union of A and B, as in set theory: an element of $A+B$ is either an element of A or an element of B, together with an indication of which one is the case. In particular, if we consider an element of $A+A$, we can still tell whether it is in the left or right component, even though the two types are the same. In programming languages, this is sometimes known as a "union" or "variant" type. We call it a "sum" type here. The type 0 is simply the empty type, corresponding to the empty set in set theory.

What should the lambda terms be that go with these new types? We know from our experience with the Curry-Howard isomorphism that we have to have precisely one term constructor for each introduction or elimination rule of natural deduction. Moreover, we know that if such a rule has n subderivations, then our term constructor has to have n immediate subterms. We also know something about bound variables: Each time a hypothesis is canceled in a natural deduction rule, there must be a binder of the corresponding variable in the lambda calculus. This information more or less uniquely determines what the lambda terms should be; the only choice that is left is what to call them!

We add four terms to the lambda calculus:

$$\text{Raw terms:} \quad M, N ::= \ldots \;\big|\; \mathrm{in}_1 M \;\big|\; \mathrm{in}_2 M \;\big|\; \mathrm{case}\, M \,\mathrm{of}\, x^A \Rightarrow N \mid y^B \Rightarrow P \;\big|\; \Box_A M$$

The typing rules for these new terms are shown in Table 5. By comparing these rules to ($\vee$-I_1), ($\vee$-I_2), ($\vee$-E), and ($\bot$-E), you can see that they are precisely analogous.

But what is the meaning of these new terms? The term $\mathrm{in}_1 M$ is simply an element of the left component of $A+B$. We can think of in_1 as the injection function $A \to A+B$. Similar for in_2. The term $(\mathrm{case}\, M \,\mathrm{of}\, x^A \Rightarrow N \mid y^B \Rightarrow P)$ is a case distinction: evaluate M of type $A+B$. The answer is either an element of the left component A or of the right component B. In the first case, assign the answer to the variable x and evaluate N. In the second case, assign the answer to the variable y and evaluate P. Since both N and P are of type C, we get a final result of type C. Note that the case statement is very similar to an if-then-else; the only difference is that the two alternatives also carry a value. Indeed, the booleans can be defined as $1+1$, in which case $\mathbf{T} = \mathrm{in}_1 *$, $\mathbf{F} = \mathrm{in}_2 *$, and **if_then_else** $MNP = \mathrm{case}\, M \,\mathrm{of}\, x^1 \Rightarrow N \mid y^1 \Rightarrow P$, where x and y don't occur in N and P, respectively.

$$(\mathit{in}_1) \quad \frac{\Gamma \vdash M : A}{\Gamma \vdash \mathrm{in}_1 M : A+B}$$

$$(\mathit{in}_2) \quad \frac{\Gamma \vdash M : B}{\Gamma \vdash \mathrm{in}_2 M : A+B}$$

$$(\mathit{case}) \quad \frac{\Gamma \vdash M : A+B \qquad \Gamma, x{:}A \vdash N : C \qquad \Gamma, y{:}B \vdash P : C}{\Gamma \vdash (\mathrm{case}\, M \,\mathrm{of}\, x^A \Rightarrow N \mid y^B \Rightarrow P) : C}$$

$$(\Box) \quad \frac{\Gamma \vdash M : 0}{\Gamma \vdash \Box_A M : A}$$

Table 5: Typing rules for sums

Finally, the term $\Box_A M$ is a simple type cast, corresponding to the unique function $\Box_A : 0 \to A$ from the empty set to any set A.

6.11 Classical logic vs. intuitionistic logic

We have mentioned before that the natural deduction calculus we have presented corresponds to intuitionistic logic, and not classical logic. But what exactly is the difference? Well, the difference is that in intuitionistic logic, we have no rule for proof by contradiction, and we do not have $A \vee \neg A$ as an axiom.

Let us adopt the following convention for negation: the formula $\neg A$ ("not A") is regarded as an abbreviation for $A \to \bot$. This way, we do not have to introduce special formulas and rules for negation; we simply use the existing rules for $\to$ and $\bot$.

In intuitionistic logic, there is no derivation of $A \vee \neg A$, for general A. Or equivalently, in the simply-typed lambda calculus, there is no closed term of type $A + (A \to 0)$. We are not yet in a position to prove this formally, but informally, the argument goes as follows: If the type A is empty, then there can be no closed term of type A (otherwise A would have that term as an element). On the other hand, if the type A is non-empty, then there can be no closed term of type $A \to 0$ (or otherwise, if we applied that term to some element of A, we would obtain an element of 0). But if we were to write a *generic* term of type $A + (A \to 0)$, then this term would have to work no matter what A is. Thus, the term would have to decide whether to use the left or right component independently of A. But for any such term, we can get a contradiction by choosing A either empty or non-empty.

Closely related is the fact that in intuitionistic logic, we do not have a principle of proof by contradiction. The "proof by contradiction" rule is the following:

$$(\textit{contra}_x)\ \frac{\Gamma, x{:}\neg A \vdash \bot}{\Gamma \vdash A}.$$

This is *not* a rule of intuitionistic propositional logic, but we can explore what would happen if we were to add such a rule. First, we observe that the contradiction rule is very similar to the following:

$$\frac{\Gamma, x{:}A \vdash \bot}{\Gamma \vdash \neg A}.$$

However, since we defined $\neg A$ to be the same as $A \to \bot$, the latter rule is an instance of ($\to$-*I*). The contradiction rule, on the other hand, is not an instance of ($\to$-*I*).

If we admit the rule (*contra*), then $A \vee \neg A$ can be derived. The following is such a derivation:

$$(\textit{contra}_y)\ \dfrac{(\to\text{-}E)\ \dfrac{(ax_y)\ \dfrac{}{y{:}\neg(A\vee\neg A) \vdash \neg(A\vee\neg A)} \qquad (\vee\text{-}I_2)\ \dfrac{(\to\text{-}I_x)\ \dfrac{(\to\text{-}E)\ \dfrac{(ax_y)\ \dfrac{}{y{:}\neg(A\vee\neg A), x{:}A \vdash \neg(A\vee\neg A)} \qquad (\vee\text{-}I_2)\ \dfrac{(ax_x)\ \dfrac{}{y{:}\neg(A\vee\neg A), x{:}A \vdash A}}{y{:}\neg(A\vee\neg A), x{:}A \vdash A\vee\neg A}}{y{:}\neg(A\vee\neg A), x{:}A \vdash \bot}}{y{:}\neg(A\vee\neg A) \vdash \neg A}}{y{:}\neg(A\vee\neg A) \vdash A\vee\neg A}}{y{:}\neg(A\vee\neg A) \vdash \bot}}{\vdash A\vee\neg A}$$

Conversely, if we added $A \vee \neg A$ as an axiom to intuitionistic logic, then this already implies the (*contra*) rule. Namely, from any derivation of $\Gamma, x{:}\neg A \vdash \bot$, we can obtain a derivation of $\Gamma \vdash A$ by using $A \vee \neg A$ as an axiom. Thus, we can *simulate* the (*contra*) rule, in the presence of $A \vee \neg A$.

$$(\vee\text{-}E_{x,y})\ \dfrac{\dfrac{(\textit{excluded middle})}{\Gamma \vdash A\vee\neg A} \qquad (\bot\text{-}E)\ \dfrac{\Gamma, x{:}\neg A \vdash \bot}{\Gamma, x{:}\neg A \vdash A} \qquad (ax_y)\ \dfrac{}{\Gamma, y{:}A \vdash A}}{\Gamma \vdash A}$$

In this sense, we can say that the rule (*contra*) and the axiom $A \vee \neg A$ are equivalent, in the presence of the other axioms and rules of intuitionistic logic.

It turns out that the system of intuitionistic logic plus (*contra*) is equivalent to classical logic as we know it. It is in this sense that we can say that intuitionistic logic is "classical logic without proofs by contradiction".

Exercise 31. The formula $((A \to B) \to A) \to A$ is called "Peirce's law". It is valid in classical logic, but not in intuitionistic logic. Give a proof of Peirce's law in natural deduction, using the rule (*contra*).

Conversely, Peirce's law, when added to intuitionistic logic for all A and B, implies (*contra*). Here is the proof. Recall that $\neg A$ is an abbreviation for $A \to \bot$.

$$(\to\text{-}E)\ \dfrac{\dfrac{(\textit{Peirce's law for } B = \bot)}{\Gamma \vdash ((A\to\bot)\to A)\to A} \qquad (\to\text{-}I_x)\ \dfrac{(\bot\text{-}E)\ \dfrac{\Gamma, x{:}A\to\bot \vdash \bot}{\Gamma, x{:}A\to\bot \vdash A}}{\Gamma \vdash (A\to\bot)\to A}}{\Gamma \vdash A}$$

We summarize the results of this section in terms of a slogan:

$$\begin{array}{ll} & \text{intuitionistic logic + }(\textit{contra}) \\ = & \text{intuitionistic logic + ``}A \vee \neg A\text{''} \\ = & \text{intuitionistic logic + Peirce's law} \\ = & \text{classical logic.} \end{array}$$

The proof theory of intuitionistic logic is a very interesting subject in its own right, and an entire course could be taught just on that subject.

6.12 Classical logic and the Curry-Howard isomorphism

To extend the Curry-Howard isomorphism to classical logic, according to the observations of the previous section, it is sufficient to add to the lambda calculus a term representing Peirce's law. All we have to do is to add a term $\mathfrak{C} : ((A \to B) \to A) \to A$, for all types A and B.

Such a term is known as *Felleisen's* $\mathfrak{C}$, and it has a specific interpretation in terms of programming languages. It can be understood as a control operator (similar to "goto", "break", or exception handling in some procedural programming languages).