Lecture Notes on Type Theory

Erik Palmgren

Fall term 2013
(Draft. Version January 28, 2014)

.

Lectures notes for the advanced level course Type Theory given at the Department of Mathematics at Stockholm University Fall term 2013.

Acknowledgements: I am grateful to Peter Dybjer and Thierry Coquand for permission to use the material of chapter 1 from the introduction of (Coquand, Dybjer and Palmgren 2007). Thanks goes to Oliver Krüger for corrections of an earlier version of these notes.

Chapter 1

Introduction

1.1 Constructivism: critique of classical logic in mathematics

Constructivism has a history well before the advent of interest in computability in mathematics. Around the turn of the century 1899-1900 there arised doubts about the consistency of the axiomatic and logical foundations for the abstract mathematics that had started to develop with the support of set theory. Bertrand Russell had 1903 with his well-known paradox shown that unrestricted formation of sets from concepts may lead to outright contradictions. (*Is there a set that consists of all sets that are not members of themselves?* Answer: No.) Also other principles in use, such as the Axiom of Choice, had been discovered to have unintuitive and unnatural consequences, even though no paradoxes where known to arise from them. Ernst Zermelo showed 1908 that the set of real numbers may be well-ordered. It was among many mathematicians considered as a serious scientific crisis of the subject, known as the *Grundlagenkrisis*. In that mood of time the outstanding Dutch mathematician, and founder of modern topology, L.E.J. Brouwer started a critical examination and reconstruction of the foundations for mathematics, which went further than previous attempts, and included the very logic and not only the axioms. By the introduction of his *intuitionistic mathematics* he wanted to put mathematics on a secure and intuitive footing. His idea was that every proof must built on a so-called *mental construction*. At that time (1910) there were no of course programming languages, and not even a mathematical notion of algorithm, but it turned out that his notion of mental construction could be interpreted as algorithmic construction in a precise way. This requirement led Brouwer to reject a logical law that had been taken for granted since Aristotle, namely the *Principle of Excluded Middle* or *Tertium Non Datur*. This states that for every proposition A, either A is true or A is false, in logical symbols:

$$A \vee \neg A \qquad \text{(PEM)}.$$

For concrete, finitely checkable, propositions there was no reason to doubt the law. The problematic case, according to Brouwer, is when A contains quantification over an infinite set, for instance the set of integers.

Brouwer demonstrated that it was possible to develop mathematics also without the principle of excluded middle, and that it in many cases lead to more informative and insightful proofs. The immediate followers of Brouwer were not very numerous, and his use of the theory of choice sequences, which is inconsistent with classical logic, repelled many mathematicians from his philosophy. Later developments of constructive mathematics avoid this theory, and its results are immediately understandable and acceptable to mainstream mathematics (see Bishop-Bridges 1985, Bridges-Richman 1987).

1.2 Non-constructive proofs

We shall illustrate how the principle of excluded middle is used in some non-constructive existence proofs, and how this may lead to the loss of algorithmic content. Here is a famous standard example, chosen for its simplicity rather than mathematical interest.

Proposition 1.2.1. *There are irrational numbers a and b such that a^b is an rational number.*

Proof. The number $\sqrt{2}$ is irrational (non-rational). Consider the number $\sqrt{2}^{\sqrt{2}}$. By the principle of excluded middle it is either rational or irrational. If it is rational, we may finish the proof by exhibiting $a = b = \sqrt{2}$. If it is irrational, let $a = \sqrt{2}^{\sqrt{2}}$ and $b = \sqrt{2}$. Then a^b is rational, indeed we have

$$a^b = (\sqrt{2}^{\sqrt{2}})^{\sqrt{2}} = (\sqrt{2})^2 = 2.$$

$\square$

Note that in this existence proof it was not decided which of the numbers $a = \sqrt{2}$ or $a = \sqrt{2}^{\sqrt{2}}$, with $b = \sqrt{2}$, that actually gives the intended example. One may actually show, using deeper mathematical theory, that $\sqrt{2}^{\sqrt{2}}$ is irrational, but the standard proof requires several pages. An alternative example could be given by the numbers $a = e$ and $b = \ln 2$, but the proofs of their irrationality are far more complicated than that for $\sqrt{2}$.

Many classical existence proofs are indirect, and start "Suppose that there is no object x such that ...". The rest of the proof is devoted to show that a contradiction arises from this assumption. Here is a simple example involving infinite sequences.

Proposition 1.2.2. *Each infinite sequence of natural numbers*

$$a_1, a_2, a_3, \ldots, a_k, \ldots$$

has a minimal term, i.e. there is some n such that $a_n \leq a_k$ for all k.

Proof. Suppose that there is no minimal term in the sequence. For every index k we may then find $k' > k$ with $a_k > a_{k'}$. Let $k_1 = 1$. By assumption there is then $k_2 > k_1$ with $a_{k_1} > a_{k_2}$. Again, by the assumption, we find $k_3 > k_2$ such that $a_{k_2} > a_{k_3}$. Continuing in this manner we obtain an infinite sequence $k_1 < k_2 < k_3 < \cdots$ such that

$$a_{k_1} > a_{k_2} > a_{k_3} > \cdots .$$

This sequence of natural numbers decrease by at least one unit for every step, so

$$a_{k_n} \le a_1 - (n-1).$$

Putting $n = a_1 + 2$, we thereby find a term in the sequence less than 0, which is impossible. □

This existence proof gives no information whatsoever where the minimal term is to be found. Not only the proof of this result is non-constructive, but the result is essentially non-constructive. Consider for example the sequence obtained in the following way: given a description of a Turing machine and an input string, let a_k be 0 if the machine has terminated after k steps, and 1 otherwise. If we would be able to find the minimum of this sequence, algorithmically, we could also solve the general halting problem algorithmically. However this is known to be impossible.

This kind of “information-less” existence proofs are not uncommon in mathematical analysis. For instance certain results on the existence of solutions to differential equations build on such proofs (Cauchy-Peano existence proof). There are rather simple examples of ordinary differential equations, whose solutions cannot computed from the parameters, though there are theoretical solutions (see Beeson 1985)

Constructive mathematics and logic are founded on the idea that existence is taken more seriously than in classical mathematics: to prove that a certain object exists is the same as giving a method for constructing it.

1.2.1 Exercises

1. Recall that an algebraic real number is a real number which is a root of a polynomial with integer coefficients. Such numbers are closed under the usual arithmetical operations. Any real number which is not algebraic, is called *transcendental.* The numbers e and π are transcendental. Prove, using only these facts, that $e+\pi$ or $e-\pi$ is transcendental. (It is unknown which one it is, or if whether both are transcendental).

2. (König's lemma). A finite string over the alphabet $\{l, r\}$ is regarded as describing a path in a binary tree, starting from the root. Suppose that P is an infinite set of such paths. Show that there is an infinite string

$$d_1 d_2 d_3 \cdots$$

such that for every n, the string $d_1 d_2 \cdots d_n$ is an initial segment of some path in P.

3. (Brouwer's Fan Theorem). Consider a set of paths P such that if w is in P, then so is each of its initial segments, i.e. if $w = uv \in P$, then $u \in P$. An infinite path $d = d_1 d_2 d_3 \cdots$ is said to be *barred by* P, if there is some n such that $d_1 \cdots d_n \notin P$. Show that if every infinite path d is barred by P, then P must be finite.

1.3 Constructive interpretation of the logical constants

According to Brouwer's idea every mathematical theorem A must rest on a (mental) construction a. The construction a may be regarded as a *witness* to the truth of A. This basic constructive interpretation was further clarified by Arend Heyting (a student of Brouwer) and by A.N. Kolmogorov (the founder of modern probability theory). Hence it is called the *Brouwer-Heyting-Kolmogorov-interpretation,* or BHK-interpretation for short.

It should be pointed out that there are some limits on what may be regarded as a construction. One may be lead to think that this is a construction

$$f(n) = \begin{cases} 1 & \text{there are } n \text{ consecutive 7's in the decimal expansion of } \pi, \\ 0 & \text{otherwise.} \end{cases}$$

This is, a priori, not a constructive function as no one has (yet) found an algorithm that can decide whether there are n consecutive 7's in the decimal expansion of π or not. Case distinction is allowed only if it can be decided effectively which case is true, for given parameters.

BHK-interpretation. We explain what it means that a is a *witness* to the truth of the proposition A, by induction on the form of A. This will be expressed more briefly as *a is a witness to A*, or that *a testifies A*.

- $\perp$ has no witnesses.
- p testifies $A \wedge B$ iff p is a pair (a, b) where a testifies A and b testifies B.
- p testifies $A \longrightarrow B$ iff p is a method which to each witness a to A gives a witness $p(a)$ to B.
- p testifies $A \vee B$ iff p has the form $\mathsf{inl}(a)$, in which case a testifies A, or p has the form $\mathsf{inr}(b)$, in which case b testifies B

- p testifies $(\forall x \in S)A(x)$ iff p is a method which to each element $d \in S$, provides a witness $p(d)$ to $A(d)$.
- p testifies $(\exists x \in S)A(x)$ iff p is a pair (d, q) consisting of $d \in S$ and a witness q to $A(d)$.

We may add for numeral equations $s = t$ the interpretation

- p testifies $s = t$ iff p is 0 and s and t are computed to the same thing.

Remark 1.3.1. The witnesses $p, q, \ldots$ are commonly called *proof constructions*, *proof objects* or simply *proofs*. We use initially the term witness to clearly distinguish from derivation in a formal system.

Example 1.3.2. A witness p to a proposition of the form

$$(\forall x \in S)(\exists y \in T)R(x, y)$$

is thus a method p which to each $a \in S$ assigns a witness $p(a)$ to

$$(\exists y \in T)R(a, y).$$

This witnesss in turn is a pair $p(a) = (p_1(a), p_2(a))$ where $p_1(a) \in T$ and $p_2(a)$ is witness to

$$R(a, p_1(a)).$$

Thus

$$a \mapsto p_1(a)$$

is a method for computing a $y \in T$ from $x \in S$, to satisfy $R(x, y)$.

Remark 1.3.3. *The Principle of Excluded Middle* (PEM)

$$A \vee \neg A$$

is not obviously valid under the BHK-interpretation, since we would need to find a method, which given the parameters in A, decides whether A is valid or not. If we restrict the possible constructions to computable functions, we may actually show that PEM is not constructively true. It is known that there is a primitive recursive function T such that $T(e, x, t) = 1$ in case t describes a terminating computation (t is, so to say, the complete "trace" of the computation) for the Turing machine e with input x, and having the value $T(e, x, t) = 0$ otherwise. By a suitable coding, the arguments to T may be regarded as natural numbers. The halting problem for e and x may now be expressed by the formula

$$H(e, x) =_{\text{def}} (\exists t \in \mathbb{N})\, T(e, x, t) = 1.$$

According to PEM

$$(\forall e \in \mathbb{N})(\forall x \in \mathbb{N})\, H(e,x) \vee \neg H(e,x).$$

If this proposition were to have a computable witness, then we could decide the halting problem, contrary to Turing's well-known result that this is algorithmically undecidable. The principle of indirect proof, *reductio ad absurdum* (RAA)

$$\neg\neg A \longrightarrow A$$

can be shown to be equivalent to PEM within intuitionistic logic, so it is not valid under the BHK-interpretation either.

Chapter 2

Martin-Löf type theory

In this chapter we follow the exposition of (Martin-Löf 1984) quite closely regarding the foundations of type theory.

2.1 Fundamental notions of Martin-Löf type theory

In Martin-Löf (1984) a general philosophy of logic is presented which revives the notion of judgement. A *judgement* is an act of knowledge, for instance asserting that something holds. When reasoning mathematically we are making a sequence of judgements about mathematical objects. One kind of judgement may be to state that some mathematical statement is true, another kind of judgement may be to state that something is a mathematical object, is a set, for instance. The logical rules give a method for producing correct judgements from earlier judgements. The judgements obtained by such rules may be presented in tree form

$$\dfrac{\dfrac{J_1 \quad J_2}{J_3}\,r_1 \qquad \dfrac{\dfrac{J_4}{J_5}\,r_5 \quad J_6}{J_7}\,r_3}{J_8}\,r_4$$

or in sequential form

(1)	J_1	axiom
(2)	J_2	axiom
(3)	J_3	by rule r_1 from (1) and (2)
(4)	J_4	axiom
(5)	J_5	by rule r_2 from (4)
(6)	J_6	axiom
(7)	J_7	by rule r_3 from(5) and (6)
(8)	J_8	by rule r_4 from (3) and (7)

The latter form is common in mathematical arguments, but we will prefer using the less verbose tree form. Such a tree formed by logical rules from axioms is a *derivation.* We have presented a very abstract and general notion of derivation here, mainly to give an idea of the shape of things to come.

2.1.1 Judgements in first-order logic

First-order reasoning may be presented using a single kind of judgement

> the proposition B is true under the hypothesis that the propositions $A_1, \dots, A_n$ are all true.

We write this as a so-called *Gentzen sequent* (not to be confused with derivability)

$$A_1, \dots, A_n \vdash B$$

When $n = 0$, then $\vdash B$ states that B is true without any assumptions.

The familiar rule for conjunctive introduction then becomes

$$\frac{A_1, \dots, A_n \vdash B \quad A_1, \dots, A_n \vdash C}{A_1, \dots, A_n \vdash B \wedge C}\ (\wedge I)$$

Usually the sequence of assumption $A_1, \dots, A_n$ are abbreviated by some Greek capital letter $\Gamma, \Delta, \dots$ when presenting the rules.

The rule for implicational introduction is

$$\frac{\Gamma, B \vdash C}{\Gamma \vdash B \to C}\ (\to I)$$

and implicational elimination (modus ponens)

$$\frac{\Gamma \vdash B \to C \quad \Gamma \vdash B}{\Gamma \vdash C}\ (\to E).$$

Moreover for any sequence of well-formed formulas $A_1, \dots, A_n$ and any $1 \leq i \leq n$ there is an assumption axiom

$$\frac{}{A_1, \dots, A_n \vdash A_i}\ (\text{as. } i)$$

The full system will be presented later, but for now we give an example of a derivation:

$$\dfrac{\dfrac{\dfrac{\dfrac{}{A \wedge B \to C, A, B \vdash A \wedge B \to C}\ (\text{as. }1) \quad \dfrac{\dfrac{}{A \wedge B \to C, A, B \vdash A}\ (\text{as. }2) \quad \dfrac{}{A \wedge B \to C, A, B \vdash B}\ (\text{as. }3)}{A \wedge B \to C, A, B \vdash A \wedge B}\ (\wedge I)}{A \wedge B \to C, A, B \vdash C}\ (\to E)}{A \wedge B \to C, A \vdash B \to C}\ (\to I)}{A \wedge B \to C \vdash A \to (B \to C)}\ (\to I)$$

One may regard this as a standard natural deduction proof, only that the open assumptions have been listed explicitly to the left of $\vdash$. For comparison the standard proof is

$$\dfrac{\dfrac{A \wedge B \to C \qquad \dfrac{\overline{A}^{\,1} \wedge \overline{B}^{\,2}}{A \wedge B}\,(\wedge I)}{C}\,(\to E)}{\dfrac{B \to C}{A \to (B \to C)}\,(\to I, 1)}\,(\to I, 2) \quad .$$

The sequent-style formulation will be especially useful in type theory.

Martin-Löf type theory is a much more complicated system than first-order logic. One reason is that more information is carried around in the derivations due to the identification of propositions and types. Another reason is that the syntax is more involved. For instance, the well-formed formulas have to be generated simultaneously with the provably true formulas.

2.1.2 Judgement forms in type theory

The type theory of (Martin-Löf 1984, p.5) has four basic forms of judgements

1. A is a set (abbreviated A set)
2. A and B are equal sets (abbreviated $A = B$)
3. a is an element of the set A (abbreviated $a : A$)
4. a and b are equal elements of the set A (abbreviated $a = b : A$)

Note: Martin-Löf (1984) uses the term *set* instead of the more widely used term *type.*

We recall three diverse interpretations of the judgement forms from (Martin-Löf 1984, p.5):

A set	$a : A$	
A is a set	a is an element of the set A	A is non-empty (or inhabited)
A is a proposition	a is a proof (construction) of the proposition A	A is true
A is a problem (task)	a is a method of solving the problem (doing the task) A	A is solvable

The first two lines indicates the so-called *Propositions-as-Sets* (or Propositions-as-Types) principle. Sometimes it is also known as the Curry-Howard correspondence (or Curry-Howard isomorphism). A proposition A can thus be regarded as the set of all possible proof constructions of A.

2.1.3 Explanation of the judgement forms

We start with a quote from E. Bishop (1967, Chapter 1) in which he defines a constructive notion of set (Bishop set, now sometimes known as a *setoid*):

> A set is not an entity which has an ideal existence: a set exists only when it has been defined. To define a set we prescribe, at least implicitly, what we (the constructing intelligence) must do in order to construct an element of the set, and what we must do to show that two elements of the set are equal.

We recount Martin-Löf's explanation of the judgments forms and in particular that of a set. Note that these are explanations of the closed versions (no free variables) of the judgement forms. Later we give the explanation of the hypothetical versions.

1. The meaning of the judgement form A set

As Martin-Löf (1984, p. 8) writes, a set A "is defined by prescribing how a canonical element of A is formed as well as how two equal canonical elements of A are formed".

Example. The canonical elements of the set of natural numbers are formed by the introduction rules

$$\frac{}{0 : \mathrm{N}} \qquad \frac{a : \mathrm{N}}{S(a) : \mathrm{N}}.$$

and equality of canonical elements are given by

$$\frac{}{0 = 0 : \mathrm{N}} \qquad \frac{a = b : \mathrm{N}}{S(a) = S(b) : \mathrm{N}}$$

Elements that are formed directly by introduction rules are called *canonical.* (In functional computer languages constants such as 0 and S are often called *constructors.*) There may however be non-canonical elements in N: $S(0) + S(0)$ does not arise directly from an introduction rule, but can be calculated to a canonical element, viz. $S(S(0) + 0)$.

Example. Another example is the disjoint union $A+B$ of two sets A and B whose introduction rules are

$$\frac{a : A}{\mathsf{inl}(a) : A + B} \qquad \frac{b : B}{\mathsf{inr}(b) : A + B}.$$

The equality of canonical elements is given by

$$\frac{a = c : A}{\mathsf{inl}(a) = \mathsf{inl}(c) : A + B} \qquad \frac{b = d : B}{\mathsf{inr}(b) = \mathsf{inr}(d) : A + B}$$

2. The meaning of the judgement form $A = B$: Two sets A and B are equal if each canonical element a of A is also a canonical element of B, and vice versa, if each canonical element a of B is also a canonical element of A:

$$\frac{a : B}{a : A} \qquad \text{and} \qquad \frac{a : A}{a : B}$$

and moreover if for all canonical a and b:

$$\frac{a = b : B}{a = b : A} \qquad \text{and} \qquad \frac{a = b : A}{a = b : B}.$$

Note that this judgement form presupposes that A and B are sets.

3. The meaning of the judgement form $a : A$ is: An element of a set is a method (or a program) which, when executed, yields a canonical element in A as a result.

This judgement form presupposes that A is a set.

4. The meaning of the judgement form $a = b : A$ is: Two elements a and b of A are equal if when they are executed yield equal canonical elements as results.

This judgement form presupposes that $a : A$ and $b : A$.

This gives the general meanings to the judgement forms. When introducing a new set we check that it makes sense by verifying that its rules are valid under the above interpretations. However, in general, we need also to make use of the extension of these explanations to hypothetical judgements.

We can already now verify the validity of some general rules for equality using this meaning explanation, namely the equivalence relation rules

$$\frac{A \text{ set}}{A = A} \qquad \frac{A = B}{B = A} \qquad \frac{A = B \quad B = C}{A = C}$$

and

$$\frac{a : A}{a = a : A} \qquad \frac{a = b : A}{b = a : A} \qquad \frac{a = b : A \quad b = c : A}{a = c : A}$$

and the set-equality rules

$$\frac{a : A \quad A = B}{a : B} \qquad \frac{a = b : A \quad A = B}{a = b : B} \; .$$

Let us verify the left set-equality rule. Suppose $a : A$ and $A = B$ holds. The judgement $a : A$ means that a is a method that computes to a canonical element c of A. According to the meaning of $A = B$, c is thus also a canonical element of B. But since a computes to c, this means that $a : B$.

For the remaining rules see (Martin-Löf 1984, pp. 14–15).

2.2 Metamathematical intermezzo: abstract computation relations

The following considerations may be helpful to understand the notion of computation referred to in (Martin-Löf 1984). These will be developed more fully when we come to the study of λ-calculus.

Let S be a set and let $\leadsto$ be a binary relation on S. We shall think of S as a set of abstract expressions and $a \leadsto b$ as the relation a may be *computed, or reduced to* b. The pair $(S, \leadsto)$ is called an *abstract computation relation* (ACR). An element $a \in S$ is *normal* if for any $b \in S$

$$a \leadsto b \text{ implies } a = b.$$

We may think of this as saying that a cannot be computed further, or computes *only* to it self. A normal element may be regarded as a *value* or *end-result.* If $a \leadsto b$, and b is normal, we say that b *is a value of* a.

An important property satisfied by some ACRs is *confluence.* $(S, \leadsto)$ is *confluent* (or *Church-Rosser*) if for any $a, b, c \in S$ with $a \leadsto b$ and $a \leadsto c$, there is $d \in S$ such that $b \leadsto d$ and $c \leadsto d$.

Proposition 2.2.1. *Let $(S, \leadsto)$ be a confluent ACR. If a computes to normal b and c, then $b = c$.*

Proof. Suppose $a \leadsto b$ and $a \leadsto c$. By confluence there is d with $b \leadsto d$ and $c \leadsto d$. But since b is normal, $b = d$, and since c is also normal, $c = d$. $\square$

Thus in a confluent ACR, values of elements are unique if they exist. Another important property of confluent ACRs is the following. For an ACR $(S, \leadsto)$ define a new binary relation *a is convertible to b*, written $a \approx b$, to hold if there is $c \in S$ with $a \leadsto c$ and $b \leadsto c$.

Theorem 2.2.2. *If $(S, \leadsto)$ is confluent and $\leadsto$ is total and transitive as a relation, then $\approx$ is an equivalence relation.*

Proof. The relation $\approx$ is obviously symmetric. It is reflexive since for any a with there is some b with $a \leadsto b$, so $a \approx a$. Suppose that $a \approx b$ and $b \approx c$, so there are x and y with $a \leadsto x$, $b \leadsto x$, $b \leadsto y$ and $c \leadsto y$. By confluence there is z such that $x \leadsto z$ and $y \leadsto z$. Hence by transitivity $a \leadsto z$ and $c \leadsto z$, so $a \approx c$ as required. $\square$

Let $(S, \leadsto)$ be an ACR. We say that $a \in S$ is *normalizable* or has normal form, if there is some normal b with $a \leadsto b$. There is stronger notion, $a \in S$ *strongly normalizable* if for any infinite sequence (a_n) starting with a contains a normal element, i.e. for some n

$$a = a_1 \;\; a_2 \;\; a_3 \;\; \cdots a_{n-1} \;\; a_n = a_{n+1} = a_{n+2} = \ldots$$

An ACR is *normalizing* if every element is normalizable, and *strongly normalizing* if every element is strongly normalizable.

2.2.1 Lazy evaluation

We give an extended example involving so-called *lazy evaluation* of arithmetical expressions. The evaluation of an expression e generally only divulges whether its value is 0 or of the form $S(e')$, that is whether it is 0 or greater than 0. In the latter case evaluation can be applied to e' to reveal whether its value is $S(0)$ (one) or greater than one. The process can be repeated as long as necessary to figure out the result.

Let A be the set of arithmetical expressions that can be formed from 0, + and $S(\,)$ (add one). Formally we define A as the smallest set that satisfies

(a) $0 \in A$

(b) if $x \in A$, then $S(x) \in A$

(c) if $x, y \in A$, then $(x + y) \in A$.

Note well that $((x+y)+z)$ is different from $(x+(y+z))$. Otherwise we drop parentheses when possible.

Let $C \subseteq A$ be the subset of expressions of *canonical form* (direct results of introduction rules for $\mathbb{N}$) defined by

$$C = \{0\} \cup \{S(a) : a \in A\}.$$

Define $\rightsquigarrow$ to be smallest binary relation on A such that

$$\frac{a \in C}{a \rightsquigarrow a}\ (\text{ref}) \tag{2.1}$$

$$\frac{b \rightsquigarrow 0 \quad a \rightsquigarrow c}{(a+b) \rightsquigarrow c}\ (\text{base}) \tag{2.2}$$

$$\frac{b \rightsquigarrow S(d)}{(a+b) \rightsquigarrow S(a+d)}\ (\text{succ}) \tag{2.3}$$

That the relation is the smallest is the same as saying that $a \rightsquigarrow b$ is true if and only if it witnessed by a finite derivation of this fact. For instance $S(0) + S(0) \rightsquigarrow S(S(0) + 0)$ holds since

$$\frac{\dfrac{S(0) \in C}{S(0) \rightsquigarrow S(0)}\ (\text{ref})}{S(0) + S(0) \rightsquigarrow S(S(0) + 0)}\ (\text{succ}).$$

Another example: $(0+S(0))+0 \rightsquigarrow S(0+0)$ is true because

$$\dfrac{\dfrac{0 \in C}{0 \rightsquigarrow 0} \quad \dfrac{\dfrac{S(0) \in C}{S(0) \rightsquigarrow S(0)}\ \text{(ref)}}{0+S(0) \rightsquigarrow S(0+0)}\ \text{(succ)}}{(0+S(0))+0 \rightsquigarrow S(0+0)}\ \text{(base)}$$

Let $A_{\text{lazy}} = (A, \rightsquigarrow)$ — the ACR of *lazy sums*. This computation rule have several good and expected properties

Proposition 2.2.3. *If $x \rightsquigarrow y$, then $y \in C$.*

Proof. We do induction on the height of derivations of $x \rightsquigarrow y$. If $x \rightsquigarrow y$ is the conclusion of a derivation ending with a (ref)– or a (succ)–rule, then $y \in C$ by definition. Consider the remaining case where $x \rightsquigarrow y$ has been derived by a (base)–rule, with $x = (a+b)$,

$$\dfrac{b \rightsquigarrow 0 \quad a \rightsquigarrow y}{(a+b) \rightsquigarrow y}\ \text{(base)}.$$

Then clearly $a \rightsquigarrow y$ has been derived by a shorter derivation, so by inductive hypothesis $y \in C$ as desired. □

Proposition 2.2.4. *For every $x \in A$, there is $y \in C$ with $x \rightsquigarrow y$.*

Proof. Induction on the build-up of expressions in A. If $x = 0$ or $x = S(a)$ for some $a \in A$, then $x \in C$, so $x \rightsquigarrow x$ by the (ref)–rule. Suppose that $x = a + b$. By inductive hypothesis, $a \rightsquigarrow y$ and $b \rightsquigarrow z$ for some $y, z \in C$. If $z = 0$, we may derive:

$$\dfrac{b \rightsquigarrow 0 \quad a \rightsquigarrow y}{(a+b) \rightsquigarrow y}\ \text{(base)}.$$

Hence $(a+b) \rightsquigarrow y$ with $y \in C$. On the other hand, if $z = S(d)$ for some $d \in A$, then we may derive

$$\dfrac{b \rightsquigarrow S(d)}{(a+b) \rightsquigarrow S(a+d)}\ \text{(succ)}$$

Thus $(a+b) \rightsquigarrow S(a+d)$ and obviously $S(a+d) \in C$. In both cases for z we obtain a canonical expression y such that $x \rightsquigarrow y$. □

The computation rule is indeed deterministic:

Proposition 2.2.5. *If $x \rightsquigarrow u$ and $x \rightsquigarrow v$, then $u = v$.*

Proof. We prove this by a double induction on the derivations of $x \rightsquigarrow u$ and $x \rightsquigarrow v$. In case $x \in C$, then the only rule that can result in these conclusions is (ref), so $u = v$. Suppose instead that $x = (a+b)$. The two conclusions may be the result of four different combinations of rules (base)-(base), (base)-(succ), (succ)-(base) and (succ)-(succ). The first combination looks like this

$$\frac{b \rightsquigarrow 0 \quad a \rightsquigarrow u}{(a+b) \rightsquigarrow u}\,\text{(base)} \qquad \frac{b \rightsquigarrow 0 \quad a \rightsquigarrow v}{(a+b) \rightsquigarrow v}\,\text{(base)}.$$

By inductive hypothesis $u = v$, since $a \rightsquigarrow u$ and $a \rightsquigarrow v$ have shorter derivations. The second combination would be

$$\frac{b \rightsquigarrow 0 \quad a \rightsquigarrow u}{(a+b) \rightsquigarrow u}\,\text{(base)} \qquad \frac{b \rightsquigarrow S(d)}{(a+b) \rightsquigarrow v}\,\text{(succ)}.$$

By inductive hypothesis $0 = S(d)$, since $b \rightsquigarrow 0$ and $b \rightsquigarrow S(d)$ have shorter derivations. This is impossible. The third case is similarly ruled out. As for the final case it looks like this, with $u = S(a+d)$ and $v = S(a+d')$:

$$\frac{b \rightsquigarrow S(d)}{(a+b) \rightsquigarrow S(a+d)}\,\text{(succ)} \qquad \frac{b \rightsquigarrow S(d')}{(a+b) \rightsquigarrow S(a+d')}\,\text{(succ)}.$$

Thus $S(d) = S(d')$ by the inductive hypothesis, since $b \rightsquigarrow S(d)$ and $b \rightsquigarrow S(d')$. Hence also $d = d'$, and it follows that $u = v$. □

Corollary 2.2.6. $\rightsquigarrow$ *is confluent.*

Proof. Suppose $x \rightsquigarrow u$ and $x \rightsquigarrow v$. Then $u = v$ by (2.2.5). Since $v \in C$ by (2.2.3), we have $v \rightsquigarrow v$ and thus also $u \rightsquigarrow v$. Hence $\rightsquigarrow$ is confluent □

Corollary 2.2.7. $\rightsquigarrow$ *is transitive.*

Proof. Suppose $x \rightsquigarrow y$ and $y \rightsquigarrow z$. Then $y \in C$, so $y \rightsquigarrow y$. Hence $y = z$ by (2.2.5). Thus $x \rightsquigarrow z$ as required. □

We conclude thus by Theorem 2.2.2 that $\approx$ defines an equivalence relation on A, and moreover that for any $a \in A$ there is $c \in C$ with $a \approx c$.

Proposition 2.2.8. *For any $a \in A$, a is normal with respect to $\rightsquigarrow$ if and only if $a \in C$.*

Proof. Left as an exercise. □

Finally the computation rules are correct:

Proposition 2.2.9. *For any expressions $x, y \in S$, if $x \rightsquigarrow y$, then x and y are equal when evaluated to natural numbers in the usual recursive way.*

Proof. Straightforward induction on derivations. □

2.2.2 Exercises

1. Complete the proofs of Propositions 2.2.8 and 2.2.9.

2. Investigate what happens to the above results, if the rule

$$\frac{a \rightsquigarrow b}{S(a) \rightsquigarrow S(b)} \ (S-\xi)$$

is added to the definition of $\rightsquigarrow$ above. Clearly, we then have $S(0+0) \rightsquigarrow S(0)$. In particular, $S(0+0)$ is not normal any more.

2.3 Dependent Types and Hypothetical Judgements

2.3.1 Families of Sets

A central notion in Martin-Löf type theory is the notion of a dependent type or set. This is a type-theoretic version of the notion of a family of sets, which is often used in mathematics and usually written in the notation

$$\{A_i\}_{i\in I} \quad \text{or possibly,} \quad A_i \text{ is a set } (i \in I)$$

Here I is an index set, and $A_{(-)}$ assigns to each index $i \in I$ a set A_i. There are some well-known constructions associated with families of sets: the *generalized cartesian product* of the family

$$\prod_{i\in I} A_i = \{f : I \longrightarrow \cup_{i\in I} A_i : \text{for all } i \in I,\ f(i) \in A_i\}$$

and the *disjoint union* of the family

$$\sum_{i\in I} A_i = \{(i, a) : i \in I, a \in A_i\}.$$

Note that if $a_i \in A_i$ for each $i \in I$, then the function f defined by $f(i) = a_i$ is an element of $\prod_{i\in I} A_i$.

These constructions also have type-theoretic counterparts and turns out to be two of the main building blocks for both the logic and the objects of the theory.

Note that if $A_i = C$ for all $i \in I$, that is the family is constant, then as sets

$$\prod_{i\in I} A_i = (I \longrightarrow C) \qquad \sum_{i\in I} A_i = I \times C, \tag{2.4}$$

2.3.2 Exercises

1. Verify the equations in (2.4).

2. Suppose that $\{A_i\}_{i\in I}$ is a family of sets, and suppose that $\{B_{i,j}\}_{i\in I, j\in A_i}$ is a doubly indexed family of set. Prove (in informal set theory) that there is a function from the set
$$\prod_{i\in I}\sum_{j\in A_i} B_{i,j}$$
to the set
$$\sum_{f\in(\prod_{i\in I} A_i)}\prod_{i\in I} B_{i,f(i)}.$$
(Did you use the axiom of choice?)

2.3.3 Hypothetical judgements

For each of the four basic judgement forms $\mathcal{J}$ above we need to consider its hypothetical version

$$\mathcal{J} \quad (x_1 : A_1, x_2 : A_2, x_3 : A_3, \ldots, x_n : A_n). \tag{2.5}$$

The general idea of its interpretation is that $\mathcal{J}$ should hold for any instantiation of the parameters $x_1, \ldots, x_n$ to elements belonging to the appropriate sets, and that appropriate equalities between elements in the parameters should be respected in the judgement instances. This will be made precise below.

The part $(x_1 : A_1, x_2 : A_2, \ldots, x_n : A_n)$ of (2.5) is called the *context* in which the judgement $\mathcal{J}$ is made. By the propositions-as-sets principle the context may be regarded as a sequence of hypotheses that certain propositions are true or as a sequence of declarations of to which sets certain variables belong, or even a mix of the two interpretations.

There are complications to this picture. In general sets/types of the context may depend on each other, so that A_2 depends on the parameter x_1, we write this as $A_2(x_1)$, moreover A_3 depend on the two previous parameters x_1 and x_2, and finally A_n may depend on all the previous parameters, which we write as $A_n(x_1, \ldots, x_{n-1})$. The general form of the hypothetical judgement thus looks like

$$\mathcal{J}(x_1, \ldots, x_n) \quad (x_1 : A_1, x_2 : A_2(x_1), x_3 : A_3(x_1, x_2), \ldots, x_n : A_n(x_1, \ldots, x_{n-1})). \tag{2.6}$$

An alternative notation for hypothetical judgements will be the sequent notation

$$x_1 : A_1, x_2 : A_2(x_1), x_3 : A_3(x_1, x_2), \ldots, x_n : A_n(x_1, \ldots, x_{n-1}) \vdash \mathcal{J}(x_1, \ldots, x_n). \tag{2.7}$$

A note on substitutions. In (Martin-Löf 1984) an informal notion of substitution is employed. If $A(x)$ and $a(x)$ is a set respectively an element then $A(b)$ and $a(b)$ are the results of substituting b for the variable x in these entities, respectively. More generally if $A(x_1, \ldots, x_n)$ and $a(x_1, \ldots, x_n)$ is a set respectively an element, then $A(b_1, \ldots, b_n)$ and $a(b_1, \ldots, b_n)$ are the results of *simultaneously* substituting $b_1, \ldots, b_n$ for the variables $x_1, \ldots, x_n$ in these entities respectively. It is assumed that substitution renames bound variables so that no clashes occurs. There several ways to make this notation precise: use syntactic substitution when formulating rules, use a logical framework with syntactic substitution on the type level, or use explicit substitution. We will return to these refinements later.

We start with a detailed analysis of the one parameter hypothetical judgements as in (Martin-Löf 1984). We distinguish again four forms of judgements

1. $B(x)$ set $\quad (x : A)$
2. $B(x) = C(x) \quad (x : A)$

3. $b(x) : B(x) \quad (x : A)$

4. $b(x) = c(x) : B(x) \quad (x : A)$

The meaning of the judgement forms are the following.

1. The meaning of the hypothetical judgement

$$B(x) \text{ set} \quad (x : A)$$

is that for each element a of A, $B(a)$ is a set, and moreover if a and b are two equal elements of A, then $B(a) = B(b)$.

This justifies two substitution rules:

$$\frac{\vdash a : A \quad x : A \vdash B(x) \text{ set}}{\vdash B(a) \text{ set}} \qquad \frac{\vdash a = b : A \quad x : A \vdash B(x) \text{ set}}{\vdash B(a) = B(b)} .$$

2. The meaning of the hypothetical judgement

$$B(x) = C(x) \quad (x : A)$$

is that for any element a of A, $B(a)$ and $C(a)$ are equal sets.

This justifies the substitution rule:

$$\frac{\vdash a : A \quad x : A \vdash B(x) = C(x)}{\vdash B(a) = C(a)} .$$

3. The meaning of the hypothetical judgement

$$b(x) : B(x) \quad (x : A)$$

is that for any element a of A, $b(a)$ is an element of $B(a)$, and whenever a and c are equal elements of A, then $b(a)$ and $b(c)$ are equal elements of $B(a)$.

By this we justify the two substitution rules

$$\frac{\vdash a : A \quad x : A \vdash b(x) : B(x)}{\vdash b(a) : B(a)} \qquad \frac{\vdash a = c : A \quad x : A \vdash b(x) : B(x)}{\vdash b(a) = b(c) : B(a)} .$$

4. The meaning of the hypothetical judgement

$$b(x) = c(x) : B(x) \quad (x : A)$$

is that for any element a of A, $b(a)$ and $c(a)$ are equal elements of $B(a)$.

This justifies the substitution rule

$$\frac{\vdash a : A \quad x : A \vdash b(x) = c(x) : B(x)}{\vdash b(a) = c(a) : B(a)} .$$

2.3.4 General hypothetical judgements

The general hypothetical judgements forms are:

1. $B(x_1, \ldots, x_n)$ set $\qquad (x_1 : A_1, x_2 : A_2(x_1), x_3 : A_3(x_1, x_2), \ldots, x_n : A_n(x_1, \ldots, x_{n-1}))$

2. $B(x_1, \ldots, x_n) = C(x_1, \ldots, x_n)$
   $$(x_1 : A_1, x_2 : A_2(x_1), x_3 : A_3(x_1, x_2), \ldots, x_n : A_n(x_1, \ldots, x_{n-1}))$$

3. $b(x_1, \ldots, x_n) : B(x_1, \ldots, x_n)$
   $$(x_1 : A_1, x_2 : A_2(x_1), x_3 : A_3(x_1, x_2), \ldots, x_n : A_n(x_1, \ldots, x_{n-1}))$$

4. $b(x_1, \ldots, x_n) = c(x_1, \ldots, x_n) : B(x_1, \ldots, x_n)$
   $$(x_1 : A_1, x_2 : A_2(x_1), x_3 : A_3(x_1, x_2), \ldots, x_n : A_n(x_1, \ldots, x_{n-1}))$$

Notice that these are instances of (2.6).

Now we need to instantiate all the hypothesis at once, that is, to make a simultaneous substitution for all the variables $x_1, x_2, \ldots, x_n$ with some elements $a_1, \ldots, a_n$ respectively. Because of the dependencies we need the following n judgements

$$\begin{array}{l} a_1 : A_1 \\ a_2 : A_2(a_1) \\ a_3 : A_3(a_1, a_2) \\ \quad\vdots \\ a_n : A_n(a_1, \ldots, a_{n-1}) \end{array} \tag{2.8}$$

to substitute correctly into the variables. We write this also as

$$\vdash a_1 : A_1, \vdash a_2 : A_2(a_1), \vdash a_3 : A_3(a_1, a_2), \ldots, \vdash a_n : A_n(a_1, \ldots, a_{n-1})$$

1. The judgement

$$B(x_1, \ldots, x_n) \text{ set} \qquad (x_1 : A_1, x_2 : A_2(x_1), x_3 : A_3(x_1, x_2), \ldots, x_n : A_n(x_1, \ldots, x_{n-1}))$$

means that for any sequence $a_1, \ldots, a_n$ of elements such that

$$\vdash a_1 : A_1, \vdash a_2 : A_2(a_1), \vdash a_3 : A_3(a_1, a_2), \ldots, \vdash a_n : A_n(a_1, \ldots, a_{n-1})$$

we have

$$B(a_1, \ldots, a_n) \text{ set}$$

and moreover for any other sequence $c_1, \ldots, c_n$ of elements with

$$\vdash c_1 : A_1, \vdash c_2 : A_2(c_1), \vdash a_3 : A_3(c_1, c_2), \ldots, \vdash c_n : A_n(c_1, \ldots, c_{n-1})$$

and

$$\vdash a_1 = c_1 : A_1, \vdash a_2 = c_2 : A_2(a_1), \vdash a_3 = c_3 : A_3(a_1, a_2), \ldots, \vdash a_n = c_n : A_n(a_1, \ldots, a_{n-1})$$

we have

$$B(a_1, \ldots, a_n) = B(c_1, \ldots, c_n).$$

2. The judgement

$$\begin{aligned} &B(x_1, \ldots, x_n) = C(x_1, \ldots, x_n) \\ &\qquad (x_1 : A_1, x_2 : A_2(x_1), x_3 : A_3(x_1, x_2), \ldots, x_n : A_n(x_1, \ldots, x_{n-1})) \end{aligned}$$

means that for any sequence $a_1, \ldots, a_n$ of elements such that

$$\vdash a_1 : A_1, \vdash a_2 : A_2(a_1), \vdash a_3 : A_3(a_1, a_2), \ldots, \vdash a_n : A_n(a_1, \ldots, a_{n-1})$$

we have

$$B(a_1, \ldots, a_n) = C(a_1, \ldots, a_n).$$

3. The judgement

$$\begin{aligned} &b(x_1, \ldots, x_n) : B(x_1, \ldots, x_n) \\ &\qquad (x_1 : A_1, x_2 : A_2(x_1), x_3 : A_3(x_1, x_2), \ldots, x_n : A_n(x_1, \ldots, x_{n-1})) \end{aligned}$$

means that for any sequence $a_1, \ldots, a_n$ of elements such that

$$\vdash a_1 : A_1, \vdash a_2 : A_2(a_1), \vdash a_3 : A_3(a_1, a_2), \ldots, \vdash a_n : A_n(a_1, \ldots, a_{n-1})$$

we have

$$b(a_1, \ldots, a_n) : B(a_1, \ldots, a_n)$$

and moreover for any other sequence $c_1, \ldots, c_n$ of elements with

$$\vdash c_1 : A_1, \vdash c_2 : A_2(c_1), \vdash a_3 : A_3(c_1, c_2), \ldots, \vdash c_n : A_n(c_1, \ldots, c_{n-1})$$

and

$$\vdash a_1 = c_1 : A_1, \vdash a_2 = c_2 : A_2(a_1), \vdash a_3 = c_3 : A_3(a_1, a_2), \ldots, \vdash a_n = c_n : A_n(a_1, \ldots, a_{n-1})$$

we have

$$b(a_1, \ldots, a_n) = b(c_1, \ldots, c_n) : B(a_1, \ldots, a_n).$$

4. The judgement

$$\begin{aligned} &b(x_1, \ldots, x_n) = b(x_1, \ldots, x_n) : B(x_1, \ldots, x_n) \\ &\qquad (x_1 : A_1, x_2 : A_2(x_1), x_3 : A_3(x_1, x_2), \ldots, x_n : A_n(x_1, \ldots, x_{n-1})) \end{aligned}$$

means that for any sequence $a_1, \dots, a_n$ of elements such that

$$\vdash a_1 : A_1, \vdash a_2 : A_2(a_1), \vdash a_3 : A_3(a_1, a_2), \dots, \vdash a_n : A_n(a_1, \dots, a_{n-1})$$

we have

$$b(a_1, \dots, a_n) = c(a_1, \dots, a_n) : B(a_1, \dots, a_n).$$

As in the one parameter case we can justify some substitutions laws.

From 1:

$$\frac{\begin{array}{l} x_1 : A_1, x_2 : A_2(x_1), x_3 : A_3(x_1, x_2), \dots, x_n : A_n(x_1, \dots, x_{n-1}) \vdash B(x_1, \dots, x_n) \text{ set} \\ \vdash a_1 : A_1 \quad \vdash a_2 : A_2(a_1) \quad \vdash a_3 : A_3(a_1, a_2) \quad \cdots \quad \vdash a_n : A_n(a_1, \dots, a_{n-1}) \end{array}}{\vdash B(a_1, \dots, a_n) \text{ set}}$$

$$\frac{\begin{array}{l} x_1 : A_1, x_2 : A_2(x_1), x_3 : A_3(x_1, x_2), \dots, x_n : A_n(x_1, \dots, x_{n-1}) \vdash B(x_1, \dots, x_n) \text{ set} \\ \vdash a_1 = c_1 : A_1 \quad \vdash a_2 = c_2 : A_2(a_1) \quad \vdash a_3 = c_3 : A_3(a_1, a_2) \quad \cdots \quad \vdash a_n = c_n : A_n(a_1, \dots, a_{n-1}) \end{array}}{\vdash B(a_1, \dots, a_n) = B(c_1, \dots, c_n)}$$

From 2:

$$\frac{\begin{array}{l} x_1 : A_1, x_2 : A_2(x_1), x_3 : A_3(x_1, x_2), \dots, x_n : A_n(x_1, \dots, x_{n-1}) \vdash B(x_1, \dots, x_n) = C(x_1, \dots, x_n) \\ \vdash a_1 : A_1 \quad \vdash a_2 : A_2(a_1) \quad \vdash a_3 : A_3(a_1, a_2) \quad \cdots \quad \vdash a_n : A_n(a_1, \dots, a_{n-1}) \end{array}}{\vdash B(a_1, \dots, a_n) = C(a_1, \dots, a_n)}$$

From 3:

$$\frac{\begin{array}{l} x_1 : A_1, x_2 : A_2(x_1), x_3 : A_3(x_1, x_2), \dots, x_n : A_n(x_1, \dots, x_{n-1}) \vdash b(x_1, \dots, x_n) : B(x_1, \dots, x_n) \\ \vdash a_1 : A_1 \quad \vdash a_2 : A_2(a_1) \quad \vdash a_3 : A_3(a_1, a_2) \quad \cdots \quad \vdash a_n : A_n(a_1, \dots, a_{n-1}) \end{array}}{\vdash b(a_1, \dots, a_n) : B(a_1, \dots, a_n)}$$

$$\frac{\begin{array}{l} x_1 : A_1, x_2 : A_2(x_1), x_3 : A_3(x_1, x_2), \dots, x_n : A_n(x_1, \dots, x_{n-1}) \vdash b(x_1, \dots, x_n) : B(x_1, \dots, x_n) \\ \vdash a_1 = c_1 : A_1 \quad \vdash a_2 = c_2 : A_2(a_1) \quad \vdash a_3 = c_3 : A_3(a_1, a_2) \quad \cdots \quad \vdash a_n = c_n : A_n(a_1, \dots, a_{n-1}) \end{array}}{\vdash b(a_1, \dots, a_n) = b(c_1, \dots, c_n) : B(a_1, \dots, a_n)}$$

From 4:

$$\frac{\begin{array}{l} x_1 : A_1, x_2 : A_2(x_1), x_3 : A_3(x_1, x_2), \dots, x_n : A_n(x_1, \dots, x_{n-1}) \\ \qquad\qquad \vdash b(x_1, \dots, x_n) = c(x_1, \dots, x_n) : B(x_1, \dots, x_n) \\ \vdash a_1 : A_1 \quad \vdash a_2 : A_2(a_1) \quad \vdash a_3 : A_3(a_1, a_2) \quad \cdots \quad \vdash a_n : A_n(a_1, \dots, a_{n-1}) \end{array}}{\vdash b(a_1, \dots, a_n) = c(a_1, \dots, a_n) : B(a_1, \dots, a_n)}$$

The assumption rule. Then the following assumption rule is justified: For any $k = 1, \ldots, n$ we can bring out the kth variable

$$\frac{\begin{array}{l}\vdash A_1 \text{ set} \\ x_1 : A_1 \vdash A_2(x_1) \text{ set} \\ x_1 : A_1, x_2 : A_2(x_1) \vdash A_3(x_1, x_2) \text{ set} \\ \qquad \vdots \\ x_1 : A_1, \ldots, x_{n-1} : A_{n-1}(x_1, \ldots, x_{n-2}) \vdash A_n(x_1, \ldots, x_{n-1}) \text{ set}\end{array}}{x_1 : A_1, \ldots, x_n : A_n(x_1, \ldots, x_{n-1}) \vdash x_k : A_k(x_1, \ldots, x_{k-1})} \text{(as.}k\text{)} \tag{2.9}$$

Suppose that

$$\vdash a_1 : A_1, \vdash a_2 : A_2(a_1), \vdash a_3 : A_3(a_1, a_2), \ldots, \vdash a_n : A_n(a_1, \ldots, a_{n-1})$$

Thus in particular

$$a_k : A_k(a_1, \ldots, a_{k-1}).$$

Moreover if

$$\vdash a_1 = c_1 : A_1, \vdash a_2 = c_2 : A_2(a_1), \vdash a_3 = c_3 : A_3(a_1, a_2), \ldots, \vdash a_n = c_n : A_n(a_1, \ldots, a_{n-1})$$

we have in particular

$$a_k = c_k : A_k(a_1, \ldots, a_{k-1}).$$

This justifies (2.9).

2.3.5 General Substitution Rules

The above rules can be further generalized and justified by the meaning of the hypothetical judgments forms. A sequence of assumptions

$$\Gamma = x_1 : A_1, x_2 : A_2(x_1), \ldots, x_n : A_n(x_1, \ldots, x_{n-1})$$

is called a *context.* Suppose that Δ is another context

$$y_1 : D_1, y_2 : D_2(y_1), \ldots, y_m : D_m(y_1, \ldots, y_{m-1}).$$

An element of A_1 in this context is

$$\Delta \vdash a_1(y_1, \ldots, y_m) : A_1.$$

Generalizing (2.8) we get

$$\begin{array}{l}\Delta \vdash a_1(y_1, \ldots, y_m) : A_1 \\ \Delta \vdash a_2(y_1, \ldots, y_m) : A_2(a_1(y_1, \ldots, y_m)) \\ \qquad \vdots \\ \Delta \vdash a_n(y_1, \ldots, y_m) : A_n(a_1(y_1, \ldots, y_m), \ldots, a_{n-1}(y_1, \ldots, y_m))\end{array} \tag{2.10}$$

The substitution rules now take their most general form as follows. Write $\bar{y} = y_1, \ldots, y_m$ where $\Delta = y_1 : D_1, y_2 : D_2(y_1), \ldots, y_m : D_m(y_1, \ldots, y_{m-1})$.

From 1:

$$\frac{\begin{array}{l} x_1 : A_1, x_2 : A_2(x_1), \ldots, x_n : A_n(x_1, \ldots, x_{n-1}) \vdash B(x_1, \ldots, x_n) \text{ set} \\ \Delta \vdash a_1(\bar{y}) : A_1 \\ \Delta \vdash a_2(\bar{y}) : A_2(a_1(\bar{y})) \\ \qquad \vdots \\ \Delta \vdash a_n(\bar{y}) : A_n(a_1(\bar{y}), \ldots, a_{n-1}(\bar{y})) \end{array}}{\Delta \vdash B(a_1(\bar{y}), \ldots, a_n(\bar{y})) \text{ set}}$$

$$\frac{\begin{array}{l} x_1 : A_1, x_2 : A_2(x_1), \ldots, x_n : A_n(x_1, \ldots, x_{n-1}) \vdash B(x_1, \ldots, x_n) \text{ set} \\ \Delta \vdash a_1(\bar{y}) = c_1(\bar{y}) : A_1 \\ \Delta \vdash a_2(\bar{y}) = c_2(\bar{y}) : A_2(a_1(\bar{y})) \\ \qquad \vdots \\ \Delta \vdash a_n(\bar{y}) = c_n(\bar{y}) : A_n(a_1(\bar{y}), \ldots, a_{n-1}(\bar{y})) \end{array}}{\Delta \vdash B(a_1(\bar{y}), \ldots, a_n(\bar{y})) = B(c_1(\bar{y}), \ldots, c_n(\bar{y}))}$$

From 2:

$$\frac{\begin{array}{l} x_1 : A_1, x_2 : A_2(x_1), \ldots, x_n : A_n(x_1, \ldots, x_{n-1}) \vdash B(x_1, \ldots, x_n) = C(x_1, \ldots, x_n) \\ \Delta \vdash a_1(\bar{y}) : A_1 \\ \Delta \vdash a_2(\bar{y}) : A_2(a_1(\bar{y})) \\ \qquad \vdots \\ \Delta \vdash a_n(\bar{y}) : A_n(a_1(\bar{y}), \ldots, a_{n-1}(\bar{y})) \end{array}}{\Delta \vdash B(a_1(\bar{y}), \ldots, a_n(\bar{y})) = C(a_1(\bar{y}), \ldots, a_n(\bar{y}))}$$

From 3:

$$\frac{\begin{array}{l} x_1 : A_1, x_2 : A_2(x_1), \ldots, x_n : A_n(x_1, \ldots, x_{n-1}) \vdash b(x_1, \ldots, x_n) : B(x_1, \ldots, x_n) \\ \Delta \vdash a_1(\bar{y}) : A_1 \\ \Delta \vdash a_2(\bar{y}) : A_2(a_1(\bar{y})) \\ \qquad \vdots \\ \Delta \vdash a_n(\bar{y}) : A_n(a_1(\bar{y}), \ldots, a_{n-1}(\bar{y})) \end{array}}{\Delta \vdash b(a_1(\bar{y}), \ldots, a_n(\bar{y})) : B(a_1(\bar{y}), \ldots, a_n(\bar{y}))}$$

$$\frac{\begin{array}{l} x_1 : A_1, x_2 : A_2(x_1), \ldots, x_n : A_n(x_1, \ldots, x_{n-1}) \vdash b(x_1, \ldots, x_n) : B(x_1, \ldots, x_n) \\ \Delta \vdash a_1(\bar{y}) = c_1(\bar{y}) : A_1 \\ \Delta \vdash a_2(\bar{y}) = c_2(\bar{y}) : A_2(a_1(\bar{y})) \\ \qquad \vdots \\ \Delta \vdash a_n(\bar{y}) = c_n(\bar{y}) : A_n(a_1(\bar{y}), \ldots, a_{n-1}(\bar{y})) \end{array}}{\Delta \vdash b(a_1(\bar{y}), \ldots, a_n(\bar{y})) = b(c_1(\bar{y}), \ldots, c_n(\bar{y})) : B(a_1(\bar{y}), \ldots, a_n(\bar{y}))}$$

From 4:

$$\frac{\begin{array}{l} x_1 : A_1, x_2 : A_2(x_1), \ldots, x_n : A_n(x_1, \ldots, x_{n-1}) \\ \qquad\qquad\qquad\qquad \vdash b(x_1, \ldots, x_n) = c(x_1, \ldots, x_n) : B(x_1, \ldots, x_n) \\ \Delta \vdash a_1(\bar{y}) : A_1 \\ \Delta \vdash a_2(\bar{y}) : A_2(a_1(\bar{y})) \\ \qquad \vdots \\ \Delta \vdash a_n(\bar{y}) : A_n(a_1(\bar{y}), \ldots, a_{n-1}(\bar{y})) \end{array}}{\Delta \vdash b(a_1(\bar{y}), \ldots, a_n(\bar{y})) = c(a_1(\bar{y}), \ldots, a_n(\bar{y})) : B(a_1(\bar{y}), \ldots, a_n(\bar{y}))}$$

2.3.6 Weakening of judgements

The general idea of weakening a hypothetical judgement is that we may insert an extra assumption and still be able to judge the same conclusion.

Suppose that

$$x_1 : A_1, x_2 : A_2(x_1), \ldots, x_n : A_n(x_1, \ldots, x_{n-1}) \vdash \mathcal{J}(x_1, \ldots, x_n) \tag{2.11}$$

is an arbitrary hypothetical judgement. Now assuming that $B(x_1, \ldots, x_k)$ is a set that depends only on the first k variables of the context of (2.11):

$$x_1 : A_1, x_2 : A_2(x_1), \ldots, x_k : A_k(x_1, \ldots, x_{k-1}) \vdash B(x_1, \ldots, x_k) \text{ set.} \tag{2.12}$$

If y is a variable different from $x_1, \ldots, x_n$, then

$$\begin{array}{l} x_1 : A_1, x_2 : A_2(x_1), \ldots, x_k : A_n(x_1, \ldots, x_{k-1}), \\ \quad y : B(x_1, \ldots, x_k), \\ \quad x_{k+1} : A_n(x_1, \ldots, x_k), \ldots, x_n : A_n(x_1, \ldots, x_{n-1}) \vdash \mathcal{J}(x_1, \ldots, x_n) \end{array} \tag{2.13}$$

is the weakening of (2.11) by (2.12). We can justify this as a general rule: Suppose that for any sequence $a_1, \ldots, a_k, b, a_{k+1}, \ldots, a_n$ of elements such that

$$\begin{array}{l} \vdash a_1 : A_1, \vdash a_2 : A_2(a_1), \vdash a_3 : A_3(a_1, a_2), \ldots, \vdash a_k : A_k(a_1, \ldots, a_{k-1}) \\ \vdash b : B(a_1, \ldots, a_k) \\ \vdash a_{k+1} : A_n(a_1, \ldots, a_k), \ldots, \vdash a_n : A_n(a_1, \ldots, a_{n-1}) \end{array}$$

By the meaning of (2.11) we immediately obtain

$$\vdash \mathcal{J}(a_1, \ldots, a_n)$$

as required. We have justified the following *weakening rule* for any form of judgement $\mathcal{J}$ and variable y not free in Γ, Δ

$$\frac{\Gamma \vdash B(x_1, \ldots, x_k) \text{ set} \quad \Gamma, \Delta \vdash \mathcal{J}(x_1, \ldots, x_n)}{\Gamma, y : B(x_1, \ldots, x_k), \Delta \vdash \mathcal{J}(x_1, \ldots, x_n)} \text{ (weakening)}$$

Having adopted the weakening rule a simple assumption rule suffices: for any variable y not free in Γ,

$$\frac{\Gamma \vdash A(x_1, \ldots, x_n) \text{ set}}{\Gamma, y : A(x_1, \ldots, x_n) \vdash y : A(x_1, \ldots, x_n)} \text{ (as.last)} \tag{2.14}$$

Suppose that

$$\begin{array}{l} \vdash A_1 \text{ set} \\ x_1 : A_1 \vdash A_2(x_1) \text{ set} \\ x_1 : A_1, x_2 : A_2(x_1) \vdash A_3(x_1, x_2) \text{ set} \\ \quad \vdots \\ x_1 : A_1, \ldots, x_{n-1} : A_{n-1}(x_1, \ldots, x_{n-2}) \vdash A_n(x_1, \ldots, x_{n-1}) \text{ set} \end{array} \tag{2.15}$$

Then the assumption rule (2.9) may be proved from (2.14): For any $k = 1, \ldots, n$ we can bring out the kth variable

$$x_1 : A_1, \ldots, x_n : A_n(x_1, \ldots, x_{n-1}) \vdash x_k : A_k(x_1, \ldots, x_{k-1}) \tag{2.16}$$

Indeed we get

$$x_1 : A_1, \ldots, x_k : A_k(x_1, \ldots, x_{k-1}) \vdash x_k : A_k(x_1, \ldots, x_{k-1})$$

by (2.14). Then weakening this by

$$x_1 : A_1, \ldots, x_k : A_k(x_1, \ldots, x_{n-1}) \vdash A_{k+1}(x_1, \ldots, x_k) \text{ set}$$

we get

$$x_1 : A_1, \ldots, x_k : A_k(x_1, \ldots, x_{k-1}), x_{k+1} : A_{k+1}(x_1, \ldots, x_k) \vdash x_k : A_k(x_1, \ldots, x_{k-1}).$$

We can continue weakening with sets from the list (2.15), until we get (2.16).

The following assumption rule is easily derived using weakening

$$\frac{\vdash A_1 \text{ set} \quad \cdots \quad \vdash A_n \text{ set}}{x_1 : A_1, \ldots, x_n : A_n \vdash x_k : A_k} \text{ (as.}k\text{)} \tag{2.17}$$

Chapter 3

Type Constructions

In this chapter we consider the basic set (type) constructions of Martin-Löf type theory. The rules axiomatizing the sets in the theory follows a general pattern. Just as in a natural deduction system there are *introduction rules* and *elimination rules*, and the elimination rules are essentially determined by the introduction rules. In addition there are *computation rules* fixing the relation between these rule. Moreover since the syntax of theory, is extensible, there are also *formation rules*.

In summary, defining a set in type theory requires

1. *Formation rule:* names the new set to be defined, possibly in terms of earlier defined sets.
2. *Introduction rules:* shows how canonical elements of the set are built and when they are equal.
3. *Elimination rule:* shows how to define a function on the set in terms of the canonical elements, by introducing an *elimination operator*.
4. *Computation rule:* Shows how the elimination operator acts on canonical elements.

In functional programming parlance, the canonical elements are given by *constructors* and elimination operator are called *destructors*.

Martin-Löf (Nordström *et al.* 1990) distinguishes between *sets* and *types* where the former are supposed to be constructed according to rules as above, whereas in the case of types, the elimination rules are not necessarily determined by the introduction rule. An alternative term for *set* could be *inductive type*.

3.1 Π-sets — generalized cartesian products

The Π-set construction presented here is a type theory version of the generalized cartesian product of ordinary set theory mentioned above.

If we display all the possibly free variables, the Π-formation rule is this pair of rules. Suppose $\Gamma = z_1 : C_1, \ldots, z_n : C_n(z_1, \ldots, z_{n-1})$ and writing $\bar{z} = z_1, \ldots, z_n$,

$$\frac{\Gamma \vdash A(\bar{z}) \text{ set} \quad \Gamma, x : A(\bar{z}) \vdash B(\bar{z}, x) \text{ set}}{\Gamma \vdash (\Pi x : A(\bar{z}))B(\bar{z}, x) \text{ set}}$$

$$\frac{\Gamma \vdash A(\bar{z}) = C(\bar{z}) \text{ set} \quad \Gamma, x : A(\bar{z}) \vdash B(\bar{z}, x) = D(\bar{z}, x) \text{ set}}{\Gamma \vdash (\Pi x : A(\bar{z}))B(\bar{z}, x) = (\Pi x : C(\bar{z}))D(\bar{z}, x) \text{ set}}$$

However, we prefer to suppress both the common context Γ and its variables $\bar{z}$ when giving the rules. So we write the rule as follows, and similarly for all the other rules.

1. Π-formation:

$$\frac{\vdash A \text{ set} \quad x : A \vdash B(x) \text{ set}}{\vdash (\Pi x : A)B(x) \text{ set}} \qquad \frac{\vdash A = C \text{ set} \quad x : A \vdash B(x) = D(x)}{\vdash (\Pi x : A)B(x) = (\Pi x : C)D(x) \text{ set}}$$

2. Π-introduction:

$$\frac{x : A \vdash b(x) : B(x)}{\vdash (\lambda x)b(x) : (\Pi x : A)B(x)} \qquad \frac{x : A \vdash b(x) = c(x) : B(x)}{\vdash (\lambda x)b(x) = (\lambda x)c(x) : (\Pi x : A)B(x)}$$

The constructor of Π is the so-called λ-abstraction, which creates a name for $b(x)$ as a function of x. In ordinary mathematical texts one may see the notation $x \mapsto b(x)$ which in essence is $(\lambda x)b(x)$. The rule on the right above is in λ-calculus known as the *ξ-rule.*

3. Π-elimination:

$$\frac{\vdash c : (\Pi x : A)B(x) \quad \vdash a : A}{\vdash \mathrm{Ap}(c, a) : B(a)} \qquad \frac{\vdash c = d : (\Pi x : A)B(x) \quad \vdash a = b : A}{\vdash \mathrm{Ap}(c, a) = \mathrm{Ap}(d, b) : B(a)}$$

In this rule $\mathrm{Ap}(c, a)$ is the method which first calculates c to a canonical element $(\lambda x)b(x)$, and then substitutes a into $b(x)$ and continues to compute $b(a)$ to a canonical element. Then we can justify the following rule relating the elimination operator Ap to the canonical element $(\lambda x)b(x)$.

4. Π-computation:

$$\frac{x : A \vdash b(x) : B(x) \quad \vdash a : A}{\vdash \mathrm{Ap}((\lambda x)b(x), a) = b(a) : B(a)}$$

In λ-calculus this rule is better known as the *β-rule.*

Remark 3.1.1. In Martin-Löf (1984, p. 29) the η-rule

$$\frac{\vdash c : (\Pi x : A)B(x)}{c = (\lambda x)\mathrm{Ap}(c, x)}$$

is also considered. However, it does not belong to the intensional version of type theory and can be avoided in applications.

3.1.1 Function set

A noteworthy special case of the above is when the family $B(x)$ is constant C. Then defining

$$A \to C =_{\text{def}} (\Pi x : A)B(x) = (\Pi x : A)C$$

we have the following rules:

1. $\to$-formation:

$$\frac{\vdash A \text{ set} \quad \vdash C \text{ set}}{\vdash A \to C \text{ set}} \qquad \frac{\vdash A = D \text{ set} \quad \vdash C = E}{\vdash A \to C = D \to E \text{ set}}$$

2. $\to$-introduction:

$$\frac{x : A \vdash c(x) : C}{\vdash (\lambda x)c(x) : A \to C} \qquad \frac{x : A \vdash c(x) = d(x) : C}{\vdash (\lambda x)c(x) = (\lambda x)d(x) : A \to C}$$

3. $\to$-elimination:

$$\frac{\vdash c : A \to C \quad \vdash a : A}{\vdash \mathrm{Ap}(c, a) : C} \qquad \frac{\vdash c = d : A \to C \quad \vdash a = b : A}{\vdash \mathrm{Ap}(c, a) = \mathrm{Ap}(d, b) : C}$$

4. $\to$-computation:

$$\frac{x : A \vdash b(x) : C \quad \vdash a : A}{\vdash \mathrm{Ap}((\lambda x)b(x), a) = b(a) : C}$$

Example 3.1.2. Given that A, B and C are sets, find some $h(x)$ such that

$$x : A \to B \vdash h(x) : (B \to C) \to (A \to C).$$

Working from this goal up to axioms, it seems reasonable to use $(\to)$-introduction twice at the end. Thus we have the unfinished derivation:

$$\cfrac{\cfrac{\cfrac{?}{x : A \to B, y : B \to C, z : A \vdash\ ? : C}}{x : A \to B, y : B \to C \vdash (\lambda z)\ ? : A \to C}\ (\to \text{i})}{x : A \to B \vdash (\lambda y)(\lambda z)\ ? : (B \to C) \to (A \to C)}\ (\to \text{i})$$

We can use $(\to)$-elimination and hypotheses to obtain an element of C. Writing $\Gamma = x : A \to B, y : B \to C, z : A$, we have

$$\dfrac{\dfrac{\dfrac{\dfrac{?}{\Gamma \vdash y : B \longrightarrow C} \quad \dfrac{?}{\Gamma \vdash ? : B}}{x : A \to B, y : B \to C, z : A \vdash \mathrm{Ap}(y, ?) : C}\,(\to \mathrm{e})}{x : A \to B, y : B \to C \vdash (\lambda z)\mathrm{Ap}(y, ?) : A \to C}\,(\to \mathrm{i})}{x : A \to B \vdash (\lambda y)(\lambda z)\mathrm{Ap}(y, ?) : (B \to C) \to (A \to C)}\,(\to \mathrm{i})$$

An element of B may be obtained in a similar way:

$$\dfrac{\dfrac{\dfrac{\dfrac{?}{\Gamma \vdash y : B \to C} \quad \dfrac{\dfrac{?}{\Gamma \vdash x : A \to B} \quad \dfrac{?}{\Gamma \vdash z : A}}{\Gamma \vdash \mathrm{Ap}(x, z) : B}\,(\to \mathrm{e})}{x : A \to B, y : B \to C, z : A \vdash \mathrm{Ap}(y, \mathrm{Ap}(x, z)) : C}\,(\to \mathrm{e})}{x : A \to B, y : B \to C \vdash (\lambda z)\mathrm{Ap}(y, \mathrm{Ap}(x, z)) : A \to C}\,(\to \mathrm{i})}{x : A \to B \vdash (\lambda y)(\lambda z)\mathrm{Ap}(y, \mathrm{Ap}(x, z)) : (B \to C) \to (A \to C)}\,(\to \mathrm{i})$$

At the top we can now apply assumptions rules.

$$\dfrac{\dfrac{\dfrac{\dfrac{\mathcal{D}_1 \quad \mathcal{D}_2 \quad \vdash A \text{ set}}{\Gamma \vdash y : B \to C}\,(\text{as.2}) \quad \dfrac{\dfrac{\mathcal{D}_1 \quad \mathcal{D}_2 \quad \vdash A \text{ set}}{\Gamma \vdash x : A \to B}\,(\text{as.1}) \quad \dfrac{\mathcal{D}_1 \quad \mathcal{D}_2 \quad \vdash A \text{ set}}{\Gamma \vdash z : A}\,(\text{as.3})}{\Gamma \vdash \mathrm{Ap}(x, z) : B}\,(\to \mathrm{e})}{x : A \to B, y : B \to C, z : A \vdash \mathrm{Ap}(y, \mathrm{Ap}(x, z)) : C}\,(\to \mathrm{e})}{x : A \to B, y : B \to C \vdash (\lambda z)\mathrm{Ap}(y, \mathrm{Ap}(x, z)) : A \to C}\,(\to \mathrm{i})}{x : A \to B \vdash (\lambda y)(\lambda z)\mathrm{Ap}(y, \mathrm{Ap}(x, z)) : (B \to C) \to (A \to C)}\,(\to \mathrm{i})$$

where $\mathcal{D}_1$ and $\mathcal{D}_2$ are respectively

$$\dfrac{\vdash A \text{ set} \quad \vdash B \text{ set}}{\vdash A \to B \text{ set}}\,(\to \mathrm{f}) \qquad \dfrac{\vdash B \text{ set} \quad \vdash C \text{ set}}{\vdash B \to C \text{ set}}\,(\to \mathrm{f}).$$

This completes the proof.

By a further $\to$-introduction we may obtain

$$\vdash (\lambda x)(\lambda y)(\lambda z)\mathrm{Ap}(y, \mathrm{Ap}(x, z)) : (A \to B) \to ((B \to C) \to (A \to C)) \tag{3.1}$$

Example 3.1.3. Suppose that A and B are sets and that $x : A, y : B \vdash R(x, y)$ set. Find some k such that

$$\vdash k : (\Pi x : A)(\Pi y : B)R(x, y) \to (\Pi y : B)(\Pi x : A)R(x, y). \tag{3.2}$$

Let us first check that the type is well-formed. We have by Π-formation twice

$$\dfrac{\dfrac{x : A, y : B \vdash R(x, y) \text{ set}}{x : A \vdash (\Pi y : B)R(x, y) \text{ set}}\,(\Pi - \mathrm{f})}{\vdash (\Pi x : A)(\Pi y : B)R(x, y) \text{ set}}\,(\Pi - \mathrm{f}).$$

Call this derivation $\mathcal{D}_1$. How do we show that $(\Pi y : B)(\Pi x : A)R(x, y)$ is a set? We can not use the same derivation, since the parameters of $x : A, y : B \vdash R(x, y)$ set are in the wrong order. We employ the substitution rule to permute them. By the assumption rules we get

$$\dfrac{\dfrac{\vdash B \text{ set} \quad \vdash A \text{ set}}{y : B, x : A \vdash x : A} \quad \dfrac{\vdash B \text{ set} \quad \vdash A \text{ set}}{y : B, x : A \vdash y : B} \quad x : A, y : B \vdash R(x, y) \text{ set}}{y : B, x : A \vdash R(x, y) \text{ set}}\ (\text{subst}).$$

Call this derivation $\mathcal{D}_2$. From this we get as above $\vdash (\Pi y : B)(\Pi x : A)R(x, y)$ set. It follows by an application of $\rightarrow$-formation that the type of (3.2) is well-formed.

Now we try to derive (3.2) by ending with two Π-introduction rules and a $\rightarrow$-introduction rule

$$\dfrac{\dfrac{\dfrac{\dfrac{? \quad ?}{p : (\Pi x : A)(\Pi y : B)R(x, y), y : B, x : A \vdash\ ? : R(x, y)}}{p : (\Pi x : A)(\Pi y : B)R(x, y), y : B \vdash (\lambda x)? : (\Pi x : A)R(x, y)}\ (\Pi - \text{i})}{p : (\Pi x : A)(\Pi y : B)R(x, y) \vdash (\lambda y)(\lambda x)? : (\Pi y : B)(\Pi x : A)R(x, y)}\ (\Pi - \text{i})}{\vdash (\lambda p)(\lambda y)(\lambda x)? : (\Pi x : A)(\Pi y : B)R(x, y) \rightarrow (\Pi y : B)(\Pi x : A)R(x, y)}\ (\rightarrow -\text{i})$$

Let $\Gamma = p : (\Pi x : A)(\Pi y : B)R(x, y), y : B, x : A$. Applying p to x we get $\text{Ap}(p, x) : (\Pi y : B)R(x, y)$ and this to y, we get $\text{Ap}(\text{Ap}(p, x), y) : R(x, y)$. I.e by adding Π-elimination rules at the top, we can finish the proof:

$$\dfrac{\dfrac{\dfrac{\dfrac{\dfrac{\dfrac{\mathcal{D}_2 \quad \vdash B \text{ set} \quad \vdash A \text{ set}}{\Gamma \vdash p : (\Pi x : A)(\Pi y : B)R(x, y)}\ (\text{as.1}) \quad \dfrac{\mathcal{D}_2 \quad \vdash B \text{ set} \quad \vdash A \text{ set}}{\Gamma \vdash x : A}\ (\text{as.3})}{\Gamma \vdash \text{Ap}(p, x) : (\Pi y : B)R(x, y)}\ (\Pi - \text{e}) \quad \dfrac{\mathcal{D}_2 \quad \vdash B \text{ set} \quad \vdash A \text{ set}}{\Gamma \vdash y : B}\ (\text{as.2})}{p : (\Pi x : A)(\Pi y : B)R(x, y), y : B, x : A \vdash \text{Ap}(\text{Ap}(p, x), y) : R(x, y)}\ (\Pi - \text{e})}{p : (\Pi x : A)(\Pi y : B)R(x, y), y : B \vdash (\lambda x)\text{Ap}(\text{Ap}(p, x), y) : (\Pi x : A)R(x, y)}\ (\Pi - \text{i})}{p : (\Pi x : A)(\Pi y : B)R(x, y) \vdash (\lambda y)(\lambda x)\text{Ap}(\text{Ap}(p, x), y) : (\Pi y : B)(\Pi x : A)R(x, y)}\ (\Pi - \text{i})}{\vdash (\lambda p)(\lambda y)(\lambda x)\text{Ap}(\text{Ap}(p, x), y) : (\Pi x : A)(\Pi y : B)R(x, y) \rightarrow (\Pi y : B)(\Pi x : A)R(x, y)}\ (\rightarrow -\text{i}) \tag{3.3}$$

3.1.2 Propositions-as-sets interpretation

The set $(\Pi x : A)B(x)$ will be regarded as a proposition in two ways. The first is when both A and $B(x)$ are viewed as propositions, and where $B(x)$ is constant C. This will interpret the implicational proposition $A \supset B$. The second is when A is still viewed as a set, but $B(x)$ is regarded as a proposition depending on x. This will interpret the universal quantified proposition $(\forall x : A)B(x)$.

Recall that any set A has a dual existence as a proposition A. Then $a : A$ reads as a is a proof construction for A. That A is *true* means there the is some a such that

$a : A$. We can understand this as a new judgement

$$\vdash A \text{ true} \quad \text{means there there is some } a \text{ such that} \vdash a : A$$

. To do logical deduction we need a hypothetical version of this judgement

$$B_1 \text{ true}, \ldots, B_n \text{ true} \vdash A \text{ true}.$$

means that there is some $a(x_1, \ldots, x_n)$ such that

$$x_1 : B_1, \ldots, x_n : B_n \vdash a(x_1, \ldots, x_n) : A. \tag{3.4}$$

Note that we are considering arbitrary proof constructions for $B_1, \ldots, B_m$ in the antecedent, and we are required find a proof construction $a(x_1, \ldots, x_n)$ of A depending on those.

Now we can interpret the implication rules writing

$$A \supset B =_{\text{def}} A \to B.$$

(This is not to be confused with the subset notation.)

1. $(\supset)$-formation follows from $(\to)$-formation:

$$\frac{\vdash A \text{ prop} \quad \vdash C \text{ prop}}{\vdash A \supset B \text{ prop}}$$

2. $(\supset)$-introduction follows from $(\to)$-introduction

$$\frac{x : A \vdash b(x) : B}{\vdash (\lambda x)b(x) : A \supset B}$$

Suppressing proof constructions as in (3.4) this reads as the usual introduction rule in intuitionistic logic (save for the annotation "true"):

$$\frac{A \text{ true} \vdash B \text{ true}}{\vdash A \supset B \text{ true}}$$

3. $(\supset)$-elimination (or modus ponens) follows from $(\to)$-elimination:

$$\frac{\vdash c : A \supset B \quad \vdash a : A}{\vdash \mathrm{Ap}(c, a) : B}$$

Again suppressing proof constructions we get the familiar modus ponens rule of intuitionistic logic

$$\frac{\vdash A \supset B \text{ true} \quad \vdash A \text{ true}}{\vdash B \text{ true}}$$

Second application: regard A as a set and $B(x)$ as a proposition. Define

$$(\forall x : A)B(x) =_{\text{def}} (\Pi x : A)B(x)$$

Then we have from the Π-formation rule:

1. ∀-formation :
$$\frac{\vdash A \text{ set} \quad x : A \vdash B(x) \text{ prop}}{\vdash (\forall x : A)B(x) \text{ prop}}$$

2. ∀-introduction follows from the Π-introduction rule:
$$\frac{x : A \vdash b(x) : B(x)}{\vdash (\lambda x)b(x) : (\forall x : A)B(x)}$$

Suppressing proof constructions for propositions as before we have the familiar introduction rule
$$\frac{x : A \vdash B(x) \text{ true}}{\vdash (\forall x : A)B(x) \text{ true}}$$

3. ∀-elimination follows from Π-elimination:
$$\frac{\vdash c : (\forall x : A)B(x) \quad \vdash a : A}{\vdash \text{Ap}(c, a) : B(a)}$$

Suppressing proof constructions for propositions as before we have the familiar elimination rule
$$\frac{\vdash (\forall x : A)B(x) \text{ true} \quad \vdash a : A}{\vdash B(a) \text{ true}}$$

Thus the intuitionistic logical rules of $\forall$ and $\supset$ are valid. In both cases it is clearly reasonable to call $(\lambda x)b(x)$ a method, so these rules are also valid under the BHK-interpretation.

3.2 Σ-sets — general disjoint unions

The Σ-sets constructions presented here is a type-theoretic version of the disjoint union of family of sets in set theory as recalled above.

1. Σ-formation:
$$\frac{\vdash A \text{ set} \quad x : A \vdash B(x) \text{ set}}{\vdash (\Sigma x : A)B(x) \text{ set}} \qquad \frac{\vdash A = C \text{ set} \quad x : A \vdash B(x) = D(x) \text{ set}}{\vdash (\Sigma x : A)B(x) = (\Sigma x : C)D(x) \text{ set}}$$

2. Σ-introduction:
$$\frac{\vdash a : A \quad \vdash b : B(a)}{\vdash (a, b) : (\Sigma x : A)B(x)} \qquad \frac{\vdash a = c : A \quad \vdash b = d : B(a)}{\vdash (a, b) = (c, d) : (\Sigma x : A)B(x)}$$

The canonical elements will thus be of the form (a, b).

3. Σ-elimination:

$$\frac{z : (\Sigma x : A)B(x) \vdash C(z) \text{ set} \quad \vdash c : (\Sigma x : A)B(x) \quad x : A, y : B(x) \vdash d(x, y) : C((x, y))}{\vdash E(c, (x, y)d(x, y)) : C(c)}$$

and moreover

$$\frac{z : (\Sigma x : A)B(x) \vdash C(z) \text{ set} \quad \vdash c = e : (\Sigma x : A)B(x) \quad x : A, y : B(x) \vdash d(x, y) = f(x, y) : C((x, y))}{\vdash E(c, (x, y)d(x, y)) = E(e, (x, y)f(x, y)) : C(c)}$$

This is one of the more complicated rules. Here $(x, y)d(x, y)$ indicates that the variables x and y are bound in the expression.

Any element c of $(\Sigma x : A)B(x)$ can be calculated to a canonical element of the form (a, b) where $a : A$ and $b : B(a)$. We let $E(c, (x, y)d(x, y))$ be the method which performs this calculation and then substitutes a and b in to d as $d(a, b)$ and continue calculating this element to a canonical one in $C((a, b))$. This also justifies the computation rule below.

4. Σ-computation:

$$\frac{z : (\Sigma x : A)B(x) \vdash C(z) \text{ set} \quad \vdash a : A \quad \vdash b : B(a) \quad x : A, y : B(x) \vdash d(x, y) : C((x, y))}{\vdash E((a, b), (x, y)d(x, y)) = d(a, b) : C((a, b))}$$

Derived rules for projections. Let A be a set and $B(x)$ set $(x : A)$ be a family of sets. Then $C(z) = A$ is a constant family of sets over $(\Sigma x : A)B(x)$, i.e.

$$z : (\Sigma x : A)B(x) \vdash A \text{ set}$$

and by the assumption rule

$$x : A, y : B(x) \vdash x : A$$

Now letting $d(x, y) = x$, we get by Σ-elimination, for $c : (\Sigma x : A)B(x)$

$$\vdash E(c, (x, y)x) : C(c)$$

Also by the Σ-computation rule:

$$E((a, b), (x, y)x) = a.$$

Define

$$\pi_1(c) =_{\text{def}} E(c, (x, y)x).$$

In summary:

$$\frac{\vdash c : (\Sigma x : A)B(x)}{\vdash \pi_1(c) : A} \text{ (proj1)} \tag{3.5}$$

and

$$\frac{\vdash a : A \quad \vdash b : B(a)}{\vdash \pi_1((a,b)) = a : A}\ (\text{proj1} - c).$$

Now we can see that by substitution

$$z : (\Sigma x : A)B(x) \vdash B(\pi_1(z)) \text{ set}.$$

Let $C(z) =_{\text{def}} B(\pi_1(z))$. Moreover, since

$$x : A, y : B(x) \vdash B(x) = B(\pi_1((x,y)))$$

we have

$$x : A, y : B(x) \vdash y : B(\pi_1((x,y)))$$

Let $d(x,y) =_{\text{def}} y$ and we get by Σ-elimination and Σ-computation

$$\frac{\vdash c : (\Sigma x : A)B(x)}{\vdash \pi_2(c) : B(\pi_1(c))}\ (\text{proj2}) \tag{3.6}$$

and

$$\frac{\vdash a : A \quad \vdash b : B(a)}{\vdash \pi_2((a,b)) = b : A}\ (\text{proj2} - c).$$

Again an important special case is when the family $B(x)$ is constant, say $B(x) = D$. Then defining

$$A \times D =_{\text{def}} (\Sigma x : A)B(x) = (\Sigma x : A)D.$$

The rules then becomes

1. $\times$-formation:

$$\frac{\vdash A \text{ set} \quad \vdash D \text{ set}}{\vdash A \times D \text{ set}} \qquad \frac{\vdash A = A' \text{ set} \quad \vdash D = D' \text{ set}}{\vdash A \times D = A' \times D' \text{ set}}$$

2. $\times$-introduction:

$$\frac{\vdash a : A \quad \vdash b : D}{\vdash (a,b) : A \times D} \qquad \frac{\vdash a = c : A \quad \vdash b = d : D}{\vdash (a,b) = (c,d) : A \times D}$$

3. $\times$-elimination:

$$\frac{z : A \times D \vdash C(z) \text{ set} \quad \vdash c : A \times D \quad x : A, y : D \vdash d(x,y) : C((x,y))}{\vdash E(c, (x,y)d(x,y)) : C(c)}$$

and moreover

$$\frac{z : A \times D \vdash C(z) \text{ set} \quad \vdash c = e : A \times D \quad x : A, y : D \vdash d(x,y) = f(x,y) : C((x,y))}{\vdash E(c, (x,y)d(x,y)) = E(e, (x,y)f(x,y)) : C(c)}$$

4. $\times$-computation:

$$\frac{z : A \times D \vdash C(z) \text{ set} \quad \vdash a : A \quad \vdash b : D \quad x : A, y : D \vdash d(x,y) : C((x,y))}{\vdash E((a,b),(x,y)d(x,y)) = d(a,b) : C((a,b))}$$

The projection rules take the form

$$\frac{\vdash c : A \times D}{\vdash \pi_1(c) : A}\ (\text{proj1}) \qquad \frac{\vdash c = d : A \times D}{\vdash \pi_1(c) = \pi_1(d) : A}\ (\text{proj1})$$

$$\frac{\vdash c : A \times D}{\vdash \pi_2(c) : D}\ (\text{proj2}) \qquad \frac{\vdash c = d : A \times D}{\vdash \pi_2(c) = \pi_2(d) : D}\ (\text{proj2})$$

and

$$\frac{\vdash a : A \quad \vdash b : D}{\vdash \pi_2((a,b)) = b : D}\ (\text{proj2} - c) \qquad \frac{\vdash a : A \quad \vdash b : D}{\vdash \pi_1((a,b)) = a : A}\ (\text{proj1} - c).$$

3.2.1 Propositions-as-sets interpretation

The set $(\Sigma x : A)B(x)$ may be regarded as a proposition in two ways. The first is when both A and $B(x)$ are viewed as propositions, and where $B(x)$ is constant C. This will interpret the conjunctional proposition $A \wedge C$. The second is when A is still viewed as a set, but $B(x)$ is regarded as a proposition depending on x. This will interpret the existentially quantified proposition $(\exists x : A)B(x)$.

Defining $(\exists x : A)B(x) =_{\text{def}} (\Sigma x : A)B(x)$ we get

1. $\exists$-formation:

$$\frac{\vdash A \text{ set} \quad x : A \vdash B(x) \text{ prop}}{\vdash (\exists : A)B(x) \text{ prop}}$$

2. $\exists$-introduction:

$$\frac{\vdash a : A \quad \vdash b : B(a)}{\vdash (a,b) : (\exists x : A)B(x)}$$

3. $\exists$-elimination:

$$\frac{z : (\Sigma x : A)B(x) \vdash C(z) \text{ prop} \quad \vdash c : (\exists x : A)B(x) \quad x : A, y : B(x) \vdash d(x,y) : C((x,y))}{\vdash E(c,(x,y)d(x,y)) : C(c)}$$

The latter is actually a stronger elimination rule than the standard $\exists$-elimination rule in intuitionistic logic, since C may depend on x (and y).

Next define

$$A \wedge B =_{\text{def}} A \times B$$

we get

1. $\wedge$-formation:

$$\frac{\vdash A \text{ prop} \quad x : A \vdash B \text{ prop}}{\vdash A \wedge B \text{ prop}}$$

2. $\wedge$-introduction:

$$\frac{\vdash a : A \quad \vdash b : B}{\vdash (a,b) : A \wedge B}$$

Now taking the projections of $\times$ in the non-dependent form we get

3. $\wedge$-elimination:

$$\frac{\vdash z : A \wedge B}{\vdash \pi_1(z) : A} \qquad \frac{\vdash z : A \wedge B}{\vdash \pi_2(z) : B}.$$

This shows as well that the BHK-interpretation verifies the rules for $\exists$ and $\wedge$ in intuitionistic logic.

3.2.2 Type-theoretic axiom of choice

For S and T sets and a predicate $x : S, y : T \vdash R(x,y)$ the following proposition is often called the *type-theoretic axiom of choice:*

$$(\forall x : S)(\exists y : T)R(x,y) \supset (\exists f : S \to T)(\forall x : S)R(x, \mathrm{Ap}(f,x)) \tag{3.7}$$

It is true in type theory, which may come as a surprise, as it is one of the axioms considered non-constructive in ordinary set theory ZF. The proof is essentially that of Example 1.3.2. See (Martin-Löf 1984) for a proof in type theory.

Why doesn't this axiom have non-constructive consequences? The reason is that sets in type theory do in general not have quotients. Instead a set A have to be equipped with an explicit equivalence relation $=_A$ in case we want to identify elements. Compare the Bishop 1967 quote above. Now to obtain something comparable to the strength of the axiom of choice in ZF would require that the choice function f of (3.7) to respect the equivalence relations of S and T in the sense that

$$x =_S y \Longrightarrow \mathrm{Ap}(f,x) =_T \mathrm{Ap}(f,y).$$

There is nothing that guarantees this property however. We refer to Martin-Löf (2004) for a further discussion on the axiom of choice in relation to type theory.

3.3 +-sets

The binary disjoint union is treated separately from the general disjoint union.

1. +-formation:

$$\frac{\vdash A \text{ set} \quad \vdash B \text{ set}}{\vdash A + B \text{ set}} \qquad \frac{\vdash A = C \text{ set} \quad \vdash B = D \text{ set}}{\vdash A + B = C + D \text{ set}}$$

2. $+$-introduction

$$\frac{a : A}{\mathsf{inl}(a) : A + B} \qquad \frac{b : B}{\mathsf{inr}(b) : A + B}$$

$$\frac{a = c : A}{\mathsf{inl}(a) = \mathsf{inl}(c) : A + B} \qquad \frac{b = d : B}{\mathsf{inr}(b) = \mathsf{inr}(d) : A + B}$$

3. $+$-elimination

$$\frac{z : A + B \vdash C(z) \text{ set} \quad \vdash c : A + B \quad x : A \vdash d(x) : C(\mathsf{inl}(x)) \quad y : B \vdash e(y) : C(\mathsf{inr}(y))}{D(c, (x)d(x), (y)e(y)) : C(c)}$$

$$\frac{\begin{array}{l} z : A + B \vdash C(z) \text{ set} \\ \vdash c = c' : A + B \quad x : A \vdash d(x) = d'(x) : C(\mathsf{inl}(x)) \quad y : B \vdash e(y) = e'(y) : C(\mathsf{inr}(y)) \end{array}}{D(c, (x)d(x), (y)e(y)) = D(c', (x)d'(x), (y)e'(y)) : C(c)}$$

$D(c, (x)d(x), (y)e(y))$ is a method which first computes c to canonical form, which is either $\mathsf{inl}(a)$, with $a : A$ or $\mathsf{inr}(b)$ with $b : B$. In the former case it makes a substitution and continues to evaluate $d(a)$ to canonical form. In the latter case it makes the substitution $e(b)$ and continues to evaluate this element to canonical form.

Thus we can justify the computations rules:

$+$-computation

$$\frac{z : A + B \vdash C(z) \text{ set} \quad \vdash a : A \quad x : A \vdash d(x) : C(\mathsf{inl}(x)) \quad y : B \vdash e(y) : C(\mathsf{inr}(y))}{D(\mathsf{inl}(a), (x)d(x), (y)e(y)) = d(a) : C(\mathsf{inl}(a))}$$

$$\frac{z : A + B \vdash C(z) \text{ set} \quad \vdash b : B \quad x : A \vdash d(x) : C(\mathsf{inl}(x)) \quad y : B \vdash e(y) : C(\mathsf{inr}(y))}{D(\mathsf{inr}(b), (x)d(x), (y)e(y)) = e(b) : C(\mathsf{inr}(b))}$$

Example 3.3.1. Given sets A and C and that $x : A \vdash B(x)$ set, find $h(z)$ such that

$$z : (\Sigma x : A)(B(x) + C) \vdash h(z) : (\Sigma x : A)B(x) + C.$$

A standard tactic in this situation is to try is to apply Σ-elimination to z. Write $H =_{\text{def}} (\Sigma x : A)(B(x) + C)$ and $G =_{\text{def}} (\Sigma x : A)B(x) + C$.

$$\frac{\dfrac{\text{form. \& weaken.}}{z : H \vdash G \text{ set}} \quad \dfrac{\text{form. \& weaken.}}{z : H \vdash z : H} \text{(as.1)} \quad \dfrac{?}{z : H, x : A, y : B(x) + C \vdash d(x, y) : G}}{z : H \vdash E(z, (x, y)d(x, y)) : G} \Sigma e \tag{3.8}$$

Write $\Gamma = z : H, x : A, y : B(x) + C$.

$$\dfrac{\mathcal{D}\quad \dfrac{\text{form. \& weaken.}}{\Gamma \vdash y : B(x)+C}\,\text{(as)} \quad \dfrac{\dfrac{\dfrac{\text{form. \& weaken.}}{\Gamma, u : B(x)\vdash x : A}\,\text{(as)} \quad \dfrac{\text{form. \& weaken.}}{\Gamma, u : B(x)\vdash u : B(x)}\,\text{(as)}}{\Gamma, u : B(x)\vdash (x,u) : (\Sigma x : A)B(x)}\,\Sigma i}{\Gamma, u : B(x)\vdash \mathsf{inl}((x,u)) : G}\,{+i} \quad \dfrac{\dfrac{\text{form. \& weaken.}}{\Gamma, v : C\vdash v : C}\,\text{(as)}}{\Gamma, v : C\vdash \mathsf{inr}(v) : G}\,{+i}}{\Gamma\vdash\ D(y,(u)\mathsf{inl}((x,u)),(v)\mathsf{inr}(v)) : G}\,{+e}$$

Here $\mathcal{D}$ is some derivation of $\Gamma, w : B(x) + C \vdash G$ set. Now letting $d(x,y) = D(y,(u)\mathsf{inl}((x,u)),(v)\mathsf{inr}(v))$ we can complete the derivation (3.8) and obtain

$$h(z) = E(z,(x,y)D(y,(u)\mathsf{inl}((x,u)),(v)\mathsf{inr}(v)))$$

as the desired proof construction. Note

$$\begin{aligned} h((x,\mathsf{inl}(u))) &= \mathsf{inl}((x,u)) \\ h((x,\mathsf{inr}(v))) &= \mathsf{inr}(v). \end{aligned}$$

3.3.1 Propositions-as-sets interpretation

We define

$$A \vee B =_{\text{def}} A + B$$

and obtain the following interpretation of the sets as propositions.

1. $\vee$-formation:

$$\dfrac{\vdash A \text{ prop} \quad \vdash B \text{ prop}}{\vdash A \vee B \text{ prop}}$$

2. $\vee$-introduction

$$\dfrac{a : A}{\mathsf{inl}(a) : A \vee B} \qquad \dfrac{b : B}{\mathsf{inr}(b) : A \vee B}$$

3. $\vee$-elimination

$$\dfrac{z : A \vee B \vdash C(z) \text{ prop} \quad \vdash c : A \vee B \quad x : A \vdash d(x) : C(\mathsf{inl}(x)) \quad y : B \vdash e(y) : C(\mathsf{inr}(y))}{D(c,(x)d(x),(y)e(y)) : C(c)}$$

This verifies the rules for $\vee$ in intuitionistic natural deduction. In fact rule 3 above is slightly stronger than the ordinary elimination rule, since it allows that C depends on z.

We can now conclude that the BHK-interpretation verifies the $\vee$-rules.

3.4 Empty Set

The empty set N_0 is not supposed to have elements so there are only formation and elimination rules.

N_0-formation:

$$\frac{}{\vdash \mathrm{N}_0 \text{ set}} \qquad \frac{}{\vdash \mathrm{N}_0 = \mathrm{N}_0}$$

N_0-elimination:

$$\frac{z : \mathrm{N}_0 \vdash C(z) \text{ set} \quad c : \mathrm{N}_0}{\mathrm{R}_0(c) : C(c)} \qquad \frac{z : \mathrm{N}_0 \vdash C(z) \text{ set} \quad c = d : \mathrm{N}_0}{\mathrm{R}_0(c) = \mathrm{R}_0(d) : C(c)}$$

$\mathrm{R}_0(c)$ is the method which is such that if c computes to a canonical element in N_0, it computes a canonical element in $C(c)$. (Since there is no canonical elements in N_0, this promise can trivially be kept!)

3.4.1 Propositions-as-sets interpretation

Define

$$\perp =_{\text{def}} \mathrm{N}_0$$

and we get

$\perp$-formation:

$$\frac{}{\perp \text{ prop}} \qquad \frac{}{\perp = \perp \text{ prop}}$$

$\perp$-elimination:

$$\frac{z : \mathrm{N}_0 \vdash C(z) \text{ prop} \quad c : \perp}{\mathrm{R}_0(c) : C(c)} \qquad \frac{z : \mathrm{N}_0 \vdash C(z) \text{ prop} \quad c = d : \perp}{\mathrm{R}_0(c) = \mathrm{R}_0(d) : C(c)}$$

This interprets the *ex-falso quod libet* rule in intuitionistic logic.

Thus we may conclude that the BHK-interpretation verifies all the rules of intutionistic logic.

Negation is defined in terms of $\perp$ and $\supset$, i.e.

$$\neg A =_{\text{def}} A \supset \perp.$$

Exercises For any propositions A and B find proof constructions for the following proposition:

1. $(A \supset B) \supset (\neg B \supset \neg A)$.

2. $A \supset \neg\neg A$.
3. $\neg\neg\neg A \supset \neg A$.
4. $\neg(A \vee B) \supset \neg A \wedge \neg B$.
5. $\neg A \wedge \neg B \supset \neg(A \vee B)$.

Exercises Let A be a set and $x : A \vdash B(x)$ prop. Prove

1. $\neg(\exists x : A)B(x)$ true $\vdash (\forall x : A)\neg B(x)$ true
2. $(\forall x : A)\neg B(x)$ true $\vdash \neg(\exists x : A)B(x)$ true

Chapter 4

Simple inductive types

4.1 Finite Sets

For each natural number k we introduce a set N_k that is to consist of the k elements

$$0_k, 1_k, \ldots, (k-1)_k.$$

The set N_k is sometimes called the *canonical k-element set.*

N_k-formation:

$$\frac{}{\vdash \mathrm{N}_k \text{ set}} \qquad \frac{}{\vdash \mathrm{N}_k = \mathrm{N}_k.}$$

N_k-introduction:

$$\frac{}{\vdash 0_k : \mathrm{N}_k} \qquad \frac{}{\vdash 1_k : \mathrm{N}_k} \qquad \cdots \qquad \frac{}{\vdash (k-1)_k : \mathrm{N}_k}$$

$$\frac{}{\vdash 0_k = 0_k : \mathrm{N}_k} \qquad \frac{}{\vdash 1_k = 1_k : \mathrm{N}_k} \qquad \cdots \qquad \frac{}{\vdash (k-1)_k = (k-1)_k : \mathrm{N}_k}$$

N_k-elimination:

$$\frac{z : \mathrm{N}_k \vdash C(z) \text{ set} \quad \vdash c : \mathrm{N}_k \quad \vdash d_0 : C(0_k) \quad \cdots \quad \vdash d_{k-1} : C((k-1)_k)}{\vdash \mathrm{R}_k(c, d_0, \ldots, d_{k-1}) : C(c)}$$

$$\frac{z : \mathrm{N}_k \vdash C(z) \text{ set} \quad \vdash c = c' : \mathrm{N}_k \quad \vdash d_0 = d_0' : C(0_k) \quad \cdots \quad \vdash d_{k-1} = d_{k-1}' : C((k-1)_k)}{\vdash \mathrm{R}_k(c, d_0, \ldots, d_{k-1}) = R_k(c', d_0', \ldots, d_k') : C(c)}$$

Here $\mathrm{R}_k(c, d_0, \ldots, d_{k-1})$ is the method that calculates c to canonical form, that is one of the constants $0_k, \ldots, (k-1)_k$, and if the result is i_k, then it continues to calculate d_i to canonical form. This justifies:

N_k-computation: for each $i = 0, \ldots, k-1$,

$$\frac{z : \mathrm{N}_k \vdash C(z) \text{ set} \quad \vdash d_0 : C(0_k) \quad \cdots \quad \vdash d_{k-1} : C((k-1)_k)}{\vdash \mathrm{R}_k(i_k, d_0, \ldots, d_{k-1}) = d_i : C(i_k)}$$

Remark 4.1.1. N_0 is the empty set introduced earlier. N_1 is also known as the *unit set*. It can be used to interpret the propositional constant $\top$ for true. N_2 may be used to represent Boolean values for instance with $\mathrm{ff} = 0_2$ and $\mathrm{tt} = 1_2$ for "false" respectively "true". We define

$$\mathrm{Bool} =_{\mathrm{def}} \mathrm{N}_2$$

and the programming construct if-then-else:

$$\texttt{if } c \texttt{ then } d_1 \texttt{ else } d_0 \;\; =_{\mathrm{def}} \mathrm{R}_2(c, d_0, d_1). \tag{4.1}$$

The following addition to type theory is useful as this stage.

$$\frac{a : N_2}{\vdash \mathrm{Tr}(a) \text{ set}}$$

$$\frac{}{\mathrm{Tr}(0_2) = N_0} \qquad \frac{}{\mathrm{Tr}(1_2) = N_1} \tag{4.2}$$

This construction allows us to relate the truth value symbols of N_2 to propositions. It is in fact definable using R_2 once we have access to type universes.

4.2 Natural Numbers

The natural numbers are represented in the Dedekind-Peano way using a base element 0 (zero) and a successor operation S, so that

$$\underbrace{S(S(\cdots S}_{n}(0)\cdots))$$

represents the number n.

N-formation:

$$\frac{}{\vdash \mathrm{N} \text{ set}} \qquad \frac{}{\vdash \mathrm{N} = \mathrm{N}}$$

N-introduction:

$$\frac{}{\vdash 0 : \mathrm{N}} \qquad \frac{\vdash a : \mathrm{N}}{\vdash S(a) : \mathrm{N}}$$

$$\frac{}{\vdash 0 = 0 : \mathrm{N}} \qquad \frac{\vdash a = b : \mathrm{N}}{\vdash S(a) = S(b) : \mathrm{N}}$$

N-elimination:

$$\frac{z : \mathrm{N} \vdash C(z) \text{ set} \quad \vdash c : \mathrm{N} \quad \vdash d : C(0) \quad x : \mathrm{N}, y : C(x) \vdash e(x,y) : C(S(x))}{\vdash \mathrm{R}(c, d, (x,y)e(x,y)) : C(c)}$$

$$\frac{z : \mathrm{N} \vdash C(z) \text{ set} \quad \vdash c = c' : \mathrm{N} \quad \vdash d = d' : C(0) \quad x : \mathrm{N}, y : C(x) \vdash e(x,y) = e'(x,y) : C(S(x))}{\vdash \mathrm{R}(c, d, (x,y)e(x,y)) = \mathrm{R}(c', d', (x,y)e'(x,y)) : C(c)}$$

N-computation:

$$\frac{z : \mathrm{N} \vdash C(z) \text{ set} \quad \vdash d : C(0) \quad x : \mathrm{N}, y : C(x) \vdash e(x,y) : C(S(x))}{\vdash \mathrm{R}(0, d, (x,y)e(x,y)) = d : C(0)}$$

$$\frac{z : \mathrm{N} \vdash C(z) \text{ set} \quad \vdash a : \mathrm{N} \quad \vdash d : C(0) \quad x : \mathrm{N}, y : C(x) \vdash e(x,y) : C(S(x))}{\vdash \mathrm{R}(S(a), d, (x,y)e(x,y)) = e(a, \mathrm{R}(a, d, (x,y)e(x,y))) : C(S(a))}$$

Remark 4.2.1. The N-elimination rule may as well be interpreted as an induction rule:

$$\frac{z : \mathrm{N} \vdash C(z) \text{ prop} \quad \vdash c : \mathrm{N} \quad \vdash C(0) \text{ true} \quad x : \mathrm{N}, C(x) \text{ true} \vdash C(S(x)) \text{ true}}{\vdash C(c) \text{ true}}$$

Example 4.2.2. Addition of natural numbers may be defined using recursive equations

$$\begin{aligned} u + 0 &= u \\ u + S(v) &= S(u + v). \end{aligned}$$

To do this in type theory we need to find a function $f(a, b)$ so that

$$u : \mathrm{N}, v : \mathrm{N} \vdash f(u, v) : \mathrm{N}$$

and

$$u : \mathrm{N} \vdash f(u, 0) = u : \mathrm{N} \tag{4.3}$$

$$u : \mathrm{N}, a : \mathrm{N} \vdash f(u, S(a)) = S(f(u, a)) : \mathrm{N} \tag{4.4}$$

Inspecting the N-elimination and computation rule it seems that we could possibly achieve this with constant $C(z) = \mathrm{N}$. We take u as parameter and try to define

$$f(u, v) =_{\text{def}} \mathrm{R}(v, d(u, v), (x, y)e(u, v, x, y))$$

for some suitable $d(u, v)$ and $e(u, v, x, y)$ in the rule application

$$\frac{\Gamma, z : \mathrm{N} \vdash \mathrm{N} \text{ set} \quad \Gamma \vdash v : \mathrm{N} \quad \Gamma \vdash d(u, v) : \mathrm{N} \quad \Gamma, x : \mathrm{N}, y : \mathrm{N} \vdash e(u, v, x, y) : \mathrm{N}}{\Gamma \vdash \mathrm{R}(v, d(u, v), (x, y)e(u, v, x, y)) : \mathrm{N}}$$

Here $\Gamma = u : \mathrm{N}, v : \mathrm{N}$. By substituting 0 for v and using the N-computation rule, we have

$$\underbrace{\mathrm{R}(0, d(u,0), (x,y)e(u,0,x,y))}_{f(u,0)} = d(u,0) \tag{4.5}$$

and substituting $S(a)$ for v, the computation rule then gives

$$\underbrace{\mathrm{R}(S(a), d(u,S(a)), (x,y)e(u,S(a),x,y))}_{f(u,S(a))} = e(u, S(a), a, \underbrace{\mathrm{R}(a, d(u,S(a)), (x,y)e(u,S(a),x,y))}_{f(u,a)}). \tag{4.6}$$

Now (4.3) can be solved by letting

$$d(u,v) =_{\mathrm{def}} u,$$

and (4.4) can solved by setting

$$e(u,v,x,y) =_{\mathrm{def}} S(y)$$

in (4.5) and (4.6) above. We need to verify that $\mathrm{R}(v, d(u,v), (x,y)e(u,v,x,y)) = \mathrm{R}(v, u, (x,y)S(y))$ is well typed with these definitions. Indeed we have

$$\dfrac{\dfrac{\dfrac{\dfrac{}{\vdash \mathrm{N}\ \mathrm{set}}\ \mathrm{N{-}f}}{(\text{appl. of weakening})}}{\Gamma, z : \mathrm{N} \vdash \mathrm{N}\ \mathrm{set}} \quad \dfrac{\vdash \mathrm{N}\ \mathrm{set}\ \cdots}{\Gamma \vdash v : \mathrm{N}}\ (\mathrm{as.2}) \quad \dfrac{\vdash \mathrm{N}\ \mathrm{set}\ \cdots}{\Gamma \vdash u : \mathrm{N}}\ (\mathrm{as.1}) \quad \dfrac{\dfrac{\vdash \mathrm{N}\ \mathrm{set}\ \cdots}{\Gamma, x : \mathrm{N}, y : \mathrm{N} \vdash y : \mathrm{N}}\ (\mathrm{as.4})}{\Gamma, x : \mathrm{N}, y : \mathrm{N} \vdash S(y) : \mathrm{N}}\ (\mathrm{N{-}i})}{\Gamma \vdash \mathrm{R}(v, u, (x,y)S(y)) : \mathrm{N}}\ (\mathrm{N{-}e})$$

Thus

$$u + v =_{\mathrm{def}} f(u,v) =_{\mathrm{def}} \mathrm{R}(v, u, (x,y)S(y))$$

is the method that adds u and v.

Exercise Multiplication, exponential and the tower operation $(*)$ on natural numbers are defined by recursive equations

$$\begin{aligned} u \cdot 0 &= 0 \\ u \cdot S(a) &= u \cdot a + u \end{aligned}$$

$$\begin{aligned} u^0 &= S(0) \\ u^{S(a)} &= u^a \cdot u \end{aligned}$$

$$\begin{aligned} u * 0 &= S(0) \\ u * S(a) &= u^{(u*a)} \end{aligned}$$

Modify the above construction for addition and show that these operations can be defined in type theory.

Example 4.2.3. We need an equality testing function $\mathsf{eq} : \mathrm{N} \to \mathrm{N} \to \mathrm{Bool}$ satisfying the recursive equations

$$\begin{array}{lcl} \mathrm{Ap}(\mathrm{Ap}(\mathsf{eq}, 0), 0) & = & \mathrm{tt} \\ \mathrm{Ap}(\mathrm{Ap}(\mathsf{eq}, 0), S(y)) & = & \mathrm{ff} \\ \mathrm{Ap}(\mathrm{Ap}(\mathsf{eq}, S(x)), 0) & = & \mathrm{ff} \\ \mathrm{Ap}(\mathrm{Ap}(\mathsf{eq}, S(x)), S(y)) & = & \mathrm{Ap}(\mathrm{Ap}(\mathsf{eq}, x), y) \end{array} \tag{4.7}$$

To do this we first define $\mathsf{isz} : \mathrm{N} \to \mathrm{Bool}$ by letting

$$\mathsf{isz} =_{\mathrm{def}} (\lambda u)\mathrm{R}(u, \mathrm{tt}, (x, y)\mathrm{ff})$$

Then

$$\mathrm{Ap}(\mathsf{isz}, 0) = \mathrm{tt} \text{ and } \mathrm{Ap}(\mathsf{isz}, S(a)) = \mathrm{ff}$$

by β-equality and N-computation. This function decides whether a natural number is zero. It is clear than if we find eq such that $\mathrm{Ap}(\mathsf{eq}, 0) = \mathsf{isz}$ then the first two equations of (4.7) are satisfied. For any $f : \mathrm{N} \to \mathrm{Bool}$ define

$$e(f) =_{\mathrm{def}} (\lambda u)\mathrm{R}(u, \mathrm{ff}, (v, z)\mathrm{Ap}(f, v))$$

Now if eq satisfies

$$\mathrm{Ap}(\mathsf{eq}, S(x)) = e(\mathrm{Ap}(\mathsf{eq}, x)) \tag{4.8}$$

then the last two equations of (4.7) are also satisfied. This follows easily from β-equality and N-computation. Our problem has now been reduced to solving the recursion equations

$$\begin{array}{lcl} \mathrm{Ap}(\mathsf{eq}, 0) & = & \mathsf{isz} \\ \mathrm{Ap}(\mathsf{eq}, S(x)) & = & e(\mathrm{Ap}(\mathsf{eq}, x)) \end{array} \tag{4.9}$$

Let $\mathsf{eq} =_{\mathrm{def}} (\lambda n)\mathrm{R}(n, \mathsf{isz}, (x, f)e(f)))$. Then by β-equality and N-computation

$$\mathrm{Ap}(\mathsf{eq}, 0) = \mathsf{isz}$$

$$\mathrm{Ap}(\mathsf{eq}, S(x)) = \mathrm{R}(S(x), \mathsf{isz}, (x, f)e(f))) = e(\mathrm{R}(x, \mathsf{isz}, (x, f)e(f))) = e(\mathrm{Ap}(\mathsf{eq}, x))$$

If we expand all the definitions above we obtain

$$\mathsf{eq} =_{\mathrm{def}} (\lambda n)\mathrm{R}(n, (\lambda u)\mathrm{R}(u, \mathrm{tt}, (x, y)\mathrm{ff}), (x, f)(\lambda u)\mathrm{R}(u, \mathrm{ff}, (v, z)\mathrm{Ap}(f, v))))$$

The predicate defined by

$$E(m, n) =_{\mathrm{def}} \mathrm{Tr}(\mathrm{Ap}(\mathrm{Ap}(\mathsf{eq}, m), n)).$$

may now be defined as the equality relation on natural numbers, which in view of (4.2) and (4.7) satisfies

$\vdash E(0,0)$ true

$\vdash (\forall y : \mathrm{N})\neg E(0, S(y))$ true

$\vdash (\forall x : \mathrm{N})\neg E(S(x), 0)$ true

$\vdash (\forall x : \mathrm{N})(\forall y : \mathrm{N})(E(x,y) \supset E(S(x), S(y)))$ true

$\vdash (\forall x : \mathrm{N})(\forall y : \mathrm{N})(E(S(x), S(y)) \supset E(x,y))$ true

Example 4.2.4. The recursion operation R may be used to define also objects of more complicated type. Suppose we are interested to define an operation $(\cdot)^n$ that takes a function f on a set A and gives the function f^n which is the nth iterate of that function, i.e. so that

$$\mathrm{Ap}(f^n, a) = \underbrace{\mathrm{Ap}(f, \mathrm{Ap}(f, \cdots \mathrm{Ap}(f}_{n}, a) \cdots)).$$

This satisfies the recursion equations

$$\begin{aligned} \mathrm{Ap}(f^0, a) &= a, \\ \mathrm{Ap}(f^{S(n)}, a) &= \mathrm{Ap}(f, \mathrm{Ap}(f^n, a)). \end{aligned}$$

Now f^n should be an element of $A \to A$, so we let $C(z)$ be constant $A \longrightarrow A$. We wish to find d, e so that

$$f^n =_{\mathrm{def}} \mathrm{R}(n, d(n,f), (x,y)e(n,f,x,y)) : C(n)$$

solves the equation above. Let us make this our "ansatz". Now we need the second equation here

$$\mathrm{Ap}(f^0, a) =_{\mathrm{def}} \mathrm{Ap}(d(0,f), a) = a : A.$$

Putting $d(n,f) = (\lambda x)x$ will solve this equality by β-equality. By N-computation we have the first equality and we need to satisfy the second equality

$$\mathrm{Ap}(f^{S(n)}, a) = \mathrm{Ap}(e(S(n), f, n, f^n), a) = \mathrm{Ap}(f, \mathrm{Ap}(f^n, a)).$$

Letting $e(n,f,x,y) = (\lambda v)\mathrm{Ap}(f, \mathrm{Ap}(y, v))$ solves this equation by the β-equality. Now it remains to check that f^n is well typed:

$$f : A \to A, n : \mathrm{N} \vdash f^n : A \longrightarrow A.$$

Let $\Gamma = f : A \to A, n : \mathrm{N}$. It suffices to show that the following derivation tree can be completed:

$$\dfrac{\dfrac{\dfrac{\dfrac{\vdash A \text{ set} \quad \vdash A \text{ set}}{\vdash A \to A \text{ set}}\,(\to f)}{(\mathit{weakenings})}}{\Gamma, z : \mathrm{N} \vdash A \to A \text{ set}} \qquad \dfrac{\cdots}{\Gamma \vdash n : \mathrm{N}}\,\text{as.1} \quad \dfrac{?}{\Gamma \vdash d(n,f) : A \to A} \quad \dfrac{?}{\Gamma, x : \mathrm{N}, y : A \to A \vdash e(n,f,x,y) : A \to A}}{\Gamma \vdash \mathrm{R}(n, d(n,f), (x,y)e(n,f,x,y)) : A \to A}$$

The third branch can be completed as

$$\dfrac{\dfrac{\cdots}{x : A \vdash x : A}\ \text{as.3}}{\Gamma \vdash (\lambda x)x : A \to A}\ (\to i)$$

The fourth branch is completed as indicated (abbreviating $\Gamma' = \Gamma, x : \mathrm{N}, y : A \to A$)

$$\dfrac{\dfrac{\dfrac{\cdots}{\Gamma', v : A \vdash f : A \to A}\ (\text{as.2}) \quad \dfrac{\dfrac{\cdots}{\Gamma', v : A \vdash y : A \to A}\ (\text{as.4}) \quad \dfrac{\cdots}{\Gamma', v : A \vdash v : A}\ (\text{as.5})}{\Gamma', v : A \vdash \mathrm{Ap}(y, v) : A}\ (\to e)}{\Gamma', v : A \vdash \mathrm{Ap}(f, \mathrm{Ap}(y, v)) : A}\ (\to e)}{\Gamma' \vdash (\lambda v)\mathrm{Ap}(f, \mathrm{Ap}(y, v)) : A \to A}\ (\to i)$$

Exercise. It is known that the Ackermann function $a : \mathbb{N} \times \mathbb{N} \longrightarrow \mathbb{N}$, defined by

$$\begin{array}{rcl} a(0, n) & = & S(n) \\ a(S(m), 0) & = & a(m, S(0)) \\ a(S(m), S(n)) & = & a(m, a(S(m), n)), \end{array}$$

grows more quickly than any primitive recursive function. Prove that it nevertheless may be defined in type theory with the help of the recursion operator R. [Hint 1: expand the definition of $a(S(m), n)$ in the third line of the definition. Hint 2: define a function $b : \mathrm{N} \to (\mathrm{N} \to \mathrm{N})$ such that $\mathrm{Ap}(a, (m, n)) = \mathrm{Ap}(\mathrm{Ap}(b, m), n)$. The example f^n above may also be useful.]

4.3 Home Grown Example: Binary Trees

T_2-formation:

$$\dfrac{}{\vdash \mathrm{T}_2 \ \text{set}} \qquad \dfrac{}{\vdash \mathrm{T}_2 = \mathrm{T}_2}$$

T_2-introduction:

$$\dfrac{}{\vdash \mathsf{leaf} : \mathrm{T}_2} \qquad \dfrac{\vdash a : \mathrm{T}_2 \quad \vdash b : \mathrm{T}_2}{\vdash \mathsf{branch}(a, b) : \mathrm{T}_2}$$

$$\dfrac{}{\vdash \mathsf{leaf} = \mathsf{leaf} : \mathrm{T}_2} \qquad \dfrac{\vdash a = c : \mathrm{T}_2 \quad \vdash b = d : \mathrm{T}_2}{\vdash \mathsf{branch}(a, b) = \mathsf{branch}(c, d) : \mathrm{T}_2}$$

T_2-elimination:

$$\frac{\begin{array}{l} z : \mathrm{T}_2 \vdash C(z) \text{ set} \\ \vdash c : \mathrm{T}_2 \quad \vdash d : C(\mathsf{leaf}) \quad x : \mathrm{T}_2, y : C(x), u : \mathrm{T}_2, v : C(u) \vdash e(x,y,u,v) : C(\mathsf{branch}(x,u)) \end{array}}{\vdash \mathrm{TR}(c, d, (x,y,u,v)e(x,y,u,v)) : C(c)}$$

$$\frac{\begin{array}{l} z : \mathrm{T}_2 \vdash C(z) \text{ set} \\ \vdash c = c' : \mathrm{T}_2 \\ \vdash d = d' : C(\mathsf{leaf}) \\ x : \mathrm{T}_2, y : C(x), u : \mathrm{T}_2, v : C(u) \vdash e(x,y,u,v) = e'(x,y,u,v) : C(\mathsf{branch}(x,u)) \end{array}}{\vdash \mathrm{TR}(c, d, (x,y,u,v)e(x,y,u,v)) = \mathrm{TR}(c', d', (x,y,u,v)e'(x,y,u,v)) : C(c)}$$

T_2-computation:

$$\frac{\begin{array}{l} z : \mathrm{T}_2 \vdash C(z) \text{ set} \\ \vdash d : C(\mathsf{leaf}) \quad x : \mathrm{T}_2, y : C(x), u : \mathrm{T}_2, v : C(u) \vdash e(x,y,u,v) : C(\mathsf{branch}(x,u)) \end{array}}{\vdash \mathrm{TR}(\mathsf{leaf}, d, (x,y,u,v)e(x,y,u,v)) = d : C(\mathsf{leaf})}$$

$$\frac{\begin{array}{l} z : \mathrm{T}_2 \vdash C(z) \text{ set} \\ \vdash a : \mathrm{T}_2 \quad \vdash b : \mathrm{T}_2 \\ \vdash d : C(\mathsf{leaf}) \quad x : \mathrm{T}_2, y : C(x), u : \mathrm{T}_2, v : C(u) \vdash e(x,y,u,v) : C(\mathsf{branch}(x,u)) \end{array}}{\begin{array}{l} \vdash \mathrm{TR}(\mathsf{branch}(a,b), d, (x,y,u,v)e(x,y,u,v)) = \\ \qquad e(a, \mathrm{TR}(a, d, (x,y,u,v)e(x,y,u,v)), b, \mathrm{TR}(b, d, (x,y,u,v)e(x,y,u,v))) \\ \qquad\qquad : C(\mathsf{branch}(a,b)) \end{array}}$$

Exercise. Using TR define count such that $z : \mathrm{T}_2 \vdash \mathsf{count}(z) : \mathrm{N}$ and

$$\begin{array}{rcl} \vdash \mathsf{count}(\mathsf{leaf}) & = & S(0) : \mathrm{N} \\ a : \mathrm{T}_2, b : \mathrm{T}_2 \vdash \mathsf{count}(\mathsf{branch}(a,b)) & = & \mathsf{count}(\mathsf{branch}(a)) + \mathsf{count}(\mathsf{branch}(b)) : \mathrm{N}. \end{array}$$

Chapter 5

Type Theory in the Proof Assistant Coq

Proof assistants are computer programs that check and help construct formal mathematical proofs, often in an interactive way. The first to use dependent type theory was AUTOMATH (1968), but was using classical logic. Some later proof assistant based on constructive dependent type theory are Nuprl, Coq, LEGO, Alf and Agda. The proof assistant Coq (1988-) uses several variants of dependent type theory as its background theories. One theory is pCiC, *predicative calculus of constructions,* which may be regarded as an extension of Martin-Löf type theory. All the theories are expressed using the GALLINA specification language.

The basic part of pCiC builds up all its sets (types) using the (generalized cartesian) product construction and various inductive set constructions. We examine here how the type theory of previous chapters is expressed in Coq.

That $a : A$ is expressed is Coq as `a:  A`. We use this special font (teletype) for syntax in Coq. That A *is a set* is expressed in Coq by `A : Set`, i.e. that `A` is an element of `Set`. `Set` is in turn an element of `Type`. The equality judgements are hidden in Coq's proof engine and will not be visible. The set formation judgement will be treated as membership judgement.

The proofs in Coq are usually built in the fashion indicated in Chapter 3 starting from the goal and working backwards to the axioms or assumptions. This is done using commands called *tactics*. We shall below revisit some the examples of previous chapters and show how to do them in Coq.

5.1 The product construction

The generalized cartesian product $(\Pi x : A)B$ is in Coq written `(forall x:A, B)`. Application of a function to an element is written `f a` instead of $\mathrm{Ap}(f, a)$. Repeated application Ap(Ap(f, a), b) is written as `f a b` etc. The notation for function abstraction is also different: $(\lambda x)b$ is written `(fun x => b)` or if the type of x is included `(fun x:A => b)`. The function set $A \to B$ is written as `A -> B`.

Revisiting some examples. We may formulate the last result from Example 3.1.2 by submitting the following to the Coq system (CoqIde). First we declare some variables to be sets in the *section* named `Example`

```
Section Example.
Variable A B C:Set.
```

and then enter

```
   Theorem Contra:  (A -> B) ->((B -> C) -> (A -> C)).
```

This generates a goal to prove

```
   (A -> B) -> (B -> C) -> A -> C
```

under the assumptions `A:Set`, `B:Set`, `C:Set`. In CoqIde this is indicated as

```
1 subgoal
A : Set
B : Set
C : Set
______________________________________(1/1)
(A -> B) -> (B -> C) -> A -> C
```

in the goal window. Above the line is the present context. Giving the command to use an introduction rule to obtain the goal, where the new assumption is called `x`:

```
   intro x.
```

yields the new goal:

```
1 subgoal
A : Set
B : Set
C : Set
x : A -> B
______________________________________(1/1)
(B -> C) -> A -> C
```

Two further introduction rule applications in succession

```
   intro y.
   intro z.
```

gives

```
1 subgoal
A : Set
B : Set
C : Set
x : A -> B
y : B -> C
z : A
______________________________________(1/1)
C
```

Then we can achieve `C` by applying `y` to some proof of `B`

```
   apply y.
```

gives the new goal `B`

```
1 subgoal
A : Set
B : Set
C : Set
x : A -> B
y : B -> C
z : A
______________________________________(1/1)
B
```

Then applying `x` to try to obtain `B`

```
   apply x.
```

gives the new goal

```
1 subgoal
A : Set
B : Set
C : Set
```

```
x : A -> B
y : B -> C
z : A
______________________________________(1/1)
A
```

The goal can be proved by the assumption rule

```
assumption.
```

The goal window now says: `Proof completed.` We save the proof by giving the command

```
Qed.
```

To print the witness, or proof construction, we type

```
Print Contra.
```

and get in the output window

```
Contra =

fun (x : A -> B) (y : B -> C) (z : A) => y (x z)
     : (A -> B) -> (B -> C) -> A -> C
```

Reading this back in type theory it says

$$\mathrm{Contra}_{\mathrm{def}} = (\lambda x)(\lambda y)(\lambda z)\mathrm{Ap}(y, \mathrm{Ap}(x, z)) : (A \to B) \to (B \to C) \to A \to C$$

which is just the proof construction in (3.1). If we close the section by giving the command

```
End Example.
```

all the variables `A B C` are abstracted on, and we get a general proof that can applied to any sets which is shown by printing `Contra` again.

```
Print Contra.
```

and get in the output window

```
Contra =
fun (A B C : Set) (x : A -> B) (y : B -> C) (z : A) => y (x z)
     : forall A B C : Set, (A -> B) -> (B -> C) -> A -> C
```

The complete *proof script* of the above is

```
     Section Example.
     Variable A B C:Set.

     Theorem Contra:
     (A-> B) -> ((B -> C) -> (A -> C)).
     intro x.
     intro y.
     intro z.
     apply y.
     apply x.
     assumption.
     Qed.

     Print Contra.

     End Example.

     Print Contra.
```

Example 3.1.3 may be solved quickly using the script

```
Section Example2.
Variable A B:Set.
Variable R:A -> B -> Set.

Theorem Switch:
(forall x:A, (forall y: B, R x y))
->
(forall y:B, (forall x: A, R x y)).
intros p y x.
apply p.
Qed.

Print Switch.

End Example2.
```

The command `intros p y x` makes three (named) intros in one go. After this the goal window displays:

```
1 subgoal
A : Set
B : Set
R : A -> B -> Set
p : forall (x : A) (y : B), R x y
x : B
y : A
______________________________________(1/1)
R y x
```

Note the abbreviated expression for the product type after `p`. The command `apply p` now finishes the proof by applying `p` to `y` and `x` in succession. The proof printed at the end is

```
Switch =
fun (p : forall (x : A) (y : B), R x y) (x : B) (y : A) => p y x
     : (forall (x : A) (y : B), R x y) -> forall (y : B) (x : A), R x y
```

Translated back to M-L type theory this just the conclusion of (3.3).

All other sets introduced in the last two chapter are inductive types or sets in Coq terminology, as we shall see in the next section.

5.2 Inductive sets

5.2.1 Finite sets and binary disjoint union

We may define each of the sets N_k of Section 4.1 using the construction `Inductive`.

```
Inductive N_2:  Set := zero_2 :  N_2 | one_2 :  N_2.
```
(5.1)

This command defines an inductive set `N_2` having two constructors `zero_2` and `one_2`. In the output window we see

```
  N_2 is defined
  N_2_rect is defined
  N_2_ind is defined
  N_2_rec is defined
```

In addition a number of elimination rules have been generated automatically from the introduction rule (5.1). The one of interest to us can be displayed with

```
  Print N_2_rect.
```

which gives the output:

```
N_2_rect =

fun (P : N_2 -> Type) (f : P zero_2) (f0 : P one_2) (n : N_2) =>
  match n as n0 return (P n0) with
      | zero_2 => f
      | one_2 => f0
  end
     : forall P : N_2 -> Type,
           P zero_2 -> P one_2 -> forall n : N_2, P n
```

`N_2_rec` narrows down `N_2_rect` to families in `Set`.

In fact a two element set (`bool`) is already defined and used in the standard library of Coq

```
  Inductive bool : Set := true : bool | false : bool.
```

The empty set is defined by

```
   Inductive N_0: Set := .
```

Note that there are no constructors in this set.

Binary disjoint union. The +- construction is already defined in the standard library as `sum`, but we shall here define a copy of it `sum'` to be able to investigate it closer.

```
Inductive sum' (A B:Set):Set :=
  inl': A -> sum' A B | inr': B -> sum' A B.
```

Here `inl'` and `inr'` are the two constructors which forms the canonical elements in `sum' A B` from elements in `A` and `B` respectively.

To follow M-L type theory closely we can define the elimination operator

```
Definition D (A B : Set)(C: sum' A B -> Set)
(c: sum' A B)
(d: (forall x:A, C (inl' A B x)))
(e: (forall y:B, C (inr' A B y))): C c :=
  match c as c0 return C c0 with
    | inl' x => d x
    | inr' y => e y
  end.
```

The defining part agrees well with the meaning explanation of +-elimination in above in Chapter 3. It computes the main argument `c` to canonical form `c_0` which is matched against the possible canonical forms `inl' x` and `inr' y`. Depending on which it is `d x` or `e y` is returned in the set `C c_0`.

One can check that `D` defined here is essentially a notational variant of the automatically generated elimination operator `sum'_rect`.

5.2.2 Sigma-sets

The Σ-set has just one constructor, the pairing operation `Spair`:

```
Inductive Sigma (A:Set)(B:A -> Set) : Set :=
  Spair : forall a : A, B a -> Sigma A B
```

The elimination operator is then defined guided by the meaning explanation, which is again a notational variant of the automatically generated eliminator (`Sigma_rect`).

```
Definition E (A:Set)(B:A -> Set)
  (C: Sigma A B -> Set)
  (c: Sigma A B)
  (d: (forall x:A, forall y:B x,
      C (Spair A B x y))): C c :=
 match c as c0 return (C c0) with
    | Spair a b => d a b
 end.
```

Remark 5.2.1. The library version of `Sigma` is called `sigT`. The non-dependent `Sigma` is in the library under the name `prod`.

5.2.3 Natural numbers and recursively inductive sets

The natural numbers are recursively inductively defined

```
Inductive N:Set :=  zero: N | succ : N -> N.
```

Remark 5.2.2. The library version of `N` is called `nat`

The elimination operator `R` is defined using the *fixed point construction* of Coq which enables (well-founded) recursive definitions. Here a function in `(forall n:N, C n)` is defined.

```
Definition R
    (C:N -> Set)
    (d: C zero)
    (e: (forall x:N, C x -> C (succ x))): (forall n:N, C n) :=
fix F (n: N): C n :=
  match n as n0 return (C n0) with
    | zero => d
    | succ n0 => e n0 (F n0)
  end.
```

The addition function of Example 4.2.2 may now be defined

```
Definition add (m n:N) : N :=
   R (fun z=> N) m (fun x y => succ y) n.
```

We see how it works by adding the numbers 2 and 2:

```
  Eval compute in
    add (succ (succ zero)) (succ (succ zero)).
```

This is displayed in the output window

```
   = succ (succ (succ (succ zero))) : N
```

The equality testing function of Example 4.2.3 may be defined as

```
Definition eq (m : N): N -> bool :=
  R (fun z=> N -> bool)
    (fun u => R (fun z: N => bool)
                true (fun x y => false) u)
    (fun x f =>
       (fun u => R (fun z:N => bool)
                false (fun v z => f v) u))
  m.
```

Trying it out with

```
Eval compute in
    eq zero (succ zero).
```

yields

```
    = false  : bool.
```

Remark 5.2.3. A more direct definition of the equality tester may be given in Coq as follows

```
Definition eq': N -> N -> bool :=
fix F (m :N)(n: N) : bool :=
  match m as m0 with
   | zero =>    match n as n0 with
                  | zero =>    true
                  | succ n0 => false
                end
   | succ m0 => match n as n0 with
                  | zero =>    false
                  | succ n0 => F m0 n0
                end
  end.
```

The binary tree set of Section 4.3 can be rendered as an inductive set in Coq

```
Inductive T2:Set :=   leaf: T2 | branch : T2 -> T2 -> T2.
```

The automatically generated elimination operator is interesting to compare to `TR` of Section 4.3:

```
T2_rect =
fun (P : T2 -> Type) (f : P leaf)
  (f0 : forall t : T2, P t -> forall t0 : T2, P t0 -> P (branch t t0)) =>
fix F (t : T2) : P t :=
  match t as t0 return (P t0) with
  | leaf => f
  | branch t0 t1 => f0 t0 (F t0) t1 (F t1)
  end
     : forall P : T2 -> Type,
       P leaf ->
       (forall t : T2, P t -> forall t0 : T2, P t0 -> P (branch t t0)) ->
       forall t : T2, P t
```

A common feature of the inductive sets considered above is that they are all of the form

```
Inductive newsetname [ possibly some parameters] : Set :=
    constructor1 : someset_1
 |  constructor2 : someset_2
 ...
 |  constructorN : someset_N.
```

there are some positivity conditions on the types `someset`$_k$ in that possible recursive invocations of `newsetname` must occur only strictly positive. More about this in later chapters. See Reference Manual.

Chapter 6

Identity Types and Equality

6.1 Equivalence Relations

According to the proposition-as-sets principle a *binary relation* R on a set A is the same as a family of sets

$$R(x, y) \text{ set } (x : A, y : A)$$

The relation is *reflexive* if

$$(\forall x : A)R(x, x) \text{ true}$$

It is *symmetric* if

$$(\forall x, y : A)(R(x, y) \supset R(y, x)) \text{ true.}$$

We call it *transitive* if

$$(\forall x, y, z : A)(R(x, y) \supset R(y, z) \supset R(x, z)) \text{ true.}$$

As usual an *equivalence relation* is a binary relation which satisfies all these conditions. A set A together with an equivalence relation $=_A$ on A is called a *setoid* or a *Bishop*[1] *set*. If $(A, =_A)$ and $(B, =_B)$ are setoids, a function $f : A \longrightarrow B$ is called *extensional* if

$$(\forall x, y : A)(x =_A y \supset \mathrm{Ap}(f, x) =_B \mathrm{Ap}(f, y)) \text{ true.}$$

A binary relation R is *smaller* than the binary relation S on A if

$$(\forall x, y : A)(R(x, y) \supset S(x, y)) \text{ true.}$$

Theorem 6.1.1. *Suppose that R is a reflexive binary relation on A which is smaller than any reflexive binary relation S on A. Then*

(i) R is an equivalence relation.

[1]after Errett Bishop

(ii) for any set B, any equivalence relation $=_B$ on B, and any function $f : A \to B$, the function $f : (A, R) \to (B, =_B)$ is extensional.

Proof. Part (i): The following relation is easily seen to be reflexive

$$S(x, y) =_{\text{def}} R(y, x).$$

Hence by the minimality property of R

$$(\forall x, y : A)(R(x, y) \supset S(x, y)) \text{ true.}$$

i.e.

$$(\forall x, y : A)(R(x, y) \supset R(y, x)) \text{ true.}$$

so R is symmetric. Similarly noting that the relation

$$T(x, y) =_{\text{def}} (\forall z : A)(R(y, z) \supset R(x, z)).$$

is reflexive gives

$$(\forall x, y : A)(R(x, y) \supset (\forall z : A)(R(y, z) \supset R(x, z))) \text{ true.}$$

which is a logically equivalent to definition of transitivity (check!).

Part (ii): To prove extensionality we need only to note that

$$F(x, y) =_{\text{def}} (\mathrm{Ap}(f, x) =_B \mathrm{Ap}(f, y))$$

defines a reflexive relation on A. Hence by minimality

$$(\forall x, y : A)(R(x, y) \supset \mathrm{Ap}(f, x) =_B \mathrm{Ap}(f, y)) \text{ true,}$$

which proves the extensionality of f. □

6.2 Identity Types

Martin-Löf type theory has also a standard propositional equality for each set, given by the so-called *identity type construction*. This gives a smallest equivalence relation on each set A (identifying fewest elements).

I-formation:

$$\frac{\vdash A \text{ set} \quad \vdash a : A \quad \vdash b : A}{\vdash \mathrm{I}(A, a, b) \text{ set}} \qquad \frac{\vdash A \text{ set} \quad \vdash a = c : A \quad \vdash b = d : A}{\vdash \mathrm{I}(A, a, b) = \mathrm{I}(A, c, d)}$$

I-introduction:

$$\frac{\vdash a : A}{\vdash \mathrm{r}(a) : \mathrm{I}(A, a, a)} \qquad \frac{\vdash a = b : A}{\vdash \mathrm{r}(a) = \mathrm{r}(b) : \mathrm{I}(A, a, a)}$$

I-elimination:

$$\frac{\begin{array}{l} x : A, y : A, z : \mathrm{I}(A, x, y) \vdash C(x, y, z) \text{ set} \\ \vdash a : A \quad \vdash b : A \quad \vdash c : \mathrm{I}(A, a, b) \quad x : A \vdash d(x) : C(x, x, \mathrm{r}(x)) \end{array}}{\vdash \mathrm{J}(c, (x)d(x)) : C(a, b, c)}$$

$$\frac{\begin{array}{l} x : A, y : A, z : \mathrm{I}(A, x, y) \vdash C(x, y, z) \text{ set} \\ \vdash a = a' : A \quad \vdash b = b' : A \quad \vdash c = c' : \mathrm{I}(A, a, b) \quad x : A \vdash d(x) = d'(x) : C(x, x, \mathrm{r}(x)) \end{array}}{\vdash \mathrm{J}(c, (x)d(x)) = \mathrm{J}(c', (x)d'(x)) : C(a, b, c)}$$

I-computation:

$$\frac{x : A, y : A, z : \mathrm{I}(A, x, y) \vdash C(x, y, z) \text{ set} \quad \vdash a : A \quad x : A \vdash d(x) : C(x, x, \mathrm{r}(x))}{\vdash \mathrm{J}(\mathrm{r}(a), (x)d(x)) = d(a) : C(a, a, \mathrm{r}(a))}$$

We have immediately the following relation between the judgemental equality of elements and the propositional equality given by the I-types.

Proposition 6.2.1. *If* $\Gamma \vdash a = b : A$*, then* $\Gamma \vdash \mathrm{r}(a) : \mathrm{I}(A, a, b)$*.*

Proof. Suppose $a = b : A$. We have $a = a : A$, so by I-formation equality

$$\mathrm{I}(A, a, a) = \mathrm{I}(A, a, b).$$

Now since $\mathrm{r}(a) : \mathrm{I}(A, a, a)$, the set-equality rule gives $\mathrm{r}(a) : \mathrm{I}(A, a, b)$ as required. □

Remark 6.2.2. Unlike in (Martin-Löf 1984) we assume no axioms that for drawing the conclusion $\Gamma \vdash a = b : A$ from $\Gamma \vdash c : \mathrm{I}(A, a, b)$. The theory of that book is called *extensional type theory*, while we are presenting the nowadays standard *intensional type theory* as in (Nordström *et al.* 1990).

Theorem 6.2.3. *For any A, the relation* $\mathrm{I}(A, \cdot, \cdot)$ *is the smallest reflexive relation on A.*

Proof. This is immediate from the I-elimination rule. Suppose that

$$x, y : A \vdash R(x, y) \text{ set}$$

and

$$\vdash p : (\forall x : A)R(x, x).$$

We let $C(x,y,z) =_{\mathrm{def}} R(x,y)$ and $d(x) =_{\mathrm{def}} \mathrm{Ap}(p,x)$. Then applying I-elimination in the context $\Gamma = u : A, v : A, w : \mathrm{I}(A,u,v)$ we get

$$\dfrac{\dfrac{\dots}{\Gamma \vdash u : A} \quad \dfrac{\dots}{\Gamma \vdash v : A} \quad \dfrac{\dots}{\Gamma \vdash w : \mathrm{I}(A,u,v)} \quad \dfrac{\Gamma \vdash p : (\forall x : A)R(x,x) \quad \dfrac{\dots}{\Gamma \vdash x : A}}{\Gamma, x : A \vdash \mathrm{Ap}(p,x) : R(x,x))}\,\Pi e}{\Gamma \vdash \mathrm{J}(w,(x)\mathrm{Ap}(p,x)) : R(u,v)}\,\mathrm{I}e$$

Thus by applications of Π-introduction we get

$$(\forall u, v : A)(\mathrm{I}(A,u,v) \supset R(u,v)) \text{ true}$$

as required. □

Corollary 6.2.4. *For any sets A, B, any equivalence relation $=_B$ on B, and any function $f : A \to B$, the function $f : (A, \mathrm{I}(A,\cdot,\cdot)) \to (B, =_B)$ is extensional. In particular, $f : (A, \mathrm{I}(A,\cdot,\cdot)) \to (B, \mathrm{I}(B,\cdot,\cdot))$ is extensional.* □

Proposition 6.2.5. *(Peano's third axiom)*

$$(\forall x, y : \mathrm{N})(\mathrm{I}(\mathrm{N}, S(x), S(y)) \supset \mathrm{I}(\mathrm{N}, x, y)) \text{ true.}$$

Proof. Define the function

$$\mathrm{pd} =_{\mathrm{def}} (\lambda u)\mathrm{R}(u, 0, (x,y)x) : \mathrm{N} \to \mathrm{N}.$$

Note that

$$\mathrm{Ap}(\mathrm{pd}, S(a)) = \mathrm{R}(S(a), 0, (x,y)x) = a. \tag{6.1}$$

Suppose $\mathrm{I}(\mathrm{N}, S(x), S(y))$ holds. Then by the extensionality of pd provided by Corollary 6.2.4, we get

$$\mathrm{I}(\mathrm{N}, \mathrm{Ap}(\mathrm{pd}, S(x)), \mathrm{Ap}(\mathrm{pd}, S(y))) \text{ true.}$$

and hence by (6.1)

$$\mathrm{I}(\mathrm{N}, x, y) \text{ true}$$

as required. □

The I-equality on N is in fact the same as the equality given by the equality tester eq and Tr

Lemma 6.2.6. *For all $x, y : \mathrm{N}$*

$$\mathrm{I}(\mathrm{N}, x, y) \leftrightarrow \mathrm{Tr}(\mathsf{eq}(x,y)).$$

Proof. Since $\mathrm{Tr}(\mathsf{eq}(x,x))$ holds for all $x : \mathrm{N}$ the direction ($\supset$) follows from Theorem 6.2.3. It remains to prove

$$(\forall x, y : \mathrm{N})(\mathrm{Tr}(\mathsf{eq}(x,y)) \supset \mathrm{I}(\mathrm{N}, x, y)) \text{ true}$$

This is proved by induction on x and then y. For $x = 0$, it follows readily since $\mathrm{Tr}(\mathsf{eq}(0,0)) = \mathrm{N}_1$ and $\mathrm{r}(0) : \mathrm{I}(\mathrm{N}, 0, 0)$, and since $\mathrm{Tr}(\mathsf{eq}(0, S(u))) = \mathrm{N}_0 = \perp$. Suppose as inductive hypothesis

$$(\forall y : \mathrm{N})(\mathrm{Tr}(\mathsf{eq}(x,y)) \supset \mathrm{I}(\mathrm{N}, x, y)) \text{ true.} \tag{6.2}$$

We show that

$$(\forall y : \mathrm{N})(\mathrm{Tr}(\mathsf{eq}(S(x),y)) \supset \mathrm{I}(\mathrm{N}, S(x), y)) \text{ true.}$$

by induction on y. For $y = 0$, this is trivial by $\perp$-elimination since $\mathrm{Tr}(\mathsf{eq}(S(x), 0)) = \perp$. Assume $\mathrm{Tr}(\mathsf{eq}(S(x), S(u)))$. But $\mathsf{eq}(S(x), S(u)) = \mathsf{eq}(x, u)$ so $\mathrm{Tr}(\mathsf{eq}(x, u))$. Hence by (6.2) $\mathrm{I}(\mathrm{N}, x, u)$. Then using the function $(\lambda x)S(x)$ we get by Corollary 6.2.4 that $\mathrm{I}(\mathrm{N}, \mathrm{Ap}((\lambda x)S(x), x), \mathrm{Ap}((\lambda x)S(x), u))$ holds. But by β-equality this the same as that $\mathrm{I}(\mathrm{N}, S(x), S(u))$ holds, as required. □

Corollary 6.2.7. *(Peano's fourth axiom)*

$$(\forall x : \mathrm{N})\neg\mathrm{I}(\mathrm{N}, S(x), 0) \text{ true.}$$

Proof. Suppose $x : \mathrm{N}$ and $p : \mathrm{I}(\mathrm{N}, S(x), 0)$. Then by the Lemma 6.2.6, $\mathrm{Tr}(\mathsf{eq}(S(x), 0))$ is true. But by the definition of eq we have the set equality $\mathrm{Tr}(\mathsf{eq}(S(x), 0)) = \perp$. Hence $\perp$ is true. We have shown $\neg\mathrm{I}(\mathrm{N}, S(x), 0)$ is true. □

6.3 Axiom of Choice and Related Principles

Recall that the type-theoretic axiom of choice is provable in type theory.

Theorem 6.3.1. *(Type-theoretic axiom of choice.) For any sets S and T, and any relation $R(x, y)$* set $(x : S, y : T)$ *we have*

$$(\forall x : S)(\exists y : T)R(x, y) \supset (\exists f : S \to T)(\forall x : S)R(x, \mathrm{Ap}(f, x)) \text{ true} \tag{6.3}$$

Proof. See Martin-Löf (1984). □

This version of the axiom of choice is much weaker in type theory than in ordinary set theory, since there are no quotient types. Quotients may however be simulated using setoids, and accordingly the axiom of choice may be reformulated using setoids. Suppose that $(S, =_S)$ and $(T, =_T)$ are setoids and that the relation $R(x, y)$ set $(x : S, y : T)$ is *extensional* with respect to these setoids, that is

$$(\forall x, u : S)(\forall y, v : T)[R(x, y) \wedge x =_S u \wedge y =_T v \supset R(u, v)].$$

What could be called *Zermelo's axiom of choice* (AC) (Martin-Löf 2004) is then following:

(AC) If $(\forall x : S)(\exists y : T)R(x, y)$, then there is an extensional function $f : (S, =_S) \longrightarrow (T, =_T)$ such that $(\forall x : S)R(x, \mathrm{Ap}(f, x))$.

This axiom is known to be non-constructive in general — it implies PEM (Diaconescu's theorem). However special cases of (AC) are provable in type theory, for instance when $(S, =_S)$ is $(S, \mathrm{I}(S, \cdot, \cdot))$.

Theorem 6.3.2. *Let* $R(x, y)$ set $(x : S, y : T)$ *be a relation, which need not be extensional with respect to the setoids* $(S, =_S)$ *and* $(T, =_T)$*. If* $(\forall x : S)(\exists y : T)R(x, y)$*, then there is an extensional function* $f : (S, \mathrm{I}(S, \cdot, \cdot)) \to (T, =_T)$ *such that* $(\forall x : S)R(x, \mathrm{Ap}(f, x))$*.*

Proof. By Theorem 6.3.1 we get a function $f : S \to T$ such that $(\forall x : S)R(x, \mathrm{Ap}(f, x))$. Now by Corollary 6.2.4 f is in fact extensional. $\square$

Another important form of choice is the Axiom of Unique Choice, which is also provable in type theory. It relates extensional functions to total functional relations.

Theorem 6.3.3. *(Axiom of Unique Choice) Let* $R(x, y)$ set $(x : S, y : T)$ *be a relation which is extensional with respect to* $(S, =_S)$ *and* $(T, =_T)$*. If for each* $x : S$*, there exists a unique* $y : T$*, up to* $=_T$ *equality, such that* $R(x, y)$*, then there is an extensional function* $f : (S, =_S) \longrightarrow (T, =_T)$ *such that* $(\forall x : S)R(x, \mathrm{Ap}(f, x))$*.*

Proof. By Theorem 6.3.1 we have a function $f : S \to T$ such that $(\forall x : S)R(x, \mathrm{Ap}(f, x))$. We show that it is extensional. Suppose $x =_S u$. Thus $R(x, \mathrm{Ap}(f, x))$ and $R(u, \mathrm{Ap}(f, u))$. By the extensionality of R in the first argument we get $R(x, \mathrm{Ap}(f, u))$. By the uniqueness assumption $\mathrm{Ap}(f, x) =_T \mathrm{Ap}(f, u)$. Hence f is extensional. $\square$

Exercise. Prove constructively the Dependent Choice Axiom in the form: for every set A and every binary relation R on A,

$$\begin{aligned}&(\forall x : A)(\exists y : A)R(x, y) \supset \\ &\qquad (\forall x : A)(\exists f : \mathrm{N} \to A)\mathrm{I}(A, \mathrm{Ap}(f, 0), x) \wedge (\forall n : N)R(\mathrm{Ap}(f, n), \mathrm{Ap}(f, S(n))).\end{aligned}$$

6.4 Decidable identity

Identity of A is *decidable* if

$$(\forall x, y : A)(\mathrm{I}(A, x, y) \vee \neg \mathrm{I}(A, x, y)) \text{ true}$$

Theorem 6.4.1. *Identity of* N_k *is decidable, for each* k*.*

Proof. We prove this for $k = 2$ and leave the remaining cases as exercises. By N_2-elimination on x it is enough to prove the case where $x = 0_2$ and $x = 1_2$

$$(\forall y : \mathrm{N}_2)(\mathrm{I}(\mathrm{N}_2, 0_2, y) \vee \neg\mathrm{I}(\mathrm{N}_2, 0_2, y)) \text{ true} \tag{6.4}$$

$$(\forall y : \mathrm{N}_2)(\mathrm{I}(\mathrm{N}_2, 1_2, y) \vee \neg\mathrm{I}(\mathrm{N}_2, 1_2, y)) \text{ true} \tag{6.5}$$

To prove (6.4) it is by another N_2-elimination on y enough to prove

$$\mathrm{I}(\mathrm{N}_2, 0_2, 0_2) \vee \neg\mathrm{I}(\mathrm{N}_2, 0_2, 0_2)) \text{ true} \tag{6.6}$$

and

$$\mathrm{I}(\mathrm{N}_2, 0_2, 1_2) \vee \neg\mathrm{I}(\mathrm{N}_2, 0_2, 1_2) \text{ true.} \tag{6.7}$$

Now $\mathrm{r}(0_2) : \mathrm{I}(\mathrm{N}_2, 0_2, 0_2)$, so (6.6) follows by $+$-introduction. We prove (6.7) using I-elimination. Let $C(x, y, z) = \mathrm{Tr}(y) \supset \mathrm{Tr}(x)$ for $x, y : \mathrm{N}_2, z : \mathrm{I}(\mathrm{N}_2, x, y)$. Then $(\lambda u)u : C(x, x, \mathrm{r}(x))$ for any $x : \mathrm{N}_2$. Suppose $c : \mathrm{I}(\mathrm{N}_2, 0_2, 1_2)$. Then by I-elimination,

$$\mathrm{J}(c, (x)(\lambda u)u) : C(0_2, 1_2, c).$$

Thus using the set equalities

$$C(0_2, 1_2, c) = (\mathrm{Tr}(1_2) \supset \mathrm{Tr}(0_2)) = (\mathrm{N}_1 \supset \bot)$$

we get

$$\mathrm{J}(c, (x)(\lambda u)u) : \mathrm{N}_1 \supset \bot.$$

But $0_1 : \mathrm{N}_1$, so $\mathrm{Ap}(\mathrm{J}(c, (x)(\lambda u)u), 0_1) : \bot$. Hence $(\lambda c)\mathrm{Ap}(\mathrm{J}(c, (x)(\lambda u)u), 0_1) : \neg\mathrm{I}(\mathrm{N}_2, 0_2, 1_2)$. An application of $+$-introduction the yields (6.7).

This establishes (6.4). The proof of (6.5) is similar, taking instead $C(x, y, z) = \mathrm{Tr}(x) \supset \mathrm{Tr}(y)$. □

Theorem 6.4.2. *Identity of* N *is decidable.*

Proof. This follows by Theorem 6.4.1 and Lemma 6.2.6 and intuitionistic logic. □

6.4.1 Decidable predicates and characteristic functions

Let A be a set. A family of sets $P(x)$ $(x : A)$ may be regarded as a predicate on A. We say that it is a *decidable predicate* if

$$(\forall x : A)(P(x) \vee \neg P(x)) \text{ true.}$$

In this case we have

$$(\forall x : A)(\exists y : \mathrm{N}_2)(P(x) \wedge \mathrm{I}(\mathrm{N}_2, y, 1_2) \vee \neg P(x) \wedge \mathrm{I}(\mathrm{N}_2, y, 0_2)) \text{ true.}$$

By the type-theoretic axiom of choice there is $f : A \to \mathrm{N}_2$ such that

$$(\forall x : A)(P(x) \wedge \mathrm{I}(\mathrm{N}_2, \mathrm{Ap}(f, x), 1_2) \vee \neg P(x) \wedge \mathrm{I}(\mathrm{N}_2, \mathrm{Ap}(f, x), 0_2)) \text{ true}.$$

Thus it follows by intuitionistic logic

$$(\forall x : A)(P(x) \leftrightarrow \mathrm{I}(\mathrm{N}_2, \mathrm{Ap}(f, x), 1_2)).$$

We have thus obtained a characteristic function for the predicate, which is computable as well.

6.5 Peano Arithmetic and Heyting Arithmetic

In the course of the previous chapters and sections it turns out that we have verified the axioms of Peano Arithmetic with intuitionistic logic, commonly called Heyting Arithmetic. In summary we have

(1a) $a + 0 = a : \mathrm{N}$

(1b) $a + S(b) = S(a + b) : \mathrm{N}$ (where $a + b =_{\mathrm{def}} \mathrm{R}(b, a, (x, y)S(y))$.)

(2a) $a \cdot 0 = 0 : \mathrm{N}$

(2b) $a \cdot S(b) = a \cdot b + a : \mathrm{N}$ (where $a \cdot b =_{\mathrm{def}} \mathrm{R}(b, 0, (x, y)y + a)$.)

(3) $\mathrm{I}(\mathrm{N}, S(a), S(b)) \supset \mathrm{I}(\mathrm{N}, a, b)$ true

(4) $\neg\mathrm{I}(\mathrm{N}, S(a), 0)$ true

(5) For any predicate $P(x)$ on N,

$$P(0) \supset (\forall x : \mathrm{N})(P(x) \supset P(S(x))) \supset (\forall x : \mathrm{N})P(x) \text{ true}.$$

Note that for the axioms (1a,1b,2a,2b) we have the stronger judgmental equalities. They imply according to Proposition 6.2.1 the corresponding propostional equality, e.g.

(1a') $\mathrm{I}(\mathrm{N}, a + 0, a)$ true

and so on.

The coding power of Peano and Heyting arithmetic is known to be enormous with respect to finite structures. An early witness of this is Gödel's incompleteness theorem (1931) which relies on the possibility of those systems to code proofs in any axiomatizable formal system. However for practical work in proof assistants such as Coq it is usually far more convenient to explicitly define the finite structures required using inductive types.

6.5.1 Exercises

1. Prove using Peano's axioms and intuitionistic logic that

$$(\forall xy : \mathrm{N})(\mathrm{I}(\mathrm{N}, x, y) \vee \neg \mathrm{I}(\mathrm{N}, x, y)) \text{ true}$$

   How can you use the proof construction to decide whether two numbers are equal or not?

2. Write $1 = S(0)$ and $2 = S(1)$. Define multiplication by

$$m \cdot n =_{\mathrm{def}} \mathrm{R}(n, 0, (x, y)y + m).$$

   Prove

$$(\forall x : \mathrm{N})(\exists y : \mathrm{N})(\mathrm{I}(\mathrm{N}, 2 \cdot y, x) \vee \mathrm{I}(\mathrm{N}, 2 \cdot y + 1, x))$$

   What does your proof construction do?

3. We have defined $a+b =_{\mathrm{R}} (b, 0, (x, y)S(y))$ so by the N-computation rules we have the definitional equalities $x + 0 = x : \mathrm{N}\ (x : \mathrm{N})$ and $x + S(y) = S(x + y) : \mathrm{N}$ $(x, y : \mathrm{N})$. However, we do not have $0 + x = x : \mathrm{N}\ (x : \mathrm{N})$. For this we need the propositional equality given by the I-type. Prove that the following are true:

   (a) $(\forall x : \mathrm{N})\mathrm{I}(\mathrm{N}, 0 + x, x)$

   (b) $(\forall x, y : \mathrm{N})\mathrm{I}(\mathrm{N}, S(x) + y, S(x + y))$

   (c) $(\forall x, y : \mathrm{N})\mathrm{I}(\mathrm{N}, x + y, y + x)$

4. Prove that $(\forall x, y, z : \mathrm{N})\mathrm{I}(\mathrm{N}, (x + y) + z, x + (y + z))$ holds.

5. The *predecessor function*

$$\mathrm{pd} =_{\mathrm{def}} (\lambda u)\mathrm{R}(u, 0, (x, y)x) : \mathrm{N} \to \mathrm{N}.$$

   satisfies the following equations definitionally $\mathrm{Ap}(\mathrm{pd}, 0) = 0$ and $\mathrm{Ap}(\mathrm{pd}, S(a)) = a$. The *modified subtraction function* $a \dot{-} b$ (*a monus b*) is given by the term $\mathrm{R}(b, a, (x, y)\mathrm{Ap}(\mathrm{pd}, y))$. Prove that

   (a) $(\forall x : \mathrm{N})\mathrm{I}(\mathrm{N}, x \dot{-} 0, x)$

   (b) $(\forall x : \mathrm{N})\mathrm{I}(\mathrm{N}, 0 \dot{-} S(x), 0)$

   (c) $(\forall x, y : \mathrm{N})\mathrm{I}(\mathrm{N}, S(x) \dot{-} S(y), x \dot{-} y)$

6.6 Alternative elimination rule for identity types

We present an alternative elimination rule due to Christine Pauline (1992). We may view this as say that for a fixed $a : A$ the predicate

$$\mathrm{I}(A, a, x) \qquad (x : A)$$

is the smallest predicate $P(x)$ on A for which $P(a)$ holds. That is if $P(a)$ and $\mathrm{I}(A, a, b)$ holds, then $P(b)$. As in the usual elimination rule P can be generalized to depend on the proof object of $\mathrm{I}(A, a, x)$ as well.

I-elimination (variant)

$$\frac{\begin{array}{l}\vdash a : A \\ x : A, z : \mathrm{I}(A, a, x) \vdash C(x, z) \text{ set} \\ \vdash b : A \quad \vdash c : \mathrm{I}(A, a, b) \quad \vdash d : C(a, \mathrm{r}(a))\end{array}}{\mathrm{Jp}_a(c, (x)d(x)) : C(b, c)}$$

$$\frac{\begin{array}{l}\vdash a = a' : A \\ x : A, z : \mathrm{I}(A, a, x) \vdash C(x, z) \text{ set} \\ \vdash b = b' : A \quad \vdash c = c' : \mathrm{I}(A, a, b) \quad \vdash d = d' : C(a, \mathrm{r}(a))\end{array}}{\mathrm{Jp}_a(c, d) = \mathrm{Jp}_{a'}(c', d') : C(b, c)}$$

I-computation (variant):

$$\frac{\vdash a : A \quad x : A, z : \mathrm{I}(A, a, x) \vdash C(x, z) \text{ set} \quad \vdash d : C(a, \mathrm{r}(a))}{\mathrm{Jp}_a(\mathrm{r}(a), d) = d : C(b, c)}$$

Chapter 7

Order and induction

7.1

Let $\prec$ be a binary relation on a set A, i.e. $x \prec y$ set $(x, y : A)$. We say that a predicate $P(x)$ set $(x : A)$ is *progressive* with respect to $(A, \prec)$ if

$$(\forall a : A)((\forall b : A)(b \prec a \supset P(b)) \supset P(a)) \text{ true.}$$

The binary relation $(A, \prec)$ is called *well founded* if for every progressive predicate P on A, $(\forall x : A)P(x)$. Now this property states that there is an induction principle on the set $(A, \prec)$.

Proposition 7.1.1. *If $\prec$ is a well founded relation on A, then $\neg x \prec x$ for every $x : A$*

Proof. Let $P(x) =_{\text{def}} \neg x \prec x$. It suffices to show that P is progressive. Assume

$$(\forall b : A)(b \prec a \supset \neg b \prec b). \tag{7.1}$$

To prove $\neg a \prec a$ assume $a \prec a$. Letting $b = a$ in (7.1) gives $\neg a \prec a$ which is a contradiction. Hence $\neg a \prec a$ as required. □

More generally, we have

Theorem 7.1.2. *If $\prec$ is a well founded relation on A, then there is no $f : \mathrm{N} \to A$ such that for all $n : \mathrm{N}$, $\mathrm{Ap}(f, S(n)) \prec \mathrm{Ap}(f, n)$.*

Proof. We prove that for each $z : A$

$$\neg(\exists f : \mathrm{N} \to A)\mathrm{I}(A, \mathrm{Ap}(f, 0), z) \wedge (\forall n : \mathrm{N})\mathrm{Ap}(f, S(n)) \prec \mathrm{Ap}(f, n)$$

by showing that this property $P(z)$ is progressive. Assume that

$$(\forall y : A)(y \prec z \supset P(y)) \text{ true} \tag{7.2}$$

Assume for contradiction that $f : \mathrm{N} \to A$ is such that $\mathrm{I}(A, \mathrm{Ap}(f, 0), z)$ and for all $n : N$, $\mathrm{Ap}(f, S(n)) \prec \mathrm{Ap}(f, n)$. By (7.2) we have $P(\mathrm{Ap}(f, S(0)))$ since $\mathrm{Ap}(f, S(0)) \prec z$. Let $g(n) = f(S(n))$. Then clearly

$$\mathrm{I}(A, \mathrm{Ap}(g, 0), \mathrm{Ap}(f, S(0))) \wedge (\forall n : \mathrm{N})\mathrm{Ap}(g, S(n)) \prec \mathrm{Ap}(g, n),$$

which is in contradiction to $P(\mathrm{Ap}(f, S(0)))$. □

The converse to the theorem is in generally true only classically.

Remark 7.1.3. In the classical set theory ZF the Regularity Axiom may equivalently be replaced by the Set Induction Axiom saying that $\in$ is well-founded.

Let us show that the standard strict order relation on natural numbers is well founded. Define the relation $<$ on N by

$$x < y =_{\mathrm{def}} (\exists z : \mathrm{N})\mathrm{I}(\mathrm{N}, x + S(z), y).$$

Lemma 7.1.4. *For all $x, y : \mathrm{N}$,*

$$x < S(y) \supset x < y \vee \mathrm{I}(\mathrm{N}, x, y).$$

Proof. Suppose $x < S(y)$. Hence there is $z : \mathrm{N}$ with

$$\mathrm{I}(\mathrm{N}, x + S(z), S(y)).$$

Hence by (Peano 1b and 3)

$$\mathrm{I}(\mathrm{N}, x + z, y). \tag{7.3}$$

Define a predicate

$$Q(u) =_{\mathrm{def}} (\mathrm{I}(\mathrm{N}, x + u, y) \supset x < y \vee \mathrm{I}(\mathrm{N}, x, y)).$$

It is enough to show $Q(z)$. We prove this by induction on z. To see $Q(0)$, note that if $\mathrm{I}(\mathrm{N}, x + 0, y)$ holds then $\mathrm{I}(\mathrm{N}, x, y)$ is true by (Peano 1a). Hence $x < y \vee \mathrm{I}(\mathrm{N}, x, y)$ We prove $Q(S(v))$: suppose $\mathrm{I}(\mathrm{N}, x + S(v), y)$. This gives by definition $x < y$ and hence $x < y \vee \mathrm{I}(\mathrm{N}, x, y)$ also in this case. Now (Peano 5) gives $Q(z)$ and by (7.3) we get $x < y \vee \mathrm{I}(\mathrm{N}, x, y)$ as required. □

Proposition 7.1.5. $(\mathrm{N}, <)$ *is well founded.*

Proof. Suppose that $P(x)$ set $(x : \mathrm{N})$ is a progressive predicate on N. Define

$$Q(x) =_{\mathrm{def}} (\forall y : \mathrm{N})(y < x \supset P(y)).$$

It suffices to prove $(\forall x : \mathrm{N})Q(x)$, since then for any $x : \mathrm{N}$ we have $x < S(x)$, and so $P(x)$. Now we may prove $(\forall z : \mathrm{N})Q(z)$ using (Peano 5). $Q(0)$ follows by (Peano 4). The inductive step $(\forall x : \mathrm{N})(Q(x) \supset Q(S(x))$ follows by Lemma 7.1.4 and (Peano 1a and 1b). We leave the details to the reader. □

This corresponds to so-called *complete induction* on N.

Recall some standard order-theoretic concepts: A binary relation $\prec$ on A is *transitive* if $x \prec y$ and $y \prec z$ implies $x \prec z$. A transitive order $\prec$ is a *linear order,* if for all $x, y : A$,

$$x \prec y \ \vee \ \mathrm{I}(A, x, y) \ \vee \ y \prec x.$$

If the relation $\prec$ is extensional with respect to an equivalence relation $=_A$ we require only that for all $x, y : A$,

$$x \prec y \ \vee \ x =_A y \ \vee \ y \prec x$$

for $\prec$ being linear with respect to $=_A$.

A well founded linear order is called a *well-order.*

Lemma 7.1.6. $(\mathrm{N}, <)$ *is a linear order.* $\square$

Proof. We prove that $<$ is transitive. Suppose $x < y$ and $y < z$. Thus there are u, v : such that

$$\mathrm{I}(N, x + S(u), y) \text{ true} \qquad\qquad \mathrm{I}(N, y + S(v), z) \text{ true}.$$

By I-elimination this gives

$$\mathrm{I}(N, (x + S(u)) + S(v), z) \text{ true}$$

Now associativity of + (Exercise) and (Peano 1b) gives

$$\mathrm{I}(N, x + S(S(u) + v), z) \text{ true}$$

Thus $x < z$.

To prove that $<$ is linear is an exercise. $\square$

Theorem 7.1.7. $(\mathrm{N}, <)$ *is a well-order.* $\square$

Proof. This follows by Proposition 7.1.5 and Lemma 7.1.6 $\square$

7.1.1 Construction of orders

A new well-order may be built by juxtaposing two well-orders. If $(A_1, \prec_1)$ and $(A_2, \prec_2)$ are two sets with relations, then we can form the juxtaposition of these $(A_1 + A_2, \prec_{1,2})$ where we define:

$$\begin{aligned} \mathsf{inl}(a) \prec_{1,2} \mathsf{inl}(b) &\ \Leftrightarrow\ a \prec_1 b \\ \mathsf{inl}(a) \prec_{1,2} \mathsf{inr}(b) &\ \Leftrightarrow\ \top \\ \mathsf{inr}(b) \prec_{1,2} \mathsf{inl}(a) &\ \Leftrightarrow\ \bot \\ \mathsf{inr}(a) \prec_{1,2} \mathsf{inr}(b) &\ \Leftrightarrow\ a \prec_2 b \end{aligned}$$

Formally in type theory we define the type

$$\begin{aligned}(x \prec_{1,2} y) \;=_{\text{def}}\; & (\exists a, b : A_1)(\mathrm{I}(A_1 + A_2, x, \mathsf{inl}(a)) \wedge \mathrm{I}(A_1 + A_2, y, \mathsf{inl}(b)) \wedge a \prec_1 b) \\ & \vee (\exists a : A_1)(\exists b : A_2)(\mathrm{I}(A_1 + A_2, x, \mathsf{inl}(a)) \wedge \mathrm{I}(A_1 + A_2, y, \mathsf{inr}(b))) \\ & \vee (\exists a, b : A_2)(\mathrm{I}(A_1 + A_2, x, \mathsf{inr}(a)) \wedge \mathrm{I}(A_1 + A_2, y, \mathsf{inr}(b)) \wedge a \prec_2 b).\end{aligned}$$

Theorem 7.1.8. *If $(A_1, \prec_1)$ and $(A_2, \prec_2)$ are well-orders, so is $(A_1 + A_2, \prec_{1,2})$.* □

Another possibility for constructing a new well-order is by lexico-graphic combination. If $(A_1, \prec_1)$ and $(A_2, \prec_2)$ are two sets with relations, we can form $(A_1 \times A_2, \prec^{1,2})$ by

$$(a, b) \prec^{1,2} (c, d) \Leftrightarrow a \prec_1 c \vee \mathrm{I}(A_1, a, c) \wedge b \prec_2 d.$$

Theorem 7.1.9. *If $(A_1, \prec_1)$ and $(A_2, \prec_2)$ are well-orders, so is $(A_1 \times A_2, \prec^{1,2})$.* □

In general it is difficult to construct large linearly ordered sets without classical logic, so the standard notion of ordinal becomes less powerful in constructive mathematics. See however (Mines *et al.* 1988, p. 24).

For a binary relation R on A define its transitive closure R^+ by

$$\begin{aligned}R^+(x, y) =_{\text{def}} & (\exists n : \mathrm{N})(\exists f : \mathrm{N} \to A)\mathrm{I}(A, \mathrm{Ap}(f, 0), x) \wedge \mathrm{I}(A, \mathrm{Ap}(f, S(n)), y) \\ & \wedge (\forall k : \mathrm{N})(k < S(n) \supset R(\mathrm{Ap}(f, k), \mathrm{Ap}(f, S(k)))\end{aligned} \tag{7.4}$$

Theorem 7.1.10. *For any binary relation R on A, R^+ is transitive and*

$$(\forall x, y : A)(R(x, y) \supset R^+(x, y)) \text{ true.}$$

If Q is a transitive relation on A with

$$(\forall x, y : A)(R(x, y) \supset Q(x, y)) \text{ true}$$

then

$$(\forall x, y : A)(R^+(x, y) \supset Q(x, y)) \text{ true}$$

Proof. R^+ is transitive. Suppose that $R^+(x, y)$ and $R^+(y, z)$ are witnessed by n, f and m, g in (7.4) respectively. We claim that R^+ is witnessed by $n + m$ and by some h such that $\mathrm{I}(h(k), f(k))$ for $k < S(n)$ and $\mathrm{I}(h(k), g(k \dot{-} S(n))$ for $\mathrm{I}(k, S(n))$ or $k > S(n)$. Details are left to the reader.

Suppose Q is a transitive relation on A with $(\forall x, y : A)(R(x, y) \supset Q(x, y))$ true. Suppose that n, f are witnesses to $R^+(x, y)$. Thus

$$(\forall k : \mathrm{N})(k < S(n) \supset Q(\mathrm{Ap}(f, k), \mathrm{Ap}(f, S(k)))$$

where $\mathrm{I}(A, \mathrm{Ap}(f, 0), x)$ and $\mathrm{I}(A, \mathrm{Ap}(f, S(n)), y)$ holds. Using the transitivity of Q it follows by induction that $Q(x, y)$. Details are left to the reader. □

Corollary 7.1.11. *Let R be binary relation on A. Then for all $x, y : A$*

$$R^+(x,y) \iff R(x,y) \vee (\exists z : A) R^+(x,z) \wedge R(z,y).$$

Proof. The direction ($\Leftarrow$) follows since R is included in R^+ and since R^+ transitive.

To prove the direction ($\Rightarrow$) we use the universal property of R^+ established in Theorem 7.1.10. Define

$$Q(x,y) =_{\text{def}} R(x,y) \vee (\exists z : A) R^+(x,z) \wedge R(z,y).$$

Clearly R is included in Q. It suffices to show that Q is transitive, and then apply the theorem. Suppose $Q(x,y)$ and $Q(y,w)$. Thus we have the following possibilities for some z_1 and z_2:

1. $R(x,y) \wedge R(y,w)$
2. $R(x,y) \wedge R^+(y,z_2) \wedge R(z_2,w)$
3. $R^+(x,z_1) \wedge R(z_1,y) \wedge R(y,w)$
4. $R^+(x,z_1) \wedge R(z_1,y) \wedge R^+(y,z_2) \wedge R(z_2,w)$

Then using that R is included in R^+ and R^+ is transitive, we get in each of these cases some z with $R^+(x,z)$ and $R(z,w)$. Hence $Q(x,w)$. □

Theorem 7.1.12. *For any binary relation R on A, R is well-founded if and only if R^+ is well founded.*

Proof. Suppose R^+ is well-founded. If the predicate P on A is progressive with respect to R, then it is also progressive with respect to R^+. From this follows immediately that R is well-founded.

Suppose now that R is well-founded. Assume that P is progressive with respect to R^+. We want to show that

$$(\forall x : A) P(x). \tag{7.5}$$

Consider the following predicate on A

$$S(x) =_{\text{def}} (\forall y : A)(R^+(y,x) \supset P(y)) \tag{7.6}$$

We claim that

$$(\forall x : A) S(x). \tag{7.7}$$

From this follows (7.5) immediately since P is progressive with respect to R^+. We can prove the claim (7.7) by demonstrating that S is progressive with respect to R, since R is well-founded. Let us do that. Suppose that

$$(\forall z : A)(R(z,x) \supset S(z)),$$

that is

$$(\forall z : A)(R(z, x) \supset (\forall y : A)(R^+(y, z) \supset P(y))) \tag{7.8}$$

Now as P is progressive with respect to R^+ this implies

$$(\forall z : A)(R(z, x) \supset P(z)) \tag{7.9}$$

To finally prove $S(x)$, suppose that $R^+(y, x)$. By Corollary 7.1.11 we have either $R(y, x)$ in which case $P(y)$ holds by (7.9), or we have some $z : A$ with $R^+(y, z)$ and $R(z, x)$, in which case (7.8) also gives $P(y)$. Hence $S(x)$ holds, and we have showed that S is progressive with respect to R. □

If R is a binary relation on A, we define its *reflexive transitive closure by*

$$R^*(x, y) =_{\text{def}} \mathrm{I}(A, x, y) \vee R^+(x, y).$$

Exercise. Find and prove a universal property for R^* analogous to Theorem 7.1.10.

7.1.2 An application: Newman's lemma

Well-founded sets have important applications in the study of termination in computation. Let $\leadsto^1$ be a binary relation on A which is supposed denote one-step computation. The computation relation is *strongly normalizing* if the relation

$$R(x, y) =_{\text{def}} (y \leadsto^1 x)$$

is well-founded. In particular, there are no infinite computation sequences (Theorem 7.1.2):

$$x_1 \leadsto^1 x_2 \leadsto^1 x_3 \leadsto^1 \cdots .$$

Let $\leadsto^*$ be the reflexive transitive closure of $\leadsto^1$, such that $x \leadsto^* y$ states that one can reach y from x in zero or more one-step computations.

The relation $(A, \leadsto^1)$ is *confluent*, if for any $x \leadsto^* u$ and $x \leadsto^* v$, there is y with $u \leadsto^* y$ and $v \leadsto^* y$. A weaker notion which is usually easier to establish is: $(A, \leadsto^1)$ is *weakly confluent* if whenever $x \leadsto^1 u$ and $x \leadsto^1 v$, there is y with $u \leadsto^* y$ and $v \leadsto^* y$.

Theorem 7.1.13. *(Newman's Lemma). Let $\leadsto^1$ be a binary relation on A which is strongly normalizing and weakly confluent. Then $\leadsto^1$ is confluent.*

Proof. The proof goes by verifying that the predicate

$$P(x) =_{\text{def}} (\forall u, v : A)((x \leadsto^* u) \wedge (x \leadsto^* v) \supset (\exists y : A)(u \leadsto^* y \wedge v \leadsto^* y).)$$

is progressive with respect to the inverse (well-founded) relation $R(y, x) = (x \leadsto^1 y)$. This is to verify

$$(\forall x : A)((\forall y : A)(x \leadsto^1 y \supset P(y)) \supset P(x)).$$

We leave this to the reader. □

7.2 W-types

As the usual set-theoretic ways of constructing higher ordinals or large well-orders are not available in type theory, one has to turn to other means of construction. Brouwer and Kleene proposed that infinite well founded trees could play much the same roles as regular ordinals do in set theory for various inductive constructions. The W-types of type theory (Martin-Löf 1984) are such constructions.

W-formation:

$$\frac{\vdash A \text{ set} \quad x : A \vdash B(x) \text{ set}}{\vdash (\mathrm{W}x : A)B(x) \text{ set}}$$

W-introduction:

$$\frac{\vdash a : A \quad \vdash f : B(a) \to (\mathrm{W}x : A)B(x)}{\vdash \sup(a, f) : (\mathrm{W}x : A)B(x)}$$

We should think of the canonical element $\sup(a, f)$ as a node in a tree, where $f(x)$ gives all the nodes immediately above that node, as x varies over $B(a)$. In case $B(a)$ is an empty set there will be no nodes above that node, and it is considered as a leaf of the tree. If $B(a)$ is N_2, then $\sup(a, f)$ has two nodes immediately above, namely $f(0_2)$ and $f(1_2)$. In case $B(a)$ is infinite, for instance is the set N, then $\sup(a, f)$ has infinitely many nodes immediately above itself. The element $a : A$ thus indicates what set $B(a)$ can be used for branching.

W-elimination. In this rule W abbreviates $(\mathrm{W}x : A)B(x)$:

$$\frac{\begin{array}{l} z : W \vdash C(z) \text{ set} \\ \vdash c : W \qquad x : A, h : B(x) \to W, k : (\Pi y : B(x))C(\mathrm{Ap}(h, x)) \vdash d(x, h, k) : C(\sup(x, g)) \end{array}}{\vdash \mathrm{T}(c, (x, h, k)d(x, h, k)) : C(c)}$$

W-computation:

$$\frac{\begin{array}{l} z : W \vdash C(z) \text{ set} \\ \vdash a : A \\ \vdash f : B(a) \to W \\ x : A, h : B(x) \to W, k : (\Pi y : B(x))C(\mathrm{Ap}(h, x)) \vdash d(x, h, k) : C(\sup(x, h)) \end{array}}{\vdash \mathrm{T}(\sup(a, f)), (x, h, k)d(x, h, k)) = d(a, f, (\lambda y)\mathrm{T}(\mathrm{Ap}(f, y), (x, h, k)d(x, h, k))) : C(\sup(a, f))}$$

Remark 7.2.1. The W-type construction may be defined in Coq as

```
Inductive W (A:Set)(B:A -> Set):Set:=
    sup : forall a : A, (B a -> W A B) -> W A B
```

There is one constructor `sup` which builds canonical elements in a set.

Example 7.2.2. *Brouwer's second number class* (Martin-Löf 1984). Suppose that $B(x)$ set $(x : \mathrm{N}_3)$ is a family of sets such that $B(0_3) = \mathrm{N}_0$, $B(1_3) = \mathrm{N}_1$ and $B(2_3) = \mathrm{N}$. Then

$$\mathcal{O} =_{\mathrm{def}} (\mathrm{W}x : A)B(x)$$

is the type theory counterpart of Brouwer's second number class. The zero is given by the tree with no branches:

$$0_{\mathcal{O}} =_{\mathrm{def}} \sup(0_3, (\lambda x)\mathrm{R}_0(x))$$

whereas the successor is obtained attaching a one branch node under the given tree $\alpha : \mathcal{O}$

$$S_{\mathcal{O}}(\alpha) =_{\mathrm{def}} \sup(1_3, (\lambda x)\mathrm{R}_1(x, \alpha))$$

Suppose that $f : \mathrm{N} \to \mathcal{O}$ is an infinite sequence of trees, then its *formal limit* is constructed as

$$\lim f =_{\mathrm{def}} \sup(2_3, f).$$

For every $n : N$ we may define a corresponding number n^* in $\mathcal{O}$ by recursion

$$\begin{array}{rcl} 0^* & = & 0_{\mathcal{O}} \\ (S(n))^* & = & S_{\mathcal{O}}(n^*) \end{array}$$

(Put $n^* = \mathrm{R}(n, 0_{\mathcal{O}}, (x, y)S_{\mathcal{O}}(y)) : \mathcal{O}$.) This sequence has a formal limit which we may call

$$\omega =_{\mathrm{def}} \lim\,(\lambda n)n^*.$$

Define the immediate subtree relation $\prec$ by

$$(\alpha \prec \sup(a, f)) =_{\mathrm{def}} (\exists x : B(a))\mathrm{I}(\mathcal{O}, \alpha, \mathrm{Ap}(f, x)). \tag{7.10}$$

It is easily seen that

$$(\forall n : N)\, n^* \prec \omega$$

holds. Moreover for any $f : N \longrightarrow \mathcal{O}$ we have

$$(\forall n : N)\mathrm{Ap}(f, n) \prec \sup(2_3, f)$$

so $\mathcal{O}$ is uncountable in a strong sense.

The immediate subtree relation $\prec$ on $\mathcal{O}$ is well-founded. In fact, we have more generally for the subtree relation on a general W-set

$$(\alpha \prec \sup(a, f)) =_{\mathrm{def}} (\exists x : B(a))\mathrm{I}((Wx : A)B(x), \alpha, \mathrm{Ap}(f, x)) \tag{7.11}$$

the following:

Theorem 7.2.3. *For any set A and any family of sets $B(x)$ $(x : A)$ the immediate subtree relation $\prec$ on* $(\mathrm{W}x : A)B(x)$ *is well founded.*

Proof. Write $W = (\mathrm{W}x : A)B(x)$. Suppose that $P(z)$ $(z : W)$ is progressive with respect to $\prec$. We show that

$$(\forall z : W)P(z) \text{ true}$$

by W-elimination on z. Assume $a : A$ and $f : B(a) \to W$ and that

$$(\forall t : B(a))P(\mathrm{Ap}(f, t)) \tag{7.12}$$

holds. According to the progressiveness of P, it suffices to show

$$(\forall y : W)(y \prec \sup(a, f) \supset P(y)).$$

Suppose $y : W$ and $(\exists x : B(a))\mathrm{I}(W, y, \mathrm{Ap}(f, x))$. Let $x : B(a)$ with $\mathrm{I}(W, y, \mathrm{Ap}(f, x))$ true. By (7.12) we get $P(\mathrm{Ap}(f, x))$. Now I-elimination gives the desired $P(y)$. □

Chapter 8

Universes

8.1 Type universes

A type universe is basically a collection of types which is closed under certain (already existing) type constructions. We may make this precise by saying that it is family of sets $T(x)$ set $(x : U)$ over a set U. This is a new kind of type construction which does not follow the usual pattern of introduction and elimination rules. The elimination rules are here weaker and do not guarantee that the universe is minimal with respect to the introduction rules (as for instance the natural numbers are).

U-formation:

$$\frac{}{\vdash U \text{ set}} \qquad \frac{}{\vdash U = U}$$

T-formation:

$$\frac{a : U}{T(a) \text{ set}} \qquad \frac{a = b : U}{T(a) = T(b)}$$

For every type construction that we have considered so far we introduce coding in U of that type construction. For instance we have

$$\frac{\vdash A \text{ set} \quad \vdash B \text{ set}}{\vdash A + B \text{ set}}$$

which is reflected in U as

$$\frac{\vdash a : U \quad \vdash b : U}{\vdash a \hat{+} b : U} \qquad \frac{\vdash a : U \quad \vdash b : U}{\vdash T(a \hat{+} b) = T(a) + T(b)}$$

For basic types N and N_k we have

$$\frac{}{\hat{\mathrm{N}} : U} \qquad \frac{}{T(\hat{\mathrm{N}}) = \mathrm{N}}$$

and

$$\frac{}{\hat{\mathrm{N}}_k : U} \qquad \frac{}{T(\hat{\mathrm{N}}_k) = \mathrm{N}_k.}$$

For dependent constructions the reflection is a bit more involved. We wish to reflect

$$\frac{\vdash A \text{ set} \quad x : A \vdash B(x) \text{ set}}{\vdash (\Pi x : A)B(x) \text{ set}}.$$

We introduce

$$\frac{\vdash a : U \quad x : T(\hat{a}) \vdash b(x) : U}{\vdash (\hat{\Pi} x : a)b(x) : U} \qquad \frac{\vdash a : U \quad x : T(\hat{a}) \vdash b(x) : U}{\vdash T((\hat{\Pi} x : a)b(x)) = (\Pi x : T(a))T(b(x))}$$

The rules for Σ and W are identical replacing Π by these signs respectively. The identity formation rule

$$\frac{\vdash A \text{ set} \quad \vdash b : A \quad \vdash c : A}{\vdash \mathrm{I}(A, b, c) \text{ set}}$$

is reflected by

$$\frac{\vdash a : U \quad \vdash b : T(a) \quad \vdash c : T(a)}{\vdash \hat{\mathrm{I}}(a, b, c) : U} \qquad \frac{\vdash a : U \quad \vdash b : T(a) \quad \vdash c : T(a)}{\vdash T(\hat{\mathrm{I}}(a, b, c)) = \mathrm{I}(T(a), b, c)}$$

We may consider yet another universe U', T' reflecting the type constructions so far. Thus we introduce

U'-formation:

$$\frac{}{U' \text{ set}} \qquad \frac{}{U' = U'}$$

T'-formation:

$$\frac{a : U'}{T'(a) \text{ set}} \qquad \frac{a = b : U'}{T'(a) = T'(b)}$$

Then two new constructors $\hat{U}$ and $\hat{T}$ with introduction rules

$$\frac{}{\hat{U} : U'} \qquad \frac{}{T'(\hat{U}) = U}$$

and

$$\frac{a : U}{\hat{T}(a) : U'} \qquad \frac{a = b : U}{\hat{T}(a) = \hat{T}(a) : U'}$$

$$\frac{a : U}{T'(\hat{T}(a)) = T(a)}$$

For each type construction $+, \Pi, \Sigma, \ldots$ we introduces codes $\hat{+}', \hat{\Pi}', \hat{\Sigma}', \ldots$ analogously as we did for U, T above. It is indeed possible to continue this process of building new universes ad infinitum (Martin-Löf 1984) and also to internalize it (Palmgren 1998), using so called *super universes.*

8.2 Applications of universes

The dependent type $\mathrm{Tr}(x)$ set $(x : \mathrm{N}_2)$ of (4.2) may now be defined by N_2-elimination. For $x : \mathrm{N}_2$, we have $\mathrm{R}_2(x, \hat{\mathrm{N}}_0, \hat{\mathrm{N}}_1) : U$ so letting

$$\mathrm{Tr}(x) =_{\mathrm{def}} T(\mathrm{R}_2(x, \hat{\mathrm{N}}_0, \hat{\mathrm{N}}_1)),$$

we get

$$\mathrm{Tr}(0_2) = T(\hat{\mathrm{N}}_0) = \mathrm{N}_0,$$

$$\mathrm{Tr}(1_2) = T(\hat{\mathrm{N}}_1) = \mathrm{N}_1$$

as required.

Transfinite sets: Let $F(n) = T(R(n, \hat{\mathrm{N}}, (x,y)y \to \hat{\mathrm{N}}))$. Then

$$F(0) = \mathrm{N} \qquad F(S(n)) = F(n) \to \mathrm{N},$$

and $(\Sigma n : \mathrm{N})F(n)$ is an example of a transfinite type which does not exists in standard Martin-Löf type theory without a universe.

Aczel's set-theoretic universe: Aczel's model of constructive set theory CZF (Aczel 1978) uses the following set of trees as the universe of iterative sets;

$$V =_{\mathrm{def}} (\mathrm{W}x : U)T(x)$$

Equality of sets $\alpha =_V \beta$ is $T(\mathrm{Ap}(e_V, (\alpha, \beta)))$ where $e_V : V \times V \to U$ is defined by V-elimination in such a way that the bisimulation property is satisfied:

$$\begin{aligned}&\sup(a,f) =_V \sup(b,g) \Leftrightarrow\\ &\qquad (\forall x : T(a))(\exists y : T(b))\mathrm{Ap}(f,x) =_V \mathrm{Ap}(g,y) \wedge\\ &\qquad (\forall y : T(b))(\exists x : T(a))\mathrm{Ap}(f,x) =_V \mathrm{Ap}(g,y)\end{aligned}$$

Then the membership relation is defined by as the immediate subtree relation (modulo $=_V$):

$$(\alpha \in_V \sup(a,f)) =_{\mathrm{def}} (\exists x : T(a))\, \alpha =_V \mathrm{Ap}(f,x).$$

Restricted power sets: For a set X, let $\mathcal{P}(X) = X \to U$. This is essentially Russell's propositional functions on X, where we think of U as a set of truth values. For any $Q : \mathcal{P}(X)$ and $x : X$ we may define a membership relation

$$x \mathrel{\dot{\in}} Q =_{\mathrm{def}} T(\mathrm{Ap}(Q,x)).$$

The natural inclusion relation for two $P, Q : \mathcal{P}(X)$ is then defined by

$$(P \subseteq Q) =_{\mathrm{def}} (\forall x : X)(x \mathrel{\dot{\in}} P \supset x \mathrel{\dot{\in}} Q)$$

In classical set theory the power set is a complete boolean algebra with respect to inclusion. We have an analogous result for the restricted power set just introduced. A lattice $(L, \wedge, \vee, \top, \bot, \leq)$ is a *Heyting algebra* if there is a binary operation $(\to)$ on L such that for all $a, b, c : L$

$$a \wedge b \leq c \iff a \leq (b \to c). \tag{8.1}$$

Theorem 8.2.1. *For a set X, $(\mathcal{P}(X), \subseteq)$ is a Heyting algebra which is complete with respect to suprema and infima indexed by sets of the form $T(a)$ where $a : U$.*

Proof. We define operations $\wedge, \vee, \to$. For $P, Q \in \mathcal{P}(X)$,

$$\begin{aligned} P \wedge Q &=_{\text{def}} (\lambda x)(\text{Ap}(P, x) \mathbin{\hat{\times}} \text{Ap}(Q, x)) \\ P \vee Q &=_{\text{def}} (\lambda x)(\text{Ap}(P, x) \mathbin{\hat{+}} \text{Ap}(Q, x)) \\ P \to Q &=_{\text{def}} (\lambda x)(\text{Ap}(P, x) \mathbin{\hat{\to}} \text{Ap}(Q, x)) \end{aligned}$$

To verify the condition (8.1): Let $P, Q, R \in \mathcal{P}(X)$. We need to check

$$P \wedge Q \subseteq R \iff P \subseteq (Q \to R). \tag{8.2}$$

Now

$$\begin{aligned} P \wedge Q \subseteq R &\iff (\forall x : X)(x \mathrel{\dot{\in}} (P \wedge Q) \supset x \mathrel{\dot{\in}} R) \\ &\iff (\forall x : X)(T(\text{Ap}(P \wedge Q, x)) \supset T(\text{Ap}(R, x))) \\ &\iff (\forall x : X)(T(\text{Ap}(P, x)) \wedge T(\text{Ap}(Q, x)) \supset T(\text{Ap}(R, x))) \\ &\iff (\forall x : X)(T(\text{Ap}(P, x)) \supset T(\text{Ap}(Q, x)) \supset T(\text{Ap}(R, x))) \\ &\iff (\forall x : X)(T(\text{Ap}(P, x)) \supset T(\text{Ap}((Q \to R), x)) \\ &\iff P \subseteq (Q \to R) \end{aligned}$$

Let $P_i : \mathcal{P}(X)$ for $i : T(a)$. The supremum and infimum may be defined by

$$\bigvee_{i:T(a)} P_i =_{\text{def}} (\lambda x)(\hat{\Sigma} i : a)\text{Ap}(P_i, x)$$

$$\bigwedge_{i:T(a)} P_i =_{\text{def}} (\lambda x)(\hat{\Pi} i : a)\text{Ap}(P_i, x)$$

The verification of their corresponding properties is left to the reader:

$$\bigvee_{i:T(a)} P_i \subseteq Q \iff (\forall i : T(a))(P_i \subseteq Q)$$

$$Q \subseteq \bigwedge_{i:T(a)} P_i \iff (\forall i : T(a))(Q \subseteq P_i)$$

□

8.3 Universes in Coq and Prop

We give a brief description of the universes in Coq, for a full account see (Coq Reference Manual 2013).

In the Coq system there is an infinite hierarchy of type universes

$$\mathrm{Set} : \mathrm{Type}_1 : \mathrm{Type}_2 : \quad \cdots \quad : \mathrm{Type}_k : \mathrm{Type}_{k+1} \quad \cdots$$

Every object (term) in the system belongs to some universe. Each of the universes is closed under the Π-construction (`forall`) and the construction of inductive types and families (`Inductive`).

The universes are formulated in the "Russell style" (cf. Martin-Löf 1984) so that the decoding functions T are suppressed. Thus we have rules (writing Type_0 for Set):

$$\frac{}{\vdash \mathrm{Type}_k : \mathrm{Type}_{k+1}} \qquad \frac{\vdash A : \mathrm{Type}_k}{\vdash A : \mathrm{Type}_{k+1}} \qquad \frac{\vdash A = B : \mathrm{Type}_k}{\vdash A = B : \mathrm{Type}_{k+1}}.$$

When working with Coq system, one generally sees only the distinction between `Set` and `Type`, the indexes k of the higher universes are not shown.

Though we may use the propositions-as-sets principle in Coq just as in standard Martin-Löf type theory, the Coq system provides a special type `Prop` of propositions should not be understood as sets. We have

$$\vdash \mathrm{Prop} : \mathrm{Type}_1$$

Two main differences from all Type_k is that Prop is not closed under construction of inductive types, and in particular not under the usual Σ construction, and that it is closed under Π in a stronger way than the universes or types:

$$\frac{\vdash A : \mathrm{Type}_k \quad x : A \vdash P(x) : \mathrm{Prop}}{\vdash (\Pi x : A)P(x) : \mathrm{Prop}}.$$

One way to motivate this is to think of Prop like the classical truth-values. It has some strong consequences that are not acceptable from a predicative point of view, and thus not strict constructivists. It allows definition of non-restricted power sets

$$\mathcal{P}(X) = X \to \mathrm{Prop}.$$

Though this leads to more powerful proof methods, a drawback is algorithms can only be extracted indirectly from existence theorems which belongs to Prop. If

$$(\tilde{\exists} x : A)P(x) : \mathrm{Prop}$$

and

$$p : (\tilde{\exists} x : A)P(x)$$

we can in general not find a projection from $(\tilde{\exists} x : A)P(x)$ to A, i.e. find the witness x from p. The reason is that the associated elimination rule for $\tilde{\exists}$ is weaker. We have the following elimination rule for the weak existential quantifier

$$\frac{\vdash C : \text{Prop} \quad \vdash p : (\tilde{\exists} x : A)P(x) \quad x : X, y : P(x) \vdash d(x, y) : C}{\vdash \tilde{E}(p, (x, y)d(x, y)) : C}$$

We see that the possibility to define the first projection as when Σ-elimination is blocked since C can only be in Prop and not in Set. Note also that C does not depend on x and y. This is just as for the usual existence elimination rule in first order logic.

The weak existential quantifier can in fact be defined in terms of the stronger Π above:

$$(\tilde{\exists} x : A)P(x) =_{\text{def}} (\Pi C : \text{Prop})((\Pi x : A)(P(x) \supset C) \supset C).$$

Remark 8.3.1. The basic library of Coq uses Prop as the preferred way of interpreting propositions. Most powerful tactics about logical reasoning deals only with Prop.

Chapter 9

Setoids

A major drawback of the sets in type theory is that they do not admit quotients of sets as in set theory. Instead one usually defines explicit equivalence relations on sets. The combination of a set and an equivalence relation is called a *setoid.* For instance we may be interested to define the integers $\mathbb{Z}$ as formal differences of two natural numbers $(a, b) : \mathrm{N} \times \mathrm{N}$. Two such numbers (a, b) and (c, d) are declared equivalent if their difference is the same:

$$(a, b) =_{\mathbb{Z}} (c, d) \Longleftrightarrow_{\text{def}} a + d =_{\mathrm{N}} b + c$$

This makes the definition of arithmetical operations algebraically smooth e.g.

$$-(a, b) =_{\text{def}} (b, a),$$

$$(a, b) + (c, d) =_{\text{def}} (a + c, b + d)$$

In this case it is strictly speaking not necessary to use an explicit equivalence relation. We could chose a leaner coding, taking for instance as underlying set

$$\mathrm{N} + \mathrm{N}$$

where the elements of the form $\mathrm{inr}(n)$ denotes non-negative integers and elements of the form $\mathrm{inl}(n)$ denotes negative number $-(n+1)$. The arithmetical operations become slightly more cumbersome, e.g.

$$\begin{array}{rcl} -\mathrm{inr}(0) & = & \mathrm{inr}(0) \\ -\mathrm{inr}(S(n)) & = & \mathrm{inl}(n) \\ -\mathrm{inl}(n) & = & \mathrm{inr}(S(n)) \end{array}$$

but the equality would be given by the standard equality $\mathrm{I}(\mathrm{N} + \mathrm{N}, x, y)$. It is not always possible to use the standard equality, for instance when want to consider equality between real numbers or infinite sequences.

9.1 Setoids and extensional functions

A *setoid* X is a pair $(|X|, =_X)$ consisting of a set $|X|$ and an equivalence relation $=_X$ on $|X|$. An *extensional function* f from a setoid X to a setoid Y is a pair $(|f|, \mathrm{ext}_f)$ consisting of a function $|f| : |X| \to |Y|$ and a proof construction witnessing extensionality

$$\mathrm{ext}_f : (\forall x, y : |X|)(x =_X y \supset \mathrm{Ap}(|f|, x) =_Y \mathrm{Ap}(|f|, y)). \tag{9.1}$$

Below we use some overloading of notation so that $x : X$ denotes $x : |X|$ and $\mathrm{Ap}(f, x)$ denotes $\mathrm{Ap}(|f|, x)$. We shall sometimes even write

$$f\,x \text{ for } \mathrm{Ap}(f, x).$$

Two extensional function $f, g : X \longrightarrow Y$ between setoids are *extensionally equal*, in symbols $f =_{\mathrm{ext}} g$, if

$$(\forall x : X) f\,x =_Y g\,x \text{ true}.$$

Let X, Y and Z be setoids. The composition $f \circ g = (h, \mathrm{ext}_h)$ of two extensional functions $f : X \to Y$, $g : Y \to Z$ is given by $h = (\lambda x)(f\,(g\,x)) : |X| \to |Y|$ and ext_h is the proof its extensionality which can be obtained by a suitable composition of the proof construction ext_f and ext_g. For every X we define $\mathrm{id}_X = ((\lambda x)x, (\lambda x)(\lambda y)(\lambda p)p)$ which is the extensional identity function on X.

Lemma 9.1.1. *1. Let X, Y and Z be setoids. Then for all extensional functions $f, h : X \to Y$, $g, k : Y \to Z$:*

$$f =_{\mathrm{ext}} h \supset g =_{\mathrm{ext}} k \supset g \circ f =_{\mathrm{ext}} k \circ h.$$

2. Let X, Y, Z and U be setoids. Then for all extensional functions $f : X \to Y$, $g : Y \to Z$ and $h : Z \to U$:

$$h \circ (g \circ f) =_{\mathrm{ext}} (h \circ g) \circ f.$$

3. For all setoids X and Y and all extensional functions $f : X \longrightarrow Y$:

$$f \circ \mathrm{id}_X =_{\mathrm{ext}} f \qquad \mathrm{id}_Y \circ f =_{\mathrm{ext}} f. \qquad \square$$

From this lemma it follows easily that the setoids and extensional functions form a category (or to be precise an E-category[1]). For every pair of setoids X and Y, let

[1]An E-category is defined by a type (or set) of objects, without imposing any equality between them. For any pair of objects A and B there is a setoid $\mathrm{Hom}(A, B)$ of arrows. There is an identity arrow id_A for each object A, and a extensional composition map $\mathrm{Hom}(B, C) \times \mathrm{Hom}(A, B) \to$ satisfying the usual equalities.

$\mathrm{Hom}(X, Y)$ be the setoid whose underlying set are the extensional functions $X \to Y$ with extensionality proofs

$$(\Sigma f : |X| \to |Y|)[(\forall x, y : X)(x =_X y \supset f\,x =_Y f\,y)]$$

and whose equality relation is given by extensional equality

$$(f, p) =_{\mathrm{Hom}(X,Y)} (g, q) \ =_{\mathrm{def}} \ (f =_{\mathrm{ext}} g).$$

Some standard definitions and results carry over from set theory to setoid theory: For extensional $f : X \longrightarrow Y$:
$f : X \longrightarrow Y$ is *injective,* if for all $x, y : X$, $f\,x =_Y f\,y$ implies $x =_X y$,
$f : X \longrightarrow Y$ is *surjective,* for every $y : Y$, there is $x : X$ with $f\,x =_Y y$,
$f : X \longrightarrow Y$ is *bijection,* if it is both injective and surjective.

Using Unique Choice it is easy to show that

Lemma 9.1.2. *An extensional $f : X \longrightarrow Y$ is a bijection if and only if it has an inverse, i.e. there is an extensional $g : Y \to X$ such that $g \circ f =_{\mathrm{ext}} \mathrm{id}_X$ and $f \circ g =_{\mathrm{ext}} \mathrm{id}_Y$.* $\square$

9.1.1 Some constructions

For any set A, define the setoid $\overline{A} = (A, \mathrm{I}(A, \cdot, \cdot))$, the *free setoid on A.* Any predicate $P(x)$ set $(x : A)$ or function $f : A \to B$ is extensional with respect to $\overline{A}$:

$$P(x) \wedge \mathrm{I}(A, x, y) \supset P(y),$$

$$\mathrm{I}(A, x, y) \supset f\,x =_B f\,y.$$

This is an easy consequence of I-elimination.

If A and B are setoids, then their *cartesian product* setoid is

$$A \times B =_{\mathrm{def}} (|A| \times |B|, =_{A \times B})$$

where

$$(x, y) =_{A \times B} (u, v) \Longleftrightarrow_{\mathrm{def}} x =_A u \wedge y =_B v.$$

The two projections from $|A| \times |B|$ form extensional functions $\pi_1 : A \times B \to A$ and $\pi_2 : A \times B \to B$ that satisfy the following property

Lemma 9.1.3. *Let A and B be setoids. If $x : A$ and $y : B$ then there is a unique $u : A \times B$ (up to $=_{A \times B}$) with $\pi_1\,u =_A x$ and $\pi_2\,u =_B y$.* $\square$

Let A and B be setoids. Their *disjoint union* setoid is

$$A + B =_{\text{def}} (|A| + |B|, =_{A+B})$$

where

$$\begin{aligned} u =_{A+B} v \Longleftrightarrow_{\text{def}} & (\exists a, c : A)(\mathrm{I}(A + B, u, \mathrm{inl}(a)) \wedge \mathrm{I}(A + B, v, \mathrm{inl}(c)) \wedge a =_A c) \vee \\ & (\exists b, d : B)(\mathrm{I}(A + B, u, \mathrm{inr}(b)) \wedge \mathrm{I}(A + B, v, \mathrm{inr}(d)) \wedge b =_B d). \end{aligned}$$

The two injections into $|A| + |B|$ form extensional functions $\mathrm{inl} : A \to A + B$ and $\mathrm{inr} : B \to A + B$ that satisfy the following property

Lemma 9.1.4. *Let A and B be setoids. If $u : A + B$, then there is a unique $a : A$ (up to $=_A$) such that $\mathrm{inl}(a) =_{A+B} u$, or there is a unique $b : B$ (up to $=_B$) such that $\mathrm{inr}(b) =_{A+B} u$.* □

Free setoids are convenient to work with as they do not require extensionality proofs. They behave well with respect to the construction $\times$ and $+$:

Theorem 9.1.5. *Let A and B be sets. Then*

1. $\mathrm{I}(A \times B, (x, y), (u, v))$ *if and only if* $\mathrm{I}(A, x, u)$ *and* $\mathrm{I}(B, y, v)$,
2. $\mathrm{I}(A + B, \mathrm{inl}(x), \mathrm{inl}(u))$ *if and only if* $\mathrm{I}(A, x, u)$.
3. $\mathrm{I}(A + B, \mathrm{inr}(y), \mathrm{inr}(v))$ *if and only if* $\mathrm{I}(B, y, v)$.
4. $\neg\mathrm{I}(A + B, \mathrm{inl}(x), \mathrm{inr}(y))$.
5. $\neg\mathrm{I}(A + B, \mathrm{inr}(y), \mathrm{inl}(x))$.

Proof. The first three items are proved by straightforward applications of I-elimination. As for (4), define $f : A + B \to \mathrm{N}$, by $f = (\lambda u)D(u, (x)0, (y)S(0))$. Then if $\mathrm{I}(A + B, \mathrm{inl}(x), \mathrm{inr}(y))$, we obtain by extensionality on free setoids,

$$\mathrm{I}(\mathrm{N}, \mathrm{Ap}(f, \mathrm{inl}(x)), \mathrm{Ap}(f, \mathrm{inr}(y))) \text{ true.}$$

Evaluating the function f on these arguments we get

$$\mathrm{I}(\mathrm{N}, 0, S(0)) \text{ true}$$

which contradicts Peano's fourth axiom (verified above). Item (5) follows by symmetry. □

Corollary 9.1.6. *For sets A and B we have the following isomorphisms of setoids*

1. $\overline{A \times B} \cong \overline{A} \times \overline{B}$,
2. $\overline{A + B} \cong \overline{A} + \overline{B}$. □

9.2 Quotients

A *quotient* of a setoid X is a setoid Q and an extensional function $q : X \to Q$ satisfying the following universal property: for any extensional function $f : X \to Y$ such that

$$(\forall x, y : X)(q\,x =_Q q\,y \supset f\,x =_Y f\,y)$$

there exist a unique extensional function $\bar{f} : Q \to Y$ with $\bar{f} \circ q =_{\mathrm{ext}} f$.

Theorem 9.2.1. *Let X be a setoid. Then an extensional function $q : X \to Q$ is a quotient if and only if it is surjective.*

Proof. ($\Leftarrow$) Suppose that q is surjective. Then by the type-theoretic axiom of choice we find $g : |Q| \to |X|$ such that

$$(\forall y : Q) q\,(g\,y) =_Q y. \tag{9.2}$$

Note that g need not be extensional. Let $f : X \to Y$ such that

$$(\forall x, y : X)(q\,x =_Q q\,y \supset f\,x =_Y f\,y) \tag{9.3}$$

Define $|\bar{f}| = (\lambda u)(f\,(g\,u))$. Suppose $y =_Q y'$. Thus by (9.2),

$$q\,(g\,y) =_Q y =_Q y' =_Q q\,(g\,y').$$

and thus by the property (9.3) of f,

$$f\,(g\,y) =_Y f\,(g\,y').$$

Thus $|\bar{f}|$ is extensional $Q \to Y$. Now

$$\bar{f}\,(q\,x) =_Y f\,(g\,(q\,x)) =_Y f\,x$$

where the last equality follows from (9.3) since, according to (9.2), with $y = q\,x$,

$$q\,(g\,(q\,x)) =_Q q\,x.$$

If $\hat{f} : Q \to Y$ is another extensional function with

$$\hat{f}\,(q\,x) =_Y f\,x \quad (x : X).$$

We get by the surjectivity of q that $\hat{f}$ and $\bar{f}$ are extensionally equal as required.

($\Rightarrow$) Suppose that $q : X \to Q$ is a quotient of X. The following argument is essentially in (Mines *et al.* 1998, Theorem I.4.1). We use a more elementary construction of (Wilander 2010, p. 565) and build the following setoid Y. Let

$$|Y| =_{\mathrm{def}} \mathrm{N}_2 \times |Q|$$

and

$$((m, u) =_Y (n, v)) =_{\text{def}} (C(m, u) \Leftrightarrow C(n, v))$$

and using

$$C(m, u) =_{\text{def}} (\text{Tr}(m) \supset (\exists x : X)(q\, x =_Q u).$$

Define $f : X \longrightarrow Y$ by $f =_{\text{def}} (\lambda x)(0_2, q\, x)$. This is an extensional (and constant) function since $C(0_2, q\, x)$ is true for all $x : X$. Define $\bar{f} =_{\text{def}} (\lambda u)(0_2, u)$ This is a constant and extensional function $Q \to Y$ and $\bar{f} \circ q =_{\text{ext}} f$. Construct a different function $\hat{f} = (\lambda u)(1_2, u)$. It is also extensional $Q \to Y$ since if $u =_Q v$, then by transitivity of $=_Q$,

$$C(1_2, u) \Leftrightarrow (\exists x : X)q\, x =_Q u \Leftrightarrow (\exists x : X)q\, x =_Q v \Leftrightarrow C(1_2, v).$$

But for $u = q\, x$ the equivalence $C(1_2, q\, x) \Leftrightarrow C(0_2, q\, x)$ holds, so $\hat{f} \circ q =_{\text{ext}} f$. By uniqueness, $\hat{f} =_{\text{ext}} \bar{f}$. For each $u : Q$, we have thus

$$C(0_2, u) \Leftrightarrow C(1_2, u).$$

Since $C(0_2, u)$ is trivially true, the right hand holds, which says

$$\text{Tr}(1_2) \supset (\exists x : X)q\, x =_Q u,$$

Hence as $\text{Tr}(1_2)$ is true, there is $x : X$, $q\, x =_Q u$. Thus we have shown that q is surjective. □

A setoid Y is *projective* if for every surjective $f : X \to Y$, there is an extensional $g : Y \to X$ with $f \circ g =_{\text{ext}} \text{id}_Y$. Using the type-theoretic axiom of choice it is easy to show that the free setoid $(A, \text{I}(A, \cdot, \cdot))$ is projective. More over if $(A, =_A)$ is a setoid, then $(\lambda x)x$ defines a surjective function $(A, \text{I}(A, \cdot, \cdot)) \to (A, =_A)$.

Corollary 9.2.2. *Every setoid is the quotient of a free setoid, and hence of a projective setoid.*

Proof. For a setoid $X = (|X|, =_X)$ we note that $(\lambda x)x$ is a surjective map from the free setoid $(|X|, \text{I}(|X|, \cdot, \cdot))$ to X. Thus X is a quotient of $(|X|, \text{I}(|X|, \cdot, \cdot))$. Every free setoid is projective. □

A setoid A is *discrete* if $=_A$ is decidable, i.e. for all $x, y : A$,

$$x =_A y \vee \neg\, x =_A y.$$

Proposition 9.2.3. *If A and B are discrete setoids, then so are $A \times B$ and $A + B$.* □

9.3 Subsets

We already have encountered one way of defining a subcollection of a setoid X: namely as a predicate $P(x)$ set $(x : X)$, that is *extensional*

$$(\forall x : X)(P(x) \wedge x =_Y y \supset P(y)) \text{ true.}$$

We can construct a new setoid set consisting of just those elements of X which satisfy P, using the Σ-type

$$\{x : X \mid P(x)\} =_{\text{def}} \big((\Sigma x : X)P(x), \pi_1(\cdot) =_A \pi_1(\cdot)\big)$$

Note that elements (x, p) and (y, q) of this setoid are equal precisely when $x =_X y$. Then the first projection π becomes an extensional injection

$$\pi_1 : \{x : X \mid P(x)\} \longrightarrow X.$$

We can denote this as $\{x : X \mid P(x)\} \hookrightarrow X$. This leads to a more abstract notion of subsetoid.

Let X be a setoid. A *subsetoid* of X is an injective function $i_A : \partial A \to X$ where A is a setoid. For an element $x : X$, we say that x *belongs to* $A = (\partial A, i_A)$, in symbols $x \in_X A$, if there is some $a : A$ with $i_A(a) =_X x$. Note that

$$x \in_X A \text{ set } (x : A)$$

is an extensional predicate on X.

If $i_A : \partial A \to X$ and $i_B : \partial B \to X$ are two subsets we write $A \subseteq B$, if for every $x : X$, $x \in_X A$ implies $x \in_X B$. This is in fact equivalent to the existence of an extensional function $f : \partial A \longrightarrow \partial B$ such that $i_B \circ f =_{\text{ext}} i_A$.

A subsetoid A of X is *detachable,* if for all $x : X$

$$x \in_X A \vee \neg x \in_X A.$$

9.4 Finite and countable sets

The canonical k-element setoid $\mathbb{N}_k$ is

$$\{n : \mathbb{N} \,|\, n < k\}.$$

Write n'_k for $(n, p_{n,k})$ where $p_{n,k} : n < k$. A setoid X is *finite* if it is isomorphic to some $\mathbb{N}_k$, $k : \mathrm{N}$; it is *finitely enumerable* if there is a surjection $\mathbb{N}_k \to X$ for some k.

Proposition 9.4.1. *(a) Every finite setoid is finitely enumerable.*

(b) A finitely enumerable setoid is finite if and only if it discrete.

(c) A detachable subsetoid of finite setoid is finite.

(d) It can be decided whether a finitely enumerable setoid, is inhabited or not.

Proof. (a): this follows since every isomorphism is surjective. (d): Suppose that $f : \mathbb{N}_k \twoheadrightarrow X$ is a surjection. Then X is inhabited precisely when $k > 0$. $\square$

In general subsetoids of finite setoids cannot be proved to be finite or even finitely enumerable. Let P be an arbitrary proposition, and form the subsetoid

$$\{x : \mathbb{N}_1 \mid P\} \hookrightarrow \mathbb{N}_1.$$

(In classical set theory, this subset is finite.) However, if we assume that it is finitely enumerable then there is k and a surjection

$$f : \mathbb{N}_k \to \{x : \mathbb{N}_1 \mid P\}.$$

Thus $k = 0$ or $k > 0$. If $k = 0$, then assuming P, we get an element $y : \{x : \mathbb{N}_1 \mid P\}$, so by surjectivity there is $u : \mathbb{N}_0$ which f maps to y. This is impossible. Hence $\neg P$. If $k > 0$, then f provides an element in $\{x : \mathbb{N}_1 \mid P\}$ which implies that P is true. The assumption of finite enumerability thus leads to $P \vee \neg P$. From this reasoning follows:

Theorem 9.4.2. *If subsetoids of finite setoids, are finitely enumerable, then PEM is valid.* $\square$

Zermelo's axiom of choice fails even for finitely enumerable sets.

Theorem 9.4.3. *(Diaconescu's theorem) For any proposition P, there is a setoid M and a surjective function $f : \mathbb{N}_2 \twoheadrightarrow M$, such that there is an extensional $g : M \to \mathbb{N}_2$ with $f \circ g =_{\mathrm{ext}} M$, then $P \vee \neg P$.*

Proof. Let P be an arbitrary proposition. Consider the finite set $\mathbb{N}_2$. Now define a quotient of this set by letting $M = (\mathrm{Fin}_2, \sim)$ where

$$a \sim b \Longleftrightarrow_{\mathrm{def}} a =_{\mathbb{N}_2} b \vee P$$

This relation is easily seen to be an equivalence relation, and the identity $(\lambda x)x$ becomes a surjective extensional function $f : \mathbb{N}_2 \twoheadrightarrow M$. Suppose now that $g : M \to \mathbb{N}_2$ is an extensional function with

$$f \circ g =_{\mathrm{ext}} \mathrm{id}_M. \tag{9.4}$$

Now $\mathbb{N}_2$ is a discrete set so there are two cases

$$g\,0_2' =_{\mathbb{N}_2} g\,1_2' \vee \neg(g\,0_2' =_{\mathbb{N}_2} g\,1_2').$$

In the former case, we get by (9.4), $0_2' \sim 1_2'$. But $\neg 0_2' =_{\mathbb{N}_2} 1_2'$, so P must hold. In the latter case, that is $\neg(g\,0_2' =_{\mathbb{N}_2} g\,1_2')$. Suppose P is true, then $0_2' \sim 1_2'$, and hence also $g\,0_2' =_{\mathbb{N}_2} g\,1_2'$ by extensionality. But this is a contradiction. Hence $\neg P$. Thus we have proved $P \vee \neg P$. $\square$

Corollary 9.4.4. *(Diaconescu) If Zermelo's axiom of choice is true, then PEM holds.*

Proof. Let P and M be as in the theorem. Let $f : \mathbb{N}_2 \longrightarrow M$ be a surjective function. Thus

$$(\forall x : M)(\exists n : \mathbb{N}_2) f\, n =_M x.$$

If Zermelo's axiom of choice holds, there is an extensional function $g : M \longrightarrow \mathbb{N}_2$ such that

$$(\forall x : M) f\,(g\, x) =_M x.$$

But this is $f \circ g =_{\text{ext}} M$, so by the theorem $P \vee \neg P$. □

A setoid X is *countable* if it is a quotient of a detachable D subsetoid of $\mathbb{N}$.

Example 9.4.5. *Any quotient of* $\mathbb{N}$ *is countable.*

Example 9.4.6. *Any finitely enumerable set is countable.*

Lemma 9.4.7. *Any detachable subsetoid of a countable setoid is countable.* □

Proof. Suppose that X is countable, so there is a detachable subsetoid $D \hookrightarrow \mathbb{N}$, and a surjection $f : D \to X$. Let $d : D \to \mathbb{N}$ be injection of the subsetoid. Assume now that we are given a detachable subsetoid $e : E \hookrightarrow X$. Form the subsetoid

$$D_E =_{\text{def}} \{n : N \mid (\exists u : D) d\, u =_{\mathbb{N}} n \wedge f\, u \in_X E\} \overset{\pi_1}{\hookrightarrow} \mathbb{N}.$$

This may be proved to be a detachable subsetsetoid of $\mathbb{N}$ by using that $n \in_X D$ is decidable and then that $f\, u \in_X E$ is decidable. For every $(n, p) : D_E$, there is $u : D$ and $v : E$ with $d\, u =_{\mathbb{N}} n$ and $e\, v =_{\mathbb{N}} f\, u$. Now both u and v are unique since d and e are injective. By unique choice we have two extensional functions $g : D_E \to D$ and $h : D_E \to E$ such that $d\,(g\,(n, p)) =_{\mathbb{N}} n$ and $e\,(h\,(n, p)) =_{\mathbb{N}} f\,(g\,(n, p))$. We claim that h is surjective. For any $z : E$, there is $u : D$ with $f\, u =_X e\, z$, since f is surjective. Thus $(d\, u, p) : D_E$ for some proof construction p. Hence $d\,(g\,((d\, u), p)) =_{\mathbb{N}} d\, u$ and $e\,(h\,((d\, u), p))) =_{\mathbb{N}} f\,(g\,((d\, u), p)))$. By injectivity of d, we have $u =_D g\,((d\, u), p)$, so

$$e\,(h\,((d\, u), p)) =_X f\, u =_X e\, z.$$

Hence by injectivity of e, $h\,((d\, u), p))) =_E z$ as required. □

Lemma 9.4.8. *If A and B are countable setoids, then so are $A \times B$ and $A + B$.*

Proof. Use some standard isomorphisms $\mathbb{N} + \mathbb{N} \cong \mathbb{N}$ and $\mathbb{N} \times \mathbb{N} \cong \mathbb{N}$. The details are left to the reader. □

Example 9.4.9. Hom($\mathbb{N}, \mathbb{N}$) is not countable. Suppose that there is a surjective function $f : D \to \mathrm{Hom}(\mathbb{N}, \mathbb{N})$ from a detachable subsetoid D of $\mathbb{N}$. Let $d : D \to \mathbb{N}$ be the injection associated with the subsetoid. Define $h : \mathbb{N} \to \mathbb{N}$ by

$$h =_{\mathrm{def}} (\lambda n) \begin{cases} ((f\,u)\,n) + 1 & \text{if } n \in_X D \text{ and } u : D \text{ unique such that } d\,u =_N n \\ 0 & \text{otherwise} \end{cases}$$

Suppose now that for some $u : D$, we have $f\,u =_{\mathrm{ext}} h$. Now by definition

$$h\,(d\,u)) =_{\mathbb{N}} ((f\,u)\,(d\,u)) + 1.$$

But this contradicts the equality $f\,u =_{\mathrm{ext}} h$ for the argument $d\,u$. Hence we must conclude that Hom($\mathbb{N}, \mathbb{N}$) is not countable.

9.4.1 Subsetoids of countable setoids and Kripke's Schema

Subsetoids of countable setoids are not necessarily countable (Mines *et al.* 1988, p. 32). To argue for this we consider a general principle called *Kripke's Schema* (KS). It is the following: (KS) for every proposition P there is a function $f : \mathbb{N} \to \mathbb{N}$ such that

$$P \iff (\exists n : \mathbb{N}) \neg f\,n =_{\mathbb{N}} 0.$$

It is trivially true, classically, but not provable constructively (as is it false in realizability models where Markov's principle holds; see Bridges and Richman (1987) and below).

Theorem 9.4.10. *The following are equivalent:*

1. *Every subsetoid of* $\mathbb{N}$ *is countable.*
2. *Kripke's Schema.*

Proof. ($1 \Rightarrow 2$): Assume (1). Let P set be a proposition. Define the subsetoid

$$\{x : \mathbb{N} \mid P\} \overset{\pi_1}{\hookrightarrow} \mathbb{N}.$$

Thus there is a detachable $d : D \hookrightarrow \mathbb{N}$ and a surjection $h : D \to \{x : \mathbb{N} \mid P\}$. Define a function $\mathrm{N} \to \mathrm{N}$,

$$f =_{\mathrm{def}} (\lambda n) \begin{cases} 1 & \text{if } n \in_{\mathbb{N}} D \\ 0 & \text{if } \neg\, n \in_{\mathbb{N}} D \end{cases}$$

Suppose P holds. Then there is $u : D$ such that $h(u) = 0$. Hence $d\,u \in_{\mathbb{N}} D$ and

$$\neg f\,(d\,u) =_{\mathbb{N}} 0.$$

Conversely, if there is n with $\neg\mathrm{I}(\mathrm{N}, f\,n, 0)$, then $n \in_{\mathbb{N}} D$ which implies there is some $u : D$ such that $d\,u =_{\mathbb{N}} n$. But then $h\,u : \{x : \mathbb{N} \mid P\}$, so P holds. Thus we have proved (2).

($\Leftarrow$) Assume (1). Let $A \hookrightarrow \mathbb{N}$ be a subsetoid of $\mathbb{N}$. Define the family of propositions

$$P(n) =_{\mathrm{def}} n \in_{\mathbb{N}} A \quad (n : \mathbb{N})$$

By Kripke's Schema we have

$$(\forall n : \mathbb{N})(\exists f : \mathbb{N} \longrightarrow \mathbb{N})(P(n) \Longleftrightarrow (\exists m)\neg\, f\,m =_{\mathbb{N}} 0).$$

By the type-theoretic axiom of choice we obtain $g : \mathrm{N} \to (\mathrm{N} \to \mathrm{N})$ such that

$$(\forall n : \mathbb{N})(P(n) \Longleftrightarrow (\exists m : \mathbb{N})\neg(g\,n)\,m =_{\mathbb{N}} 0).$$

Define a setoid

$$D = \{(n, m) : \mathbb{N} \times \mathbb{N} \mid (g\,n)\,m =_{\mathbb{N}} 0\}.$$

The first projection $\pi_1 : D \to \mathbb{N} \times \mathbb{N}$ is an injection. Let $\phi : \mathbb{N} \times \mathbb{N} \to \mathbb{N}$ be a bijective coding of pairs of natural numbers. Then $\phi \circ \pi_1 : D \to \mathbb{N}$ is an injection as well, and D becomes a detachable subsetoid of $\mathbb{N}$ since $=_{\mathbb{N}}$ is decidable. Define the following subsetoid of $\mathbb{N}$:

$$B =_{\mathrm{def}} \{n : \mathbb{N} \mid (\exists m : \mathbb{N})(n, m) \in_{\mathbb{N}\times\mathbb{N}} D\}.$$

The extensional projection $p : D \to B$ is clearly surjective and by the above we have

$$(\forall n : \mathbb{N})(n \in_{\mathbb{N}} A \Leftrightarrow P(n) \Leftrightarrow n \in_{\mathbb{N}} B)$$

Hence A and B are equal as subsetoids of $\mathbb{N}$. Hence A is countable. □

Markov's Principle (MP) is the following: for any $f : \mathbb{N} \to \mathbb{N}$,

$$\neg\neg(\exists n : \mathbb{N})f\,n =_{\mathbb{N}} 0 \supset (\exists n : \mathbb{N})f\,n =_{\mathbb{N}} 0.$$

MP is true in many realizability interpretations of constructive systems. Together with Kripke's Schema it yields PEM (Bridges and Richman, 1987). This is a reason to reject KS from a constructive point of view.

Theorem 9.4.11. *KS and MP implies PEM.*

Proof. Let P be a proposition. Then by KS there are functions $f, g : \mathbb{N} \to \mathbb{N}$ such that

$$P \Longleftrightarrow (\exists n : \mathbb{N})\neg f\,n =_{\mathbb{N}} 0$$

and

$$\neg P \Longleftrightarrow (\exists n : \mathbb{N})\neg g\,n =_{\mathbb{N}} 0$$

Thus

$$\begin{aligned} P \vee \neg P &\iff (\exists n : \mathbb{N})(\neg f\, n =_{\mathbb{N}} 0 \vee \neg g\, n =_{\mathbb{N}} 0) \\ &\iff (\exists n : \mathbb{N})(h\,((f\, n) + (g\, n)) =_{\mathbb{N}} 0) \end{aligned}$$

Here h is the function which is such that $h\,0 = S(0)$ and $h\,S(m) = 0$. However we can always prove in intuitionistic logic

$$\neg\neg(P \vee \neg P).$$

Hence it follows by the above equivalence that

$$\neg\neg(\exists n : \mathbb{N})h\,((f\, n) + (g\, n)) =_{\mathbb{N}} 0.$$

By Markov's Principle,

$$(\exists n : \mathbb{N})h\,((f\, n) + (g\, n)) =_{\mathbb{N}} 0.$$

Which then gives $P \vee \neg P$, by the same equivalence again. $\square$

Chapter 10

Inductive sets, families and predicates in Coq

10.1 Simple inductive sets

In the Coq system the simple inductive sets N_2 and N of Martin-Löf type theory can be defined as follows. The two constructors are separated by a vertical bar `|` .

```
Inductive N_2: Set :=
   zero_2:N_2  |  one_2:N_2.

Inductive N:Set :=
    zero: N  |  succ : N -> N.
```

The +-set construction can be defined as an inductive set with parameters

```
Inductive sum' (A B:Set):Set :=
   inl': A -> sum' A B  |  inr': B -> sum' A B.
```

The Σ-set construction is an inductive set with a single constructor and dependent parameters

```
Inductive Sigma (A:Set)(B:A -> Set):Set :=
    Spair: forall a:A, forall b : B a, Sigma A B.
```

The general pattern is that a new set, that is an element of the type `Set`, is defined by giving the typing of some constructor names.

10.2 Inductive families and predicates

A construction in Martin-Löf type theory that doesn't follow this pattern is the identity type construction I. This is an example of an *inductively defined family.* Here an element is introduced in a family of sets. For a fixed element $a : A$, we may think of the identity set as a family over A

$$\mathrm{I}(A, a, x) \text{ set } (x : A).$$

```
Inductive I (A: Set)(x: A) : A -> Set :=
     r :  I A x x.
```

The transitive closure of a relation may be defined as an inductive predicate

```
Inductive tc  (A:Set)(R:A->A->Set) : A -> A -> Set :=
     incl: (forall x y:A, R x y -> tc A R x y)
   | tran: (forall x y z:A,
       tc A R x y -> R y z -> tc A R x z).
```

The smallest equivalence relation that includes a given predicate is given by

```
Inductive eqc  (A:Set)(R:A->A->Set) : A -> A -> Set :=
     incl: (forall x y:A, R x y -> eqc A R x y)
   | refr (forall x :A,  eqc A R x x)
   | symm (forall x y :A,  eqc A R x y -> eqc A R y x)
   | tran: (forall x y z:A,
       eqc A R x y -> eqc R y z -> eqc A R x z).
```

10.3 General form of inductive definitions in Coq

We follow Paulin-Mohring (1992) in this presentation but adapt the notation. A *sort* is any of the types `Set`, `Type` or `Prop`. A type A is an *arity* with target sort s if it has the form

$$(\forall x_1 : M_1, \forall x_2 : M_2, \ldots, \forall x_m : M_m, s).$$

Here $\forall$ is notation for `forall`, and M_k is a type that can depend on the variables $x_1, \ldots, x_{k-1}$. Note that `Set` is an arity taking $m = 0$. The arity associated with the identity type above is `A -> Set` which can be written `(`$\forall$ `_ :  A, Set)`. The arity associated with `eqc` is similarly `(`$\forall$ `_ :  A,` $\forall$ `_ :  A, Set)`.

Let A be an arity and suppose $X : A$. In a type P of the form

$$(\forall y_1 : N_1, \forall y_2 : N_2, \ldots, \forall y_n : N_n, X\, m_1 \cdots m_k),$$

we say that X occurs *strictly positively* in P, if X does not occur in $N_1, \ldots, N_n$ or in $m_1, \ldots, m_k$. *Admissible constructor forms for* $X : A$ are built up as follows

- X is admissible,
- A term $C\,m$ is admissible, if C is admissible, and X does not occur in m.
- A term $P \to C$ is admissible, if C is admissible and X occurs strictly positively in P
- A term $(\forall x : M)C$ is admissible, if C is admissible and X does not occur in M.

The formation rule for inductive sets is

$$\frac{\Gamma \vdash A : \texttt{Type} \quad \Gamma, X : A \vdash C_1 : s \quad \cdots \quad \Gamma, X : A \vdash C_n : s}{\Gamma \vdash \mathrm{Ind}(X : A)\{C_1 | \cdots | C_n\}} \tag{10.1}$$

where A is an arity for s and each C_i is an admissible constructor form for $X : A$. Here an inductive set with n different (nameless) constructors is introduced.

Example 10.3.1. The inductive set N_2 is given by

$$I_1 =_{\mathrm{def}} \mathrm{Ind}(X : \texttt{Set})\{X | X\}$$

Example 10.3.2. The inductive set N is given by

$$I_2 =_{\mathrm{def}} \mathrm{Ind}(X : \texttt{Set})\{X | X \to X\}$$

Example 10.3.3. The inductive family $\mathrm{I}\,A\,a$ is given by

$$I_3 =_{\mathrm{def}} \mathrm{Ind}(X : A \to \texttt{Set})\{X\,a\}$$

Example 10.3.4. The inductive family $\texttt{tc}\,A\,R$ is given by

$$\begin{aligned} I_4 =_{\mathrm{def}} \mathrm{Ind}(X : A \to A \to \texttt{Set})\{&(\forall x : A, \forall y : A, R\,x\,y \to X\,x\,y) \\ &| \ (\forall x : A, \forall y : A, \forall z : A, X\,x\,y \to R\,y\,z \to X\,x\,z)\} \end{aligned}$$

The introduction rule corresponding to (10.1) is as follows. Let $I =_{\mathrm{def}} \mathrm{Ind}(X : A)\{C_1 | \cdots | C_n\}$. For each $i = 1, \ldots, n$

$$\frac{}{\mathrm{Constr}(i, I) : C_i[I/X]}$$

Here $C_i[I/X]$ denotes the result of substituting I for X in C_i.

For the examples above we have

$$\frac{}{\mathrm{Constr}(1, I_1) : I_1} \qquad \frac{}{\mathrm{Constr}(2, I_1) : I_1}$$

and

$$\frac{}{\mathrm{Constr}(1, I_2) : I_2} \qquad \frac{}{\mathrm{Constr}(2, I_2) : I_2 \to I_2}.$$

Moreover,

$$\frac{}{\mathrm{Constr}(1, I_3) : I_3\, a},$$

With $I_3 = \mathrm{I}\, A\, a$ this is as above. Further, for $I_4 = \mathtt{tc}\, A\, R$,

$$\frac{}{\mathrm{Constr}(1, I_4) : \forall x : A, \forall y : A, R\, x\, y \to I_4\, x\, y}$$

and

$$\frac{}{\mathrm{Constr}(2, I_4) : \forall x : A, \forall y : A, \forall z : A, X\, x\, y \to R\, y\, z \to I_4\, x\, z}$$

Let $A = (\forall x_1 : M_1, \forall x_2 : M_2, \ldots, \forall x_m : M_m, s)$ be an arity, and let $X : A$. Let s' be a sort and

$$Q : \forall x_1 : M_1, \forall x_2 : M_2, \ldots, \forall x_m : M_m, X\, x_1 \cdots x_m \to s'.$$

For C a constructor form and $c : C$ we define inductively a type

$$C\{X, Q, x\}.$$

There are three cases to consider.

- $(P \to C)\{X, Q, c\} =$

  $$\forall p : P, (\forall y_1 : N_1, \ldots, \forall y_r : N_r, Q\, t_1 \cdots t_m\, (p\, y_1 \cdots y_r)) \to C\{X, Q, c\, p\}$$

  where $c : P \to C$ and $P = (\forall y_1 : N_1, \ldots, \forall y_r : N_r, X\, t_1 \cdots t_m)$ is strictly positive in X a

- $P = (\forall x : M, C)\{X, Q, c\} = \forall x : M, C\{X, Q, c\, x\}$ where $c : (\forall x : M, C)$ and X does not occur in M.

- $(X\, a_1 \cdots a_m)\{X, Q, c\} = Q\, a_1 \cdots a_m\, c$ where $c : X\, a_1 \cdots a_m$.

The elimination and computation rules are then generated from these rules according to the following method.

$$\frac{\begin{array}{l}\Gamma \vdash Q' : (\forall x_1 : M_1, \forall x_2 : M_2, \ldots, \forall x_m : M_m, I\, x_1 \cdots x_m \to s') \\ \Gamma \vdash c' : I\, a_1 \cdots a_m \\ \Gamma \vdash f_1 : C_1\{X, Q, c\}[I, Q', \mathrm{Constr}(1, I)/X, Q, c] \\ \vdots \\ \Gamma \vdash f_n : C_n\{X, Q, c\}[I, Q', \mathrm{Constr}(n, I)/X, Q, c])\}\end{array}}{\mathrm{Elim}(c, Q')\{f_1 | \cdots | f_n\} : Q'\ a_1 \cdots a_m\, c'}$$

Here $C_i\{X, Q, c\}$ is a type that is constructed by recursion on its structure.

10.4 Exercises

1. One of these definitions of inductive set is rejected by the Coq system. Can you explain why with the help of the theory above?

```
Inductive Gadget (A:Set) : Set :=
   gnil : Gadget A
 | gconstr : (forall f: A-> A -> Gadget A, Gadget A).

Inductive Widget (A:Set) : Set :=
   wnil : Widget A
 | wconstr : (forall f: A-> Widget A -> A, Widget A).
```

2. Find the elimination rules for the inductive types defined in Sections 10.1 and 10.2.

3. In this problem we outline a formalization of the lazy addition example of Section 2.2. Define the set of arithmetical expressions as follows in Coq.

```
Inductive Aexp :Set :=
  zer: Aexp
| suc: Aexp -> Aexp
| pls: Aexp -> Aexp -> Aexp.
```

A predicate for recognizing canonical expressions is given by

```
Definition Canonical (x:Aexp):Set :=
or (I Aexp x zer)
   (Sigma Aexp (fun y =>
     I Aexp x (suc y))).
```

The denotational semantics of these expressions are given by recursive definition

```
Definition denotation: Aexp- > N:=
fix F (a: Aexp): N :=
  match a as a0  with
  | zer => zero
  | suc a1 => succ (F a1)
  | pls a1 a2 => add (F a1) (F a2)
  end.
```

The computation relation $\rightsquigarrow$ is given as an inductively defined relation

```
Inductive Comp : Aexp -> Aexp -> Set :=
refrule: forall a: Aexp,
         forall p: Canonical a, Comp a a
| zerrule: forall a b c:Aexp,
         forall p: Comp b zer,
         forall q: Comp a c,
           Comp (pls a b) c
| sucrule: forall a b c:Aexp,
         forall p: Comp b (suc c),
           Comp (pls a b) (suc (pls a c)).

Theorem Only_canonical_results:
(forall x y: Aexp, Comp x y -> Canonical y).
```

Correctness: the computation relation preserves denotation of expressions.

```
Theorem correct_wrt_semantics:
(forall x y: Aexp, Comp x y ->
I N (denotation x) (denotation y)).

Theorem Comp_is_total: (forall x:Aexp,
  (Sigma Aexp (fun y =>
    prod (Comp x y) (Canonical y)))).
```

Chapter 11

Semantics of Dependent Types

A main problem in semantics of dependent types is how to interpret the interaction of substitutions with dependent type constructions. We follow here Hofmann (1997).

11.1 Categories with families

Definition 11.1.1. A *category with families (CwF)* consists of the following data

(a) A category $\mathcal{C}$ with a terminal object $\top$. This is thought of as the category of contexts and substitutions. For $\Gamma \in \mathcal{C}$ denote by $!_\Gamma$ the unique morphism $\Gamma \longrightarrow \top$.

(b) For each object Γ of $\mathcal{C}$, a class $\mathrm{Ty}(\Gamma)$ and for each morphism $f : \Delta \longrightarrow \Gamma$, a class function $\mathrm{Ty}(f) : \mathrm{Ty}(\Gamma) \longrightarrow \mathrm{Ty}(\Delta)$, for which use the notation $\sigma\{f\}$ for $\mathrm{Ty}(f)(\sigma)$, to suggest that it is the result of performing the substitution f in the type σ. These functions should satisfy, for all $\sigma \in \mathrm{Ty}(\Gamma)$, and $g : \Theta \longrightarrow \Delta$, $f : \Delta \longrightarrow \Gamma$:

   – $\sigma\{1_\Gamma\} = \sigma$,

   – $\sigma\{f \circ g\} = \sigma\{f\}\{g\}$.

   (Thus Ty may be regarded as a functor $\mathcal{C}^{\mathrm{op}} \longrightarrow \mathrm{Class}$.)

(c) For each $\sigma \in \mathrm{Ty}(\Gamma)$, an object $\Gamma.\sigma$ in $\mathcal{C}$ and a morphism $\mathrm{p}(\sigma) = \mathrm{p}_\Gamma(\sigma) : \Gamma.\sigma \longrightarrow \Gamma$. This tells that each context can be extended by a type in the context, and that there is a projection from the extended context to the original one.

(d) For each $\sigma \in \mathrm{Ty}(\Gamma)$, there is a class $\mathrm{Tm}(\Gamma, \sigma)$ — thought of as the terms of type σ. It should be such that for $f : \Delta \longrightarrow \Gamma$ there is a class function $\mathrm{Tm}(f) : \mathrm{Tm}(\Gamma, \sigma) \to \mathrm{Tm}(\Delta, \sigma\{f\})$, where we write $N\{f\}$ for $\mathrm{Tm}(f)(N)$. It should satisfy the following

   – $N\{1_\Gamma\} = N$ for $N \in \mathrm{Tm}(\Gamma, \sigma)$ $(= \mathrm{Tm}(\Gamma, \sigma\{1_\Gamma\}))$.

  - $N\{f \circ g\} = N\{f\}\{g\}$ for $N \in \mathrm{Tm}(\Gamma, \sigma)$
    (Note: $\mathrm{Tm}(\Theta, \sigma\{f \circ g\}) = \mathrm{Tm}(\Theta, \sigma\{f\}\{g\})$.)

(e) For each $\sigma \in \mathrm{Ty}(\Delta)$ there is an element $\mathrm{v}_\sigma \in \mathrm{Tm}(\Delta.\sigma, \sigma\{\mathrm{p}(\sigma)\})$.

(f) For any morphism $f : \Gamma \longrightarrow \Delta$ and $M \in \mathrm{Tm}(\Gamma, \sigma\{f\})$, there is

$$\langle f, M\rangle_\sigma : \Gamma \longrightarrow \Delta.\sigma.$$

This construction should satisfy

  - $\mathrm{p}(\sigma) \circ \langle f, M\rangle_\sigma = f$
  - $\mathrm{v}_\sigma\{\langle f, M\rangle_\sigma\} = M$
    (Note: $\mathrm{v}_\sigma\{\langle f, M\rangle_\sigma\} \in \mathrm{Tm}(\Gamma, \sigma\{\mathrm{p}(\sigma)\}\{\langle f, M\rangle_\sigma\}) = \mathrm{Tm}(\Gamma, \sigma\{f\}))$
  - $\langle \mathrm{p}(\sigma), \mathrm{v}_\sigma\rangle_\sigma = 1_{\Gamma.\sigma}$

and moreover for any $g : \Theta \longrightarrow \Gamma$,

$$\langle f, M\rangle_\sigma \circ g = \langle f \circ g, M\{g\}\rangle_\sigma$$

(Remark: $M\{g\} \in \mathrm{Tm}(\Theta, \sigma\{f\}\{g\}) = \mathrm{Tm}(\Theta, \sigma\{f \circ g\})$.) $\square$

Note that for any $f : \Gamma \longrightarrow \Delta.\sigma$, we have by the equations

$$f = 1_{\Gamma.\sigma} \circ f = \langle \mathrm{p}(\sigma), \mathrm{v}_\sigma\rangle_\sigma \circ f = \langle \mathrm{p}(\sigma) \circ f, \mathrm{v}_\sigma\{f\}\rangle. \tag{11.1}$$

Hence if $g : \Gamma \longrightarrow \Delta.\sigma$ is such that $\mathrm{p}(\sigma) \circ f = \mathrm{p}(\sigma) \circ g$ and $\mathrm{v}_\sigma\{f\} = \mathrm{v}_\sigma\{g\}$, then $f = g$. In particular, we have

$$\langle f, M\rangle_\sigma = \langle f', M'\rangle_\sigma \text{ iff } f = f' \text{ and } M = M'.$$

A context Γ is called *finite* if it has the form

$$\Gamma = \top.\sigma_1.\ldots.\sigma.$$

Maps into such contexts can be described as follows.

Proposition 11.1.2. *Suppose* $\sigma_1 \in \mathrm{Ty}(\top)$, $\sigma_2 \in \mathrm{Ty}(\top.\sigma_1)$, ..., $\sigma_n \in \mathrm{Ty}(\top.\sigma_1.\ldots.\sigma_{(n-1)})$. *Every* $f : \Gamma \to \top.\sigma_1.\ldots.\sigma_n$ *can be written uniquely as*

$$f = \langle \cdots \langle\langle !_\Gamma, M_1\rangle_{\sigma_1}, M_2\rangle_{\sigma_2}, \ldots, M_n\rangle_{\sigma_n}$$

where

$$M_i \in \mathrm{Tm}(\Gamma, \sigma_i\{\langle \cdots \langle\langle !_\Gamma, M_1\rangle_{\sigma_1}, M_2\rangle_{\sigma_2}, \ldots, M_{i-1}\rangle_{\sigma_{i-1}}\}) \qquad (i = 1, \ldots, n).$$

Indeed $M_i = \mathrm{v}_i\{\mathrm{p}(\sigma_{i+1}) \cdots \mathrm{p}(\sigma_n) f\}$, *for* $i = 1, \ldots, n$. $\square$

To motivate the operation q below consider the following construction. Let $\mathrm{Fam}(\Gamma)$ be the class of families in context $\Gamma \in \mathcal{C}$

$$\{(\sigma, \tau) : \sigma \in \mathrm{Ty}(\Gamma), \tau \in \mathrm{Ty}(\Gamma.\sigma)\}$$

For $f : \Theta \longrightarrow \Gamma$ we wish to define a map that performs substitution in a family

$$\mathrm{Fam}(\Gamma) \longrightarrow \mathrm{Fam}(\Theta).$$

To do this we first define for $f : \Theta \longrightarrow \Gamma$ and $\sigma \in \mathrm{Ty}(\Gamma)$ a morphism $\mathrm{q}(f, \sigma) : \Theta.\sigma\{f\} \longrightarrow \Gamma.\sigma$ by

$$\mathrm{q}(f, \sigma) =_{\mathrm{def}} \langle f \circ \mathrm{p}(\sigma\{f\}), \mathrm{v}_{\sigma\{f\}}\rangle_\sigma.$$

Suppose $\tau \in \mathrm{Ty}(\Gamma.\sigma)$. Then $\tau\{\mathrm{q}(f, \sigma)\} \in \mathrm{Ty}(\Theta.\sigma\{f\})$. Hence $(\sigma\{f\}, \tau\{\mathrm{q}(f, \sigma)\}) \in \mathrm{Fam}(\Theta)$, whenever $(\sigma, \tau) \in \mathrm{Fam}(\Gamma)$. We write

$$\mathrm{Fam}(f)(\sigma, \tau) = (\sigma\{f\}, \tau\{\mathrm{q}(f, \sigma)\}). \tag{11.2}$$

Moreover, the following lemma will be used to prove that Fam is functorial.

Lemma 11.1.3. *(a)* $\mathrm{q}(1_\Gamma, \sigma) = 1_{\Gamma.\sigma}$

(b) $\mathrm{q}(f \circ g, \sigma) = \mathrm{q}(f, \sigma) \circ \mathrm{q}(g, \sigma\{f\})$

Proof. (a): $\mathrm{q}(1_\Gamma, \sigma) = \langle 1_\Gamma \circ \mathrm{p}(\sigma\{1_\Gamma\}), \mathrm{v}_{\sigma\{1_\Gamma\}}\rangle_\sigma = \langle \mathrm{p}(\sigma), \mathrm{v}_\sigma\rangle_\sigma = 1_{\Gamma.\sigma}$
(b):

$$\begin{aligned}
& \mathrm{q}(f, \sigma) \circ \mathrm{q}(g, \sigma\{f\}) \\
= \;& \langle f \circ \mathrm{p}(\sigma\{f\}), \mathrm{v}_{\sigma\{f\}}\rangle_\sigma \circ \langle g \circ \mathrm{p}(\sigma\{f\}\{g\}), \mathrm{v}_{\sigma\{f\}\{g\}}\rangle_{\sigma\{f\}} \\
= \;& \langle f \circ \mathrm{p}(\sigma\{f\}) \circ \langle g \circ \mathrm{p}(\sigma\{f\}\{g\}), \mathrm{v}_{\sigma\{f\}\{g\}}\rangle_{\sigma\{f\}}, \mathrm{v}_{\sigma\{f\}}\{\langle g \circ \mathrm{p}(\sigma\{f\}\{g\}), \mathrm{v}_{\sigma\{f\}\{g\}}\rangle_{\sigma\{f\}}\}\rangle_\sigma \\
= \;& \langle f \circ g \circ \mathrm{p}(\sigma\{f\}\{g\}), \mathrm{v}_{\sigma\{f\}\{g\}}\}\rangle_\sigma \\
= \;& \langle f \circ g \circ \mathrm{p}(\sigma\{f \circ g\}), \mathrm{v}_{\sigma\{f \circ g\}}\rangle_\sigma = \mathrm{q}(f \circ g, \sigma).
\end{aligned}$$

□

Corollary 11.1.4. *The operation* Fam *is functorial in the sense that*

(a) $\mathrm{Fam}(1_\Gamma) = \mathrm{id}_{\mathrm{Fam}(\Gamma)}$

(b) For $f : \Theta \longrightarrow \Gamma$ *and* $g : \Delta \longrightarrow \Theta$,

$$\mathrm{Fam}(f \circ g) = \mathrm{Fam}(g) \circ \mathrm{Fam}(f). \quad \square$$

For $(\sigma, \tau) \in \mathrm{Fam}(\Gamma)$, and $f : \Theta \longrightarrow \Gamma$ we write

$$(\sigma, \tau)\{f\} = \mathrm{Fam}(f)(\sigma, \tau).$$

Lemma 11.1.5. $\mathrm{q}(f, \sigma)\{\langle g, N\rangle\} = \langle f \circ g, N\rangle.$ □

11.2 Type constructions

We shall consider the type constructions Π, Σ, $+$, I, N_k and N below.

A CwF *supports* Π*-types* if for $\sigma \in \mathrm{Ty}(\Gamma)$ and $\tau \in \mathrm{Ty}(\Gamma.\sigma)$ there is a type $\Pi(\sigma, \tau) \in \mathrm{Ty}(\Gamma)$, and moreover for every $P \in \mathrm{Tm}(\Gamma.\sigma, \tau)$ there is an element $\lambda_{\sigma,\tau}(P) \in \mathrm{Tm}(\Gamma, \Pi(\sigma, \tau))$, and furthermore for any $M \in \mathrm{Tm}(\Gamma, \Pi(\sigma, \tau))$ and any $N \in \mathrm{Tm}(\Gamma, \sigma)$ this is an element $\mathrm{App}_{\sigma,\tau}(M, N) \in \mathrm{Tm}(\Gamma, \tau\{\langle 1_\Gamma, N\rangle_\sigma\})$, such that the following equations hold for any $f : \Theta \longrightarrow \Gamma$:

$$(\beta\text{-conv}) \quad \mathrm{App}_{\sigma,\tau}(\lambda_{\sigma,\tau}(P), N) = P\{\langle 1_\Gamma, N\rangle_\sigma\},$$

$$(\Pi\text{-subst}) \quad \Pi(\sigma, \tau)\{f\} = \Pi(\sigma\{f\}, \tau\{\mathsf{q}(f, \sigma)\}),$$

$$(\lambda\text{-subst}) \quad \lambda_{\sigma,\tau}(P)\{f\} = \lambda_{\sigma\{f\},\tau\{\mathsf{q}(f,\sigma)\}}(P\{\mathsf{q}(f, \sigma)\}),$$

$$(\mathrm{App}\text{-subst}) \quad \mathrm{App}_{\sigma,\tau}(M, N)\{f\} = \mathrm{App}_{\sigma\{f\},\tau\{\mathsf{q}(f,\sigma)\}}(M\{f\}, N\{f\}).$$

Remark: The substitution f acts as $\mathrm{Fam}(f)$ on the family σ, τ in the last three equations.

Omitting type information from terms (which is not computationally relevant) the equations above simply read:

$$\mathrm{App}(\lambda(P), N) = P\{\langle 1, N\rangle\},$$

$$\Pi(\sigma, \tau)\{f\} = \Pi(\sigma\{f\}, \tau\{\mathsf{q}(f)\}),$$

$$\lambda(P)\{f\} = \lambda(P\{\mathsf{q}(f)\}),$$

$$\mathrm{App}(M, N)\{f\} = \mathrm{App}(M\{f\}, N\{f\}).$$

A CwF *supports* Σ*-types* if for $\sigma \in \mathrm{Ty}(\Gamma)$ and $\tau \in \mathrm{Ty}(\Gamma.\sigma)$ there is a type $\Sigma(\sigma, \tau) \in \mathrm{Ty}(\Gamma)$, and for $M \in \mathrm{Tm}(\Gamma, \sigma)$ and $N \in \mathrm{Tm}(\Gamma, \tau\{\langle 1_\Gamma, M\rangle_\sigma\})$ there is an element $\mathrm{Pair}_{\sigma,\tau}(M, N) \in \mathrm{Tm}(\Gamma, \Sigma(\sigma, \tau))$. These constructions should satisfy for any $f : \Delta \longrightarrow \Gamma$

$$(\Sigma\text{-subst}) \quad \Sigma(\sigma, \tau)\{f\} = \Sigma(\sigma\{f\}, \tau\{\mathsf{q}(f, \sigma)\}),$$

$$(\mathrm{Pair}\text{-subst}) \quad \mathrm{Pair}_{\sigma,\tau}(M, N)\{f\} = \mathrm{Pair}_{\sigma\{f\},\tau\{\mathsf{q}(f,\sigma)\}}(M\{f\}, N\{f\})$$

Note that:

$$\begin{array}{rcl} N\{f\} & \in & \mathrm{Tm}(\Delta, \tau\{\langle 1_\Gamma, M\rangle_\sigma\}\{f\}) \\ & = & \mathrm{Tm}(\Delta, \tau\{\langle 1_\Gamma \circ f, M\{f\}\rangle_\sigma\}) \\ & = & \mathrm{Tm}(\Delta, \tau\{\langle f, M\{f\}\rangle_\sigma\}) \\ & = & \mathrm{Tm}(\Delta, \tau\{\langle f \circ \mathrm{p}(\sigma\{f\}), \mathrm{v}_{\sigma\{f\}}\rangle_\sigma\}\{\langle 1_\Delta, M\{f\}\rangle_{\sigma\{f\}}\} \\ & = & \mathrm{Tm}(\Delta, \tau\{\mathsf{q}(f, \sigma)\}\{\langle 1_\Delta, M\{f\}\rangle_{\sigma\{f\}}\}. \end{array}$$

So by (Σ-subst) both sides of (Pair-subst) have the same type. Next we need to specify the elimination operation E. First we construct a substitution

$$\mathrm{pair}_{\sigma,\tau} : \Gamma.\sigma.\tau \to \Gamma.\Sigma(\sigma,\tau)$$

by

$$\mathrm{pair}_{\sigma,\tau} =_{\mathrm{def}} \langle \mathrm{p}(\sigma)\mathrm{p}(\tau), \mathrm{Pair}_{\sigma\{\mathrm{p}(\sigma)\mathrm{p}(\tau)\},\tau\{\mathrm{q}(\mathrm{p}(\sigma)\mathrm{p}(\tau),\sigma)\}}(\mathrm{v}_\sigma\{\mathrm{p}(\tau)\}, \mathrm{v}_\tau)\rangle.$$

For any $\rho \in \mathrm{Ty}(\Gamma.\Sigma(\sigma,\tau))$, for any $P \in \mathrm{Tm}(\Gamma, \Sigma(\sigma,\tau))$ and for any $K \in \mathrm{Tm}(\Gamma.\sigma.\tau, \rho\{\mathrm{pair}_{\sigma,\tau}\})$ there is an element

$$\mathrm{E}_{\sigma,\tau,\rho}(P,K) \in \mathrm{Tm}(\Gamma, \rho\{\langle 1_\Gamma, P\rangle_{\Sigma(\sigma,\tau)}\})$$

such that

(Σ-conv) $\mathrm{E}_{\sigma,\tau,\rho}(\mathrm{Pair}_{\sigma,\tau}(M,N),K) = K\{\langle\langle 1_\Gamma, M\rangle_\sigma, N\rangle_\tau\}$.

Moreover for any $f : \Delta \to \Gamma$, we require

(E-subst) $\mathrm{E}_{\sigma,\tau,\rho}(P,K)\{f\} = \mathrm{E}_{\sigma\{f\},\tau\{\mathrm{q}(f,\sigma)\},\rho\{\mathrm{q}(f,\Sigma(\sigma,\tau))\}}(P\{f\}, K\{\mathrm{q}(\mathrm{q}(f,\sigma),\tau)\})$.

Again omitting type information from terms the equations above read:

$\Sigma(\sigma,\tau)\{f\} = \Sigma(\sigma\{f\}, \tau\{\mathsf{q}(f)\})$,

$\mathrm{Pair}(M,N)\{f\} = \mathrm{Pair}(M\{f\}, N\{f\})$

$\mathrm{E}(\mathrm{Pair}(M,N),K) = K\{\langle\langle 1, M\rangle, N\rangle\}$.

$\mathrm{E}(P,K)\{f\} = \mathrm{E}(P\{f\}, K\{\mathrm{q}(\mathrm{q}(f))\})$.

A CwF *supports +-types* if for $\sigma,\tau \in \mathrm{Ty}(\Gamma)$ there is a type $\sigma + \tau \in \mathrm{Ty}(\Gamma)$, and for each $M \in \mathrm{Tm}(\Gamma,\sigma)$ there is $\mathrm{inl}_{\sigma,\tau}(M) \in \mathrm{Tm}(\Gamma, \sigma+\tau)$, and for each $N \in \mathrm{Tm}(\Gamma,\tau)$ there is $\mathrm{inr}_{\sigma,\tau}(N) \in \mathrm{Tm}(\Gamma, \sigma+\tau)$. For all $f : \Delta \to \Gamma$, these construction should satisfy

(+-subst) $(\sigma+\tau)\{f\} = \sigma\{f\} + \tau\{f\}$

(inl-subst) $\mathrm{inl}_{\sigma,\tau}(M)\{f\} = \mathrm{inl}_{\sigma\{f\},\tau\{f\}}(M\{f\})$

(inr-subst) $\mathrm{inr}_{\sigma,\tau}(N)\{f\} = \mathrm{inr}_{\sigma\{f\},\tau\{f\}}(N\{f\})$

Furthermore for each $\rho \in \mathrm{Ty}(\Gamma.\sigma+\tau)$, each $P \in \mathrm{Tm}(\Gamma, \sigma+\tau)$ and each pair

$$K_1 \in \mathrm{Tm}(\Gamma.\sigma, \rho\{\langle \mathrm{p}(\sigma), \mathrm{inl}_{\sigma\{\mathrm{p}(\sigma)\},\tau\{\mathrm{p}(\sigma)\}}(\mathrm{v}_\sigma)\rangle_{\sigma+\tau}\})$$

and

$$K_2 \in \mathrm{Tm}(\Gamma.\tau, \rho\{\langle \mathrm{p}(\tau), \mathrm{inr}_{\sigma\{\mathrm{p}(\tau)\},\tau\{\mathrm{p}(\tau)\}}(\mathrm{v}_\tau)\rangle_{\sigma+\tau}\})$$

there is

$$\mathrm{D}_{\sigma,\tau,\rho}(P, K_1, K_2) \in \mathrm{Tm}(\Gamma, \rho\{\langle 1_\Gamma, P\rangle\})$$

such that

(+-conv1) $\mathrm{D}_{\sigma,\tau,\rho}(\mathrm{inl}_{\sigma,\tau}(M), K_1, K_2) = K_1\{\langle 1_\Gamma, M\rangle_\sigma\}$

(+-conv2) $\mathrm{D}_{\sigma,\tau,\rho}(\mathrm{inr}_{\sigma,\tau}(N), K_1, K_2) = K_2\{\langle 1_\Gamma, N\rangle_\tau\}$.

Moreover for any $f : \Delta \to \Gamma$, the construction D should satisfy

(D-subst) $\mathrm{D}_{\sigma,\tau,\rho}(P, K_1, K_2)\{f\} = \mathrm{D}_{\sigma\{f\},\tau\{f\},\rho\{\mathrm{q}(f,\sigma+\tau)\}}(P\{f\}, K_1\{\mathrm{q}(f,\sigma)\}, K_2\{\mathrm{q}(f,\tau)\})$

Without type information the terms, the equations above read

$(\sigma + \tau)\{f\} = \sigma\{f\} + \tau\{f\}$

$\mathrm{inl}(M)\{f\} = \mathrm{inl}(M\{f\})$

$\mathrm{inr}(N)\{f\} = \mathrm{inr}(N\{f\})$

$\mathrm{D}(\mathrm{inl}(M), K_1, K_2) = K_1\{\langle 1, M\rangle\}$

$\mathrm{D}(\mathrm{inr}(N), K_1, K_2) = K_2\{\langle 1, N\rangle\}$.

$\mathrm{D}(P, K_1, K_2)\{f\} = \mathrm{D}(P\{f\}, K_1\{\mathrm{q}(f)\}, K_2\{\mathrm{q}(f)\})$

A CwF *supports* I-*types* if for any $\sigma \in \mathrm{Ty}(\Gamma)$, and any $M, N \in \mathrm{Tm}(\Gamma, \sigma)$ there is a type $\mathrm{I}(\sigma, M, N) \in \mathrm{Ty}(\Gamma)$, and for any $M \in \mathrm{Tm}(\Gamma, \sigma)$, there is an element $\mathrm{r}_\sigma(M) \in \mathrm{I}(\sigma, M, M)$. These constructions should satisfy for each $f : \Delta \to \Gamma$

(I-subst) $\mathrm{I}(\sigma, M, N)\{f\} = \mathrm{I}(\sigma\{f\}, M\{f\}, N\{f\})$

(r-subst) $\mathrm{r}_\sigma(M)\{f\} = \mathrm{r}_{\sigma\{f\}}(M\{f\})$

For any

$$\rho \in \mathrm{Ty}(\Gamma.\sigma.\sigma\{\mathrm{p}(\sigma)\}.\mathrm{I}(\sigma\{\mathrm{p}(\sigma)\mathrm{p}(\sigma\{\mathrm{p}(\sigma)\})\}, \mathrm{v}_\sigma\{\mathrm{p}(\sigma\{\mathrm{p}(\sigma)\})\}, \mathrm{v}_{\sigma\{\mathrm{p}(\sigma)\}})),$$

any $P \in \mathrm{Tm}(\Gamma, \mathrm{I}(\sigma, M, N))$ and any $K \in \mathrm{Tm}(\Gamma.\sigma, \rho\{\langle\langle\langle \mathrm{p}(\sigma), \mathrm{v}_\sigma\rangle, \mathrm{v}_\sigma\rangle, \mathrm{r}_\sigma(\mathrm{v}_\sigma)\rangle\})$, there is

$$\mathrm{J}_{\sigma,\rho}(P, K) \in \mathrm{Tm}(\Gamma, \rho\{\langle\langle\langle 1_\Gamma, M\rangle, N\rangle, P\rangle\})$$

such that

(I-conv) $\mathrm{J}_{\sigma,\rho}(\mathrm{r}_\sigma(M), K) = K\{\langle 1_\Gamma, M\rangle_\sigma\}$

Moreover for any $f : \Delta \to \Gamma$,

(J-subst) $\mathrm{J}_{\sigma,\rho}(P, K)\{f\} = \mathrm{J}_{\sigma\{f\},\rho\{\mathrm{q}(\mathrm{q}(\mathrm{q}(f,\sigma),\sigma\{\mathrm{p}(\sigma)\}),\mathrm{I}(\sigma\{\mathrm{p}(\sigma)\mathrm{p}(\sigma\{\mathrm{p}(\sigma)\})\},\mathrm{v}_\sigma\{\mathrm{p}(\sigma\{\mathrm{p}(\sigma)\})\},\mathrm{v}_{\sigma\{\mathrm{p}(\sigma)\}}))\}}(P\{f\}, K\{\mathrm{q}(f,\sigma)\})$

Omitting the type information from the terms the equations above are follows

$\mathrm{I}(\sigma, M, N)\{f\} = \mathrm{I}(\sigma\{f\}, M\{f\}, N\{f\})$

$$\mathrm{r}(M)\{f\} = \mathrm{r}(M\{f\})$$

$$\mathrm{J}(\mathrm{r}(M), K) = K\{\langle 1, M\rangle\}$$

$$\mathrm{J}(P, K)\{f\} = \mathrm{J}(P\{f\}, K\{\mathrm{q}(f)\})$$

A CwF *supports* N_k*-types* if for every $\Gamma \in \mathcal{C}$ there is $\mathrm{N}_{k,\Gamma} \in \mathrm{Ty}(\Gamma)$ and k elements

$$0_{k,\Gamma}, \ldots, (k-1)_{k,\Gamma} \in \mathrm{Tm}(\Gamma, \mathrm{N}_{k,\Gamma})$$

such that for any $f : \Delta \to \Gamma$,

(N_k-subst) $\mathrm{N}_{k,\Gamma}\{f\} = \mathrm{N}_{k,\Delta}$

(i_k-subst) $i_{k,\Gamma}\{f\} = i_{k,\Delta}$.

Moreover for any $\rho \in \mathrm{Ty}(\Gamma.\mathrm{N}_{k,\Gamma})$ and any $M_i \in \mathrm{Tm}(\Gamma, \rho\{\langle 1_\Gamma, i_{k,\Gamma}\rangle\})$ $(i = 0, \ldots, k-1)$ and any $P \in \mathrm{Tm}(\Gamma, \mathrm{N}_{k,\Gamma})$, there is

$$\mathrm{R}_{k,\Gamma,\rho}(P, M_0, \ldots, M_{k-1}) \in \mathrm{Tm}(\Gamma, \rho\{\langle 1_\Gamma, P\rangle\}$$

which is such that

(N_k-conv) $\mathrm{R}_{k,\Gamma,\rho}(i_{k,\Gamma}, M_0, \ldots, M_{k-1}) = M_i$

and for $f : \Delta \to \Gamma$,

(N_k-subst) $\mathrm{R}_{k,\Gamma,\rho}(P, M_0, \ldots, M_{k-1})\{f\} = \mathrm{R}_{k,\Delta,\rho\{\mathrm{q}(f)\}}(P\{f\}, M_0\{f\}, \ldots, M_{k-1}\{f\})$.

Shedding the type information from terms we have the following equations

$$\mathrm{N}_k\{f\} = \mathrm{N}_k$$

$$i_k\{f\} = i_k$$

$$\mathrm{R}_k(i_k, M_0, \ldots, M_{k-1}) = M_i$$

$$\mathrm{R}_k(P, M_0, \ldots, M_{k-1})\{f\} = \mathrm{R}_k(P\{f\}, M_0\{f\}, \ldots, M_{k-1}\{f\}).$$

A CwF *supports natural numbers* if for every $\Gamma \in \mathcal{C}$ there is $\mathrm{N}_\Gamma \in \mathrm{Ty}(\Gamma)$ and $0_\Gamma \in \mathrm{Tm}(\Gamma, \mathrm{N}_\Gamma)$, and for any $N \in \mathrm{Tm}(\Gamma, \mathrm{N}_\Gamma)$, $\mathrm{S}_\Gamma(N) \in \mathrm{Tm}(\Gamma, \mathrm{N}_\Gamma)$, such that for any $f : \Delta \to \Gamma$,

(N-subst) $\mathrm{N}_\Gamma\{f\} = \mathrm{N}_\Delta$

(0-subst) $0_\Gamma\{f\} = 0_\Delta$.

(S-subst) $\mathrm{S}_\Gamma(N)\{f\} = \mathrm{S}_\Delta(N\{f\})$

Moreover for any $\rho \in \mathrm{Ty}(\Gamma.\mathrm{N}_\Gamma)$, $M \in \mathrm{Tm}(\Gamma, \rho\{\langle 1_\Gamma, 0_\Gamma\rangle\})$, $P \in \mathrm{Tm}(\Gamma, \mathrm{N}_\Gamma)$ and $K \in \mathrm{Tm}(\Gamma.\mathrm{N}_\Gamma.\rho, \rho\{\langle \mathrm{p}(\mathrm{N}_\Gamma) \circ \mathrm{p}(\rho), \mathrm{S}_{\Gamma.\mathrm{N}_\Gamma.\rho}(\mathrm{v}_{\mathrm{N}_\Gamma}\{\mathrm{p}(\rho)\})\rangle_{\mathrm{N}_\Gamma}\})$, there is

$$\mathrm{R}_{\Gamma,\rho}(P, M, K) \in \mathrm{Tm}(\Gamma, \rho\{\langle 1_\Gamma, P\rangle\})$$

which is such that

(N-conv1) $\mathrm{R}_{\Gamma,\rho}(0_\Gamma, M, K) = M$

(N-conv2) $\mathrm{R}_{\Gamma,\rho}(\mathrm{S}_\Gamma(P), M, K) = K\{\langle\langle 1_\Gamma, P\rangle_{\mathrm{N}_\Gamma}, \mathrm{R}_{\Gamma,\rho}(P, M, K)\rangle_\rho\}$

and for $f : \Delta \to \Gamma$,

(N-subst) $\mathrm{R}_{\Gamma,\rho}(P, M, K)\{f\} = \mathrm{R}_{\Delta,\rho\{\mathrm{q}(f,\mathrm{N}_\Gamma)\}}(P\{f\}, M\{f\}, K\{\mathrm{q}(\mathrm{q}(f, \mathrm{N}_\Gamma), \rho)\})$.

Without type information on the terms the above equations now read

$\mathrm{N}\{f\} = \mathrm{N}$

$0\{f\} = 0.$

$\mathrm{S}\{f\} = \mathrm{S}(N\{f\})$

$\mathrm{R}(0, M, K) = M$

$\mathrm{R}(\mathrm{S}(P), M, K) = K\{\langle\langle 1, P\rangle, \mathrm{R}(P, M, K)\rangle\}$

$\mathrm{R}(P, M, K)\{f\} = \mathrm{R}(P\{f\}, M\{f\}, K\{\mathrm{q}(\mathrm{q}(f))\})$.

11.3 A variable free formulation of type theory

From the semantic components we obtain eight judgement forms

$\Gamma \in \mathcal{C}$	Γ ctxt
$\Gamma = \Delta \in \mathcal{C}$	$\Gamma = \Delta$
$f \in \mathrm{Hom}_\mathcal{C}(\Delta, \Gamma)$	$f :: \Delta \longrightarrow \Gamma$
$f = g \in \mathrm{Hom}_\mathcal{C}(\Delta, \Gamma)$	$f = g :: \Delta \longrightarrow \Gamma$
$\sigma \in \mathrm{Ty}(\Gamma)$	$\Gamma \vdash \sigma$ set
$\sigma = \tau \in \mathrm{Ty}(\Gamma)$	$\Gamma \vdash \sigma = \tau$
$M \in \mathrm{Tm}(\Gamma, \sigma)$	$\Gamma \vdash M : \sigma$
$M = N \in \mathrm{Tm}(\Gamma, \sigma)$	$\Gamma \vdash M = N : \sigma$

The axioms for a CwF supporting the type constructions Π, Σ, +, I, N_k andN may be formulated in rule form, and gives then a variable-free formulation of type theory with explicit substitution. (A formulation of type theory with explicit substitution was given by Martin-Löf 1992).

$$\frac{\Gamma \text{ ctxt}}{1 :: \Gamma \longrightarrow \Gamma} \qquad \frac{f :: \Theta \longrightarrow \Delta \quad g :: \Delta \longrightarrow \Gamma}{g \circ f :: \Theta \longrightarrow \Gamma}$$

$$\frac{f :: \Theta \longrightarrow \Delta}{f \circ 1 = f :: \Theta \longrightarrow \Delta} \qquad \frac{f :: \Theta \longrightarrow \Delta}{1 \circ f = f :: \Theta \longrightarrow \Delta}$$

$$\frac{f :: \Theta \longrightarrow \Delta \quad g :: \Delta \longrightarrow \Gamma \quad h :: \Gamma \longrightarrow \Psi}{h \circ (g \circ f) = (h \circ g) \circ f :: \Theta \longrightarrow \Psi}$$

$$\frac{}{\top \text{ ctxt}} \qquad \frac{\Gamma \text{ ctxt}}{! :: \Gamma \longrightarrow \top} \qquad \frac{f :: \Gamma \longrightarrow \top}{f = ! :: \Gamma \longrightarrow \top}$$

$$\frac{\Gamma \text{ ctxt} \quad \Gamma \vdash \sigma \text{ set}}{\Gamma.\sigma \text{ ctxt}} \qquad \frac{\Gamma \text{ ctxt} \quad \Gamma \vdash \sigma \text{ set}}{\mathrm{p} :: \Gamma.\sigma \longrightarrow \Gamma}$$

$$\frac{\Delta \vdash \sigma \text{ set} \quad f :: \Gamma \longrightarrow \Delta}{\Gamma \vdash \sigma\{f\} \text{ set}}$$

$$\frac{\Gamma \vdash \sigma \text{ set}}{\Gamma \vdash \sigma\{1\} = \sigma} \qquad \frac{\Gamma \vdash \sigma \text{ set} \quad f :: \Theta \longrightarrow \Delta \quad g :: \Delta \longrightarrow \Gamma}{\Gamma \vdash \sigma\{g \circ f\} = \sigma\{g\}\{f\}}$$

$$\frac{\Gamma \vdash N : \sigma}{\Gamma \vdash N\{1\} = N : \sigma} \qquad \frac{\Gamma \vdash N : \sigma \quad f :: \Theta \longrightarrow \Delta \quad g :: \Delta \longrightarrow \Gamma}{\Gamma \vdash N\{g \circ f\} = N\{g\}\{f\} : \sigma\{g \circ f\}}$$

$$\frac{\Gamma \vdash \sigma \text{ set}}{\Gamma.\sigma \vdash \mathrm{v} : \sigma\{\mathrm{p}\}}$$

$$\frac{f :: \Gamma \longrightarrow \Delta \quad \Gamma \vdash \sigma \text{ set} \quad \Gamma \vdash M : \sigma\{f\}}{\langle f, M \rangle :: \Gamma \longrightarrow \Delta.\sigma}$$

$$\frac{f :: \Gamma \longrightarrow \Delta \quad \Gamma \vdash \sigma \text{ set} \quad \Gamma \vdash M : \sigma\{f\}}{\mathrm{p} \circ \langle f, M \rangle = f :: \Gamma \longrightarrow \Delta} \qquad \frac{f :: \Gamma \longrightarrow \Delta \quad \Gamma \vdash \sigma \text{ set} \quad \Gamma \vdash M : \sigma\{f\}}{\mathrm{v}\{\langle f, M \rangle\} = M : \sigma\{f\}}$$

$$\frac{}{\langle \mathrm{p}, \mathrm{v} \rangle = 1}$$

$$\frac{f :: \Gamma \longrightarrow \Delta \quad \Gamma \vdash \sigma \text{ set} \quad \Gamma \vdash M : \sigma\{f\} \quad g :: \Theta \longrightarrow \Gamma}{\langle f, M\rangle \circ g = \langle f \circ g, M\{g\}\rangle}$$

$$\frac{\Gamma \vdash \sigma \text{ set} \quad \Gamma.\sigma \vdash \tau \text{ set}}{\Gamma \vdash \Pi(\sigma,\tau) \text{ set}}$$

$$\frac{\Gamma.\sigma \vdash P : \tau}{\Gamma \vdash \lambda(P) : \Pi(\sigma,\tau)} \qquad \frac{\Gamma \vdash M : \Pi(\sigma,\tau) \quad \Gamma \vdash N : \sigma}{\Gamma \vdash \mathrm{App}(M,N) : \tau\{\langle 1, N\rangle\}}$$

$$\frac{\Gamma.\sigma \vdash P : \tau \quad \Gamma \vdash N : \sigma}{\Gamma \vdash \mathrm{App}(\lambda(P),N) = P\{\langle 1, N\rangle\} : \tau\{\langle 1, N\rangle\}}$$

$$\frac{\Gamma \vdash \sigma \text{ set} \quad \Gamma.\sigma \vdash \tau \text{ set} \quad f :: \Delta \longrightarrow \Gamma}{\Delta \vdash \Pi(\sigma,\tau)\{f\} = \Pi(\sigma\{f\}, \tau\{\langle f \circ \mathrm{p}, \mathrm{v}\rangle\})}$$

$$\frac{\Gamma.\sigma \vdash P : \tau \quad f :: \Delta \longrightarrow \Gamma}{\Delta \vdash \lambda(P)\{f\} = \lambda(P\{\langle f \circ \mathrm{p}, \mathrm{v}\rangle\}) : \Pi(\sigma\{f\}, \tau\{\langle f \circ \mathrm{p}, \mathrm{v}\rangle\})}$$

$$\frac{\Gamma \vdash M : \Pi(\sigma,\tau) \quad \Gamma \vdash N : \sigma \quad f :: \Delta \longrightarrow \Gamma}{\Delta \vdash \mathrm{App}(M,N)\{f\} = \mathrm{App}(M\{f\}, N\{f\}) : \tau\{\langle f, N\{f\}\rangle\}}$$

$$\frac{\Gamma \vdash \sigma \text{ set} \quad \Gamma.\sigma \vdash \tau \text{ set}}{\Gamma \vdash \Sigma(\sigma,\tau) \text{ set}}$$

$$\frac{\Gamma \vdash M : \sigma \quad \Gamma \vdash N : \tau\{\langle 1, M\rangle\}}{\Gamma \vdash \mathrm{Pair}(M,N) : \Sigma(\sigma,\tau)}$$

$$\frac{\Gamma.\Sigma(\sigma,\tau) \vdash \rho \text{ set} \quad \Gamma \vdash P : \Sigma(\sigma,\tau) \quad \Gamma.\sigma.\tau \vdash K : \rho\{\langle \mathrm{p} \circ \mathrm{p}, \mathrm{Pair}(\mathrm{v}\{\mathrm{p}\}, \mathrm{v})\rangle\}}{\Gamma \vdash \mathrm{E}(P,K) : \rho\{\langle 1, P\rangle\}}$$

$$\frac{\Gamma.\Sigma(\sigma,\tau) \vdash \rho \text{ set} \quad \Gamma \vdash M : \sigma \quad \Gamma \vdash N : \tau\{\langle 1, M\rangle\} \quad \Gamma.\sigma.\tau \vdash K : \rho\{\langle \mathrm{p} \circ \mathrm{p}, \mathrm{Pair}(\mathrm{v}\{\mathrm{p}\}, \mathrm{v})\rangle\}}{\Gamma \vdash \mathrm{E}(\mathrm{Pair}(M,N),K) = K\{\langle\langle 1, M\rangle, N\rangle\} : \rho\{\langle 1, \mathrm{Pair}(M,N)\rangle\}}$$

$$\frac{\Gamma \vdash \sigma \text{ set} \quad \Gamma.\sigma \vdash \tau \text{ set} \quad f :: \Delta \longrightarrow \Gamma}{\Delta \vdash \Sigma(\sigma,\tau)\{f\} = \Sigma(\sigma\{f\}, \tau\{\langle f \circ \mathrm{p}, \mathrm{v}\rangle\})}$$

$$\frac{\Gamma \vdash M : \sigma \quad \Gamma \vdash N : \tau\{\langle 1, M\rangle\} \quad f :: \Delta \longrightarrow \Gamma}{\Delta \vdash \mathrm{Pair}(M, N)\{f\} = \mathrm{Pair}(M\{f\}, N\{f\}) : \Sigma(\sigma, \tau)\{f\}}$$

$$\frac{\Gamma.\Sigma(\sigma, \tau) \vdash \rho \text{ set} \quad \Gamma \vdash P : \Sigma(\sigma, \tau) \quad \Gamma.\sigma.\tau \vdash K : \rho\{\langle \mathrm{p} \circ \mathrm{p}, \mathrm{Pair}(\mathrm{v}\{\mathrm{p}\}, \mathrm{v})\rangle\} \quad f :: \Delta \longrightarrow \Gamma}{\Delta \vdash \mathrm{E}(P, K)\{f\} = \mathrm{E}(P\{f\}, K\{\langle\langle f \circ \mathrm{p} \circ \mathrm{p}, \mathrm{v}\{\mathrm{p}\}\rangle, \mathrm{v}\rangle\}) : \rho\{\langle f, P\{f\}\rangle\}}$$

$$\frac{\Gamma \vdash \sigma \text{ set} \quad \Gamma \vdash \tau \text{ set}}{\Gamma \vdash \sigma + \tau \text{ set}}$$

$$\frac{\Gamma \vdash M : \sigma \quad \Gamma \vdash \tau \text{ set}}{\Gamma \vdash \mathrm{inl}(M) : \sigma + \tau} \qquad \frac{\Gamma \vdash \sigma \text{ set} \quad \Gamma \vdash N : \tau}{\Gamma \vdash \mathrm{inr}(N) : \sigma + \tau}$$

$$\frac{\Gamma.\sigma + \tau \vdash \rho \quad \Gamma \vdash P : \sigma + \tau \quad \Gamma.\sigma \vdash K_1 : \rho\{\langle \mathrm{p}, \mathrm{inl}(\mathrm{v})\rangle\} \quad \Gamma.\tau \vdash K_2 : \rho\{\langle \mathrm{p}, \mathrm{inr}(\mathrm{v})\rangle\}}{\Gamma \vdash \mathrm{D}(P, K_1, K_2) : \rho\{\langle 1, P\rangle\}}$$

$$\frac{\Gamma.\sigma + \tau \vdash \rho \quad \Gamma \vdash M : \sigma \quad \Gamma.\sigma \vdash K_1 : \rho\{\langle \mathrm{p}, \mathrm{inl}(\mathrm{v})\rangle\} \quad \Gamma.\tau \vdash K_2 : \rho\{\langle \mathrm{p}, \mathrm{inr}(\mathrm{v})\rangle\}}{\Gamma \vdash \mathrm{D}(\mathrm{inl}(M), K_1, K_2) = K_1\{\langle 1, M\rangle\} : \rho\{\langle 1, \mathrm{inl}(M)\rangle\}}$$

$$\frac{\Gamma.\sigma + \tau \vdash \rho \quad \Gamma \vdash N : \tau \quad \Gamma.\sigma \vdash K_1 : \rho\{\langle \mathrm{p}, \mathrm{inl}(\mathrm{v})\rangle\} \quad \Gamma.\tau \vdash K_2 : \rho\{\langle \mathrm{p}, \mathrm{inr}(\mathrm{v})\rangle\}}{\Gamma \vdash \mathrm{D}(\mathrm{inl}(N), K_1, K_2) = K_2\{\langle 1, N\rangle\} : \rho\{\langle 1, \mathrm{inr}(N)\rangle\})}$$

$$\frac{\Gamma \vdash \sigma \text{ set} \quad \Gamma \vdash \tau \text{ set} \quad f :: \Delta \longrightarrow \Gamma}{\Gamma \vdash (\sigma + \tau)\{f\} = \sigma\{f\} + \tau\{f\}}$$

$$\frac{\Gamma \vdash M : \sigma \quad \Gamma \vdash \tau \text{ set} \quad f :: \Delta \longrightarrow \Gamma}{\Gamma \vdash \mathrm{inl}(M)\{f\} = \mathrm{inl}(M\{f\}) : \sigma + \tau} \qquad \frac{\Gamma \vdash \sigma \text{ set} \quad \Gamma \vdash N : \tau \quad f :: \Delta \longrightarrow \Gamma}{\Gamma \vdash \mathrm{inr}(N)\{f\} = \mathrm{inr}(N\{f\}) : \sigma + \tau}$$

$$\frac{\Gamma.\sigma + \tau \vdash \rho \quad \Gamma \vdash P : \sigma + \tau \quad \Gamma.\sigma \vdash K_1 : \rho\{\langle \mathrm{p}, \mathrm{inl}(\mathrm{v})\rangle\} \quad \Gamma.\tau \vdash K_2 : \rho\{\langle \mathrm{p}, \mathrm{inr}(\mathrm{v})\rangle\} \quad f :: \Delta \longrightarrow \Gamma}{\Gamma \vdash \mathrm{D}(P, K_1, K_2)\{f\} = \mathrm{D}(P\{f\}, K_1\{\langle f \circ \mathrm{p}, \mathrm{v}\rangle\}, K_2\{\langle f \circ \mathrm{p}, \mathrm{v}\rangle\}) : \rho\{\langle f, P\{f\}\rangle\}}$$

$$\frac{\Gamma \vdash \sigma \text{ set} \quad \Gamma \vdash M : \sigma \quad \Gamma \vdash N : \sigma}{\Gamma \vdash \mathrm{I}(\sigma, M, N) \text{ set}}$$

$$\frac{\Gamma \vdash M : \sigma}{\Gamma \vdash \mathrm{r}(M) : \mathrm{I}(\sigma, M, M)}$$

$$\frac{\Gamma.\sigma.\sigma\{\mathrm{p}\}.\mathrm{I}(\sigma\{\mathrm{p} \circ \mathrm{p}\}, \mathrm{v}\{\mathrm{p}\}, \mathrm{v}) \vdash \rho \text{ set} \quad \Gamma \vdash P : \mathrm{I}(\sigma, M, N) \quad \Gamma.\sigma \vdash K : \rho\{\langle\langle\langle \mathrm{p}, \mathrm{v}\rangle, \mathrm{v}\rangle, \mathrm{r}(\mathrm{v})\rangle\}}{\Gamma \vdash \mathrm{J}(P, K) : \rho\{\langle\langle\langle 1, M\rangle, N\rangle, P\rangle\}}$$

$$\frac{\Gamma \vdash \sigma \text{ set} \quad \Gamma \vdash M : \sigma \quad \Gamma \vdash N : \sigma \quad f :: \Delta \longrightarrow \Gamma}{\Delta \vdash \mathrm{I}(\sigma, M, N)\{f\} = \mathrm{I}(\sigma\{f\}, M\{f\}, N\{f\})}$$

$$\frac{\Gamma \vdash M : \sigma \quad f :: \Delta \longrightarrow \Gamma}{\Delta \vdash \mathrm{r}(M)\{f\} = \mathrm{r}(M\{f\}) : \mathrm{I}(\sigma\{f\}, M\{f\}, M\{f\})}$$

$$\frac{\Gamma.\sigma.\sigma\{\mathrm{p}\}.\mathrm{I}(\sigma\{\mathrm{pp}\}, \mathrm{v}\{\mathrm{p}\}, \mathrm{v}) \vdash \rho \text{ set} \quad \Gamma \vdash P : \mathrm{I}(\sigma, M, N) \quad \Gamma.\sigma \vdash K : \rho\{\langle\langle\langle \mathrm{p}, \mathrm{v}\rangle, \mathrm{v}\rangle, \mathrm{r}(\mathrm{v})\rangle\} \quad f :: \Delta \longrightarrow \Gamma}{\Delta \vdash \mathrm{J}(P, K)\{f\} = \mathrm{J}(P\{f\}, K\{\mathrm{q}(f)\}) : \rho\{\langle\langle\langle f, M\{f\}\rangle, N\{f\}\rangle, P\{f\}\rangle\}}$$

$$\frac{\Gamma \text{ ctxt}}{\Gamma \vdash \mathrm{N}_k \text{ set}} \qquad (k = 0, 1, 2, \ldots)$$

$$\frac{}{\Gamma \vdash i_k : \mathrm{N}_k} \qquad (i = 0, \ldots, k-1)$$

$$\frac{\Gamma.\mathrm{N}_k \vdash \rho \text{ set} \quad \Gamma \vdash P : \mathrm{N}_k \quad \Gamma \vdash M_0 : \rho\{\langle 1, 0_k\rangle\} \quad \cdots \quad \Gamma \vdash M_{k-1} : \rho\{\langle 1, (k-1)_k\rangle\}}{\Gamma \vdash \mathrm{R}_k(P, M_0, \ldots, M_{k-1}) : \rho\{\langle 1, P\rangle\}}$$

$$\frac{\Gamma.\mathrm{N}_k \vdash \rho \text{ set} \quad \Gamma \vdash M_0 : \rho\{\langle 1, 0_k\rangle\} \quad \cdots \quad \Gamma \vdash M_{k-1} : \rho\{\langle 1, (k-1)_k\rangle\}}{\Gamma \vdash \mathrm{R}_k(i_k, M_0, \ldots, M_{k-1}) = M_i : \rho\{\langle 1, i_k\rangle\}}$$

$$\frac{f :: \Delta \to \Gamma}{\Delta \vdash \mathrm{N}_k\{f\} = \mathrm{N}_k}$$

$$\frac{f :: \Delta \to \Gamma}{\Delta \vdash i_k\{f\} = i_k : \mathrm{N}_k}$$

$$\frac{\Gamma.\mathrm{N}_k \vdash \rho \text{ set} \quad \Gamma \vdash P : \mathrm{N}_k \quad \Gamma \vdash M_0 : \rho\{\langle 1, 0_k\rangle \quad \cdots \quad \Gamma \vdash M_{k-1} : \rho\{\langle 1, (k-1)_k\rangle \quad f :: \Delta \to \Gamma}{\Delta \vdash \mathrm{R}(P, M_0, \ldots, M_{k-1})\{f\} = \mathrm{R}(P\{f\}, M_0\{f\}, \ldots, M_{k-1}\{f\}) : \rho\{\langle f, P\{f\}\rangle}$$

$$\frac{\Gamma \text{ ctxt}}{\Gamma \vdash \mathrm{N} \text{ set}}$$

$$\frac{\Gamma \text{ ctxt}}{\Gamma \vdash 0 : \mathrm{N} \text{ set}}$$

$$\frac{\Gamma \vdash M : \mathrm{N}}{\Gamma \vdash \mathrm{S}(M) : \mathrm{N}}$$

$$\frac{\Gamma.\mathrm{N} \vdash \rho \text{ set} \quad \Gamma \vdash P : \mathrm{N} \quad \Gamma \vdash M : \rho\{\langle 1, 0\rangle\} \quad \Gamma.\mathrm{N}.\rho \vdash K : \rho\{\langle \mathrm{p} \circ \mathrm{p}, \mathrm{S}(\mathrm{v}\{\mathrm{p}\})\rangle\}}{\Gamma \vdash \mathrm{R}(P, M, K) : \rho\{\langle 1, P\rangle\}}$$

$$\frac{\Gamma.\mathrm{N} \vdash \rho \text{ set} \quad \Gamma \vdash M : \rho\{\langle 1, 0\rangle\} \quad \Gamma.\mathrm{N}.\rho \vdash K : \rho\{\langle \mathrm{p} \circ \mathrm{p}, \mathrm{S}(\mathrm{v}\{\mathrm{p}\})\rangle\}}{\Gamma \vdash \mathrm{R}(0, M, K) = M : \rho\{\langle 1, 0\rangle\}}$$

$$\frac{\Gamma.\mathrm{N} \vdash \rho \text{ set} \quad \Gamma \vdash P : \mathrm{N} \quad \Gamma \vdash M : \rho\{\langle 1, 0\rangle\} \quad \Gamma.\mathrm{N}.\rho \vdash K : \rho\{\langle \mathrm{p} \circ \mathrm{p}, \mathrm{S}(\mathrm{v}\{\mathrm{p}\})\rangle\}}{\Gamma \vdash \mathrm{R}(\mathrm{S}(P), M, K) = K\{\langle\langle 1, P\rangle, \mathrm{R}(P, M, K)\rangle\} : \rho\{\langle 1, \mathrm{S}(P)\rangle\}}$$

$$\frac{f :: \Delta \to \Gamma}{\Delta \vdash \mathrm{N}\{f\} = \mathrm{N}}$$

$$\frac{f :: \Delta \to \Gamma}{\Delta \vdash 0\{f\} = 0 : \mathrm{N}}$$

$$\frac{\Gamma \vdash M : \mathrm{N} \quad f :: \Delta \to \Gamma}{\Delta \vdash \mathrm{S}(M)\{f\} = \mathrm{S}(M\{f\}) : \mathrm{N}}$$

$$\frac{\Gamma.\mathrm{N} \vdash \rho \text{ set} \quad \Gamma \vdash P : \mathrm{N} \quad \Gamma \vdash M : \rho\{\langle 1, 0\rangle\} \quad \Gamma.\mathrm{N}.\rho \vdash K : \rho\{\langle \mathrm{p} \circ \mathrm{p}, \mathrm{S}(\mathrm{v}\{\mathrm{p}\})\rangle\} \quad f :: \Delta \to \Gamma}{\Gamma \vdash \mathrm{R}(P, M, K)\{f\} = \mathrm{R}(P\{f\}, M\{f\}, K\{\mathrm{q}(\mathrm{q}(f))\}) : \rho\{\langle f, P\{f\}\rangle\}}$$

11.3.1 Another variant

The syntax may be further simplified and brought closer to that of (Martin-Löf 1992). See also (Coquand et. al. 2007). Write

fg for $f \circ g$, fgh for $(f \circ g) \circ h$ etc.

σf for $\sigma\{f\}$, σfg for $\sigma\{f\}\{g\}$ etc.

Nf for $N\{f\}$, Nfg for $N\{f\}\{g\}$ etc.

ϵ for !

$$\frac{\Gamma\ \mathrm{ctxt}}{1 :: \Gamma \longrightarrow \Gamma} \qquad \frac{f :: \Theta \longrightarrow \Delta \quad g :: \Delta \longrightarrow \Gamma}{g \circ f :: \Theta \longrightarrow \Gamma}$$

$$\frac{f :: \Theta \longrightarrow \Delta}{f1 = f :: \Theta \longrightarrow \Delta} \qquad \frac{f :: \Theta \longrightarrow \Delta}{1f = f :: \Theta \longrightarrow \Delta}$$

$$\frac{f :: \Theta \longrightarrow \Delta \quad g :: \Delta \longrightarrow \Gamma \quad h :: \Gamma \longrightarrow \Psi}{h(gf) = (hg)f :: \Theta \longrightarrow \Psi}$$

$$\frac{}{\top\ \mathrm{ctxt}} \qquad \frac{\Gamma\ \mathrm{ctxt}}{\epsilon :: \Gamma \longrightarrow \top} \qquad \frac{f :: \Gamma \longrightarrow \top}{f = \epsilon :: \Gamma \longrightarrow \top}$$

$$\frac{\Gamma\ \mathrm{ctxt} \quad \Gamma \vdash \sigma\ \mathrm{set}}{\Gamma.\sigma\ \mathrm{ctxt}} \qquad \frac{\Gamma\ \mathrm{ctxt} \quad \Gamma \vdash \sigma\ \mathrm{set}}{\mathrm{p} :: \Gamma.\sigma \longrightarrow \Gamma}$$

$$\frac{\Delta \vdash \sigma\ \mathrm{set} \quad f :: \Gamma \longrightarrow \Delta}{\Gamma \vdash \sigma f\ \mathrm{set}}$$

$$\frac{\Gamma \vdash \sigma\ \mathrm{set}}{\Gamma \vdash \sigma 1 = \sigma} \qquad \frac{\Gamma \vdash \sigma\ \mathrm{set} \quad f :: \Theta \longrightarrow \Delta \quad g :: \Delta \longrightarrow \Gamma}{\Gamma \vdash \sigma(gf) = \sigma gf}$$

$$\frac{\Gamma \vdash N : \sigma}{\Gamma \vdash N1 = N : \sigma} \qquad \frac{\Gamma \vdash N : \sigma \quad f :: \Theta \longrightarrow \Delta \quad g :: \Delta \longrightarrow \Gamma}{\Gamma \vdash N(gf) = Ngf : \sigma(gf)}$$

$$\frac{\Gamma \vdash \sigma\ \mathrm{set}}{\Gamma.\sigma \vdash \mathrm{v} : \sigma\mathrm{p}}$$

$$\frac{f :: \Gamma \longrightarrow \Delta \quad \Gamma \vdash \sigma\ \mathrm{set} \quad \Gamma \vdash M : \sigma f}{\langle f, M\rangle :: \Gamma \longrightarrow \Delta.\sigma}$$

$$\frac{f :: \Gamma \longrightarrow \Delta \quad \Gamma \vdash \sigma\ \mathrm{set} \quad \Gamma \vdash M : \sigma f}{\mathrm{p}\langle f, M\rangle = f :: \Gamma \longrightarrow \Delta} \qquad \frac{f :: \Gamma \longrightarrow \Delta \quad \Gamma \vdash \sigma\ \mathrm{set} \quad \Gamma \vdash M : \sigma f}{\mathrm{v}\langle f, M\rangle = M : \sigma f}$$

$$\frac{}{\langle \mathrm{p}, \mathrm{v}\rangle = 1}$$

$$\frac{f :: \Gamma \longrightarrow \Delta \quad \Gamma \vdash \sigma \text{ set} \quad \Gamma \vdash M : \sigma f \quad g :: \Theta \longrightarrow \Gamma}{\langle f, M\rangle g = \langle fg, Mg\rangle}$$

$$\frac{\Gamma \vdash \sigma \text{ set} \quad \Gamma.\sigma \vdash \tau \text{ set}}{\Gamma \vdash \Pi(\sigma, \tau) \text{ set}}$$

$$\frac{\Gamma.\sigma \vdash P : \tau}{\Gamma \vdash \lambda(P) : \Pi(\sigma, \tau)} \qquad \frac{\Gamma \vdash M : \Pi(\sigma, \tau) \quad \Gamma \vdash N : \sigma}{\Gamma \vdash \mathrm{App}(M, N) : \tau\langle 1, N\rangle}$$

$$\frac{\Gamma.\sigma \vdash P : \tau \quad \Gamma \vdash N : \sigma}{\Gamma \vdash \mathrm{App}(\lambda(P), N) = P\langle 1, N\rangle : \tau\langle 1, N\rangle}$$

$$\frac{\Gamma \vdash \sigma \text{ set} \quad \Gamma.\sigma \vdash \tau \text{ set} \quad f :: \Delta \longrightarrow \Gamma}{\Delta \vdash \Pi(\sigma, \tau)f = \Pi(\sigma f, \tau\langle f\mathrm{p}, \mathrm{v}\rangle)}$$

$$\frac{\Gamma.\sigma \vdash P : \tau \quad f :: \Delta \longrightarrow \Gamma}{\Delta \vdash \lambda(P)f = \lambda(P\langle f\mathrm{p}, \mathrm{v}\rangle) : \Pi(\sigma f, \tau\langle f\mathrm{p}, \mathrm{v}\rangle)}$$

$$\frac{\Gamma \vdash M : \Pi(\sigma, \tau) \quad \Gamma \vdash N : \sigma \quad f :: \Delta \longrightarrow \Gamma}{\Delta \vdash \mathrm{App}(M, N)f = \mathrm{App}(Mf, Nf) : \tau\langle f, Nf\rangle}$$

$$\frac{\Gamma \vdash \sigma \text{ set} \quad \Gamma.\sigma \vdash \tau \text{ set}}{\Gamma \vdash \Sigma(\sigma, \tau) \text{ set}}$$

$$\frac{\Gamma \vdash M : \sigma \quad \Gamma \vdash N : \tau\langle 1, M\rangle}{\Gamma \vdash \mathrm{Pair}(M, N) : \Sigma(\sigma, \tau)}$$

$$\frac{\Gamma.\Sigma(\sigma, \tau) \vdash \rho \text{ set} \quad \Gamma \vdash P : \Sigma(\sigma, \tau) \quad \Gamma.\sigma.\tau \vdash K : \rho\langle \mathrm{pp}, \mathrm{Pair}(\mathrm{vp}, \mathrm{v})\rangle}{\Gamma \vdash \mathrm{E}(P, K) : \rho\langle 1, P\rangle}$$

$$\frac{\Gamma.\Sigma(\sigma, \tau) \vdash \rho \text{ set} \quad \Gamma \vdash M : \sigma \quad \Gamma \vdash N : \tau\langle 1, M\rangle \quad \Gamma.\sigma.\tau \vdash K : \rho\langle \mathrm{pp}, \mathrm{Pair}(\mathrm{vp}, \mathrm{v})\rangle}{\Gamma \vdash \mathrm{E}(\mathrm{Pair}(M, N), K) = K\langle\langle 1, M\rangle, N\rangle : \rho\langle 1, \mathrm{Pair}(M, N)\rangle}$$

$$\frac{\Gamma \vdash \sigma \text{ set} \quad \Gamma.\sigma \vdash \tau \text{ set} \quad f :: \Delta \longrightarrow \Gamma}{\Delta \vdash \Sigma(\sigma, \tau)f = \Sigma(\sigma f, \tau\langle f\mathrm{p}, \mathrm{v}\rangle)}$$

$$\frac{\Gamma \vdash M : \sigma \quad \Gamma \vdash N : \tau\langle 1, M\rangle \quad f :: \Delta \longrightarrow \Gamma}{\Delta \vdash \mathrm{Pair}(M, N)f = \mathrm{Pair}(Mf, Nf) : \Sigma(\sigma, \tau)f}$$

$$\frac{\Gamma.\Sigma(\sigma,\tau) \vdash \rho \text{ set} \quad \Gamma \vdash P : \Sigma(\sigma,\tau) \quad \Gamma.\sigma.\tau \vdash K : \rho\langle \text{pp}, \text{Pair}(\text{vp}, \text{v})\rangle \quad f :: \Delta \longrightarrow \Gamma}{\Delta \vdash \text{E}(P,K)f = \text{E}(Pf, K\langle\langle f\text{pp}, \text{vp}\rangle, \text{v}\rangle) : \rho\langle f, Pf\rangle}$$

$$\frac{\Gamma \vdash \sigma \text{ set} \quad \Gamma \vdash \tau \text{ set}}{\Gamma \vdash \sigma + \tau \text{ set}}$$

$$\frac{\Gamma \vdash M : \sigma \quad \Gamma \vdash \tau \text{ set}}{\Gamma \vdash \text{inl}(M) : \sigma + \tau} \qquad \frac{\Gamma \vdash \sigma \text{ set} \quad \Gamma \vdash N : \tau}{\Gamma \vdash \text{inr}(N) : \sigma + \tau}$$

$$\frac{\Gamma.\sigma + \tau \vdash \rho \quad \Gamma \vdash P : \sigma + \tau \quad \Gamma.\sigma \vdash K_1 : \rho\langle \text{p}, \text{inl}(\text{v})\rangle \quad \Gamma.\tau \vdash K_2 : \rho\langle \text{p}, \text{inr}(\text{v})\rangle}{\Gamma \vdash \text{D}(P, K_1, K_2) : \rho\langle 1, P\rangle}$$

$$\frac{\Gamma.\sigma + \tau \vdash \rho \quad \Gamma \vdash M : \sigma \quad \Gamma.\sigma \vdash K_1 : \rho\langle \text{p}, \text{inl}(\text{v})\rangle \quad \Gamma.\tau \vdash K_2 : \rho\langle \text{p}, \text{inr}(\text{v})\rangle}{\Gamma \vdash \text{D}(\text{inl}(M), K_1, K_2) = K_1\langle 1, M\rangle : \rho\langle 1, \text{inl}(M)\rangle}$$

$$\frac{\Gamma.\sigma + \tau \vdash \rho \quad \Gamma \vdash N : \tau \quad \Gamma.\sigma \vdash K_1 : \rho\langle \text{p}, \text{inl}(\text{v})\rangle \quad \Gamma.\tau \vdash K_2 : \rho\langle \text{p}, \text{inr}(\text{v})\rangle}{\Gamma \vdash \text{D}(\text{inl}(N), K_1, K_2) = K_2\langle 1, N\rangle : \rho\langle 1, \text{inr}(N)\rangle}$$

$$\frac{\Gamma \vdash \sigma \text{ set} \quad \Gamma \vdash \tau \text{ set} \quad f :: \Delta \longrightarrow \Gamma}{\Gamma \vdash (\sigma + \tau)f = \sigma f + \tau f}$$

$$\frac{\Gamma \vdash M : \sigma \quad \Gamma \vdash \tau \text{ set} \quad f :: \Delta \longrightarrow \Gamma}{\Gamma \vdash \text{inl}(M)f = \text{inl}(Mf) : \sigma + \tau} \qquad \frac{\Gamma \vdash \sigma \text{ set} \quad \Gamma \vdash N : \tau \quad f :: \Delta \longrightarrow \Gamma}{\Gamma \vdash \text{inr}(N)f = \text{inr}(Nf) : \sigma + \tau}$$

$$\frac{\Gamma.\sigma + \tau \vdash \rho \quad \Gamma \vdash P : \sigma + \tau \quad \Gamma.\sigma \vdash K_1 : \rho\langle \text{p}, \text{inl}(\text{v})\rangle \quad \Gamma.\tau \vdash K_2 : \rho\langle \text{p}, \text{inr}(\text{v})\rangle \quad f :: \Delta \longrightarrow \Gamma}{\Gamma \vdash \text{D}(P, K_1, K_2)f = \text{D}(Pf, K_1\langle f\text{p}, \text{v}\rangle, K_2\langle f\text{p}, \text{v}\rangle) : \rho\langle f, Pf\rangle}$$

$$\frac{\Gamma \vdash \sigma \text{ set} \quad \Gamma \vdash M : \sigma \quad \Gamma \vdash N : \sigma}{\Gamma \vdash \text{I}(\sigma, M, N) \text{ set}}$$

$$\frac{\Gamma \vdash M : \sigma}{\Gamma \vdash \text{r}(M) : \text{I}(\sigma, M, M)}$$

$$\frac{\Gamma.\sigma.\sigma\mathrm{p}.\mathrm{I}(\sigma\mathrm{pp},\mathrm{vp},\mathrm{v}) \vdash \rho \text{ set} \quad \Gamma \vdash P : \mathrm{I}(\sigma, M, N) \quad \Gamma.\sigma \vdash K : \rho\langle\langle\langle \mathrm{p}, \mathrm{v}\rangle, \mathrm{v}\rangle, \mathrm{r}(\mathrm{v})\rangle}{\Gamma \vdash \mathrm{J}(P, K) : \rho\langle\langle\langle 1, M\rangle, N\rangle, P\rangle}$$

$$\frac{\Gamma \vdash \sigma \text{ set} \quad \Gamma \vdash M : \sigma \quad \Gamma \vdash N : \sigma \quad f :: \Delta \longrightarrow \Gamma}{\Delta \vdash \mathrm{I}(\sigma, M, N)f = \mathrm{I}(\sigma f, Mf, Nf)}$$

$$\frac{\Gamma \vdash M : \sigma \quad f :: \Delta \longrightarrow \Gamma}{\Delta \vdash \mathrm{r}(M)f = \mathrm{r}(Mf) : \mathrm{I}(\sigma f, Mf, Mf)}$$

$$\frac{\Gamma.\sigma.\sigma\mathrm{p}.\mathrm{I}(\sigma\mathrm{pp},\mathrm{vp},\mathrm{v}) \vdash \rho \text{ set} \quad \Gamma \vdash P : \mathrm{I}(\sigma, M, N) \quad \Gamma.\sigma \vdash K : \rho\langle\langle\langle \mathrm{p}, \mathrm{v}\rangle, \mathrm{v}\rangle, \mathrm{r}(\mathrm{v})\rangle \quad f :: \Delta \longrightarrow \Gamma}{\Delta \vdash \mathrm{J}(P, K)f = \mathrm{J}(Pf, K\langle f\mathrm{p}, \mathrm{v}\rangle) : \rho\langle\langle\langle f, Mf\rangle, Nf\rangle, Pf\rangle}$$

$$\frac{\Gamma \text{ ctxt}}{\Gamma \vdash \mathrm{N}_k \text{ set}} \qquad (k = 0, 1, 2, \ldots)$$

$$\frac{}{\Gamma \vdash i_k : \mathrm{N}_k} \qquad (i = 0, \ldots, k-1)$$

$$\frac{\Gamma.\mathrm{N}_k \vdash \rho \text{ set} \quad \Gamma \vdash P : \mathrm{N}_k \quad \Gamma \vdash M_0 : \rho\langle 1, 0_k\rangle \quad \cdots \quad \Gamma \vdash M_{k-1} : \rho\langle 1, (k-1)_k\rangle}{\Gamma \vdash \mathrm{R}_k(P, M_0, \ldots, M_{k-1}) : \rho\langle 1, P\rangle}$$

$$\frac{\Gamma.\mathrm{N}_k \vdash \rho \text{ set} \quad \Gamma \vdash M_0 : \rho\langle 1, 0_k\rangle \quad \cdots \quad \Gamma \vdash M_{k-1} : \rho\langle 1, (k-1)_k\rangle}{\Gamma \vdash \mathrm{R}_k(i_k, M_0, \ldots, M_{k-1}) = M_i : \rho\langle 1, i_k\rangle}$$

$$\frac{f :: \Delta \to \Gamma}{\Delta \vdash \mathrm{N}_k f = \mathrm{N}_k}$$

$$\frac{f :: \Delta \to \Gamma}{\Delta \vdash i_k f = i_k : \mathrm{N}_k}$$

$$\frac{\Gamma.\mathrm{N}_k \vdash \rho \text{ set} \quad \Gamma \vdash P : \mathrm{N}_k \quad \Gamma \vdash M_0 : \rho\langle 1, 0_k\rangle \quad \cdots \quad \Gamma \vdash M_{k-1} : \rho\langle 1, (k-1)_k\rangle \quad f :: \Delta \to \Gamma}{\Delta \vdash \mathrm{R}(P, M_0, \ldots, M_{k-1})f = \mathrm{R}(Pf, M_0 f, \ldots, M_{k-1}f) : \rho\langle f, Pf\rangle}$$

$$\frac{\Gamma \text{ ctxt}}{\Gamma \vdash \mathrm{N} \text{ set}}$$

$$\frac{\Gamma\ \mathrm{ctxt}}{\Gamma \vdash 0 : \mathrm{N}\ \mathrm{set}}$$

$$\frac{\Gamma \vdash M : \mathrm{N}}{\Gamma \vdash \mathrm{S}(M) : \mathrm{N}}$$

$$\frac{\Gamma.\mathrm{N} \vdash \rho\ \mathrm{set} \quad \Gamma \vdash P : \mathrm{N} \quad \Gamma \vdash M : \rho\langle 1, 0\rangle \quad \Gamma.\mathrm{N}.\rho \vdash K : \rho\langle \mathrm{pp}, \mathrm{S}(\mathrm{vp})\rangle}{\Gamma \vdash \mathrm{R}(P, M, K) : \rho\langle 1, P\rangle}$$

$$\frac{\Gamma.\mathrm{N} \vdash \rho\ \mathrm{set} \quad \Gamma \vdash M : \rho\langle 1, 0\rangle \quad \Gamma.\mathrm{N}.\rho \vdash K : \rho\langle \mathrm{pp}, \mathrm{S}(\mathrm{vp})\rangle}{\Gamma \vdash \mathrm{R}(0, M, K) = M : \rho\langle 1, 0\rangle}$$

$$\frac{\Gamma.\mathrm{N} \vdash \rho\ \mathrm{set} \quad \Gamma \vdash P : \mathrm{N} \quad \Gamma \vdash M : \rho\langle 1, 0\rangle \quad \Gamma.\mathrm{N}.\rho \vdash K : \rho\langle \mathrm{pp}, \mathrm{S}(\mathrm{vp})\rangle}{\Gamma \vdash \mathrm{R}(\mathrm{S}(P), M, K) = K\langle\langle 1, P\rangle, \mathrm{R}(P, M, K)\rangle : \rho\langle 1, \mathrm{S}(P)\rangle}$$

$$\frac{f :: \Delta \to \Gamma}{\Delta \vdash \mathrm{N}f = \mathrm{N}}$$

$$\frac{f :: \Delta \to \Gamma}{\Delta \vdash 0f = 0 : \mathrm{N}}$$

$$\frac{\Gamma \vdash M : \mathrm{N} \quad f :: \Delta \to \Gamma}{\Delta \vdash \mathrm{S}(M)f = \mathrm{S}(Mf) : \mathrm{N}}$$

$$\frac{\Gamma.\mathrm{N} \vdash \rho\ \mathrm{set} \quad \Gamma \vdash P : \mathrm{N} \quad \Gamma \vdash M : \rho\langle 1, 0\rangle \quad \Gamma.\mathrm{N}.\rho \vdash K : \rho\langle \mathrm{pp}, \mathrm{S}(\mathrm{vp})\rangle \quad f :: \Delta \to \Gamma}{\Gamma \vdash \mathrm{R}(P, M, K)f = \mathrm{R}(Pf, Mf, K\langle\langle f\mathrm{pp}, \mathrm{vp}\rangle, \mathrm{v}\rangle) : \rho\langle f, Pf\rangle}$$

Bibliography

[1] Michael Beeson (1985), *Foundations of Constructive Mathematics.* Springer-Verlag.

[2] Errett Bishop and Douglas S. Bridges (1985), *Constructive Analysis.* Springer-Verlag.

[3] Douglas S. Bridges and Fred Richman (1987), *Varieties of Constructive Mathematics.* London Mathematical Society Lecture Notes, Vol. 97. Cambridge University Press

[4] Coq Reference Manual. URL: `coq.inria.fr`

[5] Thierry Coquand, Peter Dybjer, and Erik Palmgren. *Type-theoretic Foundations of Constructive Mathematics,* to appear, preliminary version of 2007.

[6] Johan Granström (2011). *Treatise on Intuitionistic Type Theory.* Springer.

[7] Arend Heyting (1971), *Intuitionism.* North-Holland.

[8] Martin Hofmann (1997). Syntax and Semantics of Dependent Types, In: Semantics and Logic of Computation. Cambridge University Press.

[9] Per Martin-Löf (1984), *Intuitionistic Type Theory. Notes by Giovanni Sambin of a series of lectures given in Padua 1980.* Bibliopolis.

[10] Per Martin-Löf (1992). *Substitution calculus.* Seminar notes. (Formulation of type theory with explicit substitution.)

[11] Per Martin-Löf (2008) One hundred years of Zermelo's axiom of choice: what was the problem with it? In: *Logicism, Intuitionism, and Formalism: What Has Become of Them?*, Sten Lindström, Erik Palmgren, Krister Segerberg, and Viggo Stoltenberg-Hansen, editors (2008). (Text of Martin-Löf's Kolmogorov lecture 2004). URL:

`www.math.kth.se/~kurlberg/colloquium/2005/MartinLooef.pdf`

[12] Ray Mines, Fred Richman and Wim Wuitenburg (1988). *A Course in Constructive Algebra.* Springer.

[13] Bengt Nordström, Kent Peterson and Jan Smith (1990), *Programming in Martin-Löf's Type Theory.* Oxford University Press. [The book is out of print, but is accessible in electronic form at URL:
`www.cs.chalmers.se/Cs/Research/Logic/book/` .]

[14] Christine Paulin Mohring: Inductive Definitions in the System Coq: Rules and Properties. ENS Lyon, L.I.P. Research Report 92-49, 1992.

[15] Aarne Ranta (1994), *Type-theoretical Grammar.* Oxford University Press.

[16] Anne S. Troelstra och Dirk van Dalen (1988), *Constructivism in Mathematics, Vol. I & II.* North-Holland.

[17] Dirk van Dalen (1997) *Logic and Structure,* Third edition. Springer.

[18] Olov Wilander. Setoids and universes. *Mathematical Structures in Computer Science* 22(2010), pp. 563 – 576.