CSE-490 Logic in Computer Science
Course Notes

Sungwoo Park

Fall 2007

Draft of September 3, 2009

This document is in draft form and is likely to contain errors.
Please do not distribute this document outside class.

Preface

This is a collection of course notes for CSE-490 *Logic in Computer Science* at POSTECH. The material is largely based on course notes for *15-399 Constructive Logic* and *15-815 Automated Theorem Proving*, both by Frank Pfenning at Carnegie Mellon University.

Any comments and suggestions will be greatly appreciated. I especially welcome feedback from students as to which part is difficult to follow and which part needs to be improved. The less background you have in logic and proof theory, the more useful your comments will be. So please do not hesitate if you are taking this course!

Contents

Chapter 1

Inductive Definitions

This chapter discusses *inductive definitions* which are an indispensable tool in the study of programming languages. The reason why we need inductive definitions is not difficult to guess: a programming language may be thought of a system that is inhabited by *infinitely* many elements (or programs), and we wish to give a complete specification of it with a *finite* description; hence we need a mechanism of inductive definition by which a finite description is capable of yielding an infinite number of elements in the system. Those techniques related to inductive definitions also play a key role in investigating properties of programming languages. We will study these concepts with a few simple languages.

1.1 Inductive definitions of syntactic categories

An integral part of the definition of a programming language is its *syntax* which answers the question of which program (*i.e.*, a sequence of characters) is recognizable by the parser and which program is not. Typically the syntax is specified by a number of *syntactic categories* such as expressions, types, and patterns. Below we discuss how to define syntactic categories inductively in a few simple languages.

Our first example defines a syntactic category nat of natural numbers:

$$\mathsf{nat} \qquad n \; ::= \; \mathtt{O} \mid \mathtt{S}\, n$$

Here nat is the name of the syntactic category being defined, and n is called a *non-terminal*. We read ::= as *"is defined as"* and | as *"or."* $\mathtt{O}$ stands for "zero" and $\mathtt{S}$ "successor." Thus the above definition is interpreted as:

> A natural number n is either $\mathtt{O}$ or $\mathtt{S}\, n'$ where n' is another natural number.

Note that nat is defined inductively: a natural number $\mathtt{S}\, n'$ uses another natural number n', and thus nat uses the same syntactic category in its definition. Now the definition of nat produces an infinite collection of natural numbers such as

$$\mathtt{O},\ \mathtt{S}\,\mathtt{O},\ \mathtt{S}\,\mathtt{S}\,\mathtt{O},\ \mathtt{S}\,\mathtt{S}\,\mathtt{S}\,\mathtt{O},\ \mathtt{S}\,\mathtt{S}\,\mathtt{S}\,\mathtt{O},\ \cdots.$$

Thus nat specifies a language of natural numbers.

A syntactic category may refer to another syntactic category in its definition. For example, given the above definition of nat, the syntactic category tree below uses nat in its inductive definition:

$$\mathsf{tree} \qquad t \; ::= \; \mathtt{leaf}\; n \mid \mathtt{node}\; (t, n, t)$$

$\mathtt{leaf}\; n$ represents a leaf node with a natural number n; $\mathtt{node}\; (t_1, n, t_2)$ represents an internal node with a natural number n, a left child t_1, and a right child t_2. Then tree specifies a language of regular binary trees of natural numbers such as

$$\mathtt{leaf}\; n,\ \mathtt{node}\; (\mathtt{leaf}\; n_1, n, \mathtt{leaf}\; n_2),\ \mathtt{node}\; (\mathtt{node}\; (\mathtt{leaf}\; n_1, n, \mathtt{leaf}\; n_2), n', \mathtt{leaf}\; n''),\ \cdots.$$

A similar but intrinsically different example is two syntactic categories that are mutually inductively defined. For example, we simultaneously define two syntactic categories even and odd of even and odd numbers as follows:

$$\begin{array}{lrcl} \mathsf{even} & e & ::= & \mathtt{0} \mid \mathtt{S}\ o \\ \mathsf{odd} & o & ::= & \mathtt{S}\ e \end{array}$$

According to the definition above, even consists of even numbers such as

$$\mathtt{0},\ \mathtt{S}\ \mathtt{S}\ \mathtt{0},\ \mathtt{S}\ \mathtt{S}\ \mathtt{S}\ \mathtt{S}\ \mathtt{0},\ \cdots$$

whereas odd consists of odd numbers such as

$$\mathtt{S}\ \mathtt{0},\ \mathtt{S}\ \mathtt{S}\ \mathtt{S}\ \mathtt{0},\ \mathtt{S}\ \mathtt{S}\ \mathtt{S}\ \mathtt{S}\ \mathtt{S}\ \mathtt{0},\ \cdots.$$

Note that even and odd are *subcategories* of nat because every even number e or odd number o is also a natural number. Thus we may think of even and odd as nat satisfying certain properties.

Exercise 1.1. Define even and odd independently of each other.

Let us consider another example of defining a syntactic subcategory. First we define a syntactic category paren of strings of parentheses:

$$\begin{array}{lrcl} \mathsf{paren} & s & ::= & \epsilon \mid (s \mid)s \end{array}$$

ϵ stands for the empty string (*i.e.*, $\epsilon s = s = s\epsilon$). paren specifies a language of strings of parentheses with no constraint on the use of parentheses. Now we define a subcategory mparen of paren for those strings of matched parentheses:

$$\begin{array}{lrcl} \mathsf{mparen} & s & ::= & \epsilon \mid (s) \mid s\ s \end{array}$$

mparen generates such strings as

$$\epsilon\ ,\ ()\ ,\ ()()\ ,\ (())\ ,\ (())()\ ,\ ()()()\ ,\cdots.$$

mparen is ambiguous in the sense that a string belonging to mparen may not be decomposed in a unique way (according to the definition of mparen). For example, $()()()$ may be thought of as either $()()$ concatenated with $()$ or $()$ concatenated with $()()$. The culprit is the third case $s\ s$ in the definition: for a sequence of substrings of matched parentheses, there can be more than one way to split it into two substrings of matched parentheses. An alternative definition of lparen below eliminates ambiguity in mparen:

$$\begin{array}{lrcl} \mathsf{lparen} & s & ::= & \epsilon \mid (s)\ s \end{array}$$

The idea behind lparen is that the first parenthesis in a non-empty string s is a left parenthesis "(" which is paired with a unique occurrence of a right parenthesis ")". For example, $s = (())()$ can be written as $(s_1)s_2$ where $s_1 = ()$ and $s_2 = ()$, both strings of matched parentheses, are uniquely determined by s. $())$ and $(()()$, however, are not strings of matched parentheses and cannot be written as $(s_1)s_2$ where both s_1 and s_2 are strings of matched parentheses.

An inductive definition of a syntactic category is a convenient way to specify a language. Even the syntax of a full-scale programming language (such as SML) uses essentially the same machinery. It is, however, not the best choice for *investigating properties of languages.* For example, how can we formally express that n belongs to nat if $\mathtt{S}\ n$ belongs to nat, let alone prove it? Or how can we show that a string belonging to mparen indeed consists of matched parentheses? The notion of *judgment* comes into play to address such issues arising in inductive definitions.

1.2 Inductive definitions of judgments

A judgment is an object of knowledge, or simply a statement, that may or may not be provable. Here are a few examples:

- "$1 - 1$ *is equal to* 0" is a judgment which is always provable.

- "$1 + 1$ *is equal to* 0" is also a judgment which is never provable.
- "*It is raining*" is a judgment which is sometimes provable and sometimes not.
- "`S S O` *belongs to the syntactic category* `nat`" is a judgment which is provable if `nat` is defined as shown in the previous section.

Then how do we prove a judgment? For example, on what basis do we assert that "$1 - 1$ *is equal to* 0" is always provable? We implicitly use arithmetic to prove "$1 - 1$ *is equal to* 0", but strictly speaking, arithmetic rules are not given for free — we first have to reformulate them as *inference rules*.

An inference rule consists of premises and a conclusion, and is written in the following form (where J stands for a judgment):

$$\frac{J_1 \quad J_2 \quad \cdots \quad J_n}{J}\ R$$

The inference rule, whose name is R, states that if J_1 through J_n (premises) hold, then J (conclusion) also holds. As a special case, an inference rule with no premise (*i.e.*, $n = 0$) is called an *axiom*. Here are a few examples of inference rules and axioms where we omit their names:

$$\frac{m \textit{ is equal to } l \quad l \textit{ is equal to } n}{m \textit{ is equal to } n} \qquad \frac{m \textit{ is equal to } n}{m + 1 \textit{ is equal to } n + 1}$$

$$\frac{}{n \textit{ is equal to } n} \qquad \frac{}{0 \textit{ is a natural number}} \qquad \frac{\textit{My coat is wet}}{\textit{It is raining}}$$

Judgments are a general concept that covers any form of knowledge: knowledge about weather, knowledge about numbers, knowledge about programming languages, and so on. Note that judgments alone are inadequate to justify the knowledge being conveyed — we also need inference rules for proving or refuting judgments. In other words, the definition of a judgment is complete *only when there are inference rules for proving or refuting it*. Without inference rules, there can be no meaning in the judgment. For example, without arithmetic rules, the statement "$1-1$ *is equal to* 0" is nothing more than nonsense and thus cannot be called a judgment.

Needless to say, judgments are a concept strong enough to express membership in a syntactic category. As an example, let us recast the inductive definition of nat as a system of judgments and inference rules. We first introduce a judgment n nat:

$$n \ \mathsf{nat} \qquad \Leftrightarrow \qquad n \textit{ is a natural number}$$

We use the following two inference rules to prove the judgment n nat where their names, *Zero* and *Succ*, are displayed:

$$\frac{}{\mathsf{O}\ \mathsf{nat}}\ \textit{Zero} \qquad \frac{n\ \mathsf{nat}}{\mathsf{S}\ n\ \mathsf{nat}}\ \textit{Succ}$$

n in the rule *Succ* is called a *metavariable* which is just a placeholder for another sequence of `O` and `S` and is thus *not* part of the language consisting of `O` and `S`. That is, n is just a (meta)variable which ranges over the set of sequences of `O` and `S`; n itself (before being replaced by `S O`, for example) is not tested for membership in nat.

The notion of metavariable is similar to the notion of variable in SML. Consider an SML expression `x = 1` where `x` is a variable of type `int`. The expression makes sense only because we read `x` as a variable that ranges over integer values and is later to be replaced by an actual integer constant. If we literally read `x` as an (ill-formed) integer, `x = 1` would always evaluate to `false` because `x`, as an integer constant, is by no means equal to another integer constant `1`.

The judgment n nat is now defined inductively by the two inference rules. The rule *Zero* is a base case because it is an axiom, and the rule *Succ* is an inductive case because the premise contains a judgment smaller in size than the one (of the same kind) in the conclusion. Now we can prove, for example, that `S S O` nat holds with the following *derivation tree*, in which `S S O` nat is the root and `O` nat is the only leaf (*i.e.*, it is an inverted tree):

$$\cfrac{\cfrac{\cfrac{}{\mathsf{O}\ \mathsf{nat}}\ \textit{Zero}}{\mathsf{S}\ \mathsf{O}\ \mathsf{nat}}\ \textit{Succ}}{\mathsf{S}\ \mathsf{S}\ \mathsf{O}\ \mathsf{nat}}\ \textit{Succ}$$

Similarly we can rewrite the definition of the syntactic category tree in terms of judgments and inference rules:

$$t\ \mathsf{tree} \qquad \Leftrightarrow \qquad \textit{t is a regular binary tree of natural numbers}$$

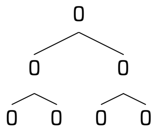
$$\frac{n\ \mathsf{nat}}{\mathtt{leaf}\ n\ \mathsf{tree}}\ \mathit{Leaf} \qquad \frac{t_1\ \mathsf{tree} \quad n\ \mathsf{nat} \quad t_2\ \mathsf{tree}}{\mathtt{node}\ (t_1, n, t_2)\ \mathsf{tree}}\ \mathit{Node}$$

A slightly more complicated example is a judgment that isolates full regular binary trees of natural numbers, as shown below. Note that there is no restriction on the form of judgment as long as its meaning is clarified by inference rules. We may even use English sentences as a valid form of judgment!

$$t\ \mathsf{ctree}\langle d\rangle \qquad \Leftrightarrow \qquad \textit{t is a full regular binary tree of natural numbers of depth } d$$

$$\frac{n\ \mathsf{nat}}{\mathtt{leaf}\ n\ \mathsf{ctree}\langle\mathtt{0}\rangle}\ \mathit{Cleaf} \qquad \frac{t_1\ \mathsf{ctree}\langle d\rangle \quad n\ \mathsf{nat} \quad t_2\ \mathsf{ctree}\langle d\rangle}{\mathtt{node}\ (t_1, n, t_2)\ \mathsf{ctree}\langle\mathtt{S}\ d\rangle}\ \mathit{Cnode}$$

The following derivation tree proves that



is a full regular binary tree of depth $\mathtt{S}\ \mathtt{S}\ \mathtt{0}$:

$$\dfrac{\dfrac{\dfrac{\dfrac{}{\mathtt{0}\ \mathsf{nat}}\ \mathit{Zero}}{\mathtt{leaf}\ \mathtt{0}\ \mathsf{ctree}\langle\mathtt{0}\rangle}\ \mathit{Cleaf} \quad \dfrac{}{\mathtt{0}\ \mathsf{nat}}\ \mathit{Zero} \quad \dfrac{\dfrac{}{\mathtt{0}\ \mathsf{nat}}\ \mathit{Zero}}{\mathtt{leaf}\ \mathtt{0}\ \mathsf{ctree}\langle\mathtt{0}\rangle}\ \mathit{Cleaf}}{\mathtt{node}\ (\mathtt{leaf}\ \mathtt{0}, \mathtt{0}, \mathtt{leaf}\ \mathtt{0})\ \mathsf{ctree}\langle\mathtt{S}\ \mathtt{0}\rangle}\ \mathit{Cnode} \quad \dfrac{}{\mathtt{0}\ \mathsf{nat}} \quad \mathit{(omitted)}}{\mathtt{node}\ (\mathtt{node}\ (\mathtt{leaf}\ \mathtt{0}, \mathtt{0}, \mathtt{leaf}\ \mathtt{0}), \mathtt{0}, \mathtt{node}\ (\mathtt{leaf}\ \mathtt{0}, \mathtt{0}, \mathtt{leaf}\ \mathtt{0}))\ \mathsf{ctree}\langle\mathtt{S}\ \mathtt{S}\ \mathtt{0}\rangle}\ \mathit{Cnode}$$

We can also show that $t = \mathtt{node}\ (\mathtt{leaf}\ \mathtt{0}, \mathtt{0}, \mathtt{node}\ (\mathtt{leaf}\ \mathtt{0}, \mathtt{0}, \mathtt{leaf}\ \mathtt{0}))$ is not a full regular binary tree as we cannot prove $t\ \mathsf{ctree}\langle d\rangle$ for any natural number d:

$$\dfrac{\dfrac{\dfrac{}{\mathtt{0}\ \mathsf{nat}} \quad d' = \mathtt{0}}{\mathtt{leaf}\ \mathtt{0}\ \mathsf{ctree}\langle d'\rangle}\ \mathit{Cleaf} \quad \dfrac{}{\mathtt{0}\ \mathsf{nat}}\ \mathit{Zero} \quad \dfrac{\cdots \quad d' = \mathtt{S}\ d''}{\mathtt{node}\ (\mathtt{leaf}\ \mathtt{0}, \mathtt{0}, \mathtt{leaf}\ \mathtt{0})\ \mathsf{ctree}\langle d'\rangle}\ \mathit{Cnode}}{\mathtt{node}\ (\mathtt{leaf}\ \mathtt{0}, \mathtt{0}, \mathtt{node}\ (\mathtt{leaf}\ \mathtt{0}, \mathtt{0}, \mathtt{leaf}\ \mathtt{0}))\ \mathsf{ctree}\langle\mathtt{S}\ d'\rangle}\ \mathit{Cnode}$$

It is easy to see why the proof fails: the left subtree of t requires $d' = \mathtt{0}$ while the right subtree of t requires $d' = \mathtt{S}\ d''$, and there is no way to solve two conflicting equations on d'.

As with the syntactic categories even and odd, multiple judgments can be defined simultaneously. For example, here is the translation of the definition of even and odd into judgments and inference rules:

$$\begin{array}{rcl} n\ \mathsf{even} & \Leftrightarrow & \textit{n is an even number} \\ n\ \mathsf{odd} & \Leftrightarrow & \textit{n is an odd number} \end{array}$$

$$\frac{}{\mathtt{0}\ \mathsf{even}}\ \mathit{ZeroE} \qquad \frac{n\ \mathsf{odd}}{\mathtt{S}\ n\ \mathsf{even}}\ \mathit{SuccE} \qquad \frac{n\ \mathsf{even}}{\mathtt{S}\ n\ \mathsf{odd}}\ \mathit{SuccO}$$

The following derivation tree proves that $\mathtt{S}\ \mathtt{S}\ \mathtt{0}$ is an even number:

$$\dfrac{\dfrac{\dfrac{}{\mathtt{0}\ \mathsf{even}}\ \mathit{ZeroE}}{\mathtt{S}\ \mathtt{0}\ \mathsf{odd}}\ \mathit{SuccO}}{\mathtt{S}\ \mathtt{S}\ \mathtt{0}\ \mathsf{even}}\ \mathit{SuccE}$$

Exercise 1.2. Translate the definition of paren, mparen, and lparen into judgments and inference rules.

1.3 Derivable rules and admissible rules

As shown in the previous section, judgments are defined with a certain (fixed) number of inference rules. When put together, these inference rules justify new inference rules which may in turn be added to the system. The new inference rules do *not* change the characteristics of the system because they can all be justified by the original inference rules, but may considerably facilitate the study of the system. For example, when multiplying two integers, we seldom employ the basic arithmetic rules, which can be thought of as original inference rules; instead we mostly use the rules of the multiplication table, which can be thought of as new inference rules.

There are two ways to introduce new inference rules: as *derivable rules* and as *admissible rules*. A derivable rule is one in which the gap between the premise and the conclusion can be bridged by a derivation tree. In other words, there always exists a sequence of inference rules that use the premise to prove the conclusion. As an example, consider the following inference rule which states that if n is a natural number, so is $\mathtt{S}\ \mathtt{S}\ n$:

$$\frac{n\ \mathsf{nat}}{\mathtt{S}\ \mathtt{S}\ n\ \mathsf{nat}}\ \mathit{Succ2}$$

The rule $\mathit{Succ2}$ is derivable because we can justify it with the following derivation tree:

$$\cfrac{\cfrac{n\ \mathsf{nat}}{\mathtt{S}\ n\ \mathsf{nat}}\ \mathit{Succ}}{\mathtt{S}\ \mathtt{S}\ n\ \mathsf{nat}}\ \mathit{Succ}$$

Now we may use the rule $\mathit{Succ2}$ as if it was an original inference rule; when asked to justify its use, we can just present the above derivation tree.

An admissible rule is one in which the premise implies the conclusion. That is, whenever the premise holds, so does the conclusion. A derivable rule is certainly an admissible rule because of the derivability of the conclusion from the premise. There are, however, admissible rules that are not derivable rules. (Otherwise why would we distinguish between derivable and admissible rules?) Consider the following inference rule which states that if $\mathtt{S}\ n$ is a natural number, so is n:

$$\frac{\mathtt{S}\ n\ \mathsf{nat}}{n\ \mathsf{nat}}\ \mathit{Succ}^{-1}$$

First observe that the rule Succ^{-1} is not derivable: the only way to derive $n\ \mathsf{nat}$ from $\mathtt{S}\ n\ \mathsf{nat}$ is by the rule Succ, but the premise of the rule Succ is smaller than its conclusion whereas $\mathtt{S}\ n\ \mathsf{nat}$ is larger than $n\ \mathsf{nat}$. That is, there is no derivation tree like

$$\cfrac{\cfrac{\mathtt{S}\ n\ \mathsf{nat}}{\vdots}\ \mathit{Succ}^{-1}}{n\ \mathsf{nat}}\ \mathit{Succ}^{-1}\ .$$

Now suppose that the premise $\mathtt{S}\ n\ \mathsf{nat}$ holds. Since the only way to prove $\mathtt{S}\ n\ \mathsf{nat}$ is by the rule Succ, $\mathtt{S}\ n\ \mathsf{nat}$ must have been derived from $n\ \mathsf{nat}$ as follows:

$$\cfrac{\begin{array}{c}\vdots\\ n\ \mathsf{nat}\end{array}}{\mathtt{S}\ n\ \mathsf{nat}}\ \mathit{Succ}$$

Then we can extract a smaller derivation tree $\begin{array}{c}\vdots\\ n\ \mathsf{nat}\end{array}$ which proves $n\ \mathsf{nat}$. Hence the rule Succ^{-1} is justified as an admissible rule.

An important property of derivable rules is that they remain valid even when the system is augmented with new inference rules. For example, the rule $\mathit{Succ2}$ remains valid no matter how many new inference rules are added to the system because the derivation of $\mathtt{S}\ \mathtt{S}\ n\ \mathsf{nat}$ from $n\ \mathsf{nat}$ is always possible thanks to the rule Succ (which is not removed from the system). In contrast, admissible rules may become invalid when new inference rules are introduced. For example, suppose that the system introduces a new (bizarre) inference rule:

$$\frac{n\ \mathsf{tree}}{\mathtt{S}\ n\ \mathsf{nat}}\ \mathit{Bizarre}$$

The rule *Bizarre* invalidates the previously admissible rule $Succ^{-1}$ because the rule $Succ$ is no longer the only way to prove $\mathsf{S}\ n$ nat and thus $\mathsf{S}\ n$ nat fails to guarantees n nat. Therefore the validity of an admissible rule must be checked each time a new inference rule is introduced.

Exercise 1.3. Is the rule $\dfrac{n \text{ even}}{\mathsf{S}\ \mathsf{S}\ n \text{ even}}\ SuccE^2$ derivable or admissible? What about the rule $\dfrac{\mathsf{S}\ \mathsf{S}\ n \text{ even}}{n \text{ even}}\ SuccE^{-2}$?

1.4 Inductive proofs

We have learned how to specify systems using inductive definitions of syntactic categories or judgments, or *inductive systems* of syntactic categories or judgments. While it is powerful enough to specify even full-scale programming languages (*i.e.*, their syntax and semantics), the mechanism of inductive definition alone is hardly useful unless the resultant system is shown to exhibit desired properties. That is, we cannot just specify a system using an inductive definition and then immediately use it without proving any interesting properties. For example, our intuition says that every string in the syntactic category mparen has the same number of left and right parentheses, but the definition of mparen itself does not automatically prove this property; hence we need to formally prove this property ourselves in order to use mparen as a language of strings of matched parentheses. As another example, consider the inductive definition of the judgments n even and n odd. The definition seems to make sense, but it still remains to formally prove that n in n even indeed represents an even number and n in n odd an odd number.

There is another important reason why we need to be able to prove properties of inductive systems. An inductive system is often so complex that its soundness, *i.e.* its definition being devoid of any inconsistencies, may not be obvious at all. In such a case, we usually set out to prove a property that is supposed to hold in the system. Then each flaw in the definition that destroys the property, if any, manifests itself at some point in the proof (because it is impossible to complete the proof). For example, an expression in a functional language is supposed to evaluate to a value of the same type, but this property (called *type preservation*) is usually not obvious at all. By attempting to prove type preservation, we can either locate flaws in the definition or partially ensure that the system is sound. Thus proving properties of an inductive system is the most effective aid in fixing errors in the definition.

First we will study a principle called *structural induction* for proving properties of inductive systems of syntactic categories. Next we will study another principle called *rule induction* for proving properties of inductive systems of judgments. Since an inductive system of syntactic category is a simplified presentation of a corresponding inductive system of judgments, structural induction is in fact a special case of rule induction. Nevertheless structural induction deserves separate treatment because of the role of syntactic categories in the study of programming languages.

1.4.1 Structural induction

The principle of structural induction states that a property of a syntactic category may be proven inductively by analyzing the structure of its definition: for each base case, we show that the property holds without making any assumption; for each inductive case, we first assume that the property holds for each smaller element in it and then prove the property holds for the entire case.

A couple of examples will clarify the concept. Consider the syntactic category nat of natural numbers. We wish to prove that $P(n)$ holds for every natural number n. Examples of $P(n)$ are:

- n has a successor.
- n is O or has a predecessor n' (*i.e.*, $\mathsf{S}\ n' = n$).
- n is a product of prime numbers (where definitions of products and prime numbers are assumed to be given).

By structural induction, we prove the following two statements:

- $P(\mathsf{O})$ holds.
- If $P(n)$ holds, then $P(\mathsf{S}\ n)$ also holds.

The first statement is concerned with the base case in which `0` has no smaller element in it; hence we prove $P(\texttt{0})$ without any assumption. The second statement is concerned with the inductive case in which $\texttt{S}\ n$ has a smaller element n in it; hence we first assume, as an *induction hypothesis*, that $P(n)$ holds and then prove that $P(\texttt{S}\ n)$ holds. The above instance of structural induction is essentially the same as the principle of mathematical induction.

As another example, consider the syntactic category tree of regular binary trees. In order to prove that $P(t)$ holds for every regular binary tree t, we need to prove the following two statements:

- $P(\texttt{leaf}\ n)$ holds.
- If $P(t_1)$ and $P(t_2)$ hold as induction hypotheses, then $P(\texttt{node}\ (t_1, n, t_2))$ also holds.

The above instance of structural induction is usually called *tree induction*.

As a concrete example of an inductive proof by structural induction, let us prove that every string belonging to the syntactic category mparen has the same number of left and right parentheses. (Note that we are not proving that mparen specifies a language of strings of matched parentheses.) We first define two auxiliary functions *left* and *right* to count the number of left and right parentheses. For visual clarity, we write $\mathit{left}[s]$ and $\mathit{right}[s]$ instead of $\mathit{left}(s)$ and $\mathit{right}(s)$. (We do not define *left* and *right* on the syntactic category paren because the purpose of this example is to illustrate structural induction rather than to prove an interesting property of mparen.)

$$
\begin{array}{rcl}
\mathit{left}[\epsilon] & = & 0 \\
\mathit{left}[(s)] & = & 1 + \mathit{left}[s] \\
\mathit{left}[s_1\ s_2] & = & \mathit{left}[s_1] + \mathit{left}[s_2] \\
\mathit{right}[\epsilon] & = & 0 \\
\mathit{right}[(s)] & = & 1 + \mathit{right}[s] \\
\mathit{right}[s_1\ s_2] & = & \mathit{right}[s_1] + \mathit{right}[s_2]
\end{array}
$$

Now let us interpret $P(s)$ as "$\mathit{left}[s] = \mathit{right}[s]$." Then we want to prove that if s belongs to mparen, written as $s \in \mathsf{mparen}$, then $P(s)$ holds.

Theorem 1.4. *If $s \in \mathsf{mparen}$, then $\mathit{left}[s] = \mathit{right}[s]$.*

Proof. By structural induction on s.

Each line below corresponds to a single step in the proof. It is written in the following format:

conclusion *justification*

This format makes it easy to read the proof because in most cases, we want to see the conclusion first rather than its justification.

Case $s = \epsilon$:
$\mathit{left}[\epsilon] = 0 = \mathit{right}[\epsilon]$

Case $s = (s')$:
$\mathit{left}[s'] = \mathit{right}[s']$ by induction hypothesis on s'
$\mathit{left}[s] = 1 + \mathit{left}[s'] = 1 + \mathit{right}[s'] = \mathit{right}[s]$ from $\mathit{left}[s'] = \mathit{right}[s']$

Case $s = s_1\ s_2$:
$\mathit{left}[s_1] = \mathit{right}[s_1]$ by induction hypothesis on s_1
$\mathit{left}[s_2] = \mathit{right}[s_2]$ by induction hypothesis on s_2
$\mathit{left}[s_1\ s_2] = \mathit{left}[s_1] + \mathit{left}[s_2] = \mathit{right}[s_1] + \mathit{right}[s_2] = \mathit{right}[s_1\ s_2]$
from $\mathit{left}[s_1] = \mathit{right}[s_1]$ and $\mathit{left}[s_2] = \mathit{right}[s_2]$
□

In the proof above, we may also say "by induction on the structure of s" instead of "by structural induction on s."

1.4.2 Rule induction

The principle of rule induction is similar to the principle of structural induction except that it is applied to derivation trees rather than definitions of syntactic categories. Consider an inductive definition of a judgment J with two inference rules:

$$\frac{}{J_b}\ R_{\mathsf{base}} \qquad \frac{J_1 \quad J_2 \quad \cdots \quad J_n}{J_i}\ R_{\mathsf{ind}}$$

We want to show that whenever J holds, another judgment $P(J)$ holds where $P(J)$ is a new form of judgment parameterized over J. For example, when J is "n nat", $P(J)$ may be "either n even or n odd." To this end, we prove the following two statements:

- $P(J_b)$ holds.
- If $P(J_1)$, $P(J_2)$, $\cdots$, and $P(J_n)$ hold as induction hypotheses, then $P(J_i)$ holds.

By virtue of the first statement, the following inference rule makes sense because we can always prove $P(J_b)$:

$$\frac{}{P(J_b)}\ R'_{\mathsf{base}}$$

The following inference rule also makes sense because of the second statement: it states that if $P(J_1)$ through $P(J_n)$ hold, then $P(J_i)$ also holds, which is precisely what the second statement proves:

$$\frac{P(J_1) \quad P(J_2) \quad \cdots \quad P(J_n)}{P(J_i)}\ R'_{\mathsf{ind}}$$

Now, for any derivation tree for J using the rules R_{base} and R_{ind}, we can prove $P(J)$ using the rules R'_{base} and R'_{ind}:

$$\frac{}{J_b}\ R_{\mathsf{base}} \quad \Longrightarrow \quad \frac{}{P(J_b)}\ R'_{\mathsf{base}}$$

$$\frac{\begin{matrix}\vdots \\ J_1\end{matrix} \quad \begin{matrix}\vdots \\ J_2\end{matrix} \quad \cdots \quad \begin{matrix}\vdots \\ J_n\end{matrix}}{J_i}\ R_{\mathsf{ind}} \quad \Longrightarrow \quad \frac{\begin{matrix}\vdots \\ P(J_1)\end{matrix} \quad \begin{matrix}\vdots \\ P(J_2)\end{matrix} \quad \cdots \quad \begin{matrix}\vdots \\ P(J_n)\end{matrix}}{P(J_i)}\ R'_{\mathsf{ind}}$$

In other words, J always implies $P(J)$. A generalization of the above strategy is the principle of rule induction.

As a trivial example, let us prove that n nat implies either n even or n odd. We let $P(n\ \mathsf{nat})$ be "either n even or n odd" and apply the principle of rule induction. The two rules *Zero* and *Succ* require us to prove the following two statements:

- $P(\mathtt{0}\ \mathsf{nat})$ holds. That is, for the case where the rule *Zero* is used to prove n nat, we have $n = \mathtt{0}$ and thus prove $P(\mathtt{0}\ \mathsf{nat})$.
- If $P(n'\ \mathsf{nat})$ holds, $P(\mathtt{S}\ n'\ \mathsf{nat})$ holds. That is, for the case where the rule *Succ* is used to prove n nat, we have $n = \mathtt{S}\ n'$ and thus prove $P(\mathtt{S}\ n'\ \mathsf{nat})$ using the induction hypothesis $P(n'\ \mathsf{nat})$.

According to the definition of $P(J)$, the two statements are equivalent to:

- Either $\mathtt{0}$ even or $\mathtt{0}$ odd holds.
- If either n' even or n' odd holds, then either $\mathtt{S}\ n'$ even or $\mathtt{S}\ n'$ odd holds.

A formal inductive proof proceeds as follows:

Theorem 1.5. *If n nat, then either n even or n odd.*

Proof. By rule induction on the judgment n nat.

It is of utmost importance that we apply the principle of rule induction to the *judgment* n nat rather than the natural number n. In other words, we analyze the structure of the proof of n nat, *not the structure of* n. If we analyze the structure of n, the proof degenerates to an example of structural induction! Hence we may also say "by induction on the structure of the proof of n nat" instead of "by rule induction on the judgment n nat."

Case $\dfrac{}{\mathsf{0\ nat}}\ Zero$ (where n happens to be equal to $\mathsf{0}$):
(This is the case where n nat is proven by applying the rule $Zero$. It is not obtained as a case where n is equal to 0, since we are not analyzing the structure of n. Note also that we do *not* apply the induction hypothesis because the premise has no judgment.)
0 even — by the rule $ZeroE$

Case $\dfrac{n'\ \mathsf{nat}}{\mathsf{S}\ n'\ \mathsf{nat}}\ Succ$ (where n happens to be equal to $\mathsf{S}\ n'$):
(This is the case where n nat is proven by applying the rule $Succ$.)
n' even or n' odd — by induction hypothesis
S n' odd or S n' even — by the rule $SuccO$ or $SuccE$

□

Rule induction can also be applied simultaneously to two or more judgments. As an example, let us prove that n in n even represents an even number and n in n odd an odd number. We use the rules $ZeroE$, $SuccE$, and $SuccO$ in Section 1.2 along with the following inference rules using a judgment n double n':

$$\frac{}{\mathsf{0\ double\ 0}}\ Dzero \qquad \frac{n\ \mathsf{double}\ n'}{\mathsf{S}\ n\ \mathsf{double\ S\ S}\ n'}\ Dsucc$$

Intuitively n double n' means that n' is a double of n (*i.e.*, $n' = 2 \times n$). The properties of even and odd numbers are stated in the following theorem:

Theorem 1.6.
If n even, *then there exists* n' *such that* n' double n.
If n odd, *then there exist* n' *and* n'' *such that* n' double n'' *and* S $n'' = n$.

The proof of the theorem follows the same pattern of rule induction as in previous examples except that $P(J)$ distinguishes between the two cases $J = n$ even and $J = n$ odd:

- $P(n$ even) is "there exists n' such that n' double n."
- $P(n$ odd) is "there exist n' and n'' such that n' double n'' and S $n'' = n$."

An inductive proof of the theorem proceeds as follows:

Proof of Theorem 1.6. By simultaneous rule induction on the judgments n even and n odd.

Case $\dfrac{}{\mathsf{0\ even}}\ ZeroE$ where $n = \mathsf{0}$:
0 double 0 — by the rule $Dzero$
We let $n' = \mathsf{0}$.

Case $\dfrac{n_p\ \mathsf{odd}}{\mathsf{S}\ n_p\ \mathsf{even}}\ SuccE$ where $n = \mathsf{S}\ n_p$:
n'_p double n''_p and S $n''_p = n_p$ — by induction hypothesis
S n'_p double S S n''_p — by the rule $Dsucc$ with n'_p double n''_p
S n'_p double n — from S S n''_p = S $n_p = n$
We let $n' = \mathsf{S}\ n'_p$.

Case $\dfrac{n_p\ \mathsf{even}}{\mathsf{S}\ n_p\ \mathsf{odd}}\ SuccO$ where $n = \mathsf{S}\ n_p$:
n'_p double n_p — by induction hypothesis
We let $n' = n'_p$ and $n'' = n_p$ — from $n = \mathsf{S}\ n_p$

□

1.5 Techniques for inductive proofs

An inductive proof is not always as straightforward as the proof of Theorem 1.5. For example, the theorem being proven may be simply false! In such a case, the proof attempt (which will eventually fail) may help us to extract a counterexample of the theorem. If the theorem is indeed provable (or is believed to be provable) but a direct proof attempt fails, we can try a common technique for inductive proofs. Below we illustrate three such techniques: introducing a lemma, generalizing the theorem, and proving by the principle of inversion.

1.5.1 Using a lemma

We recast the definition of the syntactic categories mparen and lparen as a system of judgments and inference rules:

$$\frac{}{\epsilon\ \mathsf{mparen}}\ Meps \quad \frac{s\ \mathsf{mparen}}{(s)\ \mathsf{mparen}}\ Mpar \quad \frac{s_1\ \mathsf{mparen} \quad s_2\ \mathsf{mparen}}{s_1\ s_2\ \mathsf{mparen}}\ Mseq$$

$$\frac{}{\epsilon\ \mathsf{lparen}}\ Leps \quad \frac{s_1\ \mathsf{lparen} \quad s_2\ \mathsf{lparen}}{(s_1)\ s_2\ \mathsf{lparen}}\ Lseq$$

Our goal is to show that s mparen implies s lparen. It turns out that a direct proof attempt by rule induction fails and that we need a lemma. To informally explain why we need a lemma, consider the case where the rule *Mseq* is used to prove s mparen. We may write $s = s_1\ s_2$ with s_1 mparen and s_2 mparen. By induction hypothesis on s_1 mparen and s_2 mparen, we may conclude s_1 lparen and s_2 lparen. From s_1 lparen, there are two subcases to consider:

- If $s_1 = \epsilon$, then $s = s_1\ s_2 = s_2$ and s_2 lparen implies s lparen.
- If $s_1 = (s_1')\ s_1''$ with s_1' lparen and s_1'' lparen, then $s = (s_1')\ s_1''\ s_2$.

In the second subcase, it is necessary to prove $s_1''\ s_2$ lparen from s_1'' lparen and s_2 lparen, which is not addressed by what is being proven (and is not obvious). Thus the following lemma needs to be proven first:

Lemma 1.7. *If s lparen and s' lparen, then $s\ s'$ lparen.*

Then how do we prove the above lemma by rule induction? The lemma does not seem to be provable by rule induction because it does not have the form *"If J holds, then $P(J)$ holds"* — the *If* part contains two judgments! It turns out, however, that rule induction can be applied exactly in the same way. The trick is to interpret the statement in the lemma as:

If s lparen, then s' lparen implies $s\ s'$ lparen.

Then we apply rule induction to the judgment s lparen with $P(s\ \mathsf{lparen})$ being "s' lparen implies $s\ s'$ lparen." An inductive proof of the lemma proceeds as follows:

Proof of Lemma 1.7. By rule induction on the judgment s lparen. Keep in mind that the induction hypothesis on s lparen yields "s' lparen implies $s\ s'$ lparen." Consequently, if s' lparen is already available as an assumption, the induction hypothesis on s lparen yields $s\ s'$ lparen.

Case $\frac{}{\epsilon\ \mathsf{lparen}}\ Leps$ where $s = \epsilon$:

s' lparen	assumption
$s\ s' = \epsilon\ s' = s'$	
$s\ s'$ lparen	from s' lparen

Case $\frac{s_1\ \mathsf{lparen} \quad s_2\ \mathsf{lparen}}{(s_1)\ s_2\ \mathsf{lparen}}\ Lseq$ where $s = (s_1)\ s_2$:

s' lparen	assumption
$s\ s' = (s_1)\ s_2\ s'$	
"s' lparen implies $s_2\ s'$ lparen"	by induction hypothesis on s_2 lparen

$s_2\ s'$ lparen — from the assumption s' lparen
$(s_1)\ s_2\ s'$ lparen — by the rule *Lseq* with s_1 lparen and $s_2\ s'$ lparen

□

Exercise 1.8. Can you prove Lemma 1.7 by rule induction on the judgment s' lparen?

Now we are ready to prove that s mparen implies s lparen.

Theorem 1.9. *If s mparen, then s lparen.*

Proof. By rule induction on the the judgment s mparen.

Case $\dfrac{}{\epsilon \text{ mparen}}\,Meps$ where $s = \epsilon$:
ϵ lparen — by the rule *Leps*

Case $\dfrac{s' \text{ mparen}}{(s') \text{ mparen}}\,Mpar$ where $s = (s')$:

s' lparen — by induction hypothesis

(s') lparen — from $\dfrac{s' \text{ lparen} \quad \dfrac{}{\epsilon \text{ lparen}}\,Leps}{(s')\ \epsilon \text{ lparen}}\,Lseq$ and $(s') = (s')\ \epsilon$

Case $\dfrac{s_1 \text{ mparen} \quad s_2 \text{ mparen}}{s_1\ s_2 \text{ mparen}}\,Mseq$ where $s = s_1\ s_2$:
s_1 lparen — by induction hypothesis on s_1 mparen
s_2 lparen — by induction hypothesis on s_2 mparen
$s_1\ s_2$ lparen — by Lemma 1.7

□

1.5.2 Generalizing a theorem

We have seen in Theorem 1.4 that if a string s belongs to the syntactic category mparen, or if s mparen holds, s has the same number of left and right parentheses, *i.e.*, $\mathit{left}[s] = \mathit{right}[s]$. The result, however, does not prove that s is a string of matched parentheses because it does not take into consideration positions of matching parentheses. For example, $s =)($ satisfies $\mathit{left}[s] = \mathit{right}[s]$, but is not a string of matched parentheses because the left parenthesis appears after its corresponding right parenthesis.

In order to be able to recognize strings of matched parentheses, we introduce a new judgment $k \rhd s$ where k is a non-negative integer:

$$\begin{array}{lcl} k \rhd s & \Leftrightarrow & \textit{k left parentheses concatenated with s form a string of matched parentheses} \\ & \Leftrightarrow & \underbrace{((\cdots(}_{k}\ \textit{s is a string of matched parentheses} \end{array}$$

The idea is that we scan a given string from left to right and keep counting the number of left parentheses that have not yet been matched with corresponding right parentheses. Thus we begin with $k = 0$, increment k each time a left parenthesis is encountered, and decrement k each time a right parenthesis is encountered:

$$\frac{}{0 \rhd \epsilon}\,Peps \qquad \frac{k+1 \rhd s}{k \rhd (s}\,Pleft \qquad \frac{k-1 \rhd s \quad k > 0}{k \rhd)s}\,Pright$$

The second premise $k > 0$ in the rule *Pright* ensures that in any prefix of a given string, the number of right parentheses may not exceed the number of left parentheses. Now a judgment $0 \rhd s$ expresses that s is a string of matched parentheses. Here are a couple of examples:

$$\dfrac{\dfrac{\dfrac{\dfrac{\dfrac{}{0 \rhd \epsilon}\,Peps \quad 1 > 0}{1 \rhd)}\,Pright \quad 2 > 0}{2 \rhd))}\,Pright}{1 \rhd ())}\,Pleft}{0 \rhd (())}\,Pleft \qquad \dfrac{\dfrac{\dfrac{(\textit{the rule Pright is not applicable because } 0 \not> 0)}{0 \rhd)(}}{1 \rhd))(}\,Pright}{0 \rhd ())(}\,Pleft$$

Note that while an inference rules is usually read from the premise to the conclusion, *i.e.*, "if the premise holds, then the conclusion follows," the above rules are best read from the conclusion to the premise: "in order to prove the conclusion, we prove the premise instead." For example, the rule $Peps$ may be read as "in order to prove $0 \rhd \epsilon$, we do not have to prove anything else," which implies that $0 \rhd \epsilon$ automatically holds; the rule $Pleft$ may be read as "in order to prove $k \rhd (s$, we only have to prove $k+1 \rhd s$." This bottom-up reading of the rules corresponds to the left-to-right direction of scanning a string. For example, a proof of $0 \rhd (())$ would proceed as the following sequence of judgments in which the given string is scanned from left to right:

$$0 \rhd (()) \quad \longrightarrow \quad 1 \rhd ()) \quad \longrightarrow \quad 2 \rhd)) \quad \longrightarrow \quad 1 \rhd) \quad \longrightarrow \quad 0 \rhd \epsilon$$

Exercise 1.10. Rewrite the inference rules for the judgment $k \rhd s$ so that they are best read from the premise to the conclusion.

Now we wish to prove that a string s satisfying $0 \rhd s$ indeed belongs to the syntactic category mparen:

Theorem 1.11. *If* $0 \rhd s$*, then* s mparen*.*

It is easy to see that a direct proof of Theorem 1.11 by rule induction fails. For example, when $0 \rhd (s$ follows from $1 \rhd s$ by the rule $Pleft$, we cannot apply the induction hypothesis to the premise because it does not have the form $0 \rhd s'$. What we need is, therefore, a generalization of Theorem 1.11 that covers all cases of the judgment $k \rhd s$ instead of a particular case $k = 0$:

Lemma 1.12. *If* $k \rhd s$*, then* $\underbrace{((\cdots(}_{k} s$ mparen*.*

Lemma 1.12 formally verifies the intuition behind the general form of the judgment $k \rhd s$. Then Theorem 1.11 is obtained as a corollary of Lemma 1.12.

The proof of Lemma 1.12 requires another lemma whose proof is left as an exercise (see Exercise 1.18):

Lemma 1.13. *If* $\underbrace{((\cdots(}_{k} s$ mparen*, then* $\underbrace{((\cdots((}_{k})s$ mparen*.*

Proof of Lemma 1.12. By rule induction on the judgment $k \rhd s$.

Case $\dfrac{}{0 \rhd \epsilon}\ Peps$ where $k = 0$ and $s = \epsilon$:

ϵ mparen — by the rule $Meps$

$\underbrace{((\cdots(}_{k} s$ mparen — from $\underbrace{((\cdots(}_{k} s = \epsilon$

Case $\dfrac{k+1 \rhd s'}{k \rhd (s'}\ Pleft$ where $s = (s'$:

$\underbrace{((\cdots(}_{k+1} s'$ mparen — by induction hypothesis on $k+1 \rhd s'$

$\underbrace{((\cdots(}_{k} s$ mparen — from $\underbrace{((\cdots(}_{k+1} s' = \underbrace{((\cdots(}_{k}(s' = \underbrace{((\cdots(}_{k} s$

Case $\dfrac{k-1 \rhd s' \quad k > 0}{k \rhd)s'}\ Pright$ where $s =)s'$:

$\underbrace{((\cdots(}_{k-1} s'$ mparen — by induction hypothesis on $k-1 \rhd s'$

$\underbrace{((\cdots((}_{k-1})s'$ mparen — by Lemma 1.13

$\underbrace{((\cdots(}_{k} s$ mparen — from $\underbrace{((\cdots((}_{k-1})s' = \underbrace{((\cdots(}_{k})s' = \underbrace{((\cdots(}_{k} s$

□

It is important that generalizing a theorem is different from introducing a lemma. We introduce a lemma when the induction hypothesis is applicable to all premises in an inductive proof, but the conclusion to be drawn is not a direct consequence of induction hypotheses. Typically such a lemma, which fills the gap between induction hypotheses and the conclusion, requires another inductive proof and is thus proven separately. In contrast, we generalize a theorem when the induction hypothesis is not applicable to some premises and an inductive proof does not even work. Introducing a lemma is to no avail here, since the induction hypothesis is applicable only to premises of inference rules and nothing else (*e.g.*, judgments proven by a lemma). Thus we generalize the theorem so that a direct inductive proof works. (The proof of the generalized theorem may require us to introduce a lemma, of course.)

To generalize a theorem is essentially to find a theorem that is harder to prove than, but immediately implies the original theorem. (In this regard, we can also say that we "strengthen" the theorem.) There is no particular recipe for generalizing a theorem, and some problem requires a deep insight into the judgment to which the induction hypothesis is to be applied. In many cases, however, identifying an invariant on the judgment under consideration gives a clue on how to generalize the theorem. For example, Theorem 1.11 deals with a special case of the judgment $k \rhd s$, and its generalization in Lemma 1.12 precisely expresses what the judgment $k \rhd s$ means.

1.5.3 Proof by the principle of inversion

Consider an inference rule $\dfrac{J_1 \quad J_2 \quad \cdots \quad J_n}{J}\ R$. In order to apply the rule R, we first have to establish proofs of all the premises J_1 through J_n, from which we may judge that the conclusion J also holds. An alternative way of reading the rule R is that in order to prove J, it suffices to prove $J_1, \cdots, J_n$. In either case, it is the premises, not the conclusion, that we have to prove first.

Now assume the existence of a proof of the conclusion J. That is, we assume that J is provable, but we may not have a concrete proof of it. Since the rule R is applied in the top-down direction, the existence of a proof of J does not license us to conclude that the premises $J_1, \cdots, J_n$ are also provable. For example, there may be another rule, say $\dfrac{J'_1 \quad J'_2 \quad \cdots \quad J_m}{J}\ R'$, that deduces the same conclusion, but using different premises. In this case, we cannot be certain that the rule R has been applied at the final step of the proof of J, and the existence of proofs of $J_1, \cdots, J_n$ is not guaranteed.

If, however, the rule R is the *only* way to prove the conclusion J, we may safely "invert" the rule R and deduce the premises $J_1, \cdots, J_n$ from the existence of a proof of J. That is, since the rule R is the only way to prove J, the existence of a proof of J is subject to the existence of proofs of all the premises of the rule R. Such a use of an inference rule in the bottom-up direction is called the *principle of inversion.*

As an example, let us prove that if $\mathtt{S}\ n$ is a natural number, so is n:

Proposition 1.14. *If* $\mathtt{S}\ n$ nat*, then* n nat*.*

We begin with an assumption that $\mathtt{S}\ n$ nat holds. Since the only way to prove $\mathtt{S}\ n$ nat is by the rule $Succ$, $\mathtt{S}\ n$ nat must have been derived from n nat by the principle of inversion:

$$\frac{n\ \mathsf{nat}}{\mathtt{S}\ n\ \mathsf{nat}}\ Succ$$

Thus there must be a proof of n nat whenever there exists a proof of $\mathtt{S}\ n$ nat, which completes the proof of Proposition 1.14.

1.6 Exercises

Exercise 1.15. Suppose that we represent a binary number as a sequence of digits $\mathbf{0}$ and $\mathbf{1}$. Give an inductive definition of a syntactic category bin for positive binary numbers without a leading $\mathbf{0}$. For example, $\mathbf{10}$ belongs to bin whereas $\mathbf{00}$ does not. Then define a function num which takes a sequence b belonging to bin and returns its corresponding decimal number. For example, we have $num(\mathbf{10}) = 2$ and $num(\mathbf{110}) = 6$. You may use ϵ for the empty sequence.

Exercise 1.16. Prove the converse of Theorem 1.9: if s lparen, then s mparen.

Exercise 1.17. Given a judgment t tree, we define two functions $numLeaf(t)$ and $numNode(t)$ for calculating the number of leaves and the number of nodes in t, respectively:

$$\begin{array}{rcl} numLeaf(\texttt{leaf}) & = & 1 \\ numLeaf(\texttt{node}\ (t_1, n, t_2)) & = & numLeaf(t_1) + numLeaf(t_2) \\ numNode(\texttt{leaf}) & = & 0 \\ numNode(\texttt{node}\ (t_1, n, t_2)) & = & numNode(t_1) + numNode(t_2) + 1 \end{array}$$

Use rule induction to prove that if t tree, then $numLeaf(t) - numNode(t) = 1$.

Exercise 1.18. Prove a lemma: if $\underbrace{((\cdots(}_{k}\ s$ lparen, then $\underbrace{((\cdots(}_{k}()s$ lparen. Use this lemma to prove Lemma 1.13. Your proof needs to exploit the equivalence between s mparen and s lparen as stated in Theorem 1.9 and Exercise 1.16.

Exercise 1.19. Proof the converse of Theorem 1.11: if s mparen, then $0 \rhd s$.

Exercise 1.20. Consider an SML implementation of the factorial function:

```
fun fact' 0 a = a
  | fact' n a = fact' (n - 1) (n * a)
fun fact n = fact' n 1
```

We wish to prove that `fact` $\widehat{n}$ evaluates to $\widehat{n!}$ by mathematical induction on $n \geq 0$, where $\widehat{n}$ stands for an SML constant expression for a mathematical integer n. Since `fact` $\widehat{n}$ reduces to `fact'` $\widehat{n}$ `1`, we try to prove a lemma that `fact'` $\widehat{n}$ `1` evaluates to $\widehat{n!}$. Unfortunately it is impossible to prove the lemma by mathematical induction on n. How would you generalize the lemma so that mathematical induction works on n?

Exercise 1.21. The principle of mathematical induction states that for any natural number n, a judgment $P(n)$ holds if the following two conditions are met:

1. $P(0)$ holds.
2. $P(k)$ implies $P(k+1)$ where $k \geq 0$.

There is another principle, called *complete induction*, which allows stronger assumptions in proving $P(k+1)$:

1. $P(0)$ holds.
2. $P(0), P(1), \cdots, P(k)$ imply $P(k+1)$ where $k \geq 0$.

It turns out that complete induction is not a new principle; rather it is a derived principle which can be justified by the principle of mathematical induction. Use mathematical induction to show that if the two conditions for complete induction are met, $P(n)$ holds for any natural number n.

Exercise 1.22. Consider the following inference rules for comparing two natural number for equality:

$$\frac{}{\mathsf{0} \doteq \mathsf{0}}\ EqZero \qquad \frac{n \doteq m}{\mathsf{S}\ n \doteq \mathsf{S}\ m}\ EqSucc$$

Show that the following inference rule is admissible:

$$\frac{n \doteq m \quad n \ \mathsf{double}\ n' \quad m\ \mathsf{double}\ m'}{n' \doteq m'}\ EqDouble$$

Exercise 1.23. Consider yet another inductive definition of strings of matched parentheses where we use a new judgment s tparen:

$$\frac{}{\epsilon\ \mathsf{tparen}}\ Teps \qquad \frac{s_1\ \mathsf{tparen} \quad s_2\ \mathsf{tparen}}{s_1\ (s_2)\ \mathsf{tparen}}\ Tseq$$

Give a proof of the following lemma:

Lemma 1.24. *If* s tparen *and* s' tparen, *then* $s\ s'$ tparen.

Use Lemma 1.24 to prove the following theorem:

Theorem 1.25. *If* s mparen, *then* s tparen.

In conjunction with the result from Exercise 1.16, Theorem 1.25 proves that s lparen implies s tparen. Now we wish to prove this result directly, *i.e.*, without using the judgment s mparen:

Theorem 1.26. *If* s lparen, *then* s tparen.

Try to prove Theorem 1.26 by rule induction on s lparen. If you can complete the proof, write it. If you need a lemma to complete the proof, state the lemma, prove it, and use it to complete the proof.

Exercise 1.27. Consider two SML implementations of the Fibonacci function. `fib1` is an ordinary recursive function whereas `fib2` is a tail-recursive function. All arguments to both functions are assumed to be natural numbers (which include zero).

```
fun fib1 0 = 0
  | fib1 1 = 1
  | fib1 n = fib (n - 1) + fib (n - 2)        // n ≥ 2

fun fib2 m n 0 = m
  | fib2 m n p = fib2 n (m + n) (p - 1)       // p ≥ 1
```

Use mathematical induction or complete induction to prove the following properties. We assume that every variable (p, z, m, n) ranges over natural numbers.

- `fib2 (fib1` p`) (fib1 (`p `+ 1))` z `= fib1 (`z `+` p`).`
- `fib2` m n `(`p `+ 1) =` m `* fib1` p `+` n `* fib1 (`p `+ 1).`
- `fib1 (2 *` p`) + fib1` p `* fib1` p `= 2 * fib1` p `* fib1 (`p `+ 1).`
- `fib1 (2 *` p `+ 1) = fib1` p `* fib1` p `+ fib1 (`p `+ 1) * fib1 (`p `+ 1).`

Chapter 2

Propositional Logic

This chapter develops *propositional logic*, *i.e.*, logic without universal or existential quantifications. We formulate propositional logic in the judgmental style of Pfenning and Davies [**?**], which adopts Martin-Löf's methodology of distinguishing between *propositions* and *judgments* [**?**]. It differs from the traditional style of formulating logic which relies solely on propositions.

2.1 Propositions and judgments

In a judgmental formulation of logic, a proposition is an object of verification whose *truth* can be checked by inference rules, whereas a judgment is an object of knowledge which becomes evident by a *proof*. Examples of propositions are *'1 + 1 is equal to 0'* and *'1 + 1 is equal to 2'*, both under inference rules based on arithmetic. Examples of judgments are *"'1 + 1 is equal to 0' is true"*, for which there is no proof, and *"'1 + 1 is equal to 2' is true,"* for which there is a proof.

To clarify the difference between propositions and judgments, consider a statement *'the moon is made of cheese.'* The statement is not yet an object of verification, or a proposition, since there is no way to check its truth — it becomes a proposition only when an inference rule is given. Here is an example of such an inference rule (written in a pedantic way):

$$\frac{\textit{`the moon is greenish white and has holes in it' is true}}{\textit{`the moon is made of cheese' is true}}\ \mathit{MoonCheese}$$

Now we can attempt to verify the proposition, for example, by taking a picture of the moon. That is, we still do not know whether the proposition is true or not, but by virtue of the inference rule, we know at least what counts as a verification of it. If the picture indeed shows that the moon is greenish white and has holes in it, the inference rule makes evident the judgment *"'the moon is made of cheese' is true."* Now we know *"'the moon is made of cheese' is true"* by the proof consisting of the picture and the inference rule. Thus a proposition is an object of verification which may or may not be true, whereas a judgment is an object of knowledge which we either know or do not know, depending on the existence of a proof.

It is important that the notion of judgment takes priority over the notion of proposition. Simply put, the notion of judgment does not depend on the notion of proposition, and we must introduce new kinds of judgments without using particular propositions. On the other hand, propositions are always explained with existing judgments, which include at least truth judgments because propositions must be accompanied by inference rules for establishing their truth.

In developing a formal system of propositional logic, we use two judgments: $A\ \mathit{prop}$ and $A\ \mathit{true}$.

$$\begin{array}{rcl} A\ \mathit{prop} & \Leftrightarrow & \textit{A is a proposition} \\ A\ \mathit{true} & \Leftrightarrow & \textit{A is true} \end{array}$$

$A\ \mathit{prop}$ becomes evident by the presence of an inference rule deducing $A\ \mathit{true}$. We will inductively define the set of propositions using binary connectives (*e.g.*, implication $\supset$, conjunction $\wedge$, disjunction $\vee$) and unary connectives (*e.g.*, negation $\neg$). The inference rules will be designed in such a way that the definition of a connective does not involve another connective. We say that the resultant system is *orthogonal* in the sense that all connectives can be developed independently of each other.

Exercise 2.1. Suppose that $\neg A$ is a proposition standing for the logical negation of A and that A *false* is a falsehood judgment denoting *"A cannot be true."* What is wrong with the rule $\dfrac{\neg A \; \mathit{true}}{A \; \mathit{false}} \neg\mathsf{E}$ as a means of explaining the notion of falsehood judgments? What about $\dfrac{A \; \mathit{false}}{\neg A \; \mathit{true}} \neg\mathsf{I}$?

Exercise 2.2. Why is the rule $\dfrac{\neg A \vee B \; \mathit{true}}{A \supset B \; \mathit{true}} \supset\mathsf{I}$ bad, apart from its strange meaning?

2.2 Natural deduction system for propositional logic

Natural deduction [**?**] is a principle for building a system of logic whose main concepts are *introduction* and *elimination rules*. An introduction rule explains how to deduce a truth judgment involving a particular connective, exploiting those judgments in the premise. That is, it explains how to "introduce" the connective in a derivation (when read in the top-down way). For example, an introduction rule for the conjunction connective would look like:

$$\frac{\cdots}{A \wedge B \; \mathit{true}} \wedge\mathsf{I}$$

A dual concept is an elimination rule which explains how to exploit a truth judgment involving a particular connective to deduce another judgment in the conclusion. That is, it explains how to "eliminate" the connective in a derivation (when read in the top-down way). For example, an elimination rule for the conjunction connective would look like:

$$\frac{A \wedge B \; \mathit{true}}{\cdots} \wedge\mathsf{E}$$

An introduction rule usually conveys the intuition behind a connective and is thus relatively easy to design. In contrast, an elimination rule extracts the knowledge represented by a judgment and careful design is required to ensure that the resultant system is sound and complete in a sense to be explained in Section 2.5. For example, an ill-designed elimination rule may be so strong as to extract false knowledge that cannot be justified by its corresponding introduction rule. Or it may be too weak to deduce any interesting judgment. Note that an introduction rule takes precedence over its corresponding elimination rule because without an introduction rule, there is no use in designing an elimination rule. That is, an elimination rule cannot be considered separately from its corresponding introduction rule whereas the design of an introduction rule can be an isolated task.

Below we develop a natural deduction system for propositional logic, beginning with the conjunction connective $\wedge$ (which is the easiest case).

Conjunction

Before we investigate inference rules for $\wedge$, we need to know how to build valid propositions involving $\wedge$. Hence we need a *formation rule* to state that $A \wedge B$, read as *"A and B"* or *"A conjunction B,"* is a proposition if both A and B are propositions:

$$\frac{A \; \mathit{prop} \quad B \; \mathit{prop}}{A \wedge B \; \mathit{prop}} \wedge\mathsf{F}$$

In order to justify the rule $\wedge$F, we need an inference rule for proving the truth of $A \wedge B$ on the assumption that there are inference rules for proving the truth of A and B. Since $A \wedge B$ is intended to be true whenever both A and B are true, we use the following introduction rule to admit $A \wedge B$ as a proposition:

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

The rule $\wedge$I says that if both A and B are true, then $A \wedge B$ is true. It follows the usual interpretation of an inference rule: if the premise holds, then the conclusion holds. Now we may use the rule $\wedge$I to construct a proof of $A \wedge B$ *true* from a proof $\mathcal{D}_A$ of A *true* and a proof $\mathcal{D}_B$ of B *true*; we write $\begin{array}{c}\mathcal{D}_A\\ A\ \mathit{true}\end{array}$ to mean that $\mathcal{D}_A$ is a proof of A *true*, including the last inference rule whose conclusion is A *true*:

$$\dfrac{\begin{array}{c}\mathcal{D}_A\\ A\ \mathit{true}\end{array} \quad \begin{array}{c}\mathcal{D}_B\\ B\ \mathit{true}\end{array}}{A \wedge B\ \mathit{true}}\ \wedge\mathsf{I}$$

The design of an elimination rule for $\wedge$ begins with $A \wedge B$ *true* as a premise. Since $A \wedge B$ *true* expresses that both A and B are true, we may conclude either A *true* or B *true* from $A \wedge B$ *true*, as shown in the two elimination rules for $\wedge$:

$$\dfrac{A \wedge B\ \mathit{true}}{A\ \mathit{true}}\ \wedge\mathsf{E_L} \qquad \dfrac{A \wedge B\ \mathit{true}}{B\ \mathit{true}}\ \wedge\mathsf{E_R}$$

Implication

The implication connective $\supset$ requires the notion of a *hypothetical proof* which is a proof containing *hypotheses*. We read $A \supset B$ as *"A implies B"* or *"if A, then B,"* and use the following formation rule:

$$\dfrac{A\ \mathit{prop} \quad B\ \mathit{prop}}{A \supset B\ \mathit{prop}}\ \supset\mathsf{F}$$

The intuition behind $\supset$ is that $A \supset B$ *true* holds whenever A *true* implies B *true*, or a hypothesis of A *true* leads to a proof of B *true*. We write a hypothesis of A *true* as $\overline{A\ \mathit{true}}$, and obtain the following introduction rule for $\supset$:

$$\dfrac{\begin{array}{c}\overline{A\ \mathit{true}}^{\,x}\\ \vdots\\ B\ \mathit{true}\end{array}}{A \supset B\ \mathit{true}}\ \supset\mathsf{I}^x$$

We may directly deduce A *true* using the hypothesis $\overline{A\ \mathit{true}}^{\,x}$ when necessary in the proof of B *true*.

The premise of the rule $\supset\mathsf{I}^x$ is an example of a hypothetical proof because it contains a hypothesis, *i.e.*, a judgment that is assumed to hold. We say that the rule $\supset\mathsf{I}$ *internalizes* the hypothetical proof in its premise as a proposition $A \supset B$ in the sense that the truth of $A \supset B$ compactly represents the knowledge expressed by the hypothetical proof.

There are three observations to make about the rule $\supset\mathsf{I}^x$. First we annotate both the hypothesis $\overline{A\ \mathit{true}}$ and the rule name $\supset\mathsf{I}$ with the same label x. Thus a label in a hypothesis indicates from which inference rule the hypothesis originates. It is not necessary to annotate all hypotheses with different labels as long as no conflict occurs between two hypotheses with the same label. For example, the following derivation is okay even though both hypotheses are annotated with the same label x:

$$\dfrac{\dfrac{\begin{array}{c}\overline{A\ \mathit{true}}^{\,x}\\ \vdots\\ B\ \mathit{true}\end{array}}{A \supset B\ \mathit{true}}\ \supset\mathsf{I}^x \quad \dfrac{\begin{array}{c}\overline{A'\ \mathit{true}}^{\,x}\\ \vdots\\ B'\ \mathit{true}\end{array}}{A' \supset B'\ \mathit{true}}\ \supset\mathsf{I}^x}{(A \supset B) \wedge (A' \supset B')\ \mathit{true}}$$

Second the hypothesis $\overline{A\ \mathit{true}}^{\,x}$ remains in effect only within the premise of the rule $\supset\mathsf{I}^x$. In other words, its scope is restricted to the premise of the rule $\supset\mathsf{I}^x$. After the rule $\supset\mathsf{I}^x$ is applied to deduce $A \supset B$ *true*, $\overline{A\ \mathit{true}}^{\,x}$ may no longer be used as a valid hypothesis. For example, the proof below may not use the hypothesis $\overline{A\ \mathit{true}}^{\,x}$ in the proof of in the proof $\mathcal{D}_A$ of A *true* which lies outside the scope of $\overline{A\ \mathit{true}}^{\,x}$:

$$\dfrac{\begin{array}{c}\mathcal{D}_A\\ A\ \mathit{true}\end{array} \quad \dfrac{\begin{array}{c}\overline{A\ \mathit{true}}^{\,x}\\ \vdots\\ B\ \mathit{true}\end{array}}{A \supset B\ \mathit{true}}\ \supset\mathsf{I}^x}{A \wedge (A \supset B)\ \mathit{true}}\ \wedge\mathsf{I}$$

We say that a hypothesis is *discharged* when its corresponding inference rule is applied and its scope is exited.

Note that while the premise of the rule $\supset\!\mathsf{I}^x$ is a hypothetical proof, the whole proof itself is *not* a hypothetical proof. Specifically the proof $\mathcal{D}$ below is a hypothetical proof, but the proof $\mathcal{E}$ is not:

$$\mathcal{E}\left\{\begin{array}{l} \dfrac{\mathcal{D}\left\{\begin{array}{c}\overline{A\ \mathit{true}}^{\,x}\\ \vdots \\ B\ \mathit{true}\end{array}\right.}{A \supset B\ \mathit{true}}\ \supset\!\mathsf{I}^x \end{array}\right.$$

The reason why $\mathcal{E}$ is not a hypothetical proof is that the hypothesis $\overline{A\ \mathit{true}}^{\,x}$ is discharged when the rule $\supset\!\mathsf{I}^x$ is applied, and thus is not visible to the outside. That is, we are free to use any hypothesis without turning the whole proof into a hypothetical proof as long as it is eventually discharged.

Third the hypothesis $\overline{A\ \mathit{true}}^{\,x}$ may be used not just once but as many times as necessary. In fact, we may even ignore it in the proof without using it at all. Here are examples of proofs that ignore $\overline{A\ \mathit{true}}^{\,x}$, use it once, and use it twice:

$$\dfrac{\dfrac{\overline{B\ \mathit{true}}^{\,y} \quad \begin{array}{c}\overline{A\ \mathit{true}}^{\,x}\\ \textit{(not used in the proof)}\end{array}}{A \supset B\ \mathit{true}}\ \supset\!\mathsf{I}^x}{B \supset (A \supset B)\ \mathit{true}}\ \supset\!\mathsf{I}^y \qquad \dfrac{\overline{A\ \mathit{true}}^{\,x}}{A \supset A\ \mathit{true}}\ \supset\!\mathsf{I}^x \qquad \dfrac{\dfrac{\overline{A\ \mathit{true}}^{\,x} \quad \overline{A\ \mathit{true}}^{\,x}}{A \wedge A\ \mathit{true}}\ \wedge\mathsf{I}}{A \supset (A \wedge A)\ \mathit{true}}\ \supset\!\mathsf{I}^x$$

As with the elimination rules for $\wedge$, the design of the elimination rule for $\supset$ begins with a premise $A \supset B\ \mathit{true}$. Since $A \supset B\ \mathit{true}$ expresses that $A\ \mathit{true}$ implies $B\ \mathit{true}$, the only way to exploit it is by supplying a proof of $A\ \mathit{true}$ to conclude $B\ \mathit{true}$. Hence the elimination rule for $\supset$ uses both $A \supset B\ \mathit{true}$ and $A\ \mathit{true}$ as its premises:

$$\dfrac{A \supset B\ \mathit{true} \quad A\ \mathit{true}}{B\ \mathit{true}}\ \supset\!\mathsf{E}$$

The following example proves $(A \supset B) \supset (A \supset B)\ \mathit{true}$ using the rule $\supset\!\mathsf{E}$:

$$\dfrac{\dfrac{\dfrac{\overline{A \supset B\ \mathit{true}}^{\,x} \quad \overline{A\ \mathit{true}}^{\,y}}{B\ \mathit{true}}\ \supset\!\mathsf{E}}{A \supset B\ \mathit{true}}\ \supset\!\mathsf{I}^y}{(A \supset B) \supset (A \supset B)\ \mathit{true}}\ \supset\!\mathsf{I}^x$$

(We can also prove $(A \supset B) \supset (A \supset B)\ \mathit{true}$ by directly using the hypothesis $\overline{A \supset B\ \mathit{true}}^{\,x}$.)

Here are two examples involving both $\wedge$ and $\supset$. The two proofs show that $A \supset (B \supset C)$ and $(A \wedge B) \supset C$ are logically equivalent because each one implies the other. (See Section 2.3 for further details.)

$$\dfrac{\dfrac{\dfrac{\dfrac{\overline{A \supset (B \supset C)\ \mathit{true}}^{\,x} \quad \dfrac{\overline{A \wedge B\ \mathit{true}}^{\,y}}{A\ \mathit{true}}\ \wedge\mathsf{E_L}}{B \supset C\ \mathit{true}}\ \supset\!\mathsf{E} \quad \dfrac{\overline{A \wedge B\ \mathit{true}}^{\,y}}{B\ \mathit{true}}\ \wedge\mathsf{E_R}}{C\ \mathit{true}}\ \supset\!\mathsf{E}}{(A \wedge B) \supset C\ \mathit{true}}\ \supset\!\mathsf{I}^y}{(A \supset (B \supset C)) \supset ((A \wedge B) \supset C)\ \mathit{true}}\ \supset\!\mathsf{I}^x \qquad \dfrac{\dfrac{\dfrac{\dfrac{\overline{(A \wedge B) \supset C\ \mathit{true}}^{\,x} \quad \dfrac{\overline{A\ \mathit{true}}^{\,y} \quad \overline{B\ \mathit{true}}^{\,z}}{A \wedge B\ \mathit{true}}\ \wedge\mathsf{I}}{C\ \mathit{true}}\ \supset\!\mathsf{E}}{B \supset C\ \mathit{true}}\ \supset\!\mathsf{I}^z}{A \supset (B \supset C)\ \mathit{true}}\ \supset\!\mathsf{I}^y}{((A \wedge B) \supset C) \supset (A \supset (B \supset C))\ \mathit{true}}\ \supset\!\mathsf{I}^x$$

Disjunction

Like $\wedge$ and $\supset$, the disjunction connective $\vee$ is binary:

$$\dfrac{A\ \mathit{prop} \quad B\ \mathit{prop}}{A \vee B\ \mathit{prop}}\ \vee\mathsf{F}$$

$A \vee B$, read as "*A or B*" or "*A disjunction B*," is intended to be true when either A or B is true, but we do not necessarily know which alternative is true. In our formulation of propositional logic, an introduction rule for $\vee$ concludes $A \vee B$ *true* from a proof of either A *true* or B *true*:

$$\frac{A\ \mathit{true}}{A \vee B\ \mathit{true}}\ \vee\mathsf{I_L} \qquad \frac{B\ \mathit{true}}{A \vee B\ \mathit{true}}\ \vee\mathsf{I_R}$$

The design of an elimination rule for $\vee$ is not obvious. A naive attempt would be to conclude one of A *true* and B *true* from $A \vee B$ *true*:

$$\frac{A \vee B\ \mathit{true}}{A\ \mathit{true}}\ \vee\mathsf{E_L}? \qquad \frac{A \vee B\ \mathit{true}}{B\ \mathit{true}}\ \vee\mathsf{E_R}?$$

In a certain sense, both rules are too strong (or too powerful) because they conclude a judgment that cannot be justified by $A \vee B$ *true*, which does not specify exactly which of A *true* and B *true* holds. In fact, each rule allows us to prove A *true* for any proposition A:

$$\cfrac{\cfrac{\cfrac{\overline{B\ \mathit{true}}^{\,x}}{B \supset B\ \mathit{true}}\ \supset\mathsf{I}^x}{A \vee (B \supset B)\ \mathit{true}}\ \vee\mathsf{I_R}}{A\ \mathit{true}}\ \vee\mathsf{E_L}?$$

Since it is generally unknown which of A *true* and B *true* has been supplied in a proof of $A \vee B$ *true* (*e.g.*, when $A \vee B$ *true* is a hypothesis), the only logical way to exploit $A \vee B$ *true* is by considering both possibilities simultaneously. If we can prove C *true* both from A *true* and from B *true* for a certain proposition C, then we may conclude C *true* from $A \vee B$ *true*, since C *true* holds regardless of how the proof of $A \vee B$ *true* has been built. The elimination rule for $\vee$ expresses such a way of reasoning:

$$\frac{A \vee B\ \mathit{true} \qquad \begin{array}{c}\overline{A\ \mathit{true}}^{\,x}\\ \vdots\\ C\ \mathit{true}\end{array} \qquad \begin{array}{c}\overline{B\ \mathit{true}}^{\,y}\\ \vdots\\ C\ \mathit{true}\end{array}}{C\ \mathit{true}}\ \vee\mathsf{E}^{x,y}$$

Note that A *true* and B *true* are introduced as new hypotheses and are annotated with different labels x and y. As in the elimination rule for $\supset$, their scope is limited to their respective premises of the rule $\vee\mathsf{E}^{x,y}$ (*i.e.*, $\overline{A\ \mathit{true}}^{\,x}$ to the second premise and $\overline{B\ \mathit{true}}^{\,y}$ to the third premise), which means that both hypotheses are discharged when C *true* is deduced in the conclusion.

Unlike the elimination rules for $\wedge$ and $\supset$, the elimination rule for $\vee$ exploits $A \vee B$ *true* in an indirect way in that its conclusion contains a proposition C that is not necessarily A, B, or their combination. That is, when applying the elimination rule to $A \vee B$ *true*, we ourselves have to choose a proposition C (which can be completely unrelated to A and B) such that C *true* is provable both from A *true* and from B *true*. For this reason, the inclusion of $\vee$ in a system of logic makes it hard to investigate metalogical properties of the system, as we will see later.

As a trivial example, let us prove that A *true* is stronger than $A \vee B$ *true*:

$$\cfrac{\cfrac{\overline{A\ \mathit{true}}^{\,x}}{A \vee B\ \mathit{true}}\ \vee\mathsf{I_L}}{A \supset (A \vee B)\ \mathit{true}}\ \supset\mathsf{I}^x$$

The converse does not hold, *i.e.*, $A \vee B$ *true* is strictly weaker than A *true*, because there is no way to prove A *true* from B *true* for arbitrary propositions A and B:

$$\cfrac{\cfrac{\overline{A \vee B\ \mathit{true}}^{\,x} \qquad \overline{A\ \mathit{true}}^{\,y} \qquad \begin{array}{c}\overline{B\ \mathit{true}}^{\,z}\\ \vdots\\ A\ \mathit{true}\quad \textit{(impossible)}\end{array}}{A\ \mathit{true}}\ \vee\mathsf{E}^{y,z}}{(A \vee B) \supset A\ \mathit{true}}\ \supset\mathsf{I}^x$$

As another example, let us prove that the disjunction connective is commutative:

$$(A \vee B) \supset (B \vee A) \; \mathit{true}$$

We begin by applying the rule $\supset\mathsf{I}$ so that the problem reduces to proving $B \vee A \; \mathit{true}$ from $A \vee B \; \mathit{true}$:

$$\dfrac{\begin{array}{c}\overline{A \vee B \; \mathit{true}}^{\,x} \\ \vdots \\ B \vee A \; \mathit{true}\end{array}}{(A \vee B) \supset (B \vee A) \; \mathit{true}} \supset\mathsf{I}^{x}$$

At this point, the proof may proceed either in a bottom-up way by applying an introduction rule $\vee\mathsf{I}_\mathsf{L}$ or $\vee\mathsf{I}_\mathsf{R}$ to $B \vee A \; \mathit{true}$, or in a top-down way by applying the elimination rule $\vee\mathsf{E}$ to $A \vee B \; \mathit{true}$. In the first case, we eventually get stuck because it is impossible to prove $A \; \mathit{true}$ or $B \; \mathit{true}$ from $A \vee B \; \mathit{true}$. For example, we cannot fill the gap in the proof shown below:

$$\dfrac{\dfrac{\begin{array}{c}\overline{A \vee B \; \mathit{true}}^{\,x} \\ \vdots \\ B \; \mathit{true}\end{array}}{B \vee A \; \mathit{true}} \vee\mathsf{I}_\mathsf{L}}{(A \vee B) \supset (B \vee A) \; \mathit{true}} \supset\mathsf{I}^{x}$$

In the second case, the problem reduces to separately proving $B \vee A \; \mathit{true}$ from $A \; \mathit{true}$ and from $B \; \mathit{true}$, which is accomplished by applying the introduction rules for $\vee$:

$$\dfrac{\dfrac{\overline{A \vee B \; \mathit{true}}^{\,x} \quad \dfrac{\overline{A \; \mathit{true}}^{\,y}}{B \vee A \; \mathit{true}} \vee\mathsf{I}_\mathsf{R} \quad \dfrac{\overline{B \; \mathit{true}}^{\,z}}{B \vee A \; \mathit{true}} \vee\mathsf{I}_\mathsf{L}}{B \vee A \; \mathit{true}} \vee\mathsf{E}^{y,z}}{(A \vee B) \supset (B \vee A) \; \mathit{true}} \supset\mathsf{I}^{x}$$

Exercise 2.3. We can rewrite the elimination rule for the disjunction connective by using the implication connective in place of hypothetical proofs:

$$\dfrac{A \vee B \; \mathit{true} \quad A \supset C \; \mathit{true} \quad B \supset C \; \mathit{true}}{C \; \mathit{true}} \vee\mathsf{E}$$

Why do we not use the new elimination rule which actually seems simpler than the previous one?

Truth and falsehood

Truth $\top$ is a proposition that is assumed to be always true. Hence a proof of $\top \; \mathit{true}$ requires no particular evidence and is always provable, as indicated by the empty premise in its introduction rule:

$$\dfrac{}{\top \; \mathit{prop}} \top\mathsf{F} \qquad \dfrac{}{\top \; \mathit{true}} \top\mathsf{I}$$

Then how do we exploit a proof of $\top \; \mathit{true}$ in an elimination rule? Since we have to provide no particular evidence in a proof of $\top \; \mathit{true}$, there is no logical content in it, which implies that there is no interesting way to exploit it. Therefore $\top$ has no elimination rule.

Falsehood $\perp$ is a proposition that is never true, or equivalently, whose truth is impossible to establish. The intuition is that it denotes a logical contradiction which must not be provable under any circumstance. Therefore there is no introduction rule for $\perp$. Interestingly, however, there *is* an elimination rule for $\perp$. Suppose that we have a proof of $\perp \; \mathit{true}$. If we think of $\perp \; \mathit{true}$ as something impossible to prove, or as something that is the most difficult to prove, the existence of its proof implies that we can prove everything (which is no more difficult to prove than $\perp \; \mathit{true}$)! Therefore the elimination rule for $\perp$ deduces $C \; \mathit{true}$ for an arbitrary proposition C:

$$\dfrac{}{\perp \; \mathit{prop}} \perp\mathsf{F} \qquad \dfrac{\perp \; \mathit{true}}{C \; \mathit{true}} \perp\mathsf{E}$$

Then why do we need an elimination rule for $\perp$ at all, if it is impossible to prove $\perp$ *true*? While it is impossible to prove $\perp$ *true* out of nothing, it is possible to prove $\perp$ *true* using hypotheses. For example, $\perp$ *true* in the premise of the rule $\perp$E itself may be a hypothesis, as illustrated in the proof below:

$$\dfrac{\dfrac{\overline{\perp\ \mathit{true}}^{\,x}}{C\ \mathit{true}}\,\perp\mathsf{E}}{\perp \supset C\ \mathit{true}}\,\supset\mathsf{I}^x$$

In essence, there is nothing wrong with reasoning from an assumption that something impossible to prove has been proven somehow.

We say that a system of logic is *inconsistent* if $\perp$ *true* is provable in it, and *consistent* if not. An inconsistent system is worthless because a judgment A *true* is provable for an arbitrary proposition A. We will later present a proof that our system of propositional logic is consistent, whose discovery was in fact a major milestone in the history of logic.

Truth $\top$ and falsehood $\perp$ can also be viewed as the nullary cases of conjunction and disjunction, respectively. Consider a general n-ary case $\bigwedge_{i=1}^n A_i$ of conjunction with a single introduction rule and n elimination rules:

$$\dfrac{A_i\ \mathit{true}\quad \mathit{for}\ i = 1, \cdots, n}{\bigwedge_{i=1}^n A_i\ \mathit{true}}\,\bigwedge\mathsf{I} \qquad \dfrac{\bigwedge_{i=1}^n A_i\ \mathit{true}}{A_i\ \mathit{true}}\,\bigwedge\mathsf{E}_i\ (1 \le i \le n)$$

If we let $\top = \bigwedge_{i=1}^n A_i$ with $n = 0$, the rule $\bigwedge$I turns into the rule $\top$I because it comes to have an empty premise, and each rule $\bigwedge\mathsf{E}_i$ disappears (*i.e.*, no elimination rule for $\top$). Similarly a general n-ary case $\bigvee_{i=1}^n A_i$ of disjunction has n introduction rules and a single elimination rule:

$$\dfrac{A_i\ \mathit{true}}{\bigvee_{i=1}^n A_i\ \mathit{true}}\,\bigvee\mathsf{I}_i (1 \le i \le n) \qquad \dfrac{\bigvee_{i=1}^n A_i\ \mathit{true} \qquad \begin{array}{c}\overline{A_i\ \mathit{true}}^{\,x_i}\\ \vdots \\ C\ \mathit{true}\quad \mathit{for}\ i = 1, \cdots, n\end{array}}{C\ \mathit{true}}\,\bigvee\mathsf{E}^x$$

If we let $\perp = \bigvee_{i=1}^n A_i$ with $n = 0$, each rule $\bigvee\mathsf{I}_i$ disappears (*i.e.*, no introduction rule for $\perp$), and the rule $\bigvee$E turns into the rule $\perp$E because all hypothetical proofs in its premise disappear.

Now it is clear that $\top$ and $\perp$ are identities for the binary connectives $\wedge$ and $\vee$, respectively. For example, we can identify $A \wedge \top$ with A: if A *true* is provable, then $A \wedge \top$ *true* is also provable because $\top$ *true* automatically holds; the converse follows by the rule $\wedge\mathsf{E_L}$. Similarly we can identify $A \vee \perp$ with A: if $A \vee \perp$ *true* is provable, A *true* must also be provable because the second alternative $\perp$ *true* cannot be taken; the converse follows by the rule $\vee\mathsf{I_L}$.

Negation

The only unary connective in propositional logic is negation $\neg$:

$$\dfrac{A\ \mathit{prop}}{\neg A\ \mathit{prop}}\,\neg\mathsf{F}$$

$\neg A$, read as "*not A*" or "*negation A,*" denotes the logical negation of A, and its truth means that A cannot be true. Below we consider three different approaches to designing inference rules for negation, all of which provide a means to express that A cannot be true.

The first approach is to define a falsehood judgment A *false* denoting "*A cannot be true*" and then use the following rules to deduce and exploit $\neg A$ *true*:

$$\dfrac{A\ \mathit{false}}{\neg A\ \mathit{true}}\,\neg\mathsf{I} \qquad \dfrac{\neg A\ \mathit{true}}{A\ \mathit{false}}\,\neg\mathsf{E}$$

(We do not discuss inference rules for deducing A *false*.) As in the rule $\supset$I, we say that the rule $\neg$I internalizes A *false* as a proposition $\neg A$ in the sense that the truth of $\neg A$ compactly represents the knowledge expressed by A *false*.

In the second approach, we deduce $\neg A\ true$ if an assumption of $A\ true$ leads to the provability of every truth judgment. The rationale is that if the system is known to be consistent (and thus not every truth judgment is provable), the provability of every truth judgment, *i.e.*, inconsistency of the system, as a consequence of an assumption of $A\ true$ implies that the assumption must be wrong, that is, A cannot be true.

In order to be able to express the provability of every truth judgment, we introduce a *propositional variable* p which stands for *any* proposition. We use a *parametric judgment* $p\ true$, or a judgment parametric in a propositional variable p, in the introduction rule for $\neg$:

$$\dfrac{\begin{array}{c}\overline{A\ true}^{\,x}\\ \vdots\\ p\ true\end{array}}{\neg A\ true}\ \neg\mathsf{I}^{x,p}$$

Since the premise is a hypothetical judgment, we annotate the hypothesis $\overline{A\ true}$ and the rule name $\neg\mathsf{I}$ with the same label x. Moreover we annotate the rule name $\neg\mathsf{I}$ with the propositional variable p as well, since p is a fresh variable whose scope is restricted to the premise. The elimination rule for $\neg$ states that proofs of both $\neg A\ true$ and $A\ true$ license us to prove the truth of any proposition:

$$\dfrac{\neg A\ true \quad A\ true}{C\ true}\ \neg\mathsf{E}$$

Note that C in the conclusion can be any proposition, including propositional variables. As an example, we prove that A and $\neg A$ cannot be true simultaneously:

$$\dfrac{\dfrac{\dfrac{\overline{A \wedge \neg A\ true}^{\,x}}{\neg A\ true}\ \wedge\mathsf{E_R} \quad \dfrac{\overline{A \wedge \neg A\ true}^{\,x}}{A\ true}\ \wedge\mathsf{E_L}}{p\ true}\ \neg\mathsf{E}}{\neg(A \wedge \neg A)\ true}\ \neg\mathsf{I}^{x,p}$$

The third approach uses a *notational definition* by regarding $\neg A$ as a syntactic abbreviation of $A \supset \bot$. That is, $\neg$ plays no semantic role at all and $\neg A$ is simply expanded to $A \supset \bot$. The notational definition of $\neg$ justifies the following rules:

$$\dfrac{\begin{array}{c}\overline{A\ true}^{\,x}\\ \vdots\\ \bot\ true\end{array}}{\neg A\ true}\ \neg\mathsf{I}^{x} \qquad \dfrac{\neg A\ true \quad A\ true}{\bot\ true}\ \neg\mathsf{E}$$

Note that if $\neg$ was defined as an independent connective rather than a notational convenience, these rules would destroy the orthogonality of the system because the meaning of $\neg$ would depend on the meaning of $\bot$. We use the third approach in our treatment of $\neg$ (which is the most popular definition in the literature).

As an example, we prove that if A is true, then $\neg A$ cannot be true:

$$\dfrac{\dfrac{\dfrac{\overline{\neg A\ true}^{\,y} \quad \overline{A\ true}^{\,x}}{\bot\ true}\ \neg\mathsf{E}}{\neg\neg A\ true}\ \neg\mathsf{I}^{y}}{A \supset \neg\neg A\ true}\ \supset\mathsf{I}^{x}$$

The converse $\neg\neg A \supset A\ true$ is *not* provable, however, which implies that $A\ true$ is strictly stronger than $\neg\neg A\ true$. That is, a proof that $\neg A$ cannot be true is not enough for concluding that A is true. A failed

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

$$\begin{array}{c} \overline{A\ true}^{\,x} \\ \vdots \\ B\ true \\ \hline A \supset B\ true \end{array} \supset\mathsf{I}^x \qquad \frac{A \supset B\ true \quad A\ true}{B\ true} \supset\mathsf{E}$$

$$\frac{A\ true}{A \vee B\ true} \vee\mathsf{I_L} \qquad \frac{B\ true}{A \vee B\ true} \vee\mathsf{I_R} \qquad \begin{array}{ccc} & \overline{A\ true}^{\,x} & \overline{B\ true}^{\,y} \\ & \vdots & \vdots \\ A \vee B\ true & C\ true & C\ true \\ \hline & C\ true & \end{array} \vee\mathsf{E}^{x,y}$$

$$\frac{}{\top\ true} \top\mathsf{I} \qquad \frac{\bot\ true}{C\ true} \bot\mathsf{E} \qquad \begin{array}{c} \overline{A\ true}^{\,x} \\ \vdots \\ \bot\ true \\ \hline \neg A\ true \end{array} \neg\mathsf{I}^x \qquad \frac{\neg A\ true \quad A\ true}{\bot\ true} \neg\mathsf{E}$$

Figure 2.1: Natural deduction system for propositional logic

attempt to prove $\neg\neg A \supset A\ true$ would look like:

$$\cfrac{\cfrac{\cfrac{\overline{\neg\neg A\ true}^{\,x} \qquad \begin{array}{c} \overline{\neg\neg A\ true}^{\,x} \\ \vdots\ ? \\ \neg A\ true \end{array}}{\bot\ true} \neg\mathsf{E}}{A\ true} \bot\mathsf{E}}{\neg\neg A \supset A\ true} \supset\mathsf{I}^x$$

The unprovability of $\neg\neg A \supset A\ true$ is a quintessential feature of the system of logic presented so far, or any system belonging to what is known as *constructive logic* or *intuitionistic logic*. In constructive logic, what $\neg A\ true$ proves is not exactly the direct opposite of what $A\ true$ proves. Rather it provides only indirect evidence that there is no proof of $A\ true$ by showing that the existence of such a proof leads to a logical contradiction. In contrast, *classical logic* assumes that every proposition is either true or false and has no intermediate state. Under classical logic, $\neg\neg A\ true$ is indistinguishable from $A\ true$ because A is either true or false and we have positive evidence that A cannot be false. The truth table method for proving the truth of a proposition is based on classical logic, which tries all possible combinations of truth and falsehood values for all atomic propositions. Until we come back to the topic of classical logic in Chapter 7, we focus only on constructive logic.

Figure 2.1 shows all inference rules of propositional logic where the set of propositions is inductively defined as follows:

$$\text{proposition} \quad A \quad ::= \quad P \mid A \wedge A \mid A \supset A \mid A \vee A \mid \top \mid \bot \mid \neg A$$

P is called a *propositional constant* and denotes an atomic proposition (*e.g.* *'1 + 1 is equal to 0,' '1 + 1 is equal to 2' is true,' 'the moon is made of cheese,' etc*). The rules $\neg\mathsf{I}$ and $\neg\mathsf{E}$ are derived rules under the notational definition $\neg A = A \supset \bot$. From now on, we use the following operator precedence

$$\neg > \wedge > \vee > \supset$$

where $\wedge$, $\vee$, $\supset$ are all right-associative. Examples are:

$$\begin{array}{rcl} \neg A \wedge B &=& (\neg A) \wedge B \\ A \wedge B \vee C &=& (A \wedge B) \vee C \\ A \vee B \supset C &=& (A \vee B) \supset C \\ \neg A \wedge B \vee C \supset D &=& (((\neg A) \wedge B) \vee C) \supset D \end{array} \qquad \begin{array}{rcl} A \wedge B \wedge C &=& A \wedge (B \wedge C) \\ A \vee B \vee C &=& A \vee (B \vee C) \\ A \supset B \supset C &=& A \supset (B \supset C) \end{array}$$

2.3 Logical equivalence

We say that a proposition A is logically equivalent to another proposition B, written $A \equiv B$, if A *true* implies B *true* and vice versa. A notational definition of logical equivalence $A \equiv B$ is given as follows:

$$A \equiv B \quad = \quad (A \supset B) \wedge (B \supset A) \; true$$

If A and B are logically equivalent, an occurrence of A inside any proposition may be replaced by B (or an occurrence of B by A) without changing its meaning in that the resultant proposition remains logically equivalent to the original proposition. Thus logical equivalences enable us to simplify a proof involving a proposition that is logically equivalent to a less complex proposition. For example,

$$\neg\neg\neg A \supset (\neg\neg\neg B \supset \neg(A \vee B)) \; true$$

becomes easy (or even obvious) to prove once we transform $\neg\neg\neg A \supset (\neg\neg\neg B \supset \neg(A \vee B))$ into $(\neg A \wedge \neg B) \supset \neg(A \vee B)$ by exploiting logical equivalences $\neg\neg\neg A \equiv \neg A$ and $A \supset (B \supset C) \equiv (A \wedge B) \supset C$.

Below we list logical equivalences of propositional logic which are divided into three groups.

Commutativity and idempotence. $\wedge$ and $\vee$ are commutative and idempotent. An implication $A \supset A$ is logically meaningless and reduces to $\top$.

(C1) $A \wedge B \equiv B \wedge A$
(C2) $A \vee B \equiv B \vee A$
(C3) $A \supset B \not\equiv B \supset A$
(I1) $A \wedge A \equiv A$
(I2) $A \vee A \equiv A$
(I3) $A \supset A \equiv \top$

Truth and falsehood. Each logical equivalence below deals with a proposition of the form $\top \, \phi \, A$, $\bot \, \phi \, A$, $A \supset \top$, or $A \supset \bot$ where ϕ is $\wedge$, $\vee$, or $\supset$.

(M1) $\top \wedge A \equiv A$
(M2) $\top \vee A \equiv \top$
(M3) $\top \supset A \equiv A$
(M4) $\bot \wedge A \equiv \bot$
(M5) $\bot \vee A \equiv A$
(M6) $\bot \supset A \equiv \top$
(M7) $A \supset \top \equiv \top$
(M8) $A \supset \bot = \neg A$

Interaction between connectives. Each logical equivalence below deals with a proposition of the form $A \; \phi \; (B \; \phi \; C)$ or $(A \; \phi \; B) \supset C$ where ϕ is $\wedge$, $\vee$, or $\supset$.

(L1)	$A \wedge (B \wedge C) \equiv (A \wedge B) \wedge C$	*(associativity of $\wedge$)*
(L2)	$A \wedge (B \vee C) \equiv (A \wedge B) \vee (A \wedge C)$	*(distributivity of $\wedge$ over $\vee$)*
(L3)	$A \wedge (B \supset C) \equiv ?$	*(no interaction)*
(L4)	$A \vee (B \wedge C) \equiv (A \vee B) \wedge (A \vee C)$	*(distributivity of $\vee$ over $\wedge$)*
(L5)	$A \vee (B \vee C) \equiv (A \vee B) \vee C$	*(associativity of $\vee$)*
(L6)	$A \vee (B \supset C) \equiv ?$	*(no interaction)*
(L7)	$A \supset (B \wedge C) \equiv (A \supset B) \wedge (A \supset C)$	*(distributivity of $\supset$ over $\wedge$)*
(L8)	$A \supset (B \vee C) \equiv ?$	*(no interaction)*
(L9)	$A \supset (B \supset C) \equiv (A \wedge B) \supset C$	
(L10)	$(A \wedge B) \supset C \equiv A \supset (B \supset C)$	
(L11)	$(A \vee B) \supset C \equiv (A \supset C) \wedge (B \supset C)$	
(L12)	$(A \supset B) \supset C \equiv ?$	*(no interaction)*

Exercise 2.4. Prove logical equivalences (L2), (L4), (L7), (L9), and (L11).

2.4 Hypothetical judgments

While the rules in Figure 2.1 define a natural deduction system for propositional logic, they are unwieldy for writing hypothetical proofs. This is because no rule provides visual aid for keeping track of the scope of each hypothesis or an apparatus for preventing a hypothesis from escaping its scope. For example, the following hypothetical proof contains a wrong use of a hypothesis $\overline{A\ \mathit{true}}^{\,x}$ outside its scope:

$$\dfrac{\dfrac{\overline{A\ \mathit{true}}^{\,x}}{A \supset A\ \mathit{true}}\supset\!\mathsf{I}^x \qquad \overline{A\ \mathit{true}}^{\,x}\,\textit{(wrong use)}}{(A \supset A) \wedge A\ \mathit{true}}\wedge\mathsf{I}$$

Whenever a new hypothesis is introduced, therefore, we could draw an imaginary contour to delineate its scope, as illustrated below:

$$\dfrac{\boxed{\begin{array}{c}\overline{A\ \mathit{true}}^{\,x}\\ \vdots\\ B\ \mathit{true}\end{array}}^{\,x}}{A \supset B\ \mathit{true}}\supset\!\mathsf{I}^x \qquad \dfrac{A \vee B\ \mathit{true} \quad \boxed{\begin{array}{c}\overline{A\ \mathit{true}}^{\,x}\\ \vdots\\ C\ \mathit{true}\end{array}}^{\,x} \quad \boxed{\begin{array}{c}\overline{B\ \mathit{true}}^{\,y}\\ \vdots\\ C\ \mathit{true}\end{array}}^{\,y}}{C\ \mathit{true}}\vee\mathsf{E}^{x,y}$$

Here the scope of a hypothesis $\overline{A\ \mathit{true}}^{\,x}$ is restricted to the contour labeled x. Note that a contour may be enclosed within another contour (as in a topography map), as shown in the following example:

$$\dfrac{\boxed{\begin{array}{c}\overline{A\ \mathit{true}}^{\,x}\\ \dfrac{\boxed{\dfrac{\overline{B\ \mathit{true}}^{\,y}}{A \wedge B\ \mathit{true}}\wedge\mathsf{I}}^{\,y}}{B \supset (A \wedge B)\ \mathit{true}}\supset\!\mathsf{I}^y\end{array}}^{\,x}}{A \supset (B \supset (A \wedge B))\ \mathit{true}}\supset\!\mathsf{I}^x$$

The hypothesis $\overline{A\ \mathit{true}}^{\,x}$ may be used inside the inner contour labeled y (which makes it possible to prove $A \wedge B\ \mathit{true}$ by the rule $\wedge\mathsf{I}$), but the hypothesis $\overline{B\ \mathit{true}}^{\,y}$ cannot be used outside the inner contour.

Hypothetical judgments provide a convenient way to keep track of the scope of each hypothesis in a hypothetical proof. A hypothetical judgment $J_1, \cdots, J_n \vdash J$ becomes evident by a hypothetical proof deducing a judgment J from a collection of hypotheses $\overline{J_1}, \cdots, \overline{J_n}$:

$$J_1, \cdots, J_n \vdash J \quad \Longleftrightarrow \quad \begin{array}{c}\overline{J_1} \quad \cdots \quad \overline{J_n}\\ \ddots\ \cdot^{\cdot^{\cdot}}\\ J\end{array} \quad \}\ \text{inference rules}$$

Thus we may read $J_1, \cdots, J_n \vdash J$ as "if judgments $J_1, \cdots, J_n$ hold, then a judgment J hold." When $J_1, \cdots, J_n \vdash J$ holds, we say that J_1 through J_n *entail* J, or J is a consequence of J_1 through J_n. Hence $\vdash$ is called an *entailment relation* or a *consequence relation*. We refer to J_i, $1 \leq i \leq n$, as an *antecedent* and J as the *succedent*. We often abbreviate a collection of antecedents as Γ, as in a hypothetical judgment $\Gamma \vdash J$.

In developing a natural deduction system for propositional logic, we will use hypothetical judgments of the form $A_1\ \mathit{true}, \cdots, A_n\ \mathit{true} \vdash A\ \mathit{true}$ where antecedents and succedents are all truth judgments. Before presenting its inference rules, let us investigate properties of hypothetical judgments of the general form where antecedents and conclusions can be any judgments.

The definition of hypothetical judgments justifies two principles: *reflexivity* and *substitution principle*:

- (Reflexivity) $\Gamma, J, \Gamma' \vdash J$.
- (Substitution principle) If $\Gamma \vdash J$ and $\Gamma, J \vdash J'$, then $\Gamma \vdash J'$.

Reflexivity states that we may use a hypothesis $\overline{J}$ to conclude J. The substitution principle states that if we can prove a judgment J from a collection of hypotheses, we may use J as another hypothesis whenever the same collection of hypotheses is available. That is, we may use J as a lemma once we build a proof of $\Gamma \vdash J$.

To see how the substitution principle works, let us assume $\Gamma \vdash J$ and $\Gamma, J \vdash J'$ which imply that there are two hypothetical proofs $\mathcal{D}$ and $\mathcal{E}$ as shown below:

$$\Gamma \vdash J \quad \Longleftrightarrow \quad \left.\begin{array}{c}\overline{\Gamma}\\ \vdots\\ J\end{array}\right\}\mathcal{D} \qquad\qquad \Gamma, J \vdash J' \quad \Longleftrightarrow \quad \left.\begin{array}{ccc}\overline{\Gamma} & \cdots & \overline{J}\\ & \ddots\;\cdot^{\cdot^{\cdot}} & \\ & J' & \end{array}\right\}\mathcal{E}$$

Here $\overline{\Gamma}$ is a shorthand for $\{\overline{J} \,|\, J \in \Gamma\}$. Now we locate every occurrence of the hypothesis $\overline{J}$ in $\mathcal{E}$ and *substitute* $\mathcal{D}$ for it, which results in the following hypothetical proof:

$$\begin{array}{ccc} & & \left.\begin{array}{c}\overline{\Gamma}\\ \vdots\\ J\end{array}\right\}\mathcal{D}\\ \overline{\Gamma} & \cdots & \\ & \ddots\;\cdot^{\cdot^{\cdot}} & \\ & J' & \end{array}$$

Since the same hypothesis (*e.g.*, one in $\overline{\Gamma}$) may be used as many times as necessary (see Page 20), the hypothetical judgment above makes evident the hypothetical judgment $\Gamma \vdash J'$, which is what the substitution principle concludes from $\Gamma \vdash J$ and $\Gamma, J \vdash J'$.

Our definition of hypothetical judgments makes two implicit assumptions: 1) the order of hypotheses is immaterial; 2) a hypothesis may be used zero or more times in a hypothetical proof. These assumptions are formally stated in the *structural properties* of hypothetical judgments:

- (Exchange) If $\Gamma, J_i, J_{i+1}, \Gamma' \vdash J$, then $\Gamma, J_{i+1}, J_i, \Gamma' \vdash J$.
- (Weakening) If $\Gamma, \Gamma' \vdash J$, then $\Gamma, J', \Gamma' \vdash J$ for any judgment J'.
- (Contraction) If $\Gamma, J_i, J_i, \Gamma' \vdash J$, then $\Gamma, J_i, \Gamma' \vdash J$.

Exchange states that we may ignore the order of antecedents in a hypothetical judgment. Weakening states that we may add a new antecedent without using it, thereby "weakening" what is being proven. ($\Gamma, J' \vdash J$ is weaker than $\Gamma \vdash J$ because it draws the same conclusion from more hypotheses.) By contraction, we may combine two copies of the same antecedent into one. Note that by weakening, a hypothesis may be used zero times and that by contraction, a hypothesis may be used more than once.

Here are a few further remarks on hypothetical judgments:

- Hypothetical judgments are just a *"convenient"* way, rather than a new way, to represent hypothetical proofs. That is, the entailment relation $\vdash$ is just a syntactic tool for displaying the hypotheses and the conclusion of a hypothetical judgment while hiding its internal structure, and thus does *not* introduce a new semantic notion. (In contrast, the relation $\models$ from model theory defines the notion of *semantic* consequence. Hence it has nothing to do with hypothetical judgments and is *not* a syntactic convenience.)
- There can be more than one hypothetical proof by which a given hypothetical judgment becomes evident, since hypothetical judgments is concerned only with hypotheses and conclusions.
- A hypothetical judgment itself is an example of a judgment and thus may be used as an antecedent or the conclusion in another hypothetical judgment, although we will not use hypothetical judgments in such a way in our discussion of logic. In fact, $J_1, \cdots, J_n \vdash J$ can be thought of as an abbreviation of a nested hypothetical judgment $J_1 \vdash (J_2 \vdash \cdots (J_n \vdash J) \cdots)$, where each antecedent J_i or the conclusion J may be another hypothetical judgment!

$$\frac{A\ \mathit{true} \in \Gamma}{\Gamma \vdash A\ \mathit{true}}\ \mathsf{Hyp} \qquad \frac{\Gamma, A\ \mathit{true} \vdash B\ \mathit{true}}{\Gamma \vdash A \supset B\ \mathit{true}}\ \supset\!\mathsf{I} \qquad \frac{\Gamma \vdash A \supset B\ \mathit{true} \quad \Gamma \vdash A\ \mathit{true}}{\Gamma \vdash B\ \mathit{true}}\ \supset\!\mathsf{E}$$

$$\frac{\Gamma \vdash A\ \mathit{true} \quad \Gamma \vdash B\ \mathit{true}}{\Gamma \vdash A \wedge B\ \mathit{true}}\ \wedge\mathsf{I} \qquad \frac{\Gamma \vdash A \wedge B\ \mathit{true}}{\Gamma \vdash A\ \mathit{true}}\ \wedge\mathsf{E_L} \qquad \frac{\Gamma \vdash A \wedge B\ \mathit{true}}{\Gamma \vdash B\ \mathit{true}}\ \wedge\mathsf{E_R}$$

$$\frac{\Gamma \vdash A\ \mathit{true}}{\Gamma \vdash A \vee B\ \mathit{true}}\ \vee\mathsf{I_L} \qquad \frac{\Gamma \vdash B\ \mathit{true}}{\Gamma \vdash A \vee B\ \mathit{true}}\ \vee\mathsf{I_R} \qquad \frac{\Gamma \vdash A \vee B\ \mathit{true} \quad \Gamma, A\ \mathit{true} \vdash C\ \mathit{true} \quad \Gamma, B\ \mathit{true} \vdash C\ \mathit{true}}{\Gamma \vdash C\ \mathit{true}}\ \vee\mathsf{E}$$

$$\frac{}{\Gamma \vdash \top\ \mathit{true}}\ \top\mathsf{I} \qquad \frac{\Gamma \vdash \bot\ \mathit{true}}{\Gamma \vdash C\ \mathit{true}}\ \bot\mathsf{E} \qquad \frac{\Gamma, A\ \mathit{true} \vdash \bot\ \mathit{true}}{\Gamma \vdash \neg A\ \mathit{true}}\ \neg\mathsf{I} \qquad \frac{\Gamma \vdash \neg A\ \mathit{true} \quad \Gamma \vdash A\ \mathit{true}}{\Gamma \vdash \bot\ \mathit{true}}\ \neg\mathsf{E}$$

Figure 2.2: Natural deduction system using hypothetical judgments

- While closely related to each other, a hypothetical judgment $J_1, \cdots, J_n \vdash J$ and a rule $\dfrac{J_1 \quad \cdots \quad J_n}{J}\ R$ are disparate concepts and thus impossible to compare for equivalence. The reason is simple: the former is a judgment whereas the latter is an inference rule. The existence of a proof of $J_1, \cdots, J_n \vdash J$ just implies that R is a derivable rule. Conversely, if the rule R is available, we can always prove $J_1, \cdots, J_n \vdash J$ with the following hypothetical proof:

$$\frac{\overline{J_1} \quad \cdots \quad \overline{J_n}}{J}\ R$$

Note, however, that the above hypothetical proof may not be the only way to prove $J_1, \cdots, J_n \vdash J$ if we can build another hypothetical proof $\begin{array}{ccc} \overline{J_1} & \cdots & \overline{J_n} \\ & \ddots\ \vdots\ \cdot^{\cdot^{\cdot}} & \\ & J & \end{array}$ without using the rule R at all.

- A hypothetical judgment $\cdot \vdash J$ with no antecedents is *not* equivalent to its succedent J. While the former states that J holds unconditionally (or categorically), the latter is unaware of whether there are hypotheses or not, and could be even a hypothesis in a hypothetical judgment. For example, from the assumption that J entails J' (*i.e.*, $J \vdash J'$), we can show that $\cdot \vdash J$ implies $\cdot \vdash J'$ by the substitution principle. The converse is not the case, however, because a proof of $\cdot \vdash J'$ does not necessarily extend a proof of $\cdot \vdash J$ so that J follows directly from J'. (If $\cdot \vdash J$ and J were equivalent, the converse would also be the case.)

- An important consequence of the structural properties is that the two hypothetical judgments in each rule, $\Gamma \vdash J$ from the *if* part and $\Gamma' \vdash J$ from the *then* part, represent hypothetical proofs not only of the same size (in terms of the number of applications of inference rules) but also of completely the same structure. As a result, when structural induction (or rule induction) is applicable to $\Gamma \vdash J$, *we may apply structural induction on* $\Gamma' \vdash J$ *instead.*

Figure 2.2 shows a natural deduction system for propositional logic using hypothetical judgments, which reuses the inference rule names from the previous natural deduction system. We use hypothetical judgments of the form $\Gamma \vdash A\ \mathit{true}$ where Γ is a collection of truth judgments and the exchange rule is built-in (*i.e.*, we may reorder antecedents as we like).

The rule Hyp expresses reflexivity of hypothetical judgments. All the other rules are justified by their counterparts in the previous natural deduction system. As an example, let us consider the rule ⊃I. The premise $\Gamma, A\ \mathit{true} \vdash B\ \mathit{true}$ implies the existence of a hypothetical proof deducing $B\ \mathit{true}$ from hypotheses $\overline{\Gamma}$ and $\overline{A\ \mathit{true}}$ (where $\overline{\Gamma}$ is a shorthand for $\{\overline{J}\,|J \in \Gamma\}$):

$$\Gamma, A\ \mathit{true} \vdash B\ \mathit{true} \quad \Longleftrightarrow \quad \begin{array}{ccc} \overline{\Gamma} & \cdots & \overline{A\ \mathit{true}} \\ & \ddots\ \vdots\ \cdot^{\cdot^{\cdot}} & \\ & B\ \mathit{true} & \end{array}$$

Now we apply the rule ⊃I (in Figure 2.1) to the conclusion $B\ \mathit{true}$ *with respect to the hypothesis* $\overline{A\ \mathit{true}}$.

That is, the application of the rule $\supset$I designates $\overline{A\ true}$ as its corresponding hypothesis:

$$\cfrac{\begin{array}{ccc} \overline{\Gamma} & \cdots & \overline{A\ true}^{\,x} \\ & \ddots\ \cdot^{\cdot^{\cdot}} & \\ & B\ true & \end{array}}{A \supset B\ true}\ {\supset}\mathsf{I}^x$$

Note that hypotheses in the proof include hypotheses $\overline{\Gamma}$ but *not* $\overline{A\ true}^{\,x}$, which is discharged when the rule $\supset$I is applied. That is, we have a hypothetical proof for the hypothetical judgment $\Gamma \vdash A \supset B$:

$$\cfrac{\begin{array}{ccc} \overline{\Gamma} & \cdots & \overline{A\ true}^{\,x} \\ & \ddots\ \cdot^{\cdot^{\cdot}} & \\ & B\ true & \end{array}}{A \supset B\ true}\ {\supset}\mathsf{I}^x \qquad \Longleftrightarrow \qquad \Gamma \vdash A \supset B\ true$$

Thus we can prove $\Gamma \vdash A \supset B\ true$ whenever we have a proof of $\Gamma, A\ true \vdash B\ true$, which justifies the rule $\supset$I in Figure 2.2.

The use of hypothetical judgments eliminates the need to annotate hypotheses with labels. For example, here is a proof of a hypothetical judgment $\cdot \vdash A \supset (B \supset (A \wedge B))\ true$:

$$\cfrac{\cfrac{\cfrac{\cfrac{}{A\ true, B\ true \vdash A\ true}\ \mathsf{Hyp} \qquad \cfrac{}{A\ true, B\ true \vdash B\ true}\ \mathsf{Hyp}}{A\ true, B\ true \vdash A \wedge B\ true}\ \wedge\mathsf{I}}{A\ true \vdash B \supset (A \wedge B)\ true}\ {\supset}\mathsf{I}}{\cdot \vdash A \supset (B \supset (A \wedge B))\ true}\ {\supset}\mathsf{I}$$

There are two observations to make about the new natural deduction system. First each leaf in a derivation tree for $\Gamma \vdash C\ true$ (where $\Gamma \vdash C\ true$ is regarded as the root) is an application of either the rule Hyp or the rule $\top$I. In particular, if the rule $\top$I is not used (which is often the case), the derivation tree has the following form:

$$\begin{array}{ccc} \cfrac{}{\Gamma_1 \vdash A_1\ true}\ \mathsf{Hyp} & \cdots & \cfrac{}{\Gamma_n \vdash A_n\ true}\ \mathsf{Hyp} \\ & \ddots\ \cdot^{\cdot^{\cdot}} & \\ & \Gamma \vdash C\ true & \end{array}$$

Second the set of antecedents always expands in an inference rule as we move from the conclusion to its premises (*i.e.*, in a bottom-up way). That is, an inference rule $\cfrac{\Gamma' \vdash A\ true \quad \cdots}{\Gamma \vdash C\ true}$ satisfies $\Gamma \subset \Gamma'$. Then the above derivation tree for $\Gamma \vdash C\ true$ satisfies $\Gamma \subset \Gamma_1, \cdots, \Gamma \subset \Gamma_n$.

Weakening and contraction are now stated as follows:

Proposition 2.5 (Structural properties).
(Weakening) *If* $\Gamma \vdash C\ true$, *then* $\Gamma, A\ true \vdash C\ true$.
(Contraction) *If* $\Gamma, A\ true, A\ true \vdash C\ true$, *then* $\Gamma, A\ true \vdash C\ true$.

Proof. By induction on the structure of the proof of $\Gamma \vdash C\ true$ and $\Gamma, A\ true, A\ true \vdash C\ true$. For weakening, the proof of $\Gamma, A\ true \vdash C\ true$ has exactly the same structure as the proof of $\Gamma \vdash C\ true$. When structural induction is applicable to $\Gamma \vdash C\ true$, therefore, we may apply structural induction on $\Gamma, A\ true \vdash C\ true$ instead. A similar observation holds for contraction. □

The provability of the substitution principle confirms that the system in Figure 2.2 adheres to the definition of hypothetical judgments. That is, if the substitution principle was unprovable, it would indicate that some rule in Figure 2.2 was not designed according to the relation between hypothetical judgments and hypothetical proofs.

Theorem 2.6 (Substitution). *If* $\Gamma \vdash A\ true$ *and* $\Gamma, A\ true \vdash C\ true$, *then* $\Gamma \vdash C\ true$.

Exercise 2.7. To which judgment do you think structural induction must be applied in the proof of Theorem 2.6? $\Gamma \vdash A\ \mathit{true}$ or $\Gamma, A\ \mathit{true} \vdash C\ \mathit{true}$? Why?

Before attempting to write a proof of Theorem 2.6, it is worthwhile to predict how the proof would proceed. It helps us, for example, to determine to which of $\Gamma \vdash A\ \mathit{true}$ and $\Gamma, A\ \mathit{true} \vdash C\ \mathit{true}$ structural induction must be applied. For the sake of simplicity, let us assume that the rule $\top\mathsf{I}$ is not used in the proof of $\Gamma, A\ \mathit{true} \vdash C\ \mathit{true}$. (The proof of $\Gamma \vdash A\ \mathit{true}$ may use the rule $\top\mathsf{I}$.) Then the derivation tree for $\Gamma, A\ \mathit{true} \vdash C\ \mathit{true}$ has the following form

$$\begin{array}{ccc} \dfrac{}{\Gamma_1, A\ \mathit{true} \vdash C_1\ \mathit{true}}\ \mathsf{Hyp} & \cdots & \dfrac{}{\Gamma_n, A\ \mathit{true} \vdash C_n\ \mathit{true}}\ \mathsf{Hyp} \\ & \ddots\ \ \cdot^{\cdot^{\cdot}} & \\ & \Gamma, A\ \mathit{true} \vdash C\ \mathit{true} & \end{array}$$

where each leaf $\dfrac{}{\Gamma_i, A\ \mathit{true} \vdash C_i\ \mathit{true}}\ \mathsf{Hyp}$ satisfies $\Gamma \subset \Gamma_i$ for $1 \leq i \leq n$. Now consider the i-the leaf. If $C_i\ \mathit{true} \in \Gamma_i$, the antecedent $A\ \mathit{true}$ in $\Gamma_i, A\ \mathit{true} \vdash C_i\ \mathit{true}$ plays no role in the proof and the leaf is safely replaced by $\dfrac{}{\Gamma_i \vdash C_i\ \mathit{true}}\ \mathsf{Hyp}$. If $C_i = A$, we weaken $\Gamma \vdash A\ \mathit{true} = \Gamma \vdash C_i\ \mathit{true}$ to obtain $\Gamma_i \vdash C_i\ \mathit{true}$, which is then *substituted* for $\Gamma_i, A\ \mathit{true} \vdash C_i\ \mathit{true}$. Now no leaf contains $A\ \mathit{true}$ as an antecedent, and by propagating these changes all the way down to the root, we transform the whole derivation tree into a new one for $\Gamma \vdash C\ \mathit{true}$. Thus we analyze the structure of the proof of $\Gamma, A\ \mathit{true} \vdash C\ \mathit{true}$ to located all leaves in it. That is, we apply structural induction on $\Gamma, A\ \mathit{true} \vdash C\ \mathit{true}$ rather than $\Gamma \vdash A\ \mathit{true}$.

Proof. By induction on the structure of the proof of $\Gamma, A\ \mathit{true} \vdash C\ \mathit{true}$.

We consider three cases Hyp, $\supset\!\mathsf{I}$, and $\supset\!\mathsf{E}$.

Case $\dfrac{C\ \mathit{true} \in \Gamma}{\Gamma, A\ \mathit{true} \vdash C\ \mathit{true}}\ \mathsf{Hyp}$

$\Gamma \vdash C\ \mathit{true}$	by the rule Hyp with $C\ \mathit{true} \in \Gamma$

Case $\dfrac{}{\Gamma, A\ \mathit{true} \vdash C\ \mathit{true}}\ \mathsf{Hyp}$ where $A = C$

$\Gamma \vdash C\ \mathit{true}$	from the assumption $\Gamma \vdash A\ \mathit{true}$

Case $\dfrac{\Gamma, A\ \mathit{true}, C_1\ \mathit{true} \vdash C_2\ \mathit{true}}{\Gamma, A\ \mathit{true} \vdash C_1 \supset C_2\ \mathit{true}}\ \supset\!\mathsf{I}$ where $C = C_1 \supset C_2$

$\Gamma, C_1\ \mathit{true} \vdash A\ \mathit{true}$	by weakening $\Gamma \vdash A\ \mathit{true}$
$\Gamma, C_1\ \mathit{true} \vdash C_2\ \mathit{true}$	by IH on $\Gamma, A\ \mathit{true}, C_1\ \mathit{true} \vdash C_2\ \mathit{true}$ with $\Gamma, C_1\ \mathit{true} \vdash A\ \mathit{true}$
$\Gamma \vdash C_1 \supset C_2\ \mathit{true}$	by the rule $\supset\!\mathsf{I}$ with $\Gamma, C_1\ \mathit{true} \vdash C_2\ \mathit{true}$

Case $\dfrac{\Gamma, A\ \mathit{true} \vdash C' \supset C\ \mathit{true} \quad \Gamma, A\ \mathit{true} \vdash C'\ \mathit{true}}{\Gamma, A\ \mathit{true} \vdash C\ \mathit{true}}\ \supset\!\mathsf{E}$

$\Gamma \vdash C' \supset C\ \mathit{true}$	by IH on $\Gamma, A\ \mathit{true} \vdash C' \supset C\ \mathit{true}$ with $\Gamma \vdash A\ \mathit{true}$
$\Gamma \vdash C'\ \mathit{true}$	by IH on $\Gamma, A\ \mathit{true} \vdash C'\ \mathit{true}$ with $\Gamma \vdash A\ \mathit{true}$
$\Gamma \vdash C\ \mathit{true}$	by the rule $\supset\!\mathsf{E}$ with $\Gamma \vdash C' \supset C\ \mathit{true}$ and $\Gamma \vdash C'\ \mathit{true}$

□

Thus, given a proof $\mathcal{D}$ of $\Gamma \vdash A\ \mathit{true}$ and a proof $\mathcal{E}$ of $\Gamma, A\ \mathit{true} \vdash C\ \mathit{true}$, we can always produce a proof, written as $[\mathcal{D}/A\ \mathit{true}]\mathcal{E}$, of $\Gamma \vdash C\ \mathit{true}$ by substituting $\mathcal{D}$ into $\mathcal{E}$.

2.5 Local soundness and completeness

All the inference rules presented so far seem to make sense intuitively, but their correctness is yet to be established in a formal way. For example, we would certainly expect an elimination rule for $\wedge$ by which $A\ \mathit{true}$ is deducible from $A \wedge B\ \mathit{true}$, but not an elimination rule that deduces $C\ \mathit{true}$ from $A \wedge B\ \mathit{true}$ if C is unrelated to A and B. Then, in designing a natural deduction system, what is the guiding principle to which we can appeal in order to decide whether to accept or reject an inference rule? The answer is that the system must satisfy two properties: *local soundness* and *local completeness*.

An introduction rule compresses the knowledge expressed in its premises into a truth judgment in the conclusion, whereas an elimination rule retrieves the knowledge compressed within a truth judgment in a premise to deduce another judgment in the conclusion. The local soundness property states that the knowledge retrieved from a judgment by an elimination rule is only part of the knowledge compressed within that judgment. Therefore, if local soundness fails, the elimination rule is too strong in the sense that it is capable of contriving some knowledge that cannot be justified by that judgment; thus local soundness ensures that the elimination rule is not too strong. The local completeness property states that the knowledge retrieved from a judgment by an elimination rule includes at least the knowledge compressed within that judgment. Therefore, if local completeness fails, the elimination rule is too weak in the sense that it is incapable of retrieving all the knowledge compressed within that judgment; thus local completeness ensures that the elimination rule is strong enough. If an elimination rule satisfies both properties, it retrieves exactly the same knowledge compressed within a judgment in a premise.

We verify the local soundness property by showing how to reduce a proof in which an introduction rule is immediately followed by a corresponding elimination rule. As an example, consider the following proof in which the introduction rule $\wedge\mathsf{I}$ is immediately followed by its corresponding elimination rule $\wedge\mathsf{E_L}$:

$$\dfrac{\dfrac{\stackrel{\mathcal{D}}{A\ true}\quad \stackrel{\mathcal{E}}{B\ true}}{A \wedge B\ true}\wedge\mathsf{I}}{A\ true}\wedge\mathsf{E_L}$$

The rule $\wedge\mathsf{E_L}$ is not too strong because what it deduces in the conclusion, namely $A\ true$, is one of the two judgments used to deduce $A \wedge B\ true$. Hence the whole proof reduces to a simpler proof $\mathcal{D}$:

$$\dfrac{\dfrac{\stackrel{\mathcal{D}}{A\ true}\quad \stackrel{\mathcal{E}}{B\ true}}{A \wedge B\ true}\wedge\mathsf{I}}{A\ true}\wedge\mathsf{E_L} \qquad \Longrightarrow_R \qquad \stackrel{\mathcal{D}}{A\ true}$$

If the rule $\wedge\mathsf{E_L}$ was too strong (*e.g.*, deducing $A \supset B\ true$ somehow), the proof would not be reducible.

As another example, consider the following proof in which the introduction rule $\supset\mathsf{I}$ is immediately followed by the elimination rule $\supset\mathsf{E}$:

$$\dfrac{\dfrac{\begin{array}{c}\dfrac{}{A\ true}x \\ \vdots \\ B\ true\end{array}}{A \supset B\ true}\supset\mathsf{I}^x \quad \stackrel{\mathcal{D}}{A\ true}}{B\ true}\supset\mathsf{E}$$

The rule $\supset\mathsf{E}$ is not too strong because the whole proof reduces to a smaller proof of the same judgment $B\ true$ by substituting $\mathcal{D}$ for the hypothesis $\dfrac{}{A\ true}x$ in the premise of the rule $\supset\mathsf{I}^x$:

$$\dfrac{\dfrac{\begin{array}{c}\dfrac{}{A\ true}x \\ \vdots \\ B\ true\end{array}}{A \supset B\ true}\supset\mathsf{I}^x \quad \stackrel{\mathcal{D}}{A\ true}}{B\ true}\supset\mathsf{E} \qquad \Longrightarrow_R \qquad \begin{array}{c}\mathcal{D} \\ A\ true \\ \vdots \\ B\ true\end{array}$$

For the natural deduction system based on hypothetical judgments, the substitution principle justifies $\Gamma \vdash B\ true$ when proofs of $\Gamma \vdash A\ true$ and $\Gamma, A\ true \vdash B\ true$ are given:

$$\dfrac{\dfrac{\stackrel{\mathcal{D}}{\Gamma, A\ true \vdash B\ true}}{\Gamma \vdash A \supset B\ true}\supset\mathsf{I} \quad \stackrel{\mathcal{E}}{\Gamma \vdash A\ true}}{\Gamma \vdash B\ true}\supset\mathsf{E} \qquad \Longrightarrow_R \qquad \begin{array}{c}[\mathcal{E}/A\ true]\mathcal{D} \\ \Gamma \vdash B\ true\end{array}$$

The case of $\vee$ is similar to the case of $\supset$ and uses the substitution principle:

$$\dfrac{\dfrac{\overset{\mathcal{D}}{A\ \mathit{true}}}{A \vee B\ \mathit{true}}\,\vee\mathsf{I_L} \qquad \begin{array}{c}\overline{A\ \mathit{true}}^{\,x}\\ \vdots \\ C\ \mathit{true}\end{array} \qquad \begin{array}{c}\overline{B\ \mathit{true}}^{\,y}\\ \vdots\\ C\ \mathit{true}\end{array}}{C\ \mathit{true}}\,\vee\mathsf{E}^{x,y} \qquad \Longrightarrow_R \qquad \begin{array}{c}\mathcal{D}\\ A\ \mathit{true}\\ \vdots \\ C\ \mathit{true}\end{array}$$

$$\dfrac{\dfrac{\overset{\mathcal{D}}{\Gamma \vdash A\ \mathit{true}}}{\Gamma \vdash A \vee B\ \mathit{true}}\,\vee\mathsf{I_L} \qquad \overset{\mathcal{E}_L}{\Gamma, A\ \mathit{true} \vdash C\ \mathit{true}} \qquad \overset{\mathcal{E}_R}{\Gamma, B\ \mathit{true} \vdash C\ \mathit{true}}}{\Gamma \vdash C\ \mathit{true}}\,\vee\mathsf{E} \qquad \Longrightarrow_R \qquad \begin{array}{c}[\mathcal{D}/A\ \mathit{true}]\mathcal{E}_L\\ \Gamma \vdash C\ \mathit{true}\end{array}$$

We refer to these reductions $\Longrightarrow_R$ as *local reductions*. Note that there are no local reductions for $\top$ and $\bot$, since $\top$ has no elimination rule and $\bot$ has no introduction rule.

We verify the local completeness property by showing how to expand a proof of a judgment into another proof in which one or more elimination rules are followed by an introduction rule for the same judgment. As an example, consider a proof $\mathcal{D}$ of $A \wedge B\ \mathit{true}$. The elimination rules $\wedge\mathsf{E_L}$ and $\wedge\mathsf{E_R}$ are not too weak because what they deduce in their conclusions, namely $A\ \mathit{true}$ and $B\ \mathit{true}$, are sufficient to reconstruct another proof of $A \wedge B\ \mathit{true}$:

$$\overset{\mathcal{D}}{A \wedge B\ \mathit{true}} \qquad \Longrightarrow_E \qquad \dfrac{\dfrac{\overset{\mathcal{D}}{A \wedge B\ \mathit{true}}}{A\ \mathit{true}}\,\wedge\mathsf{E_L} \qquad \dfrac{\overset{\mathcal{D}}{A \wedge B\ \mathit{true}}}{B\ \mathit{true}}\,\wedge\mathsf{E_R}}{A \wedge B\ \mathit{true}}\,\wedge\mathsf{I}$$

If the elimination rules were too weak (*e.g.*, being unable to deduce $A\ \mathit{true}$ somehow), the proof would not be expandable.

As another example, consider a proof $\mathcal{D}$ of $A \supset B\ \mathit{true}$. We can reconstruct another proof of the same judgment after applying the elimination rule $\supset$E to $\mathcal{D}$, which implies that the rule $\supset$E is not too weak:

$$\overset{\mathcal{D}}{A \supset B\ \mathit{true}} \qquad \Longrightarrow_E \qquad \dfrac{\dfrac{\overset{\mathcal{D}}{A \supset B\ \mathit{true}} \qquad \overline{A\ \mathit{true}}^{\,x}}{B\ \mathit{true}}\,\supset\mathsf{E}}{A \supset B\ \mathit{true}}\,\supset\mathsf{I}^x$$

In expanding the proof $\mathcal{D}$, we have to choose a fresh label x that is not already in use in $\mathcal{D}$, for any undischarged hypothesis $\overline{B\ \mathit{true}}^{\,x}$ with the same label x in $\mathcal{D}$ becomes associated with the rule $\supset\mathsf{I}^x$, resulting in an incorrect proof if $A \neq B$. For the natural deduction system based on hypothetical judgments, we weaken a proof $\mathcal{D}$ of $\Gamma \vdash A \supset B\ \mathit{true}$ to obtain a proof of $\Gamma, A\ \mathit{true} \vdash A \supset B\ \mathit{true}$ when reconstructing another proof of $\Gamma \vdash A \supset B\ \mathit{true}$:

$$\overset{\mathcal{D}}{\Gamma \vdash A \supset B\ \mathit{true}} \qquad \Longrightarrow_E \qquad \dfrac{\dfrac{\overset{\mathcal{D}}{\Gamma, A\ \mathit{true} \vdash A \supset B\ \mathit{true}} \qquad \dfrac{}{\Gamma, A\ \mathit{true} \vdash A\ \mathit{true}}\,\mathsf{Hyp}}{\Gamma, A\ \mathit{true} \vdash B\ \mathit{true}}\,\supset\mathsf{E}}{\Gamma \vdash A \supset B\ \mathit{true}}\,\supset\mathsf{I}$$

The case for $\vee$ is given as follows:

$$\overset{\mathcal{D}}{A \vee B\ \mathit{true}} \qquad \Longrightarrow_E \qquad \dfrac{\overset{\mathcal{D}}{A \vee B\ \mathit{true}} \qquad \dfrac{\overline{A\ \mathit{true}}^{\,x}}{A \vee B\ \mathit{true}}\,\vee\mathsf{I_L} \qquad \dfrac{\overline{B\ \mathit{true}}^{\,y}}{A \vee B\ \mathit{true}}\,\vee\mathsf{I_R}}{A \vee B\ \mathit{true}}\,\vee\mathsf{E}^{x,y}$$

$$\overset{\mathcal{D}}{\Gamma \vdash A \vee B\ \mathit{true}} \qquad \Longrightarrow_E \qquad \dfrac{\overset{\mathcal{D}}{\Gamma \vdash A \vee B\ \mathit{true}} \qquad \dfrac{\dfrac{}{\Gamma, A\ \mathit{true} \vdash A\ \mathit{true}}\,\mathsf{Hyp}}{\Gamma, A\ \mathit{true} \vdash A \vee B\ \mathit{true}}\,\vee\mathsf{I_L} \qquad \dfrac{\dfrac{}{\Gamma, B\ \mathit{true} \vdash B\ \mathit{true}}\,\mathsf{Hyp}}{\Gamma, B\ \mathit{true} \vdash A \vee B\ \mathit{true}}\,\vee\mathsf{I_R}}{\Gamma \vdash A \vee B\ \mathit{true}}\,\vee\mathsf{E}$$

We refer to these expansions $\Longrightarrow_E$ as *local expansions*.

Although there are no local reductions for $\top$ and $\bot$, there *are* local expansions for $\top$ and $\bot$. Recall that $\top$ and $\bot$ are the nullary cases of conjunction and disjunction, respectively. Hence a proof $\mathcal{D}$ of $\top$ *true* expands to another proof of $\top$ *true* that uses zero elimination rules and thus ignores $\mathcal{D}$:

$$\begin{array}{c}\mathcal{D}\\ \top\ \mathit{true}\end{array} \quad \Longrightarrow_E \quad \dfrac{}{\top\ \mathit{true}}\ \top\mathsf{I}$$

Similarly a proof of $\bot$ *true* expands to another proof of $\bot$ *true* that uses zero introduction rules:

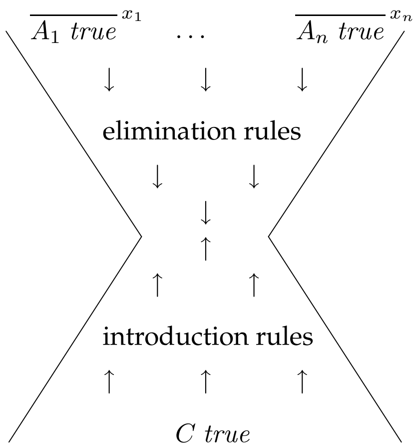
$$\begin{array}{c}\mathcal{D}\\ \bot\ \mathit{true}\end{array} \quad \Longrightarrow_E \quad \dfrac{\begin{array}{c}\mathcal{D}\\ \bot\ \mathit{true}\end{array}}{\bot\ \mathit{true}}\ \bot\mathsf{E}$$

As every connective satisfies local soundness and completeness, the natural deduction system for propositional logic is said to be locally sound and complete. When the system is extended with a new connective, quantifier, or modality, we have to check that the system remains locally sound and complete by finding its local reduction and expansion, as we will see later.

2.6 Normal proofs

We have seen that a proof containing a *detour, i.e.*, an introduction rule immediately followed by a corresponding elimination rule, can be transformed to another proof by applying a local reduction. It turns out that for every proof of A *true*, there is a sequence of local reductions that lead to another proof of A *true* containing no detour (the normalization theorem). We refer to the resultant proof as a *normal proof*. A normal proof is the most direct proof because a detour may be thought of as an example of indirect reasoning. Moreover it is minimal in size in a certain sense, irrespective of the size of its syntactic representation, because it does not reduce to another proof.

Since a normal proof contains no detour, it has the following form where top-down applications of elimination rules meet bottom-up applications of introduction rules in the middle:



Thus the structure of a normal proof conforms to our intuition in building a proof by repeatedly applying introduction rules in a bottom-up way, adding new hypotheses, and repeatedly applying elimination rules in a top-down way, starting from hypotheses. Here is an example of a normal proof of $(A \wedge B) \supset (B \wedge A)$ *true*:

$$\dfrac{\dfrac{\dfrac{\dfrac{}{A \wedge B\ \mathit{true}}\,x}{B\ \mathit{true}}\ \wedge\mathsf{E_R} \qquad \dfrac{\dfrac{}{A \wedge B\ \mathit{true}}\,x}{A\ \mathit{true}}\ \wedge\mathsf{E_L}}{B \wedge A\ \mathit{true}}\ \wedge\mathsf{I}}{(A \wedge B) \supset (B \wedge A)\ \mathit{true}}\ \supset\mathsf{I}^x$$

A proof of $(A \wedge B) \supset (B \wedge A)$ *true* that is not normal contains detours in it:

$$
\dfrac{
\dfrac{
\dfrac{
\dfrac{\dfrac{}{A \wedge B\ \mathit{true}}\,x}{A\ \mathit{true}}\,\wedge\mathsf{E_L}
\qquad
\dfrac{\dfrac{}{A \wedge B\ \mathit{true}}\,x}{B\ \mathit{true}}\,\wedge\mathsf{E_R}
}{A \wedge B\ \mathit{true}}\,\wedge\mathsf{I}\ \textit{(detour)}
}{B\ \mathit{true}}\,\wedge\mathsf{E_R}
\qquad
\dfrac{
\dfrac{
\dfrac{\dfrac{}{A \wedge B\ \mathit{true}}\,x}{A\ \mathit{true}}\,\wedge\mathsf{E_L}
\qquad
\dfrac{\dfrac{}{A \wedge B\ \mathit{true}}\,x}{B\ \mathit{true}}\,\wedge\mathsf{E_R}
}{A \wedge B\ \mathit{true}}\,\wedge\mathsf{I}\ \textit{(detour)}
}{A\ \mathit{true}}\,\wedge\mathsf{E_L}
}{
\dfrac{B \wedge A\ \mathit{true}}{(A \wedge B) \supset (B \wedge A)\ \mathit{true}}\,\supset\mathsf{I}^x
}\,\wedge\mathsf{I}
$$

Normal proofs are an indispensable tool in the study of logic because of their soundness and completeness properties: A *true* holds if and only if there is a normal proof of A *true*. The soundness property holds trivially because a normal proof is just a proof of a special form. The completeness property (that every proof has a corresponding normal proof) has two important consequences. First, in order to prove A *true*, it suffices to find a normal proof of it. When proving A *true*, for example, it is safe to ignore proofs of the following form which cannot be normal proofs:

$$
\dfrac{\begin{matrix}\vdots & \qquad & \vdots \\ B \supset A\ \mathit{true} & \qquad & B\ \mathit{true}\end{matrix}}{A\ \mathit{true}}\,\supset\mathsf{E}
$$

That is, we need to concern ourselves only with the most direct proof rather an indirect proof, for example, by introducing an intermediate proposition B as shown above. Second, in order to refute A *true*, it suffices to try to build a normal proof of it (by alternating between bottom-up applications of introduction rules and top-down applications of elimination rules) and show that the process gets stuck or does not terminate.

Exercise 2.8. Give an informal argument why $\neg\neg A \supset A$ *true* is not provable.

To formalize all these ideas, we introduce two new judgments: *neutral judgments* $A{\downarrow}$ and *normal judgments* $A{\uparrow}$. A neutral judgment $A{\downarrow}$ becomes evident by a *neutral proof* of A *true* which is either a hypothesis or an elimination rule applied to another neutral proof, whereas a normal judgment $A{\uparrow}$ becomes evident by a normal proof of A *true* which is either a neutral judgment or an introduction rule applied to another normal proof. Thus the direction of the arrow in each judgment coincides with the direction in which the proof construction should proceed. Specifically we exploit an existing neutral judgment $A{\downarrow}$ in order to deduce another judgment by determining which elimination rule to be applied; hence the proof construction from a neutral judgment always proceeds downward ($\downarrow$). For a normal judgment $A{\uparrow}$ whose proof is incomplete yet, we determine which introduction rule must be applied in order to deduce it; hence the proof construction from a normal judgment always proceeds upward ($\uparrow$). When a neutral judgment $A{\downarrow}$ meets a normal judgment $A{\uparrow}$ in the middle, the proof construction is finished.

Figure 2.3 shows the inference rules for neutral and normal judgments. The rule $\downarrow\uparrow$, called the *coercion rule*, says that a neutral proof is a normal proof, and a typical construction of a normal proof is completed with an application of the rule $\downarrow\uparrow$. All the other rules are designed according to our intuition on building the most direct proofs, and thus are best read by following the direction of the arrow in each judgment. For example, suppose that we wish to build ($\uparrow$) a new proof of $A \supset B$ *true*:

$$
\begin{matrix}\vdots \\ A \supset B{\uparrow}\end{matrix}
$$

In order to build the most direct proof of $A \supset B$ *true*, we first assume A *true* as a hypothesis which is to be exploited ($\downarrow$) in deducing another judgment:

$$
\begin{matrix}\dfrac{}{A{\downarrow}} \\ \vdots \\ A \supset B{\uparrow}\end{matrix}
$$

$$
\dfrac{\begin{array}{c}\overline{A\!\downarrow}^{\,x}\\ \vdots \\ B\!\uparrow\end{array}}{A \supset B\!\uparrow}\ \supset\!\mathsf{I}_{\uparrow}^{x}
\qquad
\dfrac{A \supset B\!\downarrow \quad A\!\uparrow}{B\!\downarrow}\ \supset\!\mathsf{E}_{\downarrow}
\qquad
\dfrac{A\!\uparrow \quad B\!\uparrow}{A \wedge B\!\uparrow}\ \wedge\mathsf{I}_{\uparrow}
\qquad
\dfrac{A \wedge B\!\downarrow}{A\!\downarrow}\ \wedge\mathsf{E}_{\mathsf{L}\downarrow}
\qquad
\dfrac{A \wedge B\!\downarrow}{B\!\downarrow}\ \wedge\mathsf{E}_{\mathsf{R}\downarrow}
$$

$$
\dfrac{A\!\uparrow}{A \vee B\!\uparrow}\ \vee\mathsf{I}_{\mathsf{L}\uparrow}
\qquad
\dfrac{B\!\uparrow}{A \vee B\!\uparrow}\ \vee\mathsf{I}_{\mathsf{R}\uparrow}
\qquad
\dfrac{A \vee B\!\downarrow \quad \begin{array}{c}\overline{A\!\downarrow}^{\,x}\\ \vdots \\ C\!\uparrow\end{array} \quad \begin{array}{c}\overline{B\!\downarrow}^{\,y}\\ \vdots \\ C\!\uparrow\end{array}}{C\!\uparrow}\ \vee\mathsf{E}_{\uparrow}^{x,y}
$$

$$
\dfrac{}{\top\!\uparrow}\ \top\mathsf{I}_{\uparrow}
\qquad
\dfrac{\bot\!\downarrow}{C\!\uparrow}\ \bot\mathsf{E}_{\uparrow}
\qquad
\dfrac{A\!\downarrow}{A\!\uparrow}\ {\downarrow}\!{\uparrow}
\qquad
\dfrac{\begin{array}{c}\overline{A\!\downarrow}^{\,x}\\ \vdots \\ \bot\!\uparrow\end{array}}{\neg A\!\uparrow}\ \neg\mathsf{I}_{\uparrow}{}^{x}
\qquad
\dfrac{\neg A\!\downarrow \quad A\!\uparrow}{\bot\!\downarrow}\ \neg\mathsf{E}_{\downarrow}
$$

Figure 2.3: Inference rules for neutral and normal judgments

Then we try to build (↑) a proof of B *true*, which is precisely what the rule $\supset\!\mathsf{I}_{\uparrow}$ expresses:

$$
\dfrac{\begin{array}{c}\overline{A\!\downarrow}\\ \vdots \\ B\!\uparrow\end{array}}{A \supset B\!\uparrow}
$$

As another example, suppose that we wish to exploit (↓) an existing proof of $A \supset B$ *true*:

$$
\begin{array}{c} A \supset B\!\downarrow \\ \vdots \end{array}
$$

In order to exploit it in the most direct manner, we need a proof of A *true*, which we do not have yet. Therefore we first build (↑) a new proof of A *true*:

$$
\begin{array}{c} A \supset B\!\downarrow \quad A\!\uparrow \\ \vdots \end{array}
$$

A proof of A *true* then allows us to deduce B *true*. Since we now have a proof of B *true* ready for use in deducing another judgment, we classify it as a neutral judgment, which is precisely what the rule $\supset\!\mathsf{E}_{\downarrow}$ expresses:

$$
\dfrac{A \supset B\!\downarrow \quad A\!\uparrow}{B\!\downarrow}
$$

Exercise 2.9. Analyze all the remaining rules in an analogous way. Note that the rule $\vee\mathsf{E}_{\uparrow}$ superficially deduces a normal judgment $C\!\uparrow$ by applying an elimination rule, thereby contradicting our intuition on a normal judgment which is supposed to be either a neutral judgment or an introduction rule applied to another normal judgment. The essence of the proof of $C\!\uparrow$, however, is found not in the application of the rule $\vee\mathsf{E}_{\uparrow}$ itself but in the two premises deducing $C\!\uparrow$. In this regard, the rule $\vee\mathsf{E}_{\uparrow}$ still adheres to our intuition on normal judgments. The rules $\top\mathsf{I}_{\uparrow}$ and $\bot\mathsf{E}_{\uparrow}$ are obtained as the nullary cases of $\wedge\mathsf{I}_{\uparrow}$ and $\vee\mathsf{E}_{\uparrow}$, respectively.

The rules in Figure 2.3 are all designed in such a way that a proof of a neutral or normal judgment contains no detour. First observe that no proof of a neutral judgment ends with an application of an introduction rule (see the rules $\supset\!\mathsf{E}_{\downarrow}$, $\wedge\mathsf{E}_{\mathsf{L}\downarrow}$, $\wedge\mathsf{E}_{\mathsf{R}\downarrow}$). Then observe that the principal premise in each elimination rule is a neutral judgment (*e.g.*, $A \supset B\!\downarrow$ in the rule $\supset\!\mathsf{E}_{\downarrow}$), which has been shown not to end with an introduction rule and thus does not give rise to a detour.

$$\frac{}{\Gamma_\downarrow, A\!\downarrow\ \vdash A\!\downarrow}\ \mathsf{Hyp}_\downarrow \qquad \frac{\Gamma_\downarrow, A\!\downarrow\ \vdash B\!\uparrow}{\Gamma_\downarrow \vdash A \supset B\!\uparrow}\ {\supset}\mathsf{I}_\uparrow \qquad \frac{\Gamma_\downarrow \vdash A \supset B\!\downarrow \quad \Gamma_\downarrow \vdash A\!\uparrow}{\Gamma_\downarrow \vdash B\!\downarrow}\ {\supset}\mathsf{E}_\downarrow$$

$$\frac{\Gamma_\downarrow \vdash A\!\uparrow \quad \Gamma_\downarrow \vdash B\!\uparrow}{\Gamma_\downarrow \vdash A \wedge B\!\uparrow}\ {\wedge}\mathsf{I}_\uparrow \qquad \frac{\Gamma_\downarrow \vdash A \wedge B\!\downarrow}{\Gamma_\downarrow \vdash A\!\downarrow}\ {\wedge}\mathsf{E_L}_\downarrow \qquad \frac{\Gamma_\downarrow \vdash A \wedge B\!\downarrow}{\Gamma_\downarrow \vdash B\!\downarrow}\ {\wedge}\mathsf{E_R}_\downarrow$$

$$\frac{\Gamma_\downarrow \vdash A\!\uparrow}{\Gamma_\downarrow \vdash A \vee B\!\uparrow}\ {\vee}\mathsf{I_L}_\uparrow \qquad \frac{\Gamma_\downarrow \vdash B\!\uparrow}{\Gamma_\downarrow \vdash A \vee B\!\uparrow}\ {\vee}\mathsf{I_R}_\uparrow \qquad \frac{\Gamma_\downarrow \vdash A \vee B\!\downarrow \quad \Gamma_\downarrow, A\!\downarrow\ \vdash C\!\uparrow \quad \Gamma_\downarrow, B\!\downarrow\ \vdash C\!\uparrow}{\Gamma_\downarrow \vdash C\!\uparrow}\ {\vee}\mathsf{E}_\uparrow$$

$$\frac{}{\Gamma_\downarrow \vdash \top\!\uparrow}\ \top\mathsf{I}_\uparrow \qquad \frac{\Gamma_\downarrow \vdash \bot\!\downarrow}{\Gamma_\downarrow \vdash C\!\uparrow}\ \bot\mathsf{E}_\uparrow \qquad \frac{\Gamma_\downarrow \vdash A\!\downarrow}{\Gamma_\downarrow \vdash A\!\uparrow}\ {\downarrow\!\uparrow} \qquad \frac{\Gamma_\downarrow, A\!\downarrow\ \vdash \bot\!\uparrow}{\Gamma_\downarrow \vdash \neg A\!\uparrow}\ \neg\mathsf{I}_\uparrow \qquad \frac{\Gamma_\downarrow \vdash \neg A\!\downarrow \quad \Gamma_\downarrow \vdash A\!\uparrow}{\Gamma_\downarrow \vdash \bot\!\downarrow}\ \neg\mathsf{E}_\downarrow$$

Figure 2.4: Inference rules for neutral and normal judgments using hypothetical judgments

As an example, we show that the proof of $(A \wedge B) \supset (B \wedge A)$ $true$ given earlier is indeed a normal proof by rewriting it in terms of neutral and normal judgments; we annotate each judgment in it with either $\downarrow$ or $\uparrow$ according to the rules in Figure 2.3 and check that no conflicting annotation arises:

$$\frac{\dfrac{\dfrac{\dfrac{\dfrac{}{A \wedge B\!\downarrow}\,x}{B\!\downarrow}\,{\wedge}\mathsf{E_R}_\downarrow}{B\!\uparrow}\,{\downarrow\!\uparrow} \quad \dfrac{\dfrac{\dfrac{}{A \wedge B\!\downarrow}\,x}{A\!\downarrow}\,{\wedge}\mathsf{E_L}_\downarrow}{A\!\uparrow}\,{\downarrow\!\uparrow}}{B \wedge A\!\uparrow}\,{\wedge}\mathsf{I}_\uparrow}{(A \wedge B) \supset (B \wedge A)\!\uparrow}\,{\supset}\mathsf{I}^x_\uparrow$$

Note that a detour is impossible to annotate with arrows $\downarrow$ and $\uparrow$:

$$\frac{\dfrac{A\!\uparrow \quad B\!\uparrow}{A \wedge B \uparrow ?\downarrow ?}\,{\wedge}\mathsf{I}_\uparrow}{A\!\downarrow}\,{\wedge}\mathsf{E_L}_\downarrow \qquad \frac{\dfrac{\begin{array}{c}\overline{A\!\downarrow}^{\,x}\\ \vdots \\ B\!\uparrow\end{array}}{A \supset B \uparrow ?\downarrow ?}\,{\supset}\mathsf{I}^x_\uparrow \quad A\!\uparrow}{B\!\downarrow}\,{\supset}\mathsf{E}_\downarrow \qquad \frac{\dfrac{A\!\uparrow}{A \vee B \uparrow ?\downarrow ?}\,{\vee}\mathsf{I_L}_\uparrow \quad \begin{array}{c}\overline{A\!\downarrow}^{\,x}\\ \vdots \\ C\!\uparrow\end{array} \quad \begin{array}{c}\overline{B\!\downarrow}^{\,y}\\ \vdots \\ C\!\uparrow\end{array}}{C\!\uparrow}\,{\vee}\mathsf{E}^{x,y}_\uparrow$$

As another example, we show that no proof of $A \vee \neg A\!\uparrow$ exists where A is an arbitrary proposition:

$$\frac{\begin{array}{c}\textit{(stuck)}\\ A\!\uparrow\end{array}}{A \vee \neg A\!\uparrow}\,{\vee}\mathsf{I_L}_\uparrow \qquad \frac{\dfrac{\begin{array}{c}\overline{A\!\downarrow}^{\,x}\\ \textit{(stuck)}\\ \vdots \\ \textit{(stuck)}\\ \bot\!\uparrow\end{array}}{\neg A\!\uparrow}\,{\supset}\mathsf{I}^x_\uparrow}{A \vee \neg A\!\uparrow}\,{\vee}\mathsf{I_R}_\uparrow$$

(Hence $A \vee \neg A$ $true$ is not provable in constructive logic, although it is a tautology in classical logic.)

Figure 2.4 shows an equivalent system for neutral and normal judgments using hypothetical judgments $\Gamma_\downarrow \vdash A\!\uparrow$ and $\Gamma_\downarrow \vdash A\!\downarrow$, where $\Gamma_\downarrow = \{A\!\downarrow \mid A \in \Gamma\}$ is a collection of neutral judgments and the exchange rule is built-in; we reuse the inference rule names from the previous system for neutral and normal judgments. The two structural properties, weakening and contraction, are stated as expected. As it is based on hypothetical judgments, the system also satisfies the substitution principle.

Proposition 2.10 (Structural properties).

(Weakening) *If* $\Gamma_\downarrow \vdash C\!\downarrow$, *then* $\Gamma_\downarrow, A\!\downarrow\ \vdash C\!\downarrow$.
If $\Gamma_\downarrow \vdash C\!\uparrow$, *then* $\Gamma_\downarrow, A\!\downarrow\ \vdash C\!\uparrow$.

(Contraction) *If* $\Gamma_\downarrow, A\!\downarrow, A\!\downarrow\ \vdash C\!\downarrow$, *then* $\Gamma_\downarrow, A\!\downarrow\ \vdash C\!\downarrow$.
If $\Gamma_\downarrow, A\!\downarrow, A\!\downarrow\ \vdash C\!\uparrow$, *then* $\Gamma_\downarrow, A\!\downarrow\ \vdash C\!\uparrow$.

Theorem 2.11 (Substitution).
If $\Gamma_\downarrow \vdash A\downarrow$ *and* $\Gamma_\downarrow, A\downarrow \vdash C\downarrow$, *then* $\Gamma_\downarrow \vdash C\downarrow$.
If $\Gamma_\downarrow \vdash A\downarrow$ *and* $\Gamma_\downarrow, A\downarrow \vdash C\uparrow$, *then* $\Gamma_\downarrow \vdash C\uparrow$.

Proof. By induction on the structure of the proof of $\Gamma_\downarrow, A\downarrow \vdash C\downarrow$ and $\Gamma_\downarrow, A\downarrow \vdash C\uparrow$. □

2.7 Normalization

We state the soundness and completeness properties of normal proofs using normal judgments as follows:

Theorem 2.12.
(Soundness) *If* $A\uparrow$ *is provable, then* $A\ \mathit{true}$ *is provable.*
(Completeness) *If* $A\ \mathit{true}$ *is provable, then* $A\uparrow$ *is provable.*

The soundness property is easy to show because every rule in Figure 2.3 is transformed into a corresponding rule in Figure 2.1 by replacing all neutral and normal judgments by truth judgments (*i.e.*, $A\uparrow$ and $A\downarrow$ by $A\ \mathit{true}$). In conjunction with the fact that a proof of a neutral or normal judgment contains no detour, it implies that $A\uparrow$ expresses a particular strategy for proving $A\ \mathit{true}$, namely a strategy that disallows detours. For the completeness property, we prove the normalization theorem shown below. For the moment, we do not consider disjunction $\lor$ and falsehood $\bot$. (Chapter 4 gives another formal proof covering all connectives in propositional logic.)

Theorem 2.13 (Normalization). *For every proof of* $A\ \mathit{true}$, *there is a sequence of local reductions that lead to a proof of* $A\uparrow$. *That is, the proof of* $A\ \mathit{true}$ *resulting from the sequence of local reductions can be annotated with arrows* $\downarrow$ *and* $\uparrow$ *according to the rules in Figure 2.3.*

Another theorem, called the *strong normalization theorem*, states that every sequence of local reductions, *regardless of their order*, eventually terminates to yield a normal proof:

Theorem 2.14 (Strong normalization). *Every sequence of local reductions starting from a proof of* $A\ \mathit{true}$ *terminates with a proof of* $A\uparrow$. *That is, there is no infinite sequence of local reductions starting from any proof of* $A\ \mathit{true}$.

In the presence of disjunction $\lor$ and falsehood $\bot$ (*i.e.*, in the full system of propositional logic), however, the normalization theorem fails! That is, a proof to which no local reduction can be applied is *not* necessarily a normal proof. (The converse that a normal proof contains no detour still holds.) For example, the following proof of $(A \lor A) \supset A\ \mathit{true}$ is not normal because annotating each judgment in it with either $\downarrow$ or $\uparrow$ fails in the conclusion of the rule $\lor\mathsf{E}^{x,y}$; here and henceforth, we abbreviate $\dfrac{A\downarrow}{A\uparrow}\ \downarrow\uparrow$ as $A\downarrow\uparrow$:

$$\dfrac{\dfrac{\dfrac{\dfrac{}{A \lor A\ \mathit{true}}\,z \quad \dfrac{\dfrac{}{A\ \mathit{true}}\,x \quad \dfrac{}{A\ \mathit{true}}\,x}{A \land A\ \mathit{true}}\,\land\mathsf{I} \quad \dfrac{\dfrac{}{A\ \mathit{true}}\,y \quad \dfrac{}{A\ \mathit{true}}\,y}{A \land A\ \mathit{true}}\,\land\mathsf{I}}{A \land A\ \mathit{true}}\,\lor\mathsf{E}^{x,y}}{A\ \mathit{true}}\,\land\mathsf{E_L}}{(A \lor A) \supset A\ \mathit{true}}\,\supset\mathsf{I}^z$$

$$\dfrac{\dfrac{\dfrac{\dfrac{}{A \lor A\downarrow}\,z \quad \dfrac{\dfrac{}{A\downarrow\uparrow}\,x \quad \dfrac{}{A\downarrow\uparrow}\,x}{A \land A\uparrow}\,\land\mathsf{I}_\uparrow \quad \dfrac{\dfrac{}{A\downarrow\uparrow}\,y \quad \dfrac{}{A\downarrow\uparrow}\,y}{A \land A\uparrow}\,\land\mathsf{I}_\uparrow}{A \land A\uparrow ?\downarrow ?}\,\lor\mathsf{E}^{x,y}_\uparrow}{A\downarrow\uparrow}\,\land\mathsf{E}_{\mathsf{L}\downarrow}}{(A \lor A) \supset A\uparrow}\,\supset\mathsf{I}^z_\uparrow$$

The proof, however, contains no detour: no introduction rule is immediately followed by its corresponding elimination rule. Note that the rule $\land\mathsf{I}$ (deducing $A \land A\ \mathit{true}$) is followed by an elimination rule, which, however, is not a corresponding elimination rule (like $\land\mathsf{E_L}$ or $\land\mathsf{E_R}$).

The example suggests that in order to reduce an arbitrary proof to a normal proof, we need another strategy for transforming proofs involving disjunction $\lor$ and falsehood $\bot$. It turns out that we need *commuting conversions* $\Longrightarrow_C$:

$$\dfrac{\dfrac{\begin{array}{c}\mathcal{D}\\ A\lor B\ \mathit{true}\end{array}\quad\begin{array}{c}\overline{A\ \mathit{true}}^{\,x}\\ \vdots\\ C\ \mathit{true}\end{array}\quad\begin{array}{c}\overline{B\ \mathit{true}}^{\,y}\\ \vdots\\ C\ \mathit{true}\end{array}}{C\ \mathit{true}}\,\lor\mathsf{E}^{x,y}}{C'\ \mathit{true}}\,R \qquad\Longrightarrow_C\qquad \dfrac{\begin{array}{c}\mathcal{D}\\ A\lor B\ \mathit{true}\end{array}\quad\begin{array}{c}\overline{A\ \mathit{true}}^{\,x}\\ \vdots\\ \dfrac{C\ \mathit{true}}{C'\ \mathit{true}}\,R\end{array}\quad\begin{array}{c}\overline{B\ \mathit{true}}^{\,y}\\ \vdots\\ \dfrac{C\ \mathit{true}}{C'\ \mathit{true}}\,R\end{array}}{C'\ \mathit{true}}\,\lor\mathsf{E}^{x,y}$$

Here the rule R is assumed to be an elimination rule, since there is no point in applying a commuting conversion when the rule R is an introduction rule. Note that a commuting conversion allows us to effectively ignore the elimination rule $\lor\mathsf{E}$ lying *between* the rule for proving $C\ \mathit{true}$ in the second or third premise and the rule R for proving $C'\ \mathit{true}$ from $C\ \mathit{true}$ in the conclusion. In a certain sense, the only role that the conclusion in the rule $\lor\mathsf{E}$ plays is to indicate that both hypotheses $\overline{A\ \mathit{true}}^{\,x}$ and $\overline{B\ \mathit{true}}^{\,y}$ lead to the same conclusion $C\ \mathit{true}$, instead of two different conclusions, say $C_1\ \mathit{true}$ and $C_2\ \mathit{true}$. In other words, $C\ \mathit{true}$ in the conclusion makes no contribution to the proof because it is the two premises that actually prove $C\ \mathit{true}$. Therefore the rule $\lor\mathsf{E}$ may be ignored as far as deducing another judgment from $C\ \mathit{true}$ in the conclusion is concerned. If we chose the following elimination rule for $\lor$ with a side condition that both hypotheses $\overline{A\ \mathit{true}}^{\,x}$ and $\overline{B\ \mathit{true}}^{\,y}$ lead to the same conclusion, no commuting conversion would be necessary:

$$\begin{array}{c}\dfrac{A\lor B\ \mathit{true}}{\overline{A\ \mathit{true}}^{\,x}\quad\overline{B\ \mathit{true}}^{\,y}}\,\lor\mathsf{E}^{x,y}\\ \vdots\qquad\quad\vdots\\ C\ \mathit{true}\quad C\ \mathit{true}\end{array}$$

Now applying a commuting conversion to the proof of $(A\lor A)\supset A\ \mathit{true}$ shown above yields another proof of the same judgment, to which a local reduction can be applied:

$$\cdots\quad\Longrightarrow_C\quad \dfrac{\dfrac{\overline{A\lor A\ \mathit{true}}^{\,z}\quad \dfrac{\dfrac{\overline{A\ \mathit{true}}^{\,x}\quad\overline{A\ \mathit{true}}^{\,x}}{A\land A\ \mathit{true}}\,\land\mathsf{I}}{A\ \mathit{true}}\,\land\mathsf{E_L}\quad \dfrac{\dfrac{\overline{A\ \mathit{true}}^{\,y}\quad\overline{A\ \mathit{true}}^{\,y}}{A\land A\ \mathit{true}}\,\land\mathsf{I}}{A\ \mathit{true}}\,\land\mathsf{E_L}}{A\ \mathit{true}}\,\lor\mathsf{E}^{x,y}}{(A\lor A)\supset A\ \mathit{true}}\,\supset\mathsf{I}^{z}$$

After removing the two detours in it, we obtain a normal proof annotated with arrows $\downarrow$ and $\uparrow$:

$$\dfrac{\dfrac{\overline{A\lor A\downarrow}^{\,z}\quad\overline{A\downarrow\uparrow}^{\,x}\quad\overline{A\downarrow\uparrow}^{\,y}}{A\uparrow}\,\lor\mathsf{E}^{x,y}_{\uparrow}}{(A\lor A)\supset A\uparrow}\,\supset\mathsf{I}^{z}_{\uparrow}$$

Exercise 2.15. Specify a commuting conversion for $\bot$. Recall that $\bot$ is the nullary case of $\lor$.

2.8 Long normal proofs

While the normalization theorem guarantees the existence of a proof of $A\uparrow$ for every proof of $A\ \mathit{true}$, it does not address the uniqueness of proofs of $A\uparrow$. In fact, such a proof of $A\uparrow$ is not always unique! To see why, observe that the rule $\dfrac{A\downarrow}{A\uparrow}\,\downarrow\uparrow$ has no restriction on proposition A. Therefore, if a proof of $A\downarrow$ is given where A is not an atomic proposition (*e.g.*, $A = A_1\supset A_2$), we may either appeal to the rule $\downarrow\uparrow$ to deduce $A\uparrow$ immediately, or apply an elimination rule to $A\downarrow$ to later build a proof of $A\uparrow$ by applying an introduction rule. For example, we may prove $(A\supset B)\supset(A\supset B)\uparrow$ by applying the rule $\downarrow\uparrow$ to $A\supset B\downarrow$:

$$\dfrac{\dfrac{\overline{A\supset B\downarrow}^{\,x}}{A\supset B\uparrow}\,\downarrow\uparrow}{(A\supset B)\supset(A\supset B)\uparrow}\,\supset\mathsf{I}^{x}_{\uparrow}$$

Alternatively we may prove the same judgment $(A \supset B) \supset (A \supset B)\uparrow$ by decomposing $A \supset B\downarrow$ until the rule $\downarrow\uparrow$ is applied to $B\downarrow$ for an atomic proposition B:

$$\dfrac{\dfrac{\dfrac{\dfrac{\overline{A \supset B\downarrow}^{\,x} \qquad \dfrac{\overline{A\downarrow}^{\,y}}{A\uparrow}\,{\downarrow\uparrow}}{B\downarrow}\,{\supset}\mathsf{E}_{\downarrow}}{B\uparrow}\,{\downarrow\uparrow}}{A \supset B\uparrow}\,{\supset}\mathsf{I}^{y}_{\uparrow}}{(A \supset B) \supset (A \supset B)\uparrow}\,{\supset}\mathsf{I}^{x}_{\uparrow}$$

If we require that proposition A in the rule $\downarrow\uparrow$ be atomic, top-down applications of elimination rules meet bottom-up applications of introduction rules only through atomic propositions. Thus every normal proof applies elimination rules until only neutral judgments $A\downarrow$ for atomic propositions A remain, and starts to apply introduction rules only to normal judgments $A\uparrow$ for atomic propositions A. We call such normal proofs as *long normal proofs*. For example, the second proof of $(A \supset B) \supset (A \supset B)\uparrow$ shown above is a long normal proof while the first proof is not.

Now consider the system in Figure 2.3 in which proposition A in the rule $\downarrow\uparrow$ is required to be atomic:

$$\dfrac{A\downarrow}{A\uparrow}\,{\downarrow\uparrow}\ (\textit{A atomic})$$

If we can show that the original rule $\downarrow\uparrow$ (without the requirement on proposition A) is derivable, all the elimination rules in the system are strong enough in the sense that even if all propositions are decomposed into atomic propositions by elimination rules, no knowledge is essentially lost. (Here it helps to think of $A\uparrow$ and $A\downarrow$ as expressing a particular strategy for proving $A\ \mathit{true}$.) As a property of *all* elimination rules collectively, it is called the *global completeness* property. (Recall that the local completeness property states that a *specific* elimination rule is strong enough.)

The system in Figure 2.3 satisfies the global completeness property. We inductively show that the original rule $\downarrow\uparrow$ without the requirement on proposition A is derivable.

Proposition 2.16. *The rule* $\dfrac{A\downarrow}{A\uparrow}$ *is derivable.*

Proof. By induction on the structure of proposition A. If A is atomic, we apply the new rule $\downarrow\uparrow$ (with the requirement on proposition A). We show the case $A = A_1 \supset A_2$:

$$\dfrac{\dfrac{\dfrac{A_1 \supset A_2\downarrow \qquad \dfrac{\overline{A_1\downarrow}^{\,x}}{A_1\uparrow}\,\textit{IH on } A_1}{A_2\downarrow}\,{\supset}\mathsf{E}_{\downarrow}}{A_2\uparrow}\,\textit{IH on } A_2}{A_1 \supset A_2\uparrow}\,{\supset}\mathsf{I}^{x}_{\uparrow}$$

□

Exercise 2.17. Complete the proof of Proposition 2.16.

Chapter 3

Proof Terms

This chapter presents an alternative formulation of propositional logic using the principle called the *Curry-Howard isomorphism* [**?**]. As a principle connecting logic and programming languages, it states that propositions in logic correspond to types in programming languages (*propositions-as-types* correspondence) and that proofs in logic correspond to programs in programming languages (*proofs-as-programs* correspondence). Thus, by applying the Curry-Howard isomorphism to a formulation of logic, we systematically derive a formulation of a corresponding programming language. In the case of propositional logic, we obtain a basic definition of the simply-typed λ-calculus.

3.1 Proof terms

The basic idea behind the Curry-Howard isomorphism is to represent a proof $\mathcal{D}$ of a truth judgment $A\ \mathit{true}$ as a *proof term* M of *type* A:

$$\begin{array}{c}\mathcal{D}\\ A\ \mathit{true}\end{array} \quad \Longleftrightarrow \quad M : A$$

That is, a *typing judgment* $M : A$ expresses that a proof term M of type A is a (concise) representation of a proof of $A\ \mathit{true}$. When $M : A$ holds, we say that proof term M typechecks with type A. Note that A can be interpreted both as a proposition and as a type, depending on the context in which it is used.

Under the correspondence between proofs and proof terms shown above, each inference rule for deducing truth judgments is translated to a corresponding *typing rule* for deducing typing judgments; by convention, a typing rule is given the same name as the inference rule from which it is derived:

$$\frac{\cdots}{A\ \mathit{true}}\ R \quad \Longleftrightarrow \quad \frac{\cdots}{M : A}\ R$$

Thus the typing rules for proof terms constitute another natural deduction system, in which an introduction rule assigns to a proof term a type involving a particular connective whereas an elimination rule uses such a proof term in its premise.

We may choose any syntax for proof terms as long as each proof term of type A provides all necessary information to extract a corresponding proof of $A\ \mathit{true}$. Below we design proof terms according to the syntax for the simply-typed λ-calculus so as to emphasize the close connection between logic and type theory. We use metavariables M, N, $\cdots$ for terms. Figure 3.1 shows all the typing rules for proof terms in propositional logic where the set of proof terms is inductively defined as follows:

$$\begin{array}{lrcl}\text{proof term} & M & ::= & x \mid (M, M) \mid \mathsf{fst}\ M \mid \mathsf{snd}\ M \mid \lambda x\!:\!A.\, M \mid M\ M \mid \\ & & & \mathsf{inl}_A\ M \mid \mathsf{inr}_A\ M \mid \mathsf{case}\ M\ \mathsf{of}\ \mathsf{inl}\ x.\, M \mid \mathsf{inr}\ x.\, M \mid () \mid \mathsf{abort}_A\ M\end{array}$$

Conjunction

Consider an application of the rule $\wedge\mathsf{I}$ in which a proof $\mathcal{D}$ of $A \wedge B\ \mathit{true}$ is constructed from a proof $\mathcal{D}_A$ of $A\ \mathit{true}$ and a proof $\mathcal{D}_B$ of $B\ \mathit{true}$. If proof terms M and N represent $\mathcal{D}_A$ and $\mathcal{D}_B$, respectively, we use a *product term* (M, N) of type $A \wedge B$ to represent $\mathcal{D}$. Thus the rule $\wedge\mathsf{I}$ is translated to the following typing rule (of the same name):

$$\frac{A\ true \quad B\ true}{A \wedge B\ true}\ \wedge\mathsf{I} \quad \Longleftrightarrow \quad \frac{M : A \quad N : B}{(M, N) : A \wedge B}\ \wedge\mathsf{I}$$

We use *projection terms* $\mathsf{fst}\ M$ and $\mathsf{snd}\ M$ in translating the rule $\wedge\mathsf{E_L}$ and $\wedge\mathsf{E_R}$; fst and snd stand for 'first projection' and 'second projection,' respectively:

$$\frac{A \wedge B\ true}{A\ true}\ \wedge\mathsf{E_L} \quad \Longleftrightarrow \quad \frac{M : A \wedge B}{\mathsf{fst}\ M : A}\ \wedge\mathsf{E_L} \qquad \frac{A \wedge B\ true}{B\ true}\ \wedge\mathsf{E_R} \quad \Longleftrightarrow \quad \frac{M : A \wedge B}{\mathsf{snd}\ M : B}\ \wedge\mathsf{E_R}$$

Implication

Suppose that we wish to convert to a proof term a proof $\mathcal{D}$ of $A \supset B\ true$ that applies the rule $\supset\mathsf{I}$ to a hypothetical proof $\mathcal{E}$ of $B\ true$:

$$\mathcal{D}\left\{\begin{array}{c} \mathcal{E}\left\{\begin{array}{c} \overline{A\ true}^{\,x} \\ \vdots \\ B\ true \end{array}\right. \\ \hline A \supset B\ true \end{array}\right. \supset\mathsf{I}^x$$

In order to build a proof term M representing $\mathcal{E}$, we first need to assign a proof term to the hypothesis $\overline{A\ true}^{\,x}$. Since $A\ true$ is just a hypothesis without a concrete proof, its corresponding proof term is also unknown. Hence we represent $\overline{A\ true}^{\,x}$ as a *variable* x, for which we can later substitute another proof term (like we substitute a concrete proof of $A\ true$ for the hypothesis $\overline{A\ true}^{\,x}$):

$$\overline{A\ true}^{\,x} \quad \Longleftrightarrow \quad \overline{x : A}$$

If M represents $\mathcal{E}$, we use a *λ-abstraction* $\lambda x\!:\!A.\,M$ to represent $\mathcal{D}$:

$$\begin{array}{c} \overline{A\ true}^{\,x} \\ \vdots \\ B\ true \\ \hline A \supset B\ true \end{array}\ \supset\mathsf{I}^x \quad \Longleftrightarrow \quad \begin{array}{c} \overline{x : A} \\ \vdots \\ M : B \\ \hline \lambda x\!:\!A.\,M : A \supset B \end{array}\ \supset\mathsf{I}$$

We say that variable x is bound in the λ-abstraction $\lambda x\!:\!A.\,M$. Note that we may rename x to another variable without changing the meaning of $\lambda x\!:\!A.\,M$. For example, both $\lambda x\!:\!A.\,(x, x)$ and $\lambda y\!:\!A.\,(y, y)$ represent the same proof, since using a different label for the same hypothesis does not alter the structure of the proof. (Renaming a bound variable in a λ-abstraction is commonly called *α-conversion.*)

Similarly to the rule $\supset\mathsf{I}$ in propositional logic, the typing rule $\supset\mathsf{I}$ restricts the scope of the hypothesis $\overline{x : A}$ to its premise. As a result, the hypothesis $\overline{x : A}$ is discharged when the rule $\supset\mathsf{I}$ is applied, and variable x in $\lambda x\!:\!A.\,M$ can be assigned type A only if it appears within M. For example, $\lambda x\!:\!A.\,x$ has type $A \supset A$, but $(\lambda x\!:\!A.\,x, x)$ cannot be assigned a type and fails to typecheck. Also the hypothesis $\overline{x : A}$ may be used not just once but as many times as necessary. Hence proof term M in $\lambda x\!:\!A.\,M$ may contain any number of occurrences of variable x, as illustrated below:

$$\dfrac{\dfrac{\overline{x : B} \quad \overset{\overline{y : A}}{\textit{(not used in the proof)}}}{\lambda y\!:\!A.\,x : A \supset B}\ \supset\mathsf{I}}{\lambda x\!:\!B.\,\lambda y\!:\!A.\,x : B \supset (A \supset B)}\ \supset\mathsf{I} \qquad \dfrac{\overline{x : A}}{\lambda x\!:\!A.\,x : A \supset A}\ \supset\mathsf{I} \qquad \dfrac{\dfrac{\overline{x : A} \quad \overline{x : A}}{(x, x) : A \wedge A}\ \wedge\mathsf{I}}{\lambda x\!:\!A.\,(x, x) : A \supset (A \wedge A)}\ \supset\mathsf{I}$$

(See Page 20 for proofs of corresponding truth judgments.)

As a proof term corresponding to the rule $\supset\mathsf{E}$, we use a *λ-application* $M\ N$:

$$\frac{A \supset B\ true \quad A\ true}{B\ true}\ \supset\mathsf{E} \quad \Longleftrightarrow \quad \frac{M : A \supset B \quad N : A}{M\ N : B}\ \supset\mathsf{E}$$

The following example uses the rule $\supset\mathsf{E}$ to typecheck $\lambda x\!:\!A \supset B.\, \lambda y\!:\!A.\, x\ y$:

$$\dfrac{\dfrac{\dfrac{\overline{x : A \supset B} \quad \overline{y : A}}{x\ y : B}\ \supset\mathsf{E}}{\lambda y\!:\!A.\, x\ y : A \supset B}\ \supset\mathsf{I}}{\lambda x\!:\!A \supset B.\, \lambda y\!:\!A.\, x\ y : (A \supset B) \supset (A \supset B)}\ \supset\mathsf{I}$$

Disjunction

As proof terms corresponding to the rule $\vee\mathsf{I_L}$ and $\vee\mathsf{I_R}$, we use *injection terms* $\mathsf{inl}_A\ M$ and $\mathsf{inr}_A\ M$; inl and inr stand for 'injection left' and 'injection right,' respectively:

$$\dfrac{A\ \mathit{true}}{A \vee B\ \mathit{true}}\ \vee\mathsf{I_L} \quad \Longleftrightarrow \quad \dfrac{M : A}{\mathsf{inl}_B\ M : A \vee B}\ \vee\mathsf{I_L} \qquad \dfrac{B\ \mathit{true}}{A \vee B\ \mathit{true}}\ \vee\mathsf{I_R} \quad \Longleftrightarrow \quad \dfrac{M : B}{\mathsf{inr}_A\ M : A \vee B}\ \vee\mathsf{I_R}$$

We annotate an injection term $\mathsf{inl}_A\ M$ or $\mathsf{inr}_A\ M$ with a type A so that whenever M typechecks, the whole injection term also typechecks with a unique type.

For the elimination rule $\vee\mathsf{E}$, we use a *case term* $\mathsf{case}\ M\ \mathsf{of}\ \mathsf{inl}\ x.\, N \mid \mathsf{inr}\ y.\, N'$; as with the rule $\supset\mathsf{I}$, we represent hypotheses $\overline{A\ \mathit{true}}^{\,x}$ and $\overline{B\ \mathit{true}}^{\,y}$ in the premise as variables x and y:

$$\dfrac{A \vee B\ \mathit{true} \qquad \begin{array}{c}\overline{A\ \mathit{true}}^{\,x}\\ \vdots \\ C\ \mathit{true}\end{array} \qquad \begin{array}{c}\overline{B\ \mathit{true}}^{\,y}\\ \vdots \\ C\ \mathit{true}\end{array}}{C\ \mathit{true}}\ \vee\mathsf{E}^{x,y} \quad \Longleftrightarrow \quad \dfrac{M : A \vee B \qquad \begin{array}{c}\overline{x : A}\\ \vdots \\ N : C\end{array} \qquad \begin{array}{c}\overline{y : B}\\ \vdots \\ N' : C\end{array}}{\mathsf{case}\ M\ \mathsf{of}\ \mathsf{inl}\ x.\, N \mid \mathsf{inr}\ y.\, N' : C}\ \vee\mathsf{E}$$

Variables x and y are bound in the case term $\mathsf{case}\ M\ \mathsf{of}\ \mathsf{inl}\ x.\, N \mid \mathsf{inr}\ y.\, N'$, and remain valid only within N and N', respectively. As an example, here is a proof term of type $(A \vee B) \supset (B \vee A)$:

$$\dfrac{\dfrac{\overline{x : A \vee B} \qquad \dfrac{\overline{y : A}}{\mathsf{inr}_B\ y : B \vee A}\ \vee\mathsf{I_R} \qquad \dfrac{\overline{z : B}}{\mathsf{inl}_A\ z : B \vee A}\ \vee\mathsf{I_L}}{\mathsf{case}\ x\ \mathsf{of}\ \mathsf{inl}\ y.\, \mathsf{inr}_B\ y \mid \mathsf{inr}\ z.\, \mathsf{inl}_A\ z : B \vee A}\ \vee\mathsf{E}}{\lambda x\!:\!A \vee B.\, \mathsf{case}\ x\ \mathsf{of}\ \mathsf{inl}\ y.\, \mathsf{inr}_B\ y \mid \mathsf{inr}\ z.\, \mathsf{inl}_A\ z : (A \vee B) \supset (B \vee A)}\ \supset\mathsf{I}$$

Truth and falsehood

We use a *unit term* $()$ as a proof term for $\top\ \mathit{true}$:

$$\dfrac{}{\top\ \mathit{true}}\ \top\mathsf{I} \quad \Longleftrightarrow \quad \dfrac{}{() : \top}\ \top\mathsf{I}$$

Just like there is no logical content in $\top\ \mathit{true}$, a unit term carries no useful information. As truth $\top$ has no elimination rule, there is no more rule for $()$.

Since falsehood $\perp$ has no introduction rule, there is no proof term for type $\perp$. For the elimination rule $\perp\mathsf{E}$, we use an *abort term* $\mathsf{abort}_C\ M$:

$$\dfrac{\perp\ \mathit{true}}{C\ \mathit{true}}\ \perp\mathsf{E} \quad \Longleftrightarrow \quad \dfrac{M : \perp}{\mathsf{abort}_C\ M : C}\ \perp\mathsf{E}$$

We annotate an abort term with a type C so that an unambiguous type can be assigned when M has type $\perp$.

3.2 Type system

Since hypothetical judgments are just a syntactic tool for displaying hypothetical proofs, it is straightforward to extend the translation in Section 3.1 to hypothetical judgments. We continue to use the same

$$\frac{M : A \quad N : B}{(M, N) : A \wedge B}\ \wedge\mathsf{I} \qquad \frac{M : A \wedge B}{\mathsf{fst}\ M : A}\ \wedge\mathsf{E_L} \qquad \frac{M : A \wedge B}{\mathsf{snd}\ M : B}\ \wedge\mathsf{E_R} \qquad \frac{\begin{array}{c}\overline{x : A}\\ \vdots \\ M : B\end{array}}{\lambda x{:}A.\, M : A \supset B}\ \supset\mathsf{I} \qquad \frac{M : A \supset B \quad N : A}{M\ N : B}\ \supset\mathsf{E}$$

$$\frac{M : A}{\mathsf{inl}_B\ M : A \vee B}\ \vee\mathsf{I_L} \qquad \frac{M : B}{\mathsf{inr}_A\ M : A \vee B}\ \vee\mathsf{I_R} \qquad \frac{M : A \vee B \quad \begin{array}{c}\overline{x : A}\\ \vdots \\ N : C\end{array} \quad \begin{array}{c}\overline{y : B}\\ \vdots \\ N' : C\end{array}}{\mathsf{case}\ M\ \mathsf{of}\ \mathsf{inl}\ x.\, N \mid \mathsf{inr}\ y.\, N' : C}\ \vee\mathsf{E}$$

$$\frac{}{() : \top}\ \top\mathsf{I} \qquad \frac{M : \bot}{\mathsf{abort}_C\ M : C}\ \bot\mathsf{E}$$

Figure 3.1: Typing rules for proof terms in propositional logic

$$\frac{x : A \in \Gamma}{\Gamma \vdash x : A}\ \mathsf{Hyp} \qquad \frac{\Gamma, x : A \vdash M : B}{\Gamma \vdash \lambda x{:}A.\, M : A \supset B}\ \supset\mathsf{I} \qquad \frac{\Gamma \vdash M : A \supset B \quad \Gamma \vdash N : A}{\Gamma \vdash M\ N : B}\ \supset\mathsf{E}$$

$$\frac{\Gamma \vdash M : A \quad \Gamma \vdash N : B}{\Gamma \vdash (M, N) : A \wedge B}\ \wedge\mathsf{I} \qquad \frac{\Gamma \vdash M : A \wedge B}{\Gamma \vdash \mathsf{fst}\ M : A}\ \wedge\mathsf{E_L} \qquad \frac{\Gamma \vdash M : A \wedge B}{\Gamma \vdash \mathsf{snd}\ M : B}\ \wedge\mathsf{E_R}$$

$$\frac{\Gamma \vdash M : A}{\Gamma \vdash \mathsf{inl}_B\ M : A \vee B}\ \vee\mathsf{I_L} \qquad \frac{\Gamma \vdash M : B}{\Gamma \vdash \mathsf{inr}_A\ M : A \vee B}\ \vee\mathsf{I_R} \qquad \frac{\Gamma \vdash M : A \vee B \quad \Gamma, x : A \vdash N : C \quad \Gamma, y : B \vdash N' : C}{\Gamma \vdash \mathsf{case}\ M\ \mathsf{of}\ \mathsf{inl}\ x.\, N \mid \mathsf{inr}\ y.\, N' : C}\ \vee\mathsf{E}$$

$$\frac{}{\Gamma \vdash () : \top}\ \top\mathsf{I} \qquad \frac{\Gamma \vdash M : \bot}{\Gamma \vdash \mathsf{abort}_C\ M : C}\ \bot\mathsf{E}$$

Figure 3.2: Typing rules using hypothetical judgments

set of proof terms to represent hypothetical proofs, but use a new typing judgment with an entailment relation $\vdash$:

$$A_1\ \mathit{true}, \cdots, A_n\ \mathit{true} \overset{\mathcal{D}}{\vdash} C\ \mathit{true} \qquad \Longleftrightarrow \qquad x_1 : A_1, \cdots, x_n : A_n \vdash M : C$$

The new typing judgment chooses a fresh variable x_i to represent each hypothesis $\overline{A_i\ \mathit{true}}$. Note that the new typing judgment itself is an example of a hypothetical judgment such that antecedents are (typing) judgments of the form $x_i : A_i$ and the succedent is a (typing) judgment of the form $M : C$. For the sake of simplicity, we maintain the invariant that all variables in antecedents are distinct.

Figure 3.2 shows a system of typing rules, or a *type system*, based on the new typing judgment. An antecedent $x : A$ is called a *type binding* because it binds variable x to type A. Γ denotes a collection of type bindings, and is called a *typing context*. We assume that the exchange rule is built into the typing judgment (*i.e.*, we may reorder type bindings as we like). The other two structural properties are stated as follows:

Proposition 3.1 (Structural properties).

(Weakening) *If* $\Gamma \vdash M : C$*, then* $\Gamma, x : A \vdash M : C$.

(Contraction) *If* $\Gamma, x : A, x : A \vdash M : C$*, then* $\Gamma, x : A \vdash M : C$.

Proof. By induction on the structure of the proof of $\Gamma \vdash M : C$ and $\Gamma, x : A, x : A \vdash M : C$. $\square$

Alternatively the proof of Proposition 3.1 may proceed by induction on the structure of proof term M. This is because the type system in Figure 3.2 is *syntax-directed*: the *syntactic* form of proof term M decides, or *directs*, a unique typing rule necessary for deducing a typing judgment $\Gamma \vdash M : C$. Hence, for example, if M is a λ-abstraction $\lambda y{:}C_1.\, M'$, then $\Gamma \vdash M : C$ is provable only by applying the rule $\supset$I, from which we conclude $\Gamma, y : C_1 \vdash M' : C_2$ (the premise of the rule $\supset$I) and $C = C_1 \supset C_2$. As an illustration, we give a proof of the weakening property for the case $M = \lambda y{:}C_1.\, M'$:

Case $M = \lambda y\!:\!C_1.\, M'$

$\Gamma, y : C_1 \vdash M' : C_2$ and $C = C_1 \supset C_2$ — by the rule $\supset$I with $\Gamma \vdash M : C$

$\Gamma, x : A, y : C_1 \vdash M' : C_2$ — by induction hypothesis on M'

$\Gamma, x : A \vdash \lambda y\!:\!C_1.\, M' : C_1 \supset C_2$ — by the rule $\supset$I

$\Gamma, x : A \vdash M : C$ — from $M = \lambda y\!:\!C_1.\, M'$ and $C = C_1 \supset C_2$

In essence, the entire proof of a typing judgment $\Gamma \vdash M : C$ can be reconstructed by analyzing proof term M, which implies that analyzing the structure of the proof of $\Gamma \vdash M : C$ is equivalent to analyzing the structure of proof term M.

As a special case of a hypothetical judgment, the typing judgment in Figure 3.2 satisfies the two general properties of hypothetical judgments: reflexivity and substitution principle. Reflexivity follows from the rule Hyp. For the substitution principle, we need an operation on proof terms that corresponds to $[\mathcal{D}/A\ \mathit{true}]\mathcal{E}$, *i.e.*, a substitution of a proof $\mathcal{D}$ for a hypothesis $\overline{A\ \mathit{true}}$ in a hypothetical proof $\mathcal{E}$. Suppose that proof terms M and N represent proofs $\mathcal{D}$ and $\mathcal{E}$, respectively, and that we use a variable x to represent hypothesis $\overline{A\ \mathit{true}}$. Then $[\mathcal{D}/A\ \mathit{true}]\mathcal{E}$ is literally translated to $[M/x]N$, which is our notation for substituting M for x in N. We define $[M/x]N$ inductively on the structure of N, where we assume $x \neq y$ and $x \neq z$:

$$
\begin{array}{rcl}
[M/x]x & = & M \\
[M/x]y & = & y \\
[M/x]\lambda x\!:\!A.\, N & = & \lambda x\!:\!A.\, N \\
[M/x]\lambda y\!:\!A.\, N & = & \lambda y\!:\!A.\, [M/x]N \\
[M/x](N_1\ N_2) & = & ([M/x]N_1)\ ([M/x]N_2) \\
[M/x](N_1, N_2) & = & ([M/x]N_1, [M/x]N_2) \\
[M/x]\mathsf{fst}\ N & = & \mathsf{fst}\ [M/x]N \\
[M/x]\mathsf{snd}\ N & = & \mathsf{snd}\ [M/x]N \\
[M/x]\mathsf{inl}_B\ N & = & \mathsf{inl}_B\ [M/x]N \\
[M/x]\mathsf{inr}_A\ N & = & \mathsf{inr}_A\ [M/x]N \\
[M/x]\mathsf{case}\ N\ \mathsf{of}\ \mathsf{inl}\ x.\, N_1 \mid \mathsf{inr}\ x.\, N_2 & = & \mathsf{case}\ [M/x]N\ \mathsf{of}\ \mathsf{inl}\ x.\, N_1 \mid \mathsf{inr}\ x.\, N_2 \\
[M/x]\mathsf{case}\ N\ \mathsf{of}\ \mathsf{inl}\ y.\, N_1 \mid \mathsf{inr}\ x.\, N_2 & = & \mathsf{case}\ [M/x]N\ \mathsf{of}\ \mathsf{inl}\ y.\, [M/x]N_1 \mid \mathsf{inr}\ x.\, N_2 \\
[M/x]\mathsf{case}\ N\ \mathsf{of}\ \mathsf{inl}\ y.\, N_1 \mid \mathsf{inr}\ z.\, N_2 & = & \mathsf{case}\ [M/x]N\ \mathsf{of}\ \mathsf{inl}\ y.\, [M/x]N_1 \mid \mathsf{inr}\ z.\, [M/x]N_2 \\
[M/x]() & = & () \\
[M/x]\mathsf{abort}_C\ N & = & \mathsf{abort}_C\ [M/x]N
\end{array}
$$

In the case of $[M/x]\lambda y\!:\!A.\, N$, we assume that y is not a *free variable* of M, where a free variable of M is a variable that is not bound in λ-abstractions or case terms within M. If y happens to be a free variable of M, we say that a *variable capture* occurs: y, a free variable before the substitution, turns into a bound variable after the substitution. For example, $\lambda y\!:\!A.\, (x, y)$ and $\lambda z\!:\!A.\, (x, z)$ represent the same proof, so $[y/x]\lambda y\!:\!A.\, (x, y)$ must be equivalent to $[y/x]\lambda z\!:\!A.\, (x, z) = \lambda z\!:\!A.\, (y, z)$, which still recognizes y as a free variable. A variable capture, however, occurs in $[y/x]\lambda y\!:\!A.\, (x, y)$ to yield $\lambda y\!:\!A.\, (y, y)$, in which y is used only as a bound variable. Thus, if a variable capture occurs in $[M/x]\lambda y\!:\!A.\, N$, we need to rename y to a different variable. A similar restriction applies to substitutions into case terms.

Theorem 3.2 (Substitution). *If $\Gamma \vdash M : A$ and $\Gamma, x : A \vdash N : C$, then $\Gamma \vdash [M/x]N : C$.*

Proof. By induction on the structure of the proof of $\Gamma, x : A \vdash N : C$. We may also use induction on the structure of proof term N.

We consider three cases Hyp, $\supset$I, and $\supset$E. In the case $\supset$I, we rename y as necessary so as to avoid variable captures.

Case $\dfrac{y : C \in \Gamma}{\Gamma, x : A \vdash y : C}\ \mathsf{Hyp}$ where $N = y$

$\Gamma \vdash y : C$ — by the rule Hyp with $y : C \in \Gamma$

$\Gamma \vdash [M/x]y : C$ — from $[M/x]y = y$

Case $\dfrac{}{\Gamma, x : A \vdash x : C}\ \mathsf{Hyp}$ where $N = x$ and $A = C$

$\Gamma \vdash M : C$ — from the assumption $\Gamma \vdash M : A$

$\Gamma \vdash [M/x]x : C$ — from $[M/x]x = M$

Case $\dfrac{\Gamma, x : A, y : C_1 \vdash N' : C_2}{\Gamma, x : A \vdash \lambda y{:}C_1.\, N' : C_1 \supset C_2} \supset\mathsf{I}$ where $N = \lambda y{:}C_1.\, N'$ and $C = C_1 \supset C_2$

$\Gamma, y : C_1 \vdash M : A$ — by weakening $\Gamma \vdash M : A$

$\Gamma, y : C_1 \vdash [M/x]N' : C_2$ — by IH on $\Gamma, x : A, y : C_1 \vdash N' : C_2$ with $\Gamma, y : C_1 \vdash M : A$

$\Gamma \vdash \lambda y{:}C_1.\, [M/x]N' : C_1 \supset C_2$ — by the rule $\supset\mathsf{I}$

$\Gamma \vdash [M/x]\lambda y{:}C_1.\, N' : C_1 \supset C_2$ — from $\lambda y{:}C_1.\, [M/x]N' = [M/x]\lambda y{:}C_1.\, N'$

Case $\dfrac{\Gamma, x : A \vdash N_1 : C' \supset C \quad \Gamma, x : A \vdash N_2 : C'}{\Gamma, x : A \vdash N_1\ N_2 : C} \supset\mathsf{E}$ where $N = N_1\ N_2$

$\Gamma \vdash [M/x]N_1 : C' \supset C$ — by IH on $\Gamma, x : A \vdash N_1 : C' \supset C$ with $\Gamma \vdash M : A$

$\Gamma \vdash [M/x]N_2 : C'$ — by IH on $\Gamma, x : A \vdash N_2 : C'$ with $\Gamma \vdash M : A$

$\Gamma \vdash [M/x]N_1\ [M/x]N_2 : C$ — by the rule $\supset\mathsf{E}$

$\Gamma \vdash [M/x](N_1\ N_2) : C$ — from $[M/x]N_1\ [M/x]N_2 = [M/x](N_1\ N_2)$

$\square$

3.3 β-reductions and η-expansions

We have seen in Section 2.5 that a local reduction removes a detour in a proof of $A\ \mathit{true}$ to yield a reduced proof of the same judgment. Since a proof of $A\ \mathit{true}$ can be represented as a proof term of type A under the Curry-Howard isomorphism, a local reduction is translated to a reduction of a proof term to another proof term of the same type. We refer to such a reduction of a proof term as a *β-reduction*; we write $M \Longrightarrow_\beta N$ for a β-reduction of M to N.

It is easy to derive β-reductions of proof terms from local reductions of proofs. For example, we obtain a β-reduction of $\mathsf{fst}\ (M, N)$ to M as follows:

$$\dfrac{\dfrac{M : A \quad N : B}{(M, N) : A \wedge B} \wedge\mathsf{I}}{\mathsf{fst}\ (M, N) : A} \wedge\mathsf{E_L} \qquad \Longrightarrow_\beta \qquad M : A$$

The following diagram explains how to obtain a β-reduction from a local reduction removing a detour in which the rule $\supset\mathsf{I}$ is immediately followed by the rule $\supset\mathsf{E}$:

$$\dfrac{\dfrac{\begin{array}{c}\overline{x : A}\\ \vdots \\ M : B\end{array}}{\lambda x{:}A.\, M : A \rightarrow B} \supset\mathsf{I} \quad N : A}{(\lambda x{:}A.\, M)\ N : B} \supset\mathsf{E} \qquad \Longrightarrow_\beta \qquad \begin{array}{c}[N/x]x : A\\ \vdots \\ [N/x]M : B\end{array}$$

The same β-reduction from a proof using hypothetical judgments is obtained as follows:

$$\begin{array}{ccc}
\dfrac{\dfrac{\begin{array}{c}\mathcal{D}\\ \Gamma, A\ \mathit{true} \vdash B\ \mathit{true}\end{array}}{\Gamma \vdash A \supset B\ \mathit{true}} \supset\mathsf{I} \quad \begin{array}{c}\mathcal{E}\\ \Gamma \vdash A\ \mathit{true}\end{array}}{\Gamma \vdash B\ \mathit{true}} \supset\mathsf{E} & \Longrightarrow_R & \begin{array}{c}[\mathcal{E}/A\ \mathit{true}]\mathcal{D}\\ \Gamma \vdash B\ \mathit{true}\end{array} \\
\Updownarrow & & \Updownarrow \\
\dfrac{\dfrac{\Gamma, x : A \vdash M : B}{\Gamma \vdash \lambda x{:}A.\, M : A \supset B} \supset\mathsf{I} \quad \Gamma \vdash N : A}{\Gamma \vdash (\lambda x{:}A.\, M)\ N : B} \supset\mathsf{E} & \Longrightarrow_\beta & \Gamma \vdash [N/x]M : B
\end{array}$$

In this way, we obtain the following β-reductions for proof terms:

$$\begin{array}{rcl}
(\lambda x{:}A.\, M)\ N & \Longrightarrow_\beta & [N/x]M \\
\mathsf{fst}\ (M, N) & \Longrightarrow_\beta & M \\
\mathsf{snd}\ (M, N) & \Longrightarrow_\beta & N \\
\mathsf{case}\ \mathsf{inl}_B\ M\ \mathsf{of}\ \mathsf{inl}\ x.\, N \mid \mathsf{inr}\ y.\, N' & \Longrightarrow_\beta & [M/x]N \\
\mathsf{case}\ \mathsf{inr}_A\ M\ \mathsf{of}\ \mathsf{inl}\ x.\, N \mid \mathsf{inr}\ y.\, N' & \Longrightarrow_\beta & [M/y]N'
\end{array}$$

Exercise 3.3. Verify that other β-reductions are obtained from their corresponding local reductions.

A β-reduction preserves the type of the proof term being reduced. This is called the *subject reduction* property because a typing judgment $M : C$ may be regarded as a sentence whose subject is M and whose predicate is C. The proof exploits the syntax-directedness of the type system: for any proof term M, there is a unique typing rule R for deducing $\Gamma \vdash M : C$; hence a proof of $\Gamma \vdash M : C$ by the rule R implies that the premise of the rule R holds as well.

Theorem 3.4 (Subject reduction). *If $\Gamma \vdash M : C$ and $M \Longrightarrow_\beta M'$, then $\Gamma \vdash M' : C$.*

Proof. By case analysis of $M \Longrightarrow_\beta M'$. We show three representative cases; the remaining two cases are similar. The proof reuses metavariable M.

Case $(\lambda x{:}A.\, M)\ N \Longrightarrow_\beta [N/x]M$

$\Gamma \vdash (\lambda x{:}A.\, M)\ N : C$	assumption
$\Gamma \vdash \lambda x{:}A.\, M : A' \supset C$ and $\Gamma \vdash N : A'$	by the rule $\supset\mathsf{E}$
$\Gamma, x : A \vdash M : C$ and $A = A'$	by the rule $\supset\mathsf{I}$ with $\Gamma \vdash \lambda x{:}A.\, M : A' \supset C$
$\Gamma \vdash [N/x]M : C$	by Theorem 3.2 with $\Gamma, x : A \vdash M : C$ and $\Gamma \vdash N : A'$ and $A = A'$

Case $\mathsf{fst}\ (M, N) \Longrightarrow_\beta M$

$\Gamma \vdash \mathsf{fst}\ (M, N) : C$	assumption
$\Gamma \vdash (M, N) : C \wedge A$	by the rule $\wedge\mathsf{E_L}$
$\Gamma \vdash M : C$	by the rule $\wedge\mathsf{I}$

Case $\mathsf{case}\ \mathsf{inl}_B\ M\ \mathsf{of}\ \mathsf{inl}\ x.\, N \mid \mathsf{inr}\ y.\, N' \Longrightarrow_\beta [M/x]N$

$\Gamma \vdash \mathsf{case}\ \mathsf{inl}_B\ M\ \mathsf{of}\ \mathsf{inl}\ x.\, N \mid \mathsf{inr}\ y.\, N' : C$	assumption
$\Gamma \vdash \mathsf{inl}_B\ M : A' \vee B'$ and $\Gamma, x : A' \vdash N : C$ and $\Gamma, y : B' \vdash N' : C$	by the rule $\vee\mathsf{E}$
$\Gamma \vdash M : A'$ and $B = B'$	by the rule $\vee\mathsf{I_L}$ with $\Gamma \vdash \mathsf{inl}_B\ M : A' \vee B'$
$\Gamma \vdash [M/x]N : C$	by Theorem 3.2 with $\Gamma, x : A' \vdash N : C$ and $\Gamma \vdash M : A'$

□

The β-reduction relation $\Longrightarrow_\beta$ can be generalized to a *structural congruence relation* $\Longrightarrow_{\beta+}$ such that $M \Longrightarrow_{\beta+} M'$ holds if a β-reduction is applied to a subterm of M to yield M'. (Such a subterm is commonly called a *redex*, or a *red*ucible *ex*pression.) For example, $((\lambda x{:}A.\, M)\ N, N') \Longrightarrow_{\beta+} ([N/x]M, N')$ holds because a subterm $(\lambda x{:}A.\, M)\ N$ reduces to $[N/x]M$ by a β-reduction.

Figure 3.3 shows the rules for the structural congruence relation $\Longrightarrow_{\beta+}$. Note that there *is* a rule for reducing an abort term $\mathsf{abort}_C\ M$. Since a proof term M may contain multiple subterms to which β-reductions are applicable, the relation $\Longrightarrow_{\beta+}$ is non-deterministic: given a proof term M, it does not always determine a unique proof term M' such that $M \Longrightarrow_{\beta+} M'$. Theorem 3.4 now extends to the *subterm subject reduction* property:

Theorem 3.5 (Subterm subject reduction). *If $\Gamma \vdash M : C$ and $M \Longrightarrow_{\beta+} M'$, then $\Gamma \vdash M' : C$.*

Proof. By induction on the structure of the proof of $M \Longrightarrow_{\beta+} M'$. The proof uses Theorem 3.4. □

Like local reductions, local expansions are translated to expansions of proof terms under the Curry-Howard isomorphism. We refer to such expansions of proof terms as η-expansions; we write $M \Longrightarrow_\eta N$ for an η-expansion of M to N.

$$
\begin{array}{rcll}
M : A \supset B & \Longrightarrow_\eta & \lambda x{:}A.\, M\ x & (x \textit{ is not free in } M) \\
M : A \wedge B & \Longrightarrow_\eta & (\mathsf{fst}\ M, \mathsf{snd}\ M) & \\
M : A \vee B & \Longrightarrow_\eta & \mathsf{case}\ M\ \mathsf{of}\ \mathsf{inl}\ x.\, \mathsf{inl}_B\ x \mid \mathsf{inr}\ y.\, \mathsf{inr}_A\ y & \\
M : \top & \Longrightarrow_\eta & () & \\
M : \bot & \Longrightarrow_\eta & \mathsf{abort}_\bot\ M &
\end{array}
$$

$$\frac{M \Longrightarrow_\beta M'}{M \Longrightarrow_{\beta+} M'} \quad \frac{M \Longrightarrow_{\beta+} M'}{\lambda x\!:\!A.\,M \Longrightarrow_{\beta+} \lambda x\!:\!A.\,M'} \quad \frac{M \Longrightarrow_{\beta+} M'}{M\ N \Longrightarrow_{\beta+} M'\ N} \quad \frac{N \Longrightarrow_{\beta+} N'}{M\ N \Longrightarrow_{\beta+} M\ N'}$$

$$\frac{M \Longrightarrow_{\beta+} M'}{(M,N) \Longrightarrow_{\beta+} (M',N)} \quad \frac{N \Longrightarrow_{\beta+} N'}{(M,N) \Longrightarrow_{\beta+} (M,N')} \quad \frac{M \Longrightarrow_{\beta+} M'}{\mathsf{fst}\ M \Longrightarrow_{\beta+} \mathsf{fst}\ M'} \quad \frac{M \Longrightarrow_{\beta+} M'}{\mathsf{snd}\ M \Longrightarrow_{\beta+} \mathsf{snd}\ M'}$$

$$\frac{M \Longrightarrow_{\beta+} M'}{\mathsf{inl}_B\ M \Longrightarrow_{\beta+} \mathsf{inl}_B\ M'} \quad \frac{M \Longrightarrow_{\beta+} M'}{\mathsf{inr}_A\ M \Longrightarrow_{\beta+} \mathsf{inr}_A\ M'}$$

$$\frac{M \Longrightarrow_{\beta+} M'}{\mathsf{case}\ M\ \mathsf{of\ inl}\ x.\,N \mid \mathsf{inr}\ y.\,N' \Longrightarrow_{\beta+} \mathsf{case}\ M'\ \mathsf{of\ inl}\ x.\,N \mid \mathsf{inr}\ y.\,N'}$$

$$\frac{N \Longrightarrow_{\beta+} N''}{\mathsf{case}\ M\ \mathsf{of\ inl}\ x.\,N \mid \mathsf{inr}\ y.\,N' \Longrightarrow_{\beta+} \mathsf{case}\ M\ \mathsf{of\ inl}\ x.\,N'' \mid \mathsf{inr}\ y.\,N'}$$

$$\frac{N' \Longrightarrow_{\beta+} N''}{\mathsf{case}\ M\ \mathsf{of\ inl}\ x.\,N \mid \mathsf{inr}\ y.\,N' \Longrightarrow_{\beta+} \mathsf{case}\ M\ \mathsf{of\ inl}\ x.\,N \mid \mathsf{inr}\ y.\,N''}$$

$$\frac{M \Longrightarrow_{\beta+} M'}{\mathsf{abort}_C\ M \Longrightarrow_{\beta+} \mathsf{abort}_C\ M'}$$

Figure 3.3: Rules for the structural congruence relation $\Longrightarrow_{\beta+}$

For example, the η-expansions for $\wedge$ and $\supset$ are obtained as follows:

$$M : A \wedge B \qquad \Longrightarrow_\eta \qquad \cfrac{\cfrac{M : A \wedge B}{\mathsf{fst}\ M : A}\,\wedge\mathsf{E_L} \quad \cfrac{M : A \wedge B}{\mathsf{snd}\ M : B}\,\wedge\mathsf{E_R}}{(\mathsf{fst}\ M, \mathsf{snd}\ M) : A \wedge B}\,\wedge\mathsf{I}$$

$$M : A \supset B \qquad \Longrightarrow_\eta \qquad \cfrac{\cfrac{M : A \supset B \quad \overline{x : A}}{M\ x : B}\,\supset\mathsf{E}}{\lambda x\!:\!A.\,M\ x : A \supset B}\,\supset\mathsf{I}$$

Exercise 3.6. Verify that other η-expansions are obtained from their corresponding local expansions.

Like a β-reduction, an η-expansion preserves the type of the proof term being expanded. We can also derive another structural congruence relation $\Longrightarrow_{\eta+}$ from the η-expansion relation $\Longrightarrow_\eta$ similarly to $\Longrightarrow_{\beta+}$.

Theorem 3.7. *If* $\Gamma \vdash M : C$ *and* $M \Longrightarrow_\eta M'$*, then* $\Gamma \vdash M' : C$.

Theorem 3.8. *If* $\Gamma \vdash M : C$ *and* $M \Longrightarrow_{\eta+} M'$*, then* $\Gamma \vdash M' : C$.

3.4 Proof terms in normal form

Since it is a special case of a proof of A *true*, a proof of a neutral judgment $A\!\downarrow$ or a normal judgment $A\!\uparrow$ can be represented as a proof term of a special form under the Curry-Howard isomorphism. We use an *elim(ination) term* E to represent a proof of $A\!\downarrow$ and an *intro(duction) term* I, or a proof term in normal form, to represent a proof of $A\!\uparrow$.

$$\begin{array}{c}\mathcal{D}\\ A\!\downarrow\end{array} \iff E : A \qquad\qquad \begin{array}{c}\mathcal{E}\\ A\!\uparrow\end{array} \iff I : A$$

Then the inference rules for neutral and normal judgments (in Figure 2.3 or Figure 2.4) are translated to the following definition of elim terms and intro terms:

$$\begin{array}{lccl}\text{elim term} & E & ::= & x \mid E\ I \mid \mathsf{fst}\ E \mid \mathsf{snd}\ E \\ \text{intro term} & I & ::= & E \mid \lambda x\!:\!A.\,I \mid (I, I) \mid \mathsf{inl}_A\ I \mid \mathsf{inr}_A\ I \mid \mathsf{case}\ E\ \mathsf{of\ inl}\ x.\,I \mid \mathsf{inr}\ x.\,I \mid () \mid \mathsf{abort}_A\ E\end{array}$$

For example, the rule $\mathsf{Hyp}_\downarrow$ specifies that variables be elim terms:

$$\frac{}{\Gamma_\downarrow, A{\downarrow} \vdash A{\downarrow}}\ \mathsf{Hyp}_\downarrow \quad\iff\quad \frac{}{\Gamma, x : A \vdash x : A}\ \mathsf{Hyp}$$

The rule $\supset\!\mathsf{I}_\uparrow$ explains why $\lambda x\!:\!A.\, I$ is defined as an intro term; similarly the rule $\supset\!\mathsf{E}_\downarrow$ explains why $E\ I$ is defined as an elim term:

$$\frac{\Gamma_\downarrow, A{\downarrow} \vdash B{\uparrow}}{\Gamma_\downarrow \vdash A \supset B{\uparrow}}\ \supset\!\mathsf{I}_\uparrow \quad\iff\quad \frac{\Gamma, x : A \vdash I : B}{\Gamma \vdash \lambda x\!:\!A.\, I : A \supset B}\ \supset\!\mathsf{I}$$

$$\frac{\Gamma_\downarrow \vdash A \supset B{\downarrow} \quad \Gamma_\downarrow \vdash A{\uparrow}}{\Gamma_\downarrow \vdash B{\downarrow}}\ \supset\!\mathsf{E}_\downarrow \quad\iff\quad \frac{\Gamma \vdash E : A \supset B \quad \Gamma \vdash I : A}{\Gamma \vdash E\ I : B}\ \supset\!\mathsf{E}$$

Note also that the inclusion of elim terms as intro terms, not the other way around, is based on the rule ${\downarrow\uparrow}$.

Exercise 3.9. Verify that the definition of elim terms and intro terms conforms to the rules in Figure 2.3 or Figure 2.4.

With the definition of intro and elim terms, we can rewrite Theorem 2.13 as the following normalization theorem for proof terms. For the moment, we do not consider proof terms for disjunction $\vee$ and falsehood $\perp$. We write $\Longrightarrow^*_{\beta+}$ for the reflexive and transitive closure of $\Longrightarrow_{\beta+}$.

Theorem 3.10 (Normalization).
For every proof term M such that $\Gamma \vdash M : A$, there exists an intro term I such that $M \Longrightarrow^*_{\beta+} I$.

Here is an example of a sequence of reductions from an ordinary proof term to an intro term, or simply a *normalization sequence*; the subterm being reduced at each step is underlined:

$$(\lambda x\!:\!A.\, \mathsf{fst}\ (x,z))\ \underline{\mathsf{fst}\ (y,z)} \Longrightarrow_{\beta+} \underline{(\lambda x\!:\!A.\, \mathsf{fst}\ (x,z))\ y} \Longrightarrow_{\beta+} \underline{\mathsf{fst}\ (y,z)} \Longrightarrow_{\beta+} y$$

There are five alternative normalization sequences:

$$\begin{array}{l}
(\lambda x\!:\!A.\, \underline{\mathsf{fst}\ (x,z)})\ \mathsf{fst}\ (y,z) \Longrightarrow_{\beta+} \underline{(\lambda x\!:\!A.\, x)\ \mathsf{fst}\ (y,z)} \Longrightarrow_{\beta+} \underline{\mathsf{fst}\ (y,z)} \Longrightarrow_{\beta+} y \\
(\lambda x\!:\!A.\, \underline{\mathsf{fst}\ (x,z)})\ \mathsf{fst}\ (y,z) \Longrightarrow_{\beta+} (\lambda x\!:\!A.\, x)\ \underline{\mathsf{fst}\ (y,z)} \Longrightarrow_{\beta+} \underline{(\lambda x\!:\!A.\, x)\ y} \Longrightarrow_{\beta+} y \\
(\lambda x\!:\!A.\, \mathsf{fst}\ (x,z))\ \underline{\mathsf{fst}\ (y,z)} \Longrightarrow_{\beta+} (\lambda x\!:\!A.\, \underline{\mathsf{fst}\ (x,z)})\ y \Longrightarrow_{\beta+} \underline{(\lambda x\!:\!A.\, x)\ y} \Longrightarrow_{\beta+} y \\
\underline{(\lambda x\!:\!A.\, \mathsf{fst}\ (x,z))\ \mathsf{fst}\ (y,z)} \Longrightarrow_{\beta+} \underline{\mathsf{fst}\ (\mathsf{fst}\ (y,z), z)} \Longrightarrow_{\beta+} \underline{\mathsf{fst}\ (y,z)} \Longrightarrow_{\beta+} y \\
\underline{(\lambda x\!:\!A.\, \mathsf{fst}\ (x,z))\ \mathsf{fst}\ (y,z)} \Longrightarrow_{\beta+} \mathsf{fst}\ (\underline{\mathsf{fst}\ (y,z)}, z) \Longrightarrow_{\beta+} \underline{\mathsf{fst}\ (y,z)} \Longrightarrow_{\beta+} y
\end{array}$$

Two other important properties of proof terms are *strong normalization* and *confluence*. Combined together, these two properties show that *every* normalization sequence produces a *unique* intro term.

Theorem 3.11 (Strong normalization, or termination).
For any proof term M such that $\Gamma \vdash M : A$, there is no infinite normalization sequence $M \Longrightarrow_{\beta+} M_1 \Longrightarrow_{\beta+} \cdots$.

Theorem 3.12 (Confluence, or Church-Rosser property).
Suppose $\Gamma \vdash M : A$. If $M \Longrightarrow^*_{\beta+} N_1$ and $M \Longrightarrow^*_{\beta+} N_2$, then there exists a proof term N such that $N_1 \Longrightarrow^*_{\beta+} N$ and $N_2 \Longrightarrow^*_{\beta+} N$.

In order for the normalization theorem to hold in the presence of proof terms for $\vee$ and $\perp$, the definition of $\Longrightarrow_{\beta+}$ needs to be extended by incorporating commuting conversions for proof terms. A commuting conversion is allowed when a case term $\mathsf{case}\ M\ \mathsf{of\ inl}\ x.\, N \mid \mathsf{inr}\ y.\, N'$ appears in a position *where an elim term is expected.* To simplify the definition of a commuting conversion, we use a *commuting conversion context* κ, which is a proof term with a hole $\Box$ in a position where an elim term is expected. We write $\kappa[\![M]\!]$ for a proof term obtained by filling the hole in κ with M. Note that κ is *not* defined inductively.

$$\text{commuting conversion context} \qquad \kappa \quad ::= \quad \Box\ M \mid \mathsf{fst}\ \Box \mid \mathsf{snd}\ \Box \mid \mathsf{case}\ \Box\ \mathsf{of\ inl}\ x.\, M \mid \mathsf{inr}\ x.\, M \mid \mathsf{abort}_A\ \Box$$

Then a commuting conversion of M to N, written as $M \Longrightarrow_c N$, is defined as follows:

$$\kappa[\![\mathsf{case}\ M\ \mathsf{of\ inl}\ x.\, N \mid \mathsf{inr}\ y.\, N']\!] \quad \Longrightarrow_c \quad \mathsf{case}\ M\ \mathsf{of\ inl}\ x.\, \kappa[\![N]\!] \mid \mathsf{inr}\ y.\, \kappa[\![N']\!]$$

Exercise 3.13. Specify a commuting conversion for $\perp$. What is the result of applying a commuting conversion to $\kappa[\![\mathsf{abort}_M\ C]\!]$?

By extending the relation $\Longrightarrow_{\beta+}$ with the following rule, we can show that the normalization theorem holds for all kinds of proof terms:

$$\frac{M \Longrightarrow_c M'}{M \Longrightarrow_{\beta+} M'}$$

It can also be shown that both strong normalization and confluence continue to hold.

3.5 Proof terms in long normal form

Theorems 3.11 and 3.12 imply that given a proof term M of type A, we can obtain a unique intro term I such that $M \Longrightarrow^*_{\beta+} I$. The result, however, does not mean that there exists a unique intro term, if any, for every type A. (Remember that there is no proof term of type A if A *true* is not provable.) For example, the two normal proofs of $(A \supset B) \supset (A \supset B)\uparrow$ in Section 2.8 correspond to the following intro terms of type $(A \supset B) \supset (A \supset B)$ under the Curry-Howard isomorphism:

$$\lambda x\!:\!A \supset B.\, x$$
$$\lambda x\!:\!A \supset B.\, \lambda y\!:\!A.\, x\ y$$

We say that intro terms representing long normal proofs are in long normal form. Since long normal proofs are obtained by requiring proposition A in the rule $\downarrow\!\uparrow$ to be atomic, an elim term E is eligible as an intro term in long normal form only if it has an atomic type:

elim term	E	$::=$	$x \mid E\ I \mid \cdots$	
intro term in long normal form	I	$::=$	$E \mid \lambda x\!:\!A.\, I \mid \cdots$	where E has an atomic type

Then, for example, $\lambda x\!:\!A \supset B.\, x$ is not in long formal form because the type of x is not atomic.

It turns out that every intro term I can be converted into another intro term I' in long normal form by applying η-expansions to its subterms:

Theorem 3.14. *For every intro term I, there exists an intro term I' in long normal form such that $I \Longrightarrow_{\eta+} I'$.*

In order to convert an intro term into long normal form, we apply an η-expansion to each elim term *that is used as an intro term but does not have an atomic type*. For example, given an intro term $\lambda x\!:\!A \supset B.\, x$, we apply an η-expansion to variable x, which is an elim term being used as an intro term but does not have an atomic type, to obtain another intro term $\lambda x\!:\!A \supset B.\, \lambda y\!:\!A.\, x\ y$ in long normal form.

Exercise 3.15. In the presence of $\perp$, there can be different intro terms in long normal form of the same type. For example, $\lambda x\!:\!\perp.\, \mathsf{abort}_{A\supset A}\ x$ and $\lambda x\!:\!\perp.\, \lambda y\!:\!A.\, y$ are both intro terms in long normal form of type $\perp \supset (A \supset A)$. Now consider the fragment of propositional logic with the implication connective $\supset$ only. Given a type A known to have proof terms, are intro terms in long normal form unique (if we identify proof terms up to renaming bound variables)?

Chapter 4

Sequent Calculus

This chapter presents a *sequent calculus* for propositional logic. Although we set out to develop it as a device for proving the completeness property of normal proofs (Theorem 2.12), the sequent calculus also serves as a basis for proof search strategies implemented in theorem provers. Due to its important role in logic, a sequent calculus is not viewed as a secondary system derivable from a corresponding natural deduction system. Rather it is accepted as a valid formulation of a system of logic in itself, whether a corresponding natural deduction system has been formulated or not.

4.1 Sequent calculus for propositional logic

The sequent calculus for propositional logic consists of inference rules for *sequents* of the form $\Gamma \longrightarrow C$ where Γ is an unordered collection of propositions. Conceptually a sequent $A_1, \cdots, A_n \longrightarrow C$ becomes evident by a proof of $C\!\uparrow$ using neutral judgments $A_1\!\downarrow, \cdots, A_n\!\downarrow$:

$$A_1, \cdots, A_n \longrightarrow C \quad \Longleftrightarrow \quad \begin{array}{c} A_1\!\downarrow \quad \cdots \quad A_n\!\downarrow \\ \ddots\;\;\cdot^{\cdot^{\cdot}} \\ C\!\uparrow \end{array}$$

Note that the exchange rule is built into sequents because Γ in a sequent $\Gamma \longrightarrow C$ is an unordered collection of propositions.

It is important that unlike a hypothetical judgment $\Gamma \vdash J$ in which a judgment $J \in \Gamma$ is interpreted as a *hypothesis* $\overline{J}$, a proposition $A \in \Gamma$ in a sequent $\Gamma \longrightarrow C$ denotes just a *neutral judgment* $A\!\downarrow$, which may happen to originate from a hypothesis $\overline{A\!\downarrow}$, but not necessarily. For example, both proofs of $C\!\uparrow$ shown below make evident the same sequent $A \wedge B, A \longrightarrow C$:

$$\begin{array}{c} \dfrac{\overline{A \wedge B\!\downarrow}}{A\!\downarrow}\,\wedge\mathsf{E_L}_\downarrow \\ \vdots \\ C\!\uparrow \end{array} \qquad\qquad \begin{array}{c} \overline{A \wedge B\!\downarrow} \qquad\qquad \overline{A\!\downarrow} \\ \ddots\;\;\cdot^{\cdot^{\cdot}} \\ C\!\uparrow \end{array}$$

In the left proof, $A \wedge B\!\downarrow$, as well as $A\!\downarrow$, is available as a neutral judgment because the same hypothesis may be used more than once. In the right proof, both neutral judgments $A \wedge B\!\downarrow$ and $A\!\downarrow$ happen to originate from hypotheses. Still, however, we may think of $A \in \Gamma$ in a sequent $\Gamma \longrightarrow C$ as denoting a hypothesis $\overline{A\!\downarrow}$ available in the proof of $C\!\uparrow$, since as far as the proof of $C\!\uparrow$ is concerned, using $A\!\downarrow$ as a neutral judgment or as a hypothesis makes no difference.

An advantage of the sequent calculus over the natural deduction system consists in the fact that a proof of $\Gamma \longrightarrow C$ always proceeds in a bottom-up way, which implies that every inference rule in the sequent calculus is best read in a bottom-up way. (This is not the case in the natural deduction system because every elimination rule is best read in a top-down way.) Consider a sequent $A_1, \cdots, A_i, \cdots, A_n \longrightarrow C$:

$$A_1, \cdots, A_i, \cdots, A_n \longrightarrow C \quad \Longleftrightarrow \quad \begin{array}{ccccc} A_1 \!\downarrow & \cdots & A_i \!\downarrow & \cdots & A_n \!\downarrow \\ & \ddots & & \iddots & \\ & & C \!\uparrow & & \end{array}$$

For the sake of simplicity, let us assume that an introduction rule applied to $C\!\uparrow$ produces a new goal $C'\!\uparrow$ without producing a new hypothesis (as is the case for the rule $\vee\mathsf{I_{L\uparrow}}$), and that an elimination rule applied to $A_i\!\downarrow$ produces a new neutral judgment $A_i'\!\downarrow$ without requiring a separate proof of a normal judgment (as is the case for the rule $\wedge\mathsf{E_{L\downarrow}}$). If we choose to apply an introduction rule to $C\!\uparrow$, a new goal $C'\!\uparrow$ is produced. Thus we now have to prove $A_1, \cdots, A_i, \cdots, A_n \longrightarrow C'$:

$$\frac{A_1, \cdots, A_i, \cdots, A_n \longrightarrow C'}{A_1, \cdots, A_i, \cdots, A_n \longrightarrow C} \quad \Longleftrightarrow \quad \begin{array}{ccccc} A_1 \!\downarrow & \cdots & A_i \!\downarrow & \cdots & A_n \!\downarrow \\ & \ddots & & \iddots & \\ & & \dfrac{C'\!\uparrow}{C\!\uparrow} & & \end{array}$$

Such an inference rule in the sequent calculus is called a *right rule* because it focuses on the right side C in a sequent $\Gamma \longrightarrow C$. A right rule then corresponds to an introduction rule in the natural deduction system. If we choose to apply an elimination rule to $A_i\!\downarrow$, a new neutral judgment $A_i'\!\downarrow$ is produced while the goal $C\!\uparrow$ remains the same. Thus we now have to prove $A_1, \cdots, A_i, A_i', \cdots, A_n \longrightarrow C$:

$$\frac{A_1, \cdots, A_i, A_i', \cdots, A_n \longrightarrow C}{A_1, \cdots, A_i, \cdots, A_n \longrightarrow C} \quad \Longleftrightarrow \quad \begin{array}{ccccc} A_1 \!\downarrow & \cdots & \dfrac{A_i\!\downarrow}{A_i'\!\downarrow} & \cdots & A_n \!\downarrow \\ & \ddots & & \iddots & \\ & & C \!\uparrow & & \end{array}$$

Such an inference rule in the sequent calculus is called a *left rule* because it focuses on a proposition in the left side Γ in a sequent $\Gamma \longrightarrow C$. A left rule then corresponds to an elimination rule in the natural deduction system.

Keeping in mind the intuition behind sequents, let us consider each inference rule in the sequent calculus. The first rule is an axiom which deals with *initial sequents* of the form $\Gamma, A \longrightarrow A$:

$$\frac{}{\Gamma, A \longrightarrow A}\ \mathit{Init}$$

Note that the rule *Init* corresponds not to the rule Hyp but to the rule $\downarrow\!\uparrow$ in the natural deduction system: it is *not* a rule using a hypothesis; rather it is a rule deducing $A\!\uparrow$ from $A\!\downarrow$.

For conjunction $\wedge$, we need two left rules $\wedge L_L$ and $\wedge L_R$, corresponding to the elimination rules $\wedge\mathsf{E_{L\downarrow}}$ and $\wedge\mathsf{E_{R\downarrow}}$, and one right rule $\wedge R$, corresponding to the introduction rule $\wedge\mathsf{I_\uparrow}$:

$$\frac{\Gamma, A \wedge B, A \longrightarrow C}{\Gamma, A \wedge B \longrightarrow C}\ \wedge L_L \qquad \frac{\Gamma, A \wedge B, B \longrightarrow C}{\Gamma, A \wedge B \longrightarrow C}\ \wedge L_R \qquad \frac{\Gamma \longrightarrow A \quad \Gamma \longrightarrow B}{\Gamma \longrightarrow A \wedge B}\ \wedge R$$

For implication $\supset$, we need one left rule, corresponding to the elimination rule $\supset\!\mathsf{E_\downarrow}$, and one right rule, corresponding to the introduction rule $\supset\!\mathsf{I_\uparrow}$. Suppose that we wish to prove $\Gamma, A \supset B \longrightarrow C$ by focusing on $A \supset B$ in the left side:

$$\Gamma, A \supset B \longrightarrow C \quad \Longleftrightarrow \quad \begin{array}{ccc} \Gamma_\downarrow & \cdots & A \supset B\!\downarrow \\ & \ddots\ \iddots & \\ & C\!\uparrow & \end{array}$$

Here $\Gamma_\downarrow$ is a shorthand for $\{A\!\downarrow \mid A \in \Gamma\}$. In order to apply the rule $\supset\!\mathsf{E_\downarrow}$ to $A \supset B\!\downarrow$, we first have to build a proof of $A\!\uparrow$ using $\Gamma_\downarrow$ and $A \supset B\!\downarrow$, which means that we need a proof of $\Gamma, A \supset B \longrightarrow A$:

$$\frac{\Gamma, A \supset B \longrightarrow A \quad \cdots}{\Gamma, A \supset B \longrightarrow C} \quad \Longleftrightarrow \quad \begin{array}{cccc} & & & \begin{array}{ccc} \Gamma_\downarrow & \cdots & A \supset B\!\downarrow \\ & \ddots\ \iddots & \end{array} \\ \Gamma_\downarrow & \cdots & A \supset B\!\downarrow & A\!\uparrow \\ & \ddots\ \iddots & & \\ & C\!\uparrow & & \end{array}$$

$$\frac{}{\Gamma, A \longrightarrow A}\,\mathit{Init} \quad \frac{\Gamma, A \wedge B, A \longrightarrow C}{\Gamma, A \wedge B \longrightarrow C}\,\wedge L_L \quad \frac{\Gamma, A \wedge B, B \longrightarrow C}{\Gamma, A \wedge B \longrightarrow C}\,\wedge L_R \quad \frac{\Gamma \longrightarrow A \quad \Gamma \longrightarrow B}{\Gamma \longrightarrow A \wedge B}\,\wedge R$$

$$\frac{\Gamma, A \supset B \longrightarrow A \quad \Gamma, A \supset B, B \longrightarrow C}{\Gamma, A \supset B \longrightarrow C}\,{\supset}L \quad \frac{\Gamma, A \longrightarrow B}{\Gamma \longrightarrow A \supset B}\,{\supset}R$$

$$\frac{\Gamma, A \vee B, A \longrightarrow C \quad \Gamma, A \vee B, B \longrightarrow C}{\Gamma, A \vee B \longrightarrow C}\,\vee L \quad \frac{\Gamma \longrightarrow A}{\Gamma \longrightarrow A \vee B}\,\vee R_L \quad \frac{\Gamma \longrightarrow B}{\Gamma \longrightarrow A \vee B}\,\vee R_R$$

$$\frac{}{\Gamma \longrightarrow \top}\,\top R \quad \frac{}{\Gamma, \bot \longrightarrow C}\,\bot L \quad \frac{\Gamma, \neg A \longrightarrow A}{\Gamma, \neg A \longrightarrow C}\,\neg L \quad \frac{\Gamma, A \longrightarrow \bot}{\Gamma \longrightarrow \neg A}\,\neg R$$

Figure 4.1: Sequent calculus for propositional logic

Then a new neutral judgment $B\!\downarrow$ becomes available for the proof of $C\!\uparrow$, which means that it now suffices to prove $\Gamma, A \supset B, B \longrightarrow C$:

$$\frac{\Gamma, A \supset B \longrightarrow A \quad \Gamma, A \supset B, B \longrightarrow C}{\Gamma, A \supset B \longrightarrow C} \quad \Longleftrightarrow \quad \begin{array}{c} \Gamma_{\downarrow} \quad \cdots \quad \dfrac{A \supset B\!\downarrow \qquad \begin{array}{c}\Gamma_{\downarrow} \quad \cdots \quad A \supset B\!\downarrow \\ \ddots\;\cdot^{\cdot^{\cdot}} \\ A\!\uparrow\end{array}}{B\!\downarrow}\,{\supset}\mathsf{E}_{\downarrow} \\ \ddots\;\cdot^{\cdot^{\cdot}} \\ C\!\uparrow \end{array}$$

Thus we obtain the following left rule ${\supset}L$; the right rule ${\supset}R$ is obtained by a similar analysis:

$$\frac{\Gamma, A \supset B \longrightarrow A \quad \Gamma, A \supset B, B \longrightarrow C}{\Gamma, A \supset B \longrightarrow C}\,{\supset}L \qquad \frac{\Gamma, A \longrightarrow B}{\Gamma \longrightarrow A \supset B}\,{\supset}R$$

For disjunction $\vee$, we need one left rule $\vee L$, corresponding to the elimination rule $\vee\mathsf{E}_{\uparrow}$, and two right rules $\vee R_L$ and $\vee R_R$, corresponding to the introduction rules $\vee\mathsf{I}_{\mathsf{L}\uparrow}$ and $\vee\mathsf{I}_{\mathsf{R}\uparrow}$; the rule $\vee L$ is obtained in a similar way to the rule ${\supset}L$:

$$\frac{\Gamma, A \vee B, A \longrightarrow C \quad \Gamma, A \vee B, B \longrightarrow C}{\Gamma, A \vee B \longrightarrow C}\,\vee L \qquad \frac{\Gamma \longrightarrow A}{\Gamma \longrightarrow A \vee B}\,\vee R_L \qquad \frac{\Gamma \longrightarrow B}{\Gamma \longrightarrow A \vee B}\,\vee R_R$$

The rule $\vee L$ is designed in such a way that commuting conversion is built into the sequent calculus. To see why, observe that $\Gamma, A \vee B \longrightarrow C$ in the conclusion describes a proof of the goal $C\!\uparrow$ in which the rule $\vee\mathsf{E}$ is to be applied to $A \vee B\!\downarrow$. Then $\Gamma, A \vee B, A \longrightarrow C$ and $\Gamma, A \vee B, B \longrightarrow C$ in the premises indicate that the rule $\vee\mathsf{E}$ applied to $A \vee B\!\downarrow$ uses the same goal $C\!\uparrow$ in its conclusion. According to the rule $\vee L$, therefore, the conclusion in any instance of the rule $\vee\mathsf{E}$ in the natural deduction system is always the current goal, which means that commuting conversion is built into the sequent calculus.

For truth $\top$, we need a right rule $\top R$ corresponding to the introduction rule $\top\mathsf{I}$, but no left rule (because there is no elimination rule for $\top$); for falsehood $\bot$, we need a left rule $\bot L$ corresponding to the elimination rule $\bot\mathsf{E}$, but no right rule (because there is no introduction rule for $\bot$):

$$\frac{}{\Gamma \longrightarrow \top}\,\top R \qquad \frac{}{\Gamma, \bot \longrightarrow C}\,\bot L$$

Figure 4.1 shows all the inference rules in the sequent calculus for propositional logic. Note that each rule R focuses on a proposition in the sequent of the conclusion, which appears in the left side if R is a left rule, in the right side if R is a right rule, and in both sides if R is the rule *Init*. For example, the rule $\wedge L_L$ focuses on $A \wedge B$ in the left side, the rule $\wedge R$ on $A \wedge B$ in the right side, and the rule *Init* on A. We refer to such a proposition as the *principal formula* of the rule.

The rules $\neg L$ and $\neg R$ are obtained from the notational definition $\neg A = A \supset \bot$. In particular, the rule $\neg L$ implicitly uses the rule $\bot L$:

$$\frac{\Gamma, A \supset \bot \longrightarrow A \quad \dfrac{}{\Gamma, A \supset \bot, \bot \longrightarrow C}\,\bot L}{\Gamma, A \supset \bot \longrightarrow C}\,{\supset}L$$

As in the natural deduction system, the weakening and contraction properties allow us to use a proposition $A \in \Gamma$ zero times and more than once, respectively, in a proof of $\Gamma \longrightarrow C$. Note that the structural properties allow us to identify two sequents $\Gamma \longrightarrow C$ and $\Gamma' \longrightarrow C$ if Γ and Γ' are equivalent as sets (rather than as multisets), *i.e.*, $\{A \mid A \in \Gamma\} = \{A \mid A \in \Gamma'\}$.

Proposition 4.1 (Structural properties).
$\Gamma_\downarrow = \{A\downarrow \mid A \in \Gamma\}$ *is a collection of neutral judgments.*
(Weakening) *If* $\Gamma \longrightarrow C$, *then* $\Gamma, A \longrightarrow C$.
(Contraction) *If* $\Gamma, A, A \longrightarrow C$, *then* $\Gamma, A \longrightarrow C$.

Proof. By induction on the structure of the proof of $\Gamma \longrightarrow C$ and $\Gamma, A, A \longrightarrow C$. □

The sequent calculus in Figure 4.1 satisfies the *subformula property* that every proposition (or formula) in the premise of a rule is a subformula of a certain proposition (or formula) in the conclusion, where the subformula relation is defined as follows: (1) A is a subformula of A; (2) A and B are subformulae of $A \supset B$, $A \wedge B$, and $A \vee B$. For example, the premise of the rule $\wedge L_L$ introduces a new proposition A, which is a subformula of $A \wedge B$ in the conclusion; the premises of the rule $\wedge R$ introduce two new propositions A and B, both of which are subformulae of $A \wedge B$ in the conclusion.

Because of the subformula property, a proof of $\Gamma \longrightarrow C$ needs to consider only subformulae of those propositions in Γ or C. For example, a proof of $\cdot \longrightarrow A \supset (B \supset C)$ never involves an analysis of $A \supset B$ by applying the rule $\supset L$ or $\supset R$ because it is not a subformula of $A \supset (B \supset C)$. In conjunction with the structural properties, therefore, the subformula property implies that the sequent calculus in Figure 4.1 is decidable: there exists a procedure for deciding whether $\Gamma \longrightarrow C$ is provable or not. Intuitively a proof of $\Gamma \longrightarrow C$ generates a finite number of sequents because only a finite number of propositions need to be considered.

Proposition 4.2. *The sequent calculus in Figure 4.1 is decidable.*

Proof. Let us write $\Gamma^\star$ for a set $\{A \mid A \in \Gamma\}$ consisting of elements of a multiset Γ. By the structural properties, $\Gamma \longrightarrow C$ is provable if and only if $\Gamma^\star \longrightarrow C$ is provable. When proving a sequent, therefore, we implicitly use only those sequents of the form $\Gamma^\star \longrightarrow C$ in which no proposition appears more than once in $\Gamma^\star$.

Suppose that we wish to check the provability of the goal sequent $\Gamma \longrightarrow C$. First we generate the set S of all possible sequents using subformulae of propositions in Γ and C. S must be a finite set because of the subformula property. (If Γ and C produce n different subformulae, there are a total of $2^n \times n$ sequents in S.) Next we check each sequent in S and mark it as "proven" if it is provable by the rule *Init*, $\top R$, or $\bot L$, which are the rules with no premise. Then, for each rule except *Init*, $\top R$, or $\bot L$, we consider all possible combinations of those sequents marked as "proven" for its premise, and mark as "proven" the sequent corresponding to the conclusion if it is not marked as "proven" yet. For example, for the rule $\wedge L_L$, we mark every sequent of the form $\Gamma, A \wedge B \longrightarrow C$ as "proven" if $\Gamma, A \wedge B, A \longrightarrow C$ is already marked as "proven." Similarly, for the rule $\supset L$, we mark every sequent of the form $\Gamma, A \supset B \longrightarrow C$ as "proven" if $\Gamma, A \supset B \longrightarrow A$ and $\Gamma, A \supset B, B \longrightarrow C$ are already marked as "proven." We repeat the procedure until no more sequent in S can be marked as "proven." The procedure must eventually terminate because the number of combinations of the rules and the sequents in S is finite. If the goal sequent is marked as "proven," we decide that it is provable; otherwise it is not provable. (The procedure described above is the basis for a practical proof search technique called the *inverse method*.) □

The soundness and completeness properties of the sequent calculus show that it is equivalent to the natural deduction system for normal judgments.

Theorem 4.3 (Soundness of the sequent calculus). *If* $\Gamma \longrightarrow C$, *then* $\Gamma_\downarrow \vdash C\uparrow$.

Theorem 4.4 (Completeness of the sequent calculus). *If* $\Gamma_\downarrow \vdash C\uparrow$, *then* $\Gamma \longrightarrow C$.

Note that while $A\downarrow \in \Gamma_\downarrow$ in a hypothetical judgment $\Gamma_\downarrow \vdash C\uparrow$ denotes a hypothesis $\overline{A\downarrow}$, $A \in \Gamma$ in a sequent $\Gamma \longrightarrow C$ denotes just a neutral judgment $A\downarrow$, which is not necessarily a hypothesis. The discrepancy does not invalidate the two theorems, however, since as far as the proof of $C\uparrow$ is concerned, using $A\downarrow$ as a hypothesis when it is a neutral judgment, or vice versa, makes no difference.

The proof of the soundness property is straightforward:

Proof of Theorem 4.3. By induction on the structure of the proof of $\Gamma \longrightarrow C$. Below we show three representative cases. We reuse metavariables Γ and C.

Case $\dfrac{}{\Gamma, A \longrightarrow A}\ \mathit{Init}$

$\Gamma_{\downarrow}, A{\downarrow} \vdash A{\downarrow}$ — by the rule $\mathsf{Hyp}_{\downarrow}$
$\Gamma_{\downarrow}, A{\downarrow} \vdash A{\uparrow}$ — by the rule ${\downarrow\uparrow}$

Case $\dfrac{\Gamma, A \supset B \longrightarrow A \quad \Gamma, A \supset B, B \longrightarrow C}{\Gamma, A \supset B \longrightarrow C}\ {\supset}L$

$\Gamma_{\downarrow}, A \supset B{\downarrow} \vdash A \supset B{\downarrow}$ — by the rule $\mathsf{Hyp}_{\downarrow}$
$\Gamma_{\downarrow}, A \supset B{\downarrow} \vdash A{\uparrow}$ — by induction hypothesis on $\Gamma, A \supset B \longrightarrow A$
$\Gamma_{\downarrow}, A \supset B{\downarrow} \vdash B{\downarrow}$ — by the rule ${\supset}\mathsf{E}_{\downarrow}$
$\Gamma_{\downarrow}, A \supset B{\downarrow}, B{\downarrow} \vdash C{\uparrow}$ — by induction hypothesis on $\Gamma, A \supset B, B \longrightarrow C$
$\Gamma_{\downarrow}, A \supset B{\downarrow} \vdash C{\uparrow}$ — by the substitution principle (Theorem 2.11)

Case $\dfrac{\Gamma, A \longrightarrow B}{\Gamma \longrightarrow A \supset B}\ {\supset}R$

$\Gamma_{\downarrow}, A{\downarrow} \vdash B{\uparrow}$ — by induction hypothesis on $\Gamma, A \longrightarrow B$
$\Gamma_{\downarrow} \vdash A \supset B{\uparrow}$ — by the rule ${\supset}\mathsf{I}_{\uparrow}$

□

On the other hand, the proof of the completeness property is not so straightforward. In fact, it is easy to see that a direct proof attempt fails because inference rules for normal judgments use neutral judgments as well, but the theorem does not mention neutral judgments at all. For example, if the proof of $\Gamma_{\downarrow} \vdash C{\uparrow}$ ends with an application of the rule ${\downarrow\uparrow}$, the premise is $\Gamma_{\downarrow} \vdash C{\downarrow}$, to which induction hypothesis cannot be applied. Therefore we need to generalize the theorem so that a hypothetical judgment $\Gamma_{\downarrow} \vdash A{\downarrow}$ is also related to sequents in a certain way.

Lemma 4.5 below generalizes Theorem 4.4. The proof itself is straightforward, but formulating the statement connecting $\Gamma_{\downarrow} \vdash A{\downarrow}$ to sequents is far from trivial. Theorem 4.4 follows as an immediate consequence of Lemma 4.5.

Lemma 4.5.
If $\Gamma_{\downarrow} \vdash A{\downarrow}$, then $\Gamma, A \longrightarrow C$ implies $\Gamma \longrightarrow C$.
If $\Gamma_{\downarrow} \vdash C{\uparrow}$, then $\Gamma \longrightarrow C$.

Proof. By simultaneous induction on the structure of the proof of $\Gamma_{\downarrow} \vdash A{\downarrow}$ and $\Gamma_{\downarrow} \vdash C{\uparrow}$. Below we reuse metavariables Γ, A, and C.

Case $\dfrac{}{\Gamma_{\downarrow}, A{\downarrow} \vdash A{\downarrow}}\ \mathsf{Hyp}_{\downarrow}$

$\Gamma, A, A \longrightarrow C$ — assumption
$\Gamma, A \longrightarrow C$ — by contraction

Case $\dfrac{\Gamma_{\downarrow}, A{\downarrow} \vdash B{\uparrow}}{\Gamma_{\downarrow} \vdash A \supset B{\uparrow}}\ {\supset}\mathsf{I}_{\uparrow}$

$\Gamma, A \longrightarrow B$ — by induction hypothesis on $\Gamma_{\downarrow}, A{\downarrow} \vdash B{\uparrow}$
$\Gamma \longrightarrow A \supset B$ — by the rule ${\supset}R$

Case $\dfrac{\Gamma_{\downarrow} \vdash A \supset B{\downarrow} \quad \Gamma_{\downarrow} \vdash A{\uparrow}}{\Gamma_{\downarrow} \vdash B{\downarrow}}\ {\supset}\mathsf{E}_{\downarrow}$

$\Gamma, B \longrightarrow C$ — assumption
$\Gamma, A \supset B, B \longrightarrow C$ — by weakening
$\Gamma \longrightarrow A$ — by induction hypothesis on $\Gamma_{\downarrow} \vdash A{\uparrow}$
$\Gamma, A \supset B \longrightarrow A$ — by weakening
$\Gamma, A \supset B \longrightarrow C$ — by the rule ${\supset}L$ on $\Gamma, A \supset B \longrightarrow A$ and $\Gamma, A \supset B, B \longrightarrow C$
$\Gamma \longrightarrow C$ — by induction hypothesis on $\Gamma_{\downarrow} \vdash A \supset B{\downarrow}$ with $\Gamma, A \supset B \longrightarrow C$

Case $\dfrac{\Gamma_{\downarrow} \vdash A{\downarrow}}{\Gamma_{\downarrow} \vdash A{\uparrow}}\ {\downarrow\uparrow}$

$\Gamma, A \longrightarrow A$ by the rule *Init*
$\Gamma \longrightarrow A$ by induction hypothesis on $\Gamma_\downarrow \vdash A\downarrow$ with $\Gamma, A \longrightarrow A$

□

The proof of Lemma 4.5 is *constructive*, as opposed to *declarative*, in the sense that it gives an algorithm for converting a proof of $\Gamma_\downarrow \vdash C\uparrow$ into a proof of $\Gamma \longrightarrow C$. The key to understanding its constructive nature is to observe that when converting $\Gamma_\downarrow \vdash A\downarrow$, a proof of $\Gamma, A \longrightarrow C$ for some proposition C is given as an assumption so that a new proof of $\Gamma \longrightarrow C$ is produced. For example, the case $\dfrac{\Gamma_\downarrow \vdash A\downarrow}{\Gamma_\downarrow \vdash A\uparrow}\ \downarrow\uparrow$ creates a proof of $\Gamma, A \longrightarrow A$ using the rule *Init*, which then serves as an assumption in converting the premise $\Gamma_\downarrow \vdash A\downarrow$. The case $\overline{\Gamma_\downarrow, A\downarrow \vdash A\downarrow}\ \mathsf{Hyp}_\downarrow$ uses the contraction property to deduce $\Gamma, A \longrightarrow C$ from such an assumption $\Gamma, A, A \longrightarrow C$.

4.2 Cut elimination

We have seen that in the natural deduction system based on hypothetical judgments, reflexivity and the substitution principle confirm that the system adheres to the definition of hypothetical judgments. That is, failure of reflexivity or the substitution principle indicates the existence of an inference rule that does not respect the definition of hypothetical judgments as concise representations of hypothetical proofs.

In the case of the sequent calculus, we may test its integrity by checking two similar principles, especially in view of the fact that $A \in \Gamma$ in a sequent $\Gamma \longrightarrow C$ can be thought of as denoting a hypothesis $\overline{A\downarrow}$. The first, corresponding to reflexivity in the natural deduction system, is the provability of every initial sequent $\Gamma, A \longrightarrow A$, which directly follows from the rule *Init*. The second, corresponding to the substitution principle, is the *admissibility of the cut rule* (where the cut rule is another rule to be explained later):

Theorem 4.6 (Admissiblity of the cut rule). *If $\Gamma \longrightarrow A$ and $\Gamma, A \longrightarrow C$, then $\Gamma \longrightarrow C$.*

Thus the admissibility of the cut rule is to the sequent calculus what the substitution principle is to the natural deduction system: if the substitution principle fails, it indicates that the natural deduction system is not sound (or even non-sense); similarly if the admissibility of the cut rule fails, it indicates that the sequent calculus is not sound (or even non-sense).

Theorem 4.6 implies that if a new rule $\dfrac{A\uparrow}{A\downarrow}\ \uparrow\downarrow$ is added to the natural deduction system for normal and neutral judgments, we can safely remove any occurrence of the rule $\uparrow\downarrow$ in a proof of $C\uparrow$. The intuition is that Theorem 4.6 may be rewritten in terms of normal and neutral judgments as follows:

$$\textit{If}\quad \begin{array}{c} \Gamma_\downarrow \\ \vdots \\ \dfrac{A\uparrow}{A\downarrow}\ \uparrow\downarrow \\ \vdots \\ C\uparrow \end{array} \quad \textit{, then} \quad \begin{array}{c} \Gamma_\downarrow \\ \vdots \\ C\uparrow \end{array}\ .$$

Since an occurrence of the rule $\uparrow\downarrow$ in a proof of $C\uparrow$ corresponds to a detour in a proof of $C\ true$, Theorem 4.6 implies in turn that we can transform the *entire* proof of $C\ true$ so as to remove any detour in it. In this sense, Theorem 4.6 states that the natural deduction system for truth judgments is *globally sound*. (Recall that the local soundness property states that a detour specific to a connective can be locally eliminated, without transforming the entire proof.) We will later formalize the global soundness property as the normalization theorem (Theorem 4.8).

The proof of Theorem 4.6 proceeds by nested induction on the structure of: 1) proposition A which is called the *cut formula*; 2) proof of $\Gamma \longrightarrow A$; 3) proof of $\Gamma, A \longrightarrow C$. Here are a few examples of applying induction hypothesis in the proof of Theorem 4.6:

- We wish to prove that $\Gamma \longrightarrow A \supset B$ and $\Gamma, A \supset B \longrightarrow C$ imply $\Gamma \longrightarrow C$. Since A is a subformula of the cut formula $A \supset B$, the induction hypothesis on A proves that $\Gamma' \longrightarrow A$ and $\Gamma', A \longrightarrow C'$

imply $\Gamma' \longrightarrow C'$ for any Γ' and C', and also regardless of the structure of the proof of $\Gamma' \longrightarrow A$ and $\Gamma', A \longrightarrow C'$.

- We wish to prove that $\Gamma \longrightarrow A$ and $\Gamma, A \longrightarrow C$ imply $\Gamma \longrightarrow C$. Suppose that the proof of $\Gamma \longrightarrow A$ has the following structure:

$$\frac{\cdots \quad \overset{\mathcal{D}}{\Gamma, B \longrightarrow A}}{\Gamma \longrightarrow A}\ R$$

  Then we weaken $\Gamma, A \longrightarrow C$ to obtain a proof $\mathcal{E}$ of $\Gamma, B, A \longrightarrow C$. Since $\mathcal{D}$ is strictly smaller than the proof of $\Gamma \longrightarrow A$, the induction hypothesis on proposition A, proof $\mathcal{D}$, and proof $\mathcal{E}$ yields $\Gamma, B \longrightarrow C$, irrespective of the structure (or size) of $\mathcal{E}$.

- We wish to prove that $\Gamma \longrightarrow A$ and $\Gamma, A \longrightarrow C$ imply $\Gamma \longrightarrow C$. Suppose that the proof of $\Gamma, A \longrightarrow C$ has the following structure:

$$\frac{\cdots \quad \overset{\mathcal{E}}{\Gamma, B, A \longrightarrow C'}}{\Gamma, A \longrightarrow C}\ R$$

  Then we weaken $\Gamma \longrightarrow A$ to obtain a proof $\mathcal{D}$ of $\Gamma, B \longrightarrow A$, which has exactly the same structure (or size) as the proof of $\Gamma \longrightarrow A$. Since $\mathcal{E}$ is strictly smaller than the proof of $\Gamma, A \longrightarrow C$, the induction hypothesis on proposition A, proof $\mathcal{D}$, and proof $\mathcal{E}$ yields $\Gamma, B \longrightarrow C'$. (Then we typically apply the same rule R to deduce $\Gamma \longrightarrow C$.)

The proof of Theorem 4.6 considers all possible combinations of the last inference rule $R_{\mathcal{D}}$ in the proof $\mathcal{D}$ of $\Gamma \longrightarrow A$ and the last inference rule $R_{\mathcal{E}}$ in the proof $\mathcal{E}$ of $\Gamma, A \longrightarrow C$. The combinations of the rules $R_{\mathcal{D}}$ and $R_{\mathcal{E}}$ are divided as follows:

1. At least one of $R_{\mathcal{D}}$ and $R_{\mathcal{E}}$ is the rule *Init*.
   (a) $R_{\mathcal{D}}$ is the rule *Init*. In this case, we have $\Gamma = \Gamma', A$.
   (b) $R_{\mathcal{E}}$ is the rule *Init*. In this case, we have either $\Gamma = \Gamma', C$ or $A = C$.
2. Neither of $R_{\mathcal{D}}$ and $R_{\mathcal{E}}$ is the rule *Init*.
   (a) A is the principal formula of both $R_{\mathcal{D}}$ and $R_{\mathcal{E}}$. In this case, $R_{\mathcal{D}}$ is a right rule and $R_{\mathcal{E}}$ is a left rule.
   (b) A is not the principal formula of $R_{\mathcal{D}}$. In this case, $R_{\mathcal{D}}$ is a left rule.
   (c) A is not the principal formula of $R_{\mathcal{E}}$. In this case, $R_{\mathcal{E}}$ can be both a left rule and a right rule.

Note that 1-(a) and 1-(b) overlap because both $R_{\mathcal{D}}$ and $R_{\mathcal{E}}$ can be the rule *Init*, and that 2-(b) and 2-(c) overlap because A may be the principal formula of neither $R_{\mathcal{D}}$ nor $R_{\mathcal{E}}$.

The proof of Theorem 4.6 is constructive because it gives an algorithm for building a proof of $\Gamma \longrightarrow C$ out of proofs of $\Gamma \longrightarrow A$ and $\Gamma, A \longrightarrow C$. The algorithm is non-deterministic because of the overlapping cases 1-(a) and 1-(b), and 2-(b) and 2-(c).

Proof of Theorem 4.6. By nested induction on the structure of: 1) cut formula A; 2) proof of $\Gamma \longrightarrow A$; 3) proof of $\Gamma, A \longrightarrow C$. Here we consider the fragment of the sequent calculus with the rules *Init*, $\supset L$, and $\supset R$ only.

We write $\mathcal{D} :: J$ or $\overset{\mathcal{D}}{J}$ to say that $\mathcal{D}$ is a proof of J. Let $R_{\mathcal{D}}$ be the last inference rule in the proof $\mathcal{D}$ of $\Gamma \longrightarrow A$ and $R_{\mathcal{E}}$ the last inference rule in the proof $\mathcal{E}$ of $\Gamma, A \longrightarrow C$.

Case 1-(a): $R_{\mathcal{D}}$ is the rule *Init*. We have $\Gamma = \Gamma', A$.

$$\mathcal{D} = \frac{}{\Gamma', A \longrightarrow A}\ \mathit{Init}$$

$\Gamma', A, A \longrightarrow C$	assumption
$\Gamma', A \longrightarrow C$	by contraction
$\Gamma \longrightarrow C$	from $\Gamma = \Gamma', A$

Case 1-(b): $R_{\mathcal{E}}$ is the rule *Init*.

Subcase: $\Gamma = \Gamma', C$

$$\mathcal{E} = \frac{}{\Gamma', C, A \longrightarrow C}\ \mathit{Init}$$

$\Gamma', C \longrightarrow C$	by the rule *Init*
$\Gamma \longrightarrow C$	from $\Gamma = \Gamma', C$

Subcase: $A = C$

$\Gamma \longrightarrow A$	assumption
$\Gamma \longrightarrow C$	from $A = C$

Case 2-(a): A is the principal formula of both $R_{\mathcal{D}}$ and $R_{\mathcal{E}}$. We have $A = A_1 \supset A_2$.

$$\mathcal{D} = \frac{\overset{\mathcal{D}_1}{\Gamma, A_1 \longrightarrow A_2}}{\Gamma \longrightarrow A_1 \supset A_2}\ {\supset}R \qquad \mathcal{E} = \frac{\overset{\mathcal{E}_1}{\Gamma, A_1 \supset A_2 \longrightarrow A_1} \quad \overset{\mathcal{E}_2}{\Gamma, A_1 \supset A_2, A_2 \longrightarrow C}}{\Gamma, A_1 \supset A_2 \longrightarrow C}\ {\supset}L$$

$\mathcal{E}_1' :: \Gamma \longrightarrow A_1$	by IH on $A_1 \supset A_2$, $\mathcal{D}$, and $\mathcal{E}_1$
$\mathcal{D}' :: \Gamma, A_2 \longrightarrow A_1 \supset A_2$	by weakening $\mathcal{D} :: \Gamma \longrightarrow A_1 \supset A_2$
$\mathcal{E}_2' :: \Gamma, A_2 \longrightarrow C$	by IH on $A_1 \supset A_2$, $\mathcal{D}'$, and $\mathcal{E}_2$
$\mathcal{D}_1' :: \Gamma \longrightarrow A_2$	by IH on A_1, $\mathcal{E}_1'$, $\mathcal{D}_1$
$\Gamma \longrightarrow C$	by IH on A_2, $\mathcal{D}_1'$, and $\mathcal{E}_2'$

Case 2-(b): A is not the principal formula of $R_{\mathcal{D}}$. We have $\Gamma = \Gamma', B_1 \supset B_2$.

$$\mathcal{D} = \frac{\overset{\mathcal{D}_1}{\Gamma', B_1 \supset B_2 \longrightarrow B_1} \quad \overset{\mathcal{D}_2}{\Gamma', B_1 \supset B_2, B_2 \longrightarrow A}}{\Gamma', B_1 \supset B_2 \longrightarrow A}\ {\supset}L$$

$\mathcal{E}' :: \Gamma', B_1 \supset B_2, B_2, A \longrightarrow C$	by weakening $\mathcal{E} :: \Gamma', B_1 \supset B_2, A \longrightarrow C$
$\mathcal{E}'' :: \Gamma', B_1 \supset B_2, B_2 \longrightarrow C$	by IH on A, $\mathcal{D}_2$, and $\mathcal{E}'$
$\Gamma', B_1 \supset B_2 \longrightarrow C$	by the rule ${\supset}L$ with $\mathcal{D}_1$ and $\mathcal{E}''$
$\Gamma \longrightarrow C$	from $\Gamma = \Gamma', B_1 \supset B_2$

Case 2-(c): A is not the principal formula of $R_{\mathcal{E}}$.

Subcase: $\Gamma = \Gamma', B_1 \supset B_2$ where $B_1 \supset B_2$ is the principal formula of $R_{\mathcal{E}}$

$$\mathcal{E} :: \frac{\overset{\mathcal{E}_1}{\Gamma', B_1 \supset B_2, A \longrightarrow B_1} \quad \overset{\mathcal{E}_2}{\Gamma', B_1 \supset B_2, A, B_2 \longrightarrow C}}{\Gamma', B_1 \supset B_2, A \longrightarrow C}\ {\supset}L$$

$\mathcal{E}_1' :: \Gamma', B_1 \supset B_2 \longrightarrow B_1$	by IH on A, $\mathcal{D}$, and $\mathcal{E}_1$
$\mathcal{D}' :: \Gamma', B_1 \supset B_2, B_2 \longrightarrow A$	by weakening $\mathcal{D} :: \Gamma \longrightarrow A$ (with $\Gamma = \Gamma', B_1 \supset B_2$)
$\mathcal{E}_2' :: \Gamma', B_1 \supset B_2, B_2 \longrightarrow C$	by IH on A, $\mathcal{D}'$, and $\mathcal{E}_2$
$\Gamma', B_1 \supset B_2 \longrightarrow C$	by the rule ${\supset}L$ with $\mathcal{E}_1'$ and $\mathcal{E}_2'$
$\Gamma \longrightarrow C$	from $\Gamma = \Gamma', B_1 \supset B_2$

Subcase: $C = C_1 \supset C_2$ is the principal formula of $R_{\mathcal{E}}$

$$\mathcal{E} :: \frac{\overset{\mathcal{E}_1}{\Gamma, A, C_1 \longrightarrow C_2}}{\Gamma, A \longrightarrow C_1 \supset C_2}\ {\supset}R$$

$\mathcal{D}' :: \Gamma, C_1 \longrightarrow A$	by weakening $\mathcal{D} :: \Gamma \longrightarrow A$
$\mathcal{E}_1' :: \Gamma, C_1 \longrightarrow C_2$	by IH on A, $\mathcal{D}'$, and $\mathcal{E}_1$

$\Gamma \longrightarrow C_1 \supset C_2$ by the rule $\supset R$ with $\mathcal{E}_1'$
$\Gamma \longrightarrow C$ from $C = C_1 \supset C_2$
□

The admissibility of the cut rule has as a corollary one of the central theorems in the study of logic: *cut elimination* (also called *Hauptsatz* meaning "main theorem"). Consider an extension of the sequent calculus with the cut rule shown below, where we use sequents of the form $\Gamma \overset{+}{\longrightarrow} C$ to distinguish the extended system from the system in Figure 4.1:

$$\frac{\Gamma \overset{+}{\longrightarrow} A \quad \Gamma, A \overset{+}{\longrightarrow} C}{\Gamma \overset{+}{\longrightarrow} C}\ \mathit{Cut}$$

Cut elimination states that the rule *Cut* is redundant:

Theorem 4.7 (Cut elimination). *$\Gamma \longrightarrow C$ if and only if $\Gamma \overset{+}{\longrightarrow} C$.*

Proof. The *only if* part is trivial. For the *if* part, we proceed by induction on the structure of the proof of $\Gamma \overset{+}{\longrightarrow} C$. The only interesting case is the rule *Cut*:

Case $\dfrac{\Gamma \overset{+}{\longrightarrow} A \quad \Gamma, A \overset{+}{\longrightarrow} C}{\Gamma \overset{+}{\longrightarrow} C}\ \mathit{Cut}$

$\Gamma \longrightarrow A$ by induction hypothesis on $\Gamma \overset{+}{\longrightarrow} A$
$\Gamma, A \longrightarrow C$ by induction hypothesis on $\Gamma, A \overset{+}{\longrightarrow} C$
$\Gamma \longrightarrow C$ by Theorem 4.6
□

Note that the rule *Cut* destroys the subformula property: it does not analyze a proposition in the conclusion, so A can be an arbitrary proposition completely unrelated to Γ and C. Thus the presence of the rule *Cut* makes it difficult to prove a sequent because each application of the rule *Cut* must "guess" such a proposition A. Fortunately the cut elimination theorem says that the rule *Cut* can be discarded without sacrificing the expressive power of the sequent calculus.

4.3 Normalization for the natural deduction system

Theorem 4.8 states that for every proof of A *true*, there exists a proof of $A\!\uparrow$. We now appeal to the cut elimination theorem to prove the same result, but covering all connectives (including $\vee$ and $\bot$). Our goal is to prove the normalization theorem stated in terms of hypothetical judgments:

Theorem 4.8 (Normalization). *$\Gamma \vdash A$ true if and only if $\Gamma_\downarrow \vdash A\!\uparrow$.*

To this end, we introduce two *annotated judgments* $\Gamma_\downarrow \vdash^+ A\!\downarrow$ and $\Gamma_\downarrow \vdash^+ A\!\uparrow$, for which we use the following rule in addition to those rules for $\Gamma_\downarrow \vdash A\!\downarrow$ and $\Gamma_\downarrow \vdash A\!\uparrow$:

$$\frac{\Gamma_\downarrow \vdash^+ A\!\uparrow}{\Gamma_\downarrow \vdash^+ A\!\downarrow}\ \uparrow\!\downarrow$$

As it is based on hypothetical judgments, the new system satisfies the substitution principle (which extends Theorem 2.11); we assume that the exchange rule is built-in:

Theorem 4.9 (Substitution).
If $\Gamma_\downarrow \vdash^+ A\!\downarrow$ and $\Gamma_\downarrow, A\!\downarrow \vdash^+ C\!\downarrow$, then $\Gamma_\downarrow \vdash^+ C\!\downarrow$.
If $\Gamma_\downarrow \vdash^+ A\!\downarrow$ and $\Gamma_\downarrow, A\!\downarrow \vdash^+ C\!\uparrow$, then $\Gamma_\downarrow \vdash^+ C\!\uparrow$.

In conjunction with the rule $\downarrow\!\uparrow$, the rule $\uparrow\!\downarrow$ effectively collapses the distinction between $A\!\uparrow$ and $A\!\downarrow$: a proof of $A\!\uparrow$ leads to a proof of $A\!\downarrow$ and vice versa. Thus both $A\!\uparrow$ and $A\!\downarrow$ in the new system are essentially no different from A *true*, as stated in the two theorems below; $\Gamma_\downarrow = \{A\!\downarrow \mid A \textit{ true} \in \Gamma\}$ is a collection of neutral judgments derived from truth judgments in Γ:

Theorem 4.10 (Soundness of the annotated judgments).
If $\Gamma_\downarrow \vdash^+ A\!\downarrow$, then $\Gamma \vdash A$ true.
If $\Gamma_\downarrow \vdash^+ A\!\uparrow$, then $\Gamma \vdash A$ true.

Proof. By simultaneous induction on the structure of the proof of $\Gamma_\downarrow \vdash^+ A\!\downarrow$ and $\Gamma_\downarrow \vdash^+ A\!\uparrow$. ☐

Theorem 4.11 (Completeness of the annotated judgments).
If $\Gamma \vdash A\ true$, *then* $\Gamma_\downarrow \vdash^+ A\!\downarrow$.
If $\Gamma \vdash A\ true$, *then* $\Gamma_\downarrow \vdash^+ A\!\uparrow$.

Proof. By induction on the structure of the proof of $\Gamma \vdash A\ true$. We use the rules $\downarrow\uparrow$ and $\uparrow\downarrow$ to convert between $A\!\downarrow$ and $A\!\uparrow$ whenever necessary. We show two cases.

Case $\dfrac{\Gamma, A_1\ true \vdash A_2\ true}{\Gamma \vdash A_1 \supset A_2\ true} \supset\mathsf{I}$ where $A = A_1 \supset A_2$

$\Gamma_\downarrow, A_1\!\downarrow \vdash^+ A_2\!\uparrow$ — by induction hypothesis on $\Gamma, A_1\ true \vdash A_2\ true$
$\Gamma_\downarrow \vdash^+ A_1 \supset A_2\!\uparrow$ — by the rule $\supset\mathsf{I}_\uparrow$, proving the second clause
$\Gamma_\downarrow \vdash^+ A_1 \supset A_2\!\downarrow$ — by the rule $\uparrow\downarrow$, proving the first clause

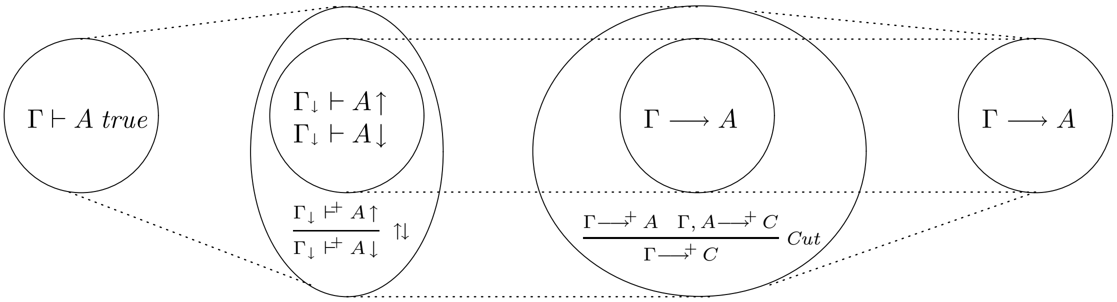
Case $\dfrac{\Gamma \vdash B \supset A\ true \quad \Gamma \vdash B\ true}{\Gamma \vdash A\ true} \supset\mathsf{E}$

$\Gamma_\downarrow \vdash^+ B \supset A\!\downarrow$ — by induction hypothesis on $\Gamma \vdash B \supset A\ true$
$\Gamma_\downarrow \vdash^+ B\!\uparrow$ — by induction hypothesis on $\Gamma \vdash B\ true$
$\Gamma_\downarrow \vdash^+ A\!\downarrow$ — by the rule $\supset\mathsf{E}_\downarrow$, proving the first clause
$\Gamma_\downarrow \vdash^+ A\!\uparrow$ — by the rule $\downarrow\uparrow$, proving the second clause

☐

Now we can complete the proof of Theorem 4.8 by showing that $\Gamma \stackrel{+}{\longrightarrow} A$ and $\Gamma_\downarrow \vdash^+ A\!\uparrow$ are equivalent (Theorems 4.12 and 4.13); we use an appropriate definition of $\Gamma_\downarrow$ depending on the definition of Γ:

$$
\begin{array}{rcll}
\Gamma_\downarrow \vdash A\!\uparrow & \iff & \Gamma \longrightarrow A & \textit{by Theorems 4.3 and 4.4} \\
& \iff & \Gamma \stackrel{+}{\longrightarrow} A & \textit{by Theorem 4.7} \\
& \iff & \Gamma_\downarrow \vdash^+ A\!\uparrow & \textit{by Theorems 4.12 and 4.13} \\
& \iff & \Gamma \vdash A\ true & \textit{by Theorems 4.10 and 4.11}
\end{array}
$$



The proof of Theorems 4.12 and 4.13 is almost the same as the proof of Theorems 4.3 and 4.4, except for the additional case in which the rule $\uparrow\downarrow$ or *Cut* is involved. The proof of of Theorem 4.13 follows from a lemma similar to Lemma 4.5.

Theorem 4.12 (Soundness of the sequent calculus with the cut rule). *If* $\Gamma \stackrel{+}{\longrightarrow} C$, *then* $\Gamma_\downarrow \vdash^+ C\!\uparrow$.

Proof. By induction on the structure of the proof of $\Gamma \stackrel{+}{\longrightarrow} C$. We show the case for the rule *Cut*.

Case $\dfrac{\Gamma \stackrel{+}{\longrightarrow} A \quad \Gamma, A \stackrel{+}{\longrightarrow} C}{\Gamma \stackrel{+}{\longrightarrow} C}\ Cut$

$\Gamma_\downarrow \vdash^+ A\!\uparrow$ — by induction hypothesis on $\Gamma \stackrel{+}{\longrightarrow} A$
$\Gamma_\downarrow \vdash^+ A\!\downarrow$ — by the rule $\uparrow\downarrow$
$\Gamma_\downarrow, A\!\downarrow \vdash^+ C\!\uparrow$ — by induction hypothesis on $\Gamma, A \stackrel{+}{\longrightarrow} C$
$\Gamma_\downarrow \vdash^+ C\!\uparrow$ — by Theorem 4.9 with $\Gamma_\downarrow \vdash^+ A\!\downarrow$ and $\Gamma_\downarrow, A\!\downarrow \vdash^+ C\!\uparrow$

☐

Theorem 4.13 (Completeness of the sequent calculus with the cut rule). *If* $\Gamma_\downarrow \vdash^+ C\!\uparrow$, *then* $\Gamma \stackrel{+}{\longrightarrow} C$.

Lemma 4.14.
If $\Gamma_\downarrow \vdash^+ A\downarrow$*, then* $\Gamma, A \overset{+}{\longrightarrow} C$ *implies* $\Gamma \overset{+}{\longrightarrow} C$*.*
If $\Gamma_\downarrow \vdash^+ C\uparrow$*, then* $\Gamma \overset{+}{\longrightarrow} C$*.*

Proof. By simultaneous induction on the structure of the proof of $\Gamma_\downarrow \vdash^+ A\downarrow$ and $\Gamma_\downarrow \vdash^+ C\uparrow$. We show the case for the rule $\uparrow\!\downarrow$.

Case $\dfrac{\Gamma_\downarrow \vdash^+ A\uparrow}{\Gamma_\downarrow \vdash^+ A\downarrow} \uparrow\!\downarrow$

$\Gamma \overset{+}{\longrightarrow} A$	by induction hypothesis on $\Gamma_\downarrow \vdash^+ A\uparrow$
$\Gamma, A \overset{+}{\longrightarrow} C$	assumption
$\Gamma \overset{+}{\longrightarrow} C$	by the rule Cut with $\Gamma \overset{+}{\longrightarrow} A$ and $\Gamma, A \overset{+}{\longrightarrow} C$

□

Implicit in the proof of Theorem 4.8 is that the proof is constructive: it gives an algorithm for converting a proof of $\Gamma \vdash A\ true$ into a proof of $\Gamma_\downarrow \vdash A\uparrow$. (Converting a proof of $\Gamma_\downarrow \vdash A\uparrow$ into a proof of $\Gamma \vdash A\ true$ is trivial.) It is a consequence of constructive proofs of all the theorems involved, in particular Theorem 4.13 (completeness of the sequent calculus with the cut rule) which is generalized to Lemma 4.14, and Theorem 4.6 (admissibility of the cut rule) which is used in the proof of Theorem 4.7 (cut elimination). To be specific, we convert a proof of $\Gamma_\downarrow \vdash A\uparrow$ into a proof of $\Gamma \vdash A\ true$ as follows:

1. $\Gamma \vdash A\ true$ to $\Gamma_\downarrow \vdash^+ A\uparrow$ by Theorem 4.11. We annotate the proof of $\Gamma \vdash A\ true$ by replacing $A\ true$ by $A\uparrow$ or $A\downarrow$, inserting the rule $\uparrow\!\downarrow$ whenever a detour is encountered.
2. $\Gamma_\downarrow \vdash^+ A\uparrow$ to $\Gamma \overset{+}{\longrightarrow} A$ by Theorem 4.13. We insert the rule Cut whenever the rule $\uparrow\!\downarrow$ is encountered.
3. $\Gamma \overset{+}{\longrightarrow} A$ to $\Gamma \longrightarrow A$ by Theorem 4.7. We use the proof of Theorem 4.6 to remove the rule Cut.
4. $\Gamma \longrightarrow A$ to $\Gamma_\downarrow \vdash A\uparrow$ by Theorem 4.3.

Thus in the heart of the proof of the normalization theorem lies the cut elimination theorem!

A corollary of the normalization theorem (or its proof) is consistency of propositional logic (or first-order logic if universal and existential quantifiers are added): $\perp\ true$ is not provable in propositional logic.

Corollary 4.15 (Consistency). *There is no proof of* $\cdot \vdash \perp\ true$*.*

Proof. It suffices to show that there is no proof of $\cdot \vdash \perp\uparrow$ (by the normalization theorem), or $\cdot \longrightarrow \perp$ (by Theorems 4.3 and 4.4). Since no rule is applicable to $\cdot \longrightarrow \perp$, there is no proof of $\cdot \longrightarrow \perp$. □

Another corollary is that $A \vee B\ true$ is provable only if either $A\ true$ or $B\ true$ is provable.

Corollary 4.16. *If* $\cdot \vdash A \vee B\ true$*, then either* $\cdot \vdash A\ true$ *or* $\cdot \vdash B\ true$*.*

Proof. $\cdot \vdash A \vee B\ true$ implies $\cdot \longrightarrow A \vee B$, as shown in the proof of the normalization theorem. Since the only way to prove $\cdot \longrightarrow A \vee B$ is by applying either $\vee R_L$ or $\vee R_R$, either $\cdot \longrightarrow A$ or $\cdot \longrightarrow B$ must hold. Therefore either $\cdot \vdash A\ true$ or $\cdot \vdash B\ true$ holds. □

Note, however, that $\Gamma \vdash A \vee B\ true$ does not necessarily imply either $\Gamma \vdash A\ true$ or $\Gamma \vdash B\ true$ if Γ is not empty. For example, $B \vee A\ true \vdash A \vee B\ true$ is provable, but neither $B \vee A\ true \vdash A\ true$ nor $B \vee A\ true \vdash B\ true$ is provable.

Finally constructive logic is shown to be different from classical logic: $A \vee \neg A\ true$, which is called *the law of excluded middle* and is an axiom in classical logic, is not provable in constructive logic.

Corollary 4.17. *There is no proof of* $\cdot \vdash A \vee \neg A\ true$ *for an arbitrary proposition* A*.*

Proof. If $\cdot \vdash A \vee \neg A\ true$ holds, then either $\cdot \longrightarrow A$ or $\cdot \longrightarrow \neg A$ holds, as shown in the proof of Corollary 4.16. The first sequent is not provable for an arbitrary proposition A. The second sequent is not provable because $A \longrightarrow \perp$ is not provable. □

Note that the law of excluded middle assumes an *arbitrary* proposition A; the use of a specific proposition A makes $A \vee \neg A\ true$ provable. For example, by letting $A = \top$, we obtain $\top \vee \neg\top\ true$, which is certainly provable.

Chapter 5

First-Order Logic

This chapter develops *first-order logic, i.e.*, logic with universal and existential quantifications. Developing first-order logic is the first step toward a practical reasoning system which inevitably demands an apparatus for expressing that a given property holds *for all, or* $\forall$, objects or that there *exists, or* $\exists$, a certain object satisfying a given property. Here we deal with pure first-order logic which does not stipulate a particular class of objects. Later we will enrich it in such a way that we can express properties of specific classes of objects such as natural numbers, trees, or boolean values.

5.1 Terms

In propositional logic, expressing properties of objects under consideration requires us to define propositional constants which denote atomic propositions. For example, in order to express that 1 is equal to 1 itself, we would need a propositional constant Eq_1 denoting an atomic proposition *'1 is equal to 1.'* While logical connectives provide us with an elegant mechanism for reasoning about such atomic propositions, the need for a separate propositional constant for each atomic proposition makes propositional logic too limited in its expressive power. For example, in order to express that every natural number is equal to itself, we would have to define an infinite array of propositional constants Eq_i denoting *'i is equal to i.'*

First-order logic replaces propositional constants in propositional logic by *predicates*. A predicate may have arguments and expresses a relation between its arguments. (For this reason, first-order logic is also called *predicate logic*.) For example, we can define a predicate Eq so that $Eq(t_1, t_2)$ denotes a proposition *'t_1 is equal to t_2.'* Here the predicate Eq has two arguments t_1 and t_2 and expresses an equality between t_1 and t_2. The arguments t_1 and t_2 are called *terms* in first-order logic and may be interpreted as particular mathematical objects (such as natural numbers). Thus first-order logic is a system in which we use predicates to express properties of terms.

Note that first-order logic itself does not enforce a specific way of interpreting terms. As an example, consider two terms $\mathbf{0}$ and $\mathsf{s}(\mathbf{0})$. As usual, we could interpret $\mathbf{0}$ as zero and $\mathsf{s}(\mathbf{0})$ as the successor of zero, but such an interpretation is just a specific way of assigning mathematical objects to terms. Thus it is also fine to interpret $\mathbf{0}$ as the natural number one and or $\mathsf{s}(\mathbf{0})$ as the predecessor of one. In general, we do not formalize how to relate terms to mathematical objects, and first-order logic in our discussion (which is based on proof theory) deals only with terms and not with their interpretations. Thus predicates directly express properties of uninterpreted terms.

Formally we define terms as follows:

$$\text{term} \quad t, s \quad ::= \quad x \mid y \mid \cdots \mid a \mid b \mid \cdots \mid f(t_1, \cdots, t_n) \mid c$$

$x, y, \cdots$ are called *term variables* which range over the set of all terms. We may substitute terms for term variables and we write $[s/x]t$ for the result of substituting s for x in t. $a, b, \cdots$ are called *parameters* and denote arbitrary/unspecified terms about which we can make no assumption. The difference between term variables and parameters is that a term variable is just a placeholder for another term whereas a parameter is understood as an arbitrary term about which nothing is known. (We will see the use of parameters in inference rules for first-order logic.)

f is called a *function symbol* and has zero or more arguments. We write $f(t_1, \cdots, t_n)$ for a term where f is a function symbol of arity n and $t_1, \cdots, t_n$ are its arguments. A *constant* c is a function symbol of zero arity; that is, c is an abbreviation of $c()$. Note that although it is usually interpreted as a function in the mathematical sense, a function symbol f is *not* a function because $f(t_1, \cdots, t_n)$ is a term in itself and does not reduce to another term. For example, $\mathbf{s}(\mathbf{0})$, which comprises of a function symbol $\mathbf{s}$ and its argument $\mathbf{0}$, does not reduce to another term, say $\mathbf{1}$, because it is a term in itself.

Now we can define a set of terms by specifying function symbols with their arities. Here are a few examples:

- To obtain terms for natural numbers, we use a constant $\mathbf{0}$ for zero and a function symbol $\mathbf{s}$ of arity one to be interpreted as the successor function.
- To obtain terms for boolean values, we use two constants $\mathbf{true}$ and $\mathbf{false}$.
- To obtain terms for binary trees, we use a constant $\mathbf{leaf}$ for leaf nodes and a function symbol $\mathbf{node}$ of arity two for inner nodes.

Terms are not to be confused with proof terms. Terms can represent any kinds of objects (*e.g.*, natural numbers, boolean values, student names, *etc.*) whereas proof terms represent proofs in logic. For example, we can say that a proof term $\lambda x\!:\!A.\, x$ represents a proof of $A \supset A$, but it makes no sense to judge the truth or falsehood of a term $\mathbf{s}(\mathbf{0})$.

5.2 Propositions in first-order logic

In addition to logical connectives from propositional logic, first-order logic uses predicates and two forms of quantifications over terms. An inductive definition of propositions is given as follows:

$$\text{proposition} \quad A \quad ::= \quad P(t_1, \cdots, t_n) \mid \cdots \mid \forall x.A \mid \exists x.A$$

Alternatively we may use three new formation rules:

$$\frac{}{P(t_1, \cdots, t_n)\ prop}\ P\mathsf{F} \qquad \frac{A\ prop}{\forall x.A\ prop}\ \forall\mathsf{F} \qquad \frac{A\ prop}{\exists x.A\ prop}\ \exists\mathsf{F}$$

P is called a *predicate symbol*. A predicate $P(t_1, \cdots, t_n)$ is a proposition that expresses a certain relation between terms $t_1, \cdots, t_n$. For example, we may use $Nat(t)$ to mean that term t is a natural number, or $Eq(t_1, t_2)$ to mean that terms t_1 and t_2 are equal. A propositional constant P is a predicate symbol of zero arity; that is, P is an abbreviation of $P()$.

$\forall x.A$ uses a *universal quantifier* $\forall$ to introduce a term variable x. Roughly speaking, the truth of $\forall x.A$ means that A is true for "every" term x. $\exists x.A$ uses an *existential quantifier* $\exists$ to introduce a term variable x. Roughly speaking, the truth of $\exists x.A$ means that we can present "some" term x for which A is true. Quantifiers $\forall$ and $\exists$ have the lowest operator precedence. For example, $\forall x.A \supset B$ is understood as $\forall x.(A \supset B)$; similarly $\exists x.A \supset B$ is understood as $\exists x.(A \supset B)$.

As quantifiers introduce term variables, there arises a need for substitutions for term variables in propositions or proofs. We write $[t/x]A$ for the result of substituting t for x in proposition A. Similarly we write $[t/x]\mathcal{D}$ for the result of substituting t for x throughout proof $\mathcal{D}$. Extending substitutions for term variables, we write $[t/a]A$ and $[t/a]\mathcal{D}$ for the result of substituting t for parameter a in A and $\mathcal{D}$, respectively. These substitutions for term variables and parameters are considerably simpler to define than substitutions in the simply-typed λ-calculus because variable captures never occur in first-order logic. That is, in a substitution $[t/x]A$ or $[t/x]\mathcal{D}$, term t is always closed and contains no free term variables.

5.3 Universal quantification

A universal quantification $\forall x.A$ is true if A is true for every term x. For example, given that $\mathbf{0}$, $\mathbf{s}(\mathbf{0})$, $\mathbf{s}(\mathbf{s}(\mathbf{0})), \cdots$ constitute the set of terms, we can deduce $\forall x.Eq(x, x)\ true$ if $Eq(\mathbf{0}, \mathbf{0})\ true$, $Eq(\mathbf{s}(\mathbf{0}), \mathbf{s}(\mathbf{0}))\ true$, $Eq(\mathbf{s}(\mathbf{s}(\mathbf{0})), \mathbf{s}(\mathbf{s}(\mathbf{0})))\ true$, $\cdots$ are all provable. Hence it helps to think of $\forall x.A$ as an infinite conjunction

$$[t_1/x]A \wedge [t_2/x]A \wedge \cdots \wedge [t_i/x]A \wedge \cdots$$

where $t_1, t_2, \cdots, t_i, \cdots$ enumerate all terms.

The inference rules for universal quantifications are given as follows:

$$\frac{[a/x]A\ \mathit{true}}{\forall x.A\ \mathit{true}}\ \forall\mathsf{I}^a \qquad \frac{\forall x.A\ \mathit{true}}{[t/x]A\ \mathit{true}}\ \forall\mathsf{E}$$

In the rule $\forall\mathsf{I}^a$, parameter a denotes an arbitrary term about which we can make no assumption. Thus we may read $[a/x]A\ \mathit{true}$ as a shorthand for a sequence of judgments

$$[t_1/x]A\ \mathit{true} \quad [t_2/x]A\ \mathit{true} \quad \cdots \quad [t_i/x]A\ \mathit{true} \quad \cdots$$

where $t_1, t_2, \cdots, t_i, \cdots$ enumerate all terms. In the rule $\forall\mathsf{E}$, t can be any term — a constant, a function symbol, a term variable, or even an existing parameter. We justify the rule $\forall\mathsf{E}$ by reading $\forall x.A\ \mathit{true}$ as

$$[t_1/x]A \wedge [t_2/x]A \wedge \cdots \wedge [t_i/x]A \wedge \cdots\ \mathit{true}$$

where $t_1, t_2, \cdots, t_i, \cdots$ enumerate all terms.

It is important that in the rule $\forall\mathsf{I}^a$, parameter a must be fresh and not found in any undischarged hypothesis. For example, a proof of $\forall x.Nat(x)\ \mathit{true}$ introducing a fresh parameter a must not contain any hypothesis of the form $\overline{P(a)}$, which is an assumption on an arbitrary term about which we can make no assumption! The presence of such a hypothesis implies that parameter a is already declared elsewhere and thus cannot be interpreted as an arbitrary term. The following example, which tries to prove that y is a natural number whenever x is a natural number, shows that using the same parameter twice in difference instances of the rule $\forall\mathsf{I}$ results in a wrong proof:

$$\cfrac{\cfrac{\cfrac{\cfrac{\cfrac{\overline{Nat(a)\ \mathit{true}}^{\,w}}{\forall x.Nat(x)\ \mathit{true}}\ \forall\mathsf{I}^a\ \textit{(wrong)}}{Nat(b)\ \mathit{true}}\ \forall\mathsf{E}}{Nat(a) \supset Nat(b)\ \mathit{true}}\ \supset\mathsf{I}^w}{\forall y.Nat(a) \supset Nat(y)\ \mathit{true}}\ \forall\mathsf{I}^b}{\forall x.\forall y.Nat(x) \supset Nat(y)\ \mathit{true}}\ \forall\mathsf{I}^a$$

Here is an example of a proof involving universal quantifiers where we exploit $[a/x](A \wedge B) = [a/x]A \wedge [a/x]B$.

$$\cfrac{\cfrac{\cfrac{\cfrac{\cfrac{\overline{\forall x.A \wedge B\ \mathit{true}}^{\,w}}{[a/x](A \wedge B)\ \mathit{true}}\ \forall\mathsf{E}}{[a/x]A\ \mathit{true}}\ \wedge\mathsf{E_L}}{\forall x.A\ \mathit{true}}\ \forall\mathsf{I}^a \quad \cfrac{\cfrac{\cfrac{\overline{\forall x.A \wedge B\ \mathit{true}}^{\,w}}{[a/x](A \wedge B)\ \mathit{true}}\ \forall\mathsf{E}}{[a/x]B\ \mathit{true}}\ \wedge\mathsf{E_R}}{\forall x.B\ \mathit{true}}\ \forall\mathsf{I}^a}{(\forall x.A) \wedge (\forall x.B)\ \mathit{true}}\ \wedge\mathsf{I}}{(\forall x.A \wedge B) \supset (\forall x.A) \wedge (\forall x.B)\ \mathit{true}}\ \supset\mathsf{I}^w$$

5.4 Existential quantification

An existential quantification $\exists x.A$ is true if there exists a term x satisfying A. For example, if $Eq(\mathbf{0}, \mathbf{0})\ \mathit{true}$ is provable, we can deduce $\exists x.Eq(x, x)$ because substituting a concrete term $\mathbf{0}$ for x makes $Eq(x, x)\ \mathit{true}$ provable. Hence it helps to think of $\exists x.A$ as an infinite disjunction

$$[t_1/x]A \vee [t_2/x]A \vee \cdots \wedge [t_i/x]A \vee \cdots$$

where $t_1, t_2, \cdots, t_i, \cdots$ enumerate all terms.

The inference rules for existential quantifications are given as follows:

$$\frac{[t/x]A\ \mathit{true}}{\exists x.A\ \mathit{true}}\ \exists\mathsf{I} \qquad \frac{\exists x.A\ \mathit{true} \qquad \begin{array}{c}\overline{[a/x]A\ \mathit{true}}^{\,w}\\ \vdots \\ C\ \mathit{true}\end{array}}{C\ \mathit{true}}\ \exists\mathsf{E}^{a,w}$$

The rule $\exists\mathsf{I}$ says that we prove $\exists x.A\ true$ by presenting a concrete term, or a *witness*, t such that $[t/x]A\ true$ is provable. We justify the rule $\exists\mathsf{I}$ by reading $\exists x.A\ true$ as

$$[t_1/x]A \lor [t_2/x]A \lor \cdots \lor [t_i/x]A \lor \cdots\ true$$

where $t_1,\ t_2,\ \cdots,\ t_i,\ \cdots$ enumerate all terms and $t_i = t$ holds. In the rule $\exists\mathsf{E}^{a,w}$, we annotate the hypothesis $\overline{[a/x]A\ true}$ with label w. We also introduce a fresh parameter a because the witness for the proof of $\exists x.A\ true$ is unknown and thus we cannot make any assumption about it. Thus we may read $\begin{array}{c}\overline{[a/x]A\ true}^{w}\\ \vdots \\ C\ true\end{array}$ as a shorthand for a sequence of hypothetical proofs

$$\begin{array}{c}\overline{[t_1/x]A\ true}^{w}\\ \vdots \\ C\ true\end{array} \qquad \begin{array}{c}\overline{[t_2/x]A\ true}^{w}\\ \vdots \\ C\ true\end{array} \qquad \cdots \qquad \begin{array}{c}\overline{[t_i/x]A\ true}^{w}\\ \vdots \\ C\ true\end{array} \qquad \cdots$$

where $t_1, t_2, \cdots, t_i, \cdots$ enumerate all terms.

In the rule $\exists\mathsf{E}^{a,w}$, parameter a must be fresh and not found in proposition A or any undischarged hypothesis. In particular, it must not be found in proposition C. Otherwise the rule ends up with a conclusion that makes too strong an assumption about the witness, namely that the witness can be an arbitrary term! For example, the following proof exploits a proof of $\exists x.Nat(x) \land Eq(x, \mathbf{0})\ true$ to draw a (nonsensical) conclusion that an arbitrary term is equal to a natural number $\mathbf{0}$, as it allows parameter a to appear in the conclusion:

$$\dfrac{\begin{array}{c}\vdots\\ \exists x.Nat(x)\land Eq(x,\mathbf{0})\ true\end{array} \qquad \dfrac{\overline{Nat(a)\land Eq(a,\mathbf{0})\ true}^{w}}{Eq(a,\mathbf{0})\ true}\,\land\mathsf{E_R}}{Eq(a,\mathbf{0})\ true}\;\exists\mathsf{E}^{a,w}$$

In essence, the rule $\exists\mathsf{E}^{a,w}$ introduces parameter a in the course of proving $C\ true$ after fixing proposition C, which implies that C is oblivious to a.

An important aspect of the rule $\exists\mathsf{I}$ is that in order to prove $\exists x.A\ true$, it is not enough to show that there only "exists" a witness x satisfying A without actually knowing what it is. The necessity of such a witness is indeed a distinguishing feature of constructive logic. In contrast, a proof of $\exists x.A\ true$ in classical logic only needs to show that there exists a term t, *which may or may not be known,* such that $[t/x]A\ true$ is provable. In other words, a proof of $\exists x.A\ true$ essentially shows that it cannot happen that there exists no term t such that $[t/x]A\ true$ is provable. As a consequence, $\exists x.A$ is no different from $\neg\forall x.\neg A$ in classical logic.

To better understand the nature of existential quantifications in constructive logic, let us consider a few examples. First $\exists x.\neg A \supset \neg\forall x.A\ true$ is provable. Intuitively a proof of $\exists x.\neg A\ true$ gives us a witness t such that $[t/x]\neg A\ true$ is provable, and we can use t to refute $\forall x.A\ true$.

$$\dfrac{\dfrac{\dfrac{\overline{\exists x.\neg A\ true}^{w} \qquad \dfrac{\overline{[a/x]\neg A\ true}^{y} \qquad \dfrac{\overline{\forall x.A\ true}^{z}}{[a/x]A\ true}\,\forall\mathsf{E}}{\bot\ true}\,\neg\mathsf{E}}{\bot\ true}\;\exists\mathsf{E}^{a,y}}{\neg\forall x.A\ true}\;\neg\mathsf{I}^{z}}{\exists x.\neg A \supset \neg\forall x.A\ true}\;\supset\mathsf{I}^{w}$$

The converse $\neg\forall x.A \supset \exists x.\neg A\ true$ is not provable, however. Intuitively a proof of $\exists x.\neg A\ true$ requires a witness t such that $[t/x]\neg A\ true$ is provable, but no proof of $\neg\forall x.A\ true$ gives such a witness.

$$\dfrac{\dfrac{\dfrac{\overline{\neg\forall x.A\ true}^{w} \qquad \begin{array}{c}?\\ \forall x.A\ true\end{array}}{\bot\ true}\,\neg\mathsf{E}}{\exists x.\neg A\ true}\,\bot\mathsf{E}}{\neg\forall x.A \supset \exists x.\neg A\ true}\;\supset\mathsf{I}^{w}$$

$$
\frac{[a/x]A\ \mathit{true}}{\forall x.A\ \mathit{true}}\ \forall\mathsf{I}^a
\qquad
\frac{\forall x.A\ \mathit{true}}{[t/x]A\ \mathit{true}}\ \forall\mathsf{E}
\qquad
\frac{[t/x]A\ \mathit{true}}{\exists x.A\ \mathit{true}}\ \exists\mathsf{I}
\qquad
\frac{\exists x.A\ \mathit{true} \qquad \begin{array}{c}\overline{[a/x]A\ \mathit{true}}^{\,w}\\ \vdots\\ C\ \mathit{true}\end{array}}{C\ \mathit{true}}\ \exists\mathsf{E}^{a,w}
$$

Figure 5.1: Natural deduction system for first-order logic

Perhaps surprisingly, $(\forall x.A) \supset (\exists x.A)\ \mathit{true}$ is *not* provable. The reason is that although $\forall x.A\ \mathit{true}$ states that $[t/x]A\ \mathit{true}$ is provable for any term t, it does not decide a concrete term t such that $[t/x]A\ \mathit{true}$ is provable. In particular, if the set of terms is empty, $\forall x.A\ \mathit{true}$ holds trivially (because there is no term), but $\exists x.A\ \mathit{true}$ never holds because it is impossible to choose a term t for x, regardless of proposition A.

$$
\frac{\dfrac{\dfrac{\overline{\forall x.A\ \mathit{true}}^{\,w}}{[t/x]A\ \mathit{true}?}\ \forall\mathsf{E}}{\exists x.A\ \mathit{true}}\ \exists\mathsf{I}}{(\forall x.A) \supset (\exists x.A)\ \mathit{true}}\ \supset\mathsf{I}^w
$$

On the other hand, $\forall y.(\forall x.A) \supset (\exists x.A)\ \mathit{true}$ *is* provable even if y does not occur free in A. The difference from the previous example is that $\forall y$ allows us to make an assumption that the set of terms is not empty. In the proof shown below, parameter a denotes an arbitrary term in the set of terms, and its presence implies that the set of terms is not empty.

$$
\frac{\dfrac{\dfrac{\dfrac{\overline{\forall x.A\ \mathit{true}}^{\,w}}{[a/x]A\ \mathit{true}}\ \forall\mathsf{E}}{\exists x.A\ \mathit{true}}\ \exists\mathsf{I}}{(\forall x.A) \supset (\exists x.A)\ \mathit{true}}\ \supset\mathsf{I}^w}{\forall y.(\forall x.A) \supset (\exists x.A)\ \mathit{true}}\ \forall\mathsf{I}^a
$$

These two examples illustrate that in constructive logic, $\forall x.A$ is not equivalent to A even if x does not occur free in A at all: $\forall x.A$ asserts A on the assumption that the set of terms is not empty, whereas A without a universal quantifier cannot exploit such an assumption.

Figure 5.1 shows the inference rules for first-order logic.

Exercise 5.1. We have seen that a logical equivalence $\neg\forall x.A \equiv \exists x.\neg A$ fails. Check whether the following logical equivalence holds or not:

$$\neg\exists x.A \equiv \forall x.\neg A$$

Exercise 5.2. Suppose that term variable x is not free in proposition A, but free in proposition B. That is, we have $[t/x]A = A$, but $[t/x]B \neq B$ in general for an arbitrary term t. Check if each of the following judgments is provable.

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

5.5 Local soundness and completeness

To show the local soundness and completeness properties of first-order logic, we consider local reductions and expansions for universal and existential quantifications. A local reduction for universal

$$\frac{[a/x]A\!\uparrow}{\forall x.A\!\uparrow}\;\forall\mathsf{I}^a_\uparrow \qquad \frac{\forall x.A\!\downarrow}{[t/x]A\!\downarrow}\;\forall\mathsf{E}_\downarrow \qquad \frac{[t/x]A\!\uparrow}{\exists x.A\!\uparrow}\;\exists\mathsf{I}_\uparrow \qquad \frac{\exists x.A\!\downarrow \quad \begin{array}{c}\overline{[a/x]A\!\downarrow}^{\,w}\\ \vdots \\ C\!\uparrow\end{array}}{C\!\uparrow}\;\exists\mathsf{E}^{a,w}_\uparrow$$

Figure 5.2: Natural deduction system for first-order logic

quantification is given as follows:

$$\dfrac{\dfrac{\begin{array}{c}\mathcal{D}\\ [a/x]A\ \mathit{true}\end{array}}{\forall x.A\ \mathit{true}}\;\forall\mathsf{I}^a}{[t/x]A\ \mathit{true}}\;\forall\mathsf{E} \quad \Longrightarrow_R \quad \begin{array}{c}[t/a]\mathcal{D}\\ [t/x]A\ \mathit{true}\end{array}$$

Since $\mathcal{D}$ proves $[a/x]A\ \mathit{true}$, we use $[t/a]\mathcal{D}$ to prove $[t/a][a/x]A\ \mathit{true} = [t/x]A\ \mathit{true}$. Similarly a local reduction for existential quantification shown below substitutes term t for parameter a:

$$\dfrac{\dfrac{\begin{array}{c}\mathcal{D}\\ [t/x]A\ \mathit{true}\end{array}}{\exists x.A\ \mathit{true}}\;\exists\mathsf{I} \qquad \left.\begin{array}{c}\overline{[a/x]A\ \mathit{true}}^{\,w}\\ \vdots\\ C\ \mathit{true}\end{array}\right\}\mathcal{E}}{C\ \mathit{true}}\;\exists\mathsf{E}^{a,w} \quad \Longrightarrow_R \quad \begin{array}{c}\mathcal{D}\\ [t/x]A\ \mathit{true}\end{array}\left.\begin{array}{c}\\ \vdots\\ C\ \mathit{true}\end{array}\right\}[t/a]\mathcal{E}$$

Note that $[t/a]\mathcal{E}$ proves the same judgment that $\mathcal{E}$ proves, namely $C\ \mathit{true}$, because parameter a does not appear in proposition C. On the other hand, $[t/a]\mathcal{E}$ changes $\overline{[a/x]A\ \mathit{true}}^{\,w}$ to $\overline{[t/a][a/x]A\ \mathit{true}}^{\,w}$, or $\overline{[t/x]A\ \mathit{true}}^{\,w}$, for which $\mathcal{D}$ is substituted. Local expansions for universal and existential quantifications are given as follows:

$$\begin{array}{c}\mathcal{E}\\ \forall x.A\ \mathit{true}\end{array} \;\Longrightarrow_E\; \dfrac{\dfrac{\begin{array}{c}\mathcal{E}\\ \forall x.A\ \mathit{true}\end{array}}{[a/x]A\ \mathit{true}}\;\forall\mathsf{E}}{\forall x.A\ \mathit{true}}\;\forall\mathsf{I}^a \qquad\qquad \begin{array}{c}\mathcal{E}\\ \exists x.A\ \mathit{true}\end{array} \;\Longrightarrow_E\; \dfrac{\begin{array}{c}\mathcal{E}\\ \exists x.A\ \mathit{true}\end{array} \quad \dfrac{\overline{[a/x]A\ \mathit{true}}^{\,w}}{\exists x.A\ \mathit{true}}\;\exists\mathsf{I}}{\exists x.A\ \mathit{true}}\;\exists\mathsf{E}^{a,w}$$

Reading $\forall x.A$ as an infinite conjunction and $\exists x.A$ as an infinite disjunction gives the rules for deducing neutral and normal judgments in Figure 5.2. It turns out that Theorem 2.12 continues to hold in first-order logic, and proving $A\ \mathit{true}$ reduces to proving $A\!\uparrow$ as in propositional logic. Theorems 2.13 (normalization) and 2.14 (strong normalization) also continue to hold, provided that the following commuting conversion for existential quantification is available where the rule R is assumed to be an elimination rule:

$$\dfrac{\dfrac{\begin{array}{c}\mathcal{D}\\ \exists x.A\ \mathit{true}\end{array} \qquad \begin{array}{c}\overline{[a/x]A\ \mathit{true}}^{\,w}\\ \vdots\\ C\ \mathit{true}\end{array}}{C\ \mathit{true}}\;\exists\mathsf{E}^{a,w}}{C'\ \mathit{true}}\;R \quad \Longrightarrow_C \quad \dfrac{\begin{array}{c}\mathcal{D}\\ \exists x.A\ \mathit{true}\end{array} \qquad \dfrac{\begin{array}{c}\overline{[a/x]A\ \mathit{true}}^{\,w}\\ \vdots\\ C\ \mathit{true}\end{array}}{C'\ \mathit{true}}\;R}{C\ \mathit{true}}\;\exists\mathsf{E}^{a,w}$$

We can derive the above commuting conversion from the commuting conversion for $\vee$ by reading $\exists x.A$ as an infinite disjunction.

5.6 Examples

As a concrete example of reasoning in first-order logic, let us characterize natural numbers. We use $\mathbf{0}$ as a term denoting zero and s as a function symbol denoting the successor function. We also use three predicates: $Nat(t)$ to mean that t is a natural number, $Eq(t, t')$ to mean that t and t' are equal, and $Lt(t, t')$ to mean that t is less than t'.

First we need axioms as a means of defining the three predicates:

$$\dfrac{}{Nat(\mathbf{0})\ \mathit{true}}\ \mathit{Zero} \qquad \dfrac{}{\forall x.Nat(x) \supset Nat(\mathbf{s}(x))\ \mathit{true}}\ \mathit{Succ}$$

$$\dfrac{}{\forall x.Eq(x,x)\ \mathit{true}}\ Eq_i \qquad \dfrac{}{\forall x.\forall y.\forall z.(Eq(x,y) \land Eq(x,z)) \supset Eq(y,z)\ \mathit{true}}\ Eq_t$$

$$\dfrac{}{\forall x.Lt(x,\mathbf{s}(x))\ \mathit{true}}\ Lt_s \qquad \dfrac{}{\forall x.\forall y.Eq(x,y) \supset \neg Lt(x,y)\ \mathit{true}}\ Lt_\neg$$

The lower four axioms may be thought of as translations of the following mathematical properties:

- $x = x$.
- If $x = y$ and $x = z$, then $y = z$.
- $x < x + 1$.
- If $x = y$, then $x \not< y$.

Combined with these axioms, first-order logic allows us to prove new theorems about these predicates. As a trivial example, here is a proof of $Nat(\mathbf{s}(\mathbf{s}(\mathbf{0})))\ \mathit{true}$, which states that $\mathbf{s}(\mathbf{s}(\mathbf{0}))$ is a natural number:

$$\dfrac{\dfrac{\dfrac{}{\forall x.Nat(x) \supset Nat(\mathbf{s}(x))\ \mathit{true}}\ \mathit{Succ}}{Nat(\mathbf{s}(\mathbf{0})) \supset Nat(\mathbf{s}(\mathbf{s}(\mathbf{0})))\ \mathit{true}}\ \forall\mathsf{E} \quad \dfrac{\dfrac{\dfrac{}{\forall x.Nat(x) \supset Nat(\mathbf{s}(x))\ \mathit{true}}\ \mathit{Succ}}{Nat(\mathbf{0}) \supset Nat(\mathbf{s}(\mathbf{0}))\ \mathit{true}}\ \forall\mathsf{E} \quad \dfrac{}{Nat(\mathbf{0})\ \mathit{true}}\ \mathit{Zero}}{Nat(\mathbf{s}(\mathbf{0}))\ \mathit{true}}\ {\supset}\mathsf{E}}{Nat(\mathbf{s}(\mathbf{s}(\mathbf{0})))\ \mathit{true}}\ {\supset}\mathsf{E}$$

Note that the two applications of the rule $\supset$E substitute different terms, namely $\mathbf{s}(\mathbf{0})$ and $\mathbf{0}$, for term variable x in $\forall x.Nat(x) \supset Nat(\mathbf{s}(x))$.

An example of using an existential quantification is a proof of $\forall x.Nat(x) \supset (\exists y.Nat(y) \land Eq(x,y))\ \mathit{true}$ which states that if x is a natural number, there exists a natural number y such that $x = y$:

$$\dfrac{\dfrac{\dfrac{\dfrac{\dfrac{}{Nat(a)\ \mathit{true}}\ z \quad \dfrac{\dfrac{}{\forall x.Eq(x,x)\ \mathit{true}}\ Eq_i}{Eq(a,a)\ \mathit{true}}\ \forall\mathsf{E}}{Nat(a) \land Eq(a,a)\ \mathit{true}}\ \land\mathsf{I}}{\exists y.Nat(y) \land Eq(a,y)\ \mathit{true}}\ \exists\mathsf{I}}{Nat(a) \supset (\exists y.Nat(y) \land Eq(a,y))\ \mathit{true}}\ {\supset}\mathsf{I}^z}{\forall x.Nat(x) \supset (\exists y.Nat(y) \land Eq(x,y))\ \mathit{true}}\ \forall\mathsf{I}^a$$

In the application of the rule $\exists$I, we use parameter a as a witness.

Here are two more examples. The first states the commutativity of equality: $x = y$ implies $y = x$ The second states that there is no term x such that $x = 0$ and $x = 1$.

- Proof of $\forall x.\forall y.Eq(x,y) \supset Eq(y,x)\ \mathit{true}$:

$$\dfrac{\dfrac{\dfrac{\dfrac{\dfrac{\dfrac{\dfrac{\dfrac{}{\forall x.\forall y.\forall z.(Eq(x,y) \land Eq(x,z)) \supset Eq(y,z)\ \mathit{true}}\ Eq_t}{\forall y.\forall z.(Eq(a,y) \land Eq(a,z)) \supset Eq(y,z)\ \mathit{true}}\ \forall\mathsf{E}}{\forall z.(Eq(a,b) \land Eq(a,z)) \supset Eq(b,z)\ \mathit{true}}\ \forall\mathsf{E}}{(Eq(a,b) \land Eq(a,a)) \supset Eq(b,a)\ \mathit{true}}\ \forall\mathsf{E} \quad \dfrac{\dfrac{}{Eq(a,b)\ \mathit{true}}\ w \quad \dfrac{\dfrac{}{\forall x.Eq(x,x)\ \mathit{true}}\ Eq_i}{Eq(a,a)\ \mathit{true}}\ \forall\mathsf{E}}{Eq(a,b) \land Eq(a,a)\ \mathit{true}}\ \land\mathsf{I}}{Eq(b,a)\ \mathit{true}}\ {\supset}\mathsf{E}}{Eq(a,b) \supset Eq(b,a)\ \mathit{true}}\ {\supset}\mathsf{I}^w}{\forall y.Eq(a,y) \supset Eq(y,a)\ \mathit{true}}\ \forall\mathsf{I}^b}{\forall x.\forall y.Eq(x,y) \supset Eq(y,x)\ \mathit{true}}\ \forall\mathsf{I}^a$$

- Proof of $\neg\exists x.Eq(x,\mathbf{0}) \wedge Eq(x,\mathbf{s}(\mathbf{0}))\ true$:

$$
\dfrac{
\dfrac{}{\exists x.Eq(x,\mathbf{0}) \wedge Eq(x,\mathbf{s}(\mathbf{0}))\ true}\,w
\qquad
\dfrac{
\dfrac{
\dfrac{
\dfrac{
\dfrac{}{\forall x.\forall y.Eq(x,y) \supset \neg Lt(x,y)\ true}\,Lt_{\neg}
}{\forall y.Eq(\mathbf{0},y) \supset \neg Lt(\mathbf{0},y)\ true}\,\forall\mathsf{E}
}{Eq(\mathbf{0},\mathbf{s}(\mathbf{0})) \supset \neg Lt(\mathbf{0},\mathbf{s}(\mathbf{0}))\ true}\,\forall\mathsf{E}
\qquad
\begin{array}{c}\mathcal{D}\\ Eq(\mathbf{0},\mathbf{s}(\mathbf{0}))\ true\end{array}
}{\neg Lt(\mathbf{0},\mathbf{s}(\mathbf{0}))\ true}\,{\supset}\mathsf{E}
\qquad
\dfrac{
\dfrac{}{\forall x.Lt(x,\mathbf{s}(x))\ true}\,Lt_s
}{Lt(\mathbf{0},\mathbf{s}(\mathbf{0}))\ true}\,\forall\mathsf{E}
}{\bot\ true}\,\neg\mathsf{E}
}{
\dfrac{\bot\ true}{\neg\exists x.Eq(x,\mathbf{0}) \wedge Eq(x,\mathbf{s}(\mathbf{0}))\ true}\,\neg\mathsf{I}^w
}\,\exists\mathsf{E}^{a,z}
$$

where we let

$$
\mathcal{D} \quad = \quad
\dfrac{
\dfrac{
\dfrac{
\dfrac{
\dfrac{}{\forall x.\forall y.\forall z.(Eq(x,y) \wedge Eq(x,z)) \supset Eq(y,z)\ true}\,Eq_t
}{\forall y.\forall z.(Eq(a,y) \wedge Eq(a,z)) \supset Eq(y,z)\ true}\,\forall\mathsf{E}
}{\forall z.(Eq(a,\mathbf{0}) \wedge Eq(a,z)) \supset Eq(\mathbf{0},z)\ true}\,\forall\mathsf{E}
}{(Eq(a,\mathbf{0}) \wedge Eq(a,\mathbf{s}(\mathbf{0}))) \supset Eq(\mathbf{0},\mathbf{s}(\mathbf{0}))\ true}\,\forall\mathsf{E}
\qquad
\dfrac{}{Eq(a,\mathbf{0}) \wedge Eq(a,\mathbf{s}(\mathbf{0}))\ true}\,z
}{Eq(\mathbf{0},\mathbf{s}(\mathbf{0}))\ true}\,{\supset}\mathsf{E}
$$

5.7 Proof terms

As in propositional logic, we use the Curry-Howard isomorphism to represent proofs of truth judgments as proof terms. Proof terms for first-order logic are given as follows:

$$\text{proof term} \quad M \quad ::= \quad \cdots \mid \lambda x.\, M \mid M\ t \mid \langle t, M\rangle \mid \mathsf{let}\ \langle x, w\rangle = M\ \mathsf{in}\ M$$

$\lambda x.\, M$, a proof term of type $\forall x.A$, is a λ-abstraction that takes a term t and returns a proof term of type $[t/x]A$. (Recall that propositions and types are equivalent under the Curry-Howard isomorphism.) It is similar to a λ-abstraction from propositional logic except that it takes a term, instead of a proof term, as its argument. For example, given a term t (denoting a natural number), $\lambda x.\, M$ may return a proof term of type $Nat(t) \supset Nat(\mathbf{s}(t))$. A corresponding λ-application $M\ t$ is a proof term of type $[t/x]A$ if M is a proof term of type $\forall x.A$.

The typing rules for $\lambda x.\, M$ and $M\ t$ are given as follows:

$$\dfrac{[a/x]M : [a/x]A}{\lambda x.\, M : \forall x.A}\,\forall\mathsf{I}^a \qquad \dfrac{M : \forall x.A}{M\ t : [t/x]A}\,\forall\mathsf{E}$$

Here we write $[t/x]M$ for a substitution of term t for term variable x in proof term M. The rule $\forall\mathsf{I}^a$ proves that $\lambda x.\, M$ has type $\forall x.A$ by introducing a fresh parameter a and proving that $[a/x]M$ has type $[a/x]A$. For example, a proof that $\lambda x.\, M$ has type $\forall x.Eq(x,x)$ could show that $[a/x]M$ has type $Eq(a,a)$ for some parameter a. Note that in the rule $\forall\mathsf{I}^a$, term a appears in both proof term M and type A. This feature of first-order logic is manifested in the rule $\forall\mathsf{E}$: terms may appear not only in types but also in proof terms. Intuitively a proof about a specific term t needs to mention t somewhere in it. (Otherwise how can we prove a fact about t at all?) Hence a proof term whose type contains t also mentions t somewhere in it. For example, we could use $\mathsf{Eq_i}\ \mathbf{0}$ as a proof term of type $Eq(\mathbf{0},\mathbf{0})$ where $\mathsf{Eq_i}$ assumes type $\forall x.Eq(x,x)$. As a consequence, a substitution $[t/x]M$ on proof term M may need a substitution $[t/x]A$ on type A if x occurs inside A in M.

$\langle t, M\rangle$ is a proof term of type $\exists x.A$. Intuitively a proof of $\exists x.A\ true$ requires a concrete witness t and a proof that t satisfies A. Hence a proof term of type $\exists x.A$ contains such a witness t and a proof term M of type $[t/x]A$. For example, a proof term of type $\exists x.Eq(x,x)$ ("there exists a term x such that x is equal to x itself") may contain a witness $\mathbf{0}$ and a proof term of type $Eq(\mathbf{0},\mathbf{0})$ ("$\mathbf{0}$ is equal to $\mathbf{0}$). Thus we obtain the following typing rule for $\langle t, M\rangle$:

$$\dfrac{M : [t/x]A}{\langle t, M\rangle : \exists x.A}\,\exists\mathsf{I}$$

Given that M has type $\exists x.A$, a proof term $\mathsf{let}\ \langle x, w\rangle = M\ \mathsf{in}\ N$ decides the type of N after binding x and w to a witness t and a proof term of type $[t/x]A$, respectively. (Note that x is a term variable

$$
\dfrac{[a/x]M : [a/x]A}{\lambda x.\, M : \forall x.A}\ \forall\mathsf{I}^a
\qquad
\dfrac{M : \forall x.A}{M\ t : [t/x]A}\ \forall\mathsf{E}
\qquad
\dfrac{M : [t/x]A}{\langle t, M\rangle : \exists x.A}\ \exists\mathsf{I}
\qquad
\dfrac{M : \exists x.A \quad \begin{array}{c}\overline{w : [a/x]A}\\ \vdots \\ [a/x]N : C\end{array}}{\mathsf{let}\ \langle x, w\rangle = M\ \mathsf{in}\ N : C}\ \exists\mathsf{E}^a
$$

Figure 5.3: Typing rules for proof terms in first-order logic

whereas w is a variable ranging over proof terms.) Since such a witness is unknown in general (*e.g.*, if M is a variable), we have to assume an arbitrary witness a and assign type $[a/x]A$ to w. Accordingly we replace x in N by a. Thus we obtain the following typing rule for $\mathsf{let}\ \langle x, w\rangle = M\ \mathsf{in}\ N$:

$$
\dfrac{M : \exists x.A \quad \begin{array}{c}\overline{w : [a/x]A}\\ \vdots \\ [a/x]N : C\end{array}}{\mathsf{let}\ \langle x, w\rangle = M\ \mathsf{in}\ N : C}\ \exists\mathsf{E}^a
$$

In practice, we may use the following typing rule with an extra assumption that x is a fresh term variable:

$$
\dfrac{M : \exists x.A \quad \begin{array}{c}\overline{w : A}\\ \vdots \\ N : C\end{array}}{\mathsf{let}\ \langle x, w\rangle = M\ \mathsf{in}\ N : C}\ \exists\mathsf{E}
$$

Figure 5.3 shows all the typing rules for proof terms in first-order logic.

Exercise 5.3. Rewrite these typing rules using hypothetical judgments.

We derive β-reductions and η-expansions for universal and existential quantifications from their corresponding local reductions and expansions of proofs given in Section 5.5. For universal quantifications, we assign proof terms as follows:

$$
\dfrac{\dfrac{[a/x]M : [a/x]A}{\lambda x.\, M : \forall x.A}\ \forall\mathsf{I}^a}{(\lambda x.\, M)\ t : [t/x]A}\ \forall\mathsf{E}
\quad \Longrightarrow_\beta \quad
[t/a][a/x]M : [t/a][a/x]A
$$

$$
M : \forall x.A \quad \Longrightarrow_\eta \quad
\dfrac{\dfrac{M : \forall x.A}{M\ a : [a/x]A}\ \forall\mathsf{E}}{\lambda x.\, M\ x : \forall x.A}\ \forall\mathsf{I}^a
\quad \textit{where } M\ a = [a/x](M\ x)
$$

As we have $[t/a][a/x]M = [t/a]M$ (and $[t/a][a/x]A = [t/x]A$), we obtain the following β-reduction and η-expansion:

$$
\begin{array}{lll}
(\lambda x.\, M)\ t & \Longrightarrow_\beta & [t/x]M \\
M : \forall x.A & \Longrightarrow_\eta & \lambda x.\, M\ x \quad (x \textit{ is not free in } M)
\end{array}
$$

For existential quantifications, we assign proof terms as follows:

$$
\dfrac{\dfrac{M : [t/x]A}{\langle t, M\rangle : \exists x.A}\ \exists\mathsf{I} \quad \begin{array}{c}\overline{w : [a/x]A}\\ \vdots \\ [a/x]N : C\end{array}}{\mathsf{let}\ \langle x, w\rangle = \langle t, M\rangle\ \mathsf{in}\ N : C}\ \exists\mathsf{E}^{a,w}
\quad \Longrightarrow_\beta \quad
\begin{array}{c}[M/w][t/a]w : [t/a][a/x]A\\ \vdots \\ [M/w][t/a][a/x]N : C\end{array}
$$

$$
M : \exists x.A \quad \Longrightarrow_\eta \quad
\dfrac{M : \exists x.A \quad \dfrac{\overline{w : [a/x]A}}{\langle a, w\rangle : \exists x.A}\ \exists\mathsf{I}}{\mathsf{let}\ \langle x, w\rangle = M\ \mathsf{in}\ \langle x, w\rangle : \exists x.A}\ \exists\mathsf{E}^a
\quad \textit{where } \langle a, w\rangle = [a/x]\langle x, w\rangle
$$

As we have $[M/w][t/a][a/x]N = [M/w][t/x]N$, we obtain the following β-reduction and η-expansion:

$$\begin{array}{rcl} \mathsf{let}\ \langle x, w\rangle = \langle t, M\rangle\ \mathsf{in}\ N & \Longrightarrow_\beta & [M/w][t/x]N \\ M : \exists x.A & \Longrightarrow_\eta & \mathsf{let}\ \langle x, w\rangle = M\ \mathsf{in}\ \langle x, w\rangle \end{array}$$

As in Section 3.4, we extend the definition of elim terms and intro terms by using an elim term E to represent a proof of $A{\downarrow}$ and an intro term I to represent a proof of $A{\uparrow}$. According to the rules for deducing neutral and normal judgments given in Section 5.5, we obtain the following definition:

$$\begin{array}{lccl} \text{elim term} & E & ::= & \cdots \mid E\ t \\ \text{intro term} & I & ::= & \cdots \mid \lambda x.\, I \mid \langle t, I\rangle \mid \mathsf{let}\ \langle x, w\rangle = E\ \mathsf{in}\ I \end{array}$$

The commuting conversion for existential quantification requires us to extend the definition of commuting conversion contexts explained in Section 3.4 and introduce another case for $M \Longrightarrow_c N$ as shown below:

$$\begin{array}{lccl} \text{commuting conversion context} & \kappa & ::= & \cdots \mid \Box\ t \mid \mathsf{let}\ \langle x, w\rangle = \Box\ \mathsf{in}\ I \end{array}$$

$$\kappa[\![\mathsf{let}\ \langle x, w\rangle = M\ \mathsf{in}\ N]\!] \quad \Longrightarrow_c \quad \mathsf{let}\ \langle x, w\rangle = M\ \mathsf{in}\ \kappa[\![N]\!]$$

5.8 Examples of proof terms

This section rewrites all the proofs in Section 5.6 using proof terms. First we need constant proof terms for axioms:

$$\dfrac{}{\mathsf{Nat_0} : \mathit{Nat}(\mathbf{0})}\ \mathit{Zero} \qquad \dfrac{}{\mathsf{Nat_s} : \forall x.\mathit{Nat}(x) \supset \mathit{Nat}(\mathbf{s}(x))}\ \mathit{Succ}$$

$$\dfrac{}{\mathsf{Eq_i} : \forall x.\mathit{Eq}(x,x)}\ \mathit{Eq_i} \qquad \dfrac{}{\mathsf{Eq_t} : \forall x.\forall y.\forall z.(\mathit{Eq}(x,y) \wedge \mathit{Eq}(x,z)) \supset \mathit{Eq}(y,z)}\ \mathit{Eq_t}$$

$$\dfrac{}{\mathsf{Lt_s} : \forall x.\mathit{Lt}(x, \mathbf{s}(x))}\ \mathit{Lt_s} \qquad \dfrac{}{\mathsf{Lt_\neg} : \forall x.\forall y.\mathit{Eq}(x,y) \supset \neg \mathit{Lt}(x,y)}\ \mathit{Lt_\neg}$$

The proof of $\mathit{Nat}(\mathbf{s}(\mathbf{s}(\mathbf{0})))\ \mathit{true}$ corresponds to a proof term $\mathsf{Nat_s}\ \mathbf{s}(\mathbf{0})\ (\mathsf{Nat_s}\ \mathbf{0}\ \mathsf{Nat_0})$ as shown in the following derivation tree:

$$\dfrac{\dfrac{\dfrac{}{\mathsf{Nat_s} : \forall x.\mathit{Nat}(x) \supset \mathit{Nat}(\mathbf{s}(x))}\ \mathit{Succ}}{\mathsf{Nat_s}\ \mathbf{s}(\mathbf{0}) : \mathit{Nat}(\mathbf{s}(\mathbf{0})) \supset \mathit{Nat}(\mathbf{s}(\mathbf{s}(\mathbf{0})))}\ \forall\mathsf{E} \quad \dfrac{\dfrac{\dfrac{}{\mathsf{Nat_s} : \forall x.\mathit{Nat}(x) \supset \mathit{Nat}(\mathbf{s}(x))}\ \mathit{Succ}}{\mathsf{Nat_s}\ \mathbf{0} : \mathit{Nat}(\mathbf{0}) \supset \mathit{Nat}(\mathbf{s}(\mathbf{0}))}\ \forall\mathsf{E} \quad \dfrac{}{\mathsf{Nat_0} : \mathit{Nat}(\mathbf{0})}\ \mathit{Zero}}{\mathsf{Nat_s}\ \mathbf{0}\ \mathsf{Nat_0} : \mathit{Nat}(\mathbf{s}(\mathbf{0}))}\ {\supset}\mathsf{E}}{\mathsf{Nat_s}\ \mathbf{s}(\mathbf{0})\ (\mathsf{Nat_s}\ \mathbf{0}\ \mathsf{Nat_0}) : \mathit{Nat}(\mathbf{s}(\mathbf{s}(\mathbf{0})))}\ {\supset}\mathsf{E}$$

In the same fashion, we obtain the following proof terms:

- Proof term of type $\forall x.\mathit{Nat}(x) \supset (\exists y.\mathit{Nat}(y) \wedge \mathit{Eq}(x,y))$:

$$\dfrac{\dfrac{\dfrac{\dfrac{\dfrac{}{z : \mathit{Nat}(a)} \quad \dfrac{\dfrac{}{\mathsf{Eq_i} : \forall x.\mathit{Eq}(x,x)}\ \mathit{Eq_i}}{\mathsf{Eq_i}\ a : \mathit{Eq}(a,a)}\ \forall\mathsf{E}}{(z, \mathsf{Eq_i}\ a) : \mathit{Nat}(a) \wedge \mathit{Eq}(a,a)}\ \wedge\mathsf{I}}{\langle a, (z, \mathsf{Eq_i}\ a)\rangle : \exists y.\mathit{Nat}(y) \wedge \mathit{Eq}(a,y)}\ \exists\mathsf{I}}{\lambda z{:}\mathit{Nat}(a).\, \langle a, (z, \mathsf{Eq_i}\ a)\rangle : \mathit{Nat}(a) \supset (\exists y.\mathit{Nat}(y) \wedge \mathit{Eq}(a,y))}\ {\supset}\mathsf{I}^z}{\lambda x.\, \lambda z{:}\mathit{Nat}(x).\, \langle x, (z, \mathsf{Eq_i}\ x)\rangle : \forall x.\mathit{Nat}(x) \supset (\exists y.\mathit{Nat}(y) \wedge \mathit{Eq}(x,y))}\ \forall\mathsf{I}^a$$

- Proof term of type $\forall x.\forall y.\mathit{Eq}(x,y) \supset \mathit{Eq}(y,x)$:

$$\dfrac{\dfrac{\dfrac{\dfrac{\dfrac{\dfrac{\dfrac{\dfrac{}{\mathsf{Eq_t} : \forall x.\forall y.\forall z.(\mathit{Eq}(x,y) \wedge \mathit{Eq}(x,z)) \supset \mathit{Eq}(y,z)}\ \mathit{Eq_t}}{\mathsf{Eq_t}\ a : \forall y.\forall z.(\mathit{Eq}(a,y) \wedge \mathit{Eq}(a,z)) \supset \mathit{Eq}(y,z)}\ \forall\mathsf{E}}{\mathsf{Eq_t}\ a\ b : \forall z.(\mathit{Eq}(a,b) \wedge \mathit{Eq}(a,z)) \supset \mathit{Eq}(b,z)}\ \forall\mathsf{E}}{\mathsf{Eq_t}\ a\ b\ a : (\mathit{Eq}(a,b) \wedge \mathit{Eq}(a,a)) \supset \mathit{Eq}(b,a)}\ \forall\mathsf{E} \quad \dfrac{\dfrac{}{w : \mathit{Eq}(a,b)} \quad \dfrac{\dfrac{}{\mathsf{Eq_i} : \forall x.\mathit{Eq}(x,x)}\ \mathit{Eq_i}}{\mathsf{Eq_i}\ a : \mathit{Eq}(a,a)}\ \forall\mathsf{E}}{(w, \mathsf{Eq_i}\ a) : \mathit{Eq}(a,b) \wedge \mathit{Eq}(a,a)}\ \wedge\mathsf{I}}{\mathsf{Eq_t}\ a\ b\ a\ (w, \mathsf{Eq_i}\ a) : \mathit{Eq}(b,a)}\ {\supset}\mathsf{E}}{\lambda w{:}\mathit{Eq}(a,b).\, \mathsf{Eq_t}\ a\ b\ a\ (w, \mathsf{Eq_i}\ a) : \mathit{Eq}(a,b) \supset \mathit{Eq}(b,a)}\ {\supset}\mathsf{I}^w}{\lambda y.\, \lambda w{:}\mathit{Eq}(a,y).\, \mathsf{Eq_t}\ a\ y\ a\ (w, \mathsf{Eq_i}\ a) : \forall y.\mathit{Eq}(a,y) \supset \mathit{Eq}(y,a)}\ \forall\mathsf{I}^b}{\lambda x.\, \lambda y.\, \lambda w{:}\mathit{Eq}(x,y).\, \mathsf{Eq_t}\ x\ y\ x\ (w, \mathsf{Eq_i}\ x) : \forall x.\forall y.\mathit{Eq}(x,y) \supset \mathit{Eq}(y,x)}\ \forall\mathsf{I}^a$$

- Proof term of type $\neg\exists x.Eq(x,\mathbf{0}) \wedge Eq(x,\mathrm{s}(\mathbf{0}))$:

$$
\dfrac{\dfrac{\dfrac{}{w : \exists x.Eq(x,\mathbf{0}) \wedge Eq(x,\mathrm{s}(\mathbf{0}))} \qquad \dfrac{\begin{array}{c}\mathcal{E}\\ (\mathsf{Lt}_\neg\;\mathbf{0}\;\mathrm{s}(\mathbf{0}))\;(\mathsf{Eq_t}\;a\;\mathbf{0}\;\mathrm{s}(\mathbf{0})\;z) : \neg Lt(\mathbf{0},\mathrm{s}(\mathbf{0}))\end{array} \quad \dfrac{\dfrac{}{\mathsf{Lt_s} : \forall x.Lt(x,\mathrm{s}(x))}\,Lt_s}{\mathsf{Lt_s}\;\mathbf{0} : Lt(\mathbf{0},\mathrm{s}(\mathbf{0}))}\,\forall\mathsf{E}}{(\mathsf{Lt}_\neg\;\mathbf{0}\;\mathrm{s}(\mathbf{0}))\;(\mathsf{Eq_t}\;a\;\mathbf{0}\;\mathrm{s}(\mathbf{0})\;z)\;(\mathsf{Lt_s}\;\mathbf{0}) : \bot}\,\neg\mathsf{E}}{\mathsf{let}\;\langle x,z\rangle = w\;\mathsf{in}\;(\mathsf{Lt}_\neg\;\mathbf{0}\;\mathrm{s}(\mathbf{0}))\;(\mathsf{Eq_t}\;x\;\mathbf{0}\;\mathrm{s}(\mathbf{0})\;z)\;(\mathsf{Lt_s}\;\mathbf{0}) : \bot}\,\exists\mathsf{E}^a}{\lambda w{:}\exists x.Eq(x,\mathbf{0}) \wedge Eq(x,\mathrm{s}(\mathbf{0})).\;\mathsf{let}\;\langle x,z\rangle = w\;\mathsf{in}\;(\mathsf{Lt}_\neg\;\mathbf{0}\;\mathrm{s}(\mathbf{0}))\;(\mathsf{Eq_t}\;x\;\mathbf{0}\;\mathrm{s}(\mathbf{0})\;z)\;(\mathsf{Lt_s}\;\mathbf{0}) : \neg\exists x.Eq(x,\mathbf{0}) \wedge Eq(x,\mathrm{s}(\mathbf{0}))}\,\neg\mathsf{I}^w
$$

where we let

$$
\mathcal{E} \quad = \quad \dfrac{\dfrac{\dfrac{\dfrac{}{\mathsf{Lt}_\neg : \forall x.\forall y.Eq(x,y) \supset \neg Lt(x,y)}\,Lt_\neg}{\mathsf{Lt}_\neg\;\mathbf{0} : \forall y.Eq(\mathbf{0},y) \supset \neg Lt(\mathbf{0},y)}\,\forall\mathsf{E}}{\mathsf{Lt}_\neg\;\mathbf{0}\;\mathrm{s}(\mathbf{0}) : Eq(\mathbf{0},\mathrm{s}(\mathbf{0})) \supset \neg Lt(\mathbf{0},\mathrm{s}(\mathbf{0}))}\,\forall\mathsf{E} \quad \begin{array}{c}\mathcal{D}\\ \mathsf{Eq_t}\;a\;\mathbf{0}\;\mathrm{s}(\mathbf{0})\;z : Eq(\mathbf{0},\mathrm{s}(\mathbf{0}))\end{array}}{(\mathsf{Lt}_\neg\;\mathbf{0}\;\mathrm{s}(\mathbf{0}))\;(\mathsf{Eq_t}\;a\;\mathbf{0}\;\mathrm{s}(\mathbf{0})\;z) : \neg Lt(\mathbf{0},\mathrm{s}(\mathbf{0}))}\,\supset\mathsf{E}
$$

where we let

$$
\mathcal{D} \quad = \quad \dfrac{\dfrac{\dfrac{\dfrac{\dfrac{}{\mathsf{Eq_t} : \forall x.\forall y.\forall z.(Eq(x,y) \wedge Eq(x,z)) \supset Eq(y,z)}\,Eq_t}{\mathsf{Eq_t}\;a : \forall y.\forall z.(Eq(a,y) \wedge Eq(a,z)) \supset Eq(y,z)}\,\forall\mathsf{E}}{\mathsf{Eq_t}\;a\;\mathbf{0} : \forall z.(Eq(a,\mathbf{0}) \wedge Eq(a,z)) \supset Eq(\mathbf{0},z)}\,\forall\mathsf{E}}{\mathsf{Eq_t}\;a\;\mathbf{0}\;\mathrm{s}(\mathbf{0}) : (Eq(a,\mathbf{0}) \wedge Eq(a,\mathrm{s}(\mathbf{0}))) \supset Eq(\mathbf{0},\mathrm{s}(\mathbf{0}))}\,\forall\mathsf{E} \quad \dfrac{}{z : Eq(a,\mathbf{0}) \wedge Eq(a,\mathrm{s}(\mathbf{0}))}}{\mathsf{Eq_t}\;a\;\mathbf{0}\;\mathrm{s}(\mathbf{0})\;z : Eq(\mathbf{0},\mathrm{s}(\mathbf{0}))}\,\supset\mathsf{E}
$$

Chapter 6

Datatypes

In pure first-order logic, term variables are assumed to range over all kinds of terms and their domains are left unspecified. Hence we can restrict the domain of a term variable only indirectly by using a predicate corresponding to a particular domain. For example, we may use a predicate $Nat(x)$ to specify that x ranges over natural numbers, as in:

$$\forall x.Nat(x) \supset A$$
$$\exists x.Nat(x) \wedge A$$

This chapter develops first-order logic with *datatypes* which explicitly specifies the domain of each term variable bound by a quantifier $\forall$ or $\exists$. We write $\forall x \in \tau.A$ and $\exists x \in \tau.A$ to specify that term variable x in proposition A ranges over datatype τ. For example, the above two propositions can now be concisely written as

$$\forall x \in \mathsf{nat}.A$$
$$\exists x \in \mathsf{nat}.A$$

where nat is a datatype for natural numbers.

The main judgment for first-order logic with datatypes is $t \in \tau$:

$$t \in \tau \qquad \Leftrightarrow \qquad \textit{term } t \textit{ has datatype } \tau$$

As in propositional logic and pure first-order logic, we base the development of datatypes on natural deduction. For example, each datatype τ is accompanied by introduction and elimination rules for deducing and exploiting judgments $t \in \tau$. We use metavariables τ and σ for datatypes, and t and s for terms.

From this chapter on, we adopt a new notation $A(x)$ to mean that proposition A contains term variable x, as in $\forall x \in \mathsf{nat}.A(x)$ and $\exists x \in \mathsf{nat}.A(x)$. Accordingly $A(t)$ stands for A in which every occurrence of x has been replaced by t. That is, we have $A(t) = [t/x]A$.

6.1 Basic constructors for datatypes

Before we consider concrete datatypes such as bool for boolean values and nat for natural numbers, we develop basic constructors for datatypes to obtain a general language similar to the simply-typed λ-calculus:

$$\text{datatype} \quad \tau \quad ::= \quad \cdots \mid \tau \rightarrow \tau \mid \tau \times \tau \mid \tau + \tau \mid \mathsf{unit} \mid \mathsf{void}$$

We call $\tau \rightarrow \sigma$ a function type, $\tau \times \sigma$ a product type, $\tau + \sigma$ a sum type, unit a unit type, and void a void type. We will use terms of these datatypes as programs for manipulating ordinary terms of such concrete datatypes. For example, we use a term of datatype $\mathsf{nat} \rightarrow \mathsf{bool}$ as a function mapping natural numbers to boolean values, and a term of datatype $\mathsf{nat} \times \mathsf{nat}$ to carry a pair of natural numbers. We assume that $\rightarrow, \times, +$ are all right-associative.

The basic constructors for datatypes have their counterparts in the simply-typed λ-calculus as follows:

datatype	$\rightarrow$	$\times$	$+$	unit	void
type	$\supset$	$\wedge$	$\vee$	$\top$	$\bot$

$$\dfrac{\begin{array}{c}\overline{x \in \tau}\\ \vdots \\ t \in \sigma\end{array}}{\lambda x \in \tau.\, t \in \tau \to \sigma}\ {\to}\mathsf{I} \qquad \dfrac{t \in \tau \to \sigma \quad s \in \tau}{t\ s \in \sigma}\ {\to}\mathsf{E} \qquad \dfrac{t \in \tau \quad s \in \sigma}{\langle t, s\rangle \in \tau \times \sigma}\ {\times}\mathsf{I} \qquad \dfrac{t \in \tau \times \sigma}{\mathbf{fst}\ t \in \tau}\ {\times}\mathsf{E_L} \qquad \dfrac{t \in \tau \times \sigma}{\mathbf{snd}\ t \in \sigma}\ {\times}\mathsf{E_R}$$

$$\dfrac{t \in \tau}{\mathbf{inl}_\sigma\ t \in \tau + \sigma}\ {+}\mathsf{I_L} \qquad \dfrac{t \in \sigma}{\mathbf{inr}_\tau\ t \in \tau \vee \sigma}\ {+}\mathsf{I_R} \qquad \dfrac{t \in \tau + \tau' \quad \begin{array}{c}\overline{x \in \tau}\\ \vdots \\ s \in \sigma\end{array} \quad \begin{array}{c}\overline{y \in \tau'}\\ \vdots \\ s' \in \sigma\end{array}}{\mathbf{case}\ t\ \mathbf{of}\ \mathbf{inl}\ x.\, s \mid \mathbf{inr}\ y.\, s' \in \sigma}\ {+}\mathsf{E}$$

$$\dfrac{}{\langle\rangle \in \mathsf{unit}}\ \mathsf{unitI} \qquad \dfrac{t \in \mathsf{void}}{\mathbf{abort}_\tau\ t \in \tau}\ \mathsf{voidE}$$

Figure 6.1: Typing rules for terms

For example, function types of the form $\tau \to \sigma$ correspond to types of the form $A \supset B$. Terms for these datatypes also have their counterparts in the simply-typed λ-calculus. For example, as we use a λ-abstraction $\lambda x\!:\!A.\, M$ as a proof term of type $A \supset B$, we use another form of λ-abstraction $\lambda x \in \tau.\, t$ as a term of datatype $\tau \to \sigma$. The definition of terms reuses the syntax for proof terms in the simply-typed λ-calculus with a few cosmetic changes:

$$\begin{array}{lll}\text{term} \quad t & ::= & \cdots \mid \lambda x \in \tau.\, t \mid t\ t \mid \langle t, t\rangle \mid \mathbf{fst}\ t \mid \mathbf{snd}\ t \mid \mathbf{inl}_\tau\ t \mid \mathbf{inr}_\tau\ t \mid \mathbf{case}\ t\ \mathbf{of}\ \mathbf{inl}\ x.\, t \mid \mathbf{inr}\ x.\, t \mid \\ & & \langle\rangle \mid \mathbf{abort}_\tau\ t\end{array}$$

Figure 6.1 shows the typing rules for terms, all of which are obtained in an analogous way to the typing rules for the simply-typed λ-calculus in Figure 3.1. We may also rewrite these typing rules using hypothetical judgments of the form $\Gamma \vdash t \in \sigma$ where typing context Γ denotes a collection of datatype bindings of the form $x \in \tau$, as in Figure 3.2.

Since the typing rules are all based on the principle of natural deduction, we obtain β-reductions and η-expansions of terms in the same way that we obtain those for the simply-typed λ-calculus:

$$\begin{array}{rcll}(\lambda x \in \tau.\, t)\ s & \Longrightarrow_\beta & [s/x]t & \\ \mathsf{fst}\ (t, s) & \Longrightarrow_\beta & t & \\ \mathsf{snd}\ (t, s) & \Longrightarrow_\beta & s & \\ \mathsf{case}\ \mathsf{inl}_\sigma\ t\ \mathsf{of}\ \mathsf{inl}\ x.\, s \mid \mathsf{inr}\ y.\, s' & \Longrightarrow_\beta & [t/x]s & \\ \mathsf{case}\ \mathsf{inr}_\tau\ t\ \mathsf{of}\ \mathsf{inl}\ x.\, s \mid \mathsf{inr}\ y.\, s' & \Longrightarrow_\beta & [t/y]s' & \\ t \in \tau \to \sigma & \Longrightarrow_\eta & \lambda x \in \tau.\, t\ x & \textit{(x is not free in t)} \\ t \in \tau \times \sigma & \Longrightarrow_\eta & \langle \mathbf{fst}\ t, \mathbf{snd}\ t\rangle & \\ t \in \tau + \sigma & \Longrightarrow_\eta & \mathbf{case}\ t\ \mathbf{of}\ \mathbf{inl}\ x.\, \mathbf{inl}_\sigma\ x \mid \mathbf{inr}\ y.\, \mathbf{inr}_\tau\ y & \\ t \in \mathsf{unit} & \Longrightarrow_\eta & \langle\rangle & \\ t \in \mathsf{void} & \Longrightarrow_\eta & \mathbf{abort}_{\mathsf{void}}\ t & \end{array}$$

Here $[s/x]t$, similar to $[N/x]M$, denotes a capture-avoiding substitution of s for x in t. With β-reductions and η-expansions available, these terms constitute a general language of their own.

6.2 Natural deduction for datatypes

We now consider two concrete datatypes bool for boolean values and nat for natural numbers. Again we explain the meaning of judgments $t \in \mathsf{bool}$ and $t \in \mathsf{nat}$ using the principle of natural deduction. For example, an introduction rule for bool specifies how to deduce a new judgment $t \in \mathsf{bool}$ whereas an elimination rule for bool specifies how to exploit an existing judgment $t \in \mathsf{bool}$. Usually we first *design* introduction rules according to the intuition behind a given datatype and then *derive* elimination rules from these introduction rules. In fact, elimination rules for a datatype can be automatically derived from its introduction rules as long as terms of the datatype are defined inductively.

Let us consider datatype bool for boolean values:

$$\text{datatype} \quad \tau \quad ::= \quad \cdots \mid \mathsf{bool}$$

As a boolean value allows us to choose one of two different options, we associate with datatype bool two terms, **true** and **false**, indicating which option to choose:

$$\frac{}{\mathbf{true} \in \mathsf{bool}}\ \mathsf{boolI_t} \qquad \frac{}{\mathbf{false} \in \mathsf{bool}}\ \mathsf{boolI_f}$$

Suppose now that we have a judgment $t \in \mathsf{bool}$ that we wish to exploit in deducing another judgment. Since it is in general unknown whether t is equivalent to **true** or **false**, we provide for both possibilities using a term matching t with **true** and **false** in turn:

$$\frac{t \in \mathsf{bool} \quad t_1 \in \tau \quad t_2 \in \tau}{\mathbf{case}\ t\ \mathbf{of\ true} \Rightarrow t_1 \mid \mathbf{false} \Rightarrow t_2 \in \tau}\ \mathsf{boolE}$$

Note that although the rule boolE eliminates a term of datatype bool, the term in its conclusion may have a different datatype τ.

We choose to abbreviate $\mathbf{case}\ t\ \mathbf{of\ true} \Rightarrow t_1 \mid \mathbf{false} \Rightarrow t_2$ as $\mathbf{if}\ t\ \mathbf{then}\ t_1\ \mathbf{else}\ t_2$ familiar from programming languages. Thus we obtain the following definition of terms for datatype bool:

$$\text{term} \quad t \quad ::= \quad \cdots \mid \mathbf{true} \mid \mathbf{false} \mid \mathbf{if}\ t\ \mathbf{then}\ t\ \mathbf{else}\ t$$

Local reductions and expansions of proofs (which we omit) are translated to the following β-reductions and η-expansion for datatype bool:

$$\begin{array}{rcl}
\mathbf{if\ true\ then}\ t_1\ \mathbf{else}\ t_2 & \Longrightarrow_\beta & t_1 \\
\mathbf{if\ false\ then}\ t_1\ \mathbf{else}\ t_2 & \Longrightarrow_\beta & t_2 \\
t \in \mathsf{bool} & \Longrightarrow_\eta & \mathbf{if}\ t\ \mathbf{then\ true\ else\ false}
\end{array}$$

Here are a few examples of functions manipulating boolean values. and and or compute the logical conjunction and disjunction, respectively, of two boolean values. not computes the logical negation of a boolean value.

$$\begin{array}{ll}
and \in \mathsf{bool} \to \mathsf{bool} \to \mathsf{bool} & and \ = \ \lambda x \in \mathsf{bool}.\, \lambda y \in \mathsf{bool}.\, \mathbf{if}\ x\ \mathbf{then}\ y\ \mathbf{else\ false} \\
or \in \mathsf{bool} \to \mathsf{bool} \to \mathsf{bool} & or \ = \ \lambda x \in \mathsf{bool}.\, \lambda y \in \mathsf{bool}.\, \mathbf{if}\ x\ \mathbf{then\ true\ else}\ y \\
not \in \mathsf{bool} \to \mathsf{bool} & not \ = \ \lambda x \in \mathsf{bool}.\, \mathbf{if}\ x\ \mathbf{then\ false\ else\ true}
\end{array}$$

Exercise 6.1. Instead of defining it as a new datatype, we can simulate bool as $\mathsf{unit}+\mathsf{unit}$. Then how do we simulate **true**, **false**, and $\mathbf{if}\ t\ \mathbf{then}\ t_1\ \mathbf{else}\ t_2$?

Datatype nat defines a natural number as either zero $\mathbf{0}$ or a successor $\mathbf{s}(t)$ of another natural number t.

$$\text{datatype} \quad \tau \quad ::= \quad \cdots \mid \mathsf{nat}$$

$$\frac{}{\mathbf{0} \in \mathsf{nat}}\ \mathsf{natI_0} \qquad \frac{t \in \mathsf{nat}}{\mathbf{s}(t) \in \mathsf{nat}}\ \mathsf{natI_s}$$

The elimination rule is similar to the rule boolE and considers two cases for a given term t of datatype nat: when t matches zero and when t matches a successor of another natural number. The difference is that in the second case, the elimination rule binds a term variable, say x, to the predecessor of t which is unknown in general. Thus the elimination rule for datatype nat uses a hypothesis of $x \in \mathsf{nat}$:

$$\frac{t \in \mathsf{nat} \quad t_0 \in \tau \quad \begin{array}{c}\overline{x \in \mathsf{nat}} \\ \vdots \\ t_s \in \tau\end{array}}{\mathbf{case}\ t\ \mathbf{of}\ \mathbf{0} \Rightarrow t_0 \mid \mathbf{s}(x) \Rightarrow t_s \in \tau}\ \mathsf{natE}$$

Here x is a local term variable whose scope is restricted to t_s, and we may rename it whenever necessary. Thus we obtain the following definition of terms for datatype nat:

$$\text{term} \quad t \quad ::= \quad \cdots \mid \mathbf{0} \mid \mathbf{s}(t) \mid \mathbf{case}\ t\ \mathbf{of}\ \mathbf{0} \Rightarrow t \mid \mathbf{s}(x) \Rightarrow t$$

A local reduction for datatype nat takes a proof in which the introduction rule $\mathsf{natI_0}$ or $\mathsf{natI_s}$ is immediately followed by the elimination rule natE:

$$\dfrac{\dfrac{}{\mathbf{0} \in \mathsf{nat}}\,\mathsf{natI_0} \quad \begin{array}{c}\mathcal{E}\\ t_0 \in \tau\end{array} \quad \begin{array}{c}\overline{x \in \mathsf{nat}}\\ \vdots \\ t_s \in \tau\end{array}}{\mathbf{case}\ \mathbf{0}\ \mathbf{of}\ \mathbf{0} \Rightarrow t_0 \mid \mathbf{s}(x) \Rightarrow t_s \in \tau}\,\mathsf{natE} \qquad \Longrightarrow_R \qquad \begin{array}{c}\mathcal{E}\\ t_0 \in \tau\end{array}$$

$$\dfrac{\dfrac{\begin{array}{c}\mathcal{D}\\ t \in \mathsf{nat}\end{array}}{\mathbf{s}(t) \in \mathsf{nat}}\,\mathsf{natI_s} \quad \begin{array}{c}\mathcal{E}\\ t_0 \in \tau\end{array} \quad \begin{array}{c}\overline{x \in \mathsf{nat}}\\ \vdots \\ t_s \in \tau\end{array}}{\mathbf{case}\ \mathbf{s}(t)\ \mathbf{of}\ \mathbf{0} \Rightarrow t_0 \mid \mathbf{s}(x) \Rightarrow t_s \in \tau}\,\mathsf{natE} \qquad \Longrightarrow_R \qquad \begin{array}{c}\mathcal{D}\\ t \in \mathsf{nat}\\ \vdots \\ [t/x]t_s \in \tau\end{array}$$

From these local reductions, we obtain the following β-reductions for datatype nat:

$$\begin{array}{rcl}\mathbf{case}\ \mathbf{0}\ \mathbf{of}\ \mathbf{0} \Rightarrow t_0 \mid \mathbf{s}(x) \Rightarrow t_s & \Longrightarrow_\beta & t_0 \\ \mathbf{case}\ \mathbf{s}(t)\ \mathbf{of}\ \mathbf{0} \Rightarrow t_0 \mid \mathbf{s}(x) \Rightarrow t_s & \Longrightarrow_\beta & [t/x]t_s\end{array}$$

A local expansion takes a proof of $t \in \mathsf{nat}$ and gives rise to the following η-expansion:

$$t \in \mathsf{nat} \qquad \Longrightarrow_\eta \qquad \mathbf{case}\ t\ \mathbf{of}\ \mathbf{0} \Rightarrow \mathbf{0} \mid \mathbf{s}(x) \Rightarrow \mathbf{s}(x)$$

Exercise 6.2. Show a local expansion of a proof of $t \in \mathsf{nat}$.

Now we can define various functions returning different results depending on whether a given natural number is zero or not. For example, we define a function returning the predecessor of a given natural number as follows:

$$\mathit{pred} \in \mathsf{nat} \rightarrow \mathsf{nat} \qquad \mathit{pred} \;=\; \lambda x \in \mathsf{nat}.\,\mathbf{case}\ x\ \mathbf{of}\ \mathbf{0} \Rightarrow \mathbf{0} \mid \mathbf{s}(y) \Rightarrow y$$

The extent to which we define such functions is limited, however, because we have no machinery for defining *recursive* functions. For example, the following definition of a function doubling a given natural number is not valid because *double* in the body of the λ-abstraction is a free term variable whose definition is still incomplete:

$$\mathit{double} \in \mathsf{nat} \rightarrow \mathsf{nat} \qquad \mathit{double} \;=\; \lambda x \in \mathsf{nat}.\,\mathbf{case}\ x\ \mathbf{of}\ \mathbf{0} \Rightarrow \mathbf{0} \mid \mathbf{s}(y) \Rightarrow \mathbf{s}(\mathbf{s}(\mathit{double}\ y))$$

In the next section, we revise the rule natE so that we can define recursive functions over natural numbers. Instead of a general form of recursion, we base the rule natE on *primitive recursion* which guarantees that every recursive call eventually terminates. Not every recursive function is definable with primitive recursion (*e.g.*, the Ackermann function), but in the study of first-order logic with datatypes, we seldom need such recursive functions.

6.3 Primitive recursion

The revised rule natE based on primitive recursion is another elimination rule for datatype nat:

$$\dfrac{t \in \mathsf{nat} \quad t_0 \in \tau \quad \begin{array}{c}\overline{x \in \mathsf{nat}} \quad \overline{f(x) \in \tau}\\ \vdots \\ t_s \in \tau\end{array}}{\mathbf{rec}\ f(t)\ \mathbf{of}\ f(\mathbf{0}) \Rightarrow t_0 \mid f(\mathbf{s}(x)) \Rightarrow t_s \in \tau}\,\mathsf{natE}$$

We may think of $\mathbf{rec}\ f(t)\ \mathbf{of}\ f(\mathbf{0}) \Rightarrow t_0 \mid f(\mathbf{s}(x)) \Rightarrow t_s$ as a primitive recursive function f applied to t. In the base case where t is $\mathbf{0}$, we take t_0. Hence t_0 is not permitted to make a recursive call to f. In the recursive case where t matches $\mathbf{s}(x)$, we take t_s. Inside t_s, we use x to denote the predecessor of t and $f(x)$ to denote a recursive call to f. Sometimes we write $\mathbf{rec}\ f(t)\ \mathbf{of}\ \left\{\begin{array}{l} f(\mathbf{0}) \Rightarrow t_0 \\ f(\mathbf{s}(x)) \Rightarrow t_s\end{array}\right.$ for visual clarity.

It is important that t_s may contain a recursive call to f, but only with x as its argument. For example, such terms as $f(\mathit{pred}\ x)$ and $f(\mathbf{s}(\mathbf{0}))$ are disallowed in t_s. This syntactic restriction is the characteristic feature of primitive recursion which prohibits an infinite sequence of recursive calls and thus guarantees that a primitive recursive function always terminates regardless of its actual argument. Since every recursive call to f within t_s always takes x as its argument, we regard $f(x)$ as not an application but a term variable in itself.

Note that if t_s contains no recursive call to f, the whole term $\mathbf{rec}\ f(t)\ \mathbf{of}\ f(\mathbf{0}) \Rightarrow t_0 \mid f(\mathbf{s}(x)) \Rightarrow t_s$ simplifies to $\mathbf{case}\ t\ \mathbf{of}\ \mathbf{0} \Rightarrow t_0 \mid \mathbf{s}(x) \Rightarrow t_s$. Since the previous rule natE is a special case of the revised rule natE, we revise the definition of terms for datatype nat as follows:

$$\text{term} \quad t \quad ::= \quad \cdots \mid \mathbf{0} \mid \mathbf{s}(t) \mid \mathbf{rec}\ f(t)\ \mathbf{of}\ f(\mathbf{0}) \Rightarrow t \mid f(\mathbf{s}(x)) \Rightarrow t$$

The β-reductions and η-expansion for datatype nat are given as follows:

$$\begin{array}{rcl}
\mathbf{rec}\ f(\mathbf{0})\ \mathbf{of}\ f(\mathbf{0}) \Rightarrow t_0 \mid f(\mathbf{s}(x)) \Rightarrow t_s & \Longrightarrow_\beta & t_0 \\
\mathbf{rec}\ f(\mathbf{s}(t))\ \mathbf{of}\ f(\mathbf{0}) \Rightarrow t_0 \mid f(\mathbf{s}(x)) \Rightarrow t_s & \Longrightarrow_\beta & [\mathbf{rec}\ f(t)\ \mathbf{of}\ f(\mathbf{0}) \Rightarrow t_0 \mid f(\mathbf{s}(x)) \Rightarrow t_s/f(x)][t/x]t_s \\
t \in \mathsf{nat} & \Longrightarrow_\eta & \mathbf{rec}\ f(t)\ \mathbf{of}\ f(\mathbf{0}) \Rightarrow \mathbf{0} \mid f(\mathbf{s}(x)) \Rightarrow \mathbf{s}(x)
\end{array}$$

In the first β-reduction, the left term reduces to t_0 because the argument to f is $\mathbf{0}$. In the second β-reduction where the argument $\mathbf{s}(t)$ matches $\mathbf{s}(x)$, we apply an equality $x = t$ throughout t_s by replacing x by t and $f(x)$ by a term denoting a call to f with an argument of t, namely $\mathbf{rec}\ f(t)\ \mathbf{of}\ f(\mathbf{0}) \Rightarrow t_0 \mid f(\mathbf{s}(x)) \Rightarrow t_s$. The η-expansion does not involve a recursive call.

Now we can define a wide range of recursive functions. For example, we specify a function double doubling a given natural number as follows:

$$\begin{array}{rcl}
\mathit{double}\ \mathbf{0} & = & \mathbf{0} \\
\mathit{double}\ \mathbf{s}(x) & = & \mathbf{s}(\mathbf{s}(\mathit{double}\ x))
\end{array}$$

The above specification translates to the following definition:

$$\begin{array}{l}
\mathit{double} \in \mathsf{nat} \rightarrow \mathsf{nat} \\
\mathit{double} \ = \ \lambda x \in \mathsf{nat}.\, \mathbf{rec}\ d(x)\ \mathbf{of}\ d(\mathbf{0}) \Rightarrow \mathbf{0} \mid d(\mathbf{s}(y)) \Rightarrow \mathbf{s}(\mathbf{s}(d(y)))
\end{array}$$

The following sequence of β-reductions (which are applied to subterms when necessary) shows that $\mathit{double}\ \mathbf{s}(\mathbf{0})$ reduces to $\mathbf{s}(\mathbf{s}(\mathbf{0}))$:

$$\begin{array}{rcl}
\mathit{double}\ \mathbf{s}(\mathbf{0}) & \Longrightarrow_\beta & \mathbf{rec}\ d(\mathbf{s}(\mathbf{0}))\ \mathbf{of}\ d(\mathbf{0}) \Rightarrow \mathbf{0} \mid d(\mathbf{s}(y)) \Rightarrow \mathbf{s}(\mathbf{s}(d(y))) \\
& \Longrightarrow_\beta & \mathbf{s}(\mathbf{s}(\mathbf{rec}\ d(\mathbf{0})\ \mathbf{of}\ d(\mathbf{0}) \Rightarrow \mathbf{0} \mid d(\mathbf{s}(y)) \Rightarrow \mathbf{s}(\mathbf{s}(d(y))))) \\
& \Longrightarrow_\beta & \mathbf{s}(\mathbf{s}(\mathbf{0}))
\end{array}$$

As another example, we specify and define a function plus adding two natural numbers as follows:

$$\begin{array}{rcl}
\mathit{plus}\ \mathbf{0}\ y & = & y \\
\mathit{plus}\ \mathbf{s}(x)\ y & = & \mathbf{s}(\mathit{plus}\ x\ y)
\end{array}$$

$$\begin{array}{l}
\mathit{plus} \in \mathsf{nat} \rightarrow \mathsf{nat} \rightarrow \mathsf{nat} \\
\mathit{plus} \ = \ \lambda x \in \mathsf{nat}.\, \lambda y \in \mathsf{nat}.\, \mathbf{rec}\ p(x)\ \mathbf{of}\ p(\mathbf{0}) \Rightarrow y \mid p(\mathbf{s}(z)) \Rightarrow \mathbf{s}(p(z))
\end{array}$$

The following sequence of β-reductions shows that $\mathbf{s}(\mathbf{0})$ and t add to $\mathbf{s}(t)$:

$$\begin{array}{rcl}
\mathit{plus}\ \mathbf{s}(\mathbf{0})\ t & \Longrightarrow_\beta & (\lambda y \in \mathsf{nat}.\, \mathbf{rec}\ p(\mathbf{s}(\mathbf{0}))\ \mathbf{of}\ p(\mathbf{0}) \Rightarrow y \mid p(\mathbf{s}(z)) \Rightarrow \mathbf{s}(p(z)))\ t \\
& \Longrightarrow_\beta & \mathbf{rec}\ p(\mathbf{s}(\mathbf{0}))\ \mathbf{of}\ p(\mathbf{0}) \Rightarrow t \mid p(\mathbf{s}(z)) \Rightarrow \mathbf{s}(p(z)) \\
& \Longrightarrow_\beta & \mathbf{s}(\mathbf{rec}\ p(\mathbf{0})\ \mathbf{of}\ p(\mathbf{0}) \Rightarrow t \mid p(\mathbf{s}(z)) \Rightarrow \mathbf{s}(p(z))) \\
& \Longrightarrow_\beta & \mathbf{s}(t)
\end{array}$$

Alternatively we may define plus in such a way that it recurses over the first argument x before taking the second argument y:

$$\mathit{plus} \ = \ \lambda x \in \mathsf{nat}.\, \mathbf{rec}\ p(x)\ \mathbf{of}\ p(\mathbf{0}) \Rightarrow \lambda y \in \mathsf{nat}.\, y \mid p(\mathbf{s}(z)) \Rightarrow \lambda y \in \mathsf{nat}.\, \mathbf{s}(p(z)\ y)$$

In this case, the reduction of $plus\ \mathbf{s}(\mathbf{0})\ t$ requires one more step:

$$\begin{array}{lll}
plus\ \mathbf{s}(\mathbf{0})\ t & \Longrightarrow_\beta & \left(\mathbf{rec}\ p(\mathbf{s}(\mathbf{0}))\ \mathbf{of}\ \left\{\begin{array}{l} p(\mathbf{0}) \Rightarrow \lambda y \in \mathsf{nat}.\, y \\ p(\mathbf{s}(z)) \Rightarrow \lambda y \in \mathsf{nat}.\, \mathbf{s}(p(z)\ y) \end{array}\right.\right)\ t \\
& \Longrightarrow_\beta & \lambda y \in \mathsf{nat}.\, \mathbf{s}(\left(\mathbf{rec}\ p(\mathbf{0})\ \mathbf{of}\ \left\{\begin{array}{l} p(\mathbf{0}) \Rightarrow \lambda y \in \mathsf{nat}.\, y \\ p(\mathbf{s}(z)) \Rightarrow \lambda y \in \mathsf{nat}.\, \mathbf{s}(p(z)\ y) \end{array}\right.\right)\ y)\ t \\
& \Longrightarrow_\beta & \mathbf{s}(\left(\mathbf{rec}\ p(\mathbf{0})\ \mathbf{of}\ \left\{\begin{array}{l} p(\mathbf{0}) \Rightarrow \lambda y \in \mathsf{nat}.\, y \\ p(\mathbf{s}(z)) \Rightarrow \lambda y \in \mathsf{nat}.\, \mathbf{s}(p(z)\ y) \end{array}\right.\right)\ t) \\
& \Longrightarrow_\beta & \mathbf{s}((\lambda y \in \mathsf{nat}.\, y)\ t) \\
& \Longrightarrow_\beta & \mathbf{s}(t)
\end{array}$$

Exercise 6.3. Design a function *plus* of datatype $\mathsf{nat} \to \mathsf{nat} \to \mathsf{nat}$ multiplying two natural numbers.

It is important to note that we have *derived*, as opposed to *designed*, the rule natE from the inductive definition of terms for datatype nat. In general, once we design introduction rules for a datatype so as to obtain an inductive definition of terms, the principle of primitive recursion determines a unique elimination rule. Below we consider another example of a datatype in order to further elucidate the process of deriving an elimination rule from introduction rules.

We use $\mathsf{list}\ \tau$ as a datatype for lists of terms of datatype τ:

$$\text{datatype} \quad \tau \quad ::= \quad \cdots \mid \mathsf{list}\ \tau$$

We use $\mathbf{nil}^\tau$ for an empty list of datatype $\mathsf{list}\ \tau$ and $t :: s$ for a list consisting of a head element t and a tail list s:

$$\frac{}{\mathbf{nil}^\tau \in \mathsf{list}\ \tau}\ \mathsf{listI_n} \qquad \frac{t \in \tau \quad s \in \mathsf{list}\ \tau}{t :: s \in \mathsf{list}\ \tau}\ \mathsf{listI_c}$$

From these introduction rules, we derive the following elimination rule based on primitive recursion:

$$\frac{t \in \mathsf{list}\ \tau \quad s_n \in \sigma \quad \begin{array}{c} \overline{x \in \tau} \quad \overline{l \in \mathsf{list}\ \tau} \quad \overline{f(l) \in \sigma} \\ \vdots \\ s_c \in \sigma \end{array}}{\mathbf{rec}\ f(t)\ \mathbf{of}\ f(\mathbf{nil}) \Rightarrow s_n \mid f(x :: l) \Rightarrow s_c \in \sigma}\ \mathsf{listE}$$

The first branch, corresponding to the rule $\mathsf{listI_n}$, uses no term variable because $\mathbf{nil}^\tau$ has no subterm. In the second branch, we use two term variables x and l because the rule $\mathsf{listI_c}$ uses two terms in its premise. As in the rule natE, we treat $f(l)$ as a term variable. Thus we obtain the following definition of terms for datatype $\mathsf{list}\ \tau$:

$$\text{term} \quad t \quad ::= \quad \cdots \mid \mathbf{nil}^\tau \mid t :: t \mid \mathbf{rec}\ f(t)\ \mathbf{of}\ f(\mathbf{nil}) \Rightarrow t \mid f(x :: x) \Rightarrow t$$

The derivation of the β-reductions and η-expansion is also similar to the case of datatype nat:

$$\begin{array}{rcl}
\mathbf{rec}\ f(\mathbf{nil}^\tau)\ \mathbf{of}\ f(\mathbf{nil}) \Rightarrow s_n \mid f(x :: l) \Rightarrow s_c & \Longrightarrow_\beta & s_n \\
\mathbf{rec}\ f(t :: t')\ \mathbf{of}\ f(\mathbf{nil}) \Rightarrow s_n \mid f(x :: l) \Rightarrow s_c & \Longrightarrow_\beta & [\mathbf{rec}\ f(t')\ \mathbf{of}\ f(\mathbf{nil}) \Rightarrow s_n \mid f(x :: l) \Rightarrow s_c / f(l)][t'/l][t/x]s_c \\
t \in \mathsf{list}\ \tau & \Longrightarrow_\eta & \mathbf{rec}\ f(t)\ \mathbf{of}\ f(\mathbf{nil}) \Rightarrow \mathbf{nil}^\tau \mid f(x :: l) \Rightarrow x :: l
\end{array}$$

As examples of recursive functions over lists, we define a function *append* concatenating two lists and another function *length* calculating the length of a list:

$$\begin{array}{rcl}
append\ \mathbf{nil}^\tau\ t & = & t \\
append\ (x :: l)\ t & = & x :: (append\ l\ t)
\end{array}$$

$$\begin{array}{l}
append \in \mathsf{list}\ \tau \to \mathsf{list}\ \tau \to \mathsf{list}\ \tau \\
append\ =\ \lambda y \in \mathsf{list}\ \tau.\, \lambda z \in \mathsf{list}\ \tau.\, \mathbf{rec}\ f(y)\ \mathbf{of}\ f(\mathbf{nil}) \Rightarrow z \mid f(x :: l) \Rightarrow x :: f(l)
\end{array}$$

$$\begin{array}{rcl}
length\ \mathbf{nil}^\tau & = & \mathbf{0} \\
length\ (x :: l) & = & \mathbf{s}(length\ l)
\end{array}$$

$$\begin{array}{l}
length \in \mathsf{list}\ \tau \to \mathsf{nat} \\
length\ =\ \lambda y \in \mathsf{list}\ \tau.\, \mathbf{rec}\ f(y)\ \mathbf{of}\ f(\mathbf{nil}) \Rightarrow \mathbf{0} \mid f(x :: l) \Rightarrow \mathbf{s}(f(l))
\end{array}$$

$$\dfrac{\begin{array}{c}\overline{x \in \tau}\\ \vdots \\ A(x)\ \mathit{true}\end{array}}{\forall x \in \tau. A(x)\ \mathit{true}}\ \forall\mathsf{I} \qquad \dfrac{\forall x\in\tau.A(x)\ \mathit{true} \quad t \in \tau}{A(t)\ \mathit{true}}\ \forall\mathsf{E}$$

$$\dfrac{t\in\tau \quad A(t)\ \mathit{true}}{\exists x\in\tau.A(x)\ \mathit{true}}\ \exists\mathsf{I} \qquad \dfrac{\exists x\in\tau.A(x)\ \mathit{true} \qquad \begin{array}{c}\overline{x\in\tau}\quad \overline{A(x)\ \mathit{true}}^{\,w}\\ \vdots\\ C\ \mathit{true}\end{array}}{C\ \mathit{true}}\ \exists\mathsf{E}^w$$

Figure 6.2: Natural deduction system for first-order logic with datatypes

6.4 First-order logic with datatypes

So far, we have designed a system for creating terms and specifying their datatypes. As our study focuses on logic rather than programming languages, we are ultimately interested in proving properties of terms rather than in manipulating terms. For example, the goal of designing a system for natural numbers is not to demonstrate how to multiply two natural numbers, but to formally prove such properties as that every non-zero natural number is a product of two prime numbers.

In order to state, let alone prove, interesting properties of terms, we need appropriate predicates. For example, we may need a predicate $LT(m, n)$ to state that a natural number m is less than another natural number n. We also need universal and existential quantifications so as to express that a property holds for all terms of a specific datatype, or that there exists a certain term satisfying a given property. For example, we may use $\forall x\in\mathsf{nat}.LT(x, \mathsf{s}(x))$ to state that for every natural number is less than its successor, or $\exists x\in\mathsf{nat}.LT(x, \mathsf{s}(\mathbf{0}))$ to state that there exists a natural number less than one. Then constructing proofs of judgments $\forall x\in\mathsf{nat}.LT(x, \mathsf{s}(x))\ \mathit{true}$ and $\exists x\in\mathsf{nat}.LT(x, \mathsf{s}(\mathbf{0}))\ \mathit{true}$ amounts to formally proving these properties.

Before we investigate how to define predicates, we consider universal and existential quantifications in the presence of datatypes. The development is similar to the case of pure first-order logic except that we now specify the datatype for each term variable.

The inductive definition of propositions is extended as follows:

$$\text{proposition} \quad A \quad ::= \quad \cdots \mid \forall x\in\tau.A(x) \mid \exists x\in\tau.A(x)$$

The formation rules for $\forall x\in\tau.A$ and $\exists x\in\tau.A$ use a hypothesis of $x \in \tau$:

$$\dfrac{\begin{array}{c}\overline{x \in \tau}\\ \vdots \\ A(x)\ \mathit{prop}\end{array}}{\forall x \in \tau. A(x)\ \mathit{prop}}\ \forall\mathsf{F} \qquad \dfrac{\begin{array}{c}\overline{x \in \tau}\\ \vdots \\ A(x)\ \mathit{prop}\end{array}}{\exists x \in \tau. A(x)\ \mathit{prop}}\ \exists\mathsf{F}$$

As we may freely rename x in $\forall x\in\tau.A(x)$ and $\exists x\in\tau.A(x)$ to a different term variable, we assume that term variables declared in universal and existential quantifications are all distinct.

Figure 6.2 shows the inference rules for first-order logic with datatypes. These inference rules have two important differences from those in pure first-order logic. First the rules $\forall$I and $\exists$E no longer replace a term variable x by a fresh parameter a as in $[a/x]A\ \mathit{true}$, but use a hypothesis of $x \in \tau$ specifying the datatype for x. Second the rules $\forall$E and $\exists$I require a separate judgment $t \in \tau$.

The rule $\forall$I resembles the rule $\supset$I in that it uses a hypothesis whose scope is local to the premise. The difference from the rule $\supset$I is that the meaning of proposition $A(x)$ in the premise is dependent on the meaning of term variable x in the hypothesis. In contrast, in the rule $\supset$I for proving $A \supset B\ \mathit{true}$, the meaning of proposition B is independent of the meaning of proposition A. Hence, if we collapse the distinction between types and datatypes and use $x \in A$ instead of $x : A$, an implication $A \supset B$ becomes a special case of a universal quantification $\forall x\in A.B$ where B contains no occurrence of x. Then the rule $\supset$E also becomes a special case of the rule $\forall$E.

$$
\frac{\begin{array}{c}\overline{x \in \tau}\\ \vdots \\ M : A(x)\end{array}}{\lambda x \in \tau.\, M : \forall x{\in}\tau.A(x)}\ \forall\mathsf{I}
\qquad
\frac{M : \forall x{\in}\tau.A(x) \quad t \in \tau}{M\ t : A(t)}\ \forall\mathsf{E}
$$

$$
\frac{t \in \tau \quad M : A(t)}{\langle t, M\rangle : \exists x{\in}\tau.A(x)}\ \exists\mathsf{I}
\qquad
\frac{M : \exists x{\in}\tau.A(x) \qquad \begin{array}{c}\overline{x \in \tau} \quad \overline{w : A(x)}\\ \vdots \\ N : C\end{array}}{\mathsf{let}\ \langle x, w\rangle = M\ \mathsf{in}\ N : C}\ \exists\mathsf{E}
$$

Figure 6.3: Typing rules for proof terms in first-order logic with datatypes

Proof terms are the same as in pure first-order logic except that we use λ-abstractions that explicitly specify the datatype of term variables:

$$
\text{proof term} \quad M \quad ::= \quad \cdots \mid \lambda x \in \tau.\, M \mid M\ t \mid \langle t, M\rangle \mid \mathsf{let}\ \langle x, w\rangle = M\ \mathsf{in}\ M
$$

Figure 6.3 shows the typing rules for these proof terms. The typing rules $\forall\mathsf{I}$ and $\forall\mathsf{E}$ show that it is no coincidence that we use λ-abstractions and λ-applications as proofs terms for universal quantifications as well as for implications, since implications are essentially a special case of universal quantifications.

Here are a few examples of proof terms whose types involve universal or existential quantifications:

- A proof term of type $(\forall x{\in}\tau.A(x) \wedge B(x)) \supset \forall x{\in}\tau.A(x)$ is

  $$\lambda z{:}\forall x{\in}\tau.A(x) \wedge B(x).\, \lambda x \in \tau.\, \mathsf{fst}\ (z\ x).$$

  Note that z is a variable whereas x is a term variable.

- A proof term of type $(\exists x{\in}\tau.A(x) \vee B(x)) \supset ((\exists x{\in}\tau.A(x)) \vee (\exists x{\in}\tau.B(x)))$ is

  $$\lambda z{:}\exists x{\in}\tau.A(x) \vee B(x).\, \mathsf{let}\ \langle x, w\rangle = z\ \mathsf{in}\ \mathsf{case}\ w\ \mathsf{of}\ \mathsf{inl}\ y_1.\, \mathsf{inl}_{\exists x \in \tau.B(x)}\ \langle x, y_1\rangle \mid \mathsf{inr}\ y_2.\, \mathsf{inr}_{\exists x \in \tau.A(x)}\ \langle x, y_2\rangle.$$

- A proof term of type $((\exists x{\in}\tau.A(x)) \vee (\exists x{\in}\tau.B(x))) \supset (\exists x{\in}\tau.A(x) \vee B(x))$ is

  $$\begin{array}{l}\lambda z{:}(\exists x{\in}\tau.A(x)) \vee (\exists x{\in}\tau.B(x)).\\ \quad \mathsf{case}\ z\ \mathsf{of}\ \mathsf{inl}\ y_1.\, \mathsf{let}\ \langle x, w\rangle = y_1\ \mathsf{in}\ \langle x, \mathsf{inl}_{B(x)}\ w\rangle \mid \mathsf{inr}\ y_2.\, \mathsf{let}\ \langle x, w\rangle = y_2\ \mathsf{in}\ \langle x, \mathsf{inr}_{A(x)}\ w\rangle\end{array}.$$

The β-reduction and η-expansion for universal quantifications

$$
\begin{array}{rcll}
(\lambda x \in \tau.\, M)\ t & \Longrightarrow_\beta & [t/x]M & \\
M : \forall x{\in}\tau.A(x) & \Longrightarrow_\eta & \lambda x \in \tau.\, M\ x & (x \textit{ is not free in } M)
\end{array}
$$

are obtained as follows:

$$
\frac{\dfrac{\begin{array}{c}\overline{x \in \tau}\\ \vdots \\ M : A(x)\end{array}}{\lambda x \in \tau.\, M : \forall x{\in}\tau.A(x)}\ \forall\mathsf{I} \quad t \in \tau}{(\lambda x \in \tau.\, M)\ t : A(t)}\ \exists\mathsf{E}
\quad \Longrightarrow_\beta \quad
\begin{array}{c} t \in \tau \\ \vdots \\ [t/x]M : A(t)\end{array}
$$

$$
M : \forall x{\in}\tau.A(x) \quad \Longrightarrow_\eta \quad
\frac{\dfrac{M : \forall x{\in}\tau.A \quad \overline{x \in \tau}}{M\ x : A(x)}\ \forall\mathsf{E}}{\lambda x \in \tau.\, M\ x : \forall x{\in}\tau.A(x)}\ \forall\mathsf{I}
$$

The β-reduction and η-expansion for existential quantifications

$$
\begin{array}{rcl}
\mathsf{let}\ \langle x, w\rangle = \langle t, M\rangle\ \mathsf{in}\ N & \Longrightarrow_\beta & [M/w][t/x]N \\
M : \exists x{\in}\tau.A(x) & \Longrightarrow_\eta & \mathsf{let}\ \langle x, w\rangle = M\ \mathsf{in}\ \langle x, w\rangle
\end{array}
$$

are obtained as follows:

$$\dfrac{\dfrac{t \in \tau \quad M : A(t)}{\langle t, M\rangle : \exists x \!\in\! \tau. A(x)}\,\exists\mathsf{I} \qquad \begin{array}{cc} \overline{x \in \tau} & \overline{w : A(x)} \\ & \vdots \\ & N : C \end{array}}{\mathsf{let}\ \langle x, w\rangle = \langle t, M\rangle\ \mathsf{in}\ N : C}\,\exists\mathsf{E} \quad \Longrightarrow_{\beta} \quad \begin{array}{cc} t \in \tau & \overline{[M/w]w : A(t)} \\ & \vdots \\ & [M/w][t/x]N : C \end{array}$$

$$M : \exists x \!\in\! \tau. A(x) \quad \Longrightarrow_{\eta} \quad \dfrac{M : \exists x \!\in\! \tau. A(x) \qquad \dfrac{\overline{x \in \tau} \quad \overline{w : A(x)}}{\langle x, w\rangle : \exists x \!\in\! \tau. A(x)}\,\exists\mathsf{I}}{\mathsf{let}\ \langle x, w\rangle = M\ \mathsf{in}\ \langle x, w\rangle : \exists x \!\in\! \tau. A}\,\exists\mathsf{E}$$

6.5 Natural deduction for predicates

We now investigate how to define predicates on terms. As with all the logical connectives in first-order logic, we base the definition of every predicate on the principle of natural deduction. That is, we use an introduction rule to deduce a new judgment involving the predicate, and an elimination rule to exploit an existing judgment involving the predicate. Moreover, as in the definition of such datatypes as bool and nat, we first *design* introduction rules to characterize the predicate and then *derive* elimination rules from these introduction rules.

As a running example, we define a predicate $LT(m, n)$ to mean a natural number m is less than another natural number n. We abbreviate $LT(m, n)$ as $m < n$.

$$\text{proposition} \quad A \quad ::= \quad \cdots \mid m < n$$

The formation rule for $m < n$ requires both m and n to be of datatype nat:

$$\dfrac{m \in \mathsf{nat} \quad n \in \mathsf{nat}}{m < n\ \mathit{prop}}\,{<}\mathsf{F}$$

By the rule $<$F, every judgment $m < n\ \mathit{true}$ implicitly assumes that both m and n are of datatype nat.

We use the following introduction rules:

$$\dfrac{}{\mathbf{0} < \mathbf{s}(n)\ \mathit{true}}\,{<}\mathsf{I}_0 \qquad \dfrac{m < n\ \mathit{true}}{\mathbf{s}(m) < \mathbf{s}(n)\ \mathit{true}}\,{<}\mathsf{I}_{\mathsf{s}}$$

The rule $<\mathsf{I}_0$ states that $\mathbf{0}$ is less than the successor of any natural number; the rule $<\mathsf{I}_{\mathsf{s}}$ states that proving $\mathbf{s}(m) < \mathbf{s}(n)\ \mathit{true}$ reduces to proving $m < n\ \mathit{true}$. Now the two introduction rules determine a unique meaning for $<$ which is a comparison relation applicable to any pair of natural numbers. If we choose to include the rule $<\mathsf{I}_0$ but omit $<\mathsf{I}_{\mathsf{s}}$, we obtain a different but still valid meaning for $<$ which is a relation testing whether a given natural number is greater than zero or not. Thus the two introduction rules provide just a specific way to characterize $<$, which can be defined in many different ways.

In order to derive elimination rules, we consider four possible cases of the judgment $m < n\ \mathit{true}$:

- $\mathbf{0} < \mathbf{0}\ \mathit{true}$ is impossible to prove. The corresponding elimination rule may deduce any judgment $C\ \mathit{true}$.
- $\mathbf{s}(m) < \mathbf{0}\ \mathit{true}$ is impossible to prove. The corresponding elimination rule may deduce any judgment $C\ \mathit{true}$.
- $\mathbf{0} < \mathbf{s}(n)\ \mathit{true}$ holds trivially by the rule $<\mathsf{I}_0$ whose premise is empty. Hence there is no corresponding elimination rule.
- $\mathbf{s}(m) < \mathbf{s}(n)\ \mathit{true}$ holds by the rule $<\mathsf{I}_{\mathsf{s}}$ whose premise is $m < n\ \mathit{true}$. Thus the corresponding elimination rule deduces $m < n\ \mathit{true}$.

We combine the first two cases to obtain a single elimination rule:

$$\dfrac{m < \mathbf{0}\ \mathit{true}}{C\ \mathit{true}}\,{<}\mathsf{E}_0 \qquad \dfrac{\mathbf{s}(m) < \mathbf{s}(n)\ \mathit{true}}{m < n\ \mathit{true}}\,{<}\mathsf{E}_{\mathsf{s}}$$

Note that the rules $<\mathsf{I_s}$ and $<\mathsf{E_s}$, which have $<$ in both the conclusion and the premise, do not destroy the orthogonality of the system because $<$ is not a logical connective but a predicate. If $<$ were a logical connective, we lose the orthogonality of the system because we explain the meaning of $<$ using $<$ itself.

Here are two examples of proofs using the rules for $<$:

$$\dfrac{\dfrac{}{\mathbf{0} < \mathbf{s}(\mathbf{0})\ \mathit{true}}\ {<}\mathsf{I_0}}{\mathbf{s}(\mathbf{0}) < \mathbf{s}(\mathbf{s}(\mathbf{0}))\ \mathit{true}}\ {<}\mathsf{I_s} \qquad \dfrac{\dfrac{\dfrac{}{m < \mathbf{0}\ \mathit{true}}\ z}{\perp\ \mathit{true}}\ {<}\mathsf{E_0}}{\neg(m < \mathbf{0})\ \mathit{true}}\ \neg\mathsf{I}^z$$

We can also represent a proof of $m < n$ *true* as a proof term of type $m < n$. Note that we refer to $m < n$ as a "type," which is nothing strange because propositions and types are equivalent under the Curry-Howard isomorphism. We use the following definition of proof terms each of which corresponds to an inference rule as shown below:

$$\text{proof term} \quad M \quad ::= \quad \cdots \mid \mathsf{ltI_0} \mid \mathsf{ltI_s}(M) \mid \mathsf{ltE_0}(M) \mid \mathsf{ltE_s}(M)$$

$$\dfrac{}{\mathsf{ltI_0} : \mathbf{0} < \mathbf{s}(n)}\ {<}\mathsf{I_0} \qquad \dfrac{M : m < n}{\mathsf{ltI_s}(M) : \mathbf{s}(m) < \mathbf{s}(n)}\ {<}\mathsf{I_s} \qquad \dfrac{M : m < \mathbf{0}}{\mathsf{ltE_0}(M) : C}\ {<}\mathsf{E_0} \qquad \dfrac{M : \mathbf{s}(m) < \mathbf{s}(n)}{\mathsf{ltE_s}(M) : m < n}\ {<}\mathsf{E_s}$$

Here are proof terms of types $\mathbf{s}(\mathbf{0}) < \mathbf{s}(\mathbf{s}(\mathbf{0}))$ and $\neg(m < \mathbf{0})$:

$$\dfrac{\dfrac{}{\mathsf{ltI_0} : \mathbf{0} < \mathbf{s}(\mathbf{0})}\ {<}\mathsf{I_0}}{\mathsf{ltI_s}(\mathsf{ltI_0}) : \mathbf{s}(\mathbf{0}) < \mathbf{s}(\mathbf{s}(\mathbf{0}))}\ {<}\mathsf{I_s} \qquad \dfrac{\dfrac{\dfrac{}{z : m < \mathbf{0}}\ z}{\mathsf{ltE_0}(z) : \perp}\ {<}\mathsf{E_0}}{\lambda z\!:\!m < \mathbf{0}.\, \mathsf{ltE_0}(z) : \neg(m < \mathbf{0})}\ {\supset}\mathsf{I}$$

As another example, we consider a predicate $EQ(m, n)$ to mean that natural numbers m and n are equal. We abbreviate $EQ(m, n)$ as $m =_{\mathsf{N}} n$.

$$\text{proposition} \quad A \quad ::= \quad \cdots \mid m =_{\mathsf{N}} n$$

$$\dfrac{m \in \mathsf{nat} \quad n \in \mathsf{nat}}{m =_{\mathsf{N}} n\ \mathit{prop}}\ {=_{\mathsf{N}}}\mathsf{F}$$

Note that $m =_{\mathsf{N}} n$, which says that m and n represent the same natural number, is different from $m = n$, which says that m and n are syntactically identical.

Similarly to the predicate $m < n$, we use two introduction rules:

$$\dfrac{}{\mathbf{0} =_{\mathsf{N}} \mathbf{0}\ \mathit{true}}\ {=_{\mathsf{N}}}\mathsf{I_0} \qquad \dfrac{m =_{\mathsf{N}} n\ \mathit{true}}{\mathbf{s}(m) =_{\mathsf{N}} \mathbf{s}(n)\ \mathit{true}}\ {=_{\mathsf{N}}}\mathsf{I_s}$$

From these introduction rules, we derive the following elimination rules:

$$\dfrac{\mathbf{0} =_{\mathsf{N}} \mathbf{s}(n)\ \mathit{true}}{C\ \mathit{true}}\ {=_{\mathsf{N}}}\mathsf{E_{0s}} \qquad \dfrac{\mathbf{s}(m) =_{\mathsf{N}} \mathbf{0}\ \mathit{true}}{C\ \mathit{true}}\ {=_{\mathsf{N}}}\mathsf{E_{s0}} \qquad \dfrac{\mathbf{s}(m) =_{\mathsf{N}} \mathbf{s}(n)\ \mathit{true}}{m =_{\mathsf{N}} n\ \mathit{true}}\ {=_{\mathsf{N}}}\mathsf{E_s}$$

There is no elimination rule for $\mathbf{0} =_{\mathsf{N}} \mathbf{0}$ *true* because the premise of the rule $=_{\mathsf{N}}\mathsf{I_0}$ is empty.

We use the following definition of proof terms for the predicate $m =_{\mathsf{N}} n$:

$$\text{proof term} \quad M \quad ::= \quad \cdots \mid \mathsf{eqI_0} \mid \mathsf{eqI_s}(M) \mid \mathsf{eqE_{0s}}(M) \mid \mathsf{eqE_{s0}}(M) \mid \mathsf{eqE_s}(M)$$

$$\dfrac{}{\mathsf{eqI_0} : \mathbf{0} =_{\mathsf{N}} \mathbf{0}}\ {=_{\mathsf{N}}}\mathsf{I_0} \qquad \dfrac{M : m =_{\mathsf{N}} n}{\mathsf{eqI_s}(M) : \mathbf{s}(m) =_{\mathsf{N}} \mathbf{s}(n)}\ {=_{\mathsf{N}}}\mathsf{I_s}$$

$$\dfrac{M : \mathbf{0} =_{\mathsf{N}} \mathbf{s}(n)}{\mathsf{eqE_{0s}}(M) : C}\ {=_{\mathsf{N}}}\mathsf{E_{0s}} \qquad \dfrac{M : \mathbf{s}(m) =_{\mathsf{N}} \mathbf{0}}{\mathsf{eqE_{s0}}(M) : C}\ {=_{\mathsf{N}}}\mathsf{E_{s0}} \qquad \dfrac{M : \mathbf{s}(m) =_{\mathsf{N}} \mathbf{s}(n)}{\mathsf{eqE_s}(M) : m =_{\mathsf{N}} n}\ {=_{\mathsf{N}}}\mathsf{E_s}$$

Now that we have a couple of predicates, we may attempt to prove interesting properties of natural numbers in conjunction with universal and existential quantifications. For example, we may attempt to prove that for any natural number x, there exists a natural number y such that $x < y$, for example, by choosing $y = \mathbf{s}(x)$. A formal proof of $\forall x \in \mathsf{nat}.\exists y \in \mathsf{nat}.x < y\ \mathit{true}$, however, is not so simple:

$$\dfrac{\dfrac{\dfrac{\overline{x \in \mathsf{nat}}}{\mathbf{s}(x) \in \mathsf{nat}}\ \mathsf{natI_s} \qquad \begin{array}{c} ??? \\ x < \mathbf{s}(x)\ \mathit{true} \end{array}}{\exists y \in \mathsf{nat}.x < y\ \mathit{true}}}{\forall x \in \mathsf{nat}.\exists y \in \mathsf{nat}.x < y\ \mathit{true}}\ \forall\mathsf{I}$$

In fact, we cannot prove even such a simple judgment $\forall x \in \mathsf{nat}.x =_{\mathsf{N}} x$. Intuitively we have to prove an infinite number of judgments $\mathbf{0} =_{\mathsf{N}} \mathbf{0}$, $\mathbf{s}(\mathbf{0}) =_{\mathsf{N}} \mathbf{s}(\mathbf{0})$, $\mathbf{s}(\mathbf{s}(\mathbf{0})) =_{\mathsf{N}} \mathbf{s}(\mathbf{s}(\mathbf{0}))$, and so on, but we do not have a mechanism by which we represent all these proofs as a single proof of finite size.

In the next section, we introduce yet another form of elimination rule for datatypes which provides such a mechanism.

6.6 Induction on terms

Suppose that we wish to prove $A(x)\ \mathit{true}$ for every boolean value x. Since there are only two terms **true** and **false**, it suffices to prove $A(\mathbf{true})\ \mathit{true}$ and $A(\mathbf{false})\ \mathit{true}$ separately, which is expressed in the following elimination rule for datatype bool:

$$\dfrac{t \in \mathsf{bool} \quad A(\mathbf{true})\ \mathit{true} \quad A(\mathbf{false})\ \mathit{true}}{A(t)\ \mathit{true}}\ \mathsf{boolE}_I$$

Note that unlike the previous elimination rule boolE which deduces only a judgment of the form $s \in \tau$, the rule boolE_I exploits a proof of $t \in \mathsf{bool}$ to deduce a judgment $A(t)\ \mathit{true}$ where $A(t)$ can be any proposition involving t. Thus we have derived a new form of elimination rule which connects different forms of judgments.

Now suppose that we wish to prove $A(x)\ \mathit{true}$ for every natural number x. Since there are an infinite sequence of natural numbers, a naive approach similar to the case of datatype bool would be clearly infeasible:

$$\dfrac{t \in \mathsf{nat} \quad A(\mathbf{0})\ \mathit{true} \quad A(\mathbf{s}(\mathbf{0}))\ \mathit{true} \quad A(\mathbf{s}(\mathbf{s}(\mathbf{0})))\ \mathit{true} \quad \cdots}{A(t)\ \mathit{true}}\ \mathsf{natE}_I$$

Thus we are led to derive an elimination rule that allows mathematical induction on natural numbers inside a proof. Specifically it needs to show that $A(\mathbf{0})\ \mathit{true}$ holds and that an induction hypothesis $A(x)\ \mathit{true}$ implies $A(\mathbf{s}(x))\ \mathit{true}$:

$$\dfrac{t \in \mathsf{nat} \quad A(\mathbf{0})\ \mathit{true} \qquad \begin{array}{c} \overline{x \in \mathsf{nat}} \quad \overline{A(x)\ \mathit{true}}^{\,u(x)} \\ \vdots \\ A(\mathbf{s}(x))\ \mathit{true} \end{array}}{A(t)\ \mathit{true}}\ \mathsf{natE}_I^{u(x)}$$

The second premise states that $A(x)\ \mathit{true}$ holds for $x = \mathbf{0}$, and corresponds to the base case in mathematical induction. The third premise states that a hypothesis of $A(x)\ \mathit{true}$ (with label $u(x)$) leads to a proof of $A(\mathbf{s}(x))\ \mathit{true}$, and corresponds to the inductive case in mathematical induction. Hence the second and third premises constitute a valid proof of $A(x)\ \mathit{true}$ for every natural number x. Note that the first premise just provides a specific natural number t which is to be substituted for x in $A(x)\ \mathit{true}$ and is thus not essential in completing a proof by mathematical induction. Often t is just a term variable, in which case it is called an *induction variable*.

Using the new elimination rule, we can now complete the proof of $\forall x \in \mathsf{nat}.\exists y \in \mathsf{nat}.x < y\ \mathit{true}$. In the proof shown below, we use x as an induction variable and let $A(x) = x < \mathbf{s}(x)$ in the rule $\mathsf{natE}_I^{u(x)}$:

$$\dfrac{\dfrac{\dfrac{\overline{x \in \mathsf{nat}}}{\mathbf{s}(x) \in \mathsf{nat}}\ \mathsf{natI_s} \qquad \dfrac{\overline{x \in \mathsf{nat}} \quad \dfrac{}{\mathbf{0} < \mathbf{s}(\mathbf{0})\ \mathit{true}}\ {<}\mathsf{I_0} \quad \dfrac{\overline{x < \mathbf{s}(x)\ \mathit{true}}^{\,u(x)}}{\mathbf{s}(x) < \mathbf{s}(\mathbf{s}(x))\ \mathit{true}}\ {<}\mathsf{I_s}}{x < \mathbf{s}(x)\ \mathit{true}}\ \mathsf{natE}_I^{u(x)}}{\exists y \in \mathsf{nat}.x < y\ \mathit{true}}}{\forall x \in \mathsf{nat}.\exists y \in \mathsf{nat}.x < y\ \mathit{true}}\ \forall\mathsf{I}$$

Generalizing the case of datatype nat, we can derive from the definition of a datatype, or from its introduction rules, an elimination rule that is based on *induction on terms* and builds inductive proofs on terms. For example, the definition of datatype $\mathsf{list}\ \tau$ results in the following elimination rule:

$$\dfrac{t \in \mathsf{list}\ \tau \quad A(\mathbf{nil}^{\tau})\ \mathit{true} \quad \begin{array}{c}\overline{x \in \tau}\quad \overline{l \in \mathsf{list}\ \tau}\quad \overline{A(l)\ \mathit{true}}^{\,u(l)} \\ \vdots \\ A(x :: l)\ \mathit{true}\end{array}}{A(t)\ \mathit{true}}\ \mathsf{list}\,\mathsf{E}_I^{u(l)}$$

We can also devise proof terms for the new elimination rules. For example, we use the following proof term for the rule natE_I:

$$\text{proof term} \quad M \quad ::= \quad \cdots \mid \mathsf{ind}\ u(t)\ \mathsf{of}\ u(\mathbf{0}) \Rightarrow M \mid u(\mathbf{s}(x)) \Rightarrow N$$

$$\dfrac{t \in \mathsf{nat} \quad M : A(\mathbf{0}) \quad \begin{array}{c}\overline{x \in \mathsf{nat}}\quad \overline{u(x) : A(x)} \\ \vdots \\ N : A(\mathbf{s}(x))\end{array}}{\mathsf{ind}\ u(t)\ \mathsf{of}\ u(\mathbf{0}) \Rightarrow M \mid u(\mathbf{s}(x)) \Rightarrow N : A(t)}\ \mathsf{natE}_I$$

We can think of $\mathsf{ind}\ u(t)\ \mathsf{of}\ u(\mathbf{0}) \Rightarrow M \mid u(\mathbf{s}(x)) \Rightarrow N$ as an *inductive function* u applied to t. If N does not use $u(x)$, it degenerates to a case analysis construct and may be written as $\mathsf{case}\ t\ \mathsf{of}\ \mathbf{0} \Rightarrow M \mid \mathbf{s}(x) \Rightarrow N$:

$$\text{proof term} \quad M \quad ::= \quad \cdots \mid \mathsf{case}\ t\ \mathsf{of}\ \mathbf{0} \Rightarrow M \mid \mathbf{s}(x) \Rightarrow N$$

$$\dfrac{t \in \mathsf{nat} \quad M : A(\mathbf{0}) \quad \begin{array}{c}\overline{x \in \mathsf{nat}} \\ \vdots \\ N : A(\mathbf{s}(x))\end{array}}{\mathsf{case}\ t\ \mathsf{of}\ \mathbf{0} \Rightarrow M \mid \mathbf{s}(x) \Rightarrow N : A(t)}\ \mathsf{natE}_I$$

As an example, here is a proof term of type $\forall x \in \mathsf{nat}. \exists y \in \mathsf{nat}. x < y$:

$$\dfrac{\dfrac{\dfrac{\overline{x \in \mathsf{nat}}}{\mathbf{s}(x) \in \mathsf{nat}}\ \mathsf{natI_s} \quad \dfrac{\overline{x \in \mathsf{nat}} \quad \dfrac{}{\mathsf{ltI_0} : \mathbf{0} < \mathbf{s}(\mathbf{0})}\ {<}\mathsf{I_0} \quad \dfrac{\overline{u(x) : x < \mathbf{s}(x)}}{\mathsf{ltI_s}(u(x)) : \mathbf{s}(x) < \mathbf{s}(\mathbf{s}(x))}\ {<}\mathsf{I_s}}{\mathsf{ind}\ u(t)\ \mathsf{of}\ u(\mathbf{0}) \Rightarrow \mathsf{ltI_0} \mid u(\mathbf{s}(x)) \Rightarrow \mathsf{ltI_s}(u(x)) : x < \mathbf{s}(x)}\ \mathsf{natE}_I}{\langle \mathbf{s}(x), \mathsf{ind}\ u(t)\ \mathsf{of}\ u(\mathbf{0}) \Rightarrow \mathsf{ltI_0} \mid u(\mathbf{s}(x)) \Rightarrow \mathsf{ltI_s}(u(x))\rangle : \exists y \in \mathsf{nat}. x < y}}{\lambda x \in \mathsf{nat}.\ \langle \mathbf{s}(x), \mathsf{ind}\ u(t)\ \mathsf{of}\ u(\mathbf{0}) \Rightarrow \mathsf{ltI_0} \mid u(\mathbf{s}(x)) \Rightarrow \mathsf{ltI_s}(u(x))\rangle : \exists y \in \mathsf{nat}. x < y : \forall x \in \mathsf{nat}. \exists y \in \mathsf{nat}. x < y}\ \forall\mathsf{I}$$

An elimination rule based on induction on terms gives rise to new β-reductions. For example, an introduction rule $\mathsf{natI_0}$ or $\mathsf{natI_s}$ (proving $t \in \mathsf{nat}$) followed by the elimination rule natE_I (proving $A(t)\ \mathit{true}$) forms a new pattern of detour, and removing such a detour corresponds to a β-reduction of a term of type $A(t)$. In the case of datatype nat, we obtain the following β-reductions:

$$\begin{array}{rcl}
\mathsf{ind}\ u(\mathbf{0})\ \mathsf{of}\ u(\mathbf{0}) \Rightarrow M \mid u(\mathbf{s}(x)) \Rightarrow N & \Longrightarrow_{\beta} & M \\
\mathsf{ind}\ u(\mathbf{s}(t))\ \mathsf{of}\ u(\mathbf{0}) \Rightarrow M \mid u(\mathbf{s}(x)) \Rightarrow N & \Longrightarrow_{\beta} & [\mathsf{ind}\ u(t)\ \mathsf{of}\ u(\mathbf{0}) \Rightarrow M \mid u(\mathbf{s}(x)) \Rightarrow N/u(x)][t/x]N
\end{array}$$

In the second β-reduction where $\mathbf{s}(t)$ matches $\mathbf{s}(x)$, we replace $u(x)$ in N by a proof term of type $A(t)$, namely $\mathsf{ind}\ u(t)\ \mathsf{of}\ u(\mathbf{0}) \Rightarrow M \mid u(\mathbf{s}(x)) \Rightarrow N$.

Note that elimination rules based on induction on terms are irrelevant to η-expansions. For example, an η-expansion of $t \in \mathsf{nat}$ must return another term of datatype nat, but the rule natE_I yields a proof term instead of a term. That is, the rule natE_I eliminates a judgment $t \in \mathsf{nat}$ to produce an incompatible judgment $M : A(t)$.

6.7 Examples

We have seen in Section 6.5 that the introduction rules for a predicate specify a unique set of elimination rules. For example, the introduction rules for the predicate $m =_{\mathsf{N}} n$

$$\frac{}{\mathbf{0} =_{\mathsf{N}} \mathbf{0}\ \mathit{true}} =_{\mathsf{N}}\mathsf{I}_0 \qquad \frac{m =_{\mathsf{N}} n\ \mathit{true}}{\mathbf{s}(m) =_{\mathsf{N}} \mathbf{s}(n)\ \mathit{true}} =_{\mathsf{N}}\mathsf{I}_{\mathsf{s}}$$

specify the following elimination rules:

$$\frac{\mathbf{0} =_{\mathsf{N}} \mathbf{s}(n)\ \mathit{true}}{C\ \mathit{true}} =_{\mathsf{N}}\mathsf{E}_{0\mathsf{s}} \qquad \frac{\mathbf{s}(m) =_{\mathsf{N}} \mathbf{0}\ \mathit{true}}{C\ \mathit{true}} =_{\mathsf{N}}\mathsf{E}_{\mathsf{s}0} \qquad \frac{\mathbf{s}(m) =_{\mathsf{N}} \mathbf{s}(n)\ \mathit{true}}{m =_{\mathsf{N}} n\ \mathit{true}} =_{\mathsf{N}}\mathsf{E}_{\mathsf{s}}$$

We have also seen in Section 6.6 that the introduction rules for a datatype specify an elimination rule based on induction on terms. For example, the introduction rules for datatype nat

$$\frac{}{\mathbf{0} \in \mathsf{nat}}\ \mathsf{natI}_0 \qquad \frac{t \in \mathsf{nat}}{\mathbf{s}(t) \in \mathsf{nat}}\ \mathsf{natI}_{\mathsf{s}}$$

specify the following elimination rule:

$$\frac{t \in \mathsf{nat} \quad A(\mathbf{0})\ \mathit{true} \quad \begin{array}{c} \overline{x \in \mathsf{nat}} \quad \overline{A(x)\ \mathit{true}}^{\,u(x)} \\ \vdots \\ A(\mathbf{s}(x))\ \mathit{true} \end{array}}{A(t)\ \mathit{true}}\ \mathsf{natE}_I^{u(x)}$$

Here we consider a few examples which use these rules to prove properties of natural numbers.

Example 1. $\forall x \in \mathsf{nat}. x =_{\mathsf{N}} x$

A judgment $\forall x \in \mathsf{nat}. x =_{\mathsf{N}} x\ \mathit{true}$ states that every natural number is equal to itself. Note that the judgment does *not* hold trivially because $=_{\mathsf{N}}$ is not a syntactic equality relation but a notational abbreviation of a predicate symbol EQ such that $EQ(m, n)$ means $m =_{\mathsf{N}} n$. That is, there is no reason that $x =_{\mathsf{N}} x$ should hold just because we intend $=_{\mathsf{N}}$ as an equality relation between natural numbers.

We begin with an inductive proof of $x =_{\mathsf{N}} x\ \mathit{true}$ where x is assumed to be an arbitrary natural number:

Proof. By induction on x.

Base case $x = \mathbf{0}$:
$\mathbf{0} =_{\mathsf{N}} \mathbf{0}\ \mathit{true}$ from $\dfrac{}{\mathbf{0} =_{\mathsf{N}} \mathbf{0}\ \mathit{true}} =_{\mathsf{N}}\mathsf{I}_0$

Inductive case $x = \mathbf{s}(x')$:
$x' =_{\mathsf{N}} x'\ \mathit{true}$ by induction hypothesis
$\mathbf{s}(x') =_{\mathsf{N}} \mathbf{s}(x')\ \mathit{true}$ from $\dfrac{x' =_{\mathsf{N}} x'\ \mathit{true}}{\mathbf{s}(x') =_{\mathsf{N}} \mathbf{s}(x')\ \mathit{true}} =_{\mathsf{N}}\mathsf{I}_{\mathsf{s}}$

□

From this inductive proof, we obtain a derivation tree for the judgment $\forall x \in \mathsf{nat}. x =_{\mathsf{N}} x\ \mathit{true}$:

$$\dfrac{\dfrac{\overline{x \in \mathsf{nat}} \quad \dfrac{}{\mathbf{0} =_{\mathsf{N}} \mathbf{0}\ \mathit{true}} =_{\mathsf{N}}\mathsf{I}_0 \quad \dfrac{\overline{x' =_{\mathsf{N}} x'\ \mathit{true}}^{\,u(x')}}{\mathbf{s}(x') =_{\mathsf{N}} \mathbf{s}(x')\ \mathit{true}} =_{\mathsf{N}}\mathsf{I}_{\mathsf{s}}}{x =_{\mathsf{N}} x\ \mathit{true}}\ \mathsf{natE}_I^{u(x')}}{\forall x \in \mathsf{nat}. x =_{\mathsf{N}} x\ \mathit{true}}\ \forall\mathsf{I}$$

Then we obtain a proof term of type $\forall x \in \mathsf{nat}. x =_{\mathsf{N}} x$ by assigning a proof term to every part of the derivation tree:

$$\dfrac{\dfrac{\overline{x \in \mathsf{nat}} \qquad \dfrac{}{\mathsf{eqI_0} : \mathbf{0} =_{\mathsf{N}} \mathbf{0}}{=_{\mathsf{N}}\mathsf{I_0}} \qquad \dfrac{\overline{u(x') : x' =_{\mathsf{N}} x'}}{\mathsf{eqI_s}(u(x')) : \mathbf{s}(x') =_{\mathsf{N}} \mathbf{s}(x')}{=_{\mathsf{N}}\mathsf{I_s}}}{\mathsf{ind}\ u(x)\ \mathsf{of}\ u(\mathbf{0}) \Rightarrow \mathsf{eqI_0} \mid u(\mathbf{s}(x')) \Rightarrow \mathsf{eqI_s}(u(x')) : x =_{\mathsf{N}} x}\mathsf{natE}_I^{u(x')}}{\lambda x \in \mathsf{nat}.\, \mathsf{ind}\ u(x)\ \mathsf{of}\ u(\mathbf{0}) \Rightarrow \mathsf{eqI_0} \mid u(\mathbf{s}(x')) \Rightarrow \mathsf{eqI_s}(u(x')) : \forall x \in \mathsf{nat}. x =_{\mathsf{N}} x}\forall\mathsf{I}$$

An equivalent but easier way to obtain such a proof term is to begin with its specification. For example, we can derive a proof term $eqNat$ of type $\forall x \in \mathsf{nat}. x =_{\mathsf{N}} x$ from the specification that $eqNat\ x$ returns a proof term of type $x =_{\mathsf{N}} x$:

$$\begin{array}{llcl} & x & & \text{proof term of type } x =_{\mathsf{N}} x \\ eqNat & \mathbf{0} & = & \mathsf{eqI_0} \\ eqNat & \mathbf{s}(x') & = & \mathsf{eqI_s}(eqNat\ x') \end{array}$$

Note that $eqNat$ may be recursively called only with argument x', just like a primitive recursive function applied to $\mathbf{s}(x')$ may be recursively called only with argument x'. Then we introduce an inductive function u and rewrite the specification into the definition of $eqNat$ where $eqNat\ x'$ changes to $u(x')$:

$$eqNat \;=\; \lambda x \in \mathsf{nat}.\, \mathsf{ind}\ u(x)\ \mathsf{of}\ u(\mathbf{0}) \Rightarrow \mathsf{eqI_0} \mid u(\mathbf{s}(x')) \Rightarrow \mathsf{eqI_s}(u(x'))$$

Example 2. $\forall x \in \mathsf{nat}. \forall y \in \mathsf{nat}. \forall z \in \mathsf{nat}. x =_{\mathsf{N}} y \supset y =_{\mathsf{N}} z \supset x =_{\mathsf{N}} z$

A judgment $\forall x \in \mathsf{nat}. \forall y \in \mathsf{nat}. \forall z \in \mathsf{nat}. x =_{\mathsf{N}} y \supset y =_{\mathsf{N}} z \supset x =_{\mathsf{N}} z\ true$ expresses the transitivity of the equality relation $=_{\mathsf{N}}$. An inductive proof of $x =_{\mathsf{N}} y \supset y =_{\mathsf{N}} z \supset x =_{\mathsf{N}} z\ true$ is given as follows:

Proof. By induction on x. We consider subcases on y and z. In each case, we assume $x =_{\mathsf{N}} y\ true$ and $y =_{\mathsf{N}} z\ true$ to show $x =_{\mathsf{N}} z\ true$.

Base case $x = \mathbf{0}$. We need to show $\mathbf{0} =_{\mathsf{N}} y \supset y =_{\mathsf{N}} z \supset \mathbf{0} =_{\mathsf{N}} z\ true$:	
Subcase $y = \mathbf{0}$:	
Subcase $z = \mathbf{0}$. We need to show $\mathbf{0} =_{\mathsf{N}} \mathbf{0}\ true$.	
$\mathbf{0} =_{\mathsf{N}} \mathbf{0}\ true$	by the rule $=_{\mathsf{N}}\mathsf{I_0}$
Subcase $z = \mathbf{s}(z')$. We need to show $\mathbf{0} =_{\mathsf{N}} \mathbf{s}(z')\ true$.	
$\mathbf{0} =_{\mathsf{N}} \mathbf{s}(z')\ true$	from the assumption $y =_{\mathsf{N}} z\ true$
Subcase $y = \mathbf{s}(y')$. We need to show $\mathbf{0} =_{\mathsf{N}} z\ true$.	
$\mathbf{0} =_{\mathsf{N}} \mathbf{s}(y')\ true$	from the assumption $x =_{\mathsf{N}} y\ true$
$\mathbf{0} =_{\mathsf{N}} z\ true$	from $\dfrac{\mathbf{0} =_{\mathsf{N}} \mathbf{s}(y')\ true}{\mathbf{0} =_{\mathsf{N}} z\ true} =_{\mathsf{N}}\mathsf{E_{0s}}$
Inductive case $x = \mathbf{s}(x')$. We need to show $\mathbf{s}(x') =_{\mathsf{N}} y \supset y =_{\mathsf{N}} z \supset \mathbf{s}(x') =_{\mathsf{N}} z\ true$:	
$x' =_{\mathsf{N}} y' \supset y' =_{\mathsf{N}} z' \supset x' =_{\mathsf{N}} z'\ true$ for any y' and z'	by induction hypothesis
Subcase $y = \mathbf{0}$. We need to show $\mathbf{s}(x') =_{\mathsf{N}} z\ true$.	
$\mathbf{s}(x') =_{\mathsf{N}} \mathbf{0}\ true$	from the assumption $x =_{\mathsf{N}} y\ true$
$\mathbf{s}(x') =_{\mathsf{N}} z\ true$	from $\dfrac{\mathbf{s}(x') =_{\mathsf{N}} \mathbf{0}\ true}{\mathbf{s}(x') =_{\mathsf{N}} z\ true} =_{\mathsf{N}}\mathsf{E_{s0}}$
Subcase $y = \mathbf{s}(y')$:	
Subcase $z = \mathbf{0}$. We need to show $\mathbf{s}(x') =_{\mathsf{N}} \mathbf{0}\ true$.	
$\mathbf{s}(y') =_{\mathsf{N}} \mathbf{0}\ true$	from the assumption $y =_{\mathsf{N}} z\ true$
$\mathbf{s}(x') =_{\mathsf{N}} \mathbf{0}\ true$	from $\dfrac{\mathbf{s}(y') =_{\mathsf{N}} \mathbf{0}\ true}{\mathbf{s}(x') =_{\mathsf{N}} \mathbf{0}\ true} =_{\mathsf{N}}\mathsf{E_{s0}}$
Subcase $z = \mathbf{s}(z')$. We need to show $\mathbf{s}(x') =_{\mathsf{N}} \mathbf{s}(z')\ true$:	
$\mathbf{s}(x') =_{\mathsf{N}} \mathbf{s}(y')\ true$	from the assumption $x =_{\mathsf{N}} y\ true$
$x' =_{\mathsf{N}} y'\ true$	from $\dfrac{\mathbf{s}(x') =_{\mathsf{N}} \mathbf{s}(y')\ true}{x' =_{\mathsf{N}} y'\ true} =_{\mathsf{N}}\mathsf{E_s}$
$\mathbf{s}(y') =_{\mathsf{N}} \mathbf{s}(z')\ true$	from the assumption $y =_{\mathsf{N}} z\ true$

$$
\begin{array}{ll}
y' =_{\mathsf{N}} z' \; true & \text{from } \dfrac{\mathsf{s}(y') =_{\mathsf{N}} \mathsf{s}(z') \; true}{y' =_{\mathsf{N}} z' \; true} =_{\mathsf{N}}\mathsf{E}_{\mathsf{s}} \\
x' =_{\mathsf{N}} z' \; true & \\
& \text{from } x' =_{\mathsf{N}} y' \supset y' =_{\mathsf{N}} z' \supset x' =_{\mathsf{N}} z' \; true,\, x' =_{\mathsf{N}} y' \; true,\, y' =_{\mathsf{N}} z' \; true \\
\mathsf{s}(x') =_{\mathsf{N}} \mathsf{s}(z') \; true & \text{from } \dfrac{x' =_{\mathsf{N}} z' \; true}{\mathsf{s}(x') =_{\mathsf{N}} \mathsf{s}(z') \; true} =_{\mathsf{N}}\mathsf{I}_{\mathsf{s}}
\end{array}
$$

□

Instead of rewriting the inductive proof as a derivation tree and then obtaining a corresponding proof term (which is tedious), we obtain a proof term $trans$ directly from its specification:

	x	y	z	$v : x =_{\mathsf{N}} y$	$w : y =_{\mathsf{N}} z$		proof term of type $x =_{\mathsf{N}} z$
$trans$	$\mathbf{0}$	$\mathbf{0}$	$\mathbf{0}$	$v : \mathbf{0} =_{\mathsf{N}} \mathbf{0}$	$w : \mathbf{0} =_{\mathsf{N}} \mathbf{0}$	$=$	eqI_0 or v or w
$trans$	$\mathbf{0}$	$\mathbf{0}$	$\mathsf{s}(z')$	$v : \mathbf{0} =_{\mathsf{N}} \mathbf{0}$	$w : \mathbf{0} =_{\mathsf{N}} \mathsf{s}(z')$	$=$	w or $\mathsf{eqE}_{0\mathsf{s}}(w)$
$trans$	$\mathbf{0}$	$\mathsf{s}(y')$	z	$v : \mathbf{0} =_{\mathsf{N}} \mathsf{s}(y')$	$w : \mathsf{s}(y') =_{\mathsf{N}} z$	$=$	$\mathsf{eqE}_{0\mathsf{s}}(v)$
$trans$	$\mathsf{s}(x')$	$\mathbf{0}$	z	$v : \mathsf{s}(x') =_{\mathsf{N}} \mathbf{0}$	$w : \mathbf{0} =_{\mathsf{N}} z$	$=$	$\mathsf{eqE}_{\mathsf{s}0}(v)$
$trans$	$\mathsf{s}(x')$	$\mathsf{s}(y')$	$\mathbf{0}$	$v : \mathsf{s}(x') =_{\mathsf{N}} \mathsf{s}(y')$	$w : \mathsf{s}(y') =_{\mathsf{N}} \mathbf{0}$	$=$	$\mathsf{eqE}_{\mathsf{s}0}(w)$
$trans$	$\mathsf{s}(x')$	$\mathsf{s}(y')$	$\mathsf{s}(z')$	$v : \mathsf{s}(x') =_{\mathsf{N}} \mathsf{s}(y')$	$w : \mathsf{s}(y') =_{\mathsf{N}} \mathsf{s}(z')$	$=$	$\mathsf{eqI}_{\mathsf{s}}(trans\ x'\ y'\ z'\ \mathsf{eqE}_{\mathsf{s}}(v)\ \mathsf{eqE}_{\mathsf{s}}(w))$

It requires a bit of thinking to obtain a correct definition of $trans$. For example, here is a wrong definition of $trans$ in which we mistakenly apply induction on x after taking y and z:

$$
\begin{array}{l}
\lambda x \in \mathsf{nat}.\, \lambda y \in \mathsf{nat}.\, \lambda z \in \mathsf{nat}. \\
\mathsf{ind}\ u(x)\ \mathsf{of} \begin{cases}
u(\mathbf{0}) \Rightarrow \mathsf{case}\ y\ \mathsf{of} \begin{cases}
\mathbf{0} \Rightarrow \mathsf{case}\ z\ \mathsf{of} \begin{cases}
\mathbf{0} \Rightarrow \lambda v\!:\!\mathbf{0} =_{\mathsf{N}} \mathbf{0}.\, \lambda w\!:\!\mathbf{0} =_{\mathsf{N}} \mathbf{0}.\, \mathsf{eqI}_0 \\
\mathsf{s}(z') \Rightarrow \lambda v\!:\!\mathbf{0} =_{\mathsf{N}} \mathbf{0}.\, \lambda w\!:\!\mathbf{0} =_{\mathsf{N}} \mathsf{s}(z').\, \mathsf{eqE}_{0\mathsf{s}}(w)
\end{cases} \\
\mathsf{s}(y') \Rightarrow \lambda v\!:\!\mathbf{0} =_{\mathsf{N}} \mathsf{s}(y').\, \lambda w\!:\!\mathsf{s}(y') =_{\mathsf{N}} z.\, \mathsf{eqE}_{0\mathsf{s}}(v)
\end{cases} \\
u(\mathsf{s}(x')) \Rightarrow \mathsf{case}\ y\ \mathsf{of} \begin{cases}
\mathbf{0} \Rightarrow \lambda v\!:\!\mathsf{s}(x') =_{\mathsf{N}} \mathbf{0}.\, \lambda w\!:\!\mathbf{0} =_{\mathsf{N}} z.\, \mathsf{eqE}_{\mathsf{s}0}(v) \\
\mathsf{s}(y') \Rightarrow \mathsf{case}\ z\ \mathsf{of} \begin{cases}
\mathbf{0} \Rightarrow \lambda v\!:\!\mathsf{s}(x') =_{\mathsf{N}} \mathsf{s}(y').\, \lambda w\!:\!\mathsf{s}(y') =_{\mathsf{N}} \mathbf{0}.\, \mathsf{eqE}_{\mathsf{s}0}(w) \\
\mathsf{s}(z') \Rightarrow \lambda v\!:\!\mathsf{s}(x') =_{\mathsf{N}} \mathsf{s}(y').\, \lambda w\!:\!\mathsf{s}(y') =_{\mathsf{N}} \mathsf{s}(z'). \\
\qquad \mathsf{eqI}_{\mathsf{s}}(u(x')\ y'\ z'\ \mathsf{eqE}_{\mathsf{s}}(v)\ \mathsf{eqE}_{\mathsf{s}}(w))
\end{cases}
\end{cases}
\end{cases}
\end{array}
$$

This definition is wrong because $u(x')\ y'\ z'\ \mathsf{eqE}_{\mathsf{s}}(v)\ \mathsf{eqE}_{\mathsf{s}}(w)$ fails to typecheck: $u(x')$ has type $x' =_{\mathsf{N}} y \supset y =_{\mathsf{N}} z \supset x' =_{\mathsf{N}} z$, but it is applied to two terms y' and z' instead of two proof terms of types $x' =_{\mathsf{N}} y$ and $y =_{\mathsf{N}} z$. Neither does dropping y' and z' help because $\mathsf{eqE}_{\mathsf{s}}(v)$ and $\mathsf{eqE}_{\mathsf{s}}(w)$ have different types $x' =_{\mathsf{N}} y'$ and $y' =_{\mathsf{N}} z'$, respectively. The problem in this definition is that y and z are already fixed when induction on x starts, leaving no chance to use $u(x')$ to build a proof term of type $x' =_{\mathsf{N}} z' \; true$ from proof terms of types $x' =_{\mathsf{N}} y'$ and $y' =_{\mathsf{N}} z'$. Thus a correct definition of proof term $trans$ starts induction on x before taking y and z as arguments:

$$
\begin{array}{l}
\lambda x \in \mathsf{nat}. \\
\mathsf{ind}\ u(x)\ \mathsf{of} \begin{cases}
u(\mathbf{0}) \Rightarrow \quad \lambda y \in \mathsf{nat}.\, \lambda z \in \mathsf{nat}. \\
\qquad \mathsf{case}\ y\ \mathsf{of} \begin{cases}
\mathbf{0} \Rightarrow \mathsf{case}\ z\ \mathsf{of} \begin{cases}
\mathbf{0} \Rightarrow \lambda v\!:\!\mathbf{0} =_{\mathsf{N}} \mathbf{0}.\, \lambda w\!:\!\mathbf{0} =_{\mathsf{N}} \mathbf{0}.\, \mathsf{eqI}_0 \\
\mathsf{s}(z') \Rightarrow \lambda v\!:\!\mathbf{0} =_{\mathsf{N}} \mathbf{0}.\, \lambda w\!:\!\mathbf{0} =_{\mathsf{N}} \mathsf{s}(z').\, \mathsf{eqE}_{0\mathsf{s}}(w)
\end{cases} \\
\mathsf{s}(y') \Rightarrow \lambda v\!:\!\mathbf{0} =_{\mathsf{N}} \mathsf{s}(y').\, \lambda w\!:\!\mathsf{s}(y') =_{\mathsf{N}} z.\, \mathsf{eqE}_{0\mathsf{s}}(v)
\end{cases} \\
u(\mathsf{s}(x')) \Rightarrow \quad \lambda y \in \mathsf{nat}.\, \lambda z \in \mathsf{nat}. \\
\qquad \mathsf{case}\ y\ \mathsf{of} \begin{cases}
\mathbf{0} \Rightarrow \lambda v\!:\!\mathsf{s}(x') =_{\mathsf{N}} \mathbf{0}.\, \lambda w\!:\!\mathbf{0} =_{\mathsf{N}} z.\, \mathsf{eqE}_{\mathsf{s}0}(v) \\
\mathsf{s}(y') \Rightarrow \mathsf{case}\ z\ \mathsf{of} \begin{cases}
\mathbf{0} \Rightarrow \lambda v\!:\!\mathsf{s}(x') =_{\mathsf{N}} \mathsf{s}(y').\, \lambda w\!:\!\mathsf{s}(y') =_{\mathsf{N}} \mathbf{0}.\, \mathsf{eqE}_{\mathsf{s}0}(w) \\
\mathsf{s}(z') \Rightarrow \lambda v\!:\!\mathsf{s}(x') =_{\mathsf{N}} \mathsf{s}(y').\, \lambda w\!:\!\mathsf{s}(y') =_{\mathsf{N}} \mathsf{s}(z'). \\
\qquad \mathsf{eqI}_{\mathsf{s}}(u(x')\ y'\ z'\ \mathsf{eqE}_{\mathsf{s}}(v)\ \mathsf{eqE}_{\mathsf{s}}(w))
\end{cases}
\end{cases}
\end{cases}
\end{array}
$$

In this definition, $u(x)$ has type $\forall y \in \mathsf{nat}.\forall z \in \mathsf{nat}.x' =_{\mathsf{N}} y \supset y =_{\mathsf{N}} z \supset x' =_{\mathsf{N}} z$, allowing us to build a proof term of type $x' =_{\mathsf{N}} z' \; true$ from proof terms of types $x' =_{\mathsf{N}} y'$ and $y' =_{\mathsf{N}} z'$.

Example 3. $\forall x \in \mathsf{nat}.\neg(x =_{\mathsf{N}} \mathbf{0}) \supset \exists y \in \mathsf{nat}.\mathsf{s}(y) =_{\mathsf{N}} x$

A judgment $\forall x \in \mathsf{nat}.\neg(x =_{\mathsf{N}} \mathbf{0}) \supset \exists y \in \mathsf{nat}.\mathsf{s}(y) =_{\mathsf{N}} x \; true$ states that every non-zero natural number is the successor of some natural number. A proof of $\neg(x =_{\mathsf{N}} \mathbf{0}) \supset \exists y \in \mathsf{nat}.\mathsf{s}(y) =_{\mathsf{N}} x \; true$, which is not an inductive proof but reuses the proof of $\forall z \in \mathsf{nat}.z =_{\mathsf{N}} z \; true$, is given as follows:

Proof. By case analysis of x.

Case $x = \mathbf{0}$. We need to show $\neg(\mathbf{0} =_{\mathsf{N}} \mathbf{0}) \supset \exists y \in \mathsf{nat}.\mathsf{s}(y) =_{\mathsf{N}} \mathbf{0}$:

$\neg(\mathbf{0} =_{\mathsf{N}} \mathbf{0})$ *true* — assumption

$\exists y \in \mathsf{nat}.\mathsf{s}(y) =_{\mathsf{N}} \mathbf{0}$ *true* — from $\neg(\mathbf{0} =_{\mathsf{N}} \mathbf{0})$ *true* and $\dfrac{}{\mathbf{0} =_{\mathsf{N}} \mathbf{0}\ \mathit{true}} =_{\mathsf{N}}\mathsf{I}_0$

Case $x = \mathsf{s}(x')$. We need to show $\neg(\mathsf{s}(x') =_{\mathsf{N}} \mathbf{0}) \supset \exists y \in \mathsf{nat}.\mathsf{s}(y) =_{\mathsf{N}} \mathsf{s}(x')$:

$\neg(\mathsf{s}(x') =_{\mathsf{N}} \mathbf{0})$ *true* — assumption (which is not used in this case)

$x' =_{\mathsf{N}} x'$ *true* — from the proof of $\forall z \in \mathsf{nat}.z =_{\mathsf{N}} z\ \mathit{true}$ and $x' \in \mathsf{nat}$

$\mathsf{s}(x') =_{\mathsf{N}} \mathsf{s}(x')$ *true* — from $\dfrac{x' =_{\mathsf{N}} x'\ \mathit{true}}{\mathsf{s}(x') =_{\mathsf{N}} \mathsf{s}(x')\ \mathit{true}} =_{\mathsf{N}}\mathsf{I}_{\mathsf{s}}$

$\exists y \in \mathsf{nat}.\mathsf{s}(y) =_{\mathsf{N}} \mathsf{s}(x')$ *true* — from $\dfrac{x' \in \mathsf{nat} \quad \mathsf{s}(x') =_{\mathsf{N}} x\ \mathit{true}}{\exists y \in \mathsf{nat}.\mathsf{s}(y) =_{\mathsf{N}} x\ \mathit{true}} \exists\mathsf{I}$

□

The specification of a proof term *pred* of type $\forall x \in \mathsf{nat}.\neg(x =_{\mathsf{N}} \mathbf{0}) \supset \exists y \in \mathsf{nat}.\mathsf{s}(y) =_{\mathsf{N}} x$ is:

$$\begin{array}{llllll} & x & v : \neg(x =_{\mathsf{N}} \mathbf{0}) & & \text{proof term of type } \exists y \in \mathsf{nat}.\mathsf{s}(y) =_{\mathsf{N}} x \\ \mathit{pred} & \mathbf{0} & v : \neg(\mathbf{0} =_{\mathsf{N}} \mathbf{0}) & = & \mathsf{abort}_{\exists y \in \mathsf{nat}.\mathsf{s}(y) =_{\mathsf{N}} \mathbf{0}}\ (v\ \mathsf{eqI}_0) \\ \mathit{pred} & \mathsf{s}(x') & v : \neg(\mathsf{s}(x') =_{\mathsf{N}} \mathbf{0}) & = & \langle x', \mathsf{eqI}_{\mathsf{s}}(\mathit{eqNat}\ x') \rangle \end{array}$$

From this specification, we obtain the following definition of *pred*:

$$\mathit{pred} \;=\; \lambda x \in \mathsf{nat}.\, \mathsf{case}\ x\ \mathsf{of}\ \left\{ \begin{array}{l} \mathbf{0} \Rightarrow \lambda v{:}\neg(\mathbf{0} =_{\mathsf{N}} \mathbf{0}).\, \mathsf{abort}_{\exists y \in \mathsf{nat}.\mathsf{s}(y) =_{\mathsf{N}} \mathbf{0}}\ (v\ \mathsf{eqI}_0) \\ \mathsf{s}(x') \Rightarrow \lambda v{:}\neg(\mathsf{s}(x') =_{\mathsf{N}} \mathbf{0}).\, \langle x', \mathsf{eqI}_{\mathsf{s}}(\mathit{eqNat}\ x') \rangle \end{array} \right.$$

Exercise 6.4. Can you give a definition of *pred* of the following form?

$$\mathit{pred} \;=\; \lambda x \in \mathsf{nat}.\, \lambda v{:}\neg(x =_{\mathsf{N}} \mathbf{0}).\, \mathsf{case}\ x\ \mathsf{of}\ \left\{ \begin{array}{l} \mathbf{0} \Rightarrow \ldots \\ \mathsf{s}(x') \Rightarrow \ldots \end{array} \right.$$

Exercise 6.5. Give a proof of $\forall x \in \mathsf{nat}.\forall y \in \mathsf{nat}.x =_{\mathsf{N}} y \supset y =_{\mathsf{N}} x\ \mathit{true}$. What is its proof term?

Exercise 6.6. Give a proof of $\forall x \in \mathsf{nat}.\forall y \in \mathsf{nat}.x < y \supset \neg(x =_{\mathsf{N}} y)\ \mathit{true}$. What is its proof term?

6.8 Induction on predicates

In Section 6.5, we have seen how to derive a set of elimination rules for a predicate from its introduction rules. For example, the introduction rules $=_{\mathsf{N}}\mathsf{I}_0$ and $=_{\mathsf{N}}\mathsf{I}_{\mathsf{s}}$ for the predicate $m =_{\mathsf{N}} n$ specify the elimination rules $=_{\mathsf{N}}\mathsf{E}_{0\mathsf{s}}$, $=_{\mathsf{N}}\mathsf{E}_{\mathsf{s}0}$, and $=_{\mathsf{N}}\mathsf{E}_{\mathsf{s}}$. Now we show how to derive yet another elimination rule from the introduction rules for a predicate. Such an elimination rule is based on *induction on predicates* and allows us to prove properties of terms satisfying certain predicates.

Suppose that we wish to show a property that whenever $m_0 =_{\mathsf{N}} n_0\ \mathit{true}$ holds, we have a proof of $A(m_0, n_0)$ where $A(m_0, n_0)$ can be any proposition involving terms m_0 and n_0. For example, we may have $A(m_0, n_0) = n_0 =_{\mathsf{N}} m_0$, in which case we attempt to prove the judgment in Exercise 6.5. We consider two cases of building a proof of $m_0 =_{\mathsf{N}} n_0\ \mathit{true}$:

- The proof of $m_0 =_{\mathsf{N}} n_0\ \mathit{true}$ uses the rule $=_{\mathsf{N}}\mathsf{I}_0$. In this case, we have $m_0 = \mathbf{0}$ and $n_0 = \mathbf{0}$, and thus need to prove $A(\mathbf{0}, \mathbf{0})\ \mathit{true}$.

- The proof of $m_0 =_{\mathsf{N}} n_0\ \mathit{true}$ uses the rule $=_{\mathsf{N}}\mathsf{I}_{\mathsf{s}}$. In this case, we have $m_0 = \mathsf{s}(m)$ and $n_0 = \mathsf{s}(n)$, and thus need to prove $A(\mathsf{s}(m), \mathsf{s}(n))\ \mathit{true}$ from the assumption of $m =_{\mathsf{N}} n\ \mathit{true}$. In addition, the principle of induction allows us to make another assumption of $A(m, n)\ \mathit{true}$ because according to the rule $=_{\mathsf{N}}\mathsf{I}_{\mathsf{s}}$, $m =_{\mathsf{N}} n\ \mathit{true}$ uses a "smaller" predicate than $\mathsf{s}(m) =_{\mathsf{N}} \mathsf{s}(n)\ \mathit{true}$ and is assumed to already satisfy the property.

The analysis in these two cases justifies the following elimination rule:

$$\dfrac{m_0 =_{\mathsf{N}} n_0\ \mathit{true} \qquad A(\mathbf{0},\mathbf{0})\ \mathit{true} \qquad \begin{array}{c} \overline{m \in \mathsf{nat}} \quad \overline{n \in \mathsf{nat}} \quad \overline{m =_{\mathsf{N}} n}^{\,w} \quad \overline{A(m,n)\ \mathit{true}}^{\,u(m,n)} \\ \vdots \\ A(\mathbf{s}(m),\mathbf{s}(n))\ \mathit{true} \end{array}}{A(m_0,n_0)\ \mathit{true}}\ {=_{\mathsf{N}}}\mathsf{E}_I^{w,u(m,n)}$$

Note that as is the case for induction on terms, the second and third premises do not use m_0 and n_0 at all and constitute a valid proof of $A(m,n)\ \mathit{true}$ for every pair of natural numbers m and n. Thus the first premise just provides two specific natural number m_0 and n_0 to be substituted for m and n in $A(m,n)\ \mathit{true}$ and are not essential in completing a proof by induction on predicates.

As an example of using the rule ${=_{\mathsf{N}}}\mathsf{E}_I$, here is a proof of the judgment in Exercise 6.5 where we let $A(m,n) = n =_{\mathsf{N}} m$:

$$\dfrac{\dfrac{\dfrac{\dfrac{\overline{x =_{\mathsf{N}} y\ \mathit{true}}^{\,w} \qquad \dfrac{}{\mathbf{0} =_{\mathsf{N}} \mathbf{0}\ \mathit{true}}\ {=_{\mathsf{N}}}\mathsf{I}_0 \qquad \dfrac{\overline{n =_{\mathsf{N}} m\ \mathit{true}}^{\,u(m,n)}}{\mathbf{s}(n) =_{\mathsf{N}} \mathbf{s}(m)\ \mathit{true}}\ {=_{\mathsf{N}}}\mathsf{I}_{\mathsf{s}}}{y =_{\mathsf{N}} x\ \mathit{true}}\ {=_{\mathsf{N}}}\mathsf{E}_I^{w,u(m,n)}}{x =_{\mathsf{N}} y \supset y =_{\mathsf{N}} x\ \mathit{true}}\ {\supset}\mathsf{I}^w}{\forall y\in\mathsf{nat}.x =_{\mathsf{N}} y \supset y =_{\mathsf{N}} x\ \mathit{true}}\ \forall\mathsf{I}}{\forall x\in\mathsf{nat}.\forall y\in\mathsf{nat}.x =_{\mathsf{N}} y \supset y =_{\mathsf{N}} x\ \mathit{true}}\ \forall\mathsf{I}$$

In a similar way, the introduction rules for the predicate $m < n$ specify the following elimination rule based on induction on predicates:

$$\dfrac{m_0 < n_0\ \mathit{true} \qquad \begin{array}{c} \overline{n \in \mathsf{nat}} \\ \vdots \\ A(\mathbf{0},\mathbf{s}(n))\ \mathit{true} \end{array} \qquad \begin{array}{c} \overline{m \in \mathsf{nat}} \quad \overline{n \in \mathsf{nat}} \quad \overline{m < n}^{\,w} \quad \overline{A(m,n)\ \mathit{true}}^{\,u(m,n)} \\ \vdots \\ A(\mathbf{s}(m),\mathbf{s}(n))\ \mathit{true} \end{array}}{A(m_0,n_0)\ \mathit{true}}\ {<}\mathsf{E}_I^{w,u(m,n)}$$

We can now simplify the proof of the judgment in Exercise 6.6 by using the rule ${<}\mathsf{E}_I$ with $A(m,n) = \neg(m =_{\mathsf{N}} n)$:

$$\dfrac{\dfrac{\dfrac{\dfrac{\overline{x < y\ \mathit{true}}^{\,w} \qquad \dfrac{\dfrac{\overline{\mathbf{0} =_{\mathsf{N}} \mathbf{s}(n)\ \mathit{true}}^{\,v}}{\bot\ \mathit{true}}\ {=_{\mathsf{N}}}\mathsf{E}_{0\mathsf{s}}}{\neg(\mathbf{0} =_{\mathsf{N}} \mathbf{s}(n))\ \mathit{true}}\ \neg\mathsf{I}^v \qquad \dfrac{\dfrac{\overline{\neg(m =_{\mathsf{N}} n)\ \mathit{true}}^{\,u(m,n)} \qquad \dfrac{\overline{\mathbf{s}(m) =_{\mathsf{N}} \mathbf{s}(n)\ \mathit{true}}^{\,v}}{m =_{\mathsf{N}} n\ \mathit{true}}\ {=_{\mathsf{N}}}\mathsf{E}_{\mathsf{s}}}{\bot\ \mathit{true}}\ \neg\mathsf{E}}{\neg(\mathbf{s}(m) =_{\mathsf{N}} \mathbf{s}(n))\ \mathit{true}}\ \neg\mathsf{I}^v}{\neg(x =_{\mathsf{N}} y)\ \mathit{true}}\ {<}\mathsf{E}_I^{w,u(m,n)}}{x < y \supset \neg(x =_{\mathsf{N}} y)\ \mathit{true}}\ {\supset}\mathsf{I}^w}{\forall y\in\mathsf{nat}.x < y \supset \neg(x =_{\mathsf{N}} y)\ \mathit{true}}\ \forall\mathsf{I}}{\forall x\in\mathsf{nat}.\forall y\in\mathsf{nat}.x < y \supset \neg(x =_{\mathsf{N}} y)\ \mathit{true}}\ \forall\mathsf{I}$$

6.9 Definitional equality

So far, we have seen how to use predicates on terms to prove various properties of datatypes. Incidentally these terms are not further reducible by β-reductions because they are either variables or built only by introduction rules, as in $\mathbf{0} =_{\mathsf{N}} \mathbf{0}$, $x =_{\mathsf{N}} y$, and $\mathsf{s}(x) =_{\mathsf{N}} \mathsf{s}(y)$. Now we consider predicates containing terms that may reduce to simpler terms by β-reductions. For example, a predicate $\mathit{plus}\ \mathbf{0}\ \mathbf{0}\ =_{\mathsf{N}} \mathbf{0}$ contains a term $\mathit{plus}\ \mathbf{0}\ \mathbf{0}$ which reduces to $\mathbf{0}$ by β-reductions. Note that $\mathit{plus}\ \mathbf{0}\ \mathbf{0}\ =_{\mathsf{N}} \mathbf{0}\ \mathit{true}$ is *not* provable because the introduction rules ${=_{\mathsf{N}}}\mathsf{I}_0$ and ${=_{\mathsf{N}}}\mathsf{I}_{\mathsf{s}}$ allow us to prove judgments of the form $\mathbf{0} =_{\mathsf{N}} \mathbf{0}\ \mathit{true}$ and $\mathsf{s}(m) =_{\mathsf{N}} \mathsf{s}(n)\ \mathit{true}$ only. Since $\mathit{plus}\ \mathbf{0}\ \mathbf{0}$ and $\mathbf{0}$ denote the same natural number, we would like to be able to prove $\mathit{plus}\ \mathbf{0}\ \mathbf{0}\ =_{\mathsf{N}} \mathbf{0}\ \mathit{true}$.

This section develops a methodology that enables us to prove such judgments as $\mathit{plus}\ \mathbf{0}\ \mathbf{0}\ =_{\mathsf{N}} \mathbf{0}\ \mathit{true}$. The basic idea is to define a notion of equality $=$, called *definitional equality*, which identifies two terms that reduce to the same term by β-reductions. For example, we have an equality $plus\ \mathbf{0}\ \mathbf{0} = \mathbf{0}$ because $plus\ \mathbf{0}\ \mathbf{0}$ reduces to $\mathbf{0}$ by β-reductions. Then the equality $plus\ \mathbf{0}\ \mathbf{0} = \mathbf{0}$ allows us to simplify the judgment $plus\ \mathbf{0}\ \mathbf{0}\ =_{\mathsf{N}}\ \mathbf{0}\ true$ to $\mathbf{0}\ =_{\mathsf{N}}\ \mathbf{0}\ true$, which is provable.

It is important that although definitional equality uses the symbol $=$ which usually stands for syntactic equality, it is a strict extension of syntactic equality. For example, $\mathbf{0} = \mathbf{0}$ holds under both syntactic equality (because x is syntactically equal to x) and definitional equality (because x reduces to x by zero β-reductions), but $plus\ \mathbf{0}\ \mathbf{0} = \mathbf{0}$ does not hold under syntactic equality (because $plus\ \mathbf{0}\ \mathbf{0}$ is syntactically different from $\mathbf{0}$) while it holds under definitional equality. We also note that definitional equality has nothing to do with the symbol $=_{\mathsf{N}}$ in the predicate $m\ =_{\mathsf{N}}\ n$ which is just a syntactic abbreviation of $EQ(m, n)$.

Formally we use a new judgment $t = s$ to mean that terms t and s are definitionally equal. (Instead of writing $t = s\ true$, we write $t = s$, omitting $true$.) As with predicates, we base the definition of $t = s$ on the principle of natural deduction:

$$\frac{t \Longrightarrow_\beta^* r \quad s \Longrightarrow_\beta^* r}{t = s}\ \mathit{DefEq}\mathsf{I} \qquad \frac{A(t)\ true \quad t = s}{A(s)\ true}\ \mathit{DefEq}\mathsf{E}$$

In the introduction rule $\mathit{DefEq}\mathsf{I}$, the judgment $t \Longrightarrow_\beta^* r$ means that t reduces to r by zero or more β-reductions. Here we assume that β-reductions may be applied to subterms of the term being reduced. For example, if $t_1 \Longrightarrow_\beta t_2$ holds, $\mathbf{s}(t_1) \Longrightarrow_\beta^* \mathbf{s}(t_2)$ holds because t_1 is a subterm of $\mathbf{s}(t_1)$. Thus the rule $\mathit{DefEq}\mathsf{I}$ states that two terms are definitionally equal if both reduce to the same term by β-reductions. As a special case, two terms t and s are definitionally equal if t reduces to s by β-reductions or vice versa.

- If $t \Longrightarrow_\beta^* s$ or $s \Longrightarrow_\beta^* t$, then $t = s$.

The elimination rule $\mathit{DefEq}\mathsf{E}$ states that once we build a proof of $t = s$, we cease to distinguish between (syntactically different) propositions $A(t)$ and $A(s)$. The corresponding typing rule allows us to change the type of a proof term silently without changing the proof term itself:

$$\frac{M : A(t) \quad t = s}{M : A(s)}\ \mathit{DefEq}\mathsf{E}$$

According to the rule $\mathit{DefEq}\mathsf{I}$, definitional equality is a relation between terms which is reflexive and commutative:

- $t = t$ holds for any term t.
- $t = s$ implies $s = t$.

Definitional equality is not necessarily transitive because the set of terms is open-ended and thus can be extended with new terms that destroy the transitivity of definitional equality. It is transitive, however, in the following weak sense:

- If $t = s$ and $s = r$, then $A(t)\ true$ implies $A(r)\ true$.

Here are a few examples of definitional equality:

- $pred\ \mathbf{0} = \mathbf{0}$ holds because we have $pred\ \mathbf{0} \Longrightarrow_\beta \mathbf{case}\ \mathbf{0}\ \mathbf{of}\ \mathbf{0} \Rightarrow \mathbf{0} \mid \mathbf{s}(y) \Rightarrow y \Longrightarrow_\beta \mathbf{0}$.
- The sequence of β-reductions on Page 79 proves $plus\ \mathbf{s}(\mathbf{0})\ t = \mathbf{s}(t)$.
- $plus\ \mathbf{0}\ x = x$ holds:

$$\begin{array}{lll} plus\ \mathbf{0}\ x & \Longrightarrow_\beta & (\lambda y \in \mathsf{nat}.\, \mathbf{rec}\ p(\mathbf{0})\ \mathbf{of}\ p(\mathbf{0}) \Rightarrow y \mid p(\mathbf{s}(z)) \Rightarrow \mathbf{s}(p(z)))\ x \\ & \Longrightarrow_\beta & \mathbf{rec}\ p(\mathbf{0})\ \mathbf{of}\ p(\mathbf{0}) \Rightarrow x \mid p(\mathbf{s}(z)) \Rightarrow \mathbf{s}(p(z)) \\ & \Longrightarrow_\beta & x \end{array}$$

- $plus\ x\ \mathbf{0} = x$ does *not* holds:

$$\begin{array}{lll} plus\ x\ \mathbf{0} & \Longrightarrow_\beta & (\lambda y \in \mathsf{nat}.\, \mathbf{rec}\ p(x)\ \mathbf{of}\ p(\mathbf{0}) \Rightarrow y \mid p(\mathbf{s}(z)) \Rightarrow \mathbf{s}(p(z)))\ \mathbf{0} \\ & \Longrightarrow_\beta & \mathbf{rec}\ p(x)\ \mathbf{of}\ p(\mathbf{0}) \Rightarrow \mathbf{0} \mid p(\mathbf{s}(z)) \Rightarrow \mathbf{s}(p(z)) \\ & \Longrightarrow_\beta & ??? \end{array}$$

As an example of a proof using definitional equality, let us find a proof term of the following type which states that every natural number is either even or odd:

$$\forall x \in \mathsf{nat}.(\exists y \in \mathsf{nat}.y + y =_{\mathsf{N}} x) \vee (\exists y \in \mathsf{nat}.\mathbf{s}(y + y) =_{\mathsf{N}} x)$$

Here we write $t + s$ for $plus\ t\ s$. Note that without definitional equality, it is impossible to find such a proof term because there is no way to prove, for example, $y + y =_{\mathsf{N}} x$.

First we observe that $\mathbf{s}(x) + y = \mathbf{s}(x + y)$ holds:

$$\begin{array}{lll} \mathbf{s}(x) + y & \Longrightarrow_\beta^* & \mathbf{rec}\ p(\mathbf{s}(x))\ \mathbf{of}\ p(\mathbf{0}) \Rightarrow y \mid p(\mathbf{s}(z)) \Rightarrow \mathbf{s}(p(z)) \\ & \Longrightarrow_\beta & \mathbf{s}(\mathbf{rec}\ p(x)\ \mathbf{of}\ p(\mathbf{0}) \Rightarrow y \mid p(\mathbf{s}(z)) \Rightarrow \mathbf{s}(p(z))) \\ \mathbf{s}(x + y) & \Longrightarrow_\beta^* & \mathbf{s}(\mathbf{rec}\ p(x)\ \mathbf{of}\ p(\mathbf{0}) \Rightarrow y \mid p(\mathbf{s}(z)) \Rightarrow \mathbf{s}(p(z))) \end{array}$$

Next we define a proof term $comp$ whose type is $\forall x \in \mathsf{nat}.\forall y \in \mathsf{nat}.x + \mathbf{s}(y) =_{\mathsf{N}} \mathbf{s}(x + y)$:

$$comp \ = \ \lambda x \in \mathsf{nat}.\ \lambda y \in \mathsf{nat}.\ \mathsf{ind}\ u(x)\ \mathsf{of}\ \left\{ \begin{array}{l} u(\mathbf{0}) \Rightarrow eqNat\ \mathbf{s}(y) \\ u(\mathbf{s}(x')) \Rightarrow \mathsf{eql}_{\mathsf{s}}(u(x')) \end{array} \right.$$

Here $eqNat\ \mathbf{s}(y)$ is assigned type $\mathbf{0} + \mathbf{s}(y) =_{\mathsf{N}} \mathbf{s}(\mathbf{0} + y)$ which is equivalent to $\mathbf{s}(y) =_{\mathsf{N}} \mathbf{s}(y)$ under definitional equality. Similarly $\mathsf{eql}_{\mathsf{s}}(u(x'))$ is assigned type $\mathbf{s}(x') + \mathbf{s}(y) =_{\mathsf{N}} \mathbf{s}(\mathbf{s}(x') + y)$ which is equivalent to $\mathbf{s}(x' + \mathbf{s}(y)) =_{\mathsf{N}} \mathbf{s}(\mathbf{s}(x' + y))$ under definitional equality. Then we use the proof term $trans$ given in Section 6.7 to obtain a proof term of the given type:

$$\begin{array}{l} \lambda x \in \mathsf{nat}. \\ \quad \mathsf{ind}\ u(x)\ \mathsf{of}\ \left\{ \begin{array}{l} u(\mathbf{0}) \Rightarrow \mathsf{inl}_{\exists y \in \mathsf{nat}.\mathbf{s}(y+y) =_{\mathsf{N}} \mathbf{0}}\ \langle \mathbf{0}, \mathsf{eql}_0 \rangle \\ u(\mathbf{s}(x')) \Rightarrow \end{array} \right. \\ \quad\quad \mathsf{case}\ u(x')\ \mathsf{of}\ \left\{ \begin{array}{ll} & \mathsf{inl}\ z.\ \mathsf{let}\ \langle y, w \rangle = z\ \mathsf{in}\ \mathsf{inr}_{\exists y \in \mathsf{nat}.y+y =_{\mathsf{N}} \mathbf{s}(x')}\ \langle y, \mathsf{eql}_{\mathsf{s}}(w) \rangle \\ \mid & \mathsf{inr}\ z.\ \mathsf{let}\ \langle y, w \rangle = z\ \mathsf{in} \end{array} \right. \\ \quad\quad\quad \mathsf{inl}_{\exists y \in \mathsf{nat}.\mathbf{s}(y+y) =_{\mathsf{N}} \mathbf{s}(x')}\ \langle \mathbf{s}(y), trans\ (\mathbf{s}(y) + \mathbf{s}(y))\ (\mathbf{s}(\mathbf{s}(y + y)))\ (\mathbf{s}(x'))\ (comp\ \mathbf{s}(y)\ y) \mathsf{eql}_{\mathsf{s}}(w) \rangle \end{array}$$

Chapter 7

Classical Logic

Unlike constructive logic which is usually explained operationally by associating each proposition A with a proof proving or refuting its truth, classical logic is a logic that is usually explained *denotationally* by associating each proposition A with a truth value, either true T or false F. As there are only two truth values in classical logic, it is both sound and complete to check the truth value of a proposition A with a truth table in which all possible combinations of truth values of atomic propositions in A are considered in turn. For example, the connectives in classical propositional logic can be explained as follows:

A	B	$A \wedge B$	$A \vee B$	$A \supset B$
T	T	T	T	T
T	F	F	T	F
F	T	F	T	T
F	F	F	F	T

In this chapter, we investigate proof-theoretic formulations of classical logic based on inference rules. We also investigate its operational interpretation, or its computational contents, to obtain useful constructs for programming languages.

7.1 A judgmental formulation of classical logic

We have already seen in Chapter 2 that the addition of the following rule to constructive logic yields classical logic:

$$\frac{}{A \vee \neg A\ \mathit{true}}\ \mathsf{EM}$$

The rule EM, called *the law of excluded middle,* asserts that for any proposition A, either $A\ \mathit{true}$ or $\neg A\ \mathit{true}$ must hold regardless of the existence of an actual proof. Another way to obtain classical logic is to add one of the following rules:

$$\frac{}{\neg\neg A \supset A\ \mathit{true}}\ \mathsf{DNE} \qquad \frac{}{((A \supset B) \supset A) \supset A\ \mathit{true}}\ \mathsf{Peirce}$$

The rule DNE, called *the law of double-negation elimination,* asserts that if A cannot be false, it must be true. The rule Peirce, called *Peirce's law*, says that a proof of $A\ \mathit{true}$ may freely assume $A \supset B\ \mathit{true}$ for an arbitrary proposition B. The three rules above are all equivalent to each other in that the addition of any of these rules renders the other two rules derivable.

Note that these rules destroy the orthogonality of the system. For example, in the presence of the rule EM which uses two connectives $\supset$ and $\neg$ in the conclusion, the meaning of $\supset$ depends on the meaning of $\neg$ (or vice versa). The rule Peirce is also bad because it tries to explain the meaning of $\supset$ presupposing the notion of $\supset$. (As a rule of thumb, an inference rule using multiple connectives, whether same or different, is always bad.) Instead of using these rules, therefore, we develop a system of classical logic based purely on judgmental notions.

We recall that at the heart of classical logic lies the principle of *proof by contradiction* (which is commonly employed in mathematical proofs). Using a truth judgment $A\ \mathit{true}$ and a falsehood judgment

A *false* (which denotes "A *is false*"), we can explain the idea as follows: in order to prove $A\ \mathit{true}$, we first assume A *false* and then show a contraction, which is established by a proof of $B\ \mathit{true}$ when an assumption B *false* is available for some proposition B. Observe that we may have to make an assumption of a falsehood judgment in the course of a proof, but we never directly conclude a falsehood judgment. Thus we use a new form of hypothetical judgment $\Gamma; \Delta \vdash_{\mathsf{K}} C\ \mathit{true}$ (where the subscript K stands for *K*lassical), in which antecedents may include falsehood judgments, but the succedent is always a truth judgment:

$$\Gamma; \Delta \vdash_{\mathsf{K}} C\ \mathit{true} \qquad \text{where} \qquad \begin{array}{lcl} \Gamma & ::= & \cdot \mid \Gamma, A\ \mathit{true} \\ \Delta & ::= & \cdot \mid \Delta, A\ \mathit{false} \end{array}$$

All previous rules from constructive propositional logic (which do not need falsehood judgments) are turned into corresponding rules for classical propositional logic by rewriting $\Gamma \vdash C\ \mathit{true}$ as $\Gamma; \Delta \vdash_{\mathsf{K}} C\ \mathit{true}$. The principle of proof by contradiction is implemented by the following two rules:

$$\frac{\Gamma; \Delta, A\ \mathit{false} \vdash_{\mathsf{K}} A\ \mathit{true}}{\Gamma; \Delta \vdash_{\mathsf{K}} A\ \mathit{true}}\ \mathsf{Contra}\!\uparrow \qquad \frac{\Gamma; \Delta, A\ \mathit{false} \vdash_{\mathsf{K}} A\ \mathit{true}}{\Gamma; \Delta, A\ \mathit{false} \vdash_{\mathsf{K}} C\ \mathit{true}}\ \mathsf{Contra}\!\downarrow$$

The rule $\mathsf{Contra}\!\uparrow$ states that in order to prove $A\ \mathit{true}$, we may assume A *false* and then show a contradiction by proving $A\ \mathit{true}$; hence it is best read in a bottom-up way (as indicated by the upward arrow in $\mathsf{Contra}\!\uparrow$). The rule $\mathsf{Contra}\!\downarrow$ states that if antecedents Γ and $\Delta, A\ \mathit{false}$ are inconsistent (because $A\ \mathit{true}$ is provable from Γ and $\Delta, A\ \mathit{false}$), we may conclude $C\ \mathit{true}$ for any proposition C as long as the same antecedents Γ and $\Delta, A\ \mathit{false}$ are available; hence it is best read in a top-down way (as indicated by the downward arrow in $\mathsf{Contra}\!\downarrow$). Note that the premise is the same in both rules.

Here are a couple of examples showing that the rules EM and DNE are now derivable.

$$\dfrac{\dfrac{\dfrac{\dfrac{\dfrac{\dfrac{}{A\ \mathit{true}; A \lor \neg A\ \mathit{false} \vdash_{\mathsf{K}} A\ \mathit{true}}\ \mathsf{Hyp}}{A\ \mathit{true}; A \lor \neg A\ \mathit{false} \vdash_{\mathsf{K}} A \lor \neg A\ \mathit{true}}\ \lor\mathsf{I_L}}{A\ \mathit{true}; A \lor \neg A\ \mathit{false} \vdash_{\mathsf{K}} \bot\ \mathit{true}}\ \mathsf{Contra}\!\downarrow}{\cdot; A \lor \neg A\ \mathit{false} \vdash_{\mathsf{K}} \neg A\ \mathit{true}}\ \neg\mathsf{I}}{\cdot; A \lor \neg A\ \mathit{false} \vdash_{\mathsf{K}} A \lor \neg A\ \mathit{true}}\ \lor\mathsf{I_R}}{\cdot; \cdot \vdash_{\mathsf{K}} A \lor \neg A\ \mathit{true}}\ \mathsf{Contra}\!\uparrow$$

$$\dfrac{\dfrac{\dfrac{\dfrac{\dfrac{}{\neg\neg A\ \mathit{true}; A\ \mathit{false} \vdash_{\mathsf{K}} \neg\neg A\ \mathit{true}}\ \mathsf{Hyp} \qquad \dfrac{\dfrac{\dfrac{}{\neg\neg A\ \mathit{true}, A\ \mathit{true}; A\ \mathit{false} \vdash_{\mathsf{K}} A\ \mathit{true}}\ \mathsf{Hyp}}{\neg\neg A\ \mathit{true}, A\ \mathit{true}; A\ \mathit{false} \vdash_{\mathsf{K}} \bot\ \mathit{true}}\ \mathsf{Contra}\!\downarrow}{\neg\neg A\ \mathit{true}; A\ \mathit{false} \vdash_{\mathsf{K}} \neg A\ \mathit{true}}\ \neg\mathsf{I}}{\neg\neg A\ \mathit{true}; A\ \mathit{false} \vdash_{\mathsf{K}} \bot\ \mathit{true}}\ \neg\mathsf{E}}{\neg\neg A\ \mathit{true}; A\ \mathit{false} \vdash_{\mathsf{K}} A\ \mathit{true}}\ \bot\mathsf{E}}{\neg\neg A\ \mathit{true}; \cdot \vdash_{\mathsf{K}} A\ \mathit{true}}\ \mathsf{Contra}\!\uparrow}{\cdot; \cdot \vdash_{\mathsf{K}} \neg\neg A \supset A\ \mathit{true}}\ \supset\!\mathsf{I}$$

7.2 Proof terms

We apply the Curry-Howard isomorphism to the new form of hypothetical judgment $\Gamma; \Delta \vdash_{\mathsf{K}} C\ \mathit{true}$ by first associating variables with judgments in Γ and Δ and then assigning a proof term of type C. For $A\ \mathit{true}$ in Γ, we bind a variable x to type A to obtain $x : A$. For A *false* in Δ, we need a new notation to indicate that a variable x is associated with a falsehood judgment A *false*. We choose a new notation $x : A\ \mathit{false}$, in which A *false* can be thought of as a type instead of a judgment; in the context of type theory, A *false* stands for the type of *continuations* of type A.

$$\begin{array}{lcl} \Gamma & ::= & \cdot \mid \Gamma, x : A \\ \Delta & ::= & \cdot \mid \Delta, x : A\ \mathit{false} \end{array}$$

As proof terms corresponding to the rules $\mathsf{Contra}\!\uparrow$ and $\mathsf{Contra}\!\downarrow$, we use $\mathsf{callcc}\ x : A\ \mathit{false}.\, M$ and $\mathsf{throw}\ M\ \mathsf{to}\ x$ which are constructs for capturing and throwing continuations in programming languages:

$$\frac{\Gamma; \Delta, x : A\ \mathit{false} \vdash_{\mathsf{K}} M : A}{\Gamma; \Delta \vdash_{\mathsf{K}} \mathsf{callcc}\ x : A\ \mathit{false}.\, M : A}\ \mathsf{Callcc} \qquad \frac{\Gamma; \Delta, x : A\ \mathit{false} \vdash_{\mathsf{K}} M : A}{\Gamma; \Delta, x : A\ \mathit{false} \vdash_{\mathsf{K}} \mathsf{throw}\ M\ \mathsf{to}\ x : C}\ \mathsf{Throw}$$

$$\frac{}{\Gamma, A \Longrightarrow A, \Delta}\ \mathit{Contra}$$

$$\frac{\Gamma, A \wedge B, A \Longrightarrow \Delta}{\Gamma, A \wedge B \Longrightarrow \Delta}\ \wedge L_L \qquad \frac{\Gamma, A \wedge B, B \Longrightarrow \Delta}{\Gamma, A \wedge B \Longrightarrow \Delta}\ \wedge L_R \qquad \frac{\Gamma \Longrightarrow A, A \wedge B, \Delta \quad \Gamma \Longrightarrow B, A \wedge B, \Delta}{\Gamma \Longrightarrow A \wedge B, \Delta}\ \wedge R$$

$$\frac{\Gamma, A \vee B, A \Longrightarrow \Delta \quad \Gamma, A \vee B, B \Longrightarrow \Delta}{\Gamma, A \vee B \Longrightarrow \Delta}\ \vee L \qquad \frac{\Gamma \Longrightarrow A, A \vee B, \Delta}{\Gamma \Longrightarrow A \vee B, \Delta}\ \vee R_L \qquad \frac{\Gamma \Longrightarrow B, A \vee B, \Delta}{\Gamma \Longrightarrow A \vee B, \Delta}\ \vee R_R$$

$$\frac{}{\Gamma \Longrightarrow \top, \Delta}\ \top R \qquad \frac{}{\Gamma, \bot \Longrightarrow \Delta}\ \bot L \qquad \frac{\Gamma, \neg A \Longrightarrow A, \Delta}{\Gamma, \neg A \Longrightarrow \Delta}\ \neg L \qquad \frac{\Gamma, A \Longrightarrow \neg A, \Delta}{\Gamma \Longrightarrow \neg A, \Delta}\ \neg R$$

$$\frac{\Gamma, A \supset B \Longrightarrow A, \Delta \quad \Gamma, A \supset B, B \Longrightarrow \Delta}{\Gamma, A \supset B \Longrightarrow \Delta}\ \supset L \qquad \frac{\Gamma, A \Longrightarrow B, A \supset B, \Delta}{\Gamma \Longrightarrow A \supset B, \Delta}\ \supset R$$

Figure 7.1: Sequent calculus for classical propositional logic

In the rule Throw, proof term M is allowed to contain variable x.

A simple case of reducing a proof term throw M to x occurs when M does not contain x:

$$\textsf{callcc}\ x : A\ \mathit{false}.\ \sigma[\![\textsf{throw}\ M\ \textsf{to}\ x]\!] \quad \Longrightarrow_R \quad M$$

Here $\sigma[\![\textsf{throw}\ M\ \textsf{to}\ x]\!]$ denotes a certain proof term containing throw M to x as a subterm. By the rule Callcc, it has type A so that the whole proof term callcc $x : A\ \mathit{false}.\ \sigma[\![\textsf{throw}\ M\ \textsf{to}\ x]\!]$ is assigned type A. By the rule Throw, proof term M also has type A, and thus the type of the proof term being reduced is preserved. For a full account of reductions of callcc $x : A\ \mathit{false}.\ M$ and throw M to x, we need to formalize the definition of σ, which we do not pursue here.

A proof term LEM of type $A \vee \neg A$ is given as follows:

$$\mathit{LEM} \quad = \quad \textsf{callcc}\ x : A \vee \neg A\ \mathit{false}.\ \textsf{inr}_A\ \lambda y\!:\!A.\ \textsf{throw}\ \textsf{inl}_{\neg A}\ y\ \textsf{to}\ x$$

A proof term DNE of type $\neg\neg A \supset A$ is given as follows:

$$\mathit{DNE} \quad = \quad \lambda x\!:\!\neg\neg A.\ \textsf{callcc}\ y : A\ \mathit{false}.\ \textsf{abort}_A\ (x\ (\lambda z\!:\!A.\ \textsf{throw}\ z\ \textsf{to}\ y))$$

7.3 Sequent calculus for classical logic

The sequent calculus for classical logic uses a new form of sequent whose definition is motivated by the principle of proof by contradiction, rather than the notion of normal proof as in constructive logic. We write $A_1, \cdots, A_n \Longrightarrow B_1, \cdots, B_m$ to mean that assumptions of A_1 *true*, $\cdots$, A_n *true* and B_1 *false*, $\cdots$, B_m *false* lead to a contradiction. Note that unlike a sequent $\Gamma \longrightarrow C$ for constructive logic, the new sequent allows multiple propositions in the right side. We use Γ for a collection of propositions in the left side and Δ for a collection of propositions in the right side (*e.g.*, $\Gamma \Longrightarrow \Delta$). We assume that Γ and Δ are unordered.

The definition of the new form of sequent justifies the following rule, which is similar to the rule *Init* in constructive logic but actually expresses the principle of proof by contradiction:

$$\frac{}{\Gamma, A \Longrightarrow A, \Delta}\ \mathit{Contra}$$

The rule *Contra* says that since assumptions of A *true* and A *false* lead to a contradiction, the proof of $\Gamma, A \Longrightarrow A, \Delta$ is completed immediately.

As in the sequent calculus for constructive logic, we give left and right rules for each connective. Although these rules appear to be mechanically derived from their corresponding rules in the sequent calculus for constructive logic, their interpretation is different because the definition of $\Gamma \Longrightarrow \Delta$ is motivated differently from the definition of $\Gamma \longrightarrow C$. As each rule focuses on a proposition in the left or right side in a given sequent, we choose to reuse the rule names from the sequent calculus for constructive logic.

Figure 7.1 shows all the rules in the sequent calculus for classical propositional logic. As in the sequent calculus for constructive logic, the proof of a sequent always proceeds in a bottom-up way. There are two important observations to make. First, unlike in the sequent calculus for constructive logic, the right side of a sequent should *not* be read as a collection of conclusions to be drawn; rather it should be read as a collection of assumptions (consisting of falsehood judgments). Second the premise in a rule (especially in a right rule) always contains more assumptions than the conclusion: we never cancel an existing assumption because the goal is to elicit a contradiction from a collection of assumptions.

The left rules $\wedge L_L$, $\wedge L_R$, $\vee L$ can be read in the same manner as their counterparts in the sequent calculus for constructive logic. The right rules, however, cannot be read in the same manner because the right side of a sequent should be read not as a collection of conclusions but as a collection of assumptions. In the rule $\wedge R$, for example, we use an assumption $A \wedge B$ *false* to yield a contradiction, rather than deduce a conclusion $A \wedge B$ *false*. Since $A \wedge B$ *false* holds when either A *false* or B *false* holds, we have to show a contradiction in each case. (Using the right side as a collection of conclusions would make it difficult, or even impossible, to make sense of the rule $\wedge R$.) The right rules $\vee R_L$ and $\vee R_R$ show that we never cancel an existing assumption.

The rule $\top R$ makes sense because an assumption $\top$ *false* expresses a contradiction: $\top$ cannot be false. Likewise the rule $\bot L$ makes sense because $\bot$ cannot be true. The rule $\neg L$ states that assuming $\neg A$ *true* is equivalent to assuming A *false*; similarly the rule $\neg R$ states that assuming $\neg A$ *false* is equivalent to assuming A *true*. (Here we do not use the notational definition of $\neg A$ as $A \supset \bot$.)

Although Figure 7.1 includes the rules $\supset L$ and $\supset R$, these rules are in fact derived rules because implication is a derived notion: classical logic has no notion of transforming a proof into another because every proposition denotes just a truth value, and thus $A \supset B$ is *defined* as $\neg A \vee B$. Lack of the notion of implication in classical logic is also the reason why $A \supset B$ is assigned a truth value T whenever A is assigned a truth value F.

The sequent calculus in Figure 7.1 satisfies the weakening and contraction properties which allow us to use a proposition A in Γ or Δ as many times as necessary in a proof of $\Gamma \Longrightarrow \Delta$. It also satisfies the subformula property in the same sense as in the sequent calculus for constructive logic, which in turn implies that the sequent calculus is decidable.

Proposition 7.1 (Structural properties).

(Weakening)	*If* $\Gamma \Longrightarrow \Delta$, *then* $\Gamma, A \Longrightarrow \Delta$.
	If $\Gamma \Longrightarrow \Delta$, *then* $\Gamma \Longrightarrow A, \Delta$.
(Contraction)	*If* $\Gamma, A, A \Longrightarrow \Delta$, *then* $\Gamma, A \Longrightarrow \Delta$.
	If $\Gamma \Longrightarrow A, A, \Delta$, *then* $\Gamma \Longrightarrow A, \Delta$.

Proof. By induction on the structure of the proof of $\Gamma \Longrightarrow \Delta$ and $\Gamma, A, A \Longrightarrow \Delta$ and $\Gamma \Longrightarrow A, A, \Delta$. $\square$

As in the sequent calculus for constructive logic, there is a cut rule whose admissibility implies that the sequent calculus is sound (*i.e.*, $\cdot \Longrightarrow \bot$ is not provable). Interestingly it expresses precisely the law of excluded middle:

$$\frac{\Gamma \Longrightarrow A, \Delta \quad \Gamma, A \Longrightarrow \Delta}{\Gamma \Longrightarrow \Delta}\ \mathit{Cut}$$

The two premises cover all possibilities because for every proposition A, either A *true* or A *false* must hold by the law of excluded middle. Hence the provability of the premises implies that a contradiction can always be reached whenever assumptions Γ and Δ are available. It turns out that the rule *Cut* is indeed admissible, and we can conclude that the law of excluded middle is built into the sequent calculus.

There is another way of interpreting a sequent $\Gamma \Longrightarrow \Delta$, which is called the *multi-conclusion view*. Under the multi-conclusion view, $A_1, \cdots, A_n \Longrightarrow B_1, \cdots, B_m$ means that if assumptions A_1 *true*, $\cdots$, "and" A_n *true* are available, a conclusion B_1 *true*, $\cdots$, "or" B_m *true* is provable. While all the rules in Figure 7.1 continue to make sense, they fail to express the essence of classical logic, namely the principle of proof by contradiction (or the law of excluded middle).

7.4 Double-negation translation and CPS translation

We have seen that classical logic is obtained by augmenting constructive logic with two new rules Contra $\uparrow$ and Contra $\downarrow$. As it comes with more inference rules, classical logic allows us to prove more truth judgments than constructive logic. That is, a proof of A *true* in constructive logic is a valid proof in classical logic as well, and therefore, if A *true* is provable in constructive logic, it is also provable in classical logic. The converse is certainly untrue, as evidenced by such judgments as $A \vee \neg A\ \mathit{true}$, $\neg\neg A \supset A\ \mathit{true}$, and $((A \supset B) \supset A) \supset A\ \mathit{true}$.

It is important that more judgments being provable does not necessarily mean more expressive power. For example, a system in which $\perp\ \mathit{true}$ is provable is totally useless (and is said to be inconsistent), even though every truth judgment is provable. In the case of classical logic, it allows more judgments to be provable, but is *less* expressive than constructive logic, which is capable of simulating classical logic via the *double-negation translation* to be explained below. In essence, constructive logic can make a finer distinction between truth and falsehood than classical logic (as it allows not only truth and falsehood but also "excluded middle").

We write A° for the proposition in constructive logic corresponding to proposition A in classical logic under the double-negation translation. The translation is structural except for $A \supset B$, which is translated to $A^\circ \supset \neg\neg B^\circ$:

$$\begin{array}{rcl}(A \wedge B)^\circ & = & A^\circ \wedge B^\circ \\ (A \vee B)^\circ & = & A^\circ \vee B^\circ \\ \top^\circ & = & \top \\ \perp^\circ & = & \perp \\ (A \supset B)^\circ & = & A^\circ \supset \neg\neg B^\circ \\ P^\circ & = & P\end{array}$$

An analogy in programming language theory is that an invocation of a "classical" function of type $A \supset B$ is in effect an invocation of a "constructive" function of type $A \supset \neg\neg B = A \supset ((B \supset \perp) \supset \perp)$, which returns an answer (of type $\perp$) when given an argument of type A and a return address (of type $B \supset \perp$) expecting an argument of type B.

Theorem 7.2 shows that classical logic is embedded in constructive logic via the double-negation translation, where we use the subscript I in a hypothetical judgment to indicate that it is valid only in *I*ntuitionistic logic, which is another name for constructive logic. Antecedents in hypothetical judgments are translated as follows:

$$\begin{array}{c}\Gamma^\circ = \{A^\circ\ \mathit{true} \mid A\ \mathit{true} \in \Gamma\} \\ \neg\Delta^\circ = \{\neg A^\circ\ \mathit{true} \mid A\ \mathit{false} \in \Delta\}\end{array}$$

Theorem 7.2 (Embedding of classical logic in constructive logic). *If* $\Gamma; \Delta \vdash_{\mathsf{K}} C\ \mathit{true}$*, then* $\Gamma^\circ, \neg\Delta^\circ \vdash_{\mathsf{I}} \neg\neg C^\circ\ \mathit{true}$.

An immediate corollary of Theorem 7.2 is that classical logic is relatively consistent with constructive logic in the sense that classical logic is consistent (*i.e.*, $\perp\ \mathit{true}$ is not provable) if and only if constructive logic is consistent, since $\neg\neg\perp$ is logically equivalent to $\perp$. As we have shown that constructive logic is consistent (*i.e.*, $\perp\ \mathit{true}$ is unprovable), we conclude that classical logic is also consistent.

Instead of proving Theorem 7.2 directly, we give another translation that converts a proof term M of type A in classical logic into a proof term M° of type A° in constructive logic. The translation is usually called *the CPS (Continuation-Passing Style) translation*, which enables us to simulate $\mathsf{callcc}\ x : A\ \mathit{false}.\, M$ and $\mathsf{throw}\ M\ \mathsf{to}\ x$ with λ-abstractions. The main idea in the CPS translation is to interpret $\neg A = A \supset \perp$ as the type of a *continuation* for type A, which, when invoked with an argument of type A, initiates the rest of the evaluation. Note that an invocation of a continuation conceptually returns an "answer" but actually never returns, for if it did, it would return a value of type $\perp$, which is impossible.

The CPS translation is obtained by translating a proof of $\Gamma; \Delta \vdash_{\mathsf{K}} M : C$ to a proof of $\Gamma^\circ, \neg\Delta^\circ \vdash_{\mathsf{I}} M^\circ : \neg\neg C^\circ$ where Γ° and $\neg\Delta^\circ$ are defined as follows:

$$\begin{array}{rcl}\Gamma^\circ & = & \{x : A^\circ \mid x : A \in \Gamma\} \\ \neg\Delta^\circ & = & \{x : \neg A^\circ \mid x : A\ \mathit{false} \in \Delta\}\end{array}$$

Theorem 7.3 (CPS translation). *If* $\Gamma; \Delta \vdash_{\mathsf{K}} M : C$*, there exists a proof term* M° *such that* $\Gamma^\circ, \neg\Delta^\circ \vdash_{\mathsf{I}} M^\circ : \neg\neg C^\circ$.

Proof. By induction on the structure of the proof of $\Gamma; \Delta \vdash_{\mathsf{K}} M : C$. The proof reuses metavariables M and C.

In each case, we only specify M°, which can be shown to satisfy $\Gamma^\circ, \neg\Delta^\circ \vdash_{\mathsf{I}} M^\circ : \neg\neg C^\circ$ by straightforward structural induction. M° is given as a λ-abstraction $\lambda k\!:\!C^\circ \supset \perp. \cdots$ (or equivalently $\lambda k\!:\!\neg C^\circ. \cdots$), where k can be thought of as a continuation expecting the result of evaluating M. A typical pattern in the CPS translation is that a proof term of type $\perp$ (returning an "answer") is built from a proof term N of type A by applying N° to a continuation $\lambda x\!:\!A^\circ.\, N'$ (as in $N^\circ\ (\lambda x\!:\!A^\circ.\, N')$) so that x is bound to the result of evaluating N and the evaluation of N' returns the final "answer."

Case $\dfrac{x : A \in \Gamma}{\Gamma; \Delta \vdash_{\mathsf{K}} x : A}\ \mathsf{Hyp}$

$x^\circ = \lambda k{:}A^\circ \supset \bot.\, k\ x$

Case $\dfrac{\Gamma; \Delta \vdash_{\mathsf{K}} M : A \quad \Gamma; \Delta \vdash_{\mathsf{K}} N : B}{\Gamma; \Delta \vdash_{\mathsf{K}} (M, N) : A \wedge B}\ \wedge\mathsf{I}$

$(M, N)^\circ = \lambda k{:}(A^\circ \wedge B^\circ) \supset \bot.\, M^\circ\ (\lambda x{:}A^\circ.\, N^\circ\ (\lambda y{:}B^\circ.\, k\ (x, y)))$

Case $\dfrac{\Gamma; \Delta \vdash_{\mathsf{K}} M : A \wedge B}{\Gamma; \Delta \vdash_{\mathsf{K}} \mathsf{fst}\ M : A}\ \wedge\mathsf{E_L}$

$(\mathsf{fst}\ M)^\circ = \lambda k{:}A^\circ \supset \bot.\, M^\circ\ (\lambda x{:}A^\circ \wedge B^\circ.\, k\ (\mathsf{fst}\ x))$

Case $\dfrac{\Gamma; \Delta \vdash_{\mathsf{K}} M : A \wedge B}{\Gamma; \Delta \vdash_{\mathsf{K}} \mathsf{snd}\ M : B}\ \wedge\mathsf{E_R}$

$(\mathsf{snd}\ M)^\circ = \lambda k{:}B^\circ \supset \bot.\, M^\circ\ (\lambda x{:}A^\circ \wedge B^\circ.\, k\ (\mathsf{snd}\ x))$

Case $\dfrac{\Gamma; \Delta \vdash_{\mathsf{K}} M : A}{\Gamma; \Delta \vdash_{\mathsf{K}} \mathsf{inl}_B\ M : A \vee B}\ \vee\mathsf{I_L}$

$(\mathsf{inl}_B\ M)^\circ = \lambda k{:}(A^\circ \vee B^\circ) \supset \bot.\, M^\circ\ (\lambda x{:}A^\circ.\, k\ (\mathsf{inl}_{B^\circ}\ x))$

Case $\dfrac{\Gamma; \Delta \vdash_{\mathsf{K}} M : B}{\Gamma; \Delta \vdash_{\mathsf{K}} \mathsf{inr}_A\ M : A \vee B}\ \vee\mathsf{I_R}$

$(\mathsf{inr}_A\ M)^\circ = \lambda k{:}(A^\circ \vee B^\circ) \supset \bot.\, M^\circ\ (\lambda x{:}B^\circ.\, k\ (\mathsf{inr}_{A^\circ}\ x))$

Case $\dfrac{\Gamma; \Delta \vdash_{\mathsf{K}} M : A \vee B \quad \Gamma, x_1 : A; \Delta \vdash_{\mathsf{K}} N_1 : C \quad \Gamma, x_2 : B; \Delta \vdash_{\mathsf{K}} N_2 : C}{\Gamma; \Delta \vdash_{\mathsf{K}} \mathsf{case}\ M\ \mathsf{of}\ \mathsf{inl}\ x_1.\, N_1 \mid \mathsf{inr}\ x_2.\, N_2 : C}\ \vee\mathsf{I_R}$

$(\mathsf{case}\ M\ \mathsf{of}\ \mathsf{inl}\ x_1.\, N_1 \mid \mathsf{inr}\ x_2.\, N_2)^\circ = \lambda k{:}C^\circ \supset \bot.\, M^\circ\ (\lambda x{:}A^\circ \vee B^\circ.\, \mathsf{case}\ x\ \mathsf{of}\ \mathsf{inl}\ x_1.\, {N_1}^\circ\ k \mid \mathsf{inr}\ x_2.\, {N_2}^\circ\ k)$

Case $\dfrac{\Gamma, x : A; \Delta \vdash_{\mathsf{K}} M : B}{\Gamma; \Delta \vdash_{\mathsf{K}} \lambda x{:}A.\, M : A \supset B}\ \supset\mathsf{I}$

$(\lambda x{:}A.\, M)^\circ = \lambda k{:}(A^\circ \supset \neg\neg B^\circ) \supset \bot.\, k\ (\lambda x{:}A^\circ.\, M^\circ)$

Case $\dfrac{\Gamma; \Delta \vdash_{\mathsf{K}} M : A \supset B \quad \Gamma; \Delta \vdash_{\mathsf{K}} N : A}{\Gamma; \Delta \vdash_{\mathsf{K}} M\ N : B}\ \supset\mathsf{E}$

$(M\ N)^\circ = \lambda k{:}B^\circ \supset \bot.\, M^\circ\ (\lambda x{:}A^\circ \supset \neg\neg B^\circ.\, N^\circ\ (\lambda y{:}A^\circ.\, x\ y\ k))$

Case $\dfrac{}{\Gamma; \Delta \vdash_{\mathsf{K}} () : \top}\ \top\mathsf{I}$

$()^\circ = \lambda k{:}\top^\circ \supset \bot.\, k\ ()$

Case $\dfrac{\Gamma; \Delta \vdash_{\mathsf{K}} M : \bot}{\Gamma; \Delta \vdash_{\mathsf{K}} \mathsf{abort}_C\ M : C}\ \bot\mathsf{E}$

$(\mathsf{abort}_C\ M)^\circ = \lambda k{:}C^\circ \supset \bot.\, M^\circ\ (\lambda x{:}\bot.\, x)$

Case $\dfrac{\Gamma; \Delta, x : A\ \mathit{false} \vdash_{\mathsf{K}} M : A}{\Gamma; \Delta \vdash_{\mathsf{K}} \mathsf{callcc}\ x : A\ \mathit{false}.\, M : A}\ \mathsf{Callcc}$

$(\mathsf{callcc}\ x : A\ \mathit{false}.\, M)^\circ = \lambda k{:}A^\circ \supset \bot.\, [k/x]M^\circ\ k$

Case $\dfrac{\Gamma; \Delta \vdash_{\mathsf{K}} M : A \quad x : A\ \mathit{false} \in \Delta}{\Gamma; \Delta \vdash_{\mathsf{K}} \mathsf{throw}\ M\ \mathsf{to}\ x : C}\ \mathsf{Throw}$

$(\mathsf{throw}\ M\ \mathsf{to}\ x)^\circ = \lambda k{:}C^\circ \supset \bot.\, M^\circ\ x$ □